

Capítulo

1

Desenvolvendo o Código da Internet do Futuro

Alberto Egon Schaeffer-Filho, Jéferson Campos Nobre, Juliano Wickboldt, Lisandro Zambenedetti Granville, Luciano Paschoal Gaspary, Weverton Luis da Costa Cordeiro

Programa de Pós-Graduação em Computação (PPGC) - UFRGS

Abstract

Research in computer networks in the 1970s and 1980s led to the emergence of the Internet, a significant technological achievement. It evolved from an academic network for simple text files exchanges into a multimedia platform that transformed global interaction and disrupted various industries. The Internet's success is attributed to robust technologies, but its inflexibility led to its "ossification." Network programmability, allowing software development on network devices, is revitalizing the Internet. This chapter explores this evolution, focusing on virtualization, network programming languages, monitoring, current challenges, and opportunities.

Resumo

As pesquisas em redes de computadores nas décadas de 1970 e 1980 culminaram com o surgimento da Internet, uma conquista tecnológica significativa. Ela evoluiu de uma rede acadêmica simples para uma plataforma multimídia que transformou a interação global e perturbou várias indústrias. O sucesso da Internet se deve a tecnologias robustas, mas sua rigidez levou à "ossificação". A programabilidade de redes, permitindo o desenvolvimento de software nos dispositivos de rede, está revitalizando a Internet. Este capítulo explora essa evolução, com foco em virtualização, linguagens de programação para redes, monitoramento, desafios atuais e oportunidades.

1.1. Introdução

As pesquisas em redes de computadores nas décadas de 1970 e 1980 culminaram com o surgimento da Internet, uma das conquistas tecnológicas recentes mais significativas da

Vídeo com a apresentação do capítulo: https://youtu.be/ma8UYCJC_kc

humanidade. A Internet passou de uma rede acadêmica voltada para a troca de arquivos de texto simples para uma plataforma multimídia que revolucionou a forma como as pessoas de todo o mundo interagem. Isso levou a disrupções em diversas áreas, abalando indústrias como a de telefonia, fonográfica, audiovisual, jornalística, entre outras. A Internet provocou a queda vertiginosa de várias indústrias tradicionais e criou as condições para o surgimento de impérios tecnológicos inovadores, como Google, Facebook e Amazon.

O sucesso da Internet se deve, entre outros fatores, a um conjunto de tecnologias de redes de computadores absolutamente robustas e estáveis, representadas principalmente pela suíte de protocolos TCP/IP [Comer 2017]. No entanto, ao longo dos anos, a Internet, que foi tão inovadora e possibilitou tantas criações, tornou-se um ambiente cujo núcleo se mostrou praticamente imutável, dificultando a absorção de novas soluções. Essa incapacidade de absorver inovações em seu núcleo ficou conhecida como a ossificação da Internet [Clark 2003], o que motivou a comunidade científica a buscar formas de tornar a Internet mais flexível, assim como era nos primórdios.

Um dos movimentos de maior sucesso, que tem transformado a Internet em um ambiente novamente propício às inovações, é aquele que ficou conhecido como programabilidade de redes [Liu et al. 2021]. Com a programabilidade de redes, adquire-se a capacidade de desenvolver e implantar software diretamente nos dispositivos de rede, como switches e roteadores. Esse software de rede pode ser desenvolvido por qualquer pessoa com conhecimento técnico, não se limitando mais ao software ou firmware fornecido pelos fabricantes de dispositivos. Dessa forma, desenvolvedores criativos podem codificar soluções com software executando não apenas na periferia da Internet (*e.g.*, em servidores, estações de trabalho, dispositivos móveis, IoT), mas também e principalmente em seu núcleo (*e.g.*, switches e roteadores). Assim, a programabilidade de redes "reabre" a Internet para acomodar inovações.

Este capítulo aborda a evolução das tecnologias da Internet e, principalmente, foca nos avanços científicos recentes que têm permitido uma Internet novamente disruptiva. Serão discutidos assuntos como virtualização em redes de computadores (Seção 1.2), linguagens de programação para redes (Seção 1.3), monitoramento e desempenho (Seção 1.4), in-network computing (Seção 1.5), redes móveis programáveis (Seção 1.6) e oportunidades para novos desenvolvimentos via redes programáveis (Seção 1.7).

1.2. Virtualização em Redes de Computadores

Comparando a indústria de redes de computadores com a indústria de desktops e servidores no que diz respeito às tecnologias de virtualização, a primeira apresenta um histórico de ciclos defasados em relação à segunda. A virtualização de desktops e servidores permite que sistemas operacionais distintos sejam executados no mesmo hardware, desacoplando assim o software do hardware. Isso possibilita que os consumidores adquiram hardware de um fornecedor e software de outro. Esse fato se tornou tão comum que agora parece óbvio. No entanto, até recentemente, o mesmo não era aplicável à indústria de redes de computadores. Os operadores de redes costumavam adquirir equipamentos e funções (em software ou firmware) sempre do mesmo fornecedor. Não havia a possibilidade de instalar software de um fornecedor em hardware de outro fornecedor. Isso resultava em custos mais elevados, limitava a liberdade de escolha e dificultava a inovação

nas redes.

A virtualização em redes de computadores é uma abordagem que ataca o problema acima ao permitir a criação de redes virtuais independentes dentro de uma infraestrutura de rede física compartilhada. Essa tecnologia é baseada em software de virtualização que divide e compartilha recursos de rede, como largura de banda, endereços IP e switchings, entre várias instâncias virtuais. Isso se traduz em benefícios significativos para a gerência e o desempenho da rede.

Uma das principais tecnologias empregadas na criação de redes com suporte a virtualização é o conceito de SDN (*Software-Defined Networking*) [Feamster et al. 2014]. O SDN separa o plano de controle do plano de dados em uma rede, desacoplando a inteligência de gerenciamento de rede (controle) dos dispositivos de rede físicos (dados) (Figura 1.1). O elemento central do SDN é o controlador, um software responsável por tomar decisões sobre como o tráfego na rede deve ser encaminhado com base em políticas definidas por software. Os dispositivos de rede, como switches e roteadores, são simplificados e se tornam dispositivos rudimentares que simplesmente encaminham o tráfego conforme as instruções do controlador.

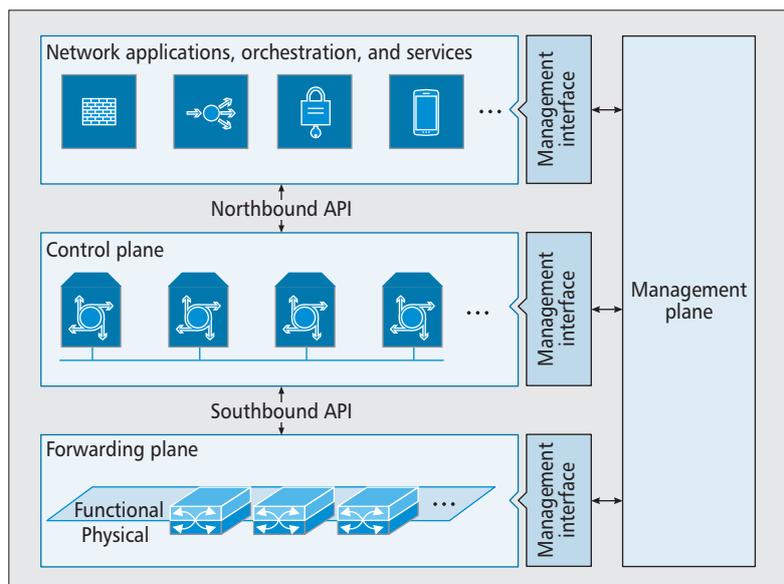


Figura 1.1. Arquitetura SDN

No contexto da virtualização de redes, o SDN desempenha um papel fundamental. Ele permite a criação de várias redes virtuais independentes em cima da infraestrutura física compartilhada. Isso é especialmente valioso em ambientes como data centers e nuvem, onde a flexibilidade e a escalabilidade são essenciais. Os benefícios do SDN incluem maior agilidade na implantação de serviços, redução de custos operacionais, melhor utilização de recursos de rede, automação avançada e capacidade de adaptação dinâmica às mudanças no tráfego.

Em termos de aplicações, o SDN é amplamente utilizado em data centers, provedores de serviços de nuvem, redes empresariais e até mesmo em ambientes de campus universitários. Em resumo, o SDN transforma a maneira como as redes são configuradas

e gerenciadas, tornando-as mais flexíveis, eficientes e adaptáveis, o que é fundamental em ambientes onde a virtualização de redes desempenha um papel crucial [Wickboldt et al. 2015].

Além de SDN, NFV (Virtualização de Funções de Rede) [Guo and McKeown 2018] é uma abordagem fundamental no contexto da virtualização de redes. NFV visa transformar funções de rede tradicionalmente executadas em hardware dedicado, como roteadores e firewalls, em software que pode ser executado em servidores de propósito geral e dispositivos de rede comuns. Isso é alcançado por meio de Funções de Rede Virtualizadas (VNFs), que são pacotes de software que representam essas funções de rede.

O coração do NFV é a virtualização das funções de rede. Essas VNFs podem ser implantadas, escalonadas e gerenciadas de forma flexível em ambientes virtualizados, proporcionando flexibilidade às operadoras de rede para adaptar suas infraestruturas às demandas em constante mudança.

O NFV também envolve um sistema de orquestração, que é responsável por gerenciar a implantação e o escalonamento das VNFs em servidores virtuais. Além disso, a infraestrutura virtualizada fornece os recursos de computação, armazenamento e rede necessários para hospedar as VNFs.

Os benefícios do NFV incluem flexibilidade, redução de custos, escalabilidade, maior inovação e gerenciamento simplificado. Ele é utilizado para aprimorar a agilidade das redes, permitir a rápida adaptação às mudanças nas demandas de tráfego e facilitar a introdução de novos serviços e funções de rede. Em resumo, o NFV desempenha um papel crucial na transformação das redes tradicionais em infraestruturas de rede virtualizadas, tornando-as mais eficientes, flexíveis e adaptáveis às necessidades dinâmicas das redes modernas.

1.2.1. Ossificação da Internet

A "ossificação da Internet" é um termo que descreve um fenômeno no qual a evolução e a inovação na arquitetura e nos protocolos da Internet se tornam mais lentas ou até mesmo estagnam devido à resistência à mudança e à dependência de sistemas existentes. Esse fenômeno ocorre devido a várias razões.

