SBRC 2021

Organizado por UFU

Minicursos do XXXIX

Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos

Organizadores

José Ferreira de Rezende (UFRJ)
Kleber Vieira Cardoso (UFG)
Pedro Frosi Rosa (UFU)
Flávio de Oliveira Silva (UFU)

SBRC 2021

Simpósio Brasileiro de Redes de Computadores

Organizado por UFU

Minicursos do XXXIX

Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos

Organizadores

José Ferreira de Rezende (UFRJ)
Kleber Vieira Cardoso (UFG)
Pedro Frosi Rosa (UFU)
Flávio de Oliveira Silva (UFU)

Sociedade Brasileira de Computação
Porto Alegre
2021

Dados Internacionais de Catalogação na Publicação (CIP)

S612 Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (39. : 2021 : Uberlândia, MG)

Minicursos do 39ª Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos [recurso eletrônico] – Organizadores: José Ferreira de Rezende, Kleber Vieira Cardoso, Pedro Frosi Rosa, Flávio de Oliveira Silva – Porto Alegre : SBC, 2022.

ISBN 978-65-87003-84-9

1. Computação. 2. Rede de computadores. 3. Sistemas distribuídos. I. Rezende, José Ferreira de. II. Cardoso, Kleber Vieira. III. Rosa, Pedro Frosi. IV. Silva, Flávio de Oliveira V. Sociedade Brasileira de Computação. VI. Laboratório Nacional de Redes de Computadores. VII. Título.

CDU 004

Catalogação elaborada por Francine Conde Cabral
CRB-10/2606

Sociedade Brasileira de Computação – SBC

Contato
Av. Bento Gonçalves, 9500
Setor 4 - Prédio 43.412 - Sala 219
Bairro Agronomia
91.509-900 – Porto Alegre RS

CNPJ: 29.532.264/0001-78
http://www.sbc.org.br

Laboratório Nacional de Redes de Computadores (LARC)

Diretor do Conselho Técnico-Científico
Ronaldo Alves Ferreira (UFMS)

Diretor Executivo
Miguel Elias Mitre Campista (UFRJ)

Vice-Diretora Executiva
Anelise Munaretto (UTFPR)

Vice-Diretor do Conselho Técnico-Científico
Eduardo Coelho Cerqueira (UFPA)

Membros Institucionais
SESU/MEC, INPE/MCT, UFRGS, UFMG, UFPE, UFCG (ex-UFPB Campus Campina Grande), UFRJ, USP, PUC-Rio, UNICAMP, LNCC, IME, UFSC, UTFPR, UFC, UFF, UFSCar, CEFET-CE, UFRN, UFES, UFBA, UNIFACS, UECE, UFPR, UFPA, UFAM, UFABC, PUCPR, UFMS, UNB, PUC-RS, PUCMG, UNIRIO, UFS e UFU.

Contato
Laboratório Nacional de Redes de Computadores (LARC)
Av. Athos da Silveira Ramos, 149, Cidade Universitária
Centro de Tecnologia, bloco H, sala 321
Rio de Janeiro - RJ - Brasil
CEP: 21941-972
AC: Miguel Elias Mitre Campista

CNPJ: 00.580.126/0001-82
http://www.larc.org.br/

Sinopse

O livro Minicursos do XXXIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC 2021) contém os minicursos selecionados para apresentação na edição do evento realizada online, no período de 16 a 20 de agosto de 2021, na cidade de Uberlândia/MG. O Livro dos Minicursos do SBRC tem sido tradicionalmente utilizado como material de estudo de alta qualidade por alunos de graduação e pós-graduação, bem como por profissionais da área. O principal objetivo dos Minicursos do SBRC é oferecer treinamento e atualização de curto prazo em temas normalmente não cobertos nas estruturas curriculares e que possuem grande interesse entre acadêmicos e profissionais.

O Livro de Minicursos SBRC 2021 aborda temas atuais e de interesse da comunidade, como sistemas de software emergentes, mobilidade e segurança em redes centradas em informação, virtualização de funções de rede na Internet das Coisas (IoT), redes vestíveis e sistemas ciber-humanos e ainda aprendizado federado aplicado à IoT.

No primeiro capítulo, intitulado "Emergent Software Systems: Theory and Practice", os autores apresentam o conceito de Sistemas de Software Emergentes. A abordagem do Software Emergente visa reduzir o esforço inicial para criar soluções autônomas; suporta a criação de sistemas totalmente adaptáveis capazes de aprender autonomamente sobre a estrutura do sistema e seu ambiente operacional sem nenhum conhecimento pré-definido.

No segundo capítulo, "Revisitando as ICNs: Mobilidade, Segurança e Aplicações Distribuídas através das Redes de Dados Nomeados", os autores apresentam uma introdução e revisão dos fundamentos das *Information-Centric Networks* (ICN) e em seguida a arquitetura *Named Data Networking* (NDN). Os autores apresentam as principais questões relacionadas à mobilidade, segurança e aplicações distribuídas através por demonstrações e atividades práticas com ambientes previamente configurados.

No terceiro capítulo, "Virtualização de Funções de Rede na IoT: Um Panorama do Gerenciamento de Desempenho x Segurança", trata do uso de *Network Function Virtualization* (NFV) em IoT do ponto de vista de gerenciamento, desempenho e segurança da rede. O texto apresenta questões e o estado da arte relacionados ao desempenho do uso de NFV na detecção e mitigação de ameaças em redes específicas de IoT.

O quarto capítulo, "Das Redes Vestíveis aos Sistemas Ciber-Humanos: Uma Perspectiva na Comunicação e Privacidade dos Dados", aborda a rápida evolução que vem ocorrendo desde os primórdios das redes vestíveis até a pesquisa de ponta em nanorredes. Essa evolução rápida fundamenta a construção dos sistemas ciberfísicos e ciber-humanos que possuem aplicações em diversas áreas. Os autores descrevem a necessidade de explorar as vulnerabilidades e desafios que essas redes possuem em relação à privacidade dos dados e resiliência de seus serviços e apresentam uma discussão, demonstrando através de exemplos práticos essas fragilidades e um levantamento do estado da arte de propostas acadêmicas para a proteção da privacidade e resiliência de seus serviços

No quinto e último capítulo, "Aprendizado Federado aplicado à Internet das Coisas", os autores apresentar os principais fundamentos do Aprendizado Federado (*Federated Learning* – FL) abrangendo as ferramentas e passos necessários para o desenvolvimento de aplicações e serviços voltadas à IoT. Os conceitos abordados incluem uma introdução à Aprendizagem de Máquina (centralizada e distribuída), o estado-da-arte em FL, uma visão geral dos trabalhos existentes, os desafios e as perspectivas futuras para o avanço da área.

Os cinco capítulos abordam de forma ampla temas atuais da comunidade, sendo uma obra útil para pesquisadores e profissionais da área de redes de computadores e sistemas distribuídos.

Synopsis

The book of Short Courses of the XXXIX Brazilian Symposium on Computer Networks and Distributed Systems (SBRC 2021) contains the minicourses selected for presentation in the online edition of the event, from August 16 to 20, 2021, in the city of Uberlândia/MG. The main objective of the SBRC Mini-Courses is to offer short-term training and updating on topics not typically covered in curricular structures and which are of great interest among academics and professionals. The SBRC Mini-Course Book has traditionally been used as high-quality study material by undergraduate and graduate students and professionals in the field.

The SBRC 2021 Short Course Book covers current topics of interest to the community, such as emerging software systems; mobility, security in information-centric networks; virtualization of network functions in the Internet of Things (IoT); wearable networks, and cyber-human systems; and federated learning applied to IoT.

In the first chapter, entitled "Emergent Software Systems: Theory and Practice," the authors present the concept of Emerging Software Systems. The Emerging Software approach aims to reduce the initial effort to create standalone solutions; it supports fully adaptable systems capable of autonomously learning about the system structure and its operating environment without any pre-defined knowledge.

In the second chapter, "Revisitando as ICNs: Mobilidade, Segurança e Aplicações Distribuídas através das Redes de Dados Nomeados", the authors present an introduction and review of the fundamentals of Information-Centric Networks (ICN) and then the Named Data Networking (NDN) architecture. The authors present the main issues related to mobility, security, and distributed applications through demonstrations and practical activities with previously configured environments.

The third chapter, "Virtualização de Funções de Rede na IoT: Um Panorama do Gerenciamento de Desempenho x Segurança", deals with the use of Network Function Virtualization (NFV) in IoT from the point of view of network management, performance, and security. The text presents issues and state-of-the-art related to the performance of NFV in the detection and mitigation of threats in specific IoT networks.

The fourth chapter, "Das Redes Vestíveis aos Sistemas Ciber-Humanos: Uma Perspectiva na Comunicação e Privacidade dos Dados", addresses the rapid evolution that has been taking place from the beginnings of wearable networks to cutting-edge research in nanonetworks. This rapid evolution underlies the construction of cyber-physical and cyber-human systems that have applications in several areas. The authors describe the need to explore the vulnerabilities and challenges that these networks have concerning data privacy and the resilience of their services and present a discussion, demonstrating these weaknesses through practical examples and a survey of the state of the art of academic proposals for privacy protection and resiliency of your services.

In the fifth and final chapter, "Aprendizado Federado aplicado à Internet das Coisas", the authors present the main foundations of Federated Learning (FL), covering the tools and steps necessary for the development of IoT applications and services. The concepts covered include an introduction to Machine Learning (centralized and distributed), the state-of-the-art in FL, an overview of current work, challenges, and future perspectives for advancing the field.

The five chapters broadly address current community issues, making them practical work for researchers and professionals in computer networks and distributed systems.

Mensagem dos Coordenadores Gerais do SBRC 2021

Com grande alegria e satisfação apresentamos o XXXIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC), o SBRC 2021.

A 39ª edição do Simpósio foi realizada de 16 a 20 de agosto de 2021 de forma online. A coordenação do evento ficou sob a responsabilidade do Programa de Pós-Graduação em Ciência da Computação (PPGCO) mantido pela Faculdade de Computação (FACOM) da Universidade Federal de Uberlândia (UFU). Em função do contexto da pandemia do COVID-19 em 2021 o evento foi planejado para ser realizado completamente de forma online.

A realização de evento desta magnitude somente é possível com o apoio, a dedicação, o comprometimento e o trabalho diversas pessoas, de diferentes instituições, e nossa primeira palavra é agradecimento.

Gostaríamos de agradecer à Sociedade Brasileira de Computação (SBC), à Comissão Especial de Redes de Computadores e Sistemas Distribuídos da SBC (CE-RESD) e ao Laboratório de Redes de Computadores (LARC) pela confiança e apoio total para a organização do evento.

Também agradecemos aos patrocinadores que permitiram a realização do evento: o Comitê Gestor da Internet no Brasil, o CGI.br na categoria Diamante. Na categoria Ouro do SBRC 2021 estão a FORTINET, a HUAWEI, o PAGSEGURO, o PAGBANK. Na modalidade Prata, o GOOGLE e a ALGAR TELECOM e ainda UFU, através da Pró-Reitoria de Pesquisa e Pós-Graduação.

A organização do evento somente foi possível pela dedicação e pelo trabalho intenso dos coordenadores das diversas atividades realizadas. Nossos profundos agradecimentos aos coordenadores do comitê de programa Alberto Egon Schaeffer Filho (UFRGS) e Fabio Luciano Verdi (UFSCAR). Aos coordenadores de workshops Jó Ueyama (USP) e Rafael Lopes Gomes (UECE). Aos coordenadores de palestras, tutoriais e mentoria, Antônio Alfredo Loureiro (UFMG) e Eduardo Coelho Cerqueira (UFPA). Aos coordenadores de Painéis, Augusto José Venâncio Neto (UFRN) e Carlos André Guimarães Ferraz (UFPE). Aos coordenadores de minicursos, José Ferreira de Rezende (UFRJ) e Kleber Vieira Cardoso (UFG). Aos coordenadores do Salão de Ferramentas, Cristiano Bonato Both (UNISINOS) e Daniel Fernandes Macêdo (UFMG). Aos coordenadores do Concurso de Teses e Dissertações, Leandro Aparecido Villas (UNICAMP) e Michele Nogueira Lima (UFMG). Aos coordenadores do Hackathon, Cesar Augusto C. Marcondes (ITA) e Linnyer Beatrys Ruiz Aylon (UEM) e ainda comitê de organização local, João Henrique de Souza Pereira (UFU), Luiz Cláudio Theodoro (UFU), Rodrigo Miani (UFU) e Rafael Pasquini (UFU).

Agradecemos ainda ao comitê consultivo do SBRC Ronaldo A. Ferreira (UFMS), Alberto Egon Schaeffer Filho (UFRGS), Miguel Elias Mitre Campista (UFRJ), Igor Monteiro Moraes (UFF), Weverton Cordeiro (UFRGS), Anelise Munaretto Fonseca (UFTPR), Marcelo Gonçalves Rubinstein (UERJ), Antônio Jorge Gomes Abelém (UFPA) e Fabíola Gonçalves Pereira Greve (UFBA) e finalmente agradecemos a todo time local formado por estudantes do Programa de Pós-graduação em Ciência da Computação e técnicos da Universidade Federal de Uberlândia.

Como resultado de todo este trabalho, o SBRC 2021 foi muito rico em conteúdo e, durante a semana, diversas atividades foram realizadas entre elas: dezessete (17) Seções Técnicas; cinco (5) palestras internacionais; cinco (5) painéis com profissionais e pesquisadores do Brasil e do exterior; cinco (5) minicursos; concurso de teses e dissertações; salão de ferramentas; evento MUlheres em redeS de computadores e sistemAs diStribuídos (MUSAS); Hackathon FORTINET e ainda dez (10) Workshops em diversos temas e que com uma rica programação envolvendo outras seções técnicas, painéis e palestras nacionais e internacionais.

Com toda esta programação, o SBRC 2021, reúne representantes nacionais e internacionais da academia, administração pública, empresas e outras organizações de forma a favorecer um ambiente estimulante para a apresentação e discussão de ideias, soluções e aprendizados com foco nos mais diferentes desafios e oportunidades existentes na área de redes de computadores e sistemas distribuídos.

Pedro Frosi Rosa (UFU)
Flávio de Oliveira Silva (UFU)
Coordenadores Gerais do SBRC 2021

Mensagem dos Coordenadores de Minicursos do SBRC 2021

É com muita satisfação que apresentamos os Minicursos do XXXIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC), realizado online entre os dias 16 e 20 de agosto de 2021. Os Minicursos do SBRC produzem os capítulos do Livro de Minicursos os quais permitem à comunidade se atualizar em temas que despertam grande interesse entre acadêmicos e profissionais. O livro, assim como as apresentações no simpósio, possui um viés altamente didático, capaz de motivar alunos de graduação, pós-graduação e profissionais da área a mergulharem nos temas propostos. A ideia é abrir horizontes e complementar a formação dos participantes em temas recentes ou ainda pouco explorados nas estruturas curriculares das instituições tradicionais de ensino e pesquisa e com grande potencial de interesse para estudantes e profissionais de forma a melhor qualificá-los para suas pesquisas acadêmicas e/ou sua atuação no mercado.

Nesta edição do evento, recebemos 10 submissões no prazo final após um adiamento. Todas as propostas foram atribuídas a três revisores, os quais aceitaram prontamente e revisaram no prazo. Diante deste fato, gostaríamos de agradecer imensamente aos 18 membros do TPC deste ano que se dedicaram à revisão das propostas de forma sempre atenciosa. Foi um prazer trabalhar com todos.

Houve uma decisão de aceitar no máximo cinco propostas. As cinco propostas selecionadas, aquelas que tiveram as maiores médias gerais, e contribuíram para atrair público para o evento. Os temas selecionados são bastante atuais e abordam temas como fog computing, quantum Internet, softwarização em 5G, deep learning e computação serverless. Vale mencionar que, dentre as propostas não contempladas, algumas teriam plenas condições de serem aceitas, o que infelizmente não foi possível desta vez.

Gostaríamos de agradecer aos organizadores gerais do SBRC 2021, os professores Pedro Frosi Rosa e Flávio de Oliveira Silva os quais, com muita dedicação, otimismo e resiliência, não desistiram de organizar o SBRC neste ano atípico de 2021. Agradecemos a confiança na organização dos Minicursos, no trato sempre muito gentil e na infinita paciência. Todas as mensagens enviadas foram sempre respondidas prontamente.

Agradecemos também aos autores dos minicursos, pois sem a dedicação e esmero de todos na escrita e na entrega pontual do material, nada disso teria sido viável. Temos certeza de que todas as apresentações brindaram todo o trabalho e possibilitaram ampla absorção técnica da plateia. O desfecho dos minicursos foi suave graças à colaboração e ao profissionalismo de todos. Agora, é rentabilizar o árduo trabalho fomentando frutíferas discussões em direção ao avanço da Ciência, cada vez mais útil e necessária nestes dias.

José Ferreira de Rezende
Kleber Vieira Cardoso
Coordenadores de Minicursos do SBRC 2021

Comitê de Programa dos Minicursos do SBRC 2021

Aldri Luiz dos Santos (UFMG)
Alex Borges Vieira (UFJF)
Anelise Munaretto (UTFPR)
Antonio Carlos de Oliveira Junior (UFG)
Carlos Alberto Campos (UNIRIOTEC)
Christian Esteve Rothenberg (UNICAMP)
Fabio Verdi (UFSCar)
Juliana Freitag Borin (UNICAMP)
Leobino Nascimento Sampaio (UFBA)
Luciano Paschoal Gaspary (UFRGS)
Lúcio Rene Prade (UNISINOS)
Magnos Martinello (UFES)
Marcel William Rocha da Silva (UFRRJ)
Natalia Castro Fernandes (UFF)
Paulo Ditarso Maciel Jr. (IFPB)
Ronaldo Alves Ferreira (UFMS)
Sérgio Teixeira de Carvalho (UFG)
Waldir Moreira (Fraunhofer Portugal AICOS)

Sumário

Capítulo 1

Emergent Software Systems: Theory and Practice

Roberto Rodrigues Filho (UFG), Barry Porter (Lancaster University),
Fábio M. Costa (UFG) and Iwens Sene Júnior (UFG)

Abstract

Autonomic Computing and related research communities have drawn attention from researchers and industry practitioners who seek techniques and tools to build large-scale, reliable self-adaptive systems. However, building autonomic solutions remains a challenge: i) the upfront effort to develop such systems is very high, making them costly to implement; ii) only specialised parts of the system are made adaptive, limiting its flexibility in handling unknown operating conditions; and iii) state-of-the-art approaches still heavily rely on design-time predictions of operating conditions, making systems execution uncertain when predictions are wrong. To address these challenges, the concept of Emergent Software Systems has been proposed. The Emergent Software approach aims to reduce the upfront effort to create autonomic solutions, and it supports fully adaptive systems able to autonomously learn about the system's structure and operating environment with no predefined knowledge or predictions. This chapter aims to disseminate Emergent Software Systems, presenting the central concept and tools to realise the approach.

1.1. Introduction

Over the last two decades, systems researchers have acknowledged the increasing complexity in developing and managing software systems and the need for new and dynamic solutions [6, 24, 26, 47]. According to Blair [6], this complexity stems from two main reasons: i) the increasing heterogeneity of modern systems, and ii) the increasing volatility ever more present in the operating environment of such systems. In addition, systems have become larger, with millions of lines of code divided into software modules running over distributed infrastructures across the globe (e.g., YouTube), making them difficult to manage and evolve to accommodate new business and user demands.

This makes systems increasingly difficult to manage. In particular, the volatility of the operating environment resulting from the frequent changes that occur in the system is a crucial challenge to overcome. These changes happen across the entire system, and they impact different layers: infrastructure, platform and application. In the infrastructure, the

addition and removal of nodes and adjustments in a node's resource capabilities (e.g., due to costs or system demands) in modern elastic environments, such as the cloud, are very common. These changes in the infrastructure may impact the software modules running on the impacted nodes, which sometimes need to be tuned accordingly. For instance, nodes being removed from the infrastructure due to faults require changes in the software to cope with such situations (e.g., adopting fault-tolerance mechanisms). Likewise, an evolution of the software features may demand changes in the system's overall architecture to improve efficiency as well as in its infrastructure to provide more resources. For instance, the addition of video streaming as a feature in a web-based application may require an increase in network resources and computing resources from the infrastructure.

These are examples of common changes, and given the characteristics of current systems (large scale and heterogeneity), these situations are difficult to handle. Nevertheless, it is not hard to imagine that it is possible to anticipate such scenarios. That gives time for engineers to prepare the system for such changes and make sure that the proper mechanisms are in place when these changes occur. Nowadays, however, due to the scale these systems operate, these examples of changes are often accompanied by difficult-to-predict situations. For instance, content provided by a system may go viral, which suddenly increases user access, demanding more resources and, sometimes, a more optimised version of the system to better handle the new workload. In other cases, due to the complex interaction among the distributed modules of a system, a predefined fault-tolerance mechanism may be ineffective because it simply did not anticipate a series of successive failures throughout the system.

The above examples make evident that it is crucial to address the volatility (i.e., the frequent changes) of modern operating environments in contemporary systems. The illustrated situations demand fast and accurate decision making that may affect many different layers of the system. In the literature [26, 40, 47], the self-adaptive and autonomic computing communities have argued that pushing the responsibility of decision making to adapt systems when facing changes to the machine itself is the best way forward.

Self-adaptive and autonomic systems are software systems able to change their behaviour or structure to accommodate changes with minimum or no human interference in the process [10]. These systems require the definition of an adaptation logic, responsible for deciding how and when adaptation occurs, and an adaptation mechanism responsible for reifying changes to the system. As the volatility of modern systems increases, the demand for self-adaptive systems follows suit. Moreover, considering that the decisions on how adaptation should be performed must be fast and accurate, relying on engineers to manually analyse the current system status and design the new system behaviour to cope with changes is a slow and unsuitable process. Self-adaptation solutions, in turn, are a perfect fit to address the increasing volatility of modern operating environments. That is because they take over the responsibility to make decisions at runtime and quickly perform online adaptations with no human interference.

However, the development of such adaptive systems presents its own challenges. First, the upfront effort of creating such systems is high. This, in turn, makes the development of self-adaptive systems costly. The effort is put into capturing the adaptation logic

and ensuring that the system will not enter a malfunctioning state as a result of adaptation. Also, the adaptation mechanisms should encompass all adaptation possibilities, and support seamless runtime adaption with minimum impact on system's performance. Second, given that it is costly to create self-adaptive systems, a common strategy is to make only specialised parts of the system self-adaptive. This vastly limits the flexibility of the systems in handling changes, particularly the unexpected ones. Lastly, current approaches heavily rely on design-time prediction to build such systems. This inevitably leaves the system incapable of handling unforeseen/uncertain conditions and often underperforming when these predictions are inaccurate. As a response to that, the concept of Emergent Software Systems have been proposed and investigated in the literature [37, 42, 43, 44, 45].

Emergent Software Systems is an approach to facilitate the development of self-adaptive systems. It applies lightweight component-based models (e.g., [7, 11, 36]) in tandem with reinforcement learning [49] algorithms to support the creation of self-adaptive systems. The component-based models enable the composition of software from a collection of small (e.g., the size of a class in a Java program) and highly reusable components. These models also enable seamless runtime adaptation of the system by replacing a set of components with others. Considering that the system is entirely composed of these small components, the entire system is made to adapt. The reinforcement learning algorithm, in turn, enables the system to start with no predefined domain-specific knowledge and learn as it executes which software composition (defined by the set of small components) is the most suitable composition for the operating environment on which the system is currently operating. Thus, the reinforcement learning algorithm allows Emergent Software Systems to autonomously build their own understanding of the system structure and their operating environment, eliminating the need for predictions.

This chapter presents an approach that aims to facilitate the creation of self-adaptive systems, named Emergent Software Systems. The goal is to motivate the adoption of the Emergent Software System approach to build self-adaptive software able to cope with the demands of contemporary systems. To achieve this, we introduce the main concepts of self-adaptive and autonomic computing systems, as well as the state-of-the-art in the field, along with the shortcomings of current approaches (Sec. 1.2). Then, based on the flaws of current approaches, we motivate and introduce the concept of Emergent Software Systems (Sec. 1.3). We introduce the concept along with the challenges involved in realising it. Next, we introduce our implementation of the concept and a framework that supports the creation of Emergent Software Systems (Sec. 1.5). We discuss the application of the concept to build microservices able to evolve their internal architectural composition and to cope with the dynamism of current operating environments (Sec. 1.6). We make available a collection of practical exercises and code examples for the reader to gain hands-on experience with the development of Emergent Software Systems. Finally, we conclude the chapter by inviting researchers and industry practitioners to join the effort to further develop the concept and contribute to create self-adaptive solutions that are able to cope with the demands of contemporary systems (Sec.1.7).

1.2. Related Work and Background

This section introduces the main concept in self-adaptive, autonomic computing systems. It also presents some of the most relevant and seminal papers in the area. We discuss the current state-of-the-art techniques used to implement self-adaptive systems. We also focus on reinforcement learning algorithms to support runtime learning of the self-adaptive systems adaptation logic. This focus on the learning algorithms is essential for understanding our proposed Emergent Software Systems approach. We conclude the section by presenting the shortcomings of current self-adaptive techniques.

1.2.1. Autonomic Computing

Motivated by the increasing complexity in contemporary systems, IBM introduced the concept of Autonomic Computing [26] in a manifesto in 2001. Their motivation were their vision of future systems being characterised by their large size, their interconnectivity with other large systems, the heterogeneity level and the constant demands to cope with changes. These type of systems require skilled software engineers to install, configure, tune and maintain them. In [26], Kephart and Chess argue that contemporary systems were reaching the limit of the human capacity to manage systems adequately and to timely react to unexpected events. Thus, inspired by the autonomic nervous system in human beings, the Autonomic Computing vision aims to enable systems to self-manage based on goals defined by administrators with minimum human interference.

The realisation of the concept of Autonomic Computing requires systems to implement autonomous behaviour towards a specific system aspect (e.g., security, performance, fault-tolerance). These systems aim to autonomously maintain specific properties. These are known as self-* properties. These properties are dimensions of autonomous behaviour incorporated in a system to address a particular system aspect. Autonomous systems may implement the following self-* properties: self-protecting, self-optimising, self-healing and self-configuring.

Self-protecting systems (e.g., [52]) are capable of identifying possible security threats and operating risks for the system well functioning. Furthermore, these systems can adequately handle and prevent malicious users from exploiting possible security threats by, for example, autonomously changing their internal structure. Self-optimising systems (e.g., [22]) are systems that can change their structure to improve some aspects of their performance autonomously, for example, by reducing the systems response time. Self-configuring systems (e.g., [25]) can autonomously set up when introduced to a distributed system or change their configuration to accommodate the insertion of new systems. Self-healing systems (e.g., [1]) is capable of maintaining a level of reliability by discovering faults in the system's behaviour and deciding on a course of action to maintain system execution.

The area of autonomic computing brought a variety of scientific and engineering challenges to be addressed. Many of these challenges were introduced and discussed in seminal paper 'The Vision of Autonomic Computing' [26]. The main challenges involve the life-cycle of autonomous systems, which comprehends systems design, test, management, monitoring and upgrading systems. The management of interactions among autonomous systems includes defining services provided and required by autonomous

systems, services discovery and negotiation of providing services to multiple autonomous entities. Another challenge is the definition and representation of the global system's goals, representing the interface between humans and autonomous systems. These challenges are still in debate in the related research communities with promising solutions and future directions.

1.2.2. Self-adaptive Systems/Self-organising Systems

The term 'self-adaptive systems' is usually used as an umbrella term by different research communities to refer to the systems ability to change its structure to accommodate changes. Whereas in distributed environments, the term 'self-organising systems' is frequently used to refer to systems capable of reorganising their distributed architecture to cope with changes and achieve global system goals. In order to build systems that are capable of self-organising and self-adapting, the system requires the definition of an **adaptation logic** to guide software adaptation, the implementation of **adaptation mechanisms** to propagate changes to the actual system structure, and **coordination** among the system nodes to ensure a coherent change and convergence towards global system goals.

The **adaptation logic** is responsible for capturing the knowledge that allows the system to detect events of interest and decide its next course of actions, which include the decision to maintain the system in its current state or to adapt to another configuration. The adaptation logic can be represented in different forms: the policy-based approaches, which represent adaptation logic using expertly-crafted rules defined in the design phase; model-driven approaches, which apply models representing system properties, QoS traits, system architectures, system goals and equations that support runtime reasoning and adaptation; and bio-inspired algorithms are used to encode adaptation logic imitating behaviours found in nature, e.g. ant-colony.

The **adaptation mechanisms** are responsible for propagating changes in the existing software. These mechanisms are classified either as parametric or architectural approaches. Parametric approaches consider the system as a black box, having only dial buttons to influence the system behaviour. This approach is limited compared to architectural-based approaches because it can only change the system within a predefined range of parametric values. The architectural adaptation approach allows the system to change its structure by replacing components or changing its architecture pattern to maintain system properties (e.g., using a load balancer architecture to maintain the performance or replicate services to cope with system failures). The architectural adaptation allows profound changes in the system, making it more flexible to cope with changes in the operating environment.

In a distributed scenario, it is essential to create mechanisms to ensure that the system achieves its global goals whilst maintaining non-functional requirements. This is imperative, particularly in situations where multiple autonomous single-goal entities make their own adaptation decisions and interact with each other. In this context, the **coordination of software adaptation** is critical to ensure a coherent adaptation of distributed systems and convergence towards a common global system goal. Many approaches for coordinating adaptation were investigated in the multi-agent research community where the autonomous entities (agents) sought cooperation and consensus to make decisions to

converge the system towards the global desired behaviour. Some approaches use voting schemes, action predictions, and other consensus schemes.

Typical activities of a self-adaptive and self-organising system involve monitoring the system execution, analysing the monitored data, deciding on and executing the course of action the system needs to maintain its desired properties. The feedback loop conceptual framework was widely adopted to implement self-adaptive systems for capturing these essential activities to equip systems with autonomous adaptation ability. The most famous feedback loop conceptual framework in self-adaptive systems and autonomic computing field is the MAPE-K loop [3], which stands for Monitoring, Analysing, Planning, Executing. The framework actions are guided by the knowledge (the K in the MAPE-K) defined by experts at the design phase. This knowledge is also expanded by the results of the previous system's actions. In today's autonomous systems, the activities defined in the MAPE-K feedback loop are imperative to realise self-adaptive systems. Even if the classic MAPE-K feedback loop is not applied, some form of it is always used to support reliable adaptation.

1.2.3. Organic Computing

The Organic Computing research initiative started in Germany with the focus to address and explore the self-organisation concept in technical software systems, inspired by neuroscience and molecular biology principles and software engineering [35, 47, 51]. Originated around the same time as the Autonomic Computing paradigm, the Organic Computing initiative also focused on the problem of having multiple instances of interacting autonomous systems, which may lead to conflicts and undesired emergent behaviour affecting the resulting system [47].

In detail, autonomous systems with multiple goals might spontaneously interact with other autonomous systems to achieve and maintain global systems goals. The interaction among autonomous system instances is not a far-fetched futuristic idea. A concrete example given in [47] draws attention to the modern cars and the multiple interacting devices that are required to support the basic cars functions. In this example, devices require data from other devices to provide their functionality, and the orchestration of such devices results in the car and all its available functions. Problems with the interactions among these systems, such as delaying the delivery of information to devices, may lead to miscalculations and compromise service execution. The problems with orchestration and synchronisation that may occur in such scenarios are not the only problem. Another fundamental problem is the emergent undesired behaviours resulted from unpredicted interactions among systems devices.

Given this scenario of interacting autonomous systems, several studies were conducted to validate multiple controlling mechanisms that could be used to prevent undesired emergent behaviour in a variety of scenarios [41, 33] as well as to address further aspects (self-* properties) of autonomous systems [8, 19].

1.2.4. Reinforcement Learning (RL) for Adaptive Systems

Reinforcement Learning is a learning paradigm in which an agent learns to correlate a set of actions to reward values [49, 46]. An agent, in this context, can be defined as software

capable of executing actions and collecting rewards as a consequence of the performed actions. This learning paradigm allows software systems to learn, at runtime, whilst the system interacts with the environment, which actions and order will yield maximum reward values. This characteristic of this learning paradigm applied to self-adaptive systems enables the system to learn the adaptation logic as it executes, driving software adaptation at runtime, and evolving its adaptation logic as the system encounters new, unforeseen situations [2, 27].

Several papers on self-adaptive systems that apply reinforcement learning algorithms in the adaptation process, such as [2], or any machine learning algorithm for that matter [54, 30, 16], focuses on parametric adaptation. Parametric adaptation consists of changes in software parameters as if the system were a black box with dials. In this context, the system can only adapt within a predefined set of values/actions, limiting the capacity of reinforcement learning algorithms to learn beyond the actions that experts established in the design phase.

This limitation of parametric tuning approaches also affects some architectural-based approaches such a [27, 32, 18], because, in these approaches, the system can only change its architecture based on a predefined range of architectural configurations, and that expanding this fixed range of architectural options requires the system and the learning algorithm to be re-evaluated and re-defined.

In [27], Kim and Park demonstrate the application of Q-learning, a well-known and widely used reinforcement learning algorithm, to architectural change robots in a simulation case study. This is one of the earlier and most representative examples of the direct application of reinforcement learning to build and evolve adaptation logic in self-adaptive systems. The paper describes how to create learning tables with actions and systems states, detailing a method to apply Q-learning to self-adaptive systems. The paper ends with evaluating the application of reinforcement learning to robot simulations, showing that a robot that applies such an approach can successfully learn and evolve its adaptation logic.

The main limitation of applying reinforcement learning to self-adaptive systems in the literature is the lack of abstraction of actions and states for the machine learning algorithm. As papers such as [2, 27] illustrate, reinforcement learning algorithms are applied to self-adaptive systems to optimise software in specific domains, with a predefined set of actions observable in specific case studies. This shows high levels of human dependency to define, for each application domain, the set of actions and states on which the learning algorithm will be developed, limiting the potential and generalisation of reinforcement learning approaches in self-adaptive systems.

We argue that in order to fully enable learning and evolution of the adaptation logic of software systems, it is required to i) apply reinforcement learning to the composition process of systems and ii) 'abstract'/remodel the system to be easily handled by machine learning algorithms. The proposed Emergent Systems concept demonstrates how these two points can be realised by unifying component-based models and reinforcement learning algorithms.

1.2.5. Critique of Existing Approaches

Although the main concepts and research areas that focus on self-adaptive software solutions have been introduced, we have not pinpointed the shortcomings of the most relevant work that describes the different existing techniques to create self-adaptive solutions. In this section, we present these studies, dividing them into different categories. The self-adaptive systems research area is vast and multi-disciplinary, having multiple research communities proposing different solutions with different terminologies. To address this issue and present a more complete picture of the state-of-the-art of such techniques, we attempt to group these relevant works based on the similarities of their proposed approach. As a result, we created the following categories: i) rule-based and policy-driven approaches, ii) mathematical models, and iii) bio-inspired approaches. Note that these categories do not contain all relevant approaches. For a better attempt at a complete picture of the different approaches, refer to [42].

Rule-based and policy-driven approaches implement the simplest and most straightforward method to represent the systems adaptation logic. Due to their simplicity, these were the most used methods in designing early self-adaptive systems. The approach consists of manually writing static rules in the design phase to describe software state or operating conditions and make the system aware of what software configuration it should change into in case those conditions are detected. Some approaches implement very straightforward and static rules, not leaving room for autonomous reasoning and decision making in the process, as described in [23], and other approaches, such as in [28], the software engineers describe adaptation rules in terms of logic expressions, which allow a small level of reasoning in the adaptation process.

These policy-driven approaches are very effective when all system states are well known before and the deployment environment is fairly static. For that reason, these approaches were used in early examples of self-adaptive solutions. Due to the **high levels of dynamism** in the operating environment of contemporary applications, the static policy-driven approach is not adequate to support the required levels of flexibility in the software adaptation process. Suppose the system suffers from fluctuations in the workload or unexpected failures in the underlying structure. In that case, these systems cannot react accordingly and accommodate the changes on time, leading the system to a malfunctioning state or degraded performance.

Mathematical models consists of approaches that capture their adaptation logic in the form of mathematical functions. Sometimes differential equations are used to establish a relation between input and output. This input-output mapping is used to capture the systems state and connect it to a particular action that changes the systems state and therefore controls the adaptation process. An example of such approach is described in [31]. Lu *et al.* present mathematical models that guide the assignment of classes of HTTP requests to threads responsible for handling these requests in a web server. The goal is to enable an adaptive assignment of requests to threads to guarantee delays below a predefined value as workload characteristic changes. Although this approach supports a dynamic request-to-thread assignment according to workload variation, its adaptation

logic is determined by a predefined set of fix equations, which are highly complex to elaborate and does not evolve to accommodate unexpected events.

A slightly different and better approach to this is shown in [18]. Elkhodary *et al.* propose a learning approach to generate mathematical functions that correlate the presence of software features in the system with the satisfaction levels of the system's goals. These mathematical functions are autonomously created as a result of offline training by running the system and exposing it to a set of operating conditions the system is expected to find in production. After the system's deployment, as it is exposed to new operating conditions, it is capable of fine-tuning the functions coefficients to incorporate new operating conditions. Although this approach reduces the complexity of defining equations to guide adaptation by generating the model autonomously, it only tunes the model to accommodate new conditions. If new software modules are added, or completely different conditions are encountered, the system will require offline re-training.

Bio-inspired approaches encode behaviours found in nature into algorithms to solve optimisation problems. A very popular example of this approach is the ant colony, where agents imitate ant behaviour in finding food by leaving a trail of pheromone that other ants can sense [15]. A great variety of papers explored the application of such methods in a variety of problems such as vehicle routing [20], graph colouring [12] and project scheduling [34]. Another popular example of this approach is evolutionary computing [17]. This concept applies the basic idea of biological evolution to solve problems. This approach defines a population of individuals, then randomly insert genetic variations into the population and submit them through a selective process eliminating some individuals and leaving the fittest according to a selection function. The remaining individuals are used to build the next generation of individuals that go through the process again.

Bio-inspired approaches are used to directly provide systems with self-adaptive capabilities. For example, Ding *et al.* [14] describe an algorithm that encodes the behaviour of Neuroendocrine Immune System to support autonomous composition and adaptation of web services. Another example is described in [40], where Ramirez *et al.* used evolutionary computing to predict operating conditions the system will be facing and how the diverse available software configurations would best suit different conditions maintaining desired QoS levels. These approaches are complementary to Emergent Software Systems, as they can be used in online learning to find optimal architectural compositions.

1.3. The Concept of Emergent Software Systems

This section introduces the concept of Emergent Software Systems and discusses the main challenges of realising the approach. First, we motivate the need for Emergent Software Systems, then we introduce and described the concept in detail. We conclude the section by presenting the main challenges inherent in the concept regardless of current technology's limitations. Note that we do not discuss the implementation of the concept in this section. That is to allow the reader to focus on the ideas behind the concept rather than implementation details. We present an implementation of the concept in Sec. 1.5.

1.3.1. Motivation

The main motivation for considering the advancement of self-adaptive solutions is the increasing dynamism of current operating environments. As previously described, this dynamism requires systems capable to react fast and accurately to new and often unexpected operating conditions. This scenario makes the advancement of self-adaptive systems crucial when considering the development and management of future systems.

An analysis of the start-of-the-art approaches to realise self-adaptive systems (Sec. 1.2) shows that current approaches share one or more of the following characteristics that prevent the advancement of self-adaptation systems: i) significant human-dependency in their design; ii) inability to autonomously evolve their adaptation logic in face of unexpected conditions; and iii) adaptation mechanisms are only applied to very specific parts of the systems. The Emergent Software System concept is introduced to enable the creation of self-adaptive system that addresses these three main concerns.

As previously described, the realisation of self-adaptive solutions requires the development of an adaptation logic, adaptation mechanisms and often a coordination strategy (in distributed contexts) to ensure seamless runtime adaptation. Analysing the most relevant work in this domain with regards to adaptation logic creation and evolution, a research gap was identified [42]. There is a lack of work addressing the entirely autonomous creation and evolution of adaptation logic, which would enable the development of solutions capable of addressing the high levels of uncertainty commonly presented in developing adaptive solutions (as described in [10]). Furthermore, the creation of systems able to autonomously develop their adaptation logic would reduce the engineering efforts of developing autonomous solutions, facilitating the development of self-adaptive systems.

Considering the adaptation logic, the state-of-the-art approaches implement different techniques including policy-driven (e.g., Grace, *et al.* [23]), system representative models (e.g., SASSY [32]) and mathematical models (e.g., Netkv [53]). Most of the approaches require the definition of adaptation information in the design time. Some techniques allow the adaptation logic to evolve as new events occur in the operating environment, enabling a limited evolution of its adaptation logic to accommodate new events from the operating environment (e.g., FUSION [18]). This approach, however, can not evolve its adaptation logic if new software is added, requiring a re-execution of its offline training approach to generate the new adaptation logic. Finally, Georgé *et al.* [21], have a complete autonomous multi-agent approach, that is constantly trying to optimise towards a goal, but never builds nor evolves an adaptation logic. This approach is not able to 'remember' optimal solutions for previously seen situations, having to search for solutions even when previously seen conditions occur.

Therefore, the concept of Emergent Software Systems is a response to the identified gap in the literature considering the development of self-adaptive systems. The concept facilitates the development of self-adaptive systems because it does not require predictions or domain-specific knowledge. The resulting system is entirely adaptive with no extra effort than writing code (e.g., matching code to a feature model). Lastly, the emergent system can learn and evolve its adaptation logic as it executes and new conditions arise.

1.3.2. Definition

Emergent Software Systems is an approach to facilitate the development of self-adaptive systems. In detail, this novel concept consists of autonomously composing systems from small reusable software components, according to metrics and events (numbers and labels that represent the system's health status and its operating condition) and user-provided goals. This process ensures that software is autonomously composed as a response to external stimuli and user-desired goals. Next we present a definition of Emergent Software System as it was first introduced in [42].

The realisation of Emergent Software Systems requires the existence of a system goal G and a finite set of small software units SU. The goal G defines the main purpose of the system (comprehending both functional and non-functional requirements) and SU is composed of a variety of software units u. For some $u \in SU$, there exist implementation variations of u, meaning that unit variants provide the same functionality in different forms. For example, if the functionality is to compress a stream of bytes, there should exist software units implementing variations of compression algorithms (e.g., gzip, zlib, etc.). The functionalities are defined in the form of interfaces that define the available functions the software unit offer or require, as well as the kind of data that can be passed in or out of those functions. The software units are connected to each other through their interfaces, connecting the units that require a specific interface to the units that provide the same interface. This process of connecting software units is the result of software architectural composition, forming a single instance of software in a single node.

The presence of variations of software units enables a variety of architectural compositions. This happens locally, in individual machines, but the same process can also be used to compose distributed systems with emergent software running on a distributed infrastructure. In this distributed setting, each resulting software assembly influences how an executing software on a given node interacts with others. The interaction of multiple emergent systems running on distinct nodes constitutes the global system architectural composition. In other words, there are no predefined specifications of the system topology, protocols or data semantics; these are determined by local architectural composition.

Furthermore, the existence of unit variants allows the resulting system to be assembled to achieve G in different ways, enabling the system to maintain the required QoS in diverse operating conditions. Furthermore, some of the software units are required to emit a stream of metrics and events. Metrics are numeric values that represent the health status (e.g., performance) of the system and are also used as indicators of the satisfaction levels of G. On the other hand, the events are used to classify the operating environment, representing systems inputs and deployment characteristics such as CPU and memory percentage usage. The classification of the environment is essential to enable a fair comparison among the metrics of multiple existing compositions, ensuring that the perception of the system health of a given composition is compared to that of another composition in a fair way, given that both were exposed to the same operating conditions.

Problem Statement: The problem that emergent systems have to solve is to learn the composition of software units to best satisfy G at runtime. The system identifies its G satisfaction levels by analysing the collected metrics, choosing the most suitable com-

position by analysing metrics within the same operating condition, characterised by the collected events. The problem is aggravated by the need to involve multiple nodes to accomplish *G*. In a distributed scenario, the system must find the best global composition to satisfy *G* coordinating local composition across the available nodes.

To conclude, the Emergent Software System paradigm differentiates itself from other approaches to self-adaptive systems by incorporating an online learning approach to compose software systems. This strategy is a step beyond current approaches. It reduces the effort of creating the system, enabling the system to build its own understanding of its components and operating conditions (i.e., its adaptation logic). Furthermore, the main focus of Emergent Software solutions is in self-composition instead of self-adaptation. As a consequence of this change of focus, the Emergent Systems paradigm is expected to create software solutions capable of handling the increasing complexity of software systems by postponing decisions that are commonly made in the systems design phase, pushing software composition and design decisions to be made at runtime whilst the system is exposed to operating conditions.

1.3.3. Emergent Software Systems Challenges

The definition of the Emergent Software Systems concept imposes some challenges when implementing it. This section discusses the main challenges and their impact on the realisation of Emergent Software Systems and their adoption to create large-scale self-adaptive systems. These challenges were first introduced in [38].

Combinatorial Explosion: The Emergent Software vision is fundamentally based on a combinatorial learning process. These systems are built from combining smaller components into architectures and testing these architectures at runtime. As components are added to the repository, the number of possible architectural assemblies increase exponentially. This scenario becomes considerably worse when considering distributed systems, where the global system architecture results from combining a set of individual software architectures. Considering that adding one component to the repository exponentially increases the number of possible architectures, this property has a cascading effect in the distributed scenario when considering the amount of participating nodes in the system. The global system architecture is the overall combination of the micro-architectures of all participating nodes. To illustrate how quickly the search space grows in a distributed scenario, consider the following example: Two identical web servers with 50 compositions each running on two distinct nodes, and one load balancer with ten valid compositions running on another node. The load balancer is responsible for forwarding requests to both web servers. This system has 25,000 valid compositions ($50 * 50 * 10$). If we add another web server node with 50 compositions, the number of global compositions increases to 1,250,000 ($50 * 50 * 50 * 10$). This is an inherent property of emergent systems and a fundamental problem to be addressed in order to show the feasibility of the emergent systems paradigm and guarantee its consolidation.

Coordination: Emergent systems are responsible for i) composing the system using software units, ii) collecting perception data generated by the components at runtime and

iii) learning about the collected information. In a distributed setting, these systems can operate in different locations with different "personalities". The system can operate by controlling arbitrary groups of nodes in the system. They can control the operation of either individual nodes, in a completely decentralised fashion, or groups of any number of nodes, up to all system nodes, thus providing degrees of centralised control. Additionally, emergent systems can adopt different personalities, for example, by acting in an entirely selfish manner according to a group's local interests and ignoring the rest of the system, or by acting in an altruistic way, making local decisions to benefit global system's interests.

Behavioural Mismatch: In a distributed systems scenario, systems hosted in different nodes might implement behaviours that do not match other interacting nodes. For example, one system might encode different schemes of encryption algorithms, making the nodes incompatible with each other. A trivial solution for this problem is to manually define rules or impose constraints regarding aspects of the system behaviour. As discussed before, such a solution would be against the philosophy of emergent systems due to its restraining nature, limiting the reinforcement learning process. Emergent systems should have the freedom to explore and learn autonomously whilst simultaneously maintaining the system in a globally valid composition.

Relative Fitness Landscape: In emergent systems, fitness landscapes are graphs representing the fitness values of the available architectures executing on an operating environment. These graphs have an oscillatory shape where the highest peak represents the architectural composition with the highest fitness value, meaning that it is the optimal available architectural composition for the observed operating environment. Other valid architectural compositions are formed with lower fitness values as components are removed or replaced from the architecture with the highest fitness value. In the context of an emergent software system, for different operating conditions (external stimuli), the fitness landscape of the system changes, meaning that the fitness values of the available compositions change, including the choice of a new composition as the most suitable one when the environment changes. This scenario characterises emergent software systems as having **dynamic fitness landscapes**. In [8], Cakar *et al.* define two types of dynamic fitness landscapes, one influenced by external stimuli and the other influenced by the system's constituent components. Cakar *et al.* refer to changes in the fitness landscape caused by the system's internal elements as **self-referential fitness landscapes**.

Emergent software system's fitness landscapes suffer from external stimuli and the interaction of the components that form the system's architecture. The fitness value of a specific architecture is determined by the metrics and events generated by the system's components. Thus the perceived conditions are directly influenced by the executing composition itself. This scenario makes it difficult for the system to differentiate real changes from a distorted perception of the environment. The dynamic fitness landscapes of emergent systems in tandem with the distortion perception problem represent a crucial challenge to realise emergent systems.

Monitoring Data Quality: The poor quality of the metrics and events collected from the system's components may give the system a false perception of its performance and operating conditions, misleading the system to accumulate false knowledge, which may in turn trigger unnecessary adaptation or the selection of suboptimal architectures. In addition, as a related challenge, the absence of important perception data hides trends in the operating environment from the system. For example, the system may find its performance degrading over time with no apparent change in the operating condition. This scenario occurs when selected events do not match all essential aspects of the environment that impact the system's performance or when event averages are miscalculated, hiding fluctuations in the environment. This apparent mismatch between how the system perceives its environment and the actual environment directly impacts the emergent system's ability to locate an optimal architectural composition. Ideally, emergent systems should detect such blind spots so that the appropriate adjustments can be made to give the system a more precise perception of its environment.

1.4. Successful Projects

As a result of realising the Emergent Software System concept, we built a number of real-world systems to evaluate our approach. In this section, we provide an overall description of the three main projects developed to this date. These projects are open-source and available for download. All instructions to replicate the results are also provided with the code. We also provide references to previous publications where we detail the design, implementation and evaluation of each of these systems, and analyse how they realise the Emergent Software Systems concepts. The tools used to develop these projects are described in Sec. 1.5.

1.4.1. Emergent Web Server

The Emergent Web Server is a fully functional Web server that implements the HTTP 1.0 protocol and appropriately handles HTTP requests. The Web server was entirely developed according to the concept of Emergent Software Systems. It can adapt and evolve its internal composition to better handle requests according to the observed workload characteristics. Its design and implementation are fully described in [37, 38, 45], where it is used as a case study to explore and evaluate the concept. The emergent Web server is also presented in a publicly available video[1] that demonstrates its execution and performance.

The emergent Web server was initially explored with 42 unique ways to compose its internal architecture and was exposed to various workload patterns. The results show that the emergent Web server can converge towards the appropriate composition in a timely manner and learn the best performing composition for different classified workload patterns, being able to recompose its internal architecture to the best performing composition as soon as a previously seen workload pattern is recognised.

Later in this chapter (Sec. 1.5.4.1), the emergent Web server will be used as a target system for the reader to experiment and practice with the concept and tools to create Emergent Software Systems.

[1]To access the video, please visit: https://youtu.be/BBNvbC6w_3Y

1.4.2. Distributed Emergent Web Server

The Distributed Emergent Web Server is an extension of the emergent Web server. It also consists of a fully functional Web server that implements the HTTP 1.0 protocol and can serve content on the Web. The difference between this project and the emergent Web server is that it can delineate parts of its architecture and relocate them to other machines in a distributed infrastructure.

This project was used to explore learning in a distributed setting. The description of the project and the results are described and presented in [39]. The paper also presents the open-source code of the project to enable replication of its results. Besides the learning approach evaluated in the context of a distributed emergent Web server, this project also explores the potential of abstracting distributed compositions as actions to adapt the Web server into different architectural compositions to be learned at runtime.

The distributed emergent Web server can learn, at runtime, and with no predefined domain-specific knowledge, which Web server composition is the most suitable for the observed operating environment. Besides that, the distributed emergent Web server can also learn whether or not it requires running on multiple machines to handle the incoming workload volume.

1.4.3. Emergent Microservices

The Emergent Microservices project introduces a novel approach for the implementation of microservices using the concept of Emergent Software Systems. The project describes the use of the approach to create a fully functional real-world microservice that can appropriately handle incoming requests in a self-adaptive fashion.

The creation of Emergent Microservices aims at enabling microservices to adapt and evolve their internal architectural composition to autonomously cope with workload volume changes and workload pattern changes. The emergent microservice can evolve its internal architectural composition to better handle incoming requests at runtime, with no human interference.

The results of implementing and evaluating the approach are described in [43]. Rodrigues Filho *et al.* show that an emergent microservice can detect and adequately adjust its composition to accommodate changes in the operating environment. The paper also discusses the roles of engineers and machines in the design and evolution of future microservice-based systems. A more detailed description of the emergent microservice is presented later in this chapter (Sec. 1.5). We also use this project as a practical exercise to familiarise the reader with the Emergent Microservices concept and the use of Emergent Software Systems tools to build autonomic systems.

1.5. Perception, Assembly, Learning (PAL) Framework

This section discusses the realisation of the Emergent Software Systems concept. We present a framework that facilitates and supports the creation of emergent systems following the definition given in Sec. 1.3. This framework was introduced in [42].

Emergent Software Systems can autonomously compose software architectures from small and reusable software components according to the observed operating envi-

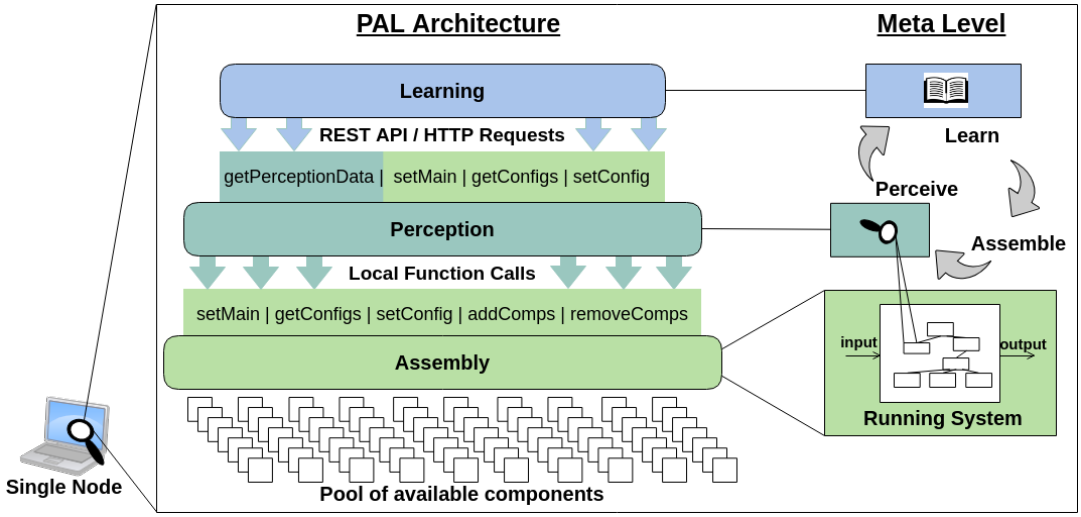


Figure 1.1. The PAL framework architecture [42].

ronment. To realise this concept, we built a framework with three main modules: **Perception**, **Assembly** and **Learning** (Fig 1.1).

The **Assembly** module searches for components in a repository, creates an in-memory representation of all available architectural compositions the system can be assembled into, and supports seamless adaptation from one composition to another at runtime. The **Perception** module, in turn, generates and adds proxy components to the system's architecture to monitor the system's health status and operating environment. Finally, the **Learning** module leads the entire autonomous software composition process through reinforcement learning approaches.

The interaction among the three modules is depicted in Fig. 1.1. The framework is organised in a multi-tier architecture in which the upper layers use the functions of the lower-level layers to implement their own functions and provide higher-level functionalities. The multi-tier modular architecture ensures that each participating module implements a well-defined set of functionalities to the upper level. Although the main modules are designed to be generic and useful in any application domain, this modular structure allows each module to be replaced by a variant, when applicable, to explore the specificities of particular domains. This characteristic makes the framework flexible, facilitating, for example, the experimentation and comparison of a variety of learning algorithms in the Learning module.

The three modules have well-defined roles in the realisation of the Emergent Software System concept. Their interaction results in a feedback loop that keeps the system constantly observing its health status and operating conditions and learning the optimal available system's composition for the observed environment. The following sections focus on the implementation details and the functionalities provided by the PAL framework.

1.5.1. Assembly Module

This section provides details on the Assembly module, covering the component-based model used to enable the concept of Emergent Software Systems and the functionalities

provided to the upper layers. The definition of Emergent Software Systems is based on the composition of fully functional software architectures from small components. To assemble components into functioning local architectures, the Assembly module searches for components in the repository, gathers information about them (and their variants) and assembles them into a fully functional system with composition options. The Assembly module uses functions provided by a component-based runtime to execute component replacement and composition at run time whilst abstracting the processes that support software architecture composition and (re)composition used by the Learning module to realise the reinforcement learning process. The following sections present the API functions provided by the Assembly module and describe the role of a component-based model in the autonomous composition process enabling architectural adaptation by component replacement.

Component-based models are essential to support the concept of Emergent Software Systems, mainly because they provide the necessary information about components to realise autonomous software composition. The information provided supports autonomous component connections to form functioning software architectures, avoiding random component compositions and offline testing to find compositions that work. Furthermore, since component models allow developers to express features through interfaces that are explicitly defined when coding components, the component-based model eliminates the need to use extra models to label the system code and define which parts of the system are adaptive and how to adapt them (e.g. what features could be replaced and how to replace them). For example, feature models (such as in [18]) are often used to represent how pieces of code interact with each other, and how they could be replaced to support adaptation. Such feature models force developers to detail the relationship between different parts of the system after implementing the system itself, increasing development effort. On the other hand, the component-based model is a way to eliminate that need, integrating essential information for component adaptation directly in the component's code. This characteristic reduces development effort during system design, facilitating software adaptation throughout the entire system rather than limiting it to specific parts that were previously modelled.

This work uses the Dana[2] component model described in [36]. Dana is a general-purpose programming language that inherently provides a component-based model and runtime support for component adaptation in fine-grained complex modular structures. Component-based models require the definition of interfaces and components. Interfaces define function signatures (i.e., function names, return types and the list of parameters with their respective data types). Each interface expects *at least one* component that implements all functions defined in it. Components can *provide* implementation for multiple interfaces and may *require* other interfaces to support their own implementation. Also, multiple components might implement the same interface using different approaches. Those are often referred to as **component variants**, and it is by means of replacing those variants in an executing software that the adaptation process occurs. This "provides-requires" policy is a central part of a component-based model, allowing the language

[2]Please refer to http://www.projectdana.com for more practical information on the language, its component-based model and its online adaptation mechanism.

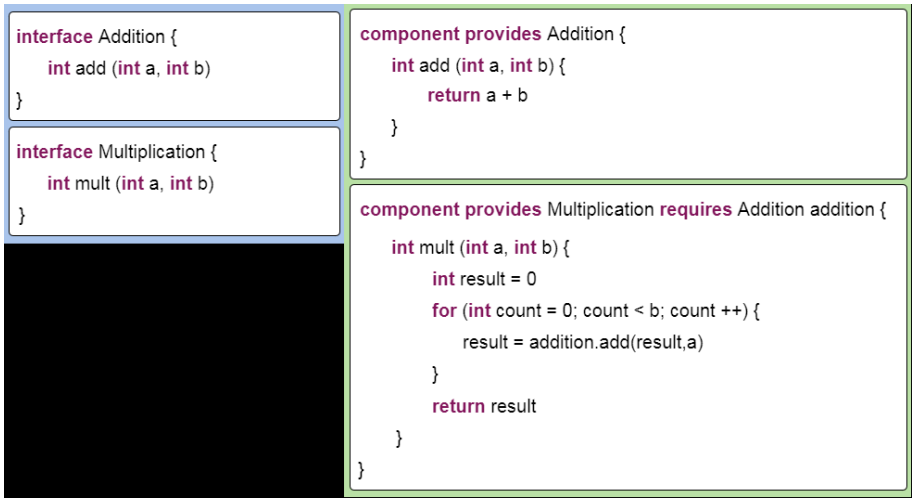


Figure 1.2. Example of interfaces and components. Note that Multiplication requires an external interface (Addition) to complete its implementation. The code is written using the Dana programming language syntax [42].

runtime to connect different components to create fully functional software. Software evolution or adaptation is realised by replacing components in the resulting software architecture at runtime without interrupting the system's services. Moreover, as part of the online replacement of a component, the system needs to examine the component's dependencies (i.e., its "requires" declaration) and perform further changes in its architecture to add any components that are required to support the new one.

An example of the Dana language syntax and its component-based model is show in Fig. 1.2. The figure shows two interfaces: *Addition* and *Multiplication*. Each interface may comprise a number of function definitions, where each function is defined by its name, return type, and a list of parameters with their types. The figure also shows two components that provide the implementation for the two interfaces. The *Multiplication* component is a simple example that illustrates a component implementation depending on another interface. In this case, the *Multiplication* component relies on the interface *Addition* to support its implementation. Therefore, whenever the *Multiplication* component is used, any component providing *Addition* is required to be connected to it. Considering the use of these interfaces and components to create a **Calculator** program, if multiple components provide the *Addition* interface, then any of those components can be connected to the *Multiplication* component to make it work. Furthermore, we could also assume the existence of a *Multiplication* component variant that, for example, does not require the *Addition* interface, thus implementing the *Multiplication* interface in a different way. Therefore, in this example, there exists a variety of architectural options to realise the **Calculator** program, each option implementing the operation *Multiplication* differently, by either connecting *Multiplication* with a variety of *Addition* components or replacing *Multiplication* with a variant that does not require the *Addition* interface.

The Dana runtime provides basic functions to abstract the online architectural composition and adaptation process. The abstractions provided by the language support, through single function calls, the realisation of software adaptation or composition, involving: `loading` new components into memory, `pausing` an old component's ex-

ecution, `transferring` component states (when applicable – i.e. *replacing* statefull components) and `connecting` a new component with its required components. Dana transparently executes the tasks mentioned above without further actions from the developer. The Assembly module supports these functions with low-level operations to determine and control the composition and adaptation process at runtime. A list of the functions provided by the Assembly module is detailed below.

The Assembly Module API:

- `bool setMain(char compName[])`: This function is responsible for starting the assembly of the target system's architecture. The Assembly module is capable of assembling an entire system's architecture and its variations from a main component.
- `String[] getConfigs()`: This function returns a list of the available architectural descriptions. These descriptions are string-based representations of architectures, containing a list of components and how these components are connected in the architecture. Furthermore, these descriptions are used by the Learning module to reason and learn about architectural compositions.
- `bool setConfig(char configDesc[])`: This function is used to change the executing architectural composition to another at runtime. This function receives as parameter the architectural description of the new composition.
- `char[] getConfig()`: This function returns the currently executing software composition.
- `bool removeComp(char compName[])`: This function removes a specific component from the list of components being used by the Assembly module to form the available software compositions. As a result, there is a decrease in the number of available software compositions. This function expects the name of the component to be removed.
- `bool addComp(char compName[])`: This function adds a component to the list of available components. As a result, there is an increase in the number of possible architectural compositions. This function receives the name of the component, searches for it in the repository and loads it into the Assembly module to create future architectural compositions.

The `setMain()` function sets the procedure to compose the software architecture and variations of it from a root component. The root component implements the main function, i.e., the point where the program starts its execution. From that component, the Assembly module extracts the name of the required interfaces. For each required interface, the Assembly module searches the repository for components that provide it. Suppose the Assembly module locates multiple components providing the same interface. In that case, it gathers the component variants (i.e., the multiple components providing the same interface). It attaches them to the interface creating an adaptation point, i.e., a point

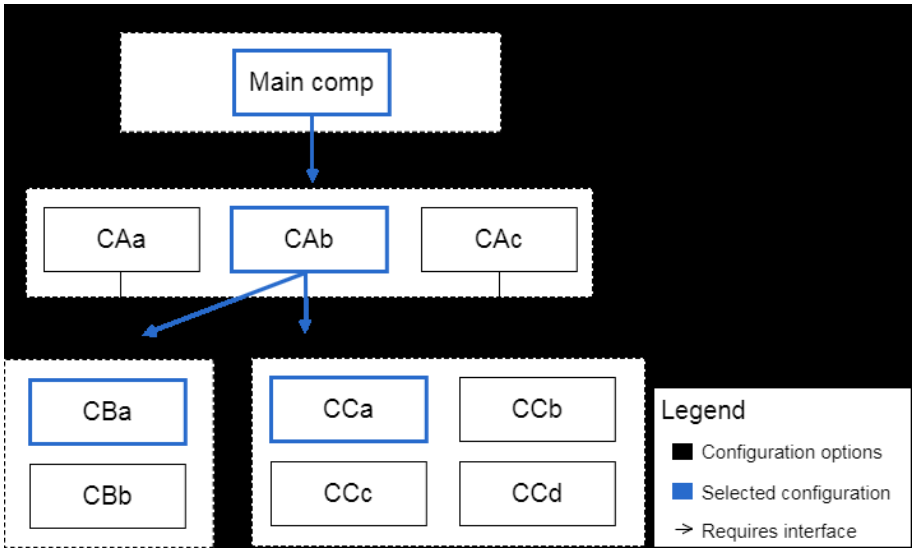


Figure 1.3. An example of a generic architecture represented by the Assembly module. This results from executing the Assembly module function *setMain* [42].

where there are multiple components to choose from. These adaptation points are represented in a tree, as shown in Fig. 1.3 for hypothetical interfaces *A*, *B* and *C*. After loading all components that provide the interfaces required by the main component, the Assembly module repeats the same steps for each of the loaded components. It continues with loading components and their variants for the required interfaces until the entire architectural tree is complete with the necessary components to realise at least one working software composition, i.e., having at least one component providing every required interface.

A working composition consists in the selection of one component in each required interface on the tree (an example is shown in Fig. 1.3). The components marked in blue in the figure are part of the currently executing composition. After loading components and their variants, the Assembly module selects and runs a random architectural composition. The example in Fig. 1.3 presents a total of 14 available architectural compositions. As show above, the Assembly also module provides functions to add new components (which were not in the repository when `setMain` was first executed) and to remove components as candidates to provide a specific interface, not allowing the removal of components that have no variants. These two functions influence the number of available architectural compositions. Furthermore, the Assembly module provides functions to return a list of architectural descriptions of the available compositions, a function to return the description of the currently executing composition, and a function to change from one composition to another. These functions work by acting on information about the available components, as structured in a tree (as exemplified in Fig. 1.3) and stored in the Assembly module.

The Assembly module provides architectural descriptions to enable external modules to reason about the running software structure and change the running composition. This text-based description provides information about the components that make up a specific composition, along with their relationships, i.e., how the components are connected. The architectural descriptions are helpful to the PAL framework for two main

reasons: i) they enable external modules to interact with the architecture variations built by the Assembly module; and ii) they enable external modules to infer and reason about information on the architectures. Thus, architectural descriptions facilitate the interactions between the other framework modules (mainly Perception and Learning) and the Assembly module. Since a description contains all the details to reassemble any architectural composition, they enable, for example, the Learning module to reason about the available architectural compositions to understand the impact of specific components in the architecture. Furthermore, the descriptions support the creation of different implementation versions of the Assembly module, in case a domain-specific version of the Assembly module may perform faster in special scenarios, enabling these different versions to internally represent compositions in different data structures whilst maintaining a consistent format for referencing valid software compositions.

An example of the architectural description is presented bellow:

$$|MainComp, CAb, CBa, CCa|MainComp : A : CAb, CAb : B : CBa, CAb : C : CCa|$$

This is the architectural description of the executing composition illustrated in Fig. 1.3. The description has two main parts: the list of components and the components relationships. On the left-hand side of the description, there is a list of all participating components in the architecture. In the example, the list of components is $MainComp, CAa, CBA, CCa$. The component relationships part of the description (on the right-hand side) describes how the participating components are connected. The first relationship, $MainComp : A : CAb$, tells that $MainComp$ is connected to CAb through the interface A.

1.5.2. Perception Module

The Perception module uses the Assembly module functions to enable the system to monitor its health status and collect information about the operating environment. Furthermore, the Perception module provides, through a RESTful API, both its own functions and the Assembly module functions to external modules interacting with Perception. This facilitates access to functions that support the learning process, such as functions to change to a new software composition, to get a list of possible software compositions, to add new components or remove components, and to get the executing composition description (i.e., Assembly functions), as well as functions to allow insertion, removal of monitoring components (proxies) to extract monitoring information from the executing system (i.e., Perception functions). This section is divided into two subsections: the first one describes the perception data (events and metrics), which represent the system's health status and classify operating environments. The second subsection describes a particular type of component (proxies) that are autonomously generated and inserted into the software architecture to collect metrics and events from executing compositions.

1.5.2.1. Perception Data (Events and Metrics)

The perception data (Events and Metrics) are data types through which the system represents its health status and operating conditions, which is essential information for the

```
data Metric {
    char name[]
    dec value
    bool preferHighValue
}

data Event {
    char name[]
    char type[]
    dec value
}
```

Figure 1.4. Metric and Event data types in the Dana programming language [42].

system to learn about its internal composition and execution environment. The learning algorithm uses these data to classify environments and determine the level of satisfaction of the system's goal, serving as the base for the system to learn and make design choices. The metrics are used to represent aspects of system health status (e.g., performance, security level, and so on), while events are created to store the values of environment features (e.g., input patterns, hardware characteristics). In tandem, the Learning module uses both events and metrics to realise its reinforcement learning approach. As an essential characteristic, both data types (see Fig. 1.4) were designed to capture information regardless of the application domain, enabling the PAL framework to learn about any target system by setting the appropriate event types and metrics to ensure a satisfying learning outcome.

The metric data type is composed of the `name`, `value` and `preferHighValue` attributes. The `name` attribute represents the aspect of the system that is being monitored. This field may be set as *'response time'* or *'the number of active threads'* or any other *string* that represents aspects of the system that can be quantitatively measured and monitored. The `value` attribute stores the corresponding value associated with the metric. For example, for *'response time'* the system may store a value in milliseconds representing the time that the system takes to process incoming requests. Additionally, metrics are used to represent system status and express the system's goals. The `preferHighValue`, when set to **true**, tells the system to find the architectural composition that maximises the `value` attribute. Contrarily, when `preferHighValue` is set to **false**, the system searches for the composition with the lowest `value` attribute. `name` and `preferHighValue` are often manually defined by a domain expert, and the system obtains the `value` as it executes. Note that it is possible to create multi-goal Emergent Systems using the concept of metrics. However, we focus on showing the feasibility of the Emergent Software Systems approach and not to explore the autonomous optimisation of multi-goal systems.

Events are used to represent different features of the operating environment. The event data type represents features with the following attributes: `name`, `type`, and `value`. The `name` attribute stores the name of the feature; for example, in cases where it is im-

portant to characterise request patterns, a possible event type could be named as *'request type'*. The attribute `type` defines the data type of the environment feature. In our request pattern example, this attribute could be set to *'text'* when the system received requests to retrieve text files, or *'image'* to represent different labels of the same *'request type'*. Finally, the attribute `value` quantifies the observed attribute. For example, it stores the size (e.g., in bytes) of the requested files. This allows the system to understand the operating environment it is running on, in this example, by understanding how differences in the file size of certain request types affect system operation. The `name` values are often manually defined at design time, while the `type` and `value` are collected from the executing target system.

After collecting events and metrics from the executing system, other important attributes can be inferred: the number of times a certain attributed was perceived (e.g., number of requests for text files). This could be essential information that assists the system in determining whether an event is recurring or not, which might be an essential aspect to characterise the operating environment. Another critical piece of information is the time when the metrics and events were collected. Time-stamped data enables the system to establish a timeline with information on how the system and the operating environment behave in a time frame. The collection of events and metrics is done by proxies, which are components that are autonomously generated to collect (and time-stamp) information about the system.

1.5.2.2. Proxy Components

Proxy Components are essential elements in the Perception module. They are responsible for extracting events and metrics from the system and operating environment. A proxy component complies with the same policy of other components in a component-based model. Thus, with regards to the Assembly module, it is indistinguishable from other components in terms of code structure, syntax, and connection to other components. The only difference is that proxies are annotated to allow the Assembly module to locate them and extract the collected metrics and events and to avoid connecting two or more proxy components.

A special characteristic of proxies is that they need to provide and require, at the same time, the interface the proxy is supposed to monitor. By requiring and providing the same interface, the proxy component can be inserted between two components; one that requires the interface (e.g. component A) and another that provides the interface (e.g. component B). Then the proxy can intercept function calls from A to B and extract information from their interaction. Fig. 1.5 illustrates a proxy component inserted in the system.

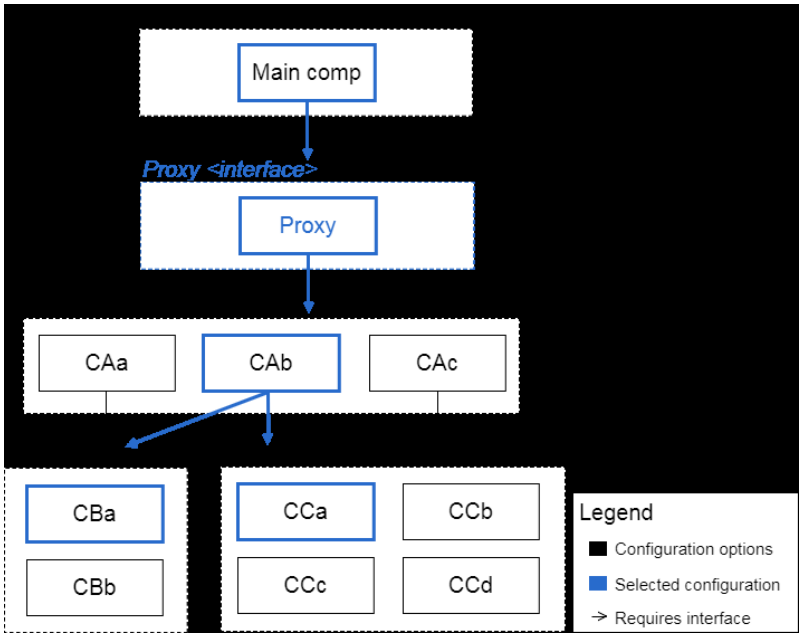


Figure 1.5. Proxy component inserted in the system's architecture [42].

The Perception module was designed to support a proxy-based monitoring solution, mainly because proxies are non-intrusive, i.e., they do not require code to be inserted into the monitored component to extract information for the system. Thus, it is possible to separate the code responsible for monitoring a specific component from the component itself, not requiring component developers to code functions to monitor the component, avoiding it becoming incompatible with the applied learning algorithm. Furthermore, this separation of concerns (component code and monitoring code) also makes the component development process more flexible, eliminating the need for a model to dictate how components should be coded to enable their monitoring.

This proxy-based approach implemented by the Perception module also enables the autonomous generation of proxy components at runtime, according to the necessity and goals of the Learning module. Proxy generation is further detailed next. Furthermore, considering that a Proxy component is generated to monitor target interfaces rather than components, and that multiple components might require the same interface throughout the system, a Proxy Expression Language was designed to allow the Learning module to express precisely what interface to place the proxies in the system structure. The Proxy Expression Language is also described in detail in what follows.

The Proxy Generation Process gives the system flexibility by autonomously generating the appropriate proxy components for the goals of the Learning module, regardless of the system, the application domain or the components to be monitored. The Learning module triggers the proxy generation process. Once it obtains the system goals and the available architectural compositions, the Learning module triggers the proxy component generation process. This generation process requires as input the target interface and components responsible for generating the metrics and events to be collected from the target

interface components.

The Perception module provides a range of components that implement the collection of specific metrics and events and allow the creation of new metric/event-collector components. For example, to collect the response time in milliseconds, component `ResponseTime` offers the code necessary to collect such metric. This component provides a *timer* that is triggered right before the function is invoked and stopped right after the function finishes processing, returning the calculated time the function took to execute. The same applies to collecting event information. An example event is, in the context of a system that handles HTTP requests, the collection of MIME types from incoming requests. The component `MimeType` is also implemented and provided. This component provides a function that receives the request in a raw-text format and returns the extracted mime-type of the requested resource. Both components described above are provided by the Perception module. As new components of this kind are implemented, the Perception module makes them available to be used in the generation of proxy components.

The metrics and events generated in the proxy are timestamped once created and stored in an object named `Container`. This object is responsible for averaging the metrics and events of the same type, updating the 'count' variable to express the number of occurrences of a specific event or metric, and making them available to the Perception module, which, in turn, formats them into a JSON string as **perception data** to send to the Learning module upon request. Note that averaging event and metric values is essential to reduce the amount of data stored in the proxy component.

Proxy Expression Language: Proxy components focus on monitoring interfaces rather than particular components. The reason is that a proxy designed to monitor an interface monitors a standardised feature and thus can monitor all component variants that provide such feature, independent of individual component implementations. Furthermore, suppose that the system adapts its architectural composition, from an implementation of the monitored component to one of its variants. In that case, the Perception module can replace the component without replacing the proxy, reducing the number of changes performed by the Assembly module. The main problem of monitoring interfaces rather than components is that an interface can be required by multiple components in the software architecture. As a result, when placing a proxy to monitor an interface, the Perception module actually needs to place multiple proxies to monitor *every* component that requires that interface throughout the architecture, thus impacting the performance of the system. The Proxy Expression Language was designed to provide fine-grained control over the process of proxy placement, while avoiding the need for the Learning module to indicate what component (instead of interface) should be monitored every time an architectural change occurs.

The Proxy Expression Language (PEL) is a tool to precisely express where to place proxy components in the architectural composition. The Learning module uses this language to ensure that the generated proxy components are kept in place, monitoring the intended interface even as the software compositions are constantly changing. The Perception module is responsible for interpreting the expression and making a list of architectural compositions without proxy components, associating these compositions with

their equivalent compositions with proxy components added at the right places (interfaces or components) according to the expression created by the Learning module. Therefore, whenever the Learning module requests the software composition to be changed, the Perception module compares the new architecture description with the list of equivalent compositions and decides. Suppose the new composition matches one of the compositions in the list. In that case, the Perception module changes the software composition to the corresponding new composition with the proxy component in the right place according to the expression. This strategy enables the Learning module to work with architectural descriptions with no added proxy components. The Perception module is responsible for applying the PEL expression provided by the Learning module once (before it starts the experimentation process) and transparently adding proxy components to the appropriate places in the software architectural composition.

The language provides levels of control to the Learning module when placing proxy components in the system, allowing it to place proxies for only one specific component, or for all components that require an interface, or any level between the two, for example, by placing a proxy component to monitor an interface but only if a specific component requires that interface (as opposed to every component that requires that interface). The Learning module can thus create expressions as generic or as specific as needed, having the Perception module to use that expression against the architectural description to determine the place to add the proxy components in the software architecture. Adding multiple proxy components to different software architecture parts gives a clearer perspective of the system status and its operating environment. However, it adds overhead to the processing and collection of events and metrics, impacting system performance. Thus, an essential challenge in adding proxies to the software architecture is identifying the number of proxy components and their optimal placement to reduce the number of proxies whilst maximising the quality of collected data. Considering that, in an Emergent Software Systems architecture, a component calls functions on lower-level components (as shown in the component graph of Fig. 1.3), by placing proxies at the highest possible level, the monitoring proxy can calculate the time a stack of function calls takes to execute. This is not a general rule and may vary according to the architecture. An exception to this scenario is when, at some point in the function call stack, a function is called to be executed in a different *thread*.

1.5.2.3. The Perception Module API

The Perception Module API is presented below:

- `void addProxy(char exp[])`: This function receives as parameters an expression in the Proxy Expression Language (PEL) format that determines where the proxy should be inserted into the software architecture. At the end of this function's execution, the user should expect the monitoring proxy to be inserted into the systems architecture;
- `void removeProxy(char exp[])`: This function removes inserted monitoring proxies. The function receives as parameters the same expression used to add the proxy in the system and uses it to remove the proxy;

- `char[] getPerceptionData()`: This function returns the collected metrics and events from the executing system in a JSON format;

These are the main functions provided by the Perception module. Note that, due to the dependency of the Perception module from the Assembly module (the Perception modules uses the Assembly module functions to implement their own), any component that interacts with the Perception module will have access to both the Perception module functions and the Assembly module functions.

1.5.3. Learning Module

The Learning module is responsible for guiding the learning process, which is crucial to the realisation of Emergent Software Systems. This module uses the lower level modules (Perception and Assembly modules) to trigger the deployment and composition process and to request Proxy generation by selecting Metrics and Events generators according to the system goals. It controls the entire online learning process, triggering the exploration and exploitation phases, classifying the operating environment and identifying the most suitable architectural composition. This section describes the learning process (exploration and exploitation phases), showing three learning strategies: i) the *Baseline* approach, ii) the *Feature-based* approach, and iii) two *Multi-armed bandit* approaches. Furthermore, this section also describes an environment classification algorithm.

The learning process executes with no prior knowledge about the target system nor any information about the operating conditions to which the system will be exposed. The main task of the Learning module is to understand the correlations between the assembled collection of components (the system's behaviour) and the system's perception of its performance in each identified operating condition. The Learning module executes its main task by requesting the Assembly module to change the system's architectural composition in order to experiment with the identified changes in the operating environment. It then observes its performance and the operating conditions through the Perception module.

1.5.3.1. Learning Algorithms for Emergent Software Systems

The generic approach to realise learning in Emergent Software Systems involves three main tasks. Firstly, the system must be able to characterise and classify features in its operating environment (derived from the stream of events being emitted) so that the performance of different compositions can be compared in equivalent environments and so that the learning module can "remember" which compositions work best in each environment (i.e., to save re-learning each time a recurring environment is encountered). Secondly, after finding the most suitable architectural composition for the perceived environment, the system exploits the optimal composition whilst observing the environment and the its own performance to detect any changes and trigger learning again, in case it detects unforeseen changes. Finally, as in any online learning system, the learning module must balance the trade-off between exploring options for which there is insufficient information and exploiting options already known to be good [49]. This third task is essential because the emergent software framework operates on live software, and sub-optimal performance has real consequences.

Performing online reinforcement learning is highly challenging. The software is not in control of its operating environment and cannot know in advance when it may reliably compare any two software compositions against the same operating condition. Moreover, there are complex interactions between the exploration process itself and the environment, as selecting a "good" composition may impact the system performance and change the perceived environment. The explored learning approaches are reinforcement learning [49] algorithms tailored to our particular problem space. The learning approaches presented in this section continually discover optimal assemblies by exploring the search space while simultaneously classifying observed features of the operating condition into labelled environments. The *Baseline* approach explores all available compositions before opting for the best performing option. This approach is presented as a baseline to compare with other learning strategies because it is guaranteed (when operating conditions are maintained during exploration) to find the optimal global composition. The *Feature-based* approach, on the other hand, explores the search space analysing features instead of individual compositions. Thus it reduces the search space by focusing on representative compositions of available features. This approach relies on domain-specific assumptions and is scalable to support learning in large search spaces (i.e., with a large number of system composition options). Finally, the *Multi-armed bandit* approaches differ from the above strategies by adopting better strategies to balance the exploration and exploitation phases. They also consider fluctuations in the performance metrics collected from the live system, making these approaches more robust. If a drastic fluctuation on a performance metric of the system occurs, the baseline and the feature-based approaches may converge towards a suboptimal system composition.

The Baseline Algorithm applies a standard "exploration activity" to both characterise the current environment and identify the best composition for that environment. The system triggers exploration whenever it encounters high uncertainty in its decision-making process – where this uncertainty comes either from (i) having no information at all (i.e., after system startup), (ii) the current environment characteristics deviating outside the expected ranges from existing experience, or (iii) current system performance deviating beyond its expected range.

The exploration activity tries every possible composition for a fixed-length "observation window" w_t, such that the total time spent exploring is:

$$w_t * \texttt{length(getConfigs())}$$

The observation window w_t is defined according to the application domain. A reasonable time frame value (w_t), based on previous investigation, is 10 seconds for high quality learning results. The `getConfigs()` function is provided by the Assembly module and returns all available software compositions. After trying every composition, the learning module then characterises the data collected over the entire exploration process to determine the best course of action.

Specifically, after an exploration activity, the learning module selects the best-performing composition for use and enters its exploitation phase. The selected action continues to be monitored and analysed for its suitability every w_t time units. A change

in the operating environment is detected if either (i) perceived events during w_t show that this is a different event pattern or (ii) perceived metrics during w_t show degraded performance. This detected change in the operating environment also triggers a change in the learning behaviour, forcing it to start exploring again. The algorithm waits for $w_t * 3$ of consistently observed behaviour to avoid frequent oscillation between exploitation and exploration phases before changing its current course. In case (i), if the detected event pattern has been previously seen, the best matching composition is simply selected. In all other cases, new exploration activity is triggered. This process of exploration/exploitation repeats continually, where the amount of exploration reduces as fewer new environmental conditions are seen. Note that this learning algorithm, based on an exhaustive exploration phase, is not designed to scale up to large systems with thousands of compositions but rather serves as a proof-of-concept and useful baseline against which to compare more sophisticated algorithms.

The Feature-based learning strategy is an alternative solution for the Learning module. Instead of experimenting with all available architectural compositions, the system experiments with representative architectures for a specific feature. A feature is a functionality defined by an interface and, therefore, in this context, the words feature and interface are used interchangeably. A variety of components can be created to provide a single feature by implementing the same functionality differently, requiring, for example, other features (sub-features) in their implementation. The algorithm exploits that property by choosing the component variant for a specific feature F considering the sub-features required by the component variants of F.

Furthermore, the algorithm does not try all variants of all features (interfaces) in the system's architecture. Instead, it only tries component variants for the most suitable features, testing only one component variant for features that were not suitable for the operating environment. Therefore, this approach reduces the search space and supports a faster learning process, providing a more scalable solution. However, this approach relies on the assumption that the worst component variant for the best feature is better than any component of the other features, which might not be true for every application.

This strategy maintains the essence of the reinforcement learning algorithm, i.e., the entire process described in the baseline strategy still applies. The only part of the algorithm that the feature-based strategy changes is the selection of the next composition to be tested. In the baseline approach, the algorithm tests them all. The feature-based approach navigates through the software architecture from top to bottom, deciding on the component variants it encounters for each feature (interface) with multiple component variants. Once it determines the best component variant for a feature, it traverses the tree downwards considering only the chosen component variant, eliminating the branches that correspond to the remaining (not selected) variants. This process continues until the algorithm decides on every component variant of every feature it considers relevant, resulting in the optimal discovered architecture. The rest of the algorithm, including the environment classification and the details of the exploitation phase, are the same as in the baseline approach.

The description of the *Feature-based* approach as well as an evaluation of its per-

formance is detailed in [42]. The algorithm considers a tree-like structure that represents all architectural compositions for the software, as illustrated in Fig. 1.3. Considering a tree-like structure representing the available architectural compositions, the algorithm selects the first interface (down from the root) with component variants. Note that Emergent Systems only have composition options `if and only if` there exist component variants for at least one required interface. The system has no option for required interfaces with only one component implementing their functionalities. Therefore, the algorithm navigates the tree from top to bottom, determining the best component variant only for the required interfaces with more than two alternative components implementing them. The algorithm tests all interfaces (that have component variants) required by a selected component at a higher level, ignoring the interfaces required by components that were not selected at any level of the tree. The selection of component variants in each required interface is carried out by executing architectural compositions containing the component variants and selecting the variant in the best performing architecture. Once a component variant is selected, it becomes part of the "optimal" composition. Then the algorithm moves on to the next interface with component variants, following the branch defined by the selected component. This process is repeated until there are no more interfaces with component variants, making the composition containing all selected components the "optimal" composition.

A quick performance comparison between the *baseline* and *feature-based* algorithms shows that the feature-based strategy converges much faster. Considering the abstract architectural compositions of Fig. 1.3, the *baseline* approach takes $w_t * 14$ time (2.3 min for w_t = 10 secs) to explore all possible compositions. For the *Feature-based* strategy, considering the worst case scenario, the algorithm explores compositions in $w_t * 11$ time (1.8 min for w_t = 10 secs). Considering the best case scenario, the *Feature-based* approach takes $w_t * 5$ time (50 secs for w_t = 10 secs), in case the best feature is B (component CAa) rather than $B + C$ (component CAb). The average case, in this example, is when the best feature is C (component CAc), resulting in an execution of $w_t * 7$ (1.16 min for w_t = 10 secs). In this particular example, the best case scenario for the feature-based approach has a significant advantage over the baseline approach. The worst case scenario, however, is not as significant, having a difference of only 30 secs (and not guaranteeing global optimality). However, if we consider that, for every exploration phase the baseline approach always executes in 2.3 secs (considering w_t = 10 secs), the feature-based search is still very advantageous. A more expressive scenario, using a real system as example, is described and explored in [42], showing the advantages of the *Feature-based* search approach.

So far, the presented algorithms were specifically developed to realise the Emergent Software Systems concept. Next, we present two pre-existing algorithms, describing their use to build Emergent Software Systems.

Multi-armed Bandit is a well-known problem in statistics with many proposed algorithms to solve it [5, 9, 48, 50]. The solutions consist in balancing the exploration and exploitation phases. This balance is widely studied and very important in reinforcement learning strategies. In the multi-armed bandit problem, we consider multiple slot ma-

chines, just like those found in casinos, that give a particular reward whenever their arms are pulled. Each machine has a probability distribution function over the presented reward, meaning that each time an arm is pulled, a different reward may be presented. Some machines may be more inclined to have a higher probability of presenting high rewards. The problem consists in discovering which machine is most likely to yield the highest average reward. The probability distribution function of each machine is not previously known. The solution should explore the available machines to get their rewards and learn their probability distribution functions. However, as the solution gathers information on the rewards of some machines, it should also *exploit* the machine (or set of machines) which are known to yield high rewards up to that point.

Modelling the Emergent Software Systems learning problem using the multi-armed bandit model is useful for two main reasons. Firstly, it allows the use of well-studied algorithms known to provide the solution. The advantage of using such algorithms is that they have theoretical guarantees regarding convergence towards the optimal solution. Although the theoretical convergence occurs in infinite time, it is guaranteed that the algorithm will eventually converge towards an optimal solution. Secondly, the fluctuations over the reward signal match the fluctuations of collected metric values from running systems that are never fixed on a specific value but rather slightly fluctuates over a range. That means that when an executing system is queried for its execution metrics (the metrics that allow the evaluation of its performance) the metric readings are never a fixed number, it always varies slightly as it is queried.

In the context of emergent software systems, pulling an arm means that an action is being performed on the running system to change its architecture from one composition to another. The bandit algorithms explore the available systems compositions at runtime to find the composition that yields the best performance. Considering that the system is in production, the algorithm must carefully balance the exploration (i.e., finding the best-performing composition) and exploitation (i.e., taking advantage of the best-performing composition) phases. This balance is crucial to avoid overly affecting the performance of the running system while it handles incoming user requests.

The two main bandit algorithms used to realise Emergent Software Systems are **Upper Confidence Bound (UCB1)** [4] and **Thompson Sampling** [9]. This section describes the working mechanisms of these two solutions and how they are used in the context of realising emergent systems.

Upper Confidence Bound (UCB1) is a well-known and widely used algorithm to solve the multi-armed bandit problem. The algorithm works as follows: it starts by taking action (i.e., pulling an arm), then it waits until it receives a reward, it normalises the reward, and then it calculates what action should be taken next based on its equation, for every single available action. The action that yields the highest value as a result of calculating the equation becomes the following action to be taken. The equation is described and shown below:

$$r_k + \sqrt{\frac{2 \ln n}{n_k}}$$

The variable k indicates a certain action. It ranges from $0 < k \leq$ *total number of*

actions, each action in an emergent systems context is a unique architecture composition. The variable n indicates the total number of times the algorithm performed actions (i.e., selected any architecture composition). The n_k term indicates the number of times action k was selected. The first part of the equation defined by r_k represents the average reward of choosing action k. The squared root part of the equation determines the algorithm's confidence level on a certain action k. The algorithm calculates the result of the equation for every single available action. The action that yields the highest value from the equation becomes the algorithm's next action. This process repeats while the system is executing, and at some point in time, when the algorithm converges, the action that yields the highest average reward will always be chosen from that point on. After convergence, the average reward value for every action becomes too large, and the squared root part of the equation (i.e., the confidence level of the algorithm on each action) becomes too small that it cannot force the algorithm to take any other action, expect the best one.

The confidence level is inversely proportional to the squared root part of the algorithms' equation. That means that as the algorithm gains confidence in a specific action, that second part of the equation yields a minimal value, and therefore it interferes less on the action that will be chosen. On the other hand, if actions are neglected (i.e., not chosen for a long while), the confidence level of the algorithm to take such actions drops, meaning that the resulting value of the squared root part of the equation increases, forcing the algorithm to choose the action. In this particular regard, the equation dictates the primary mechanism on which each UCB1 algorithm operates, making it balance between exploring and exploiting the available actions as it increases the algorithm's confidence in the returned reward of a specific action.

We refer the reader to [39] for further detail on the approach, including an example and the evaluation results of its use in the Learning module of the PAL framework.

Thompson sampling is another algorithm that solves the multi-armed bandit problem. The difference between this algorithm and UCB1 is in the mechanism of balancing the exploration and exploitation phases. Instead of being guided by the UCB1 equation, each action in the Thompson sampling approach is selected with the probability of it being the best action given the history of the algorithm's actions selection.

In detail, the algorithm considers a probability distribution over the reward of each action with densities given by bell curves. That means that the centre of the bell curve is history of the average reward on that action, and the spread of the curve is the level of uncertainty (i.e., high uncertainty is a result of few attempts at a specific action). For an action to be selected with Thompson sampling, the random sample from its bell curve must be higher than the corresponding samples from all other actions. This is true if either the centre point is high or if the spread is large, corresponding respectively to the high average observed rewards or high uncertainty.

As a result, the algorithm selects the most likely actions or actions that are likely to perform well, but the algorithm has less information (i.e., the algorithm is less confident about). The rest of the algorithm is very similar to the UCB1; the algorithm selects an action, observes its reward, normalises its reward, determines which action should be taken next according to its probability of yielding high rewards, then selects the next action and repeats. At some point, as the system executes, the algorithm gathers enough

information on the probability distribution of each action, so that it knows with a high degree of certainty which action will yield the highest reward.

For an example of the use of this approach to create an emergent web server, please see [37].

1.5.3.2. Environment Classification

Environment classification is an important part of the learning process. The Learning module classifies operating environments for two main reasons: i) to compare architectural compositions exposed to equivalent external stimuli to guarantee fair comparisons, and ii) to 'remember' the best software composition for previously seen conditions to avoid unnecessary re-learning. The process to classify environments is very challenging. That is mainly because the executing architectural composition may distort the perceived conditions. These distortions create the illusion of changes in the environment. For example, the system may perceive a workload volume increase, when in reality, it is the running architecture that can process more requests in less time. This phenomenon is described in Sec. 1.3 and it is referenced as self-referential fitness landscapes. This section describes the environment classification algorithm used in tandem with the *Baseline* and *Feature-based* learning approaches.

After collecting information about the explored architectural compositions, the Learning module receives a list of *Events* and *Metrics* attached to each executed composition. Based on that information, the system classifies the operating environment. This approach minimises the effects of environment distortion by classifying environments using ranges. The algorithm creates ranges for each collected Event and Metric, with minimum values defined by the lowest perceived values for the event or metric and maximum values defined by the highest value registered by the compositions, as illustrated in Fig. 1.6.

Fig. 1.6 illustrates the classification of two operating environments obtained from the tests with the emergent Web server; these results are detailed in [45]. Consider a Web server program and the operating environment as the patterns of requests handled by the server. The first environment consists of large-size text file requests with low variations, i.e., the majority of text files requested were repeated files. The second environment, on the right hand side of the figure, consists of large-size image file requests with low variation (i.e., large volumes of requests to repeated files).

