ARLINDO LUIS MARCON JUNIOR

almjr

AVALIAÇÃO RESILIENTE DE AUTORIZAÇÃO *UCON*$_{ABC}$ PARA COMPUTAÇÃO EM NUVEM

Tese de doutorado apresentada ao Programa de Pós-Graduação em Informática aplicada da Pontifícia Universidade Católica do Paraná como requisito parcial para obtenção do título de Doutor em Informática.

CURITIBA

2013

ARLINDO LUIS MARCON JUNIOR

AVALIAÇÃO RESILIENTE DE AUTORIZAÇÃO $UCON_{ABC}$ PARA COMPUTAÇÃO EM NUVEM

Tese de doutorado apresentada ao Programa de Pós-Graduação em Informática aplicada da Pontifícia Universidade Católica do Paraná como requisito parcial para obtenção do título de Doutor em Informática.

Área de Concentração: *Ciência da Computação*

Orientador: Prof. Dr. Altair Olivo Santin

CURITIBA

2013

Dados da Catalogação na Publicação
Pontifícia Universidade Católica do Paraná
Sistema Integrado de Bibliotecas – SIBI/PUCPR
Biblioteca Central

Marcon Junior, Arlindo Luis

M321a
2013

Avaliação resiliente de autorização $UCON_{ABC}$ para computação em nuvem / Arlindo Luis Marcon Junior ; orientador, Altair Olivo Santin. – 2013.
131 f. : il. ; 30 cm

Tese (doutorado) – Pontifícia Universidade Católica do Paraná, Curitiba, 2013.
Bibliografia: f. [87]-92

1. Informática. 2. Cliente/servidor (Computadores). 3. Tecnologia da Informação. 4. Gerenciamento de recursos de informação. 5. Computadores – Controle de acesso. 6. Computação em nuvem. I. Santin, Altair Olivo. II. Pontifícia Universidade Católica do Paraná. Programa de Pós-Graduação em Informática. IV. Título.

CDD 20. ed. – 004

Pontifícia Universidade Católica do Paraná

ATA DE DEFESA DE TESE DE DOUTORADO
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

ÁREA DE CONCENTRAÇÃO: CIÊNCIA DA COMPUTAÇÃO

DEFESA DE TESE DE DOUTORADO Nº 018/2012

Aos 14 dias de dezembro de 2012 realizou-se a sessão pública de Defesa da Tese de Doutorado intitulada "**Avaliação Resilente de Autorização UCONABC para Computação em Nuvem**" apresentada pelo aluno **Arlindo Luis Marcon Junior** como requisito parcial para a obtenção do titulo de Doutor em Informática, perante uma Banca Examinadora composta pelos seguintes membros:

Prof. Dr. Altair Olivo Santin PUCPR (Orientador)	*(assinatura)*	Aprovado *(aprov/reprov.)*
Prof. Dr. Alcides Calsavara PUCPR		Aprovado
Prof. Dr. Luiz Augusto de Paula Lima Jr PUCPR		aprovado
Prof. Dr. Carlos Alberto Maziero UTFPR		aprovado
Prof. Dr. Lau Cheuk Lung UFSC		Aprovado

Conforme as normas regimentais do PPGIa e da PUCPR, o trabalho apresentado foi considerado Aprovado (*aprovado/reprovado*), segundo avaliação da maioria dos membros desta Banca Examinadora. Este resultado está condicionado ao cumprimento integral das solicitações da Banca Examinadora registradas no Livro de Defesas do programa.

Prof. Dr. Fabrício Enembreck
Diretor do Programa de Pós-Graduação em Informática

Agradecimentos

Inicialmente, quero começar agradecendo a Deus por todos os momentos de inspiração, ocasiões estas onde eu não conseguia encontrar a saída dos problemas. Situações estas que eu não sabia mais o que fazer.

Agradeço ao meu orientador de mestrado e doutorado, professor Dr. Altair Olivo Santin, pela confiança depositada neste aluno. Suas doses de cobrança e motivação foram fundamentais.

Quero agradecer profundamente o apoio incondicional prestado pelos meus pais, Arlindo Luiz Marcon e Lidia Ortigara Marcon. Se possível, também gostaria de pedir desculpas aos meus pais pelos vários momentos em que estive ausente.

Vale lembrar aqui das argumentações feitas pelos membros da banca, os professores, doutores, Alcides Calsavara, Luiz Augusto de Paula Lima Jr, Carlos Alberto Maziero e Lau Cheuk Lung. A contribuição deles foi de grande valor.

Gostaria de agradecer a todos os meus amigos e familiares que me apoiaram durante estes longos anos de estudo e trabalho. Quero registrar aqui que essa aventura foi desgastante, porém, muito proveitosa.

Reconheço, também, a singela colaboração fornecida pelo Instituto Federal do Paraná - IFPR. Fico muito grato pelo apoio prestado por todos os integrantes com os quais tive contato.

Por fim, mas não menos importante, citar a compreensão e o apoio prestado pela jovem Carolina K. Obrigado por ter me entendido, me apoiado, e me amado durante estes últimos meses. Espero, um dia, poder retribuir a todo o seu carinho.

A Deus Pai todo poderoso,
a Jesus Cristo seu único filho, nosso Senhor,
a meus pais, Arlindo e Lidia, pela confiança e apoio.

Sumário

Resumo

O controle de acesso direcionado aos negócios, utilizado atualmente na computação em nuvem, não atende, de maneira satisfatória, as necessidades de rastreamento do consumo de serviços, com granularidade fina (fine-grained), para os usuários. O $UCON_{ABC}$ aplica a reavaliação contínua de autorização, o que requer a contabilização para cada usuário do serviço. Assim, o esquema de reavaliação de autorização $UCON_{ABC}$ permite um controle de acesso fino para a computação em nuvem. O $UCON_{ABC}$, no entanto, não está preparado para coletar informações de consumo e reavaliar as autorizações no ambiente elástico provido pela computação em nuvem. Pois, durante a reavaliação contínua (periódica), uma condição de exceção de autorização pode ocorrer - disparidade entre os atributos de consumo e os atributos de autorização. Este trabalho tem o objetivo de prover resiliência à reavaliação contínua de autorização $UCON_{ABC}$, lidando com as condições de exceção individuais, enquanto o controle de acesso para o ambiente de computação em nuvem é mantido em níveis aceitáveis. Os experimentos realizados com um protótipo, como prova-de-conceito, mostram um conjunto de medidas para um cenário de aplicação (simulação de um e-commerce) e permitem a identificação das condições de exceção no período (ótimo) de reavaliação de autorização.

Palavras-Chave: Resiliência; Autorização; Controle de Uso; Condições de Exceção; Computação em Nuvem.

Abstract

The Business-driven access control used in cloud computing is not well suited for fine-grained user service consumption tracking. $UCON_{ABC}$ applies continuous authorization reevaluation, which requires usage accounting for each service's user. Thus, $UCON_{ABC}$ authorization revaluation scheme enables fine-grained access control for cloud computing. $UCON_{ABC}$, however, is not prepared for collecting usage accounting information and to reevaluate authorizations in an elastic environment like cloud computing. During a continuous (periodical) reevaluation an authorization exception condition may occur – disparity among usage accounting and authorization attributes. This work aims to provide resilience to the $UCON_{ABC}$ continuous authorization reevaluation, by dealing with individual exception conditions while maintaining a suitable access control in the cloud environment. The experiments made with a proof-of-concept prototype show a set of measurements for an application scenario (e-commerce) and allows for the identification of exception conditions in the best authorization reevaluation period.

Key-Words: Resilience; Authorization; Usage Control; Exception Conditions; Cloud Computing.

Lista de Figuras

Lista de Abreviaturas

Abreviatura	Referência	Tradução
ACL	*Access Control List*	Lista de Controle de Acesso
AOS	- - -	Arquitetura Orientada a Serviço
CC	*Cloud Computing*	Computação em Nuvem
CPU	*Central Processing Unit*	Unidade Central de Processamento
CRL	*Compromise Recovery List*	Lista de Recuperação de Chaves Comprometidas
CRM	*Customer Relationship Management*	Gestão do Relacionamento com o Cliente
DRM	*Digital Rights Management*	Gerenciamento de Direitos Digitais
GC	*Group Controller*	Controlador de Grupo
GKEK	*Group Key Encryption Key*	Chave para Cifrar o Segredo do Grupo
GKMP	*Group Key Management Protocol*	Protocolo para Gerenciar a Chave de Grupo
GTEK	*Group Traffic Encryption Key*	Chave para Cifrar o Tráfego do Grupo
HTTP	*Hypertext Transport Protocol*	Protocolo de Transporte de Hipertexto
IaaS	*Infrastructure as a Service*	Infraestrutura como um Serviço
ISO	*International Organization for Standardization*	Organização Internacional para Padronização
OASIS	*Organization for the Advancement of Structured Information Standards*	Organização para o Avanço de Padrões de Informação Estruturados
PaaS	*Platform as a Service*	Plataforma como um Serviço
PAP	*Policy Administration Point*	Ponto de Administração de Políticas

PC	*Permission Certificates*	Certificados de Permissão
PDP	*Policy Decision Point*	Ponto de Decisão de Políticas
PEP	*Policy Enforcement Point*	Ponto de Aplicação de Políticas
PIP	*Policy Information Point*	Ponto de Informação de Políticas
SaaS	*Software as a Service*	*Software* como um Serviço
SAML	*Security Assertion Markup Language*	Linguagem de Marcação para Assertivas de Segurança
SLA	*Service Level Agreement*	Acordo em Nível de Serviço
SOA	*Service Oriented Architecture*	Arquitetura Orientada a Serviço
SOAP	*Simple Object Access Protocol*	Protocolo Simples de Acesso a Objetos
SPML	*Service Provisioning Markup Language*	Linguagem de Marcação para Configuração de Serviços
STS	*Security Token Service*	Serviço de Credenciais Seguras
TI	*Information Technology*	Tecnologia da Informação
TS	*Tuple Space*	Espaço de *Tuplas*
UDF	*Usage Decision Facility*	Mecanismo de Decisão de Uso
UEF	*Usage Enforcement Facility*	Mecanismo de Aplicação do Uso
VM	*Virtual Machine*	Máquina Virtual
W3C	*World Wide Web Consortium*	Consórcio da *World Wide Web*
WS	*Web Services*	Serviços *Web*
WSDL	*Web Services Description Language*	Linguagem de Descrição de Serviços *Web*
XACML	*eXtensible Access Control Markup Language*	Linguagem de Marcação Extensível para Controle de Acesso
XML	*Extensible Markup Language*	Linguagem de Marcação Extensível

Capítulo 1

Introdução

1.1 *Motivação e Contextualização*

A consolidada arquitetura cliente-servidor representou um avanço significativo no uso eficaz dos recursos computacionais distribuídos. Porém, esta geralmente exige muito do lado servidor, dificultando a expansão, manutenção do mesmo e exigindo um conjunto razoável de programas para mantê-la funcional. Além disto, este paradigma demanda, em cada domínio, infraestrutura física, sujeitos capacitados para administração e desenvolvimento dos serviços providos, estratégias de *backup* e contenção, etc.

O modelo cliente-servidor tradicional foi concebido em uma relação de *um-para-um*, ou seja, um único Sistema Operacional instanciado em cada servidor físico. Isto evidencia recursos computacionais ociosos ou sobrecarregados, pois o dimensionamento do sistema precisa ser feito antecipadamente, presumindo uma demanda do servidor. Em suma, a necessidade de infraestrutura computacional traz consigo problemas que podem ser brevemente citados como: a dificuldade para administrar e manter *data-centers*; investimentos iniciais em *hardware* seguido da necessidade de atualizações e expansões constantes; previsão de atendimento de cargas de trabalho inesperadas (super-dimensionamento); gastos extras com eletricidade; necessidade de mão de obra qualificada; etc. (Douglis, 2009).

Com o passar dos anos, as tarefas computacionais começaram a ser migradas dos *desktops* e servidores corporativos para ambientes externos e terceirizados. No contexto atual, as aplicações estão sendo hospedadas em centros computacionais disponíveis e acessíveis na *Internet*. Esta mudança afeta todos os níveis do ecossistema computacional (*e.g.* usuários, desenvolvedores, gerentes de Tecnologia da Informação (*TI*), fabricantes de *hardware*) (Hayes, 2008).

Na topologia atual, disponibilizada pela computação em nuvem (*Cloud Computing - CC*), o cliente pode se comunicar com muitos servidores ao mesmo tempo, sendo que estes podem estar trocando informações entre si - não existe um único ponto central (*e.g. hub*). Usuários em geral e gerentes de Tecnologia da Informação (*TI*) têm em seu portfólio a possibilidade de utilizar infraestruturas, plataformas e sistemas terceirizados.

Na computação em nuvem, a carga de trabalho é executada em servidores virtualizados instanciados de acordo com a demanda (Foster, 2008). A abordagem utilizada diminui a quantidade de recursos computacionais ociosos, pois permite o redirecionamento dinâmico de recursos para as máquinas virtuais (*Virtual Machine - VM*) com mais necessidade computacional. Neste ambiente, temos uma relação de *n-para-um*, onde "*n*" representa a quantidade de Sistemas Operacionais virtualizados que podem ser executados em um único servidor físico.

A nuvem computacional é a base de um novo paradigma para o provimento de serviços sob demanda em ambientes distribuídos (*e.g.* armazenamento e processamento de dados de propósito geral, comunidades virtuais, compartilhamento de informações de negócios). Os serviços hospedados na nuvem podem seguir o modelo de fraco acoplamento proposto pela Arquitetura Orientada a Serviço (OASIS, 2006a), sendo implementados de maneira concreta com a tecnologia dos Serviços *Web* (Booth, 2004). A interação com os Serviços *Web* ocorre usando especificações padronizadas e amplamente utilizadas. Esta abordagem favorece a interoperabilidade, permitindo que diferentes sistemas, instanciados em plataformas distintas, troquem informações com um entendimento comum sobre os dados.

Como em qualquer abordagem nova, existem muitos assuntos inexplorados e que exigem a atenção dos pesquisadores e das empresas que interagem com a nuvem. A segurança desse contexto computacional é um assunto latente entre os provedores, consumidores e também na comunidade científica.

A interação entre diferentes domínios necessita utilizar sistemas de controle de acesso e contabilização de consumo com granularidade fina. Ou seja, é necessário regrar o acesso dos usuários, conhecendo detalhes sobre quem está acessando um determinado recurso e quanto está sendo utilizado. Atualmente existem diferentes provedores de computação em nuvem, sendo que cada um destes possui características próprias (Rimal, 2009). Caso o consumidor necessite utilizar funcionalidades de diferentes provedores, a tarefa de gerenciar o ambiente se torna extremamente árdua.

Administrar e aplicar políticas com granularidade fina, em sistemas distribuídos

que possuem as características citadas anteriormente, controlando dinamicamente o acesso aos recursos e serviços, exige uma abordagem flexível, escalável, dinâmica e segura (CSA, 2011). Na literatura científica são apresentadas as abordagens *outsourcing* (Westerinen, 2001) e *provisioning* (Westerinen, 2001), ambas as quais podem ser utilizadas para implantar os controles necessários ao ambiente de nuvem.

O controle de políticas *outsourcing* (modo de operação *pull* ou terceirização) é o mais amplamente empregado e encontrado na literatura. Este modelo segue o paradigma de controle centralizado (Westerinen, 2001). A arquitetura baseada em terceirização utiliza duas entidades principais: o Guardião do Serviço e o Monitor de Referência. Sempre que uma solicitação de acesso for enviada por um usuário, o Guardião intercepta a mensagem e envia um pedido ao Monitor de Referência, este é responsável por avaliar a solicitação e responder: acesso permitido ou negado. A abordagem de terceirização está baseada no acoplamento existente entre o Guardião e o Monitor de Referência.

Alternativamente à abordagem *outsourcing*, a arquitetura de efetivação de políticas *provisioning* (modo de operação *push* ou configuração) considera que a avaliação das políticas pode ser feita localmente ao domínio do Guardião do Serviço (Westerinen, 2001). Assim, quando o Guardião é inicializado (fase de *bootstrap*) este envia uma mensagem informando suas características para o Monitor de Referência. A operação de *bootstrap* é realizada na inicialização do Guardião, ou seja, quando este necessita ser configurado com as políticas adequadas.

Na abordagem *provisioning* não há a necessidade da troca constante de mensagens entre as entidades Guardião e Monitor de Referência para toda a solicitação de acesso. Porém, mesmo após o Guardião estar configurado com as políticas apropriadas, interações assíncronas podem ocorrer entre ambos, pois o Monitor de Referência continua sendo o responsável por receber, armazenar e, em alguns casos, interpretar as políticas.

A arquitetura de controle tradicional (*outsourcing*) facilita a gerência de políticas em um único ponto, aplicando um importante acoplamento funcional entre as entidades. A abordagem baseada em configuração (*provisioning*) é mais autônoma que a *outsourcing*, porém, nem sempre haverá políticas localmente disponíveis para avaliar o acesso de um usuário. O uso do modelo *provisioning* na configuração das políticas favorece a baixa dependência funcional entre as entidades distribuídas - como almejado pela Arquitetura Orientada a Serviço. No entanto, este modelo de operação propicia a violação "inconsciente" das políticas quando houver inconsistências entre os repositórios do Monitor de Referência e do Guardião.

A utilização dos serviços da nuvem precisa ser medida para que possa ser faturada para o consumidor. Frequentemente, o provedor pode verificar e contabilizar a utilização de um recurso fornecido para seus usuários. Porém, o consumidor (representando a entidade contratante) também precisa acompanhar a taxa de consumo dos serviços alocados para seu domínio. Com esta abordagem o arrendatário (contratante) poderia: monitorar se a cota contratada está sendo fornecida integralmente; dimensionar seus gastos ou alocar os serviços contratados de acordo com as exigências de seu ambiente; acompanhar o uso de cada usuário subordinado a seu domínio de gerenciamento.

Os ambientes de nuvem não fornecem as funcionalidades de gerenciamento adequadas para monitorar o serviço adquirido. Mesmo quando o provedor de nuvem implementa contratos em nível de serviço, os consumidores não podem utilizar estes em seus processos de negócio. Geralmente, o locatário não possui os mecanismos necessários para monitorar as cláusulas do contrato, tendo pouco ou nenhum acesso às operações de gerenciamento do provedor de nuvem (Bachtold, 2012). A integração de políticas e sistemas de controle, juntamente com esquemas de monitoramento de consumo, permite a criação de um processo transparente e automatizado para o gerenciamento dos serviços contratados.

Para cada serviço que o consumidor adquire, um *SLA* é necessário para assegurar que o montante de recursos contratado será fornecido corretamente (Goiri, 2011) (IBM, 2003). Portanto, a gestão do controle de acesso, baseada no modelo de computação em nuvem, deve ser escalável, suportando agentes de monitoramento coletando atributos de consumo em um número arbitrário de prestadores de serviços.

O controle de utilização com granularidade fina não é abordado pela computação em nuvem, sendo considerado uma tarefa do consumidor. Em suma, o domínio consumidor precisa desenvolver seus próprios meios de monitorar o consumo de serviços dos seus usuários. O controle fino para a nuvem não é devidamente investigado por propostas da literatura científica.

O consumidor que adquire os serviços da nuvem deve concentrar seus esforços no desenvolvimento do negócio, sem ter a responsabilidade de desenvolver toda a arquitetura de monitoramento e controle de consumo. Portanto, é desejável um modelo de gerenciamento que seja fornecido como uma plataforma de serviço (camada de controle intermediária), a qual possa tratar das questões de monitoramento e controle de uso dos serviços.

A demanda gerada por usuários do consumidor origina diferentes cenários de carga

para os recursos que dão suporte aos serviços fornecidos pelos provedores (*e.g.* processamento, armazenamento). Uma única seção de consumo pode necessitar dos serviços de diversos provedores de nuvem. Neste caso, o emprego da abordagem tradicional de configuração de políticas estáticas nos serviços pode caracterizar um cenário de subutilização de recursos em um provedor e a sobrecarga destes em outro. Em suma, não há como antecipar com precisão a demanda que um usuário vai gerar no serviço. Em um contexto ideal, as políticas de uso deveriam ser frequentemente avaliadas, visando detectar essa disparidade de consumo. A reconfiguração dinâmica das regras de acesso asseguraria a utilização uniforme dos recursos computacionais, sem prejudicar o acesso do usuário (Stihler, 2009).

Neste trabalho será empregado o controle de uso ($UCON_{ABC}$) para coletar atributos de consumo, consolidá-los e reavaliar as políticas. O $UCON_{ABC}$ estende o controle de acesso clássico, executando a avaliação contínua das regras de consumo para serviços, recursos ou usuários (Park, 2002) (Park, 2004). A continuidade do processo de utilização é concedida de acordo com a reavaliação das políticas. O uso pode ser entendido como operações de escrita em um objeto (*e.g.* um arquivo) ou o consumo de serviços (*e.g. e-commerce*) e recursos (*e.g.* ciclos de *CPU*).

Utilizar atributos de consumo atualizados faz parte do processo de reavaliação de políticas executado pelo $UCON_{ABC}$. Com uma periodicidade de coleta adequada (*ótima*), atributos de consumo mais consistentes (*i.e.* úteis dentro do período de tempo da reavaliação do acesso) serão utilizados pelo controle de uso $UCON_{ABC}$. A frequência em que ocorre a coleta de atributos reflete diretamente no período de tempo que o usuário estará violando a política de uso, situação esta que caracteriza uma exceção (*i.e.* disparidade entre a autorização de consumo concedida e a política vigente). Evidentemente, é desejável aplicar o menor intervalo de tempo para obtenção dos atributos de consumo e respectiva reavaliação das políticas de uso.

Esta proposta pretende determinar a frequência de monitoramento ideal (*i.e.* encontrar a relação de periodicidade ótima para a coleta de atributos de consumo e a reavaliação de políticas) visando diminuir os períodos de inconsistência de autorização (*i.e.* situação na qual o usuário está em condição de exceção). A abordagem visa prover resiliência ao processo de reavaliação de autorização contínua do $UCON_{ABC}$, lidando com as condições de exceção individuais, enquanto o controle de acesso para o ambiente de nuvem é mantido em níveis aceitáveis. Resiliência, nesta proposta, é habilidade de tratar exceções individuais em atributos de autorização enquanto o montante consumido está de

acordo com a quantidade definida em contrato.

1.2 *Objetivos*

O objetivo geral do trabalho é desenvolver uma abordagem para automatizar o gerenciamento de direitos de acesso, controlar o uso dos serviços em nuvens computacionais e manter uma visão administrativa consolidada das políticas utilizadas nos provedores que formam a nuvem.

Mais especificamente, este trabalho se propõe a implementar o modelo (*ongoing*) de autorização do $UCON_{ABC}$, provendo resiliência a reavaliação das políticas de uso. A resiliência, nesta proposta, deve ser entendida como uma maneira de prover ao modelo a habilidade de tratar algumas exceções em atributos de autorização referentes ao usuário. A resiliência no processo de autorização é suportada enquanto o *SLA* para o consumo do respectivo serviço está de acordo com a quantidade contratada.

Assim sendo, os objetivos específicos para esta proposta são os seguintes:

- Propor um ambiente intermediário que facilite a interação entre as entidades que utilizam a nuvem computacional.
- Propor um *PaaS* (*Platform as a Service*) orientado a segurança que suporte a consolidação de atributos de consumo e a reavaliação contínua (*ongoing*) das autorizações $UCON_{ABC}$.
- Criar um sistema de coleta (*accounting*) individualizada de atributos de consumo. O esquema de coleta deve utilizar o modelo de memória compartilhada distribuída (*i.e.* espaço de *tuplas*). Esta abordagem deve ser dinâmica e escalável, sendo compatível com o modelo de nuvens computacionais.
- Propor um esquema para a redefinição de políticas (*i.e.* re-escrita) em função da consolidação de atributos de consumo.
- Propor uma abordagem para identificar e tratar as exceções em atributos de autorização, provendo resiliência ao modelo de autorização (*ongoing*) $UCON_{ABC}$.
- Descobrir o melhor período de tempo para a coleta de atributos de consumo e a respectiva reavaliação das políticas de uso. A relação ótima entre a coleta e a reavaliação da autorização visa diminuir os períodos de inconsistência das políticas (*i.e.* divergências entre a autorização vigente e a política em execução).

- Estabelecer meios para monitorar a execução dos contratos – acordo de nível de serviço (*Service Level Agreement – SLA*). Os provedores oferecem seus serviços com base nos *SLAs* estabelecidos com seus arrendatários.
- Projetar e implementar um protótipo para a arquitetura proposta, juntamente com o desenvolvimento de um estudo de caso para testar o sistema.

1.3 *Contribuições*

A periodicidade (frequência) que um determinado usuário é monitorado interfere diretamente no processo de avaliação de políticas do modelo $UCON_{ABC}$. Toda a avaliação executada pelo Monitor de Referência (mediador da autorização) precisa ter acesso a atributos de consumo atualizados e individualizados (*i.e.* referente a cada usuário que está utilizando o serviço). Neste contexto, a periodicidade da reavaliação define a quantidade máxima de tempo que um determinado usuário pode se encontrar em uma condição incompatível com o sistema de controle de uso, caracterizando assim uma exceção (*i.e.* disparidade com a autorização de consumo).

Evidentemente, é desejável utilizar o menor intervalo de tempo possível para a obtenção dos atributos de consumo e a respectiva reavaliação das políticas de uso. Porém, com a utilização de uma frequência muito alta, o serviço vai sofrer uma carga de trabalho muito elevada. Assim sendo, uma contribuição desta proposta é encontrar a melhor frequência de contabilização para o monitoramento dos recursos, não gerando carga de trabalho extra, enquanto se reduzem os possíveis períodos de inconsistência referentes aos atributos de autorização - momento este que a condição de exceção pode ocorrer.

O ambiente de gerenciamento de uso deve ser flexível o suficiente para se adaptar as necessidades de cada domínio consumidor (*e.g.* tipo de recurso mais utilizado, granularidade desejada) (CSA, 2010). Os dados de consumo coletados dos provedores envolvidos são utilizados para alimentar o repositório de atributos e o Monitor de Referência - entidade que reavalia as políticas de controle de uso para cada usuário. O gerenciamento de acesso para a arquitetura de nuvem deve suportar agentes de monitoramento coletando atributos de consumo em vários provedores.

É importante ressaltar que o controle de uso com granularidade fina não é rastreado/disponibilizado pela computação em nuvem, sendo considerado uma responsabilidade do domínio consumidor. Porém, se o consumidor alocar uma

infraestrutura para concentrar seus esforços na implementação da lógica do negócio, não faz sentido que este precise se preocupar com o desenvolvimento de uma arquitetura customizada de controle de acesso e uso. Com base neste contexto, estamos propondo uma abordagem de controle que seja fornecida como uma plataforma (em nível *PaaS*) que vai tratar os aspectos de segurança.

O sistema de avaliação de políticas e o repositório de atributos, nesta proposta, utilizam os serviços implementados por um ambiente de memória compartilhada distribuída (*i.e.* serviço de espaço de *tuplas* (Galernter, 1985); *Apêndice A*). Este espaço de comunicação comum está hospedado em um *pool* de servidores virtuais. Com esta abordagem, os ambientes de contabilização de atributos e reavaliação de políticas se ajustam às demandas de uso, dimensionando a arquitetura de acordo com a necessidade de cada consumidor e respectivo conjunto de provedores. A flexibilidade do sistema de avaliação e do repositório de atributos é transparente para as entidades provedoras e consumidoras, impactando positivamente na expansão ou retração do sistema de controle de acesso.

A arquitetura de controle desenvolvida para gerenciar o contexto da nuvem permite a implantação de serviços, com fina granularidade, para autorização e monitoramento de atributos. Os dados coletados são consolidados em um ambiente de gerenciamento, sendo fornecidos imediatamente para o sistema administrativo do consumidor. Estes dados permitem a reconfiguração das políticas de uso e o monitoramento dos termos definidos no *SLA*.

Os serviços de gerenciamento são fornecidos por um ambiente federado hospedado na nuvem. A federação é o domínio utilizado para gerenciar o controle de uso, sendo compartilhada pelas entidades responsáveis por avaliar as políticas, tratar os atributos monitorados, administrar os *SLAs* e intermediar o acesso aos serviços. O *Broker* é a entidade que gerencia o contato inicial dos usuários com a nuvem sendo, também, o ponto de entrada do ambiente federado para provedores e consumidores. A federação agrega em seu domínio de gerenciamento todas as entidades com interesses convergentes.

1.4 *Organização do Texto*

A proposta está organizada de acordo com a seguinte estrutura: o **Capítulo 2** descreve o modelo de computação de nuvem. Este classifica as diferentes camadas de

acordo com os respectivos serviços oferecidos por cada uma; o **Capítulo 3** aborda itens de segurança que são fundamentais para desenvolvimento do projeto. Este capítulo apresenta conceitos sobre os contratos em nível de serviço, os modelos de controle de política e de uso, e as especificações que permitem a implementação de esquemas de segurança; o **Capítulo 4** mostra alguns esforços da literatura científica aplicados no gerenciamento de nuvens computacionais. Os trabalhos relacionados visam elencar os prós e contras de cada pesquisa, embasando e fortalecendo a necessidade de novos esquemas de gerência para ambientes distribuídos; o **Capítulo 5** expõe o modelo de gerenciamento proposto para a computação em nuvem; o **Capítulo 6** trata da implementação e avaliação do protótipo de acordo com o cenário utilizado; por fim, o **Capítulo 7** contém as conclusões e os trabalhos futuros.

Capítulo 2

Computação em Nuvem

2.1 *Introdução*

As origens do modelo de computação em nuvem (*Cloud Computing - CC*) podem ser rastreadas a partir do surgimento dos sistemas operacionais de tempo compartilhado ou multitarefa. Estas abordagens podem ser vistas como as facilitadoras do modelo utilizado pela nuvem (Bhattacharjee, 2009).

Algumas das principais diferenças da nuvem em relação a outros sistemas similares (*e.g. grid, cluster*) são: aparência de ter recursos computacionais inesgotáveis disponíveis sob demanda; eliminação do compromisso de consumo, permitindo que as empresas comecem pequenas e se desenvolvam de acordo com a necessidade; possibilidade de locar recursos computacionais de acordo com o uso (*e.g.* uma hora de processador), liberando estes quando não forem mais necessários. Basicamente, na computação em nuvem, os gastos com capital (*i.e.* infraestrutura) são convertidos em gastos operacionais (*i.e.* gerenciamento) (Armbrust, 2010).

A computação em nuvem fornece diferentes níveis de serviços (*i.e.* programas, plataformas, infraestrutura), sendo que estes podem ser acessados a partir de qualquer lugar da *Internet*. O ponto de acesso provê recursos computacionais sob-demanda para os usuários da nuvem. Os recursos físicos do provedor de nuvem podem ser compartilhados utilizando a tecnologia de virtualização. Este componente é essencial para a infraestrutura da nuvem, fornecendo facilidades no compartilhamento, gerenciamento e isolamento entre os ambientes de diferentes usuários (Rimal, 2009).

A computação em nuvem pode ser vista como uma pilha de serviços, sendo que cada camada fornece funcionalidades que são construídas com base nos níveis inferiores. Alguns serviços básicos como medição, faturamento e gerenciamento são necessários em todas as camadas (Yildiz, 2009).

2.2 *Computação em Nuvem*

Através da aplicação do modelo de nuvem (*Cloud Computing - CC*), tarefas computacionais podem ser migradas dos computadores de mesa e servidores corporativos para a nuvem computacional (Erickson, 2009). A mudança de paradigma marca a inversão de uma tendência que perdurou por muitos anos, afetando todos os níveis do ecossistema computacional, incluindo usuários, desenvolvedores, gerentes de Tecnologia da Informação (*TI*) e os fabricantes de *hardware* (Hayes, 2008).

A nuvem computacional pode ser utilizada para hospedar *softwares* em centros computacionais disponíveis e acessíveis via *Internet*. Na topologia atual não existe um ponto central, ou seja, um terminal cliente pode se comunicar com muitos servidores ao mesmo tempo, sendo que estes podem estar trocando informações entre si. A nuvem pode possuir uma ou mais centrais de gerenciamento, sendo que cada uma destas pode ser formada por vários provedores, administrando diferentes domínios. Novas funcionalidades para aplicações e serviços podem ser disseminadas a partir de uma central administrativa, sem que o consumidor tenha que se preocupar com a complexidade de gerenciamento do ambiente.

As entidades que fazem parte desse contexto computacional podem ser brevemente descritas como:

i. *Nuvem* – conjunto de provedores que fornecem diferentes níveis de serviços (*e.g.* programas, plataformas, infraestrutura);

ii. *Provedores* – gerenciam as infraestruturas computacionais que fornecem suporte aos serviços disponibilizados nas camadas superiores (*e.g.* serviço *web* de *e-commerce*). O provedor controla os recursos em nível físico (*e.g.* processador, memória) e os ambientes virtuais (*e.g.* máquina virtual).

iii. *Consumidor* – empresa que contrata e utiliza os serviços da nuvem;

iv. *Usuário* – entidade que pode estar vinculada a um domínio consumidor ou agindo por conta própria (*i.e.* o usuário final do serviço).

A migração de sistemas tradicionais para os serviços fornecidos pela nuvem pretende reduzir os custos de *TI* do consumidor, oferecendo vantagens como (Zhang, 2010): redução de gastos com manutenções e atualizações nas infraestruturas de

hardware e *software* - *i.e.* economia em licenciamento de sistemas, consumo de energia, resfriamento, armazenamento, rede, quantidade de servidores necessários; redução de trabalho na administração e no tempo de configuração de sistemas; diminuição das equipes de trabalho; desenvolvimento de aplicações com ciclo de vida mais curto e consequente redução do tempo de disponibilização de novos serviços no mercado; maior confiabilidade com custos menores.

Empresas que necessitam de serviços de *TI* estão considerando a possibilidade de utilizar provedores terceirizados (*off-premise*) tirando proveito das vantagens oferecidas pela nuvem. O ambiente da nuvem computacional é altamente escalável, permitindo demandar recursos adicionais mesmo se o número de usuários aumentar de maneira imprevisível.

Para facilitar a entrada de novas empresas no mercado, a computação em nuvem aplica o modelo de negócio *pay-as-you-go*. Ou seja, a cobrança é feita de acordo com a utilização dos serviços. Na nuvem computacional os gastos com capital são convertidos em gastos operacionais (Armbrust, 2009). Os serviços fornecidos pelo provedor de nuvem podem ser distribuídos e utilizados de maneira não uniforme, de acordo com a necessidade do consumidor. Na comunidade de rede há uma situação análoga, onde o preço é cobrado de acordo com o uso (*usage-based pricing*). A utilização da nuvem pelos consumidores é contabilizada pelo provedor para que a quantidade de recursos computacionais utilizados (*e.g.* processador, memória) possa ser faturada (Bhattacharjee, 2009).

A nuvem permite que os consumidores utilizem a quantidade de recursos necessários para realizar testes com novos sistemas. Se um projeto falhar durante sua fase inicial, por exemplo, o consumidor da nuvem investiu pouco no negócio, podendo facilmente alterar seu ramo de atividade. No modelo tradicional (*on-premise*) o contratante precisa gastar previamente em licenças, infraestrutura, consultoria etc., e se o projeto falhar o investimento feito vira prejuízo.

Com a aplicação da computação em nuvem e das respectivas tecnologias de suporte (*e.g.* virtualização), a utilização dos recursos é maximizada, reduzindo o tempo ocioso de cada máquina. A camada de virtualização abstrai o *hardware*, intermediando o acesso de várias aplicações aos mesmos recursos físicos: processador, memória etc. Esta tecnologia permite realizar a consolidação (agrupamento) de cargas de trabalho em poucos servidores físicos, reduzindo os gastos com energia, resfriamento e consequentemente propiciando economia em *hardware*, manutenção etc. (Bhattacharjee, 2009).

A técnica de virtualização permite que sistemas hospedados em um servidor físico sejam transferidos para outras máquinas, executando o balanceamento de carga ou cópias de segurança dos sistemas. Com esta abordagem, a execução ou restauração de cópias de segurança é concluída em uma pequena fração do tempo que levaria com os servidores físicos tradicionais. No caso de falhas na aplicação ou serviço, uma instância de *backup* (*hot backup*) pode assumir imediatamente o lugar da faltosa. Neste caso, a interrupção no serviço pode passar despercebida para os usuários (Kandukuri, 2009).

Empresas que optarem por utilizar um provedor de computação em nuvem podem basear suas escolhas em vários fatores: preço, confiabilidade, disponibilidade, abrangência, suporte etc. Mesmo que uma única entidade atenda estes requisitos, outros provedores podem ser utilizados, fornecendo diferentes conjuntos de funcionalidades (*e.g.* armazenamento da *Amazon*, poder computacional do *Google* e CRM (*Customer Relationship Management*) da *Salesforce*).

De maneira geral, o modelo de computação em nuvem visa prover acesso sob demanda para diferentes camadas da plataforma computacional (*e.g.* rede, servidores, armazenamento, aplicações, serviços etc.). Estas funcionalidades podem ser rapidamente fornecidas ou liberadas com pouco esforço de gerenciamento ou interação manual. A nuvem visa fornecer alta disponibilidade e elasticidade, sendo norteada por cinco características fundamentais (Mell, 2009; Figura 2.1):

1. *Auto-Atendimento*: o consumidor configura um determinado recurso computacional de acordo com a sua necessidade, sem exigir interação humana com os provedores de serviço.
2. *Amplo acesso à rede*: os recursos são disponibilizados na rede e acessados através de mecanismos padronizados. Isto possibilita o uso em diferentes plataformas (*e.g.* celulares, *notebooks* etc.).
3. *Pool de recursos*: os recursos computacionais do provedor são agrupados. Isto permite servir múltiplos consumidores em um modelo multi-inquilino (*multi-tenant*). Ou seja, os recursos físicos são distribuídos e ou redistribuídos dinamicamente de acordo com a demanda do consumidor.
4. *Elasticidade*: recursos podem ser fornecidos rapidamente e, em alguns casos, automaticamente. A quantidade de recursos disponibilizados passa para o consumidor a impressão de que a nuvem possui uma infraestrutura ilimitada.
5. *Medição no uso dos serviços*: a nuvem controla e otimiza o uso de recursos,

fornecendo métricas de acordo com o tipo de serviço sendo prestado. Tanto o provedor quanto o consumidor podem monitorar e controlar a utilização dos recursos.

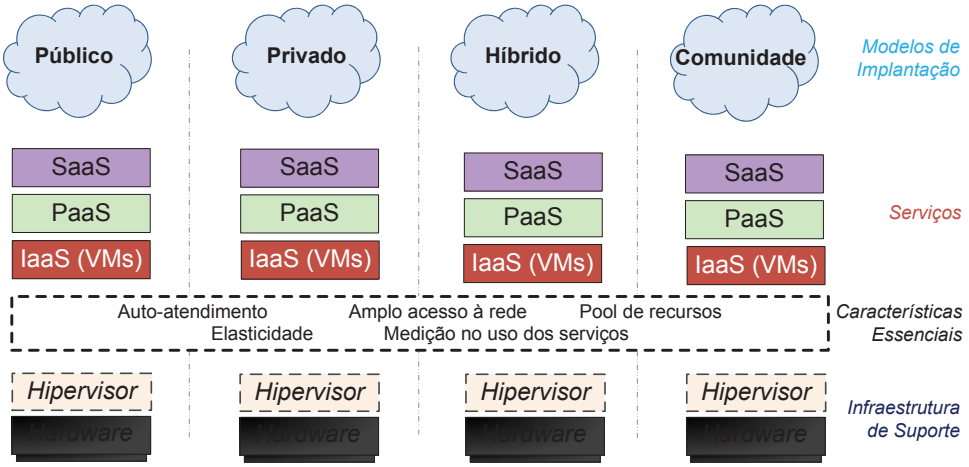

Figura 2.1: Visão geral da computação em nuvem.