Uma das principais razões para a ossificação da Internet é a ampla implantação de tecnologias e padrões estabelecidos. Quando as redes e os sistemas já estão funcionando de acordo com um conjunto específico de protocolos, é difícil convencer os operadores a adotar novos padrões, mesmo que sejam mais eficientes ou seguros. A interoperabilidade e a compatibilidade retroativa com equipamentos mais antigos podem se tornar desafios significativos.

Outro fator que contribui para a ossificação é a influência de interesses comerciais e econômicos. Grandes empresas e organizações que investiram pesadamente em tecnologias existentes podem resistir a mudanças que possam afetar seus modelos de negócios ou investimentos. Isso pode resultar na manutenção de sistemas obsoletos por mais tempo do que o ideal.

Além disso, a complexidade da Internet atual também desempenha um papel na

ossificação. À medida que a Internet cresceu e evoluiu, tornou-se uma rede global altamente interconectada, e qualquer mudança na infraestrutura ou protocolos deve ser cuidadosamente coordenada em todo o mundo. Isso pode levar a processos de tomada de decisão lentos e conservadores.

A ossificação da Internet é um desafio significativo, pois limita a capacidade da Internet de se adaptar às novas demandas e ameaças emergentes, como segurança cibernética e escalabilidade. Para combater esse fenômeno, é importante incentivar a pesquisa contínua, a padronização flexível e a adoção de novas tecnologias, enquanto também considera cuidadosamente as implicações para a interoperabilidade e a estabilidade da rede.

1.2.2. Mitigando a Ossificação da Internet com Virtualização

A virtualização de redes desempenha um papel importante no combate contra a ossificação da Internet. A ossificação ocorre quando a evolução e a inovação na arquitetura e nos protocolos da Internet diminuem devido à resistência à mudança e à dependência de sistemas e tecnologias existentes. A virtualização de redes oferece uma solução para este problema, introduzindo flexibilidade e adaptabilidade na infraestrutura de rede.

Uma das maneiras pelas quais a virtualização de redes combate a ossificação é permitindo a introdução rápida de novos serviços e funções de rede. Isso é feito por meio da criação de VNFs (como parte do arcabouço NFV), que podem ser implantadas como software sem a necessidade de modificar o hardware físico ou os protocolos existentes. Isso promove a inovação contínua sem interromper os serviços existentes.

A virtualização de redes também oferece escalabilidade sob demanda, permitindo que os recursos de rede sejam dimensionados de acordo com as necessidades do tráfego, eliminando a necessidade de adquirir hardware dedicado. Além disso, sistemas de gerenciamento centralizados e orquestração simplificam a administração e a introdução de mudanças na infraestrutura de rede.

Ao aderir a padrões abertos e desacoplar o software do hardware, a virtualização de redes promove a interoperabilidade e a colaboração entre fornecedores e desenvolvedores, facilitando a introdução de soluções inovadoras. Além disso, ela permite uma migração gradual de serviços legados para ambientes virtualizados, garantindo a compatibilidade com sistemas existentes.

Em última análise, a virtualização de redes oferece a flexibilidade necessária para que a Internet continue evoluindo e atendendo às crescentes demandas dos usuários e das aplicações, ao mesmo tempo em que mantém a estabilidade e a interoperabilidade com sistemas legados, combatendo assim a ossificação da Internet.

A virtualização de redes emerge como uma solução para combater a ossificação da Internet. Ela traz flexibilidade, agilidade e inovação para a infraestrutura de rede, permitindo a evolução contínua sem a necessidade de alterar hardware ou protocolos existentes. Isso é essencial em um cenário em que a resistência à mudança e a dependência de tecnologias estabelecidas podem impedir a adaptação da Internet às crescentes demandas e desafios. A virtualização de redes também promove a introdução rápida de novos serviços e funções de rede, escalabilidade eficiente e gerenciamento simplificado, tornando-a uma abordagem valiosa para operadoras de telecomunicações, provedores de serviços de rede

e empresas que buscam inovação e eficiência.

Relacionado a esse contexto de evolução das redes, surgem linguagens de programação específicas, como o P4 (*Programming Protocol-independent Packet Processors*), que desempenham um papel fundamental. O P4 permite a programação de dispositivos de rede para personalizar o processamento de pacotes, abrindo caminho para a criação de redes mais inteligentes e adaptáveis. Essa linguagem fornece as ferramentas necessárias para definir como os pacotes de dados são manipulados na rede, oferecendo um controle granular sobre o comportamento da rede. Em síntese, a combinação da virtualização de redes com linguagens de programação como o P4 representa uma abordagem importante para modernizar e revitalizar a Internet.

1.3. Linguagens para Programação para Redes

Redes programáveis correspondem a um paradigma que permite que a funcionalidade da rede seja definida e modificada dinamicamente por meio do software [Feamster et al. 2014]. As redes programáveis podem incluir vários componentes, como switches e roteadores programáveis, FPGAs, SmartNICs e servidores virtualizados. Esses componentes podem ser programados para executar tarefas específicas ou implementar protocolos personalizados, tornando a rede mais flexível e adaptável às mudanças de requisitos.

1.3.1. Perspectiva Histórica

Embora as redes programáveis estejam se tornando mais comuns, a ideia remonta à década de 90, quando as *active networks* foram introduzidas. O objetivo de *active networks* era tornar os roteadores mais flexíveis, permitindo que os desenvolvedores executassem programas personalizados neles. As *active networks* foram motivadas pela necessidade de alguns pesquisadores implementarem seus próprios protocolos e funcionalidades em suas redes, sem que fosse necessário esperar pelos processos de padronização da IETF para que pudessem implantar uma nova ideia [Wetherall and Tennenhouse 2019].

Conceitualmente, os pesquisadores idealizaram duas formas de processamento de código em roteadores:

- **Cápsulas:** a primeira é a substituição dos pacotes por *cápsulas* que carregam código que a infraestrutura poderia hospedar e processar;
- **Switch/roteador programável:** a outra abordagem foi o *switch programável*, permitindo instalar programas no switch que seriam responsáveis por controlar a infraestrutura [Tennenhouse and Wetherall 1996].

Porém, a ideia de *active networks* não teve a “transferência de resultados” esperada para a indústria, e acabou não sendo amplamente adotada por setores fora do meio acadêmico [Wetherall and Tennenhouse 2019]. Apesar disso, os pesquisadores pela primeira vez imaginaram a noção de uma rede programável, que era um design radicalmente diferente se comparado às arquitetura tradicionais de redes existentes na época. As *active networks* pressupunham que dispositivos de rede seriam capazes de expor seu estado a uma interface programável, um conceito posteriormente revisitado pelo OpenFlow e pelos planos de dados programáveis [Feamster et al. 2014].

Enquanto isso, os operadores de rede careciam de mecanismos adequados para gerenciar e configurar suas redes. Tipicamente, essas redes eram compostas por dispositivos de diferentes fornecedores, incluindo protocolos diferentes, e muitas vezes exigiam muito esforço manual para configurá-las. Normalmente, o plano de dados e o plano de controle estavam totalmente integrados aos dispositivos, dificultando a execução de tarefas de engenharia de tráfego ou depuração. Para simplificar o gerenciamento, os pesquisadores começaram a defender a separação do plano de controle e do plano de dados [Feamster et al. 2014]. Vários pesquisadores deram contribuições significativas mostrando que era possível separar os planos, mas esta separação só foi concretamente alcançada com o surgimento do OpenFlow [McKeown et al. 2008].

1.3.2. Surgimento de SDN e OpenFlow

O conceito de *Software-Defined Networking* (SDN) fornece uma abstração de controlador logicamente centralizado, sendo capaz de interagir com o plano de dados usando APIs específicas. A ascensão do SDN motivou a noção de sistemas operacionais de rede, como ONOS [Berde et al. 2014] e NOX [Gude et al. 2008]. Além disso, devido às preocupações relacionadas aos riscos de ter um controlador centralizado, a comunidade SDN adotou a ideia de ter um plano de controle fisicamente distribuído (porém mantendo a consistência do estado distribuído, e fornecendo suporte para aplicações de rede em geral).

Dentre as APIs destinadas à interação entre o plano de controle e o plano de dados destaca-se o protocolo OpenFlow [McKeown et al. 2008]. O OpenFlow aproveitou uma importante abstração comum entre vários switches e roteadores: a abstração *match+action*. Usando essa abstração, uma API OpenFlow permitiu que programas externos alterassem entradas de tabelas de fluxo do switch, normalmente implementadas em TCAM. A abstração *match+action* era simples, porém poderosa, e rapidamente se tornou difundida porque o hardware da tabela de fluxo era comum entre os chips de switch existentes. Isso era vantajoso para os fornecedores de equipamentos pois eles não precisavam trocar completamente seus equipamentos.

O surgimento de SDN e OpenFlow permitiu que pesquisadores e operadores pudessem usar SDN para estudar e começar a implantar suas próprias soluções em suas redes. Por exemplo, um operador de rede poderia escrever uma aplicação de plano de controle para balancear efetivamente a carga na rede. Esta aplicação poderia, por exemplo, executar um algoritmo personalizado e definir novas entradas nas tabelas de encaminhamento para pacotes subsequentes de um fluxo. A estratégia de alterar as entradas da tabela se opõe à forma como o balanceamento de carga era implementado nas redes tradicionais: uma rede tradicional estaria limitada a alterar os pesos dos links do protocolo do plano de controle (por exemplo, OSPF) implementado pelo fornecedor do equipamento, não permitindo que os operadores construíssem sua própria solução.

Ao motivar o uso de uma interface aberta entre os planos de dados e de controle, as redes OpenFlow permitiram aos pesquisadores experimentar novas ideias e simplificar o gerenciamento. Embora o OpenFlow tenha trazido esses benefícios para os operadores de rede, o grau de programabilidade na rede usando SDN ainda era pequeno. Um switch OpenFlow possui uma tabela de fluxos que mapeia fluxos para uma ação específica. A única programabilidade no switch é feita alterando as entradas da tabela de fluxo por meio

da API OpenFlow para executar ações específicas. Além disso, as entradas nas tabelas de match+action do OpenFlow correspondiam apenas a um conjunto fixo de campos de cabeçalhos do pacote. Por fim, o conjunto de ações disponíveis era fixo, incluindo apenas ações de descartar, encaminhar e enviar um pacote para o controlador.