During the exploitation phase, the system has already selected the best option to execute under the perceived operating environment and only observes metrics and events collected from the executing architectural composition. If the collected values are within the ranges established by the environment class, and if there is no extra event type and value (e.g., within a pattern of text-only requests, a few video requests start appearing), then the system understands that the environment has not changed. On the other hand, if the collected values are outside the range of any event type or metric (e.g., the system has a significant performance decrease), then after three iterations (the threshold to trigger exploration or architectural changes), the system compares the values it perceives with previously defined environments, in case these values are within all the ranges established

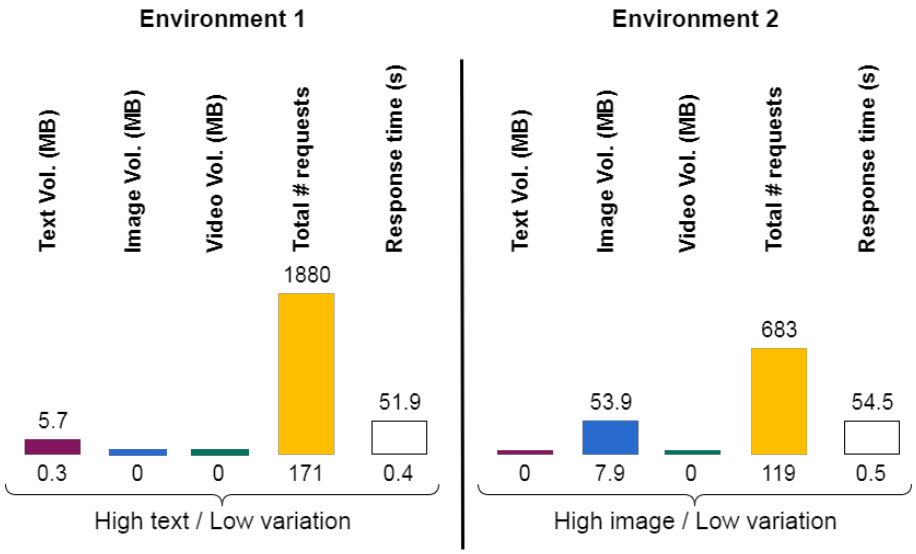


Figure 1.6. Example of classified environments using ranges of collected *Event* and *Metric* values. The first environment consists of requests of large-size text files with low variation. The second environment consists of large-size image files with low variation [45].

in classified environments, the system changes to the composition attached to the new environment. Otherwise, the system triggers exploration.

This range-based classification approach has some limitations. For a more accurate classification of environments, for instance, this approach works best when there are no changes in the environment during exploration, as it is demonstrated in [45]. Also, it is difficult to "remember" environments when there are overlaps among their defined ranges, preventing the system to accurately determine the operating environment based only on data collected from a single architecture (which is often the case to trigger exploration or composition changes during exploitation phase). Furthermore, as with any current machine learning approach, this approach suffers when essential features of the environment are not defined as Events, leaving, for example, two or more distinct environments to be classified as the same. As previously mentioned, the implementation of the Emergent Software Systems learning approach is very challenging, with several open issues to be addressed. Therefore, further research is required to better explore and overcome the range-based environment classification issues.

1.5.4. Practical: Using the PAL Framework

This section covers the main ideas and concepts of a set of practical exercises that aim to familiarise the reader with the presented framework. Note that all the details of the practical exercises along with the code for the PAL framework and the detailed instructions on how to install all the necessary software to run the exercises are available on Github[3]. The framework and the target system were developed using the Dana programming language. All code presented in this section is written in the Dana programming language[4].

[3]https://github.com/robertovrf/sbrc21_minicurso
[4]https://projectdana.com

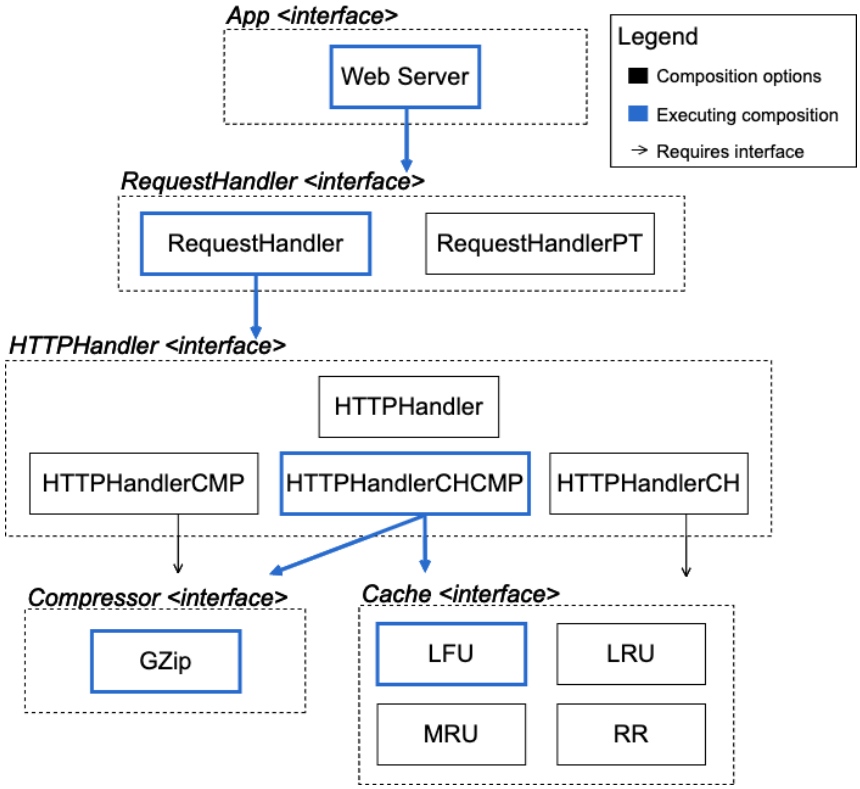


Figure 1.7. Emergent web server architecture.

The practicals presented in this section aim at providing hands-on experience to the reader on creating emergent systems. We begin by describing the target system on which practical will be based. We introduce the components used throughout the practical and describe the module that allows the target system to learn to self-adapt without human interference or predefined domain-specific knowledge.

1.5.4.1. Emergent Web Server

This section focuses on the emergent Web server as the target system. The Web server is developed using the Dana programming language following its component-based model. The Web server consists of four primary interfaces that define its main functionalities. There are component variants for some of these four interfaces, i.e., components that implement the same interface but in a slightly different manner. These component variants enable the recomposition of the web server architecture, maintaining its main functionalities but behaving in a way that may impact the Web server's performance when subjected to different request patterns.

Fig. 1.7 shows the Web server architecture. The outer dotted boxes represent the interfaces, while the inner boxes represent the components that implement the respective interface. The main interfaces are the *RequestHandler*, *HTTPHandler*, *Compressor* and *Cache*. The *RequestHandler* interface defines how the Web server handles TCP requests. For that particular interface, there are two components that implement that functional-

ity. The first component, also named *RequestHandler*, creates a thread to handle each TCP request. The component variant, named *RequestHandlerPT*, maintains a pool of threads instead and assigns incoming TCP connections to an existing thread to handle the incoming request.

The interface *HTTPHandler* defines the functions necessary to handle HTTP requests. For that interface, there are four component variants: *HTTPHandler*, which implements the functions to handle HTTP requests with no additional optimisations; *HTTPHandlerCMP*, which compresses the response before sending it to the client; *HTTPHandlerCH*, which caches the response after sending it to the client; *HTTPHandlerCHCMP*, which compresses the response before sending it to the client and caches the compressed response in memory.

The remaining interfaces, *Compressor* and *Cache*, define the functionality of both compression and caching. Both interfaces are used by the components that implement the *HTTPHandler* interface and require compression and/or caching. For the compressor interface, the *GZip* component implements the gzip compression algorithm. For the cache interface, in turn, each component variant implements a different cache replacement algorithm (to replace cached items when the cache is full and a new item needs to be cached). The cache implementations are: Least-Frequently Used (*LFU*), Least Recently Used (*LRU*), Most Recently Used (*MRU*) and Random Replacement (*RR*).

The set of components illustrated in Fig. 1.7 are used to compose a functioning Web server capable of serving files located in the *htdocs* folder. To compose a fully functional Web server, one component for each of the interfaces must be part of the application architecture. The adaptation of the Web server consists of swapping at runtime one component to a variant component that implements the same interface. For instance, we enable the adaptation of a Web server composition that uses the *HTTPHandler* component variant that implements the *HTTPHandler* interface to the *HTTPHandlerCH* that caches the response after sending the response to the client. In the new composition, the Web server performs better when the workload pattern has many repeated requests that result in the same response.

In the Github repository that hosts the code for the practicals, the reader will find the Web server code in the repository folder and the instructions to run it with a specific fixed composition. In the next section, we describe a component that enables the reader to interact with the Web server and its various compositions.

1.5.4.2. Interactive Emergent System (IES)

The Interactive Emergent System (IES) is a command-line tool that provides access to the PAL framework. Its goal is to allow the user to interact with the framework modules through the command line to inspect and control a running emergent system. In this part of the practical, we expect the reader to run IES and use it to compose the emergent Web server (explored in Sec. 1.5.4.1), change it from one composition architecture to another as it executes, and collect response time information from the executing Web server.

To execute IES, it is necessary to provide, as parameters, the path to the root com-

ponent of the Web server. A detailed description of how to execute IES to start the web server is provided in the companion Github repository. Once the IES starts, it calls the Assembly module function *setMain()*, which triggers the Assembly module to search for the components to compose the Web server, including its several architectural compositions. After the Web server is loaded and one of its compositions is executing, the IES presents a command prompt that enables interaction with the framework functions.

The user can call any function of the ESS framework modules through the IES command prompt. For instance, the user can list all Web server compositions found by the Assembly module (through the Assembly function *getAllConfigs()*) and change the executing Web server architecture to any of the available compositions in the list (*setConfig(config)*). The user is also able to query the running Web server (*getConfig()*) and collect information on the performance of its current composition (by calling the Perception function *getPerceptionData()*). Note that the Web server executes normally during the IES execution, and clients can issue HTTP requests to it at any time. Requests to the Web server can be issued from a regular web browser or through a client program that is also available in the repository.

We expect the reader to explore all functions provided by the PAL framework modules and familiarise themselves with the capabilities provided by the framework. Next, we explore the execution of the PAL framework to realise the emergent Web server, showing the potential of the emergent software systems framework to autonomously compose systems and learn the most suitable composition for different operating conditions.

1.5.4.3. Emergent Web Server

This section presents the third and practical exercise, which provides hands-on experience executing the emergent Web server. This exercise consists in executing the PAL framework targeting the Web server and experimenting with different operating conditions for the target system as it learns the best composition for each operating condition. We expect the reader to execute the framework and observe how it operates. We welcome the reader to develop different operating conditions (by creating different clients for the Web server) or to create new components to allow for further architecture variants. After such additions to the experiment, we expect the reader to test the framework to see if it locates the best composition for the new setting.

In this practical, the reader should execute the emergent Web server using the Interactive Emergent System tool. After the Web server is loaded, the reader should start a client program that generates a workload for the server. As the web server handles the requests, the reader should start the learning process and observe the system as it learns the optimal composition with no predefined domain-specific knowledge or human interference. Once the learning converges, the reader can choose a different client that generates a different workload pattern, in order to see if the learning can detect the change and converge to the new optimal composition.

We also encourage the reader to interact with the IES tool and manually find the optimal best composition for a given workload pattern generated by an available client program. This will give the reader an idea of how challenging it is for the learning al-

gorithm to perform the same task. We also encourage the reader to think of different learning approaches that could be used to make the learning module converge faster to an optimal composition by writing and experimenting with new learning strategies. A detailed description to execute this practical is in the Github companion repository.

1.6. Emergent Microservices

This section introduces the concept of Emergent Microservices [43]. This concept stems from the direct application of the Emergent Software Systems concept to build microservices able to autonomously evolve their internal architecture to accommodate changes in the workload pattern. We present the concept of microservices and motivate the need for emergent microservices. We then explore the concept in the domain of smart cities, which is known for having highly volatile execution environments. We conclude the section by presenting a set of hands-on exercises to familiarise the reader with the Emergent Microservice concept. The code for this practical is also available in the GitHub companion repository[5], along with detailed instructions to run the example. In the practical, we also expect the reader to extend the example by creating new emergent microservices and testing the concept in different domains and operating conditions.

1.6.1. Microservices

Before we dive into the concept of Emergent Microservices, we first introduce the general concept of microservices. Microservices has gained popularity in the industry and has been used as building blocks to create large-scale and flexible systems. Due to its increasing popularity, the concept has become very important in modern distributed systems and has been extensively used to build systems in various domains (e.g., Web-based systems, video streaming services, supporting platforms for IoT-based systems).

In short, microservices can be defined as self-contained small services that implement a single functionality of a larger system [29]. As a result, instead of implementing a system as a massive, highly interconnected monolith, the system can be broken down into smaller self-contained, easier-to-manage and highly reusable services. These characteristics make microservice-based systems highly maintainable, easier to evolve, highly reusable and adaptive, which are essential characteristics to create modern systems.

Each microservice is self-contained, which means that they can be deployed and run independently of other parts of the system. A microservice provides a RESTful API, with its endpoint URI, HTTP method, expected parameters, and expected result. We show an example of a microservice API function documentation below:

```
HTTP Method: GET
URI: https://189.234.22.12:80/collector/get_data
HTTP status code: (200) OK,
                  (400) Bad Request or
                  (500) Internal Server Error
Response description: Collected data (JSON format)
Response example:
```

[5]https://github.com/robertovrf/sbrc21_minicurso

```
{"resources": [ {
  "uuid": "ae9cf502-5ed2-47d4-914c-c1caec1c41c4",
  "capabilities": {
      "environment_monitoring": [{
          "temperature": "38.313",
          "humidity": "38.313",
          "date": "2016-06-21T23:27:35.000Z"}]
}}]}
```

Microservices are often deployed on cloud-based infrastructures, packaged inside containers that are managed by container-orchestration systems. The container-orchestration systems are fundamental to manage microservices in real deployments. They offer support for elasticity both horizontal and vertical, and system recovery when the container fails, triggering the creation of a new instance. These are important actions to cope with the dynamism of the system's operating environment, which constantly changes and demands the system to adapt.

1.6.2. Emergent Microservices

Microservice supporting tools, such as container orchestration systems, are well equipped to handle workload volume variation. To cope with the increasing user demands, they may create new replicas of the microservices, or increase the amount of computing resources available to the container running the microservice. However, they are not able to efficiently cope with variations that also involve changes in the workload pattern.

Changes in the workload are common in current systems. In general, these changes may affect the workload volume (e.g., the number of requests issued in a time frame), which may increase or decrease according to user demand, and the workload pattern (e.g., from requests for text-based data to requests for image-based data). Changes in the workload pattern often impact microservice performance depending on the type of functionality or data being returned. In that sense, some workload patterns require the evolution of the microservice's internal implementation to better accommodate the new pattern. For instance, if there is an increase in the size of text-based data returned by the requests to a microservice, compression of the returned data may be an effective way to improve the service's performance, enabling it to better accommodate the new workload.

Such improvements in the service implementation required by a change in the workload pattern are often conducted by the developer. The process typically consists in having an expert analyse the service logs to determine the characteristics of the new workload pattern and how they impact the service's current implementation. Then, developers are required to implement, test and deploy a new version of the service. This process is entirely human-dependent and often inefficient. Emergent Microservice [43] is a concept that aims to support the autonomous evolution of a microservice's internal composition as a response to workload pattern changes. It represents a more efficient way to evolve a microservice's internal architecture at runtime, automating the process described above. We argue that this is not only convenient but crucial to support the development of future distributed systems.

The Emergent Microservice concept is realised by applying the Emergent Software Systems framework to a component-based implementation of microservices. To facilitate microservice implementation, we developed an emergent microservice framework on top of components that implement the Web server. The framework is illustrated in Fig. 1.8 and is composed of a set of components that implement the business logic of the microservice, along with utility components that assist their implementation. Examples of utility components are implementations of common data structures (e.g., hash map, lists, queue, trees) and database drivers (e.g., MongoDB, MySQL).

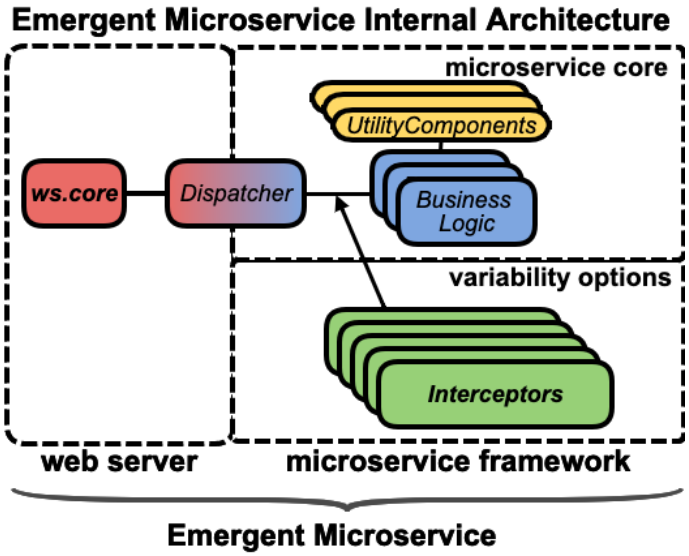


Figure 1.8. Internal architecture of an emergent microservice, showing the main components and how they connect. (Figure adapted from [43]).

.

Apart from the business logic and utility components, interceptor components comprise an essential part of the emergent microservice's framework. They intercept the requests sent to the business logic components and alter their processing to change a specific non-functional concern. For instance, a microservice that constantly accesses a database to return historical unchanged data can improve its performance by caching the data in memory. In that context, a cache interceptor can be transparently inserted into the microservices internal composition to boost its performance when needed.

Interceptors are thus a transparent method of adding variants to the microservice's internal composition. This allows the development of dynamically adaptive microservices without extra effort from the developers. They are only required to provide the core implementation of the microservice by implementing their business logic components, possibly making use of the utility components. A set of predefined interceptors are provided, adding standard variants to the microservice's internal composition. The emergent software systems framework uses these variants to compose the microservice and learn at runtime the most appropriate composition for it.

Interceptors, however, are not the only way to add variation to the emergent microservice's internal architecture. Any component variant added to the framework might be picked up by the PAL framework and used to generate different internal architectural

compositions for the microservice. If there are different ways to implement the microservice's business logic, implementing the microservices functions differently is also a way to create different microservice architectural variants.

Once the components and their variants are implemented, the deployment process of an emergent microservice is similar to the process of deploying regular microservices. Often these microservices are deployed on a cloud computing environment, using containers and container-orchestration technologies. The process often involves containerising the microservices components and their variants as well as the PAL framework. As the container starts, the PAL framework assembles the first microservice composition and starts learning, as incoming user requests are handled by the microservice, the best composition for the observed request pattern. As a regular container deployment managed by a container-orchestrator, the microservice is also susceptible to be replicated as the volume of workload increases and the service performance hits a predefined threshold. This process of configuring container-orchestration systems to manage the microservice container is better described in the practicals (Sec. 1.6.4).

1.6.3. Case Study: Emergent Microservices in the InterSCity Platform

The Smart Cities domain was chosen as a use case to illustrate the Emergent Microservice concept due to the highly volatile execution environments that it entails. In particular, we focus on the InterSCity platform [13]. It is an open-source microservice-based platform to support smart cities applications. The platform aims at abstracting the interaction between applications and city resources, providing a layer capable of scaling to handle the application and data processing demands of large cities. The platform architecture is composed of six microservices, as shown in Fig. 1.9.

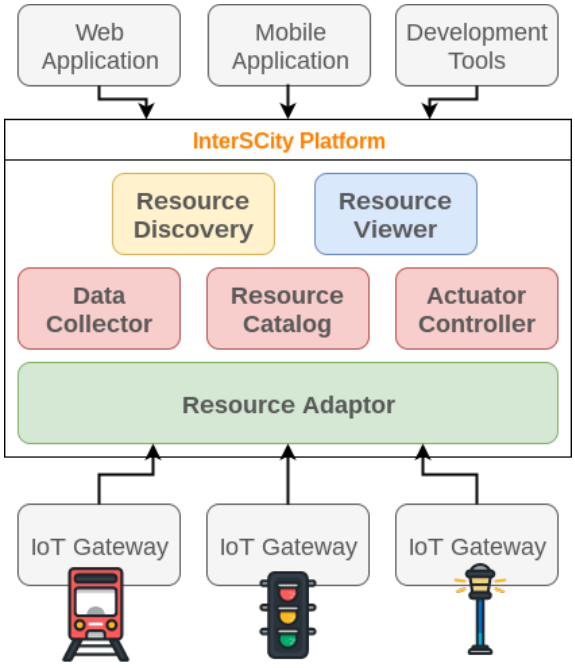


Figure 1.9. InterSCity platform microservices (https://interscity.org).

Each of the InterSCity microservices has a specific role. *Resource Adaptor* is the microservice responsible for receiving incoming requests from devices, redirecting those requests to the appropriate microservice. *Resource Catalog* is responsible for storing information about all devices available in the city. *Resource Discovery* provides information about specific devices. *Data Collector* is responsible for providing applications with access to all data collected from city devices, as well as providing the devices with a gateway to push their collected data. *Actuator Controller* is the microservice that provides access to the city actuator devices. Finally, *Resource Viewer* is responsible for showing visual representations of available city resources.

We selected the Data Collector (DC) microservice to redesign it as an emergent microservice for this case study. DC was chosen for being a crucial microservice in the platform, responsible for handling all data access generated by city applications. Thus, it plays a crucial role in the global performance and scalability of the system.

The functionality of the DC microservice is provided in terms of four operations:

- Retrieve all historical data in the database;
- Retrieve all historical data collected from a specific city resource (e.g., a bus line information);
- Retrieve the most recent data collected;
- Retrieve the most recent data collected from a specific city resource (e.g., the latest location of a specific bus);

To realise the emergent DC, we implement these operations on the business logic components of the emergent microservice framework (described in Sec. 1.6.2). The MySQL drive component is used as a utility component to provide the microservice with access to a database. As interceptors, we make available components that implement caching, compression, and both caching and compression (in the same component). They are used to generate the DC internal composition variants.

The architecture of the DC microservice, with all its components and variants is shown in Fig 1.10. The *Dispatcher* component forwards incoming HTTP requests to the *DataCollector* component for processing. This component is responsible for implementing the core functionality of the DC microservice. The interceptors, in turn, are the components that add variants to the DC microservice and enable it to have its behaviour optimised according to the pattern of incoming requests.

In total, the interceptors enable the creation of four unique microservice internal architectural compositions. In the default composition of the DC microservice, the *Dispatcher* is directly connected to *DataCollector*, and the microservice performs as in its original, non-emergent implementation. The second composition is obtained when the *Cache* interceptor is plugged between the *Dispatcher* and *DataCollector* components. This interceptor caches all responses sent back to clients. It increases the microservice performance when clients frequently request the same sets of data – caching the response in memory increases the microservices performance because it prevents the service from

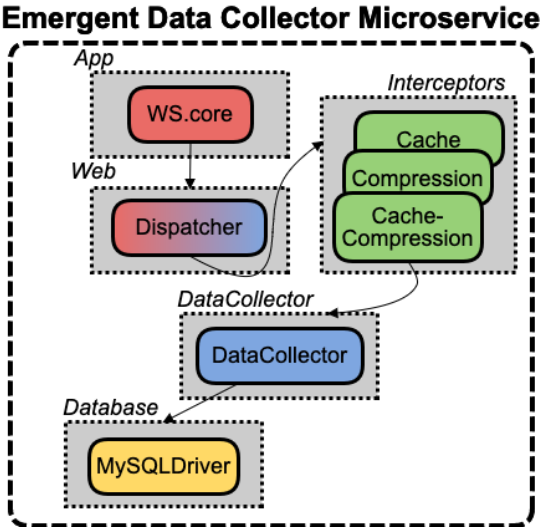


Figure 1.10. Data Collector Emergent Microservice. (Figure adapted from [43]).

connecting and retrieving data from a remote database. The compression composition, in turn, is obtained by placing the *Compression* interceptor in between the *Dispatcher* and *DataCollector* components. This interceptor compresses all responses before sending them back to the client. This composition may increase the microservice's performance when the returned data is large and highly compressible. Under these conditions, response compression may dramatically decrease the network transmission time of the response. Finally, the *CacheCompression* interceptor caches the compressed returned response. This composition improves the microservice performance in scenarios where clients request repeated data that are large and highly compressible.

We experimented with the emergent DC in a cloud-based environment to validate the concept. The result is illustrated in Fig. 1.11. We executed the emergent DC and two other implementation of DC: *Default* and *Cache-Compression*. The *Default* composition consists of a regular implementation of the DC microservice with no extra components to improve its performance. On the other hand, the *Cache-Compression* implementation consists of adding both a *Cache* and *Compression* components to the DC microservice. This last implementation compresses every response before sending it back to the client, and after sending the response, it caches the compressed response in memory. We subjected all three microservices to two different workloads, *Workload A* and *Workload B*. These workloads have distinct characteristics. *Workload A* consists of repeated requests to a specific small set of data (i.e., a workload where the cache composition would increase the performance of the system). In contrast, *Workload B* consists of requests to completely new data (i.e., a workload that would make cache-based compositions perform badly).

Fig. 1.11 shows that the *Default* composition has the worst performance when subjected to *Workload A*, but it has the best performance when subjected to *Workload B*. The graph also shows that *Cache-Compression* performs significantly better than *Defult* when subjected to *Workload A*, but it performs worse when subjected to *Workload B*. This result is expected due to the characteristics of the experimented workloads and the imple-

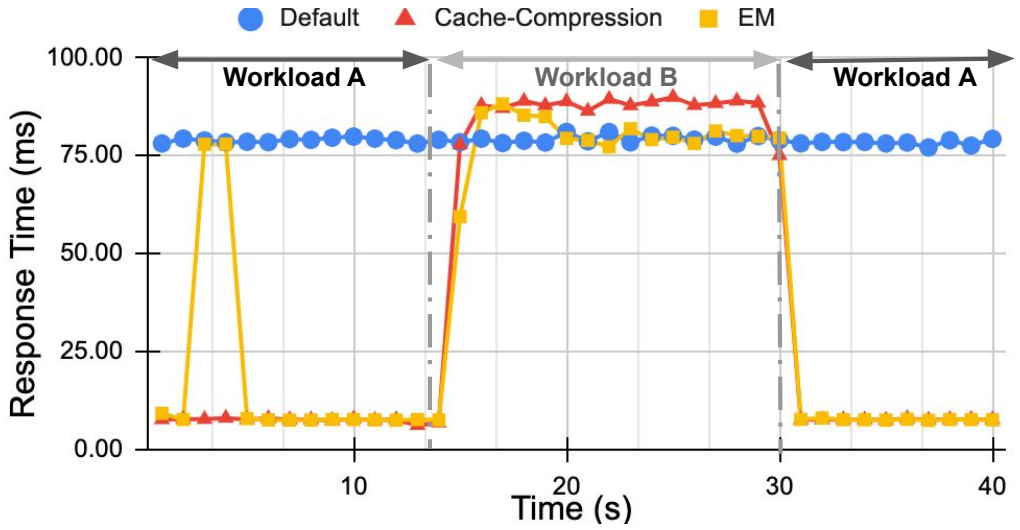


Figure 1.11. Average response time for three versions of the same microservice when subjected to two distinct workloads (A and B). EM learns and converges to the best performing composition for both Workload A (cache-compression) and Workload B (default). At 30s, EM identifies a previously seen workload pattern and is able to immediately adapt its composition.

mentation details of both microservices. The graph also demonstrates the ability of the emergent DC (the EM yellow square-dotted line) that after the learning phase (represented by the spikes on the yellow line) converges towards the best performing composition for both *Workloa A* and *B*. The graph finally shows that when the emergent DC is subjected to an already seen workload (from time 30s onwards on the graph – *Workload A* is executed again), emergent DC 'remembers' what the best composition is for *Workload A* and quickly changes its composition to *Cache-Compression* without the exploration phase.

As a result of creating and evaluating an emergent version of the DC microservice, we demonstrate that a microservice can react and evolve its internal composition when the incoming request pattern changes. Based on the application of the Emergent Software Systems concept and framework to realise Emergent Microservices, we can create microservice-based systems that are not only capable of coping with changes in the workload volume (through the use of supporting tools such as autoscalers on a cloud-based infrastructure), but are also capable of quickly and autonomously reacting in the face of workload pattern changes. Next, we describe a practical exercise that shows the necessary steps to execute the emergent DC microservice and subject it to various workload patterns in order to observe its dynamic self-adaptability.

1.6.4. Practical: Using PAL to Realise Emergent Microservices

After reading the previous sections (Sec. 1.6.2 and Sec. 1.6.3), where we introduce the concept and framework of Emergent Microservices (EM), along with its realisation using the PAL framework, we invite the reader to access the companion repository[6] and download the code and instructions to execute the emergent DC.

In the repository, the reader will find the emergent DC source code, along with

[6]https://github.com/robertovrf/sbrc21_minicurso.

the code for the PAL and emergent microservices frameworks, with instructions to run the system both on the cloud and on a local machine. If choosing to run it on a cloud-based infrastructure, the reader will find a set of scripts and a detailed tutorial to generate the emergent microservice Docker image, as well as the Kubernetes[7] scripts (YAML) to deploy emergent microservices. Otherwise, if the reader chooses to execute the microservices on a local machine, we make available an image of a virtual machine that comes configured with all necessary software. We also make available a set of client programs that simulate user behaviour and define different workload patterns for the reader to observe how the emergent DC learns at runtime, and without human interference, which microservice composition has the best performance.

If the reader has access to a cloud infrastructure, they will also find the instructions to run their emergent microservice in tandem with the Horizontal Pod Autoscaler (HPA). HPA is a supporting tool that observes the execution of a container and, in case the container's resource consumption increases above a predefined threshold, creates further replicas of the executing container. This is the state-of-the-art technique that has been widely used in production to autonomously handle sudden increases in workload volume.

The use of HPA in tandem with emergent microservices offers the potential to deal with both workload volume increase and workload pattern fluctuations. As the workload changes in pattern, the emergent microservice can learn, at runtime, the best performing composition for the detected workload pattern before HPA trigger any creation of replicas. As the workload volume increases, HPA can increase replicas to cope with the new volume. In tandem, emergent microservices can keep all replicas to their best performing composition according to the observed pattern. This, in turn, may lead to a decrease in the number of necessary replicas (i.e., resources usage) to cope with the new volume. Although this is a promising idea, it has not been fully explored in the literature.

The code available in the repository enables the reader to experiment and create their very own microservice. We encourage the reader to write different microservices and different component variants for their own microservices, as well as for the DC microservice, to familiarise with the concept. We also motivate the reader to use our available code to build on our work and solve the challenges that remain to be solved when realising the concept of emergent software systems in general and emergent microservices in particular. We list a set of research challenges and directions that the reader may consider in case they are interested in further exploring this research topic:

- Exploration of new learning approaches to better classify operating environments;
- Exploration of new learning approaches that enable better interaction among emergent microservices and other supporting tools that exist in the industrial microservice ecosystem (e.g., horizontal and vertical autoscalers);
- Explore the composition of large-scale systems using emergent microservices. Building systems with thousands or more emergent microservices;

We hope the community joins us in developing this novel technology that has

[7]https://kubernetes.io.

much potential to support the creation of future Internet systems. We believe that applying the PAL framework to build microservice architecture-based systems is a promising way to support the creation of large-scale self-adaptive systems that can handle the increasing dynamism of modern systems.

1.7. Conclusion

This chapter aimed to disseminate the concept of Emergent Software Systems, motivate the approach and advertise the tools to realise the concept. This chapter also presents the case for applying the concept to create real-world, fully functioning adaptive-systems. In particular, we presented an emergent web server and an emergent microservice. We also presented practical exercises to familiarise the reader with the concept and with the creation of real-world emergent software applications.

We have presented the concept of Emergent Software Systems as an approach to facilitate the development of autonomic computing systems. The concept consists in applying lightweight component-based models and reinforcement learning algorithms to create systems able to learn, at runtime, the most suitable available composition for the observed environment. Using these two technologies to build such systems enables the creation of everyday software systems that are emergent and adaptive. We also enable self-adaptive systems to cope with unknown conditions as our approach is shown to learn at runtime with no predefined domain-specific information.

This concept was successfully used to create real-world systems. It was explored in data centre-based applications such as the emergent Web server, the distributed Web server and emergent microservices. All information about the concept and these projects have been published elsewhere by the authors, as referenced throughout the chapter.

The chapter also presented the challenges and potential research directions to develop the concept further, hoping for a more significant adoption by the self-adaptive/ autonomic systems research community and industry practitioners. We finish the chapter by inviting everyone interested in developing future systems to join the effort to further develop and explore the main ideas that define the concept of Emergent Software Systems and their potential in supporting the creation and management of future systems.

Acknowledgements

This research is part of the INCT of the Future Internet for Smart Cities, funded by CNPq (grant 465446/2014-0), CAPES (grant 88887.136422/2017-00), and FAPESP (grants 14 /50937-1 and 15/24485-9). Dr. Rodrigues Filho would also like to thank FAPESP for supporting his research under grant 2020/07193-2.

References

[1] D. Al-Jumeily, A. Hussain, and P. Fergus. Using adaptive neural networks to provide self-healing autonomic software. *International Journal of Space-Based and Situated Computing*, 5(3):129–140, 2015.

[2] M. Amoui, M. Salehie, S. Mirarab, and L. Tahvildari. Adaptive action selection in autonomic software using reinforcement learning. In *Autonomic and Autonomous*

Systems, 2008. ICAS 2008. Fourth International Conference on, pages 175–181. IEEE, 2008.

[3] P. Arcaini, E. Riccobene, and P. Scandurra. Modeling and analyzing mape-k feedback loops for self-adaptation. In *Proceedings of the 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, pages 13–23. IEEE Press, 2015.

[4] P. Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3(Nov):397–422, 2002.

[5] D. A. Berry and B. Fristedt. Bandit problems: sequential allocation of experiments (monographs on statistics and applied probability). *London: Chapman and Hall*, 5(71-87):7–7, 1985.

[6] G. Blair. Complex distributed systems: The need for fresh perspectives. In *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, pages 1410–1421. IEEE, 2018.

[7] E. Bruneton, T. Coupaye, M. Leclercq, V. Quema, and J.-B. Stefani. An open component model and its support in java. In *Component-Based Software Engineering*, volume 3054, pages 7–22. Springer Berlin Heidelberg, 2004.

[8] E. Cakar, S. Tomforde, and C. Müller-Schloer. A role-based imitation algorithm for the optimisation in dynamic fitness landscapes. In *Swarm Intelligence (SIS), 2011 IEEE Symposium on*, pages 1–8. IEEE, 2011.

[9] O. Chapelle and L. Li. An empirical evaluation of thompson sampling. *Advances in neural information processing systems*, 24:2249–2257, 2011.

[10] B. H. C. e. a. Cheng. *Software Engineering for Self-Adaptive Systems: A Research Roadmap*, pages 1–26. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[11] M. Clarke, G. S. Blair, G. Coulson, and N. Parlavantzas. An efficient component model for the construction of adaptive middleware. In *IFIP/ACM International Conference on Distributed Systems Platforms and Open Distributed Processing*, pages 160–178. Springer, 2001.

[12] D. Costa and A. Hertz. Ants can colour graphs. *Journal of the operational research society*, 48(3):295–305, 1997.

[13] A. M. Del Esposte, F. Kon, F. M. Costa, and N. Lago. Interscity: A scalable microservice-based open source platform for smart cities. In *SMARTGREENS*, volume 1, pages 35–46, 2017.

[14] Y. Ding, H. Sun, and K. Hao. A bio-inspired emergent system for intelligent web service composition and management. *Knowledge-Based Systems*, 20(5):457–465, 2007.

[15] M. Dorigo, M. Birattari, and T. Stutzle. Ant colony optimization. *IEEE computational intelligence magazine*, 1(4):28–39, 2006.

[16] S. Duan, V. Thummala, and S. Babu. Tuning database configuration parameters with ituned. *Proceedings of the VLDB Endowment*, 2(1):1246–1257, 2009.

[17] A. E. Eiben, J. E. Smith, et al. *Introduction to evolutionary computing*, volume 53. Springer, 2003.

[18] A. Elkhodary, N. Esfahani, and S. Malek. Fusion: A framework for engineering self-tuning self-adaptive software systems. In *Proceedings of the Eighteenth ACM SIGSOFT International Symposium on Foundations of Software Engineering*, FSE '10, pages 7–16, New York, NY, USA, 2010. ACM.

[19] D. Fisch, M. Janicke, B. Sick, and C. Muller-Schloer. Quantitative emergence–a refined approach based on divergence measures. In *Self-Adaptive and Self-Organizing Systems (SASO), 2010 4th IEEE International Conference on*, pages 94–103. IEEE, 2010.

[20] L. Gambardella and G. Taillard. A multiple ant colony system for vehicle routing problems with time windows.[in:] corne d., dorigo mp: New ideas in optimization, 1999.

[21] J.-P. Georgé and M. P. Gleizes. Experiments in emergent programming using self-organizing multi-agent systems. In *CEEMAS*, pages 450–459. Springer, 2005.

[22] S. Götz, T. Kühn, C. Piechnick, G. Püschel, and U. Aßmann. A models@run.time approach for multi-objective self-optimizing software. In *Adaptive and Intelligent Systems*, pages 100–109. Springer, 2014.

[23] P. Grace, D. Hughes, B. Porter, G. S. Blair, G. Coulson, and F. Taiani. Experiences with open overlays: A middleware approach to network heterogeneity. In *Proceedings of the 3rd ACM SIGOPS/EuroSys European Conference on Computer Systems 2008*, Eurosys '08, pages 123–136, New York, NY, USA, 2008. ACM.

[24] V. Issarny, A. Bennaceur, and Y.-D. Bromberg. Middleware-layer connector synthesis: Beyond state of the art in middleware interoperability. In *International School on Formal Methods for the Design of Computer, Communication and Software Systems*, pages 217–255. Springer, 2011.

[25] K. Jeong and R. Figueiredo. Self-configuring software-defined overlay bypass for seamless inter-and intra-cloud virtual networking. In *Proceedings of the 25th ACM International Symposium on High-Performance Parallel and Distributed Computing*, pages 153–164. ACM, 2016.

[26] J. O. Kephart and D. M. Chess. The vision of autonomic computing. *Computer*, 36(1):41–50, 2003.

[27] D. Kim and S. Park. Reinforcement learning-based dynamic adaptation planning method for architecture-based self-managed software. In *Software Engineering for Adaptive and Self-Managing Systems, 2009. SEAMS'09. ICSE Workshop on*, pages 76–85. IEEE, 2009.

[28] O. Kouchnarenko and J.-F. Weber. Adapting component-based systems at runtime via policies with temporal patterns. In *Formal Aspects of Component Software*, pages 234–253. Springer, 2014.

[29] X. Larrucea, I. Santamaria, R. Colomo-Palacios, and C. Ebert. Microservices. *IEEE Software*, 35(3):96–100, 2018.

[30] G. Liao, K. Datta, and T. L. Willke. Gunther: Search-based auto-tuning of mapreduce. In *Euro-Par 2013 Parallel Processing*, pages 406–419. Springer, 2013.

[31] C. Lu, Y. Lu, T. F. Abdelzaher, J. A. Stankovic, and S. H. Son. Feedback control architecture and design methodology for service delay guarantees in web servers. *IEEE Transactions on Parallel and Distributed Systems*, 17(9):1014–1027, 2006.

[32] S. Malek, N. Esfahani, D. Menasce, J. Sousa, and H. Gomaa. Self-architecting software systems (sassy) from qos-annotated activity models. In *Principles of Engineering Service Oriented Systems, 2009. PESOS 2009. ICSE Workshop on*, pages 62–69, May 2009.

[33] P. K. McKinley, B. H. Cheng, A. J. Ramirez, and A. C. Jensen. Applying evolutionary computation to mitigate uncertainty in dynamically-adaptive, high-assurance middleware. *Journal of Internet Services and Applications*, 3(1):51–58, 2012.

[34] D. Merkle, M. Middendorf, and H. Schmeck. Ant colony optimization for resource-constrained project scheduling. *IEEE transactions on evolutionary computation*, 6(4):333–346, 2002.

[35] C. Müller-Schloer and S. Tomforde. *Organic Computing-Technical Systems for Survival in the Real World*. Springer, 2017.

[36] B. Porter. Runtime modularity in complex structures: A component model for fine grained runtime adaptation. In *Component-Based Software Engineering*, pages 26–32. ACM, June 2014.

[37] B. Porter, M. Grieves, R. Rodrigues Filho, and D. Leslie. RE$^{\text{X}}$: A development platform and online learning approach for runtime emergent software systems. In *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation*. USENIX, 2016.

[38] B. Porter and R. Rodrigues Filho. Losing control: The case for emergent software using autonomous perception, assembly and learning. In *Proc. of the 10th IEEE International Conf. on Self-Adaptive and Self-Organizing Systems*, 2016.

[39] B. Porter and R. Rodrigues Filho. Distributed emergent software: Assembling, perceiving and learning systems at scale. In *2019 IEEE 13th International Conference on Self-Adaptive and Self-Organizing Systems (SASO)*, pages 127–136. IEEE, 2019.

[40] A. J. Ramirez, A. C. Jensen, B. H. Cheng, and D. B. Knoester. Automatically exploring how uncertainty impacts behavior of dynamically adaptive systems. In *Proceedings of the 2011 26th IEEE/ACM International Conference on Automated Software Engineering*, pages 568–571. IEEE Computer Society, 2011.

[41] U. Richter, M. Mnif, J. Branke, C. Müller-Schloer, and H. Schmeck. Towards a generic observer/controller architecture for organic computing. *GI Jahrestagung (1)*, 93:112–119, 2006.

[42] R. Rodrigues Filho. *Emergent Software Systems*. PhD thesis, Lancaster University, 2018.

[43] R. Rodrigues Filho, M. P. de Sá, B. Porter, and F. M. Costa. Towards emergent microservices for client-tailored design. In *Proceedings of the 19th Workshop on Adaptive and Reflexive Middleware*, pages 1–6, 2018.

[44] R. Rodrigues Filho and B. Porter. Experiments with a machine-centric approach to realise distributed emergent software systems. In *Proceedings of the 15th International Workshop on Adaptive and Reflective Middleware*, pages 1–6, 2016.

[45] R. Rodrigues Filho and B. Porter. Defining emergent software using continuous self-assembly, perception, and learning. *ACM Transactions Autonomic Adaptive Systems*, 12(3):16:1–16:25, Sept. 2017.

[46] S. J. Russell and P. Norvig. *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,, 2016.

[47] H. Schmeck. Organic computing-a new vision for distributed embedded systems. In *Object-Oriented Real-Time Distributed Computing, 2005. ISORC 2005. Eighth IEEE International Symposium on*, pages 201–203. IEEE, 2005.

[48] S. L. Scott. A modern bayesian look at the multi-armed bandit. *Applied Stochastic Models in Business and Industry*, 26(6):639–658, 2010.

[49] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 1998.

[50] W. R. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294, 1933.

[51] S. Tomforde, B. Sick, and C. Müller-Schloer. Organic computing in the spotlight. *arXiv preprint arXiv:1701.08125*, 2017.

[52] E. Yuan, S. Malek, B. Schmerl, D. Garlan, and J. Gennari. Architecture-based self-protecting software systems. In *Proceedings of the 9th international ACM Sigsoft conference on Quality of software architectures*, pages 33–42, 2013.

[53] W. Zhang, T. Wood, and J. Hwang. Netkv: Scalable, self-managing, load balancing as a network function. In *Autonomic Computing (ICAC), 2016 IEEE International Conference on*, pages 5–14. IEEE, 2016.

[54] W. Zheng, R. Bianchini, and T. D. Nguyen. Massconf: automatic configuration tuning by leveraging user community information. In *ACM SIGSOFT Software Engineering Notes*, pages 283–288. ACM, 2011.

Capítulo 2

Revisitando as ICNs: Mobilidade, Segurança e Aplicações Distribuídas através das Redes de Dados Nomeados

Leobino N. Sampaio[1], Allan E. S. Freitas[2], Italo V. S. Brito[1,3], Francisco Renato C. Araújo[1], Adriana V. Ribeiro[1]

[1]Dep. de Ciência da Computação – Universidade Federal da Bahia (UFBA)
[2]Dep. Acadêmico de Computação – Instituto Federal da Bahia (IFBA)
[3]Florida International University (FIU)

Abstract

This chapter revisits the theme of Information-Centric Networking by exploring the architecture of Named Data Networking (NDN). NDN imposes changes in data forwarding and routing that makes the architecture more suitable for addressing distributed applications and mobile scenarios, in addition to providing a data-level security mechanism. Thus, the chapter addresses NDN properties and emergent researches on areas of mobility, security, and distributed applications, involving this architecture.

Resumo

Este capítulo revisita a temática das Redes Centradas na Informação ao explorar a arquitetura das Redes de Dados Nomeados (do inglês, Named Data Networking – NDN). A NDN impõe mudanças no encaminhamento e roteamento de dados que tornam a arquitetura mais apropriada para endereçar aplicações distribuídas e aplicações em cenários de mobilidade, além de proporcionar mecanismos de segurança em nível de dados. Assim, este capítulo aborda as propriedades da NDN e os desenvolvimentos recentes nas áreas de mobilidade, segurança e aplicações distribuídas envolvendo a arquitetura.

2.1. Introdução

Redes Centradas na Informação (do inglês, *Information-Centric Networking* – ICN) [Jacobson et al. 2009] é um modelo baseado em nomes de dados/conteúdo criado como uma

alternativa para atender os requisitos atuais e futuros da Internet. A principal característica das propostas baseadas em nome é a desassociação entre o localizador e o identificador de conteúdo, presente nas redes TCP/IP. Essa característica fundamental impôs mudanças na forma de identificação de *hosts*, encaminhamento, roteamento e segurança. Além disso, viabilizou o surgimento de arquiteturas que possibilitam o desenvolvimento de nativas para lidar com requisitos como segurança e mobilidade.

Diversos projetos (e.g., DONA, CCN/NDN e PSIRP/PURSUIT) foram desenvolvidos com o intuito de endereçar os desafios de criar uma rede baseada em nome. Esses projetos tinham o objetivo de desenvolver uma arquitetura viável para a implantação de ICN. Desta forma, buscavam definir as melhores abordagens de nomeação, roteamento, segurança, entre outros desafios advindos com o novo modelo. Dentre os projetos desenvolvidos, podemos destacar as Redes de Dados Nomeados (do inglês, *Named Data Networking* – NDN), que caracteriza-se pelo forte apoio da indústria, pela disponibilidade de um amplo conjunto de plataformas de códigos para desenvolvimento e experimentação e por uma comunidade científica internacional fortemente ativa.

A arquitetura NDN trata-se de uma proposta *clean-slate*, i.e., "tecnologia disruptiva", que oferece serviços de comunicação a partir de um modelo que é dirigido pelo receptor (*receiver-driven*), adota o encaminhamento baseado em nomes, implementa segurança ao nível dos dados (através de infraestruturas de chaves públicas), baseia-se na difusão seletiva de mensagens (*multicast*) não orientada à conexão, e faz uso de *caching* na camada de rede [Jacobson et al. 2009, Zhang et al. 2018a, Saxena et al. 2016]. Embora em NDN as características mais fundamentais das ICNs tenham sido mantidas, a arquitetura tornou-se madura e se expandiu fortemente nos últimos anos, acompanhando os novos requisitos de aplicações avançadas em diferentes cenários.

As propriedades da NDN que a tornam apropriada para sanar os requisitos atuais são decorrentes de alguns elementos de sua arquitetura: *Pending Interest Table* (PIT), *Forwarding Information Base* (FIB) e *Content Store* (CS). Esses elementos são responsáveis, respectivamente, por manter uma lista de requisições pendentes e possibilitar um plano de encaminhamento *stateful*, i.e., com guarda do estado/contexto, armazenar informações de encaminhamento e realizar *in-network caching*. Além dessas estruturas, cada nó da rede deve possuir um módulo de estratégia de encaminhamento, que será responsável por definir "se", "quando" e "para onde" encaminhar cada pacote de interesse [Jacobson et al. 2009, Zhang et al. 2014].

Os nós da rede NDN fazem uso da FIB, PIT e CS para realizar uma comunicação baseada no padrão arquitetural *publish/subscribe* [Eugster et al. 2003], por meio de um esquema de comunicação assíncrona. Este mecanismo de comunicação impulsiona o desenvolvimento de protocolos de roteamento baseado em nomes, cuja principal função é auxiliar o processo de encaminhamento e disseminar informações de alcançabilidade (prefixos de nomes) [Zhang et al. 2019a]. O roteamento baseado em nomes, em conjunto com as demais propriedades da NDN, permite a implementação de funções nomeadas de rede e de uma plataforma de invocação remota de métodos de forma distribuída e transparente quanto à localização [Król and Psaras 2017, Król et al. 2018]. No entanto, como a NDN utiliza um mecanismo de comunicação assíncrono em um ambiente de alta taxa de volatilidade, é importante manter o estado distribuído do sistema, apesar

de eventuais desconexões, ingressos e saídas de nós. Por tais motivos, protocolos de sincronização têm sido desenvolvidos de modo a viabilizar a comunicação assíncrona quando nem todas as partes podem estar *online* ao mesmo tempo [Li et al. 2018].

A mudança de paradigma proposta na arquitetura NDN, de um modelo de comunicação centrado nos *hosts* para um modelo centrado na informação, também impõe mudanças fundamentais na forma de pensar segurança de redes [Zhang et al. 2018d]. As estratégias e *frameworks* de segurança passam, portanto, a proteger os dados diretamente, e aplicações devem fazer uso da semântica associada ao esquema de nomeação para aplicar os controles de segurança, funções criptográficas e tolerância a falhas. Por conseguinte, novos desafios e problemas clássicos abrem espaço para pesquisas [Mannes and Maziero 2019, Zhang et al. 2018d], como: modelo de confiança, ataques de DDoS em IoT, modelos de ataque, autorização e propriedade na hierarquia de nomeação.

Em cenários de mobilidade, o plano de encaminhamento *stateful* da NDN oferece alternativas mais eficientes de descoberta de recursos [Araújo et al. 2019]. A desvinculação do conteúdo da sua localização é outro aspecto que permite os nós móveis recuperarem dados sem se preocupar onde os mesmos estão hospedados [Jacobson et al. 2009, Zhang et al. 2014]. Por tais motivos, a comunidade tem procurado, através da NDN, soluções leves, escaláveis, seguras e eficientes para redes móveis *ad-hoc* e redes veiculares [Brito et al. 2020, Araujo and Sampaio 2021, Zhang et al. 2019a, Mannes and Maziero 2019].

Este capítulo tem o objetivo de revisitar a temática das ICNs ao explorar a arquitetura NDN considerando três aspectos fundamentais das aplicações emergentes: **mobilidade, segurança e aplicações distribuídas**. Para atingir esse objetivo, o capítulo é iniciado com uma breve revisão dos fundamentos da arquitetura NDN (Seção 2.2). Em seguida, são discutidas as principais questões relacionadas aos três eixos temáticos escolhidos (mobilidade, segurança e aplicações distribuídas) – nas Seções 2.3, 2.4 e 2.5, respectivamente – onde são apresentados desafios de pesquisa, aplicações potenciais e casos de uso. Na Seção 2.6, são apresentados os ambientes de experimentação da arquitetura NDN e as instruções para as atividades práticas. Os principais desafios de pesquisa envolvendo a arquitetura e os eixos temáticos são elencados na Seção 2.7. Por fim, o capítulo é concluído na Seção 2.8.

2.2. Visão Geral do Paradigma ICN e da Arquitetura NDN

Nesta seção serão apresentadas algumas características das ICNs e seu comparativo com a arquitetura TCP/IP. Além disso, será discutida a arquitetura NDN, incluindo aspectos como estrutura dos nós, nomeação, roteamento e encaminhamento de dados. Também será discutido o impacto da integração da NDN com outras arquiteturas e modelos.

2.2.1. Revisão dos fundamentos de ICNs

A arquitetura atual da Internet é baseada no padrão TCP/IP, que tem uma associação entre os serviços/conteúdos disponíveis e sua localização. Desta forma, o usuário precisa saber onde o conteúdo/serviço está localizado. A ideia básica da ICN é desacoplar localizador e identificador do conteúdo e permitir que as solicitações de conteúdo sejam direcionadas à rede, como um serviço, sem que o usuário necessite saber sua localização [Jacobson et al. 2009]. A mudança de paradigma impacta no comportamento da rede em relação aos

pacotes transmitidos e serviços ofertados. O padrão TCP/IP trafega pacotes endereçados a uma localização, enquanto a ICN trafega pacotes endereçados a um conteúdo. Apesar dessa mudança, a ICN preocupa-se em manter requisitos como simplicidade, robustez e escalabilidade.

Na Figura 2.1, observa-se uma relação entre os serviços e características das arquiteturas TCP/IP e do modelo ICN. Na arquitetura atual (à esquerda), o protocolo IP é responsável por prover a comunicação entre os *hosts* e está no centro do modelo. No lado direito da figura, observa-se a substituição do protocolo IP por pedaços de conteúdos nomeados (*chunks*) que são utilizados na comunicação entre consumidor e produtor. Além disso, identifica-se que outros protocolos comuns nas redes TCP/IP (e.g., IP, UDP e TCP) podem ser usados para facilitar a comunicação em ICN. Pelas características expostas na figura, percebe-se que enquanto a arquitetura TCP/IP visa comunicar dispositivos, o modelo ICN tem o objetivo de prover conteúdos para usuários e aplicações.

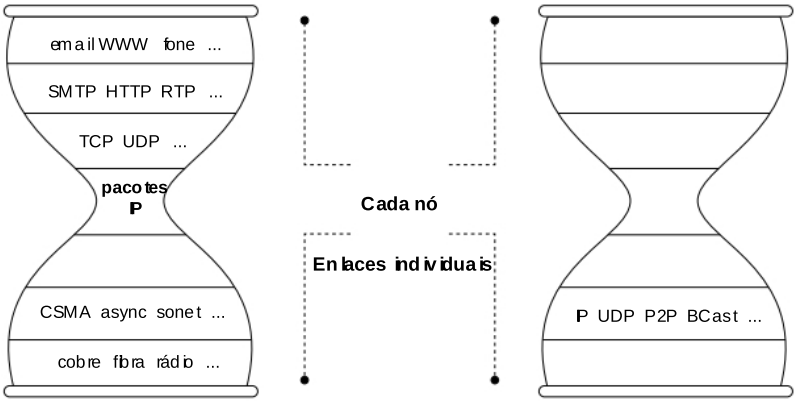


Figura 2.1. Comparação entre a arquitetura TCP/IP e as arquiteturas baseadas no modelo ICN. Traduzido de [Zhang et al. 2014].

Em busca de um modelo ICN para o futuro da Internet, pesquisadores propuseram projetos de arquiteturas centradas na informação. As arquiteturas variam em relação a alguns aspectos, como nomeação e roteamento. Os principais projetos desenvolvidos foram o DONA, PSIRP/PURSUIT e CCN/NDN. Dentre os projetos, destaca-se a importância do NDN, que foi criado há mais de 10 anos e, desde então, tem se consolidado como a principal implementação de ICN.

2.2.2. Arquitetura NDN

NDN é uma abordagem centrada no conteúdo criada para endereçar os requisitos atuais e futuros da Internet [Zhang et al. 2010]. Enquanto a arquitetura atual, centrada no *host*, preocupa-se com a comunicação entre dispositivos, a NDN foca na distribuição de conteúdo. Com esse objetivo, a arquitetura NDN usa uma comunicação inspirada no padrão arquitetural *publish/subscribe*, define novos tipos de pacote de transmissão de dados, estabelece novas estruturas para os nós da rede, faz roteamento baseado em nome e utiliza um plano de dados que armazena estado [Jacobson et al. 2009, Zhang et al. 2018a].

Em NDN, existem três papeis principais que um *host* pode assumir: consumidor de dados, produtor de dados e encaminhador. O consumidor de dados é um dispositivo que envia requisições de conteúdo para a rede. O produtor de dados é o responsável pela

criação e disponibilização dos conteúdos. Enquanto o nó encaminhador atua como um dispositivo intermediário entre esses dois *hosts*. Essas caracterizações de *host* não são excludentes. Portanto, um nó pode atuar como produtor, consumidor ou encaminhador em diferentes momentos [Zhang et al. 2018a].

A comunicação entre os *hosts* é estabelecida através do envio de pacotes de interesse e de dados (Figura 2.2). O pacote de interesse é análogo a uma requisição, já o pacote de dados inclui o conteúdo requisitado. Cada um desses pacotes é definido de acordo com diferentes campos que servem para proporcionar funcionalidades distintas, como prevenção de *loops* e medições de qualidade de serviço da rede [Saxena et al. 2016]. Além dos pacotes de interesse e de dados, a NDN também define o *Negative Acknowledgment* (NACK) para indicar que um nó não pode satisfazer a requisição ou encaminhar o pacote de interesse.

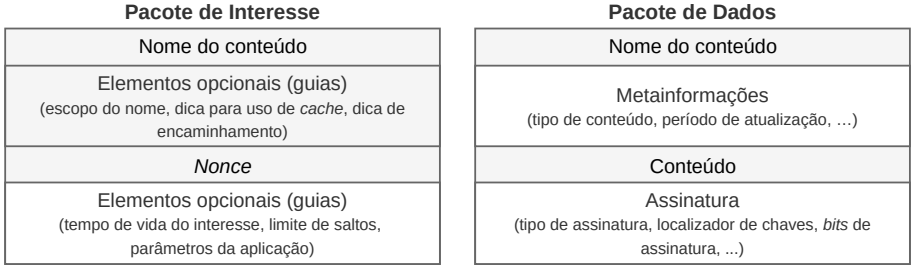


Figura 2.2. Pacotes da arquitetura NDN[1].

A NDN especifica três estruturas fundamentais para o processamento de pacotes: a tabela de interesses pendentes (do inglês, *Pending Interest Table* – PIT), a base de informações de encaminhamento (do inglês, *Forwarding Information Base* – FIB) e uma estrutura de armazenamento de conteúdo (do inglês, *Content Store* – CS). Uma visão geral do código da NDN é apresentada na Figura 2.3. Dentre os principais módulos, destaca-se o *NDN Forwarding Daemon* (NFD) [Afanasyev et al. 2018] que consiste numa estrutura que implementa o protocolo NDN e define como os pacotes de interesse e de dados são trafegados na rede e processados pelos equipamentos. NFD é composto por diversos módulos que, além de implementar as funções de encaminhamento de pacotes, definem estruturas e serviços da arquitetura. Dentre os principais módulos do NFD, é possível citar:

- ***ndn-cxx Library, Core* e *Tools:*** proporcicona funções comuns a diversos serviços do NFD. A biblioteca do ndn-cxx é essencial no desenvolvimento de aplicações NDN, como monitoramento das *faces*, DNS, roteamento e sincronização.

- ***Faces:*** trata-se de uma abstração que engloba tanto interfaces físicas quanto lógicas, incluindo túneis e chamadas do sistema entre camadas da arquitetura NDN.

[1]De acordo com a especificação: `https://named-data.net/doc/NDN-packet-spec/current/index.html`.

[2]Adaptado de: `https://named-data.net/wp-content/uploads/2019/11/5-codebase.pdf#page=5`.

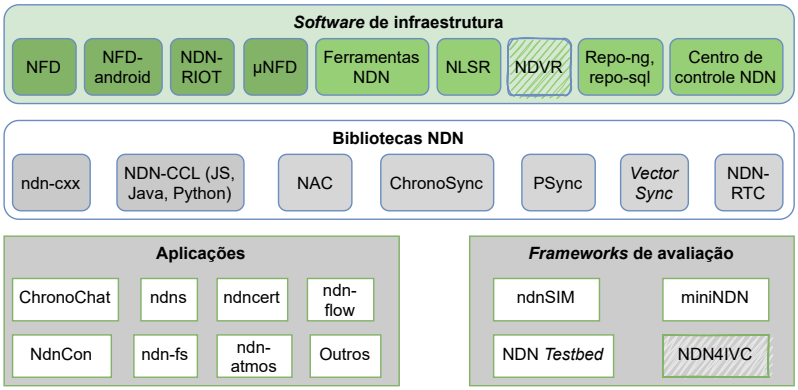


Figura 2.3. Visão geral do código base da NDN[2]. As caixas hachuradas são referentes a trabalhos apresentados em [Brito et al. 2020, Araujo et al. 2021].

- ***Tables:*** implementa as principais estruturas de processamento de pacotes em NDN: a CS, a PIT e a FIB. Além disso, engloba a definição de tabelas que implantam políticas de *cache*, medições e outras funcionalidades da rede.
- ***Forwarding:*** define o fluxo do processamento de pacotes considerando as tabelas implementadas no módulo *Tables*. Esse módulo permite o uso de diferentes estratégias de encaminhamento, como *best-route* e *multicast*.
- ***Management:*** implementa o protocolo de gerenciamento NDN e permite a verificação do desempenho da rede através de registros de informações de monitoramento dos pacotes, como latência.
- ***RIB Management:*** é responsável por gerenciar a base de informação de roteamento (do inglês, *Routing Information Base – RIB)*, que inclui rotas estáticas e dinâmicas.

Na definição da estrutura do NFD, destaca-se a composição interna das *faces* com as camadas de serviço de enlace e transporte. Na Figura 2.4, é possível observar que a camada de transporte é a camada de mais baixo nível. Essa camada é a responsável por prover entrega de pacotes para a camada de serviço. A camada de transporte utiliza a estratégia *best-effort* nessa comunicação, que pode ocorrer através de diferentes mecanismos de entrega, como transporte UNIX, Ethernet, UDP e TCP, a parir da utilização do formato de codificação TLV.

O formato de pacotes de tamanho variável (do inglês, *Time-Length-Value* – TLV) é utilizado para codificar os pacotes da camada de serviço de enlace. Essa camada proporciona um serviço de entrega *best-effort* para os pacotes das camadas de rede (interesse, dados e NACK). A camada de serviço de enlace é composta pelo serviço de enlace genérico e pelo serviço de enlace veicular. O serviço de enlace genérico é a opção padrão, enquanto o serviço de enlace veicular é uma estrutura planejada para possibilitar a implementação de uma estratégia adequada a cenários de redes veiculares.

Considerando o protocolo NDN e sua implementação por meio do NFD, temos a definição de todo o fluxo de processamento de pacotes. O processo é iniciado quando um nó encaminhador recebe um pacote através de uma *face*. As etapas do fluxo de processamento

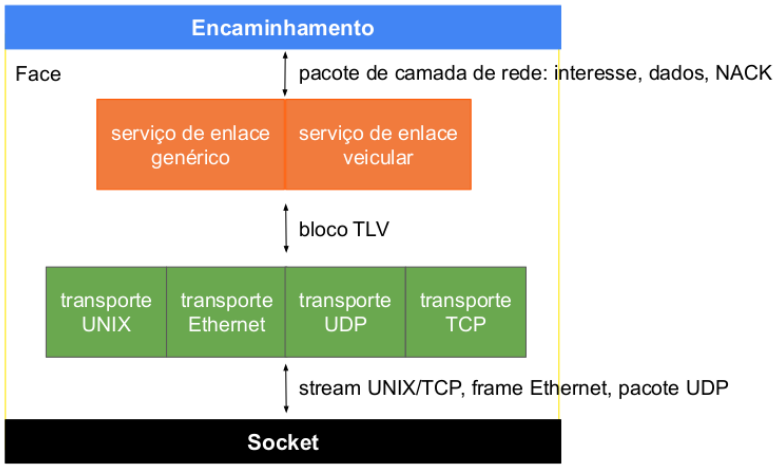


Figura 2.4. Representação da estrutura interna das *faces* NDN. Traduzido de [Afanasyev et al. 2018].

de pacotes varia de acordo com o tipo de pacote (interesse, dados ou NACK) e com a ação que está acontecendo (recebimento, envio, detecção de *loop*, etc.). Para os pacotes de interesse são definidos *pipelines* correspondentes a entrada e saída de pacotes, detecção de *loop* e contabilização de acertos e erros na CS. Para os pacotes de dados são tratadas três ações distintas: dados não solicitados, entrada e saída de pacotes de dados. Já para o NACK são definidos os processos de recebimento e envio.

Durante esses *pipelines*, as estruturas da NDN são utilizadas com finalidades variadas em momentos distintos. Em geral, a PIT possibilita a prevenção de *loop*, agregação de requisições para um mesmo interesse, encaminhamento *multicast*, manutenção de estado das requisições e cálculo de *Round Trip Time* (RTT) a partir das interfaces de saída e tempo de envio [Saxena et al. 2016]. Já a FIB é utilizada parar armazenar as melhores rotas de encaminhamento de pacotes de interesse. *Named-data Link State Routing Protocol* (NLSR) [Wang et al. 2018] é o protocolo padrão utilizado em NDN para popular a FIB. Trata-se de um protocolo que utiliza informações sobre estado de enlace ou roteamento hiperbólico para calcular as rotas e adicioná-las na tabela de roteamento. O NLSR possibilita a adição de várias interfaces de saída para um mesmo prefixo. Com isso, torna-se necessário o estabelecimento de um *ranking* para determinar as melhores rotas e adicioná-las na FIB. As múltiplas rotas também podem ser usadas simultaneamente, para enviar interesses, através do suporte nativo de *multihoming* da NDN [Zhang et al. 2018a].

Além do uso da PIT e FIB, um dos diferenciais da NDN é a adição de *in-network caching*. Em NDN, os nós encaminhadores têm uma CS que mantém uma cópia local dos conteúdos. Esse armazenamento possibilita maior resiliência e melhorias na qualidade de serviço da rede e no tempo de resposta ao usuário. A política padrão de *placement* de *cache* em NDN é a *Leave a Copy Everywhere* (LCE). Isso significa que todos os conteúdos que chegam no nó encaminhador serão armazenados. Como o tamanho da CS é limitado, é comum o uso de uma política de substituição de conteúdo que pode ser determinada considerando diversos contextos [Ioannou and Weber 2016, Pires et al. 2021] e cujo desempenho é uma preocupação sinalizada em diversos trabalhos [Ribeiro et al. 2017, Ribeiro et al. 2018, Pires et al. 2018, Pires et al. 2019].

A arquitetura NDN inclui repositórios de dados distribuídos, que têm o objetivo de armazenar cópias dos conteúdos. Diferente da CS, que se trata de um armazenamento oportunístico, os repositórios são utilizados para armazenar conteúdos específicos. Esse armazenamento pode ser realizado através da inserção estática de conteúdos ou através da configuração de prefixos de conteúdos que devem ser armazenados [Zhang 2019].

As três estruturas básicas da NDN são utilizadas no encaminhamento e roteamento de pacotes [Zhang et al. 2014]. O encaminhamento é feito de forma distinta para pacotes de interesse, de dados e NACK. Na Figura 2.5, são demonstrados os processos de chegada de pacotes de interesse e de dados em um nó encaminhador. Ao receber um pacote de interesse, o nó verifica se há um conteúdo correspondente armazenado localmente na CS. Se houver, o conteúdo é enviado ao nó solicitante e o pacote de interesse é descartado. Nesse cenário, uma cópia local do conteúdo é utilizada para resolver a requisição do usuário e a taxa de acerto do *cache* é incrementada.

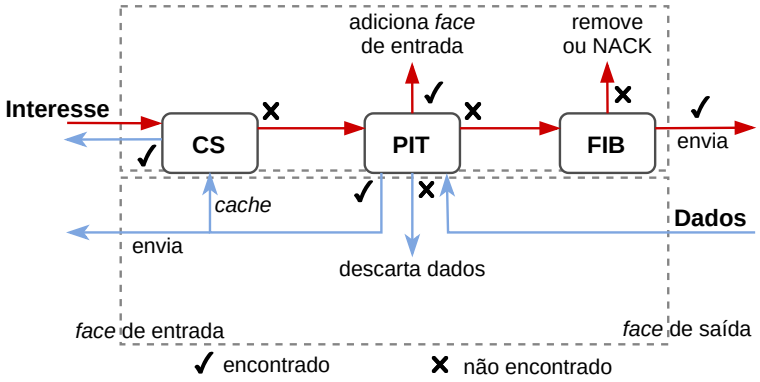


Figura 2.5. Processo de encaminhamento em um nó NDN. Adaptado de [Zhang et al. 2014].

Caso não haja uma cópia do conteúdo solicitado na CS, verifica-se a existência de uma entrada correspondente na PIT. Se existir uma entrada para o interesse, a interface de chegada do novo pacote é adicionada à entrada existente na PIT, caso ainda não tenha sido adicionada, e o novo interesse é descartado. Essa atividade demonstra o mecanismo de agregação de requisições da PIT. Se não existir nenhuma correspondência na PIT, uma nova entrada é criada para o interesse.

Quando uma nova entrada na PIT é adicionada, isso significa que o nó precisa encaminhar o pacote de interesse em busca de identificar um nó produtor que consiga satisfazê-lo. Para isso, a estrutura da FIB é consultada em busca de um próximo salto em direção ao prefixo do conteúdo que o usuário deseja. Se não houver nenhuma entrada na FIB, o pacote é removido ou o nó envia um NACK para a interface na qual o interesse chegou. Caso exista uma entrada na FIB, o pacote de interesse é encaminhado.

É importante ressaltar que, ao adicionar uma entrada na PIT, constrói-se uma trilha salto a salto entre o consumidor e o produtor. Quando o interesse alcança um produtor ou um nó encaminhador que possua o conteúdo em *cache*, o pacote de dados é encaminhado pelo caminho reverso. Ao receber o pacote de dados, o encaminhador verifica sua PIT em busca de interesses pendentes que devem ser satisfeitos. Como a PIT armazena a lista de interfaces de chegada, o pacote de dados pode ser enviado para todas as interfaces de forma

simultânea. Além disso, como a CS faz uso de uma estratégia oportunística, o conteúdo é armazenado na CS dos nós intermediários.

Além das funções básicas supracitadas, o *pipeline* de um nó encaminhador NDN inclui também operações especializadas que dão suporte à arquitetura, como o uso de diferentes estratégias de encaminhamento com base no nome e na FIB (e.g., melhor caminho, múltiplos caminhos, encaminhamento adaptativo, etc.), validação do escopo do nome com relação à *face* de entrada e saída (e.g., `/localhost` para comunicação inter-processo, `/localhop` para comunicação entre vizinhos apenas), política de dados não solicitados, supressão de retransmissão, marcação e uso de *tags* (e.g., *CachePolicyTag* para evitar a CS, *PrefixAnnouncementTag* para roteamento por auto aprendizagem, *IncomingFaceIdTag* para informar à aplicação a *face* de entrada, etc.), medições de RTT, dentre outras.

As estruturas de processamento de pacotes e as propriedades da NDN possibilitam vantagens em diferentes cenários e associação com arquiteturas variadas. Tais vantagens têm motivado propostas que envolvem a adoção da NDN em conjunto com novos modelos, paradigmas e arquiteturas, tais como SDN/P4 e *blockchain*, aplicados em cenários de mobilidade, como VANETs e FANETs.

2.2.3. Integração com outras arquiteturas e modelos

Zhang et al. 2019a exploram a escalabilidade das arquiteturas centradas na informação em relação às redes IP. Nesse trabalho, são considerados diferentes aspectos para avaliar os projetos desenvolvidos na área de ICN. Dentre os observados, é possível citar as abordagens de implementação *overlay*, *underlay* e híbrida. A maioria dos projetos desenvolvidos até 2013 propuseram o uso de ICN como um *overlay* da rede IP. A partir de 2014, tem-se predominância de projetos que propõem a arquitetura como um modelo *underlay* ou híbrido. A análise realizada pelos autores considerando a variedade da forma de implementação e dos outros aspectos elencados (e.g., requisitos da arquitetura atual), demonstram a versatilidade de implementação de arquiteturas ICN e a possibilidade de adaptação a diferentes cenários. Além disso, observa-se que não existe um consenso sobre a integração e interoperabilidade da NDN com outras arquiteturas legadas (e.g., TCP/IP).

Algumas iniciativas [Madureira et al. 2021, Siracusano et al. 2018, Conti et al. 2020, Nour et al. 2019a, Signorello et al. 2016, Miguel et al. 2018, Karrakchou et al. 2020a] têm explorado os benefícios da NDN a partir da integração com o paradigma SDN e a linguagem P4. Azgin et al. 2016 propõem a utilização da PIT apenas nas bordas da rede, removendo-a dos comutadores NDN do núcleo, de modo a tornar mais eficiente o processamento dos pacotes. Com esta mesma motivação, Madureira et al. 2021 utilizaram a linguagem P4 para desenvolver uma arquitetura e um protocolo denominados NDN-Fab. Através da programação na camada de enlace (L2), a arquitetura estabelece que os comutadores de borda da rede definem, na origem, as rotas de encaminhamento de pacotes NDN no núcleo. Assim, a NDN-Fab possibilita a interoperabilidade entre abordagens centradas no conteúdo e baseadas em localização, aproveitando a escalabilidade da NDN e o desempenho do encaminhamento L2. A NDN-Fab define o controlador de rede como o responsável pelo gerenciamento de PITs e FIBs globais, a partir da sua visão centralizada do núcleo. Já Moiseenko and Oran 2017 propõem o *ICN Path Switching*, que permite o uso de *path discovery* e *path steering*, além de realizar o encaminhamento de pacotes NDN

sem depender de pesquisas LNPM na FIB.

Alguns trabalhos [Signorello et al. 2016, Miguel et al. 2018, Karrakchou et al. 2020a] propõem a integração de todos os recursos da NDN dentro dos comutadores P4. Dada as limitações da linguagem P4, a implementação de todos os recursos previstos na arquitetura NDN não é viável. Por exemplo, a inexistência de estruturas de repetição, dificulta o processamento de pacotes no formato TLV (do inglês, *Type-Length-Value*) [Afanasyev et al. 2018]. Além disso, destaca-se o fato de que as principais estruturas de dados das redes NDN (PIT, CS e FIB) demandam espaços de armazenamento. A implementação dessas estruturas requer a utilização de módulos de memória mais lentos, levando a uma redução no desempenho de comutação de pacotes da rede.

Além das iniciativas de integração com a arquitetura IP e com SDN/P4, destaca-se a relação da NDN com livro-razão distribuído (do inglês, *distributed ledgers* – DLT), implementados sobretudo através da tecnologia *blockchain* [Fotiou and Polyzos 2016, Yu et al. 2017, Mori 2018, de Sousa et al. 2019]. Fotiou and Polyzos 2016 apresentaram um esquema para validação de integridade e fornecimento de conteúdos em NDN que permite que uma identidade (e.g., nome) possa ser usada como uma chave pública. O problema abordado reflete um cenário em que um detentor de um conteúdo deseja compartilhá-lo com alguns consumidores. Yu et al. 2017 realizaram uma discussão inicial do arcabouço NDN DeLorean, com foco na autenticação de arquivos de dados de longa duração em NDN. DeLorean fornece um serviço de auditoria de dados por meio da verificação de assinaturas em um livro-razão público. Por fim, Mori 2018 propôs um esquema para assegurar a integridade e autenticidade da fonte de dados de sensoriamento na *cache* em redes de sensores sem fio baseadas em NDN.

Zhang et al. 2019b propõem um sistema de livro-razão distribuído utilizando um grafo acíclico direcionado e *Proof-of-Authentication*. Esse sistema é construído em cima de uma arquitetura NDN, com o objetivo de facilitar a disseminação de informação entre os *hosts*. Jin et al. 2017 usam a arquitetura NDN em substituição à rede TCP/IP com a justificativa de que a NDN provê melhor suporte para implementação de *multicast* e hierarquias de *status* nas *blockchains*. Por fim, Sedky and Mougy 2018 utilizam NDN para implementar uma *blockchain*, com o intuito de otimizar a troca de informação entre as partes e aumentar a eficiência das transações.

Implementações de NDN têm sido amplamente utilizadas para cenários de redes de maior complexidade, sobretudo envolvendo mobilidade e dispositivos com capacidade limitada de recursos computacionais. Portanto, destaca-se uma tendência recente para a implementação das redes veiculares de dados nomeados (do inglês, *Vehicular Named Data Networks* – VNDN) [Grassi et al. 2014, de Sousa et al. 2018a, de Sousa et al. 2018b, Wang and Li 2020, Rondon et al. 2020, Fang et al. 2018, Wang et al. 2020, Khelifi et al. 2020, Tizvar and Abbaspour 2020, Wang et al. 2021] que são definidas como redes veiculares *ad-hoc* em que os nós (veículos) atuam no armazenamento temporário de conteúdos e encaminhamento de interesses para nós adjacentes, com o intuito de atender a demanda dos consumidores. Com motivações semelhantes, outra linha de trabalhos explora as vantagens da NDN na comunicação em redes de veículos aéreos não tripulados (do inglês, *Flying Named Data Networking* – FNDN) [Araújo et al. 2019, Barka et al. 2018, Serhane et al. 2021].

2.3. Suporte à Mobilidade em Redes de Dados Nomeados

Nesta seção, serão apresentados os aspectos referentes à mobilidade dos nós em ambientes NDN. Inicialmente, serão discutidos os requisitos das redes móveis emergentes e as vantagens oferecidas pela NDN a essas redes; em seguida, serão elencadas algumas questões sobre a mobilidade do consumidor e do produtor de dados, apresentando a classificação de soluções de suporte à mobilidade desses nós em quatro categorias principais.

2.3.1. Requisitos de redes móveis do futuro para NDN

Estima-se que o uso de conexões móveis alcançará mais de 70% da população global em 2023 [Cisco 2020]. Diante do histórico e previsões que demonstram um crescimento cada vez maior de conexões e requisitos distintos das aplicações emergentes, a NDN surge como uma arquitetura promissora com potencial para atender à demanda prevista. Há muitos trabalhos que apontam a ICN como um facilitador de comunicação, possibilitando melhorias de desempenho em redes 5G [Serhane et al. 2021], implantação e desenvolvimento de soluções em IoT [Nour et al. 2019b, Madureira et al. 2020] e computação de borda [Psaras et al. 2018]. Dentre as principais características da NDN que favorecem o suporte à mobilidade dos nós, podemos citar: necessidade de dados seguros e nomeados na camada de rede, armazenamento em *cache*, uso de repositórios, agregação de requisições e entrega *multicast*.

Dados seguros e nomeados na camada de rede

Um grande diferencial da NDN em relação à arquitetura TCP/IP é a substituição de endereços IP na camada de rede por dados nomeados, além da segurança aplicada aos dados em vez do canal de comunicação [Zhang et al. 2014]. O uso de dados nomeados na camada de rede torna possível a separação do identificador e localizador dos dados. Uma vez que os dados são seguros e tornam-se independentes de localização, qualquer nó com capacidade de armazenamento pode manter uma cópia local desses dados ao recebê-los.

O armazenamento na rede permite que os dados dos usuários permaneçam disponíveis, mesmo quando os dispositivos do usuário estão *offline*. Como é mais viável considerar que os dados estejam sempre disponíveis em algum lugar na rede do que os dispositivos estejam sempre *online*, é esperado que a NDN impulsione novas gerações de aplicações verdadeiramente P2P por meio do armazenamento na rede e da comunicação assíncrona [Zhang 2019, Psaras et al. 2018] com suporte nativo de *multihoming* na camada de rede [Zhang et al. 2018a, Amadeo et al. 2016].

Armazenamento em *cache*

A capacidade de armazenamento em *cache* nos nós da NDN é uma das principais vantagens dessa arquitetura em relação à arquitetura TCP/IP. O armazenamento em *cache* é um componente pertencente ao encaminhamento da NDN e cada nó encaminhador desempenha um *cache* oportunístico, no qual armazena passivamente os dados que o nó ajuda a encaminhar [Zhang 2019]. Os nós que armazenam os dados podem atuar como nós de réplica usando a função de armazenamento. Desta forma, os dados armazenados

em *cache* podem ser usados para responder solicitações futuras, independentemente da acessibilidade do produtor original, o que contribui com a mobilidade do produtor [Araújo et al. 2019, Araújo 2018, Araújo et al. 2018].

Este armazenamento em *cache* melhora a recuperação de dados e reduz a latência [Nour et al. 2019b], além de permitir que os nós móveis ajudem no processo de distribuição de dados, simplesmente ao mudarem de rede, desempenhando o papel de mula de dados [Araujo and Sampaio 2021]. Contudo, apesar do armazenamento em *cache* proporcionar vantagens significativas à mobilidade dos nós, apenas a adoção de *cache* pode ser insuficiente e outros tipos de armazenamento são requeridos para que possam ser usados por todas as aplicações que almejam disponibilizar seus dados [Zhang 2019].

Repositórios

Diferente da CS (*cache)*, os repositórios (*repos*) são elementos da arquitetura NDN que visam o armazenamento persistente de dados [Zhang 2019]. Um *repo* pode ser um dispositivo independente ou um módulo conectado a um dispositivo com outra finalidade, como produtores de dados ou nós roteadores [Zhang 2019]. Os *repos* são armazenamentos gerenciados que são instruídos sobre quais conteúdos armazenar e buscam proativamente por esses conteúdos na rede. O gerenciamento de um *repo* determina quais conteúdos devem ser carregados para o seu armazenamento, conforme os prefixos de nome dos conteúdos informados na configuração do *repo* [Zhang 2019].

De acordo com Psaras et al. 2018, com a expansão da IoT e da era da computação de borda, é necessário um modelo de comunicação centrado em dados que atenda o real objetivo das aplicações neste contexto, que é o acesso aos dados. Os autores defendem a implantação de *repos* na borda da rede (e.g., em pontos de acesso Wi-Fi ou similares) que possam ser usados para armazenar temporariamente os dados gerados pelos usuários e dispositivos da IoT, antes da sincronização com a nuvem – a sincronização deve ocorrer apenas quando necessário seguindo a melhor estratégia de acordo com os dados e requisitos da aplicação. A adoção de *repos* na borda tem potencial para reduzir a largura de banda necessária para enviar os dados e os custos financeiros com a sincronização com a nuvem. Além disso, pode prover suporte à mobilidade, uma vez que o *repo*, ao armazenar os dados dos produtores móveis, passa a responder as solicitações para esses dados e assim torna transparente a mobilidade do produtor [Psaras et al. 2018].

Agregação de requisições

As solicitações de pacotes de interesse para o mesmo pacote de dados, oriundas de diferentes consumidores ou não, podem ser agregadas na PIT dos nós. Essa ação evita a necessidade de todas as requisições terem que alcançar o nó produtor ou detentor dos dados solicitados. A agregação de requisições permite que apenas a primeira requisição de cada interesse seja encaminhada rumo à fonte de dados causando uma redução significativa do tráfego da rede [Ioannou and Weber 2016, Shannigrahi et al. 2017]. Ao se mover de uma rede para outra, um nó móvel pode ter suas requisições agregadas e atendidas a partir do primeiro roteador comum à sua rede antiga e atual, não havendo uma comunicação fim

a fim entre o nó móvel e o produtor dos dados.

Ao analisar uma base de dados científicos, – mais especificamente, dados climáticos usados por cientistas de todo o mundo – Shannigrahi et al. 2017 comprovaram que os padrões de solicitações são realmente agregáveis e podem reduzir a carga no servidor. Os autores investigaram a NDN sob a perspectiva de um sistema de distribuição de dados científicos e comprovaram que a agregação de interesses pode ser útil em cenários de alto tráfego, principalmente quando combinada às técnicas de *cache* e estratégias de encaminhamento. A NDN pode melhorar a entrega de dados para os usuários finais e, ao mesmo tempo, reduzir a carga nos servidores e na rede [Shannigrahi et al. 2017].

Entrega *multicast*

A entrega *multicast* de dados acontece – como uma consequência secundária da agregação de interesses de diferentes origens na PIT dos nós – quando um nó com interesses agregados recebe o pacote de dados solicitado, i.e., o dado é replicado no nó e encaminhado a cada interface de entrada dos interesses pendentes agregados na PIT. Dessa forma, um único pacote de dados consegue responder a todos os interesses pendentes agregados e correspondentes ao dado [Amadeo et al. 2016].

A NDN suporta o *multicast/anycast* naturalmente a partir da camada de rede, e quaisquer solicitações insatisfeitas durante um evento de mobilidade podem ser reemitidas sem a necessidade de soluções complexas de *handoff* como as empregadas no IP [Nour et al. 2019b]. O *multicast* nativo também atende ao objetivo de reduzir a quantidade de tráfego e as interações com os nós com restrição de energia, beneficiando principalmente os dispositivos da IoT [Amadeo et al. 2016].

2.3.2. Mobilidade de consumidor

Em NDN a mobilidade do consumidor é naturalmente suportada através da própria arquitetura [Zhang et al. 2016, Zhang et al. 2018b, Araújo et al. 2019]. Os consumidores simplesmente podem requisitar novamente os interesses para os dados não recuperados e a rede se responsabiliza pela entrega desses dados aos consumidores, independentemente de sua localização. Isso é possível graças ao esquema de trilha – i.e., "migalhas de pão" (do inglês, "*bread crumbs*") – deixado pelos interesses na PIT dos nós salto a salto ao longo do caminho do percurso do interesse em direção à fonte dos dados. As trilhas na PIT possibilitam o encaminhamento dos dados de volta ao consumidor, efetivando a entrega dos dados requisitados [Jacobson et al. 2009].

Tanto os consumidores móveis quanto os consumidores estáticos se beneficiam da funcionalidade do plano de encaminhamento com estado baseado em trilhas na PIT. Contudo, no caso de consumidores móveis, pode haver uma peculiaridade nas situações em que o consumidor troca de rede (i.e., processo conhecido como *handoff* ou *handover*) durante uma solicitação. Por exemplo, se um consumidor móvel emitir um interesse a partir de uma rede e, antes que o dado seja recebido, ocorrer o *handoff* desse consumidor para uma outra rede, a aplicação no consumidor tende a requisitar novamente os dados não recuperados que, por sua vez, serão entregues na rede atual e na rede antiga do consumidor. Isso acontece justamente pelo fato dos dados seguirem as trilhas encontradas na PIT dos

nós e especialmente devido às seguintes razões:

- **Entradas PIT ativas:** cada entrada PIT representa uma trilha para encaminhar os dados recuperados de volta ao solicitante. As entradas PIT são removidas em duas situações: (i) quando um pacote de dados correspondente chega ao nó ou (ii) quando a entrada PIT atinge seu tempo de vida (do inglês, *lifetime*) sem que um dado correspondente tenha sido recebido.
- **Privacidade dos consumidores:** na NDN, as requisições não carregam informações que identifiquem o consumidor [Zhang et al. 2014] – diferente das redes IP em que os pacotes transportam os endereços IP de origem e destino – portanto, as duas requisições do consumidor móvel (i.e., antes e depois do *handoff*) são tratadas pela rede como sendo totalmente independentes.
- **Fonte de dados alcançável:** é necessário que pelo menos uma fonte de dados (e.g., nó produtor ou nó detentor) esteja alcançável a partir das redes em que as requisições foram feitas para que os dados sejam recuperados. Os protocolos de roteamento são responsáveis por manter as informações de alcançabilidade dos nós atualizadas na tabela FIB de cada nó da rede.

Algumas das características da NDN discutidas anteriormente podem facilitar a mobilidade dos nós. O plano de encaminhamento com estado da NDN possibilita que os consumidores apenas requisitem novamente os dados não recuperados, mesmo após uma mudança de rede. Porém, essa simplicidade no suporte à mobilidade do consumidor não se aplica no suporte à mobilidade do produtor [Araújo and Sampaio 2017].

2.3.3. Mobilidade de produtor

O modelo de comunicação da NDN é focado na recuperação de dados em vez da entrega de pacotes. O foco na recuperação de dados possibilita que o suporte à mobilidade do produtor se concentre no encontro dos interesses com os dados gerados pelos produtores móveis [Zhang et al. 2016]. Os dados nomeados na camada de rede e a segurança aplicada diretamente aos dados facilitam a replicação e armazenamento em *cache* no caminho (*on-path caching*), *cache* fora do caminho (*off-path caching*) [Ioannou and Weber 2016] e nos repositórios [Zhang 2019], o que aumenta as chances do interesse encontrar os dados, mesmo que o produtor original esteja indisponível [Araújo et al. 2019, Araújo 2018]. Para possibilitar que os interesses encontrem os dados do produtor móvel, Zhang et al. 2016 apontam duas direções possíveis:

1. **Perseguição do produtor:** usa um elemento auxiliar (i.e., um "ponto de encontro") para descobrir o paradeiro do produtor na rede. Os interesses são direcionados a um produtor móvel para recuperar dados (os interesses não precisam alcançar o produtor, caso encontrem os dados solicitados na *cache* de algum roteador ao longo do caminho). Essa direção é semelhante ao suporte à mobilidade IP e as abordagens baseadas em mapeamento e rastreamento podem ser adaptadas para serem empregadas na NDN. Com exceção das abordagens baseadas em roteamento, visto que é improvável que os produtores móveis anunciem o prefixo dos dados no sistema de roteamento na escala da Internet [Zhang et al. 2016].

2. **Encontro de dados:** visa garantir que os dados produzidos por produtores móveis sejam facilmente encontrados. Essa direção pode potencializar a centralização em dados da NDN, uma vez que os dados podem ser desassociados do produtor (e.g., os dados podem ser movidos para um local estacionário e de fácil acesso) e aplicações que geram dados em um local, podem tornar o nome dos dados independente do produtor móvel e atrelado a uma região estacionária [Zhang et al. 2016].

As principais abordagens de soluções de suporte à mobilidade do produtor estão representadas na Tabela 2.1 e são discutidas individualmente nas subseções seguintes. Por questões didáticas e de padronização, durante o restante desta seção serão usados os termos mapeamento, rastreamento, depósito de dados e local de dados para classificar as abordagens de mobilidade do produtor, conforme terminologia adotada em Zhang et al. 2016 e Araújo 2018.

Tabela 2.1. Abordagens de mobilidade do produtor [Zhang et al. 2016, Araújo 2018].

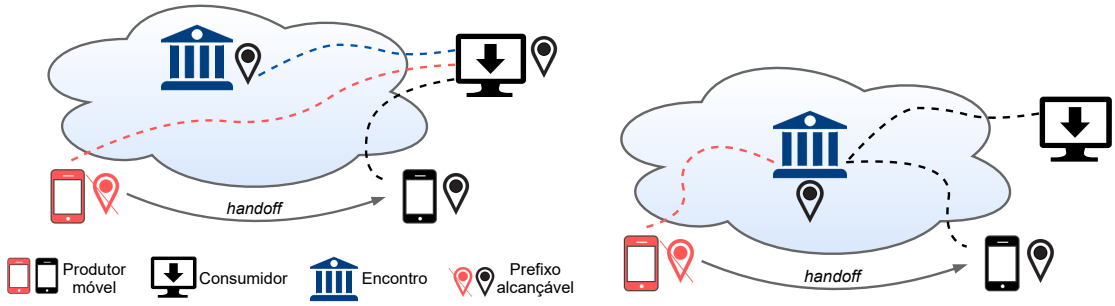
Perseguição do produtor móvel	
Mapeamento	O produtor informa ao ponto de encontro qual o ponto de fixação onde seus dados podem ser recuperados.
Rastreamento	O produtor cria uma trilha de migração para alcançá-lo, que deve ser seguida por interesses do ponto de encontro.
Encontro de dados	
Depósito de dados	Os dados produzidos pelo produtor móvel são movidos para um servidor estacionário conhecido.
Local de dados	Os dados são produzidos e disponibilizados em uma região estacionária.

Perseguição do produtor: Mapeamento

As soluções baseadas em mapeamento herdam as ideias do suporte à mobilidade IP, usando pontos de encontro estáveis para alcançar o produtor móvel. Nas abordagens baseadas em mapeamento, o produtor precisará enviar informações ao ponto de encontro sempre que fizer *handoff*. Assim, cada produtor móvel deve informar o nome do seu ponto de fixação atual "temporário" ao ponto de encontro "estável" [Zhang et al. 2016]. Os trabalhos dessa abordagem se dividem em duas dimensões com base na função do ponto de encontro e na forma como os nomes de pontos de fixação mapeados são transportados nos pacotes de interesse.

Função do ponto de encontro: pode ser um serviço que mapeia os nomes dos dados produzidos por um produtor móvel para o nome do ponto de fixação atual do produtor; um agente doméstico que faz o tunelamento de interesses em direção aos produtores móveis; ou um sistema híbrido que engloba o mapeamento de nomes e o mecanismo de tunelamento de interesses [Zhang et al. 2016]. A Figura 2.6(a) mostra quando o ponto de encontro desempenha o serviço de mapeamento de nomes, primeiro os consumidores consultam o ponto de encontro para obter o mapeamento entre o nome

dos dados e o nome do ponto de fixação atual do produtor e, em seguida, os próprios consumidores encapsulam os interesses em direção ao ponto de fixação obtido [Zhang et al. 2016]. Já na Figura 2.6(b), o ponto de encontro atua como um agente doméstico do produtor móvel, onde os dados são publicados sob o nome do ponto de encontro estacionário e globalmente alcançável. Os interesses dos consumidores alcançam o ponto de encontro que faz o tunelamento dos interesses recebidos em direção ao produtor móvel, no seu ponto de fixação, com base nas informações de mapeamento [Zhang et al. 2016]. Por fim, no modo híbrido o ponto de encontro atua como um agente doméstico para encaminhar, via túnel, o primeiro interesse ao produtor móvel, e o pacote de dados retornado traz o nome do ponto de fixação atual do produtor; que pode ser usado pelo consumidor para enviar interesses, via túnel, diretamente ao produtor [Zhang et al. 2016].



(a) Serviço de mapeamento. (b) Agente doméstico.

Figura 2.6. Abordagem de mobilidade baseada em mapeamento do produtor, a função do ponto de encontro. Adaptado de [Zhang et al. 2016, Araújo 2018].

Buscar dados através de nomes de pontos de fixação mapeados: para orientar o interesse que carrega um nome inalcançável para o ponto de fixação atual do produtor, o nome do ponto de fixação precisa ser anexado ao interesse. Para esse fim, há duas opções: (i) colocar o nome do ponto de fixação como prefixo ao nome dos dados transportado no interesse ou (ii) acrescentar um novo campo no pacote de interesse para carregar o nome do ponto de fixação, como uma dica de encaminhamento [Zhang et al. 2016].

Perseguição do produtor: Rastreamento

As soluções baseadas em rastreamento estendem o plano de encaminhamento com estado para criar a trilha de migração e recuperar interesses dos consumidores. Semelhante à abordagem de agente doméstico, o rastreamento requer que os dados sejam publicados sob o prefixo globalmente alcançável do ponto de encontro. Sempre que o produtor se move, envia interesse de comando de rastreamento ao ponto de encontro para manter o caminho reverso entre sua localização atual e o ponto de encontro.

Os interesses de comando de rastreamento recuperam interesses dos consumidores, em vez de dados. Os interesses regulares dos consumidores serão encaminhados rumo ao ponto de encontro, mas podem seguir em direção ao produtor caso encontrem o rastreamento no caminho, como mostra a Figura 2.7. Dessa forma, o ponto de encontro não precisa necessariamente participar da comunicação entre consumidor/produtor; e se

encarrega de fazer anúncios de roteamento para o prefixo do nome de dados visando atrair os interesses dos consumidores para os dados do produtor móvel e o interesse de comando de rastreamento para si [Zhang et al. 2016].

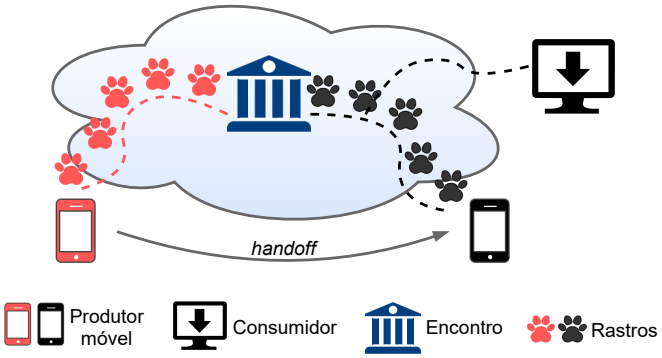


Figura 2.7. Abordagem de mobilidade baseada em rastreamento do produtor. Adaptado de [Zhang et al. 2016, Araújo 2018].

Quando o produtor se move e a nova trilha de migração (i.e., rastro) cruza com um rastro anterior, os interesses dos consumidores que estão pendentes no rastro antigo podem ser encaminhados para o novo rastro. Outra abordagem se baseia no fato do produtor enviar interesse de comando de rastreamento para, além do ponto de encontro, o seu ponto de fixação anterior para buscar os interesses pendentes e minimizar a interrupção da busca de dados. O processo de rastreamento pode ser feito de diferentes formas, incluindo o rastreamento na FIB e na PIT [Zhang et al. 2016].

Rastreamento na FIB: introduz uma FIB dedicada (tFIB) separada da FIB padrão. A tFIB é atualizada pelos interesses de comando de rastreamento. Interesses regulares serão encaminhados seguindo o rastreamento, se existir correspondência na tFIB [Zhang et al. 2016].