2.2.1 Sistemas, plataformas e infraestruturas

Os provedores de computação em nuvem podem ser classificados de acordo com o tipo de serviço oferecido e o respectivo modelo de implantação (Mell, 2009). Os principais modelos de serviço são (Figura 2.1):

1. *Software como um Serviço* (*SaaS*): oferece o produto final da computação em nuvem, o *software* que o consumidor usa. Neste nível o consumidor não gerencia ou controla a infraestrutura – rede, servidores, sistema operacional, armazenamento e funcionalidades das aplicações. O consumidor simplesmente utiliza as aplicações sendo executadas na infraestrutura da nuvem. As aplicações podem ser acessadas através de um *thin client* – navegador *web*, por exemplo.
2. *Plataforma como um Serviço* (*PaaS*): oferece suporte para o consumidor implantar na infraestrutura da nuvem suas próprias aplicações, desde que utilizem linguagens de programação e ferramentas compatíveis com o provedor. Neste nível, o consumidor gerencia sua própria aplicação e o sistema que está hospedando as configurações do ambiente, não tendo controle sobre a infraestrutura subjacente – rede, servidores, armazenamento.

3. *Infraestrutura como um Serviço* (*IaaS*): fornece recursos computacionais básicos como processamento, armazenamento, rede etc. Com estes recursos o consumidor pode implantar e executar uma grande variedade de programas – sistemas operacionais e seus aplicativos. Nesta camada, o consumidor não gerencia a infraestrutura física da nuvem, mas tem controle limitado sobre componentes da rede, como por exemplo, filtros de pacotes.

A computação em nuvem pode ser vista como uma pilha de serviços (Figura 2.1), sendo que cada camada oferece funcionalidades que são construídas com base nos níveis inferiores. Alguns componentes básicos se espalham pelos vários níveis, sendo necessários para todos os tipos de serviços oferecidos na nuvem (*e.g.* mecanismos de medição, contabilização, gerenciamento).

A camada *IaaS* abrange toda a pilha de recursos da infraestrutura – desde as instalações físicas até as plataformas de *hardware* disponíveis. Esse nível incorpora a funcionalidade de abstração de recursos, possibilitando a conectividade entre as camadas física e lógica.

A camada *PaaS* – posicionada acima do *IaaS* – adiciona um nível de integração com *frameworks* de desenvolvimento de aplicações e funcionalidades de *middleware*. Essa camada provê funções como banco de dados e recursos para troca e enfileiramento de mensagens, permitindo aos desenvolvedores a construção de aplicações sobre o *PaaS*. As linguagens de programação e as ferramentas utilizadas precisam ser suportadas pela camada subjacente.

A camada *SaaS* – posicionada acima do *PaaS* – provê um ambiente de operação auto-contido. Essa é utilizada para disponibilizar diferentes serviços para seus consumidores (*e.g.* conteúdos, aplicações, funcionalidades de gerenciamento, etc.).

As nuvens computacionais podem ser classificadas de acordo com os seguintes modelos básicos de implantação (Mell, 2009; Figura 2.1):

1. *Nuvem privada*: a infraestrutura é operada exclusivamente para atender as necessidades de uma organização, sendo que essa pode ser gerenciada pela própria entidade ou por um terceiro. O esquema de implementação pode ser local ou remoto.
2. *Nuvem baseada em comunidade*: compartilhada por várias organizações que

possuem interesses comuns – requisitos de segurança, políticas etc. Essa estrutura pode ser gerenciada pelas entidades participantes da comunidade ou por um terceiro, em implementação local ou remota.

3. *Nuvem pública*: a infraestrutura é disponibilizada para o público em geral, podendo pertencer a alguma organização que vende serviços de computação em nuvem.
4. *Nuvem híbrida*: é uma composição entre dois ou mais modelos de nuvens, por exemplo, privado e público. Estes permanecem como entidades únicas, porém, são ligados por alguma tecnologia específica – padronizada e aberta ou proprietária. Uma composição de nuvem híbrida pode viabilizar o balanceamento de carga, ou seja, quando a parte privada não consegue mais atender a demanda a parte pública pode fazer esta tarefa – se os dados não forem sensíveis.

2.3 *Conclusão*

A computação em nuvem consolida sob um único conceito características interessantes provenientes de diferentes abordagens (*e.g. grid, utility computing, cluster*). O modelo foi idealizado de acordo com as seguintes premissas: ser altamente escalável, provendo recursos computacionais que podem ser dinamicamente configurados e entregues sob-demanda; fornecer diferentes níveis de serviços para os consumidores; ser estimulada e impulsionada pelo crescimento econômico.

O capítulo seguinte aborda alguns itens de segurança relacionados com o desenvolvimento desta pesquisa. Nessa seção, conceitos básicos sobre gerenciamento, criação e aplicação de políticas de controle são explanados.

Capítulo 3

Aspectos de Segurança e Computação em Nuvem

3.1 *Introdução*

A computação em nuvem deixa em aberto algumas questões com relação a privacidade, segurança e confiabilidade. Permitir que um terceiro tome conta de informações pessoais levanta questões sobre controle e propriedade (*e.g.* interoperabilidade, retenção dos dados por parte de um terceiro) (Hayes, 2008).

A segurança dos dados é um dos itens mais citados em oposição à ampla utilização do modelo de nuvem, pois a responsabilidade pela proteção do ambiente é dividida entre as partes. O consumidor deve proteger seus dados em nível de aplicação enquanto o provedor de nuvem deve garantir a segurança física, controlar os filtros de pacotes e isolar um usuário do outro (*e.g.* proteção contra ataques de negação de serviço ou roubo de informações).

3.2 *Contrato em Nível de Serviço*

O *SLA* (*Service Level Agreement*) pode ser descrito como o documento resultante de uma negociação entre o consumidor e o fornecedor do serviço (IBM, 2003). Este documento pode especificar itens como: nível de disponibilidade e desempenho, operações permitidas, cotas de uso para um consumidor, etc. Geralmente, o contrato é escrito utilizando uma terminologia de alto nível, a qual não pode ser nativamente interpretada por mecanismos ou algoritmos de controle. Sendo assim, os itens do *SLA* devem ser convertidos em documentos que sejam entendíveis pelos sistemas que vão

efetuar a monitoração ou aplicação das cláusulas do contrato.

Cada provedor de nuvem possui uma arquitetura de acordo com a camada em que atua (*e.g. SaaS*, *PaaS*, *IaaS*), sendo que cada serviço possui suas próprias questões de segurança. O *SLA* deve descrever claramente os objetivos para cada camada, enquanto o consumidor precisa entender as regras do contrato para que possa implementar suas políticas de segurança. O *SLA* é um acordo legal entre fornecedores e clientes.

Os principais itens de um contrato podem ser brevemente descritos como (Kandukuri, 2009):

a) descrição do serviço e a maneira como este será fornecido. Estas informações devem ser precisas, contendo especificações detalhadas do que está sendo entregue;
b) tratar da monitoração do desempenho e consumo em nível de serviço. Essencialmente, deve ser possível medir o fornecimento do serviço, analisando os resultados e gerando relatórios - os objetivos e métricas utilizadas devem ser especificadas no próprio acordo;
c) tratamento de incidentes ou problemas adversos. Especifica os processos para tratar e resolver incidentes bem como as respectivas atividades preventivas que coíbam a ocorrência destes;
d) definição das responsabilidades do consumidor quanto a utilização dos serviços. O contratante deve gerenciar os usuários que acessam os serviços estipulados no contrato - *e.g.* controlar o acesso físico/lógico das instalações e informações. Da mesma maneira, o provedor deve cumprir as políticas e procedimentos de segurança estipulados pelo consumidor;
e) recuperação de desastres e o plano de continuidade dos negócios. Define itens essenciais que devem ser refletidos no *SLA*.

3.3 *Modelos de Controle de Políticas Push e Pull*

Os modelos de controle de políticas são utilizados para distinguir qual esquema de interação será utilizado pelas entidades que gerenciam e avaliam as políticas do ambiente. A abordagem *push* é utilizada para caracterizar entidades ativas, que não dependem de fatores externos ao seu contexto. Quando necessário, estas entidades podem entrar em contato com serviços externos ao seu domínio, interpretar as respostas devolvidas, ou

enviar informações adicionais. Na abordagem *pull* a entidade é passiva, delegando para um terceiro a responsabilidade de interpretar e avaliar todas as interações que ocorrem no ambiente (*e.g.* envio de informações de autorização, atributos). Os modelos de avaliação de políticas baseados em provisionamento (*push*) e terceirização (*pull*) são exemplos destes esquemas de interação.

O modelo de controle de políticas *outsourcing* (Westerinen, 2001) utiliza duas entidades centrais, sendo estas o ponto de aplicação de política ou Guardião (*Policy Enforcement Point – PEP*; Figura 3.1) e o ponto de decisão de política ou Monitor de Referência (*Policy Decision Point – PDP*). As entidades citadas trocam mensagens a cada solicitação enviada por usuários do ambiente.

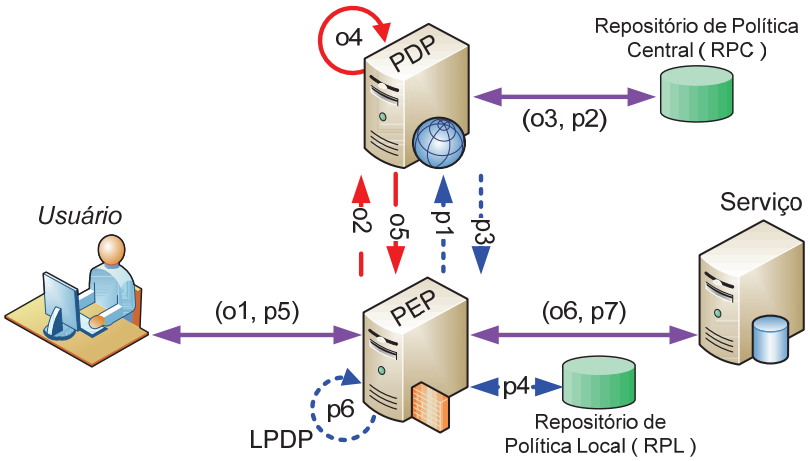

Figura 3.1: Modelos de Controle de Políticas.

A interação ocorre no momento em que o solicitante envia um pedido para acessar seu objetivo (*e.g.* serviço ou recurso; *evento o1*; Figura 3.1). Todas as solicitações são interceptadas pelo *PEP* e enviadas para o *PDP* (*pull*; *evento o2*). O Monitor de Referência, com base nas regras recuperadas do Repositório de Política Central (*RPC*; *evento o3*), informações do usuário e recurso alvo, toma uma decisão e encaminha esta para o Guardião (*eventos o4* e *05*, respectivamente). O *PEP* aplica a decisão de permitir (*evento o6*) ou negar o acesso de acordo com a resposta recebida do *PDP*. Neste modelo, a comunicação entre *PEP* e *PDP* deve ser ininterrupta, pois se houver falhas, o Guardião fica impossibilitado de tomar decisões sobre as solicitações de acesso.

O modelo de controle de políticas *provisioning* (Westerinen, 2001) utiliza as mesmas entidades centrais, porém, *PEP* e *PDP* não necessitam interagir constantemente para avaliar todas as solicitações de acesso recebidas (*evento p5*; Figura 3.1). Durante a

inicialização do Guardião, este interage com o Monitor de Referência visando receber as regras adequadas ao seu contexto (*push*; *evento p1*). As políticas recuperadas do repositório do *PDP* (*evento p2*) são enviadas e armazenadas no Repositório de Política Local (*RPL*; *eventos p3* e *p4*) para que as decisões de acesso possam ser avaliadas pelo *LPDP* (*Local Policy Decision Point*; *evento p6*); *i.e.* sem a necessidade do intercâmbio de dados com o *PDP*.

O Monitor de Referência é a entidade que gerencia as políticas do ambiente, sendo responsável por atualizar o armazenamento local ao *PEP* sempre que houver inconsistências entre o Repositório de Política Local (*RPL*) e o Repositório de Política Central (*RPC*). Mesmo após o Guardião já estar configurado com as respectivas políticas, novos recursos podem ser disponibilizados no ambiente ou usuários desconhecidos podem requisitar acesso aos recursos. Este contexto necessita de interações entre *PEP* e *PDP* visando obter uma determinada política ou uma avaliação de acesso.

3.4 *Controle de Uso*

O controle de uso (*UCON*) é mais completo que o esquema de controle de acesso clássico. Esse modelo aborda itens como obrigações, condições, continuidade dos controles e a mutabilidade dos atributos (Sandhu, 2003). Obrigações são requisitos que devem ser cumpridos pelo sujeito para que o acesso seja permitido. Condições são restrições do ambiente, independentes do sujeito ou objeto. Estas devem ser satisfeitas para que o acesso possa ser efetivado. Atualmente, com ambientes distribuídos altamente dinâmicos (*e.g.* computação em nuvem), obrigações e condições são fatores de decisão que devem ser considerados nos controles que gerenciam a utilização dos recursos digitais.

As decisões de autorização na abordagem tradicional geralmente são tomadas no momento da solicitação de acesso e, tipicamente, não reconhecem controles contínuos para os acessos com tempo de vida relativamente longo ou, para a revogação imediata da utilização. Adicionalmente, questões referentes a mutabilidade (*i.e.* as atualizações em atributos relacionados ao sujeito ou objeto como consequência do acesso) não têm sido exploradas de maneira sistemática.

O esquema de controle *UCON* define uma família de modelos *ABC* (*i.e. authorizations* - *A*, *obligations* - *B* e *conditions* - *C*) como sendo os requisitos

fundamentais para o controle de uso. O modelo *ABC* é constituído dos seguintes fatores de decisão: autorizações - *A*, baseada nos atributos do sujeito e do objeto; obrigações - *B*, necessitam de alguma ação executada pelo sujeito de modo que este possa obter ou manter o acesso, *e.g.* aceitar um contrato de licença de uso; e condições - *C*, fatores do sistema ou ambiente que interferem no acesso, *e.g.* carga do processador.

A abordagem *UCON* permite estender o controle de acesso tradicional (*i.e.* obrigatório/mandatário, discricionário, baseado em papéis), integrando as obrigações, condições e autorizações, e incluindo no modelo as propriedades de continuidade e mutabilidade (Park, 2002) (Park, 2004). No que diz respeito à autorização, o modelo considera os atributos mutáveis que são alterados como consequência do acesso. Adicionalmente, o *UCON* reconhece a continuidade do cumprimento das políticas de controle, visto que a decisão de permitir a utilização não é avaliada somente antes do acesso, mas também durante o intervalo de tempo em que o uso está ocorrendo.

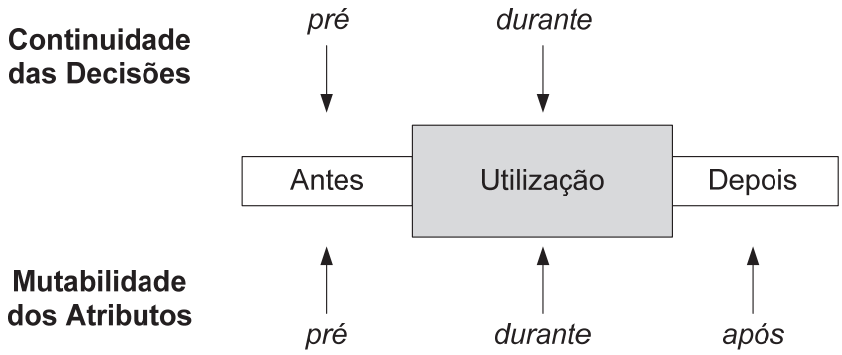

Figura 3.2: Propriedades de Continuidade e Mutabilidade. Adaptado de Sandhu, 2003.

No controle de acesso tradicional, a decisão de autorização é avaliada antes do acesso ser executado (*pré - pre*; Figura 3.2). No entanto, é bastante admissível que esta abordagem seja estendida para permitir a aplicação contínua, avaliando as políticas de uso durante todo o período de utilização do recurso (*durante - on*). Esta propriedade é chamada de continuidade, sendo utilizada para controlar o acesso a um objeto - acesso este que pode se estender por um longo período de tempo - ou para a revogação imediata de sua utilização.

Na abordagem aplicada pelo controle de acesso tradicional, os atributos são modificados somente por ações administrativas. Porém, em aplicações modernas (*e.g. Digital Rights Management - DRM*), os atributos precisam ser atualizados como um

efeito colateral, referente as ações do sujeito. No caso de atributos mutáveis, as atualizações podem ser feitas antes (*pre*), durante (*on*) ou após (*post*) a utilização do objeto. Sistemas de informação modernos necessitam utilizar propriedades de controle adicionais, considerando a continuidade do acesso e a mutabilidade dos atributos.

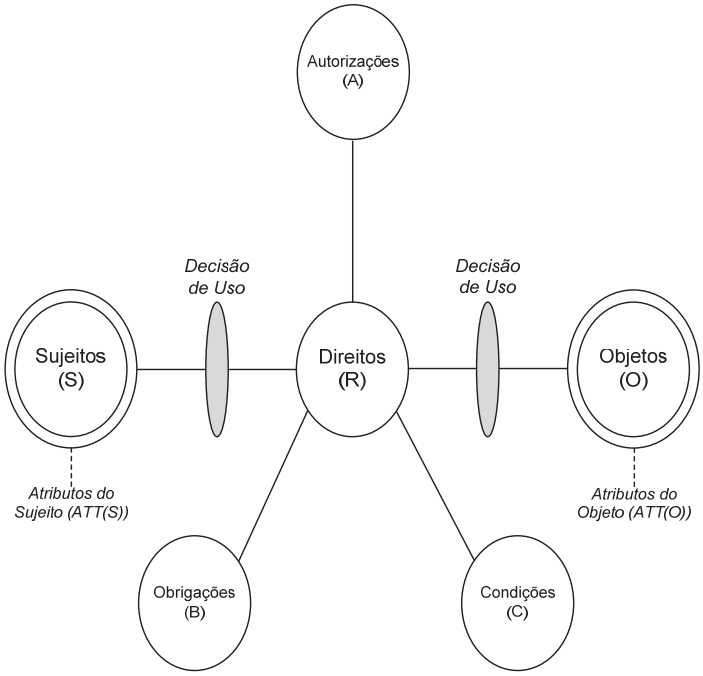

Figura 3.3: Componentes do modelo *ABC*. Adaptado de Sandhu, 2003.

O modelo *ABC* é composto por oito componentes: sujeitos, atributos do sujeito, objetos, atributos do objeto, direitos, autorizações, obrigações e condições (Figura 3.3). Sujeitos e objetos são conceitos conhecidos no controle de acesso, sendo utilizados com as mesmas funcionalidades no modelo *ABC*. Direito permite o acesso de um sujeito à um objeto em um contexto particular (*e.g.* leitura, escrita), similar ao conceito utilizado no controle de acesso clássico. O modelo *ABC* não visualiza o direito como existindo em alguma matriz de acesso, independentemente da atividade do sujeito, ao invés disto, a existência do direito é determinada quando uma tentativa de acesso é efetuada pelo sujeito. A função de decisão de uso avalia a solicitação de utilização levando em consideração as informações existentes no momento do pedido (*i.e.* os atributos do sujeito, atributos do objeto, autorizações, obrigações e condições).

Atributos do sujeito (*e.g.* identidade, papéis, saldo da conta) e do objeto (*e.g.* rótulos de segurança, lista de controle de acesso) são propriedades que podem ser utilizadas durante o processo de decisão de acesso. Uma inovação significativa do modelo

ABC é que os atributos do sujeito e do objeto podem ser mutáveis. Ou seja, estes podem ser alterados como consequência do acesso (*e.g.* políticas que decrementam o saldo da conta do sujeito de acordo com o número de acessos a um objeto). Os atributos imutáveis podem ser alterados somente por ações administrativas.

Autorizações, obrigações e condições são fatores de decisão aplicados pelas funções de decisão de uso para determinar se o sujeito tem permissão de acessar um objeto que está atrelado a um direito em particular. As autorizações são tomadas considerando os atributos do sujeito, atributos do objeto e o direito em questão. A autorização normalmente é necessária antes do acesso, mas, além disso, é possível executar um processo de autorização contínua durante o acesso ao objeto (*e.g.* verificação periódica de uma lista de revogação de certificados enquanto o acesso está ocorrendo). Caso alguma inconsistência seja verificada (*e.g.* o certificado apareceu na lista de revogação) o acesso pode ser cancelado. As autorizações podem necessitar de atualizações em atributos do sujeito ou do objeto, podendo ocorrer antes (*pre*), durante (*on*) ou após (*post*) o uso do objeto.

Obrigações são exigências que o sujeito deve executar antes (*e.g.* fornecer informações pessoais antes de acessar uma pagina *web*) ou durante o acesso (*e.g.* manter uma janela de anúncio aberta enquanto utiliza algum serviço *on-line*). Os atributos do sujeito e do objeto podem ser utilizados para decidir quais obrigações são necessárias para a liberação do acesso. As obrigações também podem ser empregadas para atualizar atributos mutáveis, podendo afetar as decisões de utilização atual ou futuras.

Condições são fatores de decisão do ambiente ou do sistema (*e.g.* hora atual, carga do sistema, estados de segurança: normal, sob ataque). As condições não estão sobre o controle direto dos sujeitos, sendo que a avaliação dessas não pode ser utilizada para atualizar atributos do sujeito ou do objeto.

Um dos assuntos mais críticos na aplicação do *UCON* é o Monitor de Referência. Esta entidade avalia as políticas e regras de controle, fornecendo a respectiva decisão de acesso para o sistema que gerencia a utilização dos objetos digitais. Os sujeitos acessam os objetos somente se estiverem autorizados pelo Monitor de Referência.

O Monitor de Referência *UCON* possui alguns itens diferentes se comparado com o *framework* de controle de acesso proposto pelo padrão *ISO* (*International Organization for Standardization*) (ISO, 2006). O monitor *UCON* (Figura 3.4) é composto por um módulo de decisão de uso (*Usage Decision Facility - UDF*) e um módulo de execução do uso (*Usage Enforcement Facility - UEF*) (Sandhu, 2003). O *UDF* possui componentes

funcionais referentes as condições, obrigações e autorizações. A parte responsável pela autorização executa um procedimento similar ao esquema de controle tradicional, utilizando as regras de uso e os atributos do sujeito e do objeto para verificar se a solicitação está autorizada ou não. O módulo *UDF* pode retornar *meta-dados* referentes à autorização de acesso e os direitos de uso permitidos para o objeto. Os *meta-dados* são utilizados pelo módulo de customização do *UEF* para personalizar o acesso ao objeto solicitado.

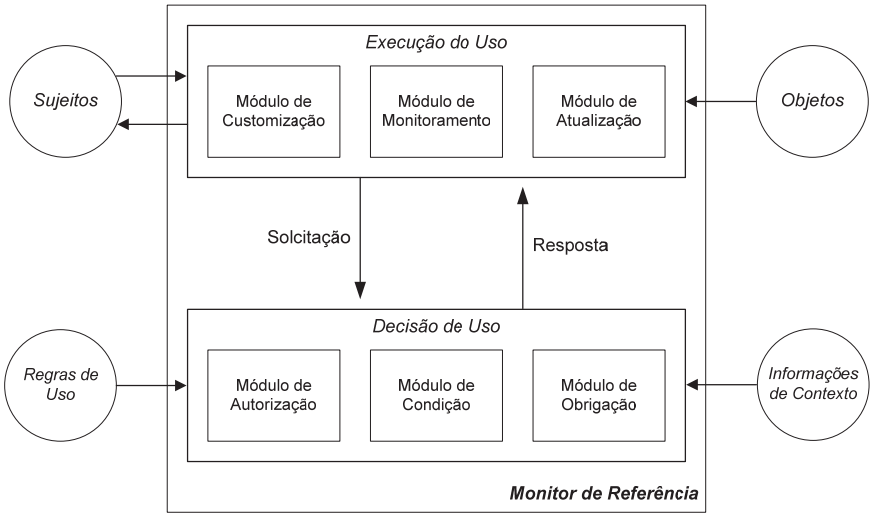

Figura 3.4: Estrutura conceitual do Monitor de Referência *UCON*. Adaptado de Sandhu, 2003.

O módulo de condição avalia se os requisitos do ambiente, para uma solicitação autorizada, foram satisfeitos de acordo com as regras de uso e informações do contexto (*e.g.* hora atual). O módulo de obrigação analisa se as restrições foram cumpridas antes ou durante a utilização ser realizada. Se existir alguma obrigação a ser exercida, esta deve ser acompanhada pelo módulo de monitoramento, sendo que o resultado deste acompanhamento precisa ser resolvido pelo módulo de atualização pertencente ao *UEF*.

3.5 *Protocolo para Gerenciamento de Chave de Grupo*

A especificação *GKMP* (*Group Key Management Protocol*) propõe um protocolo para criar chaves simétricas agrupadas e distribuir estas entre os pares de uma comunicação (Harney, 1997a). O protocolo visa ser: virtualmente invisível para o operador; a distribuição das chaves não necessita ser centralizada; somente os membros

do grupo devem possuir a chave; receptor e remetente são orientados a operação; pode utilizar a comunicação *multicast*.

A especificação descreve os seguintes itens: os papéis das entidades do sistema no gerenciamento das chaves utilizadas; a definição dos requisitos funcionais e de segurança do sistema; uma explanação detalhada sobre o protocolo de gerenciamento de chaves de grupo. Este protocolo fornece a habilidade de criar e distribuir chaves criptográficas para grupos de tamanho arbitrário, sem a intervenção de um gerenciador de chave global. O *GKMP* combina técnicas desenvolvidas para a criação de pares de chaves com métodos usados para distribuir as chaves a partir de um centro de distribuição (*i.e.* entidade responsável por enviar a chave para o grupo de *hosts*).

O protocolo *GKMP* possui as seguintes funcionalidades: criar chaves para grupos criptográficos; distribuir as chaves para os membros do grupo; assegurar o controle de acesso baseado em papéis para as chaves; impedir o acesso de *hosts* comprometidos; permitir o controle hierárquico das ações do grupo. O conceito de criação de chaves utilizado pelo *GKMP* é a geração cooperativa entre duas entidades que fazem parte do protocolo. Os algoritmos utilizados na fase de concepção utilizam a tecnologia de chaves assimétricas (*e.g. RSA*, *Diffe-Hellman*, *Elliptic Curves*) para passar informações entre duas entidades e criar uma única chave criptográfica.

O *GKMP* distribui as chaves de grupo para entidades qualificadas, pertencentes ao contexto do protocolo. As entidades encaminham certificados de permissão (*Permission Certificates - PC*) como parte do processo de distribuição da chave de grupo. Os *PCs* contêm informações de controle de acesso sobre um domínio particular. Estas informações são atribuídas por uma autoridade hierarquicamente superior que assina os certificados de permissão. Com esta abordagem, cada entidade pode verificar as permissões de qualquer entidade do *GKMP* e comparar estas com o nível de serviço sendo solicitado.

A especificação *GKMP* utiliza uma lista de entidades comprometidas que é distribuída para os membros do grupo durante as ações de gerenciamento de chaves. Em essência, a lista (*Compromise Recovery List - CRL*) permite que os membros do grupo se desconectem das entidades afetadas. O gerenciamento dos grupos é delegado para entidades controladoras específicas visando facilitar o processo de distribuição das *CRLs*.

O *GKMP* permite o controle das ações do grupo através de uma autoridade hierarquicamente superior. O protocolo permite que esta autoridade ordene remotamente as ações do grupo. As ordens são assinadas pela autoridade e verificadas por todas as

entidades envolvidas com o grupo. O *GKMP* atua em nível de aplicação, sendo independente do protocolo de comunicação subjacente.

A criação e distribuição de chaves de grupo necessita da atribuição de papéis (Harney, 1997b). Esta abordagem serve para identificar quais funções os *hosts* individuais executam no protocolo. Os papéis primários são o distribuidor de chave e o membro do grupo. O controlador inicia a criação das chaves, cria as mensagens de distribuição de chaves e coleta as confirmações de recebimento dos receptores. Os membros aguardam por uma mensagem de distribuição, decifram e validam seu conteúdo, e confirmam o recebimento da nova chave.

Os principais papéis podem ser brevemente descritos como:

- *Controlador de Grupo* (*Group Controller - GC*) - entidade com autoridade para executar ações críticas (*i.e.* criar, recodificar ou distribuir a chave), visando manter a sincronização do grupo. Se parte do grupo perder, ou de maneira inapropriada alterar a chave, este não será capaz de enviar ou receber dados de outro *host* operando com a chave criptográfica correta.
- *Membro do Grupo* - qualquer *host* que não está exercendo a função de controlador. O membro do grupo possui as seguintes funcionalidades: auxiliar o controlador na geração da chave; validar a autorização do controlador durante a execução das ações; aceitar as chaves do controlador; solicitar uma chave para o controlador; manter uma lista de *CRL* localmente; e, gerenciar a chave local.

O *GKMP* consiste de várias funções necessárias para criar, distribuir, recodificar e gerenciar grupos de chaves simétricas:

- *Criação do grupo* - a inicialização é uma função de três etapas que envolve os comandos para: criação do grupo; a criação das chaves de grupo; e a distribuição destas chaves para os demais membros do grupo. As mensagens entre o *GC* e um dos membros vai gerar duas chaves que permitem realizar ações futuras no grupo: chave para cifrar o tráfego do grupo (*Group Traffic Encryption Key - GTEK*) e a chave para cifrar a chave do grupo (*Group Key Encryption Key - GKEK*).

- *Recodificação do grupo* - função de duas etapas que envolve a troca de mensagens entre o *GC* e os demais membros do grupo. As mensagens trocadas entre o *GC* e um membro primário tem a finalidade de gerar e distribuir novas chaves para os membros do grupo (*e.g. GTEK*, *GKEK*).
- *Membro inicia o processo de adesão ao grupo* - esta funcionalidade é utilizada quando o criador do grupo precisa verificar se o futuro integrante está de acordo com as regras de ingresso previamente estabelecidas (*e.g.* limite de membros, domínio de origem). Antes de uma nova entidade filiar-se ao grupo, esta deve solicitar a chave para o *GC*, identificar-se de maneira inequívoca e encaminhar suas permissões de acesso. Caso a veracidade das informações seja comprovada, o novo membro pode receber a chave de grupo.
- *Exclusão de um membro* - existem dois cenários possíveis para este caso: *cooperativa* - remover um membro confiável devido a uma ação administrativa (*e.g.* reduzir o tamanho do grupo); e *hostil* - resultado da perda do estado seguro do grupo (*e.g.* comprometimento do serviço).

3.6 *Especificações de Segurança*

Com a adoção do modelo de serviços em nuvem, os consumidores tem a possibilidade de automatizar os processos de configuração de usuários e regras de acesso. O ambiente de nuvem permite que as organizações se afastem dos conectores personalizados e passem a utilizar especificações amplamente aceitas pelo mercado.

3.6.1 Transporte de Atributos, Contextos de Autenticação e Autorização

O perfil do usuário pode ser descrito como um conjunto de atributos utilizados pela nuvem para customizar ou restringir o acesso aos serviços. O controle de acesso gerencia o uso de recursos através da aplicação de políticas adequadas. O procedimento de controle depende de informações precisas sobre o perfil do usuário para tomar as decisões adequadas. Quando o usuário age em nome de um domínio consumidor, este se torna a fonte oficial de atributos e das respectivas políticas de controle que serão avaliadas previamente ao acesso dos serviços.

A especificação *SAML* (*Security Assertion Markup Language*) (OASIS, 2005a) define um esquema que possibilita a troca de dados (*e.g.* atributos do usuário, de autenticação ou autorização) de maneira independente de plataforma. As informações transportadas por mensagens *SAML* podem ser utilizadas para realizar o intercâmbio de atributos entre provedores e consumidores, ou para autorizar os usuários a acessar serviços disponibilizados pelo ambiente.

O emissor da assertiva (*e.g. Identity Provider - IdP*) se torna o serviço responsável pela veracidade das informações (*e.g.* autenticação, autorização, atributos). O provedor de serviço que recebe a declaração - através de relações pré-estabelecidas - confia no *IdP* que forneceu a assertiva. As declarações *SAML* podem ser embutidas no corpo de mensagens *SOAP* (*Simple Object Access Protocol*) (W3C, 2007a), sendo transportadas utilizando um protocolo de comunicação padrão (*e.g. Hypertext Transport Protocol - HTTP*).

Três tipos de declarações podem ser inseridas dentro de uma assertiva por um provedor de identidade:

i. *expressão de autenticação* - define quem emitiu a assertiva afirmando que a entidade referenciada foi autenticada;

ii. *expressão de atributo* - a entidade é associada com seus respectivos atributos;

iii. *expressão de autorização* - define o que a entidade declarada está autorizada a fazer.

3.6.2 Configuração de Objetos Distribuídos

Usuários agindo em nome do domínio consumidor necessitam que a política de controle de acesso seja transmitida da organização consumidora para o provedor de serviço ligado a nuvem. As regras de controle podem ser transmitidas periodicamente (*i.e.* modo *batch*) utilizando a especificação *SPML* (*Service Provisioning Markup Language*) (OASIS, 2006b).

A linguagem *SPML* pode ser empregada para administrar objetos distribuídos instanciados em serviços que realizam tarefas. O processo de gerenciamento pode ser feito através das operações de adicionar, remover, atualizar ou pesquisar. O elemento

alvo, onde a configuração é necessária, possui o serviço responsável por receber, interpretar e executar o provisionamento nos respectivos objetos gerenciados.

A automatização dos processos de configuração facilita o gerenciamento de serviços em ambientes distribuídos. Este procedimento pode ser utilizado para efetuar a preparação de um determinado sistema antes deste executar suas atividades (*e.g.* configurar contas de usuários, fornecer privilégios de acesso para serviços).

As principais entidades descritas pela especificação *SPML* são: a entidade solicitante (*Requesting Authority – RA*), o provedor de serviço (*Provisioning Service Provider – PSP*), os objetos do serviço (*Provisioning Service Objects – PSO*), e o serviço alvo (*Provisioning Service Target – PST*).

3.6.3 Criação, Avaliação e Transporte de Políticas de Controle de Acesso

A utilização dos serviços hospedados na nuvem, por usuários subordinados ao domínio consumidor, introduz a possibilidade das políticas de controle de acesso serem escritas internamente ao domínio contratante, sendo posteriormente transmitidas e utilizadas no provedor de serviço. A utilização de padrões da indústria permite uma avaliação inequívoca do significado de uma regra - desde que emissor e receptor concordem com a semântica utilizada na especificação das políticas. A definição de uma política de controle de acesso internamente à organização facilita a administração do ambiente, porém, é necessário transmitir essas políticas para o respectivo provedor de serviços ligado a nuvem.

A especificação *XACML* (*eXtensible Access Control Markup Language*) (OASIS, 2005b) propõe um esquema que permite criar políticas de controle de acesso utilizando o padrão de escrita *XML* (*eXtensible Markup Language*). A linguagem permite a definição de políticas globais que se aplicam a todos os recursos de um sistema (*e.g.* sujeitos, objetos, ações). Adicionalmente, a *XACML* define o protocolo pedido/resposta utilizado nas trocas de mensagens entre as entidades que decidem e executam as políticas no ambiente.

A especificação faz uso dos serviços *PEP* (*Policy Enforcement Point*), *PDP* (*Policy Decision Point*), Manipulador de Contexto, *PAP* (*Policy Administration Point*) e *PIP* (*Policy Information Point*). O *PEP* intercepta o pedido enviado por usuários, repassa a solicitação ao Manipulador de Contexto, e aplica a decisão de permitir ou negar o acesso

do usuário. O *PDP* recebe as mensagens do Manipulador de Contexto e, com base nas informações do usuário e nas políticas de controle vinculadas a determinado recurso, toma uma decisão de acesso. Para utilizar as políticas o *PDP* consulta o *PAP*, sendo este a entidade responsável por armazenar as regras do sistema. Para auxiliar na tomada de decisão do *PDP*, o Manipulador de Contexto pode interagir com *PIP* e enviar atributos adicionais para complementar o processo de avaliação (*e.g.* informações relacionados ao ambiente ou ao próprio usuário).

Os três principais elementos de uma política *XACML* podem ser brevemente descritos como:

i. *regra* - contém uma expressão booleana que é utilizada para formar uma unidade básica de gerenciamento;

ii. *política* - contém um conjunto de elementos regra e um procedimento específico para combinar o resultado da avaliação deste conjunto. A política forma a base de uma decisão de autorização;

iii. *conjunto de políticas* - pode conter várias políticas singulares ou outros agrupamentos de políticas. O conjunto possui instruções específicas para combinar os resultados de sua avaliação.

A combinação de vários elementos regra e política pode corresponder a diferentes decisões de controle de acesso. A utilização de instruções específicas permite combinar múltiplas avaliações em uma simples decisão (*e.g.* permitir ou negar). Os esquemas de junção são classificados em: algoritmos de combinação de políticas - definem um procedimento para se chegar a uma decisão de autorização considerando os resultados individuais da avaliação de um conjunto de políticas (*e.g. permit-overrides, deny-overrides, first-applicable, only-one-applicable-policy*); algoritmos de combinação de regras - definem um procedimento para se chegar a uma decisão de autorização analisando os resultados individuais da avaliação de um conjunto de regras (*e.g. permit-overrides, deny-overrides, first-applicable*).

Pedidos relacionados com o gerenciamento do controle de acesso podem ser encapsulados em diferentes linguagens (*e.g.* assertivas *SAML*), sendo transmitidos entre diferentes domínios administrativos. A solicitação de avaliação de política enviada pelo *PEP* para o *PDP*, ou a transmissão de regras e políticas de controle pode acontecer de

acordo com a necessidade (*i.e. just-in-time* - concomitante com a solicitação de acesso do usuário). Se o provedor de serviço for capaz de interpretar contextos de controle encapsulados em assertivas *SAML*, então a relação inter-domínios entre o Monitor de Referência e o Guardião pode ocorrer utilizando o *SAML profile of XACML* (OASIS, 2005c). Esta especificação permite o empacotamento dos contextos *XACML* (*e.g.* políticas de controle, solicitação de avaliação de acesso) em assertivas *SAML*.

3.6.4 Segurança das Mensagens

A troca de informações em ambientes como a *Internet* necessita de meios confiáveis de proteção de dados. Com a migração e implantação de sistemas na nuvem, a quantidade de informações que trafega na rede mundial de computadores tende a aumentar consideravelmente. Com o passar do tempo, a confiança na nuvem será cada vez maior, sendo que a sensibilidade dos dados movidos para esse ambiente também pode aumentar.

A especificação *Web Service Security (WS-Security): SOAP Message Security* (OASIS, 2004) integra e também forma a base para outros padrões de segurança disponibilizados para Serviços *Web*. O esquema proposto permite que os serviços troquem informações de maneira neutra a linguagem e plataforma, aplicando um alto nível de segurança aos dados que trafegam pela rede.

A abordagem utilizada acrescenta extensões a mensagem *SOAP* (W3C, 2007a), provendo suporte para assinatura, cifragem e a representação de credenciais de segurança - padrões como *XML Encryption* e *XML Signature* provêem suporte para um esquema de segurança fim a fim (*end-to-end*).

3.7 *Conclusão*

Em ambientes de nuvem, os procedimentos a serem aplicados no gerenciamento do controle de acesso e do perfil do usuário são mais desafiadores. As informações utilizadas nestas funções podem vir de diferentes fontes e processos, com convenções de nomes e tecnologias variadas. Adicionalmente, estas informações precisam ser transmitidas de maneira segura entre os diferentes domínios sendo que, geralmente, os dados serão conduzidos através de um ambiente hostil, a *Internet*.

O capítulo seguinte aborda alguns trabalhos diretamente relacionados com este projeto. As pesquisas expõem propostas em nível de modelo e também em caráter experimental. As idéias apresentadas por cada um dos artigos serviram de base para nortear os nossos objetivos.