1.3.3. Programabilidade do Plano de Dados

A programabilidade do plano de dados permite que o operador da rede defina a funcionalidade do plano de dados usando artefatos de software. A funcionalidade a ser implementada nos switches é frequentemente expressa usando linguagens de domínio específico como P4 [Bosshart et al. 2014] ou POF [Song 2013] e então é adaptada em um modelo abstrato de plano de dados [Hauser et al. 2021]. O código resultante é então compilado em uma arquitetura de processamento de pacotes que suporta o modelo de plano de dados. Um pipeline de processamento de pacotes inclui uma sequência de operações que um dispositivo do plano de dados da rede executa para processar um pacote [Gunturi et al. 2005]. Os componentes de um pipeline de processamento de pacotes podem variar de acordo com as diferentes arquiteturas de pipeline.

Um exemplo concreto de arquitetura de processamento de pacotes é a Protocol Independent Switch Architecture (PISA), que generaliza o modelo Reconfigurable Match-Table (RMT) [Bosshart et al. 2013]. O modelo RMT é uma tecnologia que torna as tabelas match+action dinamicamente programáveis sem modificar o hardware. Isso distingue os switches que usam RMT dos switches OpenFlow de várias maneiras. Na arquitetura de processamento de pacotes do OpenFlow, o número de tabelas, seu pipeline, seus tipos de correspondência e tamanhos são determinados durante a fabricação. Consequentemente, estes parâmetros permanecem sempre os mesmos, limitando a flexibilidade. Em contraste com os switches OpenFlow, o RMT permite que os administradores de rede definam os campos na tabela match+action para suas necessidades específicas. Além disso, o modelo RMT permite alterar a largura e a profundidade de uma tabela match+action, suportando diferentes tamanhos de entradas. A ação disparada e o número de ações disponíveis por uma correspondência também são programáveis, podendo executar funcionalidades customizadas caso um pacote corresponda a uma entrada em uma tabela. Finalmente, o programador pode definir a topologia entre tabelas match+action no pipeline em vez de ser determinada durante a fabricação.

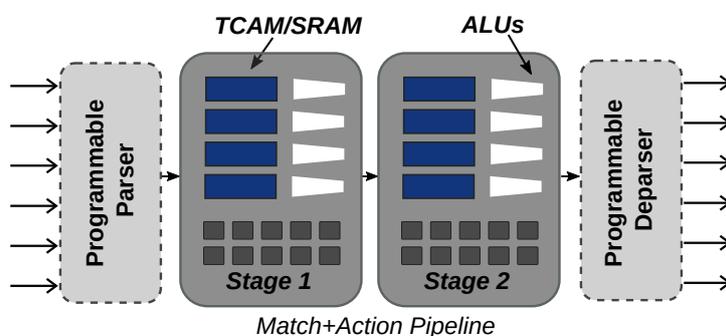


Figura 1.2. Arquitetura PISA

Arquitetura PISA. A arquitetura PISA aproveita a tecnologia RMT para fornecer

processamento de pacotes de taxa de linha. A Figura 1.2 ilustra a arquitetura PISA. Nela, os pacotes passam por um analisador de pacotes, que extrai os campos de cabeçalho necessários para o programa, como TCP, destinos IP e *metadados* importantes. *Metadados* são variáveis por pacote que armazenam valores temporários para auxiliar no processamento de pacotes, como portas de entrada e saída. Depois que o analisador processa um pacote, ele segue para um pipeline de processamento de pacotes que pode ser composto de diversas construções, como tabelas e registros *match+action*. As correspondências podem ter vários formatos, como correspondência de prefixo mais longo, correspondência ternária ou exata. Cada correspondência corresponde à execução de uma ação que executa computação e realiza operações de armazenamento. As tabelas *match+action* podem ser executadas em TCAM ou SRAM, e as ações são implementadas usando ALUs e devem ser capazes de executar na taxa de linha [Sivaraman et al. 2015]. Cada pipeline possui um conjunto fixo de estágios e cada estágio possui registradores disponíveis, que armazenam estado persistente. Cada estágio também possui um conjunto restrito de operações ALU, divididas entre operações ALU sem estado e com estado. Depois que os fluxos de controle processam um pacote, os cabeçalhos dos pacotes são emitidos por um analisador. O analisador reconstrói os cabeçalhos do pacote e o envia para uma porta de saída.

P4 - Programming Protocol-Independent Packet Processors. Linguagens de programação, como P4 [Bosshart et al. 2014] e POF [Song 2013], foram propostas para especificar a lógica de processamento de pacotes de dispositivos no plano de dados programáveis por meio de uma arquitetura de alto nível independente de abstrações. Tal linguagem pode ser usada para que operadores de rede possam rapidamente implementar novos protocolos em dispositivos de encaminhamento, personalizar suas funcionalidades e desenvolver serviços inovadores. A linguagem P4 [Bosshart et al. 2014] é uma linguagem de especificação de alto nível. A linguagem foi projetada para facilitar aos desenvolvedores a descrição do processamento de pacotes. P4 fornece uma camada de abstração acima da arquitetura PISA, fazendo com que o trabalho do compilador P4 seja mapear a funcionalidade especificada para os estágios de hardware [Hogan et al. 2022].

A linguagem P4 é frequentemente referenciada como uma linguagem declarativa [Shahbaz and Feamster 2015, Eichholz et al. 2022], com o objetivo de fornecer uma abstração de alto nível e liberar os desenvolvedores da necessidade de se preocupar como as funcionalidades são implementada no hardware do plano de dados. Os cabeçalhos dos pacotes podem ser declarados de forma semelhante a `structs` em C. O analisador (*parser*) é especificado através de uma abordagem de máquina de estado, onde os estados geralmente correspondem a uma parte do cabeçalho do pacote e as transições entre estados são transições entre cabeçalhos. Durante o tempo de execução, o analisador processa o pacote extraíndo os valores dos bits do pacote para variáveis internas do programa, que podem então ser acessadas e modificadas nos demais elementos de processamento.

O código P4 é organizado logicamente da seguinte forma (veja Figura 1.3):

1. *Declaração de dados:* é uma seção que define o formato do cabeçalho do pacote e as informações de metadados que podem ser usadas para sua análise. Esta seção é mapeada em um cabeçalho e um barramento de metadados que transporta essas informações por todos os estágios de processamento. Os tipos de cabeçalho são declarados de forma semelhante às estruturas em C, ou seja, os campos são definidos

em uma ordem específica e com um tamanho pré-determinado;

2. *Parser logic*: é uma seção que especifica como, quando e em que ordem cada um dos cabeçalhos deve ser analisado. Esta seção de um programa P4 é mapeada para os elementos *parser* e *deparser* do modelo de encaminhamento. Esses elementos são então responsáveis por extrair o campo de cabeçalho dos pacotes em seu ingresso (*parser*) e no processo de tradução de estruturas de dados em um formato que possa ser armazenado e reconstruído na sequência no mesmo ou em um ambiente computacional diferente; e
3. *Match-action tables and control flows*: é uma seção que especifica tabelas capazes de fazer match em campos de cabeçalho arbitrários e realizar modificação de cabeçalhos de pacotes (e metadados) e outras ações personalizadas. Também expressa a ordem e condições nas quais cada tabela deve ser executada.

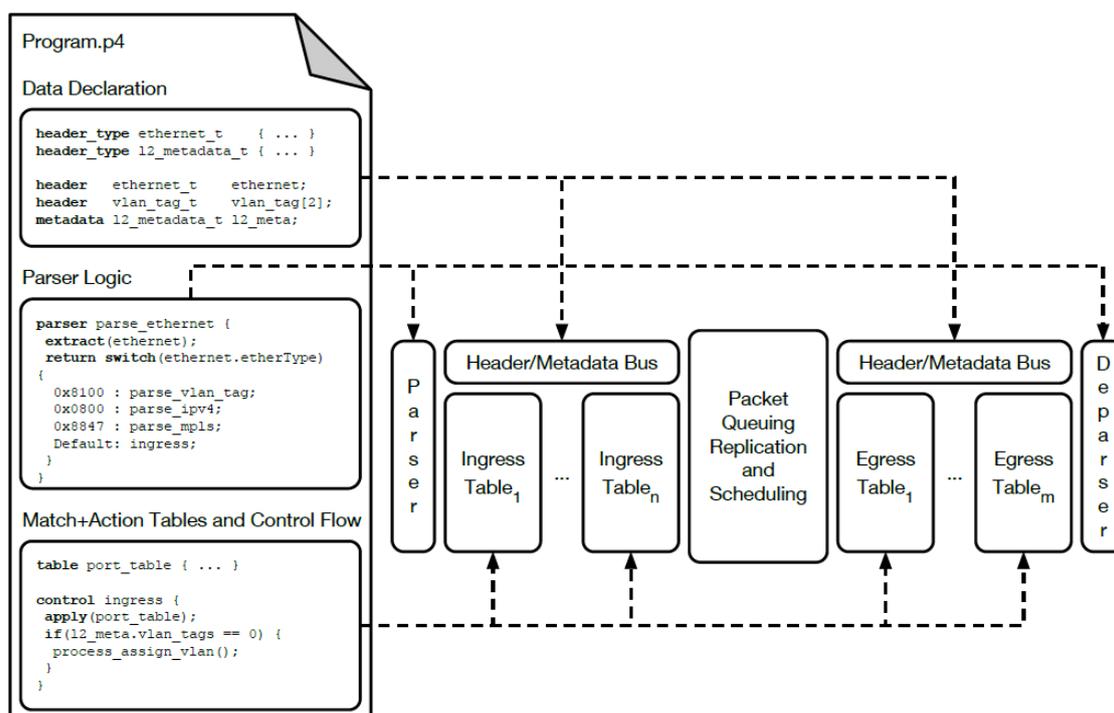


Figura 1.3. Seções de código P4 e mapeamento para o modelo de encaminhamento abstrato (Adaptado de [Kim et al. 2006])

1.4. Monitoramento e Desempenho

As redes de comunicação atuais operam com expectativas de alto desempenho em latência, largura de banda e *jitter*, especialmente com o surgimento e a proliferação de novos serviços e aplicações (por exemplo, negociação algorítmica, telecirurgia e *streaming* de vídeos de realidade virtual). Os usuários demandam garantias estritas que devem ser cumpridas, exigindo a definição de metas claras para o desempenho da rede, os chamados objetivos de nível de serviço (SLOs – *Service-Level Objectives*). Um SLO pode prescrever que o atraso fim-a-fim deve ser menor que 5 milissegundos em 95% dos pacotes de

tráfego de telecirurgia, ou que a largura de banda fornecida deve ser maior que 1 Gbps pelo menos 95% do tempo para um tráfego agregado de *streaming* de vídeo.