Rastreamento na PIT: estende a funcionalidade da PIT, de recuperar dados, para recuperar interesses. Adiciona um novo campo, `traceName`, no pacote de interesse, reservado ao nome de interesses de comando de rastreamento visando criar um rastro entre ponto de encontro e produtor móvel; e um sinalizador, `traceable`, para indicar se um interesse pode ser rastreado por outros interesses. Após receber um interesse com `traceName`, um roteador primeiro procura uma correspondência do `traceName` com as entradas da PIT. Se houver uma correspondência, o interesse será encaminhado através da interface de entrada do interesse do comando de rastreamento, caso contrário será encaminhado usando o mecanismo padrão [Zhang et al. 2016].

Encontro de dados: Depósito de dados

Como a NDN permite que os dados sejam facilmente separados dos produtores originais, em vez de perseguir o produtor móvel, uma alternativa é mover os dados gerados por esses produtores para uma localização acessível (e.g., estacionária) [Zhang et al. 2016], como representado na Figura 2.8, onde os interesses do consumidor são encaminhados para o depósito para buscar dados e podem encontrar o rastro do produtor móvel no caminho.

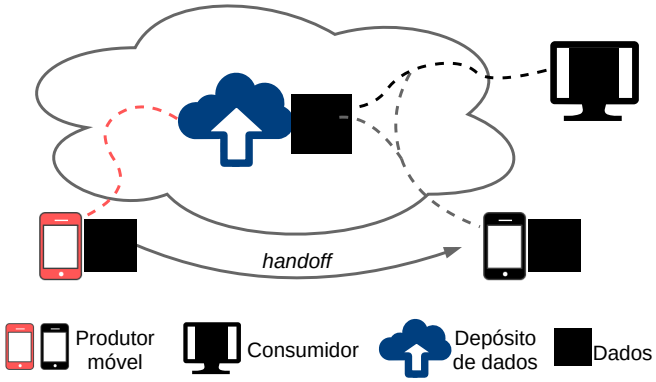


Figura 2.8. Abordagem de mobilidade baseada em depósito de dados. Adaptado de [Zhang et al. 2016, Araújo 2018].

Um depósito de dados se assemelha a um agente doméstico das soluções baseadas em mapeamento, exceto pelo fato que o depósito de dados assume a responsabilidade por hospedar os dados em vez de simplesmente encaminhar interesses [Zhang et al. 2016]. Por exemplo, o depósito pode ser configurado para armazenar sob demanda os dados dos usuários e, ao receber uma solicitação para recuperar um determinado dado, retornará o dado se este já tiver sido carregado no depósito, caso contrário, o depósito tentará recuperar o dado solicitado usando técnicas de mapeamento ou rastreamento.

Um depósito de dados representa um encontro baseado em nomes, onde os interesses e dados são atraídos para se encontrarem. A Internet atual desempenha a função de depósito na camada de aplicação. O serviço de armazenamento na nuvem é um exemplo de encontro baseado em nomes e depósito de dados. Diferente da rede atual, a NDN suporta o encontro baseado em nomes na camada de rede, permitindo que o depósito anuncie o prefixo dos dados na tabela de roteamento [Zhang et al. 2016].

Encontro de dados: Local de dados

Em algumas aplicações e ambientes de rede, os dados estão associados a uma região geográfica específica e podem ser gerados por qualquer produtor no local, como mostra a Figura 2.9. Por exemplo, em aplicações de VNDN [de Sousa et al. 2018a], os dados sobre as condições da rodovia em um determinado local podem ser gerados por qualquer veículo presente naquela região [Zhang et al. 2016].

O encaminhamento de interesses para recuperar os dados pode usar geo-roteamento ou contar com as RSUs (*Road Side Units*) para anunciar os prefixos de dados da região no sistema de roteamento. Quando um produtor na região recebe um interesse, pode usar informações de GPS (*Global Positioning System*) para gerar o dado e responder a solicitação recebida. O produtor também pode responder os interesses recebidos se houver dados correspondentes em *cache*. Uma vez que o produtor sai da região do local dos dados, deixa de receber requisições para os dados da região, que podem ser respondidas por outros produtores móveis daquele local [Zhang et al. 2016].

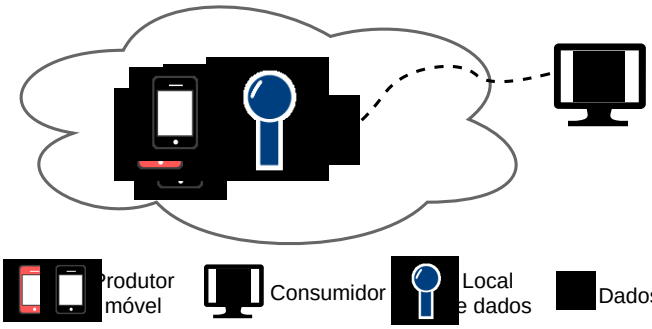


Figura 2.9. Abordagem de mobilidade baseada em local de dados. Adaptado de [Zhang et al. 2016, Araújo 2018].

2.4. Segurança em NDN

Nesta seção serão apresentados os principais conceitos em relação aos modelos de confiança, aplicações de segurança, superfícies de ataque e soluções de proteção disponíveis em NDN. A adoção de uma nova arquitetura de redes implica em uma nova forma de desenvolver aplicações, estratégias de armazenamento e manuseio das informações. Portanto, tais aspectos também serão endereçados em uma discussão geral ao final da seção.

2.4.1. Segurança nativa na arquitetura NDN

O modelo de comunicação centrado na informação leva a uma mudança fundamental no projeto e adoção de estratégias de segurança: ao invés de proteger o canal de comunicação, a arquitetura NDN provê maneiras de proteger os dados diretamente, independente do meio de transmissão e armazenamento. No alicerce do arcabouço de segurança da arquitetura NDN está a criptografia de chave pública e os certificados digitais [Zhang et al. 2018d].

Criptografia de chave pública, ou criptografia assimétrica, é um mecanismo criptográfico no qual utiliza-se um par de chaves – chave pública e chave privada – no processo de ciframento e deciframento de informações [Stallings 2020]. Os certificados digitais, por sua vez, servem para atestar a autenticidade da chave pública de uma entidade [Stallings 2020], condição essencial para uso de sistemas criptográficos assimétricos. A arquitetura NDN provê mecanismos flexíveis que automatizam e garantem a corretude do processo de gerenciamento e operação das chaves e certificados [Yu et al. 2015, Zhang et al. 2018d].

Para entender o processo de gerenciamento e operação de chaves criptográficas na NDN, considere que toda aplicação e todo componente participando de uma comunicação NDN é uma "entidade" e toda entidade tem propriedade sobre um ou mais espaços de nomes. Uma entidade garante a propriedade sobre um espaço de nomes através de um certificado digital, e pode delegar um ou mais sub-prefixos do seu espaço de nomes emitindo um certificado para outra entidade. Um espaço de nome que possui um certificado associado é denominado "identidade". Assim, os principais componentes do arcabouço de segurança da NDN são:

- **Chaves criptográficas:** podem ser vistas como um conjunto de *bytes* qualquer identificado por um nome, porém com uma semântica especial: tais *bytes* são usados criptograficamente para assinar ou cifrar outros *bytes*. Ao tratar as chaves

criptográficas como um dado nomeado qualquer, a arquitetura NDN faz uso de todos os seus componentes na troca de chaves entre entidades.

- **Certificado digital:** é usado para correlacionar uma chave pública a um prefixo de nome e, assim, permitir a validação de sua autenticidade e propriedade. O nome do certificado segue uma convenção de nomeação com "`/<prefix>/KEY/-<key-id>/<issuer-info>/<version>`", onde `/prefix` é o prefixo de nome ao qual a chave pública será atrelada, `key-id` é um identificador da chave, `issuer-info` são informações sobre o emissor e `version` é a versão do certificado. É importante ressaltar que o nome atrelado à chave não necessariamente tem relação com o emissor, o que flexibiliza o esquema de nomeação mas requer o complemento de políticas de validação para verificação da propriedade.
- **Política de confiança:** As aplicações NDN definem uma política de confiança, também conhecida como política ou regras de validação, que especifica quais entidades são confiáveis para produzir quais pedaços de dados, quais chaves devem ser usadas em quais espaços de nomeação e para quais propósitos. Por exemplo, uma aplicação NDN de roteamento que produz dados sobre alcançabilidade de prefixos define uma política de confiança que especifica que os dados produzidos em um roteador devem ser assinados pela chave específica do roteador e tal chave não pode ser usada para assinar outras chaves; ao invés disso, as chaves dos roteadores são assinadas pela chave específica da organização que gerencia os roteadores em questão.

Assim, valendo-se de um esquema de nomeação bem estruturado e semanticamente expressivo, a arquitetura NDN permite que aplicações sejam desenvolvidas seguindo convenções de nomeação para chaves/certificados. Essas convenções viabilizam um processo sistemático de assinatura, validação, cifragem e decifragem de dados [Zhang et al. 2018d]. Além disso, o esquema de nomeação permite automação e melhora a usabilidade do processo de gerenciamento de chaves: é possível identificar automaticamente qual chave foi usada para assinar um dado, utilizar pacotes de interesse/dados para obter o certificado equivalente àquela chave, verificar a confiança/autoridade da chave em relação ao dado produzido e da própria chave em relação à cadeia de confiança. Com isso, os dados são protegidos diretamente, independente do meio de transmissão/armazenamento.

Na Figura 2.10 é ilustrado um modelo de confiança para uma aplicação de *blog* em NDN provido por Yu et al. 2015. Neste exemplo, a aplicação requer que os artigos publicados sejam assinados por um autor. Os autores, por sua vez, possuem chaves assinadas pelos administradores do *site*. Tais administradores possuem chaves que são assinadas pelo dono da aplicação ou por outros administradores. A chave do dono da aplicação é considerada uma âncora de confiança e pressupõe-se que ela está pré-instalada – ou pode ser obtida por um canal seguro – nos consumidores.

A arquitetura NDN também provê mecanismos nativos para garantir a disponibilidade dos dados: o dado pode ser obtido de forma segura e independente da localização, seja através do produtor, ou através de repositórios e *caches* oportunísticos [Zhang et al. 2018d]. A disponibilidade dos certificados segue o mesmo princípio da disponibilidade dos dados, uma vez que um certificado basicamente é um conjunto de *bytes* nomeados e com uma semântica especial. Adicionalmente, a arquitetura NDN possui APIs que permitem

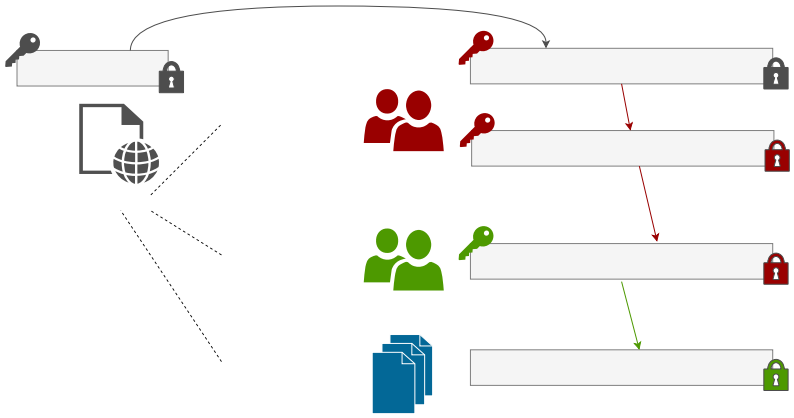


Figura 2.10. Modelo de confiança para aplicação de *blog*. Adaptado de [Yu et al. 2015].

a um produtor acumular um conjunto de certificados e distribuí-los em um único pacote de dados [Zhang et al. 2018d], facilitando sua obtenção. Conforme será discutido na Seção 2.4.4, ataques podem explorar componentes auxiliares da arquitetura, como os nós encaminhadores, para afetar a disponibilidade.

2.4.2. Inicialização do modelo de confiança, das chaves e das políticas

O arcabouço de segurança da arquitetura NDN usa um conjunto de estratégias criptográficas bem consolidadas, um esquema de nomeação expressivo e um conjunto de políticas de confiança que empoderam os desenvolvedores de aplicações na definição de regras de validação que, em última instância, garantem a autenticidade, integridade e confiabilidade dos dados. O desafio reside, portanto, na inicialização desses componentes, em particular, como obter a âncora de confiança, os certificados, as políticas de confiança e aplicar mecanismos de controle de acesso.

Os certificados que compõem a âncora de confiança possuem confiabilidade intrínseca [Stallings 2020] que, por sua vez, é usada para derivar a confiança em outros certificados. Desta forma, sua inicialização torna-se uma etapa crítica para garantir a segurança do sistema. Visando prover suporte a uma vasta gama de casos de uso, a arquitetura NDN permite que cada aplicação defina como será realizada a inicialização da âncora de confiança [Zhang et al. 2018d]. De forma geral, o processo deve ser realizado de forma segura, tipicamente por um canal fora da banda (do inglês, *out-of-band*). O caso mais simples é inserir as entradas na âncora de segurança manualmente ou durante a instalação do nó ou da aplicação NDN [Zhang et al. 2018d]. Outra opção é através de interações presenciais e uso de mecanismos auxiliares como *QR code* [Gawande et al. 2019], ou ainda pela comunicação direta entre os nós em uma rede P2P [Zhang et al. 2018d].

A obtenção dos certificados é feita através do envio de um pacote de interesse pelo nome da identidade, seguido pelo pacote de dados com o certificado obtido do produtor ou do *cache* oportunístico. A depender da aplicação, os certificados podem ser armazenados em um repositório central (e.g., aplicações de nuvem) ou obtidos diretamente de outros nós (e.g., aplicações distribuídas/P2P). A semântica associada ao certificado e sua aplicabilidade para assinar dados serão validadas através das políticas de confiança, que também podem ser instaladas manualmente na aplicação [Zhang et al. 2018d] ou obtidas

dinamicamente através de pacotes de interesse/dados [Yu et al. 2015]. Neste último caso, a API de validação dependerá de, ao menos, uma política de confiança básica para validar outras políticas de confiança [Yu et al. 2015]. Por exemplo, a política de confiança básica pode definir que pacotes de dados do escopo da políticas de confiança devem ser assinados por uma chave da âncora de confiança, garantindo assim uma inicialização segura.

A partir dos componentes anteriores, é possível alcançar requisitos de controle de acesso na rede, como confidencialidade, integridade, autenticidade e autorização. A autenticidade e integridade dos dados é alcançada através da validação da assinatura digital no pacotes de dados e da validação do nome em relação à política de confiança. O primeiro faz uso de primitivas básicas da criptografia assimétrica e algoritmos de resumo digital (*hash*) [Stallings 2020], enquanto o último utiliza expressões regulares enriquecidas para flexibilizar a validação do esquema de nomeação [Yu et al. 2015]. Embora mais comumente aplicada aos pacotes de dados, os pacotes de interesse também podem ser assinados e validados seguindo as mesmas premissas. Nesses casos, é mais comum uso de criptografia simétrica de chave compartilhada [Li et al. 2019a].

Já a confidencialidade requer um processo ligeiramente mais complexo. A estratégia geralmente utilizada para confidencialidade é baseada no protocolo de troca de chaves Diffie-Hellman [Stallings 2020], que automaticamente estabelece chaves de sessão para cifragem/decifragem ponto a ponto. Tal processo não se aplica para comunicação entre múltiplas partes, que é o caso mais comum em NDN. Para superar essa limitação, algumas propostas fazem uso da semântica do esquema de nomeação para embutir informações adicionais que permitam estabelecer chaves de sessão para múltiplas partes [Zhang et al. 2018d, Zhang et al. 2018c, Marxer and Tschudin 2017]. A ideia básica consiste em definir uma nova entidade chamada "gerenciador de acesso" (que pode ser o *proprietário* da aplicação NDN em questão), que criará políticas de controle de acesso, e o produtor de dados que criará chaves de sessão para cifrar/decifrar os dados.

As políticas de controle de acesso criadas pelo gerenciador de acesso basicamente são pares de chave pública e privada, chamadas KEK (chave de cifragem de chave) e KDK (chave de decifragem de chave) respectivamente, criadas de forma granular por espaço de nomes, que permitirão ao produtor de dados distribuir as chaves de sessão para criptografia para múltiplos consumidores [Zhang et al. 2018c]. Assim, o produtor obtém a chave pública KEK do gerenciador de acesso e cada consumir obtém a chave privada KDK do gerenciador de acesso, que deve ser criptografada com a chave pública do consumidor caso o consumir esteja autorizado a acessar aquele espaço de nomes. Em seguida o produtor cria uma chave simétrica de sessão, chamada CK (chave de conteúdo), para a comunicação com os múltiplos consumidores e distribui sob-demanda a chave CK cifrada com a chave KEK, cuja posse da KDK é garantida apenas aos consumidores autorizados, portanto restringindo o acesso à chave CK.

Tais políticas de controle de acesso também podem ser aplicadas para autorização. Neste caso, o gerenciador de acesso, beneficiando-se do esquema de nomeação, adiciona um sufixo ao espaço de nomes do produtor quando da distribuição da chave KEK, por exemplo `time/8am/6pm`, para que a aplicação produtora garanta o cumprimento da política, nesse caso produzindo dados apenas para consumidores autorizados e dentro do horário estabelecido.

Em verdade, o desafio de controle de acesso e confidencialidade entre múltiplas partes também ocorre para outras funções criptográficas, como a própria assinatura digital. Resultados preliminares apontam, inclusive, para a necessidade de revisão do modelo de confiança da NDN para ofertar suporte a tais requisitos [Zhang et al. 2021].

2.4.3. Aplicações de segurança

Construídas com base no arcabouço de segurança da NDN e nos modelos de confiança, chaves e políticas, uma série de aplicações de segurança vem sendo desenvolvidas para adicionar funcionalidades comuns às redes tradicionais e também novas funcionalidades.

Controle de acesso

Regras de controle de acesso são utilizadas para definir autorização ou privilégios de determinadas entidades para acessar determinados conteúdos ou recursos. A NDN provê diversos mecanismos que viabilizam soluções de controle de acesso, destacando-se o esquema de nomeação semanticamente expressivo. Nour et al. 2021a apresentam uma classificação detalhada dos mecanismos de controle de acesso diferenciando aqueles baseados em cifragem do conteúdo e os independente de cifragem. Essa categorização inicial é subdivida em soluções baseadas em atributo, baseadas no esquema de nomeação, baseadas em identidade, baseadas em um intermediador (*broker*), baseadas em controles no *cache*, dentre outras [Nour et al. 2021a].

Zhang et al. 2018c apresentam a modelagem de um esquema de controle de acesso que permite cifragem dos dados e distribuição de chaves. Utilizando a semântica de nomeação especialmente modelada para expressar regras de controle de acesso, os produtores são aptos a controlar acesso aos dados produzidos com base em diferentes critérios (e.g., hora do dia) e apenas os consumidores autorizados recebem chaves de decifragem que os concedem acesso aos dados.

Já Marxer and Tschudin 2017 propõem o uso de ACL para permitir o acesso à coleção de dados (i.e., sub-prefixos do espaço de nomeação) e aos pedaços do conteúdo produzido derivados do conteúdo original. Assim, o consumidor que obtém um conteúdo, em conjunto com as chaves simétricas de decifragem do conteúdo (CK) e a chave pública de decifragem da chave simétrica (KDK), obtém ainda a ACL para controle de acesso daquele pedaço de dados, podendo tornar-se um provedor independente. Em ambos os casos, observa-se uma sobrecarga extra de comunicação para atividades como identificação de consumidores autorizados e estabelecimento de chaves.

Autenticidade e confidencialidade

Soluções de confidencialidade com cifragem baseada em atributos [Lee et al. 2018] estendem as estratégias de criptografia aplicadas aos pacotes de dados a fim de prover proteção sob demanda, entre diferentes domínios administrativos e com possibilidade de agrupamento no processo de autorização. Lee et al. 2018 utilizam o conceito de organização virtual e federação de entidades, sugerindo mecanismos de gestão de membros e confiabilidade entre eles a partir do uso de cifragem baseada atributos. Ramani et al.

2019 usa os atributos como uma alternativa ao modelo padrão de assinatura de pacotes de dados da NDN.

Segundo Ramani et al. 2019, ainda é preciso resolver dois problemas do modelo tradicional: validação dos dados sem demandar requisições adicionais para certificados da cadeia de confiança e anonimidade do produtor. A proposta consiste na introdução de uma entidade Autoridade de Atributos que gera os parâmetros e assinaturas, que são obtidas apenas na inicialização da aplicação e cuja identidade fornece uma visão alto nível da aplicação, mas não do produtor. Uma abordagem híbrida, com múltiplos mecanismos para obtenção da cadeia de confiança, é apresentado em [Gawande et al. 2019]. Os autores Gawande et al. 2019 apresentam a modelagem de uma aplicação para distribuição segura de dados multimídia para uma rede social descentralizada, onde a cadeia de confiança pode ser obtida através de interação presencial dos usuários (que escaneiam um *QR code* contendo os certificados) ou através da troca de mensagens e validação das assinaturas ou através de canais fora da banda.

Esquemas de confiança

Um aspecto chave para a segurança da arquitetura NDN é a distribuição de chaves e o estabelecimento robusto/seguro das cadeias de confiança, especialmente em cenários colaborativos ou com conectividade limitada [Ramani and Afanasyev 2020b]. Diversos trabalhos têm abordado essa questão a partir de diferentes perspectivas: seja propondo modelos de confiança eficientes e robustos para redes veiculares, cuja conectividade dos nós tem curta duração [Ramani and Afanasyev 2020b]; seja através de técnicas de gestão de certificados de curta duração como forma de substituir o processo de revogação quando do comprometimento da chave [Ramani and Afanasyev 2020a]. Alguns trabalhos consideram estratégias de gerenciamento de espaço de nomeação, autoridade de prefixos e serviços de busca/mapeamento de nomes [Tehrani et al. 2019, Afanasyev et al. 2017]

Yu et al. 2015 propõem automação para distribuição de esquemas de confiança, chaves de autenticação de pacotes de dados ou mesmo criação de chaves com escopo de nomeação limitado. Sistemas de reputação [Kapetanidou et al. 2020] constituem uma outra abordagem de esquema de confiança, independente de função criptográfica. Sistemas de reputação são especialmente interessantes em cenários cuja sobrecarga da validação das assinaturas é proibitiva para que seja aplicada em todo pacote de dados, como é o caso dos roteadores NDN. Se, por um lado, os consumidores validam as assinaturas de todos os pacotes de dados, os roteadores acabam sendo alvos de muitos ataques explorando a capacidade de *cache* e reduzidas proteções de segurança. Kapetanidou et al. 2020 apresentam diferentes soluções de confiança baseada em reputação que podem ser aplicadas em roteadores para evitar ataques de negação de serviço.

Autenticação

Gestão de identidade e aplicações de autenticação utilizando o arcabouço de segurança da NDN têm sido o alvo de pesquisas principalmente para ambientes IoT [Li et al. 2019b, Asaf et al. 2020]. Li et al. 2019b propõem um protocolo para autenticação segura de

dispositivos IoT em casas inteligentes, onde assume-se como premissa o uso de um segredo pré-compartilhado entre os dispositivos e a âncora de segurança para, então, derivar o certificado da âncora de segurança – que será utilizado para que o dispositivo autentique outros dispositivos na mesma casa – e seu próprio certificado assinado pela âncora – que será utilizado para que outros dispositivos verifiquem sua identidade.

Também são propostas estratégias de autenticação mais robustas para dispositivos com recursos adicionais, a saber: geração do par de chaves no próprio dispositivo, quando da existência de bom grau de entropia no dispositivo; reuso de chaves no processo de re-autenticação, quando da existência de mídias de armazenamento seguras; caso o dispositivo possua interfaces interativas como teclas e monitor, é possível dinamicamente estabelecer o segredo com a âncora para troca dos certificados – de forma similar ao processo de empareamento de dispositivos *Bluetooth* e os códigos de autorização [Li et al. 2019b].

Asaf et al. 2020 apresentam um levantamento sobre o uso de *blockchain* em NDN, abrangendo desde a descrição em alto nível dos principais componentes/camadas de um arcabouço *blockchain* e sua integração com NDN até um levantamento de trabalhos aplicando a tecnologia de *blockchain* em contextos específicos de aplicações NDN. Exemplos de contextos apresentados incluem: segurança e privacidade – proteção do conteúdo, gerenciamento de chaves, segurança no *cache*, privacidade; suporte a funções específicas de rede em ambientes 5G, redes de sensores sem fio, redes veiculares e *ad-hoc*; e outras aplicações em geral – mobilidade, aplicações na área de saúde e concessão de recursos.

2.4.4. Superfície de ataques

A mudança no modelo de comunicação trazida com a arquitetura NDN inevitavelmente implica em novos desafios de segurança e em desafios para tratar problemas conhecidos [Mannes and Maziero 2019]. Mecanismos seguros, eficazes e flexíveis de checagem de integridade e autenticidade dos conteúdos nomeados são necessários, permitindo ao usuário validar se o conteúdo foi alterado e se a origem é legítima.

O plano de dados com manutenção de estado, *cache* oportunístico e diferentes estratégias de encaminhamento demanda mais recursos e está suscetível a diferentes tipos de ataques, incluindo ataques de negação de serviço e ataques de poluição de *cache*. Além disso, devem ser considerados os desafios envolvendo a privacidade a partir de diferentes perspectivas, englobando desde o esquema de nomeação até o armazenamento de informações sensíveis em *caches* intermediários. Como reflexo desses desafios, alguns trabalhos recentes propõem avaliações das superfícies de ataques em diferentes contextos [Mannes and Maziero 2019, Nour et al. 2021b].

Mannes and Maziero 2019 apresentam uma classificação de ameaças de segurança e entidades maliciosas envolvidas em ataques, como mostrado na Figura 2.11. Os ataques são classificados/analisados em três categorias: segurança do conteúdo, do roteamento e do *cache*. Essas categorias são subdivididas de acordo com o tipo de ataque e a entidade maliciosa que executa a exploração. Exemplos de ataques direcionados ao conteúdo incluem: fabricação, renomeação e modificação de conteúdo, monitoramento de conteúdos por nome, análise de conteúdo não cifrado, privacidade do nome, identificação do produtor através do localizador de chave, acesso não autorizado e personificação de produtor.

Ataques ao roteamento incluem: exaustão de recursos através de inundação, ataques à PIT [Seixas et al. 2017], sobrecarga no produtor, exaustão de banda, sequestro de prefixo (do inglês, *prefix hijacking*) e interceptação de tráfego. Por fim, são exemplos de ataques ao sistema de *cache*: *cache snooping*, ataques de temporização, monitoramento de requisições, poluição de *cache*, negação de serviço à CS e injeção de conteúdo ilegítimo. Mannes and Maziero 2019 provêm ainda uma relação de trabalhos que apresentam contramedidas aos ataques apresentados e um detalhamento sobre o modelo de ataque para cada vulnerabilidade.

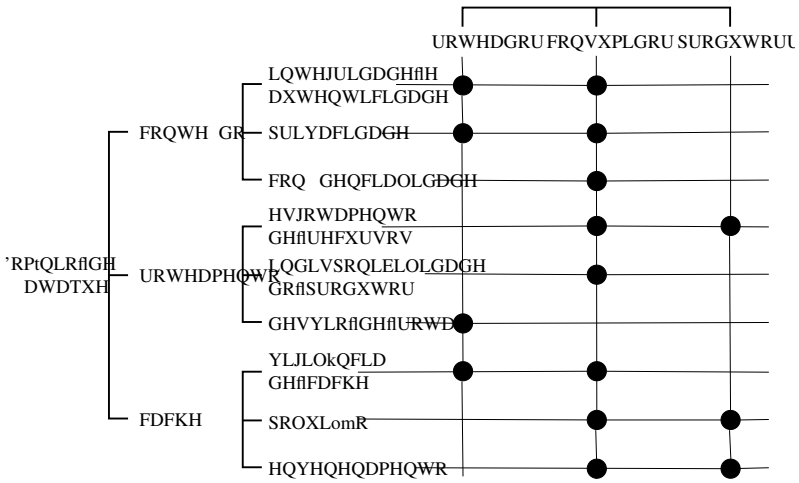


Figura 2.11. Classificação de ameaças e entidades maliciosas. Traduzido de [Mannes and Maziero 2019].

De maneira complementar, Nour et al. 2021b apresentam um levantamento dos principais ataques em redes ICN/NDN, com foco principal em ambientes de rede sem fio. A taxonomia de ataques leva em consideração os componentes da ICN: nome do conteúdo, *cache* e conteúdo em si. Três ataques ao nome do conteúdo são listados: inundação de interesses, ataques de monitoramento e ataques de lista de supervisão (onde o encaminhador NDN pode filtrar/bloquear uma lista pré-definida de nomes de conteúdo). Para ataques ao componente da *cache*, são citados apenas poluição de *cache* e envenenamento de *cache*. Por fim, ataques ao conteúdo incluem: acesso não autorizado e ataques ao encaminhador NDN (do inglês, *Mistreating Attack*). Os ataques aos encaminhadores intermediários, que consistem em um encaminhador malicioso filtrando/bloqueando ou modificando o conteúdo, são bastante danosos ao ambiente sem fio, principalmente em redes *ad-hoc*, nas quais os nós encaminham tráfego em favor de outros nós [Nour et al. 2021b].

Por fim, é importante destacar que o desenvolvimento de uma nova arquitetura implica em novas maneiras de desenvolver aplicações, novos arcabouços de *software*, novas bibliotecas e até novos modelos econômicos associados à distribuição de conteúdo. Todos esses aspectos podem implicar em vulnerabilidades e ataques às aplicações NDN, que embora não sejam diretamente associadas à arquitetura, possuem forte relação e eventualmente podem afetar componentes específicos.

2.5. Aplicações Distribuídas

Nesta seção, serão apresentadas as propriedades da NDN sob a ótica dos modelos de computação distribuída. Além disso, serão abordados os aspectos de comunicação entre processos e a utilização de protocolos para a sincronização de estado distribuído em NDN. Por fim, será apresentado um estudo de caso baseado no uso de vetores de estado em NDN.

2.5.1. Modelos de sistemas distribuídos

A arquitetura NDN possibilita o desenvolvimento de aplicações distribuídas em processos de nós distintos da rede, em que a comunicação entre quaisquer dois processos é mediada por trocas de mensagens em um canal de comunicação. Desta forma, a NDN atua como um sistema distribuído que media a informação dos produtores de dados aos consumidores de dados. Na literatura, modelos de sistemas distribuídos caracterizam um conjunto de propriedades que definem o comportamento de processos e de canais de comunicação.

No projeto de algoritmos distribuídos, o modelo de interação apresenta as premissas do ambiente quanto aos limites temporais de execução pelos processos e de troca de mensagens. A existência de limites bem conhecidos permite, por exemplo, pontos de sincronização entre os processos, que auxiliam na determinação do traço global da execução do algoritmo distribuído. A existência destes limites temporais nos ambientes reais pode ser possível pela reserva de recursos de processamento e comunicação e de um comportamento determinístico, o que é de difícil implementação (e.g., por meio do uso de sistemas operacionais de tempo-real, de redes determinísticas, como CAN e *Token-Ring*, ou por meio de QoS e reserva de recursos). Caracterizamos este conjunto de premissas no modelo síncrono. Lynch 1996 indica que este modelo permite realizar passos de forma sincronizada, isto é, a execução ocorre em rodadas síncronas. Em geral, a existência de tais limites temporais em conjunto com a sincronização periódica dos relógios locais associados aos nós dos processos, estabelece a referência necessária de *slots* de tempo e rótulos temporais para a coordenação das ações distribuídas em rodadas síncronas.

Uma outra possibilidade é assumir a premissa de que os componentes do sistema distribuído (processos e canais de comunicação) executam passos de forma arbitrária, sem limites temporais conhecidos. O modelo assíncrono não permite que os algoritmos assumam quaisquer hipóteses baseadas no tempo de execução dos passos computacionais ou do tempo gasto para troca de mensagens. Assim, ao se projetar um algoritmo para o modelo assíncrono, este não deve confiar em premissas temporais para operação, sendo mais genéricos e portáveis que no síncrono. É importante ressaltar que a NDN, em linhas gerais, não se baseia em premissas temporais.

Contudo, esta mesma ausência de limites temporais não nos permite caracterizar a falha de componentes de forma segura, e assim, algoritmos que requerem coordenação de ações, como o consenso distribuído, não têm garantia de execução em ambientes assíncronos na presença de falhas [Fisher et al. 1985]. Desta forma, obter algumas garantias de execução e comunicação associadas a limites temporais pode ser interessante para a garantia da terminação dos algoritmos. Uma hipótese usual em ambientes de execução real é que um ambiente não síncrono pode se tornar estável com um comportamento "síncrono", a partir de algum momento no tempo, denominado GST (*Global Estabilization Time*) [Dolev et al. 1987]. Esta hipótese de estabilidade pode viabilizar a execução de

algoritmos distribuídos em uma rede NDN.

Para que a aplicação distribuída se torne resiliente, é importante entender os modelos de falhas – i.e., como os processos e canais podem falhar –, o que possibilita o desenvolvimento de mecanismos de tolerância a falhas. Existem diferentes tipos de falhas que podem ocasionar problemas em diversos contextos da NDN. Falhas arbitrárias ou bizantinas estão relacionadas à execução deliberada fora da especificação do algoritmo, que pode ocorrer, por exemplo, por comprometimento do processo ou do nó de execução, ou por mensagens serem corrompidas ou adulteradas no canal de comunicação [Lamport and Fischer 1982]. Na Seção 2.4 são expostos alguns cenários na NDN caracterizados por tal comportamento malicioso. Canais de comunicação também podem falhar por omissão, com perda de mensagens. Processos podem parar a execução em falha por *crash*. Caso haja o retorno deste processo em um ponto de execução anterior ao do *crash*, temos o *crash-recovery*. A infraestrutura de comunicação da NDN é, de certa forma, resiliente à perda de mensagens, embora seja necessário considerar cenários de *crash* de processos, ou de conexão intermitente dos mesmos.

2.5.2. Propriedades da NDN para computação distribuída

A separação do identificador e localizador do conteúdo e o modelo de comunicação baseado em nomes impõem mudanças fundamentais na forma de pensar aplicações em NDN. O espaço de nomes (do inglês, *namespace*) compartilhado proposto pela arquitetura permite que o encaminhamento de pacotes não se altere com mudanças de topologia ou movimentação e conectividade de nós, mas, tão somente no alcance do *namespace*.

Conforme já discutido anteriormente, os nós da NDN podem ser produtores ou consumidores de dados, ou intermediar esta relação ao encaminhar requisições. O modelo de comunicação centrado em dados utilizado permite desacoplar as requisições de referências temporais. O modelo adotado não requer nem mesmo que produtor e consumidor estejam ativos ao mesmo tempo. Pode-se, portanto, assumir assim que a NDN permite um mecanismo de comunicação completamente assíncrono. Ou seja, não é necessário limites temporais conhecidos para o processamento e comunicação ao longo de toda a execução. Contudo, a terminação adequada de aplicações distribuídas na NDN requer que haja, em algum momento, a estabilidade nos limites temporais da rede, assumindo-se as premissas de modelos parcialmente síncronos, como os da hipótese GST.

Deve-se notar que, apesar desta possibilidade de desacoplamento temporal, é possível incorporar referências temporais ao associar um determinado interesse a um *lifetime*. Assim, há a possibilidade de premissas fortes para limites temporais, exercitando aplicações síncronas na NDN. Mastorakis et al. 2018 propõem o protocolo *Realtime Data Retrieval* (RDR), que minimiza a latência de obtenção da informação mais atual desde que o produtor (ou um nó delegado por este) e o consumidor estejam ativos ao mesmo tempo. Neste contexto, o *lifetime* de uma entrada da PIT é um requisito temporal ajustado conforme a percepção do ambiente. Embora o protocolo não necessite de sincronização de relógios ou premissas mais robustas (como uma infraestrutura de enlace determinística), na presença destas, há um ambiente síncrono que favorece aplicações de tempo real estrito.

No que reflete ao modelo de falhas, pode-se assumir que abordagens apresentadas na Seção 2.4 são capazes de mitigar ataques maliciosos relacionados aos canais de comu-

nicação. Ademais, o uso de *in-network caching* permite a entrega confiável, em caso de não haver falhas do produtor e do consumidor (i.e., canais de comunicação confiáveis – sem falhas). A assincronicidade de operação entre produtor e consumidor pode até mesmo permitir a entrega caso o dado já tenha sido encaminhado para CS em nós intermediários e, assim, conseguir alcançar o consumidor.

A NDN pode ser implantada em cenários de mobilidade com alta taxa de volatilidade (*churn*) de nós, ou seja, há ingresso e saída (ou desconexão) de nós produtores e consumidores ao longo da execução. Este cenário é similar ao da comunicação assíncrona assumida por Paxos [Lamport et al. 2001], no qual processos podem ingressar e sair a qualquer tempo, refletindo o modelo de falhas de processos *crash/recovery*. Isso significa que uma desconexão eventual pode ser representada pelo *crash* ou inacessibilidade de um nó, que pode se recuperar e ficar novamente disponível.

2.5.3. Aspectos de comunicação entre processos na NDN

Aplicações distribuídas utilizam-se da troca de mensagens para coordenar ações. Em redes TCP/IP, este mecanismo de comunicação entre processos é exercitado por meio de *sockets*. Outros padrões arquiteturais podem ser utilizados por meio de camadas de *middleware*, provendo outras semânticas para interação entre processos. Os mecanismos definidos na arquitetura NDN reduzem o *gap* semântico rede-aplicação e auxiliam na implementação de alguns padrões, como o uso de funções nomeadas e do paradigma *publish/subscribe*.

Invocação remota como suporte à computação distribuída

A invocação remota de métodos/funções é um mecanismo de comunicação entre processos que permite a execução dessas tarefas como se fossem locais, burlando aspectos de localização e provendo outra semântica para o programador da aplicação cliente [Waldo et al. 1996]. As arquiteturas baseadas no modelo ICN têm algumas limitações para lidar com conteúdo dinâmico. Consequentemente, existem desafios para lidar com suporte à execução de funções de rede. No entanto, algumas iniciativas, como *Named Function as a Service* (NFaaS) [Król and Psaras 2017], têm endereçado esse problema.

A NFaaS permite mover a computação para a borda da rede. Esta abordagem permite a execução de microsserviços e funções *stateless* baseada em um novo componente da NDN, o *Kernel Store*, que armazena as funções a serem executadas. Novas primitivas são incorporadas permitindo a migração de funções de um nó da NDN para outro de forma dinâmica, ou mesmo o balanceamento de carga com o encaminhamento de parte das requisições de um nó com sobrecarga a outros nós na NDN. Essa abordagem é estendida no RICE [Król et al. 2018], que possui primitivas para desacoplar a invocação do método do retorno dos resultados permitindo computação de longa duração, além de permitir autenticação prévia de clientes e o suporte a conjuntos de parâmetros complexos.

Publish/subscribe como suporte à computação distribuída

Publish/subscribe é um padrão arquitetural que possibilita que entidades publicadoras (do inglês, *publishers*) publiquem conteúdos associados a um ou mais tópicos, e entidades

assinantes (do inglês, *subscribers*) recebam notificação somente dos tópicos que possuam interesse. Este esquema de comunicação assíncrona é baseado em eventos e desacoplado em tempo, espaço e sincronização entre as partes, de forma que permite a implementação de sistemas distribuídos de larga escala [Eugster et al. 2003]. Nour et al. 2019c exploram diferentes possibilidades de construção semântica de sistemas *publish/subscribe* em redes ICN, por exemplo: (i) única requisição-única resposta – um nó assinante requer conteúdo já publicado e obtém uma resposta, qualquer atualização deve ser obtida pela requisição de um novo pedido de conteúdo; (ii) única requisição-várias respostas – um nó assinante requer conteúdo, mas pode receber diversas respostas ao longo do tempo, de acordo com a semântica da aplicação; (iii) entrega periódica – um nó assinante requer conteúdo periódico a cada intervalo de tempo, e.g., um sensor enviando atualizações a cada 10 minutos; (iv) *N* respostas – um nó assinante requer um número específico de respostas sobre um conteúdo (e.g., os próximos 10 quadros de um vídeo); e (v) entrega condicional – um nó assinante requer receber conteúdo somente se este atender a certos requisitos condicionais.

Na NDN o roteamento e encaminhamento de mensagens é baseado em nome. O conteúdo publicado pode ser obtido individualmente pelos assinantes por meio da troca de mensagens de interesse/dados da NDN. Ou seja, a NDN é nativamente um sistema de única requisição-única resposta. Uma aplicação na NDN pode adaptar a semântica de *publish/subscribe* utilizada a partir do envio de novos pacotes de interesse. Por exemplo, em uma rede de sensores, caracterizada por uma necessidade de comunicação com entrega periódica, requisita-se a atualização da informação em cada período de monitoramento. Esse comportamento pode gerar congestionamentos, assincronia e problemas com a agregação (duas ou mais respostas serem respondidas da mesma forma pela agregação dos interesses em nós intermediários).

Cenários em que há um grupo de *N* publicadores e *M* assinantes para o mesmo conteúdo/tópico do *publish/subscribe* são desafiadores em NDN. Nour et al. 2019c tratam o modelo *publish/subscribe* por meio da semântica de comunicação em grupo. Um tópico é representado em um grupo, ao qual assinantes podem se unir (do inglês, *join*) ou sair (do inglês, *leave*), o produtor mantém a visão do grupo e pode também prover a retirada de membros de acordo com a semântica. Nesse tipo de cenário, é importante estabelecer mecanismos para sincronização do estado distribuído entre multipartes.

2.5.4. NDN Sync: sincronização de estado distribuído na NDN

O modelo *publish/subscribe* implementado pela NDN favorece a interação entre um consumidor e um produtor. Contudo, um compartilhamento de conjunto de dados que possibilite múltiplas partes é desejável para o uso em aplicações distribuídas. Neste contexto, NDN Sync se apresenta como uma abstração para comunicação multiparte agnóstica de conexão na NDN. O compartilhamento de conjunto de dados deve observar que esta comunicação multiparte é assíncrona, ou seja, nem todas as partes podem estar conectadas ao mesmo tempo. Desta forma, o NDN Sync é um mecanismo de sincronização do espaço de nomes entre um grupo de entidades.

Entidades organizadas a partir de um *namespace* formam um grupo que permite que novos dados possam ser consumidos por todos os membros deste *namespace*. A abstração criada estende a comunicação único-produtor/único-consumidor da NDN para um modelo

de multi-produtor/multi-consumidor: qualquer produtor que distribua informação no *namespace* terá a informação sincronizada entre os consumidores do *namespace*.

O NDN Sync provê um mecanismo de notificação que permite que todos os processos interessados em um determinado *namespace* possa receber notificações quando novos dados são produzidos e obtê-los no melhor momento (se isto for de seu interesse). Isto implica em um mecanismo que possibilita detecção de mudanças no estado dos dados. Ou seja, cada participante interessado no *namespace* mantém sua cópia local do estado de dados e, por meio do protocolo NDN Sync, obtém notificações e pode se manter atualizado. A estratégia de detecção de mudanças no estado de dados considera a forma como o estado de dados é representado internamente (estrutura de dados) e a estratégia de nomeação dos dados. Por exemplo, o uso de informação sequencial permite representar mudanças e o de nomes arbitrários pode ter uma semântica associada à estratégia de sincronização.

Na Figura 2.12 é possível observar o funcionamento básico do NDN Sync: após publicar novos dados, um participante notifica as demais partes através do anúncio de dados (Passo 1), quer seja por envio de mensagem de *interesse Sync* quer seja por resposta a mensagens anteriores desses interesses. De acordo com o protocolo, esta mensagem já pode informar a alteração, ou tão somente ser uma notificação. No segundo caso, um segundo passo (Passo 2) é necessário com a troca de mensagens de interesse e dados. Nós podem sinalizar interesse de longo termo em mudanças: um *interesse Sync* de longo termo é mantido pendente em todos os nós encaminhadores por um longo tempo. Ainda, é possível utilizar mensagens periódicas de monitoramento (*heartbeats*) para detectar mudanças no *namespace* ou no conjunto de partes interessadas (*membership* do grupo).

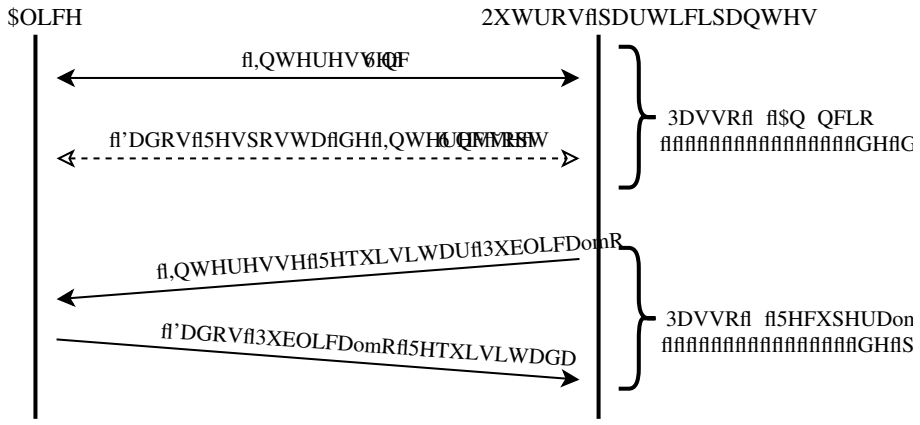


Figura 2.12. Funcionamento básico de NDN Sync. Adaptado de [Shang et al. 2017].

Diferentes protocolos de sincronização foram propostos desde 2012 [Shang et al. 2017]. Protocolos como CCNx 0.8 Sync, ChronoSync e RoundSync utilizam árvores, como *Merkle trees*, para representar o estado de dados, mantendo *hashes* ou *digests* que são utilizados para detectar mudanças e atualizar o estado. Já o CCNx 1.0 Sync utiliza manifestos para anunciar uma atualização. O manifesto é um conjunto de metadados da aplicação que são associados aos objetos mantidos na coleção local. O *hash* do manifesto é difundido através do envio de *interesse Sync* e, por meio desta difusão, os nomes associados aos manifestos são comparados e os dados de interesse são solicitados à rede. Alguns

protocolos como o iSync, syncps e PSync fazem uso do filtro de Bloom invertido (IBF) para identificar dados faltantes. O filtro de Bloom permite verificar, de forma probabilística, se elementos pertencem a um conjunto. O filtro de Bloom invertido, por sua vez, permite um conjunto de operações, inclusive comparar diferenças entre IBFs. A partir dessa comparação, é possível identificar os dados faltantes e solicitar as informações necessárias para sincronização de estado.

Em contextos como o da NDN, em que a forma assíncrona de troca de dados não utiliza informações temporais, como relógios sincronizados e rótulos de tempo (do inglês, *timestamps*), eventos de atualização de estado podem ser correlacionados de forma lógica. Lamport 2019 sugere o uso de uma notação baseada na contagem de eventos locais que utiliza um vetor denominado relógio vetorial para prover uma ordem parcial de eventos que ocorrem em um sistema distribuído. Este vetor de estados do sistema pode ser utilizado em aplicações que determinem o estado global e executem ações necessárias de coordenação distribuída [Chandy and Lamport 1985]. A partir do *Vector Sync*, propostas de protocolos baseados neste conceito foram desenvolvidas, como *State Vector Sync*, PLI-Sync e ICT-Sync [Shang et al. 2017].

2.5.5. Do NDN Sync à computação distribuída: estudo de caso

O uso de vetor de estados, inicialmente proposto no *Vector Sync*, implica em manter o contador de eventos associado a cada produtor de conteúdo. Na produção de conteúdo, há o anúncio (*interesse de notificação*): caso outra parte verifique que o vetor difundido tem componente maior que o vetor local, procede-se a atualização e os dados podem ser obtidos pela troca de mensagens de interesse e dados necessárias (ver Figura 2.13).

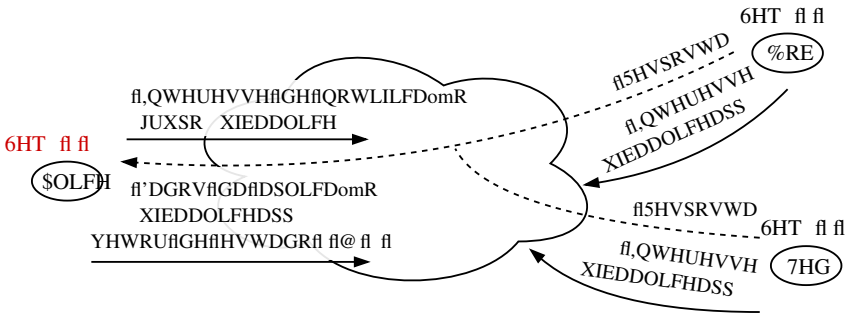


Figura 2.13. Sincronização de estado no *Vector Sync*. Adaptado de [Shang et al. 2017].

O projeto desta solução para o *Vector Sync* implica em manter qual o grupo de produtores de conteúdo ativos, o que remete a primitivas de um protocolo de comunicação em grupo clássico. O *membership* do grupo de produtores ativos associados ao *dataset* é mantido com *Heartbeats interest*. Cada participante deve enviar interesses periódicos, denominados *heartbeats*, indicando seu prefixo e vetor de estados para o grupo. Um processo escolhido como líder coleta estas mensagens e difunde a lista de membros ativos do grupo. Desta forma, é possível verificar quando há solicitações de ingresso ou de saída do grupo, ou suspeitas de falhas (participante inativo), difundindo-se a nova informação.

A abordagem baseada em um líder assume que este é o participante com maior número de ordem no prefixo. Na ausência de manifestação do líder em resposta aos

heartbeats, um novo líder pode ser escolhido, aplicando o algoritmo do *Bully* [Stoller 1997]: ou seja, o processo ativo de maior número de ordem no prefixo assumirá este papel. O uso de *heartbeats* para criar a nova visão de grupo periodicamente é uma abordagem clássica em comunicação em grupo [Cristian and Schmuck 1995]. Este padrão de comunicação estabelece o bloco de detector de defeitos, que auxilia na coordenação da computação distribuída em suspeita de falhas e cuja efetividade depende das premissas temporais do ambiente [Chandra and Toueg 1996].

A combinação da abordagem de gerenciamento de *membership* e da semântica de relógios vetoriais para a sincronização do estado do conjunto de dados permite ao *Vector Sync* manter aplicações distribuídas baseadas em múltiplos produtores e múltiplos consumidores, sem o uso de interesses de longo termo. Deve-se ressaltar que o padrão de comunicação da NDN permite seu uso em ambientes com mobilidade, comumente caracterizados por alta rotatividade de nós. Estes cenários são desafiadores para protocolos de comunicação em grupo, pois implicam em constante mudanças de visão.

A sincronia de visões geralmente implica em alta carga, em consequência a um número considerável de saídas acidentais ou exclusões por falsas suspeitas (devido a desconexões momentâneas). O *State Vector Sync* (SVS) evolui o *Vector Sync* permitindo a partição da rede, em que a abordagem de comunicação em grupo convencional mediada por líder é substituída, utilizando-se um vetor de estado dinâmico, que mantém informação atual sobre produtores conhecidos, sem necessidade de um *membership* formal. Assim, o SVS suporta maior nível de *churn*, adaptando-se ao padrão de comunicação assíncrona inerente à NDN e suportando particionamento da rede. Ou seja, participantes podem publicar e propagar novos dados a qualquer momento sem um ingresso formal, mantendo-se a disponibilidade [Shang et al. 2017].

A opção por maior resiliência em um contexto de conectividade intermitente é incrementada no PLI-Sync em que um *prefetch* oportunístico é exercitado: participantes obrigatoriamente atualizam todos os dados disponíveis e sinalizam interesse em sequências ainda a serem produzidas. Os resultados preliminares indicam que esta abordagem favorece cenários com alta perda de dados. Já o ICT-Sync propõe alterações ao *Vector Sync* com o uso de nós intermediários que copiam dados de produtores e os mantém disponíveis ao grupo mesmo quando os produtores originais estão inacessíveis. O uso de lista de mapeamento desacopla os componentes do vetor de estados dos participantes e permite uma abordagem dinâmica, dispensando o *membership* baseado em líder do *Vector Sync*.

Conforme o Teorema CAP [Simon 2000]: (C) consistência, (A) disponibilidade e (P) tolerância, a partição para um conjunto de dados compartilhado, podemos verificar que a resiliência à partições e a disponibilidade de dados destas abordagens são obtidas a cargo de consistência fraca. Desta forma, o NDN Sync se apresenta adequado a um amplo conjunto de aplicações distribuídas em que os requisitos condizem com as premissas de possibilitar disponibilidade de dados com um modelo de comunicação assíncrono, que seja resiliente a particionamento e alto *churn*.

2.6. Ambientes de Experimentação

Nesta seção, serão apresentados os ambientes de experimentação em NDN e serão descritas as informações gerais para a atividade prática do capítulo.

2.6.1. Simulação com o ndnSIM

O simulador de redes baseado em eventos discretos ndnSIM[3] [Mastorakis et al. 2017] foi desenvolvido como um módulo específico para experimentação de NDN no simulador de redes ns-3[4]. Criado em 2012, o ndnSIM se consolidou como uma plataforma largamente utilizada na comunidade de pesquisa em NDN para rápida prototipagem de aplicações, avaliações de desempenho com alta escalabilidade e vasta gama de cenários, topologias, fatores e níveis de experimentação.

O ndnSIM é apto à execução de experimentos de larga escala, com milhares de nós executando a pilha NDN em *hardware* comum [Mastorakis et al. 2017]. Em termos de diversidade de fatores e níveis para análise de desempenho, como o ndnSIM é construído no topo do ns-3, é possível testar configurações de topologias e parâmetros (e.g., largura de banda e atraso do enlace), simular diferentes modelos/protocolos da camada de enlace e modelos de propagação e mobilidade, além de permitir uma customização granular nos parâmetros/estratégias da pilha NDN. Assim, é possível escolher desde a estratégia de encaminhamento (e.g., *best route*, *multicast* e ASF) por prefixo, até o tamanho da *cache* e algoritmo de substituição de dados, incluindo ainda detalhes como política para tratamento de dados não solicitados e parâmetros especializados das *faces* NDN.

Uma das desvantagens do ndnSIM é a necessidade de adaptação no código para execução em ambiente real. Além disso, muitos estudantes iniciando a pesquisa em NDN reportam uma curva de aprendizagem menos acentuada, que vem sendo melhorada com a disponibilização de muitos exemplos pelo projeto.

2.6.2. Emulação de aplicações no Mini-NDN

Emuladores de rede fornecem uma plataforma de experimentação complementar aos simuladores: embora menos escaláveis e com menor conjunto de cenários de experimentação, o sistema de emulação tipicamente fornece características próximas ao ambiente real e permite utilizar um protótipo de aplicação similar ao que será utilizado no ambiente real.

O Mini-NDN[5] é um emulador leve de redes NDN que apoia o desenvolvimento, teste e avaliação de desempenho de aplicações NDN, em um ambiente simples ou em um *cluster* de nós. O Mini-NDN foi originalmente baseado em um projeto chamado Mini-CCNx[6], que por sua vez era um *fork* do Mininet[7], e atualmente oferece suporte às bibliotecas NDN, sistema de encaminhamento NFD, protocolos de roteamento NLSR e NDVR, além de um conjunto de ferramentas do projeto NDN, como o *ndnping* (teste de conectividade) e o *ndn-chunks* (para transferência de dados).

O Mini-NDN possui integração com o Mininet-WiFi, que permite a emulação de estações e pontos de acesso Wi-Fi (modo infraestruturado e *ad-hoc*) baseados no módulo sem fio do kernel Linux *80211_hwsim*. O Mininet-WiFi, integrado ao Mini-NDN, permite executar aplicações NDN no ambiente de rede sem fio virtual e com suporte a diferentes modelos de propagação de sinal (e.g., *Friis*, *Log Distance*, *Log Normal Shadowing*, etc.)

[3] https://ndnsim.net/current/
[4] https://www.nsnam.org/
[5] http://minindn.memphis.edu/
[6] https://github.com/chesteve/mn-ccnx
[7] https://github.com/mininet/mininet

e diferentes modelos de mobilidade (e.g., *Random Walk*, Gauss Markov, Caminhada de Levy, RPGM, etc.), além de oferecer integração com o SUMO[8] para cenários de VANET.

A principal desvantagem do Mini-NDN é a escalabilidade, sendo diretamente dependente do desempenho do *hardware* em que está sendo executado, com relatos de limite superior em torno de algumas centenas de nós NDN em uma execução. O projeto provê suporte ao modo de execução em *cluster* do Mininet, que reduz essa limitação.

2.6.3. *Testbed* NDN

O projeto NDN também conta com um *testbed*[9] internacional envolvendo 36 nós em diferentes países. Trata-se de uma rede utilizada para experimentação e validação de protocolos e aplicações em um ambiente com equipamentos reais. Atualmente, um ponto de presença do *testbed* encontra-se disponível na UFBA[10], como mostra a Figura 2.14.

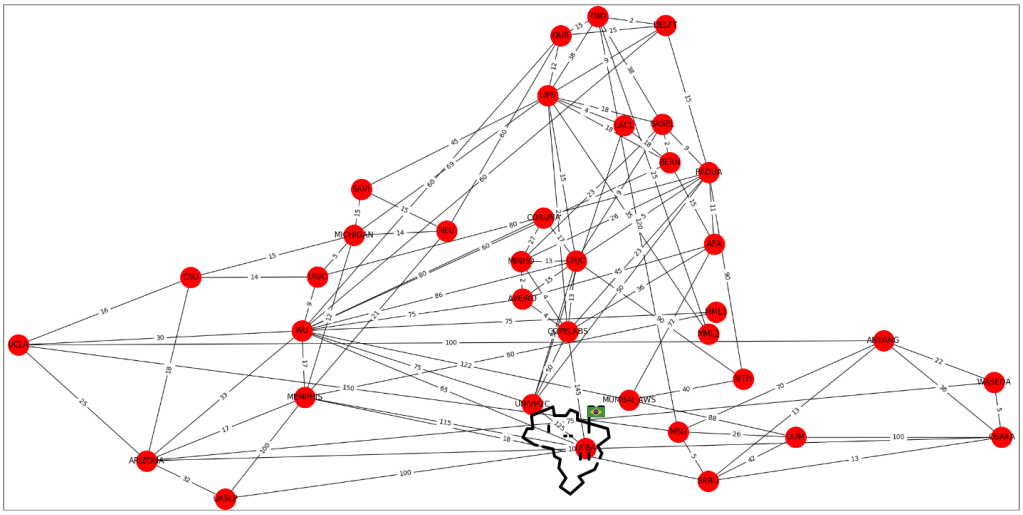


Figura 2.14. *Testbed* NDN (36 nós interconectados, 97 *links*, protocolo NLSR). Adaptado de [NDN 2021].

2.6.4. Bibliotecas, APIs e outros ambientes de experimentação NDN

Além das ferramentas de simulação, emulação e experimentação (*testbed*), o projeto NDN recentemente apresentou um conjunto de APIs e bibliotecas de suporte para aplicações NDN. Essas ferramentas abstraem os detalhes de requisições individuais de pacotes de interesse e dados, validações e tratamento de NACKs. Duas dessas bibliotecas são a NDN-Lite [Yu et al. 2021] e a NDN-CNL [Thompson et al. 2019]. Elas permitem o desenvolvimento de aplicações que fazem uso de um espaço de nomes disponível (ou automaticamente descoberto), se registram em uma ramificação do espaço de nomes e publicam dados. Isso possibilita criar uma abstração alto nível que simplifica as funções do modelo *publish/subscribe* e facilita a experimentação em cenários IoT e outros ambientes.

Outro contexto de execução importante de mencionar são os ambientes de alta capacidade de comutação de pacotes, que podem beneficiar a comunidade de *big data* e

[8] https://www.eclipse.org/sumo/
[9] https://named-data.net/ndn-testbed/
[10] https://ufba.testbed.named-data.net/

ciência de dados. Shi et al. 2020 foram pioneiros em executar o plano de encaminhamento NDN em *hardware* não especializado atingindo taxas de comutação superiores a 100 Gbps. Tais resultados foram possíveis graças a uma combinação de técnicas de aceleração do *pipeline* de processamento de pacotes através do arcabouço DPDK, além de otimizações nos algoritmos e estruturas de dados do *pipeline* de encaminhamento da NDN.

2.6.5. *HandsOn*: demonstração e prática de NDN

Nesta seção, será apresentada uma atividade prática cujo objetivo é demonstrar o funcionamento da arquitetura NDN, seus principais componentes, ambientes de execução e o desenvolvimento de uma aplicação simples utilizando as APIs de comunicação em rede.

O desenvolvimento de uma aplicação NDN requer a definição de um esquema de nomeação da aplicação, definição do modelo de confiança e o uso de uma série de funções das APIs das bibliotecas ndn-cxx, NDN-Lite ou NDN-CNL. Para simplificar a atividade prática, serão fornecidos o design de um esquema de nomeação da aplicação, modelo de confiança e um esqueleto base do código fonte. Além disso, o desenvolvimento pode ser direcionado a um dos componentes da arquitetura, tais como: desenvolvimento de uma política customizada de substituição de *cache*, desenvolvimento de uma nova estratégia de encaminhamento, desenvolvimento de um protocolo de roteamento, dentre outros. Considerando que o objetivo desta atividade é permitir que o leitor coloque em prática os conceitos apresentados nas seções anteriores, ao invés de apresentar uma exploração profunda dos componentes e APIs, ao longo da execução da prática os pontos chaves da arquitetura serão mencionados para que possam ser alvo de investigação futura.

Toda a documentação sobre o design da aplicação, todos os códigos-fonte, bibliotecas e programas auxiliares utilizados no escopo desta seção estão disponíveis em um repositório do projeto[11]. O repositório deve ser clonado para facilitar a execução da atividade prática, em seguida deve-se seguir as instruções de execução que estão disponíveis no próprio repositório. Para tal, é necessário clonar o respositório do projeto: `git clone https://github.com/insert-lab/mc-ndn-sbrc2021` e seguir as instruções contidas no arquivo `README.rst`.

2.7. Desafios de Pesquisa

Nesta seção serão apresentadas as questões de pesquisa relacionadas à mobilidade, segurança e aplicações distribuídas em NDN, incluindo roteamento e comunicação *multi-hop*, às limitações dos protocolos da camada de enlace, mecanismos de incentivo e reputação, e mecanismos de difusão, sincronização de estado e *handoff*.

2.7.1. Mobilidade

Apesar do meio sem fio favorecer a comunicação por difusão (*broadcast*), uma grande parte das implementações de redes se baseia no uso de endereçamento da camada de enlace para direcionar os quadros às interfaces de destino. No entanto, a NDN é caracterizada pela comunicação *multicast*, em que um nó toma as decisões de encaminhamento a partir das informações de estado, alcançabilidade de prefixos e conteúdos das *caches* locais.

[11] `https://github.com/insert-lab/mc-ndn-sbrc2021`

Encaminhamento NDN na camada de enlace

Em redes cabeadas, o encaminhamento da NDN é efetivo por ser realizado para um subconjunto de diferentes interfaces de saída. Contudo, em redes móveis, formadas com nós equipados por apenas uma única interface (*wlan0*), o encaminhamento resulta na inundação de pacotes na rede. Na arquitetura NDN não existe um mapeamento direto entre o nome e um endereço MAC [Kietzmann et al. 2017]. Desta forma, pacotes transmitidos por difusão causam o processamento desnecessário de quadros pelos nós da rede. Consequentemente, em cenários de mobilidade, em que os nós comunicantes possuem um única interface de rede, as informações de roteamento não são suficientes para apoiar as decisões de encaminhamento, dado que a existência de apenas uma interface resulta naturalmente numa inevitável inundação de pacotes.

A fim de mitigar a inundação da rede em cenários de mobilidade, um grande número de trabalhos tem focado na concepção de novas estratégias de encaminhamento. Em geral, os trabalhos levam em consideração métricas como localização, contexto, distância entre nós, estabilidade de enlace, quantidade de saltos, vizinhança, coordenadas geográficas e energia [Tariq et al. 2020, Wang et al. 2020]. Contudo, tais soluções são efetivas para contextos específicos e não são nativas da arquitetura.

Diferente das estratégias de encaminhamento, algumas iniciativas promovem a implementação de técnicas de autoaprendizagem de endereço MAC, através da criação dinâmica de *faces unicast* [Baccelli et al. 2014], uma abordagem semelhante ao mapeamento ARP da arquitetura IP. Tais soluções buscam manter um constante mapeamento de endereço *unicast* a partir das informações dos cabeçalhos dos pacotes de dados recuperados de transmissões *multicast* anteriores. Essas estratégias têm a vantagem de não exigir uma mudança nos protocolos de enlace existentes. Em contrapartida, as mesmas reduzem a capacidade da rede de fazer *cache* oportunístico (característica nativa da arquitetura), uma vez que os pacotes são encaminhados para endereços específicos [Kietzmann et al. 2017].

Ainda visando tratar a inundação no nível das interfaces de rede, alguns trabalhos propõem protocolos de enlace específicos para a arquitetura NDN [Shi and Zhang 2012]. No entanto, essas soluções exigem o processamento de todos os quadros passantes, no nível do enlace, o que prejudica redes formadas por nós com restrições de processamento e energia (e.g., IoT). Além disso, essas soluções tornam necessária a criação de um novo protocolo de enlace, o que limita a adoção e a interoperabilidade com redes existentes, equipadas com protocolos como o Ethernet. Por fim, destaca-se as soluções que propõem a filtragem baseada em nome no nível do enlace [Shi et al. 2016, Li et al. 2019c, Karrakchou et al. 2020b]. Além da interoperabilidade, é uma alternativa que representa uma violação dos papéis das camadas da arquitetura de rede.

Roteamento, comunicação *multi-hop* e recuperação de dados

Tendo em vista que o roteamento apoia a mobilidade atualizando direções até um produtor/repositório e divulgando dinamicamente a alcançabilidade de prefixos, um dos principais desafios consiste em definir como outros componentes da NDN fazem uso de tais informações. Entre as questões em aberto está a: (i) definição a melhor estratégia de

encaminhamento para um determinado cenário: *multicast*, *unicast*, adaptativo, ciente de coordenadas de localização; (ii) identificação de uma estratégia de múltiplos caminhos a ser adotada, considerando o encaminhamento *stateful*, visando balanceamento de carga, caminhos cientes do *cache*, caminhos disjuntos, caminhos com propriedades de tolerância a falhas; (iii) definição de como atualizar dinamicamente as políticas de confiança para permitir autoridade sobre informações de alcançabilidade para nós distintos, diante da mobilidade do produtor.

Além disso, outro aspecto importante é o modelo de mobilidade ao qual o cenário está sujeito, que pode variar em relação à velocidade dos nós, *churn*, distância entre os nós, e diversos outros aspectos. Desta forma, é preciso considerar se: (i) o modelo de mobilidade prever mobilidade do produtor e consumidor ao mesmo tempo; (ii) a possibilidade de contar com serviços de resolução de nomes eficientes; (iii) a possibilidade da adoção de repositórios e *caches* colaborativos impulsionar a mobilidade do produtor. Estas e outras questões apontam para uma necessidade de melhor investigação sobre protocolos de roteamento e suas funções, especialmente nos ambientes *ad-hoc* com alta mobilidade.

2.7.2. Segurança

A NDN se caracteriza pela segurança no nível dos dados, garantida através do uso de primitivas criptográficas conhecidas – especialmente criptografia assimétrica e infraestrutura de chaves públicas –, que, combinadas com uma semântica expressiva do nome e um conjunto de políticas de confiança flexível, permitem a validação do dado em qualquer nó da rede. Ainda que este modelo traga benefícios de segurança à arquitetura, algumas lacunas permanecem em aberto.

Modelo de ameaças

Uma série de ameaças vem sendo apresentadas na literatura, tendo como alvo diversos componentes da arquitetura NDN [Mannes and Maziero 2019, Liu et al. 2019, Nour et al. 2021b]. No entanto, constata-se uma lacuna de investigação especialmente nas seguintes áreas: ataques de negação de serviço das mais variadas formas – distribuídos, iniciados remotamente, através de inundação de interesses, através de assinaturas falsas para dados, temperando parâmetros específicos dos pacotes de interesses/dados, entre outros; nós encaminhadores maliciosos; sequestro ou falsificação de prefixos de nome no roteamento (do inglês, *route hijacking*); acúmulo de material criptográfico a partir do uso intenso de chaves, permitindo processos de criptoanálise; e ataques à privacidade.

É importante destacar, ainda, que pouco se discute sobre os modelos econômicos associados à distribuição de conteúdo na arquitetura NDN, e, historicamente, as superfícies de ataque têm uma forte relação com os aspectos econômicos envolvidos. Assim, é possível que se observe um crescimento na diversidade de ataques e identificação de vulnerabilidades quando vantagens financeiras ficarem evidentes. Verifica-se, portanto, a necessidade de um modelo de ameaças abrangente o suficiente para incorporar mecanismos adicionais ao arcabouço de segurança da NDN e, assim, reduzir a superfície de ataques.

Mecanismos de controle de acesso, autenticação e autorização

A comunicação entre múltiplas partes é uma característica comum em aplicações distribuídas, especialmente em aplicações NDN. O modelo tradicional de infraestrutura de criptografia utilizada na NDN prevê, por padrão, apenas um certificado/identidade para assinar e cifrar os dados. Pesquisas em estágio inicial sugerem o uso de uma terceira entidade central que faz a gestão de múltiplas assinaturas ou distribuição de chaves para tratar esse desafio. [Zhang et al. 2021]. Além disso, alguns trabalhos propõem soluções de controle de acesso baseado em nome e autorização [Zhang et al. 2018c]. Em ambos os casos, o uso de uma entidade centralizada contrasta com o modelo totalmente distribuído da NDN. Além disso, as propostas sugerem mudanças nos componentes da arquitetura, deixando em aberto desafios quanto à escalabilidade, tolerância à falhas e censura.

Modelo de confiança

Na Seção 2.4.2 foram apresentados os processos de estabelecimento do modelo de confiança. Um aspecto em aberto nessa área é a disponibilidade de estratégias robustas e seguras para inicialização do modelo de segurança. Soluções como uso de um certificado único pré-instalado nos nós e contato presencial utilizando canal fora da banda para estabelecimento da cadeia de confiança, podem implicar em baixa usabilidade ou efetividade. O uso de infraestruturas de chaves públicas distribuídas, teia de confiança (*web of trust*), identidade auto-soberana e outros mecanismos modernos e robustos de gestão de identidade [Pöhn and Hommel 2021] podem representar um caminho promissor para expandir o modelo de confiança e prover suporte à requisitos mais complexos de segurança.

Monitoramento de segurança e tratamento de incidentes

Questões operacionais do tratamento de incidentes de segurança, análise forense e monitoramento de segurança, podem também revelar-se em grandes desafios de pesquisa: como identificar e responsabilizar nós responsáveis por um ataque? Como viabilizar a construção de bases de conhecimento para inteligência de ameaças? Como coletar evidências para análise forense? Como realizar monitoramento de atividade maliciosa e contenção de nós envolvidos em ataques? Ou até mesmo como realizar monitoramento de disponibilidade e desempenho da rede? Estas são algumas perguntas operacionais, cujas respostas podem envolver o desenvolvimento de novas aplicações ou estratégias.

2.7.3. Aplicações distribuídas

Os protocolos de sincronização permitem uma comunicação assíncrona e multiparte centrada em dados por meio de mecanismos que suportam consistência fraca. Por outro lado, um conjunto de aplicações demanda consistência forte, em geral implementada na camada de aplicação, no topo dos protocolos existentes de sincronismo. Algumas aplicações que se baseiam no uso de mecanismos de invocação remota na NDN, como NFaaS [Król and Psaras 2017] e RICE [Król et al. 2018], visando funções *in-network*, deverão considerar mecanismos de tolerância a falhas. Uma das alternativas seria a adoção

de máquinas de estados replicadas, visando a manutenção de serviços *stateful*. Para isso, a implementação de mecanismos de consenso, acima dos protocolos de sincronização do conjunto de dados, apresenta um importante bloco para aplicações distribuídas na NDN.

O desenvolvimento de aplicações distribuídas com suporte à consistência forte, ou mesmo em cenários de consistência fraca, se torna mais desafiador quando se considera a possibilidade de atuação de falhas bizantinas. A concepção de serviços replicados tolerantes a falhas bizantinas deve considerar um conjunto de questões de implementação prática e um completo redesenho das soluções existentes na NDN. É possível pensar em consensos que considerem o modelo de sistema da NDN e utilizem seus próprios mecanismos e o NDN Sync como base para soluções de consenso a partir de abordagens clássicas, tais como Paxos [Lamport et al. 2001] e PBFT [Castro and Liskov 1999].

Uma tendência recente tem sido o uso da arquitetura NDN em cenários não permissionados de larga escala visando a composição de *distributed ledgers* [Zhang et al. 2019b, Doku et al. 2020]. Em tais cenários, em especial, os baseados no consenso não determinístico por meio de mecanismos de prova, busca-se o melhor custo-benefício entre escalabilidade e desempenho em função do contexto da aplicação subjacente. Uma possível solução seria o uso de mecanismos de sincronização de estados adequados às estruturas das *distributed ledgers* e mecanismos de reputação para mitigar nós maliciosos.

2.8. Considerações Finais

Este capítulo apresentou as principais características e desafios da arquitetura NDN em relação aos aspectos de mobilidade, segurança e aplicações distribuídas. A NDN é uma proposta relativamente nova que busca a construção de uma Internet da Informação (i.e., em que dados são o centro do processo) e, portanto, requer uma mudança conceitual na forma de pensar a rede e as aplicações que dela fazem uso. NDN, tendo sido proposta mais de três décadas após a pilha TCP/IP, tem suas bases nos requisitos desta Internet da Informação. Assim, a arquitetura é projetada para alcançar os requisitos contemporâneos, diferente da TCP/IP, que teve que se adaptar ao longo do tempo a estes novos requisitos. O capítulo explicou as propriedades fundamentais dessa arquitetura e realizou uma associação entre as características das estratégias e protocolos utilizados e o desenvolvimento de aplicações distribuídas, aplicação de segurança na rede e adequação à cenários móveis.

A NDN consegue atender alguns requisitos de mobilidade de forma nativa, mas ainda existem diversas questões em aberto e desafios de pesquisa relacionados com as características dos meios de transmissão, encaminhamento na camada de enlace, roteamento, comunicação *multi-hop* e recuperação de dados. Do ponto de vista de segurança, as mudanças que ocorreram na arquitetura geraram a necessidade de uma segurança mais focada na transmissão dos dados. Desta forma, tem-se como principais aspectos a serem observados nesse eixo temático, a definição de modelos de ameaça que exploram as propriedades da NDN, a identificação de modelos de confiança e de mecanismos de controle de acesso, autenticação e autorização. Além disso, é importante ressaltar a necessidade do desenvolvimento de estratégias de monitoramento de segurança e tratamento de incidentes, que são fundamentais na área de redes e que, muitas vezes, atendem a requisitos legislativos. As características da NDN possibilitam um modelo de comunicação temporalmente assíncrono com uso de protocolos de sincronização de mensagens. Um conjunto

de blocos de construção para sistemas distribuídos inerentes a NDN possibilita o uso de mecanismos sofisticados de *publish/subscribe* e de sincronização de dados, favorecendo a disponibilidade da computação distribuída, mesmo com alta volatibilidade de nós, ao custo de uma consistência fraca. Isto tem sido exercitado em aplicações contemporâneas, como, por exemplo, *distributed ledgers* em NDN. Um dos desafios em aberto é o de propiciar a construção de aplicações distribuídas com maior grau de resiliência e consistência.

Devido às propriedades promissoras da NDN ao atender nativamente os requisitos de rede e de aplicações atuais, tem-se observado esforços em busca da padronização da arquitetura[12]. Paralelo a isso, observa-se a importância que tem sido dada no desenvolvimento da arquitetura através do apoio da NSF (*National Science Foundation*) nos últimos anos e do envolvimento da comunidade científica. Além disso, importantes atores da indústria têm demonstrado interesse e desenvolvido soluções NDN, tais como a *Cisco Systems*, *Fujitsu Laboratories of America*, *Huawei Technologies*, *Intel Corporation*, *Juniper Networks*, *Panasonic Corporation* e *ViaSat*. Esperamos que este texto permita uma reflexão das possibilidades da arquitetura e da forma que a vemos: projetada para um cenário de alta mobilidade, com aplicações inerentemente distribuídas e em que a segurança da informação é o cerne.

Agradecimentos

Os autores agradecem o apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) e da Fundação de Amparo à Pesquisa do Estado da Bahia (FAPESB).

Referências

[Afanasyev et al. 2017] Afanasyev, A., Jiang, X., Yu, Y., Tan, J., Xia, Y., Mankin, A., and Zhang, L. (2017). Ndns: A dns-like name service for ndn. In *2017 26th International Conference on Computer Communication and Networks (ICCCN)*, pages 1–9. IEEE.

[Afanasyev et al. 2018] Afanasyev, A., Shi, J., Zhang, B., Zhang, L., Moiseenko, I., Yu, Y., Shang, W., Li, Y., Mastorakis, S., Huang, Y., Abraham, J. P., Newberry, E., DiBenedetto, S., Fan, C., Papadopoulos, C., Pesavento, D., Grassi, G., Pau, G., Zhang, H., Song, T., Yuan, H., Abraham, H. B., Crowley, P., Amin, S. O., Lehman, V., Chowdhury, M., and Wang, L. (2018). Nfd developer's guide. Technical Report NDN-0021, NDN.

[Amadeo et al. 2016] Amadeo, M., Campolo, C., Quevedo, J., Corujo, D., Molinaro, A., Iera, A., Aguiar, R. L., and Vasilakos, A. V. (2016). Information-centric networking for the internet of things: challenges and opportunities. *IEEE Network*, 30(2):92–100.

[Araújo 2018] Araújo, F. R. C. (2018). Cache colaborativo e distribuído como suporte à mobilidade de produtores em redes sem fio de dados nomeados. Dissertação (Mestrado em Ciência da Computação), Universidade Federal da Bahia, Instituto de Matemática, Programa de Pós-Graduação em Ciência da Computação, Salvador.

[12]Exemplos de padronização: `https://www.rfc-editor.org/rfc/rfc8793.html` e `https://datatracker.ietf.org/rg/icnrg/about/`

[Araújo et al. 2019] Araújo, F. R. C., de Sousa, A. M., and Sampaio, L. N. (2019). Scan-mob: An opportunistic caching strategy to support producer mobility in named data wireless networking. *Computer Networks*, 156:62–74.

[Araujo et al. 2021] Araujo, G. B., Peixoto, M. L. M., and Sampaio, L. N. (2021). Ndn4ivc: A framework for simulating and testing applications in vehicular named-data networking.

[Araujo and Sampaio 2021] Araujo, G. B. and Sampaio, L. N. (2021). An intelligent edge-traffic routing architecture for vehicular data-mule service. *IEEE Latin America Transactions*, 19(11):1976–1984.

[Araújo et al. 2018] Araújo, F. R. C., de Sousa, A. M., and Sampaio, L. N. (2018). Armazenamento oportunista em redes de dados nomeados sem fio como suporte à mobilidade de produtores. In *Anais do XXXVI Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*. SBC.

[Araújo et al. 2019] Araújo, F. R. C., de Sousa, A. M., and Sampaio, L. N. (2019). Uma estratégia de encaminhamento eficiente para redes de veículos aéreos não tripulados de dados nomeados. In *Anais do XXXVII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pages 876–889. SBC.

[Araújo and Sampaio 2017] Araújo, F. R. C. and Sampaio, L. N. (2017). Mobilidade em ndn: Consumidores versus produtores. In *Anais do XIII Workshop de Redes P2P, Dinâmicas, Sociais e Orientadas a Conteúdo*, pages 8–13. SBC.

[Asaf et al. 2020] Asaf, K., Rehman, R. A., and Kim, B.-S. (2020). Blockchain technology in named data networks: A detailed survey. *Journal of Network and Computer Applications*, 171:102840.

[Azgin et al. 2016] Azgin, A., Ravindran, R., and Wang, G. (2016). pit/less: Stateless forwarding in content centric networks. In *2016 IEEE Global Communications Conference (GLOBECOM)*, pages 1–7.

[Baccelli et al. 2014] Baccelli, E., Mehlis, C., Hahm, O., Schmidt, T. C., and Wählisch, M. (2014). Information centric networking in the iot: Experiments with ndn in the wild. In *Proceedings of the 1st ACM Conference on Information-Centric Networking*, ACM-ICN '14, page 77–86. ACM.

[Barka et al. 2018] Barka, E., Kerrache, C. A., Hussain, R., Lagraa, N., Lakas, A., and Bouk, S. H. (2018). A trusted lightweight communication strategy for flying named data networking. *Sensors*, 18(8).

[Brito et al. 2020] Brito, I. V. S., Sampaio, L., and Zhang, L. (2020). (Poster) Towards a distance vector routing protocol for named data networking. In *Named Data Networking Community Meeting (NDNcomm) 2020*.

[Castro and Liskov 1999] Castro, M. and Liskov, B. (1999). Practical byzantine fault tolerance. In *Proceedings of the Third Symposium on Operating Systems Design and Implementation*, OSDI '99, page 173–186. USENIX Association.

[Chandra and Toueg 1996] Chandra, T. D. and Toueg, S. (1996). Unreliable failure detectors for reliable distributed systems. *Journal of the ACM (JACM)*, 43(2):225–267.

[Chandy and Lamport 1985] Chandy, K. M. and Lamport, L. (1985). Distributed snapshots: Determining global states of distributed systems. *ACM Transactions on Computer Systems (TOCS)*, 3(1):63–75.

[Cisco 2020] Cisco (2020). Cisco annual internet report (2018-2023) white paper. Disponível em: `https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html`. Último acesso em: 30 de maio de 2021.

[Conti et al. 2020] Conti, M., Gangwal, A., Hassan, M., Lal, C., and Losiouk, E. (2020). The road ahead for networking: A survey on icn-ip coexistence solutions. *IEEE Communications Surveys Tutorials*, 22(3):2104–2129.

[Cristian and Schmuck 1995] Cristian, F. and Schmuck, F. (1995). Agreeing on processor group membership in timed asynchronous distributed systems. *Report CSE95-428, UC San Diego*.

[de Sousa et al. 2018a] de Sousa, A. M., Araújo, F. R. C., and Sampaio, L. N. (2018a). A link-stability-based interest-forwarding strategy for vehicular named data networks. *IEEE Internet Computing*, 22(3):16–26.

[de Sousa et al. 2019] de Sousa, A. M., Araújo, F. R. C., Greve, F. G. P., and Sampaio, L. N. (2019). Um arcabouço para validação de conteúdo em redes de dados nomeados baseado em blockchain. In *Anais do XXXVII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pages 155–168. SBC.

[de Sousa et al. 2018b] de Sousa, A. M., Araújo, F. R. C., and Sampaio, L. N. (2018b). Encaminhamento seletivo de interesses em redes veiculares de dados nomeados baseado no tempo de vida do enlace. In *Anais do XXXVI Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*. SBC.

[Doku et al. 2020] Doku, R., Rawat, D. B., Garuba, M., and Njilla, L. (2020). Fusion of named data networking and blockchain for resilient internet-of-battlefield-things. In *2020 IEEE 17th Annual Consumer Communications & Networking Conference (CCNC)*, pages 1–6.

[Dolev et al. 1987] Dolev, D., Dwork, C., and Stockmeyer, L. (1987). On the minimal synchronism needed for distributed consensus. *Journal of the ACM*, 34(1):77–97.

[Eugster et al. 2003] Eugster, P. T., Felber, P. A., Guerraoui, R., and Kermarrec, A.-M. (2003). The many faces of publish/subscribe. *ACM Computing Surveys*, 35(2):114–131.

[Fang et al. 2018] Fang, C., Yao, H., Wang, Z., Wu, W., Jin, X., and Yu, F. R. (2018). A survey of mobile information-centric networking: Research issues and challenges. *IEEE Communications Surveys Tutorials*, 20(3):2353–2371.

[Fisher et al. 1985] Fisher, M. J., Lynch, N., and Paterson, M. S. (1985). Impossibility of distributed consensus with one faulty process. *Journal of the ACM*, 32(2):374–382.

[Fotiou and Polyzos 2016] Fotiou, N. and Polyzos, G. C. (2016). Decentralized name-based security for content distribution using blockchains. In *Computer Communications Workshops (INFOCOM WKSHPS), 2016 IEEE Conference on*, pages 415–420. IEEE.

[Gawande et al. 2019] Gawande, A., Clark, J., Coomes, D., and Wang, L. (2019). Decentralized and secure multimedia sharing application over named data networking. In *Proceedings of the 6th ACM Conference on Information-Centric Networking*, ICN '19, page 19–29. ACM.

[Grassi et al. 2014] Grassi, G., Pesavento, D., Pau, G., Vuyyuru, R., Wakikawa, R., and Zhang, L. (2014). Vanet via named data networking. In *2014 IEEE conference on computer communications workshops (INFOCOM WKSHPS)*, pages 410–415. IEEE.

[Ioannou and Weber 2016] Ioannou, A. and Weber, S. (2016). A survey of caching policies and forwarding mechanisms in information-centric networking. *IEEE Communications Surveys Tutorials*, 18(4):2847–2886.

[Jacobson et al. 2009] Jacobson, V., Smetters, D. K., Thornton, J. D., Plass, M. F., Briggs, N. H., and Braynard, R. L. (2009). Networking named content. In *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies*, CoNEXT'09, pages 1–12. ACM.

[Jin et al. 2017] Jin, T., Zhang, X., Liu, Y., and Lei, K. (2017). Blockndn: A bitcoin blockchain decentralized system over named data networking. In *2017 Ninth International Conference on Ubiquitous and Future Networks (ICUFN)*, pages 75–80.

[Kapetanidou et al. 2020] Kapetanidou, I. A., Hassan, M., Sarros, C.-A., Conti, M., and Tsaoussidis, V. (2020). Reputation-based trust: A robust mechanism for dynamic adaptive streaming over named data networking. In *2020 IEEE 14th International Conference on Big Data Science and Engineering (BigDataSE)*, pages 114–121.

[Karrakchou et al. 2020a] Karrakchou, O., Samaan, N., and Karmouch, A. (2020a). Endn: An enhanced ndn architecture with a p4-programmabie data plane. In *Proceedings of the 7th ACM Conference on Information-Centric Networking*, ICN '20, page 1–11. ACM.

[Karrakchou et al. 2020b] Karrakchou, O., Samaan, N., and Karmouch, A. (2020b). Fc-trees: A front-coded family of compressed tree-based fib structures for ndn routers. *IEEE Transactions on Network and Service Management*, 17(2):1167–1180.

[Khelifi et al. 2020] Khelifi, H., Luo, S., Nour, B., Moungla, H., Faheem, Y., Hussain, R., and Ksentini, A. (2020). Named data networking in vehicular ad hoc networks: State-of-the-art and challenges. *IEEE Communications Surveys Tutorials*, 22(1):320–351.

[Kietzmann et al. 2017] Kietzmann, P., Gündoğan, C., Schmidt, T. C., Hahm, O., and Wählisch, M. (2017). The need for a name to mac address mapping in ndn: Towards

quantifying the resource gain. In *Proceedings of the 4th ACM Conference on Information-Centric Networking*, ICN '17, page 36–42. ACM.

[Król et al. 2018] Król, M., Habak, K., Oran, D., Kutscher, D., and Psaras, I. (2018). Rice: Remote method invocation in icn. In *Proceedings of the 5th ACM Conference on Information-Centric Networking*, ICN '18, page 1–11. ACM.

[Król and Psaras 2017] Król, M. and Psaras, I. (2017). Nfaas: Named function as a service. In *Proceedings of the 4th ACM Conference on Information-Centric Networking*, ICN '17, page 134–144. ACM.

[Lamport 2019] Lamport, L. (2019). Time, clocks, and the ordering of events in a distributed system. In *Concurrency: The Works of Leslie Lamport*, page 179–196. ACM.

[Lamport et al. 2001] Lamport, L. et al. (2001). Paxos made simple. *ACM Sigact News*, 32(4):18–25.

[Lamport and Fischer 1982] Lamport, L. and Fischer, M. (1982). Byzantine generals and transaction commit protocols. Technical Report 62, SRI International.

[Lee et al. 2018] Lee, C. A., Zhang, Z., Tu, Y., Afanasyev, A., and Zhang, L. (2018). Supporting virtual organizations using attribute-based encryption in named data networking. In *2018 IEEE 4th International Conference on Collaboration and Internet Computing (CIC)*, pages 188–196. IEEE.

[Li et al. 2019a] Li, T., Kong, Z., Mastorakis, S., and Zhang, L. (2019a). Distributed dataset synchronization in disruptive networks. In *2019 IEEE 16th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, pages 428–437. IEEE.

[Li et al. 2018] Li, T., Shang, W., Afanasyev, A., Wang, L., and Zhang, L. (2018). A brief introduction to ndn dataset synchronization (ndn sync). In *IEEE Military Communications Conference (MILCOM)*, pages 612–618.

[Li et al. 2019b] Li, Y., Zhang, Z., Wang, X., Lu, E., Zhang, D., and Zhang, L. (2019b). A secure sign-on protocol for smart homes over named data networking. *IEEE Communications Magazine*, 57(7):62–68.

[Li et al. 2019c] Li, Z., Xu, Y., Zhang, B., Yan, L., and Liu, K. (2019c). Packet forwarding in named data networking requirements and survey of solutions. *IEEE Communications Surveys Tutorials*, 21(2):1950–1987.

[Liu et al. 2019] Liu, G., Quan, W., Cheng, N., Zhang, H., and Yu, S. (2019). Efficient ddos attacks mitigation for stateful forwarding in internet of things. *Journal of Network and Computer Applications*, 130:1–13.

[Lynch 1996] Lynch, N. (1996). *Distributed Algorithms*. Morgan Kaufmann Publishers, Inc., San Francisco, CA, USA.

[Madureira et al. 2021] Madureira, A. L. R., Araújo, F. R. C., Araújo, G. B., and Sampaio, L. N. (2021). Ndn fabric: Where the software-defined networking meets the content-centric model. *IEEE Transactions on Network and Service Management*, 18(1):374–387.

[Madureira et al. 2020] Madureira, A. L. R., Araújo, F. R. C., Prates, L. N. B., and Sampaio, L. N. (2020). Ndn-adap: Uma arquitetura para encaminhamento eficiente de pacotes em redes de dados nomeados. In *Anais do XXXVIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pages 812–825. SBC.

[Mannes and Maziero 2019] Mannes, E. and Maziero, C. (2019). Naming content on the network layer: A security analysis of the information-centric network model. *ACM Computing Surveys*, 52(3).

[Marxer and Tschudin 2017] Marxer, C. and Tschudin, C. (2017). Schematized access control for data cubes and trees. In *Proceedings of the 4th ACM Conference on Information-Centric Networking*, ICN '17, page 170–175. ACM.

[Mastorakis et al. 2017] Mastorakis, S., Afanasyev, A., and Zhang, L. (2017). On the evolution of ndnsim: An open-source simulator for ndn experimentation. *ACM SIGCOMM Computer Communication Review*, 47(3):19–33.

[Mastorakis et al. 2018] Mastorakis, S., Gusev, P., Afanasyev, A., and Zhang, L. (2018). Real-time data retrieval in named data networking. In *2018 1st IEEE International Conference on Hot Information-Centric Networking (HotICN)*, pages 61–66. IEEE.

[Miguel et al. 2018] Miguel, R., Signorello, S., and Ramos, F. M. V. (2018). Named data networking with programmable switches. In *2018 IEEE 26th International Conference on Network Protocols (ICNP)*, pages 400–405.

[Moiseenko and Oran 2017] Moiseenko, I. and Oran, D. (2017). Path switching in content centric and named data networks. In *Proceedings of the 4th ACM Conference on Information-Centric Networking*, ICN '17, pages 66–76. ACM.

[Mori 2018] Mori, S. (2018). Secure caching scheme by using blockchain for information-centric network-based wireless sensor networks. *Journal of Signal Processing*, 22(3):97–108.

[NDN 2021] NDN (2021). Ndn testbed status. Disponível em: `http://ndndemo.arl.wustl.edu/ndn.html`. Último acesso em: 07 de maio de 2021.

[Nour et al. 2021a] Nour, B., Khelifi, H., Hussain, R., Mastorakis, S., and Moungla, H. (2021a). Access control mechanisms in named data networks: A comprehensive survey. *ACM Computing Surveys (CSUR)*, 54(3):1–35.

[Nour et al. 2019a] Nour, B., Li, F., Khelifi, H., Moungla, H., and Ksentini, A. (2019a). Coexistence of icn and ip networks: An nfv as a service approach. In *2019 IEEE Global Communications Conference (GLOBECOM)*, page 1–6. IEEE Press.

[Nour et al. 2021b] Nour, B., Mastorakis, S., Ullah, R., and Stergiou, N. (2021b). Information-centric networking in wireless environments: Security risks and challenges. *IEEE Wireless Communications*, 28(2):121–127.

[Nour et al. 2019b] Nour, B., Sharif, K., Li, F., Biswas, S., Moungla, H., Guizani, M., and Wang, Y. (2019b). A survey of internet of things communication using icn: A use case perspective. *Computer Communications*, 142-143:95–123.

[Nour et al. 2019c] Nour, B., Sharif, K., Li, F., Yang, S., Moungla, H., and Wang, Y. (2019c). Icn publisher-subscriber models: Challenges and group-based communication. *IEEE Network*, 33(6):156–163.

[Pires et al. 2019] Pires, S., Araújo, F. R. C., Freitas, A. E. S., and Sampaio, L. N. (2019). Análise de perfis de usuários de música e seus impactos no desempenho de políticas de substituição de cache. In *Anais do XXXVII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pages 848–861. SBC.

[Pires et al. 2021] Pires, S., Ziviani, A., and Sampaio, L. (2021). Contextual dimensions for cache replacement schemes in information-centric networks: a systematic review. *PeerJ Computer Science*, 7:e418.

[Pires et al. 2018] Pires, S. S., Ribeiro, A. V., de Sousa, A. M., Freitas, A. E. S., and Sampaio, L. N. (2018). On evaluating the influence of user's music listening habits on cache replacement policies. In *2018 IEEE Symposium on Computers and Communications (ISCC)*, pages 00930–00933.

[Pöhn and Hommel 2021] Pöhn, D. and Hommel, W. (2021). Proven and modern approaches to identity management. In *Advances in Cybersecurity Management*, pages 421–443. Springer.

[Psaras et al. 2018] Psaras, I., Ascigil, O., Rene, S., Pavlou, G., Afanasyev, A., and Zhang, L. (2018). Mobile data repositories at the edge. In *USENIX Workshop on Hot Topics in Edge Computing (HotEdge 18)*. USENIX Association.

[Ramani and Afanasyev 2020a] Ramani, S. K. and Afanasyev, A. (2020a). Certcoalesce: Efficient certificate pool for ndn-based systems. In *Proceedings of the 7th ACM Conference on Information-Centric Networking*, ICN '20, page 158–160. ACM.

[Ramani and Afanasyev 2020b] Ramani, S. K. and Afanasyev, A. (2020b). Rapid establishment of transient trust for ndn-based vehicular networks. In *2020 IEEE International Conference on Communications Workshops (ICC Workshops)*, pages 1–6. IEEE.

[Ramani et al. 2019] Ramani, S. K., Tourani, R., Torres, G., Misra, S., and Afanasyev, A. (2019). Ndn-abs: Attribute-based signature scheme for named data networking. In *Proceedings of the 6th ACM Conference on Information-Centric Networking*, ICN '19, page 123–133. ACM.

[Ribeiro et al. 2017] Ribeiro, A. V., Sampaio, L. N., and Ziviani, A. (2017). Explorando a afinidade de usuários para descarregamento de dados mais eficiente em redes celulares de pequeno porte. In *Anais do XXXV Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*. SBC.

