UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

CARLOS RANIERY PAULA DOS SANTOS

Uso de Serviços de Presença em Sistemas P2P de Gerenciamento de Redes

Dissertação apresentada como requisito parcial
para a obtenção do grau de
Mestre em Ciência da Computação

Prof[a]. Dr[a]. Maria Janilce B. Almeida
Orientador

Porto Alegre, Fevereiro de 2008

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Santos, Carlos Raniery Paula dos

Uso de Serviços de Presença em Sistemas P2P de Gerenciamento de Redes / Carlos Raniery Paula dos Santos. – Porto Alegre: PPGC da UFRGS, 2008.

92 f.: il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2008. Orientador: Maria Janilce B. Almeida.

1. Gerenciamento de redes. 2. Peer-to-peer. 3. Servicos de presenca. 4. Notificações. I. Almeida, Maria Janilce B.. II. Título.

“To accomplish great things,
we must not only act but also dream,
not only plan but also believe.”
— Anatole France

AGRADECIMENTOS

Agradeço antes de tudo à minha esposa Emanuella, pela prova incondicional de amor que me deste quando aceitou me acompanhar nesta jornada. Você foi a pessoa que, com seu sorriso, ternura, voz, carinho e brincadeiras, tornou estes dois anos os mais felizes que eu já tive em minha vida e ainda assim, consegue me surpreender diariamente, com pequenos gestos, curtas frases e palavras sinceras. E que apesar de não aguentar mais ouvir falar em serviços de presença, me acompanhou todos os dias (e noites) durante a escrita desta dissertação. Como eu digo, desde que te vi entrando naquela sala, sabia que era você com quem eu sempre sonhei e termino dizendo: "Te amei, te amo, te amarei eternamente!".

Agradeço aos meus queridos pais, Amador e Maria, e avós, Francisco e Iadê, pelo exemplo de vida e pelo imenso carinho que sempre tiveram por mim durante toda a minha vida. Apesar da enorme distância (4250 Km), sempre tentaram me passar calma e confiança para que eu conseguisse alcançar todos os meus sonhos. Agradeço de coração por tudo o que fizerem por mim. Muitas saudades!

Agradeço à minha orientadora Maria Janilce Bosquiroli Almeida, por ter me auxiliado no desenvolvimento deste trabalho e pelo acompanhamento durante o mestrado. Um agradecimento muito especial aos professores Lisandro Granville e Luís Cechin pelos ensinamentos, conselhos, dedicação e paciência. Gostaria de agradecer também aos amigos e professores do Grupo de Redes de Computadores, Luciano Paschoal Gaspary, Juergen Rochol e a professora Liane Tarouco, pelos ensinamentos, auxílio e competência nas aulas ministradas.

Agradeço também às primeiras pessoas que estiveram presentes no início da minha vida acadêmica, professora Rossana Andrade e os professores Cidcley Teixeira, Carlos Hairon e ao grande amigo Windson Carvalho. Pessoas estas, que sempre serão lembradas por todo apoio que me deram e que me possibilitou, hoje, estar terminando mais esta etapa da minha vida.

Agradeço aos meus colegas de mestrado Ewerton Salvador, Giovane Moura, Bruno Carreiro, tanto pela ajuda, como pelos momentos de descontração (/var e *Coke Time*). Um agradecimento especial à Clarissa Marquezan e ao André Panisson, pelo auxilio dado no desenvolvimento deste trabalho e pela escrita dos artigos internacionais.

Gostaria de agradecer também aos demais colegas, professores e funcionários do Instituto de Informática pela prestatividade e ajuda dispensada. Em especial ao Luís Otávio, por todas autorizações que me deu para usar os laboratórios da graduação nas eternas noites de testes.

Por fim, a todos que de forma direta ou indireta me apoiaram, auxiliaram e contribuíram não só na realização desta dissertação, mas em todas as atividades realizadas durante o mestrado, meus sinceros agradecimentos. Muito obrigado a todos!

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

TI	Tecnologia da Informação
WS	Web Services
P2P	Peer-to-Peer
SNMP	Simple Network Management Protocol
RMON	Remote Monitoring
IETF	Internet Engineering Task Force
CMIP	Common Management Information Protocol
PDP	Policy Decision Point
PEP	Policy Enforcement Point
IM	Instant Messaging
VO	Virtual Organization
NAT	Network Address Translation
MbD	Management by Delegation
SIMPLE	SIP for Instant Messaging and Presence Leveraging Extensions
XMPP	Extensible Messaging and Presence Protocol
WV	Wireless Village
PRIM	Presence and Instant Messaging Protocol
PAM	Presence and Availability Management
APEX	Application Exchange
ISO	International Organization for Standardization
FCAPS	Fault, Configuration, Accounting, Performance and Security
MIB	Management Information Base
IP	Internet Protocol
TCP	Transmission Control Protocol
MLM	Mid-Level Manager
TLM	Top-Level Manager

PBNM	Policy Based Network Management
PDP	Policy Decision Point
PEP	Policy Enforcement Point
API	Application Programming Interface
SOAP	Simple Object Access Protocol
UFRGS	Universidade Federal do Rio Grande do Sul
IM&P	Instant Messaging and Presence
RFC	Request for Comments
OMA	Open Mobile Alliance
IMPS	Instant Messaging and Presence Service
PA	Presence Agent
PUA	Presence User Agent
UA	User Agent
W3C	World Wide Web Consortium
SIP	Session Initiation Protocol
XML	eXtensible Markup Language
BEEP	Blocks Extensible Exchange Protocol
JCP	Java Community Process
JSR	Java Specification Request
CPIM	Common Presence and Instant Messaging
SMTP	Simple Mail Transfer Protocol
IRC	Internet Relay Chat
SMS	Short Message Service
CLI	Command Line Interface
CLP	Command Line Protocol
CSP	Client-Server Protocol
SSP	Server-Server Protocol
SMCN	Server Mobile Core Network
JID	Jabber Identificator
VoIP	Voice over IP
PIDF	Presence Information Data Format
RPID	Rich Presence Information Data Format
XCAP	XML Configuration Access Protocol
ICMP	Internet Control Message Protocol

Java ME	Java Micro Edition
Java SE	Java Standard Edition
OV	Object Value
NIST	National Institute of Standards and Technology
SER	Sip Express Router
NTP	Network Time Protocol
CSV	Comma Separated Values

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

O uso de notificações para informar o estado dos dispositivos e serviços da rede possui um papel crucial para garantir o correto funcionamento da rede e assim evitar custos desnecessários. Serviços de presença, que podem ser desenvolvidos utilizando o mecanismo de notificações, foram planejados com o objetivo de fornecer meios para que informações de estado sejam entregues corretamente aos interessados. Contudo, até agora, o uso destes serviços no contexto do gerenciamento de redes ainda permanece desconhecido. Sendo assim, os principais objetivos desta dissertação são investigar a viabilidade da utilização de serviços de presença em gerenciamento de redes e identificar os cenários mais adequados para sua utilização. Para atingir tais objetivos nesta dissertação é proposta uma arquitetura de gerenciamento que utiliza serviços de presença como mecanismo de notificação em processos de gerenciamento de redes. Com base nesta arquitetura foi desenvolvido um protótipo, onde alguns dos principais serviços de presença existentes na literatura foram integrados. Este protótipo foi então incorporado à estrutura do sistema de gerenciamento de redes ManP2P, adicionando à este funcionalidades para o envio de notificações. Por fim, o protótipo desenvolvido, junto com os serviços de presença selecionados, foram avaliados em um ambiente de testes, onde se procurou observar o comportamento de cada serviço de presença em diferentes situações. Os resultados obtidos indicam que o uso de serviços de presença como mecanismo de notificação no contexto do gerenciamento é viável, contudo, é necessário escolher qual serviço de presença que melhor atende às necessidades dos administradores. Além disso, os resultados indicam quais são os serviços mais adequados para cada situação, permitindo assim que as informações de presença sejam encaminhadas da forma mais efetiva possível aos interessados.

Palavras-chave: Gerenciamento de redes, peer-to-peer, servicos de presenca, notificações.

Employing Presence Services in P2P-Based Network Management Systems

ABSTRACT

The use of notifications to report the status of underlying communication networks has a crucial impact on the performance of the managed network itself. Presence services, which are implemented using notification messages, have been designed with the objective to provide ways of delivering accurate presence information to interested parties. However, up to now, the use of presence services in the network management discipline has not been properly addressed. Therefore, the main objectives of this work are, to investigate the viability of presence services use and to identify the best situations where this mechanism can be used. In order to accomplish the investigation, this work proposes a network management architecture that employs presence services as a notification mechanism into traditional network management processes. Based on this architecture, a prototype was built and some of the main presence services available in the literature have been integrated. This prototype was integrated into the structure of a P2P-based network management system named ManP2P, adding features for sending notifications. At last, the developed prototype and the selected presence services, have been evaluated through a set of tests performed over a testbed. The obtained results present the availability of using presence services in the context of network management. Nevertheless, they also show that is necessary to match the best presence services with the administrator's requirements. Moreover those results characterize each services are more suitable for which network management situation, thus, helping to improve the notification process.

Keywords: Network Management, Peer-to-Peer, Presence Services, Notifications.

1 INTRODUÇÃO

O aumento na complexidade que as redes de computadores têm apresentado nos últimos anos vem obrigando os pesquisadores a olharem com mais atenção para a tarefa de gerenciamento de redes. Este aumento deve-se principalmente ao surgimento de novos dispositivos e serviços que possuem cada vez mais funcionalidades e comportamentos particulares. Além disso, tem-se observado uma constante evolução nos ambientes corporativos. Infra-estruturas tradicionais de TI, onde os recursos (*i.e.*, *hardware* e *software*) eram gerenciados localmente por um única empresa, estão dando lugar a cenários onde tais recursos são gerenciados de uma forma distribuída, por parceiros isolados geograficamente e interligados através da Internet. Nesse contexto, o gerenciamento de redes apresenta-se como uma atividade que tem como objetivo manter as infra-estruturas de TI funcionando de maneira adequada evitando assim prejuízos financeiros. Dessa forma, modelos e ferramentas de gerenciamento têm sido propostos por pesquisadores para auxiliar as corporações a administrar de forma mais adequada suas infra-estruturas de rede.