O monitoramento de conformidade com SLOs e o diagnóstico imediato de violações são essenciais para a operação das redes atuais e do futuro. No entanto, o monitoramento de rede é uma tarefa inerentemente difícil, às vezes comparada à busca por uma agulha no palheiro¹. Infelizmente, as arquiteturas existentes não são projetadas para monitorar SLOs com o nível de detalhe e precisão exigidos. As ferramentas tradicionais de monitoramento passivo (por exemplo, SNMP [Fedor et al. 1990, Gaspary et al. 2005] e Net-Flow/IPFIX [Cisco Networks 2023, Aitken et al. 2013]) operam em escalas de tempo longas (dezenas de segundos ou mais) e, portanto, carecem de granularidade adequada para detectar eventos como rajadas de tráfego de curta duração (por exemplo, *micro-bursts*), que podem ser críticas para toda uma nova geração de aplicações. Técnicas de medições ativas (por exemplo, ping, traceroute, OWAMP [Zekauskas et al. 2006] e TWAMP [Babiarz et al. 2008]) também não fornecem “resolução” de tempo suficiente; além disso, não há garantia de que a rede roteará e priorizará as sondas (*probes*) da mesma maneira que os pacotes de produção. O movimento consistente em direção à heterogeneidade no tratamento de tráfego [Jeyakumar et al. 2014, Hong et al. 2013], roteamento por caminhos múltiplos [Jain et al. 2013, Kumar et al. 2015] e balanceamento de carga de fluxos [Alizadeh et al. 2014, He et al. 2015] exacerba essa limitação.

1.4.1. Telemetria de Rede em Banda

A programabilidade do plano de dados torna viável um novo método de monitoramento de rede, denominado *In-band Network Telemetry* (INT). Segundo um painel recente realizado no âmbito da iniciativa The NetworkingChannel [Foster et al. 2023], INT vem sendo considerada uma das aplicações “matadoras” (de *killer application*) por fornecer uma maneira de monitorar redes e serviços com níveis de precisão e detalhes sem precedentes, ao mesmo tempo que é capaz de operar em velocidade de linha. Em essência, o método consiste em registrar informações de operação, administração e manutenção dentro de um pacote de dados à medida que ele atravessa uma rede. Várias estruturas e técnicas foram propostas para realizar telemetria de rede em banda, por exemplo, TPP [Jeyakumar et al. 2014], INT [Kim et al. 2015] e Cisco iOAM [Cisco Sa]. Elas compartilham a ideia de permitir que pacotes de dados consultem indicadores instantâneos do estado interno de cada switch por onde passam – como tamanho e latência de filas, e utilização de enlaces – e armazenem essas informações em cabeçalhos de telemetria.

A Figura 1.4 descreve o fluxo de execução do INT [Kim et al. 2015], a estrutura de telemetria em banda atualmente mais proeminente e que pode ser implementada usando P4. A abstração da arquitetura INT é composta por um controlador de monitoramento remoto e por nós de origem, trânsito e destino, cada um dos quais representando um papel em sua instanciação. Cada switch programável no caminho de um pacote (conforme é transmitido pela rede) pode assumir uma ou mais funções. A figura ilustra um cenário em que o sistema final (*end-host*) 1 envia um pacote de dados tradicional para o sistema

¹Everflow [Zhu et al. 2015]: “[Problem diagnosis] is not only akin to searching in the proverbial haystack for needles, but for specific needles of arbitrary size, shape and color.”
Sonata [Gupta et al. 2018]: “[...] telemetry queries often require finding ‘needles in a haystack’ where the fraction of total traffic or flows that satisfies these queries is tiny.”

final 2 por meio de uma rede de switches compatíveis com INT.

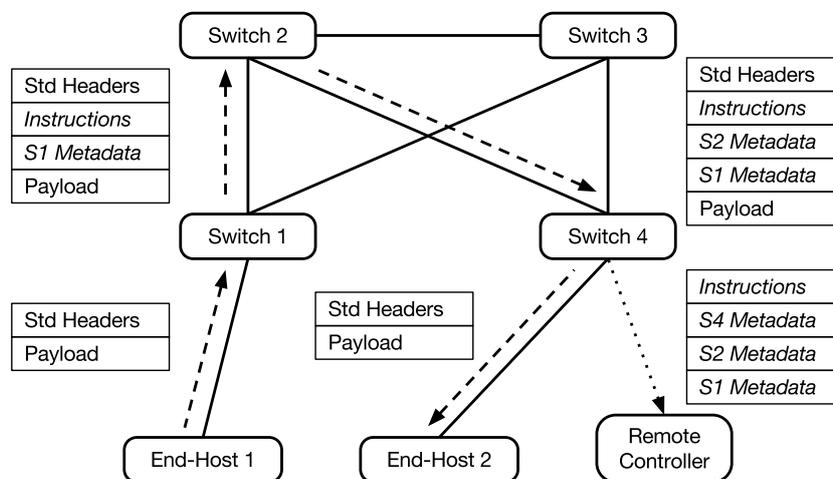


Figura 1.4. Arquitetura da estrutura INT para telemetria de rede em banda (extraída de [Marques 2022] apud [Laupkhov and Thomas 2016])

Nós de origem (no exemplo, *Switch 1*) são responsáveis por incorporar instruções de medição (normalmente na forma de valores de cabeçalho) em pacotes regulares ou de sondagem. *Nós de trânsito* executam as instruções e acrescentam valores medidos aos pacotes. No exemplo, *Switches 1, 2 e 4* assumem o papel de nós de trânsito, já que o caminho do pacote é *End-Host 1 → Switch 1 → Switch 2 → Switch 4 → End-Host 2*. Por último, *sink nodes* (no exemplo, *Switch 4*) recuperam os resultados das instruções e reportam (subconjuntos apropriados delas) a um controlador. Exemplos de metadados que podem ser coletados em cada switch são o ID do switch, o ID da porta de entrada/saída, o carimbo de data/hora, a contagem de bytes, a contagem de descartes e a utilização do enlace, bem como um ID de fila, ocupação e estado de congestionamento.

Diversos desafios vêm sendo abordados pelo Grupo de Pesquisa em Redes de Computadores no âmbito de INT [Marques et al. 2019, Marques et al. 2020, Vassoler et al. 2023]. A seguir, a título de exemplo motivador, descreve-se um desses trabalhos.

1.4.2. Sistema IntSight

Capitalizando sobre as oportunidades habilitadas por INT, projetou-se e desenvolveu-se IntSight [Marques et al. 2020], um sistema para detectar e diagnosticar violações de SLO, com foco em atraso fim-a-fim e largura de banda. Em contraste com INT clássico, IntSight usa cabeçalhos de telemetria como espaços de trabalho, onde os dispositivos de encaminhamento calculam metadados *path-wise* (por exemplo, IDs de caminho, pontos de contenção e atrasos fim-a-fim) progressivamente. A detecção no plano de dados, então, torna-se uma questão de comparar os valores calculados com os definidos pelo SLO. O sistema proposto permite que dispositivos de encaminhamento resumam dados de monitoramento e *seletivamente* relatem eventos de interesse para o plano de controle, onde ocorre o *diagnóstico* das violações.

A Figura 1.5 ilustra a abordagem do IntSight para monitoramento de rede. Ela exemplifica a trajetória de um pacote de dados em uma rede monitorada pelo sistema.

Para orientar a visão geral, considere, como exemplo, um fluxo de pacotes sujeito a SLOs relacionados a atraso fim-a-fim. Os pacotes desse fluxo ingressam na rede via o sistema final A, são roteados através do caminho $[N_x, N_y, N_z]$ e deixam a rede para chegar ao sistema final B.

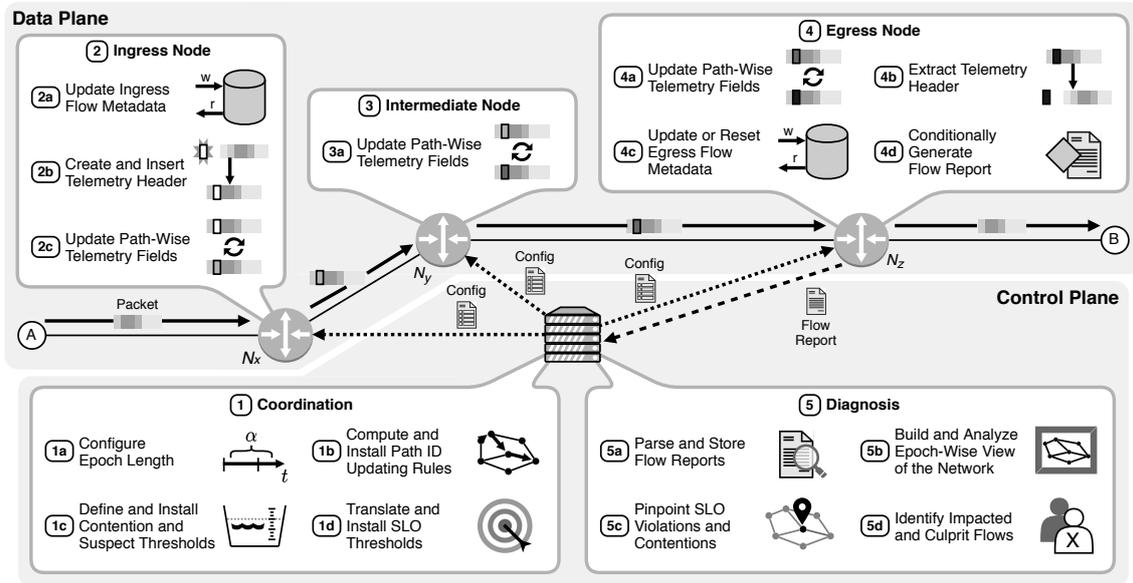


Figura 1.5. Visão geral do IntSight (extraída de [Marques et al. 2020])

No exemplo, o dispositivo N_x representa a *ingress node*, ou seja, o primeiro dispositivo na rede a receber pacotes do fluxo. Os nós de entrada têm duas tarefas principais: (i) armazenar metadados persistentes sobre o tráfego que ingressa na rede por meio deles e (ii) inicializar o processo de telemetria para os pacotes. Quando o dispositivo N_x recebe um pacote de dados vindo do sistema final A, ele segue as três etapas descritas a seguir. Inicialmente, ele atualiza os valores dos campos de metadados (que são armazenados em *arrays* de registradores) associados ao fluxo ao qual o pacote pertence (Etapa 2a na Figura 1.5). Em seguida, N_x cria e insere um cabeçalho de telemetria no pacote contendo campos como EEDelay (Etapa 2b). A lista completa de campos presentes nos cabeçalhos de telemetria é apresentada em [Marques et al. 2020].