[Ribeiro et al. 2018] Ribeiro, A. V., Sampaio, L. N., and Ziviani, A. (2018). Affinity-based user clustering for efficient edge caching in content-centric cellular networks. In *2018 IEEE Symposium on Computers and Communications (ISCC)*, pages 00474–00479.

[Rondon et al. 2020] Rondon, L. B., da Costa, J. B., Filho, G. P. R., Rosário, D., and Villas, L. A. (2020). Degree centrality-based caching discovery protocol for vehicular named-data networks. In *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*, pages 1–5.

[Saxena et al. 2016] Saxena, D., Raychoudhury, V., Suri, N., Becker, C., and Cao, J. (2016). Named data networking: a survey. *Computer Science Review*, 19:15–55.

[Sedky and Mougy 2018] Sedky, G. and Mougy, A. E. (2018). Bcxp: Blockchain-centric network layer for efficient transaction and block exchange over named data networking. In *2018 IEEE 43rd Conference on Local Computer Networks (LCN)*, pages 449–452.

[Seixas et al. 2017] Seixas, N. F. S., Ribeiro, A. V., and Sampaio, L. N. (2017). Um modelo de rede centrada na informação resiliente a ataques de negação de serviços por inundação de interesses. In *Anais do XXXV Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*. SBC.

[Serhane et al. 2021] Serhane, O., Yahyaoui, K., Nour, B., and Moungla, H. (2021). A survey of icn content naming and in-network caching in 5g and beyond networks. *IEEE Internet of Things Journal*, 8(6):4081–4104.

[Shang et al. 2017] Shang, W., Yu, Y., Wang, L., Afanasyev, A., and Zhang, L. (2017). A survey of distributed dataset synchronization in named data networking. Technical Report NDN-0053, NDN.

[Shannigrahi et al. 2017] Shannigrahi, S., Fan, C., and Papadopoulos, C. (2017). Request aggregation, caching, and forwarding strategies for improving large climate data distribution with ndn: A case study. In *Proceedings of the 4th ACM Conference on Information-Centric Networking*, ICN '17, page 54–65. ACM.

[Shi et al. 2016] Shi, J., Liang, T., Wu, H., Liu, B., and Zhang, B. (2016). Ndn-nic: Name-based filtering on network interface card. In *Proceedings of the 3rd ACM Conference on Information-Centric Networking*, ACM-ICN '16, pages 40–49. ACM.

[Shi et al. 2020] Shi, J., Pesavento, D., and Benmohamed, L. (2020). Ndn-dpdk: Ndn forwarding at 100 gbps on commodity hardware. In *Proceedings of the 7th ACM Conference on Information-Centric Networking*, ICN '20, page 30–40. ACM.

[Shi and Zhang 2012] Shi, J. and Zhang, B. (2012). Ndnlp: A link protocol for ndn. Technical Report NDN-0006, NDN.

[Signorello et al. 2016] Signorello, S., State, R., François, J., and Festor, O. (2016). Ndn.p4: Programming information-centric data-planes. In *2016 IEEE NetSoft Conference and Workshops (NetSoft)*, pages 384–389.

[Simon 2000] Simon, S. (2000). Brewer's cap theorem. *CS341 Distributed Information Systems, University of Basel (HS2012)*.

[Siracusano et al. 2018] Siracusano, G., Salsano, S., Ventre, P., Detti, A., Rashed, O., and Blefari-Melazzi, N. (2018). A framework for experimenting icn over sdn solutions using physical and virtual testbeds. *Computer Networks*, 134:245–259.

[Stallings 2020] Stallings, W. (2020). *Cryptography and Network Security: Principles and Practice*. Pearson, USA, 8th edition.

[Stoller 1997] Stoller, S. D. (1997). Leader election in distributed systems with crash failures. Technical Report 481, Computer Science Dept., Indiana University. Revised July 1997.

[Tariq et al. 2020] Tariq, A., Rehman, R. A., and Kim, B. (2020). Forwarding strategies in ndn-based wireless networks: A survey. *IEEE Communications Surveys Tutorials*, 22(1):68–95.

[Tehrani et al. 2019] Tehrani, P. F., Keidel, L., Osterweil, E., Schiller, J. H., Schmidt, T. C., and Wählisch, M. (2019). Ndnssec: Namespace management in ndn with dnssec. In *Proceedings of the 6th ACM Conference on Information-Centric Networking*, ICN '19, page 171–172. ACM.

[Thompson et al. 2019] Thompson, J., Gusev, P., and Burke, J. (2019). Ndn-cnl: A hierarchical namespace api for named data networking. In *Proceedings of the 6th ACM Conference on Information-Centric Networking*, ICN '19, page 30–36. ACM.

[Tizvar and Abbaspour 2020] Tizvar, R. and Abbaspour, M. (2020). A density-aware probabilistic interest forwarding method for content-centric vehicular networks. *Vehicular Communications*, 23:100216.

[Waldo et al. 1996] Waldo, J., Wyant, G., Wollrath, A., and Kendall, S. (1996). A note on distributed computing. In *International Workshop on Mobile Object Systems*, pages 49–64. Springer.

[Wang et al. 2020] Wang, J., Luo, J., Zhou, J., and Ran, Y. (2020). A mobility-predict-based forwarding strategy in vehicular named data networks. In *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, pages 01–06.

[Wang et al. 2018] Wang, L., Lehman, V., Hoque, A. M., Zhang, B., Yu, Y., and Zhang, L. (2018). A secure link state routing protocol for ndn. *IEEE Access*, 6:10470–10482.

[Wang and Li 2020] Wang, X. and Li, Y. (2020). Content delivery based on vehicular cloud. *IEEE Transactions on Vehicular Technology*, 69(2):2105–2113.

[Wang et al. 2021] Wang, X., Wang, X., and Wang, D. (2021). Cost-efficient data retrieval based on integration of vc and ndn. *IEEE Transactions on Vehicular Technology*, 70(1):967–976.

[Yu et al. 2021] Yu, T., Zhang, Z., Ma, X., Moll, P., and Zhang, L. (2021). A pub/sub api for ndn-lite with built-in security. Technical Report NDN-0071, NDN.

[Yu et al. 2015] Yu, Y., Afanasyev, A., Clark, D., claffy, k., Jacobson, V., and Zhang, L. (2015). Schematizing trust in named data networking. In *Proceedings of the 2nd ACM Conference on Information-Centric Networking*, ACM-ICN '15, page 177–186. ACM.

[Yu et al. 2017] Yu, Y., Afanasyev, A., Seedorf, J., Zhang, Z., and Zhang, L. (2017). Ndn delorean: An authentication system for data archives in named data networking. In *Proceedings of the 4th ACM Conference on Information-Centric Networking*, ICN '17, page 11–21. ACM.

[Zhang et al. 2018a] Zhang, H., Li, Y., Zhang, Z., Afanasyev, A., and Zhang, L. (2018a). Ndn host model. *ACM SIGCOMM Computer Communication Review*, 48(3):35–41.

[Zhang 2019] Zhang, L. (2019). The role of data repositories in named data networking. In *2019 IEEE International Conference on Communications Workshops (ICC Workshops)*, pages 1–5.

[Zhang et al. 2014] Zhang, L., Afanasyev, A., Burke, J., Jacobson, V., Claffy, K., Crowley, P., Papadopoulos, C., Wang, L., and Zhang, B. (2014). Named data networking. *SIGCOMM Comput. Commun. Rev.*, 44(3):66–73.

[Zhang et al. 2010] Zhang, L., Estrin, D., Burke, J., Jacobson, V., Thornton, J. D., Smetters, D. K., Zhang, B., Tsudik, G., Claffy, K., Krioukov, D., Massey, D., Papadopoulos, C., Abdelzaher, T., Wang, L., Crowley, P., and Yeh, E. (2010). Named data networking (ndn) project. Technical Report NDN-0001, NDN.

[Zhang et al. 2016] Zhang, Y., Afanasyev, A., Burke, J., and Zhang, L. (2016). A survey of mobility support in named data networking. In *2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 83–88.

[Zhang et al. 2019a] Zhang, Y., Xia, Z., Afanasyev, A., and Zhang, L. (2019a). A note on routing scalability in named data networking. In *2019 IEEE International Conference on Communications Workshops (ICC Workshops)*, pages 1–6.

[Zhang et al. 2018b] Zhang, Y., Xia, Z., Mastorakis, S., and Zhang, L. (2018b). Kite: Producer mobility support in named data networking. In *Proceedings of the 5th ACM Conference on Information-Centric Networking*, ICN '18, pages 125–136. ACM.

[Zhang et al. 2021] Zhang, Z., Liu, S., King, R., and Zhang, L. (2021). Supporting multiparty signing over named data networking.

[Zhang et al. 2019b] Zhang, Z., Vasavada, V., Ma, X., and Zhang, L. (2019b). Dledger: An iot-friendly private distributed ledger system based on dag.

[Zhang et al. 2018c] Zhang, Z., Yu, Y., Ramani, S. K., Afanasyev, A., and Zhang, L. (2018c). Nac: Automating access control via named data. In *MILCOM 2018-2018 IEEE Military Communications Conference (MILCOM)*, pages 626–633.

[Zhang et al. 2018d] Zhang, Z., Yu, Y., Zhang, H., Newberry, E., Mastorakis, S., Li, Y., Afanasyev, A., and Zhang, L. (2018d). An overview of security support in named data networking. *IEEE Communications Magazine*, 56(11):62–68.

Capítulo 3

Virtualização de Funções de Rede na IoT: Um Panorama do Gerenciamento de Desempenho x Segurança

Guilherme Werneck de Oliveira, Jonathan Rangel Porto, Nelson Gonçalves Prates Jr., Aldri Luiz dos Santos, Michele Nogueira, Daniel Macêdo Batista

Abstract

This chapter presents open issues and the state-of-the-art related to the use of Network Function Virtualization (NFV) in the detection and mitigation of security threats in the Internet of Things (IoT). The main characteristics of NFV, the project, architectures, and protocols of networks to connect IoT devices are also discussed as basis to understand the main concepts. It focuses also on the NFV management specifications and how they affect performance and security in IoT. Next, the chapter presents the issues related to the performance of NFV applied to security in IoT. Then, it highlights a case study on the MCTIC/FAPESP MENTORED Testbed, an experimental environment being deployed for the community that provides realistic scenarios for simulating specific vulnerabilities and for evaluating network security mechanisms in a controlled manner. Finally, the chapter discusses the main open challenges in order to increase the interest of readers to conduct research on topics that bring together virtualization, IoT and network security.

Resumo

Este capítulo apresenta questões e o estado da arte relacionados ao uso da Virtualização de Funções de Rede (NFV – Network Function Virtualization) na detecção e mitigação de ameaças de segurança na Internet das Coisas (IoT – Internet of Things). Ele aborda as características relacionadas à NFV, assim como o projeto, as arquiteturas e os protocolos de redes que suportam a conexão dos dispositivos IoT. Consideram-se também as especificações de gerenciamento de NFV e como elas afetam os requisitos de desempenho e de segurança na IoT. Na sequência, são apresentadas as questões relacionadas ao desempenho de NFV quando esta é aplicada à garantia de segurança em IoT. Então, detalha-se um estudo de caso implementado no ambiente experimental MCTIC/FAPESP MENTORED, um ambiente em implantação que será aberto à comunidade e oferecerá cenários realísticos para simulação de vulnerabilidades específicas e avaliação de mecanismos de segurança de redes de forma controlada. Por fim, discutem-se os principais desafios em aberto nas pesquisas que unam virtualização, IoT e segurança na IoT.

3.1. Introdução

A Internet das Coisas (IoT – *Internet of Things*) tem passado por uma rápida popularização, alcançando uma grande diversidade de domínios de aplicações, como por exemplo cuidados da saúde, monitoramento ambiental, automação residencial, mobilidade inteligente e Indústria 4.0 [da Silva et al. 2021, Batista et al. 2016, Rosário et al. 2014]. Como consequência, cada vez mais dispositivos IoT[1] com características diversas são implantados em uma variedade de ambientes públicos e privados, tornando-se progressivamente objetos comuns da vida cotidiana. A IoT tem sido vista por muitos como o próximo estágio da Internet. Sua implementação possibilita a evolução na sociedade e indústria, como no caso das cidades inteligentes, pois com a IoT os sistemas de transporte, de controle de resíduos, de energia, entre outros, tornam-se mais eficientes e melhoram a qualidade de vida dos cidadãos. Por outro lado, a infraestrutura física de sistemas heterogêneos é complexa e exige soluções eficientes e dinâmicas para o gerenciamento e a configuração das redes num nível que permita a implantação padronizada e de fácil replicação em casas, prédios e cidades inteligentes [Alam et al. 2020, Chi et al. 2019].

A Figura 3.1 ilustra uma visão geral da IoT. Os dispositivos IoT, na camada mais baixa da figura, interagem com o ambiente físico nos papéis de sensores e atuadores. Esses dispositivos são necessários para o correto funcionamento das aplicações IoT, na camada mais alta da figura. As comunicações entre os vários componentes do cenário acontecem graças à infraestrutura de tecnologia da informação disponível na Internet por meio de recursos na borda e no núcleo da rede. Como ocorre em redes de telecomunicações de um modo geral, dois dos requisitos importantes em IoT são desempenho e segurança. No entanto, muitos dispositivos na IoT não oferecem suporte a mecanismos de segurança fortes e, portanto, podem ser alvos e, até mesmo, meios para uma série de ataques [Meneghello et al. 2019]. De acordo com a empresa Kaspersky [Kaspersky 2020, Kaspersky 2019], os ataques a dispositivos IoT têm se tornado frequentes. Nos primeiros seis meses de 2019, foram contabilizados 105 milhões de ataques a dispositivos IoT oriundos de 276 mil endereços IP exclusivos. Esses ataques são dos mais variados tipos e se aproveitam de diversas vulnerabilidades, como a descoberta em agosto de 2020 pela empresa Check Point [CheckPoint 2020] contra a assistente pessoal Alexa, que permitia a manipulação de seus *tokens* e a realização de ações em nome da vítima. Outro dado importante é que 28% das empresas que utilizam plataformas IoT reportam ter encontrado incidentes envolvendo dispositivos conectados. Com o grande volume de dados gerados por sensores e dispositivos inteligentes, inclusive dados sigilosos, as consequências de tais incidentes podem ser graves.

Uma abordagem que vem ganhando espaço quando o intuito é responder às ameaças na IoT consiste no uso de recursos virtualizados de rede por meio da Virtualização de Funções de Rede (NFV – *Network Functions Virtualization*) [Alam et al. 2020, Farris et al. 2019, Zarca et al. 2018]. A NFV apresenta um novo grau de flexibilidade e escalabilidade criando recursos de rede virtuais sob demanda, como *firewalls*, sistemas de detecção de intrusão (IDS – *Intrusion Detection Systems*) e sistemas de inspeção profunda

[1]No decorrer deste capítulo, o termo "dispositivos IoT" será usado para descrever dispositivos de hardware voltados para aplicações de Internet das Coisas, como assistentes pessoais, relógios inteligentes, sensores e atuadores de um modo geral.

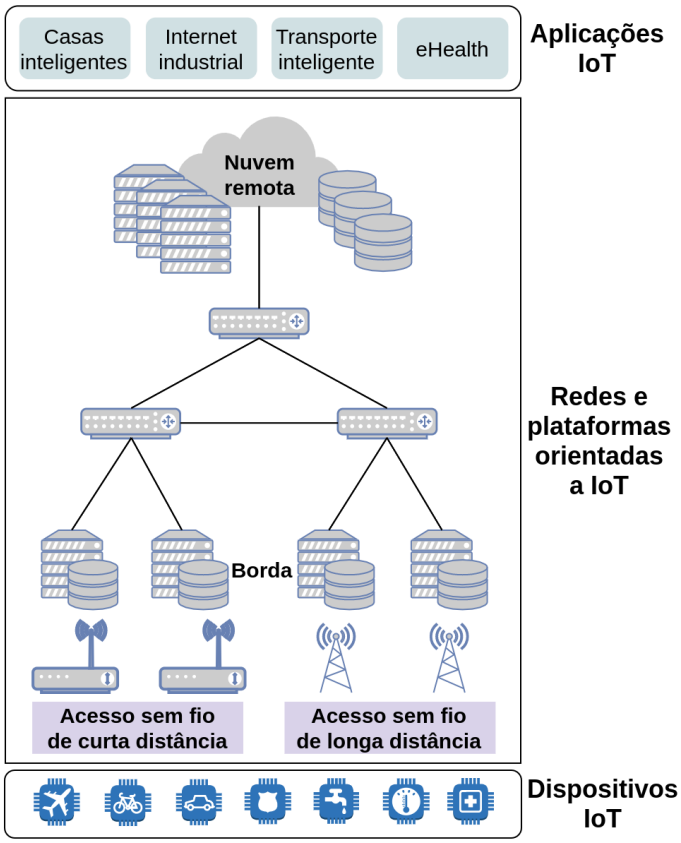


Figura 3.1. Uma visão geral da IoT [Farris et al. 2019].

de pacotes (DPI – *Deep Packet Inspection*). A abordagem de virtualização permite que várias instâncias de um mecanismo de detecção ou mitigação específico possam ser implementadas em diferentes locais na rede, considerando as restrições impostas pela ocorrência de eventos maliciosos [Farris et al. 2019] e pelo poder limitado de processamento e armazenamento de muitos dispositivos IoT.

Além da localização das funções virtualizadas, limitada pela capacidade dos dispositivos IoT e pelas particularidades dos ataques, há outras questões relacionadas ao uso de NFV. Uma delas diz respeito ao gerenciamento do desempenho dos recursos virtualizados. Para [Laghrissi and Taleb 2019], nesse cenário, o verdadeiro desafio é produzir *software* eficiente e escalável para orquestrar as redes virtuais, a fim de permitir que essas redes sejam facilmente configuradas e tenham seus ciclos de vida gerenciados. Em [Mijumbi et al. 2016], além da gestão e da orquestração de recursos, os autores também consideram a eficiência energética, a própria eficiência do recurso provisionado, a modelagem e a alocação de recursos como desafios em aberto. Tudo isso se relaciona com o gerenciamento de desempenho em ambientes virtualizados, tópico que tem levantado diversos desafios de pesquisa, como orquestração de funções de rede virtualizadas (VNF – *Virtual Network Function*), otimização da cadeia de funções (SFC – *Service Function Chaining*), posicionamento de VNFs em localidades distintas, entre outros [Zarca et al. 2020a, Gupta et al. 2019, Zhang et al. 2016].

Este capítulo apresenta um apanhado do estado da arte relacionado ao desempenho de NFV na detecção e mitigação de ameaças na IoT, enfatizando as questões de pesquisa em aberto. O foco na utilização da tecnologia NFV busca não só considerar aspectos ligados à redução de custos, devido à sua utilização sob demanda, mas também à redução de forma eficaz dos riscos e dos prejuízos causados por ataques na IoT. Além disso, por se tratar de um tópico recente, espera-se que sua divulgação por meio do capítulo tenha um papel importante no avanço do estado da arte. O capítulo apresenta características relacionadas à NFV, bem como à construção, arquiteturas e protocolos de redes que possuam dispositivos IoT. O texto também considera as especificações de gerenciamento de NFV, assim como requisitos de desempenho e segurança na IoT. Em seguida, são apresentadas questões de pesquisa relacionadas ao desempenho de NFV quando esta é aplicada à detecção e à mitigação de ameaças na IoT. Na sequência, será apresentado um estudo de caso sobre o MENTORED *Testbed*, ferramenta aberta à comunidade que disponibiliza cenários realísticos para simulação de vulnerabilidades específicas e avaliação de mecanismos de segurança de redes de forma controlada. Também são apresentadas uma análise e uma discussão sobre os principais desafios em aberto na área.

Este capítulo está organizado da seguinte forma: a Seção 3.2 apresenta a definição e as características da tecnologia NFV, descreve aspectos relacionados à camada física nos dispositivos IoT, redes e suas aplicações, e detalha as características da camada de orquestração de NFV; ainda, a Seção 3.2 relaciona as questões de desempenho de NFV às camadas física, virtualização, serviços e orquestração, aplicadas à detecção e à mitigação de ameaças na IoT; a Seção 3.3 apresenta a revisão da literatura sobre desempenho da NFV quando empregada para a detecção e mitigação de ameaças na IoT; a Seção 3.4 descreve um estudo de caso preparado sobre o MENTORED *Testbed*, ambiente de experimentação aberto à comunidade que disponibiliza cenários realísticos para simulação de vulnerabilidades específicas e para avaliação de mecanismos de segurança de redes de forma controlada. Por fim, o capítulo traz uma análise e uma discussão na Seção 3.5 sobre os principais desafios em aberto na área.

3.2. Conceitos Básicos

A Figura 3.2 ilustra um cenário de virtualização de funções de rede na IoT. Os dispositivos reais, na camada **física**, utilizam os serviços das VNFs disponíveis na camada de **serviços**. As VNFs são instanciadas graças à infraestrutura NFV, na camada de **virtualização**, e são gerenciadas pelo gerenciador de VNF e pelo gerenciador de infraestrutura virtualizada na camada de **orquestração**. Em um cenário ideal, algumas das VNFs fornecem serviços de segurança e nenhuma das VNFs possui vulnerabilidades. O foco deste capítulo é na utilização de VNFs que implementem mecanismos de segurança da rede, como *firewall*, IDS e DPI. Às leitoras e aos leitores com interesse no tópico de detecção de vulnerabilidades nas implementações de VNFs, recomenda-se a leitura de [De Benedictis and Lioy 2019, Marku et al. 2020].

As subseções seguintes apresentam os conceitos básicos para que os componentes apresentados na Figura 3.2 sejam compreendidos: a Subseção 3.2.1 apresenta a definição e as características de NFVs, relacionando-as com as camadas de virtualização, serviços e orquestração. A Subseção 3.2.2 define a IoT e apresenta aplicações nesse contexto, relacionando com a camada física. A Subseção 3.2.3 descreve conceitos presentes no ge-

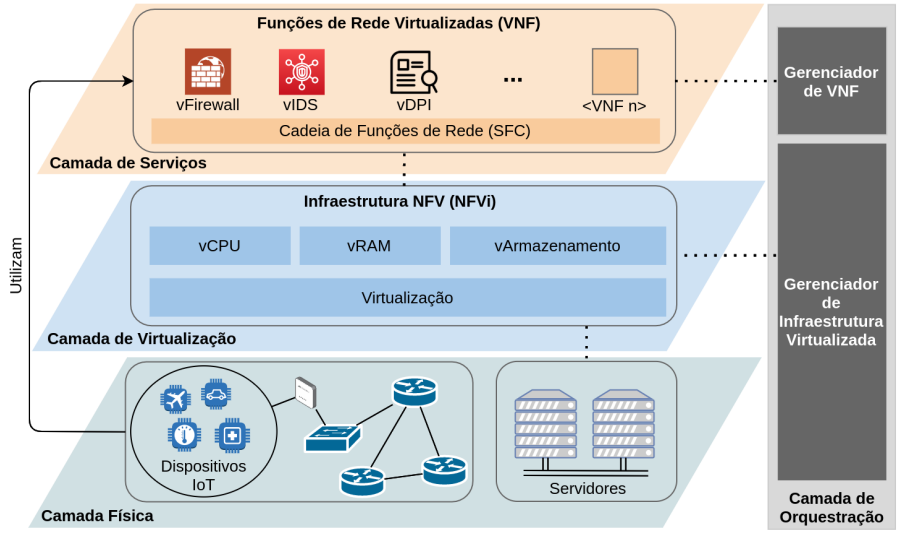


Figura 3.2. Visão geral da virtualização de funções de rede na IoT.

renciamento de redes que utilizam NFV, relacionando-os com a camada de orquestração; e a Subseção 3.2.4 apresenta conceitos que ajudam a compreender o *tradeoff* entre desempenho e segurança quando utiliza-se NFV para detecção e mitigação de ameaças de segurança na IoT, relacionando as camadas **física**, de **virtualização**, de **serviços** e de **orquestração**.

3.2.1. Virtualização de Funções de Rede (NFV)

As funções de rede como filtragem de pacotes, tradução de endereços e balanceamento de carga são essenciais em muitas redes de computadores. Mesmo em uma rede residencial de pequeno porte, em que os usuários locais agem como clientes, acessando servidores externos localizados na Internet, os modems fornecidos pelo provedor de Internet costumam já vir com algumas destas funções embutidas de fábrica. Em uma rede de grande porte, como as existentes em *campi* universitários, as funções de rede costumam ser implementadas em hardware dedicado e são configuradas de modo a formar um encadeamento de funções de rede ou cadeias de funções (SFCs). Um exemplo de SFC aplicada a um fluxo de pacotes entrando em uma rede para acesso a um serviço web está ilustrado pelos dispositivos interconectados da Figura 3.3. As funções de rede realizadas pelos dispositivos na figura são:

1. **Filtragem de pacotes:** para verificar se as características do fluxo (endereço IP de origem, endereço IP de destino, porta origem, protocolos) coincidem com alguma regra que represente um potencial ataque. Caso represente, os pacotes são descartados. Caso não represente, eles seguem para a próxima função de rede;

2. **Tradução de endereços:** para repassar os pacotes da requisição para uma rede interna com endereços IP não roteáveis na Internet, onde o serviço *web* será fornecido por um conjunto de servidores;

3. **Balanceamento de carga:** para distribuir as requisições entre o conjunto de servidores *web*, reduzindo a chance de sucesso de um ataque de negação de serviço.

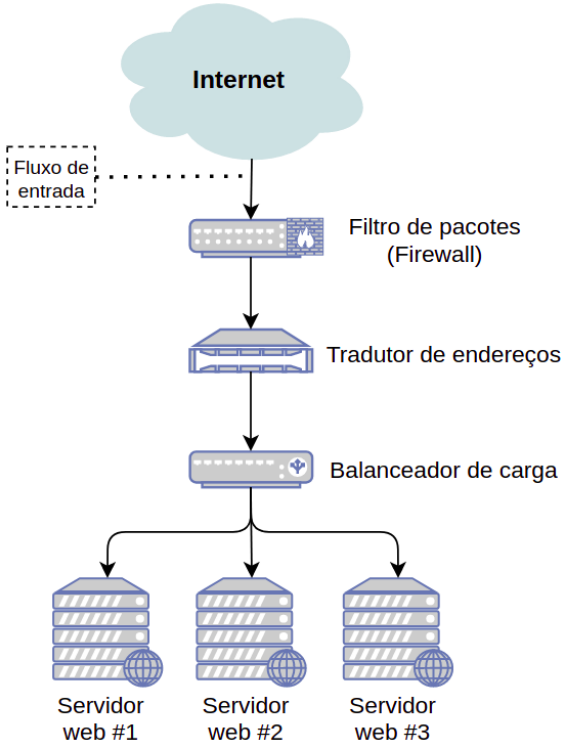


Figura 3.3. Exemplo de um encadeamento de funções em uma rede tradicional.

Uma rede como a ilustrada na Figura 3.3, em que as funções de rede são fornecidas por equipamentos de hardware dedicados conhecidos como *appliances* ou *middleboxes*, será chamada no decorrer deste capítulo de uma **rede tradicional**. Nessa rede, se o operador tiver que adicionar uma nova função, como um sistema de detecção de intrusão (IDS), por exemplo, depois do *firewall* (filtro de pacotes), ele precisará adquirir o *appliance* específico e fazer sua instalação física e configuração na rede local, considerando possíveis mudanças em rotas nos *appliances* existentes.

A NFV permite a criação de uma arquitetura de redes que flexibiliza o processo de aquisição e reconfiguração de funções de rede por meio da virtualização destas funções. Diferente de uma rede tradicional, em uma rede baseada em NFV, as VNFs são fornecidas como *software* e instanciadas em hardware de propósito geral. Neste caso, o operador poderia contratar um serviço de IDS, que estaria implementado completamente em *software* e instanciar o mesmo em um servidor localizado no *data center* de um provedor de nuvem ou em computadores na névoa ou na borda. Possíveis mudanças nas configurações de *appliances* existentes poderiam ser necessárias, assim como no caso da rede tradicional, mas todas as ações relacionadas à aquisição do equipamento específico e sua instalação física na rede local passam a ser desnecessárias. Neste caso da rede baseada em NFV, caso o operador resolva passar a utilizar outra implementação de IDS, bastaria alterar o *software* instanciado na nuvem/névoa/borda, sem necessidade de aquisição de um novo hardware. Além disso, em momentos de pico, em que o IDS precisasse realizar

tarefas intensivas em memória e CPU, bastaria solicitar ao provedor que ampliasse esses recursos ou migrasse a VNF para um recurso com maior capacidade. Essas facilidades nas mudanças da implementação das VNFs e da capacidade das mesmas tornam a NFV ideal para ambientes heterogêneos, com constantes mudanças e com recursos limitados, como é o caso da IoT [Alam et al. 2020].

A Figura 3.4 ilustra uma possível implementação da SFC da Figura 3.3 em uma arquitetura de NFV. Neste caso, o fluxo de pacotes alcança a primeira função de rede, o *firewall*, que está instanciado na nuvem, na névoa ou na borda. O fluxo percorre as demais funções, também instanciadas na nuvem/névoa/borda, até alcançar os servidores web. Vale notar que neste exemplo, o filtro de pacotes e o tradutor de endereços estão compartilhando uma mesma máquina física, mas essa configuração não precisa ser estática. Em um momento em que o tráfego na rede esteja relativamente alto, as instanciações podem ser feitas em máquinas separadas para que cada uma possa tirar proveito de todos os recursos físicos de cada um dos servidores. O compartilhamento de um mesmo servidor físico seria interessante em momentos de tráfego baixo, permitindo que menos servidores permaneçam ligados, levando a uma economia em termos de energia elétrica. Além disso, também não é necessário que todas as funções estejam virtualizadas. Seria possível, por exemplo, instanciar apenas o *firewall*.

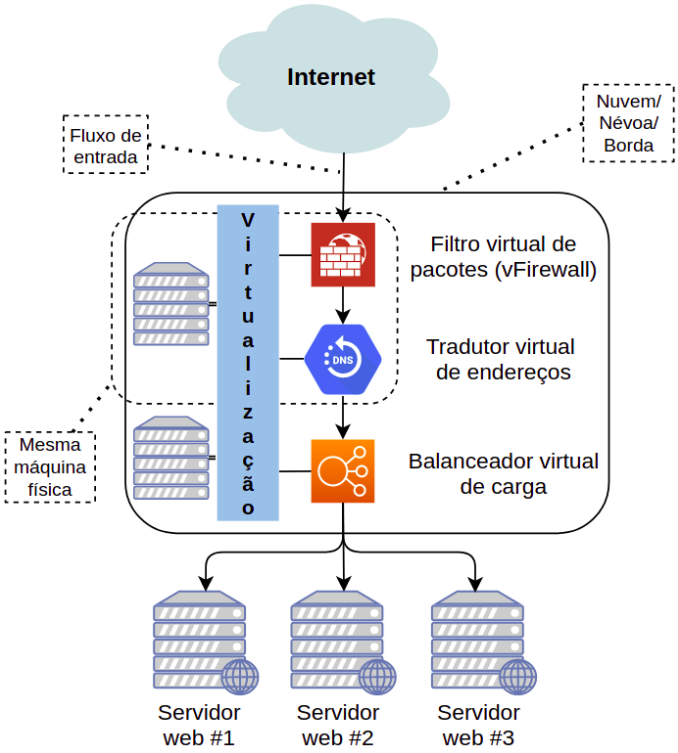


Figura 3.4. Exemplo de um encadeamento de funções de rede com NFV.

Para que a NFV funcione corretamente é necessário que haja a interação de diversos componentes, que podem ser agrupados em camadas, como apresentado na Figura 3.2. As VNFs precisam ser implementadas como *software* (contêineres ou máquinas virtuais, por exemplo) e mantidas em algum repositório, compondo a camada de servi-

ços. A camada de serviços também pode fornecer SFCs inteiras. Sempre que uma VNF é requisitada, ela deve ser instanciada em alguma máquina física. A infraestrutura da NFV (NFVi), na camada de virtualização, é responsável por abstrair os recursos da infraestrutura física, que possivelmente estarão em um ou mais provedores, permitindo que equipamentos de diferentes fabricantes possam oferecer os recursos físicos necessários para a instanciação das VNFs. A NFVi também cuida da interconexão das VNFs entre si e com o mundo exterior, permitindo a interação com a Internet. Para garantir que os recursos da infraestrutura sejam alocados da melhor forma possível, é necessário gerenciar o ciclo de vida de cada uma das VNFs. O gerenciador de VNF na camada de orquestração tem a responsabilidade de cuidar desse ciclo de vida, instanciando, atualizando e encerrando as VNFs. Ainda na camada de orquestração, o gerenciador de infraestrutura virtualizada tem o papel de manter um controle do mapeamento dos recursos físicos para os recursos virtuais.

3.2.2. Redes IoT e Aplicações

A IoT é um paradigma de comunicação com o objetivo de conectar diversos tipos de objetos à Internet [Meneghello et al. 2019]. Dentre esses objetos há, por exemplo, relógios, veículos, prédios, sensores e atuadores de um modo geral. Adicionar suporte à IoT em um objeto consiste na inclusão de recursos como bateria, memória, processador e interface de rede sem fio. Estes recursos são utilizados para execução de protocolos e aplicações da IoT. Com esse suporte, é comum que os objetos passem a ser chamados de "inteligentes". Este adjetivo também é usado para descrever aplicações executadas nos objetos. No cenário da Figura 3.2, os objetos inteligentes estão localizados na camada física.

A IoT não é uma rede à parte da Internet, mas sim uma extensão da mesma que permite a conexão de diversos tipos de objetos do cotidiano. Diferente de um computador convencional, seja ele um computador pessoal ou um servidor, é comum que um dispositivo IoT seja exposto a ambientes críticos para equipamentos eletroeletrônicos, com altas temperaturas e umidade, por exemplo. A depender dos ambientes e das aplicações que serão executadas nos objetos, eles têm restrições nas dimensões físicas. As limitações impostas pelo ambiente e pela dimensão física impedem que dispositivos IoT tenham um alto poder computacional, exigindo que seja dada preferência pela execução de protocolos e aplicações leves, que garantam a qualidade de experiência dos usuários sem que haja um consumo elevado dos recursos do dispositivo, principalmente da bateria.

A Figura 3.5 ilustra a típica arquitetura em três camadas da IoT [Lin et al. 2017]. A **camada de percepção** está na parte inferior da arquitetura. Ela é responsável por interagir com os dispositivos e componentes físicos dos objetos inteligentes, obtendo dados e enviando comandos. A **camada de rede** integra diversos dispositivos de interconexão, como *switches* e pontos de acesso, a várias tecnologias de comunicação, como Bluetooth e Wi-Fi. Nesta camada são determinadas as rotas das comunicações. A **camada de aplicação** é a camada no topo da arquitetura e é onde as aplicações são propriamente implementadas. Aplicativos de cidades inteligentes, indústrias inteligentes e casas inteligentes estão nesta camada. O ideal é que estes aplicativos utilizem protocolos leves da camada de aplicação da Arquitetura Internet, como o MQTT, o CoAP e o AMQP [Naik 2017].

A limitação de recursos dos dispositivos conectados à IoT, unida às configurações

Figura 3.5. Arquitetura em camadas da IoT

padrão dos dispositivos, que não levam em conta alguns aspectos básicos de segurança como criptografia e a obrigatoriedade de senhas fortes [Giaretta et al. 2019], têm tornado a IoT alvo de ataques de segurança. Parte desses ataques é realizada para obter o controle remoto dos dispositivos, que passam a fazer parte das chamadas *botnets*, redes criadas para disparar ataques de larga escala contra a Internet [Schwengber et al. 2020]. No decorrer deste capítulo, o termo Redes IoT será usado para se referir a redes de comunicação que possuam dispositivos IoT.

3.2.3. Gerenciamento de Redes com NFV

A ausência de dispositivos com recursos abundantes para a execução de funções de rede relacionadas à segurança tornam a arquitetura de NFV interessante para prover segurança na IoT. Neste caso, as VNFs de segurança podem ser instanciadas em dispositivos externos, fornecidos por provedores de nuvem, de névoa ou na borda. Para que isso seja possível, os atores da camada de orquestração (Figura 3.2) têm papel fundamental [Medhat et al. 2017].

O gerenciador de infraestrutura virtualizada está presente em cada um dos domínios de infraestrutura virtual. No caso de uma infraestrutura mantida pela plataforma OpenStack[2], por exemplo, o gerenciador utilizado é o `neutron`. Um gerenciador de infraestrutura virtualizada como o `neutron` é responsável por cuidar do endereçamento entre os dispositivos virtualizados, da topologia virtual, e da potencial conexão entre diversas infraestrutura por meio de redes privadas virtuais (VPNs - *Virtual Private Networks*), entregando "rede como um serviço".

O gerenciador de VNF é responsável por gerenciar o ciclo de vida das VNFs, como instanciação, atualização, mudança de configuração, escala vertical (mudança na quantidade de recursos físicos disponibilizados por um computador hospedeiro para uma NF), escala horizontal (mudança na quantidade de computadores hospedeiros usados para instanciar uma NF) e finalização.

[2]OpenStack é uma plataforma de código aberto desenvolvida para gerenciar múltiplos recursos virtualizados. `https://www.openstack.org`.

3.2.4. Questões de Desempenho x Segurança com NFV na IoT

Um fator a ser considerado no uso de qualquer recurso computacional é o seu desempenho. Este caracteriza-se por dois fatores: *métricas* e *medições*. As métricas são definições padrões de quantidades produzidas em um experimento, com uma utilidade pretendida. Elas devem ser cuidadosamente especificadas para transmitir o significado exato de um valor medido. A medição remete-se a um conjunto de operações com o objetivo de determinar o valor de uma métrica [IETF 2011, IETF 1998]. Dessa forma, os indicadores de desempenho computacionais estão diretamente relacionados às métricas dos recursos que demonstram o grau de funcionalidade de um sistema computacional e, com isso, representam de forma abstrata o estado desse sistema. Por exemplo, o tempo necessário para o carregamento de um aplicativo baseado na *web* é o indicador mais perceptível de que o sistema está funcionando no nível esperado. Assim, tempos de carregamento mais longos do que o normal indicam que o sistema pode estar com problemas, exigindo atuações de seus administradores. Esses indicadores mudam ao longo do tempo e são fontes iniciais de informação sobre a saúde do sistema [Moghaddam et al. 2019].

Do ponto de vista de NFV, a Tabela 3.1 apresenta métricas de desempenho relacionadas à velocidade, acurácia e confiabilidade de acordo com as categorias referentes à orquestração, operação de máquina virtual, estabelecimento e operação de redes e componentes de tecnologia como serviço. Essas métricas foram definidas pelo Instituto Europeu de Normas de Telecomunicações (ETSI – *European Telecommunications Standards Institute*) [ETSI 2014a, ETSI 2014b] para entidades que fornecem VNFs e que gerenciam infraestruturas virtualizadas, no intuito de garantir que os recursos oferecidos tenham o desempenho necessário de acordo com os requisitos de qualidade previstos. Por exemplo, uma rede que está sendo altamente demandada por requisições benignas, ou seja, que não são ataques, e possui a latência de provisionamento de máquina virtual (VM – *Virtual Machine*) alta para o serviço virtualizado de *firewall*, poderá gerar grandes atrasos nas respostas às requisições, o que afeta diretamente a qualidade de experiência do usuário final do serviço.

Na mesma linha do ETSI, a Força-Tarefa de Engenharia da Internet (IETF – *Internet Engineering Task Force*) [IETF 2017a] designou um grupo específico, chamado Grupo de Trabalho de Metodologia de *Benchmarking* (BMWG – *Benchmarking Methodology Working Group*), para produzir uma série de recomendações sobre as principais características e análise de desempenho de dispositivos, sistemas e serviços de rede. Nele a Metodologia de *Benchmarking* para Desempenho de Virtualização de Redes foi desenvolvida, considerando como métricas de desempenho a **taxa de transferência**, a **taxa de perda de quadros**, o **consumo de CPU**, o **consumo de memória** e a **latência**.

É possível encontrar, também, métricas relacionadas à NFV em outros trabalhos. Em [Gupta et al. 2019], por exemplo, os autores abordam questões relacionadas à tolerância a falhas de NFV e apresentam métricas específicas para o problema, tais como: perda de pacotes, taxa de transferência média de pacotes, congestionamento de pontos de interconexão, entre outras. Em [Kim et al. 2015], os autores apresentam métricas de comparação de desempenho para controladores de redes definidas por *software* (SDN – *Software Defined Network*) e NFV. As métricas apresentadas sobre NFV são concentradas em vCPU e vMemória (associadas à computação), latência e taxa de transferência

Tabela 3.1. Resumo das métricas de qualidade de serviço em NFV [ETSI 2014b].

Categoria da métrica	Velocidade	Acurácia	Confiabilidade
Primeira etapa da orquestração (por exemplo, alocação de recursos, configuração e instalação)	Latência de provisionamento de VM	Política de posicionamento de VM e conformidade	Confiabilidade de provisionamento de VM
Operação de VM	Interrupção da VM (duração e frequência do evento) e Latência de programação da VM	Erro de *clock* da VM	Taxa de liberação prematura de VM
Estabelecimento de rede virtual (VN – *Virtual Network*)	Latência de provisionamento da VN	Conformidade de diversidade de VN	Confiabilidade de provisionamento de VN
Operação de rede virtual	Atraso de pacote, *jitter* (variação no atraso) e taxa de transferência de pacotes entregues	Taxa de perda de pacotes	Conexão interrompida
Segunda etapa da orquestração (por exemplo, liberação de recursos)	–	–	Taxa de liberação de VM com falha
Componente de tecnologia como serviço (TcaaS)	Latência do serviço TcaaS	–	Confiabilidade TcaaS (por exemplo, relação de transação defeituosa) e interrupção do TcaaS

(associadas à comunicação), taxa de E/S e tempo de recuperação de VNFs (associadas ao armazenamento). Em [Zhang et al. 2016], os autores utilizam métricas como uso de CPU, taxa de transferência de pacotes e latência para apresentar como o efeito do acesso não uniforme a memória (NUMA – *Non-Uniform Memory Access*) e o posicionamento da cadeia de serviços impactam no desempenho de NFV.

Quando as características de dispositivos IoT são consideradas na implementação de uma rede, como baixo consumo de energia e recursos computacionais limitados, deve-se observar principalmente o comportamento da comunicação entre os dispositivos e os serviços disponibilizados a eles. Por exemplo, um determinado dispositivo IoT acoplado a um semáforo deve solicitar a um servidor remoto a resposta de um serviço de cálculo, baseado nas informações de volume de automóveis coletadas, para que o semáforo tenha seus tempos de abertura e fechamento alterados de acordo com o tráfego de veículos. Se essa comunicação não levar em consideração métricas como latência entre o dispositivo IoT e o servidor, o tempo de resposta para a requisição fará com que o serviço prestado não tenha a qualidade pretendida em relação à otimização da gestão do tráfego de veículos. Além disso, é necessário observar o comportamento dos serviços disponibilizados na rede IoT. Nesse contexto, a Figura 3.6 apresenta um cenário onde o serviço de IDS, provido por uma VNF, é consumido sob demanda de acordo com o volume de solicitações das câmeras, independentemente do *hardware* que o sustenta. A Figura 3.7 ilustra não só o consumo do serviço virtualizado de *firewall* feito pelo dispositivo IoT A em movimento, mas também a migração do estado em que ele se encontra entre os diferentes provedores

de VNF, nesse caso, Nuvem #1 e Nuvem #2. Um exemplo prático desse cenário seria um sistema de transporte urbano de uma cidade inteligente onde os ônibus poderiam ser representados pelos dispositivos IoT e as nuvens pelos diferentes provedores de computação disponíveis em localidades geográficas distintas. Para os dois cenários ilustrados, a RFC 8172 [IETF 2017b] adiciona outras métricas ao desempenho de NFV, como tempo para implantar e migrar VNFs, que devem ser levadas em consideração para que o tempo de resposta em uma rede IoT não seja afetado negativamente.

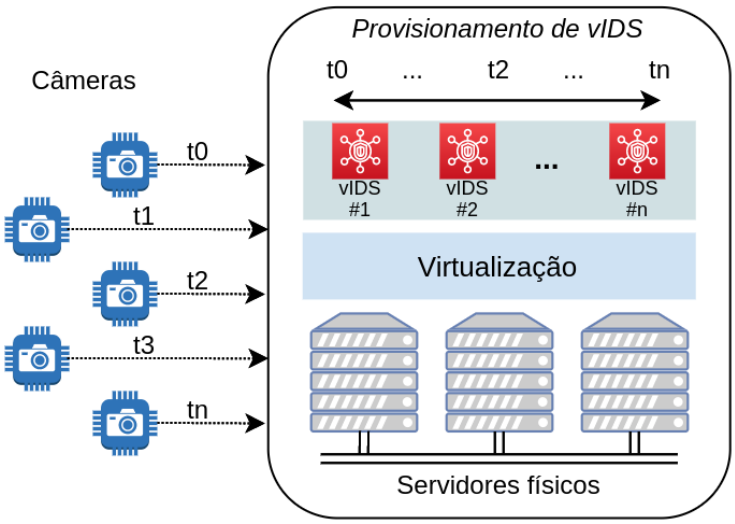


Figura 3.6. Escalando recursos de VNFs de segurança em redes IoT [Farris et al. 2019].

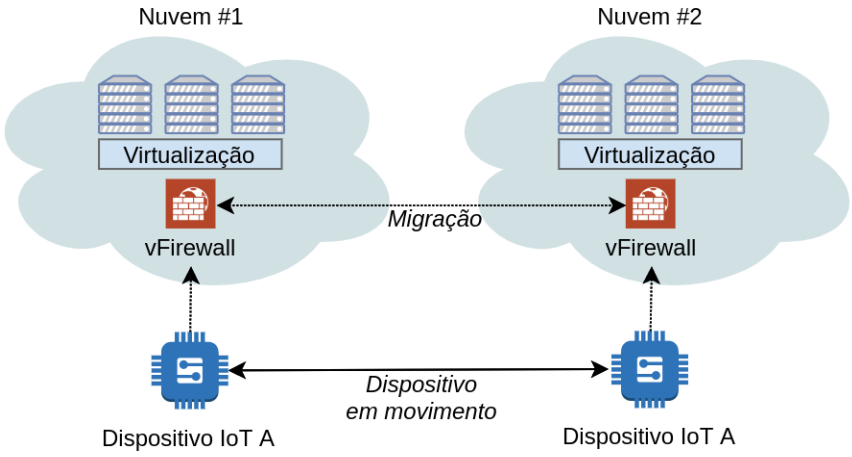


Figura 3.7. Migrando recursos de VNFs de segurança em redes IoT [Farris et al. 2019].

A aplicação de NFV especificamente para segurança em redes IoT é outro fator a ser considerado, abordagem que vem ganhando espaço atualmente [Alam et al. 2020, Farris et al. 2019, Zarca et al. 2018]. Nesse caso, além das justificativas para as métricas apresentadas na Tabela 3.1, a própria funcionalidade de segurança fornecida pela VNF

pode ter requisitos mais rígidos, principalmente em termos de tempo de resposta, para garantir que a detecção e a mitigação de um ataque sejam realizadas antes que seja tarde demais. Por exemplo, um IDS instalado na rede local onde os dispositivos de IoT estão conectados pode agir dentro do tempo esperado pelo fato de ele estar localizado no exato local onde os fluxos de interesse estão trafegando. Repassar esse tráfego para um ambiente computacional remoto, mesmo que esse ambiente tenha alta capacidade de processamento, pode tornar o IDS inútil caso a latência entre a rede local e o ambiente remoto seja muito alto. Esse é um caso em que não há diferentes gradientes para a qualidade do serviço, como no caso de um serviço que, mesmo se estiver "lento", ainda é útil. O não atendimento de um tempo mínimo, permitindo que o ataque tenha sucesso, significa que o serviço requisitado não foi entregue.

De um ponto de vista da camada de serviços, ilustrada na Figura 3.2, o trabalho de [Farris et al. 2019] apresenta abordagens para a aplicação de NFV voltadas à segurança de redes IoT. Isto possibilita a abrangência e a customização de serviços de proteção à rede oferecidos como, por exemplo, segurança como serviço (SECaaS – *Security as a Service*). Os autores classificam os trabalhos da literatura de acordo com quatro aspectos: separação do *software* de segurança do hardware; escalabilidade sob demanda e tolerância a falhas para VNF de segurança; suporte à mobilidade de VNF de segurança; e encadeamento de serviços de segurança de rede. Estendendo essa classificação proposta pelos autores, é possível considerar também o uso de VNF para implementação de modelos de aprendizado de máquina no intuito de detectar e mitigar ataques na rede, denominando a proposta de VNF para modelos de aprendizado de segurança.

Sobre a separação do *software* de segurança do hardware, três trabalhos se destacam. Em [Bremler-Barr et al. 2014], os autores apresentam uma abordagem de virtualização de serviços de DPI cujo desempenho, medido em taxa de transferência de pacotes, foi melhorado quando comparado a soluções implementadas em *middleboxes*. Em [Montero et al. 2015], os autores propõem virtualizar as aplicações de segurança na borda da rede baseadas em domínios virtuais de confiança (TVD – *Trusted Virtual Domain*), um contêiner lógico instanciado na rede composto por aplicativos de segurança do usuário e dados de controle de acesso a outras TVDs. Em [Yu et al. 2015], os autores propõem uma arquitetura de segurança chamada IoTSec, a qual contempla *micro-middleboxes* customizados (*µboxes*) que atuam como *gateways* de segurança para cada dispositivo IoT e cuja implementação pode se dar por meio de VNF. A IoTSec é composta por um controlador centralizado que monitora os contextos dos dispositivos e o ambiente operacional para gerar uma visão global e aplicar políticas entre os dispositivos. Com base nessa visualização, ele instancia e configura *µboxes* individuais e os mecanismos de encaminhamento necessários para rotear pacotes para elas.

Sobre a escalabilidade sob demanda e tolerância a falhas para VNF de segurança, em [Cao et al. 2015], os autores apresentam uma solução chamada de NFV-VITAL. Seu objetivo é determinar a configuração que produz o maior desempenho de uma VNF, ou seja, ela computa a carga de trabalho máxima que uma VNF suporta antes que a qualidade do serviço reduza, usando diferentes tamanhos de implantação e opções de virtualização.

A análise de soluções de IDS virtualizadas, como Snort[3] e Suricata[4], demonstrou os benefícios de selecionar o dimensionamento e as configurações ideais para os mecanismos de segurança. Em [Hafeez et al. 2016], os autores abordam a manutenção de réplicas com reconhecimento de estado de *middleboxes* virtuais com o objetivo de, em caso de falhas, instanciar novos serviços.

Sobre o suporte à mobilidade de VNF de segurança, é possível citar uma estrutura de suporte à migração de instâncias virtuais de segurança perto dos dispositivos do usuário final, como proposto por [Montero and Serral-Gracià 2016]. A abordagem aproveita o uso de tecnologias de virtualização e de SDN para gerenciar a migração de aplicativos de segurança na extremidade da rede, enquanto minimiza a interrupção das conexões em andamento. A solução é composta por quatro componentes: um contêiner virtual de segurança, cuja função é fornecer aplicativos como *firewall* ou alguma composição entre eles; um controlador de rede, que é responsável por configurar o direcionamento do tráfego de rede para os aplicativos de segurança; um migrador de recursos, que realiza a função de mover o estado de um aplicativo de segurança específico para o novo local do usuário; e um orquestrador que coordena a alocação dos recursos, a configuração de rede e a migração dos contêineres virtuais de segurança.

A respeito do encadeamento de serviços de segurança de rede, em [Gember-Jacobson et al. 2014], os autores apresentam o OpenNF, uma solução que implementa um plano de controle dedicado para garantir o controle coordenado dos estados da VNF e dos estados de encaminhamento da rede. Em [Qazi et al. 2013], os autores desenvolveram uma abordagem baseada na aplicação de políticas de SDN para simplificar o encaminhamento de tráfego entre *middleboxes*. A solução, chamada de SIMPLE (*Software-defIned Middlebox PoLicy Enforcement*), é composta pelo processamento de políticas nas quais os administradores de rede configuram suas lógicas de processamento, abstraindo informações sobre onde esse processamento ocorre ou como o tráfego precisa ser roteado. Assim, o SIMPLE traduz as políticas configuradas para a infraestrutura física considerando os dados da topologia e do tráfego da rede. Por fim, a solução também considera restrições de dispositivos como CPU, memória, aceleradores para diferentes *middleboxes* e quantidade de memória TCAM (*Ternary Content Addressable Memory*) disponível para instalar regras de encaminhamento nos *switches* SDN. O trabalho de [Wang et al. 2021] não se restringe a serviços de segurança. Nele os autores propõem uma abordagem baseada em aprendizagem por reforço profundo (do inglês, *deep reinforcement learning*) para posicionar as cadeias de serviços de rede de forma adaptativa. O algoritmo extrai características da rede física em tempo de execução por meio de um grafo de rede convolucional (GCN – *Graph Convolutional Network*) e gera estratégias de posicionamento de serviços através de um modelo de sequência a sequência (Seq2Seq – *Sequence-to-Sequence*), no qual aprende a tomar decisões de posicionamento de cadeias de serviços observando o desempenho correspondente de decisões anteriores.

Dentre os trabalhos da literatura relacionados ao uso de VNF para modelos de aprendizado de segurança, é possível citar [Bülbül and Fischer 2020], onde é proposta

[3]Snort é um sistema de prevenção de intrusão (IPS – *Intrusion Prevention System*) baseado em código aberto. `https://www.snort.org`.

[4]Suricata é um IDS e um IPS também baseado em código aberto. `https://suricata.io`.

uma abordagem de mitigação de ataques de negação de serviço distribuído (DDoS – *Distributed Denial of Service*). Este utiliza SDN e NFV por meio de um algoritmo de aprendizado de máquina baseado em generalização e sumarização e LMP (do inglês, *Longest Matching Prefix*), que derivam padrões para descrever ataques na rede. Após a identificação do ataque, o módulo de geração de padrões deriva uma regra OpenFlow[5] para os demais *switches* da rede, solicitando a filtragem do tráfego. Caso o ataque extrapole a rede local, as regras de filtros de tráfego também podem ser propagadas para roteadores externos à rede local.

Em [Jia et al. 2020], os autores têm como objetivo propor técnicas de defesa contra ataques DDoS de IoT e apresentam um esquema de defesa centrado na borda da rede, denominado FlowGuard. Este consiste na composição de dois fluxos, o de filtro e o de manipulação. O fluxo de filtro utiliza uma tabela que define as regras de filtragem de fluxo para diferentes tipos de ataque DDoS. Também é composto por um algoritmo que detecta anomalias no tráfego de dispositivos IoT que passa por servidores de borda. Uma vez identificada a anomalia na rede, o fluxo de manipulação é ativado e a classificação do ataque DDoS ocorre utilizando técnicas de aprendizado de máquina, como memória de curto prazo longo (LSTM – *Long Short-Term Memory*) e rede neural convolucional (CNN – *Convolutional Neural Network*), que podem ser implementadas por meio de VNFs.

Pode-se dizer que métricas e medições são as bases para o processo de monitoramento de qualquer tipo de rede de computadores, sendo utilizadas para determinar o estado dos componentes e, também, para servir de insumos para o gerenciamento de desempenho. Em redes tradicionais, o desempenho da rede leva em conta questões como transferência de dados, o tempo de resposta da rede, a perda de pacotes, a porcentagem do uso de recursos, a utilização da conexão de dados, entre outras. Contudo, quando um contexto de rede envolve dispositivos IoT juntamente com abordagens de segurança baseadas em NFV, outras questões devem ser incluídas para nortear a análise de seu comportamento e o impacto que elas trazem para o seu desempenho. Alguns exemplos são: no domínio de IoT, tipos de dispositivos e protocolos de comunicação utilizados por eles; em NFV, provisionamento, escalabilidade, conexões de rede e orquestração de VNFs; e no contexto de segurança, ferramentas, algoritmos, tempo de detecção e mitigação de ameaças.

3.3. Gerenciamento de Desempenho e Segurança em IoT com NFV

O gerenciamento de desempenho parte do princípio de como utilizar os indicadores ou métricas coletadas da rede para provisionar de forma eficiente recursos necessários para a manutenção da qualidade dos serviços entregues aos seus usuários. Por isso, para que a operação de um recurso de NFV seja realizada de forma eficiente, os indicadores de desempenho devem ser bem definidos e aferidos a fim de demonstrarem o comportamento e a saúde do sistema. Por serem recursos virtualizados, um fator a ser ponderado por meio da medição dos indicadores é a capacidade máxima da infraestrutura física em que estão hospedados, ou seja, o orquestrador de VNFs não deve permitir a hiper alocação de recursos em uma infraestrutura compartilhada, fazendo com que outros recursos

[5]OpenFlow é um protocolo de comunicação que permite o acesso ao plano de encaminhamento de um *switch* ou roteador, possibilitando o direcionamento e até mesmo o descarte de pacotes da rede. `https://opennetworking.org`.

disponibilizados virtualmente tenham seu desempenho afetado. Nessa mesma linha, o orquestrador também não deve manter alocados recursos que não têm utilidade em um determinado contexto que a rede contempla. Em outras palavras, os indicadores de desempenho contribuem para a tomada de decisão no uso ou não de um determinado recurso para um propósito identificado. Por exemplo, um recurso virtualizado de DPI pode ser alocado a partir do momento em que a rede começa a receber requisições de origens que ela geralmente não recebe, fazendo com que os pacotes sejam inspecionados de forma profunda, com o objetivo de detectar códigos maliciosos. Uma vez não identificado algum tipo de ameaça, a desconexão com o endereço que originou a análise ou até mesmo após a identificação e mitigação do ataque, o recurso virtualizado de DPI deve ser desalocado, liberando recursos computacionais às demais necessidades da rede. Assim, a perspectiva de **gerenciamento de desempenho** apresentada nesta seção, analisa como as métricas e medições de desempenho, apresentadas na Seção 3.2.4, foram abordadas por diversos autores em suas soluções.

Além disso, ao se tratar de dispositivos IoT, é preciso investigar a fundo suas características antes mesmo de propor qualquer tipo de infraestrutura ou mecanismos de segurança dedicados a eles [Lin et al. 2017]. Assim, a perspectiva **IoT** leva em consideração as seguintes características para a análise dos trabalhos da literatura apresentada nesta seção:

- **Dispositivos** utilizados no experimento. A variedade de dispositivos contidos no guarda-chuva de IoT é grande e tende a aumentar. Segundo a Corporação Internacional de Dados (IDC – *International Data Corporation*) [IDC 2020], em 2025, 75% dos mais de 55 bilhões de dispositivos previstos em todo o mundo estarão conectados a uma plataforma IoT. Além disso, a corporação estima que os dados gerados a partir de dispositivos IoT conectados estejam na casa dos 73 ZB (73×10^{21} Bytes), sendo a maioria proveniente de segurança e vigilância por vídeo, além de aplicações industriais. É possível dizer que grande parte dos dados gerados e transitados nessas redes envolve informações privativas de usuários, tais como imagens, documentos, localizações, preferências, entre outros. Por isso, identificar como esses dispositivos estão sendo utilizados auxilia no reconhecimento de possíveis limitações de segurança que podem ser supridas por recursos de NFV;

- **Protocolo** abordado. A aplicação de padrões da Internet já consolidados a dispositivos inteligentes - como o IP - pode simplificar a integração dos cenários previstos em IoT. No entanto, os protocolos de comunicação convencionais da Internet precisam ser modificados ou estendidos para oferecer suporte a necessidades específicas de aplicações IoT. Nessa linha, [Sobin 2020] contempla em seu trabalho um estudo dos protocolos e tecnologias de comunicação dedicadas às camadas: física, contida pelo protocolo IEEE 802.12.4; camada MAC, por IEEE802.15.4E, ZigBee e TSMP (do inglês, *Time Synchronized Mesh Protocol*); camada de convergência, contida por 6LoWPAN; camada de transporte, por UDP (do inglês, *User Datagram Protocol*), IPv6, uIP (*micro* IP), ROLL (do inglês, *Routing Over Low power and Lossy networks*), RPL (do inglês, *IPv6 Routing Protocol for Low Power and Lossy Networks*) e DTLS (do inglês, *Datagram Transport Layer Security*); e camada de aplicação, por CoAP (do inglês, *Constrained Application Protocol*), MQTT (do in-

glês, *Message Queuing Telemetry Transport*), DDS (do inglês, *Data-Distribution Service*), AMQP (do inglês, *Advanced Message Queuing Protocol*), SMCP (do inglês, *Secure Mobile Crowdsensing Protocol*) e LLAP (do inglês, *Lightweight Local Automation Protocol*). Diferentemente de [Sobin 2020], [Lin et al. 2017] consideram que a camada de percepção é dedicada a tecnologias como RFID (do inglês, *Radio Frequency Identification*) e demais sensores de redes IoT. Os protocolos IEEE 802.15.4, 6LoWPAN, ZigBee, Z-Wave, MQTT, CoAP, DDS, AMQP e XMPP (do inglês, *Extensible Messaging and Presence Protocol*) são categorizados na camada de rede, pois a camada de aplicação refere-se ao contexto de negócio do uso de IoT como, por exemplo, cidades e transportes inteligentes. Outros trabalhos [Meneghello et al. 2019, Neshenko et al. 2019, Deogirikar and Vidhate 2017, Yang et al. 2017, Gao et al. 2013] também fazem estudos de camadas e protocolos de comunicação a fim de direcionar soluções, desafios e limitações no contexto de IoT. Portanto, identificar e relacionar os protocolos de comunicação ajuda a compreender de que forma a NFV tem contribuído com esses desafios. Baseado nisso, a Tabela 3.2 traz um consenso entre as camadas e os protocolos de comunicação IoT abordados nos trabalhos analisados;

Tabela 3.2. Protocolos referentes às camadas IoT.

Camada	Protocolo
Aplicação	HTTP(S), MQTT, CoAP, AMQP, LLAP, SMCP, DDS, XMPP
Rede	TCP, UDP, UIP, ROLL, RPL, DTLS, Zigbee, Z-Wave, IPV6, 6LoWPAN, LoRaWAN, BLE
Percepção	IEEE 802.15.4, IEEE 802.15.4E, TSMP

- **Arquitetura** apresentada. Existem diversas vantagens em considerar a utilização da computação em nuvem para aplicações tradicionais, como a virtualização de recursos, escalabilidade, redundância, pagamento de acordo com o uso (do inglês, *pay-as-you-go*), entre outros. No entanto, para aplicações IoT nas quais os dispositivos possuem capacidades limitadas, o processamento de determinadas tarefas deve ser dividido e realizado entre os demais nós da rede em diferentes localidades, e quando esse cenário abrange milhares de dispositivos interconectados, é preciso pensar em outras estratégias de infraestrutura para que o desempenho das aplicações não seja afetado negativamente. Por essa razão, outros paradigmas, como a computação em névoa ou *fog computing*, surgiram para suportar tal realização [Yousefpour et al. 2019, Lin et al. 2017, Singh et al. 2016]. Diferentes abordagens podem ser encontradas no intuito de aproximar o processamento computacional dos dispositivos IoT e reduzir a latência entre o provisionamento de serviços. Por exemplo, a Figura 3.8 ilustra a quantificação dos atributos quando próximos ou distantes da nuvem, além de diferentes níveis de processamento, partindo do nível mais capacitado (nuvem) para o menos capacitado (borda), contemplando um nível intermediário (névoa). Dessa forma, entender as características de infraestrutura consideradas pelos autores dos trabalhos analisados afeta diretamente o desempenho do uso de recursos NFV.

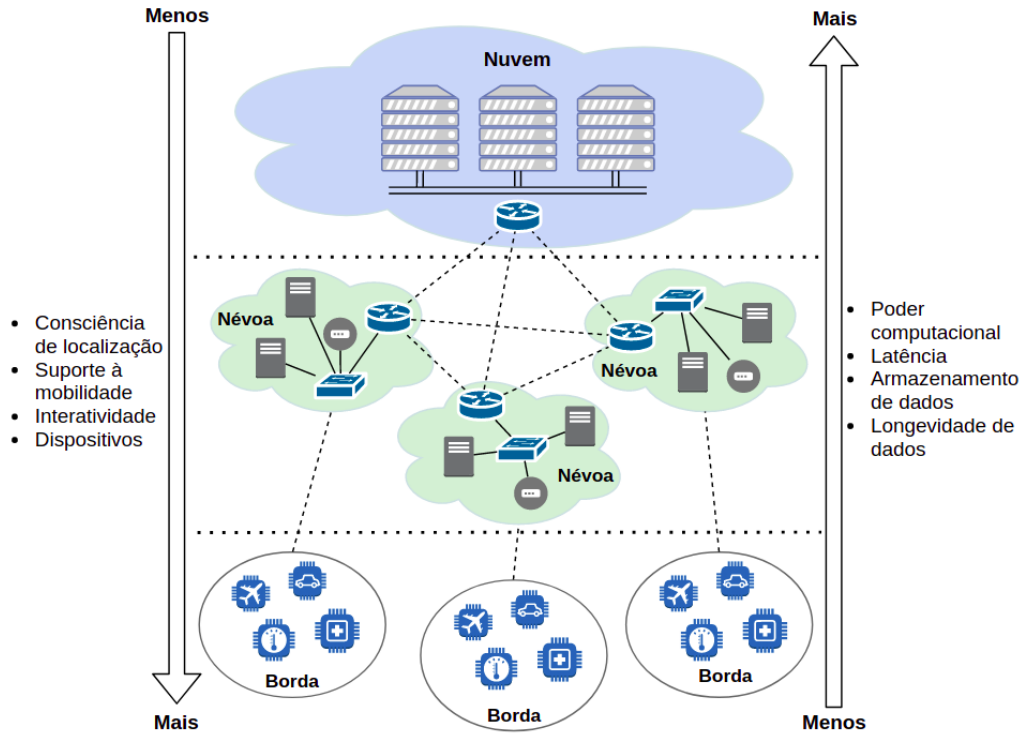


Figura 3.8. Níveis de processamento e capacidades em diferentes arquiteturas distribuídas [Yousefpour et al. 2019, Lin et al. 2017].

É fato que dispositivos IoT possuem características específicas e são desenvolvidos para atuar em cenários bem definidos, ou seja, não possuem um propósito geral de atuação, como computadores pessoais. Por exemplo, equipamentos hospitalares são desenvolvidos visando alcançar a máxima redução de tempo entre a coleta dos sinais vitais de uma pessoa e sua exibição no monitor. Em um contexto industrial, sensores são utilizados para determinar a abertura ou fechamento de comportas de acordo com o volume de determinado líquido. Na gestão de tráfego urbano, dispositivos inteligentes podem coletar informações referentes ao número de carros nas vias e determinar o tempo de fechamento de um ou mais semáforos. Os exemplos são extensos. No entanto, baixo consumo de energia, protocolos de comunicação otimizados e suporte à conexão entre dispositivos heterogêneos são alguns dos requisitos em comum que devem ser observados durante o desenvolvimento de mecanismos de segurança. Assim, do ponto de vista de **segurança**, a análise dos trabalhos da literatura apresentada nesta seção foi baseada nos seguintes itens:

- **Abordagem** da solução proposta. O propósito de um determinado mecanismo de segurança deve ser definido logo na etapa de planejamento de um projeto, sempre levando em conta os fundamentos de confidencialidade, integridade e disponibilidade. Quando em pleno funcionamento, o mecanismo deve objetivar duas principais ações: a detecção e a mitigação de ameaças à rede. A ação de detectar ameaças pode ser considerada uma etapa preliminar do processo de resposta a incidentes de segurança. Do mesmo modo, a mitigação pode ser vista como uma etapa subsequente, cujo objetivo é desabitar ou neutralizar a ameaça presente na rede. Por

exemplo, uma vez identificado um ataque DDoS pelo mecanismo de detecção, o sistema de mitigação pode entrar em ação no intuito de negar ou redirecionar os pacotes maliciosos para o descarte. Entender e avaliar esses mecanismos ajuda a compreender em que nível de completude e maturidade do uso de NFV a solução proposta está;

- **Ataque** considerado. Qualquer computador conectado via rede pode ser alvo de um ataque, da mesma forma que pode participar de um ataque e isso não é diferente em plataformas IoT. Neste contexto, o trabalho realizado por [Meneghello et al. 2019] apresenta vulnerabilidades em aberto de tecnologias como Zigbee, BLE (do inglês, *Bluetooth Low Energy*), 6LoWPAN e LoRaWAN da camada de rede. [Yang et al. 2017] pontua as principais limitações de dispositivos IoT, além de apresentar e analisar questões de segurança classificadas pelas camadas de percepção, rede, transporte e aplicação. No trabalho de [Deogirikar and Vidhate 2017], além de considerar os ataques a redes IoT em camadas física, de rede e de *software*, são pontuados ataques específicos de encriptação, tais como: *side-channel*, *cryptanalysis of a simple modular exponentiation*, *ciphertext only*, *known plaintext*, *chosen plaintext*, *chosen ciphertext* e homem no meio (MITM – *Man-in-the-Middle*). [Neshenko et al. 2019] vai além da classificação feita pelas camadas baseadas em dispositivo, rede e *software*, considerando o impacto que os ataques têm em relação à confidencialidade, integridade, disponibilidade e responsabilização. Para isso, os autores consideram que para a camada baseada em dispositivo, as vulnerabilidades identificadas são referentes à deficiência física ou captação de energia insuficiente. As vulnerabilidades de autenticação inadequada, encriptação inapropriada e portas desnecessariamente abertas são citadas para a camada baseada em rede e, para a camada de *software*, são considerados itens como controle de acesso insuficiente, gerenciamento impróprio de atualizações de correção, práticas inseguras de desenvolvimento de código e mecanismos insuficientes de autenticação. A Tabela 3.3 apresenta uma descrição dos diferentes tipos de ataques em redes IoT e auxilia na compreensão das ameaças que têm sido detectadas e mitigadas pelo uso de NFV.

[Farris et al. 2017] apresentam uma proposta de um *framework* cujo objetivo é gerenciar e orquestrar políticas de segurança em redes IoT, na qual a heterogeneidade de dispositivos físicos e virtuais prevalece. A arquitetura do *framework* é composta pelos planos de usuário, de orquestração, de aplicação de segurança e de gerenciamento. O plano de usuário implementa e oferece uma interface para a definição de políticas de segurança, tais como autenticação e autorização, filtragem e encaminhamento e proteção de canal. Após as políticas de segurança serem definidas pelo usuário, o plano de orquestração as interpreta e as incorpora no processo de execução. Além disso, o plano de orquestração também inclui os módulos de monitoramento, reação, orquestração de segurança e ativadores, que são responsáveis, respectivamente, por coletar as informações em tempo real dos componentes da rede; interpretar as informações recebidas do módulo de monitoramento e selecionar as contramedidas de acordo com as políticas de segurança definidas; selecionar os ativadores de acordo com a contramedida; e aplicar as políticas de segurança. Os ativadores, neste caso, são considerados os recursos provisionados via VNFs. O plano de aplicação de segurança é contido pelos domínios de gerenciamento e controle,

Tabela 3.3. Tipos de ataques a redes IoT e suas descrições.

Ataque	Descrição
Vírus, *worms*, cavalo de tróia, *spyware* e *adware*	O invasor pode alterar o comportamento do sistema, roubar dados ou invalidar acessos usando esses códigos maliciosos.
Scripts maliciosos	Ao injetar um *script* malicioso, o invasor obtém acessos privilegiados ao sistema.
Phishing	O invasor obtém as informações confidenciais como nome de usuário e senhas através da divulgação de *links* via e-mail, redirecionando a vítima para páginas falsas.
Negação de serviço (DoS) ou negação de serviço distribuída (DDoS)	O invasor faz com que o sistema alvo pare de fornecer seus serviços por meio do consumo de largura de banda da rede.
Sinkhole	O invasor compromete um nó dentro da rede e executa o ataque a partir desse nó. O nó comprometido envia informações de roteamento falsas para os nós vizinhos de que tem o caminho mínimo e, em seguida, atrai o tráfego para ele.
Roteamento	O invasor modifica e envia informações de roteamento anormais. Isso pode resultar no descarte de pacotes, encaminhamento de dados errados ou particionamento da rede.
Homem no meio (MITM)	O invasor intercepta a comunicação entre dois nós da rede e obtém acesso aos pacotes trocados por eles.
Sybil	O nó malicioso assume ilegalmente várias identidades e age de acordo com elas.
Captura de tráfego	O invasor intercepta e examina as mensagens para obter informações da rede.
Esgotamento de bateria	O invasor consome mais energia do que o previsto e, consequentemente, desativa o dispositivo.
Interferências de rádio frequência (RF) em RFIDs	O invasor envia sinais com ruídos via radio frequência a fim de clonar a etiqueta do dispositivo alvo para obter acesso não autorizado ou desabilitar a comunicação.

infraestrutura e virtualização, VNF e IoT. O domínio de gerenciamento e controle supervisiona o uso dos recursos em tempo de execução, além de orquestrar as VNFs sobre a infraestrutura virtualizada seguindo as recomendações de gerenciamento e orquestração da ETSI. O domínio de infraestrutura e virtualização é responsável por disponibilizar, de forma virtualizada, os recursos de CPU, memória e elementos de redes. O domínio de VNF provê os recursos virtualizados e utilizados sob demanda para a rede, tais como *firewall*, IDS, DPI, entre outros. Por fim, o domínio de IoT inclui os dispositivos que fazem parte da rede a ser gerenciada.

Para demonstrar a aplicabilidade do *framework*, os autores descrevem dois estudos de caso e experimentam seu desempenho em uma prova de conceito em [Zarca et al. 2018]. O primeiro estudo de caso considera um cenário MEC (do inglês, *Multi-access Edge Computing*), no qual o principal objetivo do *framework* é suportar a capacidade de prover VNFs mais próximas dos dispositivos IoT, fazendo com que os requisitos de qualidade da rede sejam atendidos. O segundo estudo de caso apresenta um cenário relacionado a prédios inteligentes, BMS (do inglês, *Building Management System*), onde a proteção de dispositivos distintos, como computadores e aparelhos de ar condicionado, pode se dar por políticas de segurança que diferem os canais de comunicação. A

prova de conceito foi focada na aplicação de políticas para provisionar mecanismos de segurança em SDN. Os experimentos consideraram a comparação entre três abordagens. A primeira e a segunda foram implementadas pelos controladores SDN *OpenDaylight* (ODL) e *Open Network Operating System* (ONOS), respectivamente, e a terceira abordagem por um componente de VNF de *firewall* baseado em NETCONF e iptables. Também consideraram um ambiente sob ataque do tipo MITM já detectado, utilizando como estratégia de mitigação o isolamento dos sensores comprometidos. Sobre os dispositivos IoT, foi utilizada uma máquina virtual executando um emulador Cooja Contiki, contido por sensores IoT conectados por meio do protocolo 6LoWPAN.

Os resultados foram aferidos de acordo com o tempo de reação necessário para cada abordagem implementada em um intervalo de 1 a 1000 aplicações de políticas de segurança para a mitigação do ataque. Em comparação com os controladores SDN ODL e ONOS, a implementação do *firewall* em VNF teve um desempenho inferior, principalmente na etapa de aplicação das políticas. Segundo os autores, o tipo de comunicação foi o principal motivo para essa perda de desempenho, ou seja, enquanto que o SDN estabelece inicialmente uma conexão TLS e a mantém aberta durante a seção em que está ativo, o NETCONF realiza a abertura de um canal SSH para cada solicitação.

A evolução do *framework* proposto inicialmente por [Farris et al. 2017] e experimentado por [Zarca et al. 2018] teve sua continuação em outros trabalhos. Por exemplo, em [Zarca et al. 2019b], os autores apresentam um abordagem para gerenciar políticas de autenticação, autorização e responsabilização (AAA – *Authentication, Authorization e Accounting*) na IoT através de uma VNF chamada de vAAA. Ela se estabelece em nível de névoa juntamente com políticas de segurança, sendo executadas em um controlador SDN em nível de nuvem. Os autores também propõem realizar a comunicação entre dispositivos IoT e serviços da rede por meio de canais virtuais protegidos (vChannel-Protection), em específico, utilizando o DTLS. Para a experimentação, foram utilizados dispositivos IoT em uma versão personalizada do Contiki OS 2.7 e do servidor erbium CoAP, juntamente com o protocolo 6LoWPAN.

O desempenho dos componentes propostos foi avaliado de acordo com o processo de comunicação segura entre os dispositivos IoT e o *broker*[6], que consiste nas etapas de autenticação, autorização e canal protegido a partir das ações de refinamento, tradução e aplicação da política de segurança. Sob essa dimensão, os valores apresentados para as etapas de autenticação, autorização e canal protegido se mantiveram constantes entre 0 e 0,1 segundos para todas as ações, exceto para a de aplicação da política na etapa de canal protegido, que consumiu 0,35 segundos. Os autores também avaliaram o consumo de tempo para a aplicação de políticas, CPU e memória RAM de acordo com a escalabilidade de dispositivos IoT em um intervalo de 1 a 500. Os resultados mostraram que o tempo médio para as ações de refinamento, tradução e aplicação das políticas foi em média de 30 segundos para 500 dispositivos IoT na rede. O consumo de CPU tornou-se constante, próximo dos 100%, a partir de 100 dispositivos IoT e o consumo de memória RAM se manteve estabilizado para as ações de refinamento e tradução de políticas, em torno de 40 MB, enquanto que para a ação de aplicação da política o consumo chegou aos 120 MB.

[6]Um *broker* é um elemento de software usado em protocolos do tipo *publish-subscribe* que recebe as mensagens dos publicadores e as envia para os assinantes.

Em [Zarca et al. 2019a], os autores apresentam uma arquitetura referente ao gerenciamento de segurança para o *framework*. Nessa etapa, os autores desenvolveram um módulo de gerenciamento composto por um componente chamado de filtragem e pré-processamento de dados, que realiza a captura de informações da rede e as encaminha ao detector de incidentes. Este último executa a análise de segurança, procurando possíveis ataques na rede. O módulo também contém um banco de dados de assinaturas de ataques e um outro componente baseado em inteligência artificial, que aplica técnicas de aprendizado de máquina para detectar anomalias de comportamento, componente esse desenvolvido por [Bagaa et al. 2020]. Os experimentos implementados consideraram os dois cenários citados anteriormente, MEC e BMS, e os mesmos dispositivos de [Zarca et al. 2019b]. Além disso, foram realizados ataques DDoS no cenário MEC e a simulação de um dispositivo IoT infectado por um *malware* a fim de demonstrar a capacidade do módulo em detectar e mitigar a ameaça na rede.

Os resultados apresentados na análise do desempenho foram relacionados à detecção e mitigação dos ataques. Para a detecção, a solução mostrou um tempo médio de 460 milissegundos em 100 testes realizados. Para a mitigação, ou seja, a aplicação de políticas de segurança na rede, o tempo médio entre tradução e aplicação foi de 406 milissegundos também para as 100 iterações. Portanto, a execução do ciclo completo de resposta a ataques de DDoS levou menos de um segundo.

Seguindo a mesma linha de evolução do *framework*, o trabalho de [Zarca et al. 2020b] traz uma nova abordagem de mitigação de ataques à rede IoT por meio da implementação automatizada de *honeynets* baseadas em VNFs. A Figura 3.9 ilustra a criação de uma rede de dispositivos IoT virtualizados (vIoTHoneynet) sem nenhuma função para que, após a detecção de um ataque, o redirecionamento do tráfego seja feito para a vIoTHoneynet através de regras atualizadas no controlador SDN. Os experimentos foram baseados em dois cenários BMS e os experimentos foram executados 100 vezes para o mesmo caso de teste.

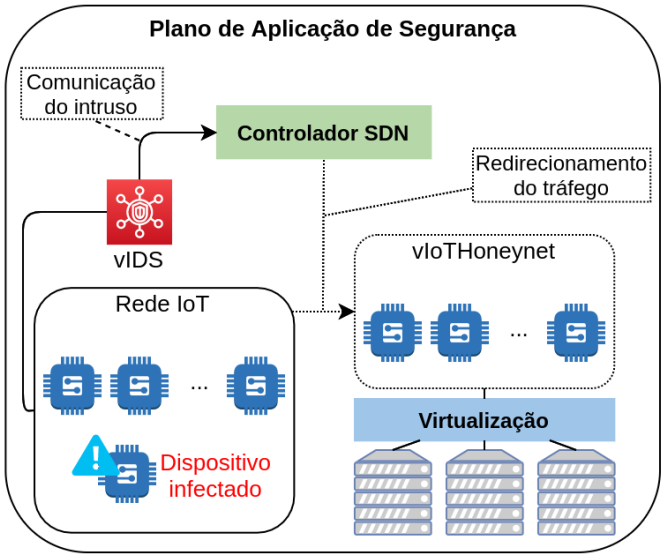


Figura 3.9. Arquitetura vIoTHoneynet [Zarca et al. 2020b].

Considerando que a vIoTHoneynet é instanciada sob demanda, ou seja, apenas quando um ataque é detectado na rede IoT, os experimentos consideraram dois tipos de tempos para compor o tempo total de mitigação. O primeiro tempo foi referente à configuração da vIoTHoneynet composta pelos tempos de compilação e carregamento dos códigos dos novos dispositivos IoT, além do tempo de configuração da rede para esses dispositivos. O segundo tempo foi referente ao estabelecimento da vIoTHoneynet na rede, por exemplo, pela configuração do roteamento das conexões do atacante através da implementação de novas regras no controlador SDN e pelo tempo de aplicação das políticas de segurança em si.

O primeiro cenário de experimentação contou com 20 e o segundo com 50 dispositivos IoT remotamente acessíveis, distribuídos em sensores de umidade, temperatura, luz, CO2, RFIDs para a abertura de portas e roteadores RPL. No entanto, os cenários não se diferenciaram apenas pela quantidade de dispositivos, mas também pela versão do sistema operacional Contiki utilizado para a simulação dos dispositivos IoT, respectivamente, 2.7 e 3.1. Os autores também consideraram a implantação de diferentes tamanhos de vIoTHoneynets para cada cenário. Esses tamanhos de teste foram de no mínimo 10 dispositivos a no máximo 50. Os resultados mostraram que para o primeiro cenário, o tempo médio para a configuração da vIoTHoneynets foi de 15 segundos. Para o segundo cenário, o tempo médio de configuração foi de 45 segundos. Quando os tempos de configuração da vIoTHoneynets foram somados ao tempo de aplicação das políticas de segurança para mitigar o ataque, o tempo médio de mitigação do ataque para o primeiro cenário ficou em torno de 51,5 segundos, e para o segundo cenário, 160 segundos. Os autores também aferiram o consumo de CPU e memória RAM das vIoTHoneynets criadas. Em ambos os cenários, o consumo de CPU se manteve em 100% para todos os tamanhos de teste, enquanto que o consumo de memória RAM no primeiro cenário cresceu de acordo com o aumento dos testes se mantendo no limite de 20% de uso, diferentemente do segundo cenário, que alcançou um consumo máximo de 25%.

Em relação à aplicação de aprendizagem de máquina para a detecção de anomalias em redes IoT, o trabalho de [Bagaa et al. 2020] propõe um modelo baseado em aprendizagem supervisionada desenvolvido sobre o *framework* proposto por [Farris et al. 2017]. Nele, os autores implementam um agente de inteligência virtual que recebe as informações do módulo de monitoramento da rede e as analisa no intuito de detectar padrões referentes a ataques como DDoS, *Probing*, U2R (do inglês, *User to Root*) e R2L (do inglês, *Remote to Local*). Os modelos de aprendizagem *Random Forest*, J48, *Bayesian Network* e *Hoeffding Tree* foram desenvolvidos e treinados a partir de conjuntos de dados que incluem os tipos de ataque citados anteriormente. Os experimentos foram baseados na coleta de dados de sensores de temperatura e CO2 de salas distintas e subdivididos em quatro categoria: (SV), conjunto de dados contento apenas os valores de medição e data; (P5V), conjunto de dados que inclui data, valor, valor precedente, valor 2º precedente, valor 5º precedente; (PD3V), que inclui data, valor, valor precedente diferente, 2º precedente diferente, 3º precedente diferente; e (CR), conjunto de valores cruzados entre as salas sendo sua estrutura data, sala 1, sala 2, sala 3, sala 4, etiqueta. Os resultados apresentados mostraram que para os dados de sensores de temperatura, o modelo desenvolvido teve uma acurácia de 99,71% para SV e P5V, de 99,28% para PD3V e de 99,23% para CR. Para o dados de sensores de CO2, o modelo mostrou uma acurácia de 98,86%

para SV, de 99,24% para P5V e de 99,13% para PD3V.

No mesmo contexto referente à aplicação de aprendizagem de máquina, o trabalho de [Sairam et al. 2019] apresenta uma solução chamada NETRA, que disponibiliza VNFs na borda da rede IoT no intuito de melhorar sua segurança. A solução foi desenvolvida para ser executada em dispositivos de baixa capacidade de processamento computacional. Por isso, utiliza uma estrutura baseada em contêineres Docker[7] para suportar as VNFs de segurança. A Figura 3.10 ilustra como os componentes da solução são distribuídos. A camada de núcleo da rede é composta pela conexão com o provedor de Internet (ISP – *Internet Service Provider*) e pelo Docker Hub[8], responsável por armazenar as imagens utilizadas em cada VNF. A camada de borda contém o *gateway* IoT desenvolvido em Raspberry Pi 3, onde se implementa as VNFs para aquele domínio de segurança. A camada de dispositivos IoT é composta pelos dispositivos inteligentes utilizados na rede. Todo o tráfego de rede realizado na borda é analisado pelo componente vAnalytics que implementa um modelo de aprendizado de máquina baseado em *Random Forest* para classificar os ataques a partir de anomalias identificadas. O modelo utilizado foi treinado a fim de detectar e mitigar ataques do tipo DDoS.

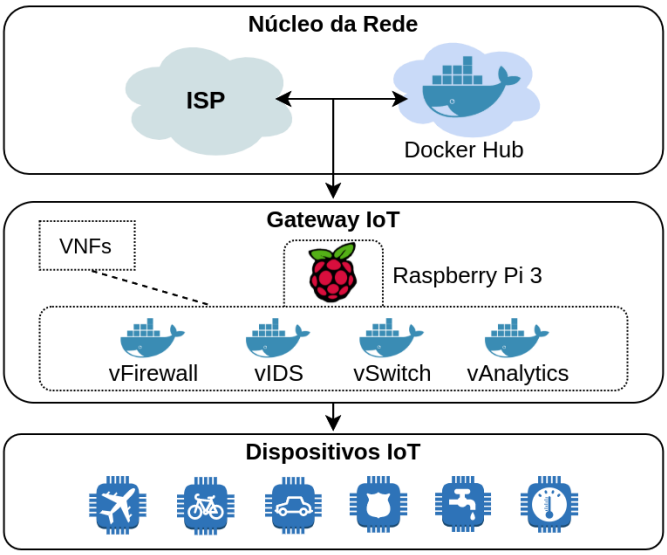


Figura 3.10. Arquitetura NETRA [Sairam et al. 2019].

Os experimentos foram realizados sob duas perspectivas de implementação, a de máquinas virtuais e a de contêineres, com o objetivo de avaliar os indicadores de desempenho: armazenamento, memória, latência, taxa de transferência e escalabilidade. Sobre o desempenho de armazenamento, as máquinas virtuais requisitaram 72 GB de espaço, enquanto que os contêineres já instanciados com as VNFs requisitaram 465 MB. Sobre o consumo de memória, as máquinas virtuais tiveram um pico de 140 MB, e os contêineres de 110 MB, para quatro instâncias em execução. Os valores apresentados de latência

[7]Docker é uma plataforma que utiliza a virtualização para executar aplicações em pacotes chamados contêineres. Os contêineres são isolados uns dos outros e agrupam suas próprias aplicações, bibliotecas e arquivos de configuração `https://www.docker.com`.

[8]Docker Hub é o principal repositório de imagens de aplicações disponibilizadas em contêineres `https://hub.docker.com`.

entre VNFs e entre dispositivos IoT e VNFs foram, em média, 0,83 milissegundos para a implementação baseada em máquinas virtuais e 0,53 milissegundos para a implementação em contêineres. Para a taxa de transferência, os resultados mostraram que os contêineres podem suportar até 16,8 Mbit/s para pacotes TCP e aproximadamente 1 Mbit/s para pacotes UDP. Sobre a escalabilidade, a perspectiva baseada em máquinas virtuais apresentou um tempo aproximado de 20 minutos para configurar uma VNF, e de cinco minutos para iniciá-la/pará-la. A perspectiva baseada em contêineres apresentou um tempo aproximado de quatro minutos para configurar uma VNF pela primeira vez, e a partir desse momento, um tempo de nove segundos. Além disso, os experimentos mostram que o tempo para iniciar uma VNF em um contêiner é menor que um segundo. Os autores também apresentam uma avaliação do desempenho do modelo de aprendizagem para a detecção e mitigação de ataques DDoS. Nesse caso, o modelo apresentou uma acurácia de 94,4% e um tempo médio de resposta da VNF para mitigação menor que um segundo.

O trabalho de [Boudi et al. 2019] também apresenta uma abordagem baseada em contêineres para arquiteturas de computação de borda. Nele, os autores realizaram a comparação de desempenho onde o Suricata foi executado em um dispositivo Raspberry Pi 3, que contempla dois cenários diferentes: o primeiro em máquina física (SoBM) e o segundo, em um contêiner Docker (SoDC). Os indicadores de desempenho utilizados na comparação foram: número de pacotes processados, taxa de transferência, consumo de memória RAM e CPU e número de pacotes descartados. Os resultados mostraram que, em relação ao número de pacotes processados, taxa de transferência e consumo de memória RAM, os valores foram muito próximos, podendo considerar um empate entre as duas implementações. No entanto, os indicadores de consumo de CPU e número de pacotes descartados apontaram que a abordagem SoDC possui uma vantagem sobre SoBM, onde o consumo de CPU foi 10% menor e o número de pacotes descartados foi de 4,5% menor.

Em [Guizani and Ghafoor 2020], os autores utilizam das vantagens da escalabilidade providas pela NFV para implementar um IDS baseado em modelos de aprendizagem de máquina para mitigar *malwares* em redes IoT. A solução apresentada é composta por um modelo RNN-LSTM, executando em uma rede centralizada, que detecta os ataques na rede, e cria zonas de vigilância por meio de diferentes NFVs para realizar a aplicação de *patches* de segurança nos dispositivos. Para demonstrar o desempenho da abordagem, os autores realizaram experimentos considerando o modelo epidêmico SEIR, usado na área de epidemiologia. Nesse modelo, há indivíduos suscetíveis (S), expostos (E), infectados (I) e resistentes (R). No caso de [Guizani and Ghafoor 2020], os indivíduos são os dispositivos IoT. A partir da disseminação de um *malware* na rede IoT, o desempenho da abordagem foi aferido de acordo com a escalabilidade da rede, ou seja, a criação de zonas de vigilância em NFV e a quantidade de vírus na rede. Foram consideradas diferentes quantidades de dispositivos na rede, dentre 1000, 10.000 e 100.000. Os resultados mostraram que a criação de zonas de vigilância virtualizadas se mantém constante para 1000 dispositivos na rede. Nesse caso cinco zonas foram criadas. Já o aumento do número de zonas de vigilância, para 50, iniciou-se a partir de 10.000 dispositivos na rede. No entanto, se manteve constante até os 100.000.

Levando em consideração a computação em névoa, [Zhou et al. 2019] propõem a implementação de VNFs para a realização da detecção de ataques DDoS em redes IIoT (do inglês, *Industrial Internet of Things and Services*). A abordagem de detecção de

anomalias proposta pelos autores considera o uso do Snort para implementar máquinas de estado modeladas a partir de protocolos de comunicação TCP e proprietários utilizados em sistemas de controle industrial, nesse caso, o *Modbus*. Os resultados apresentados mostraram que em uma arquitetura que considera o processamento do sistema de detecção próximo aos dispositivos IoT, há um tempo médio de 137 milissegundos e uma taxa de detecção de 93,93%, enquanto que o tempo médio de resposta em uma arquitetura local é de aproximadamente 240 milissegundos e uma taxa de detecção de 84,95%.

A complexidade no gerenciamento de segurança de redes IoT também é uma questão a ser abordada quando há mais de um provedor de serviços na rede. As cidades inteligentes são alguns dos exemplos. Nelas, diversos provedores de computação em nuvem podem disponibilizar diferentes serviços aos vários dispositivos IoT conectados pela cidade. No entanto, a oferta desses serviços deve levar em consideração as características de redes IoT, como a utilização de *gateways* de borda e até mesmo domínios específicos para este tipo de conexão. Por isso, [Massonet et al. 2017] apresentam uma extensão de uma arquitetura de segurança em nuvem federada para redes IoT, baseada na aplicação de políticas de segurança. A segurança em uma arquitetura de nuvem federada é composta por um componente gerenciador central que realiza a gestão das políticas de segurança da rede e as disponibiliza por meio dos demais gerenciadores de nuvens, que são federadas. A proposta apresentada pelos autores considera que a implementação das políticas de segurança de rede, feita por cada provedor de nuvem, seja realizada por meio das cadeias de VNF de segurança e pelo roteamento do tráfego via regras no controlador SDN. No contexto de IoT, a arquitetura proposta se estende para a utilização de *gateways* de borda, onde se realiza a conexão direta com os dispositivos IoT. Além disso, a proposta também considera que o processamento das VNFs de segurança deve ser realizado na borda, devido ao baixo poder computacional e ao possível aumento da latência devido à distância entre o provedor de serviço e os dispositivos. Os autores não avaliaram o desempenho dos recursos propostos.

Em um contexto de soluções domésticas inteligentes, [Al-Shaboti et al. 2018] apresentam uma proposta da utilização de controladores SDN para reforçar o controle de acesso estático e dinâmico à rede IoT. Além disso, apresentam a implementação de uma VNF para mitigar ataques de falsificação ARP (do inglês, *Address Resolution Protocol*). A implementação do servidor ARP foi realizada de duas formas, uma utilizando a biblioteca Scapy[9] (SC), e a outra utilizando o *Data Plane Development Kit* (DPDK). Esse servidor tem como objetivo ser uma entidade confiável que realiza a gestão das requisições ARP na rede. Desta forma, todos os pacotes ARP na rede são filtrados pelo controlador SDN e são encaminhados ao servidor para resolução. Para demonstrar a abordagem de mitigação, os experimentos foram executados utilizando o Mininet[10], de acordo com diferentes quantidades de máquinas presentes na rede, começando em dez e finalizando em 50. Os resultados mostraram que o tempo de resposta da VNF implementada em SC foi de mais de 70 milissegundos para dez dispositivos, e de aproximadamente um segundo para 50 dispositivos. Para a implementação em DPDK, o tempo de resposta

[9]Scapy é uma biblioteca do Python específica para a criação e manipulação de pacotes de redes. `https://scapy.net`.

[10]Mininet é um emulador de redes que possibilita a criação virtual de máquinas, *switches*, controladores e conexões. `http://mininet.org`.

Tabela 3.4. Panorama do desempenho de NFV para o contexto de segurança.

Trabalho	Desempenho avaliado	Segurança	
		Abordagem	Ataque
[Farris et al. 2017]	—	Prevenção	—
[Massonet et al. 2017]	—	Prevenção	—
[Al-Shaboti et al. 2018]	Tempo de resposta ARP	Prevenção e mitigação	MITM
[Zarca et al. 2018]	Tempo de aplicação de políticas de segurança	Prevenção e mitigação	MITM
[Boudi et al. 2019]	Número de pacotes processados, número de pacotes descartados, taxa de transferência, consumo de CPU e de memória RAM	Detecção e mitigação	—
[Sairam et al. 2019]	Armazenamento, consumo de memória RAM, latência, taxa de transferência, escalabilidade, tempo de implantação de ambiente e acurácia do modelo	Detecção e mitigação	DDoS
[Zarca et al. 2019a]	Tempo de detecção e de aplicação de políticas de segurança	Detecção e mitigação	DDoS e *malware*
[Zarca et al. 2019b]	Tempo de aplicação de políticas de segurança, consumo de CPU e de memória RAM	Prevenção e mitigação	MITM
[Zhou et al. 2019]	Tempo de detecção	Detecção e mitigação	DDoS
[Afek et al. 2020]	—	Prevenção	—
[Bagaa et al. 2020]	Acurácia do modelo	Detecção	DDoS, Probing Attack, U2R e R2L
[Guizani and Ghafoor 2020]	Escalabilidade	Detecção e mitigação	*Malware*
[Zarca et al. 2020b]	Tempo de implantação de ambiente e de aplicação de políticas de segurança	Detecção e mitigação	DDoS

médio foi de 0,57 milissegundo. Em relação à mitigação do ataque de falsificação ARP, os autores apresentam apenas a taxa de 100% de descarte para os pacotes maliciosos sem contabilizar o tempo de execução do descarte.

