Capítulo

# 5

# Gerência de Identidades Federadas em Nuvens: Enfoque na Utilização de Soluções Abertas

Guilherme Feliciano[1], Lucio Agostinho[1], Eliane Guimarães[2], Eleri Cardozo[1]

[1]Faculdade de Engenharia Elétrica e de Computação
Universidade Estadual de Campinas - UNICAMP

[2]Centro de Tecnologia da Informação Renato Archer
13083-970 - Campinas - SP

{gof,larocha,eleri}@dca.fee.unicamp.br, eliane.guimaraes@cti.gov.br

***Abstract***

*Cloud computing emerges as a new paradigm for the offering of services in the Web. The main idea is to transfer most of the processing and storage of user applications to a remote cloud of services. Although this approach leads to new business opportunities, the issue of security is still an open and difficult to solve problem. This is because in a cloud there are several applications available as services, many of which have their own access control systems. Furthermore, applications that support service compositions across distinct domains require authentication mechanisms that take into account this collaborative nature. This short course addresses the offering of federated services under an Identity Management perspective. The main concepts related to this subject are introduced in the context of cloud computing as well as the main solutions employed in open federated cloud environments. Finally, a case study in a field of networked robotics that uses one of the solutions discussed in this short course is presented.*

***Resumo***

*A computação em nuvem surge como um novo paradigma para a oferta de serviços na Web. A ideia principal é transferir a maior parte do processamento e armazenamento das aplicações dos usuários para uma nuvem remota de serviços. Apesar desta abordagem trazer novas oportunidades de negócios, a questão da segurança ainda é um problema em aberto e de difícil solução. Isso porque em uma nuvem há diversas aplicações*

*oferecidas como serviços, sendo que muitas delas possuem seus próprios sistemas para controle de acesso. Além disso, aplicações que suportam a composição de serviços entre domínios distintos exigem mecanismos de autenticação que levem em conta esta realidade colaborativa. Este minicurso explora a oferta de serviços em federações sob a ótica da Gerência de Identidades. São apresentados os principais conceitos relacionados a esse tema no contexto de computação em nuvem e as principais soluções abertas empregadas em nuvens federadas. Ao final, é apresentado um estudo de caso no domínio da robótica em rede que utiliza uma das soluções discutidas neste minicurso.*

## 5.1. Introdução

No final da década de 90, a Gerência de Identidades (IdM - *Identity Management*) e a segurança da informação eram tratadas separadamente. Mais especificamente, a infraestrutura de IdM se destinava à provisão de serviços, especialmente serviços centralizados de autenticação [Suess and Morooney 2009]. Nesta autenticação centralizada empregava-se tecnologias, tais como GSS-API (*Generic Security Service Application Programming Interface*), SASL (*Simple Authentication and Security Layer*) e/ou SSL/TLS (*Secure Sockets Layer/Transport Layer Security*) para o estabelecimento de um contexto de segurança entre dois pares comunicantes [Johansson 2009].

Neste cenário, organizações (empresas ou universidades) empregavam serviços de diretórios baseados em LDAP (*Lightweight Directory Access Protocol*). Esses serviços eram destinados a fornecer mecanismos de autenticação de forma centralizada, com o objetivo de facilitar a gerência deste ambiente e prover uma forma de autenticação única, conhecida como SSO (*Single Sign-On*). Como a identidade do usuário e o seu identificador geralmente eram os mesmos, um serviço de *e-mail*, por exemplo, poderia utilizar o identificador do usuário como chave para armazenar todas as suas mensagens e configurações de sua caixa postal no serviço de diretório [Johansson 2009].

Este modelo ainda é muito utilizado e funciona bem em ambientes onde há uma única organização ou um único serviço de diretório. As limitações deste modelo centralizado surgem quando as organizações precisam utilizar serviços compartilhados, como por exemplo, em cenários de B2B (*business-to-business*) ou necessitam fundir seus repositórios, como por exemplo, em fusões de empresas [Johansson 2009].

A falha deste modelo centralizado em tratar a autenticação em cenários fora do domínio de uma organização, ocorre em razão da falha do LDAP em fornecer um mecanismo de autenticação escalável, que não prenda a organização a uma solução de um fabricante específico ou que não obrigue a organização a permitir acesso externo às suas bases de dados [Johansson 2009].

Com a realidade da computação em nuvem e o surgimento da chamada Web 2.0, o cenário para a IdM tornou-se mais complexo. A Web 2.0 aprofundou os conceitos de colaboração, interoperabilidade entre aplicações e compartilhamento de informações na Internet [Shelly and Frydenberg 2010]. A nuvem surge como uma alternativa para oferecer serviços na Web de forma rápida, escalável e com custos proporcionais à demanda.

A gerência de identidades federadas (*Federated IdM*) têm como desafio oferecer

uma infraestrutura que permita tanto aos usuários uma experiência de SSO, quanto aos administradores um mecanismo de autenticação e controle de acesso aos recursos federados entre parceiros [Olden 2011]. Segundo Stallings [Stallings 2011], um ambiente de gerência de identidades federadas deve ter principalmente os seguintes elementos: autenticação, autorização, *logging*, provisionamento de usuários, automação com *workflow*, delegação, federação, SSO e mecanismo de *password reset* oferecido aos usuários.

Esta infraestrutura é formada por um sistema integrado de políticas, processos de negócios e tecnologias para o tratamento das identidades, definição, certificação e gerência do ciclo de vida das identidades, mecanismos para troca segura e validação destas informações e aspectos legais [Wangham et al. 2010]. A gerência de identidades federadas também têm como meta permitir que os usuários possam estabelecer ligações entre suas diversas identidades. Esta ligação lógica é denominada federação de identidades [Bertino and Takahashi 2011].

Este minicurso tem como principal objetivo explorar os padrões para gerência de identidades federadas, bem como as tecnologias que implementam estes padrões, com vistas à aplicação em ambientes de computação em nuvem. As soluções são apresentadas com um enfoque prático e um estudo de caso ilustra o estabelecimento de uma nuvem federada para a realização de experimentos robóticos via rede.

O capítulo está dividido em sete seções, incluindo esta. A seção 5.2 apresenta os conceitos gerais relacionados à computação em nuvem e suas técnicas fundamentais. A seção 5.3 descreve os principais conceitos e técnicas relacionados a gerência de identidades federadas. A seção 5.4 apresenta o cenário da gerência de identidades federadas em ambientes de computação em nuvem, bem como seus principais desafios e tendências da área. A seção 5.5 apresenta os principais padrões e soluções para a implantação da gerência de identidades federadas em ambientes de computação em nuvem. A seção 5.6 apresenta um estudo de caso detalhando um ambiente de computação em nuvem desenvolvido pelos autores e a motivação para a aplicação da gerência de identidades federadas neste ambiente. Por fim, a seção 5.7 resume os principais assuntos abordados e relata a experiência dos autores no uso de ferramentas para implantação de ambientes federados e integração de recursos em nuvens.

## 5.2. Visão Geral sobre Computação em Nuvem

Computação em nuvem é um modelo de computação distribuída que deriva características da computação em grades, no que diz respeito à provisão de informação sob demanda para múltiplos usuários concorrentes. Um domínio oferece aplicações na nuvem sem se preocupar com o local onde os serviços estão sediados ou como eles são oferecidos. Fatias do poder computacional dos nós da rede são oferecidas, reduzindo os custos para fornecer uma infraestrutura própria para prover os serviços. Os recursos são cedidos apenas durante o período de uso, reduzindo o consumo de energia quando a utilização não for mais necessária [Endo et al. 2010]. A virtualização fornece a tecnologia base para muitas soluções de nuvem. Além disso, em muitas soluções são oferecidos ambientes onde os usuários são capazes de escolher seus recursos virtualizados tais como linguagem de programação, sistema operacional e outros serviços

personalizados. Os principais benefícios são a redução dos custos de investimento em infraestrutura, dos custos operacionais e a escalabilidade para a provisão de serviços sob demanda [Verdi et al. 2010].

Segundo [Mell and Grace 2010] a computação em nuvem possui características, definições e atributos que ainda estão sendo elaborados. Ainda que possua o agrupamento de muitos modelos, as seguintes características próprias podem ser enumeradas:

1. Oferta de serviços sob demanda: alocação dinâmica dos serviços requisitados (*resource pooling*), sem interação humana com o provedor dos serviços.

2. Amplo acesso aos recursos computacionais: acesso por meio de diversos protocolos padronizados, para uma grande variedade de dispositivos como PCs, *laptops*, dispositivos móveis, dentre outros.

3. Transparência: o usuário não precisa conhecer a localização física dos recursos computacionais oferecidos.

4. Elasticidade: os serviços devem ser alocados e desalocados rapidamente, apenas no decorrer da requisição do usuário.

5. Gerência: a infraestrutura deve oferecer mecanismos para a gerência de recursos, de armazenagem, de processamento e largura de banda, dentre outros.

Outras características comuns de ambientes que utilizam computação em nuvem são [Endo et al. 2010]:

- Escalabilidade: a oferta de recursos sob demanda viabiliza a oferta de serviços para um número maior de usuários. Os recursos serão alocados apenas pelo período contratado, reduzindo a sub-utilização da rede de serviços. Essa característica implica na elasticidade da oferta de recursos para muitos usuários concorrentes.

- Modelo *pay-per-use*: cobrança proporcional ao uso dos recursos. A computação em nuvem é um exemplo de *utility computing* (computação vista como uma utilidade) porque a oferta desses serviços é similar a outros tradicionais, onde o usuário paga pelo fornecimento de eletricidade, água, gás natural ou serviços de telefonia [Breitman and Viterbo 2010].

- Virtualização: o usuário tem a ilusão de que interage com os recursos de um *host* real quando, na verdade, utiliza um ambiente que simula o acesso físico do *host* no qual estão hospedados.

Na computação em nuvem um provedor de serviços pode utilizar um ou mais modelos para a oferta de serviços. Ainda assim, a administração do domínio é responsável por controlar a infraestrutura, sistema operacional, servidores, operações de persistência e demais requisitos para a oferta de serviços para uma grande quantidade de usuários concorrentes. Os modelos para prestação de serviços em nuvem são os seguintes:

- IaaS (*Infrastructure as a Service*): destaca a importância da infraestrutura na provisão de serviços. O provedor é capaz de oferecer uma infraestrutura de armazenagem, processamento e demais recursos de *hardware*, tais como servidores e componentes de rede, de maneira transparente para o usuário. Exemplo de provedores: Amazon AWS [Amazon 2010] e FlexiScale [FlexiScale 2010].

- PaaS (*Platform as a Service*): destaca a importância de plataformas na provisão de serviços. O usuário é capaz de desenvolver suas próprias aplicações, respeitando o modelo de desenvolvimento da plataforma. Também são oferecidos serviços de comunicação (serviços Web, por exemplo), armazenagem e linguagens de programação. Exemplo de provedores: Ning [Ning 2010] e Microsoft Windows Azure Platform [Windows Azure 2010].

- SaaS (*Software as a Service*): o provedor de serviços habilita a execução de aplicações de uso exclusivo do usuário e/ou aplicações fornecidas pelo próprio provedor e/ou terceiros, tais como aplicativos de *e-mail* empresariais, grupos de discussão, ferramentas para edição de sites e demais aplicações que são compartilhadas por um grande número de usuários. Nesse modelo, os serviços são mantidos em um provedor de serviços e são acessíveis pela Internet. O uso de tecnologias como serviços Web, arquiteturas orientadas a serviço (*SOA - Service Oriented Architecture*) e AJAX (*Asynchronous JavaScript and XML*) impulsionaram o desenvolvimento de aplicações remotas que inclusive podem ser incorporadas a Web sites como serviços. Aliado a isso, o aumento da largura de banda entre os usuários finais e os provedores de serviços contribui para o aumento de aplicações sediadas remotamente. Exemplo de provedores: Salesforce [Salesforce 2010] e Google Apps [Google 2010].

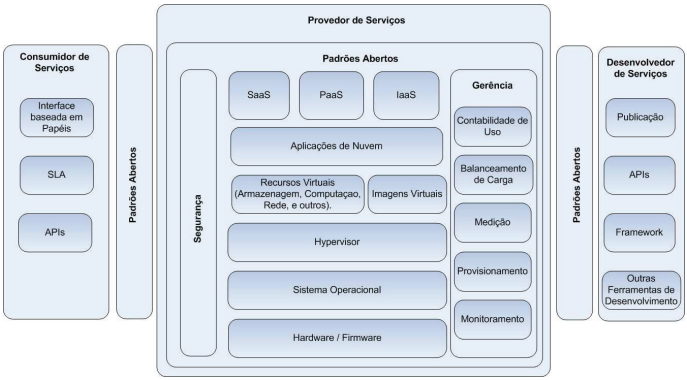


**Fig. 5.1. Visão Geral dos Principais Componentes para Ambientes de Computação em Nuvem. Adaptado de [Zappert 2010].**

A Fig. 5.1 apresenta uma visão geral dos principais componentes encontrados em ambientes de computação em nuvem. A descrição detalhada de cada um desses

componentes pode ser encontrada em [Zappert 2010]. Nessa figura destaca-se os relacionamentos entre os elementos Provedor de Serviços, Consumidor de Serviços e Desenvolvedor de Serviços, os quais são interligados por tecnologias de comunicação baseadas em padrões abertos. No Provedor de Serviços é observado um nível crescente de abstração em cada uma das camadas. Por exemplo, infraestruturas de nuvem são altamente dependentes do tipo de *hardware* escolhido, além do sistema operacional base de toda a infraestrutura. O suporte a virtualização é oferecido pelo componente Hypervisor que é mantido sobre o sistema operacional. A virtualização de recursos de *hardware* e *software* oferece a base para que um amplo conjunto de aplicações de nuvem sejam oferecidas de várias formas diferentes (*aaS - *Everything as a Service*).

O componente de Gerência deve incluir funcionalidades para contabilidade de uso, balanceamento de carga, medição de tráfego e uso de recursos, provisionamento de máquinas virtuais e demais funções de monitoramento. O componente Segurança deve incluir funcionalidades tanto para a segurança do tráfego de dados, quanto para a segurança relacionada ao acesso ao ambiente. Tanto os componentes de gerência quanto os componentes de segurança devem abordar desde as camadas inferiores do *hardware/firmware* até as camadas superiores no nível das aplicações e infraestrutura de suporte. O Consumidor de Serviços tem acesso aos recursos por meio de interfaces baseadas em regras de acesso compatíveis com os mecanismos de segurança da nuvem. Além disso, acordos de nível de serviço (SLAs - *Service Level Agreements*) são necessários para garantir a qualidade da oferta de serviços e disponibilidade de acesso, o qual é realizado por meio de interfaces de programação para aplicações (APIs - *Application Programming Interfaces*) que se comunicam com os serviços oferecidos por meio de interfaces públicas.

O Desenvolvedor de Serviços utiliza uma gama de serviços para desenvolver novas aplicações para a nuvem. Esses serviços utilizam padrões abertos para se comunicar e estender as funcionalidades dos serviços já existentes. O desenvolvimento de novos serviços demanda o uso de *frameworks* próprios para o desenvolvimento e demais ferramentas de suporte, bem como a publicação dos mesmos com o auxílio de APIs (componentes Publicação e APIs).

A infraestrutura da nuvem é oferecida em diversos níveis de abrangência e disponibilidade, de acordo com as finalidades da organização e de seus usuários. Cada um desses modelos admite que uma entidade terceirizada mantenha a nuvem: a) nuvem privada: centrada no domínio de uma organização; b) nuvem pública: acessível pela Internet, geralmente para uso público em geral; c) nuvem comunitária: nuvens compartilhadas entre várias organizações com finalidades em comum; d) nuvem híbrida: uma combinação das anteriores. O uso desse último modelo é uma alternativa para estender o *pool* de recursos utilizando serviços de outros provedores de nuvem. Nesses casos, a organização utiliza os serviços de uma nuvem pública para prover a maioria das funcionalidades para seus clientes, mas os dados restritos são mantidos e gerenciados em *datacenters* particulares na organização. Outra funcionalidade de destaque é a possibilidade de migração de serviços para redução de custos durante o ciclo de vida das máquinas virtuais (*live migration*) para cumprir os requisitos de armazenagem, desempenho, capacidade de processamento, largura de banda e demais quesitos relacionados a qualidade de serviço (QoS - *Quality of Service*). Isso sugere que

relacionamentos de confiança precisam assegurar SLA entre os domínios parceiros.

De acordo com essa análise, com os recentes avanços das soluções de virtualização e de computação em nuvem, a interoperabilidade será um fator fundamental para manter a cooperação mútua entre provedores de nuvem. Nesse cenário, uma federação de nuvens híbridas é uma solução onde cada domínio é capaz de expandir seus recursos virtualizados sob demanda, requisitando mais recursos computacionais de outros domínios federados, quando necessário. A gerência de identidades federadas aplica-se a esse cenário de forma a gerenciar o uso seguro de recursos por parte dos usuários e sistemas de sites parceiros.

### 5.2.1. Técnicas Fundamentais em Computação em Nuvem

**Virtualização**

Em essência, a virtualização consiste em imitar um comportamento, seja por extensão ou substituição de um recurso por outro [Carissimi 2008]. A virtualização também é definida como um sistema ou um método de dividir os recursos computacionais em múltiplos ambientes isolados [OpenVZ 2010]. O conceito de virtualização remonta à virtualização de recursos em sistemas operacionais. Soluções de alto nível como interfaces gráficas, bibliotecas e APIs são exemplos de recursos de *software* que tornam transparente para o usuário o acesso aos recursos de *hardware*, em especial, o acesso aos periféricos de entrada e saída. Ou seja, cria-se a ilusão no sistema operacional de que se tem a interação direta com os recursos de *hardware*. Diz-se também que a virtualização é uma metodologia para dividir os recursos de um computador em múltiplos ambientes de execução conhecidos como máquinas virtuais, aplicando conceitos de particionamento, *time-sharing*, simulação completa ou parcial de máquina, emulação, QoS, entre outros [Carissimi 2008].