Capítulo 4

Trabalhos Relacionados

4.1 *Introdução*

Embora todo o ecossistema de nuvens computacionais tenha evoluído significativamente desde sua proposição e efetiva implantação, novos desafios continuam a surgir a cada dia. A solução dos problemas ou a melhoria das propostas existentes necessita de abordagens inovadoras, sendo imperativo que estudos constantes sejam direcionados para esta área. As seções subsequentes são exemplos destes esforços.

4.2 *An Integrated Service Model Approach for Enabling SOA*

O artigo aborda um modelo (*ISM - Integrated Service Model*) (Zhang, 2009) que decompõe o serviço em: lógica do negócio, interface e implementação. O *ISM* reforça o esquema proposto pela arquitetura orientada a serviço (*Service Oriented Architecture - SOA*; *i.e.* composto por provedor, consumidor e repositório) aumentando a flexibilidade, extensibilidade e adaptabilidade dos serviços.

De acordo com o modelo *SOA*, aplicações podem ser encapsuladas com interfaces e publicadas como Serviços *Web* (*WS*). Porém, a *SOA* não identifica assuntos relacionados a manipulação do serviço (*e.g.* decomposição, agregação, transformação). Adicionalmente, *SOA* não provê suporte para configurar e ou re-configurar os serviços e seus componentes. Uma solução baseada no modelo atual de *SOA* e Serviços *Web* poderia sofrer de baixa re-usabilidade.

O *ISM* decompõe serviço em três camadas (Figura 4.1):

i. *camada de processos do negócio* - todo o processo é dividido em pequenas tarefas

que são mapeadas na camada de serviço (*i.e. SOA*). Esta camada poderia localizar *dez* processos existentes e agregar estes em *três* grandes grupos, enquanto monitora e gerencia a colaboração entre os mesmos;

ii. *camada de serviço* - permite o registro, descoberta, conexão, decomposição, agregação e gerenciamento do ciclo de vida do serviço. A camada trata do mapeamento de processos para serviços reais, sendo responsável por: encontrar o provedor de serviço adequado; localizar onde o serviço está hospedado; controlar o acesso; e fazer a ligação (*binding*) com a interface do serviço;

iii. *camada de componente de serviço* - pode invocar serviços na camada de serviços ou processos na camada de processos de negócio. Esta camada visa implementar e ou adaptar a assinatura do método que será utilizado na camada de serviço.

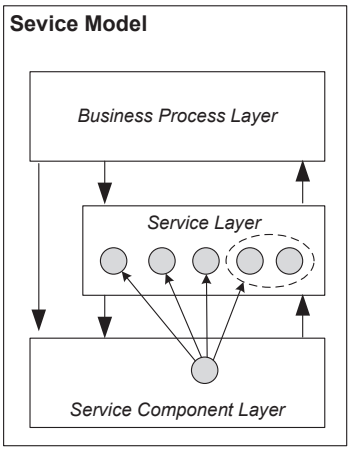

Figura 4.1: Modelo de Serviço Um Para Muitos. Adaptado de Zhang, 2009.

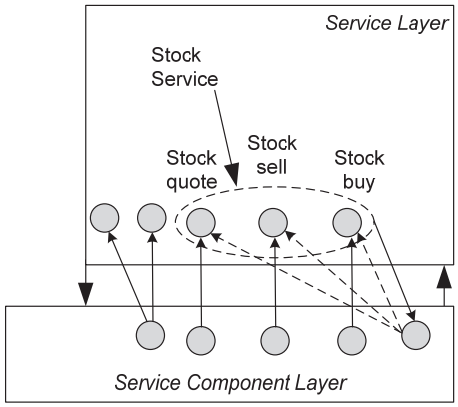

Figura 4.2: Modelo de Serviços Compostos. Adaptado de Zhang, 2009.

Um componente pode ser utilizado por vários serviços, sendo assim, o número de serviços pode exceder o número de elementos na camada de componentes (Figura 4.1). A Figura 4.2 ilustra uma nova interface de serviço (*Stock Service*) formada por três elementos individuais (*i.e. quote, sell,* e *buy*).

O modelo proposto pelo artigo atua principalmente na camada *SaaS*, promovendo a agregação de diferentes serviços em uma única interface composta. Cada serviço sendo fornecido é constituído de pequenas partes, provenientes de um conjunto de componentes. Esta abordagem assemelha-se com a presente proposta no sentido de oferecer uma

interface de acesso homogênea, formada por provedores de serviço distintos. A necessidade computacional do contratante ou dos respectivos usuários associados ao domínio consumidor (*i.e.* a cota de uso estabelecida no contrato) é quem gera a demanda pela alocação dos recursos.

4.3 *Automated Control in Cloud Computing*

O artigo aborda a construção de um controlador externo aos serviços hospedados na nuvem – um *add-on* para o consumidor (Lim, 2009). Este componente deve funcionar de acordo com a *API* fornecida pelo provedor. Algumas *APIs* disponibilizadas pela nuvem oferecem opções para a implantação de controladores baseados em *feedback* (*feedback controllers*), visando automatizar a configuração e o fornecimento de recursos.

Normalmente o consumidor (*guest*; Figura 4.3) controla os sistemas no servidor virtual (*Virtual Machine – VM*) instanciado no provedor. As nuvens alocam dinamicamente fatias (*slices*) de seus recursos (*e.g.* processamento, armazenamento), balanceando a qualidade do serviço em relação ao custo gerado para o consumidor. A utilização de políticas de controle auxiliadas por dados fornecidos pelo provedor (*off-the-shelf feedback*) podem auxiliar na tarefa de monitoramento das aplicações hospedadas na nuvem.

Baseado nestas premissas, os consumidores deveriam estar autorizados a operar seus próprios controladores dinâmicos fora da plataforma da nuvem, ou como uma extensão da mesma (Figura 4.3). Estes gerenciadores externos são chamados de *slice controllers*. Para ser possível esta abordagem, o provedor de recursos deve expor sensores a atuadores que aceitem as políticas de controle definidas pelo consumidor.

As plataformas de nuvem disponibilizam interfaces de serviço que visam separar os interesses das entidades envolvidas: consumidores são isolados de detalhes físicos do recurso; provedores são isolados das nuances referentes aos sistemas de seus arrendatários. Considerando isso, a estrutura do controlador deveria refletir essa separação de funcionalidades, ou seja: o provedor de nuvem deve executar seu próprio sistema de controle (*cloud controller*) para gerenciar as solicitações de recurso; o controle da aplicação deve ser implementado e administrado pelo domínio consumidor (*slice controller*).

A abordagem proposta permite ao cliente adequar suas políticas, customizando os

controladores de acordo com as necessidades da aplicação. Esta divisão em camadas exige que a nuvem forneça uma *API* rica em funcionalidades, a qual suporte a interação entre os diferentes controladores (*e.g.* o *cloud controller* interage com o *slice controller* para obter informações como a qualidade de serviço solicitada pelo cliente ou o posicionamento da *VM*).

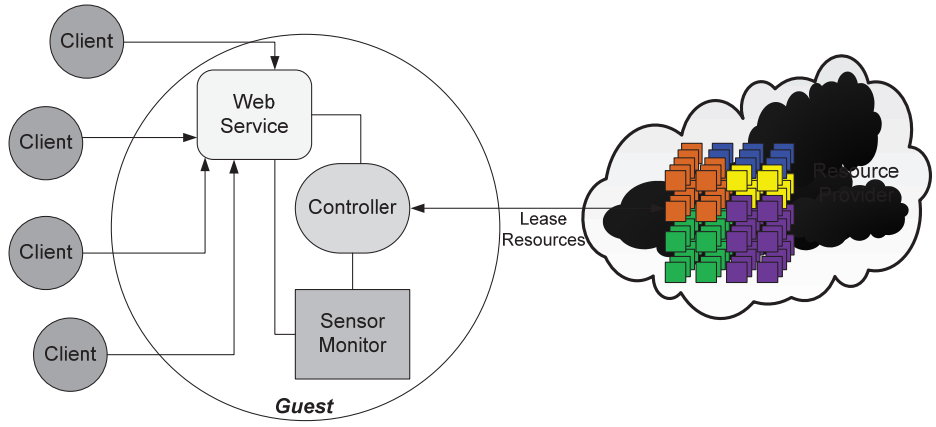

Figura 4.3: Sistema de Monitoração e Controle. Adaptado de Lim, 2009.

Nas plataformas de nuvem atuais, os clientes não possuem acesso aos atuadores em nível de *hypervisor* (*e.g.* permite controlar a alocação de memória e *CPU* para as *VMs*). Neste contexto existe uma lacuna entre os sensores/atuadores fornecidos pelo provedor e estes desejados pelo *slice controller* do consumidor. A técnica de *proportional thresholding* pode ser aplicada em casos onde *slice controllers* precisam utilizar gerenciadores com granularidade grossa.

A utilização de um limiar proporcional (*proportional thresholding*) é uma técnica de controle para sistemas com atuadores de granularidade grossa (*coarse-grained*). O controlador utiliza como referência uma faixa de valores dinâmicos ao invés de um valor estático. A política criada pelo consumidor pode utilizar este limiar para controlar a alocação de recursos. Seria como usar uma escala de crescimento horizontal (*i.e.* mais *VMs*) para lidar com picos de consumo inesperados, visando assegurar a alta utilização dos recursos (*e.g. CPU*).

A proposta de criação de um sistema de controle automatizado, ressalta a necessidade de se consolidar os dados recuperados do ambiente do provedor e aplicar estes em políticas de controle. O modelo traz à tona a necessidade que o locatário tem de monitorar a cota contratada, regrando o uso e alocando os recursos de acordo com as necessidades do ambiente consumidor. Porém, adicionalmente ao controle de uso

aplicado aos recursos, é necessário regrar qual usuário subordinado ao domínio consumidor está utilizando os serviços na nuvem. Monitores e atuadores de granularidade fina, auxiliados por um domínio intermediário responsável por recuperar e consolidar os dados do ambiente, podem proporcionar uma solução mais completa para o gerenciamento do ambiente.

4.4 *Determining Service Trustworthiness in Inter Cloud Computing Environments*

As aplicações atuais podem exigir recursos de várias plataformas distribuídas, necessitando de uma federação de nuvens autônomas. A federação precisa confiar nos provedores que formam a nuvem, tendo em vista que os recursos podem ser provenientes de entidades públicas ou privadas. O *framework* apresentado permite que as entidades interessadas determinem a confiabilidade dos provedores integrados a federação. A agregação de nuvens tem como objetivo o intercâmbio de recursos (*e.g.* armazenamento, processamento) de modo unificado e uniformizado (Abawajy, 2009).

A natureza dinâmica das nuvens dificulta o compartilhamento de recursos em ambientes de interação inter-nuvens (*Inter-Cloud Computing - ICC*). Para concretizar esta tarefa é necessário: padronizar os protocolos de rede; criar interfaces de comunicação que forneçam serviços confiáveis; implantar políticas e mecanismos para equiparar as diferentes nuvens; gerenciar as solicitações de acesso provenientes de outros domínios; selecionar nuvens confiáveis para terceirizar serviços.

Um sistema de gerenciamento baseado em reputação possibilita determinar a fiabilidade do serviço em ambientes inter-nuvens (*ICC*). Este sistema permite ao solicitante obter o nível de confiabilidade do serviço (*e.g.* utilizar um serviço de nuvem que satisfaça determinados requisitos de *QoS*). O *ICC* visa facilitar o compartilhamento de funcionalidades para entidades que fazem parte da rede de colaboração (*e.g.* serviços para aplicações, recursos de armazenamento).

Consumidores, usuários e provedores interagem entre si sem ter garantias sobre o comportamento dos serviços para quem estes confiam suas informações. Com a crescente expansão das interações entre nuvens, existe a necessidade de avaliar e manter a reputação das entidades. Associado a isto é necessário avaliar que tipo de informação

coletar, como obter dados precisos, e de que maneira utilizar os níveis de confiança.

A nuvem possui uma arquitetura dividida em camadas, sendo que os serviços são fornecidos de acordo com o tipo da nuvem e as respectivas políticas utilizadas pelo provedor. O compartilhamento dos recursos é feito por acordos entre os participantes (*e.g.* uma nuvem adquire cotas da outra de acordo com uma política - *SLA*).

Os usuários submetem seus pedidos de alocação de recursos para um gerente local (*Cloud Resource Manager* - *CRM*; Figura 4.4). Esta entidade é responsável pelo provisionamento e alocação do recurso na nuvem. Se o *CRM* não pode atender a solicitação localmente (*e.g.* devido a necessidade de mais recursos), este encaminha uma solicitação para o gerente de recursos inter-nuvens (*Inter-Cloud Resource Manager* - *IRM*).

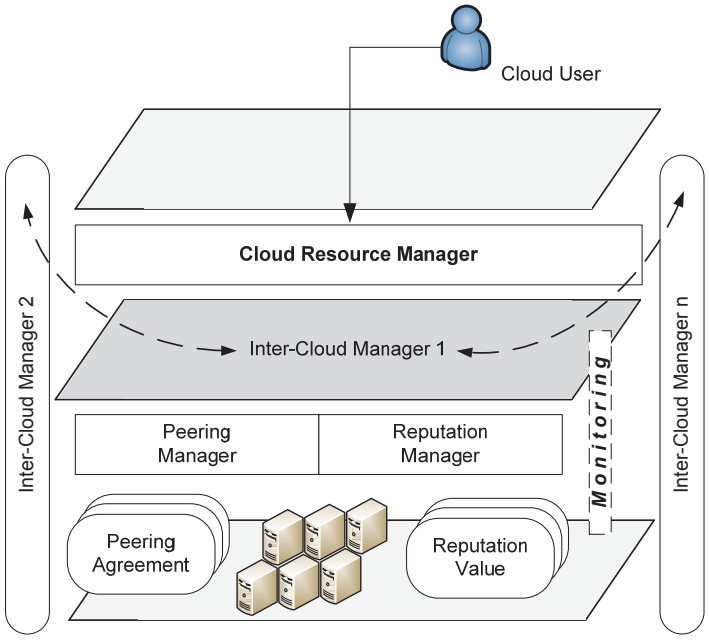

Figura 4.4: Infraestrutura de computação inter-nuvens. Adaptado de Abawajy, 2009.

O *IRM* é responsável por mediar a troca de recursos entre as nuvens. Este gerenciador estabelece os termos do acordo entre os provedores de nuvens (*e.g.* tipo do recurso que as entidades listadas no contrato podem adquirir umas das outras). O gerente de recursos inter-nuvens fornece funcionalidades para: selecionar a nuvem adequada para o provimento de um determinado recurso; gerenciar as solicitações de recursos e serviços vindas de outros *IRMs*; monitorar a execução das aplicações entre as nuvens; interagir com o sistema de contabilização de consumo de cada nuvem.

O gerente de reputação (*Reputation Manager - RM*; Figura 4.4) é responsável por manter uma medida de confiança referente a um conjunto de nuvens (*peering clouds*). A classificação é baseada na experiência obtida durante a interação (*i.e. first-hand information*) e pelo compartilhamento de informações com outras entidades (*i.e. second-hand information*).

A confiança é representada pelo relacionamento entre duas entidades (*peering entities*) em um contexto particular. O gerente de reputação (*RM*) usa os seguintes dados para avaliar uma entidade: *a) experiência pessoal* - informações primárias (*first-hand*) obtidas após efetuar transações com determinado "*nó*"; *b) avaliação da reputação* - o nível de confiança que o "*nó A*" formou sobre o comportamento do "*nó B*"; *c) avaliação de honestidade* - o julgamento do "*nó A*" sobre a credibilidade do "*nó B*" enquanto provedor de informações secundárias (*second-hand*).

Uma vez que os dados sobre a classificação tenham sido recolhidos, o gerente de reputação (*RM*) pode atualizar seus registros. A cada período de tempo, cada entidade publica sua avaliação para um subconjunto dos "*nós*" com a qual possui algum tipo de contrato. Ao fechar novos contratos, os provedores podem obter informações de confiabilidade de um "*nó*" (*i.e. peer*) previamente conhecido.

A abordagem descrita neste documento presume a utilização de uma federação. Este ambiente consolida os atributos oriundos dos provedores e fornece um domínio confiável. Neste domínio os consumidores podem obter informações precisas sobre as entidades que disponibilizam serviços na nuvem. Adicionalmente, a proposta contempla o controle de acesso regrado por políticas de granularidade fina, sendo este um item essencial para gerenciar o consumo dos usuários subordinados ao consumidor.

4.5 *A RESTful Approach to the Management of Cloud Infrastructure*

O artigo propõe um Sistema de Gerenciamento para Nuvens (*Cloud Management System – CMS*) utilizando o modelo *REST* (*REpresentational State Transfer*) (Han, 2009). Os elementos gerenciados pelo sistema (*e.g.* rede, armazenamento) podem ser modelados como os recursos do *REST*, sendo que as operações de gerenciamento podem ser efetuadas através dos métodos: *POST – Create*: criar um novo recurso; *GET – Retrieve*: recuperar o estado atual do recurso; *PUT – Update*: atualizar o estado do recurso; *DELETE – Delete*: apagar um recurso.

Os princípios fundamentais do *REST* são: cada recurso (*e.g.* um registro, um item) é referenciado usando uma *URI*; os recursos são manipulados utilizando os métodos disponibilizados pelo protocolo *HTTP* (*i.e. GET*, *PUT*, *DELETE*, *POST*); cada interação com o recurso é *stateless*.

Os componentes básicos do sistema de gerenciamento são (Figura 4.5):

- *Frontend Interface* – interface de gerenciamento para entidades externas (*e.g.* usuários, sistemas). As solicitações recebidas são encaminhadas para o *Management Module*, sendo posteriormente respondidas;
- *Asynchronous Event Handler* – recebe as notificações dos elementos gerenciados, armazena estas em um banco de dados, enviando as mais significativas para o *Management Module*;
- *Management Client* – coleta informações dos elementos gerenciados e as armazena em um banco de dados. Este módulo envia mensagens de controle para os elementos gerenciados;
- *Management Module* – administra os parâmetros de configuração dos elementos gerenciados; manipula a topologia lógica dos elementos; obtêm informações de monitoramento do banco de dados ou do *Management Client*; registra as informações necessárias; gerencia as notificações do *Asynchronous Event Handler*.

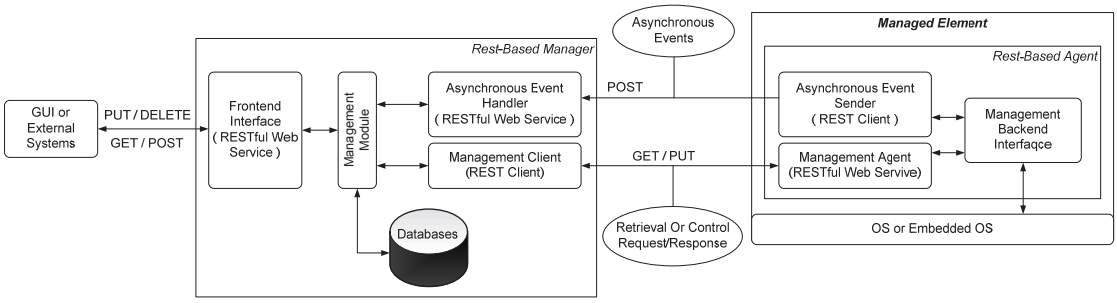

Figura 4.5: Sistema de Gerenciamento para Nuvens. Adaptado de Han, 2009.

A *Interface* de Administração (*Management Backend Interface*; Figura 4.5) é utilizada para fornecer acesso ao Agente de Gerenciamento (*Management Agent*) e ao Emissor de Eventos (*Asynchronous Event Sender*). Se um acontecimento inesperado (*e.g.* sobrecarga da *CPU*) for detectado, a *Interface* de Administração envia este para o

Emissor de Eventos. Essa *Interface* também retorna as respostas adequadas para o Agente de Gerenciamento de acordo com as solicitações do mesmo (*e.g.* aplicar configurações em uma lista de controle de acesso). O Emissor de Eventos recebe informações da *Interface* de Administração e envia notificações para o Manipulador de Eventos. O Agente de Gerenciamento recebe comandos de controle ou pedidos de informação referentes ao cliente e envia as respostas para a Administração do Cliente (*Management Client*) através de uma interface *REST*.

Todas as informações referentes a elementos gerenciados (*e.g.* memória, grupo de usuários) em um serviço *REST* são descritas através de *URIs*. Alguns *MIMs* (*Management Information Models*) são organizados como "*árvores*", sendo que os nodos internos representam subdivisões na árvore (*e.g.* por função). Cada valor de variável é armazenado na "*folha*" correspondente da "*árvore*", criando assim um caminho único desde a "*raiz*" da "*árvore*". Os filhos de um nodo são numerados sequencialmente para que cada nodo na "*árvore*" tenha um nome único. De acordo com o *MIM* do agente, cada nodo na "*árvore*" representa as informações de gerenciamento.

As informações são divididas em três partes: informações comuns - a configuração de dispositivos; informações de gerenciamento padrão - definida por especificações; informações de gerenciamento privada - específicas para um consumidor. Se o cliente (*Management Client*) deseja recuperar a *(a) lista de administradores* e a *(b) descrição do sistema* do "*cloud01.snu.ac.kr*" então este deve enviar pedidos *HTTP* transportando a operação *GET*, respectivamente, para as seguintes *URIs*:

(a) "*http://cloud01.snu.ac.kr/deviceInfo/ci/AdminList*";

(b) "*http://cloud01.snu.ac.kr/deviceInfo/smi/mib/iso/org/dod/internet/mgmt/mib/system/sysDesc*".

O gerenciamento do ambiente utilizando *REST* fornece poucos modos de interação, além dos métodos suportados pelo *HTTP*. Adicionalmente, cada consumidor necessita gerenciar seus próprios serviços, independentemente de quantos tipos de provedores forem utilizados. Para serviços simples, que envolvem poucas entidades, ou que não necessitam manter o estado durante as transações, o *REST* é uma boa alternativa. Porém, se a variedade de serviços a serem administrados for grande, o conjunto de dados gerados também será. Supondo que cada entidade gerenciada utilize uma convenção de nomes particular, o ambiente pode originar uma enorme quantidade de atributos heterogêneos.

4.6 *Policy-based Event-driven Services-oriented Architecture for Cloud Services Operation & Management*

O artigo propõe um modelo de aplicação de políticas que permite o gerenciamento de serviços entre nuvens. A abordagem *PESA* (*Policy-based Event-driven Service-oriented Architecture*) visa controlar as características do serviço (*e.g.* disponibilidade, desempenho, segurança) (Goyal, 2009). O modelo usa o conceito de particionamento lógico e virtual de sistemas (*i.e.* serviços que abrangem os limites geográficos e ou tecnológicos da empresa).

A implantação de um negócio qualquer pode utilizar serviços de várias nuvens. O serviço *S1*, através do *S5* (Figura 4.6), pode implementar um fluxo de trabalho que combine entidades públicas e ou privadas. Para o consumidor utilizar *SLAs* em seu fluxo de negócio, o provedor de nuvem deve suportar estes contratos e executar o gerenciamento adequado em nível de serviço.

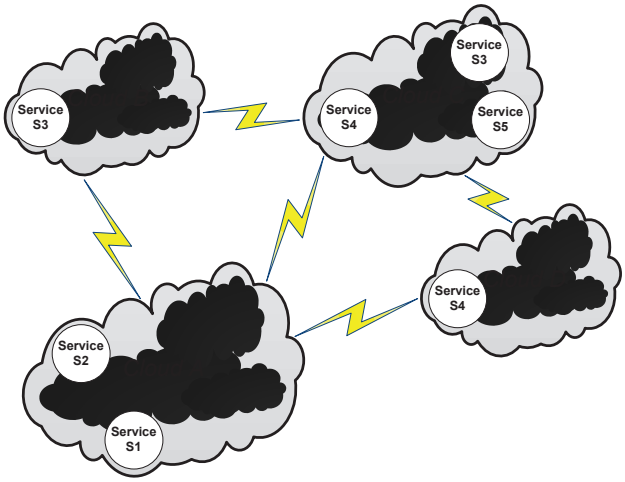

Figura 4.6: Fluxo de Trabalho Entre Nuvens. Adaptado de Goyal, 2009.

A política na arquitetura *PESA* expressa um conjunto de regras para controlar o comportamento e o uso do serviço em contextos específicos. A integração de políticas com sistemas de medidas necessita que o provedor permita que os consumidores apliquem regras para gerenciar a utilização dos serviços subjacentes (*e.g.* otimizar desempenho, prover segurança, gerenciar recursos).

O volume de dados gerado por vários serviços pode ser muito alto para um administrador centralizado coletar, armazenar, analisar e processar. Sendo assim, o

particionamento do ambiente em domínios cria escopos administrativos reduzidos (*i.e.* limitar as interações entre as partes; classificar os tipos de recursos). Cada partição oferece um isolamento fim-a-fim das funcionalidades da nuvem através da utilização de ambientes virtuais e serviços de gerenciamento (Figura 4.7). Esta divisão permite a implementação de níveis com aspectos similares (*e.g.* qualidade de serviço, controle de acesso).

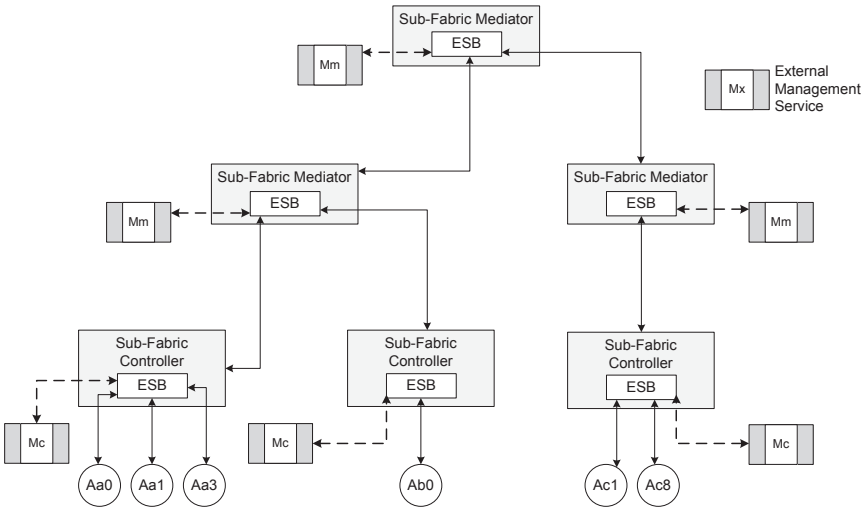

Figura 4.7: Serviços de Gerenciamento. Adaptado de Goyal, 2009.

Cada nível da partição pode possuir mais de um serviço de gerenciamento (*SG*). Os *SGs* de nível mais alto possuem um escopo e uma visibilidade maior, permitindo análises mais complexas, correlações e demais funcionalidades. Exceto para o *SG root*, cada serviço de gerenciamento está subordinado a um superior. Cada *SG* pode ser dedicado para um aspecto de gerenciamento particular (*e.g.* uma determinada política).

A proposta contempla principalmente o gerenciamento dos recursos consumidos na nuvem. A avaliação de acesso dos usuários ou o controle sobre quem está fazendo uso do serviço não são abordados pela arquitetura *PESA*. Porém, o artigo traz algumas idéias interessantes sobre a integração de monitores de recursos com políticas de controle. Este esquema permite automatizar o gerenciamento das entidades que interagem com o ambiente. A segmentação utilizada na arquitetura visa somente o ambiente local, internamente ao domínio de um provedor. O esquema de gerenciamento sendo proposto nesta pesquisa permite consolidar todos os atributos do ambiente em um mesmo domínio administrativo. Esta abordagem visa facilitar o acesso do consumidor as informações de uso referentes a seus usuários.

4.7 *A Runtime Model Based Monitoring Approach for Cloud*

A proposta descreve um modelo (*Runtime Model for Cloud Monitoring - RMCM*) que fornece uma representação intuitiva de uma nuvem em execução (Shao, 2012). O modelo foi utilizado na implementação de um *framework* de monitoramento flexível. Este pode obter um balanceamento entre a carga gerada em tempo de execução e a capacidade de monitoramento.

O *RMCM* abstrai as informações de tempo de execução coletadas durante o funcionamento da nuvem e as apresenta de forma compreensível e operacional. O modelo está vinculado de tal maneira com a nuvem que este reflete constantemente o sistema, seu estado atual e comportamento. Qualquer mudança no sistema em execução será refletida imediatamente no modelo. Quando problemas aparecem, são executadas ações no modelo ao invés de intervenções diretamente na nuvem.

Geralmente existem três tipos de papéis em uma nuvem (Figura 4.8): *1)* operadores de nuvem - possuem os recursos computacionais. Este papel precisa inspecionar o funcionamento da nuvem e seu ambiente para obter os indicadores sobre quando expandir/contrair o fornecimento do serviço; *2)* desenvolvedor de serviço - tentam descobrir os hábitos de consumo dos usuários finais. O monitoramento executado por este papel produz informações utilizadas na evolução das aplicações e na melhoria da experiência do usuário; *3)* usuários finais - utilizam o serviço fornecido pela nuvem para dar suporte aos seus negócios.

A nuvem é apresentada com modelos de tempo de execução consistentes com as necessidades de cada tipo de papel (Figura 4.8). Os operadores da nuvem têm uma visão integral do ambiente em funcionamento, concentrando-se na utilização e alocação de recursos. Os desenvolvedores de serviço concentram-se no estado de sua aplicação, verificando se a nuvem forneceu recursos suficientes conforme o contratado. Os usuários finais concentram-se em uma comparação entre vários serviços. Estas exibições fornecem um modelo de tempo de execução consistente para o monitoramento de um sistema de nuvem.

As entidades modeladas são dividas em quatro categorias (Figura 4.8): *1)* infraestrutura de suporte (*e.g.* sistemas operacionais das máquinas físicas e virtuais) - concentra-se no estado atual da utilização dos recursos (*e.g.* utilização da *CPU*); *2) middleware* - fornece um ambiente de execução para os componentes que auxiliam no fornecimento de serviços para os usuários finais (*e.g.* gerenciamento da execução); *3)*

aplicações implementadas pelos desenvolvedores - sistemas em nuvem são monitorados conforme duas perspectivas: *a)* de acordo com seu projeto e implementação, visando validar se a aplicação satisfaz seus requisitos; *b)* conforme as regras pré-definidas pelo operador da nuvem, visando prevenir que algum código malicioso e ou defeituoso degrade o desempenho da nuvem; *4)* monitoramento das interações entre papéis e entre nuvens.

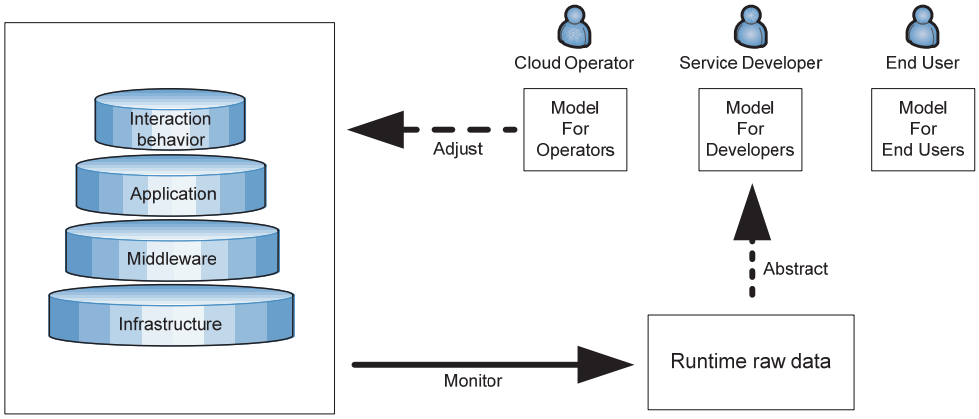

Figura 4.8: Modelo de Processos no *RMCM*. Adaptado de Shao, 2012.

Em uma nuvem, várias entidades precisam ser monitoradas simultaneamente, sendo assim, é necessário coordenar a utilização dos recursos para obter um bom balanceamento. Para isso, a estrutura de monitoramento deve acompanhar cada uma das entidades, porém, tomar decisões centralizadas. A estrutura de monitoramento foi implementada no estilo agente-servidor (*server-agent*). Cada agente é implantado em uma máquina virtual, sendo encarregado de coletar as informações em tempo de execução.

Diferentes tipos de perfis de monitoração são implantados em um mesmo agente, permitindo que este colete informações das entidades de cada nível. Os dados coletados na forma bruta serão comparados com algumas regras pré-definidas, determinando assim se algum alarme deveria ser enviado para um centro de monitoração. Se o estado de funcionamento representado pelo modelo *RMCM* for aceitável, este será guardado em um armazenamento próprio. Sempre que necessário, os administradores podem verificar as informações armazenadas no centro de monitoramento e modificar as configurações dos agentes.

A utilização de agentes permite que os recursos e serviços fornecidos por provedores de nuvem sejam monitorados pelas entidades que gerenciam o ambiente. Em nossa proposta, os dados coletados são armazenados e analisados por um mecanismo de

controle federado (*i.e. Broker*). Estes atributos permitem identificar violações no consumo individual de cada usuário que está acessando o ambiente da nuvem computacional.

4.8 *A Usage Control Based Architecture for Cloud Environments*

A proposta expõe uma arquitetura para aplicar o modelo *UCON* em ambientes de nuvem e uma abordagem para manipular a mutabilidade dos atributos. Para possibilitar a implementação da proposta, a sintaxe e a semântica da linguagem de políticas *XACML* foi estendida (Tavizi, 2012).

As principais entidades da arquitetura são descritas a seguir (Figura 4.9):

- *Enforcement Point* - a tarefa deste componente é: receber os pedidos de acesso; emitir as solicitações de decisão para o módulo *Decision Point*; e executar as respostas retornadas.
- *Decicion Point* - este componente avalia toda a solicitação de decisão enviada pelo *Enforcement Point* de acordo com as regras armazenadas no *Policy Manager* (*e.g.* políticas de autorização, condição e obrigação). No processo de avaliação de políticas, o *Decicion Point* interage com o *Attibute Manager* para adquirir os atributos do sujeito, objeto e ambiente. O resultado da decisão é enviado para o *Enforcement Point* e as obrigações são enviadas para o *Obligation Handler*.
- *Context Handler* - responsável por converter as mensagens de um contexto para outro, proporcionando um entendimento comum das informações trocadas entre os componentes envolvidos.
- *Policy Manager* - administra e armazena as políticas de autorização, obrigação e condição. Adicionalmente, expressa os tipos de dados para o *Attibute Manager*. Este evento especifica a maneira como a mutabilidade dos atributos deve acontecer.
- *Attibute Manager* - recebe e armazena informações atualizadas referentes aos atributos do sujeito, objeto e ambiente (*i.e.* dados enviados por

sensores presentes no ambiente destes elementos). Visando manter um nível adequado entre segurança e sobrecarga computacional, o modelo executa um controle contínuo, de acordo com o grau de sensibilidade do atributo desejado (*e.g.* para atributos sensíveis, o *Decicion Point* é informado quando o atributo é alterado, fazendo com que a política seja re-avaliada).

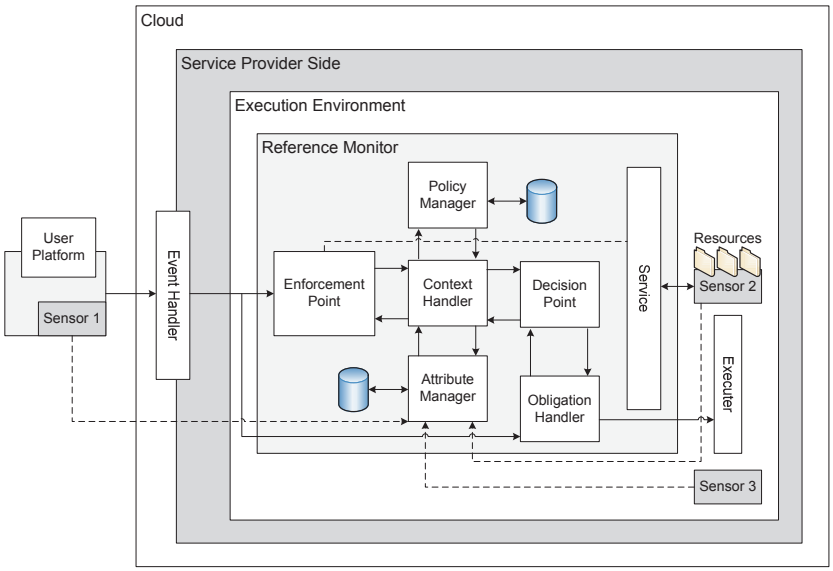

Figura 4.9: Arquitetura baseada em *UCON* para Ambientes de Nuvem. Adaptado de Tavizi, 2012.

- *Service* - através deste componente o usuário obtém acesso aos recursos protegidos.
- *Event Handler* - recebe e analisa eventos externos e encaminha estes para o componente adequado. Nos eventos incluem-se: receber solicitações de acesso ao serviço; executar ações que alterem o estado das obrigações do sujeito; disparar uma reavaliação de políticas; etc.
- *Executer* - este componente recebe as obrigações analisadas pelo *Obligation Handler* e executa estas em um *scheduler*.
- *Obligation Handler* - recebe e processa as obrigações enviadas pelo *Decicion Point*, emitindo os comandos apropriados para os módulos relacionados.

As entidades descritas a seguir fazem parte da abordagem para tratamento de obrigações (Figura 4.10):

Obligation analyzer and convertor: este módulo é responsável por analisar e converter as obrigações em um conjunto ordenado de tarefas, guardando estas em um armazenamento secundário. Cada registro armazenado corresponde a uma obrigação, sendo que cada uma destas é definida pelo seu campo descritor (*e.g.* tempo limite para a tarefa ser cumprida). Após analisar e armazenar os registros recebidos, o módulo envia a descrição das obrigações para o usuário do sistema. Antes do *Enforcement Point* conceder um privilégio para o usuário, este deve esperar pelos resultados do *Obligation Handler* (*i.e.* se existe alguma obrigação vinculada a decisão feita pelo *Decision Point*). Depois de analisar as obrigações, e caso não existam pendências, o comando de liberação será enviado para o usuário.

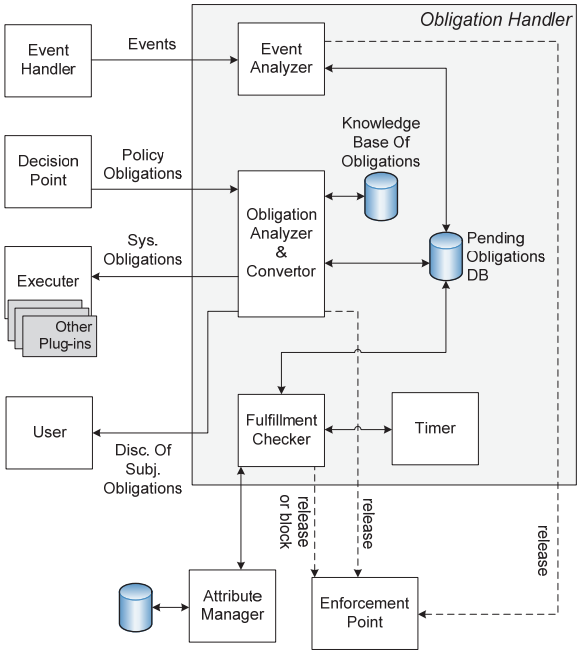

Figura 4.10: Arquitetura interna do módulo para tratamento de obrigações. Adaptado de Tavizi, 2012.

Event Analyser: este módulo verifica se existe alguma obrigação pendente relacionada ao evento atual. Se a ocorrência do evento implica em alguma obrigação ser cumprida, o *Event Analyser* apaga esta do armazenamento. Adicionalmente, se o valor limite da obrigação cumprida for zero, significa que a seção de consumo do usuário está aguardando. Neste caso, o módulo envia um comando de liberação para o componente

Enforcement Point.