A descrição de um dispositivo IoT e seu comportamento na rede podem ser utilizados para a criação de melhorias em mecanismos de segurança. Dessa forma, a [IETF 2020] orienta os fabricantes de dispositivos IoT e de sistemas de comunicação sobre a implementação da MUD (do inglês, *Manufacturer Usage Description*). Armazenadas em arquivos, as MUDs consistem em listas de permissões que descrevem domínios e *endpoints* para cada tipo de dispositivo na rede. Assim, logo ao conectar, os dispositivos IoT solicitam a MUD ao servidor dedicado a esse propósito para, então, se integrar com os demais componentes e serviços.

Nesse sentido, o trabalho de [Afek et al. 2020] apresenta uma proposta de uti-

Tabela 3.5. Panorama do desempenho de NFV para o contexto de IoT.

Trabalho	Desempenho avaliado	IoT		
		Protocolo	Arquitetura	Dispositivo
[Farris et al. 2017]	—	—	—	—
[Massonet et al. 2017]	—	—	Névoa	Dispositivos em geral
[Al-Shaboti et al. 2018]	Tempo de resposta ARP	—	—	Dispositivos em geral
[Zarca et al. 2018]	Tempo de aplicação de políticas de segurança	6LoWPAN	—	Dispositivos em geral
[Boudi et al. 2019]	Número de pacotes processados, número de pacotes descartados, taxa de transferência, consumo de CPU e de memória RAM	—	Névoa	Raspberry Pi 3
[Sairam et al. 2019]	Armazenamento, consumo de memória RAM, latência, taxa de transferência, escalabilidade, tempo de implantação de ambiente e acurácia do modelo	—	Névoa	Raspberry Pi 3 e câmeras IP
[Zarca et al. 2019a]	Tempo de detecção e de aplicação de políticas de segurança	CoAP e 6LoWPAN	Névoa	Dispositivos em geral
[Zarca et al. 2019b]	Tempo de aplicação de políticas de segurança, consumo de CPU e de memória RAM	CoAP e 6LoWPAN	Névoa	Dispositivos em geral
[Zhou et al. 2019]	Tempo de detecção	TCP e Modbus	Névoa	Dispositivos em geral
[Afek et al. 2020]	—	—	—	Dispositivos que suportam MUD
[Bagaa et al. 2020]	Acurácia do modelo	—	—	Sensores de temperatura e CO2
[Guizani and Ghafoor 2020]	Escalabilidade	—	—	Dispositivos em geral
[Zarca et al. 2020b]	Tempo de implantação de ambiente e de aplicação de políticas de segurança	CoAP, RPL e LoWPANs	—	Sensores de umidade, temperatura, luz, CO2 e RFID

lização de VNFs em nível de ISP para a aplicação de políticas de segurança em redes

domésticas em larga escala baseadas em MUD. O objetivo da proposta é garantir que os pacotes que trafegam na rede de ou para um dispositivo IoT sejam analisados e confrontados com suas regras contidas no arquivo MUD. Este mecanismo consiste em dois componentes, o de monitoramento (WLM – *whitelist monitoring*) e o de aplicação (WLE – *whitelist/MUD enforcement*). O componente WLM é responsável por analisar se os pacotes estão ou não de acordo com as regras. O componente WLE recebe as informações da WLM e aplica as políticas de exclusão ou de manutenção da conexão. A proposta dos autores é de que o WLM seja desenvolvido como uma VNF, sendo solicitada pelo WLE que está no caminho principal dos pacotes ou roteadores. Nesse trabalho, os autores apresentam uma prova de conceito. Porém, não analisaram o desempenho dos componentes nem do mecanismo implementados.

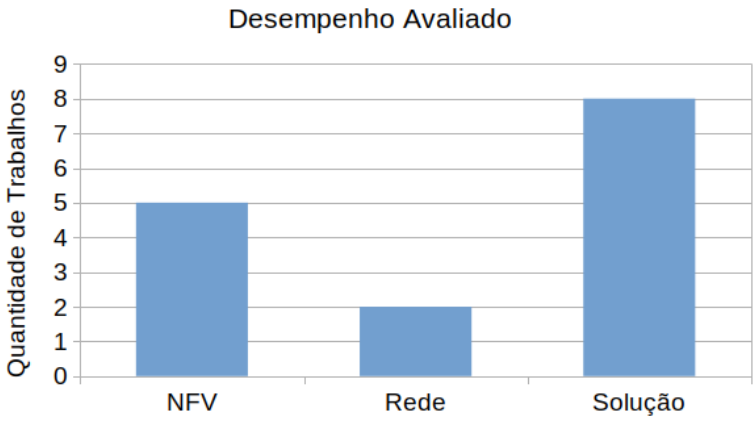
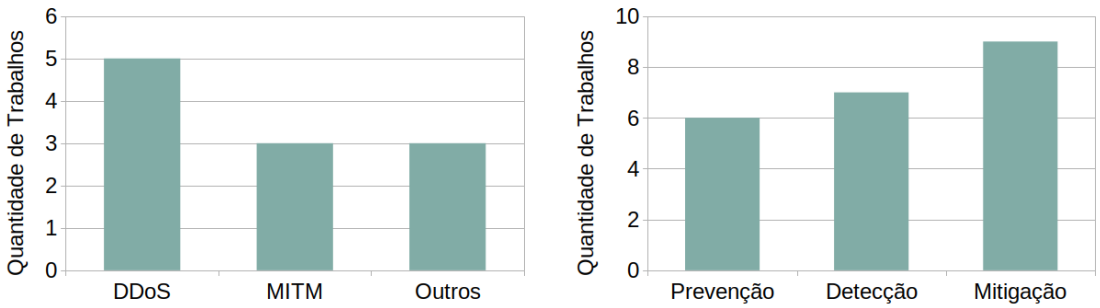


Figura 3.11. Métricas utilizadas na avaliação de desempenho.



(a) Ataque (b) Abordagem

Figura 3.12. Tipos de ataques considerados e abordagens apresentadas.

A Tabela 3.4 traz um panorama do desempenho de NFV para o contexto de segurança e a Figura 3.11 contabiliza a quantidade de trabalhos e os tipos de métricas utilizadas na avaliação de desempenho dos recursos virtualizados. É possível observar que as métricas relacionadas aos componentes implementados por meio da virtualização (NFV), como consumo de CPU, memória RAM, armazenamento, escalabilidade e tempo de implantação de ambiente, estão presentes em 50% dos trabalhos. Já as métricas relacionadas à rede (Rede), como latência, taxa de transferência e número de pacotes

processados/descartados, estão em 20% dos trabalhos. As que mais se destacam são as métricas que apresentam a eficácia da solução proposta (Solução), como tempo de resposta, de aplicação de políticas, de detecção e mitigação de ataques e acurácia do modelo de aprendizado de máquina, que estão em 80% dos trabalhos. A Figura 3.12 contabiliza os ataques e as abordagens apresentadas nos trabalhos analisados. O ataque mais utilizado para demonstrar o uso da NFV é o DDoS (55,55%), a frente do MITM (33,33%) e os demais (33,33%). Além disso, também é possível observar que 46,15% dos trabalhos levam em consideração a NFV para a prevenção de ataques na rede IoT, 53,85% para detecção e 69,23% para mitigação. Para o contexto de IoT, a Tabela 3.5 mostra que 46,15% dos trabalhos abordam as características de computação em névoa para a aplicação de NFV. No entanto, apenas [Zhou et al. 2019] apresentam uma comparação entre o desempenho de detecção e mitigação para arquiteturas distintas.

3.4. Estudo de Caso: MENTORED Testbed

O principal objetivo desta Seção é divulgar os esforços que estão sendo realizados no escopo do projeto MCTI/FAPESP MENTORED[11] em direção à criação de um ambiente controlado para experimentação em cibersegurança e introduzir a utilização de ferramentas de experimentação para avaliar soluções de redes como aquelas baseadas em NFV que foram apresentadas nas seções anteriores deste capítulo. Para este fim, apresenta-se um estudo de caso focado em uma ferramenta para detecção de *botnets* baseada em uma NFV de monitoramento de tráfego. Neste estudo de caso, empregaram-se ferramentas que permitem a reprodução do experimento conduzido pelo público. Assim, a Subseção 3.4.1 detalha o *testbed* em desenvolvimento e a Subseção 3.4.2 ilustra o emprego de VNFs em um cenário experimental.

3.4.1. MENTORED Testbed

O projeto MENTORED tem como objetivos identificar, modelar e avaliar comportamentos maliciosos associados à IoT de forma a auxiliar na construção de soluções avançadas para possibilitar: prevenção, predição, detecção e mitigação de ataques DDoS. O projeto se divide em quatro pacotes de trabalho (WPs – *Work Packages*), sendo os dois primeiros pacotes de trabalho (WP1 e WP2) dedicados à prevenção e à predição de *botnets* e ataques DDoS, respectivamente; o terceiro pacote (WP3) é dedicado à detecção e à mitigação de ataques DDoS; e o quarto pacote (WP4) foca em criar um ambiente, denominado MENTORED *Testbed*, para a experimentação de sistemas de cibersegurança.

O MENTORED *Testbed* é um ambiente acadêmico experimental, em desenvolvimento, que oferece aos pesquisadores a possibilidade de demonstrar a viabilidade de suas soluções de cibersegurança em cenários de rede com escala realista. Este ambiente vem sendo definido tomando como base a Infraestrutura Definida por *Software* da RNP (IDS-RNP) e as demandas dos outros pacotes de trabalho do projeto. A infraestrutura IDS-RNP se baseia na plataforma de orquestração de contêineres Kubernetes[12] e funciona como um ambiente externo para o desenvolvimento de projetos de rede e nuvem.

[11] `https://mentoredproject.org/`

[12] Kubernetes é uma plataforma de orquestração de contêineres que automatiza a implantação, o dimensionamento e a gestão de aplicações em contêineres. `https://kubernetes.io`.

A Figura 3.13 apresenta a visão geral dos servidores dispostos fisicamente em 13 Pontos de Presença (do inglês, *Point of Presence* - PoP) da RNP, sendo cinco servidores localizados nas regiões norte e nordeste e oito localizados nas regiões sul, sudeste e centro-oeste. Os circuitos de rede do nordeste e uma conexão com o sudeste possuem 100 Gbit/s. Os demais circuitos possuem entre 10 Gbit/s e 40 Gbit/s de capacidade. Cada nó do IDS-RNP possui um *switch* gerenciável e um servidor de virtualização equipados minimamente com dois processadores Intel® Xeon® E5-2630, 64GB de memória RAM e disco com 4TB de armazenamento.

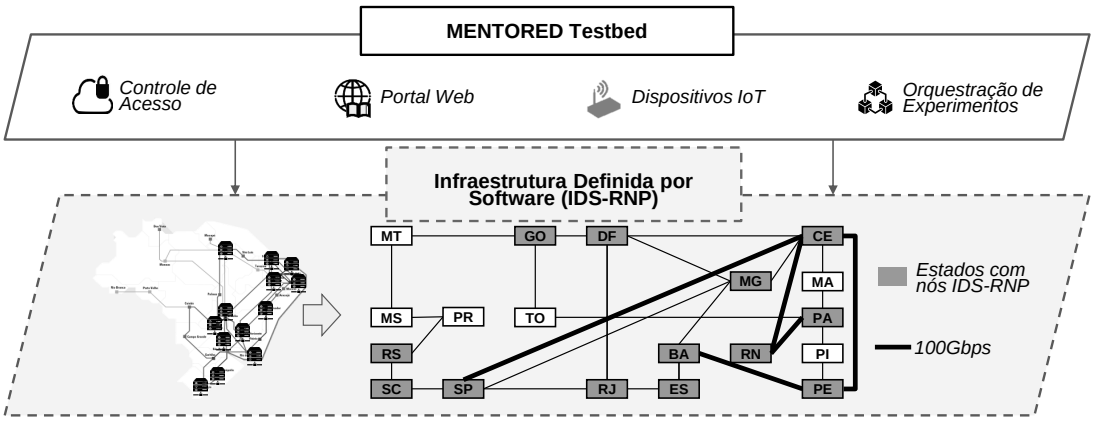


Figura 3.13. Visão geral do ambiente.

O WP4 do projeto MENTORED tem como objetivo a implantação de um ambiente usando tecnologias e metodologias avançadas para pesquisa experimental sob demanda em cibersegurança com foco em cenários realísticos da IoT. Portanto, a equipe do projeto realizou um levantamento de requisitos considerando os requisitos do *testbed* de cibersegurança DETERLab [Benzel 2011], do *testbed* de IoT FIT-IoT Lab [Adjih et al. 2015] e as definições apresentadas pelos pesquisadores dos outros pacotes de trabalho do projeto. O grupo escolheu analisar estes *testbeds*, pois estes são pioneiros em experimentação em redes com acesso aos dispositivos físicos e porque possibilitam que o próprio usuário defina o comportamento dos agentes em cada cenário experimental, diferentemente dos demais *testbeds*, que emulam os dispositivos ou são baseados em aplicações específicas da IoT. Assim, os seguintes requisitos para o MENTORED *Testbed* foram definidos:

- **Fidelidade**: se refere à capacidade de obter precisão suficiente na reprodução dos fenômenos específicos, sendo estudados em um experimento particular, dentro de um ambiente que pode ou não corresponder a qualquer rede real existente. Assim, além da representação física de uma rede real, o *testbed* pode também considerar e empregar modelos acurados para realizar avaliações fiéis;

- **Validade**: as limitações do próprio *testbed* não devem distorcer acidentalmente os resultados de um experimento. O *testbed* deve identificar e denunciar as violações dessas condições experimentais exigidas, alertando o usuário para possíveis falhas de validade do experimento;

- **Escalabilidade**: o *testbed* deve ser capaz de prover suporte a experimentos representativos para capturar efeitos complexos dos ataques relacionados ao tráfego maciço de dados na escala da Internet;
- **Segurança**: exige que o *testbed* garanta que nenhum código ou usuários maliciosos obtenham o acesso indevido ou prejudiquem outras infraestruturas de rede, informações ou códigos do próprio *testbed* ou da Internet em geral;
- **Reprodutibilidade**: garante que um experimento, uma vez executado, possa ser exportado e, em seguida, executado em um ambiente idêntico em um momento posterior, para produzir resultados idênticos;
- **Transparência**: o *testbed* deve possibilitar o monitoramento em tempo real e não intrusivo do tráfego de rede e dos recursos computacionais, além de empregar ferramentas de visualização destes recursos tanto de forma gráfica quanto pela linha de comandos;
- **Perspectiva centrada no usuário**: o *testbed* deve oferecer liberdade para os usuários desenvolverem novas classes de ferramentas que facilitam as pesquisas experimentais, além das funções tradicionais da pesquisa experimental, como a configuração de experimentos e o monitoramento do tráfego;
- **Acesso em Tempo Real**: o *software* de orquestração deve fornecer acesso em tempo real aos dispositivos, para que um usuário possa redefinir, reprogramar e monitorar o estado de cada dispositivo durante a execução dos experimentos.

Como mencionado, o MENTORED *Testbed* vem sendo construído sobre a infraestrutura IDS-RNP, a qual está fundamentada na plataforma de orquestração de contêineres Kubernetes. A plataforma Kubernetes gerencia o ciclo de vida de aplicativos e serviços em contêineres, usando métodos que fornecem monitoramento, escalabilidade e alta disponibilidade. Cada contêiner é uma área isolada dentro do sistema operacional e possui seu próprio sistema de arquivos, compartilhamento de CPU, memória e espaço de processo para executar serviços ou aplicações individualmente. Estes contêineres são logicamente alocados em *pods*, conjuntos de contêineres que compartilham a mesma área do sistema operacional e de rede. Um conjunto de *pods* se comunica por meio de uma rede e forma um *cluster*. Os servidores físicos (*Workers*) executam o sistema operacional, os *pods* e os módulos de controle. A plataforma Kubernetes emprega um módulo *Master* com uma API (do inglês, *Application Programming Interface*) que administra a comunicação de rede entre os *Workers* para criar *clusters* em múltiplos *Workers* remotos. O Kubernetes *Master* recebe *scripts* de configuração e controla localmente cada *Worker* para executar serviços e aplicações de forma portátil em *clusters*.

Os *Workers* executam três módulos locais básicos: o *Kubelet*, *Proxy* e de *Networking*. O *Kubelet* é um agente local da plataforma Kubernetes para administrar os *pods* hospedados localmente em cada *Worker*. O módulo de *Proxy* administra as regras locais de rede, intermediando as conexões e controlando os acessos. O módulo de *Networking* distribui as regras para cada *Proxy* e estabelece a comunicação entre os contêineres. Então, um usuário da plataforma Kubernetes pode definir como seus aplicativos devem ser

executados e como eles devem interagir com outros aplicativos ou com a Internet. Estas características permitem o desenvolvimento modular do *testbed*. Além disso, a plataforma Kubernetes compreende primitivas de *software* livre, para o desenvolvimento de novas versões capazes de cumprir os requisitos estudados sobre redes heterogêneas de longa distância.

O IDS-RNP auxilia na concepção do MENTORED *Testbed* ao fornecer esta estrutura de nível nacional e a equipe técnica que auxilia na implementação dos novos requisitos. Esta infraestrutura permite criar experimentos em escala com redes de longa distância, recursos computacionais e *softwares* bem conceituados tanto na academia quanto no mercado. O MENTORED *Testbed* utiliza estas características do IDS-RNP para configurar automaticamente experimentos e para testar mecanismos de cibersegurança, com cenários de rede realísticos, escaláveis e com segurança suficiente para não prejudicar a disponibilidade da infraestrutura. A Figura 3.14 ilustra a arquitetura do MENTORED *Testbed*, composta por três componentes principais: as *Ilhas MENTORED*, as *Redes Virtuais de Controle e Teste* e o *MENTORED Master*.

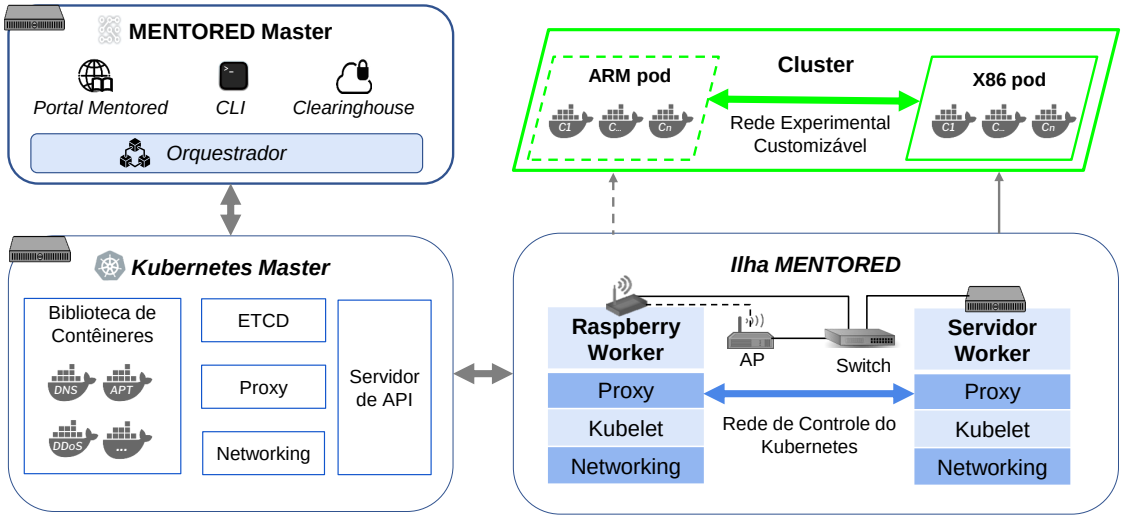


Figura 3.14. Arquitetura do MENTORED *Testbed*.

As **Ilhas MENTORED** constituem-se de ilhas do IDS-RNP que foram acrescidas de um Ponto de Acesso sem fio (AP – *Access Point*) e um conjunto de placas Raspberry PI 4 para formarem uma rede local sem fio. Esses novos dispositivos poderão ser usados para aumentar a heterogeneidade de equipamentos (ARM e X86) e meios de transmissão e, com isso, prover a fidelidade do *testbed* ao criar redes IoT. Para suportar tal heterogeneidade, utiliza-se a plataforma Kubespray[13], um módulo Kubernetes que administra *clusters* multi-arquiteturais. As placas Raspberry PI 4 também possuem conexão de rede cabeada, porém, a mesma só é usada para o tráfego de controle e gerência dos dispositivos. Ou seja, o tráfego dos experimentos só trafega pela interface de rede sem fio.

As **Redes Virtuais de Controle do Kubernetes e Experimental Customizável** servem para trafegar a comunicação entre os componentes administrativos, responsáveis

[13] `https://kubespray.io`.

pela configuração dos componentes do *testbed* e a carga de trabalho gerada pelos experimentos, respectivamente. Estas redes podem ser configuradas por diferentes módulos baseados nas CNIs (do inglês, *Conteiners Networks Interfaces*) do Kubernetes. Diversas CNIs podem ser implantadas em conjunto e permitem que o *testbed* defina os melhores módulos de rede ou que seja desenvolvido um próprio conforme as especificidades do projeto. Além disso, permite-se a criação de experimentos em redes com tecnologias heterogêneas.

O **MENTORED Master** é uma camada de *software* sobre a API do Kubernetes que administra a interação entre os quatro módulos de *software*. Ele controla os demais componentes do *testbed*: o *Portal MENTORED*, a *CH (ClearingHouse)*, a *Interface de Linha de Comandos (do inglês, Command-Line Interface - CLI*) e o *Orquestrador*. O *Portal MENTORED* e a *CH* lidam com a interação com o usuário, gestão de identidade e controle de acesso mediante uma interface gráfica para o usuário (GUI – *Graphical User Interface*). A CLI fornece comandos específicos para utilizar o orquestrador *MENTORED* e controlar os componentes do *testbed*. O *Portal MENTORED* será um site para os usuários se cadastrarem e, por meio da *CH*, adquirirem as credenciais necessárias para estabelecerem uma conexão remota e segura com a CLI. O Orquestrador faz a interface entre os demais módulos e a API do Kubernetes para fornecer as funções específicas de criação de experimentos, controle e monitoramento dos recursos. A API do Orquestrador simplifica a configuração dos cenários de rede e emprega as ferramentas necessárias automaticamente para cumprir os requisitos definidos. Assim, o usuário autenticado reserva os recursos de hardware, configura os contêineres a serem executados e, por fim, cria e submete um *script* de descrição do experimento pretendido conforme os comandos específicos da API. O Orquestrador interpreta o *script* e configura automaticamente a infraestrutura com base no Kubernetes para disponibilizar os recursos necessários e alocar os módulos para cada *namespace*.

Este conjunto de componentes é coordenado a fim de permitir que o MENTORED *Testbed* atinja os requisitos do ambiente experimental e aloque serviços pré-estabelecidos em uma biblioteca de contêineres. Então, caso um usuário precise de qualquer NFV para executar experimentos, o *testbed* possui um contêiner com um conjunto de configurações prontas para executar avaliações sobre este tipo de serviço na biblioteca. O requisito *Perspectiva Centrada no Usuário* compreende que as metodologias de pesquisa experimental são, elas mesmas, uma área de pesquisa ativa e os avanços nesta área permitem a criação de novas metodologias de pesquisa [Benzel 2011]. Neste sentido, tanto os pesquisadores do WP4, quanto os usuários gerais podem criar conjuntos de contêineres com experimentos configurados para colaborar com novas metodologias de pesquisa e com a biblioteca de contêineres.

3.4.2. Estudo de Caso

Esta Subseção detalha um estudo de caso projetado para uso no ambiente de experimentação, tendo como objetivo geral apresentar à comunidade acadêmica uma introdução na criação de cenários experimentais de redes e atrair novos pesquisadores para o desenvolvimento de metodologias de pesquisa experimentais em cibersegurança. O estudo de caso compreende operações simples de rede, que envolvem uma VNF de análise do tráfego de rede e detecção de *bots* (*i.e.*, dispositivos infectados na rede). Tanto o MENTORED *Test-*

bed quanto a IDS-RNP estão em desenvolvimento e ainda não possuem suporte para os usuários finais. Portanto, este estudo de caso é descrito de uma forma que um usuário comum possa simular um *cluster* semelhante ao do IDS-RNP e reproduzir os experimentos localmente. Os arquivos de apoio estarão disponíveis no github do projeto[14].

Avaliar propostas de detecção de *bots* de forma eficiente exige que o *testbed* possua suporte a diferentes topologias, protocolos e aplicações. Os trabalhos de detecção de intrusão geralmente monitoram pontos estratégicos do tráfego e enviam sinais customizados para a ferramenta de defesa tomar as providências necessárias. Isto ocorre principalmente quando consideram-se as redes de dispositivos IoT, que possuem desafios relacionados à heterogeneidade e ao volume de dados. O estudo de caso compreende avaliar uma ferramenta de detecção em um cenário de rede customizável e fácil de configurar.

A plataforma Kubernetes oferece um conjunto de procedimentos para o desenvolvimento de módulos que agregam funcionalidades na orquestração de contêineres. Neste contexto, os cenários criados para desenvolver este estudo de caso utilizam estes módulos de CNI para organizar a comunicação de rede entre os contêineres. A principal ferramenta considerada neste estudo para desenvolver os cenários de rede é o Knetlab[15], uma CNI desenvolvida pela equipe da RNP para simular topologias de rede entre os *pods* do Kubernetes. Além disso, a ferramenta Kind[16] monta o *cluster* Kubernetes com os servidores *Workers* e o Kubernetes *Master* conteinerizados localmente.

A Figura 3.15 ilustra a arquitetura do cenário de experimentação. Toda esta estrutura será executada sobre um computador *host*. A ferramenta Kind simula, com contêineres Docker, um *cluster* Kubernetes com múltiplos *Workers* e *Masters*. Cada *Worker* representa uma Ilha do IDS-RNP e recebe uma etiqueta de identificação baseada nas localizações do país. O Kubernetes *Master* instala os módulos de CNI como o Knetlab, o Multus e o OpenVSwitch (OVS) nos *Workers* distribuídos. O Knetlab emprega outras CNIs, como o Multus, que cria múltiplas interfaces de rede em cada *pod* e as regras OVSDB (do inglês, *Open vSwitch Database Management Protocol*), que implementam *switches* virtuais e criam *bridges* de VLANs (do inglês, *Virtual Local Area Network*) entre os *pods* ou as VXLANs (do inglês, *Virtual Extensible LAN*) entre contêineres *Masters* e *Workers*.

Esta estrutura permite criar *pods* que se comunicam através de uma rede controlada. Eles implementam VNFs sobre qualquer camada de rede. Cada *pod* na topologia customizada representa um dispositivo em rede e executa funções específicas para o experimento. Neste estudo de caso, considera-se que a rede possui duas classes de dispositivos, os dispositivos de borda e os de rede. Os dispositivos de borda atuam gerando três modelos de tráfego: normal, atacante e de servidor alvo. Os dispositivos de rede atuarão como OpenVSwitch, para o encaminhamento de tráfego, ou como uma *middlebox*, executando a VNF que pretendemos avaliar. A VNF classifica aqueles dispositivos que estiverem gerando tráfego atacante e executa o comando para bloquear este tráfego.

O Knetlab emprega o conceito de laboratórios para executar diferentes experimentos de forma isolada. Cada laboratório recebe um *namespace* e uma série de *scripts yaml*

[14] `https://github.com/mentoredtestbed`.
[15] `https://git.rnp.br/cnar/knetlab/knetlab-doc`.
[16] `https://kind.sigs.k8s.io`.

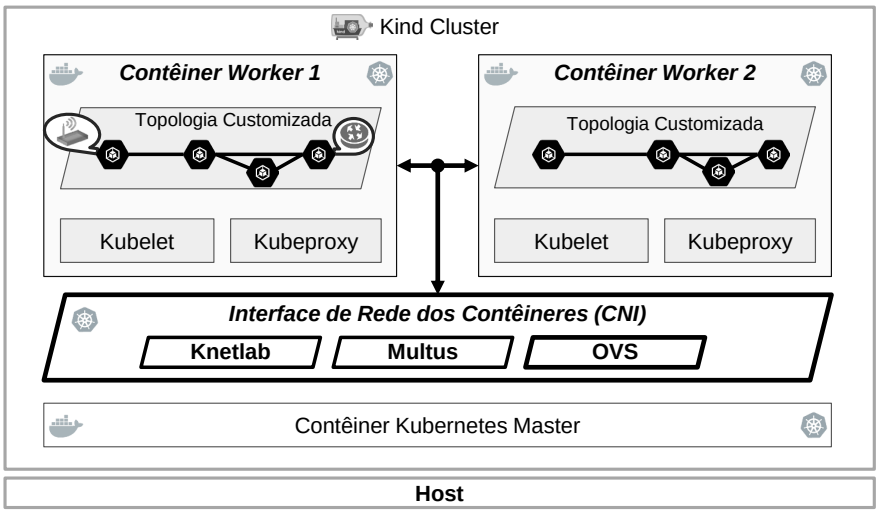


Figura 3.15. Arquitetura do experimento.

com as descrições de cada tipo de dispositivo, as configurações específicas de cada *pod* e a topologia de rede. Neste estudo de caso, o cenário inteiro pode ser criado através da execução de um *script* de inicialização, desde que no *host* esteja instalada a ferramenta de comandos da plataforma Kubernetes (*kubectl*), o Kind e o Docker.

O cenário criado automaticamente apenas inicializa as interfaces, os dispositivos e as conexões entre os *pods*. A topologia considerada neste estudo está ilustrada na Figura 3.16. Cada *Worker* pode simular uma topologia inteira sozinho, no entanto, esta topologia mostra que o Knetlab pode também usar as conexões de longa distância do IDS-RNP e suportar diferentes serviços e protocolos de rede entre as rotas virtuais.

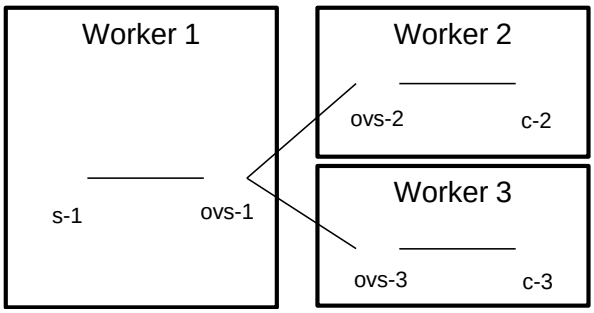


Figura 3.16. Topologia do estudo de caso.

Os *pods* que desempenham o papel de dispositivos de rede (OVS-1, OVS-2 e OVS-3) possuem um terminal linux e o OVS instalados por padrão. Os dispositivos de borda executam os comandos conforme as pré-configurações definidas para cada modelo. Portanto, os usuários podem desenvolver os experimentos como se fossem uma estrutura de rede real apenas com os cabos de rede conectando os dispositivos. Neste contexto, os *pods* contidos na topologia executam os serviços para desempenhar as funções de rede. Os *pods* podem executar contêineres específicos disponibilizados no repositório público do MENTORED *Testbed*, com exemplos práticos que executam aplicações pré-configuradas.

O *script* de configuração dos *pods* insere os recursos necessários para a execução do estudo de caso. Os *pods* que executam os dispositivos de rede recebem e instalam os *scripts* de configuração das VLANs. Os *pods* de borda executam contêineres com um servidor *web* nginx, um *script* gerador de tráfego no modo cliente e outro gerador de tráfego no modo ataque. O *switch* OVS-1 também executará o *script* representando uma VNF que monitora o tráfego a fim de detectar *botnets* e bloqueia, com um comando simples, os IPs supostamente infectados. O *script* extrai informações dos cabeçalhos dos pacotes de rede, realiza uma análise simples para determinar qual o IP responsável por gerar o tráfego atacante e envia um sinal ao OVS para bloqueá-lo - lembrando que o objetivo maior deste estudo de caso é demonstrar a metodologia de experimentação empregada para avaliar a VNF e não a solução de detecção em si.

Os *pods* também rodam um serviço agente para determinar quando o experimento será inicializado. O serviço agente envia sinais através da rede administrativa, que é configurada por padrão pela plataforma Kubernetes na interface `eth0`. Desta forma, o serviço recebe o sinal para começar e determina um horário para os demais *pods* executarem as funções especificadas e salvarem os resultados obtidos, ou seja, se a ferramenta de detecção foi eficiente e se o bloqueio da entrada do tráfego aliviou a carga no servidor.

Este cenário permite a execução de experimentos com topologias específicas e emprega VNFs em diferentes contextos de rede. Assim, cada usuário deve implementar os próprios dispositivos, topologias e serviços. O MENTORED *Testbed* visa simplificar a criação destes experimentos, alimentando uma biblioteca de cenários, que podem ser criados automaticamente e reconfigurados conforme as demandas do usuário, como foi apresentado. Desta forma, os usuários gastam menos tempo com as configurações complexas dos cenários realísticos de rede e ficam mais tempo focados no problema de pesquisa em si.

3.5. Principais Desafios e Limitações

Aprimorar os mecanismos de segurança levando em consideração a heterogeneidade de dispositivos IoT é um grande desafio a ser encarado. Dessa forma, a NFV tem se mostrado uma alternativa viável para auxiliar no cumprimento dos requisitos de segurança em redes IoT. No entanto, existem desafios e limitações em aberto que precisam de atenção para que a aplicação dessa tecnologia seja cada vez mais abrangente e eficaz.

Como visto na Seção 3.3, a definição e a aplicação de políticas de segurança na IoT têm sido uma das abordagens suportadas pela NFV. Expressar requisitos de segurança para controlar sistemas IoT distribuídos representa uma tarefa desafiadora, especialmente quando diferentes domínios administrativos e tecnológicos estão envolvidos. Isso significa que redes IoT possuem características dinâmicas que exigem que as políticas de segurança sejam estendidas, incluindo informações de contextos dinâmicos de rede. Assim, as políticas de segurança também podem depender das interações entre objetos inteligentes localizados no mesmo ambiente. Todos esses aspectos exigem capacitar os modelos de política para incluir as peculiaridades dos sistemas de IoT, bem como o potencial de novos mecanismos de segurança orientados por *software*.

A flexibilidade na seleção de mecanismos virtualizados de segurança pode afetar diretamente a qualidade dos serviços da rede, especialmente quando aplicações IoT com

requisitos rigorosos em termos de latência e confiabilidade são consideradas. Nessa linha, surgem desafios, como: quais serviços de segurança implementar, onde colocá-los e como configurá-los, resultando em um problema complexo de otimização para determinar a melhor alocação de serviços virtualizados. Além disso, três fatores podem aumentar ainda mais a complexidade da decisão. O primeiro é relacionado ao ambiente *multicloud*, ou seja, um ambiente composto por mais de um provedor de nuvem, pública ou privada, como Amazon AWS, Google GCP, Red Hat OpenStack/OpenShift, entre outros. Nesse tipo de ambiente, diversas VNFs disponibilizadas por dois ou mais provedores de nuvem podem ser selecionadas para a composição da SFC, o que aumenta as variáveis de decisão do orquestrador de VNFs. O segundo é sobre a execução em tempo real do otimizador de seleção das VNFs, uma vez que a carga de trabalho atual da rede deve ser considerada na reconfiguração dos mecanismos de segurança, especialmente para os recursos dinâmicos dos sistemas IoT. O terceiro fator são os requisitos de segurança que englobam o tempo de reação a ameaças e os custos envolvidos para a implementação de seus mecanismos.

Além da baixa latência e do aumento na largura de banda da rede, a tecnologia 5G também traz um recurso chamado de fatiamento da rede ou *network slicing* para atender, principalmente, os aspectos de redes IoT. Essa abordagem permite a criação de recursos personalizados de redes de acordo com requisitos de funcionalidade, isolamento, desempenho, entre outros. Em um contexto de NFV, para aprimorar a segurança em redes IoT, o fatiamento da rede pode ser implementado de acordo com o encadeamento de funções, levando em consideração a compensação entre a flexibilidade, desempenho e custo. O desafio aqui está relacionado à granularidade da fatia, que contém os recursos de rede, podendo ser fornecida por domínio de negócio, fluxo de tráfego, tipos de dispositivos e, até mesmo, por dispositivo unitário. Além disso, outro desafio nesse contexto é criar e gerenciar de forma dinâmica as fatias de recursos de rede.

Do ponto de vista de segurança inerente às estruturas lógicas da NFV ETSI, é possível citar o desafio do isolamento dos ambientes virtualizados sobre uma mesma infraestrutura física. Por exemplo, uma falha não corrigida de isolamento no *hypervisor* pode ser explorada por um superusuário do ambiente (*root*), possibilitando seu acesso às VNFs e explorando seus dados privados. Esse desafio pode ser ampliado quando o provisionamento de VNFs for realizado por uma entidade diferente da provedora de NFVi, por exemplo, provedores de computação em nuvem distintos. Além disso, o gerenciamento das imagens que são utilizadas para implementar as VNFs deve ser observado de forma cuidadosa. As imagens contêm informações de configuração das VNFs que podem ser interceptadas por um atacante e redefinidas para que se comportem de acordo com suas necessidades. Outro desafio é a composição de políticas de autenticação e autorização para a utilização de VNFs e suas cadeias por domínios de redes distintos e isolados, responsabilidade do componente de gerenciamento e orquestração NFV MANO (do inglês, *Network Function Virtualization Management and Orchestration*).

3.6. Considerações Finais

A introdução de recursos de rede virtualizados no ecossistema IoT traz diversas vantagens que aumentam o valor agregado da rede, principalmente pelo desacoplamento de funções de segurança de dispositivos com capacidades limitadas. Além disso, a escalabilidade é um dos aspectos mais fortes da virtualização quando comparada a recursos estáticos

em uma rede tradicional, permitindo, assim, a alocação dinâmica de funções de rede de acordo com o contexto requisitado, como o volume de tráfego, quantidade de dispositivos, ataques, entre outros. No entanto, o comportamento de uma tecnologia nova em um ambiente de redes deve ser observado para que seu desempenho não interfira de forma negativa na qualidade dos serviços prestados aos usuários finais. Este capítulo apresentou questões e o estado da arte relacionados ao uso de NFV na detecção e mitigação de ataques em redes IoT. Foram apresentados os conceitos iniciais de IoT, características relacionadas à NFV, questões de desempenho quando aplicadas à segurança de IoT, o estado da arte e um estudo de caso implementado no MENTORED *Testbed*, além dos desafios e limitações que essa tecnologia apresenta.

A análise da literatura, Seção 3.3, permite concluir que os indicadores designados para avaliar o desempenho da virtualização de NF são menos explorados que os indicadores utilizados para demonstrar a eficácia dos métodos propostos pelos diversos autores. Um exemplo claro disso é o tempo aferido para a implantação de uma ou mais máquinas virtuais que é encontrado apenas em dois dos trabalhos que analisam o desempenho da abordagem proposta. Outro ponto importante é que o desempenho da rede também é pouco explorado, mesmo em trabalhos que consideram a implementação de uma arquitetura em névoa. O desempenho da rede é observado pelos autores em apenas dois dos trabalhos analisados. Avaliar o comportamento de recursos virtualizados por meio dessas métricas é essencial, pois impacta diretamente no gerenciamento de desempenho da rede e se torna ainda mais importante quando funções de segurança são implementadas por meio de NFV.

Ademais, a NFV tem suportado a criação customizada de ambientes para a realização de simulações de ataques em redes IoT, como mostra o MENTORED *Testbed*. O estudo de caso apresentado na Seção 3.4 demonstrou na prática operações, que envolvem VNFs, em diferentes contextos de redes. Isto não só possibilita a simplificação da criação de experimentos em redes IoT, mas também a redução de custos relacionados com a implantação de ambientes dinâmicos e complexos.

Por fim, gerenciar o desempenho da rede torna-se uma atividade ainda mais complexa quando desafios como a aplicação de políticas de segurança de forma dinâmica, a seleção de serviços e onde implementá-los, a granularidade do fatiamento da rede e questões inerentes às estruturas de provisionamento de NFV são incluídos na rede IoT. Isso faz com que pesquisas relacionadas a esses contextos tendam a aumentar no decorrer do tempo, fazendo com que a virtualização de funções de rede alcance níveis de aplicabilidade cada vez mais abrangentes.

Agradecimentos

O presente trabalho foi realizado com apoio do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) e da Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP). Agradecemos as agências de pesquisa CNPq pelo auxílio financeiro 130762/2021-0 e a FAPESP pelos processos 18/23098-0, 14/50937-1 e 18/22979-2.

Referências

[Adjih et al. 2015] Adjih, C., Baccelli, E., Fleury, E., Harter, G., Mitton, N., Noel, T., Pissard-Gibollet, R., Saint-Marcel, F., Schreiner, G., Vandaele, J., and Watteyne, T. (2015). FIT IoT-LAB: A Large Scale Open Experimental IoT Testbed. In *Anais do IEEE World Forum on Internet of Things*, Milan, Italy.

[Afek et al. 2020] Afek, Y., Bremler-Barr, A., Hay, D., Goldschmidt, R., Shafir, L., Avraham, G., and Shalev, A. (2020). NFV-based IoT Security for Home Networks using MUD. In *Anais do NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium*, pages 1–9.

[Al-Shaboti et al. 2018] Al-Shaboti, M., Welch, I., Chen, A., and Mahmood, M. A. (2018). Towards Secure Smart Home IoT: Manufacturer and User Network Access Control Framework. In *Anais da 2018 IEEE 32nd International Conference on Advanced Information Networking and Applications (AINA)*, pages 892–899.

[Alam et al. 2020] Alam, I., Sharif, K., Li, F., Latif, Z., Karim, M. M., Biswas, S., Nour, B., and Wang, Y. (2020). A Survey of Network Virtualization Techniques for Internet of Things Using SDN and NFV. *ACM Computing Surveys*, 53(2):1–40.

[Bagaa et al. 2020] Bagaa, M., Taleb, T., Bernabe, J. B., and Skarmeta, A. (2020). A Machine Learning Security Framework for Iot Systems. *IEEE Access*, 8:114066–114077.

[Batista et al. 2016] Batista, D. M., Goldman, A., Hirata, R., Kon, F., Costa, F. M., and Endler, M. (2016). InterSCity: Addressing Future Internet research challenges for Smart Cities. In *Anais da 7th International Conference on the Network of the Future (NOF)*, pages 1–6.

[Benzel 2011] Benzel, T. (2011). The Science of Cyber Security Experimentation: The DETER Project. In *Anais da 27th Annual Computer Security Applications Conference*, ACSAC '11, page 137–148, New York, NY, USA. Association for Computing Machinery.

[Boudi et al. 2019] Boudi, A., Farris, I., Bagaa, M., and taleb, T. (2019). Assessing Lightweight Virtualization for Security-as-a-Service at the Network Edge. *IEICE Transactions on Communications*, E102.B(5):970–977.

[Bremler-Barr et al. 2014] Bremler-Barr, A., Harchol, Y., Hay, D., and Koral, Y. (2014). Deep Packet Inspection as a Service. In *Anais da 10th ACM International Conference on Emerging Networking Experiments and Technologies (CoNEXT '14)*, page 271–282. Association for Computing Machinery.

[Bülbül and Fischer 2020] Bülbül, N. S. and Fischer, M. (2020). SDN/NFV-based DDoS Mitigation via Pushback. In *Anais da ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, pages 1–6.

[Cao et al. 2015] Cao, L., Sharma, P., Fahmy, S., and Saxena, V. (2015). NFV-VITAL: A Framework for Characterizing the Performance of Virtual Network Functions. In *Anais da 2015 IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN)*, pages 93–99.

[CheckPoint 2020] CheckPoint (2020). Keeping the gate locked on your IoT devices: Vulnerabilities found on Amazon's Alexa. Disponível em: `https://research.checkpoint.com/2020/amazons-alexa-hacked`. Acessado em 14 de Junho de 2021.

[Chi et al. 2019] Chi, Z., Li, Y., Sun, H., Yao, Y., and Zhu, T. (2019). Concurrent Cross-Technology Communication among Heterogeneous IoT Devices. *IEEE/ACM TON*, 27(3):932–947.

[da Silva et al. 2021] da Silva, M. d. V. D., Rocha, A., Gomes, R. L., and Nogueira, M. (2021). Lightweight Data Compression for Low Energy Consumption in Industrial Internet of Things. In *Anais da IEEE 18th Annual Consumer Communications Networking Conference (CCNC)*.

[De Benedictis and Lioy 2019] De Benedictis, M. and Lioy, A. (2019). A Proposal for Trust Monitoring in a Network Functions Virtualisation Infrastructure. In *Anais da 2019 IEEE Conference on Network Softwarization (NetSoft)*.

[Deogirikar and Vidhate 2017] Deogirikar, J. and Vidhate, A. (2017). Security Attacks in IoT: A Survey. In *Anais da International Conference on IoT in Social, Mobile, Analytics and Cloud*, pages 32–37.

[ETSI 2014a] ETSI (2014a). Network Functions Virtualisation (NFV): NFV Performance & Portability Best Practises. Disponível em: `https://www.etsi.org/deliver/etsi_gs/NFV-PER/001_099/001/01.01.01_60/gs_NFV-PER001v010101p.pdf`. Acessado em 13 de Maio de 2021.

[ETSI 2014b] ETSI (2014b). Network Functions Virtualisation (NFV): Service Quality Metrics. Disponível em: `https://www.etsi.org/deliver/etsi_gs/NFV-INF/001_099/010/01.01.01_60/gs_NFV-INF010v010101p.pdf`. Acessado em 13 de Maio de 2021.

[Farris et al. 2017] Farris, I., Bernabe, J. B., Toumi, N., Garcia-Carrillo, D., Taleb, T., Skarmeta, A., and Sahlin, B. (2017). Towards provisioning of SDN/NFV-based security enablers for integrated protection of IoT systems. In *Anais da 2017 IEEE Conference on Standards for Communications and Networking (CSCN)*, pages 169–174.

[Farris et al. 2019] Farris, I., Taleb, T., Khettab, Y., and Song, J. (2019). A Survey on Emerging SDN and NFV Security Mechanisms for IoT Systems. *IEEE COMST*, 21(1):812–837.

[Gao et al. 2013] Gao, Y., Peng, Y., Xie, F., Zhao, W., Wang, D., Han, X., Lu, T., and Li, Z. (2013). Analysis of security threats and vulnerability for cyber-physical systems. In *Anais da 2013 3rd International Conference on Computer Science and Network Technology*, pages 50–55.

[Gember-Jacobson et al. 2014] Gember-Jacobson, A., Viswanathan, R., Prakash, C., Grandl, R., Khalid, J., Das, S., and Akella, A. (2014). OpenNF: Enabling Innovation in Network Function Control. In *Anais da 2014 ACM Conference on SIGCOMM*, pages 163–174.

[Giaretta et al. 2019] Giaretta, A., Dragoni, N., and Massacci, F. (2019). IoT Security Configurability with Security-by-Contract. *Sensors*, 19(19).

[Guizani and Ghafoor 2020] Guizani, N. and Ghafoor, A. (2020). A Network Function Virtualization System for Detecting Malware in Large IoT Based Networks. *IEEE Journal on Selected Areas in Communications*, 38(6):1218–1228.

[Gupta et al. 2019] Gupta, L., Salman, T., Zolanvari, M., Erbad, A., and Jain, R. (2019). Fault and Performance Management in Multi-Cloud Virtual Network Services using AI: A Tutorial and a Case Study. *Computer Networks*, 165:106950.

[Hafeez et al. 2016] Hafeez, I., Ding, A. Y., Suomalainen, L., Kirichenko, A., and Tarkoma, S. (2016). Securebox: Toward Safer and Smarter IoT Networks. In *Anais do 2016 ACM Workshop on Cloud-Assisted Networking*, pages 55–60.

[IDC 2020] IDC (2020). IoT Growth Demands Rethink of Long-Term Storage Strategies, says IDC. Disponível em: `https://www.idc.com/getdoc.jsp?containerId=prAP46737220`. Acessado em 12 de Junho de 2021.

[IETF 1998] IETF (1998). RFC 2330: Framework for IP Performance Metrics. Disponível em: `https://datatracker.ietf.org/doc/html/rfc2330`. Acessado em 13 de Maio de 2021.

[IETF 2011] IETF (2011). RFC 6390: Guidelines for Considering New Performance Metric Development. Disponível em: `https://datatracker.ietf.org/doc/html/rfc6390`. Acessado em 13 de Maio de 2021.

[IETF 2017a] IETF (2017a). Benchmarking Methodology for Virtualization Network Performance. Disponível em: `https://tools.ietf.org/id/draft-huang-bmwg-virtual-network-performance-03.html`. Acessado em 13 de Maio de 2021.

[IETF 2017b] IETF (2017b). RFC 8172: Considerations for Benchmarking Virtual Network Functions and Their Infrastructure. Disponível em: `https://datatracker.ietf.org/doc/html/rfc8172`. Acessado em 13 de Maio de 2021.

[IETF 2020] IETF (2020). RFC 8520: Manufacturer Usage Description Specification. Disponível em: `https://datatracker.ietf.org/doc/rfc8520`. Acessado em 21 de Junho de 2021.

[Jia et al. 2020] Jia, Y., Zhong, F., Alrawais, A., Gong, B., and Cheng, X. (2020). FlowGuard: An Intelligent Edge Defense Mechanism Against IoT DDoS Attacks. *IEEE Internet of Things Journal*, 7(10):9552–9562.

[Kaspersky 2019] Kaspersky (2019). Brasil é o segundo país com mais ataques a dispositivos IoT. Disponível em: `https://www.kaspersky.com.br/about/press-releases/2019_brasil-e-o-segundo-pais-com-mais-ataques-a-dispositivos-iot`. Acessado em 9 de Março de 2021.

[Kaspersky 2020] Kaspersky (2020). Quase 30% das empresas que usam IoT sofreram incidentes de segurança. Disponível em: `https://www.kaspersky.com.br/blog/empresas-iot-seguranca-dicas/14869`. Acessado em 9 de Março de 2021.

[Kim et al. 2015] Kim, T., Koo, T., and Paik, E. (2015). SDN and NFV Benchmarking for Performance and Reliability. In *Anais do 2015 17th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, pages 600–603.

[Laghrissi and Taleb 2019] Laghrissi, A. and Taleb, T. (2019). A Survey on the Placement of Virtual Resources and Virtual Network Functions. *IEEE COMST*, 21(2):1409–1434.

[Lin et al. 2017] Lin, J., Yu, W., Zhang, N., Yang, X., Zhang, H., and Zhao, W. (2017). A Survey on Internet of Things: Architecture, Enabling Technologies, Security and Privacy, and Applications. *IEEE Internet of Things Journal*, 4(5):1125–1142.

[Marku et al. 2020] Marku, E., Biczok, G., and Boyd, C. (2020). Securing Outsourced VNFs: Challenges, State of the Art, and Future Directions. *IEEE Communications Magazine*, 58(7):72–77.

[Massonet et al. 2017] Massonet, P., Deru, L., Achour, A., Dupont, S., Croisez, L.-M., Levin, A., and Villari, M. (2017). Security in Lightweight Network Function Virtualisation for Federated Cloud and IoT. In *Anais da 2017 IEEE 5th International Conference on Future Internet of Things and Cloud (FiCloud)*, pages 148–154.

[Medhat et al. 2017] Medhat, A. M., Taleb, T., Elmangoush, A., Carella, G. A., Covaci, S., and Magedanz, T. (2017). Service Function Chaining in Next Generation Networks: State of the Art and Research Challenges. *IEEE Communications Magazine*, 55(2):216–223.

[Meneghello et al. 2019] Meneghello, F., Calore, M., Zucchetto, D., Polese, M., and Zanella, A. (2019). IoT: Internet of Threats? A Survey of Practical Security Vulnerabilities in Real IoT Devices. *IEEE Internet of Things Journal*, 6(5):8182–8201.

[Mijumbi et al. 2016] Mijumbi, R., Serrat, J., Gorricho, J.-L., Bouten, N., Turck, F. D., and Boutaba, R. (2016). Network Function Virtualization: State-of-the-Art and Research Challenges. *IEEE COMST*, 18(1):236–262.

[Moghaddam et al. 2019] Moghaddam, S. K., Buyya, R., and Ramamohanarao, K. (2019). Performance-Aware Management of Cloud Resources: A Taxonomy and Future Directions. *ACM Computing Surveys*, 52(4).

[Montero and Serral-Gracià 2016] Montero, D. and Serral-Gracià, R. (2016). Offloading Personal Security Applications to the Network Edge: A Mobile User Case Scenario. In *Anais da 2016 International Wireless Communications and Mobile Computing Conference (IWCMC)*, pages 96–101.

[Montero et al. 2015] Montero, D., Yannuzzi, M., Shaw, A., Jacquin, L., Pastor, A., Serral-Gracia, R., Lioy, A., Risso, F., Basile, C., Sassu, R., Nemirovsky, M., Ciaccia, F., Georgiades, M., Charalambides, S., Kuusijarvi, J., and Bosco, F. (2015). Virtualized Security at the Network Edge: a User-Centric Approach. *IEEE Communications Magazine*, 53(4):176–186.

[Naik 2017] Naik, N. (2017). Choice of Effective Messaging Protocols for IoT Systems: MQTT, CoAP, AMQP and HTTP. In *Anais do 2017 IEEE International Systems Engineering Symposium (ISSE)*, pages 1–7.

[Neshenko et al. 2019] Neshenko, N., Bou-Harb, E., Crichigno, J., Kaddoum, G., and Ghani, N. (2019). Demystifying IoT Security: An Exhaustive Survey on IoT Vulnerabilities and a First Empirical Look on Internet-Scale IoT Exploitations. *IEEE Communications Surveys Tutorials*, 21(3):2702–2733.

[Qazi et al. 2013] Qazi, Z. A., Tu, C.-C., Chiang, L., Miao, R., Sekar, V., and Yu, M. (2013). SIMPLE-Fying Middlebox Policy Enforcement Using SDN. *SIGCOMM Comput. Commun. Rev.*, 43(4):27–38.

[Rosário et al. 2014] Rosário, D., Zhao, Z., Santos, A., Braun, T., and Cerqueira, E. (2014). A Beaconless Opportunistic Routing based on a Cross-layer Approach for Efficient Video Dissemination in Mobile Multimedia IoT Applications. *Computer Communications*, 45:21–31.

[Sairam et al. 2019] Sairam, R., Bhunia, S. S., Thangavelu, V., and Gurusamy, M. (2019). NETRA: Enhancing IoT Security Using NFV-Based Edge Traffic Analysis. *IEEE Sensors Journal*, 19(12):4660–4671.

[Schwengber et al. 2020] Schwengber, B. H., Vergütz, A., Prates, N. G., and Nogueira, M. (2020). A Method Aware of Concept Drift for Online Botnet Detection. In *Anais da 2020 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6.

[Singh et al. 2016] Singh, J., Pasquier, T., Bacon, J., Ko, H., and Eyers, D. (2016). Twenty Security Considerations for Cloud-Supported Internet of Things. *IEEE Internet of Things Journal*, 3(3):269–284.

[Sobin 2020] Sobin, C. (2020). A Survey on Architecture, Protocols and Challenges in IoT. *Wireless Personal Communications*, 112:1383–1429.

[Wang et al. 2021] Wang, T., Fan, Q., Li, X., Zhang, X., Xiong, Q., Fu, S., and Gao, M. (2021). Drl-sfcp: Adaptive service function chains placement with deep reinforcement learning. In *Anais da ICC 2021 - 2021 IEEE International Conference on Communications (ICC)*.

[Yang et al. 2017] Yang, Y., Wu, L., Yin, G., Li, L., , and Zhao, H. (2017). A Survey on Security and Privacy Issues in Internet-of-Things. *IEEE Internet of Things Journal*, 4(5):1250–1258.

[Yousefpour et al. 2019] Yousefpour, A., Fung, C., Nguyen, T., Kadiyala, K., Jalali, F., Niakanlahiji, A., Kong, J., and Jue, J. P. (2019). All one needs to know about fog computing and related edge computing paradigms: A complete survey. *Journal of Systems Architecture*, 98:289–330.

[Yu et al. 2015] Yu, T., Sekar, V., Seshan, S., Agarwal, Y., and Xu, C. (2015). Handling a Trillion (Unfixable) Flaws on a Billion Devices: Rethinking Network Security for the Internet-of-Things. In *Anais do 14th ACM Workshop on Hot Topics in Networks (HotNets-XIV)*.

[Zarca et al. 2020a] Zarca, A. M., Bagaa, M., Bernal Bernabe, J., Taleb, T., and Skarmeta, A. F. (2020a). Semantic-Aware Security Orchestration in SDN/NFV-Enabled IoT Systems. *Sensors*, 20(13).

[Zarca et al. 2018] Zarca, A. M., Bernabe, J. B., Farris, I., Khettab, Y., Taleb, T., and Skarmeta, A. (2018). Enhancing IoT security through network softwarization and virtual security appliances. *International Journal of Network Management*, 28(5):e2038. e2038 nem.2038.

[Zarca et al. 2020b] Zarca, A. M., Bernabe, J. B., Skarmeta, A., and Alcaraz Calero, J. M. (2020b). Virtual IoT HoneyNets to Mitigate Cyberattacks in SDN/NFV-Enabled IoT Networks. *IEEE Journal on Selected Areas in Communications*, 38(6):1262–1277.

[Zarca et al. 2019a] Zarca, A. M., Bernabe, J. B., Trapero, R., Rivera, D., Villalobos, J., Skarmeta, A., Bianchi, S., Zafeiropoulos, A., and Gouvas, P. (2019a). Security Management Architecture for NFV/SDN-Aware IoT Systems. *IEEE Internet of Things Journal*, 6(5):8005–8020.

[Zarca et al. 2019b] Zarca, A. M., Garcia-Carrillo, D., Bernal Bernabe, J., Ortiz, J., Marin-Perez, R., and Skarmeta, A. (2019b). Enabling Virtual AAA Management in SDN-Based IoT Networks. *Sensors*, 19(2).

[Zhang et al. 2016] Zhang, W., Hwang, J., Rajagopalan, S., Ramakrishnan, K. K., and Wood, T. (2016). Performance Management Challenges for Virtual Network Functions. In *2016 IEEE NetSoft Conference and Workshops*, pages 20–23.

[Zhou et al. 2019] Zhou, L., Guo, H., and Deng, G. (2019). A fog computing based approach to DDoS mitigation in IIoT systems. *Computers & Security*, 85:51–62.

Capítulo 4

Das Redes Vestíveis aos Sistemas Ciber-Humanos: Uma Perspectiva na Comunicação e Privacidade dos Dados

Michele Nogueira, Ligia F. Borges, Fernando Nakayama

Abstract

Cyber-physical and cyber-human systems have different characteristics from existing networks, including security properties and vulnerabilities from those found in traditional systems and networks. These systems require a high level of interoperability between devices and subsystems, predictability to allow control and they are subject to a broader range of attack models. With the evolution and popularization of cyber-physical and cyber-human systems, new security threats are emerging or increasing all the time. Data privacy and the guarantee of the availability of services need to be systematically investigated in this context. This short course contributes in this direction, attracting, at first, attention to the rapid evolution that has been taking place from the beginnings of wearable networks to cutting-edge research in nanonetworks. This rapid evolution underlies the construction of cyber-physical and cyber-human systems that have applications in several areas that will be explored in this short course.

Resumo

Os sistemas ciberfísicos e ciber-humanos possuem características diferentes das redes existentes, incluindo propriedades e vulnerabilidades de segurança diversas daquelas encontradas nos sistemas e redes tradicionais. Esses sistemas requerem um alto nível de interoperabilidade entre dispositivos e subsistemas, maior previsibilidade para permitir seu controle e estão sujeitos a um maior grupo de modelos de ataques. Com a evolução e a popularização dos sistemas ciberfísicos e ciber-humanos, novas ameaças de segurança emergem ou são potencializadas a todo momento. A privacidade dos dados e a garantia de disponibilidade dos serviços são questões que precisam ser investigadas de forma sistemática nesse contexto. Este minicurso contribui nesta direção atraindo, em um primeiro momento, a atenção para a rápida evolução que vem ocorrendo desde os primórdios das redes vestíveis até a pesquisa de ponta em nanorredes. Essa evolução rápida fundamenta a construção dos sistemas ciberfísicos e ciber-humanos que possuem aplicações em diversas áreas, exploradas neste minicurso.

4.1. Introdução

Diante da crise gerada pelo novo coronavírus, o relatório de Agosto de 2020 da empresa californiana *Global Industry Analysts, Inc.*, especializada em prover pesquisa de mercado, revisou as estimativas para o mercado global de sensores vestíveis, trazendo uma projeção de US$ 2,5 bilhões de dólares americanos para o período de 2020 a 2027, ou seja, um crescimento de 25,2% [Analysts 2020]. As projeções de mercado refletem tendências de uso de dispositivos que integram tais sensores. A medida que a popularidade e a confiança do usuário em dispositivos dotados desses sensores aumentam, ocorrerão novos e variados vetores de ataques direcionados a invasões de privacidade, roubos de dados e negação de serviços [Ashibani and Mahmoud 2017, Yaacoub et al. 2020]. Esses sensores, posicionados dentro e fora do corpo, coletam dados de atividades dos usuários e do ambiente, permitindo a identificação de informações sensíveis por técnicas de inferência [Dong et al. 2019, Trimananda et al. 2020, Al-Shawabka et al. 2020].

Em todos esses casos, é de extrema importância garantir a disponibilidade, a privacidade e a transmissão confidencial dos dados do usuário, do ambiente e das aplicações a um repositório centralizado de dados/nuvem de processamento [Datta et al. 2018, Hafeez et al. 2020, Nakayama et al. 2019, Nogueira et al. 2009]. Por exemplo, diferentes tipos de sensores geram tráfego de dados característico, que pode ser detectado no meio de outras transmissões e usado para estimar o tipo específico de monitoramento realizado em um determinado indivíduo e outras informações importantes [Dong et al. 2019, Trimananda et al. 2020, Al-Shawabka et al. 2020, Prates et al. 2020, Tahaei et al. 2020]. Esse tipo de vazamento de privacidade do usuário ocorre por meio da análise do tráfego de dados gerado pelos sensores. Esses vazamentos de informação e a disponibilidade dos serviços precisam ser abordados de forma sistemática e com profundidade, diferente do que ocorre em geral, em que essas questões são exploradas de forma *ad hoc* por técnicas clássicas para a Internet e sem a preocupação de construir um embasamento científico para entender as ameaças contra a privacidade dos dados e a negação de serviços.

Os sistemas ciber-físicos (*Cyber-Physical Systems* – CPS) e ciber-humanos (*Cyber-Human Systems* – CHS) evoluem rapidamente, dando suporte à conexão de dispositivos heterogêneos, vestíveis, implantáveis no corpo humano (na escala nano, por exemplo) ou no ambiente. Os sistemas ciber-físicos e ciber-humanos tendem a transformar a maneira como interagimos com o mundo físico ao nosso redor em diferentes setores, como transporte, manufatura, agricultura, saúde, energia, defesa, aeroespaço e construções [Rajkumar et al. 2010, Song et al. 2016]. O projeto, a implementação e a verificação de sistemas ciber-físicos e ciber-humanos apresentam uma infinidade de desafios técnicos que devem ser tratados por uma comunidade multidisciplinar de pesquisadores e educadores [Song et al. 2016]. Esses sistemas expandem o conceito de Internet das Coisas (IoT), pois são projetados considerando como parte nata a integração e a comunicação do ambiente físico ou humano com o ciberespaço [Yaacoub et al. 2020, Sharma et al. 2020].

Os sistemas CPS e CHS possuem características diferentes das redes existentes, incluindo propriedades e vulnerabilidades de segurança diversas daquelas encontradas nos sistemas e redes tradicionais [Ashibani and Mahmoud 2017, Ding et al. 2018]. Esses sistemas requerem um alto nível de interoperabilidade entre dispositivos e subsistemas, maior previsibilidade para permitir seu controle e estão sujeitos a um maior grupo de

modelos de ataques. Com a evolução e a popularização dos CFS e CHS, novas ameaças de segurança emergem ou são potencializadas a todo momento. A privacidade dos dados e a garantia de disponibilidade dos serviços são questões que precisam ser investigadas de forma sistemática nesse contexto, considerando a reprodução dos experimentos e a identificação de seus limites fundamentais.

Este capítulo de livro contribui nesta direção atraindo, em um primeiro momento, a atenção para a rápida evolução que vem ocorrendo desde os primórdios das redes vestíveis até a pesquisa de ponta em nanorredes. Essa evolução rápida fundamenta a construção dos sistemas ciber-físicos e ciber-humanos que possuem aplicações em diversas áreas que serão exploradas neste minicurso. A forte aplicabilidade dessas redes, em conjunto com regulações mundiais relacionadas à proteção dos dados, nos remetem à necessidade de explorar as vulnerabilidades e desafios que essas redes possuem em relação à privacidade dos dados e resiliência de seus serviços. Assim, trazemos uma discussão neste sentido, demonstrando através de exemplos práticos essas fragilidades e também um levantamento do estado da arte de propostas acadêmicas para a proteção da privacidade e resiliência de seus serviços. Por fim, serão ressaltadas as perspectivas futuras associadas aos desafios e questões de pesquisa em aberto.

4.2. Fundamentação

Esta seção apresenta uma síntese do contexto histórico e evolutivo das redes vestíveis rumo aos sistemas ciber-físicos e ciber-humanos, a motivação para esses sistemas e uma visão geral. Esta seção se organiza nas seguintes subseções: (i) contexto histórico, (ii) conceitos importantes e (iii) formas de comunicação e tecnologias. Esta estrutura foi concebida a fim de prover um entendimento amplo do tema e suportar ideias em prol a evoluções no tema.

4.2.1. Contexto histórico

Resgatando o contexto histórico, consideramos justo mencionar que a proposição de equipamentos vestíveis iniciou-se no século XIII com a invenção dos óculos de grau com o objetivo claro de melhorar a visão dos usuários. A maioria dos dispositivos atuais são considerados inteligentes por terem a capacidade de processar dados e acessar a Internet. Entretanto, dispositivos que trazem uma melhor experiência aos usuários também podem ser considerados inteligentes. Sendo assim, os óculos de grau podem ser considerados um caso precoce de dispositivo vestível inteligente [Ometov et al. 2021]. A linha evolutiva até a aparição dos dispositivos computacionais vestíveis compreende uma série de pequenos avanços em conceitos e equipamentos, dentre os quais podemos citar os relógios de bolso, ábacos portáteis, rádios de comunicação, câmeras portáteis, relógios de pulso, entre outros. A Figura 4.1 ilustra os marcos históricos que contribuíram para o desenvolvimento das redes vestíveis e das nanorredes.

O primeiro exemplo de dispositivo vestível com poder computacional foi concebido por Edward Thorp em 1955. O dispositivo era um pouco maior que uma caixa de fósforos e podia ser furtivamente posicionado na sola de um sapato com o objetivo de predizer os resultados da roleta de um casino [Thorp 1998]. Nos anos seguintes, surgiram os relógios com calculadora e os sistemas de áudio portáteis. Aos poucos, a evolução

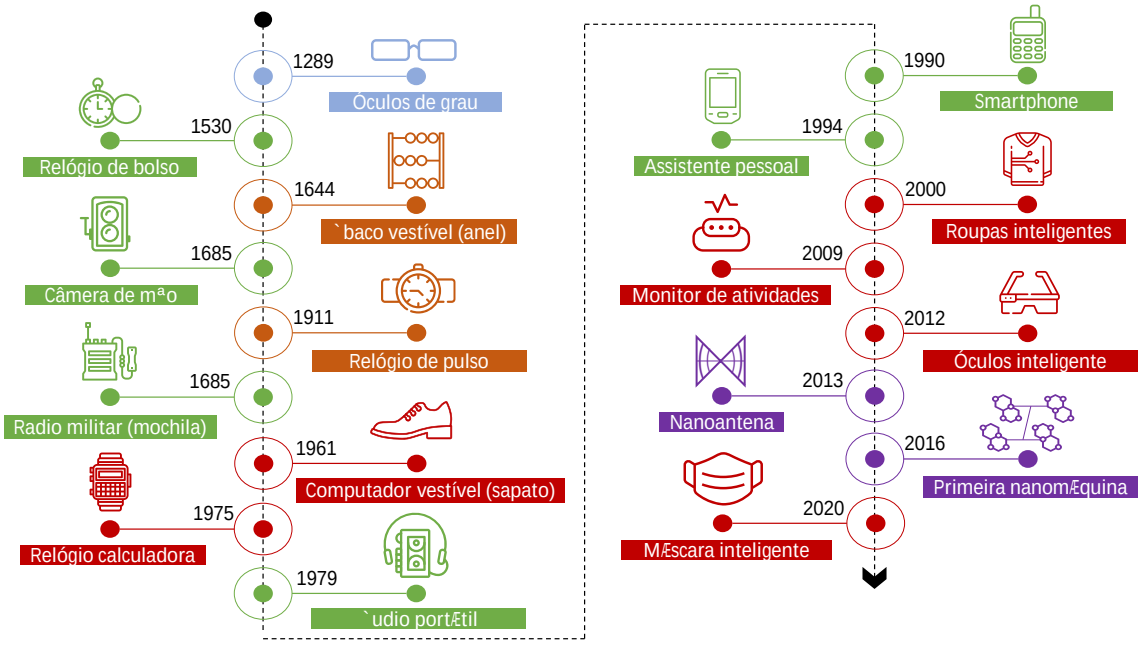


Figura 4.1: Marcos históricos na evolução das redes vestíveis e nanorredes

tecnológica e os avanços nos processos de engenharia permitiram que novos dispositivos fossem criados e se tornassem parte do nosso cotidiano. Hoje, a utilização dos dispositivos portáteis e vestíveis existentes no mercado se tornou tão corriqueira que quase não se nota a importância dos seus predecessores no contexto histórico da evolução dos vestíveis.

Os dispositivos computacionais portáteis, como assistentes pessoais digitais, sistemas de localização e sistemas de pagamento móveis, pavimentaram o caminho para o surgimento de novos conceitos. Seguindo a linha evolutiva dos dispositivos computacionais portáteis e vestíveis podemos citar os *smartphones*, relógios inteligentes, monitores de atividades físicas e roupas inteligentes. Esses novos equipamentos e tecnologias apoiados nos novos patamares de portabilidade e usabilidade oferecem suporte aos sistemas ciber-físicos e ciber-humanos.

A evolução dos dispositivos computacionais segue uma clara tendência rumo a dispositivos mais compactos e portáteis. A Figura 4.2 representa essa evolução histórica considerando os principais marcos na evolução dos dispositivos computacionais comerciais. Os primeiros computadores disponíveis eram grandes em tamanho, ocupavam um grande volume de espaço e tinham alto custo de aquisição e operacional. Sendo assim, seu uso era restrito às grandes empresas e corporações. Com o advento de tecnologias como o circuito integrado o processo de miniaturização dos dispositivos ganhou força e os computadores se tornaram menores, mais baratos, e mais disponíveis[Moore 1998]. As redes vestíveis e em escala nano representam o presente e futuro dessa evolução.

Os sistemas ciber-físicos representam a integração de processos computacionais, processos relacionados às redes de comunicação e processos físicos. Embora grande parte dos sistemas computacionais tenha o foco em pessoas utilizando os sistemas, muitos sistemas complexos são uma combinação de computadores e pessoas em busca de um objetivo comum. As pessoas são uma parte primordial desses sistemas, uma vez que elas intera-

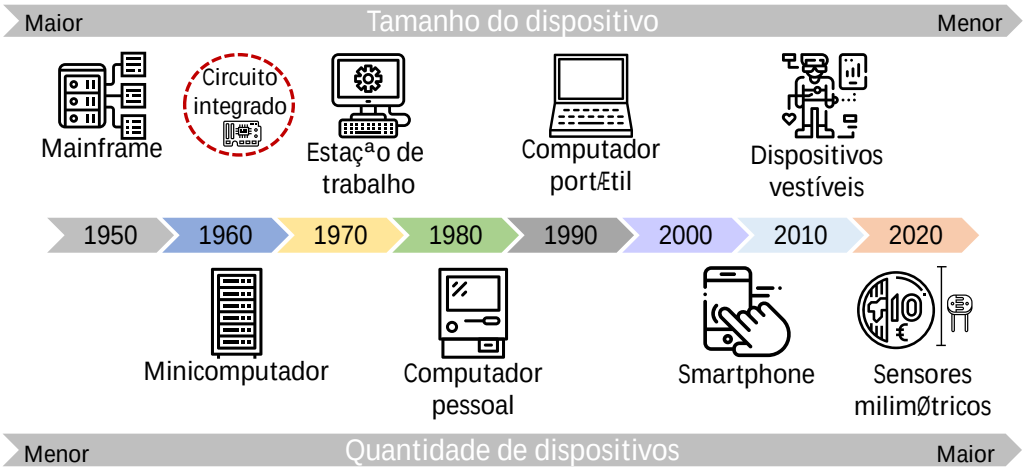


Figura 4.2: Linha do tempo da miniaturização dos computadores

gem ativamente e também fazem parte dos mesmos. Sendo assim, os componentes dos sistemas ciber-humanos trabalham em conjunto para atingir os objetivos estipulados para o sistema, que na maioria dos casos estão alinhados com os objetivos humanos. Nesse contexto, os dispositivos vestíveis (ou apenas "vestíveis") desempenham um papel fundamental e atuam como facilitadores na construção do ambiente ciber-humano. Através dos dispositivos e das redes vestíveis, os usuários interagem de forma fácil, rápida e fluida com os outros componentes do sistema e com o sistema ciber-humano como um todo.

Os dispositivos vestíveis têm atraído a atenção da comunidade acadêmica e da indústria, aumentando sua popularidade por diferentes aplicações. Em Janeiro de 2021, a empresa Gartner publicou as principais tendências tecnológicas para o próximo ciclo de ápice [IDTechEX 2021]. Dentre elas, estão as associadas com os ambientes inteligentes com uma maior integração entre os dispositivos, os usuários e os negócios através de dispositivos vestíveis capazes de coletar e fornecer informações, de modo transparente, sobre os usuários e criar condições para maior qualidade de vida. Por dispositivos vestíveis, entendem-se aqueles usados, acoplados ou implantados no corpo humano para monitorar continuamente as atividades de um indivíduo, sem interromper ou limitar os movimentos [Gao et al. 2016].

Os dispositivos vestíveis disponíveis atualmente são um notável avanço em relação às gerações anteriores, mesmo assim ainda estão em uma fase embrionária de desenvolvimento. Muitos usuários sentem a necessidade de combinar dispositivos, um rastreador de atividades físicas e um *smartphone* por exemplo, pois nenhum deles oferece todas as características desejadas como a coleta constante de sinais vitais, bateria de longa duração, alto poder de processamento, entre outras. De fato, o ecossistema dos equipamentos vestíveis compreende uma grande variedade de dispositivos que isoladamente ou de forma integrada podem oferecer uma qualidade de experiência sem precedentes. Sendo assim, os dispositivos vestíveis, mesmo com algumas restrições, têm o potencial de mudar a vida dos usuários da mesma forma que os *smartphones* fizeram nos últimos anos. Isso também se refletirá nos sistemas sem fio de telemetria médica, que serão compostos por diversos sensores e dispositivos em micro e nanoescala que serão implantados no corpo humano para fornecer um monitoramento mais sensível da saúde, detecção e tratamento de doenças em tempo real.

Os principais fatores que moldaram a atual geração de dispositivos vestíveis foram a miniaturização dos componentes que permitiu avanços na portabilidade, o avanço das redes de comunicação sem fio, os avanços na computação baseada em eficiência energética, e os avanços nos sensores e telas empregados nos vestíveis. Mesmo com melhorias notáveis nas tecnologias envolvidas na construção dos dispositivos, alguns temas continuam sendo foco de pesquisa da indústria e da comunidade acadêmica. Os dispositivos vestíveis dependem de baterias e/ou processadores de baixa capacidade que requerem ajustes e otimizações tanto em nível de hardware quanto em nível de software. É importante observar que as pesquisas relacionadas aos dispositivos vestíveis podem impactar beneficamente algumas áreas relacionadas, como o desenvolvimento de novos protocolos de comunicação sem fio e melhorias nos protocolos existentes, além de avanços relacionados à eficiência energética.

4.2.2. Conceitos

Ao observar a evolução e miniaturização dos dispositivos computacionais, nota-se que os atuais dispositivos de pequeno porte dotados apenas de sensores e sistemas embarcados desempenham funções equivalentes a dispositivos de grande porte das gerações anteriores. Aliados a outros equipamentos de comunicação e sensoriamento esses dispositivos têm a capacidade de monitorar as condições de uma pessoa ou de um ambiente, e interagir com o mundo físico através de sinais visuais, atuadores, ou robôs, por exemplo. Entretanto, as ações que acontecem no mundo virtual carecem de uma maior integração com as possíveis respostas apresentadas no mundo físico. O termo ciber-físico foi cunhado por Raj Rajkumar, professor da Universidade *Carnegie Mellon* nos Estados Unidos da América, enquanto ele realizava uma apresentação. A definição apresentada e consolidada desde então é de um sistema que combine os aspectos computacionais, de comunicação e de armazenamento de dados, com o objetivo de monitorar e, eventualmente, controlar as entidades que existem no mundo físico [Rajkumar et al. 2010].

O sistemas ciber-humanos possuem as mesmas bases fundamentais dos sistemas compostos de dispositivos ciber-físicos (sensores, atuadores, dispositivos, tecnologias de comunicação, armazenamento de dados, etc). Entretanto, nos sistemas ciber-humanos as pessoas também desempenham funções primordiais. Algumas vezes o papel das pessoas é ativo e exige interação direta com os outros componentes do sistema. Em outras, o papel é passivo como por exemplo quando se analisa o comportamento humano. Em ambas situações as pessoas são peça chave para o correto funcionamento do sistema, seja atuando como operadores ou como parte passiva do sistema. Um dos conceitos que ilustra os sistemas ciber-humanos é o de "pessoas como sensores", onde as observações de um sistema são feitas não somente por sensores e dispositivos mas também pelas observações subjetivas dos seres humanos [Resch 2013].

É possível diferenciar uma pessoa participante do sistema ciber-humano de um sistema ciber-físico tradicional através de três características: (i) cognição, (ii) previsibilidade e (iii) motivação [Sowe et al. 2016]. A cognição humana é extremamente diferente dos dispositivos ciber-físicos que empregam sensores e processadores. Seres humanos utilizam a função cerebral, a visão e a audição como forma de cognição. Essas diferenças na forma de interpretar o ambiente que nos cerca representam os desafios e as oportunidades na integração homem-máquina. As pessoas são mais imprevisíveis que os dispositivos

computacionais, mesmo ao realizar uma mesma tarefa de forma repetitiva. Entretanto, as pessoas têm mais facilidade de se adaptar a situações de mudança constante e podem oferecer soluções inovadoras. Finalmente, as pessoas necessitam de algum tipo de motivação ou compensação para executar uma função.

A atual geração de dispositivos computacionais portáteis incorpora uma categoria específica de equipamentos que beneficiam intensamente os sistemas ciber-humanos, os dispositivos computacionais vestíveis. Os vestíveis são dispositivos computacionais posicionados sobre uma parte do corpo humano ou implantados no mesmo. Geralmente, eles combinam tecnologias permitindo a visualização de informações através de telas, a entrada de dados através de toque ou voz, a transferência de dados através de tecnologias de comunicação, baterias internas, e o processamento e armazenamento de dados. Os vestíveis buscam atender a dois objetivos principais: (i) a mobilidade, (ii) e a sensibilidade ao contexto. Por serem extremamente portáteis, eles estão sempre com o usuário e acompanham-o em qualquer situação, integrando diversos sensores (ex. sistema de posicionamento global, giroscópio, monitores de sinais vitais) e capazes de interagir com o ambiente e com o usuário.

As funcionalidades dos dispositivos vestíveis dependem dos sensores integrados a eles e do posicionamento do mesmo no corpo humano. A Figura 4.3 ilustra as possibilidades de posicionamento dos vestíveis no corpo humano. Os dispositivos vestíveis podem ser posicionados na cabeça, região do tronco, mãos, braços e pernas, além de poderem ser implantados em diferentes partes do corpo. As principais funcionalidades são auxiliar a localização através do sistema de posicionamento global, a medição de sinais vitais (variação cardíaca, oxigenação do sangue, etc), auxílios visuais (realidade aumentada), e a troca de informações através de tecnologias de comunicação. Os dispositivos vestíveis têm a capacidade de se comunicar entre si e/ou com *gateways* utilizando tecnologias de comunicação sem fio. Os dispositivos vestíveis conectados entre si e com os demais componentes de uma rede ciber-humana formam uma rede vestível. As redes vestíveis são usualmente compostas de múltiplos dispositivos, gerando um grande volume de dados.

Particularmente no domínio da saúde, as redes IoT e vestíveis impulsionaram o conceito da Internet das coisas da saúde, do inglês *Internet of Health Things* (IoHT). IoHT refere-se a uma infraestrutura composta de dispositivos relacionados à saúde conectados entre si e/ou à Internet. A IoHT revolucionou a medicina e a saúde, fornecendo acesso fácil a dados relacionados à saúde para usuários e profissionais da área e abrindo um mundo de possibilidades. Ela oferece a comunicação entre dispositivos de saúde e a *Internet*, a coleta contínua de características pessoais e do ambiente, a possibilidade de atendimento remoto e orientações de sintomas e tratamentos, proporcionando aos pacientes um maior controle sobre suas vidas. Existe um grande número de aplicações IoHT voltadas para a medicina inteligente, como a supervisão contínua do tratamento do câncer e monitoramento de glicose, a automação da administração de insulina, o controle dos sintomas da asma entre outros. A IoHT ganha gradualmente a atenção da academia e da indústria que, motivada pelo envelhecimento da população, muda a prestação de cuidados ao paciente de dentro das instituições tradicionais para um atendimento contínuo em qualquer lugar e a qualquer hora.

Os dados transmitidos através de uma rede vestível são de extrema importância

Figura 4.3: Posicionamento dos dispositivos vestíveis no corpo humano

pois representam caraterísticas pessoais de um usuário como: quem ele é, sua localização, e seu estado de saúde. Sendo assim, é essencial que se adote medidas de segurança para assegurar a privacidade dessas informações. Entretanto, os vestíveis possuem particularidades que dificultam a implementação de mecanismos de segurança tradicionais como criptografia complexa e assinaturas e chaves digitais. Dentre os fatores limitantes os mais evidentes são o poder de processamento moderado, a baixa largura de banda nos canais de comunicação, e restrições no armazenamento das informações. Adicionalmente, as redes vestíveis utilizam tecnologias de comunicação sem fio, tornando-as mais expostas a ameaças. Sendo assim, adotar mecanismos de segurança em ambientes ciber-humanos é uma tarefa complexa e um problema a ser confrontado.

Os mecanismos de segurança computacional visam fornecer simultaneamente a disponibilidade no uso dos dispositivos para usuários autorizados, a confidencialidade e a integridade dos dados disseminados na rede vestível [Laprie et al. 2004]. O vazamento de dados pessoais é uma das principais preocupações no tocante à segurança nas redes vestíveis. Estudos apontam que grande parte dos consumidores não se sentem à vontade ao compartilhar dados pessoais nem em receber informações privadas de colegas ou parentes [Perez and Zeadally 2017]. As principais preocupações associadas à quebra da privacidade em redes vestíveis estão relacionadas ao uso indevido dos dados e suas implicações, como a divulgação de informações em redes sociais, o uso das informações para extorsão ou ameaças, e a divulgação de dados não autorizada por parte de prestadores de serviço. Outro aspecto de segurança relevante é a resiliência na coleta e na transmissão das informações. As redes vestíveis fornecem suporte a aplicações que envolvem rico de vida, como o monitoramento de pacientes em tempo real, tornando-se essencial um ambiente capaz de lidar com eventuais problemas de sensoriamento e comunicação.

Os dispositivos intracorporais são compostos por componentes biológicos e eletrônicos com dimensões de milímetros a escala micrométrica/nanométrica. São denominados de nanomáquinas, ou seja, pequenos componentes (0.1-10 micrômetros) consistindo

em um conjunto de moléculas organizadas capazes de realizar tarefas de computação, detecção e/ou atuação muito simples. As nanomáquinas naturais são amplamente encontradas na natureza e são responsáveis pela contração muscular, pela locomoção das bactérias e dos espermatozoides, pela divisão celular, pela replicação de DNA e outros. As nanomáquinas sintéticas são produzidas pelo homem e já são uma realidade. Recentemente, Sir Fraser Stoddart (vencedor do prêmio Nobel de química, 2016) apresentou uma nanomáquina com quatro motores moleculares que desempenham o papel de rodas e são capazes de realizar movimentos unidirecionais quando expostas à luz ultravioleta [Richard Van Noorden 2016].

A biologia sintética é o campo de pesquisa interdisciplinar da engenharia abrindo um caminho viável para a realização prática de sistemas ciber-físicos com capacidade de comunicação molecular através da programação do código genético das células [Akyildiz et al. 2019]. Por meio da biologia sintética, os engenheiros têm projetado bio-dispositivos em micro e nanoescala baseados em células que são semelhantes aos dispositivos eletrônicos convencionais [Koucheryavy et al. 2021]. Uma rede/nanorrede surge da comunicação entre esses dispositivos em micro/nanoescala baseados em células por meio de sistemas de comunicação molecular, i.e., sistemas que utilizam moléculas para codificar a informação. A comunicação intencional e controlada de dados entre as nanomáquinas e bio-dispositivos é essencial para viabilizar o trabalho coordenado e a realização de tarefas mais complexas, pois os dispositivos miniaturizados possuem restrições computacionais de processamento, armazenamento de dados e energia.

4.2.3. Formas de comunicação e tecnologias

Os sistemas ciber-humanos abrangem três tipos fundamentais de comunicação e suas tecnologias: (i) intracorporal, (ii) curta distância, (iii) longa distância. A Figura 4.4 ilustra os tipos de comunicação existentes. Os sistemas de comunicação intracorporais compreendem um conjunto de dispositivos e sensores que atuam em áreas especificas do corpo humano. Para se comunicar com redes externas, incluindo a Internet, esses sistemas utilizarão um dispositivo tradutor (ou seja, interface bio-cibernética) que converterá qualquer sinal biológico em sinal elétrico aplicado para redes de computadores de nível macro. O dispositivo tradutor pode se comunicar com os vestíveis a partir de tecnologias de comunicação de curta distância (Bluetooth, NFC, Wi-Fi). Essas tecnologias são empregadas na comunicação entre os vestíveis e também para acessar dispositivos coordenadores ou *gateways* de acesso à Internet. As tecnologias de acesso de longa distância (banda larga fixa, rede celular) permitem que as informações obtidas a partir dos dispositivos nanoescala ou vestíveis cheguem até a Internet.

De forma geral, os sistemas de comunicação (em nano, micro e macro escalas) seguem os passos de codificação dos dados no sinal, transmissão, propagação do sinal, recepção do sinal e decodificação dos dados (mensagem). Na comunicação intracoporal a codificação e a transmissão são realizadas pelo nó transmissor (nanomáquina, biodispositivo ou nanorrobô) e a recepção e decodificação realizadas pelo nó receptor (nanomáquina, bio-dispositivo ou nanorrobô). A propagação do sinal ocorre no canal de comunicação. O canal de comunicação das redes internas é o corpo humano (sangue, células, órgãos, entre outros) que faz o papel de caminho físico em que o sinal se propaga entre os nós transmissores e receptores. Na literatura existem três principais paradigmas

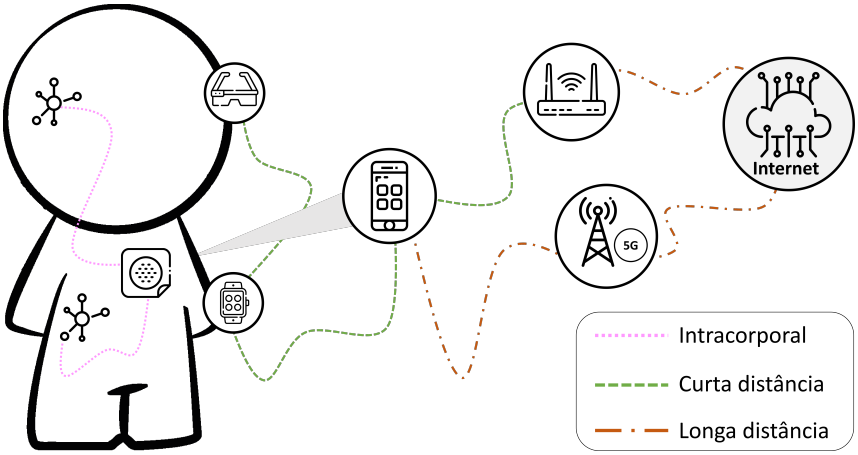


Figura 4.4: Tipos de comunicação em sistemas ciber-humanos

de comunicação em micro e nanoescala que foram consideradas para fornecer a interconexão entre dispositivos em uma rede intracorporal: comunicação acústica, comunicação eletromagnética, e comunicação molecular.

Na comunicação acústica, utiliza-se a variação de pressão para transmitir informações. As ondas acústicas são caracterizadas pela propagação em diferentes tipos de ambientes com bom desempenho, principalmente em comparação com as ondas eletromagnéticas. O atrativo para o uso intracorporal desse tipo de comunicação é a alta eficiência dessas ondas ao penetrarem tecidos e organismos biológicos [Hogg and Freitas Jr 2012]. A literatura atual das nanorredes baseadas em ondas acústicas focam principalmente no desenvolvimento de dispositivos acústicos como sensores, geradores e detectores de som habilitados para serem construídos com nanomateriais leves [Ding et al. 2019, Tao et al. 2020]. Embora muitos instrumentos de diagnóstico sejam baseados em microondas, que são consideradas não perigosas para os seres humanos [Loscri and Vegni 2015], o aquecimento causado pelas ondas ultrassônicas no tecido deve ser profundamente investigado para a viabilidade da comunicação acústica interna. As variações de pressão da onda de ultrassom causam bolhas no meio de propagação que podem atingir valores elevados e causar efeitos biológicos indesejáveis. Como o efeito é um fenômeno dependente da frequência de oscilações e pressão, o ciclo de trabalho e frequência devem ser investigados para definir valores seguros [Santagati and Melodia 2014].

As nanorredes baseadas em ondas eletromagnéticas estão fundamentadas na transmissão e recepção de radiação eletromagnética a partir de nanodispositivos (*i.e.*, nanoantenas, nanorrádio, nanotransceptor eletromecânico) compostos por novos nanomateriais como nanotubos de carbono e nanofitas de grafeno [Pfeiffer et al. 2018]. Estudos já provaram que antenas de grafeno podem funcionar na banda Terahertz (THz) [Da Costa et al. 2009]. Dessa forma, acredita-se que os nanodispositivos se comunicarão potencialmente entre 0,1-10 THz. Entretanto, os efeitos da radiação eletromagnética no tecido vivo pode ser considerado um problema na implementação dessa comunicação em sistemas ciber-físicos. Mesmo sob o atual limite de segurança de 1 mW/cm^2 os efeitos sobre a radiação THz dentro do corpo humano ainda é uma

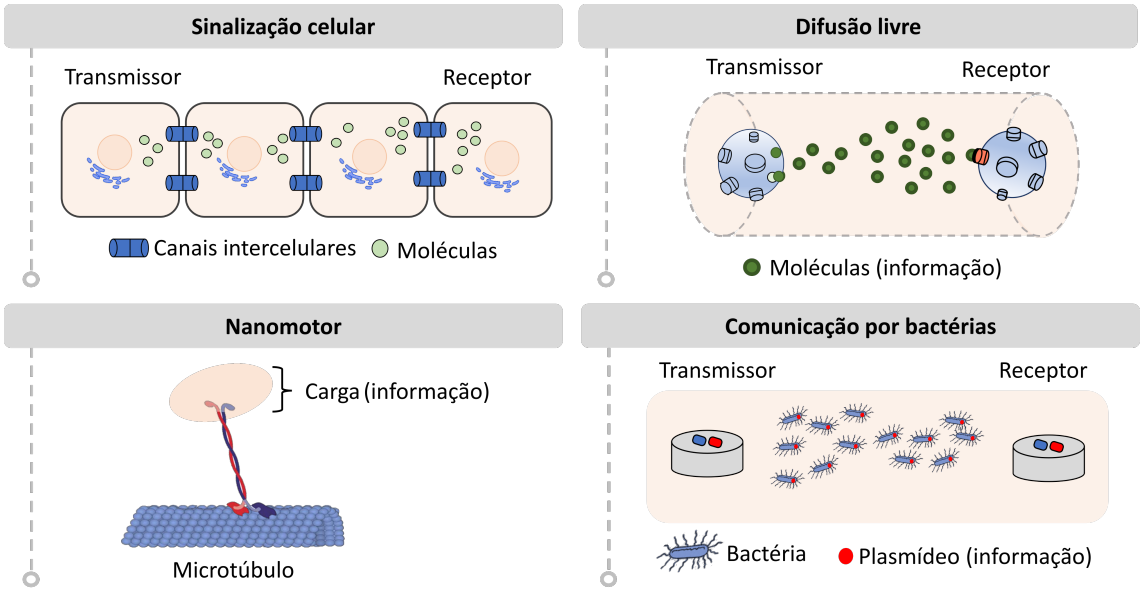


Figura 4.5: Representação das comunicações moleculares

questão em aberto [Mcguiness et al. 2019]. Existem análises experimentais em outras espécies que mostram danos no DNA sob a influência de curto prazo da radiação THz [Titova et al. 2013, De Amicis et al. 2015]. Contudo, sabe-se atualmente que o corpo humano tem adaptações contra a radiação eletromagnética se a fonte de radiação estiver fora do corpo [Jablonski and Chaplin 2010].

As redes baseadas em comunicação molecular (CM) são diretamente inspiradas por sistemas de comunicação entre entidades presentes na natureza e seus sistemas biológicos. Para projetar sistemas de CM sintética a comunidade de pesquisa adotou elementos distintos de processos bioquímicos e os resumiu em modelos matemáticos. Duas classes principais de esquemas de codificação de informação são considerados. Um depende do tipo de partícula empregada (por exemplo, neurotransmissores, mensageiros intracelulares, moléculas, bactérias) e o outro depende da maneira como as partículas são propagadas no meio (por exemplo, difusão livre e sinalização célula-célula). Entre os vários mecanismos diferentes que foram considerados para a comunicação molecular estão: (*i*) comunicações por difusão molecular como a sinalização celular e a difusão livre que codificam a informação em moléculas; (*ii*) nanorredes baseadas em bactérias (a informação é codificada no plasmídeo da bactéria); (*iii*) nanorredes por motores moleculares que transportam nanopartículas de informação (Figura 4.5).