Portanto, a virtualização é uma técnica para ocultar características físicas de recursos computacionais, de forma que os sistemas, aplicações e *end users* interajam com esses recursos. Nesta técnica, um único recurso físico, como um servidor, dispositivo de armazenagem ou sistema operacional, passa a ser visto como múltiplos recursos lógicos.

A virtualização remonta às décadas de 60 e 70. Com o uso de máquinas virtuais era possível executar e migrar aplicações legadas entre plataformas distintas, desde que houvesse uma versão da máquina virtual para a plataforma alvo. A principal motivação era ampliar e melhorar a utilização e o compartilhamento de recursos nos *mainframes* [Carissimi 2008].

Com a redução do custo do *hardware* em meados da década de 80, ocorreu uma mudança do foco de processamento centralizado em *mainframes* para o processamento distribuído em microcomputadores. O modelo cliente-servidor foi estabelecido para a computação distribuída, reduzindo a necessidade da virtualização para a integração de recursos computacionais. A redução do custo de aquisição do *hardware* e do compartilhamento de informações tornaram a virtualização pouco explorada por alguns anos [Menascé 2005].

Apenas em meados da década de 90, com o aumento do poder de processamento do *hardware* e dos computadores pessoais (PCs), a virtualização voltou a ganhar

destaque em produtos tais como o VMware [VMware Inc. 2011], User Mode Linux (UML) [Dike 2006], Xen [Xen 2010], KVM [Warnke and Ritzau 2010] e VirtualBox [Oracle 2010]. De certa forma, esses produtos trazem o conceito de virtualização como uma alternativa para executar diversos sistemas operacionais sem a necessidade de se aumentar proporcionalmente o número de *hosts* físicos que os mantêm. Isso implica em reduzir os custos relativos à aquisição de *hardware*, infraestrutura física, consumo de energia, ventilação, suporte e manutenção de vários *hosts*.

A virtualização é recomendada para consolidar múltiplos servidores em um mesmo *host*, isolar diferentes aplicações de usuários em um mesmo *host*, executar/depurar *softwares* e sistemas operacionais construídos para uma arquitetura em outra, além de simplificar a instalação de infraestruturas de *software* em diferentes domínios e testar aplicativos em *hardwares* não existentes.

O sistema operacional que executa o *software* de virtualização é denominado hospedeiro (*host*). O sistema operacional virtualizado é denominado convidado (*guest*). Múltiplos sistemas operacionais convidados podem executar no mesmo hospedeiro, sem interferência entre eles. Uma máquina virtual (VM - *Virtual Machine*) é uma camada de *software* que simula um computador real (físico) e que é capaz de executar instruções como se fosse a máquina física. O núcleo do sistema operacional hospedeiro fornece bibliotecas para suportar múltiplas máquinas virtuais. Sistemas operacionais convidados executam sobre máquinas virtuais.

### *SOC - Service Oriented Computing*

A computação em nuvem supre as necessidades de provedores de serviços na Internet quanto à maneira de se aumentar a capacidade de processamento sob demanda, sem a necessidade de se ampliar os investimentos na própria infraestrutura. Além disso, reduz os custos com o treinamento de pessoal e aquisição de licenças adicionais de *software*. Muitas soluções de nuvem utilizam serviços Web para disponibilizar interfaces Web (APIs) para acesso aos seus serviços. Serviços Web seguem a arquitetura SOA que é baseada em um modelo cliente/servidor em que o cliente faz o papel de requisitante de serviços e o servidor de provedor de serviços. A comunicação é baseada na troca de mensagens síncronas ou assíncronas independentes do protocolo de transporte utilizado [Ma 2005]. A Fig. 5.2 ilustra uma visão geral da arquitetura SOA.

A computação orientada a serviços (SOC - *Service Oriented Computing*) define um conjunto de princípios, modelos arquiteturais, tecnologias e *frameworks* para o projeto e desenvolvimento de aplicações distribuídas. Segundo [Gonçalves et al. 2011], orientação a serviço é um paradigma que propõe a criação de unidades lógicas (ou serviços) bem definidas que podem ser utilizadas coletivamente e repetidamente.

Na arquitetura SOA as aplicações distribuídas utilizam os serviços como blocos de construção [IBM 2011]. Para simplificar a localização desses serviços em provedores distintos, SOA integra um componente para o registro e a descoberta. Com isso, os provedores registram os seus serviços em um repositório centralizado. O registro descreve as informações de cada serviço e a localização do provedor que os mantêm. Quando o cliente deseja utilizar um serviço, ele faz uma consulta nesse repositório e

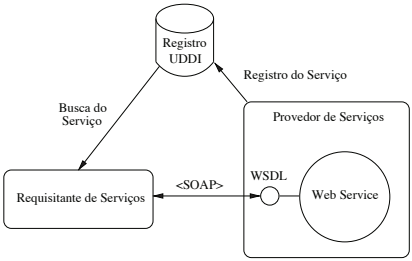


**Fig. 5.2. Visão Geral da Arquitetura Orientada a Serviço.**

informa quais os requisitos desejados. O repositório então devolve uma lista de serviços que satisfazem os requisitos solicitados. Após a escolha do serviço, o cliente acessa diretamente o provedor de serviços com base na informação de sua localização. Essa localização dinâmica pode ser feita no momento da execução do aplicativo. UDDI (*Universal Description, Discovery, and Integration*) especifica um método padrão para publicação e descoberta de componentes de *software* na Web segundo a arquitetura SOA [OASIS 2006]. Interfaces de serviços são descritas em WSDL (*Web Services Description Language*) e o protocolo SOAP (*Simple Object Access Protocol*) é empregado na interação cliente/servidor. WSDL e SOAP são padrões baseados em XML (*Extensible Markup Language*).

Em SOA, as unidades de *software* podem ser desenvolvidas separadamente. Tais unidades encapsulam a lógica da implementação do serviço e expõem apenas a interface de acesso à funcionalidade. Isso torna possível o desenvolvimento de aplicações distribuídas com fraco acoplamento. Além disso, a possibilidade de reuso de código e de disponibilidade na Web permite o desenvolvimento de aplicações em ambientes colaborativos distintos.

### Datacenters

Infraestruturas de virtualização para nuvens geralmente utilizam um ou mais *datacenters*. Os *datacenters* são infraestruturas formadas por componentes que fornecem capacidades em larga escala para processamento, armazenagem e serviços de rede para uma ou mais organizações [Veras 2009]. *Datacenters* podem ser agrupados nas categorias de PDC (*Private Data Center*) e IDC (*Internet Data Center*). Um PDC é voltado para empresas privadas, instituições ou órgãos governamentais, com a finalidade de processar informações internas. Um IDC, por outro lado, geralmente é mantido por um provedor de serviços de telecomunicações ou informação. Um *datacenter* agrupa de centenas a milhares de componentes, desde *switches* e roteadores, a poderosos servidores, balanceadores de carga e dispositivos de armazenagem. Essas características exigem que a infraestrutura tenha suporte adequado para o funcionamento, incluindo a provisão adequada de energia, ventilação/ar condicionado, segurança, monitoramento e redundância de dados. Além disso, equipamentos especializados requerem configuração

e treinamento, o que encarece os custos para a provisão de uma infraestrutura particular [Gonçalves et al. 2011].

Em ambientes de computação em nuvem, o uso de *datacenters* é uma solução adequada para o oferecimento de ambientes virtualizados, visando o desenvolvimento e rápido provisionamento de aplicações empresariais. Empresas não precisam empregar grandes quantias para a aquisição e configuração de infraestruturas de tecnologia que mudam rapidamente, mas sim utilizar infraestruturas pré-configuradas de ambientes, como os da Amazon ou IBM, por exemplo. O custo para o processamento e armazenagem de dados pode reduzir ou ampliar, conforme a demanda.

### 5.2.2. Sistemas Abertos para Gerência de Nuvens

Atualmente existem diversas soluções de código aberto para a gerência de recursos virtuais em nuvens. Uma classificação detalhada dos principais ambientes é apresentada em [Endo et al. 2010]. Dentre estas soluções destaca-se a plataforma XCP (*Xen Cloud Platform*) [Xen 2010]. A plataforma inclui o agrupamento e o isolamento de recursos de *hardware* e rede, provisionamento de sistemas, APIs para acessar os recursos virtuais e compatibilidade com um grande número de sistemas operacionais. XCP herda características dos ambientes de virtualização para-virtualizados dos produtos Xen.

O projeto Nimbus [Nimbus 2011] disponibiliza um *toolkit* de código aberto para oferecer um ambiente de *cluster* como uma IaaS. Esse *toolkit* oferece interfaces WSDL para o serviço Amazon EC2, Amazon EC2 Query API, e WSRF (*Web Services Resource Framework*) para aplicações em grades. Também são oferecidas interfaces que seguem o padrão arquitetural REST (*Representational State Transfer*) para o acesso à base de dados de recursos e escalonamento de VMs por meio de uma interface de gerência. A implementação da virtualização é baseada em Xen e KVM (*Kernel Virtual Machine*).

O projeto OpenNebula [OpenNebula 2010] oferece um *toolkit* de código aberto para a provisão de nuvens públicas, privadas e híbridas. A infraestrutura do sistema oferece recursos sob demanda para usuários finais, e foi projetada para ser integrada com outras soluções de armazenagem e rede. As VMs são utilizadas em um *pool* de recursos e toda a alocação de recursos é baseada em políticas. O ambiente virtualizado é acessível por meio de interfaces OCCI (*Open Cloud Computing Interface*), definidas pelo Open Grid Forum. Além disso, OpenNebula oferece interfaces para a EC2 Query API da Amazon, através da interface Web OpenNebula Sunstone e um subconjunto de chamadas da vCloud API da VMware. A infraestrutura tem suporte para Xen, KVM/Linux e VMware.

Eucalyptus [Murari 2010] é uma plataforma de *software* de código aberto para a criação e gerência de nuvens públicas e privadas, que possui APIs interoperáveis com as APIs da plataforma Amazon EC2/S3. A empresa Canonical, mantenedora da distribuição Ubuntu, adotou inicialmente o Eucalyptus, juntamente com outros *softwares* de código aberto, como solução de nuvem para o Ubuntu Server Edition. Essa solução ficou conhecida como Ubuntu Enterprise Cloud (UEC). Eucalyptus suporta o uso de Xen, KVM e VMware. A versão UEC, porém, suporta apenas o uso do KVM. Recentemente, a Canonical demonstrou interesse no uso de outra solução para nuvem, conhecida como OpenStack [Rackspace 2010].

É interessante notar que nenhuma dessas soluções contempla funcionalidades necessárias para gerenciar identidades de usuários pertencentes a múltiplas nuvens. Geralmente as iniciativas adotadas para a criação de federações contemplam o uso de outras infraestruturas específicas para essa tarefa, ou seja, infraestruturas para gerência de identidades federadas são agregadas como serviços adicionais ao ambiente de computação em nuvem.

## 5.3. Gerência de Identidades: Conceitos e Técnicas

O conceito de identidade, segundo [Cao and Yang 2010], é difícil de precisar, uma vez que a definição de identidade está relacionada ao ambiente onde é empregada, a contextos semânticos e a casos de uso. Como uma definição mais geral, pode-se dizer que uma identidade é uma representação de uma entidade ou sujeito que seja suficiente para identificar esta entidade em um contexto particular [El Maliki and Seigneur 2007]. Uma entidade, por sua vez, é qualquer coisa existente no mundo real. Como exemplos de entidades temos uma pessoa, um *host* ou uma aplicação. Em geral, a relação de cardinalidade de uma entidade é que ela pode ter múltiplas identidades. De acordo com a norma ITU-T Y.2720 [ITU-T 2009], uma identidade pode consistir de:

- Identificador: conjunto de dígitos, caracteres e símbolos ou qualquer outra forma de dados usada para identificar unicamente uma identidade. Podem ser delimitados pelo tempo e/ou espaço. Por exemplo, uma URL (*Uniform Resource Locator*) é única ao longo do tempo. Como exemplo de identificadores temos CPF, RG, número de matrícula e número de passaporte.
- Credenciais: uma credencial é um atestado de qualificação, competência ou autoridade, expedida por terceiros com autoridade relevante ou competência para tal ato e que atesta a veracidade da identidade. Na computação, exemplos de credenciais incluem certificados digitais X.509 assinados por uma autoridade certificadora (CA - *Certificate Authority*), senha, asserções SAML (*Security Assertions Markup Language*), dentre outros.
- Atributos: um conjunto de dados que descreve as características fundamentais de uma identidade. Como exemplo temos: nome completo, domicílio, data de nascimento e papéis (*roles*).

A Fig. 5.3 ilustra a relação entre os componentes de uma identidade na notação UML (*Unified Modeling Language*).

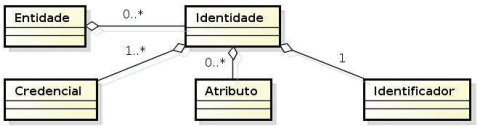


**Fig. 5.3. Relação entre os Componentes da Identidade.**

### 5.3.1. Ciclo de Vida da Identidade

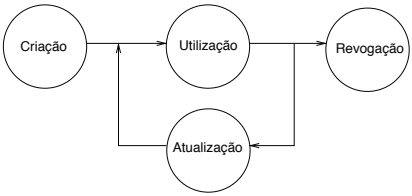


**Fig. 5.4. Ciclo de Vida da Identidade.**

A Fig. 5.4 ilustra a ciclo de vida de uma identidade e a relação entre as suas fases. De acordo com esta figura o ciclo de vida de uma identidade pode ser categorizado em [Bertino and Takahashi 2011]:

- Criação: a criação de uma identidade está relacionada com a provisão da infraestrutura de tecnologia da informação e comunicação (TIC). É necessário um mecanismo para automatizar a sua criação mediante informação de dados de um usuário e para armazenar estas em um repositório (por exemplo, banco de dados). Esta fase possui três subfases:
  - Verificação dos atributos. Um atributo é verificado por uma autoridade confiável por parte dos destinatários destes atributos. Por exemplo, em situações onde o cadastro em um site requer atributos obrigatórios, tais como RG e CPF. Há casos também onde este passo não ocorre. Por exemplo, em cadastros de *blogs*, onde os atributos requisitados não requerem nenhuma verificação.
  - Emissão da credencial. Após a verificação dos atributos, as credenciais são emitidas. Elas podem ser emitidas por alguma autoridade, pelo próprio sujeito ou pela entidade onde ocorreu o cadastro. As credenciais podem ser de várias formas (certificados digitais, *passwords*, entre outros). É importante ressaltar que cada tipo de credencial implica em um nível de confiança sobre a mesma.
  - Formação da identidade. A identidade é formada pelos atributos verificados, as credenciais emitidas e identificadores que são atribuídos por terceiros ou pelo próprio sujeito.
- Utilização: uma vez que a identidade foi criada, ela pode ser utilizada por vários sistemas e serviços. Neste caso mecanismos para autenticar e autorizar são ações geralmente executadas. Pode também incluir outras ações, tais como bilhetagem. O uso da identidade implica na utilização de forma segura e privativa, principalmente em ambientes federados onde os pares comunicantes devem ser capazes de descobrir, distinguir e autenticar identidades de forma confiável.
- Atualização: a fase de atualização envolve a alteração de valores de atributos já existentes (por exemplo, um novo endereço) ou a inserção de novos atributos para

refletir novas políticas ou regras de negócio. Com a alteração da identidade é necessário também que os sistemas de IdM permitam que estes novos parâmetros sejam propagados para domínios federados. Outra questão interessante é deixar alguns serviços mais procurados à disposição do usuário. Um exemplo é o serviço de *password reset*, que pode ser oferecido ao usuário que esqueceu ou teve a sua senha bloqueada.

- Revogação: identidades e credenciais devem ser canceladas se estas ficarem obsoletas (por exemplo, expirou a data de validade) ou inválidas. Essa fase é importante para manter o sistema de IdM atualizado.

### 5.3.2. Autenticação e Controle de Acesso (Autorização)

O processo de verificar a existência de uma identidade é chamado de autenticação [Stallings 2011]. Para que tal processo se inicie é necessário que o usuário forneça uma credencial válida para o método de autenticação utilizado pela entidade autenticadora. De posse da credencial válida, a entidade autenticadora poderá verificar em sua base de identidades se há alguma identidade cuja credencial seja a fornecida pelo usuário. Caso a entidade autenticadora encontre alguma entrada referente à credencial, o usuário é autenticado. Com isso entende-se que o usuário provou ser quem ele informou [Stallings 2011].

Uma vez realizada a autenticação do usuário, a entidade autenticadora poderá verificar qual ou quais são as permissões de acesso ao recurso originalmente requisitado. Este processo de verificar as permissões de acesso a um determinado recurso denomina-se autorização [Stallings 2011].

### 5.3.3. Modelos para Gerência de Identidades

Os modelos para IdM são classificados de acordo com a sua arquitetura. O modelo mais simples é conhecido como modelo isolado ou tradicional. Neste modelo não há divisão de responsabilidade e um único servidor é responsável pelo tratamento de todo o ciclo de vida da identidade e da autenticação e autorização. Uma desvantagem deste modelo é que o usuário precisa se lembrar da identidade que possui em cada serviço que este usuário acessa. Isto torna a experiência do usuário tediosa. Para o provedor de serviços isto representa um custo maior para gerenciar sua infraestrutura de IdM [Wangham et al. 2010].