Preocupado com a necessidade de gerenciamento, o *Internet Engineering Task Force* (IETF) propôs o *framework Simple Network Management Protocol* (SNMP) (HARRINGTON; PRESUHN; WIJNEN, 2002), que tem como objetivo fornecer funcionalidades de monitoração e configuração para o gerenciamento de redes. O SNMP foi definido na década de 80, revisto nos anos 90 e ainda hoje se apresenta como a solução *de facto* para o gerenciamento de redes TCP/IP. Contudo, soluções convencionais como o SNMP não são mais suficientes considerando as necessidades de gerenciamento das modernas redes de computadores (SOLDATOS; ALEXOPOULOS, 2007). Por exemplo, como a Internet se tornou uma rede fragmentada com a introdução de NATs (*Network Address Translation*) e *firewalls*, os administradores de redes são impedidos de descobrir e acessar dispositivos internos à outros domínios administrativos. Isto impossibilita, por exemplo, o gerenciamento de ambientes dinâmicos e de múltiplos domínios, como os formados pelas organizações virtuais (*Virtual Organization* – VOs) (FOSTER; KESSELMAN; TUECKE, 2001).

Nesse contexto, a comunidade de gerenciamento de redes começou a investigar o uso de tecnologias que originalmente foram definidas em outras áreas, como por exemplo Web Services (WS) (CURBERA et al., 2002) e Peer-to-Peer (P2P) (ANDROUTSELLIS-THEOTOKIS; SPINELLIS, 2004). Diversas pesquisas (MOURA et al., 2007) (GRANVILLE et al., 2005) já vêm mostrando que tais tecnologias podem aprimorar sensivelmente os processos de gerenciamento de redes. O uso de tecnologias de Web Services condiz com a atual realidade encontrada em muitas empresas, onde percebe-se uma mudança na maneira como os negócios são conduzidos. Empresas totalmente integradas estão sendo substituídas por redes de negócios, nas quais cada participante oferece serviços especializados a outros parceiros. Além disso, o uso de Web Services permite que sejam

criados sistemas de gerenciamento mais sofisticados a partir da composição de serviços (PELTZ, 2003) utilizando modelos de orquestração e coreografia de WS.

Tecnologias P2P, por sua vez, também apresentam características interessantes que podem ser aproveitadas no gerenciamento de redes, por exemplo: distribuição, escalabilidade e alta disponibilidade. Outra grande vantagem na utilização de tecnologias P2P é que estas possibilitam que a tarefa de gerenciamento seja realizada ao longo de diferentes domínios administrativos, atendendo assim às necessidades dos administradores em face das modernas redes de computadores. Isto é possível porque tais tecnologias possuem facilidades para permitir a troca de mensagens, mesmo quando há elementos que possam dificultar tal comunicação (*e.g.*, NATs e *firewalls*). Uma infra-estrutura P2P para gerenciamento de redes pode ser vista como um conjunto de *peers* interligados e que constituem um *overlay* de gerenciamento a partir de onde as ações de gerenciamento são executadas.

Apesar das vantagens apresentadas pelas novas tecnologias quando utilizadas no gerenciamento de redes, o uso das mesmas costuma vir associado, se comparadas às soluções de gerenciamento tradicionais, com um consumo maior de recursos (*e.g.*, banda de rede, processamento). No intuito de diminuir este consumo de recursos, mecanismos já utilizados no gerenciamento de redes no passado podem ser re-empregados junto às novas tecnologias. Um destes mecanismos é o de notificações (CARZANIGA; ROSENBLUM; WOLF, 2001), que pode ser utilizado para informar aos administradores o estado dos recursos da rede, permitindo que estes administradores tenham uma imagem consistente dos recursos gerenciados. Este mecanismo pode ser utilizado também para reportar falhas que ocorram em tais recursos, abrangendo duas (falhas e desempenho) das cinco áreas funcionais do gerenciamento de redes definidas pela *International Organization for Standardization* (ISO), cuja classificação chama-se FCAPS (*Fault, Configuration, Accounting, Performance and Security*) (LAKSHMI, 1998). Com o uso de notificações, o tradicional processo de *polling*, que sobrecarrega a rede com mensagens de requisição-resposta, pode ser substituído por mensagens assíncronas que são geradas e enviadas aos interessados apenas quando necessário. Uma forma de se implementar o suporte a notificações é utilizando o paradigma *publish/subscribe* (FELBER; GUERRAOUI; KERMARREC, 2003), onde produtores publicam informações para consumidores que registram interesse em recebê-las. Esse processo diminui sensivelmente o número de mensagens de gerenciamento que circulam na rede, melhorando assim sua eficiência.

Além da escolha de um mecanismo adequado para informar o estado dos recursos da rede, os sistemas de gerenciamento devem possuir características que garantam o envio correto das informações aos administradores interessados. Serviço de presença é um mecanismo que pode ser utilizado para este propósito. Tais serviços foram concebidos inicialmente em sistemas de mensagens instantâneas (*Instant Messaging* – IM) (*e.g.*, ICQ e Microsoft MSN) e que se tornaram aplicações muito populares entre os usuários da Internet. Os serviços de presença, nesse contexto, são usados para informar o estado (*e.g.*, *online*, *offline*, falando no telefone) de usuários a um grupo remoto de pessoas, normalmente representado por uma lista de contatos. Além do contexto das aplicações de IM, o conceito dos serviços de presença pode ser freqüentemente encontrado em outros cenários com objetivos variados, por exemplo em sistemas P2P. Neste caso, as informações de presença são utilizadas para informar detalhes de operação do *overlay* P2P (*e.g.*, conectividade dos *peers*, recursos compartilhados, etc.) aos *peers* que fazem parte deste *overlay*.

Como já citado, o uso de tecnologias P2P como ferramenta para gerenciamento de redes já vem sendo investigado no ambiente acadêmico, mas não existem ainda pesqui-

sas que explorem os serviços de presença como possível mecanismo para notificar os administradores sobre o estado dos elementos da rede gerenciada. Por exemplo, um administrador interessado no estado de algum dispositivo poderia registrar no serviço de presença seu interesse em ser notificado e o serviço de presença se encarregaria de encaminhar as notificações contendo o estado do dispositivo ao administrador, liberando-o para desempenhar outras atividades.

Assim, considerando o desconhecimento do comportamento dos serviços de presença quando empregados como mecanismo de notificação em processos de gerenciamento de redes, esta dissertação tem como objetivo fazer uma investigação do uso de serviços de presença em conjunto com sistemas P2P para o gerenciamento de redes. Para tanto, foi proposta uma arquitetura de gerenciamento baseada em tecnologias P2P e que utiliza serviços de presença como mecanismo de notificação. A arquitetura proposta possibilita que serviços de presença tradicionais sejam adicionados à sua estrutura permitindo que o serviço mais adequado para diferentes situações seja utilizado. A responsabilidade de decidir qual é o serviço de presença mais adequado pertence a um dos elementos definidos na arquitetura (*i.e.*, *broker*), que utiliza parâmetros definidos por um operador humano (que desempenha o papel de controlador do sistema) para fazer esta escolha. Por exemplo, um controlador pode informar que em determinada situação deve ser utilizado um serviço de presença capaz de enviar notificações mais rapidamente, enquanto que em outra situação deve ser utilizado um que gere menos tráfego.

A arquitetura proposta foi implementada através da API para criação de serviços de gerenciamento disponível no ManP2P (PANISSON et al., 2006), que é um sistema de gerenciamento de redes desenvolvido pelo Grupo de Redes da UFRGS, o qual combina funcionalidades de gerenciamento de redes e distribuição de tarefas por diferentes domínios administrativos. O ManP2P foi implementado utilizando JXTA, que é um *framework* para o desenvolvimento de aplicações P2P criado pela Sun Microsystems e atualmente mantido por um grupo independente. O JXTA também foi utilizado no desenvolvimento de um novo serviço de presença, que foi integrado à implementação da arquitetura e que utiliza exclusivamente os meios de comunicação disponíveis no JXTA para notificar o estado dos recursos gerenciados. Tanto a implementação da arquitetura, como o protótipo de serviço de presença foram então incorporados ao ManP2P, adicionando à este funcionalidades para o envio de notificações.

Para selecionar quais os serviços de presença que poderiam ser adicionados à implementação da arquitetura proposta, foram estudados 6 dos principais serviços disponíveis na literatura. Foram então estabelecidos critérios que possibilitaram uma comparação entre os serviços de presença, evidenciando as características de cada um e possibilitando o levantamento de requisitos que os serviços deveriam possuir que os qualificariam para serem utilizados no gerenciamento de redes. Os serviços selecionados, junto ao protótipo de serviço de presença desenvolvido, foram avaliados através de medições de desempenho para que o comportamento de cada um fosse observado e, assim, fosse possível decidir quando seria melhor empregar um serviço ou outro, ajudando os operadores de redes a configurarem mais adequadamente o sistema proposto de acordo com as suas necessidades.

Esta dissertação também preocupou-se em manter o suporte a diferentes dispositivos e serviços que podem ser encontrados nas modernas redes de computadores. A forma como é realizado o acesso a estes recursos foge do escopo deste trabalho de dissertação. Contudo, foi desenvolvida uma API que permite a criação de módulos de *software* capazes de operar como *gateways* que recebem as informações de presença dos recursos gerenciados

(comunicação a ser implementada por um desenvolvedor) e geram notificações que são encaminhadas através do *overlay* P2P aos administradores interessados.

O restante deste trabalho está organizado da seguinte forma. No Capítulo 2 são revistos os principais trabalhos realizados na área de gerenciamento de redes e alguns dos principais modelos de serviços de presença existentes. O Capítulo 3 apresenta a arquitetura da solução proposta neste trabalho. Os detalhes da implementação da proposta são apresentados no Capítulo 4. No Capítulo 5 são apresentados os resultados das avaliações de desempenho realizadas. Por fim, as considerações finais e os trabalhos futuros são apresentados no Capítulo 6.

2 GERENCIAMENTO DE REDES E SERVIÇOS DE PRESENÇA

Neste capítulo são revistos os principais trabalhos da área de gerenciamento de redes, procurando apresentar como as pesquisas evoluíram para tentar solucionar os problemas que surgiram com as modernas redes de computadores. Em seguida, serão apresentados conceitos sobre o modelo de serviços de presença proposto pelo IETF. Logo após, são apresentados 6 dos principais serviços de presença encontrados na literatura e, em seguida, foram estabelecidos critérios que permitiram realizar uma comparação das características de tais serviços e assim, selecionar quais poderiam ser adicionados à implementação da arquitetura proposta. O levantamento destas características também contribuiu para o desenvolvimento do serviço de presença proposto e que será apresentado no próximo capítulo.