Fulfillment Checker: este componente compara as condições de obrigações pendentes com atributos atualizados e, se cada uma das obrigações foi cumprida, as partes relacionadas são informadas. Se o valor limite de uma obrigação qualquer está configurado com zero, significa que a seção do usuário está esperando, sendo assim esse módulo informa o *Enforcement Point* e então apaga o registro do armazenamento. Porém, se o valor limite da obrigação é maior que zero, após apagar o registro do armazenamento, o processo pode ser finalizado. Adicionalmente, este componente é responsável por verificar se o tempo limite de qualquer obrigação terminou ou não. Caso isto aconteça, este módulo notifica o *Enforcement Point* para bloquear a seção de consumo do usuário e executar ações compensatórias contra este.

A proposta aplica o modelo *UCON* no que se refere ao tratamento das obrigações. Itens como a periodicidade do sistema de monitoramento de atributos, os quais refletem diretamente na autorização fornecida ao usuário, não são abordados pelo modelo.

4.9 *An architecture model of management and monitoring on Cloud services resources*

O artigo trata de uma abordagem para executar a administração de recursos para a nuvem (Sun, 2010). Esta tarefa envolve o monitoramento de serviços e processos de gerenciamento de *TI* (*IT Service Management - ITSM*). O principio básico do *ITSM* é o estudo orientado a processo, o qual adota os seguintes métodos: *carding* - transforma o gerenciamento técnico em gerenciamento de processo; *package* - transforma o gerenciamento de processo em gerenciamento de serviço.

A implementação do esquema proposto visa identificar os recursos que precisam ser gerenciados. O monitoramento dos serviços da nuvem pode ser dividido em basicamente quatro aspectos: disponibilidade, desempenho, rendimento (*throughput*) e utilização dos serviços. O modelo necessita coletar indicadores em tempo real e aprender sobre as condições de operação do serviço.

A abordagem proposta pode ser dividida em três seções (Figura 4.11): *1)* camada inferior - recupera as informações dos recursos utilizados nos serviços; *2)* camada intermediária - coleta e monitora as informações do nível inferior; *3)* seção de

monitoramento - acompanha a qualidade do serviço (*QoS*) referente aos recursos fornecidos para a nuvem (*e.g.* recursos físicos e virtuais).

O monitoramento dos recursos ocorre quando executa-se o acesso a plataforma dos serviços da nuvem. Para coletar as informações, um agente de monitoramento pode ser instalado em cada nodo de serviço. Visando ampliar o alcance do gerenciamento, esta camada utiliza *plugins* para dar mais escalabilidade a monitoração. A plataforma de acesso para os serviços da nuvem transfere as informações coletadas para a camada de monitoração dos recursos, através da tecnologia *JMX* (*Java Management Extension*).

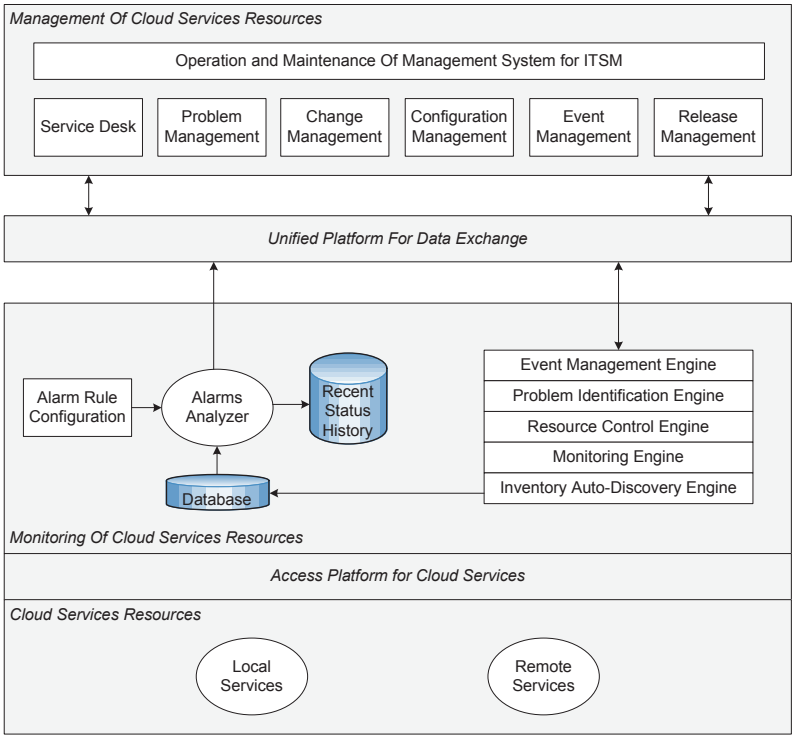

Figura 4.11: Modelo de arquitetura para gerenciamento e monitoramento. Adaptado de Sun, 2010.

O monitoramento da nuvem pode ser dividido em duas partes: processamento de dados e alarmes. A função do processamento é recolher informações de monitoramento, pesquisar e analisar estes dados. O processamento possui conhecimento quanto as informações monitoradas através da mineração de dados e inteligência nos negócios (*Business Intelligence - BI*).

Os processos de filtragem, tratamento e processamento dos dados podem ser executados pelos mecanismos presentes na camada de monitoração (Figura 4.11): *auto-discovery engine* - cria relatórios dos recursos monitorados; *monitoring engine* - monitora

os dados que são transferidos pelos agentes; *resource control engine* - controla o recurso monitorado com a ajuda dos agentes; *problem identification engine* - visa encontrar falhas na camada de monitoramento; *event management engine* - registra os *logs* e gerencia os eventos da camada de monitoramento.

As informações armazenadas na camada de monitoramento são fornecidas ao analisador de alarmes. Esta entidade avalia estes dados em relação aos detalhes de configuração para verificar se alguma condição de alarme foi satisfeita. O analisador de alarmes verifica seu histórico para detectar se um alarme é novo, foi previamente detectado, ou já é considerado inválido.

A proposta descrita apresenta um modelo centralizado de gerenciamento e monitoramento. Itens como políticas de acesso ou o controle do consumo de serviços não são tratados pela abordagem. Um ponto interessante referente a arquitetura é a utilização de agentes de monitoramento instalados em cada nodo de serviço. Ou seja, cada máquina virtual instanciada no provedor, ou até mesmo cada serviço, pode possuir um agente que execute a coleta dos atributos do sistema.

4.10 *Usage Management in Cloud Computing*

A proposta descrita nesta pesquisa fornece uma análise das características e desafios envolvidos na implantação de uma estrutura de gerenciamento de uso no ambiente de nuvem. A análise permite a interpretação e aplicação automatizada de políticas (Jamkhedkar, 2011).

O gerenciamento de uso para a nuvem se refere a uma coleção de processos e mecanismos que permitem administrar e controlar como os dados são utilizados dentro do sistema. Este processo engloba os seguintes itens: linguagens de especificação de políticas; os mecanismos de licença; a forma de expressar as políticas; os mecanismos de aplicação de políticas; o registro de uso; e o suporte a mecanismos de autenticação e criptografia.

As políticas de uso precisam ser administradas por um *framework* de gerenciamento que vai funcionar sobre uma infraestrutura distribuída. O usuário e o provedor da nuvem negociam a política de uso e o respectivo preço associado de maneira automatizada, via agentes de *software* (*e.g. Java Agent Development Framework*). No próximo passo, a política é representada em um formato entendível e executável pela

máquina. Esta política é então interpretada e aplicada na infraestrutura da nuvem.

O *framework* de gerenciamento deve suportar a interpretação dinâmica das políticas de uso, permitindo a aplicação das regras através de composições ou cadeias de serviços. A interpretação dinâmica permite que políticas sejam escritas em diferentes níveis de abstração, sendo interpretadas apropriadamente, de acordo com as particularidades de cada ambiente de serviço. Esta abordagem assegura que as políticas sejam utilizadas corretamente, mesmo se os dados dos usuários sejam enviados para nuvens previamente desconhecidas.

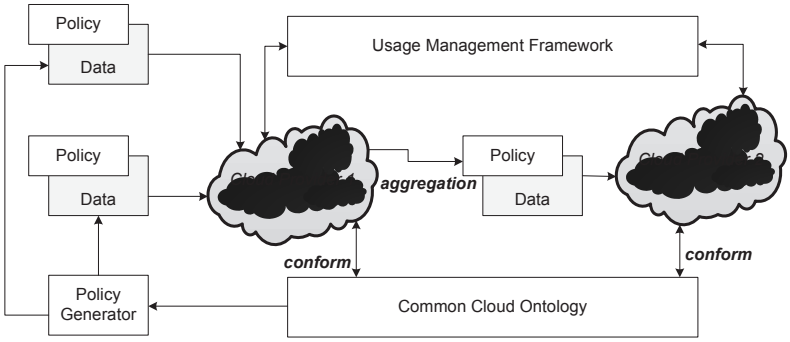

Figura 4.12: Persistência de políticas entre as agregações de dados. Adaptado de Jamkhedkar, 2011.

Um serviço fornecido pelo provedor de nuvem *1* (Figura 4.12) pode agregar dados de diferentes fontes. Estes dados são enviados para o serviço fornecido pelo provedor de nuvem *2*. Neste cenário, cada uma das fontes de dados são governadas por suas próprias políticas de uso. Depois de combinados, o conjunto de dados agregado é controlado por uma política de uso única, formada pela associação das duas políticas originais.

A operação da arquitetura proposta é divida em duas fases: fase de configuração - trata da conformidade das políticas de uso e a configuração do ambiente de nuvem; fase de trabalho - trata da interpretação e execução das políticas de uso. Na fase de configuração, as políticas são avaliadas dentro de um contexto particular, estando este separado da sintaxe e da semântica das linguagens de política.

O contexto do ambiente pode ser composto por sub-serviços de processamento de dados (*dps*) com os seguintes parâmetros: {*date*, *dps_location*, *dps_trust-level*}. Onde *date* - representa a data global; *dps_location* - representa a localização do sub-serviço; *dps_trust-level* - representa o nível de confiança do sub-serviço, variando de *1* a *5*. Uma política de uso poderia declarar que: o conjunto de dados *X* pode ser processado por um *dps* localizado nos Estados Unidos, com níveis de confiança superior a *4*, e não após *16*

de dezembro de *2013*.

Para o exemplo supracitado é importante notar que todo serviço de nuvem possui um contexto representativo para o qual os parâmetros precisam ser preservados. Durante a fase de configuração, os serviços geram seu próprio contexto a partir de uma ontologia comum, compartilhada entre todos os serviços da nuvem. Dependendo da natureza dos serviços, estes compartilham parâmetros comuns (*e.g.* hora, data). Utilizando as restrições de políticas, os parâmetros são definidos de acordo com o contexto da nuvem onde a política vai ser interpretada.

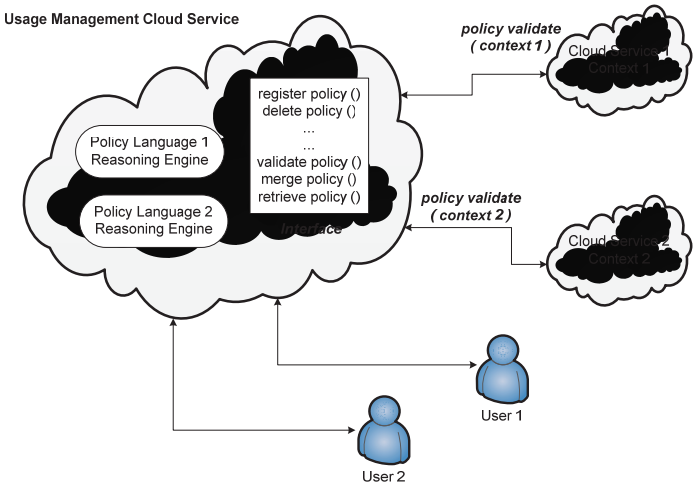

Figura 4.13: Operações de gerenciamento de uso no ambiente da nuvem. Adaptado de Jamkhedkar, 2011.

A fase de trabalho consiste no gerenciamento, interpretação e validação das políticas. Estes procedimentos ocorrem através de um serviço de gerenciamento de uso compartilhado pelas entidades que fazem parte da nuvem (Figura 4.13). Ao invés de padronizar uma linguagem de política para todos os serviços da nuvem, a arquitetura utiliza um serviço de nuvem para o gerenciamento de uso. Este serviço fornece uma interface *web* para que as políticas sejam registradas, recuperadas, anexadas aos conjuntos de dados, validadas, interpretadas, ou até mesmo mescladas com outras políticas provenientes de diferentes provedores.

Para validar as políticas, os serviços de nuvem interagem com a interface de gerenciamento de uso por uma ação particular em um conjunto de dados, fornecendo o contexto através do qual a ação pretende ser executada. O serviço de gerenciamento avalia se a política e o contexto são inter-operáveis através da comparação de suas condições. Se as condições são consistentes, as restrições da política são avaliadas

considerando o estado atual do contexto fornecido. Caso as restrições sejam satisfeitas, a ação é liberada para ser executada, caso contrário, a ação é negada.

A proposta apresentada sugere que os serviços apliquem políticas recuperadas de uma interface *web* padronizada. Esta abordagem assegura que a sintaxe e a semântica das diferentes linguagens fiquem escondidas dos serviços que necessitam aplicar as respectivas políticas de uso. O esquema apresentado evita que todo provedor de serviço precise implementar um interpretador para cada linguagem de política. No presente trabalho de pesquisa, utiliza-se um ambiente federado, responsável por tornar homogêneas as políticas específicas de cada domínio consumidor.

4.11 *Conclusão*

A implantação de novos mecanismos de gerenciamento é imperativa para as nuvens computacionais. A utilização de sistemas de controle de acesso com granularidade fina, sendo auxiliados por atributos recuperados dinamicamente do ambiente, é claramente exposta nos trabalhos relacionados. A dinamicidade oferecida pela nuvem computacional necessita de abordagens flexíveis, e que sejam ao mesmo tempo altamente seguras e confiáveis. O Apêndice B complementa estes trabalhos, esta seção visa explorar esforços adicionais disponíveis na literatura.

O próximo capítulo introduz a proposta executada por este projeto de pesquisa. A abordagem visa contemplar alguns itens que não foram tratados nos trabalhos relacionados, oferecendo um esquema que seja compatível com o modelo de computação em nuvem.

Capítulo 5

Proposta

Esta proposta visa aplicar o modelo de autorização contínua do $UCON_{ABC}$, provendo resiliência para o processo de reavaliação das políticas de uso. Resiliência, nesta proposta, significa fornecer ao modelo a habilidade de tratar algumas situações de exceção que ocorrem com os atributos de autorização de um usuário. A resiliência é garantida enquanto o *SLA* do respectivo serviço consumido está de acordo com a quantidade definida no contrato (Figura 5.1; *evento* SLA_{CO}).

O Domínio Consumidor (*DC*) escreve as políticas de uso para seus usuários (*evento* U_{AQ}), estabelecendo os atributos de autorização individuais (*i.e.* as cotas de uso de serviço para o usuário). A cota é utilizada durante o processo de reavaliação em substituição aos atributos de autorização. O objetivo desta abordagem é relaxar/flexibilizar a política de uso quando possível, lidando com situações de exceção de autorização.

Os atributos de consumo são obtidos para cada serviço sendo utilizado pelo usuário (*evento* A_{UU}), ou seja, as informações que contabilizam a seção de consumo do usuário são obtidas do provedor através de Agentes de Monitoramento (*AGM*). A resiliência para o processo de reavaliação de autorização contínua (R_{AR}, (***1***)) é definida somente se o SLA_{CO} menos a soma que contabiliza os atributos de consumo dos usuários (*sum of user accounting - sua*) é maior que *t*. A constante *t* é uma cota reserva livremente definida pelo consumidor para um determinado tipo de serviço ou recurso.

$$R_{AR} \exists \left[\left(SLA_{CO} - \left(\sum_{i=1}^{n} A_{UU_i} \right) \right) > t \right] \quad (1)$$

Isto significa que, quando a soma do consumo total dos usuários (*sua*) estiver próxima da quantidade definida em "SLA_{CO} - *t*", um novo *SLA*, contendo um montante

adicional, deveria ser negociado para tentar evitar uma violação de contrato. Visando facilitar o entendimento da proposta, um cenário envolvendo um sistema de arquivos é utilizado. O objetivo é explicar como a cota é aplicada no modelo, provendo resiliência ao processo de autorização $UCON_{ABC}$. Para o contexto descrito (Figura 5.1), o esquema de resiliência está representado como as linhas pontilhadas para os atributos de consumo e de autorização.

Considerando um consumidor que negocia um SLA_{CO} para um serviço (*e.g.* espaço de armazenamento de *600Gb*, sendo *t* igual a *100Gb*). O consumidor escreve políticas de uso que definem *userA: 200Gb*, *userB: 200Gb* e *userC: 100Gb*. Quando o usuário solicita acesso ao provedor (Figura 5.1; *evento AR*), o Guardião do Serviço (*GS*) envia uma solicitação de avaliação de autorização referente ao usuário para o Monitor de Referência *UCON* (*MRU*; *evento PER*). Por sua vez, o *MRU* configura a cota de uso para o usuário solicitante do serviço (*i.e.* inicialmente, o valor da cota é igual aos atributos de autorização definidos pelo consumidor; *userA: 200Gb*, *userB: 200Gb* e *userC: 100Gb*) e fornece uma resposta de permissão de consumo para o mecanismo de execução de controle de acesso (Guardião do Serviço - *GS; evento PDE*).

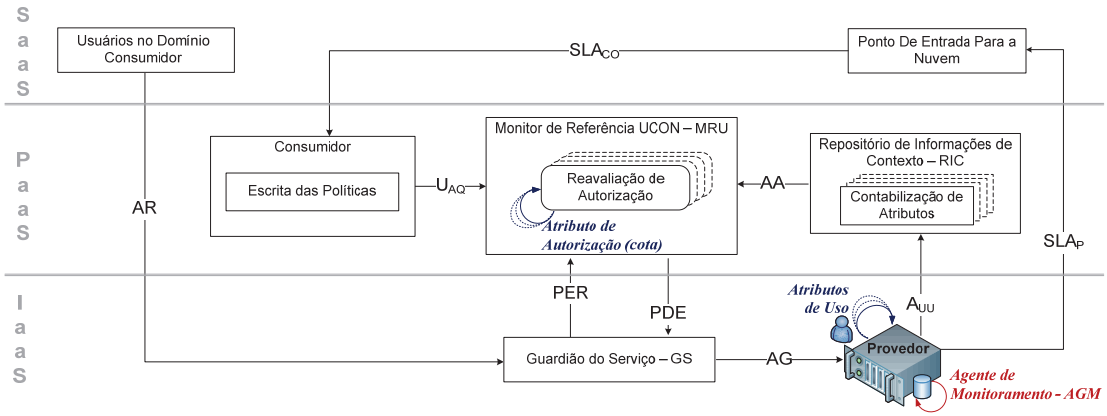

Figura 5.1: Modelo de autorização contínua $UCON_{ABC}$.

Após ter o acesso ao provedor liberado pelo Guardião do Serviço (*evento AG*), o usuário inicia a utilização do armazenamento. Posteriormente, os Agentes de Monitoramento (*AGM*) enviam atributos de consumo de cada usuário (*e.g. userA: 190Gb*, *userB: 10Gb* e *userC: 0Gb*) para o Repositório de Informações de Contexto (*RIC*). Depois de algum tempo, o Guardião do Serviço solicita uma reavaliação da autorização. Durante a reavaliação (autorização em tempo de execução - *ongoing*), o Monitor de Referência *UCON* (*MRU*) compara os atributos de consumo de cada usuário (*evento AA*)

com a cota (*i.e.* mecanismo que implementa a resiliência para o processo de autorização). Neste caso, o consumo do espaço de armazenamento, referente a cada usuário, está abaixo das cotas previamente estabelecidas.

Um período de tempo depois, os Agentes de Monitoramento (*AGM*) enviam novamente os atributos de consumo referentes a cada usuário (*e.g. userA: 250Gb*, *userB: 20Gb* e *userC: 10Gb*) para o Repositório de Informações de Contexto (*RIC*) e uma reavaliação de autorização é necessária. Durante a reavaliação de autorização, o Monitor de Referência *UCON* (*MRU*) percebe que os atributos de consumo do *userA* estão excedendo a cota previamente estabelecida (*i.e.* o *userA* foi autorizado a utilizar *200Gb* de espaço de armazenamento). Neste momento, a condição de soma de consumo para os usuários do consumidor (*sua*) é de *280Gb*. Tendo em vista que o "SLA_{CO} - *t*" permite *500Gb* de consumo de armazenamento para o consumidor, e o *userA* está excedendo o limite de uso em *50Gb*, a cota para este usuário em situação irregular é automaticamente expandida para *userA: 250Gb*.

Com a continuidade da utilização, os Agentes de Monitoramento prosseguem enviando os atributos de consumo referente a cada usuário (*e.g. userA: 240Gb*, *userB: 160Gb* e *userC: 100Gb*) para o Repositório de Informações de Contexto (*RIC*), enquanto o Guardião do Serviço (*GS*) prossegue solicitando as reavaliações de política. Durante a reavaliação de autorização feita pelo Monitor de Referência *UCON* (*MRU*), este percebe que os atributos de consumo do *userA* estão abaixo da cota expandida, porém, a condição de soma de consumo para os usuários do consumidor (*sua*), é de *500Gb* de espaço de armazenamento utilizado. Neste momento, o processo de reavaliação (*evento* R_{AR} (***1***)) detecta que o armazenamento consumido, subtraído da cota reserva (SLA_{CO} - *t*), está completamente preenchido.

Adicionalmente, a cota do *userA* está excedendo a política de uso, sendo que a parcela de armazenamento extra, adicionada anteriormente, pode ser equiparada com o parâmetro definido pelo atributo de autorização (*i.e.* política de uso), *userA: 200Gb*. Se, na próxima reavaliação, os atributos de consumo do usuário *userA*, informados pelo Agente de Monitoramento (*AGM*), mantiverem os mesmos valores além da cota, e a soma total dos atributos (*sua*) estiver próxima do "SLA_{CO} - *t*", o *userA* vai estar em uma condição de exceção (*i.e.* a cota do *userA* é *200Gb* mas os atributos de consumo contabilizam um valor excedente). Caso contrário, a cota do *userA* poderia ser alterada novamente.

As condições de exceção ocorrem quando o usuário, durante uma reavaliação, é

detectado violando uma cota que autorizava seu acesso em uma avaliação prévia. A condição de exceção acontece porque durante o período de reavaliação (atual e anterior), o usuário continua a consumir o serviço e exceder o limite de sua cota. Além disso, essa situação pode ser desencadeada pela resiliência do modelo, que redefine a cota para o valor do atributo de autorização. Uma exceção também pode ocorrer se, entre os períodos de reavaliação, houver alguma mudança que torne a política atual mais restritiva do que a política que foi previamente definida e avaliada.

Quando um usuário é detectado em condição de exceção, o consumidor é informado através dos atributos de consumo consolidados mantidos pela federação. Neste contexto, o consumidor deve reescrever as políticas de uso visando trazer a condição do usuário ao estado normal, antes que a condição de exceção seja alcançada. Caso contrário, o consumo do serviço pelo usuário será negado enquanto a condição estiver pendente.

A condição de exceção pode indicar para o consumidor que mais serviços precisam ser contratados (*i.e.* quando a soma total dos atributos de consumo (*sua*) estiver próxima do SLA_{CO}, o consumidor pode solicitar a alocação de uma quantidade adicional (*evento* SLA_P; Figura 5.1), desde que o provedor possua recursos disponíveis). A elasticidade fornecida pela nuvem pode ser utilizada em uma solicitação automática por mais serviços, quando estes se esgotarem para o consumidor. Esta abordagem permite que o modelo de resiliência seja fornecido de maneira contínua.

Para propósitos administrativos, se muitos usuários estão passando por uma condição de exceção, o consumidor pode concluir que mais serviços precisam ser contratados (*e.g.* atualização de *SLA*). Nas abordagens tradicionais, quando o usuário alcança o limite definido pela cota, não existe suporte para atender valores de consumo excedentes nos processos de autorização. Ou seja, o administrador precisa pesquisar por usuários que não estão utilizando a quantidade de recursos permitida pela cota e ajustar as políticas manualmente.

A atualização das regras é feita sem o auxílio de atributos de consumo que refletem as reais condições do sistema, sendo uma tarefa manual e cansativa. Basicamente, não há suporte para decidir como alterar a política para um determinado usuário. Com a utilização do modelo de resiliência proposto, através da aplicação de cotas, obtêm-se um balanço automático para o limite estabelecido na política e a quantidade contratada em *SLA*. Adicionalmente, somente quando a soma dos atributos de consumo de todos os usuários (*sua*) alcançar o limite definido no SLA_{CO}, o administrador do consumidor precisa agir. Porém, como o administrador possui acesso a atributos de consumo

consolidados para cada usuário, fica fácil decidir o que pode ser feito. A reavaliação provê resiliência ao processo de autorização, explorando a ociosidade na utilização do serviço quando a soma dos atributos de consumo dos usuários (*sua*) estiver abaixo do valor definido pelo "SLA_{CO} - t".

Outra opção para tratar desta situação de exceção seria aguardar um período de tempo. Isto significa que, durante um determinado período, a condição de exceção seria tolerada, esperando que a utilização do serviço pelo usuário seja finalizada ou volte ao parâmetro normal. Esta abordagem visa preservar o trabalho em andamento no serviço, na tentativa de não desperdiçar o que já foi consumido. A exceção também pode ser tolerada para permitir ao consumidor avaliar se a política para o usuário deveria ser modificada, visando adequar as regras ao perfil de consumo do usuário. Nesta circunstância, a utilização do serviço pelo usuário poderia ser suspensa até que a política de uso seja atualizada. Após a alteração das regras, o consumo do serviço pode ser retomado pelo usuário.

Nas abordagens tradicionais, a autorização é avaliada somente no início da utilização (pré-autorização - *pre-authorization*), porém isto pode gerar inconsistências entre a autorização concedida e o atributo de autorização ativo (*i.e.* política atual). Mesmo a reavaliação de autorização durante (*ongoing*) o acesso não é suficiente para garantir que algum consumo autorizado para um usuário não estará violando a política. Isto pode acontecer devido à condições de exceção serem disparadas entre os períodos de reavaliação. Nesta proposta, estes períodos são menores do que nas abordagens tradicionais, dependendo somente do intervalo de tempo entre as reavaliações de atributos de autorização.

Na computação em nuvem, quando um usuário está em condição de exceção e o prazo para regularizar a situação chega ao fim, ainda é possível suspender a utilização do serviço durante um período de tempo. Durante este período, o consumidor pode ajustar a condição do usuário visando trazer a situação novamente ao estado normal. Por outro lado, a utilização do serviço pode ser revogada ao custo de perder o trabalho executado até o momento.

A elasticidade fornecida pela computação em nuvem é utilizada para adicionar, de maneira automática, mais instâncias de Monitores de Referência *UCON* e Repositórios de Informações de Contexto. Estes procedimentos ocorrem quando a demanda por reavaliações de autorização ou a recuperação de atributos de consumo aumentar. A abordagem proposta, agindo como uma plataforma de serviços (*PaaS*), libera o

consumidor da responsabilidade de tratar dos aspectos de controle, deixando o contratante livre para concentrar seus esforços somente no desenvolvimento e implantação de serviços para atender ao seu ramo de negócio.

Em resumo, esta proposta permite o monitoramento, reavaliação contínua e reconfiguração das políticas de uso definidas para o usuário de acordo com a necessidade, ou enquanto o contrato permitir (*i.e.* enquanto o limite definido no *SLA* não for alcançado). Adicionalmente, a flexibilidade fornecida pelo ambiente de avaliação, suportando vários monitores de referência trabalhando em paralelo, fornece suporte a elasticidade no controle de uso.

5.1 *Arquitetura Proposta*

Este trabalho é baseado em uma nuvem computacional formada por vários tipos de serviços, provedores e consumidores, sendo que cada um destes pode estar trocando dados com diferentes camadas do sistema (*e.g. SaaS*, *PaaS*, *IaaS*). Um ambiente de intermediação (*i.e.* federação) é utilizado visando facilitar a interação entre as entidades que estão acessando a nuvem computacional (Figura 5.2). A federação oferece um *PaaS* que é orientado pelo modelo de segurança e controle de acesso $UCON_{ABC}$, este utiliza o esquema de reavaliações contínua (*ongoing*) para as autorizações. Com a divisão do ambiente de nuvem em diferentes domínios é possível criar escopos de proteção reduzidos que permitem regrar e monitorar as interações entre as partes.

O sistema de gerenciamento foi projetado levando em consideração vários aspectos: *i)* controle de uso aplicando políticas definidas pelo Domínio Consumidor (*DC*) para usuários individuais; *ii)* monitoramento de uso (*i.e.* contabilização de atributos) para cada usuário pertencente ao consumidor; *iii)* reavaliação contínua da autorização concedida para o usuário; *iv)* possibilidade de redefinição das políticas de controle de uso considerando atributos consolidados e a disponibilidade do contrato de serviço (*SLA*).

O ponto de entrada para o ambiente da nuvem computacional para os provedores, consumidores e respectivos usuários é disponibilizado por um *Broker* (Figura 5.2). O *Broker* executa as seguintes funções: intermediação das ofertas de serviços pelos provedores; a negociação com os consumidores para que estes obtenham acesso ao serviço; definição dos *SLAs* entre as partes interessadas; redirecionamento dos usuários do consumidor para o endereço de rede do provedor, local onde o serviço contratado está

disponível.

O provedor se apresenta para o *Broker* como um fornecedor de recursos e serviços para o ambiente da federação. Para cada serviço oferecido, o provedor negocia um *SLA* com o *Broker* e envia para este a descrição da interface de acesso. O *SLA* define os parâmetros para o serviço, este é a fonte de informações para o *Broker* derivar as quantias (*e.g.* cotas de uso para armazenamento em disco) que podem ser oferecidas para o Domínio Consumidor (*DC*). Cada consumidor pode ser direcionado para um ou mais provedores, dependendo da quantia contratada. A descrição da interface de acesso é fornecida para os usuários do consumidor, permitindo que estes implementem seus sistemas de consumo de acordo com os serviços oferecidos pelos provedores (W3C, 2007b).

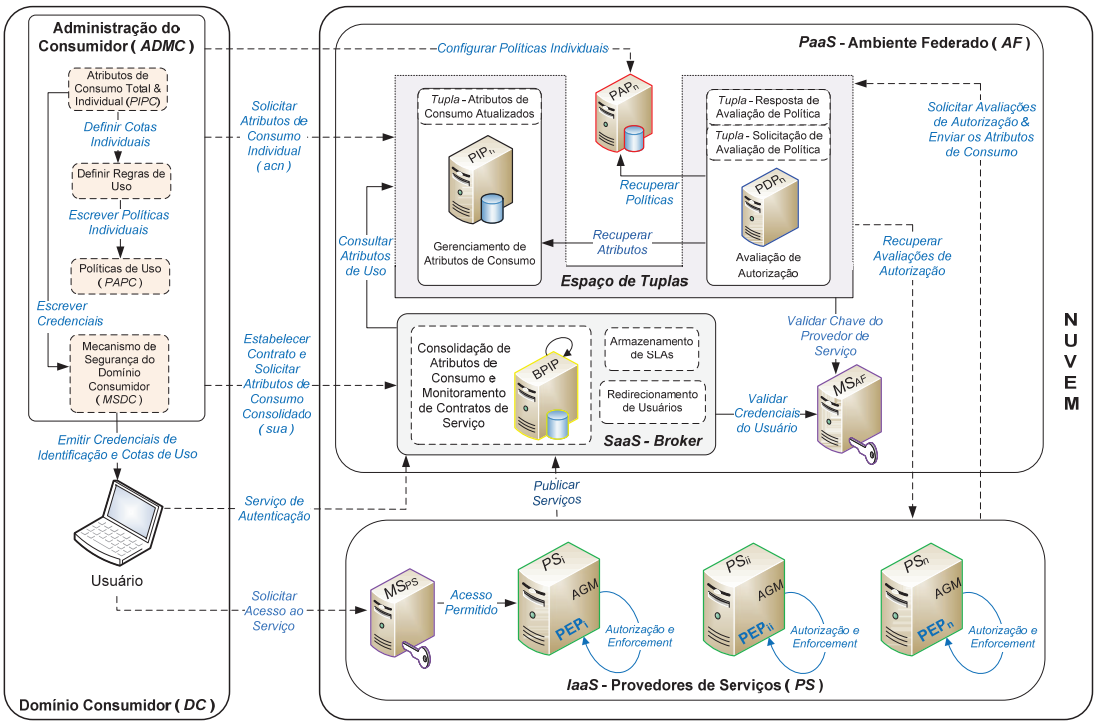

Figura 5.2: Visão geral da proposta.

O consumidor estabelece um *SLA* com o *Broker*, componente que representa os provedores federados, e define as políticas de uso para os usuários de seu domínio de acordo com o serviço contratado. As políticas de controle de uso serão configuradas no ponto de administração de políticas (*Policy Administration Point - PAP*) do Ambiente Federado (*AF*). O *SLA* contratado fornece para cada consumidor um conjunto de serviços utilizados para gerenciar o ambiente (*e.g.* serviços para o armazenamento de atributos,

políticas, credenciais e chaves criptográficas). As credencias de autenticação geradas pelo Domínio Consumidor para os usuários consideram o total de serviços descritos no *SLA*. Estas credenciais permitem aos usuários interagirem com o *Broker* para que estes possam obter uma referência para o fornecedor do serviço solicitado.

O usuário do Domínio Consumidor (*DC*) recebe uma cota que define os atributos de consumo (direitos) para a utilização de um serviço. A cota inicial é igual a quantidade definida na política de uso. Antes do usuário poder utilizar a sua cota, este deve se apresentar ao *Broker*. Neste contato inicial, o usuário deve fornecer sua credencial de autenticação (*e.g.* a identificação do solicitante assinada pelo Domínio Consumidor) (Stihler, 2012). O *Broker* utiliza a política de uso, disponibilizada pelo *DC*, para selecionar o provedor que melhor se enquadra na solicitação de acesso ao serviço. Na resposta do *Broker*, o usuário recebe a descrição da interface do serviço e, a partir deste momento, passa a acessar diretamente o Provedor de Serviço (*PS*). Esta abordagem permite ao *Broker* alocar os usuários em provedores específicos, de acordo com as demandas do *SLA* de cada consumidor.

As solicitações de acesso recebidas pelos provedores que formam a federação são interceptadas pelo Guardião do Serviço (*Policy Enforcement Point - PEP*), e avaliadas por implementações de Monitores de Referência *UCON* (*Policy Decision Point - PDP*) instanciados no Ambiente Federado (*AF*). Os atributos de consumo do usuário são armazenados e fornecidos pelo Repositório de Informações de Contexto (*Policy Information Point - PIP*). O repositório de contabilização de atributos é um serviço atualizado por Agentes de Monitoramento (*AGM*) sendo executados nos Provedores de Serviço da federação.

A reavaliação de políticas executada pelo *PDP* provê resiliência para o processo de autorização. Esta avaliação compara os atributos de consumo do usuário com relação a sua cota. Este processo visa explorar a ociosidade na utilização dos serviços, caso os atributos de consumo de um usuário forem maiores que sua cota. Sendo assim, se a soma dos atributos de consumo dos usuários (*sua*) estiver abaixo da quantia definida em um determinado contrato SLA_{CO} (Figura 5.1), significa que existe ociosidade na utilização dos serviços. Neste caso, a cota para o usuário que excedeu o limite inicial pode ser expandida. Quando a *sua* alcançar o limite de uso definido no "SLA_{CO} - *t*", todas as cotas expandidas são retornadas para os valores pré-definidos na política de uso. Lembrando que a constante "*t*" é uma cota reserva, livremente definida pelo consumidor.

Depois do consumidor configurar as políticas de uso, baseado no *SLA*, e o usuário

iniciar a utilização do serviço, o *PIP* passa a armazenar os atributos de consumo de cada usuário. Periodicamente, o *Broker* consolida os atributos de consumo para cada consumidor, verificando assim se o SLA_{CO} do serviço está sendo honrado. No Domínio Consumidor, os atributos consolidados são analisados para decidir se a política de um usuário precisa ser ajustada. Aplicando esta abordagem, o consumidor é capaz de redefinir as políticas de uso de acordo com a necessidade de cada usuário, otimizando o consumo do serviço e minimizando as exceções nas políticas de uso.

A federação provê interoperabilidade, alta disponibilidade e elasticidade, aplicando uma abordagem baseada em memória compartilhada distribuída. O espaço de memória comum, implementado como um serviço de espaço de *tuplas*, oferece serviços de gerenciamento que permitem: o envio de solicitações de avaliação e reavaliação de autorização - função dos *PEPs*; o armazenamento das respostas das avaliações e reavaliações - função dos *PDPs*; e o monitoramento dos atributos de consumo de cada usuário do Domínio Consumidor - função dos *PIPs*. Esta abordagem permite que o serviço de espaço de *tuplas* seja acessado por vários Agentes de Monitoramento instanciados no provedor para a escrita dos atributos de consumo. Uma descrição mais completa, referente ao ambiente de memória compartilhada distribuída, pode ser encontrada no Apêndice *A*.

O ambiente de avaliação de políticas é baseado no modelo de terceirização (*outsourcing*). Na abordagem proposta, isto significa que a avaliação é feita no Ambiente Federado (*AF*). Porém, não de maneira centralizada, considerando que se aplica o serviço de espaço de *tuplas* que suporta vários *PDPs* trabalhando em paralelo. Analisando a abordagem de avaliação utilizada, constatou-se que o modelo *outsourcing* é mais adequando ao contexto da computação em nuvem devido à dinamicidade inerente ao ambiente (*i.e.* frequente instanciação e destruição de máquinas virtuais).

No modelo de terceirização, nenhum tempo é gasto transferindo políticas e gerenciando sistemas de *cache* no ambiente do provedor para cada serviço instanciado (Marcon, 2009a) (Marcon, 2009b). Ou seja, para cada novo serviço disponibilizado, as políticas para os usuários do Domínio Consumidor deveriam ser configuradas no respectivo serviço. Se vários serviços forem instanciados devido a um pico na demanda de uso, os pedidos efêmeros de avaliação de políticas de usuário não justificam o custo da transferência e gerenciamento das políticas no provedor. Na atual proposta, o modelo *outsourcing* consolida as políticas no *PAP*, proporcionado um controle mais consistente das regras de acesso e evitando a disparidade entre estas (*i.e.* inconsistências causadas

pelo *cache* de políticas no domínio do provedor).

A expansão ou retração dos serviços de gerenciamento (*e.g.* ambiente de avaliação de políticas) ocorre de acordo com a demanda do contexto (*i.e.* conforme a quantidade de consumidores e agentes utilizando o espaço de *tuplas*). A aplicação do espaço de *tuplas* para o compartilhamento de dados em contextos distribuídos, como a computação em nuvem, visa criar um desacoplamento temporal entre consumidores e provedores de informação. Neste contexto, as entidades que escrevem ou fazem a leitura de conteúdos (*i.e. tuplas*) podem ser assíncronas, não necessitando conhecer umas as outras. A interação pode ocorrer durante o período de tempo que as entidades concordem com o formato do *template* utilizado para armazenar dados dentro do espaço comum.

A expansão ou retração dos serviços ocorre automaticamente, sendo transparente para ambos, consumidor e provedor. Quanto mais consumidores, maior será o *pool* de máquinas virtuais para dar suporte ao espaço de *tuplas*. Cada consumidor necessita de serviços de gerenciamento e Provedores de Serviços adequados para atender seu domínio. O esquema proposto, baseado no modelo de nuvem computacional, fornece alta disponibilidade e elasticidade, com baixo acoplamento funcional entre as entidades do ambiente. Estes objetivos são alcançados aplicando a infraestrutura de nuvem para gerenciar o *pool* de servidores que hospedam os serviços de espaço de *tuplas*, *PDPs*, *PIPs* e *PEPs*.