Para amenizar os problemas existentes no modelo isolado, surge o modelo centralizado. Neste modelo cliente-servidor, há o conceito de provedor de identidades (IdP - *Identity Provider*) e provedores de serviço (SPs - *Service Providers*). Os provedores de serviço não armazenam as identidades dos usuários em seus sistemas e também delegam a etapa da autenticação para um provedor de identidades. Neste modelo todos os SPs utilizam os serviços de identidade oferecidos por um único IdP. Assim, é possível para um usuário que possua uma identidade em um site na Web e que utiliza este modelo, realizar SSO. Com o mecanismo de SSO, o usuário apresenta seu identificador e credenciais apenas uma vez no IdP parceiro do site. O SP, por utilizar o mesmo IdP, pode confiar na autenticação prévia do usúario e assumir que o usuário esteja autenticado sem necessitar de outro processo de verificação das credenciais.

Sistemas tais como Kerberos, PKI (*Public Key Infraestructure*), CAS (*Central Authentication Service*) e Microsoft Passport Network são comunente utilizados em modelos centralizados [Cao and Yang 2010]. Apesar deste modelo oferecer uma separação de responsabilidades, permitindo o SSO, ainda assim oferece desvantagens do ponto de vista da escalabilidade (o IdP é um ponto de falha), da privacidade (o IdP pode ter controle total sobre a identidade) e do fato que os SPs e IdP necessitam estar no mesmo domínio administrativo.

O modelo federado é uma evolução do modelo centralizado. Neste modelo, os SPs e o IdP podem estar em domínios administrativos diferentes. O SSO, no caso federado, também pode ser realizado entre domínios administrativos diferentes (denominado *Cross Domain SSO*). No modelo federado é possível também existir vários IdPs, com uma relação MxN entre IdPs e SPs.

Uma federação representa um conjunto de organizações que cooperam entre si de acordo com regras de confiança pré-estabelecidas para a autenticação de usuários e compartilhamento de recursos [Switch 2011]. Para tal, estas unem-se através de círculos de confiança (CoT - *Circles of Trust*). Uma solução de federação é considerada desejável quando organizações crescem com a aquisição de novos sites, para a manutenção distribuída de repositórios e para simplificar a autenticação e autorização de usuários a recursos e aplicações entre domínios parceiros [Ping Identity 2010a]. As entidades a seguir são normalmente encontradas em uma federação:

- Provedor de serviço: é a entidade mais próxima ao recurso do domínio em questão. É responsável por verificar as permissões de acesso aos recursos por usuários autenticados. Como exemplo, o SP poderia ser um provedor, que oferece serviços de nuvem para seus clientes.
- Provedor de identidades: é um provedor de serviços de identidades. Sua responsabilidade é manter a base de dados de usuários do domínio e validar as credenciais de usuários. Como exemplo, pode ser uma empresa que gerencia contas para um grande número de usuários que precisam de acesso seguro a transações bancárias. Por meio dessa entidade os usuários fornecem as suas credenciais para acessarem os recursos federados.
- *IdP proxy*: é a entidade responsável por interrogar o usuário a respeito de seu domínio de origem. O domínio de origem é o local onde o usuário possui uma identidade em um IdP. Geralmente, o IdP *proxy* apresenta uma lista de IdPs para o usuário selecionar o mais apropriado.

As duas primeiras entidades são geralmente encontradas em federações onde os participantes estão localizados em um mesmo domínio administrativo. Neste caso dizemos que a federação é do tipo intra-domínio e ocorre entre as diferentes aplicações do domínio. Em federações cujos elementos estão dispostos em domínios administrativos distintos, dizemos que a federação é do tipo inter-domínios. Neste caso, e dependendo da arquitetura adotada, o elemento IdP *proxy* atua como uma ponte entre esses domínios.

Por fim, o modelo centrado no usuário, uma evolução do modelo federado, tem como objetivo resolver uma das principais críticas em relação aos modelos anteriores,

que diz respeito à identidade do usuário, ou seja, esta fica armazenada nos provedores de identidades. Estes podem ter total controle sobre estas informações. O modelo centrado no usuário visa então oferecer mecanismos para que um usuário possa escolher que tipo de informações deseja liberar para um determinado provedor de identidades. Exemplos de soluções que utilizam o modelo centrado no usuário são: OpenID [OpenID Foundation 2011], Microsoft CardSpace [Microsoft 2011] e o Projeto Higgins [Eclipse Foundation 2011]. A Fig. 5.5 apresenta os modelos para a IdM.

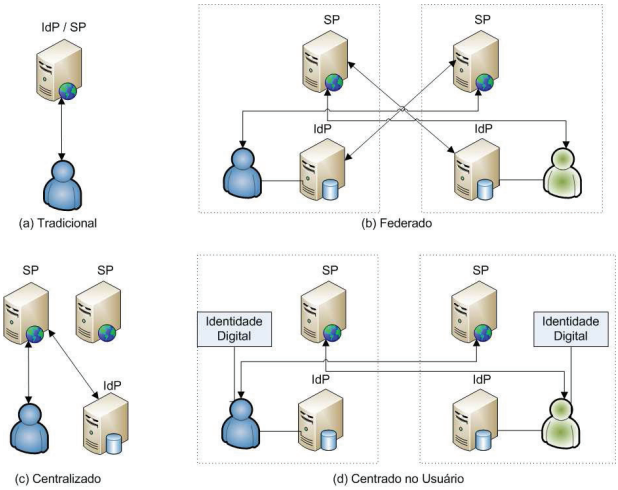


**Fig. 5.5. Modelos para a IdM. Baseado em [Wangham et al. 2010]**

### 5.3.4. Mecanismos para Localização de Provedores de Identidades

Em arquiteturas cujo modelo é federado ou centrado no usuário há a problemática da localização de provedores de identidades para a autenticação do usuário. Nestas arquiteturas, há a possibilidade de haver diversos provedores de identidades, e estes, por sua vez, podem estar espalhados em domínios administrativos diferentes. Há duas técnicas empregadas pelos padrões em gerência de identidades federadas:

- Localização baseada em um serviço: nesta técnica utiliza-se um provedor de serviço de localização de identidades. Este serviço conhece todos os possíveis provedores de identidade alcançáveis a partir do provedor. No processo de autenticação de um usuário, este serviço será invocado para apresentar uma lista com os possíveis provedores de identidades conhecidos e confiáveis. O usuário então escolhe um provedor de identidades (onde ele possui uma identidade) e então é redirecionado para este provedor. Caso o usuário não tenha nenhuma identidade em nenhum provedor conhecido, este usuário deverá criar uma identidade. Esta abordagem é utilizada pelo padrão SAML, no perfil *IdP Discovery*.

- Localização baseada em atributos da identidade: esta abordagem utiliza um atributo existente na própria identidade para descobrir onde está o provedor de identidades que autentica esta identidade. O padrão OpenID utiliza como identificador da identidade o formato URL.

### 5.3.5. Opções para Implantação de SSO

A implantação de sistemas para IdM geralmente envolve a instalação de um *middleware* que ficará responsável pelos serviços de identidade (autenticação, autorização, entre outros). Uma questão importante no processo de implantação de sistemas para IdM diz respeito ao SSO. Principalmente se o SSO envolver diferentes domínios, questões relacionadas à infraestrutura desses domínios, que desejam ingressar na federação, devem ser levadas em consideração. As seguintes opções para a implantação de SSO devem ser avaliadas [Bertino and Takahashi 2011]:

#### Baseada em *Broker*

Nesta abordagem, há um servidor central responsável por autenticar os usuários (sujeitos) e emitir uma credencial em forma de *ticket*. Através destes *tickets* é possível ao usuário obter acesso aos recursos protegidos. O Kerberos é um exemplo de implantação de SSO que utiliza a abordagem baseada em *broker*. Esta abordagem está relacionada com o modelo centralizado de IdM. A Fig. 5.6 ilustra esta abordagem.

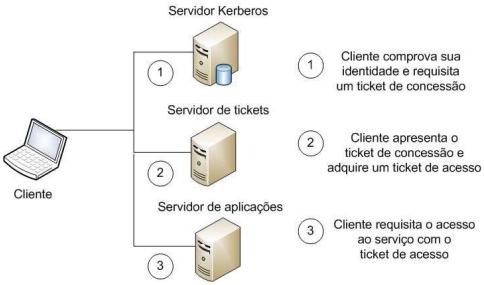


**Fig. 5.6. Processo de SSO que Utiliza Abordagem Baseada em *Broker*.**

#### Baseada em Agentes

Esta abordagem utiliza um filtro instalado em um servidor de aplicação Web. Este filtro intercepta todas as requisições que passam neste servidor (passo 1) e, com base em regras pré-estabelecidas, redireciona a requisição para um servidor de autenticação (provedor de identidades, passo 2). Após a autenticação bem sucedida (passo 3), o agente verifica se há alguma credencial válida e, se houver, libera o acesso ao recurso solicitado (passo 4). A Fig. 5.7 ilustra esta abordagem.

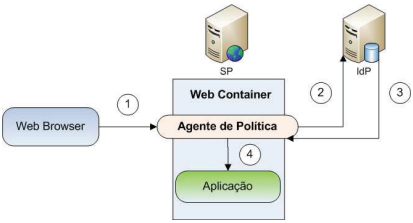


**Fig. 5.7. Processo de SSO que Utiliza Abordagem Baseada em Agente.**

### Baseada em *Proxy* Reverso

Esta abordagem utiliza um *proxy* localizado na borda da rede ou em uma zona desmilitarizada (DMZ). Através deste *proxy*, requisições de acesso a recursos (por exemplo, aplicações) são filtradas (com o uso de um agente, passo 1) e o usuário é redirecionado para um servidor de autenticação (provedor de identidades, passo 2). Após a autenticação bem sucedida, o usuário é redirecionado de volta para o *proxy* (passo 3), que por sua vez estará apto a liberar o acesso redirecionando o usuário para o recurso protegido (passo 4). Nesta abordagem, é possível através do *proxy*, o redirecionamento para qualquer domínio ao qual este tenha alcance. Um problema com esta abordagem é que há uma perda de eficiência uma vez que todas as requisições passam pelo mesmo *proxy*. Uma vantagem é que não é necessário instalar componentes adicionais nos servidores de aplicação para protegê-los. A Fig. 5.8 mostra esta abordagem.

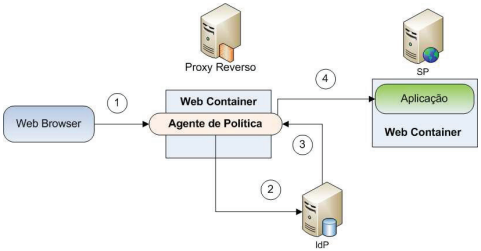


**Fig. 5.8. Processo de SSO que Utiliza Abordagem Baseada em *Proxy* Reverso.**

### Baseada em API

Nesta abordagem há a inclusão de uma API relacionada com o *middleware* para IdM nas aplicações a serem protegidas. Este mecanismo é o mais intrusivo de todos, uma vez que cada aplicação necessita chamar as primitivas oferecidas pela API. Um fator positivo é que nesta abordagem é possível obter um nível de controle de acesso mais sofisticado se comparado com as demais abordagens. No passo 1, a requisição para acessar a aplicação Web é interceptada pela API que está na própria aplicação. No passo 2, a API redireciona

o *browser* para o provedor de serviço de identidades. No passo 3, após a autenticação bem sucedida, este provedor redireciona o *browser* do usuário para a API. No passo 4, a API informa à aplicação que o usuário está autenticado e a aplicação segue o seu fluxo de execução. A Fig. 5.9 ilustra esta abordagem.

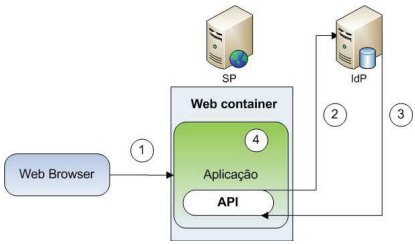


**Fig. 5.9. Processo de SSO que Utiliza Abordagem Baseada em API.**

**Baseada em Serviços**

Esta abordagem é parecida com a abordagem baseada em API. A diferença é que neste caso as chamadas são realizadas remotamente (passos 2 e 3), pois são oferecidas diretamente pelo provedor de serviço de identidades. A Fig. 5.10 mostra esta abordagem.

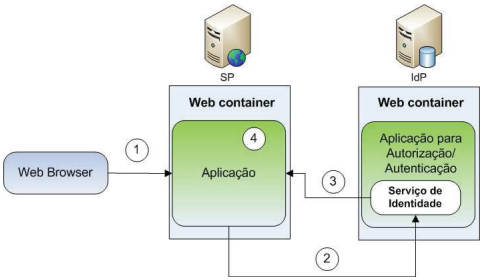


**Fig. 5.10. Processo de SSO que Utiliza Abordagem Baseada em Serviços.**

### 5.3.6. Auditoria

O propósito da auditoria para a IdM é auxiliar a resolução de questões relacionadas com o inventário de identidades. A auditoria deve ser realizada periodicamente, por exemplo uma vez ao ano. No processo de auditoria, o inventário é atualizado visando oferecer informações para a avaliação de riscos referente às políticas de segurança, privacidade, entre outras [Windley 2005].

Um desafio importante para que a auditoria funcione em um ambiente de computação em nuvem diz respeito ao problema da falta de visibilidade no acesso de

aplicações oferecidas como SaaS. Este fato é uma consequência dos usuários estarem acessando as aplicações através de uma rede pública (por exemplo, a Internet) ao invés de estarem conectados em uma rede privada (por exemplo, a rede de uma universidade ou empresa). Isto prejudica o uso de ferramentas de monitoramento de rede, uma importante aliada da auditoria [Olden 2011].

#### 5.3.7. Privacidade

O termo privacidade é definido como o direito de uma entidade (sujeito) em determinar o grau de interação que suas informações pessoais (atributos de sua identidade) devem ter perante o contexto onde a identidade está inserida. Este contexto inclui o grau de comprometimento na divulgação destas informações a terceiros [Júnior et al. 2010]. No âmbito da IdM, tendo em vista que as identidades estão logicamente inter-relacionadas entre os diferentes domínios de segurança, a privacidade pode ser vista como a possibilidade de um sujeito determinar quais informações (atributos) de sua identidade serão divulgadas no processo de federação de forma a manter o anonimato entre as diferentes identidades na federação. Uma técnica para a preservação da privacidade em IdM é utilizar pseudônimos. Pseudônimos são identificadores que não permitem inferências em relação à identidade real (e seus atributos) do usuário [Wangham et al. 2010]. Os pseudônimos podem ter significado local (dependente do contexto entre o usuário e um SP específico) ou global, e podem ter validade temporária (durante o tempo de uma sessão) ou permanente.

### 5.4. Gerência de Identidades Federadas em Nuvens

No setor acadêmico, ambientes de computação em nuvem geralmente pertencem a domínios privados mantidos por centros de pesquisa e universidades. Muitas vezes, nestes domínios, estão recursos cujo acesso pode ser realizado via Internet. Neste caso, é essencial que existam mecanismos para prover a autenticação e a autorização de usuários no acesso aos recursos, tanto para usuários vindos do próprio domínio, quanto de outros domínios parceiros. Estes mecanismos possibilitam a interoperabilidade de recursos localizados em domínios distintos, habilitando então a cooperação entre centros de pesquisa parceiros.

Para que a cooperação seja realizada com êxito, as entidades parceiras devem definir políticas para o compartilhamento de recursos junto aos domínios federados [Gomes and Kowalczyk 2011]. Esse processo envolve mecanismos de SSO para que, uma vez que o usuário esteja autenticado em seu domínio de origem, ele possa acessar recursos em quaisquer outros domínios federados, desde que esse usuário esteja inserido no contexto de autorização do domínio em questão. Em domínios não-federados, o mesmo usuário precisaria ter credenciais em ambos os domínios para ter permissões de acesso aos recursos distribuídos [Cao and Yang 2010].

O uso de padrões abertos assegura a interoperabilidade em ambientes de computação em nuvens híbridas. No entanto, os serviços oferecidos em nuvens ainda estão em processo de desenvolvimento e ainda existem muitos desafios para a provisão de segurança na integração de aplicações de domínios distintos. Outro fato é que nem todas as aplicações de uma empresa podem ser disponibilizadas na nuvem. Ao

mesmo tempo em que alguns serviços podem ser contratados de terceiros (tais como espaço de armazenagem, aplicações Web para *e-mail*, publicação *on-line* de documentos e hospedagem de páginas Web), outros serviços são exclusivos do uso interno da empresa e não devem ser disponibilizados publicamente (folha de pagamentos, relatórios empresariais e aplicativos internos de controle de produção, dentre outros).

Uma área da computação em nuvem onde a IdM tem papel importante é no cenário de *mash-ups*. Um *mash-up* pode ser caracterizado por uma página Web ou aplicação que compartilha, utiliza ou integra dados, apresentação e funcionalidade de uma ou mais fontes criando um novo serviço [Crupi and Warner 2008]. Um exemplo seria um *mash-up* entre um *blog* e um serviço de armazenamento de fotos, oferecendo aos usuários a possibilidade de postar mensagens com foto, sendo que estas fotos podem estar em diferentes sites. Neste cenário, o controle de acesso aos recursos compartilhados (fotos) deve ser baseado em políticas bem definidas e deve avaliar o pedido de acesso ao recurso com base na autenticação prévia do usuário feito pelo *blog*.

Provedores tais como Oracle, Google, Salesforce, IBM, Amazon e Microsoft são exemplos de empresas que começaram a estabelecer novos *datacenters* para a provisão de serviços de nuvem. A infraestrutura desses provedores deve comportar a habilidade de expandir/reduzir a capacidade de suas aplicações sob demanda para o fornecimento de conteúdo para redes sociais, aplicações comerciais, multimídia, jogos, entre muitos outros.