Seguindo as duas etapas anteriores (ou seja, 2a e 2b), N_x atualiza os campos do cabeçalho de telemetria *path-wise*, de acordo com o que o pacote observou neste nó (Etapa 2c). Por exemplo, o campo CPs (*Contention Points*) é marcado caso tenha sido observada contenção no dispositivo, caracterizada pela alta ocupação da fila. O campo SPs (*Suspicion Points*) é marcado quando a vazão do fluxo é responsabilizada pela alta utilização da porta ou enlace de saída, considerando sua capacidade. Para ambos os campos CPs e SPs, a contenção e a suspeita são determinadas pela comparação dos valores medidos com limites pré-configurados em tabelas de *lookup*. Por fim, o atraso ao qual o pacote foi submetido no dispositivo é adicionado ao campo EEDelay (*End-to-End Delay*).

Nós intermediários (N_y no exemplo) executam apenas uma etapa: eles atualizam os campos *path-wise* no cabeçalho de telemetria (Etapa 3a). Isso é equivalente à Etapa 2c nos nós de ingresso. Em comparação com as demandas de processamento nos nós de

ingresso e saída, a nos nós intermediários é a mais simples e que consome menos recursos. Por exemplo, nós intermediários não armazenam persistentemente quaisquer metadados relativos ao tráfego que encaminham. Isso isenta os dispositivos centrais da rede de terem a mesma capacidade de processamento e memória que os dispositivos de borda.

Um *nó de saída*, ou seja, o último dispositivo a processar um pacote antes de ele deixar a rede (N_z na Figura 1.5), é encarregado das etapas mais importantes no procedimento de monitoramento. Depois de atualizar os campos *path-wise* (Etapa 4a, igual às Etapas 2c e 3), ele extrai o cabeçalho de telemetria do pacote (Etapa 4b). Isso faz com que o pacote retorne ao seu formato original antes de ser encaminhado ao sistema final (B no exemplo). Em seguida, o nó de saída atualiza os campos de metadados persistentes em relação ao fluxo, considerando os valores dos campos de telemetria do cabeçalho extraído (Etapa 4c). Semelhante aos nós de entrada, os nós de saída armazenam campos de metadados em *arrays* de registradores.

A última etapa em um nó de saída é gerar, *condicionalmente*, relatórios de fluxo (Etapa 4d), que são pacotes de controle com todos os metadados armazenados para o respectivo fluxo durante uma época de monitoramento. Um relatório é gerado se, e somente se, uma época acabou de terminar e um evento de interesse foi observado para o fluxo. Um evento de interesse pode ser uma violação de SLO, uma contenção experimentada pelo fluxo em um dispositivo ou uma suspeita em relação ao fluxo (de ser agressor). Ao final dessa etapa, o relatório gerado é enviado ao plano de controle para ser analisado e informar o diagnóstico do problema, conforme descrito a seguir.

A aplicação do IntSight que executa no plano de controle possui duas tarefas principais: a coordenação do processo de monitoramento e o diagnóstico de problemas. As etapas relacionadas a ambas as tarefas estão ilustradas na seção inferior da Figura 1.5. Focando no diagnóstico de problemas, enquanto os dispositivos do plano de dados processam e encaminham pacotes, o plano de controle “escuta” continuamente os relatórios de fluxo provenientes desses dispositivos e executa as etapas de diagnóstico a seguir. Primeiro, ele analisa os relatórios recebidos e armazena suas informações em um banco de dados de metadados (Etapa 5a). Essa etapa salva informações de forma persistente, permitindo análises históricas e em tempo real. Os relatórios são armazenados no banco de dados considerando características temporais, ou seja, sua época. IntSight constrói uma visão global da rede para cada época e analisa seu estado, comportamento e desempenho em busca de eventos de interesse (Etapa 5b).

Quando um evento de interesse (ou seja, violação, contenção ou suspeição) é detectado, o IntSight usa as informações relatadas para identificar onde (ou seja, em quais dispositivos) tal evento aconteceu (Etapa 5c). Por fim, o IntSight diagnostica o evento de interesse, analisando o comportamento de todos os fluxos presentes nos dispositivos identificados e indicando as vítimas e os culpados (Etapa 5d).

1.5. In-network Computing

In-network computing (INC) ou computação em rede é um paradigma emergente na área de redes programáveis onde uma parte significativa do processamento de um sistema distribuído é descarregada dos servidores para o plano de dados dos dispositivos de rede [Michel et al. 2021, Benson 2019, Sapio et al. 2017]. A ideia por trás do INC é realizar

tarefas de computação próximas à fonte de dados, reduzindo a sobrecarga de transmissão de dados e melhorando assim a eficiência geral do sistema.

Além disso, ao invés de adicionar novos equipamentos à infraestrutura, a computação em rede concentra-se em dispositivos que já existem na infraestrutura para encaminhar o tráfego. Isso se traduz na redução e necessidade de equipamentos especializados adicionais, como aceleradores ou middleboxes.

1.5.1. Princípios da In-network Computing

A adoção de uma estratégia de descarregamento ou *offloading* para a rede é motivada principalmente pelas vantagens de desempenho que ela pode proporcionar. Com INC, é possível **reduzir a latência** interceptando e processando pacotes no plano de dados dos dispositivos de rede em vez de enviar pacotes para serem processados pelos servidores finais. Dessa forma, em vez de exigir uma solicitação para concluir um RTT inteiro enviando o pacote a um servidor, alguns pacotes podem ser rapidamente processados no hardware da rede e encaminhados ao seu destino final. Além disso, ao processar pacotes nos dispositivos de rede, o INC também **economiza largura de banda** entre o switch que executa o INC e os servidores que executam a funcionalidade do host final. Ao economizar largura de banda, é possível evitar congestionamentos e obstruções que muitas vezes causam ocupação de buffer, perdas de pacotes e degradação de desempenho.

Por fim, a **redução do consumo de energia** é uma vantagem surpreendente do paradigma INC. Switches programáveis que hospedam INCs consomem recursos iguais ou menores que os tradicionais. Ao descarregar funcionalidades INC para um dispositivo programável, é possível observar a mesma quantidade de consumo de recursos observada em um switch ocioso [Tokusashi et al. 2019]. Além disso, aumentar a taxa de tráfego aumenta o consumo de energia de acordo com uma taxa constante. Portanto, quando comparados aos servidores, que podem dobrar o consumo à medida que a taxa aumenta, os switches podem escalar para taxas de tráfego maiores. No entanto, embora o consumo de um servidor seja geralmente menor do que um switch, se assumirmos que os switches já estão disponíveis na infraestrutura, é possível economizar o consumo do servidor movendo a funcionalidade usando INC [Tokusashi et al. 2019].

1.5.2. Exemplos de Sistemas INC: NetLock, NetGVT e SwitchML

Esforços recentes mostraram muitas aplicações que podem ser usadas na rede. Os casos de uso mais simples são funções de rede, como detecção de DDoS, balanceamento de carga e NAT. Casos de uso mais sofisticados incluem sistemas para treinamento e inferência de Machine Learning [Sanvito et al. 2018], que foi estudado em diversas aplicações, como classificação de tráfego, controle de braços robóticos e até visão computacional [Glebke et al. 2019]. O armazenamento em cache de pares chave-valor [Jin et al. 2017, Jin et al. 2018] e o gerenciamento de locks [Yu et al. 2020] também são possíveis, permitindo transações rápidas de bancos de dados em data centers.

Para melhor ilustrar o uso e as aplicações de INC, apresentamos a seguir um conjunto de sistemas do estado-da-arte:

Controle de concorrência. Realizar o controle de concorrência dentro da rede em switches permite o processamento de transações com RTT reduzido em comparação

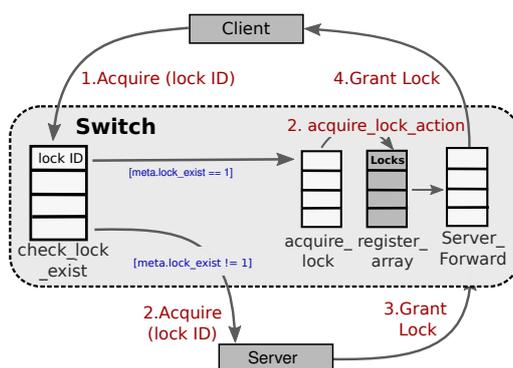


Figura 1.6. NetLock

com abordagens baseadas em servidores. No sistema Netlock [Yu et al. 2020], quando um cliente deseja adquirir um *lock* para um objeto, ele primeiro tenta obtê-lo do switch. O switch verifica em uma tabela de encaminhamento se ele é responsável por gerenciar aquele objeto. Se o *lock* existir no switch, o pacote é processado por outra tabela responsável por chamar uma ação que grava o *lock* em um array de registradores persistentes. Por outro lado, caso o switch não seja o responsável, a solicitação do *lock* será encaminhada normalmente para um servidor. Esse funcionamento é ilustrado na Figura 1.6.

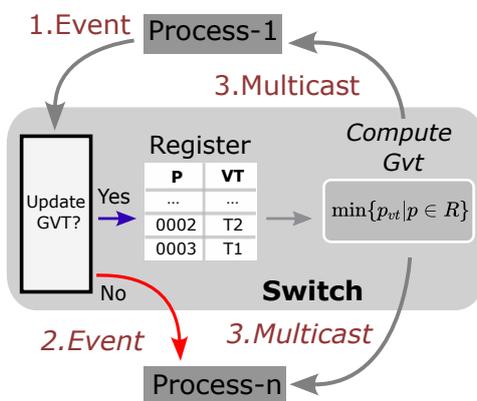


Figura 1.7. NetGVT

Sincronização de eventos. NetGVT [Parizotto et al. 2022] é um sistema que emprega in-network computing para realizar a sincronização eficiente de eventos em sistemas distribuídos. Ele foi projetado para descarregar o cálculo de um tempo virtual global (GVT) em switches de rede, permitindo-lhes sincronizar eventos com um RTT reduzido em comparação com uma solução tradicional baseada em servidores. O sistema opera interceptando pacotes de eventos enviados entre processos em execução em servidores e armazenando a *clock* virtual local do processo remetente em um registrador dentro do switch. O sistema então realiza uma comparação entre o tempo virtual armazenado nos registradores e o tempo virtual mínimo de todos os processos existentes no sistema. Se o tempo virtual do processo for o novo mínimo, o switch executa uma ação de registro que grava o novo mínimo em um registrador persistente e transmite o novo valor para todos os processos. Por outro lado, se o tempo virtual do processo não for o novo mínimo, o pacote

de eventos é encaminhado normalmente ao seu destino. Este funcionamento é ilustrado na Figura 1.7.