5.1.1 Domínio Consumidor

O Domínio Consumidor (*DC*) abrange os seguintes principais componentes: módulo de gerenciamento (*ADMC*) - entidade responsável por contratar os serviços oferecidos pelo Ambiente Federado (*AF*; *evento ad*; Figura 5.3); mecanismo de segurança do domínio (*MSDC*) - incumbido de emitir as credencias de autenticação para os usuários (*evento au*; Figura 5.3); administração de políticas (*PAPC*; Figura 5.2) - encarregado de armazenar as políticas de uso referentes aos usuários do consumidor; gerenciamento de atributos (*PIPC*; Figura 5.2) - responsável por consolidar os atributos de consumo disponibilizados pela federação; usuários - utilizam os serviços contratados pelo domínio (*evento ac*; Figura 5.3).

A Administração do Consumidor (*ADMC*; Figura 5.3) escreve as credencias de autenticação para seus usuários no *MSDC* (Mecanismo de Segurança do Domínio

Consumidor; *evento re*) de acordo com os parâmetros extraídos do *SLA*. A definição das políticas de uso para os usuários é feita com base no contrato (SLA_{CO}) negociado com o *Broker* (*evento ad*).

As políticas definem para cada usuário as restrições de uso individuais, referente a utilização do serviço, e as credenciais de acesso que serão utilizadas (*evento au*) para solicitar a descrição do serviço no *Broker* (*evento is*). A descrição informa ao usuário como acessar e consumir os serviços (*evento ac*) nos provedores (PS_n) associados a federação (*evento sp*) (W3C, 2007b).

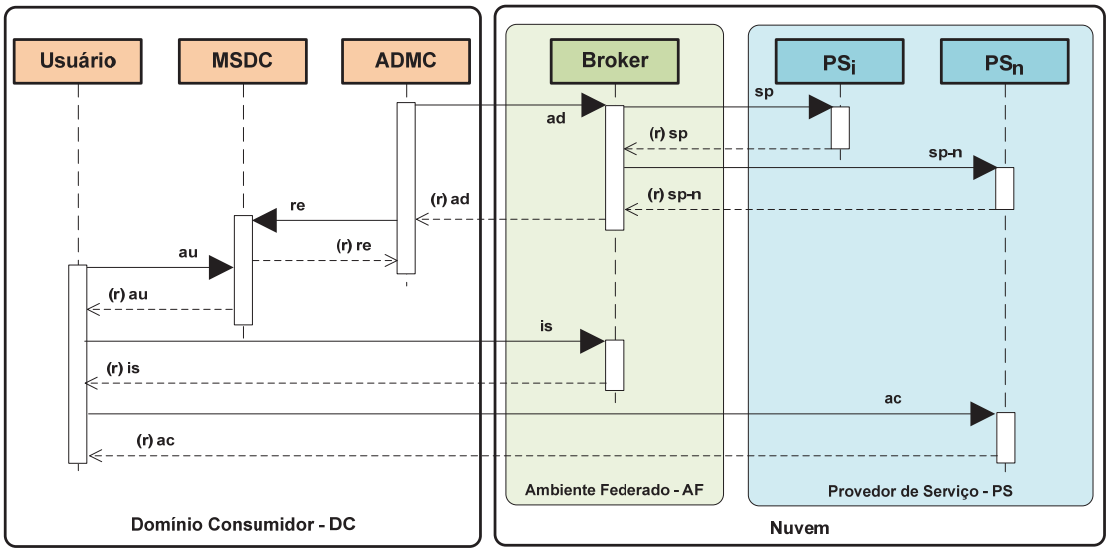

Figura 5.3: Domínio Consumidor.

5.1.2 Ambiente Federado e Provedores

O Ambiente Federado (*AF*), representado pelo *Broker*, disponibiliza para o usuário o ponto de entrada para acessar os recursos e serviços fornecidos por todos os provedores associados a federação. A federação, através do *Broker*, se torna a entidade responsável por administrar o contrato *SLA* estabelecido com o consumidor. Quando um usuário necessita de um serviço, o *Broker* escolhe entre os provedores federados aquele que melhor atende as necessidades do consumidor (*i.e.* as políticas de uso). É responsabilidade do *Broker* encontrar os provedores que satisfaçam as necessidades do consumidor, e aplicar o redirecionamento dos usuários visando otimizar a alocação e utilização dos recursos e serviços (Amit, 2011).

O Provedor de Serviço (*PS*) é uma entidade associada a federação. O processo de filiação é gerenciado externamente ao contexto desta proposta. Porém, uma vez associado, o provedor começa a usufruir do ambiente de gerenciamento e da infraestrutura de suporte do Ambiente Federado. Os recursos e serviços disponibilizados neste ambiente podem ser divididos de acordo com a área de cobertura geográfica da nuvem ou o tipo de funcionalidade oferecida pelo provedor. Alguns provedores de serviço (camada *SaaS*) contratam provedores de infraestrutura de nuvem (camada *IaaS*) para hospedar os sistemas ofertados aos consumidores. Esta abordagem dificulta o gerenciamento, de maneira consolidada, do ambiente distribuído implementado pela computação em nuvem.

O provedor mantêm seus próprios serviços *web* (camada *SaaS*), os quais fornecem acesso aos níveis subjacentes (camadas *PaaS* e *IaaS*). Na camada intermediária (*PaaS*), os serviços de controle interceptam as solicitações de acesso (*i.e.* recebidas no nível *SaaS*), monitorando e aplicando as políticas para cada pedido enviado por usuários pertencentes ao Domínio Consumidor. Quando o acesso é liberado na camada *PaaS*, o usuário está autorizado a utilizar os recursos sendo fornecidos pelo nível de infra-estrutura (camada *IaaS*).

O modelo de terceirização adotado nesta proposta libera o provedor e o consumidor das tarefas de implementar o gerenciamento dos atributos de consumo e o Monitor de Referência $UCON_{ABC}$. Ambos os mecanismos são instanciados no Ambiente Federado. Neste contexto, existe sempre uma política de controle de uso disponível, independentemente de qual provedor o *Broker* designou para fornecer o serviço para o usuário. Módulos de aplicação de políticas (*PEP*) e Agentes de Monitoramento podem ser empacotados junto com o sistema desenvolvido pelo Domínio Consumidor.

5.1.3 Atributos

Os atributos pertencentes a camada *IaaS* se referem ao total de recursos utilizados pela máquina virtual ou instância do serviço (Figura 5.4; *evento rsv*). Estes atributos são referentes a quantidade alocada para o consumidor (*e.g. CPU-v*, *Memory-v*, *HD-v*; onde "*v*" representa o recurso fornecido pela máquina virtual). A camada *PaaS* fornece atributos de monitoramento relacionados aos usuários que estão consumindo os serviços (*e.g. Attr-User*; *evento ua*).

Os dados gerenciados pelo *PIP* (*Policy Information Point*) são fornecidos por Agentes de Monitoramento sendo executados nas máquinas virtuais instanciadas nos Provedores de Serviço (*PS*; *evento tp*). O *PIP* disponibiliza os atributos referentes a utilização individual de cada usuário para o *PIPC* (*PIP do Consumidor*; *evento st*). As informações de consumo armazenados no *PIP* também são fornecidas para o gerente de atributos do *Broker* (*Broker PIP - BPIP*; *evento ar*). O *Broker* consolida os dados de consumo referentes a todo Domínio Consumidor visando monitorar os *SLAs*.

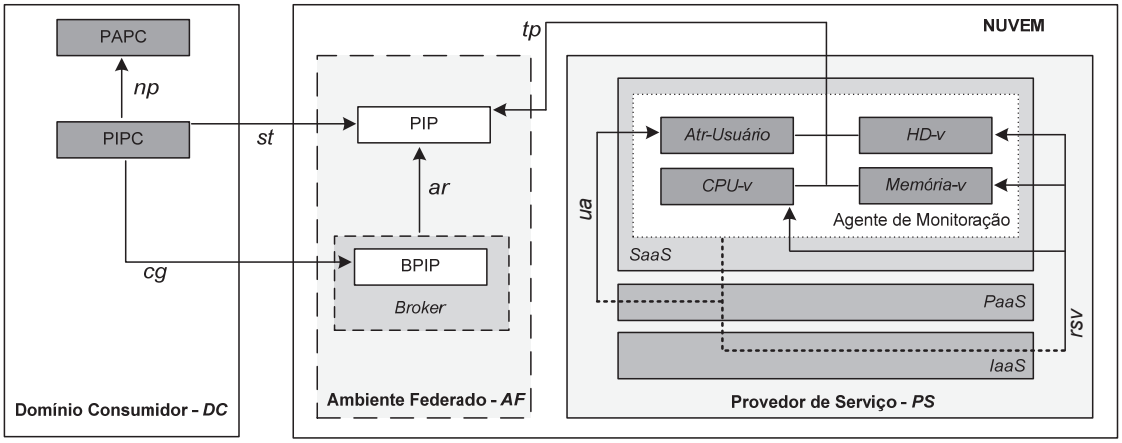

Figura 5.4: Gerenciamento de Atributos.

Considerando os atributos individuais (*evento st*) e o consumo consolidado (*evento cg*) é possível identificar qual usuário está utilizando os serviços no provedor (*evento ua*). Estas informações permitem descobrir se existem recursos ociosos no ambiente do provedor (*evento rsv*). Os atributos de consumo são fornecidos para o *PAPC* (*PAP do Consumidor*; *evento np*) para que este possa gerar novas políticas para os usuários do Domínio Consumidor (*DC*), justificar a necessidade de contratar mais serviços, ou otimizar a taxa de consumo dos recursos alocados no provedor.

De acordo com o tipo de serviço que o usuário vai utilizar no provedor (*e.g.* processamento), o *PIP* pode armazenar somente os atributos pertinentes aquele recurso (*e.g. CPU-v*). Neste esquema, o consumidor e o *Broker* monitoram somente os atributos referentes ao serviço que o usuário está consumindo. Baseado nas informações coletadas, é possível executar a reavaliação da cota de uso. Por consequência, a partir dos atributos de consumo consolidados, obtêm-se automaticamente um balanceamento entre as instâncias específicas de um serviço e a possível necessidade de reconfiguração das políticas de uso.

A coleta e o envio dos atributos de consumo, executadas pelo Agente de Monitoramento instanciado no Provedor de Serviço, podem acontecer com uma periodicidade bem definida. Porém, a utilização de uma frequência de monitoramento muito alta pode degradar o desempenho do serviço, tendo em vista que o Agente de Monitoramento vai utilizar os recursos do sistema de maneira mais intensiva. O emprego de uma frequência de monitoramento muito baixa reduz a carga de trabalho, porém, deixa o sistema vulnerável a perda dos picos de consumo. Neste contexto, é possível ao usuário violar uma regra de acesso entre os períodos de coleta de atributos e a respectiva reavaliação da política de uso. Encontrar o *tradeoff* para estas situações é um dos objetivos deste trabalho.

5.1.4 Mecanismos de Segurança

Para acessar o Provedor de Serviço (*PS*), cada usuário recebe uma credencial de acesso assinada pelo consumidor (Mecanismo de Segurança do Domínio Consumidor - *MSDC*; Figura 5.3; *evento au*). Esta credencial emitida pelo consumidor permite que o usuário seja identificado perante a federação. A credencial informa ao *Broker* qual é o domínio solicitante e a identificação do usuário (Stihler, 2012).

Os dados de identificação permitem ao *Broker* autenticar o usuário e fornecer a este a descrição da interface para acessar o serviço instanciado no provedor. A descrição da interface do serviço, além de informar ao usuário como acessar os recursos do provedor, também fornece informações sobre os privilégios de acesso necessários para utilizar o serviço. Estas informações são transportadas pela credencial de acesso emitida para o usuário.

Tendo em vista que a proposta foi implementada em um ambiente distribuído, o sistema de avaliação de políticas utiliza a criptografia na interação entre as entidades. A mensagem de avaliação de política enviada pelo Guardião do Serviço (*PEP*; *evento rpe*; Figura 5.5) para o espaço de *tuplas* é assinada com a chave do *PEP* e cifrada na chave de grupo. O processo criptográfico de cifragem utiliza uma chave de grupo compartilhada pelos *PDPs*, de acordo com Harney e Muckenhirn (Harney, 1997a) (Harney, 1997b).

Nesta abordagem, qualquer *PDP* do grupo pode processar um pedido de avaliação que chega ao serviço de espaço de *tuplas* (*eventos pr* e *epr*). A decisão de avaliação de política correspondente é cifrada na chave pública do *PEP*, assinada com chave de grupo,

e enviada para o espaço de *tuplas* correspondente (*evento spr*). Toda a instância de *PDP* que precisa acessar o serviço de espaço de *tuplas* para avaliar autorizações, deve ter a chave de grupo para abrir as solicitações enviadas pelo *PEP*. Adicionalmente, todo *PEP* que precisa solicitar avaliações de política deve ser afiliado ao grupo e compartilhar a chave com os *PDPs*. O gerenciamento das chaves e afiliações ao grupo é responsabilidade do *Broker*.

Os atributos de consumo enviados pelos Agentes de Monitoramento instanciados no provedor são assinados com a chave do agente e cifrados na chave pública do *PIP* da federação. Estes dados são enviados e armazenados no serviço de espaço de *tuplas* para atributos. Os procedimentos de assinatura e cifragem são de responsabilidade do serviço de credenciais do provedor (Mecanismo de Segurança do Provedor de Serviço - *MSPS*; Figura 5.2).

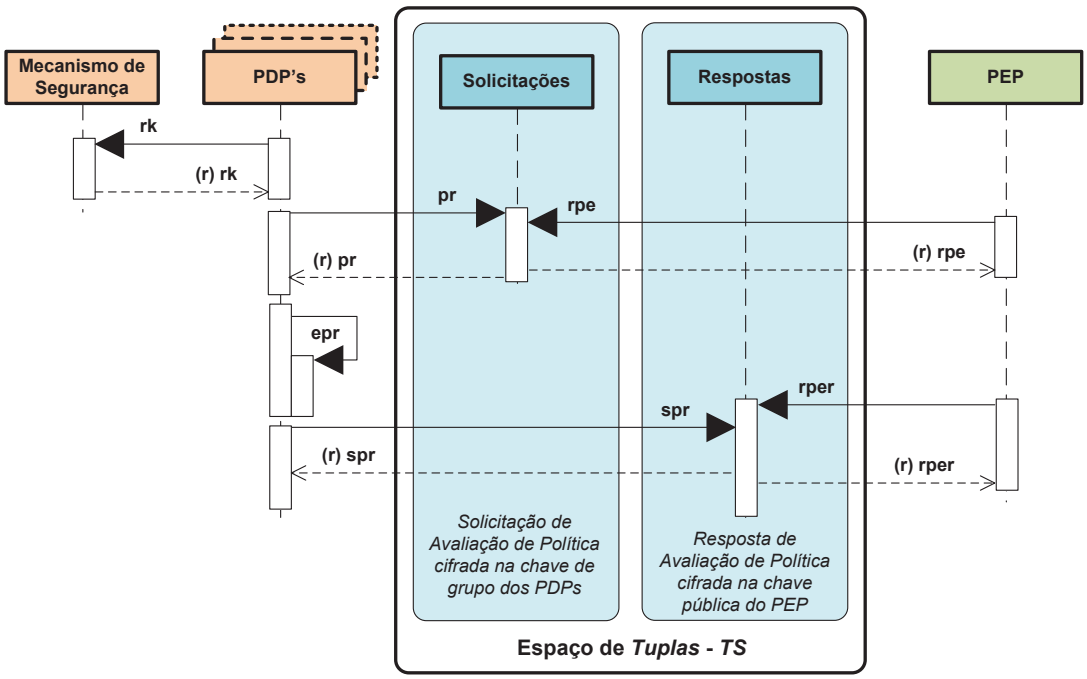

Figura 5.5: Segurança na avaliação de políticas baseada em espaço de *tuplas*.

Consumidor e provedor possuem um relacionamento confiável previamente estabelecido com o *Broker*. No ambiente do provedor, o usuário precisa ser autenticado pela infraestrutura de chave pública do serviço de credenciais (*MSPS*). A credencial fornecida pelo usuário (emitida pelo Domínio Consumidor) prova que este é parte da federação, tendo em vista que o consumidor e o provedor compartilham um relacionamento confiável com o *Broker* da federação.

5.1.5 Controle de Uso e Gerenciamento de Políticas

O *PAP* do Consumidor (*PAPC*) e o Mecanismo de Segurança do Domínio Consumidor (*MSDC*) transformam o *SLA* em regras e credencias de acesso (*i.e.* regras são a unidade básica para a criação de políticas; *evento re*; Figura 5.6) para os usuários do domínio (*evento au*). Utilizando estas credenciais, os usuários do consumidor acessam o Repositório de *Interfaces* (*RI*; *event is*) no *Broker* e o respectivo serviço *web* no provedor (*WS*; *evento ac*).

As políticas derivadas do *SLA* são armazenadas no repositório do consumidor (*PAPC*) e configuradas no *PAP* da federação (Figura 5.6; *evento en*). O *PAPC* é um serviço utilizado no gerenciamento das políticas de controle internamente ao Domínio Consumidor (*DC*). As políticas criadas e armazenadas localmente serão transferidas para o repositório do Ambiente Federado (*AF*) (*i.e.* o *PAP* recebe e armazena as políticas enviadas pelo *PAPC*). O *PDP* da federação vai utilizar as políticas armazenadas no *PAP* (*evento rp*) para avaliar as solicitações de autorização enviadas pelo *PEP* (*evento dc*) instanciado na máquina virtual do Provedor de Serviço (*PS*). As políticas criadas pelo consumidor e armazenadas no *PAP* também são utilizadas para configurar a cota do usuário.

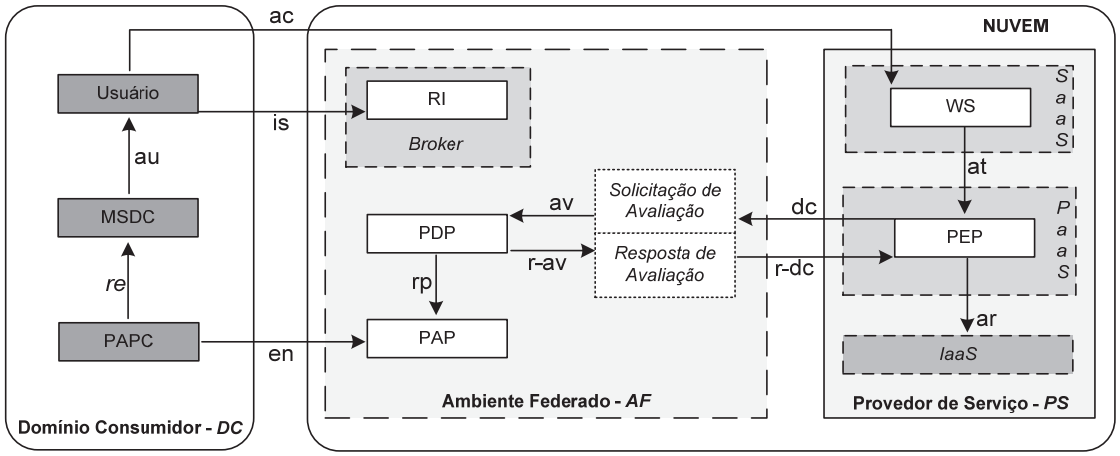

Figura 5.6: Gerenciamento de Políticas.

Como o controle de acesso é implementado utilizando o modelo de terceirização, o usuário deve ser autorizado pelo *PDP* e liberado pelo *PEP* (*evento at*) para poder consumir o serviço (*evento ac*). O processo de avaliação de autorização é executado nos serviços de gerenciamento da federação (*eventos dc* e *av*, respectivamente). As políticas

recuperadas do *PAP* (*evento rp*) são avaliadas pelo *PDP*. Quanto o resultado da avaliação é positivo (*eventos r-av* e *r-dc*, respectivamente), o usuário está autorizado a acessar o recurso disponível na camada *IaaS* (*evento ar*), sendo que esta decisão é honrada pelo *PEP*.

As solicitações de acesso recebidas pelo *PEP* (*evento at*) são encaminhadas para o serviço de espaço de *tuplas* da federação. Este espaço compartilhado é constantemente monitorado pelos *PDPs*. Com esta abordagem, qualquer servidor *PDP* (*i.e.* qualquer Monitor de Referência *UCON* pertencente ao *pool* de servidores *PDP*) recupera a solicitação de avaliação do serviço de espaço de *tuplas*, avalia a política e endereça a resposta ao *PEP* solicitante. O *PEP* que enviou a solicitação monitora o espaço de *tuplas* correspondente visando obter a resposta da avaliação de política. A avaliação de autorização utiliza os atributos de consumo disponíveis no repositório (*PIP*) da federação.

De acordo com a necessidade do ambiente (*i.e.* conforme este se expandir), mais *PDPs* podem ser instanciados para avaliar políticas referentes aos serviços do provedor e usuários dos consumidores. Esta característica provê elasticidade aos ambientes de avaliação e gerenciamento de atributos. Esta estratégia é a mesma utilizada para expandir ou contrair o número de *PIPs*.

Nesta proposta, políticas criadas pelo consumidor são armazenadas no *PAP* e utilizadas para configurar a cota de uso para o usuário. A cota é utilizada durante as reavaliações, em conjunto com as políticas de uso. O objetivo desta abordagem é relaxar as regras individuais da política quanto possível. Isto significa que a cota permite ao usuário extrapolar sua política de uso quando a condição de soma de consumo para os usuários do consumidor (*sua*) não alcançar o limite estabelecido no *SLA* vigente.

A abordagem proposta permite a reavaliação contínua (*i.e. ongoing* - durante o consumo do serviço) e a reconfiguração das políticas de uso para o usuário de acordo com a necessidade e enquanto possível (*i.e.* enquanto o limite definido no *SLA* não for alcançado). Adicionalmente, a flexibilidade fornecida pelo esquema de avaliação, suportando vários *PDPs* trabalhando em paralelo, fornece suporte para um ambiente de controle de uso elástico.

5.2 *Conclusão*

Os controles de segurança na computação em nuvem apresentam diferentes riscos

para a organização consumidora quando comparado com as abordagens tradicionais. O consumidor que necessitar de uma grande quantidade de recursos computacionais, pode precisar de vários provedores de nuvem diferentes para satisfazer sua demanda.

O *pool* de serviços fornecido é bastante heterogêneo, tornando a tarefa de utilização complexa, envolvendo o gerenciamento de diferentes contratos, políticas, interfaces, atributos, usuários etc. Na computação em nuvem o consumidor abre mão de alguns controles (físico, por exemplo) enquanto mantêm as responsabilidades sobre o gerenciamento operacional do ambiente.

A abordagem proposta se adéqua ao nível de acesso permitido atualmente a infraestrutura (*IaaS*) não exigindo mudanças neste ambiente, pois a gestão é feita por serviços que estão sob o controle do consumidor ou *Broker*. Isto significa que, sem a utilização do esquema proposto, o consumidor não poderia efetuar um controle de uso com granularidade fina para a nuvem, pois nenhum provedor de *IaaS* oferece serviço similar.

Capítulo 6

Protótipo e Testes de Avaliação da Proposta

6.1 *Introdução*

As seções descritas abaixo descrevem as principais tecnologias utilizadas no desenvolvimento do protótipo, o esquema de avaliação e o resultado de testes executados com a implementação do cenário proposto.

6.2 *Tecnologias*

O protótipo referente a proposta (Figura 6.1) foi implementado utilizando um sistema operacional de nuvem, Serviços *Web* e o espaço de *tuplas*.

- O sistema operacional de nuvem utilizado pelo ambiente foi o *VMWare vSphere 4.1* (*www.vmware.com/products/vsphere/overview.html*).
- O acesso aos atributos do ambiente foi obtido com o suporte das seguintes bibliotecas de código: *VMware Infrastructure Java API* (*http://vijava.sourceforge.net*); *APIs* para acessar a *Java Virtual Machine* fornecidas pelo *Java Development Kit* (*java.lang.management*); os projetos *SIGAR* (*hyperic.com/support/docs/sigar*) e *JavaSysMon* (*jezhumble.github.com/javasysmon*).
- Os serviços fornecidos pelas *VMs* (*Virtual Machine*) estão hospedados no servidor de aplicações *Apache TomCat* (*tomcat.apache.org*).
- O acesso aos serviços é efetuado através da *engine SOAP Axis2* (*axis.apache.org/axis2/java/core*).
- O módulo *Rampart* (*axis.apache.org/axis2/java/rampart*) integrado ao *AXIS2*

fornece a segurança (*i.e.* assinatura e cifragem) necessária para as mensagens *SOAP* (especificações *WS-Security* (OASIS, 2004) e *WS-Trust* (OASIS, 2008)).

- O transporte e o provisionamento das políticas utilizam as bibliotecas dos projetos *Open SPML* (*java.net/projects/OpenSPML*) e *SAML* (*www.opensaml.org*).
- Políticas de uso foram criadas e manipuladas utilizando a implementação da *XACML* (*eXtensible Access Control Markup Language*) (OASIS, 2005b) fornecida pela *Sun XACML API* (*sunxacml.sourceforge.net*).
- O espaço de *tuplas*, implementação do espaço de memória compartilhada no Ambiente Federado, utilizou a tecnologia fornecida pelo *JavaSpaces*. Esta *API* foi desenvolvida e implementada pelo projeto *River* (*river.apache.org*).

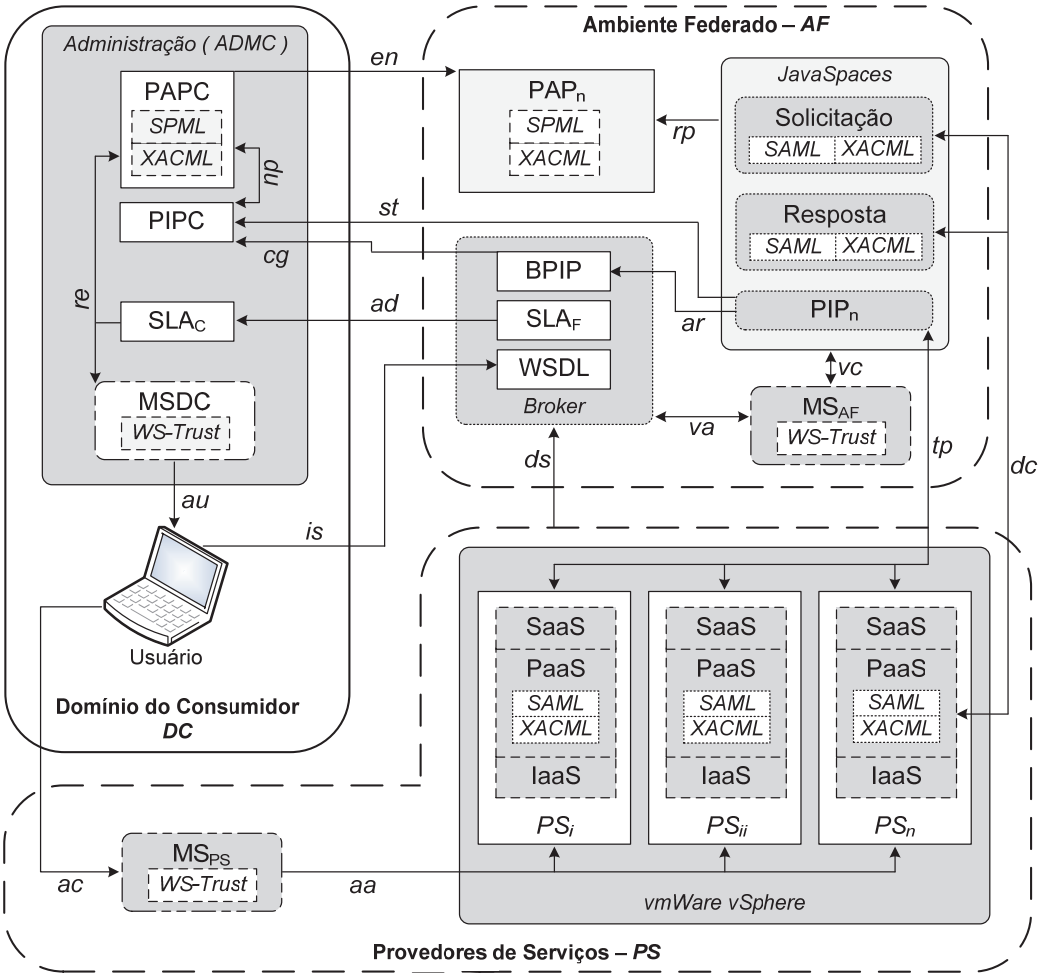

Figura 6.1: Protótipo.

Os testes foram executados em um ambiente que implementa a proposta (Figura 6.1), de acordo com a seguinte especificação: o Domínio Consumidor está sendo executado em *Intel Core i7 2Ghz*, *6GB* de *RAM* e *Windows 7 x64* como sistema

operacional; as instâncias dos Provedores de Serviço utilizam um *Intel Xeon 2.6Ghz*, processador *dual* de 6 núcleos, *48GB* de *RAM* e *3TB* de armazenamento; o servidor do Ambiente Federado é um *Intel 2.0Ghz*, processador *dual* de 4 núcleos, *16GB* de RAM e *3.3TB* de armazenamento. Os servidores possuem a instalação do *VMware vSphere ESX 4.1*, hospedando as máquinas virtuais e os serviços descritos na proposta. As máquinas estão conectadas através de uma rede *Gigabit Ethernet*.

6.3 *Avaliação do Protótipo*

O serviço oferecido para o usuário neste cenário de testes não é um sistema de *e-commerce* real. Assim, as cargas de trabalho para a *CPU* foram simuladas utilizando algoritmos criptográficos com diferentes tamanhos de chaves. Todas as cargas foram ajustadas para criar diferentes ambientes de simulação (*e.g.* carga < *10%*, *40%* < carga < *60%*, carga > *90%*). Neste contexto, cada Agente de Monitoramento enviou *17* atributos de consumo diferentes para o serviço de espaço de *tuplas*.

As principais informações enviadas para o espaço de *tuplas* são: endereço de *IP* (*Internet Protocol*) e o nome da máquina virtual que hospeda o serviço *web* disponibilizado para os usuários; quantidade de memória *heap* gerenciada pela máquina virtual *Java* (*i.e.* valores: inicial, utilizado, reservado e o máximo disponibilizado); identificação do Agente de Monitoramento e da *thread* responsável por executar o serviço solicitado pelo usuário (*i.e.* nome e *id* da *thread*); tempo gasto pelo agente para monitorar os atributos do sistema; tempo gasto pela *thread* de serviço para atender uma solicitação do usuário (*user time*) e para o sistema responder a solicitação (*overhead time*); informações relativas a máquina virtual que hospeda o serviço *web* utilizado nos testes (*e.g.* taxas de utilização referentes ao processador, memória e rede).

As medidas (Figuras 6.2, 6.3 e 6.4) foram obtidas executando-se *1000* interações e calculando a média entre estas para cada item apresentado nos gráficos. Procedimento similar foi utilizado para as Tabelas 1 e 2. O coeficiente de variabilidade foi abaixo de *5%* em todos os casos. Os testes foram executados em um ambiente controlado de rede local, com isto foi possível conhecer o comportamento do sistema sem interferências externas. O tempo para recuperar os atributos da máquina virtual *Java* está representado em nano segundos. Nas medidas realizadas, testou-se vários tempos de monitoramento (*i.e.* a partir de zero milissegundos até dois segundos), visando encontrar o melhor valor entre a carga

de trabalho gerada pelo sistema e o tempo para fornecer o serviço *web* de *e-commerce*.

O eixo "*x*" representa o tempo de espera (*i.e.* periodicidade) da tarefa de monitoramento (*i.e.* Agente de Monitoramento implementado por uma *thread Java*): "*S0*" significa que a tarefa de monitoramento não está sujeita a espera induzida (*sleep time*), processando continuamente (*i.e.* de acordo com o esquema de prioridade da máquina virtual *Java*); o "*x*" em "*Sx*" representa o tempo, em milissegundos, que a tarefa esperou entre os monitoramentos consecutivos (*e.g.* em "*S2*", a tarefa esperou "*2ms*"; para "*S2048*", a tarefa esperou por aproximadamente "*2s*"). Todas as tarefas e serviços foram executados com o mesmo nível de prioridade. Esta abordagem visa evitar a disparidade nas medidas devido ao esquema de escalonamento, baseado em prioridade, adotado pelo algoritmo implementado pela máquina virtual *Java* (*JVM*).

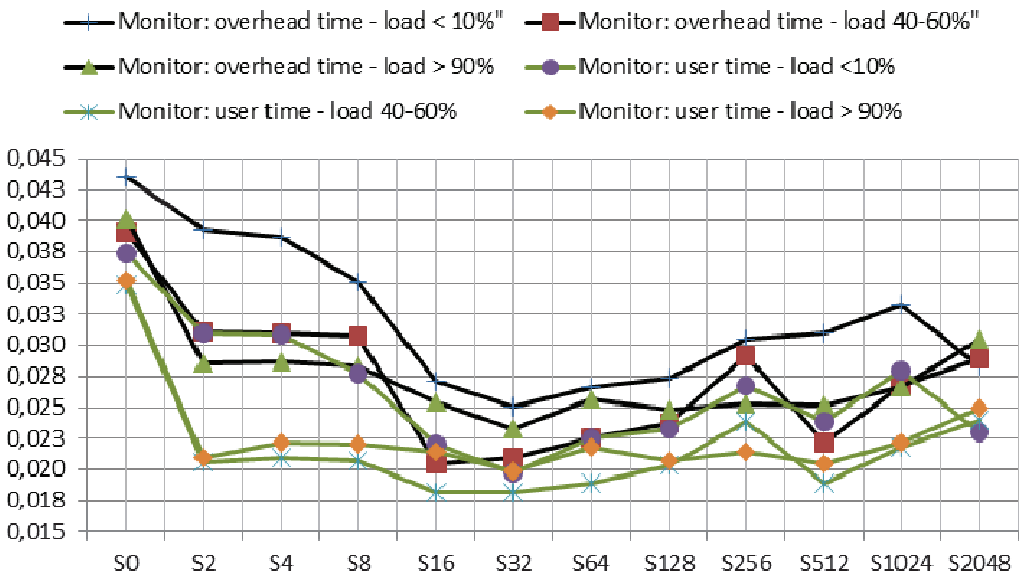

Figura 6.2: Melhor periodicidade para o monitoramento de atributos.

O tempo da *CPU* foi obtido utilizando a classe *Java ThreadMXBean*. A carga da *CPU* na máquina virtual foi medida com o auxílio das *APIs SIGAR* e *Java SysMon*. As medidas foram executadas para a tarefa de monitoramento (Figura 6.2) e para a tarefa responsável por executar o serviço disponibilizado para o usuário (Figuras 6.3 e 6.4). As medidas apresentadas nas figuras se referem a um cenário de teste com diferentes cargas de trabalho: o tempo de CPU gasto para oferecer o serviço para o usuário da nuvem (*user time*); e o tempo gasto pelo sistema para suportar o serviço sendo fornecido para o usuário

(*overhead time*).

A Figura 6.2 mostra que uma frequência de monitoramento com intervalos muito curtos pode degradar o desempenho do serviço (*i.e.* entre *S0* e *S8*, em média, o tempo de sistema é alto - *overhead time*) porque o agente monitor vai utilizar os recursos do sistema de maneira mais intensiva.

Na Figura 6.2 é possível observar que os experimentos mostram a melhor periodicidade de monitoramento de serviço para o cenário desenvolvido. Neste caso, podemos perceber que em torno de "*S32*" (periodicidade de monitoramento de *32ms*) é obtido o melhor valor entre carga de trabalho e tempo de processamento (*user time*) considerando três cargas de *CPU* para o serviço de *e-commerce*.

Um dos objetivos deste trabalho foi encontrar a melhor periodicidade de monitoramento. Assim, o período aproximado de *32ms* define a quantidade máxima de tempo que um determinado usuário pode estar em uma condição de exceção (*i.e.* que neste caso considerou o monitoramento de um recurso altamente dinâmico como a *CPU*). Consequentemente, com a utilização de valores próximos a esta periodicidade, obtêm-se a melhor frequência de reavaliação de políticas.

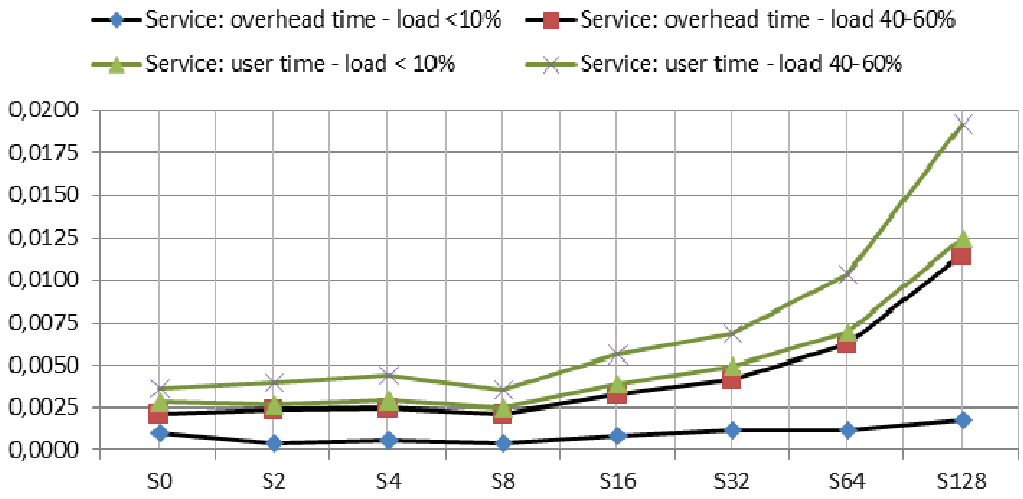

Figura 6.3: Carga de trabalho para o serviço e usuário - carga entre *40% e 60%*.

As Figuras 6.3 e 6.4 mostram os valores entre a carga de trabalho e o tempo de processamento (*user time*) para o caso de um serviço *web* de *e-commerce*. Os gráficos foram divididos para facilitar a visualização, considerando a escala de uso mostrada em cada caso: Figura 6.3 entre *zero* nano segundos e *200* nano segundos; Figura 6.4 entre *50*

nano segundos e *2300* nano segundos.

Pode-se perceber que o intervalo de monitoramento de *32ms* ("*S32*"; Figuras 6.2 & 6.3) confirma o melhor valor. Acima deste tempo, a carga de trabalho e o tempo de processamento mostram um crescimento exponencial, situação mostrada na Figura 6.4. Para facilitar a visualização desta informação, somente os tempos entre *0ms* (*zero* milissegundos) e *128ms* foram mostrados. Valores acima destes aumentam o tempo exponencialmente, tornando difícil a visualização dos detalhes mostrados nos gráficos.

Alguns testes de *stress* foram executados no serviço de espaço de *tuplas* (Tabela 6.1) visando identificar o número de entradas que o espaço consegue armazenar consecutivamente. Pode-se perceber que, na média, enquanto o tamanho da *tupla* (atributos de consumo) dobra de valor, o número de entradas armazenadas reduz pela metade. Adicionalmente, o tempo gasto para armazenar cada tipo de *tupla* é alterado significativamente, resultando no melhor rendimento para entradas com tamanhos entre *4KB* e *16KB*. No caso dos experimentos mostrados na Figura 6.2, o Agente de Monitoramento enviou *tuplas* com aproximadamente *2KB* (*i.e.* transportando 17 tipos de atributos de consumo).