2.1 Evolução e Cenário Atual do Gerenciamento de Redes

Nesta seção, buscou-se mostrar a evolução dos modelos e técnicas utilizadas no gerenciamento e que são seguramente aceitos como importantes pela comunidade de gerenciamento de redes. Pretende-se também deixar claros os problemas destes modelos diante dos novos cenários encontrados nas corporações.

2.1.1 Gerenciamento Centralizado

Historicamente, o modelo centralizado de gerenciamento de redes foi o primeiro a ser utilizado, principalmente devido à sua facilidade de implementação e simplificada arquitetura. Nele há uma única estação de gerenciamento, chamada de gerente, responsável por concentrar todo o processamento das tarefas de gerenciamento (LEINWAND; CONDROY, 1996). Além desta estação, há um conjunto de agentes localizados dentro dos dispositivos físicos a partir de onde as informações de gerenciamento são obtidas (MARTIN-FLATIN; ZNATY; HABAUX, 1999). A Figura 2.1 apresenta este modelo centralizado.

Exemplos de tecnologias que podem implementar o modelo de gerenciamento centralizado (MARTIN-FLATIN; ZNATY; HABAUX, 1999) incluem o *Simple Network Management Protocol* (SNMP) (CASE et al., 1990) e o *Remote Network Monitoring MIB* (RMON) (STALLINGS, 1999), propostos pelo IETF, o *Common Management Information Protocol* (CMIP) (STALLINGS, 1993) proposto pelo *Open Systems Interconnection* (OSI), que faz parte do conjunto de recomendações da série X.700 (ITU-T Rec. X.700, 1992) e o NetFlow (Cisco, 2001), desenvolvido pela Cisco. É importante ressaltar que o SNMP também pode ser utilizado no gerenciamento não centralizado, como é no caso do

uso da Script MIB (SCHöNWALDER; QUITTEK; KAPPLER, 2000).

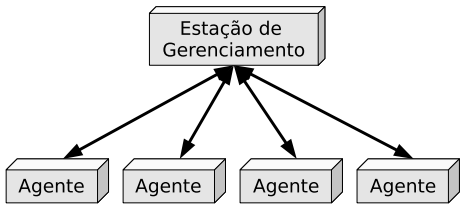

Figura 2.1: Modelo de gerenciamento centralizado

Apesar de ser amplamente utilizado nos sistemas de gerenciamento tradicionais, o modelo centralizado apresenta diversas limitações para o gerenciamento das modernas redes de computadores (SOLDATOS; ALEXOPOULOS, 2007). Uma das limitações do modelo centralizado diz respeito à escalabilidade, pois o aumento do número de equipamentos gerenciados na rede acaba por elevar a carga computacional requerida na estação de gerenciamento, dificultando ou mesmo inviabilizando o gerenciamento adequado dos equipamentos. Outro problema de escalabilidade é gerado pelo fato do sistema de gerenciamento precisar consultar todos os equipamentos gerenciados a partir de um único ponto, o que acarreta um alto tráfego nos enlaces de rede mais próximos à estação de gerenciamento. Além disso, estes sistemas apresentam limitações quanto à robustez. Quando, por exemplo, uma conexão entre a estação de gerenciamento e a rede é danificada, todas as funções de gerenciamento tornam-se indisponíveis. Por fim, o fato de centralizar todas as tarefas de gerenciamento em uma única estação, faz com que este modelo de gerenciamento possua um ponto vulnerável, pois se esta estação falha, toda a atividade de gerenciamento é interrompida.

2.1.2 Gerenciamento por Delegação

Devido às deficiências citadas anteriormente do gerenciamento centralizado, novos modelos de gerenciamento são definidos, resultando em soluções mais sofisticadas e introduzindo novas entidades de gerenciamento com responsabilidades especiais. No caso do modelo de gerenciamento por delegação (*Management by Delegation* – MbD) (GOLDSZMIDT; YEMINI, 1995), os gerentes podem delegar responsabilidades de gerenciamento a outros gerentes, chamados nesse caso de gerentes intermediários (*Mid-Level Manager* - MLM). Estes gerentes intermediários, de fato, são entidades híbridas que atuam como agentes ao receber tarefas delegadas de gerentes de nível mais alto (*Top-Level Manager* - TLM), mas também atuam como gerentes quando gerenciam agentes de nível mais baixo, que também podem ser outros MLMs. A Figura 2.2 apresenta os elementos deste modelo.

O gerenciamento por delegação introduz um bom nível de descentralização, uma vez que tarefas de gerenciamento são delegadas a diferentes gerentes que podem dividir assim a carga de processamento das ações de gerenciamento. Entretanto, no MbD não existe troca de informações entre gerentes de mesmo nível ou de diferentes hierarquias. Por exemplo, considerando os elementos apresentados na Figura 2.2, podem ocorrer as seguintes trocas de informações: (i) o “MLM-A” só pode se comunicar com o “TLM-1”, (ii) o “MLM-B” só pode se comunicar diretamente com o “TLM-1” e com o “MLM-C”, (iii) e o “MLM-C” só poderá trocar informações com o “TLM-2”, não podendo se reportar diretamente ao “TLM-1”.

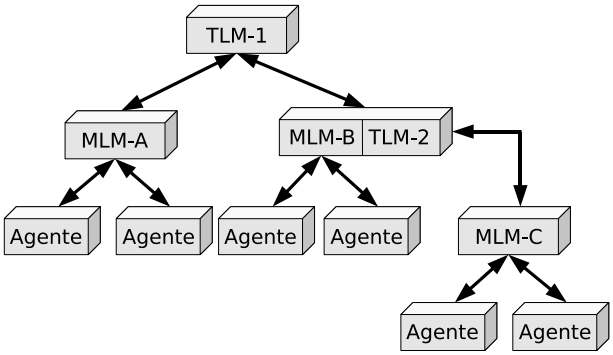

Figura 2.2: Modelo de gerenciamento por delegação

2.1.3 Gerenciamento de Redes Baseado em Políticas

Outro modelo de gerenciamento amplamente aceito pela comunidade é o baseado em políticas (*Policy Based Network Management* – PBNM), cujo objetivo é controlar o comportamento do sistema de acordo com políticas previamente definidas. Em geral, os sistemas baseados em políticas tendem a apresentar certo grau de centralização, o que pode não ser escalável quando o número de elementos gerenciados cresce. Nesse caso, ainda existe a necessidade da descentralização de sistemas de gerenciamento quando eles devem ser utilizados em cenários com um grande número de elementos gerenciados, como é o caso das redes de computadores atuais.

O modelo PBNM foi proposto tanto pela indústria como pelo meio acadêmico, sendo que hoje, existem diversas arquiteturas e sistemas baseados neste modelo, mas a definição mais aceita é a arquitetura PBNM proposta pelo IETF (WESTERINEN et al., 2001). O modelo PBNM do IETF é composto por quatro elementos: ferramenta de políticas (*Policy Tool*), repositório de políticas (*Policy Repository*), ponto de decisão da política (*Policy Decision Point* – PDP) e ponto de aplicação da política (*Policy Enforcement Point* – PEP). A Figura 2.3 apresenta os componentes do modelo PBNM do IETF e seus relacionamentos.

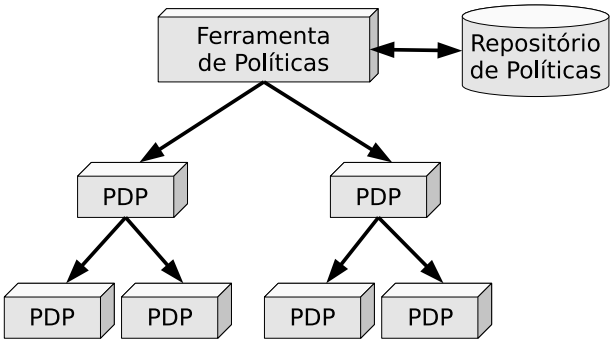

Figura 2.3: Modelo de gerenciamento de redes baseado em políticas do IETF

O modelo PBNM do IETF também possui fortes características distribuídas, uma vez que o sistema pode ser formado por diferentes PDPs, que são os elementos encarregados de dispararem as ações de gerenciamento na rede. Essa abordagem pode ser utilizada mesmo em cenários de gerenciamento formados por diferentes domínios administrativos como mostra o trabalho de Marquezan *et al*. (MARQUEZAN et al., 2005).

2.1.4 Gerenciamento Baseado em Tecnologias P2P

Uma tecnologia que poder ser utilizada para resolver os problemas citados anteriormente é a de sistemas *peer-to-peer* (P2P) (ANDROUTSELLIS-THEOTOKIS; SPINELLIS, 2004). Os sistemas P2P foram construídos inicialmente para permitir que usuários compartilhassem recursos (*e.g.*, arquivos, processamento) com outros usuários, provavelmente localizados em diferentes domínios administrativos. Atualmente, os sistemas P2P são utilizados com objetivos variados (*e.g.*, VoIP e videoconferência) e devido às suas características (*e.g.*, distribuição, escalabilidade e alta disponibilidade) são interessantes de serem aplicados no gerenciamento de redes. Por exemplo, sistemas P2P podem auxiliar na cooperação de administradores, localizados em diferentes domínios administrativos, para executar uma tarefa de gerenciamento. Outra característica interessante de sistemas P2P quando aplicados no gerenciamento, é que eles podem aumentar a conectividade entre as entidades de gerenciamento (*e.g.*, gerentes e agentes), tornando o sistema de gerenciamento mais confiável. Por fim, o uso de sistemas P2P pode resolver o problema do aumento no número de dispositivos e serviços gerenciados, uma vez que as tarefas de gerenciamento podem ser distribuídas em vários *peers*, contribuindo também para evitar o problema do ponto único de falha, como citado no modelo de gerenciamento centralizado.