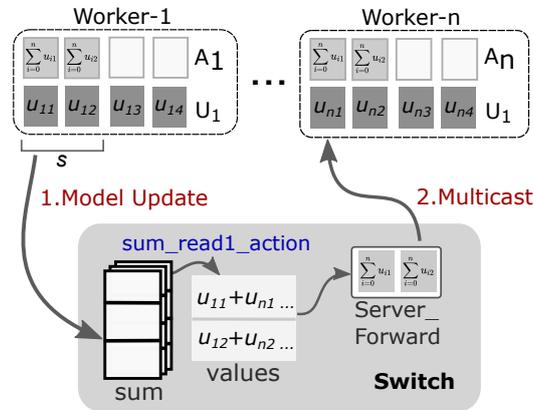


Figura 1.8. SwitchML

Agregação na rede. SwitchML [Sapio et al. 2021] é um sistema para acelerar o treinamento de aprendizado de máquina distribuído que descarrega a agregação de parâmetros de treinamento (gradientes) para o hardware do switch. Ao realizar a agregação de parâmetros nos switches, essa computação ocorre mais perto dos trabalhadores e evita a sobrecarga de comunicação imposta por soluções tradicionais baseadas em servidores. Uma tabela mapeia um pacote para uma ação que agrega os parâmetros de treinamento de cada trabalhador e os armazena em um conjunto de registradores. O gradiente resultante é encapsulado em um pacote e enviado de volta aos trabalhadores para atualizar seu modelo local. Este processo se repete para todas as partes do modelo de forma síncrona. O SwitchML oferece diversas vantagens em relação aos sistemas de agregação tradicionais, incluindo melhor escalabilidade, latência reduzida e maior eficiência devido à proximidade da agregação com os trabalhadores e às capacidades de taxa de linha dos switches. Este funcionamento é ilustrado na Figura 1.8.

1.6. Redes Móveis Programáveis

A evolução das redes móveis tem sido marcada por avanços significativos, sendo as transições para as redes de quarta e quinta geração (4G e 5G) dois dos marcos recentes mais importantes. Uma mudança notável nesse processo tem sido a transição de implementações baseadas em hardware de propósito específico, desenvolvidas por um grupo restrito de grandes fabricantes, para implementações baseadas em software, inclusive de código aberto, e em tecnologias como o Rádio Definido por Software (*Software-Defined Radio – SDR*) [Bonati et al. 2020]. Essa mudança é relevante do ponto de vista científico e tecnológico, pois permite maior flexibilidade e participação de diversos atores no desenvolvimento de componentes de rede. Além disso, torna cada vez mais economicamente viável a implantação de redes móveis (5G ou 4G) privadas, sem vínculo com operadoras utilizando software aberto e hardware de propósito geral [Aijaz 2020]. Atualmente, essa mudança de paradigma permite uma inovação mais distribuída tanto nos componentes de Rede de Acesso via Rádio (*Radio Access Network – RAN*) quanto na Rede de Núcleo (*Core Network – CN*).

Projetos de software de código aberto, juntamente com a tecnologia SDR, têm desempenhado um papel crucial na promoção da inovação na academia e na indústria de telecomunicações. Entre os projetos mais notáveis estão o Open Air Interface² e o srs-RAN³, que oferecem implementações completas e funcionais dos componentes da RAN compatíveis com os padrões do 3GPP (3rd Generation Partnership Project), tanto para 4G quanto para 5G, e podem ser executados sobre dispositivos SDR programáveis via FPGA, como Ettus USRP, LimeSDR e Nuand bladeRF [Mihai et al. 2022]. Isso permite que pesquisadores conduzam experimentos realistas e contribuam para o desenvolvimento de futuras gerações de redes móveis, uma vez que podem modificar e personalizar essas implementações de acordo com suas necessidades de pesquisa.

Além disso, projetos como o Open5GS⁴ e o Free5GC⁵ fornecem implementações funcionais dos componentes de núcleo da rede, que são responsáveis pelo registro e autenticação de usuários, gestão de mobilidade, fatiamento de rede, entre outras funções. Isso amplia ainda mais as oportunidades para a pesquisa e desenvolvimento em redes móveis de próxima geração. Essas implementações abertas associadas a tecnologias nativas da nuvem (*cloud-native*), como o Docker e Kubernetes, vem sendo gradualmente adotadas pela indústria de telecomunicações [Sekigawa et al. 2022]. Elas desempenham um papel primordial na transformação das redes móveis, permitindo a virtualização e orquestração de recursos de rede de forma eficiente e escalável, o que é essencial para a revolução baseada em software que está ocorrendo nas redes móveis.

Outra iniciativa importante para a democratização das redes de telecomunicações de futura geração é a OpenRAN (O-RAN) Alliance [Garcia-Saavedra and Costa-Pérez 2021]. No Brasil, encabeçada pela Rede Nacional de Ensino e Pesquisa (RNP), temos a OpenRAN@Brasil⁶. Esses projetos visam abrir o mercado de RAN para mais fornecedores, promovendo a concorrência e a inovação. Eles também apresentam uma arquitetura aberta que permite a construção de redes mais flexíveis e personalizáveis, fortemente baseadas em tecnologias de Inteligência Artificial, adaptando as redes às necessidades específicas de diferentes operadoras, usuários, aplicações e serviços.

As iniciativas e projetos de software de código aberto e mencionados trazem consigo uma mudança fundamental no cenário das telecomunicações e uma série de desafios e oportunidades de pesquisa. Anteriormente, apenas um pequeno grupo de engenheiros experientes das grandes empresas fornecedoras de equipamentos de redes tinham acesso ao código para atuar no desenvolvimento de protocolos padronizados. No entanto, a democratização proporcionada pelos projetos de código aberto faz com que qualquer pessoa possa contribuir efetivamente para esses projetos e influenciar seu desenvolvimento. Nesse contexto, o teste de software de redes se torna uma peça crucial para garantir que as implementações sejam corretas, conformes e robustas [Lando et al. 2023].

Os testes de conformidade desempenham um papel fundamental na garantia de que as implementações de software aberto para redes 5G e de gerações futuras estejam

²Site do projeto Open Air Interface: <https://openairinterface.org/>

³Site do projeto srsRAN: <https://www.srsran.com/>

⁴Site do projeto Open5GS: <https://open5gs.org/>

⁵Site do projeto Free5GC: <https://free5gc.org/>

⁶Site oficial da iniciativa: <https://openranbrasil.org.br/>

em conformidade com os padrões estabelecidos por organismos internacionais, como o 3GPP, responsável pelas especificações de redes móveis atuais (e.g., LTE, 5G-NR, etc.). Esses testes verificam se os protocolos e funcionalidades estão sendo implementados de acordo com as especificações técnicas, garantindo a interoperabilidade entre diferentes componentes de rede e dispositivos de usuários. Isso é essencial, uma vez que a heterogeneidade de implementações em um ambiente de código aberto pode levar a problemas de compatibilidade.

Além dos testes de conformidade, os testes de robustez são igualmente cruciais. Eles visam identificar vulnerabilidades e pontos fracos nas implementações de software, tornando-as mais resistentes a falhas, ataques e condições adversas, maliciosas ou não. Com a natureza aberta das implementações de software, é importante que a comunidade de desenvolvedores trabalhe em conjunto para identificar e corrigir vulnerabilidades, e os testes de robustez auxiliam nesse processo.

Outro aspecto essencial é a realização de testes de desempenho padronizados, também conhecidos como benchmarks. Esses testes avaliam o desempenho das implementações de software em cenários realistas de uso. Para redes privadas 5G, por exemplo, os benchmarks podem fornecer medições de latência, taxa de bits, qualidade de serviço e escalabilidade, considerando uma variedade de aplicações de interesse (e.g., vídeo, voz, jogos, IoT). Essas métricas são vitais para garantir que não apenas as implementações de software atendam às expectativas de desempenho de operadoras e usuários finais, mas também para auxiliar na identificação de gargalos das cada implementações e sua relação com as plataformas de hardware sobre as quais são implantadas.

Em termos de oportunidades de pesquisa e inovação, o campo de desenvolvimento e testes de software para redes abertas 5G e gerações futuras é vasto. Os membros do Grupo de Pesquisa em Redes de Computadores vêm se concentrando no desenvolvimento de metodologias avançadas de testes para software de redes móveis, na criação de ferramentas automatizadas de teste e na exploração de técnicas de análise de desempenho atuando em projetos como o PORVIR-5G (Programmability, ORchestration and VIRTUALization of 5G Networks)⁷. Além disso, o grupo vem instalando e aprimorando um ambiente de testes (testbed) realístico com equipamentos e sistemas para realizar pesquisas nessa área utilizando as plataformas abertas mencionadas além de tecnologias de computação em nuvem e borda.

1.7. Desafios e Oportunidades em Redes Programáveis

As infraestruturas de redes e serviços, e a própria Internet, estão passando por mudanças dramáticas, com avanços rápidos sendo feitos em tópicos como IoT, 5G e, mais recentemente, 6G. Esses sistemas prometem uma super alta taxa de transferência e baixas latências fim-a-fim, e será interessante observar sua evolução. Igualmente importante é o surgimento de aplicações como realidade virtual e aumentada, aplicações holográficas, veículos autônomos e IoT industrial, que levantam à questão de se está pronto para suportá-las e se dispõe das ferramentas necessárias para gerenciá-las. Redes programáveis têm o potencial de fornecer a flexibilidade necessária para projetar soluções de rede que atendam aos requisitos operacionais cada vez mais rigorosos desses serviços emer-

⁷Site do projeto PORVIR-5G: <https://porvir-5g-project.github.io/>

gentes. Existem oportunidades enormes para pesquisas ainda mais impactantes, tanto de natureza mais aplicada quanto fundamental. Algumas delas estão resumidas a seguir.

Avanços na computação por pacote. Realizar cálculos por pacote usando planos de dados programáveis requer um esforço significativo. As primitivas atuais são inadequadas até mesmo para tarefas de gerenciamento básicas. Superar limitações de linguagem ou hardware requer considerável criatividade. A pergunta é: como se pode progredir nessa área? Os desafios residem em compreender as demandas e formular construções que equilibrem os *trade-offs* entre flexibilidade e segurança, flexibilidade e desempenho, e outros fatores.