O modelo de acesso a aplicações em plataformas de computação em nuvem segue o modelo *multi-tenant* (multi-locatário), em que uma mesma aplicação passa a ser utilizada por múltiplos locatários (*tenants*), que são desde usuários convencionais a empresas. Esse modelo é interessante para aplicações em nuvem porque múltiplos usuários podem compartilhar os recursos físicos utilizados para prover um aplicativo, ao mesmo tempo em que as operações realizadas por um usuário não são vistas por outro.

Provedores de serviços tais como a Amazon, Salesforce.com e Aol, entre outros, possuem os seus próprios sistemas de gerência de identidades, mas não realizam processos de SSO. Infraestruturas para gerência de nuvens tais como Abiquo, OpenStack, Eucalyptus, Proxmox e OpenNebula não possuem um mecanismo para gerência de identidades federadas padronizado. Os sistemas Web existentes ainda são limitados quanto à distribuição automatizada de recursos entre nuvens, respeitando quesitos de QoS. Ambientes de computação federados devem ser capazes de adotar medidas para a provisão escalável de serviços, respeitando quesitos de QoS, condições de rede e variação de carga entre os diversos servidores associados.

A gerência de identidades federadas pode ser implantada de diversas maneiras. Segundo [Bertino and Takahashi 2011], a primeira delas é o chamado *in-house*. Nesta configuração, as identidades são expedidas e gerenciadas pelo próprio domínio (organização). A outra maneira é oferecer como um serviço tercerizado. Este cenário é chamado de identidade como um serviço (IDaaS - *Identity as a Service*). Nesta configuração, as identidades são expedidas e gerenciadas por provedores IDaaS. Em um cenário de hospedagem gerenciada, a organização contratante exporta toda a sua base de usuários ao provedor IDaaS terceirizado. Em outro cenário, o provedor IDaaS mantém somente pseudônimos das identidades dos usuários da organização, que por sua vez são

mapeadas para identidades de seus usuários reais.

### 5.4.1. Desafios da Gerência de Identidades Federadas em Nuvens

O desafio da gerência de identidades federadas está relacionado com a problemática de diferentes provedores de serviços utilizarem diferentes mecanismos de mapeamento de usuários em contextos de autenticação [El Maliki and Seigneur 2007]. A gerência de identidades federadas tem o objetivo de unificar a validação de contexto de usuários nos diferentes domínios parceiros. Isso significa que é preciso manter as devidas restrições e controle de acesso de acordo com o contexto em que o usuário está inserido no momento. As identidades de usuários fornecidas para aplicações podem ser do tipo usuário/senha, certificados digitais, *tokens*, biometria, cartões, entre outros [Cao and Yang 2010].

Dentre os principais desafios da gerência de identidades federadas está o acesso seguro a dados de outros domínios, sem que seja necessário a replicação dos serviços administrativos. Por isso é necessário que os mesmos protocolos sejam utilizados pelos domínios parceiros, ou que esses protocolos sejam interoperáveis. Como exemplo, a interface OCCI [Open Grid Forum 2009] é relativamente recente (início do projeto em 2009) e é um dos esforços da comunidade do Open Grid Forum (OGF) para definir como provedores de serviços podem oferecer seus recursos de computação, dados e rede através de interfaces padronizadas. O objetivo do projeto é promover interoperabilidade entre provedores de nuvem, sem a necessidade de tradução de padrões de formatação de dados para a criação, gerência e representação de recursos de um domínio para outro. Infraestruturas de nuvem como OpenNebula e Eucalyptus têm suporte para a interface OCCI, além de oferecerem interfaces para a EC2 Query API da Amazon, bem como APIs próprias de interconexão com outros provedores de nuvem.

Nesse novo modelo, um conjunto de aplicativos, infraestrutura e mesmo plataformas são oferecidos como serviços que não precisam estar fisicamente mantidos nos domínios de uma mesma organização. Os serviços são distribuídos em uma nuvem de recursos armazenados em diversas organizações. Em virtude desse modelo, muitos desafios se tornam inerentes para a provisão de segurança no acesso a serviços de múltiplos parceiros. A seguir são apresentados alguns deles [Buyya et al. 2010]:

- *Single Sign-On*: o acesso a serviços de domínios externos deve ser protegido por meio de serviços de autenticação. O desafio nesse caso está em garantir que uma única credencial seja fornecida e validada entre múltiplos domínios, sem a necessidade do usuário manter uma conta por domínio.

- Segurança para o acesso a recursos distribuídos: o uso de serviços de *datacenters* remotos deve contemplar mudanças nas políticas de provisão de recursos, uma vez que esses recursos poderão ser utilizados por qualquer um dos domínios parceiros. Em nuvens é desejável que tanto os usuários quanto as aplicações da nuvem sejam capazes de acessar múltiplos recursos, e que os usuários de um domínio sejam reconhecidos em outro, sem a replicação das bases de dados de usuários. Ainda que cada domínio possa manter suas próprias políticas de acesso aos serviços por ele oferecidos, outras políticas de controle de acesso podem estar distribuídas em muitas localizações da rede, e um mesmo usuário pode requerer

múltiplas regras para acessar os recursos de um mesmo serviço. Além disso, um mesmo usuário pode ter diferentes identidades, uma por domínio, e o ambiente da nuvem deve ser capaz de mapeá-las para um mesmo usuário. Por outro lado, as declarações (*entitlements*), que definem o conjunto de políticas de acesso, devem ser interoperáveis quanto ao formato de representação, ainda que cada domínio tenha a liberdade de especificar as suas próprias políticas de controle de acesso aos seus recursos locais.

- Dinamicidade: o modelo da computação em nuvem representa o cenário atual do comércio eletrônico, onde os relacionamentos entre os recursos ofertados/produzidos mudam rapidamente. A IdM em nuvens deve considerar a natureza dinâmica da provisão de recursos virtuais e físicos, seja por meio da alteração dinâmica de servidores para fins de balanceamento de carga durante uma migração, ou da provisão sob demanda (aplicações elásticas) de mais recursos para os usuários/aplicações. É comum que nesses ambientes servidores e máquinas virtuais sejam iniciados e finalizados dinamicamente, sendo que a IdM deveria ser informada que o acesso futuro foi permitido ou revogado [Gopalakrishnan 2009]. Além disso, o acesso deve ser monitorado e todos os provedores de serviços devem cumprir os acordos de SLA entre múltiplos domínios. Atrasos na (des)provisão de serviços poderia levar a sérios riscos de segurança [Gopalakrishnan 2009]. Para esses casos, a linguagem SPML (*Service Provisioning Markup Language*) [OASIS 2011] fornece uma descrição em estruturas XML para representar a (des)provisão de requisições para a IdM. Apesar de não ser obrigatório, a SPML utiliza o protocolo SAML.

- Escalabilidade: ambientes de computação em nuvem devem ser capazes de atender a um número relativamente grande de conexões, identidades e transações em um curto espaço de tempo. Dessa forma, o tempo de resposta para realizar as autenticações/autorizações deveria considerar os momentos de pico de consumo de recurso na rede.

- Interoperabilidade: atualmente existe uma corrida em torno de padrões para IdM. Dentre as várias alternativas podem ser citadas as soluções de conectividade da Microsoft e os padrões abertos OAuth, OpenID e SAMLv2, entre outros.

Outro desafio é a oportunidade dos provedores escolherem os domínios mais apropriados para atenderem às suas necessidades atuais. Dessa forma, um típico serviço de rede social, por exemplo, pode utilizar serviços de muitos outros provedores de nuvem, desde a simples armazenagem de dados até o processamento de grande quantidade de informações. Ainda assim, essa oferta deve ser dinâmica, ou seja, os mecanismos de interação com outros domínios devem facilitar tanto a requisição quanto a liberação de recursos adicionais, quando estes não forem mais necessários.

Outro desafio ligado à oferta de aplicações elásticas em ambientes federados é o fato de que os provedores podem estar localizados em diferentes regiões geográficas. Os clientes devem ser capazes de indicar em qual localidade preferem que seus dados sejam armazenados, além de indicar suas preferências sobre aumentar ou não a provisão de seus

recursos de acordo com a demanda. Por outro lado, um provedor pode terceirizar a sua própria oferta de serviços para atender aos quesitos de QoS de seus clientes. De acordo com [Buyya et al. 2010], uma vez que é inviável o estabelecimento físico de servidores em cada uma das regiões onde os serviços são acessados, os seguintes requisitos devem ser atendidos por ambientes de computação em nuvem federados:

- Oferta dinâmica de armazenagem e recursos computacionais de outros provedores de nuvem.
- Negociação de acordos quanto ao nível de serviço ofertado/consumido (SLA) entre os domínios federados.
- Garantias quanto à oferta de serviços para garantir padrões de QoS e minimizar os custos para o cliente final, de forma semelhante às tarifas de água, luz ou gás natural.

De acordo com o consumo de seus clientes a infraestrutura de nuvem deve ser capaz de prever comportamentos para manter-se em plena operação a maior parte do tempo utilizando, por exemplo, algoritmos de aprendizagem, análise estatística do uso da rede e consumo de processamento e/ou armazenagem. Em infraestruturas do tipo SaaS a IdM se preocupa em gerenciar o controle de acesso às aplicações. Por outro lado, infraestruturas do tipo PaaS requerem o controle de acesso à plataforma e às aplicações desenvolvidas na plataforma. Os requisitos para IdM em IaaS são similares às de PaaS, com a preocupação de controlar o acesso à infraestrutura de armazenagem, processamento e demais recursos de *hardware* disponibilizados na nuvem [Gopalakrishnan 2009]. A principal limitação da IdM em todos esses modelos é a perda de interoperabilidade caso os serviços de uma nuvem sejam migrados para outra.

O controle de acesso tradicional, centrado na aplicação, onde cada aplicação mantém e gerencia uma base de dados de usuários ou compartilha esta base de dados entre diversas aplicações não são adequados para ambientes de computação em nuvem federada. Isso porque em tais ambientes há o compartilhamento de informações entre múltiplos provedores de serviços. A replicação de identidades para servir cada aplicação (*one user-to-one app*) não é adequado devido ao crescimento exponencial [Olden 2011]. Além disso, este modelo exige que os usuários memorizem múltiplas identidades para cada novo serviço oferecido pela nuvem. O compartilhamento de bases entre aplicações de um mesmo domínio pode ser uma alternativa, porém entre domínios federados não é adequado, pois implica em questões de segurança no acesso à base. Olden argumenta que para limitar a replicação de identidades em uma nuvem, o modelo de uma identidade para muitas aplicações (*one user-to-many apps*) é o mais adequado, ou seja, a identidade do usuário deve estar federada com estas aplicações.

Outro desafio para a IdM em nuvens é que atualmente cada solução para gerência de recursos da nuvem utiliza uma solução própria para gerir as identidades e o acesso a estes recursos. Dado o contexto de federação/cooperação ao qual os provedores de nuvem estão situados, a demanda por uma solução padronizada, que contemple todas as nuances, ainda é esperada. Por exemplo, para a realização de SSO entre provedores

de nuvem, a Cloud Security Alliance recomenda que os provedores de serviços devem ser flexíveis e que aceitem os formatos utilizados pelos provedores de identidades [Cloud Security Alliance 2009]. A entidade também faz ressalvas no uso de protocolos proprietários.

O IDaaS deve ser capaz de suportar diferentes mecanismos de autenticação (LDAP, RADIUS, Certificados X.509, etc.), autorização (XACML, OAuth, dentre outros), federação (SAML, OpenID, InfoCard, etc.), diversas bases de dados (LDAP, MySQL, OpenDS, etc.), ser fracamente acoplado e não invasivo nas aplicações SaaS [Gopalakrishnan 2009] [Olden 2011]. Olden argumenta também que as organizações não devem integrar diretamente as aplicações oferecidas em nuvem para realizar SSO e acesso a recursos. Ao invés disto, devem federar com o IDaaS, que por sua vez está pré-integrado às diversas aplicações.

#### 5.4.2. Desafios da Privacidade em Nuvens

Os desafios da privacidade em nuvens estão nos modelos de controle de acesso empregados. Os modelos centrados no usuário são mais adequados para ambientes de computação em nuvem: as requisições aos SPs são tratadas de acordo com a identidade do usuário e suas credenciais. O controle de acesso centrado no usuário precisa conter informações que definam unicamente um usuário no domínio. Isso implica em manter um contexto de informação por usuário, ao invés de procurar a melhor forma de reagir, em determinada situação, a determinada requisição. O modelo deve suportar pseudônimos e múltiplas e discretas identidades na proteção da privacidade do usuário [Gopalakrishnan 2009]. O modelo centrado no usuário não coloca apenas o usuário no controle de sua identidade, mas também adiciona mais transparência em relação a quais dados estão sendo compartilhados e com quem. Os usuários ficam mais confiantes dado que estão dando permissões apropriadas a cada serviço utilizado.

### 5.5. Padrões e Soluções para Implantação de Gerência de Identidades Federadas

Esta seção apresenta alguns dos principais padrões e soluções utilizadas para a implantação de gerência de identidades federadas. Dividimos as subseções a seguir em padrões para autenticação e para autorização.

#### 5.5.1. Padrões para Autenticação

**SAML (Security Assertions Markup Language)**

SAML é um padrão aberto baseado em XML para a troca de informações de autenticação e autorização em domínios que participam de um mesmo CoT [OASIS 2008]. A especificação SAML define uma linguagem e um protocolo para a troca segura de informações, com a finalidade de prover o SSO e identidades federadas entre os domínios do CoT, independente da arquitetura que utiliza o SAML.

O protocolo SAML é interoperável com outros protocolos, tais como WS-Trust, WS-Federation, e desacopla o provedor de identidades (ou seja, o IdP atua como provedor de asserções SAML) e o provedor de serviços (ou seja, o SP atua como um consumidor

de asserções SAML). Uma asserção SAML é um documento em formato XML que é gerado por um IdP e contém declarações (*statements*) utilizadas pelo SP para tomar as devidas decisões sobre o controle de acesso aos recursos do domínio protegido. As seguintes declarações são definidas em uma asserção SAML: a) Declarações de autenticação: descrição que indica se o usuário foi autenticado pelo IdP em alguma interação anterior; b) Declarações de atributos: descrição de atributos (um par do tipo nome/valor) que é utilizado pelo CoT para tomar decisões sobre o controle de acesso, com base nesses atributos; c) Declaração de decisão de autorização: descrição que indica se o usuário/aplicação, de acordo com uma determinada restrição, pode ou não acessar o recurso.

O protocolo não especifica como os serviços do IdP devem ser implementados, mas espera que o IdP forneça os serviços de autenticação para o domínio local. Ao receber as asserções SAML do IdP, o SP decide quais serão as políticas quanto ao controle de acesso. Mais detalhes serão apresentados nas seções 5.5.3 e 5.6.4.

**OpenID**

OpenID é um protocolo aberto para a provisão de identidades federadas que permite o SSO para os usuários cadastrados em sites confiáveis do mesmo CoT. Com o OpenID, a credencial do usuário é fornecida para apenas um provedor de identidades, o qual identifica o acesso desse usuário aos outros sites que ele acessa [OpenID Foundation 2011]. Portanto, não é necessário que o usuário crie uma nova conta para acessar recursos de outros sites do mesmo CoT, mas é necessário que o usuário seja autenticado em pelo menos um deles.

Grandes provedores de identidades como Google, Facebook, Yahoo, Microsoft, Aol, MySpace e PayPal, entre muitos outros, fornecem suporte para o OpenID. O SSO é realizado sem a necessidade de relacionamento de confiança pré-existente entre os IdPs e os RPs (*relaying parties*), o que facilita a sua adoção [Ping Identity 2010b].

OpenID é um protocolo que utiliza redirecionamentos HTTP entre o usuário/aplicação e o provedor de identidades. As requisições para acessar o serviço de autenticação são baseadas no uso do protocolo HTTP. Primeiramente, o usuário deve se registrar em um provedor de identidades com suporte ao OpenID. O provedor de identidades, por sua vez, utiliza o nome da conta do usuário para gerar uma URL específica por usuário. Essa URL é utilizada pelo aplicativo cliente do usuário como argumento de localização do serviço remoto de autenticação do usuário, ou seja, a autenticação é vista como um serviço fornecido por um dos provedores de identidade do CoT. A partir dessa URL, o aplicativo cliente é capaz de identificar qual é o provedor de identidades remoto no qual o usuário possui uma conta. A seguir, caso o usuário ainda não tenha se autenticado, ele deve fornecer as suas credenciais no serviço de autenticação do provedor de identidades especificado na URL. OpenID também confere mecanismos de delegação de regras entre os provedores do mesmo círculo de confiança.

A Fig. 5.11 ilustra o mecanismo básico de autenticação federada com o OpenID. No passo 1, o usuário com uma identidade registrada em um provedor OpenID (usuário com conta de *e-mail* no Google, por exemplo), deseja acessar recursos de um Web

site no qual ele não está cadastrado. No entanto, esse Web site possui um serviço de autenticação OpenID. Para proceder com o acesso aos recursos protegidos do site, no passo 2 o usuário fornece a URL para acesso federado que recebeu de seu provedor de identidades. No passo 3, o serviço OpenID do Web site redireciona o *browser* do usuário para o serviço de autenticação do provedor de identidades, informado na URL fornecida anteriormente. No passo 4, duas opções são possíveis. Caso o usuário já esteja autenticado no provedor, o *browser* é redirecionado para o serviço de verificação do endereço de origem que solicitou a autenticação com o OpenID. Caso o usuário não esteja autenticado, o provedor de identidades solicita primeiro o nome de usuário e senha, para depois proceder com o redirecionamento. No passo 5, o Web site do provedor de identidades utiliza um serviço de verificação, para validar o endereço do serviço que solicitou a autenticação com o OpenID. Uma razão é que grande parte dos sites solicitam informações adicionais dos novos usuários, como o nome do usuário, *e-mail* e telefone, dentre outros. Essas informações podem ser fornecidas nesse passo, sem a necessidade de solicitá-las novamente após a autenticação. Caso o usuário permita que essas informações sejam fornecidas, no passo 6 o provedor de identidades redireciona o usuário para o Web site inicial.