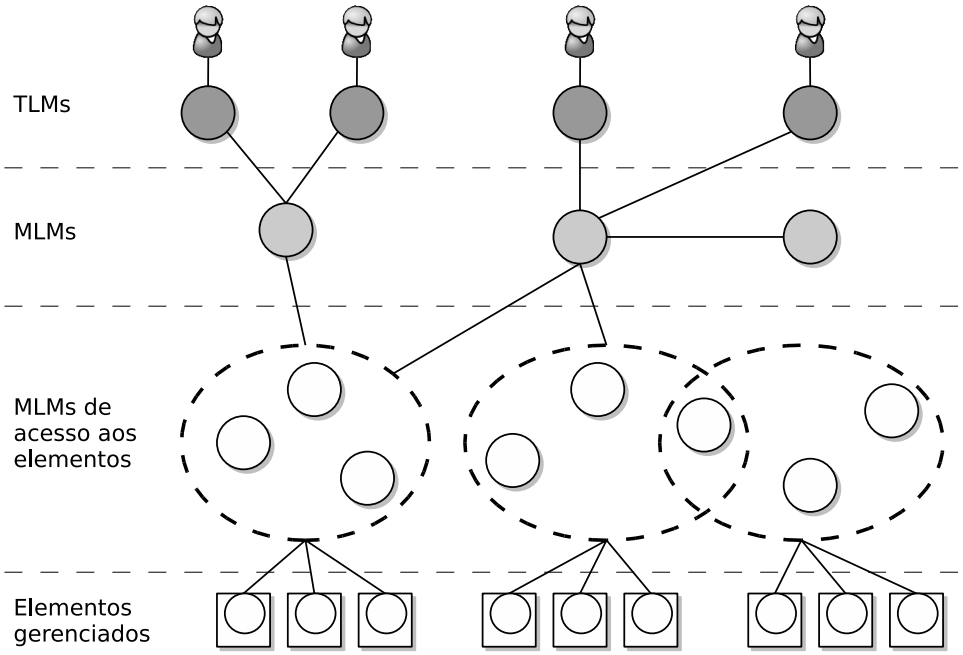

Figura 2.4: Modelo de gerenciamento de redes baseado em tecnologias P2P

Dentro deste contexto, o trabalho apresentado por Granville *et al*. (GRANVILLE et al., 2005) introduziu o conceito de gerenciamento de redes de computadores baseado em tecnologias P2P, onde a infra-estrutura P2P para gerenciamento pode ser vista como um conjunto de *peers* interligados e que constituem um *overlay* de gerenciamento a partir de onde as ações de gerenciamento são executadas. A Figura 2.4 apresenta o ambiente de gerenciamento de redes proposto por Granville *et. al* (GRANVILLE et al., 2005).

Este modelo de gerenciamento de redes baseado em tecnologias P2P baseia-se no modelo de gerenciamento por delegação, acrescentando características próprias de sistemas P2P, tais como alta distribuição das entidades de gerenciamento, auto-organização,

tolerância a falhas, escalabilidade e alta conectividade entre os *peers* do sistema. Essas características possibilitam o aperfeiçoamento dos sistemas de gerenciamento atuais, por exemplo, permitindo a introdução do suporte ao gerenciamento inter-domínios, troca confiável de mensagens de gerenciamento, replicação de serviços e distribuição de tarefas. Em sua proposta, Granville *et al*. definiu que os TLMs são os *peers* que implementam a interface gráfica com o administrador. É através desta interface que o administrador pode entrar em contato com outros administradores, executar tarefas de gerenciamento ou então receber notificações que foram encaminhadas por MLMs através do *overlay* P2P.

Um conceito importante na proposta de Granvile *et al*. é a noção de serviços de gerenciamento (*Management Services*). Um serviço de gerenciamento é um serviço oferecido por uma entidade de gerenciamento (*i.e*., MLM ou TLM) a outras entidades, tendo como resultado a execução de uma tarefa de gerenciamento. A disponibilização destes serviços em mais de um *peer* forma um grupo de *peers* (*peer group*), com o qual pode-se balancear a carga das tarefas de gerenciamento. Na implementação do modelo proposto, chamada de ManP2P e desenvolvida pelo Grupo de Redes de Computadores da UFRGS, estes serviços são disponibilizados na forma de Web Services, através do uso da *Application Programming Interface* (API) JXTA-SOAP (AMORETTI; ZANICHELLI; CONTE, 2005) que é uma implementação do *Simple Object Access Protocol* (SOAP) (MITRA, 2003) sobre a infra-estrutura P2P do JXTA (GONG, 2005), que por sua vez é um *framework* para o desenvolvimento de aplicações P2P, permitindo desta forma que Web Services executem em cima dos protocolos do JXTA.

O sistema de gerenciamento de redes ManP2P também fornece uma API que os desenvolvedores de novos serviços de gerenciamento devem utilizar para adicionar tais serviços à estrutura do ManP2P, possibilitando que os novos serviços desenvolvidos possam ser disponibilizados a *peers* remotos e também possam formar grupos, quando são disponibilizados por mais de um *peer*. Entre as principais funcionalidades que o ManP2P oferece estão:

- Trabalho cooperativo entre os administradores;
- Balanceamento de carga das tarefas de gerenciamento usando grupos de *peers*;
- Suporte para o gerenciamento inter-domínio;
- Possibilidade de integração com padrões de gerenciamento tradicionais.

Em trabalhos anteriores, avaliações de desempenho do ManP2P foram realizadas. Granville *et al*. (GRANVILLE et al., 2005) mostrou que o tráfego de mensagens de gerenciamento, quando corretamente manipuladas, pode ser menor que o gerado por outras soluções (*e.g*., SNMP) em situações em que um grande número de informações de gerenciamento é enviado. Santos *et al*. (SANTOS et al., 2008) avaliou o desempenho do ManP2P em relação ao envio de mensagens de notificação, considerando o tráfego na rede e o tempo total necessário para encaminhar tais notificações dos dispositivos físicos aos gerentes interessados. As notificações são enviadas dos dispositivos aos MLMs por meio de *traps* do SNMP. Os MLMs, por sua vez, encarregam-se de repassar tais notificações aos TLMs, onde os administradores recebem avisos dessas notificações. Observou-se com os resultado obtidos que o serviço de notificação implementado possui um desempenho melhor quando um maior número de MLMs é utilizado. Outra importante constatação é que aproximadamente 99% do tempo total das notificações é gasto dentro dos MLMs. Estes resultados serão apresentados em maiores detalhes em próximos capítulos.

Marquezan *et al*. (MARQUEZAN et al., 2007) deu maior atenção ao tempo de processamento gasto dentro dos MLMs quando são acrescentados novos *peers* (*i.e*., MLMs) ao *overlay*. Entretanto a simples visualização dos tempos de processamento não é o suficiente para avaliar se é melhor ou não acrescentar novos MLMs para melhorar o desempenho do envio de notificações. Assim observou-se o *speedup* relativo para concluir se há ganho de desempenho com a inclusão de novos *peers* intermediários. A análise do tempo de processamento indica que ao adicionar mais MLMs o tempo de notificação diminui quando vários administradores devem ser notificados. Por outro lado, a análise do *speedup* relativo mostra que ainda há um custo em adicionar tais MLMs, limitando o número total de MLMs a serem utilizados. Sendo assim, deve-se escolher uma configuração que leve em conta esses dois parâmetros para se obter o melhor cenário.

Com tudo o que foi apresentado, pode-se perceber que de fato, o uso de tecnologias P2P aplicadas no gerenciamento de redes podem ser uma boa alternativa aos modelos de gerenciamento tradicionais em face das modernas redes de computadores. Contudo, estas tecnologias P2P não estão sendo propostas com o objetivo de substituir os modelos de gerenciamento tradicionais, mas sim de complementá-los e aprimorá-los possibilitando um gerenciamento inter-domínio com *peers* localizados em diferentes domínios administrativos e que formam um ambiente flexível e dinâmico onde as tecnologias tradicionais podem ser utilizadas.

2.2 Serviços de Presença

Serviços de presença foram utilizados inicialmente em conjunto com aplicações de mensagens instantâneas (*e.g*., ICQ, MSN Messenger, GTalk, etc.), por isso são normalmente referenciados junto a este tipo de aplicação, dando origem ao termo *Instant Messaging and Presence* (IM&P). A seguir, é apresentada uma descrição geral do modelo de serviço de presença proposto pelo IETF e que serve como base para o desenvolvimento de novos serviços de presença. Em seguida, são revistos os principais modelos de serviço de presença existentes na literatura e ao final é realizada uma análise comparativa entre estes serviços.

2.2.1 Normas do IETF - RFCs 2778 e 2779

Os primeiros esforços para padronizar os serviços de presença foram realizados pelo *Internet Engineering Task Force* (IETF) nas RFCs 2778 (DAY; ROSENBERG; SUGANO, 2000) e 2779 (DAY et al., 2000). Estas RFCs apresentam um modelo abstrato para sistemas de mensagem instantânea e de presença, além de apresentar as entidades envolvidas, a terminologia, os requisitos e algumas funcionalidades que devem existir. A RFC 2778 define:

> "Um serviço de presença tem como função permitir que um usuário seja notificado da mudança do estado de outro recurso da rede, sendo que este outro recurso pode ser um humano, um dispositivo ou uma aplicação".

Uma informação de presença deve conter pelo menos o estado e/ou a disponibilidade. O estado tem como função informar apenas se o elemento está ativo ou não, enquanto que a disponibilidade (*e.g*., ocupado, livre, disponível em breve, etc.) serve para informar se o elemento pode ser acessado. Outras informações também podem estar presentes em uma informação de presença, por exemplo, a localização e a forma de acesso ao elemento.

A RFC2778 define a existência de dois componentes principais em sua arquitetura:

os *Presentities* e os *Watchers*. Os *Presentities* (termo formado pela combinação das palavras *presence* e *entity*), são representações dos recursos (*i.e.*, pessoas, dispositivos ou aplicações), ou conjunto de vários recursos, sob a forma de uma informação de presença. Estes recursos, chamados de *Principals*, são as entidades que não fazem parte da estrutura do serviço de presença e que possuem os estados a serem acessados. Os *Watchers* são os interessados em receber as informações de presença e dividem-se em dois grupos, os *Fetchers*, que buscam pela informação de presença, e os *Subscribers*, que registram-se para receber notificações do serviço de presença. Há também um tipo especial de *Fetcher*, denominado *Poller*, que busca por informações de presença regularmente. Por fim, há os *Presence User Agents* (PUAs) e os *Presence Agents* (PAs), sendo que PUAs têm como função colher as informações de presença dos *Principals* e os PAs são responsáveis por capturar todas as informações de presença de um ou vários PUAs e gerar eventos que são passados aos *Watchers*.