A difusão é uma das principais formas de transporte de materiais que as células usam para receber componentes que darão suporte a sua funcionalidade e sobrevivência. O modelo de canal de comunicação mais estudado em CM são os baseado em difusão. Nessa comunicação os bio-dispositivos que fazem o papel de nós transmissores e receptores são células hibridas, ou seja, que possuem partes orgânicas e circuitos eletrônicos. A difusão por sinalização celular (sinalização por íons de cálcio, potássio entre outros) reside em uma comunicação célula-célula geralmente mediada por junções comunicantes (canais que ligam células adjacentes). Após uma reação química (isto é, reação-difusão)

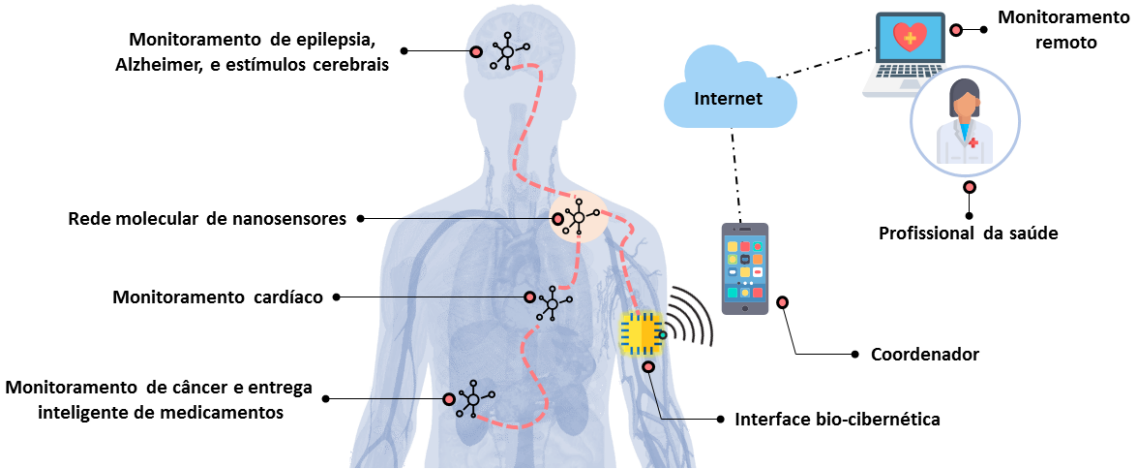


Figura 4.6: Representação das comunicações moleculares

que regula a amplificação de moléculas, os sinais são transmitidos ao longo das membranas celulares e geram uma determinada reação (*e.g.*, duplicação celular, liberação de hormônios, entre outros) na célula emissora ou as células imediatamente adjacentes.

Na difusão livre as moléculas se propagam livremente pelo espaço entre as nanomáquinas transmissora e receptora executando o movimento Browniano (aleatório). A informação é codificada na concentração e/ou tipo de molécula. O canal de comunicação dentro do corpo é por exemplo a corrente sanguínea e os neurônios. Nas comunicações moleculares baseadas em bactérias a informação é codificada no plasmídeo da bactéria. A nanomáquina receptora libera no canal um componente para atrair as bactérias que se locomovem no fluido em movimento aleatório entre o transmissor e receptor. Nas comunicações moleculares baseadas em nanomotores um nanomotor carrega uma grande partícula por um filamentos de microtúbulos até o seu receptor. Um nanomotor é uma classe de máquina molecular capaz de converter uma fonte de energia aplicada sobre ele (por exemplo moléculas de trifosfato de adenosina - ATP) em energia mecânica para realizar um trabalho específico como andar e girar.

A Figura 4.6 representa a variedade de sistemas de comunicação que podem ser implementadas dentro do corpo humano e interconectadas por meio de interfaces biocibernéticas. A interface bio-cibernética troca dados com o coordenador que possui maior capacidade computacional que por suas vez interage com a Internet para permitir o monitoramento e controle remoto dos dispositivos intracorporais. As comunicações moleculares baseadas em sinalização celular sintética ao se conectarem com tecidos celulares naturais através de suas junções comunicantes podem monitorar e tratar patologias que ocorrem em tecidos e órgãos. As comunicações baseadas em difusão livre são uma opção para transportar as informações das redes de sinalização celular até a interface biocibernética através da corrente sanguínea. As ondas acústicas podem ser utilizadas para ativar dispositivos de estimulação cerebral ou uma população de bactérias a produzem sinais de detecção de quorum que podem afetar as populações bacterianas naturais (como por exemplo as bactérias das paredes do intestino) ou geneticamente modificadas.

Além da evolução computacional que impacta no tamanho e no poder de proces-

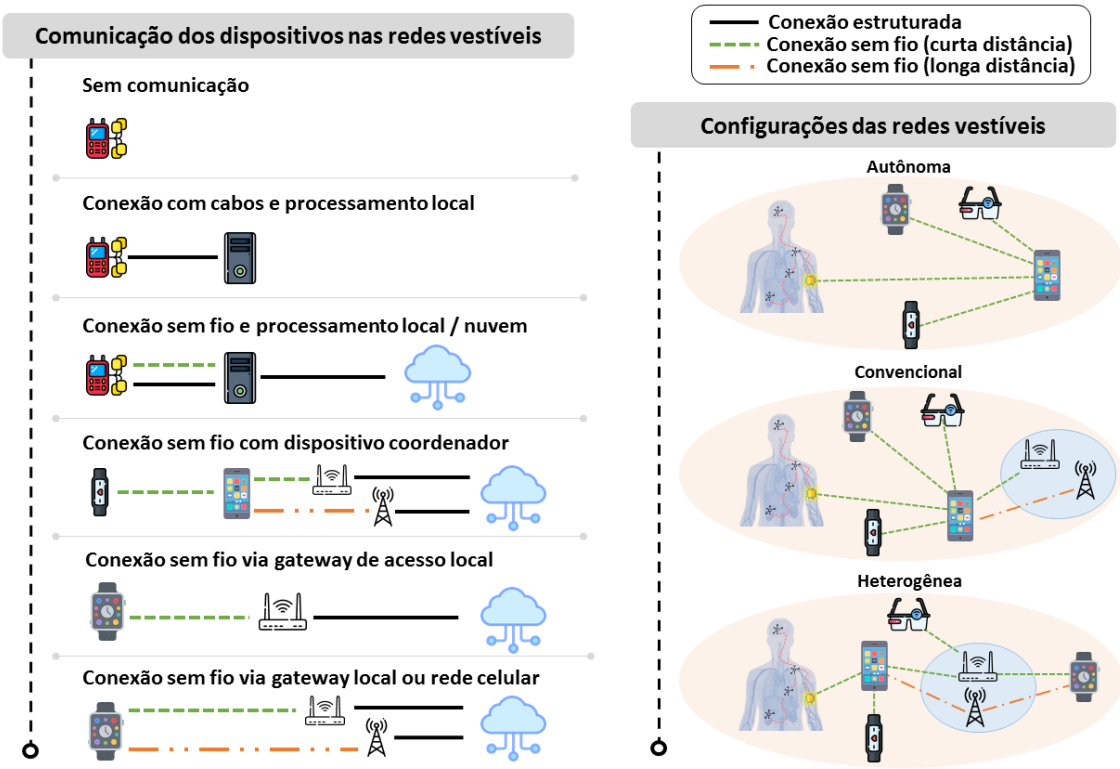


Figura 4.7: Comunicação dos dispositivos e evolução das redes vestíveis

samento dos dispositivos, a evolução nas tecnologias de comunicação sem fio também proporciona novas perspectivas para as redes vestíveis. Essa evolução beneficia a mobilidade das pessoas, trazendo maior praticidade e qualidade de vida ao usuário e permitindo o uso dos dispositivos em diversos cenários. As novas gerações de dispositivos vestíveis incorporam as tecnologias de comunicação emergentes, mas ao mesmo tempo precisam ser compatíveis com dispositivos e equipamentos de comunicação existentes. A variedade de dispositivos e tecnologias de comunicação proporcionam diversas formas comunicação entre os vestíveis e configurações para as redes. A Figura 4.7 apresenta os principais tipos de comunicação existentes para os vestíveis e as configurações para as redes vestíveis.

Os primeiros dispositivos vestíveis eram desprovidos de comunicação externa, cabendo ao próprio dispositivo coletar os dados, realizar o processamento, e exibir as informações. Em uma segunda etapa da evolução, os dispositivos eram conectados a computadores com maior capacidade de processamento. Entretanto, a comunicação era realizada através de cabos o que limitava a movimentação dos usuários. Com o avanço nas tecnologias de comunicação sem fio os vestíveis se libertaram do uso dos cabos para comunicação, ao mesmo tempo, o conceito de processamento em nuvem começava a se estabelecer. O processamento distribuído é importante para os sistemas ciber-humanos devido às limitações no poder de processamento de alguns vestíveis. Novas tecnologias de comunicação sem fio de curta distância como Bluetooth e NFC gradualmente foram incorporadas aos vestíveis, trazendo novas possibilidades de comunicação e uma grande diversidade de configurações para as redes vestíveis.

Considerando a diversidade de dispositivos e tecnologias de comunicação existem três tipos principais de configurações para as redes vestíveis: (i) autônoma, (ii) convenci-

onal e (iii) heterogênea. As redes vestíveis autônomas concentram todas as funcionalidades de sensoriamento, processamento e exibição das informações nos próprios vestíveis participantes da rede. Essa configuração descarta o processamento e o armazenamento em nuvem. Eventualmente, emprega-se um dispositivo com maior poder de processamento como coordenador da rede vestível, um *smartphone*, por exemplo. O coordenador executa as funções complexas que exigem maior poder computacional e pode transferir dados a partir de múltiplas tecnologias de comunicação e com diversos dispositivos simultaneamente. Nas redes vestíveis convencionais, os dispositivos se comunicam entre si e com o coordenador, com a possibilidade de acesso à Internet a partir do coordenador através de um *gateway* sem fio local ou uma rede celular. Essa configuração é a mais comum atualmente e representa uma evolução sobre as redes vestíveis autônomas impulsionada pela popularização das tecnologias de comunicação sem fio.

A próxima geração de dispositivos vestíveis fortalecerá ainda mais o conceito de computação ubíqua aplicada aos sistemas ciber-humanos. Beneficiando-se novamente dos avanços tecnológicos, os novos dispositivos terão a capacidade se comunicar através de múltiplas tecnologias de comunicação, sem a necessidade de um dispositivo coordenador. De fato, alguns vestíveis já possuem essas características. Podemos citar como exemplo alguns relógios inteligentes que tem a capacidade de se comunicar com outros dispositivos usando *bluetooth*, realizar pagamentos através de comunicação NFC, se conectar à Internet através de um *gateway* Wi-Fi estático ou em qualquer lugar através de redes celulares. A tendência é que os novos dispositivos vestíveis sejam capazes de enviar e receber dados através de tecnologias de comunicação variadas, fortalecendo a mobilidade, o processamento distribuído, e a diversidade de aplicações rumo a sistemas ciber-humanos ainda mais integrados.

A vasta gama de vestíveis e tecnologias de comunicação existentes oferecem suporte a diferentes soluções de conectividade. A tecnologia de comunicação deve se ajustar aos requisitos do vestível levando em consideração os aspectos desejados como: alcance, largura de banda, eficiência energética, possibilidades de configurações, entre outros. A definição da tecnologia de comunicação também estabelece outros atributos importantes, entre eles os tipos de criptografia aceitos, os esquemas de codificação e transmissão de dados, e os esquemas de correção de erros. Muitas aplicações das redes vestíveis dependem dessa variedade de tecnologias de comunicação para funcionar corretamente, entretanto, o maior alcance das redes e problemas de compatibilidade entre as tecnologias podem tornar árdua a efetivação de mecanismos de segurança e resiliência. Atualmente, as tecnologias mais utilizadas em redes vestíveis incluem a *Near Field Communication* (NFC), *Radio Frequency Identification* (RFID), *Bluetooth Low Energy* (BLE), *Wireless Fidelity* (Wi-Fi), redes de longo alcance e baixo custo energético, do inglês *Low Power Wide Area Networks* (LPWAN), e outras tecnologias de transmissão de dados.

A grande parte dos dispositivos de uma rede vestível e, principalmente, os vestíveis menores em tamanho e capacidade computacional, se comunica entre si de maneira ponto-a-ponto, do inglês *peer-to-peer* (P2P). O Bluetooth é uma tecnologia de comunicação sem fio para redes pessoais, do inglês *Personal Area Networks* (PAN). Em sua versão atual que visa o baixo consumo de energia (BLE), essa tecnologia atinge uma distância de transmissão de até 100 metros em campo aberto e uma taxa de transmissão de dados de até 1Mbps. Para isso, ela utiliza a frequência de 2.4 GHz não licenciada da banda

industrial, científica e médica (ISM). O padrão de conexão mais usual em redes Bluetooth é chamado Piconet e compreende até oito dispositivos conectados em topologia estrela, sendo um mestre e os demais escravos [Ferro and Potorti 2005]. Dentro dessa topologia o dispositivo mestre representa o coordenador da rede vestível que se conecta a um *gateway*, via Wi-Fi por exemplo, e oferece acesso à nuvem aos vestíveis conectados a ele.

O conjunto de padrões IEEE 802.11 desempenha um papel fundamental nas redes vestíveis, pois provê a conexão entre os dispositivos vestíveis capacitados e um *gateway* local sem fio, comercialmente esse padrão é conhecido como Wi-Fi. Através dos padrões 802.11, os dispositivos acessam a Internet com velocidades que podem chegar a 9.6 Gbps no Wi-Fi versão 6. Os dispositivos podem se conectar a um ponto de acesso, do inglês *Access Point* (AP), ou em modo ad hoc, e a distância de transmissão chega a 250 metros, dependendo da versão em uso. Umas das principais vantagens do Wi-Fi é estar presente nativamente nos principais dispositivos que podem ser aplicados como coordenadores de uma rede vestível. Entretanto, o alto consumo de energia é um fator impactante [Losilla et al. 2011].

A tecnologia Zigbee é uma das tecnologias de comunicação sem fio que oferece suporte a dispositivos com baterias de baixa capacidade. Em modo de repouso, os dispositivos que utilizam Zigbee podem se manter operacionais por longos períodos de tempo. A tecnologia Zigbee cobre distâncias de até 100 metros e utiliza a frequência de 2.4 Ghz, podendo sofrer interferência da tecnologia Bluetooth e algumas versões do Wi-Fi. A principal limitação da Zigbee é a baixa taxa de transferência de dados, apenas 250Kbps, o que pode limitar seu uso em aplicações de tempo real e baixa latência [Negra et al. 2016]. Por esse motivo, emprega-se a tecnologia Zigbee em poucos dispositivos de uma rede vestível e geralmente para funções que não requerem largura de banda elevada.

As tecnologias de comunicação aplicadas em redes vestíveis que possuem o menor alcance são a RFID/NFC. As etiquetas RFID possuem uma microantena e um microchip de memória que ao serem energizadas por um leitor disponibilizam um conjunto de informações armazenadas. A tecnologia NFC é uma versão mais recente e aprimorada da RFID com a capacidade de comunicação unilateral ou bidirecional. Um vestível equipado com NFC tem a capacidade de ler informações e também enviar informações para outros dispositivos conectados. A comunicação NFC é ponto-a-ponto e compreende uma distância de apenas 20 centímetros, sendo eficiente e segura para troca de informações por aproximação, pagamentos por exemplo. A taxa de transferência para tecnologia NFC corresponde a até 424 Kbps [Poongodi et al. 2020].

As tecnologias de comunicação de curta distância promovem um uso mais eficiente das baterias e são empregadas em dispositivos mais baratos e simples. Entretanto, as tecnologias de comunicação de longa distância habilitam os dispositivos vestíveis a um número maior de oportunidades e aumentam o leque de possíveis aplicações. Através dessas tecnologias, os vestíveis se comunicam com um ponto de acesso remoto. Alguns vestíveis disponíveis no mercado estão equipados com módulos de identificação de assinante, do inglês *subscriber identification module* (SIM). Utilizando um cartão SIM de uma operadora de serviços de telecomunicação o usuário tem a possibilidade de conectar o vestível diretamente à Internet, dispensando a obrigatoriedade de um coordenador.

O aumento na quantidade de dispositivos aptos a utilizarem as redes de longa dis-

tância faz com que novas soluções de comunicação sejam propostas. Entre elas está a comunicação tipo máquina, do inglês *Machine Type Comunication* (MTC) e sua variação para quantidade massiva de dispositivos mMTC. O objetivo da mMTC é oferecer comunicação de longa distância e baixo custo energético para dispositivos de baixa complexidade. Essa tecnologia impulsiona as redes IoT de banda estreita, do inglês*Narrowband Internet of Things* (NBIoT) e as redes de evolução de longo prazo, do inglês *Long Term Evolution* (LTE)-M. As especificações para essas redes preveem a transmissão de uma pequena quantidade de dados enviados e baixa periodicidade no envio, com mínimo uso de energia e grandes áreas de cobertura. Tecnologias mais antigas e com motivações iniciais que não previam os vestíveis também são alvo de avaliação e adaptações para possibilitar sua integração às redes vestíveis.

As redes de baixo consumo energético e longa distância, do inglês *Low-Power Wide Area* (LPWA), são uma realidade em cidades inteligentes. Inicialmente, elas visavam integrar dispositivos IoT de baixa complexidade como sensores de monitoramento e automação. Entretanto, as soluções oferecidas pela tecnologia LPWA atraíram a atenção para outras aplicações. Um exemplo de tecnologia LPWA não licenciada é o protocolo de longo alcance LoRa. É uma tecnologia sem fio de longo alcance que opera na banda isenta de licença da ISM e utiliza a frequência de 868MHz, cobrindo distâncias de até 25 km. Essa tecnologia está presente fundamentalmente em aplicações da saúde, como monitoramento de temperatura e variação cardíaca através de vestíveis [Ometov et al. 2021]. Outra tecnologia LPWA alternativa é utilizada pela empresa Sigfox em seus dispositivos vestíveis proprietários. A Sigfox opera na frequência sub-GHz da ISM e oferece longo alcance dentro da área de cobertura contratada e mantida pela própria empresa.

A Tabela 4.1 detalha as tecnologias de comunicação de curta e longa distância que são empregadas em redes sem fio para oferecer suporte às aplicações vestíveis. A configuração mais utilizada nas redes vestíveis ainda emprega um coordenador para atuar como retransmissor das informações coletadas nos vestíveis, apoiando-se nas tecnologias de comunicação de curta distância. Entretanto, a próxima geração de redes sem fio promete oferecer aos vestíveis a possibilidade de estabelecer comunicação direta com a nuvem. Outras formas de oferecer acesso à nuvem para os dispositivos vestíveis também são alvo de estudo, ainda que dependam de tecnologias que não estão amplamente disponíveis comercialmente. Dentre as formas de comunicação estão as redes IoT com comunicação satelital e comunicação dispositivo-a-dispositivo, do inglês *device-to-device* (D2D).

A comunicação D2D é alvo de pesquisas em redes vestíveis principalmente com o objetivo de aumentar a capacidade das redes sem fio em termos de número de dispositivos e área de cobertura. Na comunicação D2D os dispositivos se comunicam entre si, respeitando as métricas de segurança previamente estipuladas. Uma das formas de integrar os vestíveis através da comunicação D2D é utilizar a comunicação social. Apoiada na comunicação D2D, a sinergia entre as redes sociais e a IoT tornou possível o conceito de redes das coisas sociais, do inglês *Social Internet of Things* (SIoT). Na SIoT os dispositivos podem socializar, colaborar, e estabelecer níveis de comunicação de acordo com as relações sociais do usuário. O aumento da área de cobertura proporcionado pela comunicação D2D e a integração dos dispositivos através da SIoT cria novas possibilidades, como por exemplo o gerenciamento do processamento e armazenamento das informações de forma distribuída entre os dispositivos participantes da rede.

	Tecnologia	Bandas de frequência	Alcance	Taxa de transmissão	Uso de energia
Curta distância	RFID	125 - 134 kHz, 13.56 MHz, 860 - 960 MHz	Até 100m	Depende da frequência	Muito baixo
	NFC	13.56 MHz	<0.2 m	Até 424 kbps	Muito baixo
	BLE (802.15.1)	2.4 - 2.48 GHz	Até 100m	Até 24 Mbps	Baixo
	Zigbee (802.15.4)	868 - 868.6 MHz, 902 - 928 MHz, 2.4 - 2.49 GHz	Até 100m	Depende da frequência	Muito baixo
	Wi-Fi (802.11a/b/g/n)	2.4 - 2.48 GHz, 4.9 - 5.8 GHz	20-250 m	2-600 Mbps	Médio
	Wi-Fi 5 (802.11ac)	4.9 - 5.8 GHz	Até 70m	Até 3.5 Gbps	Alto
	Wi-Fi 6 (802.11ax)	1 - 6 GHz	Até 120m	Até 9.6 Gbps	Alto
Longa distância	NB-IoT	Frequências da LTE	Até 15Km	Até 250 kbps	Baixo
	LTE-M	Frequências da LTE	Até 10Km	Até 1 Mbps	Baixo
	LoRa	867 - 869 MHz	Até 25Km	50 kbps	Muito baixo
	Sigfox	868–878.6 MHz	Até 40Km	100 bps	Muito baixo

Tabela 4.1: Tecnologias de comunicação sem fio empregadas em redes vestíveis

A comunicação via satélite auxiliando os serviços IoT também é alvo de pesquisas tanto da comunidade acadêmica quanto da indústria. Dentro do contexto das redes vestíveis, a comunicação via satélite poderia desempenhar o papel de rede de acesso. Isso significa que atualmente os dispositivos vestíveis não acessam diretamente as redes satelitais. Entretanto, em muitos casos esse é o único tipo de comunicação disponível em regiões muito distantes ou de difícil acesso. Nesse cenário, as redes de comunicação por satélite podem oferecer acesso à nuvem para uma rede vestível através de gateways de comunicação específicos. Recentemente, a empresa Starlink iniciou a comercialização de kits portáteis de acesso à Internet via satélite, com uma cobertura que abrange quase todo o território dos Estados Unidos da América.

4.3. Aplicações

Esta seção apresenta as propostas recentes de aplicações para os seguintes setores da sociedade: tecnologia da informação, medicina, e neurociência. Os sistemas ciber-físicos oferecem suporte a uma vasta gama de aplicações em múltiplas áreas do conhecimento. As aplicações são usualmente centradas no usuário, utilizando dados relacionados à saúde, atividades realizadas, localização e segurança [Dhanvijay and Patil 2019]. A Tabela 4.2 ilustra as principais aplicações para os sistemas ciber-humanos.

Sistemas Ciber-humanos	
Aplicações médicas	**Aplicações não médicas**
Monitoramento de sinais vitais	Ambientes inteligentes
Monitoramento de atividades	Indústria 4.0
Detecção / Controle de doenças	Entretenimento
Telemedicina	Segurança
Sistemas de suporte à vida	
Monitoramento do sono	
Entrega inteligente de medicamentos	

Tabela 4.2: Aplicações para os sistemas ciber-humanos

Os sistemas ciber-humanos oferecem suporte a uma variedade de aplicações. Entretanto, as características dos dispositivos utilizados nesses sistemas (portabilidade, proximidade com o corpo, comunicação com outros dispositivos e com a Internet), favorecem vigorosamente as aplicações médicas. A Tabela 4.2 especifica as principais aplicações para os sistemas ciber-humanos considerando as aplicações médicas e não médicas. Em geral, as aplicações médicas dos sistemas ciber-humanos envolvem o monitoramento de funções vitais de uma pessoa e o uso das informações para uma tomada de decisão por parte do próprio usuário, de um profissional da saúde local ou remotamente, ou ainda de um agente ciber-físico que interaja com o usuário. Existem ainda aplicações não médicas para os sistemas ciber-humanos que visam melhorar a qualidade de vida, aumentar a segurança, e promover a integração social e humano-computador utilizando o usuário como parte passiva do sistema ou como operador do mesmo.

Os sistemas ciber-físicos baseados em comunicações moleculares prometem ser uma área que reúne tanto a engenharia eletromecânica, como também a nanotecnologia, biotecnologia e biologia sintética para uma reformulação das práticas médicas. Com o objetivo de trazer a personalização e a integração de diversas formas de monitoramento da saúde, diagnose e tratamento de doenças, essa tecnologia de comunicação permite a implementação de redes de sensores orientadas para a saúde em tempo real. Os dados provenientes das redes internas podem ser remotamente observados por médicos por meio dos sistemas ciber-humanos apoiando o tratamento de doenças crônicas, bem como a detecção precoce de câncer e novos vírus. Uma das aplicações que chama muita atenção para a comunicação molecular (CM) em medicina é a entrega inteligente de medicamentos, do inglês *Intelligent Drug Delivery*), que consiste em controlar adaptativamente a taxa e frequência de medicamento a entrar no corpo humano, dependendo de que local deseja-se obter uma maior concentração do medicamento [Chahibi et al. 2013].

Pesquisadores investigaram os sistemas ciber-físicos para a entrega de medicamentos. Os trabalhos analisaram, por exemplo, como as nanomáquinas podem ser removidas do corpo humano após concluída a etapa de medicação [Chude-Okonkwo et al. 2017] e o desenvolvimento de nanopropulsores para realizar a quimioterapia segura ao liberar o medicamente somente no local do tumor [Jia et al. 2019]. Em [Ye et al. 2007], um experimento com dispositivos implantáveis para a entrega remota de medicamentos às células foi conduzido. Os dispositivos são controlados remotamente por radiofrequência para manipular o microambiente químico e biológico. Em [Mcguiness et al. 2019], um protocolo de comunicação para sistema ciber-humano que utiliza o sistema circulatório (sanguíneo) como uma topologia em anel foi apresentado. Três nanorrobôs são projetados para essa topologia, sendo um para carregar o medicamento, outro nanorrobô estático com papel de nó transmissor e receptor com funções de recupera/injeta o medicamento e trocar moléculas de informação e, por fim, o terceiro nanorrobô *gateway* para monitoramento interno (*i.e.*, verificação de status do nó e contagem de nanorrobôs que transportam o medicamento no sistema circulatório) e envia as informação utilizando ondas eletromagnéticas para dispositivos externos nas proximidades, por exemplo, alertando quando a quantidade de medicamento é insuficiente.

Recentemente, a comunidade de pesquisa foca também em trabalhos que apliquem conceitos e sistemas ciber-humanos para novas tecnologias que ajudem nas doenças cerebrais como neurodegeneração. Unidades de controle que são capazes de tradu-

zir os sinais naturais do cérebro em comandos de máquina, ou seja, interfaces cérebro-máquina foram desenvolvidas e propostas como um novo método de tratamento ou terapia para doenças neurais e problemas de habilidades cognitivas [Koucheryavy et al. 2021]. Em [Wirdatmadja et al. 2017], um método de estimulação cerebral profunda foi proposto. O dispositivo externo se comunica por meio de ondas eletromagnéticas com o dispositivo subdural que, por sua vez, se comunica com os dispositivos *neural dusts i.e.*, nanodispositivo que possui um LED, um circuito retificador e um conjunto de nanofios para ativar o LED pela conversão eletromecânica do balanço dos nanofios (ultrassom). Esse sistema ciber-humano permite um controle externo das estimulações cerebrais, o que torna essa técnica mais atrativa pelo fato de ser menos invasiva e não exige cirurgia quando é implantada. As técnicas de neuroestimulação usando interfaces cérebro-máquina são uma opção de tratamento para pacientes que sofrem de graves déficits no controle motor ou enfrentam dificuldades para manipular objetos usando próteses.

As interfaces cérebro-máquina também são aplicadas a diagnósticos médicos e terapêuticas. Por exemplo, um microssistema baseado em eletrocorticografia, uma técnica para o registo gráfico das ondas elétricas cerebrais com elétrodos aplicados sobre o córtex cerebral foi presentada em [Muller et al. 2014]. O sistema consiste em um arranjo de eletrodos de 64 canais e uma antena flexível. Este sistema é adequado para o monitoramento de longo prazo devido ao uso de materiais biocompatíveis e pelo fato do implante ser minimamente invasivo não havendo a necessidade de perfuração. Em [Kim et al. 2016], um sistema semelhante de sonda neural sem fio capaz de realizar a leitura das atividades neurais e também a estimulação cerebral foi apresentado. Esse sistema possui duas antenas que permitem a transferência de dados (*i.e.*, entre o dispositivo implantado com o dispositivo externo) e a alimentação sem fio. Os sistemas de eletroencefalografia implantáveis podem ser aplicados principalmente em pacientes epilépticos, eles são promissores para monitorar e controlar quando ocorre episódios de atividade neuronal elétrica incontrolável e desorganizada que resultam em crise epiléptica.

A interação humano-máquina mudará com os serviços que habilitem mais eficiência tanto nas experiências dos usuários de sistemas de informação, quanto na aquisição de informações desses usuários. Por exemplo, os sistemas de nanorredes baseados em interface entre cérebro-máquina mudam o sistema de entrada de comandos em sistemas digitais por fazer uma ponte direta entre a consciência humana e os dispositivos elétricos. Essa integração humano-máquina já começou a ser desenvolvida de uma maneira mais próxima da realidade. Um exemplo é a pesquisa apresentada em [Hochberg et al. 2006]. Neste estudo, um participante de 25 anos que é incapaz de mover seus membros devido à tetraplegia pode manipular diferentes dispositivos externos por meio de uma interface cérebro-máquina implantada no córtex pré-motor (arranjo de 96 microeletrodos). Utilizando o controle neural, o participante foi capaz de abrir e fechar uma mão protética, realizar ações rudimentares com um braço robótico multiarticulo e mover o cursor do computador por meio de seu cérebro.

As comunicações moleculares dentro do corpo humano são diretamente ligadas à saúde do organismo e, também, às práticas medicinais que permitem o diagnóstico, o tratamento e a cura de doenças. As comunicações moleculares como parte dos sistemas ciber-humanos tem potencial para permitir o monitoramento, detecção e controle da informações à medida que se propagam in vivo através das reações bioquímica. As co-

municações entre células, órgãos e sistemas no corpo humano envolvem a transmissão, propagação e recepção de inúmeras moléculas e proteínas que transmitem informações necessárias para as reações energéticas, ajudando no metabolismo das partes envolvidas. Quando células, órgãos ou até sistemas falham na transmissão e recepção de informações moleculares, doenças naturalmente surgem. Acredita-se que o desenvolvimento e a progressão do câncer são decorrentes da anormalidade na propagação da informação molecular subjacente à diferenciação e proliferação celular, entre outras [Sakkaff et al. 2018]. Da mesma forma, a doença de Alzheimer foi associada com células gliais que tem sua comunicação molecular e equilíbrio iônico perturbado pelo acumulo de transmissores no ambiente [Mattson 2004].

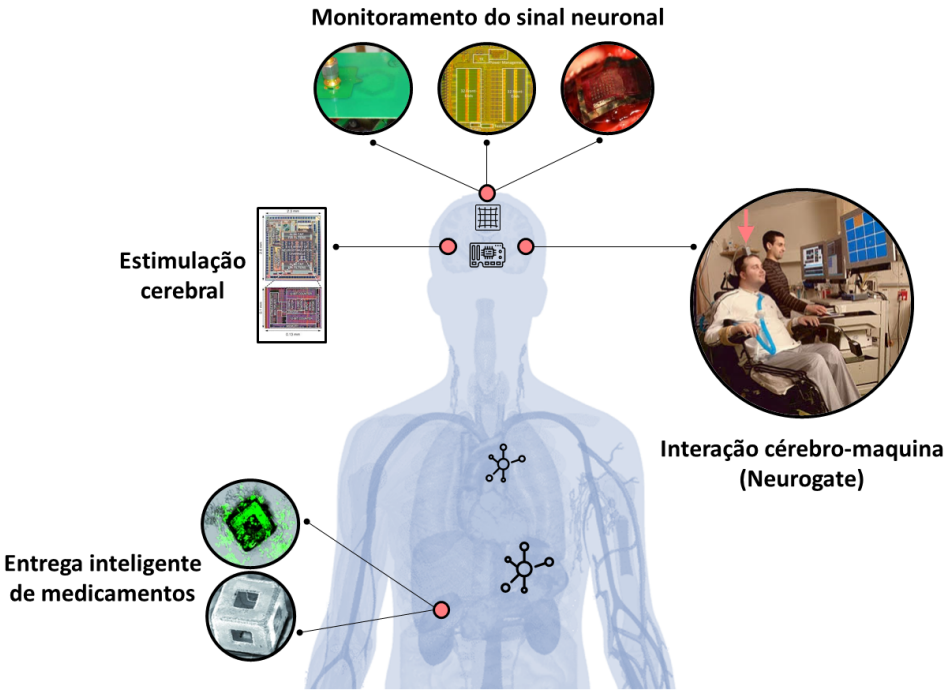


Figura 4.8: Exemplos reais de dispositivos intracoporais utilizados em sistemas ciberfísicos: Sistema de entrega de medicamentos com controle remoto. O nanocubo que transporta o medicamento apresenta os estados ativo e inativo [Ye et al. 2007]; Gravador de sinal neuronal para o córtex cerebral [Muller et al. 2014]; Sistema e sonda neural sem fio para realizar a estimulação cerebral e leitura das atividades neurais simultaneamente [Kim et al. 2016]; Sensor BrainGate implantado no cérebro do participante para a interação cérebro-máquina. O participante do estudo está sentado em uma cadeira de rodas, ventilado mecanicamente por meio de uma traqueostomia [Hochberg et al. 2006].

A medicina, de certo modo, vem aplicando práticas de correção das doenças causadas por falhas nas comunicações moleculares, por exemplo a radioterapia. Entretanto, só recentemente, práticas médicas vêm sendo redesenhadas com os avanços de outras áreas do conhecimento, como a engenharia eletromecânica, a nanotecnologia e biotecnologia. A nanomedicina é oficialmente uma área que promete materiais e sistemas minimamente invasivos ao corpo humano para acessar níveis de detalhes sobre o seu funcionamento e também repensar novas práticas medicinais. Abordagem que fazem o uso

das comunicações moleculares para a detecção de tumores são um exemplo. Um sensor com receptores sintéticos de CM para moléculas de glicoproteína (moléculas diferenciadas por suas cadeias de carboidratos) foi proposto em [Stephenson-Brown et al. 2015]. O sensor melhora a eficiência e a precisão da detecção do câncer de próstata ao identificar a presença da molecula associada ao tumor. Soluções inovadoras para tratamento de potenciais condições de saúde prejudiciais ao nível celular, podem vir de décadas de conhecimento sobre o processamento de informações coletadas dos sistemas intracoporais. Essa compreensão, combinada com as comunicação em sistemas elétricos e sua tradução em sistemas bioquímicos e vice-versa através dos sistemas ciber-humanos, proporcionarão novas práticas medicinais com resultados inimagináveis [Akyildiz et al. 2019].

As aplicações para os sistemas ciber-humanos não se restringem exclusivamente à escala nano e à comunicação intracorporal. Apoiados pelos dispositivos e redes vestíveis os sistemas ciber-humanos oferecem uma infinidade de aplicações médicas e não médicas em escala macro. A Figura 4.9 ilustra as etapas essenciais para que uma rede vestível ofereça suporte às aplicações médicas: (i) a coleta dos dados, (ii) a transmissão dos dados através das tecnologias de comunicação, e (iii) a observação dos dados. A detecção precoce de doenças, o monitoramento de sinais vitais, e os sistemas de suporte à vida são as principais aplicações médicas. No contexto não-médico a detecção de gestos a partir de sensores corporais, a integração com ambientes inteligentes, aplicações de segurança para indústria e para o usuário, e a integração social são os principais destaques. Para todas as aplicações suportadas pelos sistemas ciber-humanos utilizando redes vestíveis as pessoas são fundamentais e estão direta ou indiretamente envolvidas nos processos, o objetivo final é melhorar a qualidade de vida.

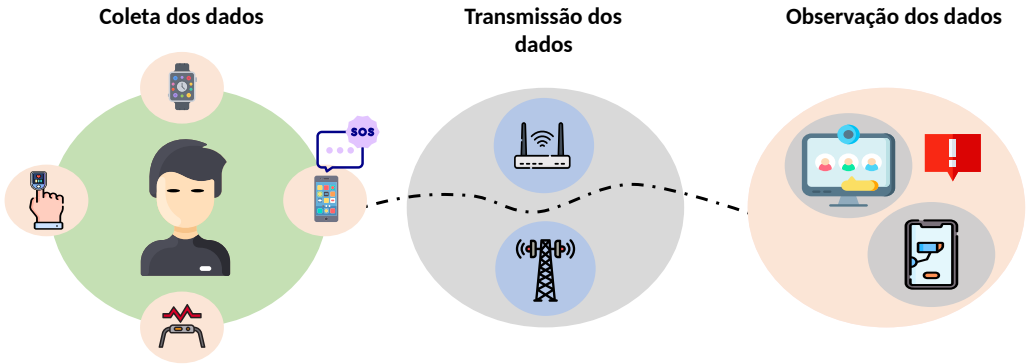


Figura 4.9: Redes vestíveis e aplicações médicas

Os sistemas ciber-humanos apoiados pelas redes vestíveis têm o potencial de transformar os sistemas tradicionais de saúde uma vez que oferecem ao usuário uma forma sofisticada porém simples de monitorar os sinais vitais e detectar doenças. Doenças como câncer, Parkinson, e asma podem ser fatais, entretanto, a detecção precoce dessas doenças tem um impacto positivo sobre o tratamento e a qualidade de vida do usuário [Movassaghi et al. 2014]. As aplicações médicas baseadas em redes vestíveis buscam melhorar os sistemas de saúde para que através dos vestíveis seja possível monitorar os sinais vitais de um paciente e e tornar mais eficiente a detecção e o controle de doenças. As informações obtidas e disseminadas a partir da rede vestível tem reflexo no mundo físico com ações que partem de um profissional da saúde ou eventualmente do próprio usuá-

rio. Em algumas situações específicas, dispositivos ciber-físicos podem interagir com um usuário para administração de medicamentos, é o caso de pacientes em leito hospitalar ou com dificuldades motoras.

Uma das aplicações de maior importância proporcionadas pelas redes vestíveis é o monitoramento contínuo dos sinais vitais de um usuário. Geralmente, as aplicações identificam as condições de saúde de um usuário e oferecem algum tipo de resposta em tempo real, com informações que possam colaborar na recuperação do paciente. A principal diferença entre o monitoramento de pacientes conduzido em ambiente hospitalar e a partir das redes vestíveis é a comodidade proporcionada ao usuário. A partir das redes vestíveis o paciente pode ser observado remotamente por um profissional da saúde considerando os seus estados fisiológicos e ações naturais, sem restringir as suas atividades regulares e reduzindo o impacto dos altos custos de uma internação hospitalar. Esse tipo de aplicação ganha cada vez mais relevância na medida em que a evolução tecnológica permita o desenvolvimento de sistemas mais precisos, seguros, convenientes e baratos.

As aplicações que monitoram as atividades de um usuário também utilizam as características fisiológicas. Adicionalmente, essas aplicações consideram outras informações oferecidas pelos vestíveis como localização e movimentação do usuário. Ao contrário do monitoramento de pacientes (leito hospitalar, cuidado domiciliar), essas aplicações visam propiciar ao usuário convencional a possibilidade de acompanhar suas atividades cotidianas e oferecer informações e alertas para o próprio usuário ou para terceiros. Além da capacidade de identificar padrões fisiológicos como a oxigenação do sangue e varição da frequência cardíaca, os vestíveis também são dotados de sensores de posicionamento global e giroscópios. Isso permite que além de monitorar as condições de saúde do usuário seja possível identificar aonde ele está e variações bruscas de movimentação, uma queda por exemplo. Sendo assim, através das redes vestíveis é possível monitorar a frequência cardíaca de um atleta durante o treino, ou identificar o trajeto de um idoso e sua condição respiratória. Dentro de uma rede vestível convencional, ainda é possível gerar avisos para o próprio usuário (visual, sonoro), ou usar as tecnologias de comunicação disponíveis para alertar remotamente um profissional da saúde ou um contato pessoal.

Os dados fisiológicos coletados continuamente a partir dos dispositivos vestíveis impulsionam outras aplicações como a detecção precoce e o controle de doenças. Para cada nível de evolução, os sensores disponíveis nos dispositivos vestíveis oferecem dados mais precisos e confiáveis. As informações coletadas podem ser processadas e através de técnicas de aprendizagem de máquina é possível identificar padrões escondidos nas estruturas de informação [Al-Turjman and Baali 2019]. Esses padrões podem indicar a presença de condições pré-existentes, como uma propensão a doenças cardiovasculares [Oresko et al. 2010]. Da mesma forma, o processamento das informações relacionadas à oxigenação do sangue pode identificar problemas respiratórios, como a asma. As aplicações responsáveis pelo controle de doenças também utilizam os dados dos vestíveis para identificar possíveis mudanças nos padrões fisiológicos dos usuários perante um tratamento específico. Ao analisar os dados após a introdução de um tratamento, é possível identificar o impacto positivo ou negativo da terapia, como a responsividade do paciente a um medicamento por exemplo [Kubota et al. 2016].

As aplicações da telemedicina são consolidadas no mercado e seus benefícios são

reconhecidos pela comunidade médica e científica. Entretanto, as redes vestíveis proporcionam novas funcionalidades e aplicações adicionais para a telemedicina tradicional. A telemedicina possibilita o acesso remoto aos cuidados médicos através da integração dos sistemas de saúde e das tecnologias de telecomunicação. Nesse contexto, os dispositivos vestíveis agregam novas informações que complementam e diversificam as aplicações na telemedicina. Uma consulta remota em que um paciente é assistido por um profissional da saúde, por exemplo, pode ganhar novos contornos com a utilização dos vestíveis. Além das características multimídia de uma consulta convencional, o profissional da saúde pode observar as informações coletadas a partir dos dispositivos na rede vestível, propiciando um diagnóstico mais completo e correto. A partir das informações dos vestíveis e com o suporte da telemedicina, novas aplicações podem emergir incluindo aplicações voltadas para diagnósticos em tempo real, auxiliares na manutenção de doenças crônicas, e para assistência remota na recuperação de procedimentos médicos.

Os sistemas ambientais de suporte à vida, do inglês Ambient Assited Living Systems (AAL), são conjuntos de aplicações que também se beneficiam com a introdução dos dispositivos e redes vestíveis. Os avanços tecnológicos e na medicina permitem que a expectativa de vida aumente gradualmente, fazendo com a população de pessoas idosas seja cada vez maior. As aplicações AAL monitoram aspectos relacionados ao bem-estar dos idosos através de sensores, atuadores, e dispositivos posicionados no ambiente (casas, apartamentos). O objetivo é migrar o local de observação dos idosos dos hospitais e clínicas de repouso para as residências, melhorando a qualidade de vida do paciente e reduzindo os custos operacionais. As redes vestíveis se integram perfeitamente aos sistemas AAL, incorporando os dados coletados a partir dos vestíveis aos dados do ambiente e possibilitando realizar a correlação dessas informações. Além das redes vestíveis as aplicações AAL se apoiam em diferentes conceitos como as residências inteligentes e a própria IoT. As aplicações AAL visam trazer independência aos usuários para realizar suas atividades cotidianas em um ambiente familiar e assistidos por mecanismos de suporte à saúde sejam locais ou remotos.

O sono é uma necessidade básica e fundamental para o ser humano. Dormir de forma saudável é fundamental para o equilíbrio físico e mental. As consequências da privação do sono incluem diversos transtornos como a narcolepsia e doenças cardiovasculares [Khan and Pathan 2018]. Alguns dispositivos vestíveis tem a capacidade de monitorar a qualidade do sono utilizando dados como a variação cardíaca e o movimento corporal. Esse monitoramento só é possível pois os fabricantes dos dispositivos utilizam grandes bases de dados, aprendizagem de máquina, e alto investimento financeiro para reconhecer padrões de sono através das bases de dados. A forma mais comum de monitorar as atividades de um paciente durante o sono é através de uma polissonografia, um exame que deve ser conduzido em ambiente controlado. Recentemente, pesquisadores identificaram que os dados obtidos a partir dos vestíveis são tão eficientes quanto os exames em ambiente controlado para medir e contribuir em melhorias na qualidade do sono [Baron et al. 2018]. A partir dos vestíveis o próprio usuário consegue mensurar a qualidade do seu sono e criar um histórico de eventos, sendo possível avaliar a necessidade de exames mais complexos.

As redes vestíveis suportam aplicações não-médicas nos setores de ambientes inteligentes, indústria, segurança e entretenimento, a Figura 4.10 ilustra algumas das apli-

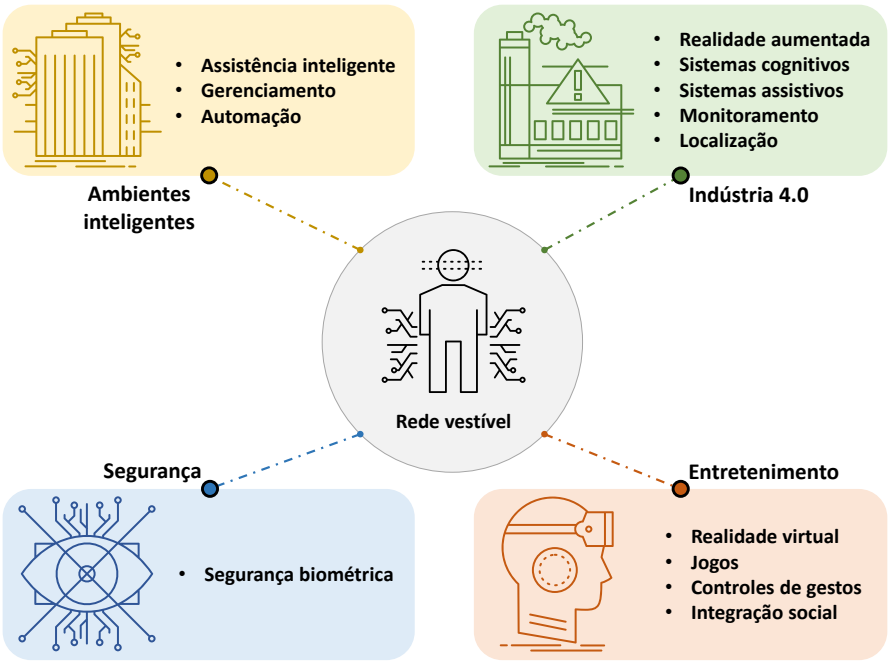


Figura 4.10: Redes vestíveis e aplicações não médicas

cações não médicas para as redes vestíveis. Ambientes inteligentes oferecem aos usuários diversos serviços de assistência e de gerenciamento de recursos que podem ser potencializados através dos vestíveis. Os serviços auxiliares utilizam os dados dos dispositivos vestíveis para construir um ambiente de acordo com as preferências do usuário. Um exemplo de integração entre os vestíveis e o ambiente inteligente é o controle automático da temperatura do ambiente de acordo com a intensidade de movimentação do usuário capturada a partir de um vestível. Outras aplicações incluem o controle de iluminação e o acionamento automático de equipamentos conectados (televisão, aparelho de som) de acordo com o comportamento e posicionamento do usuário. Os serviços de gerenciamento também se beneficiam dos vestíveis pois gerenciam aspectos de segurança e automação do ambiente. Alertas de risco (invasão, vazamento de gás, sobrecarga elétrica) podem ser enviados diretamente para um vestível, otimizando a resposta do usuário.

Nos últimos anos, a materialização dos conceitos de Internet das coisas, computação em nuvem e computação ubíqua, propiciaram mudanças drásticas na indústria. Essas mudanças, aliadas aos sistemas e aplicações cada vez mais voltados à Internet, alavancaram o conceito da Indústria 4.0. Os sistemas ciber-humanos terão papel importante no desenvolvimento da indústria inteligente, uma vez que além de máquinas e dispositivos, esses sistemas inteligentes também visam a integração do ser humano. A sustentabilidade é um dos principais aspectos na Indústria 4.0 [Roda-Sanchez et al. 2018]. Nos próximos anos, os sistemas industriais serão constituídos de dispositivos de baixa complexidade (IoT, vestíveis) e de redes de comunicação heterogêneas, tornando-se obrigatória a utilização racional de energia para a longevidade desses sistemas. A inserção das pessoas nesse ambiente conectado promove a evolução de um sistema ciber-físico para um sistema centrado no usuário, capaz de fortalecer aspectos de segurança, conforto, e bem-estar dos trabalhadores em direção a melhores condições de trabalho e aumentos nos níveis de produção. As principais tecnologias suportadas pelas redes vestíveis e empregadas na In-

dústria 4.0 são a realidade aumentada, os sistemas computacionais cognitivos e assistivos, e a possibilidade de monitoramento e localização de um trabalhador.

A realidade aumentada é uma tecnologia que permite sobrepor elementos virtuais à nossa visão da realidade através de telas ou projeções em capacetes ou óculos inteligentes. A realidade aumentada pode ser aplicada na indústria para integração humano-computador, melhoria nos processos de decisão e treinamentos personalizados [Posada et al. 2015]. Através dos dados coletados a partir dos vestíveis (movimentação, gestos) e da aplicação de técnicas de auto-aprendizagem os sistemas cognitivos podem detectar fadiga em um operário, ou ajustar automaticamente parâmetros que contribuam para produtividade, como melhoria na iluminação, ajustes ergonômicos, entre outros. Sistemas de monitoramento podem identificar as atividades do trabalhador e gerar alertas no tocante a sua saúde e segurança. Pessoas que trabalham em posição sentada por longos períodos precisam fazer pausas periódicas para evitar problemas circulatórios e lesões na região lombar, os vestíveis podem monitorar e alertar automaticamente tanto o trabalhador quanto a equipe de prevenção de acidentes [Kong et al. 2019]. A localização do usuário também é importante pois pode ser utilizada para fornecer alertas sonoros e visuais sobre áreas de maior periculosidade (piso escorregadio, necessidade de equipamentos de proteção individual).

Os setores de esporte e entretenimento também são atrativos para as aplicações baseadas em redes vestíveis. As aplicações para esses setores se apoiam em tecnologias habilitadas pelos dispositivos vestíveis, dentre elas a realidade aumentada e virtual, a captura de movimentos, e a integração social. Já há algum tempo, a indústria de jogos aplica sensores de movimentos e realidade virtual em seus produtos mas a popularização dos vestíveis impulsiona novas possibilidades. Durante os processos de confinamento causados pela COVID-19 a empresa Nintendo lançou o jogo de interpretação de personagens, do inglês *Role-playing Game* (RPG), Ring Fit Adventure para incentivar a prática de exercícios físicos em ambientes fechados. O jogo utiliza um anel de Pilates com sensores (giroscópio, medidor de frequência cardíaca por infravermelho, acelerômetro) e um vestível (faixa coxal) para movimentar um personagem e realizar ações de ataque e defesa dentro do mundo virtua. O conjunto emite alertas de sedentarismo através dos vestíveis e possui integração direta com redes sociais, sendo uma boa mostra da capacidade dos vestíveis para aplicações de entretenimento. Mais de 10 milhões de cópias foram comercializadas, demonstrando tanto a aplicabilidade quanto o interesse despertado pelos vestíveis.

As características fisiológicas, físicas e comportamentais dos usuários coletadas a partir dos vestíveis colaboram com o desenvolvimento de aplicações de segurança, sendo as principais a autenticação baseada em sinais biológicos. Sinais biológicos dinâmicos são identificados a partir de sensores óticos, elétricos e mecânicos disponíveis nos vestíveis. Os sinais biológicos possuem diferentes origens, sendo as mais comuns o coração (variação cardíaca), o cérebro (atividade cerebral), pulmões (frequência respiratória), os músculos (contração muscular), e a pele (atividade eletrodérmica). Em geral, esses sinais são únicos e exclusivos para cada pessoa, tornando-os elegíveis para utilização como chave em sistemas de autenticação [Nakayama et al. 2019]. Dessa forma, ações não intrusivas para o usuário como caminhar, respirar, ou realizar gestos podem se tornar uma padrão para iniciar ou manter continuamente a autenticação em um sistema. Mecanismo de autenticação baseados em biometria estática (digitais, face) também se beneficiam dos

vestíveis valendo-se do posicionamento dos dispositivos, geralmente instalados próximos aos rosto e mãos, facilitando a coleta dos dados.

4.4. Desafios e soluções existentes para os sistemas ciber-humanos

Os principais desafios para o desenvolvimento dos sistemas ciber-físicos e ciber-humanos, incluindo das redes vestíveis, nanorredes, são o foco desta seção. Além disso, serão apontadas soluções para os desafios identificados.

4.4.1. Redes vestíveis e a Internet das coisas médicas

O futuro para as redes vestíveis e para Internet das coisas médicas é muito promissor. Entretanto, para qualquer nova tecnologia, a transição do conceito para aplicação prática é repleta de desafios. Na verdade, ainda existem desafios notáveis para as redes vestíveis transporem tanto atualmente quanto no futuro. As soluções existentes podem confrontar alguns desafios e minimiza o impacto negativo, entretanto, muitos problemas ainda não têm solução ou em alguns casos as soluções são específicas para um contexto. A Tabela 4.3 apresenta os principais desafios e possíveis soluções considerando as características das redes vestíveis. É importante salientar que em várias situações a possível solução para um problema pode trazer novos conflitos. Por exemplo, aplicar técnicas de aprendizagem de máquina para aprimorar a precisão dos dados coletados pode aumentar a necessidade de processamento e consequentemente reduzir a duração das baterias.

Grande parte dos desafios associados à aquisição dos dados está concentrada no primeiro componente da cadeia de processamento dos dados, ou seja, no dispositivo vestível. Parâmetros importantes como a quantidade e qualidade dos dados são definidos nessa primeira etapa. Os dados disponíveis nas redes vestíveis são obtidos a partir de uma infinidade de dispositivos, com características e qualidades de construção distintas. Eventualmente, os sensores de baixo custo ou simplesmente mal calibrados podem produzir dados de baixa qualidade ou inconsistentes. Obter dados de maior qualidade respeitando a diversidade dos dispositivos é um desafio para as redes vestíveis. Dados inconsistentes podem gerar informações incorretas ao serem processados e causar alertas impróprios, como um falso alerta de aumento na frequência cardíaca, por exemplo. Possíveis soluções para o problema incluem aumentar a frequência de coleta e a quantia dos dados coletados para melhorar a amostragem, entretanto, isso pode afetar a eficiência energética e a capacidade de armazenamento do dispositivo.

O desafio de processar os dados coletados está relacionado à utilização dos dados depois da etapa de aquisição. Conforme citado acima, os vestíveis têm características e sensores com qualidades diferentes, dependendo do tipo, preço e marca dispositivo. Converter os dados brutos coletados a partir dos dispositivos em informações úteis é o objetivo do processamento dos dados. O principal desafio é gerar as informações de forma eficiente, considerando o limitado poder de processamento e armazenamento dos dispositivos. A filtragem dos dados inclui identificar possíveis erros de captura e valores discrepantes através de métodos estatísticos, para obter um conjunto de dados mais homogêneo. A quantidade de informações aproveitáveis obtidas a partir dos dados do usuário dependerá da eficiência na filtragem e no processamento dos dados. Atualmente, aplica-se cada vez mais a aprendizagem de máquina no processamento dos dados obtidos através de dis-

Características	Desafios	Possíveis soluções
	Aquisição dos dados	
-Baixa qualidade dos sensores -Inconsistência na coleta dos dados	-Obter dados de maior qualidade	-Aumentar a quantia de dados coletados -Aumentar a frequência de coleta
	Processamento dos dados	
-Baixa capacidade computacional -Baixa capacidade de coleta e armazenamento dos dados	-Gerar informações de forma eficiente	-Otimizar a análise de dados -Distribuição do processamento -Aplicar aprendizagem de máquina
	Transmissão dos dados	
-Baixas taxas de transmissão -Possibilidade de alta latência -Possibilidade de sobrecarga na rede	-Atender aos requisitos exigidos pelas aplicações	-Computação de borda (reduzir latência) -Redes vestíveis heterogêneas -Empregar protocolos multicaminhos
	Eficiência energética	
-Baterias de pequena capacidade -Dispositivos pequenos	-Ampliar o tempo de funcionamento	-Uso racional da energia -Sistemas auto-alimentados -Colheita de energia
	Interoperabilidade	
-Falta de padronização na construção dos dispositivos -Falta de compatibilidade (comunicação, serviços, aplicações)	-Integrar os componentes da rede vestível	-Cumprimento das normas estabelecidas -Dispositivos compatíveis com múltiplos padrões
	Escalabilidade	
-Infraestruturas inadequadas para o possível crescimento	-Lidar com o aumento exponencial dos dispositivos e dados	-Gerenciamento da infraestrutura
	Aceitação do usuário	
-Tecnologias consideradas intrusivas -Desconfiança do usuário -Aversão à tecnologia	-Popularizar as redes vestíveis	-Detalhar o uso dos dispositivos -Simplificar as explicações de uso -Detalhar as medidas de segurança aplicadas
	Segurança e Privacidade	
-Múltiplas tecnologias de comunicação -Informações privadas / pessoais -Dados sensíveis (saúde) -Difícil aplicar mecanismos eficientes	-Manter a segurança do usuário e a privacidade dos dados	-Aplicar mecanismos de autenticação e autorização para usuários e dispositivos -Aplicar criptografia dos dados -Corrigir problemas de software através de atualizações

Tabela 4.3: Características, desafios e possíveis soluções para as redes vestíveis

positivos vestíveis. Outra possibilidade é distribuir o processamento dos dados entre os componentes da rede vestível ou uma estrutura na nuvem. Entretanto, essa alternativa pode sobrecarregar os canais de transmissão de dados de curta e longa distância.

Na medida em que caminhamos para sistemas mais interconectados, a produção, o processamento e a transmissão de dados vão continuar crescendo. Tecnologias como o 5G apoiam essa expansão ao oferecer taxas de transmissão cada vez maiores, possibilitando o envio massivo de dados. Contudo, as redes vestíveis abrangem diversas tecnologias de comunicação de curta e longa distância com taxas de transmissão diferentes. A variedade nas tecnologias de comunicação promove a diversidade de dispositivos, porém é limitante para algumas aplicações com requisitos restritos (baixa latência, baixo atraso). Estruturas centralizadas na nuvem, por exemplo, são comumente utilizadas para gerenciamento, processamento e armazenamento dos dados. No entanto, elas possuem dois problemas principais (i) a latência para enviar e processar os dados, e (ii) a sobrecarga na rede vestível devido ao envio contínuo e simultâneo dos dados de múltiplos dispositivos. Soluções para problemas relacionados à latência incluem aplicar a computação de borda e empregar protocolos que utilizem múltiplos caminhos simultâneos [Nakayama et al. 2021] quando possível. As redes vestíveis heterogêneas reduzem os problemas de sobrecarga, uma vez

que diferentes dispositivos empregam a comunicação de longa distância, dispensando o uso de um coordenador e aliviando o tráfego na rede vestível.

A energia é um dos recursos mais importantes em uma rede vestível, especialmente com a crescente demanda dos usuários por novas funcionalidades. Tarefas complexas implicam, na maioria dos casos, em consumo adicional de energia e reduzem sensivelmente a duração das pequenas baterias presentes nos vestíveis. Para a maioria dos usuários não é conveniente recarregar frequentemente os dispositivos pois isso significa conectá-los a uma fonte de energia externa e limitar ou interromper o seu uso. Em geral, a eficiência energética pode ser alcançada de duas formas: (i) avanços nas tecnologias das baterias (materiais, componentes) e (ii) minimizando o consumo de energia. As pesquisas em relação às baterias compreendem tanto os dispositivos vestíveis quanto uma infinidade de dispositivos portáteis (*tablets*, fones de ouvido sem fio, *smartphones*), atraindo mais atenção da indústria e da academia e obtendo resultados mais lineares. Os estudos que visam obter um uso mais racional da energia geralmente contemplam melhorias nos protocolos de comunicação e variações nos padrões de aquisição dos dados (reduzir tempo por coleta, reduzir quantidade de dados). Recentemente, pesquisadores da área estudam a viabilidade e aplicar técnicas como o carregamento sem fio e sistemas auto-alimentados através da colheita de energia do ambiente, às redes vestíveis [Seneviratne et al. 2017].

Os primeiros dispositivos comerciais com a capacidade de carregamento sem fio foram os *smartphones*. Atualmente, essa tecnologia atende aos computadores portáteis, veículos elétricos, e vestíveis, porém ela não dispensa a necessidade de carregar os dispositivos, simplesmente oferece uma alternativa prática. A principal limitação é o tempo elevado para carregamento das baterias em comparação às tecnologias tradicionais com fios. Técnicas de colheita de energia permitem que o vestível se mantenha operacional por tempo indefinido ao coletar a energia do ambiente que cerca o usuário. A colheita de energia do ambiente pode ocorrer através de movimento, variações de temperatura, luz, radiação eletromagnética, entre outras. Esses métodos incluem a coleta da energia cinética gerada através do movimento humano e a captura da energia solar para reabastecer os dispositivos[Ometov et al. 2021]. Todavia, os vestíveis tem limitações de tamanho que dificultam a instalação dos componentes adicionais necessários para aplicação desses métodos. Adicionalmente, a disponibilidade de energia no ambiente é limitada e pode não ser suficiente para alimentar um dispositivo. Sendo assim, embora promissores, esses métodos carecem de maiores pesquisas considerando os dispositivos vestíveis.

Embora existam avanços notáveis por parte da academia, indústria, e órgãos de padronização para desenvolver soluções para as redes vestíveis, alguns aspectos necessitam atenção especial. Entre eles estão a interoperabilidade entre os dispositivos, a escalabilidade e os aspectos de segurança. As redes vestíveis são constituídas por uma grande diversidade de sensores, dispositivos, tecnologias de comunicação, serviços, e aplicações que devem interagir e cooperar de forma transparente para atingir todo o potencial concebível em um sistema ciber-humano. Dispositivos vestíveis ainda sofrem de problemas de compatibilidade entre si e com outros componentes da rede, muitas vezes causados pela falta de padronização na construção e pela utilização de tecnologias proprietárias. A falta compatibilidade entre os dispositivos e tecnologias de comunicação se propaga para os serviços e aplicações, obrigando muitas vezes que o usuário utilize exclusivamente a plataforma de um fabricante específico, prejudicando a disseminação das redes vestí-

veis e reduzindo a aceitação do usuário. A falta de compatibilidade promove ainda uma escalada desordenada no número de vestíveis. Para realizar todas as ações desejadas, o usuário necessita de mais dispositivos, que por sua vez empregam tecnologias de comunicação distintas e apoiam aplicações exclusivas, podendo sobrecarregar a rede vestível e as estruturas de suporte (redes de acesso, estruturas centralizadas na nuvem).

A popularização das redes vestíveis favorece um cenário onde uma grande quantidade de dispositivos conectados à Internet ocupa grande parte do espaço que nos cerca. Individualmente, esses dispositivos produzem uma quantia limitada de dados, mas considerando o conjunto de dispositivos a quantidade de dados é incalculável. Na medida em que aspectos de segurança e interoperabilidade são aprimorados e a aceitação do usuário em favor da tecnologia aumente, espera-se que a quantidade de informações partindo das redes vestíveis seja ainda maior. A atual infraestrutura que suporta a comunicação e as aplicações das redes vestíveis deve comportar o aumento exponencial dos dispositivos e dados. Uma das dificuldades para o gerenciamento pró-ativo rumo às melhorias nas infraestruturas de comunicação reside na dificuldade em mensurar a quantidade e capacidade dos dispositivos móveis em uma área específica. Isso torna difícil aplicar soluções que reduzam a carga sobre a infraestrutura centralizada, como o compartilhamento de recursos (processamento, comunicação), e a integração dos dispositivos e usuários através da Internet das coisas sociais (SIoT), por exemplo.

Para que qualquer tecnologia evolua de forma satisfatória é necessário que além de trazer benefícios ela consiga se disseminar e construir uma base sólida de utilizadores. A popularização das redes vestíveis permite a continuidade nas pesquisas e no desenvolvimento das tecnologias, que impulsionam por sua vez o surgimento de novas e melhores aplicações. No entanto, algumas características próprias dos dispositivos vestíveis podem reduzir o interesse ou até afastar os usuários. A disposição dos vestíveis sobre o corpo pode trazer desconforto físico e psicológico para alguns usuários que podem considerar o seu uso intrusivo. Muitos usuários se sentem intimidados pois não sabem exatamente como os dados coletados serão utilizados e qual o nível de segurança efetivo. Na maioria dos casos a desconfiança não está na tecnologia em si mas no fabricante do dispositivo e também nas plataformas de armazenamento de dados. Na situação dos vestíveis aplicados ao setor industrial, os trabalhadores podem não compreender a necessidade de monitoramento ou a funcionalidade dos vestíveis, sentindo-se vigiados e promovendo a insegurança. Finalmente, alguns usuários são avessos à tecnologia em geral, tornando difícil a inserção dos vestíveis em uma rotina diária.

A possibilidade de detectar e capturar dados continuamente é um dos destaques da tecnologia vestível. Entretanto, os dados coletados muitas vezes representam características e informações pessoais do usuário, sendo necessário uma preocupação ainda maior com a segurança e privacidade dos dados. Além da apreensão em razão das ameaças às tecnologias de comunicação sem fio (ataques, espionagem), os usuários também se preocupam com a segurança das plataformas de armazenamento de dados na nuvem e dos serviços baseados em localização. Países ao redor do mundo possuem leis específicas que lidam com a privacidade dos dados, incluindo as informações obtidas a partir dos vestíveis. No Brasil, desde setembro de 2020 a Lei Geral de Proteção de Dados (LGPD)visa estabelecer um controle mais rigoroso ao acesso à informações de usuários captadas através de ferramentas tecnológicas [Pinheiro 2020]. Isso deve impactar a maneira como os

dados coletados através do vestíveis são armazenados e compartilhados. As preocupações com a segurança e a privacidade dos dados são ainda maiores quando os dados relacionados à saúde estão envolvidos.

Um terço de todos os dispositivos da IoT está relacionado à área da saúde e estima-se que 87% das organizações de saúde já adotam a Internet das Coisas Médicas, do inglês Internet of Health Things (IoHT). Entre as preocupações da IoHT estão a segurança, a resiliência e o desempenho da comunicação, uma vez que as aplicações em saúde requerem um alto nível de segurança e proteção [Wolf and Serpanos 2017]. Nessa aplicação dos vestíveis a vida das pessoas depende dos novos dispositivos e serviços médicos, exigindo atenção especial à resiliência (ou seja, disponibilidade de conexão contínua - 99,99999% de confiabilidade, latência ultrabaixa e suporte à mobilidade), mesmo no advento de falhas e ameaças, que podem ser imprevisíveis durante o desenvolvimento ou podem surgir durante a utilização [Laprie 2008]. No entanto, as vulnerabilidades da IoT em termos de segurança, privacidade e confiabilidade são amplamente conhecidas a partir de resultados acadêmicos [Coelho et al. 2019]. Essas vulnerabilidades não são diferentes para o contexto das redes vestíveis e requerem mais cuidados dadas as circunstâncias da IoHT.

4.4.2. Desafios das nanorredes

A comunicação intracorporal é uma tecnologia em fase de amadurecimento e muitos desafios existem. Diferentes adversidades são geradas pelo fato de que os dispositivos em micro e nanoescala têm capacidades computacionais reduzidas. A alimentação energética é um exemplo de limitação que dificulta a comunicação em nanorredes que utilizam dispositivos exclusivamente eletrônicos. Como os dispositivos alimentados por bateria têm vida útil limitada, métodos de captação e transferência de energia sem fio para alimentar os dispositivos foram investigados. Essas técnicas são promissoras para o desenvolvimento de implantáveis neutros em termos de energia, i.e., todas as operações do dispositivo são alimentadas por energia coletada do ambiente (mecânica, vibracional e química) [Kuscu et al. 2019]. No entanto, experimentos indicam que essas soluções fornecem energia de forma limitada aos dispositivos [Koucheryavy et al. 2021]. Como resultado, isso torna o desenvolvimento de até mesmo protocolos mais básicos como endereçamento e roteamento altamente desafiador. Esquemas de captação de energia ocorrem naturalmente nas CM e são propostas como soluções para os sistemas bioinspirados. O desafio para essas redes é então elaborar formas de mensurar a energia necessária para alcançar a codificação utilizando métricas como energia necessária por bit.

A baixa taxa de transmissão de dados é uma característica dos sistemas intracoporais e é causada por diversos fatores incluindo o atraso de propagação da informação, bem como o excesso de ruído no ambiente. Para a comunicação eletromagnética vários fenômenos quânticos afetam a propagação de ondas eletromagnéticas no grafeno. A frequência de ressonância dessas estruturas em escala nano pode ter magnitude duas vezes menor que as antenas não carbonadas e a eficiência de radiação também pode ser prejudicada por causa desse fenômeno [Akyildiz 2012]. Ademais, espera-se que os transceptores baseados em grafeno sejam limitados em termos de complexidade devido as restrições de espaço e limites de integração [Akyildiz and Jornet 2010]. Nas comunicações internas, a absorção de moléculas presentes no ambiente atenua o sinal transmitido e introduz ruído [Akyildiz and Jornet 2010]. Nas comunicações moleculares, os ruídos es-

tão relacionados ao tipo de canal e técnica de propagação empregada [Borges et al. 2021]. Portanto, o gerenciamento do ruído na CM sintética deve ser feito com base na caracterização da dinâmica do ambiente. Embora essas características desafiem as aplicações intracoporais, uma grande oportunidade surge para o desenvolvimento de esquemas de comunicação confiáveis ou técnicas adaptativas para lidar com essas adversidades.

Uma das questões de pesquisa mais desafiadora a ser abordada na comunicação intracorporal é a segurança para a saúde com os dispositivos implantáveis e a biocompatibilidade desses itens. A composição dos dispositivos intracorporais é importante para garantir biocompatibilidade. Atualmente, os dispositivos intracorporais que possuem milímetros ou micrômetros de tamanho têm sido desenvolvidos com materiais como piezoelétricos, platina, ouro, titânio, cobre, silício, prata e alumínio [Koucheryavy et al. 2021]. Entre os problemas da não compatibilidade levantados na literatura estão a toxicidade que os materiais em nanoescala podem apresentar, a corrosão de implantes dentro do corpo e a rejeição do tecido na área do transplante que impede o funcionamento dos biossensores, assim como a transmissão dos dados [Kuscu et al. 2019, Barbone et al. 2019]. Como os problemas levantados se aplicam especificamente aos dispositivos desenvolvidos a partir de nanomateriais, autores defendem que esses problemas podem ser evitados pela escolha da arquiteturas biológicas para o desenvolvimento dos componentes (tecidos e organismos geneticamente modificados) [Kuscu et al. 2019, Akyildiz et al. 2015].

Com o objetivo de trazer soluções para o desenvolvimento de novos dispositivos com aplicações médicas, uma extensa gama de estudos sobre materiais biocompatíveis como polímeros, denrímeros, eletrônica orgânica e hidrogéis foram realizados na última década [Kuscu et al. 2019]. Além desses, o grafeno tem sido investigado por permitir o desenvolvimento de componentes em nanoescala resistentes e possuir boa condutividade elétrica e térmica. Com o surgimento de novos materiais, as questões de biocompatibilidade tanto dos dispositivos fabricados a partir de componentes sintéticos como biológicos ou híbridos deverão ser investigadas por meio de testes in-vivo e ensaios clínicos apropriados tal como ocorre para qualquer outro dispositivo médico. Uma vez que as aplicações intracoporais de saúde prometem explorar a propagação de informações dentro do corpo, a segurança dos sistemas de comunicação desenvolvidos também deve ser uma preocupação primária [Akyildiz et al. 2019].

Para se comunicar com redes externas, incluindo a Internet, os sistemas intracorporais exigirão uma interface tradutora para realizar um link de dados entre o dispositivo implantado e o dispositivo externo. Um dos principais desafios para a realização das interfaces bio-cibernéticas está na engenharia de processos químicos e físicos capaz de identificar com precisão as características da molécula onde a informação está codificada, e traduzi-las em sinais elétricos [Akyildiz et al. 2015]. Uma possível solução neste sentido pode vir de novos nanosensores que possuem capacidade de detecção de sinais químicos e biológicos. Como as moléculas e organismos vivos (bactérias, fungos, entre outros) podem apresentar bioluminescência quando uma parcela de energia das ligações químicas é liberada na forma de luz visível, este fenômeno pode ser uma opção para realizar a transdução do sinal bioquímico para sinais elétricos [Fouad et al. 2020], assim como a resposta térmica e bioluminescência de moléculas para a transdução de um sinal elétrico para um sinal bioquímico [Chude-Okonkwo et al. 2016]. Esse processo pode ocorrer através das propriedades do sensor que ao ser ativado pela presença de moléculas

modula a corrente em um circuito elétrico [Akyildiz et al. 2015].

A possibilidade de interagir do ambiente externo com as comunicações intracoporais e as informação do corpo através de interfaces bio-cibernéticas adiciona outra dimensão à segurança. Um ataque ao sistema ciber-humano pode ser usado para acessar a rede intracorporal e obter informações pessoais relacionadas à saúde ou até mesmo criar situações que coloque em risco a saúde do paciente [Akyildiz et al. 2015]. Os dispositivos intracorporais utilizados atualmente não possuem comunicações criptografadas uma vez que as contramedidas resultam no consumo de recursos computacionais adicionais ou até mesmo sendo necessário um módulo de segurança externo resultando em dispositivos maiores [Koucheryavy et al. 2021]. A segurança da informação é um assunto importante no desenvolvimento de aplicações voltadas a saúde e interação homem-máquina. Portanto, o progresso desses sistemas depende dos avanços de pesquisa relacionados à cibersegurança para prover a implementação de uma infraestrutura que possa garantir a integridade dos dados. Entretanto, as soluções devem atender aos requisitos computacionais dos dispositivos intracorporais.

Recentes avanços de pesquisa na biologia sintética permitiram através da manipulação artificial criar células que produzissem moléculas modificadas que podem ser utilizadas em redes baseadas em sinalização celular [Akyildiz et al. 2019]. Embora já existam técnicas de comunicação baseadas em nanomateriais e em células sintéticas com circuitos genéticos (*i.e.*, uma rede de reações químicas envolvendo genes e espécies moleculares trabalhando juntos) para codificar e modular símbolos, a implementação desses sistemas artificiais in-vivo deve validar suas capacidades de comunicação confiável. Da mesma forma, a implementação de esquemas de controle de erro para a sinalização celular que consideram as limitações das nanomáquinas biológicas e as diferentes fontes de ruídos dos tecidos permanece uma questão em aberto e deve levar em consideração dados experimentais do ambiente de aplicação e dos componentes biológicos utilizados.

4.4.3. Considerações sobre segurança, privacidade e resiliência

Os dispositivos empregados nas redes vestíveis comumente possuem falhas de segurança relacionadas à mecanismos de autenticação ineficientes, implementação deficiente das tecnologias de comunicação, e soluções defasadas [IBM 2020]. O resultado de ataques e exploração de falhas inclui a perda de informações resultante do sequestro de dispositivos, o tratamento impreciso de falhas, a negação de serviços, entre outros [FBI 2020]. Além disso, dados os recursos computacionais restritos como por exemplo, o processamento, a memória, a largura de banda, e o armazenamento em dispositivos vestíveis, frequentemente emprega-se versões simplificadas de mecanismos de segurança tradicionais. Isso pode levar à exposição parcial ou total de informações restritas [Blasco et al. 2019]. Adicionalmente, os canais de comunicação nas redes vestíveis estão sujeitos a ataques tanto durante o emparelhamento quanto na transmissão de dados, resultando em falhas no processo de autenticação do usuário [Classen et al. 2018], e criptografia fraca ou nenhuma criptografia [Wood et al. 2017].

As redes vestíveis tem um modelo de rede e comunicação segmentado e interdependente. Seu ecossistema abrange um grande número de dispositivos heterogêneos, protocolos de comunicação e mecanismos de processamento. Portanto, analisar a informação

correspondente a uma situação adversa torna-se uma tarefa complexa. Considerando as características dinâmicas dos dispositivos móveis, o ambiente vestível está em constante e drástica mudança. Portanto, devido a natureza intricada das redes vestíveis, os mecanismos convencionais de segurança tornam-se inadequados para proteção contra as ameaças que estão se tornando cada vez mais sofisticadas [Hassija et al. 2019]. Os mecanismos de segurança atuais dependem principalmente de informações estáticas ou de uma parte específica das informações contextuais coletadas em um uma infraestrutura específica, sendo inadequadas para fornecer segurança em uma arquitetura dinâmica.

Para oferecer suporte a uma ampla gama de aplicações, as redes vestíveis empregam várias tecnologias de comunicação sem fio com alcances distintos. Nas redes vestíveis convencionais, o vestível utiliza uma tecnologia de comunicação sem fio de curta distância para se comunicar com o coordenador. Apesar da pequena distância propiciada pela tecnologia, é possível que um atacante dentro do alcance capture os dados transmitidos, ou utilize o canal de comunicação para atacar diretamente o vestível ou o coordenador. Os ataques em que um invasor monitora os canais de comunicação e captura os dados da rede representam uma ameaça à confidencialidade enquanto ataques que visam desabilitar serviços nos dispositivos representam uma ameaça à disponibilidade. Ataques que modificam os dados em trânsito ou armazenados representam uma ameaça à integridade dos dados. Estruturas na nuvem representa uma parte fundamental das redes vestíveis e também pode ser alvo de ataques que ameaçam à confidencialidade, a disponibilidade e a integridade dos dados. A Figura 4.11 ilustra as principais ameaças de segurança nas redes vestíveis.

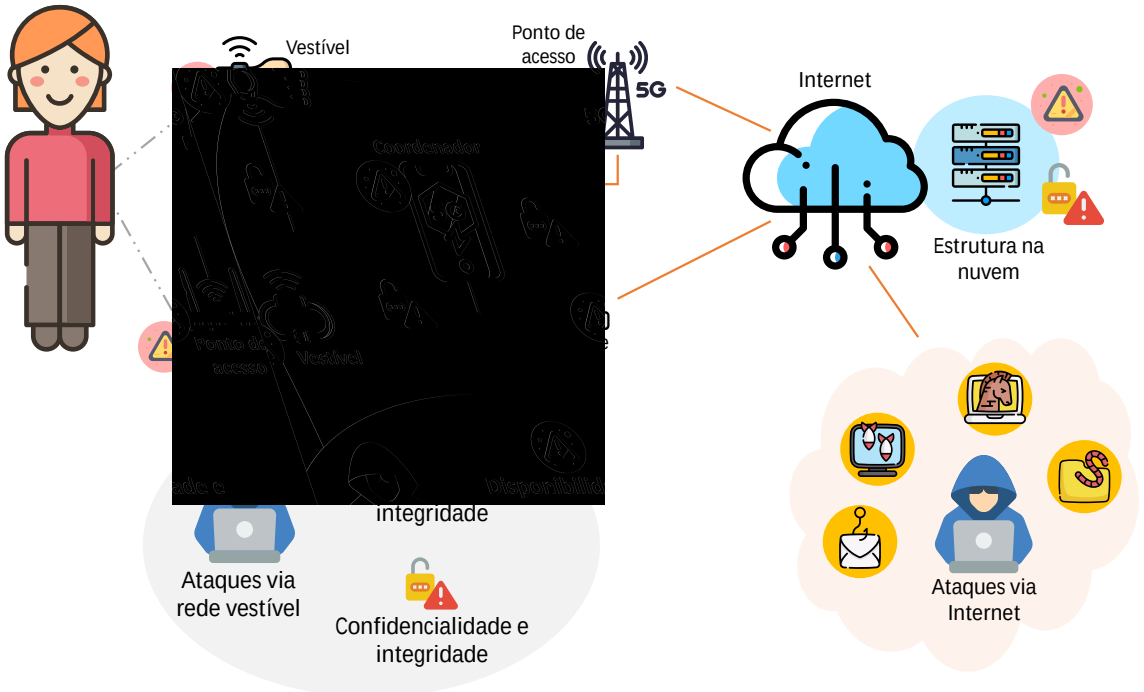


Figura 4.11: Ameaças de segurança nas redes vestíveis

A grande maioria dos ataques contra a disponibilidade é de negação de serviço, do inglês *Denial of Service* (DoS). Esses ataques visam inundar a rede vestível com uma grande quantidade de informação supérflua e podem causar interrupções na comunicação

e sobrecarga no armazenamento de dados dos vestíveis. As tecnologias de comunicação empregadas nas redes vestíveis possuem mecanismos de prevenção aos ataques DoS como sistemas de autorização e autenticação na fase de emparelhamento dos dispositivos. Todavia, nem sempre os fabricantes aplicam esses sistemas ou fazem de maneira ineficiente, deixando os dispositivos suscetíveis aos ataques DoS [Seneviratne et al. 2017]. As estruturas na nuvem podem ser alvo de um ataque ainda mais elaborado, a negação de serviço distribuída (DDoS). Nessa modalidade um atacante infecta uma série de dispositivos (computadores, roteadores de banda larga) através de técnicas de engenharia social como o *Phishing*, e usando códigos maliciosos como *Trojans* e *Worms*. Os dispositivos infectados formam uma rede estruturada sob domínio do atacante chamada *BotNet*, que pode ser usada de forma orquestrada para realizar ataques às estruturas na nuvem. Embora os ataques às estruturas de apoio não impossibilitem o funcionamento da rede vestível, eles podem afetar diretamente o funcionamento pleno do sistema.

Os ataques que ameaçam a confidencialidade envolvem a coleta de informações do usuário ou do dispositivo e o uso não autorizado dessas informações. Eventualmente, esses ataques visam os dados armazenados nos vestíveis ou na nuvem, mas em geral envolvem o monitoramento e interceptação dos dados transmitidos através dos canais de comunicação [Nakayama et al. 2019]. Devido às características das redes sem fio (comunicação pervasiva, curta distância) o principal ataque é o de espionagem, do inglês *Eavesdropping*. Esse ataque consiste na interceptação em tempo real de uma comunicação privada e representa um problema notável de segurança pois coloca em risco as informações pessoais de um usuário. Adicionalmente, ao revelarem informações do usuário e do dispositivo (senhas, números de identificação pessoais), esses ataques podem servir como porta de entrada para ataques mais elaborados. Outro ataque que segue o mesmo padrão é o ataque de análise de tráfego. Nesse tipo de ataque o invasor utiliza o modo promíscuo de uma interface sem fio para monitorar todo o tráfego ao seu alcance. Posteriormente, o invasor analisa o tráfego da rede e consegue identificar padrões que permitem relacionar um usuário a um dispositivo. Ataques de espionagem e análise de tráfego são considerados passivos pois somente monitoram a rede vestível.

As informações coletadas através de ataques de espionagem e análise de tráfego são empregadas para ataques de obtenção de informações, do inglês *Information Gathering Attacks*, que visam identificar as chaves empregadas no emparelhamento dos dispositivos. A partir da chave obtida e utilizando um equipamento imitando o dispositivo original, os atacantes ganham acesso a dispositivos vestíveis e subtraem informações confidenciais. Ataques de informações colaterais, do inglês *Side-channel Attacks*, empregam a coleta de informações para obter dados indiretamente relacionados ao ataque desejado, por exemplo capturar os movimentos do pulso do usuário a partir de um *smartwatch* para inferir a digitação de uma senha em um teclado numérico [Maiti et al. 2015]. Além dos ataques convencionais direcionados aos canais de comunicação, os softwares maliciosos para vestíveis também estão se proliferando. A incidência de softwares maliciosos é menor nos dispositivos com sistemas operacionais embarcados devido à complexidade de elaboração. Entretanto, com a popularização de sistemas operacionais específicos para vestíveis como o watchOS da Apple e o Google Wear, a tendência é de crescimento na quantidade de softwares maliciosos explorando falhas nos sistemas.

Um dos principais problemas de segurança nas redes vestíveis é a falta de cripto-

grafia dos dados enviados pela rede sem fio e também no armazenamento das informações. Vários dispositivos comerciais armazenam os dados coletados em uma base local, desprotegida, e em formato de texto simples [Seneviratne et al. 2017], incentivando e facilitando os ataques. A falta de criptografia se torna um problema ainda mais grave pois os fabricantes dos dispositivos enviam os dados para múltiplas estruturas de processamento e armazenamento na nuvem, aumentando os pontos de interceptação para um atacante. Outro problema que afeta os dispositivos vestíveis é a falta de autenticação. Alguns vestíveis são desprovidos de telas para exibição de informações ou toque, tornando complexa a tarefa de implantar mecanismos para inserção de senhas. Ressalta-se que essas falhas de segurança estão mais relacionadas ao desenvolvimento inadequado por parte dos fabricantes do que limitações nas tecnologias de comunicação. Mesmo considerando as dificuldades de aplicar mecanismos de segurança aos vestíveis por conta de seus recursos limitados, é papel do fabricante oferecer um uso descomplicado e seguro dos dispositivos.

É importante assegurar que os dados armazenados ou em trânsito não sejam alterados e sejam recebidos somente por entidades autorizadas. Os principais ataques que ferem a integridade dos dados são os de modificação, *replay* e de mascaramento. Nos ataques de modificação o atacante intercepta a transmissão de dados entre os vestíveis e altera o conteúdo dos pacotes ou os valores de data/hora. Dessa forma o atacante pode forjar um problema físico (arritmia, queda na oxigenação do sangue) e também a hora e data do suposto problema. Nos ataques de *replay* um atacante intercepta uma quantia de dados para reutilizar em uma situação posterior, uma mensagem requisitando o uso de um sensor, por exemplo. Se aceita, a mensagem falsa pode acionar um sensor ou atuador e causar a aplicação incorreta de um medicamento, ou causar exaustão da bateria pelo funcionamento contínuo. Nos ataques de mascaramento, o atacante personifica ou clona o dispositivo autenticado para atuar como legítimo. O objetivo é subtrair os dados ou injetar informações falsas no sistema. Dispositivos intracorporais controlados remotamente também pode ser alvo de ataques. Exitem exemplos na literatura onde atacantes remotos conseguem acesso às bombas de insulina, marcapassos, e implantes médicos [Zheng et al. 2017].

Quando empregadas na área de saúde, as redes vestíveis suportam o conceito da internet das coisas médicas (IoHT). Para as aplicações da IoHT, além das práticas de segurança citadas, é indispensável aplicar mecanismos para aumentar a resiliência da comunicação. As aplicações na IoHT lidam com dados que exigem urgência em sua transmissão pois o atraso ou falta de informações representam risco à vida do usuário. Isso torna os requisitos de comunicação extremamente restritos (baixa latência, baixa variação no atraso) e difíceis de serem cumpridos. Ao observar a estrutura da IoHT três serviços essenciais se destacam: (i) os serviços de coleta de dados, (ii) os serviços de conectividade e (iii) os serviços de comunicação fim-a-fim. A Figura 4.12 ilustra os serviços essenciais da IoHT. Os serviços de coleta atuam sobre os dispositivos; os serviços de conectividade atuam na comunicação entre dispositivos, coordenadores, e pontos de acesso; e os serviços de comunicação fim-afim oferecem comunicação multisaltos entre a rede vestível e as estruturas de suporte na nuvem.

Os requisitos de comunicação rigorosos, as dificuldades em estabelecer mecanismos de segurança, e as tecnologias de comunicação heterogêneas, tornam árdua a aplicação de mecanismos de resiliência na IoHT. Para cada serviço essencial, uma série de condições devem ser cumpridas para atingia a resiliência da comunicação. No serviços de

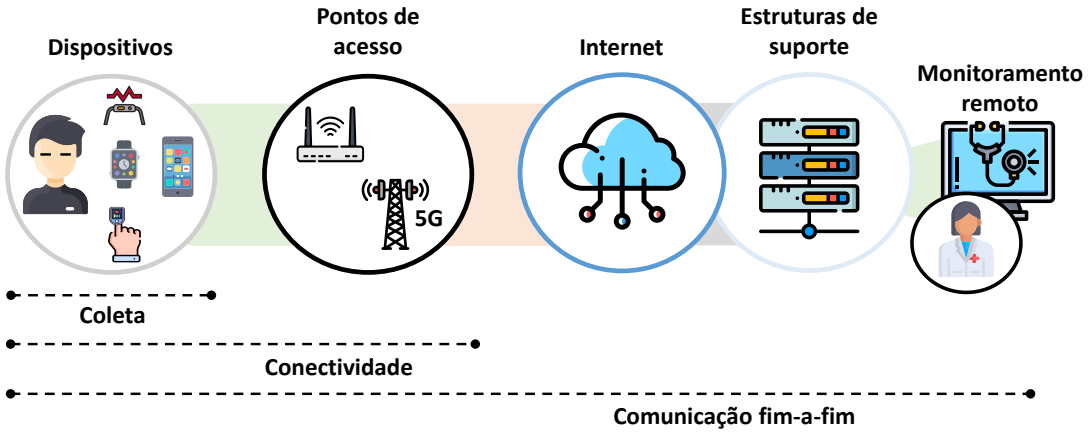


Figura 4.12: Internet das coisas médicas (IoHT)

coleta de dados, analisa-se a necessidade de redundância nos dados coletado (múltiplos sensores, múltiplos dispositivos) e quais as ações para armazenar esses dados no caso de falhas de comunicação. No serviço de conectividade, observa-se a interferência entre as tecnologias de comunicação sem fio para promover a coexistência e analisar a possibilidade de redundância nas tecnologias de comunicação. No serviço de comunicação fim-a-fim, lida-se com os possíveis atrasos ao transferir dados para a nuvem e a aplicação de mecanismos de segurança que assegurem a integridade e confidencialidade dos dados.

4.5. Integração das redes em direção aos sistemas ciber-humanos

O objetivo dos sistemas ciber-humanos é criar um ambiente integrado para que dispositivos computacionais e seres humanos possam cooperar e convergir para um mesmo propósito. Para alcançar esse objetivo, torna-se necessário gerenciar a operação entre as tecnologias de comunicação existentes e vislumbrar os possíveis cenários de crescimento, incluindo novas tecnologias e aplicações. A habilidade de um sistema trabalhar em conjunto ou usar componentes de outros sistemas é chamada interoperabilidade. Atualmente o ecossistema ciber-humano é construído de forma granular e com baixa adaptação entre as diversas tecnologias de comunicação e os dispositivos participantes, dificultando a interoperabilidade. A geração atual dos sistemas ciber-humanos foi construída sobre os fundamentos da IoT e herdou muitos dos seus problemas conceituais. Os principais problemas incluem a pobre interoperabilidade considerando as tecnologias de comunicação e protocolos, a formatação dos dados, e a adaptação entre serviços em níveis operacionais distintos (coleta de dados, conectividade e comunicação fim-a-fim).

Atualmente as tecnologias de comunicação em sistemas ciber-físicos operam verticalmente dentro dos serviços. A integração compreende somente a comunicação dentro de um serviço específico, dificultando a interoperabilidade. Entretanto, novas tecnologias podem alterar esse panorama estabelecido, e as redes vestíveis podem ser um ponto de partida. Umas das possibilidade para promover a integração das comunicações nos sistemas ciber-humanos é a tecnologia de comunicação dispositivo a dispositivo (D2D), já empregada nas redes vestíveis. A comunicação D2D não necessita de uma infraestrutura complexa de comunicação uma vez que os dispositivos se comunicam diretamente entre si. Atualmente, as redes vestíveis aplicam a comunicação D2D verticalmente e sem con-

siderar a conexão com a Internet, na comunicação entre vestível e coordenador, por exemplo. Todavia, com a introdução de dispositivos vestíveis que se conectem diretamente à Internet, a comunicação nos sistemas ciber-humanos pode alcançar novos patamares.

A comunicação D2D apoiada por tecnologias de comunicação celular como a LTE, permitirá a conexão direta dos dispositivos vestíveis com a Internet. Essa configuração reduz sensivelmente os problemas de comunicação causados pela falta de compatibilidade entre os dispositivos na rede vestível. No entanto, antes da inclusão em massa das novas tecnologia às redes vestíveis, alguns pontos devem ser considerados. A incorporação de novas tecnologias de comunicação eleva o preço dos dispositivos, reduzindo a velocidade de aceitação. Novos protocolos de comunicação devem ser aplicados aos vestíveis para comportar as novas tecnologias, porém, deve-se respeitar às características dos dispositivos que serão mantidas como baixo poder de processamento, baixa quantidade de memória e baterias de pequeno porte, entre outras. Finalmente, as novas tecnologias de comunicação atenderão à próxima geração de dispositivos, não sendo uma solução para enorme gama de dispositivos disponíveis atualmente.

No presente, a falta de interoperabilidade é o principal problema no tocante à comunicação nos sistemas ciber-físicos. Novas tecnologias de comunicação podem minimizar os problemas no futuro mas não representam uma solução para a atual geração de sistemas ciber-humanos. As principais propostas para os sistemas atuais envolvem o gerenciamento dos serviços de forma isolada e sem integração. Embora essas propostas sejam suficientes para manter o funcionamento das aplicações atuais, novos propostas como a Indústria 5.0, Sociedade 5.0, e Internet Sem Coisas [Maier et al. 2020], vão exigir maiores níveis de integração e operabilidade para funcionarem perfeitamente.

4.6. Ferramentas, simuladores e ambientes de experimentação

Esta seção apresenta as principais ferramentas empregadas pela comunidade científica em pesquisas relacionadas às redes vestíveis, sistemas ciber-físicos e ciber-humanos e nanorredes. Essas ferramentas incluem simuladores e emuladores de redes, plataformas de software e hardware abertos, entre outros. O estudo experimental de sistemas de comunicação molecular é uma tarefa difícil pois a experimentação em laboratório apresenta alto custo e a realização destes consumem muito tempo. Portanto, os avanços neste campo de pesquisa ocorrem principalmente através de simulações que são utilizadas para identificar e avaliar novas soluções de comunicação. Devido ao número considerável de propostas de canais para os sistemas ciber-físicos baseados em sistemas biológicos ou não, inúmeras ferramentas de simulações vêm sendo propostas nos últimos anos. A Tabela 4.4 apresenta os simuladores propostos e as informações associadas ao tipo de comunicação utilizada, à forma de propagação (*i.e.*, a forma de disseminação da portadora de informação), a linguagem de programação e se dispõe a opção *open source* (*i.e.*, "código aberto", o código fonte pode ser adaptado para diferentes fins). As principais características dos simuladores são apresentadas a seguir.

A ferramenta BINS para nanorredes biológicas foi desenvolvida para simular a troca de informações em canais baseados em difusão livre. BiNS apresenta uma abordagem "*plug-in-play*"para diferentes tipos de transmissores, canais, receptores, e é adaptável a qualquer tipo de partícula de informação [Felicetti et al. 2012]. Como forma de valida-

ção da metodologia de simulação, o sistema imunológico do corpo humano foi implementado no nBiNS e apresentado pelos autores. No sistema, os vasos sanguíneos fazem o papel do canal de comunicação para a propagação de moléculas do tipo IL-4 cytokines, as quais realizam o movimento aleatório entre o transmissor e o receptor. O nó transmissor é uma célula do tipo linfócito T e a célula linfócito B, o nó receptor. Os resultados obtidos no ambiente de simulação foram validados com dados experimentais.

O BNSIM (do inglês, Bacteria Network Simulator) é um simulador Java com paralelização de tarefas para simular redes de comunicação molecular baseadas em bactérias [Wei et al. 2013]. BNSim apresenta várias propriedades de redes bacterianas sintéticas e naturais. A plataforma de modelagem integra circuitos genéticos, modelos de comunicações químicas em um ambiente molecular 3D, e diferentes algoritmos de simulação (*i.e.*, algoritmo de Gillespie, equações diferenciais estocásticas, e um algoritmo híbrido que integra os dois métodos). Além disso, a ferramenta fornece orientações para a implementação do sistema de comunicação bacteriana em ambiente real.

Simulador	**Comunicação**	**Propagação**	**Linguagem / Ferramenta**	**Open Source**
BiNS	CM	Difusão livre	Java	Não
BNSIM	CM	Bactérias	Java	Sim
CalComSim	CM	Sinalização Celular	Python	Sim
dMCS	CM	Difusão livre	Java	Não
MUCIN	CM	Difusão livre	MATLAB	Sim
NanoNS	CM	Difusão livre	NS-2, Tcl, Java	Não
Nano-Sim	Eletromagnética	Banda Terahethz	NS-3	Sim
NCSim	CM	Bactérias	C++	Sim
Neuron Simulator	CM	Neurônios	Python	Sim
N3Sim	CM	Difusão livre	Java	Sim

Tabela 4.4: Simuladores para nanorredes

O simulador de comunicação molecular baseada em sinalização de cálcio CalComSim (do inglês, Calcium Signalling-Based Molecular Communication System Simulator) foi projetado para simular tanto as comunicações sintéticas quanto as naturais encontradas no tecido humano [Barros et al. 2015]. O simulador integra diferentes modelos do processo de sinalização celular e o fechamento e abertura estocásticos das junções comunicantes entre as células. O algoritmo estocástico de Guillespie é aplicado para simular as reações múltiplas e paralelas em cada célula. Todos os modelos biológicos que são incorporados no simulador são baseados em dados experimentais e consideram um cenário 3D. O simulador pode ser usado não apenas por engenheiros de telecomunicações, mas também por farmacêuticos para projetar novos medicamentos e tratamentos para doenças que prejudicam a sinalização de cálcio.