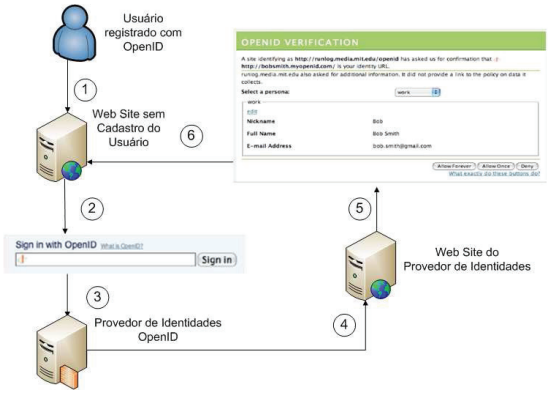


**Fig. 5.11. Mecanismo Básico de Autenticação Federada com o OpenID.**

### 5.5.2. Padrões para Autorização

#### XACML (Extensible Access Control Markup Language)

XACML 2.0 [OASIS 2005] é um padrão que define uma linguagem declarativa, utilizada para a descrição de políticas de controle de acesso, e uma linguagem para formular consultas (e obter respostas) utilizada para verificar se uma determinada ação é ou não permitida. Como resposta à consulta, um dos seguintes valores pode ser retornado: *Permit*, *Deny*, *Indeterminate* (um erro ocorreu, ou algum valor não foi especificado, impedindo o prosseguimento da consulta), ou *Not Applicable* (quando a resposta não puder ser retornada pelo serviço solicitado).

O padrão define também uma arquitetura para a aplicação de políticas baseada no modelo abstrato definido por [IETF 2000] e [ISO/IEC 1996]. Nesta arquitetura estão presentes os seguintes componentes:

- *Policy Enforcement Point (PEP)*: componente da arquitetura responsável por interceptar requisições a recursos/serviços e aplicar a decisão vinda do PDP.
- *Context Handler*: entidade responsável por traduzir as requisições a recursos de um formato nativo ao sistema para o formato XACML e vice-versa.
- *Policy Decision Point (PDP)*: componente responsável por avaliar as políticas aplicáveis e devolver ao PEP uma decisão de autorização.
- *Policy Administration Point (PAP)*: responsável pela criação das políticas.
- *Policy Information Point (PIP)*: componente responsável por obter informações/atributos adicionais e encaminhá-los ao PDP mediante solicitação.

A Fig. 5.12 apresenta uma visão geral, com os componentes centrais, PEP e PDP, atuando em uma requisição a um recurso protegido.

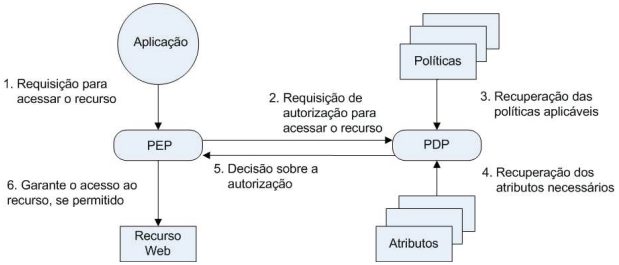


**Fig. 5.12. Visão Geral da Arquitetura do XACML. Adaptado de [Anderson 2005]**

Uma típica requisição solicita um recurso protegido por um servidor Web, o qual atua como um PEP (passo 1). O PEP forma uma requisição com os atributos da requisição original, que são as informações sobre o(s) recurso(s) que o usuário/aplicativo deseja interagir, a ação para a interação e outras informações específicas da requisição. Essa requisição é enviada para o PDP (passo 2), que verifica se existe uma política que se aplica a essa requisição (passo 3), verifica atributos adicionais para a tomada de decisão (passo 4) e retorna uma resposta para o PEP, contendo a decisão de autorização (passo 5). O PEP, por sua vez, irá aplicar a decisão do PDP de permitir ou negar o acesso ao recurso (passo 6).

Os documentos desta linguagem utilizam o formato XML estruturado em três níveis (*policy language model*): a) *PolicySet*: descrição de um conjunto de políticas ou outros *PolicySets*; b) *Policy*: descrição de uma política específica. Políticas são definidas como um conjunto de regras; c) *Rule*: descrição de uma regra. Regras definem se

um recurso pode ou não ser acessado, a partir dos atributos fornecidos e das restrições definidas em cada regra. Um documento XACML também especifica um *Target*. Caso todas as condições para o *Target* forem satisfeitas, então *PolicySet*, *Policy* e *Rule* relativas a esse *Target* são aplicadas à requisição. As condições para um *Target* são satisfeitas por meio de comparações dos valores da requisição com os definidos para o *Target*.

Os atributos de uma requisição são características do usuário/aplicação (sujeito), tais como o nome, perfil de acesso, recurso que o usuário quer acessar, período (data e hora) em que se quer acessar o recurso, entre outros. Os atributos também são características definidas para o *recurso* a ser acessado, para a *ação* a ser tomada e para o *ambiente* (localização do sujeito). Portanto, o PDP lida com comparações lógicas sobre esses valores (por exemplo, por meio de consultas XPath), de forma a responder à requisição do PEP para o acesso ao recurso protegido. O Quadro 5.1 ilustra um exemplo de documento XACML para descrever uma política de controle de acesso que permite a todos os usuários do domínio *realcloud.dca.fee.unicamp.br* acessarem recursos protegidos.

```
<?xml version="1.0" encoding="UTF-8"?>
<Policy PolicyId="identifier:example:SimplePolicy1"
RuleCombiningAlgId=
  "identifier:rule-combining-algorithm:deny-overrides">
  <Target>
      <Subjects><AnySubject/></Subjects>
      <Resources><AnyResource/></Resources>
      <Actions><AnyAction/></Actions>
  </Target>
  <Rule RuleId="identifier:example:SimpleRule1? Effect="Permit">
      <Target>
        <Subjects>
          <Subject>
              <SubjectMatch MatchId="function:rfc822name-equal">
                <SubjectAttributeDesignator AttributeId="identifier:subject:subject-id"
                 DataType="identifier:datatype:rfc822name"/>
                <AttributeValue DataType="identifier:datatype:rfc822name">
                 *@realcloud.dca.fee.unicamp.br</AttributeValue>
              </SubjectMatch>
          </Subject>
        </Subjects>
        <Resources><AnyResource/></Resources>
        <Actions><AnyAction/></Actions>
      </Target>
  </Rule>
</Policy>
```

**Quadro 5.1. Exemplo de Documento XACML para Descrever uma Política. Baseado em [Stoller 2011].**

Dentre as vantagens da linguagem XACML para a definição de políticas está o fato dela ser descrita em XML. Com isso XACML pode ser estendida e reutilizada entre diversas aplicações/organizações. Não há a necessidade de reescrever uma política em diversas linguagens diferentes para cada cenário. Outro fator é que ela pode ser utilizada nos mais diversos cenários, uma vez que ela é de propósito geral [Stoller 2011].

### OAuth

O protocolo OAuth é um padrão aberto de autenticação que fornece um mecanismo para que um usuário/aplicação possa compartilhar recursos na Web com terceiros sem ter que compartilhar a sua senha [OAuth Community 2011]. Esse protocolo também fornece uma alternativa para autorizar o acesso a esses recursos por um determinado tempo.

Dessa forma, um usuário proprietário de um recurso (um álbum de fotos, por exemplo) pode oferecer acesso temporário a esse recurso sem compartilhar a sua

identidade (nome de usuário e senha). Usuários que desejam ter acesso a esse recurso o fazem por meio de um serviço de delegação. No modelo OAuth existem três elementos principais no processo de autorização: o dono do recurso, o cliente e o servidor. Para o cliente acessar um recurso ele deve primeiramente obter uma autorização do dono do recurso, ou seja, o cliente recebe do dono do recurso uma permissão para acessar temporariamente o recurso, na forma de um *token* e uma chave secreta compartilhada. OAuth também verifica a autorização do dono do recurso e a identidade do cliente que faz a requisição a esse recurso [OAuth Community 2010].

O cenário descrito acima é semelhante ao utilizado pelo OAuth para autorizar o acesso a recursos Web a partir de provedores como o Twitter [Twitter 2011]. A Fig. 5.13 ilustra um cenário onde um usuário publica as suas fotos no site *ServidorRecurso.net* e deseja utilizar outro site para imprimi-las (*Cliente.net*). No entanto, o usuário dono do recurso não quer compartilhar a sua identidade (nome de usuário e senha) com o site de impressão.

No passo 1, ocorre o registro da aplicação cliente do site *Cliente.net* que deseja acessar o site *ServidorRecurso.net*. Um modelo similar aos de chave pública e privada é utilizado. Para cada aplicação registrada será fornecida uma *OAuth consumer key* (chave pública) e uma *OAuth consumer secret* (chave privada). Essas chaves são utilizadas para autenticar o usuário/aplicação (por exemplo, via Twitter API), garantindo que o tráfego seja do usuário portador dessas chaves. Por padrão, a Twitter API envia requisições no cabeçalho das mensagens HTTP (*HTTP Authorization header*).

No passo 2, as aplicações Web registradas devem fornecer uma URL de *callback* que o servidor do recurso irá utilizar para redirecionar o *browser* do usuário após a autenticação. O conjunto de *endpoints* que será utilizado nas interações também deve ser configurado: a) *Temporary Credential Request* - endereço para iniciar a solicitação de acesso ao recurso; b) *Resource Owner Authorization* - endereço para solicitar o serviço de autorização de acesso ao recurso; c) *Token Request URI* - endereço para solicitar *tokens* de acesso.

Antes do cliente ser autorizado a acessar os recursos, no passo 3 ele deve solicitar credenciais temporárias, a partir do endereço informado em *Temporary Credential Request*. O *ServidorRecurso.net* valida a solicitação porque *Cliente.net* possui uma *oath_consumer_key* (chave pública) válida. No passo 3.1, após validar a requisição, o servidor *ServidorRecurso.net* retorna uma mensagem HTTP contendo a *oauth_token* e *oauth_token_secret*.

No passo 4, a aplicação cliente em *Cliente.net* redireciona o *browser* do usuário para o endereço do *Resource Owner Authorization*, com a finalidade de obter a autorização do dono do recurso para acessar as suas fotos. No passo 5, *ServidorRecurso.net/authorize* solicita o acesso do dono do usuário ao recurso, usando o nome de usuário e senha do dono do recurso. Note que esse processo de autenticação ocorre no *ServidorRecurso.net*, ou seja, o dono do recurso se autentica em seu próprio domínio para delegar o acesso ao site *Cliente.net*. Quando confirmada a autenticação, o *ServidorRecurso.net* solicita a autorização do dono do recurso para que *Cliente.net* acesse o recurso. Caso o dono do recurso aprove, o *browser* é redirecionado para a URL de *callback* informada no registro da aplicação.

Até aqui, *Cliente.net* completou o seu processo de autorização. No passo 6, ele solicita o acesso a um *token* utilizando as credenciais temporárias que já obteve até então, ou seja, *oauth_consumer_key*, *oauth_token* e *oauth_verifier*. No passo 6.1, o servidor valida a requisição (como o fez no passo 3.1), porque a requisição possui uma *oauth_consumer_key* válida. O servidor também valida a requisição de acesso ao *token* porque recebeu *oauth_token* e *oauth_verifier* válidos. Após essa validação, o *ServidorRecurso.net* retorna no corpo da mensagem HTTP um conjunto de credenciais temporárias, ou seja, um novo *oauth_token* e um *oauth_token_secret*. Note que o *oauth_consumer_key* não muda durante a interação porque está relacionado com a autenticação do usuário dono do recurso.

No passo 7, o *Cliente.net* possui um conjunto de credenciais temporárias que lhe autorizam a acessar o recurso. Nesse passo, *Cliente.net* solicita um recurso de *ServidorRecurso.net* enviando *oauth_consumer_key* e o novo *oauth_token*. O *ServidorRecurso.net* irá validar o acesso porque recebeu um *oauth_consumer_key* e *oauth_token* válidos. O acesso de *Cliente.net* é permitido enquanto durar o tempo de vida do *token*, ou enquanto o dono do recurso autorizar o acesso.

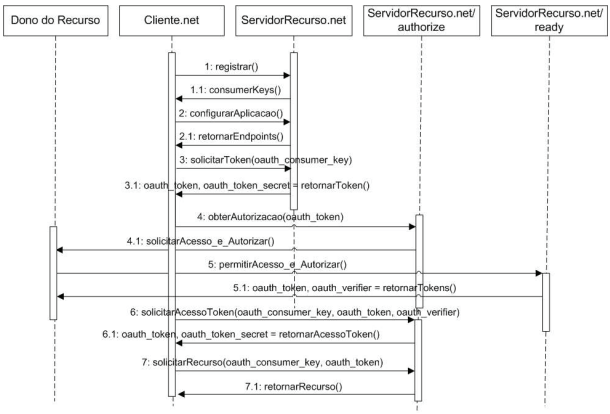


**Fig. 5.13. Diagrama de Sequência Simplificado do OAuth.**

### 5.5.3. Soluções

#### Shibboleth

Shibboleth [Internet2 2011] é um *middleware* gratuito e de código aberto desenvolvido pela comunidade Internet2 que provê uma solução de SSO, baseado em padrões abertos (principalmente XML e SAML) para autenticação e autorização na Web. Shibboleth é um sistema para a criação de federações que oferece funcionalidades para a troca segura de dados para acessar recursos entre domínios. Utilizado inicialmente para integrar instituições acadêmicas, hoje ele é usado por uma variedade de sites em todo o mundo.

Os principais elementos do Shibboleth são o *SP* e o *IdP*. O processo para

determinar o IdP do usuário pode fazer uso de um serviço de descoberta conhecido como *IdP Discovery* que exibe uma lista dos IdP confiáveis cadastrados no círculo de confiança.

Na arquitetura do Shibboleth, o acesso do usuário a recursos é realizado por meio de asserções SAML recebidas de um IdP confiável. Nessa arquitetura, os seguintes componentes são definidos para o SP:

- Recurso Alvo: os recursos Web são protegidos no SP por meio de serviços de Controle de Acesso, o que impede usuários não autenticados/autorizados de acessarem esses recursos.
- Serviço Consumidor de Asserções: é responsável por processar a asserção de autenticação do Serviço de SSO, que foi retornada pelo IdP. Além disso, esse componente redireciona o *browser* do usuário para o recurso desejado.
- Requisitante de Atributos: uma vez que um contexto de segurança tenha sido estabelecido por meio da validação fim-a-fim das asserções SAML trocadas entre IdP e SP, atributos podem ser trocados diretamente entre o SP (com o uso do componente Requisitante de Atributos) e o IdP (com o uso do componente Autoridade de Atributos).

O *IdP Discovery* é tratado na arquitetura como um serviço de *proxy* para realizar o redirecionamento do usuário para o domínio que possui o serviço de autenticação. O *IdP Discovery* também lista o conjunto de IdPs disponíveis para que o usuário escolha o domínio de autenticação mais adequado.

Para o IdP são definidos os seguintes componentes:

- Autoridade de Autenticação: lida com questões relacionadas à autenticação para outros componentes e é responsável por gerar a asserção de autenticação no IdP.
- Serviço de SSO: é responsável por iniciar o processo de autenticação no IdP e verificar a existência e validade dos *cookies* de sessão, e interagir com o componente Autoridade de Autenticação para gerar a asserção de autenticação no IdP.
- Serviço de Resolução: quando o SP define um perfil que utiliza a troca de asserções por referência, esse serviço é responsável por tratar essas requisições. Um artefato é uma referência para uma asserção de autenticação. Ou seja, ao invés de enviar a asserção de resposta de autenticação via o *browser* do usuário, é enviada uma referência a asserção expedida. Esta forma do SP obter a asserção é denominada *artifact binding*, conforme o padrão SAMLv2.
- Autoridade de Atributos: é responsável por autorizar e autenticar requisições que lidam com atributos de usuários.