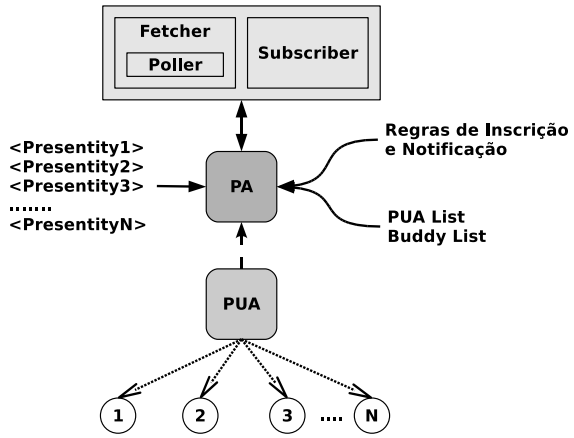

Figura 2.5: Modelo de serviço de presença proposto pelo IETF – RFC 2778

A Figura 2.5 apresenta os elementos descritos na RFC 2778 e a forma como os mesmos se relacionam. Nela, o PUA se encarrega de obter os estados dos *Principals* (enumerados de 1 a N) e também de encaminhar ao PA as informações de presença (*i.e.*, *Presentity1*, *Presentity2*, etc.) que representam os recursos que serão acessadas pelos *Watchers*. O PA também é responsável por armazenar uma relação de todos os PUAs e dos contatos (*buddy list*) autorizados a acessar informações destes PUAs. Finalmente os *Watchers*, dependendo de como operam (*i.e.*, *Fetcher* ou *Subscriber*), buscam por informações de presença ou se registram no serviço de presença para recebê-las.

Serviços de presença têm sido utilizados desde 1996, quando a primeira aplicação de mensagem instantânea, o ICQ, foi desenvolvida. Desde então, várias outras aplicações que fazem o uso destes serviços vêm sendo desenvolvidas (*e.g.,* MSN Messenger, AIM, Yahoo! Messenger, etc.), contudo, por serem aplicações proprietárias, os serviços de presença utilizados por tais aplicações não foram avaliados nesta dissertação. Sendo assim, 6 das principais propostas abertas de serviços de presença disponíveis na literatura são apresentadas a seguir.

2.2.2 XMPP

O *Extensive Message and Presence Protocol* (XMPP) é um protocolo aberto para troca de mensagens e informações de presença. O XMPP foi desenvolvido inicialmente pela comunidade Jabber, a qual submeteu no ano de 2000 um *Internet draft* para o grupo

de trabalho *Instant Messaging and Presence Protocol* (IMPP), documentando o protocolo Jabber, mas que foi recusado. Em 2002 a comunidade Jabber tentou novamente submeter o protocolo ao IETF, desta vez com sucesso, onde formou-se o grupo de trabalho XMPP. Desde então este grupo produziu quatro RFCs, sumarizadas a seguir:

- ***XMPP Core*** (SAINT-ANDRE; FOUNDATION, 2004a): define as principais características do XMPP (*e.g.*, arquitetura e endereçamento);
- ***XMPP Instant Messaging and Presence*** (SAINT-ANDRE; FOUNDATION, 2004b): especifica como o XMPP (*Core*) deve ser utilizado para enviar mensagens instantâneas e informações de presença;
- ***Mapping the XMPP to Common Presence and Instant Messaging (CPIM)*** (SAINT-ANDRE; FOUNDATION, 2004c): define o mapeamento entre mensagens do XMPP e do *Common Presence and Instant Messaging* (CPIM) (KLYNE et al., 2004) [1];
- ***End-to-End Signing and Object Encryption for the XMPP*** (SAINT-ANDRE; FOUNDATION, 2004d): especifica como pode ser implementada autenticação e cifragem de dados em uma comunicação XMPP.

A arquitetura do XMPP segue uma abordagem centralizada, uma vez que toda a comunicação entre os usuários (*i.e.*, clientes) do XMPP é realizada por intermédio de servidores, que são os componentes responsáveis por armazenar as listas de contatos dos usuários, fazer o roteamento das mensagens e controlar as conexões (*i.e.*, sessões) de/para usuários autorizados. Esta situação pode gerar problemas de escalabilidade, uma vez que todo o tráfego de dados gerado em um determinado domínio passa por um ponto único do sistema (*i.e.*, o servidor). O XMPP também define a existência de *gateways* para permitir a comunicação com aplicações de mensagem instantânea e serviços de presença que utilizem outros protocolos (*e.g.*, ICQ, MSN Messenger e Yahoo! Instant Messenger) e também com redes de outros tipos de sistemas de comunicação (*e.g.*, *Internet Relay Chat* (IRC), *Short Messsage Service* (SMS), *Simple Mail Transfer Protocol* (SMTP)). A Figura 2.6 apresenta os componentes da arquitetura proposta no XMPP.

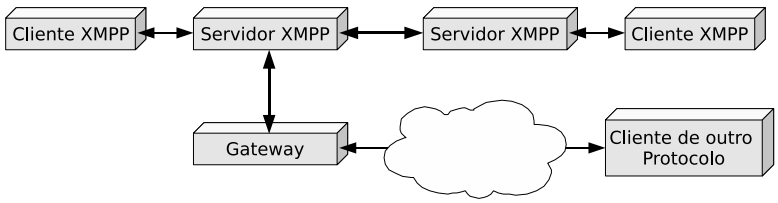

Figura 2.6: Componentes da arquitetura de serviço de presença proposta no XMPP

Qualquer recurso (*e.g.*, aplicação do usuário) existente no XMPP é endereçável por um identificador único, chamado *Jabber ID* (JID). Este identificador é semelhante à combinação de um endereço de email e uma URL. Por exemplo, um usuário com o nome "Bob" executando a aplicação "gaim" em "ufrgs.br" possuirá o seguinte JID: "Bob@ufrgs.br/gaim". Da mesma forma, recursos compartilhados, como salas de discussão (*chat rooms*), também são representados por JIDs. Por exemplo, "reuniao1@ufrgs.br" representa uma discussão "reuniao1" que está disponível no endereço "ufrgs.br".

[1] O CPIM é um formato de mensagens definido pelo IETF com o objetivo de ser um formato padrão para representar informações de presença e mensagens instantâneas.

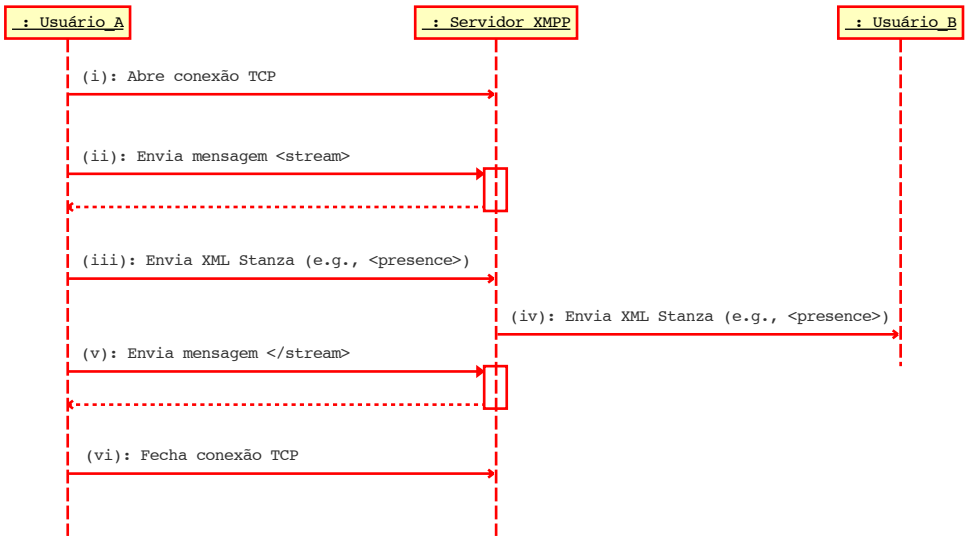

Figura 2.7: Interação entre dois usuários no XMPP

As mensagens utilizadas no XMPP, formatadas em XML, são chamadas de *XML Stanzas* e são transmitidas através de conexões persistentes, chamadas de *XML Streams*. A interação básica, apresentada na Figura 2.7, entre dois usuários "A" e "B" por meio de um "Servidor XMPP" ocorre da seguinte forma: (i) "A" abre uma conexão TCP com "Servidor XMPP", (ii) "A" inicia um *XML Stream* com "Servidor XMPP" enviando a mensagem ***<stream>***, (iii) "A" envia *XML Stanzas* (*e.g.*, <presence>) para o "Servidor XMPP" e este repassa esta mensagem para "B" (iv), "A" fecha o *XML Stream* enviando a mensagem ***</stream>*** (v), "A" fecha a conexão TCP (vi). Se for necessária uma comunicação bidirecional, ocorre a mesma seqüência de passos a partir de "B".

2.2.3 SIMPLE

O *SIP for Instant Messaging and Presence Leveraging Extensions* (SIMPLE) (B. CAMPBELL et al., 2002) foi desenvolvido inicialmente como uma extensão do *Session Initiation Protocol* (SIP) (ROSENBERG et al., 2002), que é um protocolo de sinalização capaz de iniciar, gerenciar e finalizar sessões multimídia entre pessoas geograficamente distantes e que é amplamente utilizado em aplicações de voz sobre IP (*Voice over IP – VoIP*). O propósito do SIMPLE é o de permitir a troca de mensagens instanâneas e informações de presença utilizando a infra-estrutura do SIP. Seu grupo de trabalho no IETF propôs três métodos adicionais ao SIP, apresentados a seguir, para possibilitar esta troca de mensagens.

- ***Subscribe***: usado quando um usuário quer ser notificado do estado de outro usuário;
- ***Notify***: usado para enviar a informação de presença de um *Presentity*;
- ***Message***: usado quando um usuário quer enviar uma mensagem instantânea.

No SIMPLE, a comunicação entre dois participantes pode ser realizada diretamente, utilizando uma abordagem P2P, desde que ambos os participantes saibam antecipadamente o endereço um do outro. No entanto, o SIMPLE também possibilita que dois participantes se comuniquem mesmo quando não conhecem os endereços dos outros participantes. Para isto, são utilizados *proxies* responsáveis por redirecionar a chamada para

onde o usuário estiver localizado baseado nas informações contidas nos *Registrars*. Este cenário também permite que os clientes se comuniquem mesmo quando estão localizados em domínios administrativos diferentes. A arquitetura usada no SIMPLE pode ser vista na Figura 2.8.