Equilibrar os ciclos de controle curtos (plano de dados) e longos (plano de controle). Ao projetar uma solução de gerenciamento definida por software, o desenvolvedor muitas vezes se depara com o dilema de determinar o que implementar nos planos de dados e de controle. O debate sobre o gerenciamento por delegação, que ocorreu na comunidade de pesquisa na década de 1990, permanece relevante até hoje. Pode valer a pena revisar os fundamentos desse debate e alinhá-los com a realidade tecnológica atual. Ao fazê-lo, poder-se-ia aproveitar essas valiosas contribuições para orientar essa questão.

Abstrações de linguagem adequadas para PDPs. Em nosso grupo de pesquisa, provou-se complicada a utilização das abstrações oferecidas por linguagens para planos de dados programáveis. Defende-se a disponibilização de abstrações de linguagem apropriadas, pois elas poderiam simplificar a programação do plano de dados e reduzir a probabilidade de erros de programação. Apesar dos desafios envolvidos, há espaço para revisar a pesquisa pioneira em políticas de rede (ou *intents*). Essas políticas ditariam o comportamento esperado da rede e poderiam ser refinadas para produzir código P4.

Sistemas convenientes de compilação e implantação. Uma possível área de pesquisa reside no desenvolvimento de sistemas de compilação e implantação simplificados. As ferramentas disponíveis atualmente são bastante rudimentares, tornando o processo de compilar programas para alvos de rede específicos uma tarefa complexa, devido à falta de mecanismos de depuração e solução de problemas. Da mesma forma, a implantação de programas, atualmente, é um processo relativamente *ad hoc*, dependente de interfaces pouco adequadas. Dadas essas limitações, seria interessante investigar o potencial de uma abordagem DevOps para redes programáveis, apoiada por técnicas robustas de gerenciamento de ciclo de vida de software.

Gerenciamento orientado por IA e ML na rede. A integração de técnicas de IA e ML tem recebido muita atenção para auxiliar em diferentes tarefas de gerenciamento de rede e serviços. Uma vez que os programas do plano de dados operam na taxa de linha e os dispositivos possuem portas com dezenas de gigabits por segundo, vale a pena explorar a possibilidade de transferir mecanismos baseados em ML para os dispositivos.

Quais técnicas de ML são adequadas para planos de dados? É viável utilizar métodos simplificados? Essa é uma área de pesquisa promissora.

Gerenciamento de redes híbridas. Redes híbridas, compostas por dispositivos tradicionais e programáveis, estão se tornando cada vez mais comuns. Em tais configurações, a implementação de uma solução de monitoramento baseada em Telemetria de Rede em Banda requer levar em consideração os dispositivos na infraestrutura que não podem ser programados. Para obter medições detalhadas, é crucial identificar as localizações ideais para implantar dispositivos programáveis, minimizando quaisquer possíveis “sacrifícios” na precisão. O desenvolvimento de abordagens de gerenciamento para essas redes híbridas apresenta uma área de pesquisa promissora que pode produzir resultados significativos.

De construções *ad hoc* para bibliotecas de gerenciamento reutilizáveis. O processo de implementação de procedimentos de gerenciamento normalmente envolve a criação de construções “do zero”. No entanto, há espaço para novos desenvolvimentos que permitam evoluir de construções *ad hoc* para bibliotecas de gerenciamento reutilizáveis (para planos de dados programáveis). Tal evolução facilitaria o desenvolvimento de procedimentos corretos e reutilizáveis.

Planos de dados confiáveis. A pesquisa em planos de dados confiáveis tem recebido muita atenção nos últimos anos devido à sua relevância na programação de redes. O problema de garantir a correção dos programas de rede continua é chave. A literatura apresenta inúmeras propostas de métodos e sistemas de verificação que se concentram na detecção de erros ou na garantia de conformidade com propriedades desejadas. A maioria dessas soluções depende de abordagens demoradas, como análise estática ou execução simbólica do código. No entanto, dada a natureza altamente dinâmica das redes e fluxos, ainda há espaço considerável para melhorar a escalabilidade. Além disso, a questão da segurança continua sendo uma preocupação premente. Em caso de injeção de código malicioso em um dispositivo de plano de dados, o comportamento da rede pode ser comprometido. Portanto, mais pesquisas são necessárias para aprimorar a segurança dos planos de dados programáveis.

Aplicativos conscientes de gerenciamento. Operar em um plano de dados pode proporcionar vantagens se tal funcionalidade vier acompanhada da possibilidade de acesso às informações mantidas pelas aplicações. Investigar como isso pode ser realizado de forma sistemática e modular apresenta um problema interessante para exploração adicional.

Agradecimentos

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

Referências

- [Aijaz 2020] Aijaz, A. (2020). Private 5g: The future of industrial wireless. *IEEE Industrial Electronics Magazine*, 14(4):136–145.
- [Aitken et al. 2013] Aitken, P., Claise, B., and Trammell, B. (2013). Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information. RFC 7011.
- [Alizadeh et al. 2014] Alizadeh, M., Edsall, T., Dharmapurikar, S., Vaidyanathan, R., Chu, K., Fingerhut, A., Lam, V. T., Matus, F., Pan, R., Yadav, N., and Varghese, G. (2014). Conga: Distributed congestion-aware load balancing for datacenters. In *Proceedings of the 2014 Conference of the ACM Special Interest Group on Data Communication, SIGCOMM '14*, pages 503–514, New York, NY, USA. ACM.
- [Babiarz et al. 2008] Babiarz, J., Krzanowski, R. M., Hedayat, K., Yum, K., and Morton, A. (2008). A Two-Way Active Measurement Protocol (TWAMP). RFC 5357.
- [Benson 2019] Benson, T. A. (2019). In-Network Compute: Considered Armed and Dangerous. In *Proceedings of the Workshop on Hot Topics in Operating Systems, HotOS '19*, pages 216–224, New York, NY, USA. ACM.
- [Berde et al. 2014] Berde, P., Gerola, M., Hart, J., Higuchi, Y., Kobayashi, M., Koide, T., Lantz, B., O'Connor, B., Radoslavov, P., Snow, W., et al. (2014). Onos: towards an open, distributed sdn os. In *Proceedings of the third workshop on Hot topics in software defined networking*, pages 1–6.
- [Bonati et al. 2020] Bonati, L., Polese, M., D'Oro, S., Basagni, S., and Melodia, T. (2020). Open, programmable, and virtualized 5g networks: State-of-the-art and the road ahead. *Computer Networks*, 182:107516.
- [Bosshart et al. 2014] Bosshart, P., Daly, D., Gibb, G., Izzard, M., McKeown, N., Rexford, J., Schlesinger, C., Talayco, D., Vahdat, A., Varghese, G., and Walker, D. (2014). P4: Programming Protocol-independent Packet Processors. *SIGCOMM Comput. Commun. Rev.*
- [Bosshart et al. 2013] Bosshart, P., Gibb, G., Kim, H.-S., Varghese, G., McKeown, N., Izzard, M., Mujica, F., and Horowitz, M. (2013). Forwarding metamorphosis: Fast programmable match-action processing in hardware for sdn. *ACM SIGCOMM Computer Communication Review*, 43(4):99–110.
- [Cisco Sa] Cisco (S.a.). In-band OAM (iOAM). <https://github.com/CiscoDevNet/iOAM>.
- [Cisco Networks 2023] Cisco Networks (2023). Cisco IOS NetFlow. <http://www.cisco.com/c/en/us/products/ios-nx-os-software/ios-netflow/index.html>.
- [Clark 2003] Clark, D. D. (2003). The tussle in cyberspace: defining tomorrow's Internet. *ACM SIGCOMM Computer Communication Review*, 33(1):347–356.
- [Comer 2017] Comer, D. E. (2017). *Internetworking with TCP/IP, Vol. 1: Principles, Protocols, and Architecture*. Pearson.

- [Eichholz et al. 2022] Eichholz, M., Campbell, E. H., Krebs, M., Foster, N., and Mezini, M. (2022). Dependently-typed data plane programming. *Proceedings of the ACM on Programming Languages*, 6(POPL):1–28.
- [Feamster et al. 2014] Feamster, N., Rexford, J., and Zegura, E. (2014). The Road to SDN: An Intellectual History of Programmable Networks. *SIGCOMM Comput. Commun. Rev.*, 44(2):87–98.
- [Fedor et al. 1990] Fedor, M., Schoffstall, M. L., Davin, J. R., and Case, D. J. D. (1990). Simple Network Management Protocol (SNMP). RFC 1157.
- [Foster et al. 2023] Foster, N., McKeown, N., and Rexford, J. (2023). Network programmability: The road ahead. In *The Networking Channel*. <https://networkingchannel.eu/network-programmability-the-road-ahead/>.
- [Garcia-Saavedra and Costa-Pérez 2021] Garcia-Saavedra, A. and Costa-Pérez, X. (2021). O-ran: Disrupting the virtualized ran ecosystem. *IEEE Communications Standards Magazine*, 5(4):96–103.
- [Gasparly et al. 2005] Gasparly, L., Sanchez, R., Antunes, D., and Meneghetti, E. (2005). A snmp-based platform for distributed stateful intrusion detection in enterprise networks. *IEEE Journal on Selected Areas in Communications*, 23(10):1973–1982.
- [Glebke et al. 2019] Glebke, R., Krude, J., Kunze, I., RÜth, J., Senger, F., and Wehrle, K. (2019). Towards executing computer vision functionality on programmable network devices. In *Proceedings of the 1st ACM CoNEXT Workshop on Emerging in-Network Computing Paradigms*, pages 15–20.
- [Gude et al. 2008] Gude, N., Koponen, T., Pettit, J., Pfaff, B., Casado, M., McKeown, N., and Shenker, S. (2008). Nox: towards an operating system for networks. *ACM SIGCOMM computer communication review*, 38(3):105–110.
- [Gunturi et al. 2005] Gunturi, R., Johnson, E., and Seow, C. (2005). Packet processing pipeline. US Patent App. 10/766,282.
- [Guo and McKeown 2018] Guo, Y. and McKeown, N. (2018). *Network Function Virtualization*. Morgan & Claypool.
- [Gupta et al. 2018] Gupta, A., Harrison, R., Canini, M., Feamster, N., Rexford, J., and Willinger, W. (2018). Sonata: Query-driven streaming network telemetry. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication, SIGCOMM '18*, pages 357–371, New York, NY, USA. ACM.
- [Hauser et al. 2021] Hauser, F., Häberle, M., Merling, D., Lindner, S., Gurevich, V., Zeiger, F., Frank, R., and Menth, M. (2021). A survey on data plane programming with p4: Fundamentals, advances, and applied research. *arXiv preprint arXiv:2101.10632*.
- [He et al. 2015] He, K., Rozner, E., Agarwal, K., Felter, W., Carter, J., and Akella, A. (2015). Presto: Edge-based load balancing for fast datacenter networks. In *Proceedings of the 2015 Conference of the ACM Special Interest Group on Data Communication, SIGCOMM '15*, pages 465–478, New York, NY, USA. ACM.