Como ilustrado na Fig. 5.14, uma requisição de autenticação com o Shibboleth é uma mensagem enviada pelo SP e que contém uma URL enviada para o Serviço de SSO do IdP [Scavo and Cantor 2005]. No passo 1, essa requisição é feita para acessar um recurso protegido pelo SP. Caso já exista um contexto de segurança válido, segue-se para

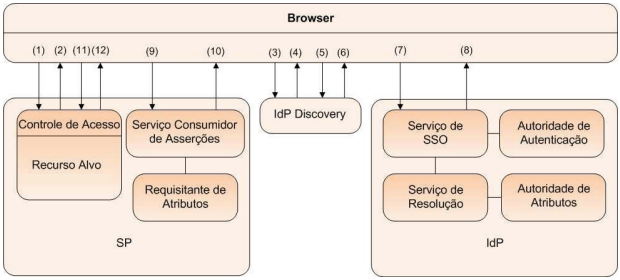


**Fig. 5.14. Fluxo de Mensagens na Arquitetura do Shibboleth.**

o passo 8. No passo 2, o *browser* do usuário é redirecionado para o *IdP Discovery*. No passo 3, o *IdP Discovery* verifica se existe um *cookie* de sessão e sua validade, e processa uma requisição de autenticação do usuário. Caso o usuário tenha um *cookie* válido, os passos 4 e 5 não são realizados. Caso contrário, um formulário HTML com uma lista de IdPs é fornecida. No passo 4, o *IdP Discovery* fornece uma lista de IdP disponíveis para que o usuário escolha onde deseja ser autenticado. No passo 5, o usuário seleciona o IdP desejado e uma requisição HTTP/GET é enviada novamente para o *IdP Discovery*. No passo 6, o *IdP Discovery* atualiza o *cookie* de sessão com as informações do IdP escolhido pelo usuário e redireciona o *browser* do usuário para o Serviço de SSO do IdP escolhido no passo anterior. No passo 7, o Serviço de SSO é requisitado no IdP e este serviço adquire uma declaração de autenticação (uma asserção SAML) da Autoridade de Autenticação. No passo 8, a asserção SAML é retornada para o *browser* do usuário, após o usuário fornecer as suas credenciais por meio de uma mensagem HTTP/POST. No passo 9, o Serviço Consumidor de Asserções do SP consulta a asserção SAML do IdP. No passo 10, o Serviço Consumidor de Asserções processa a resposta da autenticação (além de outras verificações, a asserção será válida caso o usuário tenha fornecido credenciais válidas no passo 8). Caso o usuário tenha fornecido credenciais válidas, um contexto de segurança é criado no SP, com o posterior redirecionamento do *browser* do usuário para o recurso protegido. No passo 11, o *browser* requisita novamente o acesso ao recurso protegido. No passo 12, com um contexto de segurança válido, o *browser* é redirecionado para a URL do recurso solicitado.

O processo de configuração de federações com Shibboleth não é trivial, uma vez que envolve a adaptação dos serviços do SP, das regras de controle de acesso do domínio, do *IdP Discovery* e da descrição dos atributos necessários para acessar o recurso [Hämmerle 2011]. A configuração de um sistema Shibboleth também envolve a manutenção de um intricado conjunto de arquivos. A federação CAFe [RNP 2011] é um exemplo brasileiro de instituições acadêmicas de ensino e de pesquisa que implementa uma federação com um conjunto de atributos próprios, com o esquema de dados conhecido como brEduPerson. Esses atributos definidos nas asserções SAML estabelecem quais recursos essas entidades poderão requisitar umas das outras, de forma que os provedores de identidades saibam quais atributos devem ser oferecidos.

**Projeto Higgins**

Higgins é um projeto em desenvolvimento pela Eclipse Foundation [Eclipse Foundation 2011]. Seu objetivo é integrar identidades, perfis e informações relacionadas a redes sociais entre diversos sites, aplicações e dispositivos utilizando componentes extensíveis. Múltiplos protocolos de identidades foram propostos ao longo do tempo, tais como LDAP, WS-Trust, SAML, XDI, OpenID, entre outros. Para os desenvolvedores de sistemas de gerência de identidades federadas, há a necessidade de suportar estes diversos protocolos, o que resulta em complexidade no *software* bem como na gerência de identidades federadas. Para os usuários, este fator pode causar confusão pois estes precisarão gerenciar para qual entidade foi compartilhada qual informação. Por exemplo, o número do cartão de crédito não é interessante de compartilhar em uma rede social, ao passo que em uma loja de vendas *on-line* sim. Para tratar estas questões, o projeto Higgins apresenta um *framework* que provê as seguintes tecnologias [Eclipse Foundation 2011]:

- Um seletor de identidades multiplatforma (Mac OS X, Linux, Windows, etc.) e navegadores (Firefox, Safari, IE, Chrome, etc.) que podem ser utilizados para autenticação em sistemas e sites compatíveis com o modelo centrado no usuário, denominado *Information Card* (i-card). Segundo o projeto, este modelo oferece ao usuário menos senhas, mais conveniência e melhor segurança onde todas as informações do usuário ficam armazenadas em um cartão.

- Provedores de identidades, baseado em serviços Web, que trabalham com os padrões WS-Trust (STS - *Security Token Service*) e SAML 2.0, ambos expedem i-cards. Provê também o código necessário para que provedores de serviço incorporem e aceitem estes i-cards.

- Implementa um modelo de dados chamado na versão 2.0 de *Persona Data Model* (PDM), estendendo o modelo da versão 1.x, denominado *Higgins Data Model* (HDM) 1.0. O Higgins oferece também um serviço de atributos de identidade denominado *Higgins Identity Attribute Service* (IdAS). Com estas soluções, os desenvolvedores possuem uma camada de abstração para obter interoperabilidade e portabilidade entre diferentes silos de informação de identidade, diferentes fontes de dados, tais como diretórios, bancos de dados relacionais e redes sociais.

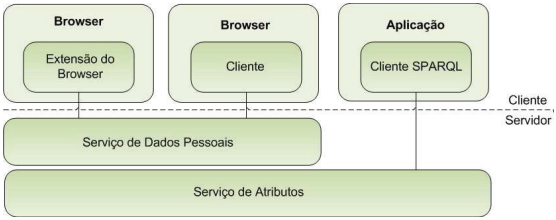


**Fig. 5.15. Arquitetura do Higgins. Adaptado de [Eclipse Foundation 2011]**

A Fig. 5.15 apresenta a arquitetura em alto nível do Higgins 2.0. Nesta figura, o componente Serviço de Dados Pessoais é responsável por prover serviços para o componente Cliente, localizado no *browser* do usuário. Dentre os serviços prestados estão aqueles para gerência de contas, alteração de *passwords*, etc. O componente Cliente é uma interface em JavaScript desenvolvida para que o usuário possa ver e editar seus cartões. Este JavaScript é carregado do Serviço de Dados Pessoais para o *browser* do usuário via o componente Extensão do Browser. Este componente é responsável por estender as funcionalidades de um *browser* padrão. Ele também provê uma API para que o *script* possa escrever e ler do componente Serviço de Atributos. Isto permite, por exemplo, que o *script* armazene informações vindas de uma página Web. O componente Serviço de Atributos é um repositório do tipo RDF/OWL (*Resource Description Framework*/*Web Ontology Language*) contendo dados do usuário. Estes dados utilizam um vocabulário definido pelo modelo *Persona Data Model* 2.0. O Serviço de Atributos expõe estes dados para o Portal e para o componente Extensão do Browser via uma interface baseada em mensagens HTTP e métodos do tipo *push*. Ele também expõe os dados via uma interface SPARQL (*SPARQL Protocol and RDF Query Language* - acrônimo recursivo) para integração entre servidores [Eclipse Foundation 2011].

**OpenAM**

O *middleware* OpenAM é uma solução de código aberto responsável por oferecer serviços de autenticação e autorização, federação, SSO, monitoração, *logging* e provisão de identidades. Com a aquisição da empresa Sun Microsystems pela Oracle, o produto originalmente conhecido como OpenSSO passou a ser mantido pelo grupo ForgeRock, porém com o nome de OpenAM. Os serviços de autenticação, autorização, verificação de validade de *tokens*, *logging* e provisão de identidades, oferecidos pelo OpenAM, podem ser acessados tanto via HTTP, quanto via serviços Web, ou por meio de um agente. Este modelo pode ser visto como um IDaaS para controle de acesso e provisão de identidades federadas [ForgeRock 2010]. O OpenAM possui uma arquitetura de *software* cliente/servidor. Desenvolvido em Java e distribuído sob a forma de uma aplicação Web J2EE, ele trabalha com diversos padrões abertos. A Fig. 5.16 ilustra a arquitetura simplificada do OpenAM.

Basicamente, o OpenAM está subdividido em 3 camadas (Interface Cliente, Núcleo e Camada de Integração). Uma descrição mais detalhada da arquitetura pode ser obtida em [Thangasamy 2011]. A primeira camada é responsável por prover interfaces de comunicação para que aplicações possam utilizar os serviços providos pela camada Núcleo. Esta camada pode ser acessada via serviços Web, através de SOAP ou HTTP. A camada Núcleo é onde estão os componentes fundamentais da arquitetura do OpenAM. Esta camada atua como um *broker* entre a Interface Cliente e os componentes servidores [ForgeRock 2010]. Os serviços providos por componentes desta camada são: autenticação, autorização, gerência de sessão (SSO), *logging*, acesso a repositórios de identidades e federação.

Os cinco primeiros serviços são implementados como *servlets* e as requisições e respostas são baseadas em mensagens XML sobre HTTP. O componente de federação

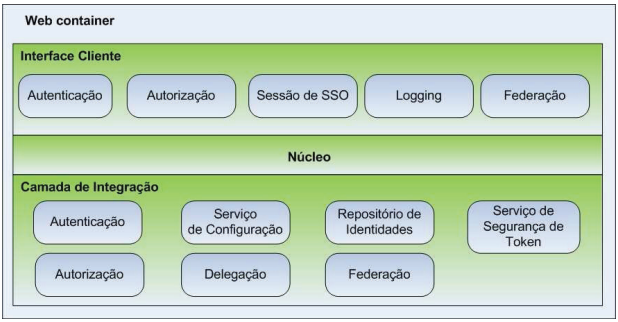


**Fig. 5.16. Arquitetura Simplificada do OpenAM.**

trabalha com as seguintes especificações: OASIS SAML (SAML1.1 e SAML 2.0), Liberty Alliance (ID-FF 1.2 e ID-WSF 1.1), WS-Security (WS-I BSP), WS-Trust e WS-Federation (WS-Federation 1.1). Cada serviço da camada Núcleo oferece uma SPI (*Service Provider Interface*) para extensão. A Camada de Integração provê as interfaces para extensão destes serviços. Como requisito para sistemas de gerência de identidades federadas, esta camada permite, por exemplo, a extensão para múltiplos mecanismos de autenticação e para múltiplos repositórios de identidades.

O OpenAM suporta também agentes. Estes são parte da arquitetura do OpenAM e são implementados como filtros instalados em servidores de aplicação tais como Apache, Tomcat, WebLogic e Glassfish. Os agentes são utilizados para interceptar requisições a recursos Web e encaminhar estas requisições para o OpenAM. Os agentes também utilizam as interfaces clientes do OpenAM para interagir com o serviço de autenticação, autorização, federação, sessão e *logging*. A Fig. 5.17 ilustra uma implantação básica do OpenAM e do agente protegendo recursos Web.

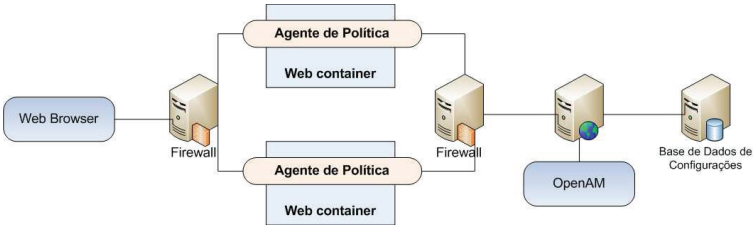


**Fig. 5.17. Implantação básica do OpenAM.**

O agente é instalado em servidores de aplicação onde os recursos a serem protegidos estão executando. Em geral estes servidores estão localizados atrás de um *firewall* externo, ou seja, em uma zona desmilitarizada. O OpenAM executa em outro servidor de aplicação, juntamente com um servidor de banco de dados, ambos atrás de um *firewall* interno. Uma requisição a uma página, por exemplo, feita por um usuário,

através de um *browser*, será interceptada pelo agente instalado no servidor onde a página é servida e fará com que o agente verifique se o usuário possui credenciais para acesso a este recurso. Em caso negativo, o *browser* do usuário será redirecionado para o OpenAM. O OpenAM fará o processo de autenticação deste usuário e irá redirecionar o mesmo de volta para a página, caso a autenticação seja bem sucedida.

### 5.5.4. Tabela Comparativa

A Tab. 5.1 apresenta as soluções de gerência de identidades federadas e os padrões estudados neste minicurso.

| Soluções | SAML 2.0 | OpenID 2.0 | XACML 2.0 | OAuth 1.0 | Paradigma |
|---|---|---|---|---|---|
| Shibboleth 2.x | √ | - | Extensão | - | Federado |
| Higgins 2.0 | √ | - | - | - | C.Usuário |
| OpenAM 9.5.x | √ | *Plugin* | √ | *Plugin* | Federado |

**Tab. 5.1. Tabela Comparativa entre Soluções Abertas em IdM.**

Como podemos observar na Tab. 5.1, o padrão para autenticação em gerência de identidades federadas mais utilizado entre as soluções abertas estudadas é o SAML. O OpenID, por sua vez, não está presente no Shibboleth e nem no Higgins, mas pode ser integrado ao OpenAM mediante a instalação de um *plugin*. Com relação aos padrões para autorização, o XACML é contemplado apenas pelo OpenAM. O Shibboleth requer uma extensão baseada em uma solução de repositório do Fedora. O OAuth não é suportado nativamente por nenhuma das soluções. Apenas o OpenAM oferece a possibilidade de instalação de um *plugin*.

Nenhuma das soluções apresentadas oferece suporte nativo a todos os padrões de gerência de identidades federadas estudados. Para ambientes de computação nuvem esta característica é importante, dado que o número de aplicações oferecidas como serviço são variadas. Apesar disso, o OpenAM é uma solução flexível que permite a possibilidade de extensão a partir da criação e instalação de *plugins*. Para um provedor IDaaS esta característica é interessante.

A utilização do OpenID e OAuth são mais adequados para ambientes de redes sociais (*blogs, mash-ups*), já que se propõem a serem soluções sem muita preocupação com a troca confiável de mensagens. O padrão SAML possui uma preocupação maior com a segurança, já que utiliza certificados digitais na troca de mensagens. Uma comparação mais detalhada entre o SAML e o OpenID pode ser encontrada em [Hodges 2009]. O XACML é um padrão que pode ser utilizado em conjunto com o SAML, podendo assim oferecer uma solução de autenticação e autorização em ambientes organizacionais.

O Higgins é a única solução estudada que adota o paradigma centrado no usuário, porém ainda é um projeto que está em evolução. Dentre as vantagens já citadas este paradigma possibilita maior autonomia ao usuário no compartilhamento de informações de sua identidade. O OpenAM, por ter uma arquitetura flexível, pode operar segundo o paradigma centrado no usuário, desde que seja instalado um *plugin* que implemente um padrão neste paradigma.

Observamos que as soluções estudadas, apesar de não suportarem a maioria dos novos padrões e paradigmas de gerência de identidades federadas, estão no processo de evolução para suportá-los. Isto é importante para provedores de nuvem, especialmente provedores IDaaS, que necessitam de uma infraestrutura flexível para atender à demanda por serviços federados.

### 5.6. Estudo de Caso: Plataforma REALcloud

Uma das motivações para o desenvolvimento de uma arquitetura para computação em nuvem é oferecer a plataforma REALabs como serviço (PaaS). Esta plataforma foi desenvolvida pelos autores para a interação remota com recursos robóticos [Guimarães et al. 2011]. REALabs oferece uma infraestrutura de *software* para o acesso remoto a um laboratório de robótica móvel. Nesta plataforma os experimentos robóticos são executados no computador do usuário e operam os robôs por meio de uma rede de comunicação (usualmente a Internet). Esta forma de operação, apesar de funcional, traz algumas limitações. A principal delas é com relação ao atraso imposto pela rede. Sem uma rede de alta velocidade o controle dos robôs fica prejudicado devido ao atraso na comunicação com o robô. Além disso, os usuários necessitam de CPU e memória consideráveis para execução dos experimentos. Um requisito também necessário é qualidade de serviço que não é garantida na Internet [Cardozo et al. 2010].

Outro ponto a ser considerado é a questão da segurança. Os recursos robóticos são caros e por isso necessitam de um controle de acesso aprimorado para permitir que apenas os usuários autenticados e autorizados possam acessar tais recursos. O acesso a estes recursos necessita de um modelo baseado em políticas compatível com o modelo de utilização dos recursos estabelecido pelo administrador do laboratório. Com a difusão de laboratórios de acesso remoto na Internet (WebLabs) surge a oportunidade de operar estes WebLabs de forma federada. Este conceito reforçou a necessidade de um controle de acesso aprimorado, que precisa contemplar acessos de usuários vindos de domínios parceiros.

Com o advento da computação em nuvem e os conceitos agregados de virtualização, computação orientada a serviços, computação sob demanda, modelo de acesso ubíquo, dentre outros, observou-se que a aplicação destes conceitos e tecnologias no contexto de WebLabs trazem ganhos significativos. A execução de experimentos em nuvem, tendo a máquina virtual do usuário na rede física dos recursos, permite um modelo mais adequado de experimentação. Neste modelo observa-se um ganho de desempenho na interação dos experimentos robóticos com os recursos, dado que tais experimentos executam na máquina virtual do usuário localizada no mesmo enlace de rede dos recursos.

Do ponto de vista da IdM, o desafio é oferecer uma infraestrutura que seja aderente à realidade colaborativa. O oferecimento da identidade como um serviço unificado de nuvem (IDaaS) é essencial para que as aplicações SaaS, PaaS e IaaS possam utilizar seus serviços sem necessitar recorrer a mecanismos individualizados de controle de acesso, fato este que inviabilizaria a gerência à medida que o número de serviços e colaborações aumentam. Outra questão importante é que este serviço permite realizar SSO entre os diferentes domínios federados. Esta infraestrutura também permite a integração de contas

de usuários provindas de diferentes domínios. Outra consideração é que o serviço IDaaS permite utilização de diferentes padrões de federação.

### 5.6.1. Visão Geral da Plataforma REALcloud

Para tratar os problemas relacionados no tópico anterior, desenvolvemos a plataforma REALcloud [Feliciano et al. 2011] [Agostinho et al. 2011]. Nesta plataforma, há uma infraestrutura de nuvem privada para os estudantes utilizarem os servidores conectados aos robôs através de uma rede local para realizarem os seus experimentos. Como o WebLab é um ambiente colaborativo, são utilizados os conceitos definidos pela gerência de identidades federadas para gerenciar o uso dos recursos distribuídos entre os diferentes domínios parceiros e seus experimentadores.