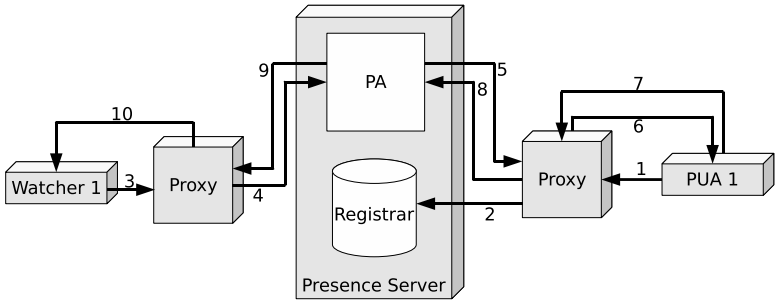

Figura 2.8: Componentes da arquitetura de serviços de presença utilizada no SIMPLE

Como apresentado na Figura 2.9, a interação entre um PUA e um *Watcher* ocorre da seguinte forma: "PUA-1" utiliza o método *Register* (método SIP) para publicar suas informações (*e.g.*, seu endereço). Esta requisição é enviada para o *Proxy* (1) e deste para o "Registrar" (2). "Watcher-1" envia uma requisição do tipo *Subscribe* pedindo informações do "PUA-1". Se o PA não possuir informações sobre a aceitação do "Watcher-1" pelo "PUA-1", a requisição será enviada ao "PUA-1" (5, 6), então o "PUA-1" irá informar ao PA se aceita enviar informações sobre seu estado ao "Watcher-1" (7, 8). Quando o estado do PUA mudar, será enviada uma mensagem do tipo *Notify* ao PA (9,10) e este irá reencaminhar esta mensagem, através dos *Proxys*, para o "Watcher-1" (11, 12).

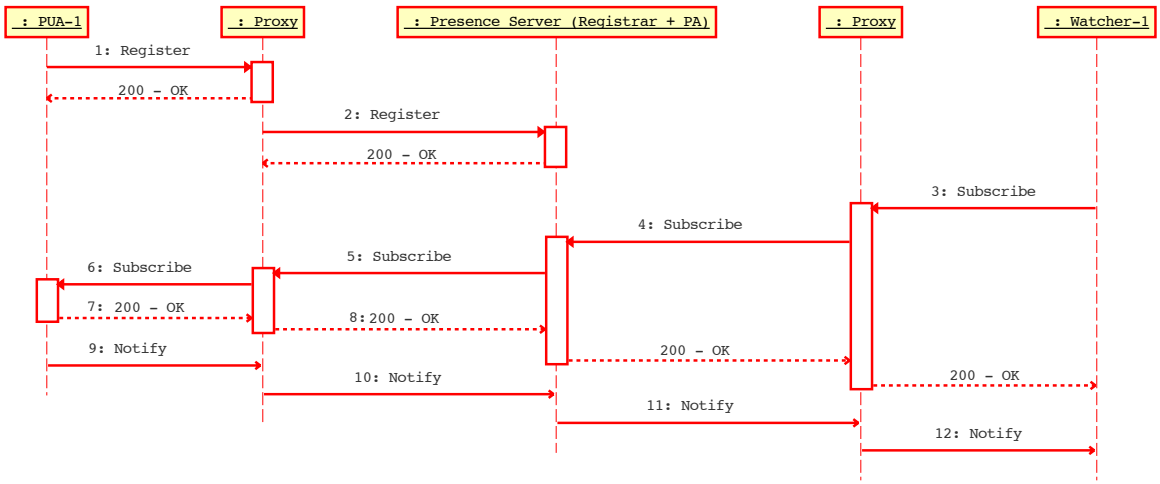

Figura 2.9: Interação entre dois usuários no SIMPLE

As mensagens de notificação no SIMPLE, formatadas em XML, se baseiam na especificação *Presence Information Data Format* (PIDF) (KLYNE et al., 2004), contudo expressam apenas o estado dos *Principals*. Para permitir que outras informações de presença (*e.g.*, atividade do usuário, localização atual) sejam utilizadas, o grupo de trabalho do SIMPLE sugere a utilização do *Rich Presence Information Data Format* (RPID) (SCHULZRINNE et al., 2006). Entretanto, mesmo assim, ainda não há suporte à informações de disponibilidade dos *Principals*. Como solução para controle acesso é sugerido o

uso do *XML Configuration Access Protocol* (XCAP) (ISOMAKI; LEPPANEN; NOKIA, 2007) para autorizar os *Watchers* a acessarem as informações de presença dos PUAs. O XCAP é um protocolo que permite que clientes acessem, criem e modifiquem informações de controle de acesso (formatadas em XML) localizadas em servidores.

2.2.4 Wireless Village

O *Wireless Village* (WV) (ERICSSON; MOTOROLA; NOKIA, 2002) é um conjunto de especificações abertas com o foco em sistemas móveis e que está sendo desenvolvido pelo grupo de trabalho *Instant Messaging and Presence Service* (IMPS) da *Open Mobile Alliance* (OMA). O objetivo do WV é possibilitar a troca de mensagens instantâneas e informações de presença entre dispositivos móveis e sistemas que utilizam redes fixas.

O WV apresenta uma arquitetura cliente/servidor onde são definidas duas entidades, o *WV Client* e o *WV Server*. O *WV Server* possui as seguintes funcionalidades associadas a ele: serviço de presença, serviço de mensagem instantânea, serviço de grupos, serviço de conteúdo e serviço de ponto de acesso (*Service Access Point*). O *Service Access Point* é responsável pela seguintes funções: autenticação, autorização, descoberta, controle dos perfis dos usuários e reencaminhamento de mensagens. Os *WV Clients* são os componentes responsáveis por se conectar aos *WV Servers* para enviar ou receber notificações de estado e são divididos em dois tipos: *WV Embedded Client*, que é uma estação (*e.g.*, telefone celular) móvel dotada de acesso à rede de dados da operadora de telefonia móvel, e *Command Line Interface Client* (CLI) é a estação móvel que usa mensagens de texto (SMS) para comunicar-se com o *WV Server*.

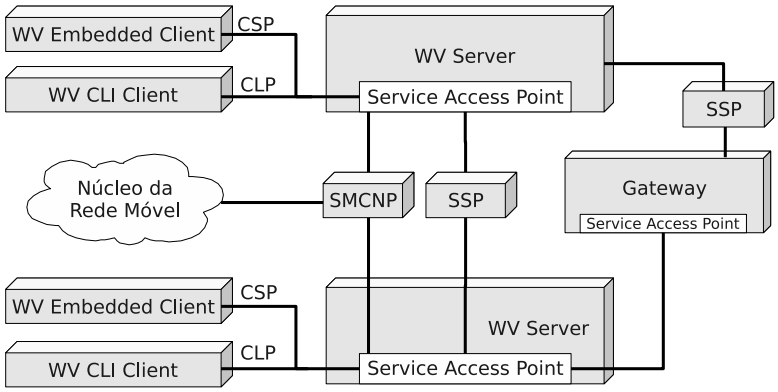

Figura 2.10: Componentes da arquitetura para serviços de presença proposta pela OMA – *Wireless Village*

O *Client-Server Protocol* (CSP) permite que estações fixas ou móveis acessem os *WV Servers*, que por sua vez comunicam-se utilizando o *Server-Server Protocol* (SSP). O *Command Line Protocol* (CLP) é o protocolo utilizado pelo CLI para permitir que antigos dispositivos móveis, sem capacidade de acesso direto à rede de dados da operadora de telefonia móvel, possam acessar o *WV Server* utilizando mensagens SMS. O *Server Mobile Core Network Protocol* (SMCNP) permite que o *WV Server* acesse informações da rede móvel para autenticação e autorização dos usuários. Todos estes elementos são apresentados na Figura 2.10. O WV, da mesma forma que o XMPP, também propõe o uso de *gateways* para comunicar-se com aplicações de outros serviços de presença usando mensagens construídas com base nas recomendações do CPIM.

2.2.5 APEX

O *Application Exchange (APEX) Access Service* (ROSE; KLYNE; CROCKER, 2003) foi construído com o objetivo de facilitar o desenvolvimento de aplicações de comunicação (*e.g.*, sistemas de mensagens instantâneas) que possuam como requisitos envio e recepção de mensagens e comunicação assíncrona. O grupo de trabalho APEX, do IETF, produziu três RFCs, sumarizadas a seguir.

- ***The Application Exchange core***: (ROSE et al., 2002a) principal documento contendo as principais definições do APEX, por exemplo, arquitetura e formato das mensagens;
- ***APEX Access Service***: (ROSE et al., 2002b) especificação de como implementar o controle de acesso;
- ***APEX Option Party Pack, Part Deux!***: (KLYNE et al., 2002) opções para modificar o comportamento do mecanismo de reencaminhamento do APEX.

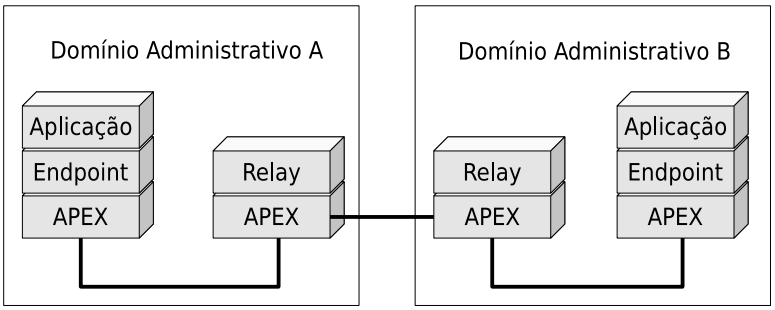

Figura 2.11: Arquitetura de serviço de presença utilizada no APEX

O APEX *core* fornece os principais mecanismos para envio de mensagens entre as aplicações dos usuários finais. Sua arquitetura, apresentada na Figura 2.11, define dois elementos: *Relays* e *Endpoints*. As aplicações dos usuários finais fazem o uso de abstrações chamadas *Endpoints*, que são responsáveis por mascarar o tipo de camada de transporte que está sendo usada, permitindo assim que as aplicações enviem e recebam dados independentemente da camada de transporte. Os *Relays*, por sua vez, representam os servidores do APEX, sendo que cada *Relay* é responsável por um domínio administrativo. Os *Relays* também possibilitam que os clientes localizados em diferentes domínios administrativos possam se comunicar.

O APEX foi construído utilizando o *Blocks Extensible Exchange Protocol* (BEEP) (ROSE, 2001), que é um protocolo genérico formado por um conjunto de *frameworks* que foram desenvolvidos para servir de base para que as aplicações possam trocar informações através de conexões permanentes, assíncronas e confiáveis. O APEX também utiliza o BEEP como base para a implementação das seguintes funções: compressão, roteamento, interoperabilidade, cifragem e autenticação. Por fim, as mensagens utilizadas no APEX, a exemplo dos outros serviços de presença, também são formatadas em XML.

2.2.6 PRIM

O *Presence and Instant Messaging Protocol* (PRIM) (SUGANO et al., 2001) é mais uma proposta do grupo de trabalho IMPP do IETF. Contudo, seu estágio atual está bastante inicial, tendo produzido apenas um *Internet Draft*. O PRIM define um conjunto de

protocolos para serviços de presença e de mensagens instantâneas e segue as recomendações das RFCs 2778, 2779 e do CPIM. O fato de seguir as recomendações do CPIM torna o PRIM um protocolo interoperável com outros serviços de presença, por exemplo, com o XMPP ou o WV. O PRIM assume que o protocolo único para o transporte de mensagens a ser utilizado é o TCP.