O dMCS é uma arquitetura de simulação de alto nível para comunicação molecular com foco na escalabilidade [Akkaya et al. 2014]. A arquitetura visa simular o canal de comunicação baseado em difusão livre, e não considerara os efeitos dos ruídos e da interferência no sistema. Os resultados da validação do dMCS mostram que diferentes opções de escalabilidade podem ser usadas para se beneficiar do poder de processamento e encurtar o tempo de execução das simulações. O simulador MolecUlar CommunicatIoN (MUCIN) também foi proposto para sistemas de CM baseados em difusão livre.

Este simulador suporta o ambiente molecular 1D e 3D, o envio de símbolos consecutivos, a recepção imperfeita de moléculas, a modulação/demodulação e uma opção para filtragem da interferência intersimbólica. O simulador está disponível na central de troca de arquivos do MATLAB. O código-fonte está disponível sob licença *Berkeley Software Distribution* BSD para pesquisadores.

A estrutura do NanoNS é construída sobre os principais componentes do simulador NS-2 e escrita em C++ e Tcl [Gul et al. 2010]. Ele incorpora os módulos de simulação para a comunicação em nanoescala com base em um canal de comunicação molecular difusivo. Os autores focaram principalmente nas propriedades do canal e no número de moléculas recebidas em função do tempo. O algoritmo divide o meio de propagação em malha. Assim, a posição exata da partícula não foi avaliada. No NanoNS, o processo de reação que descreve a interação molecular entre os nós e moléculas foi modelado de três maneiras diferentes, sendo uma delas sem reação. A validação do simulador foi realizada através da analise comparativa dos resultados experimentais e numéricos obtidos.

O NCSim simula nanorredes biológicas de bactérias flageladas [Balasubramaniam and Lio 2013]. As informações são codificadas em diferentes formatos pelo DNA dessas bactérias. Após a codificação, as bactérias transmitem as informações para nós *relays*, ou receptores que são outras bactérias do mesmo tipo. O NCSim consiste nos diferentes módulos: (*i*) camada física de rede bacteriana, que integra a mobilidade de bactérias, conjugação, técnicas de codificação e decodificação, (*ii*) modulo para gerar cenários e monitorar as simulações e (*iii*) módulo destinado ao pós-processamento de dados e plotagem de gráficos. O módulo da camada física, como o mais intensivo computacional, é implementado em C++. Os demais módulos são escritos em Python para simplicidade de manutenção e extensão. Scripts em Python são necessários para a definição de cenários por parte dos usuários.

NEURON é um ambiente utilizado para implementar modelos de neurônios individuais e pequenas redes neuronais [Hines and Carnevale 2001]. A ferramenta possui uma extensão para a linguagem de programação Python e foi desenvolvida pela Universidade de Yale. Devido a sua precisão em implementar modelos bio-físicos (processo de propagação de impulsos neuronais e a propagação das sinapses), o simulador é bastante popular na disciplina da neurociência. Os modelos bio-físicos podem ser adaptados para simular experimentos *in-vitro*, onde a partir de uma imagem da célula, a configuração morfológica dela pode ser obtida para compor um novo modelo. O simulador é utilizado pelo *Blue Brain Project*, uma iniciativa Europeia de digitalização do córtex cerebral, os dados desse projeto podem ser acessados virtualmente e seus modelos podem utilizados por pesquisadores.

N3Sim é um pacote Java para simular a comunicação baseada em difusão livre. Diferente dos demais simuladores, o processo de comunicação considera receptores não absorventes (*i.e.*, os nós transmissores podem ter um determinado volume e mensurar as moléculas emitidas pela sua reflexão, mas não possui a capacidade de absorver as moléculas)[Llatser et al. 2014]. O receptor realiza a contagem das partículas localizadas dentro de uma determinada área próxima a ele durante intervalos de tempo predefinidos. N3Sim simula a difusão de partículas em um ambiente 2-D, com uma expansão contínua para modelos de processo de difusão 3-D. O simulador considera fontes de ruídos e erros

Ambientes experimentais para redes vestíveis		
Simuladores		
Nome	Descrição	Open Source
NS2	Simulador de rede	Sim
NS-3	Simulador de rede	Sim
OMNeT++	Biblioteca C++ para construção de simulações	Sim
OPNET	Simulador de rede com ambiente visual para modelagem	Não
Programas		
Nome	Característica	Open Source
Matlab	Plataforma de programação e cálculo numérico que pode simular o comportamento de aspectos da rede vestível (mobilidade, segurança)	Não
R Project	Ambiente para computação estatística e produção de gráficos que pode ser utilizada para simular o comportamento de aspectos da rede vestível	Sim
AVISPA	Programa para análise de segurança de aplicações e protocolos da Internet	Não
Scyther	Programa para verificação de protocolos de segurança	Sim
Alloy	Programa para modelagem e avaliação de segurança (entre outros parâmetros) dos modelos construídos	Sim
Tamarin	Programa para verificação de segurança de protocolos de modelos simbólicos	Sim
Ferramentas adicionais		
Tipo	Funcionalidades	
Microcontroladores	Simular a capacidade de processamento de um dispositivo vestível comercial	
Baterias	Permitir a avaliação da eficiência energética nos ambientes de teste	
Conjunto de dados	Testar os ambientes em situações de mundo real através de dados previamente armazenados	
Sensores	Coletar dados que complementem o ambiente experimental (sinais biológicos, movimento, som, imagem, etc)	

Tabela 4.5: Ferramentas para avaliação das redes vestíveis

através de programas específicos, e a avaliação de desempenho do protocolo conduzida em ambiente experimental físico ou virtual. Outro aspecto da segurança que também é alvo de experimentação é a autenticação através de senhas, características biométricas e proximidade dos dispositivos, entre outros.

Avaliações de segurança que incluem a autenticação de usuários geralmente são conduzidas em ambiente de experimentação real pois consideram a carga e o tempo de processamento do dispositivo avaliado. Nos casos em que as características biométricas (variação cardíaca, impressão digital) são utilizadas para autenticação, geralmente emprega-se um conjunto de dados para avaliar o mecanismo construído, por exemplo uma base de sinais contendo padrões de frequência cardíaca para validar um mecanismo de autenticação baseado na variação cardíaca do usuário. Eventualmente, a construção dos conjuntos de dados pode fazer parte do experimento. Nesses casos, através de sensores específicos (sinais biológicos, movimento, som) acoplados a microcontroladores, cria-se uma base de dados para experimentação. A complexidade e diversidade de tenologias de comunicação e parâmetros de segurança nas redes vestíveis torna praticamente inviável o uso de uma ferramenta exclusiva para simulações.

4.7. Considerações finais

A evolução dos dispositivos computacionais rumo à miniaturização pavimentou o caminho para novas tecnologias como as redes vestíveis e as nanorredes. Essas novas tecnologias por sua vez apoiam o surgimento de novos conceitos, como os sistemas ciber-humanos, por exemplo. De fato, os pilares fundamentais para a integração dos dispositivos vestíveis e das nanorredes aos sistemas ciber-físicos e ciber-humanos serão a pesquisa

e o desenvolvimento tecnológico em pontos chave como comunicação, energia sustentável e progressos na miniaturização dos componentes, entre outros.

Neste capítulo de livro, apresentamos um contexto histórico para evolução das tecnologias que constituem as redes vestíveis e as nanorredes e que culminaram na disseminação de dispositivos vestíveis e em nanoescala para uma variedade de aplicações. A introdução dos conceitos fundamentais, considerando os sistemas ciber-humanos, ciber-físicos, dispositivos e redes vestíveis e nanorredes, abre caminho para uma melhor percepção sobre as funcionalidades e dificuldades de integração das tecnologias. A grande variedade de tecnologias de comunicação presente nos sistemas ciber-humanos e a diversidade de configurações propiciadas por essas tecnologias alancam uma infinidade de aplicações médicas e não médica. Entretanto, a multiplicidade de tecnologias e a sua integração representam um dos desafios a serem transpostos.

Grande parte das informações coletadas por nanodispositivos e vestíveis representam características pessoais, tornando-se necessário aplicar medidas que assegurem a segurança do usuário e dos dispositivos, a privacidade das informações e a resiliência na comunicação. A variedade de dispositivos, protocolos e tecnologias de comunicação e mecanismos de segurança dificulta a integração das redes em direção aos sistemas ciber-humanos. A dificuldade ao construir ambientes de experimentação virtuais ou práticos reflete a falta de integração nos sintemas ciber-humanos. Da mesma forma que os sistemas ciber-humanos carecem de maior integração das ferramentas de simulação, estas ainda não contemplam toda a complexidade desses sistemas, sendo necessária a aplicação de diversas ferramentas para criar um ambiente experimental.

As redes vestíveis e nanorredes, assim como os sistemas ciber-humanos, estão em sua infância e têm um grande potencial de desenvolvimento. Grande parte da evolução acontecerá em consonância com o avanço tecnológico, mas para que os sistemas ciber-físicos se tornem uma realidade mais rapidamente, serão necessárias ações que contribuam para uma maior integração entre as tecnologias existentes. A inclusão de novas tecnologias de comunicação nas nanorredes e nas redes vestíveis permitirá novas possibilidades de integração para os sistemas ciber-humanos e sua utilização em novas aplicações. No entanto, a forte conexão entre os componentes dos sistemas proprietários é um dos principais desafios, pois desconsidera os padrões existentes e inibe a interoperabilidade no ecossistema ciber-humano.

No decorrer dos próximos anos observaremos um grande avanço nas tecnologias que impulsionam as nanorredes e redes vestíveis. Vários desafios ainda devem ser confrontados como melhorias no processo de aquisição de dados e processamento, aprimoramentos na comunicação, segurança, privacidade e resiliência. Somente ao superar esses desafios poderemos idealizar um ambiente de cooperação entre máquinas, computadores e seres humanos que seja ao mesmo tempo funcional, integrado e seguro.

Agradecimentos

Os autores agradecem o apoio da UFPR e da UFMG e o auxílio financeiro do CNPq processos, #313844 /2020-8, #426701/2018-6 e da CAPES, processos #88882.382196/2019-01 e #88882.382204/2019-01.

Referências

[Akkaya et al. 2014] Akkaya, A., Genc, G., and Tugcu, T. (2014). Hla based architecture for molecular communication simulation. Simulation Modelling Practice and Theory, 42:163–177.

[Akyildiz 2012] Akyildiz, I. F. (2012). Nanonetworks: A new frontier in communications. In Proceedings of the 18th annual international conference on Mobile computing and networking, pages 1–2. ACM.

[Akyildiz and Jornet 2010] Akyildiz, I. F. and Jornet, J. M. (2010). Electromagnetic wireless nanosensor networks. Nano Communication Networks, 1(1):3–19.

[Akyildiz et al. 2019] Akyildiz, I. F., Pierobon, M., and Balasubramaniam, S. (2019). Moving forward with molecular communication: From theory to human health applications [point of view]. Proceedings of the IEEE, 107(5):858–865.

[Akyildiz et al. 2015] Akyildiz, I. F., Pierobon, M., Balasubramaniam, S., and Koucheryavy, Y. (2015). The internet of bio-nano things. IEEE Communications Magazine, 53(3):32–40.

[Al-Shawabka et al. 2020] Al-Shawabka, A., Restuccia, F., D'Oro, S., Jian, T., Costa Rendon, B., Soltani, N., Dy, J., Ioannidis, S., Chowdhury, K., and Melodia, T. (2020). Exposing the ngerprint: Dissecting the impact of the wireless channel on radio ngerprinting. In IEEE INFOCOM 2020, pages 646–655.

[Al-Turjman and Baali 2019] Al-Turjman, F. and Baali, I. (2019). Machine learning for wearable iot-based applications: A survey. Transactions on Emerging Telecommunications Technologies, page e3635.

[Analysts 2020] Analysts, G. I. (2020). Wearable sensors - global market trajectory and analytics. Technical report, Global Industry Analysts, Inc.

[Ashibani and Mahmoud 2017] Ashibani, Y. and Mahmoud, Q. H. (2017). Cyber physical systems security: Analysis, challenges and solutions. Computers & Security, 68:81–97.

[Balasubramaniam and Lio 2013] Balasubramaniam, S. and Lio, P. (2013). Multi-hop conjugation based bacteria nanonetworks. IEEE Transactions on NanoBioscience, 12(1):47–59.

[Barbone et al. 2019] Barbone, A. S., Meftah, M., Markiewicz, K., and Dellimore, K. (2019). Beyond wearables and implantables: a scoping review of insertable medical devices. Biomedical Physics & Engineering Express, 5(6):062002.

[Baron et al. 2018] Baron, K. G., Duffecy, J., Berendsen, M. A., Mason, I. C., Lattie, E. G., and Manalo, N. C. (2018). Feeling validated yet? a scoping review of the use of consumer-targeted wearable and mobile technology to measure and improve sleep. Sleep medicine reviews, 40:151–159.

[Barros et al. 2015] Barros, M. T., Balasubramaniam, S., and Jennings, B. (2015). Comparative end-to-end analysis of ca 2+-signaling-based molecular communication in biological tissues. IEEE Transactions on Communications, 63(12):5128–5142.

[Blasco et al. 2019] Blasco, J., Chen, T. M., Patil, H. K., and Wolff, D. (2019). Wearables security and privacy. In *Mission-Oriented Sensor Networks and Systems: Art and Science*, pages 351–380. Springer.

[Borges et al. 2021] Borges, L. F., Barros, M. T., and Nogueira, M. (2021). Toward reliable intra-body molecular communication: An error control perspective. *IEEE Communications Magazine*, 59(5):114–120.

[Chahibi et al. 2013] Chahibi, Y., Pierobon, M., Sang, O. S., and Akyildiz, I. F. (2013). A molecular communication system model for particulate drug delivery systems. *IEEE Transactions on Biomedical Engineering*, 60(12):3468–3483.

[Chude-Okonkwo et al. 2016] Chude-Okonkwo, U. A., Malekian, R., and Maharaj, B. T. (2016). Biologically inspired bio-cyber interface architecture and model for internet of bio-nanothings applications. *IEEE Trans. on Communications*, 64(8):3444–3455.

[Chude-Okonkwo et al. 2017] Chude-Okonkwo, U. A., Malekian, R., Maharaj, B. T., and Vasilakos, A. V. (2017). Molecular communication and nanonetwork for targeted drug delivery: A survey. *IEEE Communications Surveys & Tutorials*, 19(4):3046–3096.

[Classen et al. 2018] Classen, J., Wegemer, D., Patras, P., Spink, T., and Hollick, M. (2018). Anatomy of a vulnerable fitness tracking system: Dissecting the fitbit cloud, app, and firmware. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 2(1):5.

[Coelho et al. 2019] Coelho, K., Damião, D., Noubir, G., Borges, A., Nogueira, M., and Nacif, J. (2019). Cryptographic algorithms in wearable communications: An empirical analysis. *IEEE Communications Letters*, 23(11):1931–1934.

[Da Costa et al. 2009] Da Costa, M. R., Kibis, O., and Portnoi, M. (2009). Carbon nanotubes as a basis for terahertz emitters and detectors. *Microelectronics Journal*, 40(4-5):776–778.

[Datta et al. 2018] Datta, T., Apthorpe, N., and Feamster, N. (2018). A developer-friendly library for smart home IoT privacy-preserving traffic obfuscation. In *Proceedings of the Workshop on IoT Security and Privacy*.

[De Amicis et al. 2015] De Amicis, A., De Sanctis, S., Di Cristofaro, S., Franchini, V., Lista, F., Regalbuto, E., Giovenale, E., Gallerano, G. P., Nenzi, P., Bei, R., et al. (2015). Biological effects of in vitro thz radiation exposure in human foetal fibroblasts. *Mutation Research/Genetic Toxicology and Environmental Mutagenesis*, 793:150–160.

[Dhanvijay and Patil 2019] Dhanvijay, M. M. and Patil, S. C. (2019). Internet of things: A survey of enabling technologies in healthcare and its applications. *Computer Networks*, 153:113–131.

[Ding et al. 2018] Ding, D., Han, Q.-L., Xiang, Y., Ge, X., and Zhang, X.-M. (2018). A survey on security control and attack detection for industrial cyber-physical systems. *Neurocomputing*, 275:1674 – 1683.

[Ding et al. 2019] Ding, H., Shu, X., Jin, Y., Fan, T., and Zhang, H. (2019). Recent advances in nanomaterial-enabled acoustic devices for audible sound generation and detection. *Nanoscale*, 11(13):5839–5860.

[Dong et al. 2019] Dong, S., Li, Z., Tang, D., Chen, J., Sun, M., and Zhang, K. (2019). Your smart home can't keep a secret: Towards automated fingerprinting of iot traffic with neural networks. *ArXiv*, abs/1909.00104.

[FBI 2020] FBI (2015 (Accessed January 10, 2020)). *Internet of Things Poses Opportunities for Cyber Crime*.

[Felicetti et al. 2018] Felicetti, L., Assaf, S. S., Femminella, M., Reali, G., Alarcon, E., and Sole-Pareta, J. (2018). The molecular communications markup language (molcomml). *Nano communication networks*, 16:12–25.

[Felicetti et al. 2012] Felicetti, L., Femminella, M., and Reali, G. (2012). A simulation tool for nanoscale biological networks. *Nano Communication Networks*, 3(1):2–18.

[Ferro and Potorti 2005] Ferro, E. and Potorti, F. (2005). Bluetooth and wi-fi wireless protocols: a survey and a comparison. *IEEE Wireless Communications*, 12(1):12–26.

[Fouad et al. 2020] Fouad, H., Hashem, M., and Youssef, A. E. (2020). A nanobiosensors model with optimized bio-cyber communication system based on internet of bio-nano things for thrombosis prediction. *Journal of Nanoparticle Resc.*, 22:1–17.

[Gao et al. 2016] Gao, W., Emaminejad, S., Nyein, H. Y. Y., Challa, S., Chen, K., Peck, A., Fahad, H. M., Ota, H., Shiraki, H., and Kiriya, D. (2016). Fully integrated wearable sensor arrays for multiplexed in situ perspiration analysis. *Nature*, 529:509–527.

[Gul et al. 2010] Gul, E., Atakan, B., and Akan, O. B. (2010). Nanons: A nanoscale network simulator framework for molecular communications. *Nano Communication Networks*, 1(2):138–156.

[Hafeez et al. 2020] Hafeez, I., Antikainen, M., Ding, A., and Tarkoma, S. (2020). Iotkeeper: Detecting malicious IoT network activity using online traffic analysis at the edge. *IEEE Transactions on Network and Service Management*, 17:45–59.

[Hassija et al. 2019] Hassija, V., Chamola, V., Saxena, V., Jain, D., Goyal, P., and Sikdar, B. (2019). A survey on iot security: application areas, security threats, and solution architectures. *IEEE Access*, 7:82721–82743.

[Hines and Carnevale 2001] Hines, M. L. and Carnevale, N. T. (2001). Neuron: a tool for neuroscientists. *The neuroscientist*, 7(2):123–135.

[Hochberg et al. 2006] Hochberg, L. R., Serruya, M. D., Friehs, G. M., Mukand, J. A., Saleh, M., Caplan, A. H., Branner, A., Chen, D., Penn, R. D., and Donoghue, J. P. (2006). Neuronal ensemble control of prosthetic devices by a human with tetraplegia. *Nature*, 442(7099):164–171.

[Hogg and Freitas Jr 2012] Hogg, T. and Freitas Jr, R. A. (2012). Acoustic communication for medical nanorobots. *Nano Communication Networks*, 3(2):83–102.

[IBM 2020] IBM (2019 (Accessed January 18, 2020)). *Cost of a Data Breach Report*.

[IDTechEX 2021] IDTechEX (2021). Wearable technology market by 2028. `https://www.gartner.com/en/newsroom/press-releases/2021-01-11-gartner-forecasts-global-spending-on-wearable-devices-to-total-81-5-billion-in-2021`. Accessed: 2021-26-05.

[Jablonski and Chaplin 2010] Jablonski, N. G. and Chaplin, G. (2010). Human skin pigmentation as an adaptation to uv radiation. *Proceedings of the National Academy of Sciences*, 107(Supplement 2):8962–8968.

[Jia et al. 2019] Jia, X., He, J., Shen, L., Chen, J., Wei, Z., Qin, X., Niu, D., Li, Y., and Shi, J. (2019). Gradient redox-responsive and two-stage rocket-mimetic drug delivery system for improved tumor accumulation and safe chemotherapy. *Nano letters*, 19(12):8690–8700.

[Khan and Pathan 2018] Khan, R. A. and Pathan, A.-S. K. (2018). The state-of-the-art wireless body area sensor networks: A survey. *International Journal of Distributed Sensor Networks*, 14(4):1550147718768994.

[Kim et al. 2016] Kim, H., Yves, P., and Lee, K. (2016). Development of chip-less and wireless neural probe functioning stimulation and reading in a single device. *Microelectronic Engineering*, 158:118–125.

[Kong et al. 2019] Kong, X. T., Luo, H., Huang, G. Q., and Yang, X. (2019). Industrial wearable system: the human-centric empowering technology in industry 4.0. *Journal of Intelligent Manufacturing*, 30(8):2853–2869.

[Koucheryavy et al. 2021] Koucheryavy, Y., Yastrebova, A., Martins, D. P., and Balasubramaniam, S. (2021). A review on bio-cyber interfaces for intrabody molecular communications systems. *arXiv preprint arXiv:2104.14944*.

[Kubota et al. 2016] Kubota, K. J., Chen, J. A., and Little, M. A. (2016). Machine learning for large-scale wearable sensor data in parkinson's disease: Concepts, promises, pitfalls, and futures. *Movement disorders*, 31(9):1314–1326.

[Kuscu et al. 2019] Kuscu, M., Dinc, E., Bilgin, B. A., Ramezani, H., and Akan, O. B. (2019). Transmitter and receiver architectures for molecular communications: A survey on physical design with modulation, coding, and detection techniques. *Proceedings of the IEEE*, 107(7):1302–1341.

[Laprie 2008] Laprie, J.-C. (2008). From dependability to resilience. In *IEEE/IFIP Int. Conf. On Dependable Systems and Networks*, pages 1–2.

[Laprie et al. 2004] Laprie, J.-C., Randell, B., and Landwehr, C. (2004). Basic concepts and taxonomy of dependable and secure computing. *IEEE Transaction Dependable Security Computer*, 1(1):11–33.

[Llatser et al. 2014] Llatser, I., Demiray, D., Cabellos-Aparicio, A., Altilar, D. T., and Alarcón, E. (2014). N3sim: Simulation framework for diffusion-based molecular communication nanonetworks. *Simulation Modelling Practice and Theory*, 42:210–222.

[Loscri and Vegni 2015] Loscri, V. and Vegni, A. M. (2015). An acoustic communication technique of nanorobot swarms for nanomedicine applications. *IEEE transactions on nanobioscience*, 14(6):598–607.

[Losilla et al. 2011] Losilla, F., Garcia-Sanchez, A.-J., Garcia-Sanchez, F., Garcia-Haro, J., and Haas, Z. J. (2011). A comprehensive approach to wsn-based its applications: A survey. *Sensors*, 11(11):10220–10265.

[Maier et al. 2020] Maier, M., Ebrahimzadeh, A., Rostami, S., and Beniiche, A. (2020). The internet of no things: Making the internet disappear and"see the invisible". *IEEE Communications Magazine*, 58(11):76–82.

[Maiti et al. 2015] Maiti, A., Jadliwala, M., He, J., and Bilogrevic, I. (2015). (smart) watch your taps: Side-channel keystroke inference attacks using smartwatches. In *Proceedings of the 2015 ACM Int. Symposium on Wearable Computers*, pages 27–30.

[Mattson 2004] Mattson, M. P. (2004). Pathways towards and away from alzheimer's disease. *Nature*, 430(7000):631–639.

[Mcguiness et al. 2019] Mcguiness, D. T., Selis, V., and Marshall, A. (2019). Molecular-based nano-communication network: A ring topology nano-bots for in-vivo drug delivery systems. *IEEE Access*, 7:12901–12913.

[Moore 1998] Moore, G. E. (1998). Cramming more components onto integrated circuits. *Proceedings of the IEEE*, 86(1):82–85.

[Movassaghi et al. 2014] Movassaghi, S., Abolhasan, M., Lipman, J., Smith, D., and Jamalipour, A. (2014). Wireless body area networks: A survey. *IEEE Communications surveys & tutorials*, 16(3):1658–1686.

[Muller et al. 2014] Muller, R., Le, H.-P., Li, W., Ledochowitsch, P., Gambini, S., Bjorninen, T., Koralek, A., Carmena, J. M., Maharbiz, M. M., Alon, E., et al. (2014). A minimally invasive 64-channel wireless μecog implant. *IEEE Journal of Solid-State Circuits*, 50(1):344–359.

[Nakayama et al. 2019] Nakayama, F., Lenz, P., Banou, S., Nogueira, M., Santos, A., and Chowdhury, K. R. (2019). A continuous user authentication system based on galvanic coupling communication for s-health. *Wireless Commun. and Mobile Computing*, 2019(1):1–11.

[Nakayama et al. 2021] Nakayama, F., Lenz, P., LeFloch, A., Beylot, A.-L., Santos, A., and Nogueira, M. (2021). Performance management on multiple communication paths for portable assisted living. *IFIP Int. Symposium on Integrated Network Management*.

[Neema et al. 2019] Neema, H., Sztipanovits, J., Steinbrink, C., Raub, T., Cornelsen, B., and Lehnhoff, S. (2019). Simulation integration platforms for cyber-physical systems. In *Proceedings of the Workshop on Design Automation for CPS and IoT*, pages 10–19.

[Negra et al. 2016] Negra, R., Jemili, I., and Belghith, A. (2016). Wireless body area networks: Applications and technologies. *Procedia Computer Science*, 83:1274–1281.

[Nogueira et al. 2009] Nogueira, M., dos Santos, A. L., and Pujolle, G. (2009). A survey of survivability in mobile ad hoc networks. *IEEE Commun. Surv. Tutorials*, 11:66–77.

[Ometov et al. 2021] Ometov, A., Shubina, V., Klus, L., Skibińska, J., Saafi, S., Pascacio, P., Flueratoru, L., Gaibor, D. Q., Chukhno, N., Chukhno, O., et al. (2021). A survey on wearable technology: History, state-of-the-art and current challenges. *Computer Networks*, 193.

[Oresko et al. 2010] Oresko, J. J., Jin, Z., Cheng, J., Huang, S., Sun, Y., Duschl, H., and Cheng, A. C. (2010). A wearable smartphone-based platform for real-time cardiovascular disease detection via electrocardiogram processing. *IEEE Transactions on Information Technology in Biomedicine*, 14(3):734–740.

[Perez and Zeadally 2017] Perez, A. J. and Zeadally, S. (2017). Privacy issues and solutions for consumer wearables. *It Professional*, 20(4):46–56.

[Pfeiffer et al. 2018] Pfeiffer, M., Senkovskiy, B. V., Haberer, D., Fischer, F. R., Yang, F., Meerholz, K., Ando, Y., Grüneis, A., and Lindfors, K. (2018). Enhanced light–matter interaction of aligned armchair graphene nanoribbons using arrays of plasmonic nanoantennas. *2D Materials*, 5(4):045006.

[Pinheiro 2020] Pinheiro, P. P. (2020). *Proteção de Dados Pessoais: Comentários à Lei n. 13.709/2018-LGPD*. Saraiva Educação SA.

[Piro et al. 2013] Piro, G., Grieco, L. A., Boggia, G., and Camarda, P. (2013). Simulating wireless nano sensor networks in the ns-3 platform. In *2013 27th Int. Conference on Advanced Information Networking and Applications Workshops*, pages 67–74. IEEE.

[Poongodi et al. 2020] Poongodi, T., Rathee, A., Indrakumari, R., and Suresh, P. (2020). Iot sensing capabilities: sensor deployment and node discovery, wearable sensors, wireless body area network (wban), data acquisition. In *Principles of Internet of Things (IoT) Ecosystem: Insight Paradigm*, pages 127–151. Springer.

[Posada et al. 2015] Posada, J., Toro, C., Barandiaran, I., Oyarzun, D., Stricker, D., De Amicis, R., Pinto, E. B., Eisert, P., Döllner, J., and Vallarino, I. (2015). Visual computing as a key enabling technology for industrie 4.0 and industrial internet. *IEEE computer graphics and applications*, 35(2):26–40.

[Prates et al. 2020] Prates, N., Vergutz, A., Macedo, R., Santos, A., and Nogueira, M. (2020). A defense mechanism for timing-based side-channel attacks on IoT traffic (to appear). In *IEEE GLOBECOM*.

[Rajkumar et al. 2010] Rajkumar, R., Lee, I., Sha, L., and Stankovic, J. (2010). Cyber-physical systems: The next computing revolution. In *Design Automation Conference*, pages 731–736.

[Resch 2013] Resch, B. (2013). People as sensors and collective sensing-contextual observations complementing geo-sensor network measurements. In *Progress in location-based services*, pages 391–406. Springer.

[Richard Van Noorden 2016] Richard Van Noorden, D. C. (2016). World's tiniest machines win chemistry nobel. *Nature*, 538(7624):152–153.

[Roda-Sanchez et al. 2018] Roda-Sanchez, L., Garrido-Hidalgo, C., Hortelano, D., Olivares, T., and Ruiz, M. C. (2018). Operable: an iot-based wearable to improve efficiency and smart worker care services in industry 4.0. *Journal of Sensors*, 2018.

[Sakkaff et al. 2018] Sakkaff, Z., Immaneni, A., and Pierobon, M. (2018). Applying molecular communication theory to estimate information loss in cell signal transduction: An approach based on cancer transcriptomics. In *Proceedings of the 5th ACM International Conference on Nanoscale Computing and Communication*, pages 1–7.

[Santagati and Melodia 2014] Santagati, G. E. and Melodia, T. (2014). Opto-ultrasonic communications for wireless intra-body nanonetworks. *Nano Communication Networks*, 5(1):3 – 14.

[Seneviratne et al. 2017] Seneviratne, S., Hu, Y., Nguyen, T., Lan, G., Khalifa, S., Thilakarathna, K., Hassan, M., and Seneviratne, A. (2017). A survey of wearable devices and challenges. *IEEE Communications Surveys & Tutorials*, 19(4):2573–2620.

[Sharma et al. 2020] Sharma, M., Elmiligi, H., and Gebali, F. (2020). *Network Security and Privacy Evaluation Scheme for Cyber Physical Systems (CPS)*, pages 191–217. Springer International Publishing, Cham.

[Song et al. 2016] Song, H., Rawat, D. B., Jeschke, S., and Brecher, C. (2016). *Cyber-Physical Systems: Foundations, Principles and Applications*. Academic Press, Inc., USA, 1st edition.

[Sowe et al. 2016] Sowe, S. K., Simmon, E., Zettsu, K., de Vaulx, F., and Bojanova, I. (2016). Cyber-physical-human systems: Putting people in the loop. *IT professional*, 18(1):10–13.

[Stephenson-Brown et al. 2015] Stephenson-Brown, A., Acton, A. L., Preece, J. A., Fossey, J. S., and Mendes, P. M. (2015). Selective glycoprotein detection through covalent templating and allosteric click-imprinting. *Chemical science*, 6(9):5114–5119.

[Tahaei et al. 2020] Tahaei, H., Afifi, F., Asemi, A., Zaki, F., and Anuar, N. B. (2020). The rise of traffic classification in IoT networks: A survey. *Journal of Network and Computer Applications*, 154:102538.

[Tao et al. 2020] Tao, X., Jin, H., Mintken, M., Wolff, N., Wang, Y., Tao, R., Li, Y., Torun, H., Xie, J., Luo, J., et al. (2020). Three-dimensional tetrapodal zno microstructured network based flexible surface acoustic wave device for ultraviolet and respiration monitoring applications. *ACS Applied Nano Materials*, 3(2):1468–1478.

[Thorp 1998] Thorp, E. O. (1998). The invention of the first wearable computer. In *Digest of Papers. Second Int. Symposium on Wearable Computers*, pages 4–8. IEEE.

[Titova et al. 2013] Titova, L. V., Ayesheshim, A. K., Golubov, A., Fogen, D., Rodriguez-Juarez, R., Hegmann, F. A., and Kovalchuk, O. (2013). Intense thz pulses cause h2ax phosphorylation and activate dna damage response in human skin tissue. *Biomedical optics express*, 4(4):559–568.

[Trimananda et al. 2020] Trimananda, R., Varmarken, J., Markopoulou, A., and Demsky, B. (2020). Packet-level signatures for smart home devices. In *Network and Distributed System Security Symposium (NDSS)*.

[Wei et al. 2013] Wei, G., Bogdan, P., and Marculescu, R. (2013). Efficient modeling and simulation of bacteria-based nanonetworks with bnsim. *IEEE Journal on Selected Areas in Communications*, 31(12):868–878.

[Wirdatmadja et al. 2017] Wirdatmadja, S. A., Barros, M. T., Koucheryavy, Y., Jornet, J. M., and Balasubramaniam, S. (2017). Wireless optogenetic nanonetworks for brain stimulation: Device model and charging protocols. *IEEE Trans. on NanoBioscience*.

[Wolf and Serpanos 2017] Wolf, M. and Serpanos, D. (2017). Safety and security in cyber-physical systems and internet-of-things systems. *Proceedings of the IEEE*, 106(1):9–20.

[Wood et al. 2017] Wood, D., Apthorpe, N., and Feamster, N. (2017). Cleartext data transmissions in consumer iot medical devices. In *Proceedings of the 2017 Workshop on Internet of Things Security and Privacy*, pages 7–12. ACM.

[Yaacoub et al. 2020] Yaacoub, J.-P. A., Salman, O., Noura, H. N., Kaaniche, N., Chehab, A., and Malli, M. (2020). Cyber-physical systems security: Limitations, issues and future trends. *Microprocessors and Microsystems*, 77:103–201.

[Ye et al. 2007] Ye, H., Randall, C. L., Leong, T. G., Slanac, D. A., Call, E. K., and Gracias, D. H. (2007). Remote radio-frequency controlled nanoliter chemistry and chemical delivery on substrates. *Angewandte Chemie Int. Edition*, 46(26):4991–4994.

[Zheng et al. 2017] Zheng, G., Zhang, G., Yang, W., Valli, C., Shankaran, R., and Orgun, M. A. (2017). From wannacry to wannadie: Security trade-offs and design for implantable medical devices. In *2017 17th International Symposium on Communications and Information Technologies (ISCIT)*, pages 1–5. IEEE.

Capítulo 5

Aprendizado Federado aplicado à Internet das Coisas

Heitor S. Ramos (UFMG), Guilherme Maia (UFMG), Gisele L. Papa (UFMG), Mário S. Alvim (UFMG), Antonio A. F. Loureiro (UFMG), Isadora Cardoso-Pereira (UFMG), Diego H. C. Campos (UFMG), Giovanna Filipakis (UFMG), Giovanna Riquetti (UFMG), Eduarda T. C. Chagas (UFMG), Pedro H. Barros (UFMG), Gabriel N. Gomes (UFMG), Héctor Allende-Cid (PUC-Valparaíso)

Abstract

The main objective of this short course is to present the main fundamentals of Federated Learning (FL), covering the tools and steps necessary for the development of applications and services aimed at the Internet of Things (IoT). The concepts covered during this short course include introducing Machine Learning (centralized and distributed), the state-of-the-art in FL, an overview of existing work, challenges, and future perspectives for advancing the field. FL is a recent research field that still has a lot to be explored. Therefore, it has research fronts that remain open, with non-trivial challenges. Thus, this short course intends to deepen the following questions: (i) How does FL differ from centralized and decentralized Machine Learning? (ii) What are the main characteristics of FL that enable the development of applications in the context of IoT. (iii) What are the technological challenges for implementing FL? (iv) What are the main methodologies used for the development of IoT applications based on FL? (v) What are the main LF research problems? (vi) What are the representative applications of this area?

Resumo

O objetivo principal deste minicurso é apresentar os principais fundamentos do Aprendizado Federado (Federated Learning – FL), abrangendo as ferramentas e passos necessários para o desenvolvimento de aplicações e serviços voltados à Internet das Coisas (Internet of Things – IoT). Os conceitos abordados durante o presente minicurso incluem uma introdução à Aprendizado de Máquina (centralizada e distribuída), o estado-da-arte

em FL, uma visão geral dos trabalhos existentes, os desafios e as perspectivas futuras para o avanço da área. FL é um campo de pesquisa recente e que ainda possui muito a ser explorado. Assim sendo, possui frentes de pesquisa que seguem em aberto, com desafios não triviais. Dessa forma, este minicurso pretende aprofundar as seguintes questões: (i) Como FL se diferencia do Aprendizado de Máquina centralizado e descentralizado? (ii) Quais as principais características de FL que viabilizam o desenvolvimento de aplicações no contexto de IoT. (iii) Quais os desafios tecnológicos para a implantação de FL? (iv) Quais as principais metodologias utilizadas para o desenvolvimento de aplicações de IoT baseadas em FL? (v) Quais são os principais problemas de pesquisa de FL? (vi) Quais são as aplicações representativas desta área?

5.1. Introdução a Aprendizado Federado no Contexto de IoT

Dispositivos móveis como *smartphones*, dispositivos vestíveis e veículos autônomos são exexemplos de sistemas distribuídos que geram uma grande quantidade de dados diariamente. Esses dispositivos têm habilitado técnicas de inteligência computacional, mais especificamente aprendizado de máquina (ML), a produzirem aplicações cada vez mais atrativas e poderosas como visão computacional, processamento de linguagem natural, sistemas de recomendação, reconhecimento de fala, para citar algumas. O sucesso de várias aplicações de ML, especialmente o aprendizado profundo (DL), tem sido alavancado pela disponibilização de grandes quantidades de dados para viabilizar o treinamento e a avaliação adequada desses modelos.

Utilizando-se desses dados, as técnicas avançadas de ML viabilizam aplicações que muitas vezes superam a capacidade humana, por exemplo, os sistemas de detecção de face modernos conseguem reconhecer uma quantidade grande de indivíduos que talvez nenhum ser humano sozinho consiga fazê-lo. Por exemplo, há relatos de que o sistema de detecção de objetos do Facebook tem sido treinado com pelo menos 3,5 bilhões de imagens oriundas do Instagram [Yang et al. 2019a].

Apesar de estarmos vivendo na era do chamado *big data*, vários autores [Yang et al. 2019a] têm relatado que na prática, na maioria das vezes, temos pequenos volumes de dados distribuídos em uma grande quantidade de entidades, sejam corporações ou indivíduos. Muitas vezes, esses dados não estão adequadamente preparados, por exemplo, não estão devidamente rotulados, ou apresentam muitos dados faltatantes, devido ao custo associado a ter-se esse tipo de qualidade de dados. Por exemplo, é comum termos na área de saúde, diversas bases de dados sem a devida preparação por conta dessa tarefa ser associada a um alto custo de ter um especialista analisando e anotando esses dados. Como resultado disso, frequentemente encontramos o que é conhecido por silos de dados (*data silos*) que não são facilmente conectados para produção de modelos de ML adequados.

Deve-se levar em conta também que o modo como os dados são utilizados no treinamento de ML pode infringir algumas leis locais voltadas para a privacidade dos usuários, como por exemplo: a *General Data Protection Regulation* (GDPR), da União Européia, que entrou em vigor em 2018 e tem como algumas das normas a transparência, a não coleta de dados desnecessários para o propósito, proteção contra ataques aos dados, e, caso seja requisitado, ter dados imediatamente apagados do sistema; e a le-

gislação dos Estados Unidos, em especial a da Califórnia (*California Consumer Privacy Act*, ou CCPA), que entre suas regras permite aos usuários barrarem que seus dados sejam vendidos por companhias [Yang et al. 2019a]. No Brasil, temos também a Lei Geral de Proteção de Dados (LGPD), promulgada em 14 de agosto de 2018 e que tem como princípio o respeito à privacidade, a invioabilidade da intimidade, da honra e da imagem, dentre outras coisas. Dessa maneira, o uso dos dados deve sempre ter a preocupação de não expor a privacidade e a intimidade dos indivíduos que estão sujeitos a terem seus dados expostos. Iremos abordar com mais detalhes a questão de privacidade dos dados em ambientes de FL na Seção 5.3.

Os métodos de ML que estão sendo disseminados rapidamente, terão que se adequar a um ambiente em que os dados existem, porém, estão espalhados em um grande sistema distribuído. Além disso, transferir os dados desse sistema distribuído para uma entidade central que irá treinar os modelos de ML, pode não ser uma opção viável, seja por questões relativas à privacidade e regulações ou até mesmo do alto custo da transferência de grandes quantidades de dados em redes que podem não se adequar a esse cenário.

A Internet das Coisas (IoT) é uma das tecnologias viabilizadoras da geração de grandes quantidades de dados. Tanto a IoT mais voltada para uso individual, como vigilância patrimonial, aplicações de conforto, saúde (*e-health*), dentre outras, como as aplicações de IoT industriais/corporativas como monitoramento de chão de fábrica, gestão patrimonial e monitoramento de equipamentos, têm um enorme potencial para viabilizar aplicações avançadas de ML. Entretanto, é uma tecnologia que tem potencial para expor dados privados de usuários e também de não permitir a comunicação desse grande volume de dados em dispositivos que podem esgotar bateria ou serem equipados por redes sem fio que, por usarem canal compartilhado, não são muito adequadas para essa grande transferências de dados. Por outro lado, não está claro que as corporações que utlizam IoT industrial/corporativa irão ter disponibilidade para compartilhar dados, afinal, a possibilidade de perder o controle sobre dados estratégicos da empresa ou dados privados dos clientes é um dificultador para o modelo mais usual em que os dados são compartilhados.

O Aprendizado Federado (*Federated Learning – FL*) surge como uma solução natural para o problema de treinar modelos de ML em ambientes que apresentam alta fragmentação de dados sem violar as diretrizes das regulações vigentes nos países que estão se voltando para a proteção da privacidade do indivídulo. FL é um modelo de ML distribuída onde a privacidade do dado é uma premissa essencial. Dessa maneira, estamos interessados em construir modelos de ML (sejam preditivos ou descritivos) em que o compartilhamento de dados não seja uma opção. A ideia geral, é treinar um modelo em cada usuário, utilizando os dados locais dos usuários e comunicar os modelos (não os dados) de forma que as entidades federadas possam computar um modelo global que tenha expressividade sobre todos os dados dos clientes da federação, mesmo sem ter tido contato direto com eles. Os modelos devem ser comunicados em canais seguros, de maneira que o acesso aos parâmetros do modelo não seja suficiente para que se quebre a privacidade dos usuários.

A implementação de modelos de ML em ambientes com alta capacidade de processamento e memória para treinamento dos modelos, por exemplo, em máquinas com alto paralelismo dotadas de GPU(graphical processor units) tem apresentado bons resul-

tados. Entretanto, a maioria das entidades encontradas em IoT são equipamentos com capacidade limitada, o que torna a utilização desses modelos de DL impraticável. Por exemplo, a AlexNet [Krizhevsky et al. 2012], uma das arquiteturas mais famosas em visão computacional, possui 61 milhões de parâmetros, ocupando 249 MB de memória. Sua utilização em dispositivos como celulares rapidamente sobrecarregará os recursos escassos [Rastegari et al. 2016]. Esses recursos limitados também inviabilizam a transferência de parâmetros entre os dispositivos e o servidor, impossibilitando a adoção de FL. Dessa forma, é necessário encontrar alternativas que possam acelerar a computação dos modelos de DL para superar as dificuldades apresentadas e assim possibilitar a utilização de DL no contexto de FL em IoT. Esses aspectos serão discutidos na Seção 5.4.

A crescente utilização de aplicações de IoT tem habilitado um modelo conhecido por Computação de Borda (*edge computing*). Nesse cenário, o grande volume de dados não se encontra em um ambiente centralizado como a nuvem, mas espalhado entre diversos dispositivos. Por exemplo, veículos autônomos deverão processar modelos de ML para tomoda de decisão sem que seja necessário acionar a nuvem, pois esta pode impor uma latência de comunicação indesejável ou até mesmo proibitiva para a aplicação. Dessa maneira, os veículos devem processar essa informação localmente, ou com a ajuda de infraestrutura que esteja mais próxima a eles, viabilizando os requisitos da aplicação. Iremos discutir aspectos de Computação na Borda e FL na Seção 5.6.

A primeira aplicação de FL surgiu em 2016 no contexto de processamento natural de linguagem natural e foi proposto pelo Google no artigo [McMahan et al. 2016a]. Neste artigo, os autores propuseram um modelo federado para predição de digitação de texto em *smartphones* Android. O sistema ajuda os usuários a autocompletarem os textos digitados. Os autores propuseram o modelo de agregagação FedAvg (ver detalhes na Seção 5.2.3). Nesse modelo, os dados do usuário não são enviados para uma entidade central. Um modelo é treinado localmente e os parâmetros do modelo são enviados, de maneira criptografada, para a nuvem. Esta, computa um modelo global a partir dos modelos de cada usuário e retroalimenta esses parâmetros para os clientes, que atualizam seus modelos e produzem modelos com maior capacidade. Essa foi a primeira aplicação bem sucedida de FL reportada na literatura.

O sistema de FL do teclado do Google (Gboard) é um bom exemplo de B2C (business-to-consumer), entretanto, FL não está limitado apenas a esse modelo. Aplicações de B2B (business-to-business) também foram concebidas de modo que mesmo em ambientes em que os mesmos usuários possuam dados diferentes armazenados em diversas corporações podem se beneficiar do FL. Iremos abordar esses modelos na Seção 5.2.

Este minicurso tem a seguinte estrutura. Na Seção 5.2 são apresentados os modelos mais comuns de FL, como o FL horizontal, vertical e o FL *trasnfer learning*. A Seção 5.3 apresenta os aspectos relativos à preservação de privacidade em FL. A Seção 5.4 apresenta um aspecto de grande importância para IoT, que é a compressão de modelos para que dispositivos de baixa capacidade de processamento e memória, comumente encontrados em cenários de IoT estejam habilitados a participar de uma federação. A Seção 5.5 introduz aspectos de aprendizado por reforço no contexto de FL. A Seção 5.6 discute a relação entre Computação de Borda e FL. As Seções 5.7, 5.8 e 5.9 apresentam oportunidades de pesquisa, aplicações e um estudo de caso de FL, respectivamente. Fi-

Living area (feet^2)	#bedrooms	Price (1000$s)
2104	3	400
1600	3	330
2400	3	369
1416	2	232
3000	4	540
⋮	⋮	⋮

Figura 5.1. Exemplo de um conjunto de dados de treinamento

nalmente, a Seção 5.10 apresenta nossas considerações finais.

5.1.1. Introdução a Aprendizado de Máquina

Aprendizado de Máquina (ML) pode ser definido como um conjunto de métodos capazes de detectar automaticamente padrões nos dados e, em seguida, utilizar os padrões descobertos para realizar previsões a partir da observação de novos dados ou para realizar outros tipos de tomada de decisão.

ML geralmente é dividido em três tipos principais, (i) aprendizado supervisionado, (ii) aprendizado não-supervisionado e (iii) aprendizado por reforço. No aprendizado supervisionado, o objetivo é aprender um mapeamento das entradas $x^{(i)}$, também chamadas de *features* ou atributos, para as saídas $y^{(i)}$, dado um conjunto de dados rotulados de entrada e saída $\mathscr{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{n}$. O par $(x^{(i)}, y^{(i)})$ é tido como um exemplo de dado de treinamento. Já o conjunto $\mathscr{D}$ é chamado de conjunto de dados de treinamento, onde n é a quantidade de exemplos de treinamento. Na configuração mais simples, cada entrada de dado de treinamento $x^{(i)}$ é um vetor D-dimensional de números, representando, por exemplo, a altura e o peso de uma pessoa, ou a área e a quantidade de quartos em uma residência. No entanto, $x^{(i)}$ pode ser um objeto estruturado complexo, como uma imagem, um texto, uma série temporal, ou a estrutura de uma molécula. De maneira similar, a variável de saída $y^{(i)}$ pode, em princípio, ser qualquer coisa. No entanto, na maioria dos casos assume-se que é uma variável categórica de algum conjunto finito, ou um valor do conjunto dos números reais. Quando $y^{(i)}$ é uma variável categórica, o problema de ML é conhecido como classificação, e quando possui um valor real, o problema é conhecido como regressão.

No aprendizado supervisionado, o objetivo é, dado um conjunto de dados de treinamento, aprender uma função $f : \mathscr{X} \mapsto \mathscr{Y}$ de forma que $f(x)$ é um "bom"preditor para o valor correspondente y. Aqui, $\mathscr{X}$ denota o espaço dos valores de entrada, enquanto que $\mathscr{Y}$ denota o espaço dos valores de saída. Como um exemplo de aprendizado supervisionado, considere o conjunto de dados de treinamento mostrado na Figura 5.1 que contém algumas características a respeito de imóveis que estão à venda[1]. As entradas x são vetores de duas dimensões em $\mathbb{R}^2$, onde $x_1^{(i)}$ representa a área do imóvel da i-ésima residência no conjunto de treinamento, enquanto que $x_2^{(i)}$ representa o número de quartos.

Para realizar aprendizado supervisionado, é necessário antes determinar como a função f será representada. Uma estratégia é aproximar y como uma função linear de

[1] `http://cs229.stanford.edu/notes2020spring/cs229-notes1.pdf`

x denotada por $f_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$, onde os θ_i's são os parâmetros (ou pesos) que estão parametrizando o espaço de mapeamento de funções lineares de $\mathscr{X}$ para $\mathscr{Y}$.

Dado um conjunto de dados de treinamento, como selecionar (ou aprender) os valores para os parâmetros θ? Um método razoável parece ser fazer $f(x)$ ser próximo de y, pelo menos para os exemplos no conjunto de dados de treinamento. Tal processo é conhecido como fase de treinamento de um modelo de ML. De maneira formal, pode-se definir uma função que mede, para cada um dos valores de θ, quão próximo são os valores de $f(x^{(i)})$ com os valores correspondentes de $y^{(i)}$, isso para cada exemplo i no conjunto de dados de treinamento. Logo, define-se o que é chamada de função de custo $J(\theta) = \frac{1}{2}\sum_{i=1}^{n}(f_\theta(x^{(i)}) - y^{(i)})^2$, onde neste exemplo utiliza-se a função de custo conhecida como mínimos-quadrados (*least-squares*).

O próximo passo é escolher valores para θ que minimizam a função $J(\theta)$. Uma estratégia é utilizar o algoritmo conhecido como descida do gradiente (*gradient descent*), o qual escolhe algum valor inicial para θ, e repetidamente executa $\theta_j = \theta_j - \alpha\frac{\partial}{\partial\theta_j}J(\theta)$ simultaneamente para cada um dos valores de $j = 0,\ldots,m$. Aqui, α é chamado de *learning rate*, o qual representa um dos hiper-parâmetros do modelo e, portanto, deve ser definido antes do início da etapa de treinamento. Já $\frac{\partial}{\partial\theta_j}$ é o gradiente e indica a inclinação de J em um dado ponto θ. Perceba que este é um algoritmo muito natural que repetidamente dá um passo na direção que vai diminuir de forma mais acentuada o valor de J.

No aprendizado não-supervisionado apenas entradas são fornecidas, $\mathscr{D} = \{x^{(i)}\}_{i=1}^{n}$, e o objetivo é encontrar padrões interessantes nos dados. Isso torna o problema menos bem definido, já que não somos informados sobre quais tipos de padrões procurar e não há uma métrica de erro óbvia para utilizar. Isso é diferente do aprendizado supervisionado, onde pode-se comparar a previsão para uma saída y dado a entrada x com o valor observado. Já no aprendizado por reforço, a ideia é aprender como agir em um determinado ambiente quando são dados sinais ocasionais de recompensa ou punição.

5.1.2. Introdução a Aprendizado de Máquina Distribuído

No mundo da IoT, grandes volumes de dados são produzidos de forma onipresente. Diante disso, o principal obstáculo dos métodos de aprendizado de máquina (ML) tradicionais mudou de ser capaz de realizar inferência a partir de pequenas quantidades de dados de treinamento para como lidar com conjuntos de dados de treinamento com grande escala e altas dimensões. Nesse cenário, o poder computacional e o tempo não escalam bem com o volume do conjunto de dados. Isso torna o processo de aprendizado a partir de amostras de treinamento de grande escala com esforço e tempo de computação razoáveis uma tarefa muito difícil. Além disso, é comum lidar com cenários em que os dados estão espalhados em diferentes localidades (por exemplo, diferentes dispositivos na borda da rede - ver Seção 5.6), pertencem a diferentes indivíduos ou organizações, e não há uma solução simples para reunir os dados para realizar o processo de treinamento. A seguir, são apresentados os principais desafios que os métodos de ML tradicionais enfrentam ao lidar com conjuntos de dados de grande escala e que estão distribuídos em diferentes localidades: (i) **Falta de memória:** Os métodos de ML tradicionais são treinados carregando-se as amostras de treinamento inteiramente na memória principal. Portanto, se a complexidade

computacional das amostras de treinamento exceder a memória principal, os seguintes problemas podem surgir: (i) o modelo treinado pode não convergir ou pode resultar em baixo desempenho (como baixa precisão ou *recall*), e (ii) no pior cenário, os modelos de ML podem não ser treinados devido à falta de memória. (ii) **Tempo de treinamento inviável:** Alguns processos de otimização utilizados por algoritmos de ML podem não escalar bem em relação ao tamanho das amostras de treinamento. Como consequência, ao lidar com amostras de treinamento de grande escala, o tempo consumido pelo processo de treinamento pode ser muito longo para fins práticos. Além disso, o ajuste de hiper-parâmetros dos modelos de ML também leva muito tempo, pois normalmente é necessário avaliar muitas configurações diferentes. Portanto, se o processo de treinamento demorar muito, o ajuste de hiper-parâmetros pode não ser realizado de forma eficaz, o que pode resultar em modelos de ML ineficientes. (iii) **Violação de privacidade:** Usuários estão cada vez mais preocupados que suas informações pessoais estão sendo utilizadas para os mais diversos fins sem permissão. Diante desse cenário, coletar e compartilhar dados para treinar modelos de ML tem se tornado uma tarefa cada vez mais difícil.

Aprendizado de Máquina Distribuído (DML) [Yang et al. 2019a], ou Aprendizado Distribuído, refere-se ao estudo de algoritmos e sistemas de aprendizado de máquina que são projetados para alcançar pelo menos um dos objetivos a seguir: (i) escalar para mais dados de treinamento e modelos de inferência maiores; (ii) melhorar o desempenho da solução como um todo; (iii) preservar a privacidade das entidades detentoras dos dados.

De maneira geral, pode-se dizer que as soluções de DML possuem duas principais motivações: melhorar a escalabilidade e preservar a privacidade. No DML motivado pela escalabilidade, a principal preocupação é em como projetar soluções que são capazes de lidar com os requisitos computacionais cada vez maiores dos sistemas de ML de grande escala. Por exemplo, é fato notório que a escala dos problemas que os métodos de ML têm que lidar aumentaram de maneira exponencial nos últimos anos. Treinar modelos de aprendizado sofisticados e que possuem uma grande quantidade de dados de treinamento pode facilmente exceder a capacidade do modelo tradicional de ML que depende de uma única entidade computacional. Não é incomum encontrar exemplos de modelos na literatura que requerem várias unidades de processamento e que podem levar vários dias apenas para realizar a etapa de pré-treinamento. A seguir, são apresentadas as principais estratégias de DML motivadas por escalabilidade: (i) **Paralelismo de dados:** Uma estratégia natural em DML é particionar os dados de treinamento em subconjuntos, os quais são distribuídos para diferentes entidades de computação. Essas entidades mantêm réplicas do mesmo modelo e utilizam os subconjuntos de dados para treiná-lo em paralelo. Uma desavantagem dessa estratégia é que as réplicas do modelo devem residir na memória das entidades de computação, logo ela não escala bem para modelos muito grandes. Há duas abordagens para DML baseadas em paralelismo de dados: (i) treinamento síncrono e (ii) treinamento assíncrono. No treinamento síncrono, as entidades de computação treinam réplicas do modelo utilizando diferentes partes dos dados de treinamento de maneira síncrona, e os gradientes e pesos são agregados após cada etapa de treinamento (cada *epoch*). Já no treinamento assíncrono, as entidades computacionais treinam de maneira independente réplicas do mesmo modelo utilizando subconjuntos dos dados de treinamento e atualizam os gradientes e pesos de maneira assíncrona. (ii) **Paralelismo do modelo:** Conforme destacado anteriormente, modelos de treinamento muito

grandes podem não caber inteiramente na memória de uma única entidade computacional. Nessa situação, uma estratégia é dividir o modelo e então distribuir partes do modelo para diferentes entidades computacionais. Por exemplo, ao considerar um modelo de redes neurais profundas (DNN), pode-se dividir o modelo de maneira lógica, colocando-se diferentes camadas do modelo em diferentes entidades de computação. Dessa forma, as etapas de *forward propagation* e *backpropagation* envolvem a comunicação da saída de um dispositivo para a entrada em um outro dispositivo computacional de maneira serial. (iii) **Paralelismo de tarefas:** Normalmente utilizada em conjunto com a estratégia de paralelismo de dados, o paralelismo de tarefas diz respeito à execução de um programa em vários processadores localizados em um único computador ou em vários computadores. Ou seja, o objetivo é executar diferentes operações em paralelo de forma a utilizar totalmente os recursos de computação disponíveis na forma de processadores e memória. Considere, por exemplo, um programa de computador que utiliza várias *threads*, onde cada uma é responsável por realizar uma operação diferente.

É importante observar que na prática, normalmente se faz necessário combinar mais de uma forma de paralelismo, o que é conhecido como paralelismo híbrido. Além disso, tais estratégias podem utilizar recursos de computação elásticos e escaláveis, o que possibilita adicionar mais entidades de computação sob demanda. Tal característica é particularmente útil na era da computação em nuvem e *edge computing*, onde pode-se solicitar mais processadores (como CPUs e GPUs) e memória sob demanda. Finalmente, tais estratégias são comumente aplicadas nos cenários com conjuntos de dados particionados horizontalmente, onde subconjuntos separados de dados de treinamento são armazenados em diferentes entidades computacionais (ver Seção 5.2.2).

Já no DML motivado pela privacidade, a principal preocupação é preservar a privacidade do usuário. ML com preservação de privacidade tem se tornado uma tendência na comunidade de ML, já que questões como a privacidade dos usuários e a segurança dos dados têm recebido atenção especial de maneira global. Em um sistema DML motivado pela privacidade, existem várias entidades e cada uma contém um subconjunto dos dados de treinamento. Devido a questões de privacidade, as entidades não desejam expor seus dados umas às outras. Desse modo, as abordagens de DML são obrigadas a utilizar os dados de cada entidade participante para treinar um modelo de ML de maneira colaborativa. Nesse cenário, os conjuntos de dados mantidos por diferentes entidades podem ter atributos diferentes, resultando na chamada partição vertical dos dados de treinamento. Diante disso, DML motivado por privacidade é frequentemente utilizado em cenários com conjuntos de dados particionados verticalmente, com subconjuntos de dados de treinamento com diferentes atributos mantidos por diferentes entidades (ver Seção 5.2.4). Para mais detalhes sobre as principais estratégias de preservação de privacidade em um ambiente distribuído, consultar a Seção 5.3.

5.2. Aprendizado Federado e Modelos de Aprendizado Federado

Uma abordagem distribuída de ML chamada Aprendizado Federado (FL) [McMahan et al. 2016b] foi proposta para garantir que os dados de treinamento permaneçam em dispositivos pessoais e facilite modelos complexos de aprendizado de máquina colaborativo em dispositivos distribuídos.

Em FL, os dispositivos pessoais usam dados armazenados localmente para treinar um modelo local. A junção de vários modelos locais é usada para treinar um modelo de ML cooperativamente. Este modelo é disponibilizado por um servidor FL. De modo geral, o servidor recebe a atualização dos modelos locais (por exemplo, os pesos ou gradientes do modelo), e realiza uma agregação do valores recebidos. As etapas são repetidas em várias rodadas até que uma precisão desejável seja alcançada. Esse fato implica que FL pode ser uma tecnologia capacitadora para o treinamento de modelos de ML localizados em dispositivos conectados na borda da rede.

Em comparação com as abordagens convencionais de treinamento centrado na nuvem, a implementação de FL para treinamento de modelo em dispositivos na borda da rede apresenta as seguintes vantagens [Lim et al. 2020]: uso altamente eficiente da largura de banda da rede, privacidade e baixa latência.

5.2.1. Aprendizado Federado

Sejam N usuários $\{u_1, \cdots, u_N\}$, todos os quais desejam treinar um modelo de aprendizado de máquina por seus respectivos conjuntos de dados disjuntos $\{\mathbb{X}_1, \cdots, \mathbb{X}_N\}$. O método usual de treinamento de aprendizado de máquina é agrupar todos os dados no conjunto $\mathbb{X} = \mathbb{X}_1 \cup ... \cup \mathbb{X}_N$ para treinar um modelo $T_{\mathbb{X}}$. Um sistema de aprendizado federado é um processo de aprendizagem no qual os proprietários dos dados treinam colaborativamente um modelo T_{Fed}, no qual qualquer proprietário dos dados u_i não expõe seus dados $\mathbb{X}_i$ a outros [Yang et al. 2019a]. Em geral, o processo de treinamento do Aprendizado Federado inclui as seguinte três etapas[2] [Lim et al. 2020]: **Etapa 1** (Inicialização do modelo): O sistema inicializa os modelos locais $T_{\mathbb{X}_i}$, bem como define todos os valores de hiper-parâmetros necessários para iniciar a tarefa de aprendizagem. **Etapa 2** (treinamento e atualização do modelo local): Cada usuário u_i, respectivamente, usa seus dados locais $\mathbb{X}_i$ e atualiza o modelo local $T_{\mathbb{X}_i}$. O objetivo do usuário u_i é encontrar os parâmetros ideais para o modelo local que minimizam a função de perda $L(T_{\mathbb{X}_i})$, ou seja, $\Theta^*_{\mathbb{X}_i} := \arg\min_{\Theta_{\mathbb{X}_i}} L(T_{\mathbb{X}_i})$, onde $\Theta_{\mathbb{X}_i}$ são os parâmetros do modelo $T_{\mathbb{X}_i}$. **Etapa 3** (agregação e atualização do modelo): O servidor agrega os modelos locais dos participantes e envia as atualizações para o modelo agregado T_{Fed}. Depois da agregação, o modelo agregado envia o novo modelo de volta para os proprietários dos dados. O servidor minimiza a função de perda agregada L (T_{Fed}), ou seja, $L(T_{Fed}) = \sum_{i=1}^{N} [L(T_{\mathbb{X}_i}) p_i]$, onde p_i é um valor definido para o usuário u_i e $\sum_{i=1}^{N} p_i = 1$. Tipicamente, $p_i = 1/N$ ou $p_i = n_i / \sum_{i=1}^{N} n_i$, com $n_i = |\mathbb{X}_i|$ como podemos ver em [Li et al. 2020c]. No processo de treinamento, as etapas 2-3 são repetidas até que a função de perda agregada convirja ou uma precisão de treinamento desejável seja alcançada.

5.2.2. Aprendizado Federado Horizontal

Aprendizado Federado Horizontal (HFL), também conhecido como aprendizado federado particionado por amostra, pode ser aplicado em cenários nos quais conjuntos de dados em locais diferentes compartilham características sobrepostas, mas diferente no espaço de amostra, conforme ilustrado na Figura 5.2. É semelhante à situação em que os dados são particionados horizontalmente em uma visualização de uma tabela. Por exemplo,

[2] o modelo local refere-se ao modelo treinado em cada usuário u_i, enquanto o modelo global se refere ao modelo agregado T_{Fed}

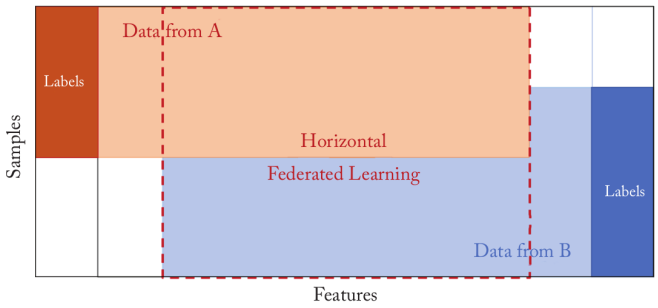


Figura 5.2. Ilustração do Aprendizado Federado Horizontal. Fonte: [Yang et al. 2019a]

dois bancos regionais podem ter grupos de usuários muito diferentes de suas respectivas regiões, e o conjunto de interseção de seus usuários é muito pequeno. No entanto, seus modelos de negócios são muito semelhantes. Assim, descreveremos duas arquiteturas populares para sistemas HFL, a saber, a arquitetura cliente-servidor e a arquitetura ponto a ponto.

Na arquitetura cliente-servidor, K participantes (também conhecidos como clientes ou usuários) com a mesma estrutura de dados treinam de forma colaborativa um modelo de aprendizado de máquina com a ajuda de um servidor (também conhecido como servidor de agregação ou coordenador). Uma suposição típica é que os participantes são honestos, enquanto o servidor é honesto, mas curioso. Portanto, o objetivo é evitar o vazamento de informações de quaisquer participantes para o servidor.

As iterações de treinamento continuam até que a função de perda convirja ou até que o número máximo de iterações permitidas é atingido. Esta arquitetura é independente de algoritmos de ML específicos (por exemplo, regressão logística ou redes neurais), e todos os participantes compartilharão os mesmos parâmetros finais do modelo. Para HFL, um proprietário de dados tem total autonomia para operar em seus dados locais, podendo decidir quando e como ingressar e contribuir para um sistema HFL. Além disso, o HFL leva em consideração a proteção da privacidade dos dados durante o treinamento do modelo. Na prática, em um sistema HFL, os dados obtidos pelos diferentes participantes não são distribuídos de forma idêntica na maioria das aplicações.

Além da arquitetura cliente-servidor discutida acima, um sistema HFL também pode fazer uso da arquitetura ponto-a-ponto (P2P). Na arquitetura P2P, não há servidor central ou coordenador. Em tais cenários, os participantes de um sistema HFL também são chamados de treinadores ou trabalhadores distribuídos. Cada treinador é responsável por treinar o mesmo modelo ML ou DL (por exemplo, um modelo DNN) usando apenas seu dado local. Além disso, os treinadores precisam de canais seguros para transferir os pesos do modelo um para o outro. Para garantir comunicações seguras entre quaisquer dois treinadores, podem ser adotadas medidas de segurança, como esquemas de criptografia com base em chave pública.

Uma vez que não há um servidor central, os treinadores devem concordar com a ordem de envio e recebimento dos pesos do modelo com antecedência. Existem duas maneiras simples de fazer isso: (i) **Transferência cíclica**: No modo de transferência cíclica, os treinadores são organizados em uma cadeia. O primeiro treinador (ou seja, o topo

da cadeia) envia os pesos do modelo atual para seu vizinho subsequente. Um treinador recebe pesos de modelo e atualiza o modelo recebido usando mini-lotes de dados de treinamento de seu próprio conjunto de dados. Então, depois do treinamento, envia os pesos do modelo atualizado para seu treinador subsequente. Por exemplo, do treinador 1 para o treinador 2, do treinador 2 para o treinador 3; e assim sucessivamente até do treinador (K - 1) para o treinador K, e do treinador K de volta ao treinador 1. Este procedimento é repetido até que os pesos do modelo convirjam ou até que o número máximo de iterações seja alcançado. (ii) **Transferência aleatória**: O k-ésimo treinador seleciona um outro treinador i aleatoriamente e envia os pesos do modelo para o treinador i. Quando o treinador i recebe os pesos do modelo do k-ésimo treinador, ele atualiza os pesos do modelo recebido usando mini-lotes de dados de treinamento de seu próprio conjunto de dados. Este procedimento ocorre simultaneamente entre os k treinadores até eles concordarem que os pesos do modelo convergiram ou até que o tempo máximo de treinamento permitido é atingido. Este método também é conhecido como aprendizagem por sussurro.

Comparado com a arquitetura cliente-servidor, a vantagem óbvia da arquitetura P2P é a possibilidade de remover o servidor central, que pode não estar disponível em aplicações práticas, e elimina a chance de vazar informações para o servidor. No entanto, existem várias desvantagens. Por exemplo, no modo de transferência cíclica, uma vez que não existe um servidor central, os parâmetros de peso são atualizado em série, em vez de em lotes paralelos, o que leva mais tempo durante o treinamento do modelo.

5.2.3. Algoritmos de Média Federada

Em [McMahan et al. 2016a], o algoritmo de média federada (FedAvg) foi empregado para treinamento de modelo federado em sistemas HFL. Esta seção revisa o algoritmo FedAvg, assumindo uma arquitetura cliente-servidor.

A quantidade de computação é controlada por três parâmetros principais, a saber: (1) ρ, a fração de clientes que realizam cálculos durante cada rodada; (2) S, o número de etapas de treinamento que cada cliente executa em seu conjunto de dados local durante cada rodada (ou seja, o número de épocas locais); e (3) M, o tamanho do mini-lote usado para o cliente realizar atualizações.

Este algoritmo seleciona uma fração de participantes durante cada rodada e calcula o gradiente e a função de perda sobre todos os dados mantidos por esses participantes. Portanto, este algoritmo controla o tamanho do lote global, com $\rho = 1$ correspondendo ao lote completo do gradiente descendente usando todos os dados mantidos por todos os participantes. Uma vez que são selecionados lotes usando todos os dados dos participantes escolhidos, este algoritmo é referido como FederatedSGD.

É comumente assumido que o coordenador ou servidor tem o modelo de ML inicial, e os participantes conhecem as configurações do otimizador. Para uma implementação típica de gradiente descendente com uma taxa de aprendizagem fixa η, na t-ésima rodada de atualização de peso do modelo global, o k-ésimo participante calcula o gradiente médio $g_i = \nabla L(T_{\mathbb{X}_i})$ do peso do modelo atual Θ_i, e o servidor agrega esses gradientes e aplica a atualização dos pesos do modelo de acordo com $\Theta_{Fed} \leftarrow \Theta_{Fed} - \eta \sum_{k=1}^{K} \frac{n_k}{n} g_k$, onde $n_k = |\mathbb{X}_k|$. Além disso, note que o servidor pode enviar os pesos do modelo atualizado Θ_{Fed} de volta para os clientes. Este método é chamado Gradiente médio.

Alternativamente, o servidor pode enviar o gradiente médio $\sum_{k=1}^{K} \frac{n_k}{n} g_k$ de volta para os usuários, e os usuários calculam a atualização dos pesos. Logo, pode-se descrever

$$\Theta_{\mathbb{X}_i} \leftarrow \Theta_{Fed} - \eta g_k, \tag{1}$$

$$\Theta_{Fed} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} \Theta_{\mathbb{X}_i}. \tag{2}$$

Ou seja, cada cliente localmente realiza uma etapa (ou várias etapas) da descida do gradiente com os pesos obtidos pelo servidor Θ_{Fed} (Equação 1), e envia os pesos locais $\Theta_{\mathbb{X}_i}$. O servidor recebe os pesos locais dos usuários e atualiza seu peso – (Equação 2).

5.2.4. Aprendizado Federado Vertical

Conforme visto na seção 5.2.2, o aprendizado federado horizontal (HFL) é aplicável a cenários IoT onde os conjuntos de dados dos participantes compartilham o mesmo espaço de recursos, mas diferem em espaços de amostra. Desse modo, o HFL é conveniente para ser aplicado na construção de aplicativos movidos por uma grande quantidade de dispositivos móveis, i.e. usuários diferentes mas com as mesmas características. Nesses casos, os usuários sendo federados são os consumidores individuais dos aplicativos.

No entanto, em muitos cenários práticos, os participantes da aprendizado federado são usuários de organizações que coletam diferentes características de dados para o mesmo grupo de usuários. Essas organizações tem como objetivo a cooperação entre si a fim de melhorar a eficiência de seus serviços. Por exemplo, suponha que haja um usuário com alguns registros em um banco que refletem ao saldo bancário, comportamento das despesas e classificação de crédito. Além disso, o mesmo usuário possui algumas outras informações armazenadas em um site de comércio eletrônico que retém o histórico de compra e navegação online do usuário.

Embora as características nas duas organizações sejam bastante diferentes, elas têm um aspecto próximo de associação uns com os outros. Por exemplo, o histórico de compras do usuário pode determinar de alguma forma a classificação de crédito do usuário. Esses cenários são comuns na vida real. Os varejistas podem fazer parceria com bancos para oferecer serviços ou produtos personalizados com base no histórico de compras do mesmo usuário. Os hospitais podem colaborar com empresas farmacêuticas para fazer uso de prontuários de pacientes comuns para tratar doenças crônicas e reduzir riscos de uma futura hospitalização.

Pode-se categorizar o aprendizado vertical (VFL) federado através participantes cujos conjuntos de dados compartilham o mesmo espaço amostral, mas diferentes características. A palavra “vertical” vem do termo "partição vertical", que é amplamente usado no contexto de visualização de um banco de dados (por exemplo, colunas de uma tabela são verticalmente particionadas em grupos diferentes e cada coluna representa uma característica de todas as amostras), como mostrado na Figura 5.3.

Os conjuntos de dados mantidos por diferentes organizações com diferentes objetivos de negócios geralmente têm informações diferentes sobre os usuários, embora essas organizações possam compartilhar usuários em comum. Com VFL, também chamado de aprendizado federado particionado por recurso, pode-se aproveitar os espaços de recursos

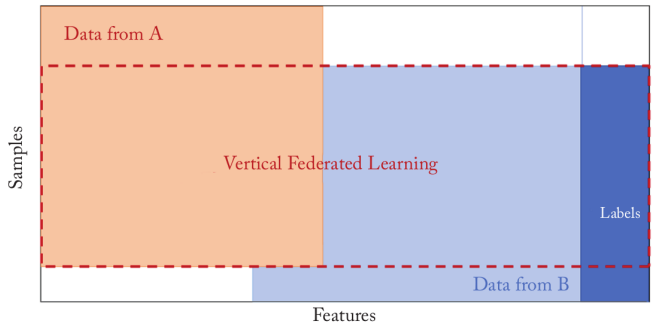


Figura 5.3. Ilustração do Aprendizado Federado Vertical. Fonte: [Yang et al. 2019b]

heterogêneos de conjuntos de dados distribuídos mantidos por aquelas organizações para construir modelos melhores de ML sem trocar e expor os dados privados.

Nas configurações de VFL, existem várias suposições subjacentes para obter segurança e preservar privacidade. Primeiro, presume-se que os participantes são honestos, mas curiosos. Isso significa que os participantes tentem deduzir o máximo possível das informações recebidas de outros participantes, embora cumpram o protocolo sem prejudicá-lo de forma alguma. Segundo, presume-se que o processo de transmissão de informações é seguro e confiável o suficiente para defender contra ataques. Além disso, assume-se que a comunicação é sem perdas, sem interferir com os resultados intermediários.

Um terceiro sistema semi-honesto (STP) também pode se juntar aos participantes para ajudar ambos. O STP é independente de ambas as partes. O STP coleta o resultados intermediários para calcular os gradientes e distribuir os resultados para cada parte. As informações que o STP recebe dos participantes são criptografadas ou ofuscadas. Os dados brutos dos participantes não são expostos uns aos outros, e cada participante recebe apenas o parâmetros do modelo relacionados aos seus próprios recursos.

5.2.5. *Transfer Learning* para FL

Federated Transfer Learning (FTL) é um caso especial de FL e diferente do FL horizontal (HFL) e vertical (VFL) [Saha and Ahmad 2021]. No FTL, dois conjuntos de dados diferem no espaço de características. Isso se aplica a conjuntos de dados coletados de empresas de natureza diferente. Devido às diferenças na natureza dos negócios, essas empresas compartilham apenas uma pequena sobreposição no espaço de características. Isso também se aplica às empresas estabelecidas em todo o mundo. Assim, em tais cenários, os conjuntos de dados diferem tanto nas amostras quanto no espaço de recursos.

Para [Li et al. 2020b] o principal problema neste cenário é a falta ou baixa qualidade dos rótulos. O Transfer Learning (TF) permite que o conhecimento de um domínio (ou seja, o domínio de origem) seja movido para outro domínio (o domínio de destino) para obter melhores resultados de aprendizagem, o que é apropriado para esta situação. Desta forma, alguns autores conceberam FTL para generalizar FL para ter uma aplicação mais ampla para uma pequena interseção de clientes e características em comum. Os autores criaram um modelo, o FedHealth, que reúne dados pertencentes a diferentes organizações por meio da FL e oferece atendimento médico personalizado por meio de

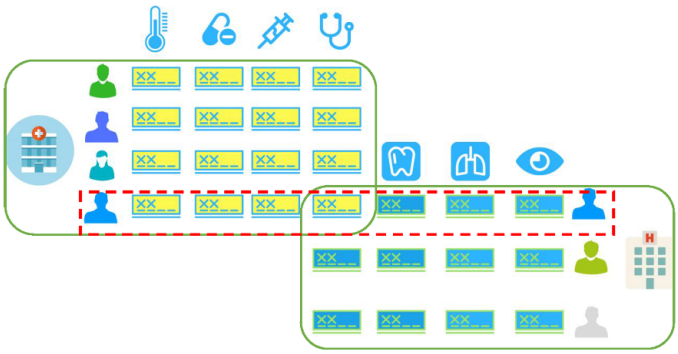


Figura 5.4. Exemplo de aplicação de FTL [Li et al. 2020b]

aprendizado por transferência. Alguns conhecimentos de diagnóstico e tratamento de doenças em um hospital podem ser transferidas para outro hospital para ajudar a diagnosticar outras doenças por FTL.

A pesquisa de FTL ainda não está madura, então ainda há muito espaço para crescimento para torná-la mais flexível com diferentes estruturas de dados. Os silos de dados e as questões de proteção de privacidade são problemas significativos encontrados na industrialização em grande escala do aprendizado de máquina atual. No entanto, o aprendizado por transferência federada é uma forma eficaz de proteger a segurança dos dados e a privacidade do usuário, quebrando as barreiras dos silos de dados.

A Figura 5.4 mostra um exemplo de FTL em dados médicos, onde o mesmo paciente pode ser tratado em hospitais diferentes, mas os mesmos não podem compartilhar os dados devido à questões de privacidade e os dados não possuírem muitos pontos em comum. Nesse caso, utilizando o ponto comum possível, é feito um FTL para que os dados do paciente possam ser aproveitados pelo modelo para gerar outro modelo que consiga compreender os novos dados do paciente.

[Saha and Ahmad 2021] cita algumas aplicações habilitadoras da tecnologia FTL, a saber: dispositivo vestível que está rapidamente se tornando parte da vida cotidiana de pacientes e profissionais de saúde. Múltiplos recursos e funcionalidades de dispositivos vestíveis incluem monitoramento remoto de pacientes, rastreamento e coleta de dados, melhorando a saúde diária e os padrões de estilo de vida, detectando condições crônicas, entre outros. Os dados de saúde, no entanto, são geralmente fragmentados e privados, dificultando a geração de resultados robustos entre as populações. Os dados do usuário gerados por dispositivos de saúde geralmente existem na forma de ilhas isoladas. Para mais avaliações e análises dos resultados obtidos, os dados coletados são carregados para o servidor remoto baseado em nuvem. Mas esse método possui alguns problemas como: (i) **restrições regulatórias** - falta de aquisição de dados massivos de usuário, (ii) **segurança e privacidade** - restringe o compartilhamento de dados que existem na forma de silos isoladas, (iii) **questão de personalização** - o processo de treinamento do modelo de aprendizado de máquina carece de personalização.

Ainda para [Saha and Ahmad 2021] o aprendizado de máquina depende da disponibilidade de grandes quantidades de dados para treinamento. No entanto, em cenários da vida real, os dados estão espalhados, principalmente, por diferentes organizações e

não podem ser facilmente integrados devido a muitas restrições legais e práticas. O FTL ajuda a melhorar a modelagem estatística em uma federação de dados. A federação de dados no caso de FTL permite que o conhecimento seja compartilhado sem comprometer a privacidade do usuário e permite que o conhecimento complementar seja transferido na rede. Assim, permitindo que a parte de domínio alvo desenvolva um modelo mais flexível e poderoso, aproveitando rótulos ricos da parte de domínio de origem.

A ampla aplicação de FTL é atualmente dificultada pela disponibilidade limitada de conjunto de dados para treinamento e validação de algoritmo, devido à ausência de abordagens técnicas e jurídicas para proteger a privacidade do usuário. A FTL tem requisitos estritos de preservação da privacidade, portanto, para evitar o comprometimento da privacidade do usuário enquanto promove a pesquisa científica em grandes conjuntos de dados, a implementação de soluções e desenvolvimento/implementação de quadros jurídicos para atender simultaneamente às demandas de proteção e utilização de dados são obrigatórios. A FTL para preservação da privacidade normalmente envolve várias partes, com ênfase nas garantias de segurança para realizar o aprendizado de máquina. Alguns dos métodos mais utilizados foram apresentados na Seção 5.3.2.

5.3. Preservação de Privacidade em FL

Apesar de sua alta aplicabilidade, modelos centralizados podem não ser uma alternativa aceitável em contextos em que, devido à preocupações com privacidade, deseja-se manter dados sensíveis (dados hospitalares, conversas pessoais, informações bancárias) em seu ponto de origem.

Devido ao pouco conhecimento popular sobre o assunto e à pouca transparência dos processos de Inteligência Artificial, há também desconfianças sobre a fiabilidade do uso de ML na sociedade. Algumas polêmicas recentes que podemos destacar são: o algoritmo do YouTube censurar vídeos de conteúdo LGBTQIA+ por considerá-los de cunho impróprio e sexual[3]; a inteligência artifical do Twitter, responsável por fazer o enquadramento de rostos em fotos postadas na plataforma, ter viés racista e reconhecer mais rostos de pessoas brancas[4]; o caso de um bot no Telegram que tinha a capacidade de "tirar a roupa" de mulheres em fotos, sendo muito usado em casos de *revenge porn* e colaborar para o cyberbullying e assédio sexual online[5]; o caso do crescimento do uso de *DeepFakes*, que, apesar de poderem ser identificadas através de métodos eficientes, podem mudar a opinião popular e gerar uma maior desconfiaça sobre IAs[6]; e, por fim, o caso de "filtros de embelezamento" serem ativados automaticamente, sem consentimento e sem possibilidade de serem desativados, em contas de *influencers*[7]. Um outro aspecto relevante é a regulação de vários países no sentido de preservar a posse e a privacidade

[3] https://www.metropoles.com/entretenimento/youtube/estudo-revela-mais-de-900-palavras-desmonetizaveis-no-youtube

[4] https://www.bbc.com/news/technology-57192898

[5] https://www.cnet.com/news/deepfake-bot-on-telegram-is-violating-women\--by-forging-nudes-from-regular-pics/

[6] https://www.cnet.com/features/deepfakes-threat-2020-us-election-isnt-\-what-youd-think/

[7] https://olhardigital.com.br/2021/06/10/internet-e-redes-sociais/tiktok-mudou-a-fisionomia-de-usuarios-sem-consulta-los/

dos dados dos indivíduos, como já discutido na Seção 5.1.

Neste sentido, técnicas de aprendizado federado têm o potencial de minimizar problemas de privacidade decorrentes de modelos centralizados de ML, uma vez que elas diminuem (ou eliminam) o tráfego de informações sensíveis entre usuários. Entretanto, FL não está imune à vulnerabilidades decorrentes da exposição de parâmetros encontrados no treinamento colaborativo. Os valores de parâmetros podem estar correlacionados aos dados sensíveis e, portanto, podem revelar informações sobre estes dados.

A necessidade de um número maior de iterações, comunicações e compartilhamento de informações entre os clientes expõe o ambiente federado a novos possíveis riscos e ataques, pois aumentam a viabilidade de manipulação de outputs do modelo de IA e do acesso a dados confidenciais de usuários. Diante ao exposto, três grandes fundamentos de segurança da informação, também conhecidos como CIA, devem ser aplicados em todas tecnologias adeptas ao aprendizado federado: confidencialidade, integridade e disponibilidade.

Devido ao aumento da consciência pública em relação à problemáticas de preservação de privacidade dos dados, um novo enfoque de desenvolvimento científico é o estudo de soluções específicas para preservação de privacidade em ambientes de aprendizado de máquina (*privacy-preserving machine learning* - PPML).

5.3.1. Tipos de Adversários

Um conceito inicial extremamente importante para a compreensão de técnicas de segurança e privacidade é a definição de adversário. Adversário é todo aquele usuário ou entidade que por algum motivo teve acesso à uma informação do sistema que não deveria ter. Desse modo, podemos dividir os tipos de adversários em duas categorias: Os semi-honestos e os maliciosos.

Adversários maliciosos são aqueles que possuem intenção de sabotar o funcionamento do algoritmo, seja roubando informações privadas ou injetando erros nos modelos. Tais adversários podem burlar os protocolos estabelecidos e agir de modo inesperado para o sistema. No entanto, no contexto específico de aprendizado federado, uma vez que este funciona por meio de práticas colaborativas, ataques realizados por adversários maliciosos não são muito comuns, dando um maior espaço para ataques provenientes de adversários semi-honestos. Adversários semi-honestos são aqueles que cumprem honestamente as regras estabelecidas pelo sistema/software, mas por curiosidade aprendem mais informações sensíveis do que deveriam.

Para executar suas ações, um adversário pode explorar seu conhecimento a respeito de como o modelo de aprendizado funciona. Existem duas classificações principais sobre o que o adversário sabe a respeito do modelo: (i) **Modelo caixa branca**: assume que o adversário possui conhecimento da arquitetura do modelo em si, porém sem acesso aos dados de treinamento utilizados para otimizá-lo. (ii) **Modelo caixa preta**: assume que o adversário não possui acesso ao modelo, pois o mesmo se encontra criptografado. Desse modo, este pode apenas consultar o modelo enviando dados e obtendo suas respectivas respostas.

5.3.2. Principais Técnicas de PPML

Secure Multi-party computation (SMC ou MPC): No ambiente federado, o objetivo consiste em construir uma função única em conjunto com todos os clientes, de tal modo que essas entradas sejam preservadas e privadas para as outras partes da federação, ao mesmo tempo em que permanecem adequadas para o treinamento do modelo. Neste caso, é mais usado para proteger os dados de atualização dos clientes. No entanto, um problema dessa técnica é o fato de exigir muito tempo de treinamento, o que pode acabar resultando em perda de dados em um contexto federado [Mothukuri et al. 2021].

Computação Homomórfica: Consiste em um conjunto de técnicas de criptografia que codificam os dados de modo que operações algébricas possam ser aplicadas eficientemente no texto cifrado. Assim, podemos executar algoritmos de aprendizado de máquina em cima dos dados sem a necessidade de descriptografá-los, garantindo a privacidade dos clientes envolvidos na federação.

Privacidade Diferencial: Consiste em inserir ruídos em dados sensíveis, de forma controlada, para que não seja possível identificar a entrada original. Logo, pode ser aplicada para confundir adversários em ataques a dados sensíveis (como por exemplo, em ataques de inferência de associação). Uma vez que o ruído é conhecido pelo sistema, é possível levá-lo em conta no modelo e assim garantir que a inferência continue aproximadamente válida, embora não consigamos identificar mais a entrada. Em FL, esses ruídos são introduzidos nos parâmetros dos usuários. Apesar de aumentar muito a segurança em comparação com a pouca perda estatística de dados, essa técnica gera incerteza perante os resultados de treino, além de dificultar a avaliação do comportamento do usuário [Mothukuri et al. 2021].

5.3.3. Tipos de Ataques

Uma vez que o aprendizado federado é feito de forma descentralizada, a proteção de privacidade foca na proteção dos gradientes transmitidos na fase de treinamento dos modelos, impedindo que informações sensíveis dos clientes sejam vazadas ou possam ser reconstruídas. Logo, o foco será nos principais ataques cujo objetivo consiste em reconstruir e/ou aprender informações dos clientes/modelo com base nos dados trocados no processo de treinamento federado.

Ataques de reconstrução: Ataques de reconstrução possuem como objetivo extrair informações dos dados brutos dos clientes no modelo federado ou das *features* que os representam. Como no modelo federado cada usuário treina localmente o modelo e apenas os gradientes são compartilhados, novos ataques vêm surgindo para inferir informações dos dados utilizados para realizar o treinamento local [Aono et al. 2016]. **Formas de se proteger contra ataques de reconstrução**: modelos que utilizam as *features* explicitamente, como máquina de vetores de suporte (SVM) e os k-vizinhos mais próximos (knn) devem ser evitados. Durante o treinamento do modelo, a computação multipartidária segura (MPC) [Yao 1982] e a criptografia homomórfica (HE) [Rivest et al. 1978] devem ser usados para proteger a privacidade da consulta do usuário durante a inferência do modelo caixa preta.

Ataques de inversão de modelo: Ataques de inversão de modelo capturam informações sobre os dados usados durante o treinamento ao conseguir acesso ao modelo (seja por caixa branca ou por caixa preta). Com base nas soluções retornadas pelo modelo, o adversário consegue reconstruí-lo (mesmo que aproximadamente), seja por meio de ataques de resolução de equações (teoricamente, um modelo linear N-dimensional pode ser descoberto com apenas N - 1 consultas) ou treinando um modelo semelhante passando entradas e as respectivas respostas obtidas pelo modelo original a que se quer copiar. **Formas de proteção contra ataques de inversão de modelo**: A regra principal para se proteger desse tipo de ataque é evitar expor ao máximo o modelo e suas saídas. Técnicas de agregação de resultados durante o treinamento [Fredrikson et al. 2015, Al-Rubaie and Chang 2016] e combinação de criptografia homomórfica vem sendo amplamente utilizadas [Xie et al. 2019].

Ataques de inferência de associação: Usa a saída do modelo caixa preta para verificar se uma determinada amostra se encontra ou não presente em um dado conjunto de treinamento. **Formas de proteção contra ataques de inferência de associação**: Novamente, técnicas de generalização de resultados podem ser utilizadas para evitar ataques. Porém, técnicas de privacidade diferencial também vêm se mostrando eficazes [Shokri et al. 2017].

Ataque de re-identificação: O objetivo do ataque aqui é desanonimizar informações de identificação pessoal nos dados de usuários utilizados no treinamento do modelo. Mesmo que as features sensíveis sejam removidas anteriormente, adversários se utilizam de conhecimentos prévios ou cruzamento de diferentes bases de dados para obter tais informações. **Formas de proteção contra ataques de re-identificação**: Técnicas de privacidade e anonimato em grupo (tal técnica também é baseada na privacidade diferencial) baseadas em algoritmos de generalização são implementadas para evitar esses ataques.

Ataque de envenenamento de modelos: Também conhecidos como ataques backdoor, esse tipo de ataque consiste em treinar maliciosamente um modelo ML, modificando os seus parâmetros, com o objetivo de invalidá-lo ou treiná-lo para um propósito específico (por exemplo, aumentar a atualização do modelo do participante malicioso). Até mesmo clientes maliciosos no aprendizado federado com um pequeno conjunto de dados possuem altas probabilidades de realizar um envenenamento do modelo com sucesso. **Formas de proteção contra ataques de envenenamento de modelos**: Modelos baseados em algoritmos de blockchain [Preuveneers et al. 2018].

A Tabela 1 sumariza os principais ataques de privacidade em FL, as garantias que as técnicas devem fornecer para evitá-los e os principais algoritmos de prevenção dos mesmos.

5.4. Compressão de Modelos no Contexto de FL

Os modelos de aprendizado de máquina costumam demandar alta capacidade de recursos em termos de processamento e de memória para que se possa treinar modelos com alta capacidade de expressividade. No contexto de IoT, raramente encontraremos as exigências

Tabela 5.1. Principais ataques de privacidade em Federated Learning, as garantias que as técnicas devem fornecer para evitá-los e os principais algoritmos de prevenção a ataques.

	Computação multipartidária segura	Criptografia Homomórfica	Agregação de resultados	Privacidade Diferencial	Blockchain
Reconstrução (Garantia da privacidade do usuário)	✓	✓			
Inversão de modelo (Garantia da privacidade do usuário)		✓	✓		
Inferência de associação (Garantia da privacidade do usuário)			✓	✓	
Re-identificação (Garantia da privacidade do usuário)				✓	✓
Envenenamento de modelos (Garantia da consistência e integridade do modelo)					✓

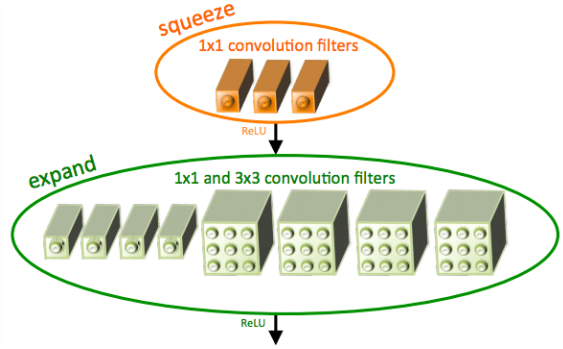


Figura 5.5. Fire Module, parte da SqueezeNet, proposto em [Iandola et al. 2016]

para que esses modelos sejam treinados adequadamente. Nesta seção, apresentaremos três possíveis abordagens para endereçar este problema, enquanto que na Seção 5.6 apresentamos uma abordagem, em outra direção, onde a Computação de Borda podeerá ser utilizada para dar suporte a esse tipo de demanda.

Adicionalmente, tenha em mente que todas as técnicas apresentadas aqui, embora discutidas separadamente, não são excludentes. Ou seja, pode-se utilizar uma ou mais técnicas de compressão para formar um framework de IoT em cenários federados. Por exemplo, é possível podar modelos compactos.

5.4.1. Desenvolvimento de Modelos Compactos

Os modelos compactos, também chamados de micro-arquiteturas, são modelos profundos focados em eficiência de memória e computação, além de bons resultados de classificação. Essa eficiência é atingida não somente diminuindo a quantidade de camadas, mas também a complexidade das mesmas, o que resulta em modelos com menos parâmetros (ou seja, modelos que ocupam menos memória) [Rastegari et al. 2016, Iandola et al. 2016]. Esse é um tópico bastante estudado em ML tradicional e que pode ser facilmente utilizado em FL, inclusive implementado em aparelhos IoT com recursos limitados. A seguir, são apresentados os principais modelos da literatura.

5.4.1.1. SqueezeNet

SqueezeNet [Iandola et al. 2016] é uma arquitetura de CNN que atinge resultados competitivos com grandes arquiteturas utilizando menos parâmetros. Por exemplo, esse modelo chega a possuir 50 vezes menos parâmetros que a AlexNet. Isso é conseguido através do *fire module*, mostrado na Figura 5.5, que é composto de uma camada *squeeze* (em laranja) e uma camada *expand* (em verde). O design dessas camadas segue as seguintes estratégias: (i) **Substituição dos filtros e limitação dos canais de entrada:** a quantidade total de parâmetros presente em uma camada convolucional é *(número de canais de entrada)* · *(número de filtros)* · *(tamanho do filtro)*. Logo, é preciso encontrar um balanço entre essas três variáveis a fim de atingir o objetivo de diminuição de parâmetros com preservação de acurácia. Com isso em mente, no *fire module*, substitui-se a maioria dos filtros 3×3 (comumente usados na literatura) por filtros 1×1, que possui 9 vezes menos parâmetros

que o filtro anterior. Além disso, limita-se a quantidade de filtros das camadas *squeeze* para ser menor que o número de filtros nas camadas *expand*, o que diminui a quantidade de canais de entrada nos filtros 3×3. (ii) ***Downsampling* mais próximo ao final da arquitetura para criação de mapas de ativação maiores:** mapas de ativação determinam para qual parte da imagem uma camada convolucional vai ativar. Intuitivamente, mapas de ativações maiores podem levar a melhores resultados de classificação, uma vez que a camada convolucional capturará os atributos essenciais de uma imagem sem ser perturbada por detalhes (que podem ser ruídos). O tamanho desse mapa é controlado pelo tamanho da imagem e em qual parte da arquitetura há um *downsampling* (isto é, em quais camadas de convolução e *pooling* aumenta-se o *stride* para maior que um). Se camadas no começo da arquitetura possuem *stride* maior, então a maioria das camadas vai ter um mapa pequeno. Por outro lado, se *stride* maiores aparecem mais no final da arquitetura, então a maioria das camadas terá mapas de ativação grande. Portanto, na construção da SqueezeNet, os *fire module* iniciais possuem um *stride* pequeno, que é aumentado mais ao fim da arquitetura.