A Fig. 5.18 apresenta uma visão em alto nível da plataforma REALcloud. A figura apresenta um conjunto de nuvens privadas e públicas conectadas através da Internet e/ou uma rede privada de alta velocidade. A nuvem privada oferece recursos tais como CPU, unidades de armazenamento e conexões de rede para as aplicações que executam os experimentos. Esta nuvem oferece também recursos de *hardware* especializados tais como GPUs (*Graphics Processing Units*) e FPGAs (*Field-Programmable Gate Arrays*) e também plataformas de *software* específicas, tais como Matlab.

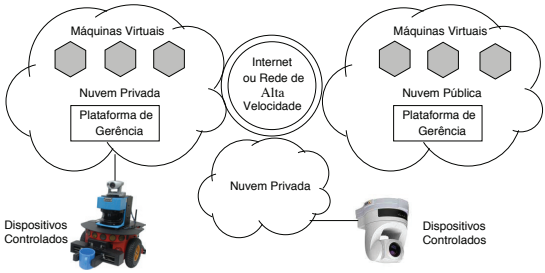


**Fig. 5.18. Arquitetura de Alto Nível da Plataforma REALcloud.**

A nuvem pública oferece recursos computacionais para aplicações e componentes que não necessitam de um tratamento de qualidade de serviço e segurança aprimorados. Alguns destes recursos podem ser oferecidos como serviços, por exemplo, serviços de gerência de WebLabs. A nuvem pública pode atuar como um provedor de identidades, permitindo que os usuários possam se autenticar, por exemplo, com uma conta do Yahoo! e com isso utilizar recursos habilitados para este tipo de usuário. A nuvem pública pode também servir para ampliar uma facilidade que alguma nuvem privada porventura necessite.

### 5.6.2. Arquitetura da Plataforma REALcloud

A plataforma REALcloud possui as seguintes características:

- Gerência de identidades federadas, para permitir o compartilhamento seguro de recursos e SSO entre os domínios federados.

- Controle de acesso para prevenir acessos não autorizados e manipulação acidental dos dispositivos físicos controlados.
- Plataforma para gerência de máquinas virtuais.
- Controle de QoS de rede, para que as aplicações tenham baixo atraso e largura banda adequadas.

Cada requisito acima é atendido por um serviço especializado. O Serviço de Gerência de Identidades Federadas é responsável pelo estabelecimento de federações entre as nuvens públicas e privadas e proteção para os serviços restantes. Este serviço provê também SSO e oferece uma API baseada em REST para utilização da identidade como um serviço (IDaaS).

O Serviço de Gerência de Acesso aos Dispositivos permite ao administrador do domínio registrar os dispositivos físicos disponíveis e outros recursos, bem como estabelecer políticas para sua manipulação (por exemplo, uso mediante reserva). Este serviço permite também o estabelecimento de sessões seguras de acesso aos dispositivos controlados.

Virtualização é uma tecnologia chave para a computação em nuvem. Máquinas virtuais oferecem isolamento e recursos computacionais de acordo com a demanda do usuário. No nosso caso, a virtualização também oferece a possibilidade das aplicações estarem próximas dos dispositivos controlados, reduzindo o atraso inerente que degrada a operação dos dispositivos. O Serviço de Gerência de VMs permite aos administradores atribuir VMs aos usuários ou grupo de usuários, bem como permite aos usuários acessar (mediante credenciais) e controlar suas VMs.

Para proteger os dispositivos controlados contra acessos não autorizados, o Serviço de Gerência de Acesso aos Dispositivos utiliza-se de mecanismos de *firewall* providos pelo Serviço de Controle de Rede. Este serviço realiza um controle de acesso no nível de encaminhamento de pacotes que bloqueia o acesso não autorizado e estabelece privilégios no nível de rede para as aplicações que estabeleceram uma sessão de acesso. A Fig. 5.19 ilustra os serviços de gerência definidos para a plataforma REALcloud (Plataforma de Gerência na Fig. 5.18). Cada pacote representa um serviço e as linhas pontilhadas relacionam as dependências entre estes serviços.

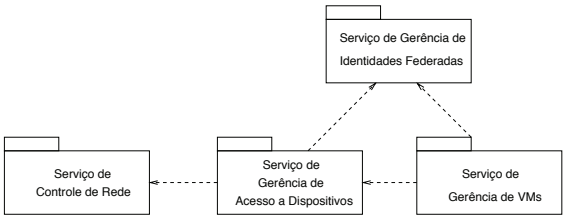


**Fig. 5.19. Serviços de Gerência da Plataforma REALcloud e suas Dependências.**

### 5.6.3. Serviços da Plataforma REALcloud

A plataforma REALcloud especializa os serviços identificados na arquitetura proposta (Fig. 5.19).

#### Gerência de Identidades Federadas

O serviço de Gerência de Identidades Federadas na plataforma REALcloud adota o OpenAM. Este *middleware* foi configurado para funcionar com os perfis *Web SSO* e *IdP Discovery*, definidos pela especificação do SAMLv2. Na plataforma é empregado um modelo para a gerência de identidades federadas onde os componentes SP, IdP e IdP *proxy* estão dispostos em um modelo em camadas. A Fig. 5.20 mostra este modelo acrescido das funcionalidades dos componentes PIP, PAP, PEP e PDP, conforme empregados no padrão XACML.

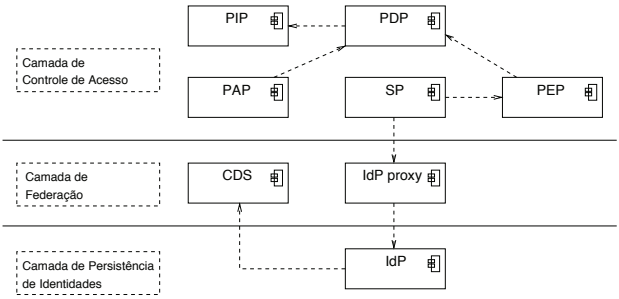


**Fig. 5.20. Modelo em Camadas dos Componentes para IdM [Feliciano et al. 2011].**

Quando um usuário necessita acessar algum recurso da plataforma, o SP, com o auxílio do componente PEP, intercepta a requisição e verifica se a requisição possui as credenciais necessárias. Se nenhuma credencial for encontrada, o SP redireciona o usuário para o IdP *proxy*, que por sua vez apresenta uma listagem contendo os possíveis IdPs. O usuário seleciona um IdP relacionado ao domínio, onde o mesmo possui uma identidade e é redirecionado para autenticação. Uma vez autenticado, o IdP expede uma credencial (uma asserção SAML) e redireciona o usuário para o componente CDS (*Common Domain Service*), que adiciona um *cookie* de domínio comum (*cookie* de federação). Na sequência, o componente CDS redireciona o *browser* para o IdP *proxy*. Este, por sua vez, apenas verifica qual foi o SP requisitante e, com isso, redireciona o usuário para este SP. O SP novamente verifica a existência de uma credencial de acesso e, desta vez, a presença da credencial permitirá o acesso ao recurso originalmente solicitado. Como a autenticação foi realizada com SSO, o usuário também está apto a acessar recursos existentes nos domínios pertencentes ao círculo de confiança, caso as políticas de autorização permitam tal operação.

**Gerência de Acesso a Dispositivos**

A Gerência de Acesso a Dispositivos realiza três funções, implementadas como serviço:

1. Registro de Recursos: este serviço permite a criação de uma lista de recursos a serem protegidos. Esta lista armazena, para cada recurso, as suas propriedades (nome, dono do recurso, modelo, etc.), largura de banda requerida para a realização do experimento e a URI (*Uniform Resource Identifier*) que identifica o recurso. Para a realização de um experimento, os recursos podem ser agrupados. Por exemplo, um braço robótico e uma câmera podem ser agrupados para que a câmera mostre os movimentos do braço. Este agrupamento é feito mediante reserva, provida por este serviço. O serviço permite que o administrador separe fatias de tempo para a utilização de cada recurso ou grupo de recursos e a duração máxima da reserva no dia.

2. Reserva: é um serviço que permite aos usuários agendar reservas futuras para a utilização dos recursos, respeitando as fatias de tempo disponíveis e a duração máxima da reserva estipulada pelo administrador.

3. Controle de Sessão: é usado para estabelecer a sessão de acesso aos dispositivos, usualmente durante o período da reserva. As sessões de acesso mantêm o estado da utilização dos recursos, por exemplo, a identidade do usuário, período de acesso e os recursos permitidos.

**Gerência de VMs**

O serviço de Gerência de VMs na plataforma REALcloud permite que os usuários iniciem e desliguem suas respectivas VMs mantidas nas nuvens públicas e privadas. Os usuários podem acessar suas VMs a qualquer horário, sem necessidade de reserva de horário. Porém, este acesso não habilita o usuário acessar os recursos protegidos do laboratório. O acesso aos recursos protegidos só é liberado se o usuário possuir uma sessão de acesso obtida do serviço de Gerência de Acesso a Dispositivos.

Assim que o usuário estabelece uma sessão, o serviço de Gerência de Acesso a Dispositivos interage com o serviço de Controle de Rede para habilitar o acesso da VM que estabeleceu a sessão para o dispositivo físico e armazenar as regras de encaminhamento de tráfego de acordo com o dispositivo acessado. Esses privilégios de acesso são descartados quando a sessão de acesso terminar. Uma sessão pode terminar explicitamente pelo usuário ou pelo serviço de Gerência de Acesso a Dispositivos, quando o período da reserva terminar.

**Virtualizador, Controle de Rede e Encaminhamento de Pacotes**

Virtualizador é um serviço para estabelecer a interação com diferentes soluções de virtualização, tais como libvirtd, KVM e VirtualBox. Utiliza-se estes pacotes para abstrair

a solução de virtualização, por exemplo, alterações na tecnologia de virtualização tem impacto reduzido em toda a arquitetura.

O serviço de Controle de Rede é necessário para interagir com o mecanismo de encaminhamento de pacotes, empregar políticas de *firewall* e para realizar diferenciação de tráfego.

O serviço de Encaminhamento de Pacotes filtra o tráfego direcionado aos dispositivos físicos de/para a VM que possui uma sessão de acesso válida. A priorização de tráfego permite estipular classes de serviços (prioridades) de acordo com o dispositivo físico sendo acessado. Por exemplo, imagens em tempo real utilizadas para controlar o robô móvel devem ter maior prioridade do que imagens geradas por uma câmera panorâmica, utilizada para visualização dos experimentos. A classe de serviço e a banda requerida são especificadas através do serviço de Registro de Recursos. A atual versão da plataforma REALcloud não suporta SLAs.

A Fig. 5.21 apresenta os serviços oferecidos pela plataforma REAcloud e a sua inter-relação em notação UML. É importante ressaltar que as linhas tracejadas modelam os fluxos da interação.

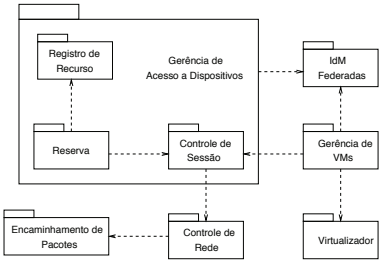


**Fig. 5.21. Serviços de Gerência da Plataforma REALcloud.**

### 5.6.4. Implementação da Plataforma REALcloud

O serviço de Gerência de VMs é um *servlet* Java que expõe uma interface Web para controlar o ciclo de vida das VMs, utilizando os serviços do Virtualizador. Os demais serviços oferecidos pela plataforma REALcloud estão descritos na sequência.

#### Serviço de Gerência de Identidades Federadas

A implementação deste serviço utiliza como base o OpenAM. Na plataforma REALcloud cada componente do OpenAM (SP, IdP proxy e IdP) está instalado em uma VM. A utilização de componentes deste serviço em VMs oferece à plataforma o aumento no tempo de disponibilidade do serviço e também uma diminuição do tempo de instalação deste serviço em outras nuvens.

O estabelecimento do CoT da federação se dá através da configuração de um perfil (SP, IdP ou IdP proxy) em cada instância do OpenAM. Para tal, foi necessário a criação

de um metadado no padrão SAMLv2. Além de conter informações sobre o perfil, este metadado possui um certificado X.509 gerado em cada entidade. A Fig. 5.22(a) ilustra a troca de metadados realizada entre os componentes de IdM da nuvem e entre as nuvens privadas (b), respectivamente. No passo 1, gera-se os metadados do IdP *proxy* com o perfil SP e IdP. No passo 2, o SP importa a parte IdP do metadado do IdP *proxy*. No passo 3, o IdP importa a parte SP do metadado do IdP *proxy*. No passo 4, o IdP *proxy* importa o metadado do SP. Finalmente, no passo 5 o IdP *proxy* importa o metadado do IdP. O IdP proxy atuará, sob o ponto de vista do SP, como um IdP e do ponto de vista de um IdP, como um SP. Isto foi importante para permitir a agregação de múltiplos IdPs (de domínios diferentes) para fazer parte da federação. Na Fig. 5.22(b), a troca de metadados ocorre entre duas nuvens privadas. No passo 1, no IdP *proxy* do domínio 1 é importado o metadado do IdP do domínio 2. No passo 2, é importado no IdP do domínio 2, a parte SP do IdP *proxy* do domínio 1. O mesmo processo (passos 3 e 4) é realizado entre os componentes IdP e IdP *proxy* dos domínios 1 e 2, respectivamente.

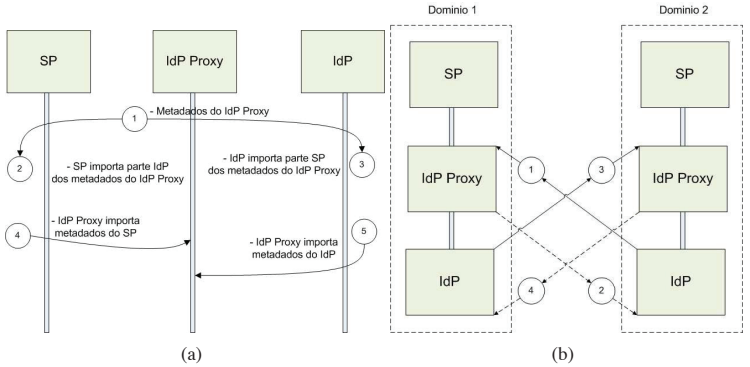


**Fig. 5.22. Troca de Metadados: (a) Dentro da Nuvem Privada, (b) entre Nuvens Privadas.**

Uma parte importante da infraestrutura de gerência de identidades federadas é o agente. O agente atua como um PEP e a filtragem ocorre na requisição de um recurso hospedado. Na nuvem privada, foi instalado um agente no servidor Apache para proteger páginas de gerência do laboratório (reserva de horário, cadastro de usuários, cadastro de experimentos, provisão de experimentos, etc.). Neste servidor Apache também está instalado o módulo *mod_proxy*. Com isso a plataforma utiliza um modelo de implantação de SSO misto, que combina o uso de *proxy* reverso e um agente instalado neste servidor para filtrar as requisições a todos os recursos protegidos da plataforma.

**Serviço de Gerência de Acesso a Dispositivos**

O serviço de Gerência de Acesso a Dispositivos providos pela plataforma REALcloud foi implementado com a tecnologia J2EE (*Java 2 Enterprise Edition*), como provido pelo servidor de aplicação Tomcat. Para a persistência dos dados foi utilizado o servidor de banco de dados MySQL.

**Serviço de Controle de Rede**

O serviço de Controle de Rede foi implementado como um *servlet* que executa um *shell script* para interagir com as regras de *firewall* padrão do Linux, oferecido pelo serviço de Encaminhamento de Pacotes. O serviço de Controle de Rede não provê serviços no nível de usuário, tendo como cliente o serviço de Gerência de Acesso a Dispositivos. Os serviços da plataforma estão distribuídos de acordo com a Fig. 5.23.

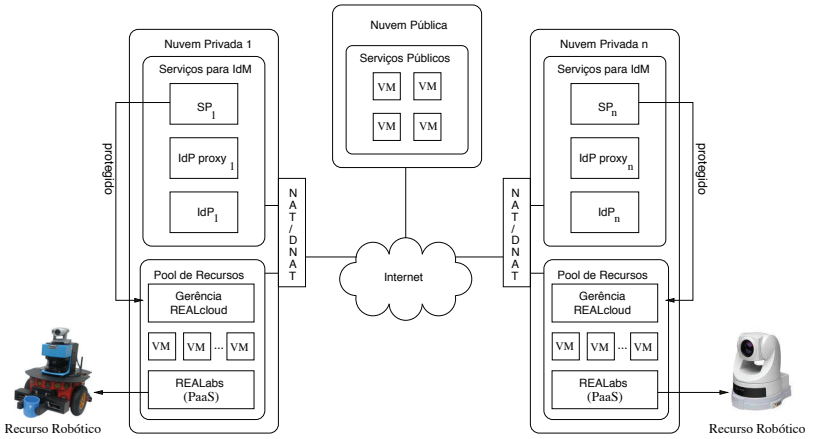


**Fig. 5.23. Plataforma REALcloud.**

As nuvens privadas 1 e N na Fig. 5.23 representam dois domínios privados que possuem recursos a serem compartilhados. O serviço de Gerência de VM, o serviço de Gerência de Acesso a Dispositivos e o serviço de Controle de Rede estão distribuídos como componentes da plataforma (Gerência REALcloud). Os componentes de gerência da plataforma REALcloud e os recursos tais como robôs, câmeras panorâmicas, máquinas virtuais, a plataforma REALabs, entre outros, localizados no *pool* de recursos, são protegidos pelo componente SP do serviço de gerência de identidades federadas.