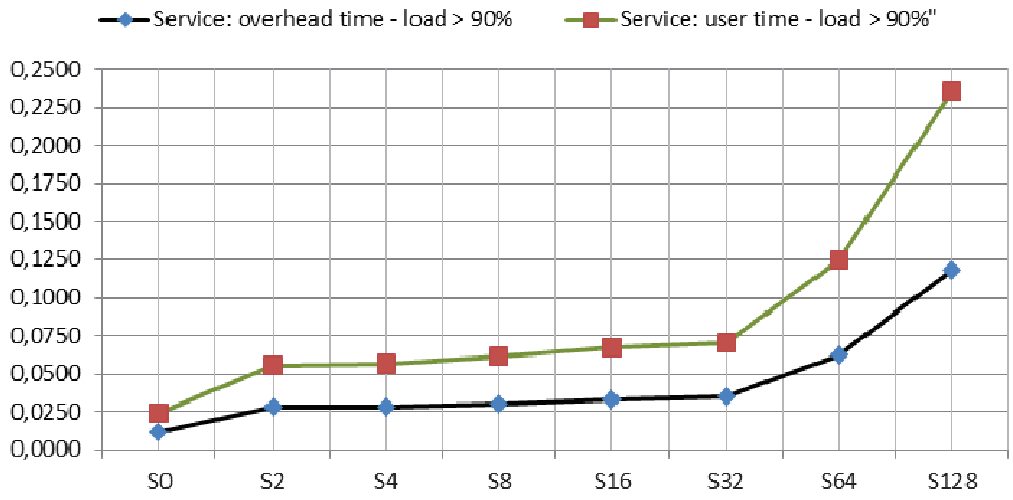

Figura 6.4: Carga de trabalho para o serviço e usuário - carga maior que *90%*.

Um teste similar foi executado no *PDP* para o processo de reavaliação de políticas (Tabela 6.2). Porém, neste caso, o *PDP* recupera a solicitação de avaliação de política do serviço de espaço de *tuplas*, com aproximadamente *2KB* (considerando que o cabeçalho da *tupla* possui aproximadamente *1KB*), e a política do respectivo repositório (*PAP*). Uma

política *XACML* simples, contendo apenas uma regra de autorização, tem aproximadamente *4KB*.

Na sequência do processo, o *PDP* avalia a solicitação e envia o resultado para o serviço de espaço de *tuplas* correspondente. A *tupla* de resposta possui um tamanho total de *2KB*, já estando incluso o cabeçalho de aproximadamente *1KB*. Este experimento mostrou a capacidade do *PDP* em atender as solicitações de reavaliação de política (Tabela 6.2; *Número de políticas reavaliadas paralelamente antes do serviço parar de responder*) e o tempo total gasto para executar o processo de avaliação (Tabela 6.2; *Tempo gasto para avaliar uma política*).

De maneira semelhante, pode-se considerar o tempo gasto pelo *PIP* para recuperar os atributos do serviço de espaço de *tuplas* e armazenar estes localmente como sendo análogo ao tempo gasto pelo *PDP* para avaliar uma política (Tabela 6.2). Adicionalmente, o tempo gasto pelo *PEP* para escrever uma solicitação de avaliação no serviço de espaço de *tuplas* é similar ao tempo gasto pelos agentes de monitoramento para escrever um atributo de consumo. Portanto, considerando-se o tempo para executar o envio de uma solicitação de reavaliação de política para o serviço de espaço de *tuplas*, acrescido do tempo necessário para o *PDP* avaliar e responder a solicitação pode-se concluir que, para várias combinações de tamanho de solicitação e tamanho de política, o tempo total gasto para completar o processo de reavaliação permanece abaixo dos *32ms* (Figura 6.2).

Tabela 6.1: Teste de *stress*: espaço de tuplas.

Tamanho da tupla (incluindo 1KB de cabeçalho)	***Número de tuplas armazenadas antes do serviço recusar conexões***	***Tempo gasto para armazenar uma tupla (ms)***	***Throughput (KB/ms)***
2 KB	69864	2,54	55011
4 KB	45333	2,57	70557
8 KB	26651	3,01	70833
16 KB	14605	3,40	68729
32 KB	7670	4,94	49684
64 KB	3935	7,90	31878
128 KB	1492	13,44	14209
256 KB	1002	25,39	10102
512 KB	502	48,84	5262
1024 KB	250	98,88	2589
2048 KB	124	200,83	1264

Por inferência, o tempo real necessário para se obter um atributo de consumo, armazenar este no *PIP* e efetivamente usar está informação no processo de reavaliação de

política necessita de dois períodos de *32ms*. Um dos períodos para disponibilizar os atributos de consumo no Ambiente Federado e outro período para usar estes atributos no processo de reavaliação das políticas. Assim sendo, reconsiderando a análise feita anteriormente, *64ms* pode ser considerado como o melhor período de reavaliação de um acesso.

As medidas apresentadas nas Tabelas 1 e 2 fornecem uma idéia a respeito da elasticidade do esquema proposto. A avaliação mostra que o número de servidores pertencentes ao *pool* deveria ser incrementado quando a demanda estiver próxima de provocar uma parada nos serviços (*i.e.* quando os serviços param de atender as solicitações). O serviço de espaço de *tuplas* juntamente com o esquema de avaliação distribuído (*i.e.* vários *PDPs* instanciados sob demanda) fornecem elasticidade para o $UCON_{ABC}$. O espaço para o gerenciamento de atributos, responsabilidade do *PIP*, segue a mesma abordagem, provendo elasticidade ao sistema de contabilização de atributos.

Tabela 6.2: Teste de *stress*: reavaliação de políticas no *PDP*.

Tamanho da política recuperada do PAP	***Número de políticas reavaliadas paralelamente antes do serviço parar de responder***	***Tempo gasto para avaliar uma política (ms)***
4 KB	1690	0,129916
8 KB	1360	0,129123
16 KB	1080	0,128985
32 KB	640	0,130264
64 KB	470	0,131786
128 KB	310	0,133511
256 KB	220	0,144560
512 KB	130	0,144513
1024 KB	90	0,161233
2048 KB	70	0,186473

6.4 *Conclusão*

Os testes iniciais mostraram que é possível efetuar o gerenciamento e a consolidação dos atributos de serviços e usuários através da utilização de padrões abertos, como é o caso dos Serviços *Web*.

A gerência de usuários que acessam domínios administrativos distribuídos, utilizando diferentes tipos de serviços, gera um volume considerável de dados que precisam ser consolidados e avaliados. A consolidação permite que o Domínio

Consumidor tenha uma visão global do ambiente. Esta abordagem possibilita ao contratante alocar melhor a sua cota de uso entre os usuários, otimizando a utilização dos serviços alocados.

Capítulo 7

Considerações Finais

7.1 *Conclusão*

O trabalho apresentado é uma abordagem inovadora para a reavaliação contínua da autorização (*ongoing*) durante o consumo do serviço ou recurso. O monitoramento constante de serviços e a reavaliação dos atributos de autorização permite que sejam identificadas disparidades entre autorizações escritas nas políticas e os atributos de consumo vigentes. Adicionalmente, a proposta provê resiliência (*i.e.* relaxando as regras da política) para os atributos de autorização em algumas circunstâncias, sem qualquer perda para o consumidor (*e.g.* violação de *SLA*).

Quando uma disparidade é identificada e o esquema de resiliência não pode ser aplicado, o usuário estará em uma condição de exceção. Neste caso, o consumidor possui algumas alternativas para reparar a situação se comparado com as abordagens tradicionais apresentadas pela literatura.

A monitoração contínua de serviços produz uma grande quantidade de atributos que precisam ser consolidados e avaliados. Assim, a adoção de uma abordagem elástica, implementada por um *pool* de servidores que hospedam o serviço de espaço de *tuplas*, juntamente com um esquema de avaliação distribuído (*i.e.* vários *PDPs* instanciados de acordo com a demanda) fornece elasticidade ao $UCON_{ABC}$ durante a reavaliação das políticas de uso.

O espaço de gerenciamento de atributos, sob a responsabilidade do *PIP*, segue a mesma abordagem, provendo elasticidade ao sistema de monitoramento utilizado pelo $UCON_{ABC}$. A proposta para o serviço de monitoramento, juntamente com a reavaliação contínua para cada usuário, fornece um esquema de controle de acesso e contabilização de atributos com fina granularidade para o ambiente da computação em nuvem.

O esquema de resiliência utilizado na proposta torna os atributos de autorização (*i.e.* cotas) definidos para cada usuário do consumidor mais flexíveis. Adicionalmente, violações em políticas de controle são monitoradas e tratadas pelo ambiente de gerenciamento da federação (*SLA*) e em nível de Domínio Consumidor (condições de exceção). Este esquema permite o consumo flexível dos recursos computacionais, ajustando as cotas de maneira semi-automática, sem desperdício, ociosidade ou abuso do serviço contratado.

A abordagem proposta mostrou que é possível executar o gerenciamento e a consolidação de atributos utilizando o baixo-acoplamento e padrões abertos (*e.g.* serviços *web*). Adicionalmente, o esquema é adequado para o nível de acesso permitido atualmente em nível de infraestrutura (*IaaS*), não necessitando de mudanças neste ambiente.

O gerenciamento é executado por serviços que trabalham de acordo com a demanda do consumidor. Isto significa que, sem o auxílio da proposta, o consumidor não poderia controlar o uso dos serviços e recursos no provedor de nuvem, tendo em vista que nenhum fornecedor de *IaaS* oferece um serviço similar. Por fim, durante o período de desenvolvimento deste trabalho, não foi encontrada nenhuma proposta similar a esta na literatura técnica, nem mesmo para os níveis *PaaS* ou *SaaS*.

Os principais resultados alcançados no doutoramento podem ser brevemente elencados como:

- mini-curso: "*Aspectos de Segurança e Privacidade em Ambientes de Computação em Nuvem*", publicado nos anais do Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais (*SBSeg 2010*);
- artigo: "*Integral Federated Identity Management for Cloud Computing*", publicado no *X IFIP International Conference on New Technologies, Mobility and Security* (*NTMS 2012*);
- artigo: "*Uma arquitetura para auditoria de nível de serviço para computação em nuvem*", publicado nos anais do Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais (*SBSeg 2012*);
- artigo: "*A $UCON_{ABC}$ Resilient Authorization Evaluation for Cloud Computing*" publicado em "*IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS (TPDS 2013)*".
- artigo: "*Avaliação Resiliente de Autorização $UCON_{ABC}$ para Computação*

em Nuvem", a ser publicado nos anais do Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais (*SBSeg 2013*);

- modelo de autorização resiliente para o $UCON_{ABC}$;
- protótipo que implementa o cenário de avaliação.

7.2 *Trabalhos Futuros*

Para os trabalhos futuros, percebe-se a necessidade do desenvolvimento de testes adicionais para a proposta, visando encontrar a relação ótima (periodicidade) para o monitoramento e reavaliação de políticas em diferentes cenários de aplicação.

Adicionalmente, a proposta precisa de testes com cargas de trabalho reais, em um modelo de implantação híbrido (*i.e.* público e privado), visando representar de maneira mais fiel as abordagens adotadas por organizações que trabalham com nuvens computacionais.

Referências

Abawajy J., (2009). *Determining Service Trustworthiness in Intercloud Computing Environments*, *10th International Symposium on Pervasive Systems, Algorithms, and Networks* (I-SPAN 2009), Kaohsiung, Taiwan.

Amit N., Sanjay C. e Gaurav S., (2011). *Policy based resource allocation in IaaS cloud. Future Generation Computer Systems* (FGCS). pg. 94-103.

Armbrust M., Fox A., Griffith R., Joseph A. D., Katz R., Konwinski A., Lee G., Patterson D., Rabkin A., Stoica I., e Zaharia M., (2010). *A view of cloud computing, Commun. ACM*, vol. 53, no. 4, pp. 50-58.

Bachtold J., Santin A. O., Stihler M., Marcon Jr. A. L. e Viegas E., (2012). Uma arquitetura para auditoria de nível de serviço para computação em nuvem. Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais (SBSeg 2012). Curitiba - PR, Brasil.

Bertram S., Boniface M., Surridge M., Briscombe N., Hall-May M., (2010). *On-Demand Dynamic Security for Risk-Based Secure Collaboration in Clouds. Proceedings of the IEEE 3rd International Conference on Cloud Computing* (CLOUD 2010), Southampton, UK.

Bhattacharjee R., (2009). *An Analysis of the Cloud Computing Platform, System Design and Management Program. Massachusetts Institute Of Technology* – MIT. Dissertação de mestrado.

Booth D., Haas H., McCabe F., Newcomer E., Champion M., Ferris C. e Orchard D., (2004). *Web Services Architecture*, *World Wide Web Consortium* (W3C).

Capizzi S. e Messina A., (2008). *A Tuple Space Service for Large Scale Infrastructures. Proceedings of the IEEE 17th Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises* (WETICE 2008). Rome, pg. 182-187.

Chunlin L. e Layuan L., (2003). *An Agent-Based Approach For Grid Computing. Proceedings of the Fourth International Conference on Parallel and Distributed Computing, Applications and Technologies* (PDCAT 2003). Chengdu, pg. 608-611.

CSA. (2010). *Domain 12: Guidance for Identity & Access Management V2.1.* Acesso: Maio 2010. Disponível em: *http://www.cloudsecurityalliance.org/guidance/csaguide-dom12-v2.10.pdf*.

CSA. (2011). *Security Guidance for Critical Areas of Focus in Cloud Computing v3.0*. Acesso: Março 2012. Disponível em: *cloudsecurityalliance.org/guidance/csaguide. v3.0.pdf*.

Danwei C., Xiuli H. e Xunyi R., (2009). *Access Control of Cloud Service Based on UCON*. *Proceedings of the 1st International Conference on Cloud Computing* (CloudCom 2009). *Lecture Notes in Computer Science* (LNCS). Springer-Verlag Berlin, Heidelberg. pg. 559-564.

Douglis F., (2009). *Staring at clouds*. *Internet Computing*, *IEEE*, vol. 13, no. 3.

Erickson J. S., Spence S., Rhodes M., Banks D., Rutherford J., Simpson E., Belrose G., e Perry R., (2009). *Content-Centered Collaboration Spaces in the Cloud*. *IEEE Internet Computing*, pp. 34-42, October.

Foster I., Zhao Y., Raicu I., e Lu S., (2008). *Cloud computing and grid computing 360-degree compared*. *Proceedings of the Grid Computing Environments Workshop*. (GCE 2008).

Gelernter D., (1985). *Generative Communication in Linda*. *Proceedings of the ACM Transactions on Programming Languages and Systems* (TOPLAS 1985). Vol. 7, No. 1. New York, NY. pg. 80-112.

Goiri Í., Julií F., Fitó J. O., Macías M. e Guitart J., (2011). *Supporting CPU-Based Guarantees in Cloud SLAs Via Resource-Level QoS Metrics*. *Future Generation Computer Systems* (FGCS). Volume 28. pg. 1295-1302.

Goyal P., e Mikkilineni R., (2009). *Policy-based Event-driven Services-oriented Architecture for Cloud Services Operation & Management*. *Proceedings of the IEEE International Conference on Cloud Computing* (CLOUD 2009). Bangalore, India.

Han H., Kim S., Jung H., Yeom H. Y., Yoon C., Park J., Lee Y., (2009). *A RESTful Approach to the Management of Cloud Infrastructure*. *Proceedings of the IEEE International Conference on Cloud Computing* (CLOUD 2009), Bangalore, India.

Harney H. e Muckenhirn C., (1997a). *Group Key Management Protocol* (GKMP) *Specification*. *Internet Engineering Task Force* (IETF), *Network Working Group*, *Request for Comments* 2093, Categoria Experimental, pg. 23.

Harney H. e Muckenhirn C., (1997b). *Group Key Management Protocol* (GKMP) *Architecture*. *Internet Engineering Task Force* (IETF), *Network Working Group*, *Request for Comments* 2094, Categoria Experimental, pg. 22.

Hayes B., (2008). *Cloud computing*. *Commun. ACM*, vol. 51, no. 7, pp. 9-11, July.

Huang H. e Wang L., (2010). *P&P - A Combined Push-Pull Model for Resource Monitoring in Cloud Computing Environment*. *Proceedings of the IEEE 3rd International Conference on Cloud Computing* (CLOUD, 2010). Miami, Florida. pg. 260-267.

Hwang K., Kulkareni S. e Hu Y., (2009). *Cloud Security with Virtualized Defense and Reputation-based Trust Management*. *Proceedings of the Eighth IEEE International Conference on Dependable, Autonomic and Secure Computing* (DASC 2009). Chengdu. pg. 717-722.

Hwang K., Kulkareni S., e Hu Y., (2009). *Cloud Security with Virtualized Defense and Reputation-Based Trust Mangement*. *Proceedings of the Eighth IEEE International Conference on Dependable, Autonomic and Secure Computing* (DASC 2009), Chengdu, China.

IBM, (2003). *Web Service Level Agreement* (WSLA) *Language Specification*. Acesso Maio 2010. *Disponível em http://www.research.ibm.com/wsla/WSLASpecV1-20030128.pdf.*

ISO - *International Organization for Standardization*, (2006). *Information technology - Open Systems Interconnection - Security frameworks for open systems: Access control framework*. ISO/IEC 10181-3:1996. pg. 36.

Jamkhedkar P. A., Lamb C. C. e Heileman G. L., (2011). *Usage Management in Cloud Computing*. *Proceedings of the IEEE International Conference on Cloud Computing* (CLOUD 2011), Washington, DC, pg. 525-532.

Kandukuri B. R., Paturi V. R., e Rakshit A., (2009). *Cloud Security Issues. Proceedings of the International Conference on Services Computing* (SCC 2009), Bangalore, India.

Lamb C. C., Jamkhedkar P. A., Heileman G. L. e Abdallah C. T. (2011). *Managed control of composite cloud systems*. *Proceedings of the 6th International Conference on System of Systems Engineering* (SoSE, 2011). Albuquerque, NM. pg. 167-172.

Lee J. e Taylor S., (2001). *Advances in Computational Resiliency*. *Proceedings of the IEEE Aerospace Conference* (AERO 2001). Big Sky, MT. pg. 2829-2835.

Li N., Tripunitara M. V. e Wang Q., (2006). *Resiliency Policies in Access Control*. *Proceedings of the 3th ACM Conference on Computer and Communications Security* (CCS 2006). Alexandria, VA. pg. 113-123.

Lim H. C., Babu S., Chase J. S. e Parekh S. S. (2009). *Automated Control in Cloud Computing: Challenges and Opportunities*. *Proceedings of the 1st workshop on Automated control for datacenters and clouds* (ACDC). Barcelona, Spain.

Marcon Jr. A. L. , Santin A. O., Lima Jr. L. A. de P. e Stihler M., (2009a). *Policy Management Architecture Based on Provisioning Model and Authorization Certificates*. *Proceedings of the 24th Annual ACM Symposium on Applied Computing* (SAC 2009). Waikiki Beach, Honolulu, Hawaii, USA.

Marcon Jr. A. L. , Santin A. O., Lima Jr. L. A. de P., Obelheiro R. R. e Stihler M., (2009b). *Policy control management for Web Services*. *Proceedings of the 11th IFIP/IEEE International Symposium on Integrated Network Management* (IM 2009). Long Island, New York, USA.

Mell P., e Grance T., (2009). *The NIST Definition of Cloud Computing*. *National Institute of Standards and Technology, Information Technology Laboratory*.

Meng S. e Liu L., (2012). *Enhanced Monitoring-as-a-Service for Effective Cloud Management*. *IEEE Transactions on Computers*. *IEEE Computer Society Digital Library*. *IEEE Computer Society*. pg. 1-14.

OASIS, (2004). *Web Services Security SOAP Message Security 1.1* (*WS-Security*). Acesso Maio 2010. Disponível em *http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss*.

OASIS, (2005a). *Assertions and Protocols for the OASIS Security Assertion Markup Language* (SAML) *V2.0*. Acesso: Maio 2010. Disponível em: *http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf*.

OASIS, (2005b). *eXtensible Access Control Markup Language* (XACML) *Version 2.0*. Acesso: Set. 2010. Disponível em: *www.oasis-open.org/committees/xacml*.

OASIS, (2005c). *SAML 2.0 profile of XACML v2.0*. Acesso: Set. 2010. Disponível em: *http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-saml-profile-spec-os.pdf*.

OASIS, (2006a). *Reference Model for Service Oriented Architecture v 1.0*. Acesso Fev. 2010. Disponível em *http://www.oasis-open.org/specs/index.php#soa-rmv1.0*.

OASIS, (2006b). *Service Provisioning Markup Language* (SPML) *Version 2*. Acesso: Jun. 2010. Disponível em: *www.oasis-open.org/committees/provision*.

OASIS, (2008). *WS-Trust 1.4*. Acesso: Mar. 2013. Disponível em: *http://docs.oasis-open.org/ws-sx/ws-trust/v1.4/*.

Park J. e Sandhu R., (2002). *Towards Usage Control Models: Beyond Traditional Access Control*. *Proceedings of the Seventh ACM Symposium on Access Control Models and Technologies* (SACMAT 2002), Monterey, California, pg. 57-64.

Park J. e Sandhu R., (2004). *The $UCON_{ABC}$ Usage Control Model*. *Proceedings of ACM Transactions on Information and System Security (TISSEC)*, vol. 7, no. 1, pp. 128-174, New York - NY, USA.

Picco G. P., Balzarotti D. e Costa P., (2005). *LighTS: A Lightweight, Customizable Tuple Space Supporting Context-Aware Applications*. *Proceedings of the 20th ACM Symposium on Applied Computing* (SAC 2005). Santa Fe, New Mexico. pg. 413-419.

Rimal B. P., Choi E., e Lumb I., (2009). *A Taxonomy and Survey of Cloud Computing Systems*. *Fifth International Joint Conference on INC*, IMS and IDC, Seoul, Korea.

Salvador Z., (2009). *Context Information Provisioning in Tuple Spaces*. *Proceedings of the Proceedings of the 6th Middleware Doctoral Symposium* (MDS 2009). Illinois, USA, pg. 1-6.

Sandhu R. e Park J., (2003). *Usage Control: A Vision for Next Generation Access Control*. *Computer Network Security*, *Lecture Notes in Computer Science (LNCS)*, Springer-Verlag, Berlin, Heidelberg, Volume 2776, 2003, pg. 17-31.

Shao J. e Wei H., (2010). *A Runtime Model Based Monitoring Approach for Cloud*. *Proceedings of the IEEE 3rd International Conference on Cloud Computing* (CLOUD 2010), Miami, Florida, pg. 313-320.

Stihler M., Santin A. O., Calsavara A. e Marcon Jr. A. L., (2009). *Distributed Usage Control Architecture for Business Coalitions*. *Proceedings of the 44th IEEE International Conference on Communications* (ICC 2009). pg. 1-6.

Stihler M., Santin A. O., Marcon Jr. A. L. e Fraga J. da S., (2012*). Integral Federated Identity Management for Cloud Computing*. *Proceedings of the 5th International Conference on New Technologies, Mobility and Security* (NTMS 2012). Istanbul. pg. 1-5.

Sun Y., Xiao Z., Bao D. e Zhao J., (2010). *An architecture model of management and monitoring on Cloud services resources*. *Proceedings of the 3rd International Conference on Advanced Computer Theory and Engineering* (ICACTE 2010), Chengdu, pg. 207-211.

Tavizi T., Shajari M. e Dodangeh P., (2012). *A Usage Control Based Architecture for Cloud Environments*. *Proceedings of the IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum* (IPDPSW 2012), Shanghai, pg. 1534-1539.

W3C, (2007a). *SOAP Version 1.2. Part 1: Messaging Framework* (*Second Edition*). Access: Sep. 2007. Available at: *http://www.w3.org/TR/soap12-part1/.*

W3C, (2007b). *Web Services Description Language* (WSDL) *Version 2.0 Part 1: Core Language.* Access: Sep. 2007. Available at: *http://www.w3.org/TR/wsdl20/.*

Wang Q. e Li N., (2010). *Satisfiability and Resiliency in Workflow Authorization Systems*. *ACM Transactions on Information and System Security* (*TISSEC*), *Volume* 13, *Issue* 4, *Article* No. 40, New York, NY.

Westerinen A., Schnizlein J., Strassner J., Scherling M., Quinn B., Herzog S., Huynh A., Carlson M., Perry J. e Waldbusser S., (2001). *Terminology for Policy-Based Management*, *Internet Engineering Task Force*, *Informational*.

Yildiz M., Abawajy J., Ercan T., e Bernoth A., (2009). *A Layered Security Approach for Cloud Computing Infrastructure*. *Proceedings of the 10th International Symposium on Pervasive Systems, Algorithms, and Networks*. Kaohsiung, Taiwan.

Zhang L. J., e Zhang J. (2009). *An Integrated Service Model Approach for Enabling SOA*. *IT Pro*. IEEE Computer Society.

Zhang Q., Cheng L., e Boutaba R., (2010). *Cloud computing: state-of-the-art and research challenges*, Springer J. Internet Serv. Appl., pp. 7-18, April.

Apêndice A

Comunicação Generativa

A.1. Generative Communication in Linda

Na comunicação generativa as mensagens do ambiente computacional são adicionadas em um formato estruturado em *tuplas* (Gelernter, 1985). Estas existem como entidades que possuem um nome e são independentes até que algum processo escolha por receber a mensagem. A abordagem de comunicação implementada pela linguagem *Linda* possui as seguintes propriedades: completamente distribuída no espaço e no tempo; permite o compartilhamento distribuído de dados; passagem de parâmetros; estrutura de nomes.

Um programa em *Linda* representa uma coleção de *tuplas*, sendo que algumas destas entradas possuem código executável enquanto outras armazenam coleções de dados. O ambiente computacional abstrato, chamado de espaço de *tuplas* (*Tuple Space* - *TS*), é a base do modelo de comunicação da linguagem. Como exemplo, considere um cenário em que dois processos *A* e *B* desejam se comunicar: para enviar um dado para *B*, o processo *A* gera uma *tupla* e adiciona esta ao *TS*; o processo *B* remove a entrada previamente inserida (Figuras A.1 e A.2, respectivamente).

Este modelo de comunicação é chamado de generativo porque, até ser explicitamente removida, a *tupla* gerada por *A* possui uma existência independente dentro do *TS*. Uma entrada depositada no espaço de *tuplas* está acessível a todos os processos dentro do *TS*, porém, não está vinculada a nenhum destes. As principais operações definidas para o *TS* são: *out()* - adiciona uma *tupla* ao *TS*; *in()* - remove uma entrada do *TS*; *read()* - faz a leitura da *tupla* armazenada no *TS* sem remover a mesma.

As variáveis e constantes inseridas na *tupla* possuem rótulos de identificação. Uma declaração do tipo "*out(N, P_2, ..., P_j)*" possui: "*P_2, ..., P_j*" os quais representam uma lista

de parâmetros que podem ser formais ou reais; *N* é um parâmetro do tipo "*name*", referindo-se a um conjunto de caracteres que representam um identificador. Assumindo que todos os "P_i" são parâmetros reais, a execução da declaração *out()* resulta na inserção da *tupla* "*N, P_2, ..., P_j*" no *TS*.

A declaração "*in(N, P_2, ..., P_j)*" possui: "*P_2, ..., P_j*" como sendo uma lista de parâmetros que podem ser formais ou reais; *N* é o parâmetro do tipo "*name*". Para este cenário, todos os "P_i" são parâmetros formais. Quando a operação "*in()*" é executada, se alguma *tupla* com tipo consonante existe no *TS*, tendo como primeiro componente *N*, esta respectiva entrada é removida do espaço. Para o processo executor prosseguir, os valores dos reais são atribuídos para os formais da declaração "*in()*". Se nenhuma *tupla* compatível estiver armazenada no *TS*, a operação "*in()*" é suspensa enquanto não houver uma entrada apropriada. A declaração "*read(N, P_2, ..., P_j)*" é idêntica a operação "*in()*" exceto que, quando uma *tupla* compatível é encontrada, está permanece no *TS*.

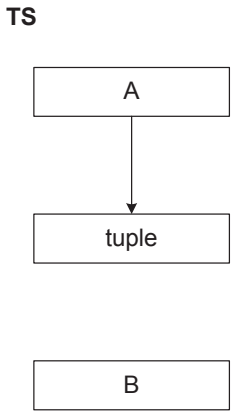

Figura A.1: Processo *A* gera uma *tupla*. Adaptado de Gelernter, 1985.

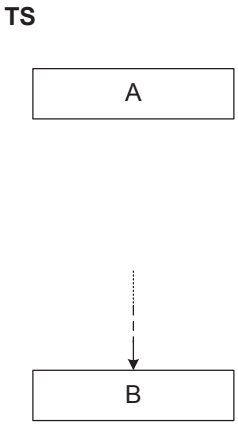

Figura A.2: Processo *B* remove a *tupla*. Adaptado de Gelernter, 1985.

Várias declarações "*in(N,...)*" podem ser executadas e ou suspensas simultaneamente em diferentes partes do sistema distribuído. A execução da operação "*out(N,...)*" envia uma cópia simples da *tupla* para o *TS*. Algumas declarações "*in(N,...)*" suspensas vão receber a entrada, porém, não se pode determinar qual destas operações vai receber a *tupla*. O mesmo acontece para as declarações "*in(N,...)*" ou "*read(N,...)*" sendo executadas subsequentemente a uma grande quantidade de "*out(N,...)*". Algumas das *tuplas* armazenadas serão comparadas, porém, não é possível determinar qual delas.

Se duas declarações "*in()*" estão competindo por uma *tupla*, somente uma destas vai obter a entrada correspondente - a *tupla* não pode ser dividida. Vale ressaltar que, se

duas operações "*read()*" e "*in()*" estão competindo por uma entrada, ambas podem ser servidas. Se a declaração "*read()*" for servida por primeiro, a operação "*in()*" pode subsequentemente remover a referida *tupla*. Se a declaração "*in()*" for servida por primeiro, esta remove a entrada e a operação "*read()*" vai ficar bloqueada até outra *tupla* estar disponível no *TS*.

Supondo que "*+t*" faça referência a qualquer *tupla* adicionada ao *TS* por uma declaração "*out(+t)*". Então, "*-t*" indica a lista de parâmetros das operações "*in()*" ou "*read()*" que pretendem receber a entrada "*+t*". Os parâmetros reais que aparecem na *tupla* "*-t*" constituem, coletivamente, um nome estruturado global para um determinado *TS*. Sendo assim, a declaração "*in(P, i:integer, j:boolean)*" solicita uma entrada com o nome "*P*". Porém, também é possível escrever "*in(P, 2, j:boolean)*", solicitando a *tupla* com o nome estruturado "*P, 2*".

Declarações nas formas "*in(P, i:integer, FALSE)*" e "*in(P, 2, FALSE)*" também podem ser aplicadas. No primeiro caso, o nome é "*P, FALSE*" enquanto para o segundo caso, o nome estruturado é "*P, 2, FALSE*". Todos os componentes reais de uma lista "*-t*" devem ser combinados, de forma idêntica, com a *tupla* "*+t*" para que a correspondência ocorra.

Assim como os componentes de "*-t*" podem ser reais, os elementos de uma *tupla* "*+t*" podem ser formais. As operações "*out(P, 2, FALSE)*" e "*out(P, i:integer, FALSE)*" representam estas declarações. A *tupla* "*P, i:integer, FALSE*" pode ser recebida por qualquer declaração que determine como primeiro componente "*P*", último componente "*FALSE*", e como componente intermediário algum real do tipo inteiro.

A comunicação generativa possui duas características fundamentais: *1)* ortogonalidade da comunicação - assim como o receptor não tem conhecimento sobre quem é o emissor, o emissor não tem conhecimento sobre quem é o receptor. Essa ortogonalidade possui duas características importantes: desacoplamento no espaço e desacoplamento no tempo; *2)* compartilhamento distribuído - sendo uma consequência das propriedades anteriores.

Desacoplamento no espaço (nomenclatura distribuída): refere-se ao fato de que uma *tupla* no *TS* pode ser utilizada como entrada para qualquer processo, independentemente do espaço de endereços. Um processo "*j*", sendo executado em "*x*" nodos distintos, pode aceitar todas as *tuplas* rotuladas com um determinado nome. Esse esquema de nomenclatura significa que *Linda* é completamente distribuída no espaço. Sendo assim, a linguagem permite que uma instrução de entrada aceite dados oriundos de

qualquer espaço de endereços. Para a Figura A.3, cenário (*a*), a *tupla* "*t*" é uma entrada para "*R*", porém, esta pode ter sido gerada em qualquer endereço. No cenário (*b*), a *tupla* "*t*", envida por "*S*", pode ser recebida por qualquer espaço de endereços.

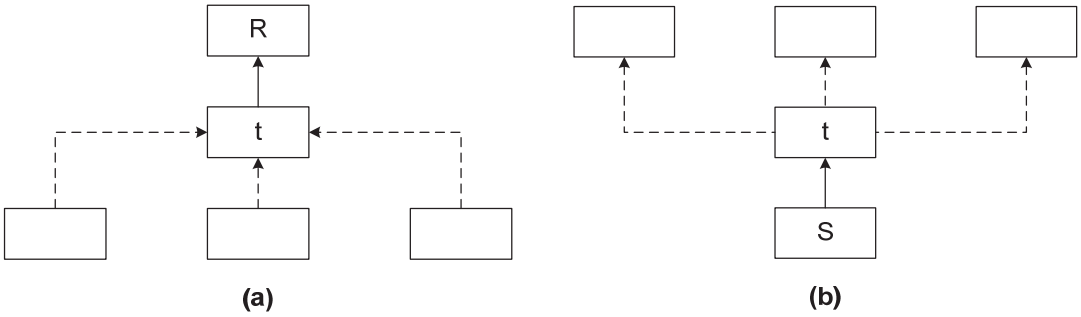

Figura A.3: Cenário com a *tupla* "*t*". Adaptado de Gelernter, 1985.

Uma *tupla* adicionada ao *TS* pela operação "*out()*" permanece no espaço até ser removida pela declaração "*in()*". Se a entrada nunca for removida, em teoria, será mantida no *TS* para sempre. Na prática, *tuplas* adicionadas ao *TS* serão removidas uma vez que todos os processos tenham terminado, a menos que o programador tenha indicado explicitamente o contrário. *Linda* permite que os programas sejam distribuídos no tempo (desacoplamento temporal), ou seja: um processo *A*, executando declarações "*out()*", pode ser executado e finalizado antes do processo "*B*" executar as operações "*in()*".

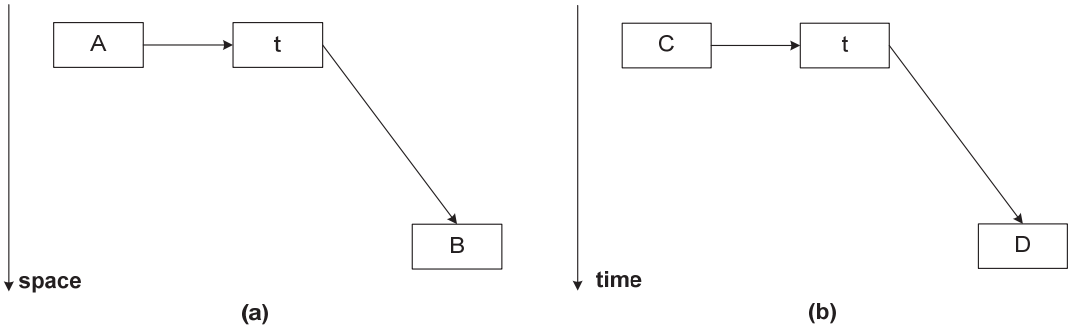

Figura A.4: Processos separados no espaço e no tempo. Adaptado de Gelernter, 1985.

A linguagem *Linda* permite a comunicação entre processos que são separados temporalmente. Para o cenário (*a*) (Figura A.4), existem processos "*A*" e "*B*", separados no espaço, que se comunicam via *TS*. No cenário (*b*), existem processos "*C*" e "*D*" separados no tempo. O processo "*C*" finaliza antes do processo "*D*" ser inicializado - da mesma forma, ambos se comunicam via *TS*.

Linda permite que "*j*" processos distintos compartilhem alguma variável "*v*" depositando esta no *TS* (compartilhamento distribuído). As definições do operador do *TS* asseguram que "*v*" será mantido atomicamente, não sendo necessário que uma variável compartilhada seja implementada por outro módulo.

Apêndice B

Trabalhos Complementares

B.1. Cloud Security with Virtualized Defense and Reputation-based Trust Management

A proposta sugere a adoção de um sistema de reputação visando a integração de *clusters* virtuais, *data-centers*, e o respectivo acesso a dados confiáveis (Hwang, 2009). Uma hierarquia de sistemas de reputação *peer-to-peer* (*P2P*) é sugerida para proteger a nuvem. Sendo os serviços um conceito chave da nuvem, os principais assuntos a serem considerados pela proposta são: integridade e confidencialidade dos dados; modelos de confiança entre provedores e usuários.

Os autores identificaram algumas das características de segurança e privacidade almejadas por consumidores da nuvem: implantação de interfaces para a autenticação de usuários; utilização de protocolos de segurança – *HTTPS*, *SSL*; controle de acesso com granularidade fina visando proteger a integridade dos dados e inibir o acesso de intrusos; proteger dados compartilhados contra alterações maliciosas, violação dos direitos de acesso, cópia ou eliminação.

A aplicação de medidas de segurança deve proteger os *hypervisors* e monitores de *VMs* de ataques que visam a exploração de vulnerabilidades, negação ou comprometimento do serviço. A virtualização pode melhorar a segurança da nuvem, porém, as *VMs* adicionam uma camada de *software* que pode se tornar um ponto de falhas. Neste contexto, algumas técnicas foram sugeridas pelos autores: *a) Segurança Através da Virtualização:* instanciar a *VM* em uma partição protegida de ataques de negação de serviço vindos de outras partições (*e.g.* fazer com que as falhas relacionadas a uma *VM* não afetem as outras instâncias); *b) Máquina Virtual como uma Sandbox:* controlar os recursos fornecidos para o sistema hospedeiro (*e.g.* considerar a *VM* um

componente de *software*).

A confiança entre diferentes domínios pode ser negociada através de políticas não conflitantes. As abordagens sugeridas para este contexto são: *gerenciamento de confiança e reputação* - construção de uma rede *overlay* que permita o desenvolvimento de um sistema de gerenciamento entre os *data-centers* (Figura B.1); *manter a consistência dos dados replicados* - utilização de mecanismos que implementam múltiplas réplicas visando garantir a alta disponibilidade dos dados; *privacidade de dados em nuvens públicas* - aplicação de mecanismos de defesa contra ataques e o estabelecimento de políticas consistentes.

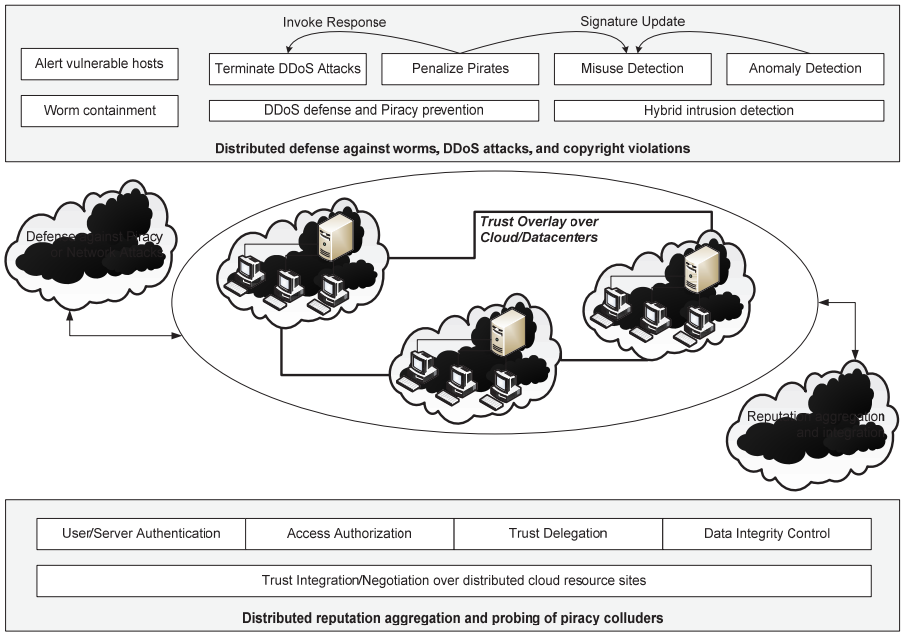

Figura B.1: Gerenciamento de Confiança *Overlay*. Adaptado de Hwang, 2009.