5.4.1.2. MobileNet

MobileNets [Howard et al. 2017] são arquiteturas de CNN projetadas para uso em aparelhos móveis, assim sendo, são leves e de baixa complexidade. A ideia principal por trás dessas arquiteturas é a convolução separável em profundidade (*Depthwise Separable Convolution*). A Figura 5.6 mostra a diferença entre a convolução comum (esquerda) e a convolução separável em profundidade (direita). Uma camada convolucional comum aplica o filtro convolucional em todos os canais da imagem, somando os pesos dos pixels da imagem de entrada. Com isso, todos os canais da imagem de entrada são combinados durante a operação. Já a convolução separável em profundidade realiza a convolução em cada canal da imagem separadamente, com cada canal tendo seus próprios pesos. Após essa convolução em profundidade, os pesos dos canais são somados através de uma convolução *pointwise* (que é o mesmo que uma convolução comum, mas com filtro de tamanho 1×1), que combina os canais de saída.

Dessa forma, enquanto a convolução comum filtra e combina os canais de uma única vez, a convolução separável em profundidade realiza as duas operações separadamente. Embora os resultados obtidos sejam similares (filtram e fazem novas *features*), uma convolução comum utiliza muito mais poder computacional e precisa aprender muito mais pesos. Por isso, a convolução separável em profundidade é muito mais rápida que a convolução comum.

Em [Sandler et al. 2018], é proposta uma extensão das MobileNets, chamada de MobileNetV2. Essa nova arquitetura é bem semelhante ao MobileNet original, porém é muito mais rápida, atingindo a mesma acurácia. Por exemplo, esse novo modelo usa duas vezes menos operações, precisa de 30% menos parâmetros e é cerca de 30% a 40% mais rápida que os modelos da primeira versão, atingindo cerca de 4% a mais de acurácia[8].

Além do uso das convoluções separáveis em profundidade, nessa segunda versão é

[8] `https://ai.googleblog.com/2018/04/mobilenetv2-next-generation-of-on.html`

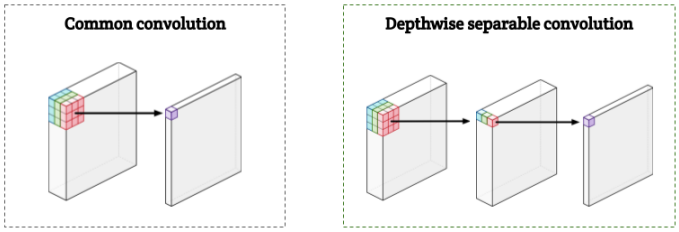


Figura 5.6. Depthwise Separable Convolution [Hollemans 2017]

introduzido um novo componente chamado *Inverted Residual Block*. Esse bloco é uma variação do *Residual Block*, parte da famosa arquitetura profunda ResNet [He et al. 2016]. Na MobileNetV2, o bloco proposto é formado por três camadas convolucionais comuns com filtros 1×1, 3×3 e 1×1, respectivamente. As duas últimas camadas são as camadas convolucionais já utilizadas na primeira versão dessa arquitetura, uma convolução separável em profundidade e uma convolução *pointwise*. Porém, a última camada agora tem um papel diferente. Enquanto na primeira versão essa camada mantinha o número de canais ou os dobrava, nessa segunda versão acontece o oposto: a quantidade de canais diminui através da projeção dos dados de uma dimensão maior em um tensor com menor número de dimensões (por isso essa camada agora é chamada de camada de projeção). A primeira camada, por sua vez, é uma camada semelhante a última, mas em vez de projetar os dados, os expande. Chamada de camada de expansão, a ideia é ampliar a quantidade de canais antes da convolução separável em profundidade.

Intuitivamente, esse bloco funciona como um compressor e descompressor de informações: a camada de expansão descomprime a informação para que possam ser extraídas com mais facilidade pela camada do meio e a camada de projeção comprime as informações novamente, tornando a representação pequena mais uma vez. Dessa forma, essa diminuição, aumento e diminuição novamente das dimensões força a rede neural a aprender uma representação mais compacta sem perder tanta qualidade nos resultados.

Além disso, as entradas e saídas para esse bloco possuem uma conexão residual, o que permite a rede a oportunidade de acessar ativações que não foram modificadas pela convolução, essencial para construir redes neurais muito profundas e permitir um treinamento mais rápido e acurado, como nas conexões residuais das ResNet [Sandler et al. 2018].

5.4.2. Compressão de Modelos Pré-treinados

O objetivo da compressão de modelos pré-treinados é simplificar um modelo já existente, de forma a obter arquiteturas de DL mais simples, porém sem perder acurácia de forma significativa. Essa simplificação tem dois principais focos: (i) **Redução de tamanho:** o modelo comprimido possui menos parâmetros, logo, utiliza menos memória RAM durante sua execução; (ii) **Redução de latência:** o modelo comprimido leva menos tempo para fazer uma inferência, o que leva a um menor consumo de energia.

Em ambientes com recursos escassos, como IoT, ambas as simplificações são desejáveis. Elas são usualmente atingidas concomitantemente, uma vez que modelos maiores precisam de mais memória e mais energia para sua execução [Cheng et al. 2017].

Similar ao tópico anterior, essa também é uma área muito estudada em ML tradicional que está sendo aos poucos aplicada em FL. Porém, muitas dessas técnicas não tem um mapeamento fácil para o universo federado ou tornam o processo de treinamento mais complexo, complicando sua adoção nesse cenário [Kairouz et al. 2021]. A seguir, discute-se os métodos mais famosos em compressão de modelos, destacando suas vantagens e desvantagens, especialmente em ambientes de IoT federado.

5.4.2.1. Poda

A motivação para podar uma rede neural é remover parâmetros redundantes e desimportantes que não vão ter grande impacto no desempenho, uma vez que redes neurais tendem a ser super-parametrizadas, com vários atributos contendo praticamente a mesma informação. Há dois tipos de poda, conforme o componente que é removido da rede neural: (i) **poda desestruturada:** remove-se pesos ou neurônios individualmente, ao zerar seus valores na matriz de pesos da rede neural, o que aumenta sua esparsidade. Por um lado, uma vez que diversos hardwares e softwares são especializados em operações nesse tipo de matriz, pode-se melhorar o desempenho do modelo em relação ao tempo e energia. Por outro lado, essa necessidade de especialização para acelerar a computação é uma limitação significativa, especialmente em aparelhos genéricos de IoT. (ii) **poda estruturada:** esse tipo de estratégia remove blocos inteiros de pesos, canais e filtros, o que não resultará em matrizes esparsas, superando a necessidade de hardware e software especializado [Han et al. 2015, Cheng et al. 2017].

Uma das formas mais simples e mais utilizadas de poda, tanto estruturada quanto desestruturada, é a poda baseada em magnitude (*magnitude-based pruning*): primeiramente, treina-se a rede neural para aprender quais as conexões importantes; o segundo passo é podar a estrutura desejada com valor abaixo de um valor de corte; e, finalmente, retreina-se a rede neural para que a mesma possa se ajustar de forma a compensar pelos pesos perdidos. Esse processo é repetido por várias iterações. Com esse método, por exemplo, a AlexNet é diminuída de 61 milhões para 6,7 milhões de parâmetros, sem perda de acurácia [Han et al. 2015]. Esse resultado impressionante possibilita a implementação dessas redes grandes em aparelhos IoT.

Apesar da sua simplicidade e capacidade de aplicação em camadas convolucionais e totalmente conectadas, a poda necessita de muito *fine-tuning* e não está claro quão bem generaliza entre diferente arquiteturas. Em muitos casos, é mais efetivo usar uma arquitetura mais eficiente (como modelos compactos) que podar um modelo pré-treinado [Blalock et al. 2020].

A poda pode ser utilizada em ambientes federados de diversas maneiras. Por exemplo, pode-se manter um modelo pré-treinado no servidor e aumentar ou diminuir (podar) suas "cópias" a depender da capacidade do cliente, ou seja, uma poda adaptativa, de forma a acelerar a computação necessária sem perder acurácia [Jiang et al. 2019].

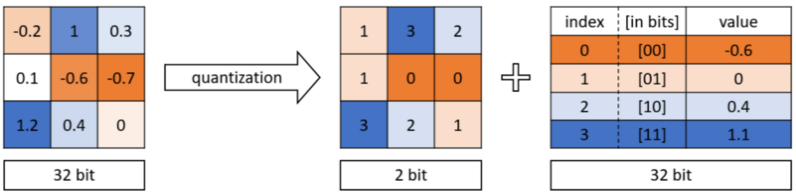


Figura 5.7. Exemplo de binarização, um tipo de quantização que resulta em uma representação em 2 bits [Tamuly 2018]

5.4.2.2. Quantização

Enquanto a poda comprime os modelos de redes neurais através da diminuição da quantidade de pesos, a quantização diminui o tamanho dos parâmetros (pesos, ativações ou gradientes) que existem através da mudança de representação, isto é, alterando o número de bits utilizados.

Há diversas formas de realizar uma quantização, como pode ser visto em [Guo 2018]. A mais simples consiste em pegar os valores máximos e mínimos e dividir pela quantidade de bits esperada, encaixando os valores no intervalo em que estão mais próximos. Por exemplo, em redes neurais profundas, os pesos são tipicamente armazenados como números de ponto flutuante de 32 bits. Ao diminuir essa representação para 2 bits, o que é chamado de binarização, passamos de 2^{32} possíveis valores para somente 4. A Figura 5.7 exemplifica a binarização. Perceba que esse tipo de transformação naturalmente levará a uma perda de "precisão" numérica; como visto na imagem, os valores $-0,6$ e $-0,7$ em 32 bits são representados como 0 em 2 bits. Porém, esse tipo de perda não deve afetar muito o resultado da rede neural profunda, devido a capacidade de lidar com ruídos que estas possuem.

Além disso, embora a quantização de outros parâmetros como ativações ou gradientes utilize ferramentas semelhantes, essa é uma atividade mais complicada, pois deve-se lidar com *bit overflow* e métodos de aproximação de valores [Guo 2018]. Alguns softwares famosos na área de DL possuem funções para quantizar os operadores mais utilizados, como o Tensorflow[9]. Porém, esse tipo de abordagem requer o uso de softwares especializados, o que pode ser visto como uma desvantagem.

Outra desvantagem no uso de quantização é a necessidade de entender o hardware utilizado. Ou seja, é necessário saber como o hardware realiza as operações bit a bit para implementar a quantização desejada. Isso pode ser um problema especialmente em ambientes IoT federados, onde os vários objetos podem possuir diferentes hardwares.

Mais ainda, a quantização torna mais difícil a convergência das redes neurais, necessitando de uma taxa de aprendizado menor para garantir um bom desempenho [Guo 2018]. Isso também torna seu uso mais complicado em ambientes federados, por necessitar de mais rodadas para garantir um bom aprendizado.

Um exemplo de uso de quantização em ambiente federado pode ser visto em [Konečnỳ et al. 2016], onde os autores treinam os modelos sem nenhuma restrição

[9] `https://www.tensorflow.org/lite/performance/post_training_quantization`

nos clientes e quantizam esses valores para diminuir o custo de comunicação entre os clientes e o servidor. O servidor é responsável por decodificar os modelos antes de agregá-los. Essa abordagem é interessante por contornar os problemas descritos anteriormente, apesar deste uso não acelerar a computação dos parâmetros dos modelos, que é o objetivo principal da compressão.

5.4.2.3. Low-rank Factorization

Como dito anteriormente, redes neurais profundas tendem a ser super-parametrizadas, com muita similaridade entre várias camadas ou canais. A ideia de *Low-rank factorization* é então decompor a matriz de pesos de uma camada em uma combinação linear de matrizes menores. Em outras palavras, uma camada mais complexa é decomposta em camadas menores que aproximam os valores da camada original; como discutido na seção anterior, as redes neurais profundas conseguem lidar com essas aproximações sem ter seu desempenho tão afetado, pela sua capacidade de lidar com ruídos.

Diversas vantagens podem ser obtidas utilizando essa estratégia. Por exemplo, a fatoração diminui a memória utilizada pela rede neural e torna as operações menos complexas, o que pode acelerar significativamente o desempenho. Além disso, não há a necessidade de softwares ou hardwares específicos para seu uso. Porém, deve-se tomar cuidado com a necessidade de *fine-tuning* para o reajuste dos pesos das novas camadas, o que pode ser visto como uma desvantagem, principalmente dentro do ambiente federado, por necessitar de mais rodadas.

Há diversas técnicas de fatoração, sendo a decomposição em valores singulares (*Singular Value Decomposition* – SVD) uma das mais conhecidas. Essa técnica é uma fatoração de matrizes, porém, as redes neurais são comumente implementadas como tensores (uma abstração de escalares, vetores e matrizes). Com isso, é necessário generalizar SVD para tensores, o que é bem estudado no contexto de redes neurais profundas. Por exemplo, [Kim et al. 2015] utilizaram a decomposição de Tucker na AlexNet, diminuindo o modelo para 11 MB, o que permitiu sua execução em um *smartphone* com uma perda mínima de acurácia.

Em FL, temos a aplicação descrita em [Qiao et al. 2021], a título de exemplo. Após os clientes realizarem seu treino local, utiliza-se técnicas de *Low-rank factorization* para comprimir os parâmetros do modelo, o qual é enviado para o servidor. O servidor agrega os modelos comprimidos e também fatora esse modelo global antes de enviá-lo aos clientes. Essa fatoração dupla, além de acelerar a computação, diminui o custo de comunicação tanto no envio quanto no recebimento dos modelos pelos clientes.

5.4.2.4. Knowledge Distillation

A motivação para *Knowledge Distillation* (KD) vem ao considerar que treino e inferência são duas atividades diferentes, então, é possível utilizar modelos diferentes para cada tarefa. Assim, um modelo complexo, chamado de professor, é treinado e seu conhecimento é transferido para um modelo de menor capacidade, chamado de estudante, que realizará

as inferências.

Nesse contexto, o conhecimento que será transferido pode ser construído de diversas formas. Tipicamente, esse conhecimento é a distribuição de saída de um modelo professor pré-treinado, mas também pode ser um *ensemble* da distribuição de saída de outros modelos estudantes (esse processo é chamado de co-destilação). Assim, o modelo estudante atualiza seus pesos através da minimização de sua própria função de perda e um regularizador de destilação que penaliza o estudante quando a saída do estudante e do professor (ou do *ensemble* de estudantes) possui uma distância grande.

Uma desvantagem do KD é que muitas decisões devem ser tomadas pelo usuário; por exemplo, os modelos professor e estudante não precisam ter a mesma arquitetura, o que significa decisões de treinamento (como escolher hiper-parâmetros) para duas redes neurais. Porém, isso também significa que esta é uma técnica muito flexível e que pode se adaptar a um grande número de atividades. Isso é especialmente importante no contexto de IoT, que possui as mais diversas aplicações.

Adicionalmente, note que para a construção do conhecimento é necessário que os modelos estudante e professor tenham amostras de treinamento comuns a ambos. Ou seja, é necessário que todos os modelos possam observar os mesmos dados para calcular sua função de perda, demandando troca de dados – o que fere o princípio do aprendizado federado. Uma forma de evitar esse problema, possibilitando o uso dessa técnica em ambientes federados, é a criação de uma base de dados *proxy*, onde as amostras são agrupadas de acordo com suas classes da seguinte maneira: cada estudante armazena a média da distribuição de saída de cada classe obtida durante seu treino local; periodicamente, essa média é enviada para o servidor, que agrega todas as distribuições enviadas por todos Fos modelos estudantes; essa média global é considerada o conhecimento do modelo professor e é usado para auxiliar no aprendizado [Seo et al. 2020].

5.4.2.5. Atenção Seletiva

A atenção seletiva é uma característica do processamento humano de informações: quando olhamos ou ouvimos algo, a informação é processada por partes; vamos nos tornando conscientes das partes quando necessário, ignorando os outros detalhes. Ou seja, vamos aprendendo a dar mais atenção ao que é útil para a identificação do que queremos. Isso permite integrar e processar de forma rápida e eficiente uma quantidade massiva de informação usando recursos limitados de processamento.

Com essa inspiração, mecanismos de atenção seletiva tem por objetivo focar nos elementos de interesse, descartando os que são irrelevantes para a atividade em curso. Em cenários de IoT, onde muitos objetos possuem poder computacional limitado, tais mecanismos podem ser utilizados como um esquema de alocação de recursos, ajudando a selecionar e processar as informações importantes. Por exemplo, na detecção de objetos, pode-se menosprezar o fundo das imagens e objetos de cores diferentes do desejado. Esse método traz diversas vantagens, como aceleração da inferência, modelos menores e ganho de acurácia. Porém, é necessário adicionar o componente de atenção seletiva a rede neural, o que pode tornar o treinamento um pouco mais custoso [Niu et al. 2021].

Umas das formas de utilização de mecanismos de atenção seletiva em FL é descrita em [Ji et al. 2019]. Nela, os autores propõem um método de agregação com atenção (*Attentive Federated Aggregation* - FedAtt), onde aplica-se um mecanismo de atenção em todas as camadas dos clientes, calculando a contribuição de cada camada (e, consequentemente, focando nas camadas mais importantes), de forma a minimizar a distância entre o modelo do servidor e dos clientes.

5.4.2.6. Loterry Ticket Hypothesis

Como visto anteriormente, embora seja possível podar redes neurais densas para pouquíssimos parâmetros (criando uma sub-rede esparsa) sem degradar muito o desempenho, por muito tempo não se conseguia treinar uma sub-rede esparsa *from scratch*. Isso quer dizer que treinar uma rede neural com a mesma topologia que a sub-rede levava a resultados piores de acurácia, dessa forma, sempre era necessário construir a rede neural densa e podá-la.

Em 2019, [Frankle and Carbin 2019] notaram que, após a obtenção das sub-redes esparsas, as mesmas eram inicializadas aleatoriamente. Com isso, eles propuseram a hipótese do tíquete de loteria, definida da seguinte maneira: *"uma rede neural profunda aleatoriamente inicializada contém uma sub-rede que, se reinicializada e treinada isoladamente, pode conseguir resultados de acurácia melhores que a rede original depois de treinada por no máximo o mesmo número de iterações."*. Usando a metáfora do título, vencer a loteria é equivalente a treinar uma rede neural com boa acurácia. Logo, um tíquete de loteria é equivalente aos parâmetros de uma rede neural: comprar um milhão de tíquetes (construir uma rede neural super-parametrizada) te dá mais chances de ganhar na loteria que comprar somente um tíquete (construir uma rede mais simples).

O processo para obter uma sub-rede "vencedora da loteria" é o seguinte: inicializa-se aleatoriamente uma rede neural e a treina; após o treino, a rede é podada, por exemplo, através da poda baseada em magnitude; por fim, reestabelece-se os pesos da sub-rede para os pesos ao qual a rede original foi inicializada. Esse processo é feito iterativamente até se encontrar o nível desejado de esparsidade ou caso a acurácia caia significativamente.

As sub-redes "vencedoras da loteria" são até 20% menores que as originais e podem ter a mesma, ou até superar, a acurácia das redes originais, com no máximo o mesmo número de iterações. Porém, as iterações necessárias para a poda é uma desvantagem, por ser muito caro computacionalmente.

Em [Li et al. 2020a], os autores aplicam os conceitos da hipótese do tíquete de loteria no contexto de aprendizado federado. Neste trabalho, cada cliente deve aprender uma sub-rede esparsa, onde a poda é feita de acordo com os dados locais. O servidor agrega os modelos utilizando somente os parâmetros das sub-redes, e cada cliente recebe os parâmetros do seu próprio modelo. Além das vantagens do aprendizado federado, essa técnica permite a criação de um modelo personalizado para cada cliente.

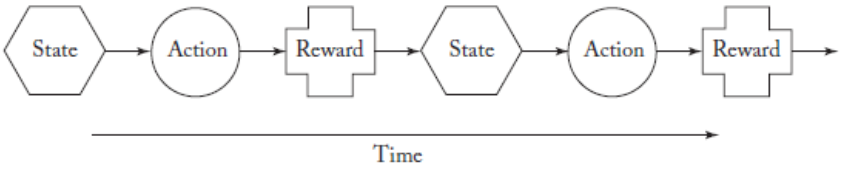


Figura 5.8. Ciclo de estado-ação-recompensa-estado. Fonte: [Yang et al. 2019a]

5.5. Aprendizado por Reforço para FL

Aprendizado por reforço (RL) é um ramo do aprendizado de máquina (ML) que lida principalmente com a tomada de decisão sequencial [Sutton et al. 1998]. Um problema de RL geralmente consiste em um ambiente dinâmico e um agente (ou agentes) que interage(m) com o ambiente. O ambiente evolui uma vez que o agente seleciona uma ação com base no estado atual do ambiente, apresentando uma recompensa pela avaliação do desempenho do agente. O agente busca atingir uma meta no ambiente tomando decisões sequenciais. Os problemas tradicionais de RL podem ser formulados como Processos de Decisão de Markov (MDPs). O agente tem que enfrentar um problema de tomada de decisão sequencial para maximizar uma função de valor (ou seja, a soma esperada de recompensas descontadas ou expectativas de recompensa).

Conforme mostrado na Figura 5.8, o agente observa o estado do ambiente e, a seguir, seleciona uma ação com base no estado. O agente deve receber uma recompensa do ambiente com base nesta ação selecionada. Em MDPs, o próximo estado do ambiente depende do último estado e da ação selecionada pelo agente. A ação que resulta na "maior recompensa esperada" geralmente se refere à ação que coloca o agente no estado com o maior potencial para ganhar recompensas no futuro. O agente se move em ciclos de estado-ação-recompensa-estado (SARS).

Existem várias dificuldades baseadas nesse problema, dentre eles: (i) Um agente tem conhecimento limitado sobre as ações ideais para um determinado estado do ambiente. Considerando o processo de tomada de decisão de uma rodada no problema RL com espaço de ação contínuo, o agente tem que lidar com um problema de otimização com espaço contínuo, o que pode exigir um grande esforço de computação. (ii) As ações do agente podem afetar os estados futuros do ambiente, afetando assim as opções e oportunidades disponíveis para o agente no futuro. Portanto, ao lidar com problemas sequenciais de tomada de decisão, cada agente não deve escolher ações avidamente, mesmo que essas ações possam obter boas recompensas em um curto prazo. Ou seja, o agente precisa fazer um trade-off entre a recompensa atual e as expectativas de recompensa futura. (iii) A seleção de ações ideais requer levar em consideração as consequências indiretas e atrasadas das ações e, portanto, pode exigir previsão ou planejamento.

5.5.1. Algoritmos de Aprendizado por reforço

Os algoritmos RL podem ser categorizados de acordo com os seguintes elementos-chave.

Baseado em modelo vs. livre de modelo: Os métodos baseados em modelo tentam construir um modelo virtual do ambiente primeiro e, em seguida, agir de acordo com a

melhor política derivada do modelo virtual. Já os métodos livres de modelo assumem que o modelo do ambiente não pode ser construído e estimam a função de política e valor por tentativa e erro.

Baseado em valor vs. baseado em política: Os métodos baseados em valor tentam aprender uma função de valor e inferir uma política ótima a partir dela. Os métodos baseados em política pesquisam diretamente no espaço dos parâmetros de política para encontrar uma política ótima.

Atualização por Monte Carlo vs. atualização por diferença temporal (TD): A atualização por Monte Carlo avalia uma política usando a recompensa acumulada durante todo o episódio. Isso é direto na implementação, mas eles requerem um grande número de iterações para convergência e sofrem uma grande variação ao estimar sua função de valor. Em vez de usar a recompensa total acumulada, o TD update calcula um erro temporal, que é a diferença entre a nova e a antiga estimativa da função de valor, para atualizar a política. Este tipo de atualização só precisa das iterações mais recentes e reduz a variância. No entanto, o viés aumenta durante a estimativa, pois a visão global de todo o episódio não é considerada.

Política-On vs. política-off: Os métodos política-On usam a política atual para gerar ações e atualizar a própria política atual de acordo. Os métodos política-off usam uma política exploratória diferente para gerar ações e a política de destino é atualizada com base nessas ações. Dois algoritmos TD que têm sido amplamente usados para resolver problemas de RL são Estado-Ação-Recompensa-Estado-Ação (SARSA) e Q-Learning.

SARSA é um algoritmo TD com uma abordagem política-on [Rummery and Niranjan 1994], pois segue a mesma política para encontrar a próxima ação. Ele tenta aprender uma função de valor de ação em vez de uma função de valor. A etapa de avaliação de política usa o erro temporal para a função de valor da ação, que é semelhante à função de valor.

Q-Learning é um algoritmo TD política-off [Watkins and Dayan 1992], pois seleciona a próxima ação de forma gananciosa, em vez de seguir a mesma política. A função Q é atualizada usando uma política que é diretamente gananciosa em relação à função Q atual.

5.5.2. Aprendizado por Reforço Federado

O RL tem muitos aspectos técnicos e não técnicos durante as implementações que dem ser observados. Um das questões mais críticas é como evitar o vazamento de informações e preservar a privacidade do agente. Essa preocupação leva a versões que preservam a privacidade de RL — Federated Reinforcement Learning (FRL). Aqui nós categorizamos as pesquisas FRL em Aprendizado por Reforço Federado Horizontal (HFRL) e Aprendizado por Reforço Federado Vertical (VFRL).

5.5.2.1. Aprendizado por Reforço Federado Horizontal

RL Paralelo [Kretchmar and Jacobvitz 2002, Grounds and Kudenko 2005] tem sido estudado há muito tempo na comunidade de pesquisa em RL, na qual vários agentes são

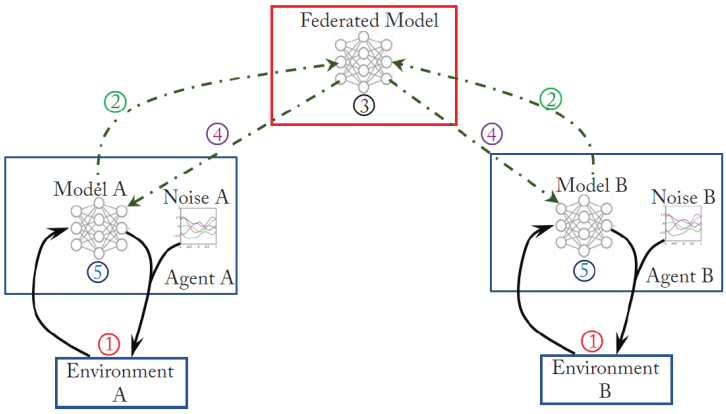


Figura 5.9. Aprendizado por Reforço Federado Horizontal. Fonte: [Yang et al. 2019a]

assumidos para executar a mesma tarefa (com as mesmas recompensas para estados e ações). Os agentes podem realizar o aprendizado em diferentes ambientes. Observe que a maioria das configurações de RL paralelas adotam as operações de transferência de experiência ou gradientes dos agentes. É simples que tais operações não podem ser conduzidas ao considerar questões de preservação de privacidade. Portanto, é natural adotar o HFRL para questões de preservação da privacidade. A comunidade HFRL adota essas configurações básicas em RL paralelo com o objetivo de preservação da privacidade como uma restrição extra (para o servidor e os agentes).

Um framework básico para a condução do HFRL é apresentado na figura 5.9.

Como pode ser visto na Figura 5.9, HFRL contém vários agentes RL paralelos (apresentamos dois agentes por simplicidade) para diferentes sistemas que podem ser geograficamente distribuídos. Os agentes RL têm a mesma tarefa em sistemas distintos. Um servidor federado assume a função de agregar os modelos de diferentes agentes RL. As etapas básicas para conduzir o HFRL são listadas a seguir.

Etapa I. Todos os agentes RL participantes treinam seus próprios modelos RL local e independentemente, sem nenhuma troca de experiência de dados, gradientes de parâmetro e perdas.

Etapa II. Os agentes RL enviam seus parâmetros de modelo mascarados ao servidor.

Etapa III. O servidor federado criptografa os modelos de agentes RL não idênticos e conduz métodos de agregação para obter um modelo federado.

Etapa IV. O servidor federado envia o modelo federado aos agentes RL.

Etapa V. Os agentes RL atualizam o modelo local.

[Nadiger et al. 2019] descreveu completamente a arquitetura geral para HFRL, que contém a política de agrupamento, a política de aprendizagem e a política de federação para os agentes RL participantes. Os autores demonstraram ainda a eficácia da arquitetura proposta com base no jogo Pong do Atari. Nas demonstrações, os autores

mostraram que com a abordagem proposta, há uma melhora mediana de 17 % no tempo de personalização. Embora o objetivo de preservação da privacidade possa apresentar mais desafios, podemos nos beneficiar do HFRL das seguintes maneiras: (i) **Para evitar amostras não independentes e identicamente distribuídas (Non-iid)**: É comum que o agente de tarefa única possa encontrar amostras não-i.i.d. durante o processo de aprendizagem. É claro que um dos principais motivos é que para tarefas RL com uma configuração de agente único, a experiência adquirida posteriormente pode estar fortemente relacionada com a experiência anterior, o que pode quebrar a suposição i.i.d. dos dados. O HFRL pode fornecer benefícios para a construção de um sistema de aprendizado de reforço mais preciso e estável. (ii) **Para aumentar a eficiência da amostra**: Outra desvantagem dos métodos convencionais de RL é a baixa capacidade de construir rapidamente modelos estáveis e precisos com amostras limitadas (conhecido como problema de baixa eficiência de amostra), o que impede que os métodos convencionais de RL sejam aplicados no mundo real. Sob o HFRL, podemos agregar conhecimento extraído por diferentes agentes de ambientes não idênticos para resolver o problema de baixa eficiência da amostra. (iii) **Para acelerar o processo de aprendizagem**: Na verdade, esse benefício pode ser obtido a partir das duas vantagens acima. Combinado com o poderoso framework de FL para agregar diferentes conhecimentos aprendidos por agentes não idênticos, experiência de mais amostras não-i.i.d. podem acelerar o aprendizado de RL e obter melhores resultados.

5.5.2.2. Aprendizado por Reforço Federado Vertical

Da mesma maneira que foi discutido anteriormente o aprendizado federado vertical, em FRL, existe a possibilidade de diversas empresas colaborarem para compartilhar dados de natureza distinta, porém, de ambientes comuns. Esta estrutura cooperativa se enquadra na categorização de VFRL. No VFRL, existem diferentes agentes RL que mantêm observações não idênticas do mesmo ambiente. Cada agente RL mantém uma política de ação correspondente (alguns agentes podem não ter uma política de ação). O objetivo principal da estrutura cooperativa é treinar um agente de RL mais eficaz com o conhecimento misto extraído das observações de diferentes agentes cooperativos. Durante o treinamento ou no processo de inferência, qualquer transformação direta de dados brutos é proibida. A seguir apresentamos uma estrutura possível para VFRL — Federated DQN [Yang et al. 2019a]. Como pode ser observado a seguir, nomeamos o agente RL que obtém a recompensa do ambiente como agente Q-network, e todos os outros agentes como agentes RL cooperativos:

Etapa I. Todos os agentes participantes do aprendizado por reforço realizam ações de acordo com as observações atuais do ambiente e do conhecimento extraído. Observe que alguns agentes podem não fazer nenhuma ação, o que apenas mantém suas próprias observações do ambiente.

Etapa II. Os agentes RL obtêm o feedback correspondente do ambiente, incluindo as observações do ambiente atual, a recompensa, etc.

Etapa III. Os agentes RL calculam os produtos intermediários alimentando as observa-

ções obtidas em sua rede neural e, em seguida, enviam os produtos intermediários mascarados ao agente Q-network.

Etapa IV. O agente Q-network criptografa todos os produtos intermediários e treina a Q-network com as perdas atuais por meio de propagação reversa.

Etapa V. O agente Q-network envia de volta os gradientes de peso mascarados para os agentes cooperativos.

Etapa VI. Cada agente cooperativo criptografa os gradientes e atualiza sua própria rede.

Na literatura, o trabalho existente que se enquadra na categoria VFRL é [Zhuo et al. 2020], que investiga o problema do sistema RL multiagente de forma cooperativa, ao considerar os requisitos de preservação da privacidade dos dados, gradientes e modelos do agente. O framework FRL estudado corresponde à arquitetura VFRL que apresentamos acima (que é denominada VFRL posteriormente). O autor apresentou sistemas detalhados da vida real onde o VFRL é significativo. Modelar e explorar os comportamentos em sistemas com múltiplos agentes cooperativos ou adversários tem sido um desafio interessante para a comunidade RL [Mao et al. 2018] [Foerster et al. 2016]. No domínio da FL, os agentes podem realizar tarefas heterogêneas, com diferentes estados, ações ou recompensas (alguns podem não ter recompensas ou ações). O principal objetivo de cada agente VFRL é construir um modelo RL estável e preciso de forma cooperativa ou competitiva, sem troca direta de experiência (incluindo estados, ações e recompensas) ou os gradientes correspondentes. Comparado com o RL multiagente, as vantagens do VFRL podem ser resumidas como segue: (i) **Para evitar vazamento de informações do agente e do usuário.** Nos sistemas de caldeiras a carvão, um benefício direto apresentado para o departamento de gerenciamento de dados meteorológicos é que ele pode aumentar a eficiência da produção sem qualquer vazamento de dados meteorológicos brutos em tempo real. Isso pode ser lançado como um serviço que pode ser publicado para todos os usuários externos em potencial. (ii) **Para melhorar o desempenho do aprendizado por reforço**. Com os métodos adequados de extração de conhecimento adotados, podemos treinar um agente RL mais razoável e robusto para aumentar a eficiência. O VFRL é vantajoso no sentido de que pode permitir que um sistema de aprendizado aproveite essas informações enquanto preserva a privacidade.

5.6. Computação de borda e Aprendizado Federado

Computação de borda trata do processamento de aplicativos e serviços executados em uma infraestrutura de computação (nós de borda) mais perto das entidades finais (usuários e dispositivos, e genericamente chamados de dispositivos ou nós de borda) e fora de um centro de processamento de dados (*data center*), normalmente na nuvem. Assim, Computação de borda leva a computação e o armazenamento de dados para mais perto de onde são necessários, para melhorar o tempo de resposta e a largura de banda de comunicação.

Este paradigma de computação distribuída é um mecanismo promissor para lidar com as deficiências e limitações da computação na nuvem na assistência a aplicativos e serviços que precisam de eficiência energética e são sensíveis tanto a atrasos quanto ao

contexto em uma ampla gama de cenários, incluindo a Internet das Coisas, computação urbana e mobilidade inteligente. Em vez de usar recursos de computação e armazenamento para executar esses aplicativos e serviços em uma nuvem, computação de borda tem o potencial de prover soluções rápidas, eficientes e inteligentes, que simplesmente não são possíveis em uma solução de nuvem tradicional.

Computação de borda traz vários benefícios para o uso de aprendizado federado. Na segurança, a natureza centralizada da computação em nuvem torna aprendizado federado particularmente vulnerável a ataques de DDoS (*Distributed Denial of Service*), o que não é o caso de computação de borda. No projeto de sistemas confiáveis, computação de borda permite o acesso mais fácil a serviços de aprendizado federado quando comparado a uma solução em nuvem. A classificação e reconstrução de dados pode ser feita de forma mais efetiva por aprendizado federado em uma solução de computação de borda.

5.6.1. Aplicações de Aprendizado Federado em Cenários de Computação de Borda

Dadas as características comuns tanto da computação de borda quanto da aprendizado federado, a computação de borda é um ambiente naturalmente adequado para aplicar aprendizado federado. Assim, o aprendizado federado de borda (*edge federated learning*) é uma área de investigação cada vez mais atraente tanto na academia quanto na indústria.

O aprendizado federado de ponta resolve o problema do isolamento de dados (*data island problem*) ao explorar o enorme potencial de dados existentes em dispositivos terminais "isolados", i.e., que estão na ponta, sem violar a privacidade do usuário. Esse tratamento conjunto permite melhorar a eficiência do aprendizado de modelo em sistemas de computação de borda. Portanto, essa estratégia pode ser amplamente utilizada em cenários nos quais a privacidade e a utilização de recursos são críticas. A seguir, apresentaremos alguns cenários para o aprendizado federado de ponta e alguns trabalhos recentes aplicados nesses cenários.

Saúde: O aprendizado profundo normalmente apresenta um excelente desempenho em tarefas complexas de reconhecimento de padrões, o que faz com que essa técnica seja amplamente utilizada na área da saúde. Suponha o cenário onde há um conjunto de instituições médicas (e.g., hospitais, clínicas, laboratórios) onde em cada uma os seus dados são armazenados e processados separadamente no nó de borda (cenário típico de hoje em dia). Nesse caso, um modelo treinado com dados coletados de uma instituição médica individual dificilmente terá uma precisão satisfatória quando aplicado aos dados "invisíveis" que de alguma forma não são corrigidos com os dados de treinamento. Para termos um modelo médico adequado precisamos de uma grande quantidade de prontuários médicos, o que é difícil de obter devido à privacidade dos dados médicos. O aprendizado federado de borda pode ajudar a superar esse problema, permitindo às instituições médicas colaborarem em modelos de treinamento sem compartilhar dados de pacientes e, assim, atendendo aos requisitos de proteção de privacidade de dados. Além das instituições médicas, o método de transferência de aprendizado federado de borda (edge federated transfer learning) pode ser aplicado a dispositivos de saúde pessoal, ou seja, a algum tipo de *wearable device*. Dispositivos de saúde pessoal, como medidor de pressão arterial e dispositivo de reconhecimento de atividade, são utilizados para observar

as condições de saúde e enviar alarmes em tempo real, sendo uma atividade central em sistemas inteligentes de saúde [Xu et al. 2021]. Para os usuários, é necessário ter um modelo disponível no início e treinar um modelo personalizado que deve ser atualizado periodicamente em função das condições físicas. Neste cenário, modelos de aprendizado federado podem ser usados para aprender de forma colaborativa o comportamento de indivíduos na borda da rede mantendo a personalização dos envolvidos [Sheller et al. 2020].

Redes veiculares: Veículos modernos têm literalmente centenas de sensores que podem gerar um grande volume de dados como localização, orientação, imagens e estado de seus diversos componentes como motor, suspensão e freio. Esses dados são valiosos para os fabricantes de veículos que podem projetar serviços inteligentes de navegação e avisos antecipados, por exemplo. O sistema computacional do veículo coleta dados de sensoriamento gerados localmente e, em seguida, repassa para o sistema de computação de borda veicular (VEC – Vehicle Edge Computing) que treina o modelo de aprendizado local. O aprendizado federado de borda no VEC pode atender às necessidades dos usuários para a tomada de decisão em veículos inteligentes. Por exemplo, à medida que tivermos mais veículos elétricos sendo utilizados, será importante haver uma solução de aprendizagem de demanda de energia federada para fazer a previsão de demanda de energia onde esses veículos elétricos se encontram/planejam ir. No caso de veículos autônomos, teremos mais sensores do que veículos comuns, como LiDAR e sensores ultrassônicos para perceber o ambiente ao redor sem a necessidade da interação humana. Nesse cenário, aprendizado federado de borda é uma solução adequada na computação de borda veicular pois preserva a privacidade de dados veiculares.

Recomendação inteligente: Em aplicativos para smartphones e desktops, a recomendação inteligente (smart intelligence) é uma função útil para prever escolhas de usuários e, assim, oferecer opções. Em comparação com as abordagens de aprendizado de máquina, o aprendizado federado de borda é capaz de treinar modelos flexíveis para tarefas de recomendação. Como os nós de borda estão localizados tendem a ter tarefas semelhantes por razões de eficiência e custo, esse semelhança entre os nós de borda pode ser usada para treinar modelos adaptativos por aprendizado federado de borda. Por exemplo, o Google Keyboard (Gboard) treina modelos usando aprendizado federado de borda em escala global para sugestão de pesquisa do teclado virtual e previsão de emojis. Os resultados da avaliação mostram que os modelos em cada nó de borda têm um bom desempenho já que os modelos são ajustados a diferentes estilos de linguagem e cultura em uma área específica. A mesma estratégia pode ser aplicada a um navegador (*browser*) treinado com aprendizado federado que pode ajudar os usuários a encontrar rapidamente o endereço da página que precisam, inserindo menos caracteres. Essa estratégia pode ser aprimorada em sistemas de aprendizado federado de ponta para fornecer a diferentes usuários modelos relativamente personalizados, explorando as semelhanças do usuário sem violar a sua privacidade.

5.6.2. Segurança e Privacidade

Segurança e privacidade são dois grandes problemas para implementar o aprendizado federado em computação de borda. Devido ao ambiente heterogêneo natural do aprendizado federado e da computação de borda, é sempre difícil prever atividades de outras entidades no sistema. Por exemplo, no processo de treinamento de aprendizado federado de borda, devido ao desconhecimento do papel exercido por outras entidades, algumas delas podem ser maliciosas e atacar o processo de treinamento. Além disso, entidades "curiosas" podem querer aprender/saber/furtar dados pessoais e privados. Neste cenário, a segurança e a privacidade dos dispositivos de borda podem ser prejudicadas.

Questões de segurança surgem na ponta do aprendizado federado devido à heterogeneidade tanto da computação de borda quanto do aprendizado federado. Mais ainda, nesse ambiente heterogêneo, nós de borda podem não ser confiáveis. Por outro lado, no aprendizado federado, geralmente assumimos que todos os nós mantêm seus próprios dados privados e não os compartilham com outras entidades. Esses recursos melhoram a generalidade do aplicativo da computação de borda e mantêm mais privacidade para os dados privados dos usuários, mas também aumentam o risco de sofrer ataques maliciosos. Muito trabalho tem sido feito para resolver questões de segurança no aprendizado federado de ponta. Primeiro introduziremos duas maneiras principais de injetar ataques seguidos por alguns algoritmos existentes para defender ataques.

Atualmente, os métodos de defesa propostos para aprendizado federado em computação de borda assumem que a distribuição de dados é I.I.D. em todos os nós ou a maioria dos nós opera corretamente. No entanto, esses pressupostos não são práticos no aprendizado federado de ponta. Primeiro, geralmente os nós de borda coletam seus dados de suas próprias fontes, de modo que a distribuição de dados não é necessariamente I.I.D. Em segundo lugar, no momento de uma sincronização, o servidor central seleciona aleatoriamente alguns dos nós de borda para executar a computação. Nesse caso, os nós honestos podem não ser a maioria e, consequentemente, pode não ser razoável supor haver uma maioria honesta durante todo o processo de treinamento.

Portanto, é necessário investigarmos algoritmos mais práticos sem assumir a distribuição de dados I.I.D. e maioria honesta para defender o aprendizado federado de borda de ataques. Essa é uma área de pesquisa promissora que precisa de mais estudos.

O aprendizado federado é projetado para proteger os dados privados de treinamento de cada nó sem depender da transmissão dos dados de treinamento entre servidores. No entanto, a violação de privacidade ainda pode ocorrer quando as informações (por exemplo, pesos de modelo) são compartilhadas entre os elementos federados. É possível que alguns nós ou servidores maliciosos possam extrair informações privadas durante o processo de treinamento. Portanto, a fim de atacar a privacidade de alguns dos nós de borda, o processo do adversário é recuperar informações privadas do conjunto de dados de pesos carregados ou pesos agregados [Melis et al. 2019]. Esse problema equivale à recuperação dos dados da atualização de peso. Em geral, existem dois tipos de ataques de privacidade na área de aprendizado federado: inferência de membros e inferência de dados. O ataque de inferência de membros tem como objetivo determinar se um registro de dados está contido no conjunto de dados de treinamento de um nó. Quando o conjunto de dados é sensível, este ataque pode vazar informações úteis. O ataque de inferência

de dados tem como objetivo recuperar dados de treinamento ou uma classe de dados de treinamento das informações que o nó fornece.

O problema de privacidade pode se tornar mais grave na computação de borda, já que os dispositivos de borda podem vazar informações sobre dados, uso e localização para usuários mal-intencionados. Como preservar a privacidade no aprendizado federado de borda considerando as especificidades da computação de borda é uma nova direção de pesquisa que deve ser investigada.

5.7. Desafios e Oportunidades de Pesquisa

Nessa seção, discutiremos alguns problemas de pesquisa que são de grande interesse da comunidade de aprendizado federado e de IoT. Essa seção não tem como objetivo exaurir todas os problemas de pesquisa em aberto na área. Para uma discussão mais completa, recomendamos o texto [Kairouz et al. 2021].

5.7.1. Dados Não Independentes e Identicamente Distribuídos - Non-IID

Em modelos de aprendizado de máquina tradicional, coletamos amostras da população de interesse seguindo os preceitos da amostragem estatística, que nos orientam a termos amostras que represente bem a população em estudo para que possamos realizar inferências confiáveis. Nesse sentido, a técnica mais empregada é a amostragem realizada de maneira aleatória e que tenha uma quantidade de elementos suficiemente grande de modo a representar toda a varabilidade da população. As aplicações de aprendizado de máquina centralizado tradicionais, em geral, seguem esse princípio que busca produzir amostras independentes e identicamente distribuídas (IID). Entrentanto, de acordo com [Zhao et al. 2018], na prática, não é realista assumir que os dados locais em cada dispositivo de borda são sempre IID. [McMahan et al. 2017] demonstrou que FedAvg pode trabalhar com certos dados não-IID. No entanto, a precisão das redes neurais convolucionais (CNN) treinadas com o algoritmo FedAvg pode reduzir significativamente, até 11 % para MNIST, 51 % para CIFAR-10 e 55 %, com alta quantidade de dados não-IID altamente assimétricos (quando cada cliente treina com apenas 1 classe).

Para lidar com o desafio estatístico do aprendizado federado, [Zhao et al. 2018] demonstram o limite na divergência no treinamento pela distância do movimentador de terra (EMD) entre a distribuição por classes em cada dispositivo (ou cliente) e a distribuição da população. Esse limite é afetado pela taxa de aprendizado, etapas de sincronização e gradientes. Em seguida, ele propõe uma estratégia de compartilhamento de dados para melhorar o FedAvg com dados não-IID, distribuindo uma pequena quantidade de dados compartilhados globalmente contendo exemplos de cada classe. Essa estratégia apresenta uma compensação entre precisão e centralização. Os experimentos no artigo mostram que pode aumentar a precisão no CIFAR-10 em 30% se estivermos dispostos a compartilhador 5% dos dados.

Para os experimentos IID, as curvas de convergência de FedAvg com um tamanho de lote B se sobrepõem principalmente às curvas SGD (Gradiente Decendente Estocástico Centralizado) com B $\times$ 10 para todos os três conjuntos de dados analisados pelos autores. Isso acontece porque o modelo global produzido pelo FedAVG representa a média dos 10 clientes avaliados (cada cliente foi treinado com 1 classe). Dessa maneira, o FedAVG para

dados IID deve ser comparável com SGD com um tamanho de lote 10 vezes maior. Assim, o FedAvg, nos experimentos de [Zhao et al. 2018], atinge precisão consistente com os resultados obtidos por [McMahan et al. 2017]. Uma redução significativa na precisão do teste é observada para FedAvg em dados não-IID em comparação com SGD, em tamanhos de lote correspondentes. Além disso, os modelos pré-treinados pelo SGD podem não convergir com o treinamento FedAvg em dados não-IID extremos. Para CIFAR-10, a precisão cai quando o FedAvg treina CNNs pré-treinadas em dados não-IID. Assim, foi demonstrado uma redução na precisão do teste FedAvg para dados não-IID. A precisão demonstrada pelo modelo centralizado utilizado não é o estado da arte, mas as CNNs treinadas foram suficientes para avaliar o aprendizado federado em dados não-IID.

Para [Hsieh et al. 2019], existem algumas maneiras de lidar com dados não-IID. Uma deles seria o *batch normalization* para dados não-IID. O *batch normalization* (*BatchNorm*) é um dos mecanismos de DL mais populares (mais de 20.000 citações em agosto de 2020). *BatchNorm* visa estabilizar uma DNN (Deep Neural Network) normalizando a distribuição de entrada para média zero e variância unitária. Como a média e a variância globais são inatingíveis com o treinamento estocástico, o *BatchNorm* usa a média e a variância de *minibatch* para estimar a média e a variância globais. Especificamente, para cada *minibatch* β, o *BatchNorm* calcula a média do *minibatch* μ_β e a variação θ_β, e então usa μ_β e θ_β para normalizar cada entrada em β. *BatchNorm* permite um treinamento mais rápido e estável porque permite maiores taxas de aprendizado.

Ainda para [Hsieh et al. 2019], como o problema de *BatchNorm* na configuração não-IID é devido à sua dependência de *minibatch*, a solução natural é substituir *BatchNorm* por mecanismos de normalização alternativos que não dependem de *minibatch*. Infelizmente, a maioria dos mecanismos de normalização alternativos existentes, Normalização de Camada e Renormalização em Lote, têm suas desvantagens. Em vez disso, um mecanismo particular pode ser o *Group Normalization* (*GroupNorm*). *GroupNorm* é um mecanismo de normalização alternativo que visa superar a normalização de *BatchNorm* e de camada (*LayerNorm*). *GroupNorm* divide canais adjacentes em grupos de um tamanho pré-especificado g_{size} e calcula a média e variância por grupo para cada amostra de entrada. Consequentemente, *GroupNorm* não depende de *minibatch* para normalização (a deficiência de *BatchNorm*), e *GroupNorm* não assume que todos os canais fazem contribuições iguais (a deficiência de *LayerNorm*). Eles avaliaram *GroupNorm* com BN-LeNet sobre CIFAR-10. Eles selecionaram $g_{size=2}$, que funciona melhor com esta DNN. Primeiro, o *GroupNorm* recupera com sucesso a perda de precisão do *BatchNorm* com BSP na configuração Não-IID. *GroupNorm* com BSP atinge 79,2% de precisão de validação na configuração Não-IID, que é tão boa quanto a precisão da configuração IID. Isso mostra que o *GroupNorm* pode ser usado como uma alternativa ao *BatchNorm* para superar o desafio de dados não-IID para o BSP. Em segundo lugar, o *GroupNorm* ajuda drasticamente os algoritmos de aprendizado descentralizado também na configuração não-IID. Com *GroupNorm*, houve 14,4%, 8,9% e 8,7% de perda de precisão para Gaia, *FederatedAveraging* e *DeepGradientCompression*, respectivamente. Embora as perdas de precisão ainda sejam significativas, elas são melhores do que suas contrapartes *BatchNorm* por um aditivo de 10,7%, 19,8% e 60,2%, respectivamente.

Como uma tentativa de contornar a situação, [Jeong et al. 2018] propôs o *Federated Augmentation* (FAug) para lidar com a sobrecarga do balanceamento de dados. Cada

dispositivo recebe rótulos de dados de outros dados e gera localmente amostras de dados ausentes com base nos rótulos, usando um modelo generativo, e o modelo generativo é treinado no servidor com alto poder de computação e conexão à Internet. Cada dispositivo reconhece os rótulos ausentes nas amostras de dados, conhecidos como rótulos de destino, e carrega algumas sementes de amostras de dados desses rótulos de destino para o servidor. Em seguida, o servidor cria novas amostras das sementes de amostras de dados carregadas usando a pesquisa de imagens do Google para os dados visuais a serem treinados usando uma *Generative Adversarial Network* (GAN). Em seguida, baixa o gerador da GAN treinado que permite que cada dispositivo reabasteça os rótulos de destino até atingir um conjunto de dados de treinamento IID, o que reduz significativamente a sobrecarga de comunicação em comparação com trocas diretas de amostra de dados, além de não copiar diretamente dados dos clientes, quebrando assim, uma prermissa básica do FL.

Para [Duan et al. 2019], os dados de treinamento de cada cliente devem ser reequilibrados para resolver o problema de degradação da precisão. Um método é redistribuir os dados do cliente local até que a distribuição seja uniforme. No entanto, o compartilhamento de dados levanta um problema de privacidade e causa alta sobrecarga de comunicação. Outra forma de reequilibrar o treinamento é atualizar o modelo global de forma assíncrona. Cada cliente calcula as atualizações com base no modelo global mais recente e aplica suas atualizações ao modelo global sequencialmente. Essas atualizações implicam em uma sobrecarga de comunicação e o consumo de tempo do método maiores do que FL convencional. Os autores propuseram um framework chamado Astraea, que combina essas duas ideias, apresentando mediadores entre o servidor FL e os clientes para reequilibrar o treinamento. O fluxo de trabalho do Astraea inclui inicialização, reequilíbrio, treinamento e agregação. Na fase de inicialização, o servidor FL primeiro espera que os dispositivos móveis ingressem na tarefa de treinamento do modelo FL. Os dispositivos participam do treinamento enviando suas informações de distribuição de dados locais para o servidor. Depois de determinar os dispositivos (clientes) envolvidos no treinamento, o servidor FL conta a distribuição global de dados e inicializa os pesos e o otimizador do modelo de aprendizagem. Na fase de reequilíbrio, o servidor primeiro calcula o número de aumentos para cada classe com base na distribuição global. Em seguida, todos os clientes realizam aumento de dados em paralelo de acordo com os resultados do cálculo. A função de aumento de amostra pega uma amostra e gera aumentos, incluindo deslocamento aleatório, rotação aleatória, cisalhamento aleatório e zoom aleatório, para a amostra. O objetivo do aumento de dados é mitigar o desequilíbrio global, em vez de eliminá-lo. Ao mesmo tempo, um aumento significativo no aumento de dados irá gerar muitas amostras semelhantes, tornando o treinamento do modelo mais sujeito a *overfitting*.

Apesar de termos apresentado algumas propostas para endereçar o problema de dados não-IID, essa é uma área que ainda tem muito a ser explorada no contexto de FL [Kairouz et al. 2021].

5.7.2. Busca Automática de Modelos

Algoritmos de DL tem avançado o estado-da-arte em diversas áreas, como visão computacional e processamento de linguagem natural. Dessa forma, naturalmente, estes tornaram-se a primeira opção para as mais diversas atividades. Porém, o design de uma rede neural

profunda que atinja bons resultados é um processo tedioso, complexo e que exige grande esforço humano, em um longo exercício de tentativa e erro, visto que diversos aspectos das redes devem ser considerados, como a topologia, hiper-parâmetros, algoritmos de aprendizado (e seus hiper-parâmetros), etc. Isso levou ao surgimento da Busca de Arquitetura Neural (*Neural Architecture Search* – NAS), ou seja, sistemas semi-automáticos que buscam o melhor modelo de rede neural para a resolução de um problema, com mínima intervenção humana [Zhu et al. 2021, Elsken et al. 2019]. NAS é parte do Aprendizado de Máquina Automático (*Automated Machine Learning* – AutoML).

Para uma certa atividade, há possibilidades infinitas de arquiteturas e modelos que podem ser explorados. Consequentemente, há muitas formas de buscar através de tais possibilidades. De acordo com [Elsken et al. 2019], o procedimento para a construção de métodos de NAS devem considerar os três seguintes aspectos: (i) **Espaço de busca:** visto que redes neurais podem ser compostas dos mais variados elementos, como convolução, *pooling*, funções de ativação, etc., definir o espaço de busca consiste em determinar quais os possíveis componentes que podem ser descobertos durante o processo de busca. Inevitavelmente, essas determinações inserem um viés humano no processo, o que pode impedir o método de encontrar arquiteturas eficientes que vão além do conhecimento humano. Além disso, um espaço de busca muito grande pode dificultar a otimização pela alta dimensionalidade. Dessa forma, é necessário encontrar um balanço entre essas duas características, o que é um desafio que ainda tem muito a ser explorado. (ii) **Estratégia de busca:** o algoritmo que guia a exploração do espaço de busca, isto é, o que determina a próxima arquitetura que será explorada. É possível utilizar um algoritmo aleatório, o que pode não ser uma boa escolha, devido as enormes possibilidades de arquiteturas dentro do espaço de busca. Dessa forma, tipicamente se usa algoritmos que consideram o desempenho de arquiteturas já descobertas, como Otimização Bayesiana, Algoritmos Evolucionários, Aprendizado por Reforço e Métodos baseados no Gradiente. Note que esse processo consome tempo e poder computacional. Dessa forma, é essencial buscar alternativas que diminuam o custo dessa atividade. (iii) **Estratégias para estimativa de desempenho:** determina como medir o desempenho da arquitetura candidata em dados não vistos, para guiar a estratégia de busca a encontrar as melhores arquiteturas. A forma mais simples é treinar as arquiteturas encontradas e medir sua acurácia nas bases de dados de teste; mas isso pode ser muito custoso, podendo levar muitos dias para finalizar. Assim, outras estratégias estão sendo exploradas para aceleração da estimação de desempenho, como *Weight Sharing*, *One-shot Models*, entre outros. Essa aceleração é uma questão importante nesse contexto e que permanece como um desafio em aberto.

Pode-se notar que NAS é um grande campo de pesquisa, o qual ainda é pouco explorado, mas que apresenta grande potencial, especialmente em ambientes federados. [Zhu et al. 2021] apresenta uma introdução ao NAS federado, com os diversos estudos feitos na área recentemente.

A grande vantagem apresentada por essa abordagem é a eliminação do esforço humano da busca da arquitetura, o que pode ajudar a superar diversas desvantagens, como a computação desnecessária, acelerando o processo de treino e inferência das redes neurais. Isso é especialmente interessante no contexto de IoT federado, onde os aparelhos geralmente possuem pouca capacidade de processamento. Dessa forma, é possível utilizar métodos de NAS para encontrar arquiteturas acuradas e pequenas o suficiente

para serem implementadas nesses dispositivos da borda das redes. Além disso, NAS pode ajudar na busca de arquiteturas mais adequadas para dados com distribuição não-IID [Kairouz et al. 2021].

Adicionalmente, os estudos atuais de NAS são basicamente focados em aprendizado federado horizontal. Como visto, o aprendizado federado vertical é totalmente diferente, o que impede o uso das mesmas técnicas de NAS em ambos os cenários, tornando o aprendizado federado vertical praticamente inexplorado [Zhu et al. 2021].

Há também outros desafios de aprendizado federado que podem ser transportados para o uso de NAS federado, como a necessidade de técnicas *lightweight* de criptografia e tornar o modelo robusto a ataques através de sistemas adversários [Zhu et al. 2021].

5.7.3. Preservação de Privacidade

Apesar do aprendizado federado promover por definição uma proposta com maior enfoque na privacidade dos dados removendo a importância e necessidade de um servidor central, esta vem apresentando uma série de novos desafios e vulnerabilidades. Por exemplo, em uma abordagem federada cliente-servidor, um adversário mal-intencionado ao controlar um servidor pode invalidar os métodos de segurança de um sistema ao comprometer muito mais participantes em uma rodada do que o esperado. Logo, um problema natural de privacidade trata-se de limitar a capacidade do servidor em reconstruir os dados de um cliente com base em suas entradas. Isso envolve definir muito bem as respostas para as seguintes questões: (i) Quais informações dos clientes o servidor tem acesso como resultado de uma rodada no ambiente federado? (ii) Qual o nível de vazamento de privacidade obtido quando o servidor tem acesso a visualização desses dados?

Embora já existam diversas soluções (principalmente envolvendo criptografia), um dos problemas promissores no modelo federado ainda consiste em desenvolver técnicas que garantam a privacidade dos clientes durante o processo de agregação dos resultados individuais pelo servidor [Kairouz et al. 2021].

Devido ao surgimento recente da proposta federada, uma série de questões se encontram em aberto e podem oferecer futuras vias de pesquisa [Mothukuri et al. 2021], como:

Ataques adversários de dia zero - A grande maioria dos métodos federados foram projetados visando garantir a proteção contra ataques pré-definidos. Como ainda não existem muitos modelos federados em produção (quando comparamos com modelos centralizados), tal abordagem torna o modelo frágil a ataques ainda não documentados. Propostas com aprendizado profundo vêm mostrando resultados promissores quanto a essa vulnerabilidade [Saharkhizan et al. 2020, Karimipour et al. 2019].

Processos bem definidos - Ainda não existe uma padronização em relação aos processos usados em FL. Logo, precisam ser feitas pesquisas adicionais com o objetivo de realizar uma padronização das técnicas e abordagens de privacidade existentes.

Trade-off entre privacidade e eficiência - Ao garantir uma maior proteção dos dados, modelos federados atuais apresentam uma perda da sua precisão. Logo, necessitamos de estudos voltados à compreensão do nível adequado de criptografia aplicado aos dados para garantir a segurança dos clientes e manter a eficiência do modelo.

Criação de frameworks de privacidade em FL - Poucas bibliotecas ou frameworks atualmente permitem a implantação de técnicas de proteção à privacidade. Além de facilitar o estudo e desenvolvimento de novas propostas, a implementação de novos pacotes podem facilitar também a adoção de métodos federados na indústria.

Seleção de clientes - Embora existam trabalhos apresentando estratégias seguras de seleção de clientes para as rodadas de treinamento do modelo [Nishio and Yonetani 2019], a proposta de abordagens padronizadas ainda é um tema de extrema importância no aprendizado federado.

5.8. Aplicações

Como já foi explicado nas seções anteriores, FL é um paradigma de modelagem inovador que permite construir modelos compartilhados e personalizados utilizando dados espalhados em vários dispositivos e organizações, sem comprometer a privacidade e a segurança dos usuários. O FL também possibilita uma forma mais eficiente de se lidar com os dados heterogêneos em relação a abordagem tradicional de ML. Essa característica se amplifica com a utilização do FL com a técnica de Transfer Learning que permite utilizar modelos já pré-treinados.

Essas características possibilitam que FL seja aplicado em diversas áreas como Saúde, Cidades Inteligentes (*Smart Cities*), Financeira [Yang et al. 2019a] e em serviços para *smartphones* [Li et al. 2020c]. Nesta seção, serão exemplificadas algumas dessas aplicações por meio da apresentação de trabalhos relacionados.

5.8.1. Cidades Inteligentes (*Smart Cities*)

Internet of Vehicles (IoV) compreende uma rede de veículos que, além de possuírem inúmeros sensores e softwares para obtenção, armazenamento e análise de dados, estão no processo de se tornarem autônomos e interconectados, trocando informações entre si a fim de otimizar a sua função de transportar passageiros com o máximo de segurança, conforto, e mínimo impacto ambiental [Gerla et al. 2014]. Naturalmente, há vários desafios a serem solucionados durante esse processo de evolução, dentre eles a necessidade de comunicação com baixa latência.

Com isso e mente, [Samarakoon et al. 2018] propõem a utilização de princípios de FL para possibilitar uma comunicação entre veículos que seja extremamente confiável e de baixa latência. Isso é possível porque, ao contrário do caso de uma abordagem centralizada, os *vehicular users (VUEs)* não precisam compartilhar informações com *roadside units (RSUs)* e outros VUEs para que os modelos de ML sejam treinados, mas o VUE constroi e envia seu próprio modelo para a RSU, que então agrega todos os modelos, realiza o ajuste dos pesos e envia o modelo atualizado de volta aos VUEs. Adicionalmente, a aplicação de FL também oferece a vantagem de não depender da sincronização entre VUE. Ou seja, mesmo no caso de perda de conexão entre VUEs e RSUs, VUEs ainda conseguem construir seus próprios modelos e continuar navegando.

5.8.2. Financeiro

Como já foi mencionado anteriormente, um dos grandes benefícios da aplicação de FL é a confidencialidade e segurança de dados privados durante o treinamento dos modelos de

ML, e um cenário em que esses fatores são de extrema importância é no setor financeiro. Não apenas pessoas não devem ter acesso aos dados privados de outras pessoas, mas os bancos também não podem trocar entre si dados referentes aos seus usuários.

[Yang et al. 2019c] apontam que há muito mais registros de transações legítimas do que fraudulentas nos *databases* dos bancos, o que torna difícil treinar modelos de ML para deteção de fraudes de cartão de crédito quando um único banco não tem dados suficientes, e ele não pode utilizar dados de outros bancos. Por isso, [Yang et al. 2019c] apresentam a criação do *framework* FFD (*Federated learning Fraud Detection*), com o objetivo de possibilitar que os bancos possam todos se beneficiarem de um treino coletivo do modelo de detecção de fraudes de cartão de crédito, sem o comprometimento da segurança dos dados.

5.8.3. Saúde

Com os avanços da tecnologia, mais hospitais estão adotando os sistemas de dados médicos eletrônicos, o que aumenta o armazenamento de dados históricos dos pacientes e de laudos médicos. Em paralelo a isso, os *smartwatches* conseguem coletar informações pessoais de saúde como as medições dos batimentos cardíacos e do nível de oxigenação do sangue. Utilizando esses dados, é possível alimentar algoritmos de aprendizado de máquina que podem ajudar no diagnóstico precoce de algumas doenças. No entanto, esses dados estão segmentados em diversos dispositivos e hospitais além de serem de natureza sensível por estarem acompanhados de uma forte política de proteção à privacidade do usuário [Xu et al. 2021].

O FL seria uma alternativa para lidar com esses dados segmentados e sensíveis uma vez que é possível treinar um modelo global compartilhado mantendo os dados localmente [Xu et al. 2021] de modo a facilitar a aplicação de modelos de aprendizado de máquina. No trabalho [Lee et al. 2018], por exemplo, o FL é utilizado para encontrar similaridade entre pacientes de hospitais diferentes mantendo as informações pessoais dos pacientes restritas à sua instituição de origem. Outro exemplo de trabalho é o [Brisimi et al. 2018] que constrói um sistema de FL para prever futuras hospitalizações de pacientes com doenças cardíacas.

5.8.4. *Smartphones*

Os dados fornecidos por usuários de *Smartphones* podem ser utilizados para treinar modelos que visam melhorar alguns serviços importantes e cotidianos ofertados em aplicativos celulares como predição de próximas palavras, detecção de rosto e detecção de voz. No entanto, existe um empecilho na utilização desses dados, pois muitos usuários não querem compartilhar os seus dados por questões de privacidade, e para evitar gastos de bateria e de largura de banda. Tendo em vista esse problema, o FL poderia ser uma ótima solução para que esses dados possam ser utilizados sem comprometer a privacidade do cliente e evitando problemas como o gasto de bateria e de banda larga [Li et al. 2020c].

Um dos exemplos mais conhecidos da utilização de FL em aplicações celulares é o trabalho [Yang et al. 2018] que destaca o FL como solução para melhorar os serviços ofertados pelo *Google Keyboard*(GBoard) que é um teclado para dispositivos móveis que possui algumas automatizações como auto-correção de textos, predição das próximas

palavras, completar palavras e expressões que podem representar um *emoji*. Nesse trabalho é destacado que o FL permite que os modelos de *machine learning* sejam treinados sem que seja preciso coletar dados sensíveis de usuários ao mesmo tempo que diminui a latência de resposta.

5.9. Estudo de caso - Hands on

Para colocar em prática os conhecimentos adquiridos ao longo dos capítulos anteriores, finalizaremos com uma aplicação em reconhecimento de atividades humanas (Human Activity Recognition). Usaremos dados obtidos por meio de sensores presentes em dispositivos móveis que encontram disponíveis no Github[10]. Embora o objetivo inicial do projeto que originou o conjunto de dados tenha sido apenas a avaliação de atividades de classificação e reconhecimento de atividades com o uso de redes neurais, queremos aqui utilizar esse background para mostrar a implementação de importantes conceitos de FL em um sistema.

5.9.1. Dataset e Modelo de Aprendizado

O conjunto de dados apresenta cerca de 20.000 leituras de sensores, obtidos após a coleta realizada com 6 participantes realizando 5 ações diferentes, onde cada leitura consiste em: (i) Medições de pose (roll, pitch, yaw), (ii) Dados provenientes do acelerômetro (medições de aceleração linear), (iii) Dados provenientes do giroscópio (medições da velocidade de rotação).

Cada feature será então representada por um vetor 3D apontando na direção da leitura em um determinado passo de tempo para os quatro diferentes sensores (cinto, braço, antebraço, haltere). Assim, normalizamos cada leitura em relação às amostras na mesma categoria no conjunto de dados e concatenadas em um único intervalo de tempo, formando um vetor de features de dimensão 40×1.

Ao longo do tutorial optamos pelo uso do framework de aprendizado federado Flower em conjunto com a biblioteca de deep learning Pytorch. Todo o código fonte e dados utilizados se encontram disponíveis em nosso repositório presente no Github[11]. Para começar a nossa análise, o primeiro passo consiste em importar todas as bibliotecas necessárias para a execução:

```
1 import torch
2 import torch.nn as nn
3 import flwr as fl
4 from torchvision import datasets, transforms
5 from torch.utils.data import DataLoader, Dataset
6 from torchvision import datasets, transforms
```

Logo após, precisaremos de uma função auxiliar para carregar os dados e definir os conjuntos de treinamento e teste. Os dados serão particionados horizontalmente, assim os subconjuntos de treinamento e teste irão ser divididos em mini-batches (pequenos lotes) com base no número total de clientes. Nessa função, precisaremos dos seguintes parâmetros: (i) **data_root (str)**: Diretório onde os datasets finais serão armazenados. (ii) **train_batch_size (int)**: Tamanho do mini-batch usado nos dados de treinamento.

[10] https://github.com/jchiang2/Human-Activity-Recognition
[11] https://github.com/EduardaChagas/Aprendizado-Federado-aplicado-IOT

(iii) **test_batch_size (int)**: Tamanho do mini-batch usado nos dados de teste. (iv) **id (int)**: Client ID usado para selecionar uma partição específica. (v) **nb_clients (int)**: Número total de clientes usados no treinamento.

No fim, ela retornará uma tupla contendo DataLoaders para os conjuntos de treinamento e teste. Usamos como padrão a taxa de divisão de 0.2 para dados de treinamento e validação (80% de treinamento, 20% de validação).

```
def load_data(
    data_root: str,
    train_batch_size: int,
    test_batch_size: int,
    cid: int,
    nb_clients: int,
    ) -> Tuple[DataLoader, DataLoader]:
    train_dataset = Dataset([e for e in range(19622)], data_root)
    test_dataset = Dataset([e for e in range(19622)], data_root)
    # Particionando os dados de treinamento com base no número de clientes
    train_loader = dataset_partitioner(
        dataset = train_dataset,
        batch_size = train_batch_size,
        client_id = cid,
        number_of_clients = nb_clients
    )
    # Particionando os dados de teste com base no número de clientes
    test_loader = dataset_partitioner(
        dataset = test_dataset,
        batch_size = test_batch_size,
        client_id = cid,
        number_of_clients = nb_clients,
    )
    return (train_loader, test_loader)
```

Atualmente o modelo de classificação mais adequado e vantajoso para a modelagem de um ambiente federado são as redes neurais. Desse modo, em nosso exemplo o modelo de aprendizado utilizado será uma CNN 1D com três camadas convolucionais e duas camadas totalmente conectadas, onde cada camada é seguida por uma função de ativação ReLU e uma camada de dropout. A taxa de aprendizado aplicada aqui será de 0.003 e uma taxa de decaimento de 0.95 por época. Definimos essa configuração de arquitetura por meio da criação de uma classe em Pytorch denominada **HARmodel**:

```
class HARmodel(nn.Module):
    """Modelo para reconhecimento de atividades humanas."""
    def __init__(self, input_size, num_classes):
        super().__init__()
        # Camada 1D convolucional
        self.features = nn.Sequential(
            nn.Conv1d(input_size, 64, 1),
            nn.ReLU(),
            nn.Dropout(),
            nn.Conv1d(64, 64, 1),
            nn.ReLU(),
            nn.Dropout(),
            nn.Conv1d(64, 64, 1),
            nn.ReLU(),
            nn.Flatten(),
            )
        # Camadas totalmente conectadas
        self.classifier = nn.Sequential(
            nn.Dropout(),
            nn.Linear(64, 128),
            nn.ReLU(),
            nn.Dropout(),
            nn.Linear(128, num_classes),
            )
    def forward(self, x):
        x = self.features(x)
        out = self.classifier(x)
        return out
```

Até o momento mostramos como criar uma aplicação padrão de Deep Learning, nas próximas subseções iremos abordar como realizar a construção de um ambiente federado ao implementar o algoritmo de média federada. Para isso, precisaremos nos preocupar com dois scripts que funcionaram paralelamente: um responsável pela interação do

cliente com o sistema federado e outro que definirá as regras do servidor central.

5.9.2. Cliente Flower

O próximo passo é definir a alocação dos dispositivos no ambiente federado. Quando o servidor seleciona um dispositivo específico do ambiente federado para realizar um treinamento, ele envia as instruções pela rede, por meio de uma interface chamada Client. Assim, o cliente recebe as instruções do servidor e chama um dos métodos desta classe para executar seu código (ou seja, para treinar a sua rede neural local). O framework Flower fornece uma classe chamada NumPyClient, que torna mais fácil implementar a interface do cliente quando utilizamos PyTorch. Quando implementamos um NumPyClient devemos definir os seguintes métodos: (i) **get_parameters**: retorna o peso do modelo como uma lista de ndarrays (ii) **set_parameters (opcional)**: atualiza os pesos do modelo local com os parâmetros recebidos do servidor (iii) **fit**: define os pesos do modelo local, treina o modelo localmente e recebe o update dos pesos locais (iv) **evaluate**: define como o modelo local será testado. Abaixo mostramos como a classe Client foi implementada para o exemplo apresentado:

```
class HARClient(fl.client.Client):
    def __init__(
        self,
        cid: int,
        train_loader: datasets,
        test_loader: datasets,
        epochs: int,
        device: torch.device = torch.device("cpu"),
    ) -> None:
        self.model = HARmodel(40, 5).to(device)
        self.cid = cid
        self.train_loader = train_loader
        self.test_loader = test_loader
        self.device = device
        self.epochs = epochs

    #Obtendo os pesos do modelo como uma lista de ndarrays NumPy
    def get_weights(self) -> fl.common.Weights:
        return [val.cpu().numpy() for _, val in self.model.state_dict().items()]

    #Configurando os pesos do modelo como uma lista de ndarrays NumPy
    def set_weights(self, weights: fl.common.Weights) -> None:
        state_dict = OrderedDict(
            {
                k: torch.Tensor(v)
                for k, v in zip(self.model.state_dict().keys(), weights)
            }
        )
        self.model.load_state_dict(state_dict, strict=True)

    #Encapsulando os pesos em parâmetros flowers
    def get_parameters(self) -> fl.common.ParametersRes:
        weights: fl.common.Weights = self.get_weights()
        parameters = fl.common.weights_to_parameters(weights)
        return fl.common.ParametersRes(parameters=parameters)

    #Treinando o modelo com o dataset local
    def fit(self, ins: fl.common.FitIns) -> fl.common.FitRes:

        # Definindo a semente para que possamos gerar os mesmos índices de batches para todos os clientes
        np.random.seed(123)
        weights: fl.common.Weights = fl.common.parameters_to_weights(ins.parameters)
        fit_begin = timeit.default_timer()

        # Configurando os pesos do modelo
        self.set_weights(weights)

        # Treinando o modelo
        num_examples_train: int = train(
            self.model, self.train_loader, epochs=self.epochs, device=self.device, cid=self.cid
        )

        # Retornando os pesos e o número de exemplos usados para treinamento
        weights_prime: fl.common.Weights = self.get_weights()
        params_prime = fl.common.weights_to_parameters(weights_prime)
        fit_duration = timeit.default_timer() - fit_begin
        return fl.common.FitRes(
            parameters = params_prime,
            num_examples = num_examples_train,
```

```
            num_examples_ceil = num_examples_train,
            fit_duration = fit_duration,
        )

    #Função de avaliação do modelo
    def evaluate(self, ins: fl.common.EvaluateIns) -> fl.common.EvaluateRes:

        weights = fl.common.parameters_to_weights(ins.parameters)

        # Usando os pesos fornecidos para atualizar o modelo local
        self.set_weights(weights)
        (
            num_examples_test,
            test_loss,
            accuracy,
        ) = test(self.model, self.test_loader, device=self.device)

        # Retornando o número de exemplos de avaliação e o resultado da avaliação (loss)
        return fl.common.EvaluateRes(
            num_examples=num_examples_test,
            loss = float(test_loss),
            accuracy = float(accuracy),
        )
```

Uma vez definidos os métodos e as classes, precisaremos apenas instanciar o cliente e inicializá-lo. O flower nos fornece a possibilidade de rodar o servidor e o cliente na mesma máquina, configurando o endereço do servidor como "[::]: 8080". Porém, se quisermos implementar uma aplicação realmente federada com o servidor e clientes em execução em diferentes máquinas, precisaremos apenas alterar o server_address para o respectivo endereço da máquina do cliente.

```
#id do atual cliente
cid = 0

#número de épocas de treinamento
epochs = 14

#Definindo a alocação de dispositivos no pytorch
DEVICE = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")

#Instanciando o cliente
client = HARClient(
        cid = cid,
        train_loader = train_loader,
        test_loader = test_loader,
        epochs = epochs,
        device = device,
)

# Endereço do servidor que iremos nos conectar
# "[::]: 8080" - define que podemos rodar o servidor e o cliente na mesma máquina usamos
server_address = "[::]:8080"

# Inicializando o cliente
fl.client.start_client(server_address, client)
```

5.9.3. Servidor Flower

Flower permite que o usuário personalize o processo de aprendizagem de acordo com a sua aplicação através da abstração chamada **Strategy**. Existem três maneiras de personalizar a maneira como Flower orquestra o processo de aprendizagem no servidor: (i) Usando uma estratégia já existente, como por exemplo o FedAvg; (ii) Personalizando uma estratégia já existente; (iii) Implementando uma nova estratégia do zero. O Flower contém uma série de estratégias de aprendizado federado já integradas, logo optamos pelo uso de uma delas, o algoritmo de média federada, como orquestrador de atualização do modelo. Uma vez escolhido como será realizado o processo de orquestramento de aprendizado, também podemos definir os seguintes parâmetros: (i) Percentual de clientes que deverão estar disponíveis para o treinamento e avaliação do modelo, assim como também o número mínimo de clientes necessários. (ii) Podemos definir uma função própria de avaliação do modelo (iii) Realizar a configuração dos parâmetros iniciais do modelo global (iv) Definir o número total mínimo de clientes que deverão estar no sistema (v) Definir

manualmente como deverá ocorrer o processo de treinamento e avaliação

Aqui apenas definiremos manualmente a função de avaliação do nosso modelo. Uma vez que estamos seguindo uma proposta cliente-servidor, é de extrema importância saber como se comporta a acurácia do servidor central e não apenas as métricas individuais dos clientes. Como o framework flower não possui por padrão a implementação de métricas importantes para entender a convergência do servidor, implementamos manualmente.

```
# Implementação da rotina de teste do servidor
def test(
    model: torch.nn.Module,
    test_loader: torch.utils.data.DataLoader,
    device: torch.device = torch.device("cpu"),
    ) -> Tuple[int, float, float]:

    model.eval()
    test_loss: float = 0
    correct: int = 0
    num_test_samples: int = 0

    with torch.no_grad():
        for data, target in test_loader:
            data, target = data.to(device), target.to(device)
            num_test_samples += len(data)
            output = model(data.unsqueeze(1).permute(0, 2, 1))

                # Realizando a soma dos loss
                # Definindo a entropia cruzada como função de loss
            test_loss += torch.nn.CrossEntropyLoss()(
                output, target).item()

            # Pegando o índex com máxima log-probabilidade
            pred = output.argmax(
                dim = 1, keepdim = True
            )
            correct += pred.eq(target.view_as(pred)).sum().item()

    test_loss /= num_test_samples
    return (test_loss, correct / num_test_samples, num_test_samples)

def eval(w):
    train_loader, test_loader = load_data(
        data_root = DATA_ROOT,
        train_batch_size = 64,
        test_batch_size = 4,
        cid = 5,
        nb_clients = 6,
    )

    device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

    server = HARClient(
        cid = 999,
        train_loader = train_loader,
        test_loader = test_loader,
        epochs = 1,
        device = device
    )

    server.set_weights(w)
    return test(server.model, train_loader, device)
```

Definida a função de avaliação, agora precisaremos apenas configurá-la na estratégia de aprendizado:

```
strategy = fl.server.strategy.FedAvg(
            eval_fn = eval)

fl.server.start_server(config={"num_rounds": 10}, strategy = strategy)
```

Com os resultados obtidos pelo modelo federado proposto observamos que em pouquíssimas rodadas conseguimos alcançar uma acurácia similar ao modelo centralizado (neste experimento, apenas em 10 épocas). No entanto, com o uso do modelo federado conseguimos construir de um sistema mais seguro e que forneça uma garantia maior quanto a privacidade do usuário.

5.10. Considerações Finais

Este minicurso introduziu o conceito de Aprendizado Federado no contexto de Internet das Coisas. Inicialmente apresentamos uma breve introdução as conceitos de aprendizado de máquina e aprendizado de máquina distribuído, no intuito de diferenciar o Aprendizado Federado deste último. Em seguida, apresentamos os conceitos fundamentais de Aprendizado Federado e introduzimos os modelos mais comumente encontrados na literatura. Introduzimos também, as técnicas e conceitos relativos à preservação de privacidade no contexto de aprendizado federado. No contexto de IoT, entendemos que o modelo geral empregado por Aprendizado Federado, não necessariamente atende às restrições de hardware impostas por dispositivos IoT. Dessa maneira, discutimos como podemos comprimir modelos para que eles sejam operacionais em dispositivos com restrições de recursos. Apresentamos também aspectos relacionados à aprendizado por reforço e computação de borda no contexto de aprendizado federado. Finalmente, apresentamos os principais desafios de pesquisa, aplicações habilitadoras da tecnologia e introduzimos um estudo de caso de uma aplicação prática de Aprendizado Federado.

Entendemos que o Aprendizado Federado é um campo amplo e multidisciplinar, que envolve fundamentos dos algoritmos de aprendizado máquina, aprendizado distribuído, criptografia, segurança, miniração de dados com preservação de privacidade, além de disciplinas que não foram discutidas nesse texto, mas que são bastante relevantes para a habilitação da tecnologia, como questões regulatórias e legais e mecanismos de incentivo para criação bem sucedida da federação. Aprendizado Federado é um área nova de pesquisa que apresenta um campo fértil para o desenvolvimento de estudos e para a criação de novas aplicações que irão beneficiar a sociedade.

Referências

[Al-Rubaie and Chang 2016] Al-Rubaie, M. and Chang, J. M. (2016). Reconstruction attacks against mobile-based continuous authentication systems in the cloud. *IEEE Transactions on Information Forensics and Security*, 11(12):2648–2663.

[Aono et al. 2016] Aono, Y., Hayashi, T., Trieu Phong, L., and Wang, L. (2016). Scalable and secure logistic regression via homomorphic encryption. In *Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy*, pages 142–144.

[Blalock et al. 2020] Blalock, D., Ortiz, J. J. G., Frankle, J., and Guttag, J. (2020). What is the state of neural network pruning? *arXiv preprint arXiv:2003.03033*.

[Brisimi et al. 2018] Brisimi, T. S., Chen, R., Mela, T., Olshevsky, A., Paschalidis, I. C., and Shi, W. (2018). Federated learning of predictive models from federated electronic health records. *International journal of medical informatics*, 112:59–67.

[Cheng et al. 2017] Cheng, Y., Wang, D., Zhou, P., and Zhang, T. (2017). A survey of model compression and acceleration for deep neural networks. *arXiv preprint arXiv:1710.09282*.

[Duan et al. 2019] Duan, M., Liu, D., Chen, X., Tan, Y., Ren, J., Qiao, L., and Liang, L. (2019). Astraea: Self-balancing federated learning for improving classification ac-

curacy of mobile deep learning applications. *Proceedings - 2019 IEEE International Conference on Computer Design, ICCD 2019*, (Iccd):246–254.

[Elsken et al. 2019] Elsken, T., Metzen, J. H., Hutter, F., et al. (2019). Neural architecture search: A survey. *J. Mach. Learn. Res.*, 20(55):1–21.

[Foerster et al. 2016] Foerster, J., Assael, I. A., de Freitas, N., and Whiteson, S. (2016). Learning to communicate with deep multi-agent reinforcement learning. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc.

[Frankle and Carbin 2019] Frankle, J. and Carbin, M. (2019). The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*.

[Fredrikson et al. 2015] Fredrikson, M., Jha, S., and Ristenpart, T. (2015). Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1322–1333.

[Gerla et al. 2014] Gerla, M., Lee, E.-K., Pau, G., and Lee, U. (2014). Internet of vehicles: From intelligent grid to autonomous cars and vehicular clouds. In *2014 IEEE world forum on internet of things (WF-IoT)*, pages 241–246. IEEE.

[Grounds and Kudenko 2005] Grounds, M. and Kudenko, D. (2005). Combining reinforcement learning with symbolic planning. In *Adaptive Agents and Multi-Agent Systems III. Adaptation and Multi-Agent Learning*, pages 75–86. Springer.

[Guo 2018] Guo, Y. (2018). A survey on methods and theories of quantized neural networks. *arXiv preprint arXiv:1808.04752*.

[Han et al. 2015] Han, S., Pool, J., Tran, J., and Dally, W. J. (2015). Learning both weights and connections for efficient neural networks. *arXiv preprint arXiv:1506.02626*.

[He et al. 2016] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

[Hollemans 2017] Hollemans, M. (2017). Google's mobilenets on the iphone. [Online; posted 14-June-2017].

[Howard et al. 2017] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.

[Hsieh et al. 2019] Hsieh, K., Phanishayee, A., Mutlu, O., and Gibbons, P. B. (2019). The non-iid data quagmire of decentralized machine learning. *Arxiv preprint arXiv:1910.00189*.

[Iandola et al. 2016] Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J., and Keutzer, K. (2016). SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size. *arXiv preprint arXiv:1602.07360*.

[Jeong et al. 2018] Jeong, E., Oh, S., Kim, H., Park, J., Bennis, M., and Kim, S.-L. (2018). Communication-efficient on-device machine learning: Federated distillation and augmentation under non-iid private data. *Arxiv preprint arXiv:1811.11479*.

[Ji et al. 2019] Ji, S., Pan, S., Long, G., Li, X., Jiang, J., and Huang, Z. (2019). Learning private neural language modeling with attentive aggregation. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE.

[Jiang et al. 2019] Jiang, Y., Wang, S., Valls, V., Ko, B. J., Lee, W.-H., Leung, K. K., and Tassiulas, L. (2019). Model pruning enables efficient federated learning on edge devices. *arXiv preprint arXiv:1909.12326*.

[Kairouz et al. 2021] Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., D'Oliveira, R. G. L., Eichner, H., Rouayheb, S. E., Evans, D., Gardner, J., Garrett, Z., Gascón, A., Ghazi, B., Gibbons, P. B., Gruteser, M., Harchaoui, Z., He, C., He, L., Huo, Z., Hutchinson, B., Hsu, J., Jaggi, M., Javidi, T., Joshi, G., Khodak, M., Konecný, J., Korolova, A., Koushanfar, F., Koyejo, S., Lepoint, T., Liu, Y., Mittal, P., Mohri, M., Nock, R., Özgür, A., Pagh, R., Qi, H., Ramage, D., Raskar, R., Raykova, M., Song, D., Song, W., Stich, S. U., Sun, Z., Suresh, A. T., Tramèr, F., Vepakomma, P., Wang, J., Xiong, L., Xu, Z., Yang, Q., Yu, F. X., Yu, H., and Zhao, S. (2021). Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1–2):1–210.

[Karimipour et al. 2019] Karimipour, H., Dehghantanha, A., Parizi, R. M., Choo, K.-K. R., and Leung, H. (2019). A deep and scalable unsupervised machine learning system for cyber-attack detection in large-scale smart grids. *IEEE Access*, 7:80778–80788.

[Kim et al. 2015] Kim, Y.-D., Park, E., Yoo, S., Choi, T., Yang, L., and Shin, D. (2015). Compression of deep convolutional neural networks for fast and low power mobile applications. *arXiv preprint arXiv:1511.06530*.

[Konečnỳ et al. 2016] Konečnỳ, J., McMahan, H. B., Yu, F. X., Richtárik, P., Suresh, A. T., and Bacon, D. (2016). Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*.

[Kretchmar and Jacobvitz 2002] Kretchmar, M. D. and Jacobvitz, D. B. (2002). Observing mother-child relationships across generations: Boundary patterns, attachment, and the transmission of caregiving. *Family process*, 41(3):351–374.

[Krizhevsky et al. 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105.

[Lee et al. 2018] Lee, J., Sun, J., Wang, F., Wang, S., Jun, C.-H., and Jiang, X. (2018). Privacy-preserving patient similarity learning in a federated environment: development and analysis. *JMIR medical informatics*, 6(2):e20.

[Li et al. 2020a] Li, A., Sun, J., Wang, B., Duan, L., Li, S., Chen, Y., and Li, H. (2020a). Lotteryfl: Personalized and communication-efficient federated learning with lottery ticket hypothesis on non-iid datasets. *arXiv preprint arXiv:2008.03371*.

[Li et al. 2020b] Li, L., Fan, Y., Tse, M., and Lin, K. Y. (2020b). A review of applications in federated learning. *Computers and Industrial Engineering*, 149(September).

[Li et al. 2020c] Li, T., Sahu, A. K., Talwalkar, A., and Smith, V. (2020c). Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3):50–60.

[Lim et al. 2020] Lim, W. Y. B., Luong, N. C., Hoang, D. T., Jiao, Y., Liang, Y. C., Yang, Q., Niyato, D., and Miao, C. (2020). Federated learning in mobile edge networks: A comprehensive survey. *IEEE Communications Surveys Tutorials*, 22(3):2031–2063.

[Mao et al. 2018] Mao, H., Zhang, Z., Xiao, Z., and Gong, Z. (2018). Modelling the dynamic joint policy of teammates with attention multi-agent ddpg. *Arxiv preprint arXiv:1811.07029*.

[McMahan et al. 2017] McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. (2017). Communication-Efficient Learning of Deep Networks from Decentralized Data. In Singh, A. and Zhu, J., editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 1273–1282, Fort Lauderdale, FL, USA. PMLR.

[McMahan et al. 2016a] McMahan, H. B., Moore, E., Ramage, D., and Arcas, B. A. Y. (2016a). Federated learning of deep networks using model averaging. *ArXiv preprint arXiv:1602.05629*, abs/1602.05629.

[McMahan et al. 2016b] McMahan, H. B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. (2016b). Communication-efficient learning of deep networks from decentralized data. *Arxiv preprint arXiv:1602.05629*.

[Melis et al. 2019] Melis, L., Song, C., Cristofaro, E. D., and Shmatikov, V. (2019). Exploiting unintended feature leakage in collaborative learning. In *IEEE Symposium on Security and Privacy (SP)*, pages 691–706.

[Mothukuri et al. 2021] Mothukuri, V., Parizi, R. M., Pouriyeh, S., Huang, Y., Dehghantanha, A., and Srivastava, G. (2021). A survey on security and privacy of federated learning. *Future Generation Computer Systems*, 115:619–640.

[Nadiger et al. 2019] Nadiger, C., Kumar, A., and Abdelhak, S. (2019). Federated reinforcement learning for fast personalization. In *2019 IEEE Second International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*, pages 123–127. IEEE.

[Nishio and Yonetani 2019] Nishio, T. and Yonetani, R. (2019). Client selection for federated learning with heterogeneous resources in mobile edge. In *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, pages 1–7. IEEE.

[Niu et al. 2021] Niu, Z., Zhong, G., and Yu, H. (2021). A review on the attention mechanism of deep learning. *Neurocomputing*, 452:48–62.

[Preuveneers et al. 2018] Preuveneers, D., Rimmer, V., Tsingenopoulos, I., Spooren, J., Joosen, W., and Ilie-Zudor, E. (2018). Chained anomaly detection models for federated learning: An intrusion detection case study. *Applied Sciences*, 8(12):2663.

[Qiao et al. 2021] Qiao, Z., Yu, X., Zhang, J., and Letaief, K. B. (2021). Communication-efficient federated learning with dual-side low-rank compression. *arXiv preprint arXiv:2104.12416*.

[Rastegari et al. 2016] Rastegari, M., Ordonez, V., Redmon, J., and Farhadi, A. (2016). XNOR-Net: ImageNet classification using binary convolutional neural networks. In *European conference on computer vision*, pages 525–542. Springer.

[Rivest et al. 1978] Rivest, R. L., Adleman, L., Dertouzos, M. L., et al. (1978). On data banks and privacy homomorphisms. *Foundations of secure computation*, 4(11):169–180.

[Rummery and Niranjan 1994] Rummery, G. A. and Niranjan, M. (1994). *On-line Q-learning using connectionist systems*, volume 37. University of Cambridge, Department of Engineering Cambridge, UK.

[Saha and Ahmad 2021] Saha, S. and Ahmad, T. (2021). Federated transfer learning: concept and applications. *Arxiv preprint arXiv:2010.15561*.

[Saharkhizan et al. 2020] Saharkhizan, M., Azmoodeh, A., Dehghantanha, A., Choo, K.-K. R., and Parizi, R. M. (2020). An ensemble of deep recurrent neural networks for detecting iot cyber attacks using network traffic. *IEEE Internet of Things Journal*, 7(9):8852–8859.

[Samarakoon et al. 2018] Samarakoon, S., Bennis, M., Saad, W., and Debbah, M. (2018). Federated learning for ultra-reliable low-latency v2v communications. In *2018 IEEE Global Communications Conference (GLOBECOM)*, pages 1–7. IEEE.

[Sandler et al. 2018] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520.

[Seo et al. 2020] Seo, H., Park, J., Oh, S., Bennis, M., and Kim, S.-L. (2020). Federated knowledge distillation. *arXiv preprint arXiv:2011.02367*.

[Sheller et al. 2020] Sheller, M. J., Edwards, B., Reina, G. A., Martin, J., Pati, S., Kotrotsou, A., Milchenko, M., Xu, W., Marcus, D., Colen, R. R., and Bakas, S. (2020). Federated learning in medicine: Facilitating multi-institutional collaborations without sharing patient data. *Scientific Reports*, 10(12598):1–12.

[Shokri et al. 2017] Shokri, R., Stronati, M., Song, C., and Shmatikov, V. (2017). Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 3–18. IEEE.

[Sutton et al. 1998] Sutton, R. S., Barto, A. G., et al. (1998). *Introduction to reinforcement learning*, volume 135. MIT press Cambridge.

[Tamuly 2018] Tamuly, K. (2018). Compression and acceleration of high-dimensional neural networks. [Online; posted 15-September-2018].

[Watkins and Dayan 1992] Watkins, C. J. and Dayan, P. (1992). Q-learning. *Machine learning*, 8(3-4):279–292.

[Xie et al. 2019] Xie, P., Wu, B., and Sun, G. (2019). Bayhenn: combining bayesian deep learning and homomorphic encryption for secure dnn inference. *arXiv preprint arXiv:1906.00639*.

[Xu et al. 2021] Xu, J., Glicksberg, B. S., Su, C., Walker, P., Bian, J., and Wang, F. (2021). Federated learning for healthcare informatics. *Journal of Healthcare Informatics Research*, 5(1):1–19.

[Yang et al. 2019a] Yang, Q., Liu, Y., Chen, T., and Tong, Y. (2019a). Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):1–19.

[Yang et al. 2019b] Yang, Q., Liu, Y., Cheng, Y., Kang, Y., Chen, T., and Yu, H. (2019b). *Federated learning*, volume 13. Morgan & Claypool Publishers.

[Yang et al. 2018] Yang, T., Andrew, G., Eichner, H., Sun, H., Li, W., Kong, N., Ramage, D., and Beaufays, F. (2018). Applied federated learning: Improving google keyboard query suggestions. *arXiv preprint arXiv:1812.02903*.

[Yang et al. 2019c] Yang, W., Zhang, Y., Ye, K., Li, L., and Xu, C.-Z. (2019c). Ffd: a federated learning based method for credit card fraud detection. In *International Conference on Big Data*, pages 18–32. Springer.

[Yao 1982] Yao, A. C. (1982). Protocols for secure computations. In *23rd annual symposium on foundations of computer science (sfcs 1982)*, pages 160–164. IEEE.

[Zhao et al. 2018] Zhao, Y., Li, M., Lai, L., Suda, N., Civin, D., and Chandra, V. (2018). Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*.

[Zhu et al. 2021] Zhu, H., Zhang, H., and Jin, Y. (2021). From federated learning to federated neural architecture search: a survey. *Complex & Intelligent Systems*, 7(2):639–657.

[Zhuo et al. 2020] Zhuo, H. H., Feng, W., Lin, Y., Xu, Q., and Yang, Q. (2020). Federated deep reinforcement learning. *Arxiv preprint arXiv:1901.08277*.

Realização

Organização

Patrocinadores