A nuvem pública oferece apenas serviços públicos, tais como: hospedagem de páginas, armazenagem de dados e aplicações específicas. Cada serviço está inserido em uma VM ou em VMs que agregam serviços co-relacionados. A nuvem pública pode estabelecer federações entre as nuvens privadas ou entre um provedor de nuvem pública (por exemplo, Amazon). Nesse caso, o serviço de gerência de identidades federadas deve ser instalado na nuvem pública.

Finalmente, cada nuvem está interconectada à Internet. Para se traduzir endereços IP privados para públicos e vice-versa, foi utilizado a função de rede NAT/DNAT. Assim cada requisição originada através da Internet para o par *IP:porta* pública de cada nuvem privada é mapeado para o par *IP:porta* privada, onde os recursos existentes no *pool* estão conectados.

## 5.7. Considerações Finais

A gerência de identidades acompanha as novas necessidades colaborativas de uma Internet que está em constante evolução. A integração de aplicações na Web com ambientes federados tem o objetivo de suprir as expectativas de uma Web de serviços independentes, porém interoperáveis.

Apesar das vantagens quanto à elasticidade na oferta de serviços em nuvem, existem alguns desafios que ainda precisam ser superados. Dentre estes estão as questões legais quanto à garantia de restrições sobre o tipo de conteúdo armazenado, além de garantias quanto ao nível de acesso do provedor de serviços aos dados hospedados em seus *datacenters*. Por outro lado, o acesso aos recursos do domínio deve ser gerenciado, para que apenas os usuários com contas no domínio, ou em domínios parceiros, tenham acesso a esses recursos. Além disso, o provedor de serviços de nuvem deve comportar o aumento do número de usuários, sem que isso reduza o desempenho das conexões individuais. Também, aplicativos empresariais que envolvem sistemas de intranet, ambientes colaborativos, redes sociais, aplicativos para celular, dentre outros, também podem se beneficiar do uso da nuvem como porta de acesso para uma ampla gama de serviços federados, ou seja, não exclusivos de uma única nuvem. Nesse sentido, a gerência de identidades federadas contribui para assegurar mecanismos confiáveis para a provisão de gerência de múltiplas identidades.

Em nossa experiência na utilização de soluções para implantação de IdM em nuvens verificamos que estas não estão totalmente adaptadas para tratar os desafios da computação em nuvem. A configuração desses sistemas requer um conhecimento prévio dos padrões e, principalmente, da teoria que lhes dá suporte, visto que a terminologia utilizada nestes sistemas é baseada nos conceitos destes padrões. O Shibboleth por ser o sistema precursor, carece de suporte aos padrões emergentes, tais como OpenID e OAuth. A instalação deste sistema é dispendiosa, visto que parte de seus componentes estão em Java e em C/C++. Outra questão é que a instalação é baseada em arquivos XML, o que dificulta o entendimento para um iniciante na área. Já o OpenAM oferece todo o seu pacote de *software* encapsulado de um único arquivo (.war), o que facilita a instalação. A configuração, por sua vez, também demanda um conhecimento prévio dos padrões, principalmente na configuração de federações, que requer um considerável esforço. No caso do Higgins, por adotar o paradigma centrado no usuário e ser um projeto ainda em desenvolvimento, existe pouca informação quanto a sua utilização.

A plataforma REALcloud é um exemplo de solução de código aberto para a realização de experimentos em rede. Como características desta solução, citamos a capacidade de agregar nuvens privadas distintas com compartilhamento de recursos utilizando relações de confiança da federação e também a capacidade de integrar nuvens públicas com a finalidade de aumentar a oferta de recursos computacionais para as demais nuvens privadas.

## Referências

[Agostinho et al. 2011] Agostinho, L., Olivi, L., Feliciano, G., Paolieri, F., Rodrigues, D., Guimarães, E., and Cardozo, E. (2011). A Cloud Computing Environment for Supporting Networked Robotics Applications. *The 3rd International Workshop on*

*Workflow Management in Service and Cloud Computing*.

[Amazon 2010] Amazon (2010). Amazon Elastic Compute Cloud (Amazon EC2). Disponível em: http://aws.amazon.com. Acessado em 10 de Setembro de 2011.

[Anderson 2005] Anderson, A. (2005). A Comparison of Two Privacy Policy Languages: EPAL and XACML. Technical report. Disponível em: http://labs.oracle.com/techrep/2005/smli_tr-2005-147/TRCompareEPALandXACML.html.

[Bertino and Takahashi 2011] Bertino, E. and Takahashi, K. (2011). *Identity Management: Concepts, Technologies, and Systems*. Artech House.

[Breitman and Viterbo 2010] Breitman, K. and Viterbo, H. (2010). Computação na Nuvem - Uma Visão Geral. *Congresso Internacional Software Livre e Governo Eletrônico - Consegi*, pages 17–45.

[Buyya et al. 2010] Buyya, R., Ranjan, R., and Calheiros, R. N. (2010). InterCloud: Utility-Oriented Federation of Cloud Computing Environments for Scaling of Application Services. In *Proceedings of the 10th ICA3PP*, pages 21–23. Springer.

[Cao and Yang 2010] Cao, Y. and Yang, L. (2010). A Survey of Identity Management Technology. *IEEE International Conference on Information Theory and Information Security (ICITIS)*.

[Cardozo et al. 2010] Cardozo, E., Guimarães, E. G., Rocha, L. A., Souza, R. S., Paolieri, F., and Pinho, F. (2010). A platform for networked robotics. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2010), Taipei, Taiwan*.

[Carissimi 2008] Carissimi, A. (2008). Virtualização: da teoria a soluções. In *Mini-curso - SBRC 2008 - Rio de Janeiro - RJ*.

[Cloud Security Alliance 2009] Cloud Security Alliance (2009). Security Guidance for Critical Areas of Focus in Cloud Computing V2.1. Technical report.

[Crupi and Warner 2008] Crupi, J. and Warner, C. (2008). Enterprise Mashups Part I: Bringing SOA to the People . *SOA Magazine*, (18).

[Dike 2006] Dike, J. (2006). *User-Mode Linux*. Prentice Hall.

[Eclipse Foundation 2011] Eclipse Foundation (2011). Higgins - Open Source Identity Framework. Disponível em: http://www.eclipse.org/higgins. Acessado em 20 de Agosto de 2011.

[El Maliki and Seigneur 2007] El Maliki, T. and Seigneur, J.-M. (2007). A Survey of User-centric Identity Management Technologies. In *Proceedings of the The International Conference on Emerging Security Information, Systems, and Technologies*, Washington, DC, USA.

[Endo et al. 2010] Endo, P. T., Gonçalves, G. E., Kelner, J., and Sadok, D. (2010). A survey on open-source cloud computing solutions. *XXVIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos - VIII Workshop em Clouds, Grids e Aplicações - Gramado - RS*.

[Feliciano et al. 2011] Feliciano, G., Agostinho, L., Guimarães, E., and Cardozo, E. (2011). Uma Arquitetura para Gerência de Identidades em Nuvens Híbridas. *IX Workshop em Clouds, Grids e Aplicações (WCGA 2011) - XXIX Simpósio Brasileiro de Redes de Computadores (SBRC - 2011) - Campo Grande - MS*.

[FlexiScale 2010] FlexiScale (2010). FlexiScale Public Cloud. Disponível em: http://www.flexiant.com/products/flexiscale. Acessado em 2 de Agosto de 2011.

[ForgeRock 2010] ForgeRock (2010). OpenAM. Disponível em: http://www.forgerock.com/openam.html. Acessado em 9 de Agosto de 2011.

[Gomes and Kowalczyk 2011] Gomes, E. R.and Vo, Q. B. and Kowalczyk, R. (2011). Pure exchange markets for resource sharing in federated clouds. In *Proceedings of Concurrency and Computation: Practice and Experience*.

[Gonçalves et al. 2011] Gonçalves, G. E., Endo, P. T., Cordeiro, T. D., and et al. (2011). Resource Allocation in Clouds: Concepts, Tools and Research Challenges. *XXIX SBRC - Gramado - RS*.

[Google 2010] Google (2010). Google Apps. Disponível em: http://www.google.com/apps. Acessado em 2 de Setembro de 2011.

[Gopalakrishnan 2009] Gopalakrishnan, A. (2009). Cloud Computing Identity Management. *SETLabs Briefings*, 7(7).

[Guimarães et al. 2011] Guimarães, E. G., Cardozo, E., Moraes, D. H., and Coelho, P. R. (2011). Design and Implementation Issues for Modern Remote Laboratories. *IEEE Transactions on Learning Technologies*, 4(1).

[Hodges 2009] Hodges, J. (2009). Technical Comparison: OpenID and SAML - Draft 07a. Technical report. Disponível em: http://identitymeme.org/doc/draft-hodges-saml-openid-compare.html. Acessado em 2 de Agosto de 2011.

[Hämmerle 2011] Hämmerle, L. (2011). Enabling Interfederation Support for a Shibboleth Service Provider (SP) in SWITCHaai. Disponível em: https://www.switch.ch/aai/docs/interfederation/sp-deployment.html. Acessado em 15 de Setembro de 2011.

[IBM 2011] IBM (2011). New to SOA and Web Services. Disponível em: http://www-128.ibm.com/developerworks/webservices/newto/websvc.html. Acessado em 4 de Agosto de 2011.

[IETF 2000] IETF (2000). Framework for Policy-based Admission Control IETF RFC 2753. Technical report. Disponível em: http://www.ietf.org/rfc/rfc2753.txt. Acessado em 25 de Agosto de 2011.

[Internet2 2011] Internet2 (2011). Shibboleth - A Project of the Internet2 Middleware Initiative. Disponível em: http://shibboleth.internet2.edu. Acessado em 15 de Setembro de 2011.

[ISO/IEC 1996] ISO/IEC (1996). Information technology – Open Systems Interconnection – Security frameworks for open systems: Access control framework ISO/IEC 10181-3:1966. Technical report.

[ITU-T 2009] ITU-T (2009). NGN Identity Management Framework. Recommendation Y.2720. Technical report. Disponível em: http://www.itu.int/itu-t/recommendations/rec.aspx?id=9574. Acessado em 20 de Agosto de 2011.

[Johansson 2009] Johansson, L. (2009). It's the F-Word. *IETF Journal*, 6(1).

[Júnior et al. 2010] Júnior, A. M., Laureano, M., Santin, A., and Maziero, C. (2010). Aspectos de segurança e privacidade em ambientes de Computação em Nuvem. In *Mini-curso - SBSeg 2010 - Fortaleza - CE*.

[Ma 2005] Ma, K. J. (2005). Web Services: What's Real and What's Not? *IT Professional*, 7(2):14–21.

[Mell and Grace 2010] Mell, P. and Grace, T. (2010). NIST Working Definition of Cloud Computing. Technical report. Disponível em: http://groups.google.com/group/cloudforum/web/nist-working-definition-of-cloud-computing. Acessado em 15 de Setembro de 2011.

[Menascé 2005] Menascé, D. A. (2005). Virtualization: Concepts, applications, and performance modeling.

[Microsoft 2011] Microsoft (2011). CardSpace. Disponível em: http://msdn.microsoft.com/en-us/library/aa480189.aspx. Acessado em 2 de Setembro de 2011.

[Murari 2010] Murari, K. e. a. (2010). *Eucalyptus Beginner's Guide - UEC Edition - Ubuntu 10.04 - Lucid*. CSS Corp.

[Nimbus 2011] Nimbus (2011). Nimbus - University of Chicago. Disponível em: http://www.nimbusproject.org/. Acessado em 2 de Setembro de 2011.

[Ning 2010] Ning (2010). Ning Inc. Disponível em: www.ning.com. Acessado em 23 de Agosto de 2011.

[OASIS 2005] OASIS (2005). eXtensible Access Control Markup Language (XACML) Version 2.0. Technical report. Disponível em: http://docs.oasis-open.org/xacml/2.0. Acessado em 9 de Setembro de 2011.

[OASIS 2006] OASIS (2006). UDDI 101. Disponível em: http://uddi.xml.org/uddi-101. Acessado em 5 de Setembro de 2011.

[OASIS 2008] OASIS (2008). Security Assertion Markup Language (SAML) V2.0 Technical Overview. Technical report. Disponível em: http://docs.oasis-open.org/security/saml/v2.0. Acessado em 17 de Agosto de 2011.

[OASIS 2011] OASIS (2011). SPML. Disponível em: http://www.oasis-open.org/committees/provision. Acessado em 9 de Setembro de 2011.

[OAuth Community 2010] OAuth Community (2010). The OAuth 1.0 Protocol. Technical report. Disponível em: http://tools.ietf.org/html/rfc5849. Acessado em 18 de Agosto de 2011.

[OAuth Community 2011] OAuth Community (2011). The OAuth 1.0 Guide. Disponível em: http://hueniverse.com/oauth/guide/intro/. Acessado em 2 de Agosto de 2011.

[Olden 2011] Olden, E. (2011). Architecting a Cloud-Scale Identity Fabric. volume 44, pages 52–59. Journal Computer - IEEE Computer Society.

[Open Grid Forum 2009] Open Grid Forum (2009). Occi - open cloud computing interfaces. Disponível em: http://occi-wg.org/. Acessado em 10 de Agosto de 2011.

[OpenID Foundation 2011] OpenID Foundation (2011). OpenID. Disponível em: http://openid.net/get-an-openid/what-is-openid/. Acessado em 2 de Setembro de 2011.

[OpenNebula 2010] OpenNebula (2010). OpenNebula. Disponível em: http://opennebula.org. Acessado em 10 de Setembro de 2011.

[OpenVZ 2010] OpenVZ (2010). OpenVZ Wiki. Disponível em: http://wiki.openvz.org. Acessado em 10 de Agosto de 2011.

[Oracle 2010] Oracle (2010). Virtualbox. Disponível em: http://www.virtualbox.org. Acessado em 10 de Agosto de 2011.

[Ping Identity 2010a] Ping Identity (2010a). About Identity Federation and SSO. Disponível em: http://pingidentity.com. Acessado em 20 de Setembro de 2011.

[Ping Identity 2010b] Ping Identity (2010b). OpenID Tutorial. Disponível em: https://www.pingidentity.com/resource-center/openid.cfm. Acessado em 20 de Setembro de 2011.

[Rackspace 2010] Rackspace (2010). Openstack cloud software. Disponível em: www.openstack.org. Acessado em 10 de Setembro de 2011.

[RNP 2011] RNP (2011). CAFe - Comunidade Acadêmica Federada. Disponível em: http://www.rnp.br/servicos/cafe.html. Acessado em 15 de Setembro de 2011.

[Salesforce 2010] Salesforce (2010). Disponível em: http://salesforce.com. Acessado em 2 de Setembro de 2011.

[Scavo and Cantor 2005] Scavo, T. and Cantor, S. (2005). Technical report. Disponível em: http://shibboleth.internet2.edu/docs/draft-mace-shibboleth-tech-overview-latest.pdf. Acessado em 13 de Agosto de 2011.

[Shelly and Frydenberg 2010] Shelly, G. B. and Frydenberg, M. (2010). *Web 2.0: Concepts and Applications*. Course Technology.

[Stallings 2011] Stallings, W. (2011). *Cryptography and Network Security: Principles and Practice*. Pearson Education.

[Stoller 2011] Stoller, S. D. (2011). XACML. Disponível em: http://www.cs.sunysb.edu/ stoller/cse608/6-XACML.pdf. Acessado em 14 de Agosto de 2011.

[Suess and Morooney 2009] Suess, J. and Morooney, K. (2009). Identity Management and Trust Services: Foundations for Cloud Computing. Technical report. Disponível em: http://www.educause.edu/node/178404. Acessado em 10 de Setembro de 2011.

[Switch 2011] Switch (2011). SWITCH - Serving Swiss Universities. Disponível em: http://www.switch.ch. Acessado em 2 de Setembro de 2011.

[Thangasamy 2011] Thangasamy, I. (2011). *OpenAM*. Packt Publishing.

[Twitter 2011] Twitter (2011). Using OAuth 1.0a. Disponível em: https://dev.twitter.com/docs/auth/oauth. Acessado em 20 de Setembro de 2011.

[Veras 2009] Veras, M. (2009). *Datacenter:Componente Central da Infraestrutura de TI*.

[Verdi et al. 2010] Verdi, F., Rothenberg, C. E., Pasquini, R., and Magalhães, M. F. (2010). Novas Arquiteturas de Data Center para Cloud Computing. *XXVIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos - Gramado - RS*.

[VMware Inc. 2011] VMware Inc. (2011). VMware. Disponível em: http://www.vmware.com. Acessado em 22 de Agosto de 2011.

[Wangham et al. 2010] Wangham, M. S., de Mello, E. R., da Silva Böger, D., Gueiros, M., and da Silva Fraga, J. (2010). Gerenciamento de Identidades Federadas. In *Mini-curso - SBSeg 2010 - Fortaleza - CE*.

[Warnke and Ritzau 2010] Warnke, R. and Ritzau, T. (2010). *qemu-kvm & libvirt*. Books on Demand GmbH.

[Windley 2005] Windley, P. (2005). *Digital Identity*. O'Reilly.

[Windows Azure 2010] Windows Azure (2010). Windows Azure Platform. Disponível em: http://www.microsoft.com/windowsazure. Acessado em 2 de Setembro de 2011.

[Xen 2010] Xen (2010). Xenserver. Disponível em: http://www.citrix.com. Acessado em 10 de Setembro de 2011.

[Zappert 2010] Zappert, F. e. a. (2010). Cloud Computing Use Cases White Paper - Version 4.0. Technical report.