O artigo propõe a utilização de uma rede sobreposta responsável por estabelecer relacionamentos confiáveis entre diferentes provedores de nuvem (*e.g.* utilizar uma *DHT* (*Distributed Hash Table*) para distribuir e gerenciar os dados do ambiente). A proposta avalia a utilização de um repositório compartilhado distribuído (*i.e. tuple space*) para armazenar e recuperar os atributos do ambiente (*e.g.* informações sobre o usuário, serviço).

A confiança entre as entidades que interagem com o ambiente é estabelecida através de um domínio intermediário. Este domínio administra a alocação dos serviços para os usuários do consumidor. A indireção causada pela intermediação do acesso permite que diferentes provedores forneçam serviços para um mesmo contratante,

tornando este processo transparente para consumidores.

B.2. On-Demand Dynamic Security for Risk-Based Secure Collaboration in Clouds

A proposta apresenta uma infraestrutura em nível *PaaS* que combina ferramentas de gerenciamento de risco com *frameworks* de política para Serviços *Web* (Bertram, 2010). Esta combinação visa auxiliar na mitigação de ameaças de segurança durante o ciclo de vida de uma aplicação implantada na nuvem. A infraestrutura orientada a serviço (*Service Oriented Infrastructure – SOI*) descrita no artigo visa proporcionar uma solução em nível *PaaS* para o gerenciamento de riscos de segurança. O *SOI* aborda o gerenciamento de dados, permitindo o planejamento de redes de serviço e a avaliação das consequências que decisões de segurança podem acarretar ao ambiente (*e.g.* mitigação de ameaças). O esquema proposto concentra-se na necessidade de políticas dinâmicas entre federações, considerando o controle de acesso, identidades e a respectiva comunicação dos requisitos entre as diferentes entidades.

Um dos componentes em nível *PaaS* se refere a segurança operacional da empresa. Esta é responsável pelo planejamento da rede de serviço considerando os requisitos do negócio (Figura B.2). Esta tarefa é executada utilizando uma ferramenta de suporte a decisão (*Cloud Security Decision Support Tool – CS-DST*) que permite gerenciar o risco inerente ao ciclo de vida das aplicações. Como resultado, um modelo é produzido com base em: amostras de segurança conhecidas, caminhos de comunicação, ameaças e vulnerabilidades com suas respectivas mitigações, processo de estabelecimento de confiança, direitos de acesso e papéis do usuário.

A implantação de políticas durante o provisionamento da aplicação visa fornecer as regras de gerenciamento (*Governance*; Figura B.2). Esta ação informa ao provedor para provisionar o *SOI* (*Service-Oriented Infrastructure*) de acordo com o modelo de segurança previamente definido. O *SOI* possui um serviço de credencias seguras para identidades (*e.g. X.509*), interceptadores para a aplicação de políticas, e um serviço de avaliação de políticas. Os processos de negócio usados para manipular os ativos também são fontes de eventos de segurança. Estes eventos são usados durante a fase de análise para identificar as ameaças e executar o processo de avaliação dentro da ferramenta *CS-DST*.

A solução *PaaS* inclui um conjunto de serviços que podem ser utilizados para alterar as políticas de segurança em tempo de execução, acarretando mínimas perdas de funcionalidade e pouca assistência manual. Esta capacidade de re-configurar as políticas visa atender eventos como alterações em: contratos, equipes de trabalho, ambiente de execução (*e.g.* de um ambiente confiável para um não confiável).

A arquitetura proposta suporta a federação de identidades e políticas de autorização para os serviços da nuvem. O sistema de autorização considera diferentes eventos (*e.g.* intrusão) quando decide se um certo pedido deveria ser concedido ou revogado. Esta abordagem permite expor níveis de serviço de acordo com o contexto da avaliação.

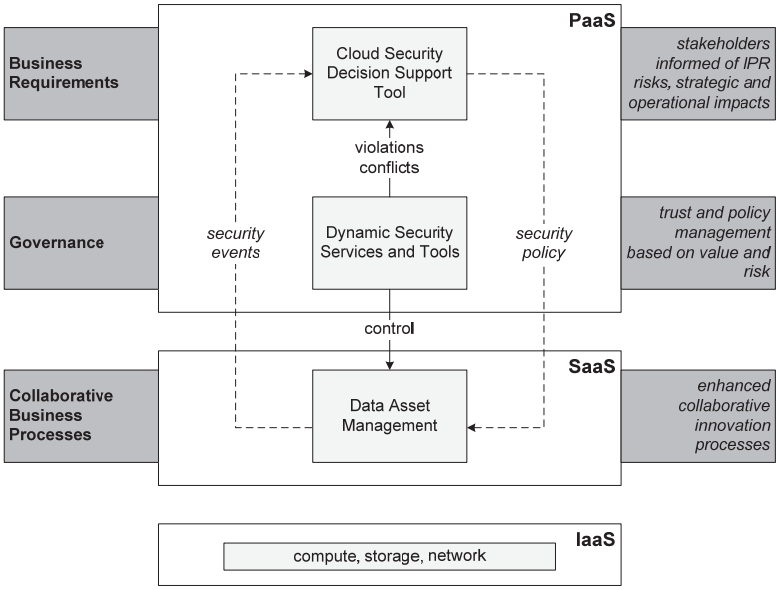

Figura B.2: Serviços de Segurança. Adaptado de Bertram, 2010.

As informações extraídas do contexto permitem que funcionalidades adaptativas sejam inseridas, fazendo com que a plataforma reaja aos eventos do ambiente (*e.g.* modificar o comportamento de um componente) ao invés de falhar em caso de problemas ou ameaças. A política pode exigir que as funções que retornam ativos digitais sensíveis se tornem indisponíveis quando um ataque for detectado.

A abordagem proposta baseia-se em atributos de segurança definidos pelo provedor e implantados em políticas de controle de acesso. O emissor é a entidade que define quais atributos deveriam ser emitidos para cada usuário, porém, como esses serão interpretados depende do provedor de serviço. O provedor, tendo entendido como interpretar os atributos da política de acesso, pode transferir estas informações para outras entidades provedoras.

Um serviço pode controlar vários recursos, sendo que cada um destes é identificado por um *ID* (*i.e. URI - Uniform Resource Identifier*) e possui um estado referente a um fluxo de trabalho. Cada uma das operações que podem ser executadas em um determinado estado relaciona-se com um conjunto de papéis (*i.e.* papéis de processo ou de fluxo de trabalho) que podem invocar a operação. A definição do estado e dos papéis do fluxo de trabalho resultam em uma política estática.

Para acessar uma operação, o sujeito solicita um atributo de segurança que é mapeado para um papel de fluxo de trabalho. O sujeito pode invocar a operação se a intersecção dos dois conjuntos não for vazia (*i.e.* possui um papel que lhe garante a permissão). As regras para determinar os papéis de um sujeito podem ser alteradas dinamicamente, permitindo que os usuários deleguem seus direitos de acesso para outros - desde que a política permita o processo de concessão.

Para o usuário possuir um papel, este deve corresponder a uma das seguintes regras: *Sufficient*, *Necessary*, *Deny*. Existem 5 tipos de regras: *1) Subject DN is*: corresponde a algum sujeito com um nome distinto (*DN - Distinguished Name*); 2) *Certificate is signed by*: qualquer sujeito cuja identidade seja aferida por uma determinada autoridade certificadora; 3) *Has SAML attribute*: qualquer sujeito com uma assertiva *SAML* assinada por uma emissor específico; 4) *Member of group*: qualquer sujeito que seja membro de um determinado grupo; 5) *Anyone*: corresponde a qualquer sujeito.

A proposta para efetuar a avaliação de risco quanto aos dados sendo armazenados, processados ou acessados em nuvens computacionais considera somente o nível de negócio e a camada *PaaS*. A aplicação de especificações amplamente aceitas e difundidas - como é o caso do Serviços *Web* - mostra que a integração de diferentes abordagens em uma única solução é possível e também necessária. A comunicação de dados inter-domínios, a criação de políticas de controle de acesso ou a utilização de mecanismos padronizados pode fazer amplo uso das propostas e modelos disponibilizados para os Serviços *Web*.

B.3. Access Control of Cloud Service Based on UCON

O módulo de negociação proposto é aplicado para aumentar a flexibilidade do sistema de controle de acesso para a nuvem. Quando a solicitação de uso não é compatível com as políticas, esse módulo permite ao usuário obter uma segunda chance

de acesso, por meio de negociação, ao invés de negar o acesso diretamente (Danwei, 2009). A proposta possui os seguintes principais componentes (Figura B.3): *user* - solicita o acesso ao serviço; *SAML Server* - responsável por emitir e avaliar assertivas, proteger atributos sensíveis e negociar com o servidor de nuvem por atributos, obrigações e condições.

O *Cloud Service* é o provedor de serviço, possuindo os seguintes módulos (Figura B.3): *1) PEP* - responsável por receber as solicitações dos usuários e executar as decisões do *PDP*; *2) PDP* - avalia as decisões de autorização, baseado nas políticas *ABC*, de acordo com os atributos da entidade, condições e obrigações; *3) PIP* - recupera as condições e os atributos da entidade, fornecendo estas para o *PDP* avaliar uma decisão; *4) PAP* - gerencia as políticas; *5)* repositório de políticas *XACML* - armazena as políticas *ABC* escritas com a linguagem *XACML*; *6)* módulo de negociação - utilizado para negociar com o usuário da nuvem por atributos, obrigações e condições.

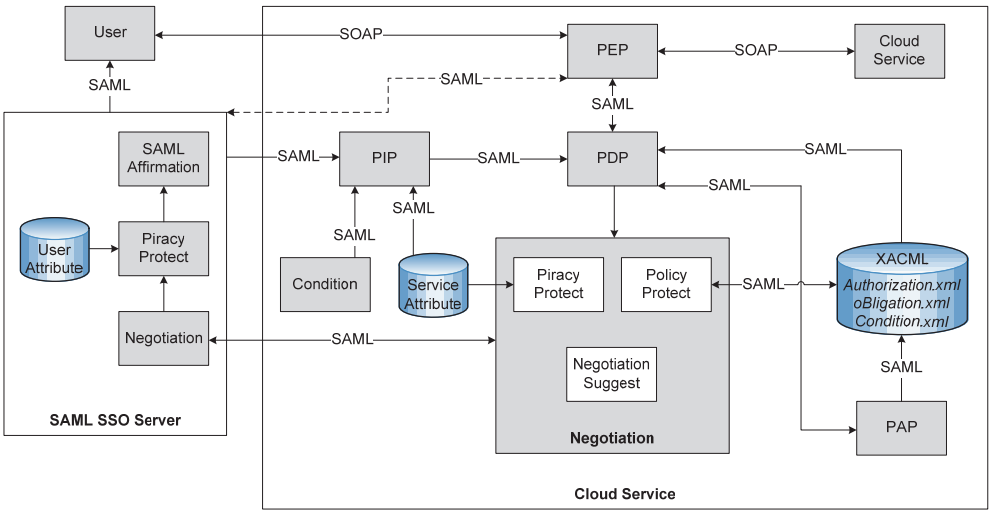

Figura B.3: Modelo de Negociação *UCON*$_{ABC}$. Adaptado de Danwei, 2009.

O módulo de negociação foi adicionado para aumentar a flexibilidade do modelo de controle de acesso (Figura B.4). Quando os atributos do usuário são insuficientes ou os parâmetros de condição são inconsistentes, o módulo de negociação é executado. Este módulo é dividido em três níveis de negociação: *1)* pesquisa de atributos - procura por atributos do usuário quando estes são insuficientes; *2)* negociação automática de atributos - auxilia a obter os atributos desejados conforme a política de privacidade de ambos os lados; *3)* negociação artificial - utilizada quando as abordagens anteriores não obtiverem sucesso. Neste nível, o serviço de nuvem envia sugestões de negociação para que o

usuário altere sua política de privacidade ou as condições da solicitação.

O módulo de negociação possui duas partes: *1) Servidor SAML*, o qual possui três módulos, *a)* assertivas *SAML* - encarregado de responder as solicitações de atributos do usuário enviadas pelo serviço, *b)* proteção de atributos - garantir a privacidade dos atributos durante a negociação automática, *c)* negociação - fornece uma maneira de acessar os serviços da nuvem através da negociação artificial; *2) Serviço de Nuvem*, o qual possui três módulos, *a)* proteção de política - preservar a política que decide a sequência de pesquisa segura para os atributos, *b)* proteção de atributo - garantir a privacidade dos atributos durante a negociação automática, *c)* negociação - fornece uma maneira alternativa de acessar os serviços da nuvem através de uma transação artificial.

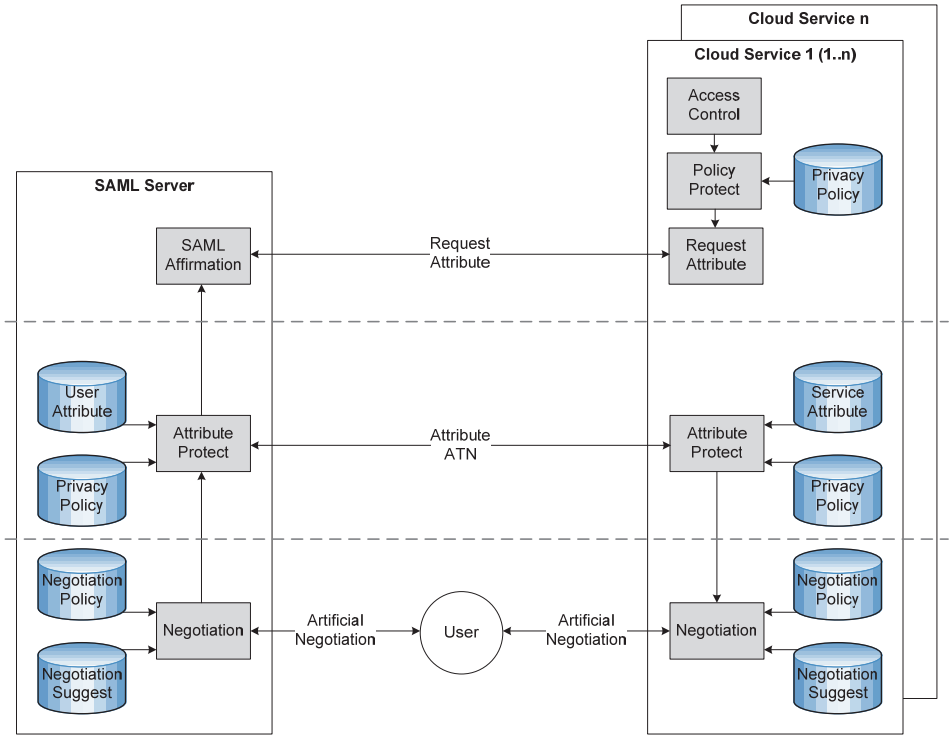

Figura B.4: Módulo de Negociação *UCON*$_{ABC}$. Adaptado de Danwei, 2009.

B.4. Enhanced Monitoring-as-a-Service for Effective Cloud Management

A proposta introduz o conceito de Monitoramento como um Serviço (*Monitoring-as-a-Service* - *MaaS*). Se comparado com as abordagens convencionais, o *MaaS* deveria suportar funcionalidades adicionais (*e.g.* detectar violações instantaneamente, monitoramento de estado periódico) (Meng, 2012). A proposta considera somente os

provedores de serviço em nível *IaaS* e *PaaS* para a aplicação do monitoramento.

O artigo apresenta as três principais funcionalidades do serviço *MaaS*: monitoramento de estado baseado em janelas; monitoramento de estado baseado em probabilidade de violação; monitoramento de estado com suporte a vários arrendatários. Os três requisitos essenciais identificados para o serviço *MaaS* são: minimizar a carga de trabalho na comunicação; minimizar o ruído; e maximizar a efetividade da tarefa de monitoramento.

Em nível de detecção de violação global, o monitoramento baseado em janelas provê flexibilidade para capturar a violação de estado, preserva significativamente os gastos com comunicação e reduz os ruídos se comparado com o monitoramento instantâneo. No nível de acompanhamento de estado local, a probabilidade de violação baseada em amostragem permite ajustar dinamicamente a intensidade do monitoramento, baseando-se na probabilidade de uma violação ser detectada.

A monitoração de estado avalia continuamente se um determinado aspecto da aplicação distribuída diverge de seu comportamento normal (*e.g.* tempo limite das solicitações). Esta abordagem é ideal para ser fornecida como um serviço devido a sua flexibilidade em atender as necessidades do usuário (*e.g.* o estado pode ser definido considerando as medidas de desempenho - *e.g.* utilização da *CPU* - ou medidas definidas pelos usuários - *e.g.* utilização do *cache* em nível de aplicação).

O modelo de detecção baseado em janelas dispara um alerta quando o valor monitorado exceder um limiar durante, pelo menos, *L* unidades de tempo. Alterando-se o tamanho da janela de monitoramento, os usuários podem ajustar a sensibilidade da detecção de violação. Este modelo economiza significativamente os custos de comunicação. O desenvolvimento desta solução envolve os seguintes pontos: elaboração de um algoritmo apropriado para a detecção distribuída; utilização da janela de monitoramento para minimizar os custos de comunicação; desenvolvimento de técnicas para aperfeiçoar automaticamente os parâmetros de monitoramento (*e.g.* limiar local, tamanho da janela), mantendo sempre um custo de comunicação baixo independentemente das condições de trabalho.

Picos de tráfego que causam as violações de estado não acontecem de forma súbita. Antes da violação acontecer, o nível de tráfego cresce em direção ao limiar e, com isto, a probabilidade de detectar violações aumenta. Baseado nesta observação, um esquema alternativo pode ajustar os intervalos de amostragem durante a execução do sistema de monitoramento.

Porém, aplicar a probabilidade de violação baseada em amostragem apresenta os seguintes desafios: *1)* encontrar uma técnica eficiente para medir a probabilidade de violação (*measure the violation likelihood - VL*). A abordagem deveria diferenciar as tarefas que apresentam mudanças estáveis e estas que apresentam mudanças de maneira mais volátil; *2)* criar um esquema para ajustar os intervalos de amostragem baseado na estimação da *VL*. Esta abordagem deveria realizar ajustes baseado na precisão especificada pelo usuário (*e.g.* tolerar no máximo *1%* de violações não detectadas se comparado com uma amostragem executada a cada *10* segundos); *3)* tarefas de monitoramento de estado com vários monitores, onde cada um destes observe diferentes faixas de valores.

Para dar suporte a vários arrendatários é apresentada uma técnica de consolidação consciente. Devido aos monitores confiarem na topologia para entregar os dados, o suporte a vários arrendatários é crítico para o monitoramento de serviços. A técnica utilizada na proposta explora a oportunidade de compartilhamento de custos entre várias tarefas de monitoramento. Esta abordagem remove as cargas de trabalho duplicadas e reduz a frequência de mensagens erradas.

O cenário mostrado utiliza seis nodos de monitoramento, sendo que cada um destes acompanha um conjunto de atributos (Figura B.5). A figura mostra as topologias amplamente utilizadas: *A* - estrela, todo nodo envia suas atualizações para o ponto central onde os coordenadores das tarefas estão hospedados; *B* - árvore simples, todos os nodos estão incluídos. Enquanto se reduz a carga de processamento nos nodos raiz e central, os demais nodos da árvore precisam confiar nos dados de monitoramento para todas as tarefas; *C* - árvore dividida em tarefas, é construída uma árvore separada para organizar os monitores de cada tarefa. Porém, se um nodo monitor executar várias tarefas, este deve participar de várias árvores; *D* - os nodos são divididos em duas árvores, uma árvore para as tarefas *A* e *B* e outra para a tarefa *C*.

Dentro de cada *cluster* podem ser construídas árvores com base nos recursos disponíveis em cada nodo - exemplo *E*. Tendo em vista que os nodos da nuvem dedicam a maior parte dos seus recursos para hospedar as aplicações, é crucial que nenhum destes esteja sobrecarregado com a carga de trabalho de monitoramento. Esta abordagem, por sua vez, assegura o bom desempenho dos serviços de monitoramento.

O planejamento da topologia consciente do recurso adota um método dirigido de pesquisa local. O esquema inicia criando uma árvore separada para cada tarefa (exemplo *C*) e, interativamente, a abordagem mescla ou divide as árvores visando otimizar a

topologia de monitoramento. A otimização é aplicada até que nenhuma melhoria possa ser executada, sendo baseada em dois objetivos prioritários: *1)* maximizar o número de tarefas suportadas pela topologia, enfatizando a escalabilidade do serviço; *2)* minimizar os custos de entrega global para os dados (por mensagem e retransmissão) tentando poupar recursos para as tarefas futuras. Adicionalmente, a proposta se ajusta naturalmente aos serviços de monitoramento em nuvem, pois pode melhorar continuamente a topologia conforme as tarefas são adicionadas/removidas.

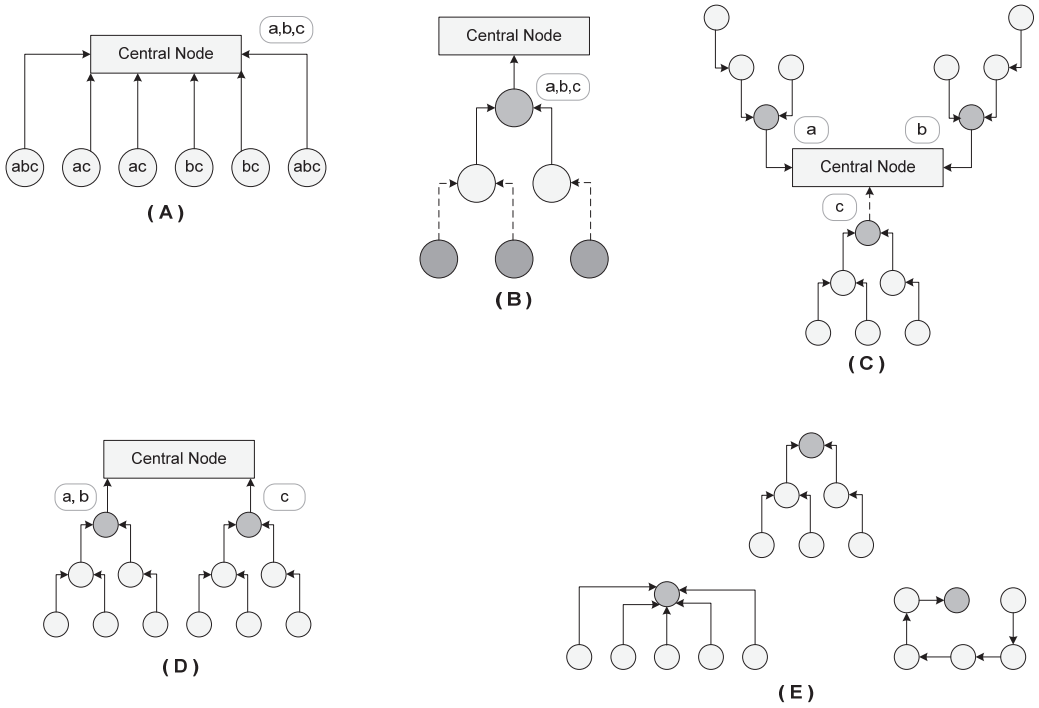

Figura B.5: Cenários para a topologia proposta. Adaptado de Meng, 2012.

B.5. Managed control of composite cloud systems

Para gerenciar efetivamente os recursos da nuvem, os sistemas devem automatizar a coleta das métricas de *QoS* (*quality-of-service*). A abordagem pode ser utilizada para atender as necessidades de grandes sistemas, os quais se estendem além dos limites de um único provedor (Lamb, 2011). Sistemas que podem ser controlados permitem aos provedores fornecer serviços bem definidos e com um custo mais efetivo. Considerando estes itens, propõe-se um sistema de gerenciamento que utiliza os parâmetros de *QoS* (*e.g* atributos do sistema como largura da banda, alocação de memória).

Para gerenciar um sistema, atendendo a um conjunto de métricas, é necessário ter acesso a informações de monitoramento relacionadas aos fatores que afetam estas métricas. Adicionalmente, é necessário ter a capacidade de ajustar o desempenho do sistema considerando os valores obtidos (*e.g.* tempo de resposta para o sistema monitorado).

Através da identificação do que precisa ser controlado, obteve-se um grupo de requisitos que podem ser utilizados para montar a arquitetura do sistema de gerenciamento: *desempenho* - o sistema vai funcionar em conformidade com as condições de tempo (*i.e. soft real-time frame*). Para isso, é necessário coletar e processar as medidas (*i.e. feedback*), tomando as decisões para evitar uma inconformidade com os parâmetros de desempenho; *acessibilidade* - para executar o controle é necessário ter acesso aos componentes do sistema; *capacidade de ser controlado* (*controllability*) - habilidade de acessar as primitivas de gerenciamento no sistema a ser ajustado (*e.g.* finalizar nodos com uma configuração e criar nodos com requisitos mais adequados as necessidades do ambiente).

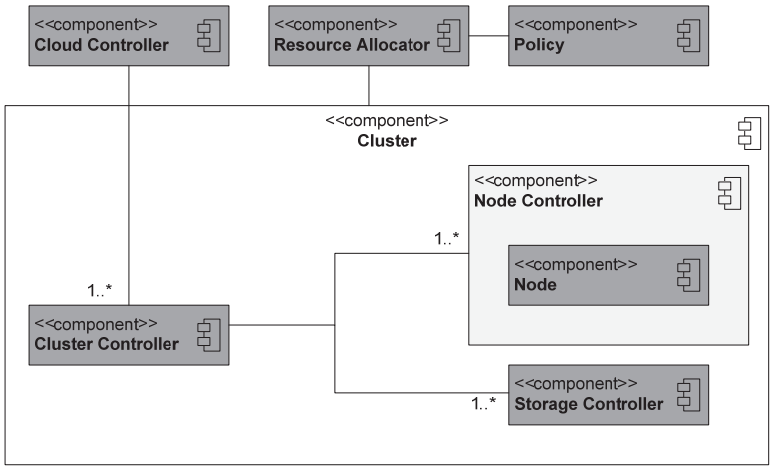

Figura B.6: Provedor de Nuvem e *QoS*. Adaptado de Lamb, 2011.

O sistema possui sete componentes primários que fornecem serviços para os consumidores (Figura B.6). Essencialmente, existe um elemento controlador de nuvem que inicia a configuração dos nodos computacionais dentro de um determinado *cluster*. Este *cluster* é gerenciado por um controlador de *cluster*, que por sua vez administra os controladores de armazenamento e os controladores de nodo, juntamente com os respectivos nodos. Os nodos e os controladores de nodos são monitorados por um alocador de recursos referente a um conjunto de requisitos de *QoS*.

Os principais componentes do modelo proposto são: controlador de nuvem -

fornece a interface para o administrador gerenciar a nuvem; controlador de *cluster* - módulo que administra os recursos de um único *cluster*, sendo gerenciado pelo controlador de nuvem; controlador de armazenamento - contêm as imagens do sistema; controlador de nodo - responsável por alocar, instanciar e gerenciar nodos computacionais que executam os sistemas do consumidor; nodo computacional - disponibiliza os serviços para os usuários finais; política - regras de *QoS* que o provedor de nuvem concordou em honrar para o consumidor (*e.g.* desempenho, configuração); alocador de recurso - responsável pelos ajustes do sistema da nuvem. Este componente visa manter a qualidade de serviço previamente definido.

Os comandos para inicializar a nuvem são disparados pelo controlador de nuvem em direção ao controlador de *cluster*. Este último envia os parâmetros para provisionar um conjunto inicial de recursos no controlador de nodo e no controlador de armazenamento. Nesta etapa, o sistema se encontra configurado e em execução, fornecendo os programas hospedados para o consumidor. Nesta fase, as métricas de desempenho são enviadas pelos nodos, controladores de nodos e controladores de armazenamento para o alocador de recursos. Este módulo processa rapidamente os dados de eventos de acordo com os parâmetros de *QoS*. Esta restrição temporal (*soft real-time frame*) evita que se gaste muito tempo na execução de análises complexas nos dados. No final, caso necessário, o alocador de recursos envia mensagens para os componentes do sistema ajustarem seus parâmetros (*i.e.* controlador de *cluster*, controlador de nodo, nodos e controlador de armazenamento), assegurando que estes permaneçam dentro de intervalos de desempenho aceitáveis.

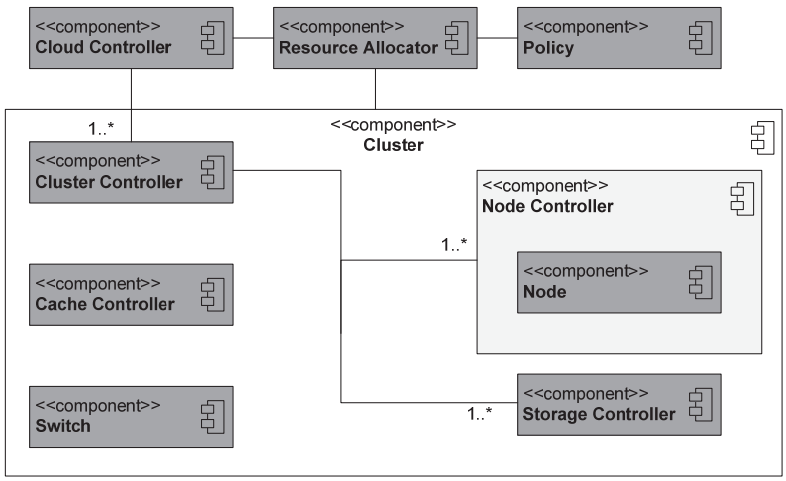

Figura B.7: Provedor de Nuvem e Gerenciamento de Uso. Adaptado de Lamb, 2011.

Para um fluxo de dados qualquer, emitido por um nodo, podem existir situações que precisam ser controladas. O objetivo disto é proibir atualizações no fluxo de dados e limitar quem pode ler este fluxo. A restrição de legibilidade exige a adição do controle de uso sobre o fluxo de dados (*i.e.* informações transmitidas através do sistema devem ser monitoradas, sendo que o acesso ao fluxo precisa ser ajustado dinamicamente). Isto implica na habilidade de controlar o encaminhamento e o *caching* da transmissão de dados conforme as condições definidas pelo usuário, controlando, também, quais nodos são capazes de enviar dados para quais nodos.

O sistema descrito acima possui novos componentes (Figura B.7), estes são responsáveis por limitar o acesso aos fluxos de rede. Os novos módulos são: controladores de *cache* - fluxos de dados podem ser armazenados temporariamente em sistemas de *cache* estrategicamente posicionados; *switch* - qualquer tipo de componente físico que controle a entrega dos dados (*e.g.* roteadores).

B.6. P&P - A Combined Push-Pull Model for Resource Monitoring in Cloud Computing Environment

A proposta estende os esquemas de monitoramento aplicados em *grid* (*i.e.* modelos *Push & Pull*) para o ambiente de nuvem. Baseado nas propriedades complementares dos dois modelos, propõem-se uma abordagem de monitoramento chamada de *P&P* (Huang, 2010). A proposta emprega as duas técnicas, alternando estas de acordo com as necessidades do usuário e as condições dos recursos monitorados.

Os dois modelos básicos de comunicação entre produtores e consumidores podem ser definidos da seguinte maneira: *pull* - os consumidores são responsáveis por puxar as informações dos produtores para saber o estado dos recursos. Este modelo reduz os custos de transmissão quando o intervalo de consulta está configurado de maneira adequada; *push* - quando ocorrem atualizações no produtor, este envia o novo estado do recurso para o consumidor. Este modelo é mais preciso quando o limiar (*i.e.* condição que determina o envio do estado) é definido de maneira apropriada.

O esquema de monitoramento híbrido, chamado de *P&P*, é composto por algoritmos *push*/*pull* sendo executados simultaneamente no consumidor e no provedor. As principais características desta abordagem são: melhor desempenho, visto que o

modelo *P&P* pode alternar entre os esquemas *push*/*pull* de acordo com as necessidades do usuário ou o estado do recurso; mais apropriado para a nuvem que o modelos *push*/*pull* puros, considerando que os recursos virtuais podem ter diferentes privilégios ou estilos de acesso.

O grau de mudança (***2***) descreve a dimensão da alteração entre o estado atual de um produtor e a situação preservada pelo respectivo consumidor. Toda a informação de estado contém um carimbo de tempo que registra o momento no qual o produtor coletou a informação. O t_p representa o carimbo de tempo do produtor previamente ao tempo *t*. O t_c representa o carimbo de tempo do consumidor anteriormente ao tempo *t*. Temos assim $t_p >= t_c$ devido a atualização do produtor ser sempre anterior a atualização do consumidor. O $P(t_p)$ representa o estado real do produtor no tempo *t*. O $C(t_c)$ significa o estado da informação que o consumidor possui no tempo *t*, esta representa a última atualização que o consumidor recebeu. Os símbolos *MAX* e *MIN* são os valores máximo e mínimo possíveis para o valor do estado.

$$\text{change\_degree} = \frac{|P(t_p) - C(t_c)|}{\text{MAX-MIN}} \leq \text{UTD} \ \ (t_p \geq t_c) \qquad \textit{(2)}$$

As necessidades dos usuários são expressadas conforme o conceito do grau de tolerância (*User Tolerant Degree* - *UTD*; ***2***). Esta descreve quão tolerante um usuário é com relação a imprecisão do estado. Um *UTD* pequeno significa que o usuário possui requisitos de precisão rigorosos. Um *UTD* amplo significa que o usuário está preparado para tolerar níveis significativos de imprecisão.

No primeiro caso (***3***), o usuário possui exigências rigorosas para os recursos monitorados. Este submete tarefas computacionais para alguns recursos, necessitando de informações de estado atualizadas. Neste cenário, o método *pull* será ativado periodicamente, visando evitar a indisponibilidade do produtor.

Para o segundo caso, o usuário deseja somente informações com granularidade grossa em relação ao recurso (*i.e.* deseja conhecer os atributos aproximados). Adicionalmente, para alguns cenários, o estado de alguns recursos monitorados se altera de modo perceptível, sendo assim, operações de envio de informações podem ser necessárias durante as operações de busca.

No último caso (***3***), a necessidade do consumidor é relativamente moderada. Neste cenário, a decisão correta é ambígua, devido ao nível de exigência estar situado entre os dois exemplos supracitados. Dependendo das circunstâncias, qualquer operação (*i.e. push*/*pull*) pode ser disparada.

$$\text{monitoring strategy} = \begin{cases} \text{push-based dominates (UTD is relatively small)} \\ \text{pull-based dominates (UTD is relatively large)} \\ \text{none dominates (UTD is relatively moderate)} \end{cases} \quad \textbf{(3)}$$

Basicamente, o modelo *P&P* é composto por dois algoritmos: *P&P-Push* sendo executado no produtor; *P&P-Pull* sendo implantado no consumidor. O monitoramento considera as necessidades do usuário e as mudanças de estado nos recursos (*i.e.* alternando entre as operações *push*/*pull*). Ambos os algoritmos estão configurados para serem mutuamente exclusivos, evitando que os processos aconteçam concorrentemente (*e.g.* se o algoritmo *pull* for executado, a operação *push* é abandonada para o intervalo de tempo correspondente).

B.7. Advances in Computational Resiliency

O conceito de resiliência computacional refere-se a habilidade de tolerar intrusões quando as informações estão sofrendo algum tipo de ataque (*Information Warfare Attack - IW*). Mesmo se este evento não for detectado, deveria ser possível prosseguir com as operações e alcançar o objetivo almejado pela tarefa (Lee, 2001). A proposta aplica a resiliência computacional em um modelo que combina: a avaliação de ataques em tempo real com a respectiva reconfiguração dos processos, replicação em tempo de execução e uma política de segurança baseada em camadas.

Para dar suporte a este modelo de resiliência, os autores desenvolveram uma tecnologia de programação distribuída que é independente da aplicação. O processo de desenvolvimento incorpora o conceito de resiliência através da utilização de uma biblioteca de transmissão de mensagens. A biblioteca oculta os detalhes dos protocolos de comunicação que são necessários para se conseguir a reconfiguração e a replicação em tempo de execução.

Para tolerar os ataques, as aplicações podem optar por replicar processos do tipo missão crítica, formando assim os grupos de tarefas (*thread groups*; Figura B.8). Porém, esta técnica não garante a continuidade da operação do sistema quando os recursos necessários estão disponíveis em qualquer lugar da rede. Em um sistema real, pode não haver recursos suficientes para replicar todas as tarefas. Neste contexto são necessários métodos baseados em políticas para controlar a replicação.

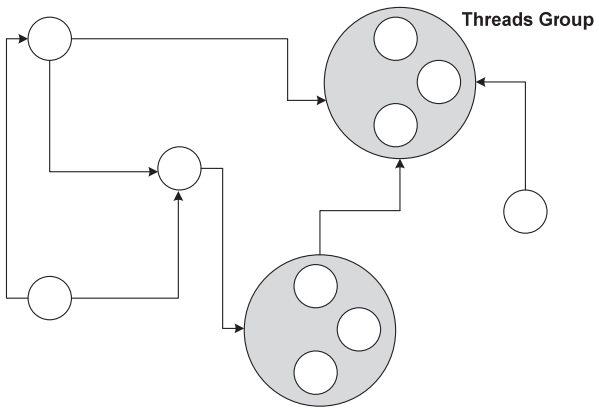

Figura B.8: Replicação de Tarefas. Adaptado de Lee, 2001.

Uma abordagem alternativa é recriar dinamicamente o nível de replicação das tarefas (*threads*) diante de um ataque. Isto assegura que a disponibilidade operacional acabe sendo restaurada, estando sujeito somente as restrições impostas pela disponibilidade dos recursos. Para o cenário abaixo (Figura B.9) o programador da aplicação descreve a estrutura necessária para a tarefa e o nível de resiliência para cada processo. No diagrama existem três tarefas: a primeira e a segunda são resilientes para o nível três, enquanto a terceira é resiliente para o nível dois. A comunicação entre as tarefas na camada de aplicação é substituída pela comunicação de grupo em nível de biblioteca. As tarefas são mapeadas considerando que as réplicas de um mesmo grupo serão alocadas em diferentes processadores.

A biblioteca proposta implementa três tipos de protocolos: *1)* associação no grupo; *2)* recuperação e execução de verificações periódicas no objeto; *3)* controle de fluxo.

1) protocolo de associação no grupo: fornece mecanismos para criar e destruir tarefas, fazendo com que estas entrem ou saiam de um grupo. No início da execução de um programa, grupos são construídos através da criação de réplicas e, em seguida, criam-se grupos utilizando o mecanismo de junção. Durante o processo de falha e recuperação, quando uma tarefa é comprometida, esta é forçada a deixar seu grupo. Nesta etapa, uma

nova réplica é criada e inserida no grupo, assumindo assim o lugar da tarefa afetada.

2) protocolo de recuperação e execução de verificações: fornece uma interface para que rotinas específicas da aplicação possam detectar o comprometimento e implementar o mecanismo de recuperação. O protocolo utiliza a estrutura subjacente para a movimentação do estado da tarefa, recriando o processo afetado e re-configurando a comunicação no grupo. Quando uma verificação ocorre, os membros de cada grupo se comunicam e selecionam uma tarefa que não esteja comprometida para ser a líder do grupo. O líder coordena a replicação, sendo responsável por alterar a comunicação do grupo para que o processo afetado possa ser removido e a nova tarefa possa ser incluída.

3) protocolo de controle de fluxo: assegura a entrega de mensagens de maneira ordenada e confiável entre os grupos, mesmo na presença de algum tipo de comprometimento.