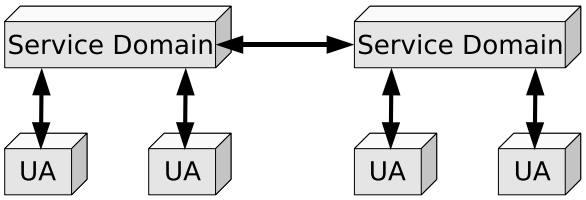

Figura 2.12: Arquitetura de serviço de presença utilizada no PRIM

A arquitetura do PRIM é bastante simples, seguindo uma abordagem cliente/servidor e definindo apenas dois componentes: *Service Domains* e *User Agents* (UAs). O *Service Domain* é um conjunto de servidores localizados dentro de um mesmo domínio administrativo e responsáveis por um conjunto de *Principals*, enquanto que é através dos *User Agents* que os *Principals* acessam funções do serviço de mensagens instantâneas e de presença.

O PRIM possui a característica de utilizar sessões de longa duração entre os clientes e servidores. Esta característica traz vantagens como a diminuição do consumo de banda, já que funções como autenticação são executadas apenas uma vez, no início de cada sessão. Por utilizar conexões persistentes iniciadas pelos *Principals* localizados dentro das redes, o PRIM pode ser utilizado em ambientes que possuam *firewalls*, permitindo assim que notificações ou mensagens geradas de fora da rede possam ser entregues para dentro da rede protegida. O PRIM também especifica que os servidores devem armazenar as informações de presença dos usuários e estes, por sua vez, se encarregam de renová-las periodicamente. As informações de presença também são renovadas sempre que há mudança no estado dos *Principals*.

2.2.7 PAM

O *Presence and Availability Management* (PAM) (GROUP, 2001) *Forum* é um consórcio de empresas estabelecido para criar um padrão para o controle e distribuição de informações de presença entre diferentes redes e aplicações. Diferentemente dos modelos de serviços de presença citados até o momento, o PAM não propõe uma arquitetura nem um protocolo de comunicação para serviços de presença, uma vez que seu objetivo é apenas o de desenvolver um conjunto de APIs e especificações que forneçam mecanismos que permitam um melhor controle das informações de estado e disponibilidade. Estas APIs foram implementadas na linguagem de programação Java, gerando a especificação JSR-123 (LOBO, 2003) disponibilizada pelo *Java Community Process* (JCP).

2.2.8 Comparação

Com o objetivo de selecionar quais dos modelos de serviços de presença, apresentados na seção anterior, podem ser adicionados à implementação da arquitetura proposta (a ser apresentada nos próximos capítulos), foram estabelecidos critérios qualitativos que permitiram realizar uma comparação entre tais serviços. Com esta comparação também

foi possível observar algumas características dos serviços de presença que possibilitaram conhecer um pouco mais do comportamento de cada um e assim, obter indícios de quando seria melhor empregar um serviço de presença ou outro. Além disso, a observação de tais características auxiliou na definição de detalhes de implementação do serviço de presença desenvolvido nesta dissertação, a ser detalhado no capítulo 3. Os critérios utilizados nesta comparação são apresentados a seguir.

- Arquitetura: se o serviço de presença segue uma arquitetura centralizada (passível de problemas devido à centralização em um ponto único) ou uma arquitetura distribuída (mais escalável);
- Formato das Mensagens: nível de detalhe (*i.e.*, granularidade) das informações de presença dos *Principals*;
- Grupo de Trabalho: empresas, consórcios ou pesquisadores por trás do desenvolvimento de cada serviço de presença;
- Pilha de Protocolos: camadas de protocolos utilizadas ou exigidas por cada serviço de presença em sua execução;
- Considerações sobre NATs e *firewalls*: existência de mecanismos que facilitem a transposição de elementos que possam dificultar a comunicação entre usuários localizados em domínios administrativos diferentes;
- Estágio de Desenvolvimento: nível de maturidade atual de cada serviço de presença (existência de lacunas ainda não propostas pelos grupos de trabalho);
- Controle de Acesso: impedir que usuários que não possuam as devidas credenciais de acesso possam obter informações sobre recursos que não sejam de sua responsabilidade;
- Criptografia: evitar que dados sigilosos não possam ser analisados por um possível usuário malicioso.

A comparação, baseada nos critérios estabelecidos, dos serviços de presença estudados é apresentada na Tabela 2.1.

Algumas conclusões podem ser feitas com a observação dos valores desta tabela. A primeira delas é o fato de que a maioria das soluções segue uma arquitetura centralizada. Isto ocorre talvez porque os grupos de trabalho preocuparam-se inicialmente com a simplicidade de implementação de cada serviço de presença. Esta característica pode apresentar problemas de escalabilidade quando há um grande número de usuários. Além disso, o elemento centralizador apresenta-se como um ponto único suscetível a falhas que podem indisponibilizar todo o sistema. No contexto do gerenciamento de rede, mais especificamente no uso de notificações para informar o estado dos dispositivos gerenciados aos interessados, esta característica de seguir uma arquitetura centralizada pode ser ruim, uma vez que o atraso na entrega das informações de presença pode aumentar de acordo com o número de notificações geradas pelos dispositivos ou serviços gerenciados.

Outra característica evidente é a preferência por utilizar dados formatados em XML, que pode ser explicado pela facilidade de manipulação dos dados em tal formato. O uso de mensagens formatadas em XML também pode facilitar na definição de mapeamentos entre mensagens de diferentes serviços de presença, provavelmente através de entidades

que operem como *gateways* entre redes de diferentes protocolos. Isto acontece, por exemplo, no uso do CPIM, utilizado pelo XMPP, PRIM e WV, tornando possível que usuários dessas três redes recebam notificações geradas por usuários das outras redes. Contudo, apesar das vantagens na formatação das mensagens em XML, seu uso pode aumentar o consumo de recursos da rede (*e.g.*, banda e processamento), uma vez que XML é um formato de dados que possui uma grande quantidade de informações repetidas (proveniente de suas *tags*). Esta situação pode ser especialmente ruim quando muitas notificações são enviadas a um único interessado.

Tabela 2.1: Comparação entre o XMPP, SIMPLE, APEX, PRIM e WV.

Critério	**XMPP**	**SIMPLE**	**APEX**	**PRIM**	**WV**
Arquitetura	Cliente/ Servidor	Híbrida	P2P	Cliente / Servidor	Cliente/ Servidor
Formato dos Dados	XML Stanza (Próprio)	PIDF + RPID	XML Próprio	PIDF	XML Próprio
Grupo de Trabalho	XMPP (IETF)	SIP/SIMPLE (IETF)	APEX (IETF)	PRIM (IETF)	WG-OMA
Pilha de Protocolos	IP/TCP/XMPP	IP/TCP&UDP/SIP/SIMPLE	IP/TCP/BEEP/ APEX	IP/TCP/PRIM	UDP, TCP, WAP, HTTP, SMS/CIR/CSP
Considerações sobre NATs/Firewalls	Abertura de portas / Endereçamento lógico	O SIP carrega o IP / Abertura de portas (dinâmicas)	-	Utiliza conexões abertas de dentro da rede	Utiliza conexões abertas de dentro da rede
Estágio de Desenvolvimento	Em operação	Em desenvolvimento	Em operação	Em desenvolvimento	Em operação
Controle de Acesso	Sim (XEP-0074)	Sim (XCAP)	Sim	Sim (Próprio)	Sim (Próprio)
Segurança:					
Autenticação	Sim (SASL)	Opcional (HTTP Digest)	Sim	Sim (SASL)	Sim
Cifragem	Sim (TLS)	Sim (TLS ou S/MIME)	Sim	Sim	Sim (HTTPS ou S/MIME)
Informação de Presença:					
Estado e Disponibilidade	Sim	Apenas Estado	Sim	Sim	Sim
Presença extendida	Sim	Em progresso (PIDF)	Não	Não	Sim

Observou-se também que quase todas as soluções, com exceção do SIMPLE e do WV, utilizam exclusivamente a arquitetura TCP/IP para enviar mensagens. O fato de usar o protocolo TCP traz vantagens no sentido de que este possui mecanismos de confirmação do envio das mensagens e mecanismos de controle de fluxo. Isto pode ser útil quando há a necessidade de ter um serviço de presença mais confiável no envio das notificações aos interessados, tornando desnecessária a reimplementação desta funcionalidade, o que pode contribuir para o aumento na complexidade da implementação do serviço de presença.

Todas os serviços de presença apresentados procuraram seguir as recomendações do IETF nas RFCs 2778 e 2779. Isto ocorre porque tais normas tem como objetivo fornecer os principais conceitos para quem quer desenvolver um serviço de presença. Elas não definem detalhes de funcionamento, de implementação ou mesmo se os componentes de sua arquitetura devem, ou não, ser distribuídos em vários computadores. Quatro das soluções encontradas possuem todas suas propostas iniciais implementadas, enquanto que as outras duas (*i.e.*, SIMPLE e PRIM) ainda possuem pontos que não foram desenvolvidos ou mesmo sugeridos. Apesar de o SIMPLE ainda ter alguns pontos de sua proposta em aberto, para funções básicas como o envio de informações de presença já pode ser utilizado, inclusive no contexto desta dissertação.

Com base nos resultados obtidos com a comparação das características dos 5 serviços de presença avaliados, chegou-se à conclusão de que apenas 3 soluções poderiam ser utilizadas no contexto desta dissertação. Estas soluções são: XMPP, SIMPLE e APEX. O PRIM não foi selecionado uma vez que seu grupo de trabalho no IETF não finalizou as especificações de como o PRIM deve funcionar para encaminhar as notificações de presença e mensagens instantâneas. O único documento gerado pelo seu grupo foi um *Internet Draft*, que expirou em 2002, e desde então, não houve mais indícios de que este grupo voltaria a se reunir para finalizar as especificações. O WV por sua vez, apesar de ter suas especificações bem definidas e ser um projeto aberto, possui um foco diferente do tratado nesta dissertação. O objetivo principal do WV é o de ser utilizado em sistemas móveis (*e.g.*, sistema de telefonia celular), diferente portanto, do propósito desta dissertação, que é de ser utilizada em ambientes de redes fixas, como os existentes nas corporações. Contudo, seu uso não está descartado em trabalho futuros.