- [Hogan et al. 2022] Hogan, M., Landau-Feibish, S., Arashloo, M. T., Rexford, J., and Walker, D. (2022). Modular switch programming under resource constraints. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, pages 193–207.
- [Hong et al. 2013] Hong, C.-Y., Kandula, S., Mahajan, R., Zhang, M., Gill, V., Nanduri, M., and Wattenhofer, R. (2013). Achieving high utilization with software-driven wan. In *Proceedings of the 2013 Conference of the ACM Special Interest Group on Data Communication, SIGCOMM '13*, pages 15–26, New York, NY, USA. ACM.
- [Jain et al. 2013] Jain, S., Kumar, A., Mandal, S., Ong, J., Poutievski, L., Singh, A., Venkata, S., Wanderer, J., Zhou, J., Zhu, M., Zolla, J., Hözlze, U., Stuart, S., and Vahdat, A. (2013). B4: Experience with a globally-deployed software defined wan. In *Proceedings of the 2013 Conference of the ACM Special Interest Group on Data Communication, SIGCOMM '13*, pages 3–14, New York, NY, USA. ACM.
- [Jeyakumar et al. 2014] Jeyakumar, V., Alizadeh, M., Geng, Y., Kim, C., and Mazières, D. (2014). Millions of little minions: Using packets for low latency network programming and visibility. In *Proceedings of the 2014 Conference of the ACM Special Interest Group on Data Communication, SIGCOMM '14*, pages 3–14, New York, NY, USA. ACM.
- [Jin et al. 2018] Jin, X., Li, X., Zhang, H., Foster, N., Lee, J., Soulé, R., Kim, C., and Stoica, I. (2018). Netchain: Scale-free sub-rtt coordination. In *15th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 18)*, pages 35–49.
- [Jin et al. 2017] Jin, X., Li, X., Zhang, H., Soulé, R., Lee, J., Foster, N., Kim, C., and Stoica, I. (2017). Netcache: Balancing key-value stores with fast in-network caching. In *Proceedings of the 26th Symposium on Operating Systems Principles, SOSP '17*, page 121–136, New York, NY, USA. Association for Computing Machinery.
- [Kim et al. 2015] Kim, C., Sivaraman, A., Katta, N., Bas, A., Dixit, A., and Wobker, L. J. (2015). In-band network telemetry via programmable dataplanes. In *Proceedings of the 2015 ACM Symposium on SDN Research, SOSR '15*, New York, NY, USA. ACM.
- [Kim et al. 2006] Kim, Y., Lau, W. C., Chuah, M. C., and Chao, H. J. (2006). PacketScore: a statistics-based packet filtering scheme against distributed denial-of-service attacks. *IEEE transactions on dependable and secure computing*, 3(2):141–155.
- [Kumar et al. 2015] Kumar, A., Jain, S., Naik, U., Raghuraman, A., Kasinadhuni, N., Zermeno, E. C., Gunn, C. S., Ai, J., Carlin, B., Amarandei-Stavila, M., Robin, M., Siganporia, A., Stuart, S., and Vahdat, A. (2015). Bwe: Flexible, hierarchical bandwidth allocation for wan distributed computing. In *Proceedings of the 2015 Conference of the ACM Special Interest Group on Data Communication, SIGCOMM '15*, pages 1–14, New York, NY, USA. ACM.
- [Lando et al. 2023] Lando, G., Schierholt, L. A. F., Milesi, M. P., and Wickboldt, J. A. (2023). Evaluating the performance of open source software implementations of the 5g network core. In *NOMS 2023-2023 IEEE/IFIP Network Operations and Management Symposium*, pages 1–7.

- [Laupkhov and Thomas 2016] Laupkhov, P. and Thomas, J. (2016). Using int to build a real-time network monitoring system @ scale. 3rd P4 Workshop. <https://2016p4workshop.sched.com/event/6otq/using-int-to-build-a-real-time-network-monitoring-system-scale>.
- [Liu et al. 2021] Liu, P., Kim, H., Li, Y., Xu, X., and Wang, X. (2021). Network Programming and Automation: A Survey. *IEEE Communications Surveys & Tutorials*, 23(4):3029–3058.
- [Marques et al. 2020] Marques, J., Levchenko, K., and Gaspar, L. (2020). Insight: Diagnosing slo violations with in-band network telemetry. In *Proceedings of the 16th International Conference on Emerging Networking EXperiments and Technologies, CoNEXT '20*, page 421–434, New York, NY, USA. Association for Computing Machinery.
- [Marques 2022] Marques, J. A. (2022). Advancing network monitoring and operation with in-band network telemetry and data plane programmability.
- [Marques et al. 2019] Marques, J. A., Luizelli, M. C., da Costa Filho, R. I. T., and Gaspar, L. P. (2019). An optimization-based approach for efficient network monitoring using in-band network telemetry. *J. Internet Serv. Appl.*, 10(1):12:1–12:20.
- [McKeown et al. 2008] McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., and Turner, J. (2008). Openflow: enabling innovation in campus networks. *ACM SIGCOMM computer communication review*, 38(2):69–74.
- [Michel et al. 2021] Michel, O., Bifulco, R., Rétvári, G., and Schmid, S. (2021). The programmable data plane: Abstractions, architectures, algorithms, and applications. *ACM Computing Surveys (CSUR)*, 54(4):1–36.
- [Mihai et al. 2022] Mihai, R., Craciunescu, R., Martian, A., Li, F. Y., Patachia, C., and Vochin, M.-C. (2022). Open-source enabled beyond 5g private mobile networks: From concept to prototype. In *2022 25th International Symposium on Wireless Personal Multimedia Communications (WPMC)*, pages 181–186.
- [Parizotto et al. 2022] Parizotto, R., Mello, B., Haque, I., and Schaeffer-Filho, A. (2022). Netgvt: Offloading global virtual time computation to programmable switches. In *Proceedings of the Symposium on SDN Research, SOSR '22*, page 16–24, New York, NY, USA. Association for Computing Machinery.
- [Sanvito et al. 2018] Sanvito, D., Siracusano, G., and Bifulco, R. (2018). Can the network be the AI accelerator? In *Proceedings of the 2018 Morning Workshop on In-Network Computing*, pages 20–25.
- [Sapio et al. 2017] Sapio, A., Abdelaziz, I., Aldilaijan, A., Canini, M., and Kalnis, P. (2017). In-Network Computation is a Dumb Idea Whose Time Has Come. In *Proceedings of the 16th ACM Workshop on Hot Topics in Networks, HotNets-XVI*.

- [Sapio et al. 2021] Sapio, A., Canini, M., Ho, C.-Y., Nelson, J., Kalnis, P., Kim, C., Krishnamurthy, A., Moshref, M., Ports, D., and Richtarik, P. (2021). Scaling distributed machine learning with in-network aggregation. In *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21)*, pages 785–808. USENIX Association.
- [Sekigawa et al. 2022] Sekigawa, S., Sasaki, C., and Tagami, A. (2022). Toward a cloud-native telecom infrastructure: Analysis and evaluations of kubernetes networking. In *2022 IEEE Globecom Workshops (GC Wkshps)*, pages 838–843.
- [Shahbaz and Feamster 2015] Shahbaz, M. and Feamster, N. (2015). The case for an intermediate representation for programmable data planes. In *Proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Networking Research*, pages 1–6.
- [Sivaraman et al. 2015] Sivaraman, A., Budiu, M., Cheung, A., Kim, C., Licking, S., Varghese, G., Balakrishnan, H., Alizadeh, M., and McKeown, N. (2015). Packet transactions: A programming model for data-plane algorithms at hardware speed. *CoRR*, vol. abs/1512.05023.
- [Song 2013] Song, H. (2013). Protocol-Oblivious Forwarding: Unleash the Power of SDN through a Future-Proof Forwarding Plane. In *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, HotSDN '13*, page 127–132, New York, NY, USA. Association for Computing Machinery.
- [Tennenhouse and Wetherall 1996] Tennenhouse, D. L. and Wetherall, D. J. (1996). Towards an active network architecture. *ACM SIGCOMM Computer Communication Review*, 26(2):5–17.
- [Tokusashi et al. 2019] Tokusashi, Y., Dang, H. T., Pedone, F., Soulé, R., and Zilberman, N. (2019). The case for in-network computing on demand. In *Proceedings of the Fourteenth EuroSys Conference 2019*, pages 1–16.
- [Vassoler et al. 2023] Vassoler, G., Marques, J. A., and Gaspary, L. P. (2023). Vermont: Towards an in-band telemetry-based approach for live network property verification. In *NOMS 2023-2023 IEEE/IFIP Network Operations and Management Symposium*, pages 1–7.
- [Wetherall and Tennenhouse 2019] Wetherall, D. and Tennenhouse, D. (2019). Retrospective on "towards an active network architecture". *ACM SIGCOMM Computer Communication Review*, 49(5):86–89.
- [Wickboldt et al. 2015] Wickboldt, J., Jesus, W. D., Isolani, P., Both, C., Rochol, J., and Granville, L. Z. (2015). Software-Defined Networking: Management Requirements and Challenges. *IEEE Communications Magazine*, 53(1):278–285.
- [Yu et al. 2020] Yu, Z., Zhang, Y., Braverman, V., Chowdhury, M., and Jin, X. (2020). NetLock: Fast, Centralized Lock Management Using Programmable Switches. In *SIGCOMM*.

[Zekauskas et al. 2006] Zekauskas, M. J., Karp, A., Shalunov, S., Boote, J. W., and Teitelbaum, B. R. (2006). A One-way Active Measurement Protocol (OWAMP). RFC 4656.

[Zhu et al. 2015] Zhu, Y., Kang, N., Cao, J., Greenberg, A., Lu, G., Mahajan, R., Maltz, D., Yuan, L., Zhang, M., Zhao, B. Y., and Zheng, H. (2015). Packet-level telemetry in large datacenter networks. *ACM SIGCOMM Computer Communication Review (CCR)*, 45(4):479–491.