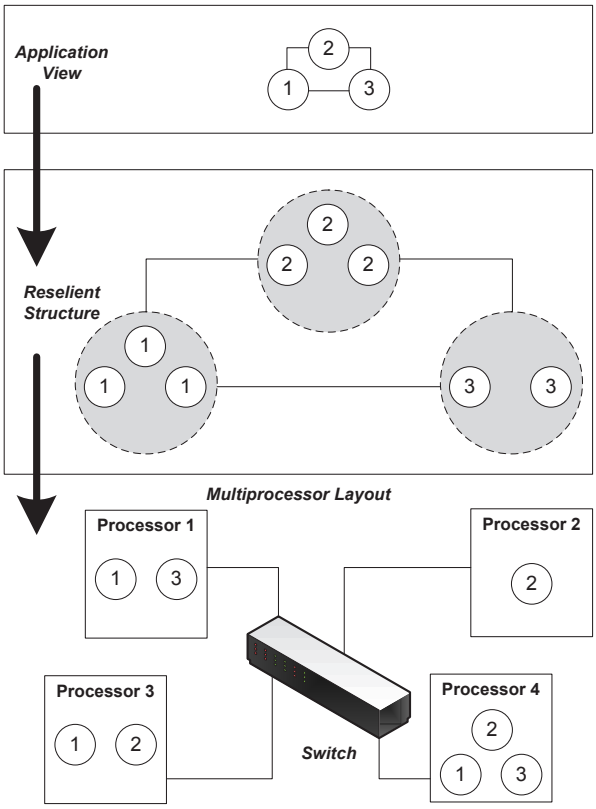

Figura B.9: Resiliência computacional. Adaptado de Lee, 2001.

B.8. Resiliency Policies in Access Control

A proposta apresenta o conceito de políticas resilientes para o contexto de sistemas de controle de acesso (Li, 2006). As políticas necessitam que este sistema seja tolerante a

ausência de usuários. A política resiliente visa assegurar que, mesmo quando situações de emergência façam com que alguns usuários se ausentem, ainda devem existir times independentes de usuários que possuam as permissões necessárias para executar tarefas críticas.

No caso de uma instituição que possui um sistema de controle que necessita de três permissões diferentes para liberar seus fundos: *1)* a aprovação, para um pedido de fundos legítimo; *2)* a emissão do cheque; *3)* o registro da transação. O escritório financeiro desta instituição é composto por um tesoureiro sênior e vários tesoureiros secundários. Em conformidade com o princípio da separação de tarefas, o tesoureiro sênior possui todas as permissões, exceto para registrar a transação. Como a emissão de fundos é uma tarefa crítica, a instituição gostaria de assegurar que, mesmo se alguns tesoureiros (*e.g.* dois, sendo que um pode ser o sênior) estiverem ausentes, o pessoal restante ainda possua privilégios suficientes para liberar os fundos.

Uma política resiliente possui o seguinte formato "*rp[P, s, d, t]*": "*rp*" é uma palavra chave; "$P = \{p_1, ..., p_n\}$" representa um conjunto de permissões; "$s >= 0$" e "$d >= 1$" são inteiros; "*t*" pode ser um inteiro positivo ou o símbolo especial "∞". Um estado de controle de acesso satisfaz a política resiliente se, e somente se, após a remoção de qualquer conjunto de usuários "*s*", ainda existam "*d*" conjuntos mutuamente disjuntos de usuários tal que, cada conjunto, contem não mais que "*t*" usuários, sendo que os usuários agregados dos conjuntos estão autorizados para todas as permissões em "*P*".

Uma política resiliente "*rp[P, s, d, t]*" define uma condição de tolerância a faltas com respeito a certas tarefas críticas. O conjunto "*P*" contém todas as permissões necessárias para executar a tarefa, sendo que a falta que precisa ser tolerada é a ausência de usuários. O parâmetro "*s*" especifica o número de usuários ausentes que se pretende tolerar. O elemento "*d*" é motivado pelo requisito que várias equipes podem ser utilizadas para executar as instâncias da tarefa. Caso somente uma equipe seja necessária, "*d*" pode ser configurado para "*1*". O parâmetro "*t*" define o tamanho limite de cada equipe. Este elemento é motivado por limitações no número máximo de usuários que podem estar envolvidos com a tarefa. Caso esta limitação não exista, "*t*" pode ser configurado para "∞".

Considere o exemplo referente ao controle de acesso no seguinte contexto (Figura B.10): para a emissão de fundos, três tipos de permissões precisam estar de posse dos usuários (*i.e. Endorse*, *Issue*, *Log*). A linha conectando o usuário a uma permissão indica que este possui a autorização. Para uma política resiliente, a configuração seria "$P =$

{Endorse, Issue, Log}". Se a política tiver o valor de "*s = 1*", então, queremos que o sistema seja resiliente a ausência de "*1*" usuário qualquer. Se definirmos o valor de "*d = 2*", significa que precisamos de dois conjuntos tal que, o conjunto agregado de usuários possua todas as permissões. Para um valor de "$t = \infty$", denota que o conjunto de usuários que possui as permissões pode ser de qualquer tamanho.

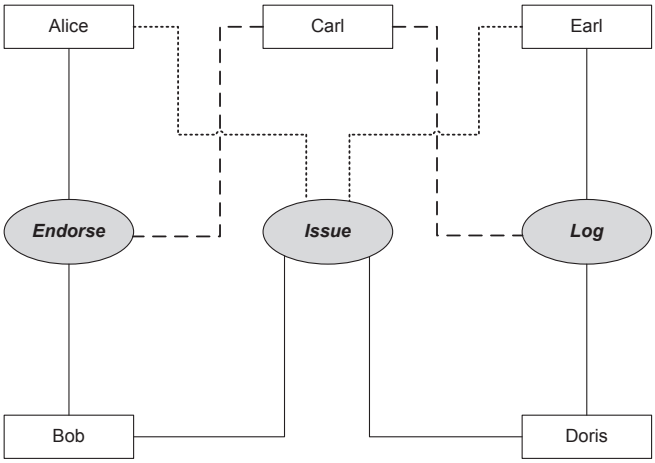

Figura B.10: Controle do Fluxo. Adaptado de Li, 2006.

Observamos no exemplo (Figura B.10) que a política "*rp[P, 1, 2, ∞]*" é satisfeita. Depois de remover "*Alice*", os dois usuários "*Carl*" e "*Earl*" possuem todas as permissões, da mesma maneira que "*Bob*" e "*Doris*". A política "*rp[P, 2, 2, ∞]*" não é satisfeita porque se "*Alice*" e "*Bob*" estiverem ausentes, o usuário que possui o direito de "*Endorse*" é "*Carl*", e este não pode pertencer a dois conjuntos distintos. O caso "*rp[P, 3, 1, ∞]*" não pode ser satisfeito se "*Alice*", "*Bob*" e "*Carl*" estiverem ausentes, porque nenhum outro usuário possui o direito de "*Endorse*". O caso "*rp[P, 1, 1, 2]*" pode ser satisfeito, porém, para "*rp[P, 1, 1, 1]*" não existe nenhum único usuário que possua as três permissões.

O estado do controle de acesso é fornecido por uma relação binária "$UP \subseteq U \times P$", onde "*U*" representa o conjunto de todos os usuários, "*P*" representa o conjunto de todas as permissões. O algoritmo de verificação processa um determinado estado do controle de acesso para remover usuários e permissões irrelevantes. Na segunda etapa, minimiza o número de conjuntos ausentes que precisam ser considerados. Por fim, para cada conjunto ausente "*A*", reduz o problema de verificação se um determinado estado do controle de acesso pode tolerar a remoção de "*A*" usuários para uma determinada instância do problema. O *SAT* (*propositional satisfiability*) ajuda a determinar se as variáveis de um

problema podem ser atribuídas de tal maneira que a fórmula seja avaliada como verdadeira.

O objetivo de uma política com separação de tarefas (*Separation of Duty - SSoD*) é impedir que um grupo de usuários possua muitas permissões. Se um sistema de controle de acesso possuir somente políticas resilientes, então este pode ser satisfeito fornecendo-se todas as permissões para todos os usuários. De maneira similar, se um sistema de controle de acesso possuir somente políticas *SSoD*, então este pode ser satisfeito não fornecendo permissões aleatórias para qualquer usuário, resultando que nenhuma tarefa pode ser executada individualmente. Uma política *SSoD* "*ssod[P,2]*" exige que nenhum usuário individual possua todas as permissões em "*P*". Uma política resiliente "*rp[P, s, d, 1]*" exige a existência de um usuário que possua todas as permissões em "*P*". Claramente, as duas políticas não podem ser satisfeitas simultaneamente.

Adicionalmente, é preciso considerar o problema de consistência de política em nível de tarefa, ou seja, políticas aplicadas a mesma tarefa, ou com a mesma permissão. Neste contexto é preciso verificar a consistência entre políticas com separação estática de tarefas (*SSoD*) e as políticas resilientes. A combinação das duas abordagens é útil em situações onde uma tarefa precisa da prevenção de fraudes e da tolerância a faltas. Para este caso, o artigo propõe a política resiliente com separação de tarefas (*Resilient Separation of Duty - ReSoD*).

Considere o exemplo de um escritório comercial onde a tarefa de emissão de recursos necessita de um conjunto "*P*" com três permissões e temos cinco usuários. Uma política resiliente afirma que na ausência de qualquer utilizador individual, os usuários remanescentes ainda deveriam ter as permissões suficientes para completar a tarefa. Adicionalmente, afim de evitar fraudes, é necessário que a tarefa não seja executada por um único usuário. Estes dois requisitos são capturados por uma política resiliente "*rp [P, 1, 1, ∞]*" e uma política com separação estática de tarefas "*ssod[P, 2]*".

Quando ambas as políticas abordam o mesmo conjunto de permissões, pode-se considerar que estas são intimamente relacionadas umas com as outras (*i.e.* o aumento da resiliência pode vir a um custo de diminuir a separação de tarefas). Sendo assim, utiliza-se uma única política, chamada de *ReSoD*, para se expressar a combinação das restrições.

Uma política *ReSoD* é representada pela *tupla* "*resod [P, k, s]*", onde "*P*" é o conjunto de permissões, "*k*" e "*s*" são inteiros onde "*k > 1*" e "*s >= 0*". O estado do controle de acesso "*UP*" satisfaz a *tupla* "*resod [P, k, s]*" se, e somente se, "*UP*" satisfaz ambos "*ssod[P, k]*" e "*rp [P, s, 1, ∞]*". Considerando uma política *ReSoD*, é necessário

verificar se o estado do controle de acesso "*UP*" satisfaz a política.

B.9. Satisfiability and Resiliency in Workflow Authorization Systems

A proposta aborda o controle de acesso baseado em papéis e relações (*role-and-relation-based access control - R^2BAC*) para sistemas de autorização de fluxo de trabalho (Wang, 2010). No modelo *R^2BAC*, os relacionamentos entre os usuários ajudam a determinar as permissões para se executar determinados passos em um fluxo de trabalho. A abordagem proposta estuda a complexidade do problema, verificando se um conjunto de usuários pode completar um fluxo de trabalho. Adicionalmente, estudam-se os casos onde não é suficiente assegurar que um fluxo de trabalho pode ser finalizado em situações normais (*i.e.* se um fluxo de trabalho pode ser concluído mesmo se alguns usuários estiverem ausentes).

O fluxo de trabalho divide uma tarefa em um conjunto de passos. As políticas em sistemas de autorização de fluxo de trabalho são normalmente definidas utilizando restrições de autorização. O administrador pode especificar, para cada passo, quais são os usuários autorizados. Adicionalmente, podem ser definidas as restrições entre os usuários que executam os diferentes passos no fluxo de trabalho (*e.g.* dois passos quaisquer devem sem executados por usuários diferentes para garantir a separação de tarefas).

O modelo *R^2BAC* é baseado em: *a)* papéis - os passos individuais de um fluxo de trabalho são autorizados para os papéis; *b)* relações - os relacionamentos definidos pelo usuário podem ser utilizados para especificar restrições, ou seja, um usuário autorizado pode ser impedido de executar um passo a menos que este satisfaça as restrições.

A resiliência em sistemas de autorização de fluxo de trabalho aborda dois aspectos: *1)* devido a existência de restrições de autorização, mesmo se um conjunto de usuários estiver autorizado a executar todos os passos de um fluxo, ainda é possível que estes não possam completar a tarefa; *2)* visto que um fluxo de trabalho consiste de uma sequência de passos, e a finalização de todos estes pode levar um tempo relativamente longo, é possível que o conjunto de usuários possa ser alterado durante a execução do fluxo.

O seguinte exemplo ajuda a entender a utilização do modelo *R^2BAC*: em uma instituição acadêmica, apresentar uma proposta de subvenção, através de um serviço específico (*Sponsor Program Services - SPS*), pode ser representado como um fluxo de trabalho com os seguintes passos (Figura B.11): *1)* preparação - um membro da faculdade

escreve uma proposta e envia esta para o escritório do departamento; *2)* orçamento - o funcionário responsável pelas contas prepara o orçamento, verifica as propostas e submete estas para o escritório *SPS*; *3)* revisão especializada - um especialista do escritório revisa a proposta, analisando a conformidade com as regras do programa; *4)* avaliador de contas - o gerente revisa a proposta e o orçamento; *5)* submissão - o gerente submete a proposta para um patrocinador externo.

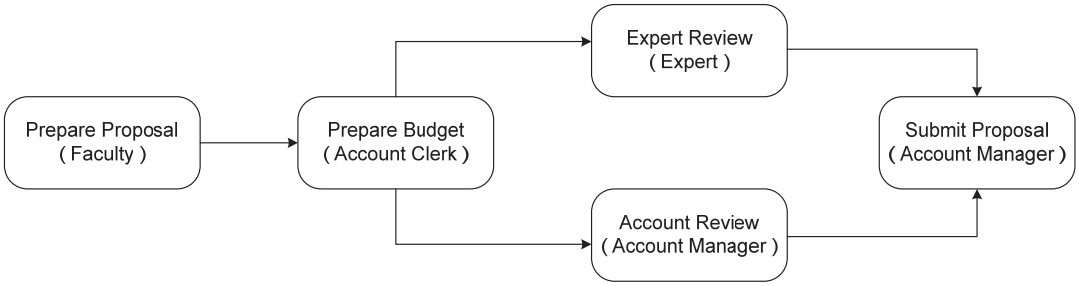

Figura B.11: Fluxo de trabalho. Adaptado de Wang, 2010.

O fluxo de trabalho descrito possui as seguintes restrições: *1)* os eventos "preparação, orçamento, revisão especializada e avaliador de contas" devem ser executados por usuários diferentes; *2)* o funcionário responsável pelas contas, e quem assina a proposta, devem estar no mesmo departamento do membro da faculdade que preparou a proposta; *3)* os revisores não devem possuir conflito de interesses com quem submeteu a proposta; *4)* o gerente de contas que revisa a proposta é responsável por submeter esta para o patrocinador externo.

Analisando o fluxo de trabalho descrito, temos: a restrição *2* reflete certas exigências processuais; a restrição *1* impõe o princípio da separação de tarefas; a restrição *3* segue o esquema da separação de tarefas, exigindo que os passos sejam executados por pessoas diferentes e que ambas não possuam conflito de interesses; a restrição *4* aplica uma política ligada a obrigação, exigindo que duas tarefas sejam executadas pelo mesmo usuário.

O modelo R^2BAC segue de acordo com as definições abaixo. Alguns dos símbolos utilizados são: "U" para os nomes de todos os possíveis usuários, "R" para os papéis, e "B" para as relações binárias.

Definição 1 - Estado do Controle de Acesso: sendo representado por uma *tupla* "*[U, UR, B]*": onde "$U \subseteq U$" é um conjunto de usuários; "$UR \subseteq U \times R$" é a relação de participação do usuário nos papéis; e "$B = \{p_1, ..., p_m\} \subseteq B$" é o conjunto de relações

binárias. Adicionalmente, assume-se que "*B*" contém duas relações binárias pré-definidas "=" e "≠" simbolizando a igualdade e a desigualdade, respectivamente. O estado de um controle de acesso "*[U, UR, B]*" define o ambiente no qual o fluxo de trabalho vai ser executado. O item "*B*" define todas as relações binárias que aparecem em qualquer restrição de fluxos de trabalho a serem executados no ambiente.

Definição 2 - Restrições e Fluxo de Trabalho: o fluxo de trabalho é representado como uma *tupla* "*[S, ≤, SA, C]*": onde "*S*" é o conjunto de passos; "$\leq \subseteq S \times S$" define a ordem parcial dos passos em "*S*"; "$SA \subseteq R \times S$" especifica como os papéis são autorizados para os passos; e "*C*" é o conjunto de restrições, sendo que cada uma destas possui o seguinte formato: *1)* "$[p(s_1, s_2)]$" o usuário que executa "s_1" e o usuário que executa "s_2" devem satisfazer a relação binária "*p*"; *2)* "$[p(\exists X, s)]$" existe um passo "$s' \in X$" de tal forma que "*[p(s', s)]*", ou seja, o usuário que executa " s' " e o usuário que executa "*s*" satisfaz "*p*"; *3)* "$[p(s, \exists X)]$" existe uma passo "$s' \in X$" de tal forma que "$[p(s, s')]$". O item "*SA*" é chamado de autorização para passo referente ao papel (*role-step authorization*) e "$(r, s) \in SA$" indica que os membros do papel "*r*" são autorizados a executar o passo "*s*". De maneira intuitiva, em um fluxo de trabalho "*[S, ≤, SA, C]*", em que "$s_i \leq s_j \ (i \neq j)$" indica que o passo "$s_i$" deve ser executado antes do passo "s_j". Quando um fluxo de trabalho é destinado a completar uma tarefa crítica, é necessário garantir que, mesmo se alguns usuários se ausentarem de maneira inesperada, o fluxo de trabalho ainda pode ser finalizado.

Considere o seguintes cenários: *1)* a execução de um fluxo de trabalho é finalizada em um curto período de tempo - *e.g.* 15 minutos. Neste caso o conjunto de usuários disponíveis para o fluxo de trabalho é estável; *2)* a execução de um fluxo leva um tempo relativamente longo - *e.g.* um dia. Nesta situação, o conjunto de usuários disponíveis pode se tornar cada vez menor ao longo do tempo (*e.g.* os usuários podem se ausentar durante o dia); *3)* a execução do fluxo de trabalho se estende por um longo período de tempo - *e.g.* um único passo executado por dia. Neste caso, o conjunto de usuários que comparecem ao trabalho pode ser diferente de um dia para o outro.

Os cenários supracitados nos proporcionam três níveis de resiliência para sistemas de fluxo de trabalho: *nível 1*, resiliência estática - o estado de um sistema de controle de acesso atende a este nível se o fluxo ainda for satisfatório depois de remover "*t*" usuários; *nível 2*, resiliência decremental - o estado de um sistema de controle de acesso atende a esta categoria se houver sempre uma maneira de completar o fluxo, desconsiderando

quando e quais usuários podem ser removidos; *nível 3*, resiliência dinâmica - o estado de um sistema de controle de acesso atende a este nível se o fluxo de trabalho pode ser completado mesmo se "*t*" usuários estiverem ausentes. Ou seja, a insuficiência é tolerada em qualquer momento durante a execução do fluxo, sendo que o conjunto de usuários ausentes pode variar de tempos em tempos. Para todos os níveis supracitados, assume-se que o número de usuários faltosos, em qualquer período, é limitado pelo parâmetro "*t*".

B.10. LighTS: A Lightweight, Customizable Tuple Space Supporting Context-Aware Applications

A abordagem descreve as extensões empregadas no espaço de *tuplas LighTS* (Picco, 2005). Estas extensões são úteis no desenvolvimento de aplicações que tem conhecimento do contexto (*context aware*). O *LighTS* foi projetado para ser um sistema leve, fornecendo um espaço de *tuplas* com suporte as operações básicas da linguagem *Linda*.

As principais contribuições da proposta são: *1)* apresentar a interface de programação do *LighTS*, descrevendo os mecanismos que suportam a customização e a extensão; *2)* mostrar como estas estruturas podem ser exploradas para atender as necessidades de aplicações que tem conhecimento do contexto. Fazem parte da proposta a descrição de duas semânticas de verificação (*i.e.* baseadas na comparação de intervalos de valores e na lógica *fuzzy*), e características que permitem a agregação de dados em nível de *tuplas*.

As principais otimizações no *LighTS* são: alteração do mecanismo do espaço de *tuplas* - as estruturas de dados que dão suporte as *tuplas* são objetos "*java.util.Vector*", estes são verificados linearmente mediante uma operação de consulta; alteração da semântica de comparação - o modo padrão do *LighTS* depende do método "*equals*", não permitindo subtipos de campos ou *tuplas*. Dados de contexto podem ser armazenados no espaço de *tuplas*, porém, o padrão de comparação baseado em valores exatos é insuficiente para aplicações que tem conhecimento do contexto.

A proposta descreve duas alternativas para a checagem semântica: *1)* comparação por faixa de valores - em aplicações que tem conhecimento do contexto, as consultas precisam analisar se um determinado valor está dentro de uma faixa permitida (*e.g.* a temperatura obtida por um sensor está entre *35ºC* e *38º C*). A abordagem do projeto

LighTS supera esta limitação utilizando a classe "*RangeField*" do pacote "*extensions*"; *2)* comparação *fuzzy*: para a *API* proposta, conjuntos *fuzzy* e suas funções de associação são combinadas no chamado "*termo fuzzy*". Uma coleção de "*termos fuzzy*" representa um "*tipo fuzzy*". A comparação baseada na lógica *fuzzy* necessita do "*tipo fuzzy*" de dois campos para executar a combinação.

O bloco de código abaixo mostra como representar uma faixa de valores. A primeira linha define um novo "*termo fuzzy*", representando uma determinada temperatura - *warm*. O termo é definido por um nome e uma função de associação - neste caso, a função "*PiFunction*" centralizada em *50ºC* e com uma amplitude de *25ºC*. A segunda linha cria um novo "*tipo fuzzy*", vinculando este ao termo criado previamente. Um "*tipo fuzzy*" é caracterizado por um nome e dois parâmetros que delimitam seu domínio.

```
FuzzyTerm ft =
            new FuzzyTerm("warm", new PiFunction (50.0f, 25.0f);
      FloatFuzzyType temp =
            new FloatFuzzyType("Temperature", -100, 100).addTerm(ft);
```

Integrando a lógica *fuzzy* e o espaço de *tuplas*: um campo *fuzzy* pode ser incluído em um modelo convencional (*template*). Neste caso, o método de comparação avalia as *tuplas*, retornando verdadeiro somente se o valor de associação do dado encontrado no campo for mais alto que um limiar pertinente a respectiva função.

Uma necessidade comum em aplicações que tem conhecimento do contexto é a habilidade de trabalhar com informações agregadas. O projeto *LighTS* aborda este problema desacoplando a representação das *tuplas* armazenadas no espaço, das entradas manipuladas pela aplicação. Como exemplo, considere o acesso a *tuplas* conforme o seguinte modelo "*p = [?UserID, ?int, ?int]*" como se estas fossem "*tuplas virtuais*" com o seguinte formato "*p' = [?UserID, ?int]*". Tendo como segundo campo da *tupla* " *p'* " a soma dos dois últimos campos da *tupla* " *p* ". Se isto fosse possível, a "*tupla virtual*" "*t = [?UserID, 50]*" poderia ser compatível com as *tuplas* reais "*['u15', 20, 30]*" e "*t = ['u23', 1, 49]*".

B.11. A Tuple Space Service for Large Scale Infrastructures

A proposta descreve o desenvolvimento de um espaço de comunicação para o

Globus Toolkit utilizando Serviços *Web* (Capizzi, 2008). O espaço de *tuplas Grinda* foi projetado como um *framework* de coordenação genérico para aplicações e serviços. Esta abordagem pode ser utilizada para a coordenação de tarefas ou para armazenar informações referentes ao aproveitamento de recursos.

A arquitetura do serviço *Grinda* é composta por dois módulos principais (Figura B.12): *1)* cliente - esconde os detalhes de comunicação com o servidor, visando simplificar o desenvolvimento das aplicações. Este módulo contêm três categorias de objetos: *a)* objetos utilizados diretamente pela aplicação para operar o espaço, *b)* objetos aplicados na serialização/de-serialização de dados para o formato *XML*, *c)* objetos empregados para simular operações síncronas; *2)* servidor - contém a lógica responsável por armazenar as entradas e implementar as operações no espaço de *tuplas*. O componente *GrindaContext* é utilizado para criar e gerenciar o espaço de *tuplas* conforme as configurações do ambiente. Neste módulo, o contexto de operação é separado da lógica de armazenamento.

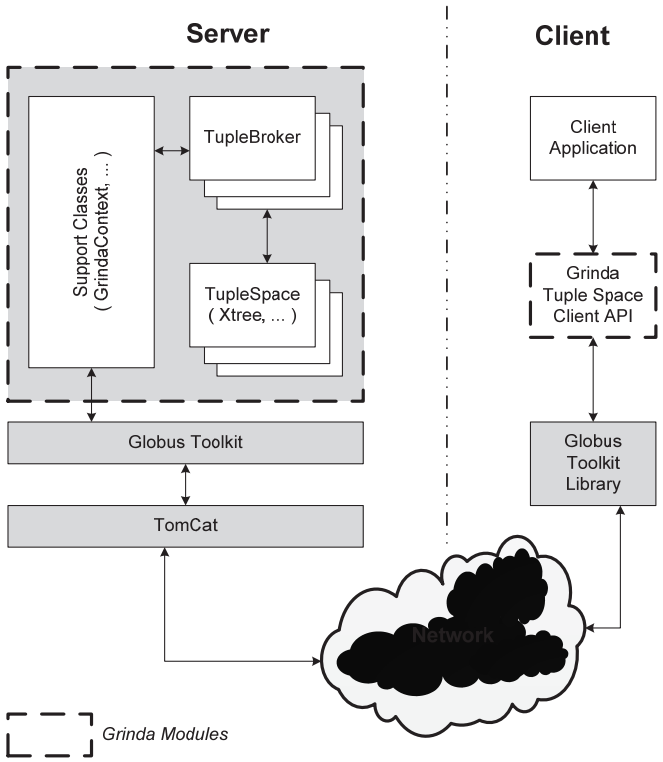

Figura B.12: Arquitetura do *Grinda*. Adaptado de Capizzi, 2008.

O módulo no lado servidor é um serviço *web* implementado em *Java*. Este utiliza o *framework WSRF* (*Web Services Resource Framework*) fornecido pelo *Globus Toolkit*. O componente no lado cliente é projetado para ser fracamente acoplado com o serviço,

permitindo a utilização de diferentes linguagens de programação.

O modelo utilizado na proposta, baseado no *JavaSpaces*, proporciona os seguintes benefícios: simplifica o desenvolvimento de aplicações, porque não necessita de código adicional para mapear dados para as *tuplas*; é facilmente traduzido para mensagens *XML*; e necessita de menos esforços no processo de serialização. Com este modelo, objetos normais podem ser diretamente inseridos ou removidos do espaço, sem a necessidade de uma conversão manual.

B.12. An Agent-Based Approach For Grid Computing

A proposta apresenta uma *grid* computacional baseada em agentes (*Agent based Computational Grid - ACG*). O sistema implementa o gerenciamento uniforme de recursos e serviços computacionais, fornecendo aos usuários uma interface consistente para o acesso aos serviços (Chunlin, 2003).

Os agentes presentes no ambiente são registrados em um serviço de gerenciamento de *grid*. Os membros da *grid* se comunicam utilizando um espaço de comunicação implementado como um espaço de *tuplas*. Esta abordagem visa fornecer um sistema de descoberta de recursos flexível e eficiente, permitindo a colaboração dos agentes que fazem parte da *grid*.

O objetivo do trabalho é formar uma comunidade de agentes em um ambiente completamente distribuído. O *ACG* visa fornecer uma camada de acesso uniforme para uma grande variedade de serviços da *grid* (*e.g.* bibliotecas, aplicações). Em essência, o *ACG* é responsável pelo fornecimento de serviços e a alocação de recursos entre os membros. Todas as entidades do ambiente (*e.g.* recursos computacionais e serviços) podem ser representadas como agentes da *grid*, sendo que cada uma destas é registrada com um gerente de serviço. O *ACG* possui um *framework* que combina funcionalidades de um *grid* com a arquitetura de um sistema de agentes, este visa fornecer os recursos da *grid* para os respectivos usuários.

Para permitir que agentes do serviço de *grid* sejam adicionados, atualizados ou removidos da rede, um Serviço de Registro da *Grid* é aplicado (*GSR*; Figura B.13). O *GSR* possui um banco de dados que pode ser utilizado para localizar os serviços registrados. O serviço *GSR* fornece as seguintes funcionalidades: mecanismo de inscrição para todos os nodos; interface para pesquisar os serviços registrados; estrutura para

assegurar que o banco de dados armazena informações atualizadas. O Serviço de Registro utiliza um Gerente de Serviço (*GSM*) presente em cada nodo da *grid*. Todo nodo possui um gerente individual, sendo assim, o sistema completo é definido pela interação entre os *GSMs*.

As principais ações envolvidas no processo de descoberta dos serviços em *grid* são: registro - o *GSR* é ativado quando os agentes registram sua presença através da publicação das descrições do serviço; pesquisa - fornece um mecanismo para descobrir serviços de *grid* para os agentes solicitantes.

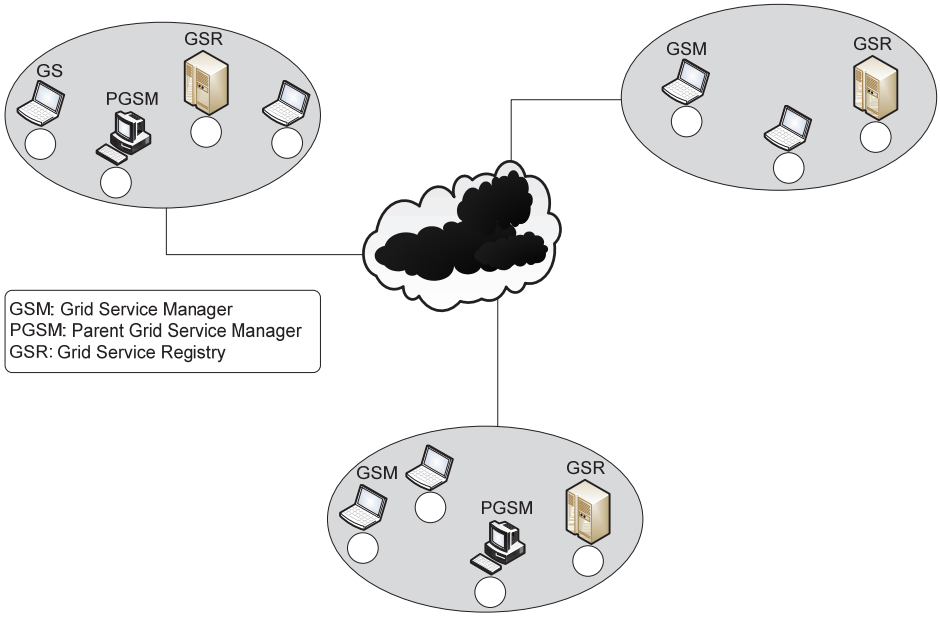

Figura B.13: Ambiente de *Grid ACG*. Adaptado de Chunlin, 2003.

No esquema proposto, os agentes interagem através do espaço de *tuplas*. O espaço de comunicação suporta mensagens assíncronas e a cooperação entre grupos assíncronos. No sistema *ACG*, os membros da *grid* são empacotados como agentes móveis, necessitando se comunicar sobre a rede. Porém, a comunicação com o agente acarreta problemas causados pela mobilidade do agente. Enviar mensagens para um agente remoto necessita do endereço da entidade receptora. Porém, está abordagem apresenta dificuldades se o receptor se mover para um novo local antes da mensagem ser entregue.

Para um serviço de *grid* se tornar parte do *ACG*, este deve ser conhecido pelo *GSM*. O processo de cadastro torna a descrição do serviço disponível para o gerente de serviço da *grid*. Este evento é utilizado para registrar serviços representados por agentes da *grid* (*GSA*). O *GSA* é a entidade que disponibiliza seus serviços para os outros membros.

O processo de pesquisa é utilizado para localizar os serviços publicados (Figura B.14). A descoberta permite a um solicitante encontrar anúncios na unidade da *grid*, esse procedimento é implementado por três tipos de componentes: agentes de serviço da *grid* (*GSA*); agentes que solicitam os serviços (*SRA*); entidades que fazem parte do próprio serviço de descoberta da *grid*. O *GSM* pode encontrar outros gerentes e verificar quais serviços estão registrados.

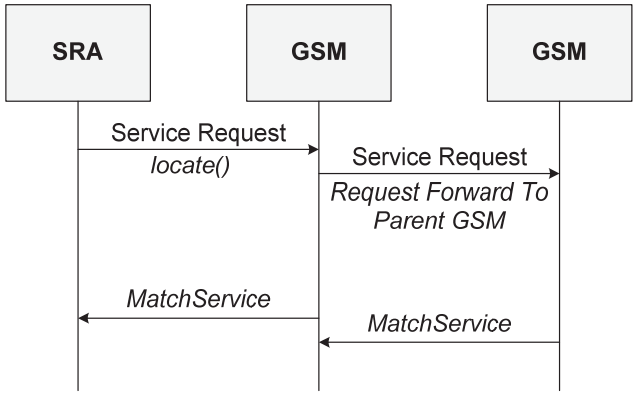

Figura B.14: Processo de descoberta de serviço. Adaptado de Chunlin, 2003.

A interação via espaço de *tuplas* pode ser utilizada na comunicação indireta entre as entidades do ambiente. O modelo generativo (Apêndice A) introduz um paradigma de comunicação desacoplado, sendo baseado na abstração de um espaço de *tuplas*. Ou seja, o agente que insere as *tuplas* no espaço é desconhecido, com isto, os parceiros do processo de comunicação permanecem anônimos uns para os outros.

B.13. Context Information Provisioning in Tuple Spaces

A proposta explora o provisionamento de contextos utilizando o espaço de *tuplas* (Salvador, 2009). Um assunto referente a aplicação da computação pervasiva/ubíqua é que esta deve ser sensível ao contexto onde é executada. Tendo em vista que muitas informações de contexto são externas ao sistema, os sistemas que provêem suporte precisam gerenciar os dados e encaminhá-los para a aplicação.

O esquema de gerenciamento necessita assegurar todo o ciclo de vida das informações de contexto. Ou seja: recuperar dados brutos do sensor; combinar as leituras efetuadas em diferentes fontes; entregar em tempo hábil a informação resultante para as aplicações; e eliminar dados desatualizados. A natureza dinâmica das tarefas de

gerenciamento e o número de partes envolvidas tornam necessário o baixo acoplamento.

Os espaços de contexto (*Context Spaces - CS*) são resultantes da adaptação de espaços de *tuplas* para as necessidades de governar as informações do ambiente. Os *CS* são compostos de um espaço de compartilhamento e um mecanismo de gerenciamento. Esta estrutura é responsável por coordenar o acesso às *tuplas* utilizando operadores de contexto.

Os operadores de contexto (*Context Operators - CO*) são processos que interagem com o espaço de *tuplas* subjacente a um *CS* para criar, ler, modificar e apagar as entradas. Cada *CO* pode manipular tipos específicos de *tuplas* em suas operações de entrada e saída. As entradas que podem ser manipuladas são definidas em tempo de compilação, não sendo alteradas em tempo de execução. Quando o tipo de entrada, de um determinado operador de contexto, corresponde a saída de outro *CO*, ambos os operadores são considerados ligados.

O *CS* define três tipos de operadores (Figura B.15): produtores de contexto (*Context Producers - P*) - fazem a interface com fontes de informação externas e periodicamente produzem *tuplas* que refletem os dados recuperados; transformadores de contexto (*Context Transformers - T*) - fazem a leitura das entradas e executam a modificação ou escrita de novas *tuplas* de acordo com o resultado do processamento; consumidores de contexto (*Context Consumers - C*) - consomem as entradas criadas por outros *COs* e oferecem alguma funcionalidade para os usuários finais.

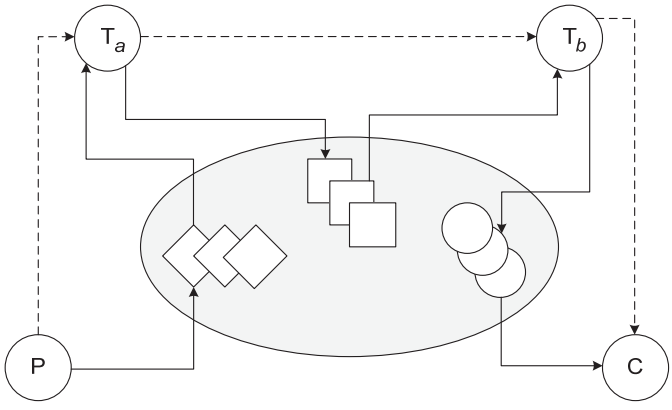

Figura B.15: Espaço de Contexto. Adaptado de Salvador, 2009.

O cenário acima (Figura B.15) ilustra um espaço simples onde quatro operadores de contexto (*i.e. P, T_a, T_b, C*) estão ligados, formando assim um fluxo contínuo entre as *tuplas* de informação. A elipse central representa o espaço de *tuplas*, as linhas pontilhadas

representam uma ligação conceitual entre uma série de operadores de contexto - uma cadeia de contexto.

As cadeias de contexto expõem as ligações conceituais entre os *COs* (Figura B.16). Devido a natureza fracamente acoplada dos paradigmas de comunicação e coordenação baseado em espaço de *tuplas*, os operadores nunca são conectados aos demais. Ao invés disto, estes fazem uso da indireção provida pelo espaço de contexto para trocar, de maneira anônima, os tipos compatíveis de *tuplas*.

Os *CSs* tipicamente possuem várias cadeias simultâneas, sendo que um operador de contexto pode fazer parte de uma ou mais destas cadeias. O fenômeno de entrelaçamento das cadeias de contexto permite a reutilização transparente das *tuplas*, sendo esta uma das principais características dos espaços de contexto.

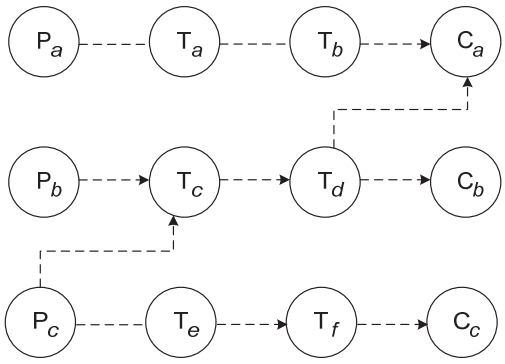

Figura B.16: Cadeias de Contexto. Adaptado de Salvador, 2009.

O exemplo acima (Figura B.16) ilustra várias cadeias de contexto, sendo que as *tuplas* de informação fluem unidirecionalmente a partir dos produtores para os consumidores de contexto. A quantidade de caminhos que conectam um conjunto de produtores a um conjunto de consumidores representa o número total de cadeias distintas nestes espaço. Neste exemplo, existem seis diferentes caminhos conectando produtores e consumidores, resultando em seis cadeias únicas as quais podem ser expressadas utilizando a seguinte notação:

```
CC1 = {(Pa,Ta),(Ta,Tb),(Tb,Ca)}
CC2 = {(Pb,Tc),(Tc,Td),(Td,Ca)}
CC3 = {(Pb,Tc),(Tc,Td),(Td,Cb)}
CC4 = {(Pc,Tc),(Tc,Td),(Td,Ca)}
CC5 = {(Pc,Tc),(Tc,Td),(Td,Cb)}
CC6 = {(Pc,Te),(Te,Tf),(Tf,Cc)}
```

Tuplas de contexto encapsulam informações para os *COs* acessarem e modificarem. O tipo das informações que podem ser encontradas nestas entradas não é pré-definido, ou seja, depende das aplicações construídas encima de um determinado espaço de contexto (*e.g.* as *tuplas* podem armazenar leituras de um sensor em modo bruto).

A separação de interesses fornecida pelo baixo acoplamento pode levar a um sistema distribuído escalável e robusto. Este pode ser capaz de enfrentar falhas e utilizar os recursos computacionais disponíveis de maneira eficiente. A combinação de pesquisas existentes nos campos da agregação-disseminação de contextos, juntamente com a comunicação baseada em espaço de *tuplas*, podem contribuir para melhorar o estado da arte na área da computação em nuvem.