Apesar do APEX poder ser utilizado como mecanismo capaz de enviar notificações de estado dos recursos gerenciados ao administradores interessados, ele não foi utilizado nesta dissertação. Isto se deve ao fato de não terem sido encontrados *frameworks* que permitissem sua implementação e integração à arquitetura proposta. Por fim, tanto o XMPP como o SIMPLE foram as duas soluções escolhidas para serem utilizadas nesta dissertação. O XMPP é uma solução madura e é amplamente utilizado como solução para comunicação instantânea, principalmente devido ao fato ser implementado no GTalk como protocolo de comunicação. O SIMPLE por sua vez, pode ser utilizado sem grandes modificações em ambientes que já utilizam o SIP como protocolo utilizado em aplicações de voz sobre IP. Embora o SIMPLE não possa enviar informações de disponibilidade, seu uso não pode ser descartado, uma vez que ele ainda pode ser utilizado para enviar informações de presença que contenham apenas o estado dos *Principals*.

3 SOLUÇÃO PROPOSTA

Como apresentado anteriormente, o uso de serviços de presença como mecanismo de notificação em processos de gerenciamento de redes é um tema ainda inexplorado pela comunidade de gerenciamento. Não sabe-se qual o comportamento que estes serviços apresentam quando empregados neste contexto. Com o objetivo de responder a esta pergunta, foi definida uma arquitetura de gerenciamento baseada em tecnologias P2P que utiliza serviços de presença tradicionais como mecanismo de notificação. Com base nesta arquitetura (SANTOS et al., 2008) foi possível avaliar o comportamento dos serviços de presença selecionados na seção 2.2.8 quando aplicados no gerenciamento de redes, o que será apresentado no capítulo 5. Sendo assim, neste capítulo serão apresentados os componentes da arquitetura proposta, os passos das etapas de registro e notificação e o protótipo de serviço de presença desenvolvido.

3.1 Arquitetura

A arquitetura proposta, de fato, se caracteriza como uma solução híbrida, que utiliza componentes e termos definidos tanto no modelo de serviço de presença do IETF, como no sistema de gerenciamento de redes ManP2P. Do modelo do IETF são utilizados os seguintes componentes: *Watchers*, PUAs e PAs. Além destes componentes, a arquitetura proposta acrescenta outros que são necessários para o seu funcionamento e que serão apresentados no decorrer desta seção. Também se utiliza o termo *Principal* para designar as entidades que não pertencem à estrutura do serviço de presença e que no contexto desta dissertação representam os dispositivos e serviços gerenciados. Do ManP2P, utilizou-se os MLMs que são *peers* responsáveis por receber ou delegar ações de gerenciamento à outros *peers* e TLMs que são *peers* que implementam a interface gráfica com o administrador. Nesse contexto, os PUAs e PAs foram projetados como *peers* MLM, enquanto que os *Watchers* podem ser disponibilizados tanto por MLMs como por TLMs, fato este que será explicado no decorrer desta seção.

A arquitetura proposta possibilita que serviços de presença tradicionais sejam adicionados a sua estrutura e então é capaz de selecionar qual é o serviço de presença mais adequado a ser utilizado em diferentes situações para informar o estado dos dispositivos e serviços gerenciados aos interessados. Os componentes da arquitetura proposta, bem como a forma que eles se relacionam são apresentados na Figura 3.1.

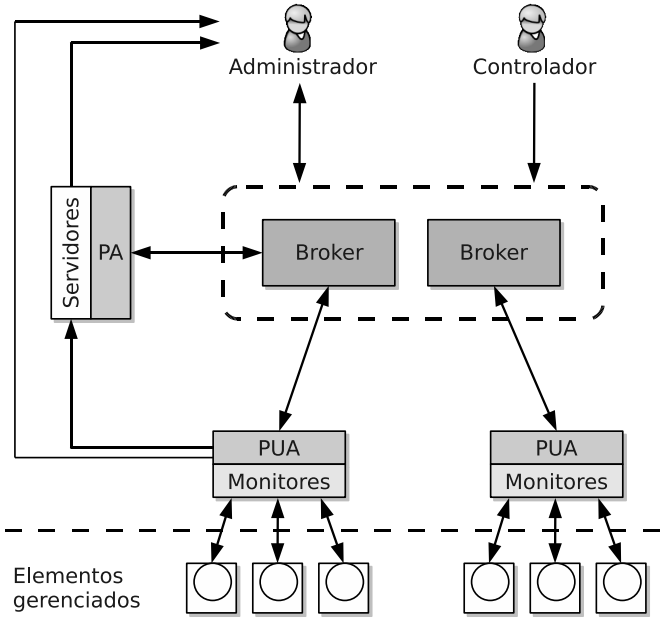

Figura 3.1: Arquitetura de gerenciamento proposta

Em nossa proposta, assumimos que existem dois tipos de operadores humanos e que possuem papéis diferentes: os administradores e os controladores. Em ambos os casos, a interação com o sistema de gerenciamento baseado em tecnologias P2P ocorre através de *peers Top-level Managers* (TLMs). O administrador é o operador humano interessado em receber as informações de presença dos recursos gerenciados, assim como um *Watcher*. O controlador, por sua vez, tem como função definir o funcionamento do sistema de gerenciamento especificando parâmetros que indicam em qual situação cada serviço de presença deve ser utilizado. Este papel de controlador não faz parte das normas do IETF, sendo assim, um dos acréscimos desta dissertação. De fato, um mesmo operador pode assumir ambos papéis, porém, funcionalmente, são dois papéis diferentes.

Como apresentado na Figura 3.1, o componente central da arquitetura é denominado *Broker*, que tem como responsabilidade controlar o sistema de gerenciamento para que as informações de presença sejam encaminhadas corretamente aos interessados (*i.e.*, *Watchers*). Mais próximos aos dispositivos e serviços (*i.e.*, *Principals*) gerenciados estão os PUAs, que tem como função capturar as informações de estado e enviá-las aos interessados. O PUA obtém as informações de estado dos *Principals* utilizando componentes denominados monitores. Estes monitores implementam a comunicação com os diferentes tipos de dispositivos (*e.g.*, roteadores, *switches*) ou serviços (*e.g.*, DNS, LDAP) gerenciados. Por fim, existem *peers* chamados PAs, que têm como função encaminhar ao sistema de gerenciamento detalhes de operação de cada servidor dos serviços de presença escolhidos.

Na arquitetura proposta, todos os componentes apresentados foram projetados como serviços de gerenciamento do ManP2P. Este fato aumenta a disponibilidade do sistema de gerenciamento e permite que as funções destes componentes possam ser acessadas por operadores localizados em diferentes domínios administrativos. Alguns destes componentes, de fato, foram projetados de forma que seus serviços de gerenciamento pudessem ser disponibilizados em mais de um *peer*, formando assim um grupo de *peers* (*peer group*), com o qual pode-se balancear a carga das tarefas destes componentes. A seguir,

todos os componentes da arquitetura proposta serão apresentados em maiores detalhes.

3.1.1 Broker

Os *Brokers* são as entidades responsáveis por decidir qual é o serviço de presença mais adequado a ser utilizado como mecanismo de notificação. Esta decisão é realizada por um mecanismo de seleção que utiliza tanto os parâmetros definidos pelos controladores, em um momento inicial de utilização do sistema de gerenciamento, como os pedidos de registro feitos pelos administradores, para então decidir qual é o serviço de presença que deve ser utilizado de acordo com cada situação. Por exemplo, um controlador pode informar que em determinada situação deve ser utilizado um serviço de presença capaz de enviar notificações mais rapidamente, enquanto que em outra situação deve ser utilizado um que gere menos tráfego. Este *Broker*, é um dos componentes da arquitetura proposta nesta dissertação que não fazem parte do modelo de serviço de presença do IETF.

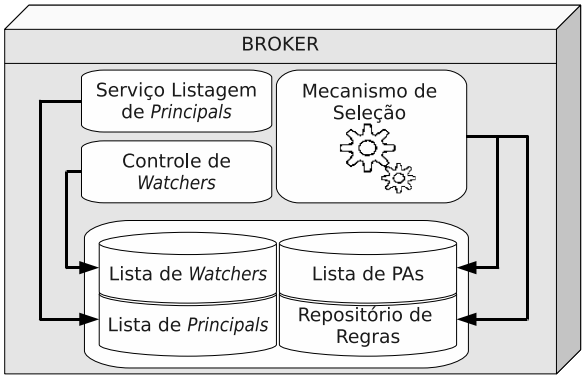

Figura 3.2: Estrutura interna do *Broker*

Além do mecanismo de seleção, a estrutura interna do *Broker*, apresentada na Figura 3.2, é formada por um serviço de gerenciamento, um controle de *Watchers* e quatro listas de dados que contêm informações sobre o funcionamento dos PAs, informações sobre os *Principals* disponíveis no sistema de gerenciamento, registro de *Watchers* que estão conectados ao sistema e dados de configurações (*i.e.*, regras) definidas pelos controladores. As funções que os *Brokers* desempenham são:

1. Configurar o sistema de gerenciamento: O *Broker* recebe os pedidos de registro dos *Watchers* e com base nas informações contidas nestes pedidos, o mecanismo de seleção decide qual é o serviço de presença mais adequado a ser utilizado. Após esta escolha, o *Broker* encaminha mensagens de configuração para os *peers* envolvidos na comunicação (*e.g.*, *Watchers*, PUAs e PAs), para que estes possam se comunicar devidamente;

2. Controlar quais os *peers* estão presentes no *overlay*: Cada *Broker* conhece quais *Watchers* estão conectados ao sistema de gerenciamento, de forma que, quando um *Watcher* sai do sistema, o *Broker* envia mensangens informando aos PUAs para removerem de seus registros o *Watcher* que saiu. O *Broker* também conhece quais PAs estão disponíveis no sistema de gerenciamento, auxiliando na escolha de qual mecanismo de notificação deve ser utilizado;

3. Armazenar uma lista contendo informações dos recursos disponíveis: Cada *Broker* conhece todos os dispositivos e serviços que estão disponíveis no sistema de gerenciamento e que podem ter suas informações de presença acessadas. Desta forma, quando um administrador está interessado em escolher sobre quais recursos quer ser notificado, ele envia um pedido de listagem ao serviço de gerenciamento localizado nos *Brokers* e então, um *Broker*, eleito por balanceamento de carga, responde a esta requisição informando quais são os *Principals* disponíveis.

De fato, os *Brokers* podem ser organizados em grupos de *peers* a fim de oferecer os serviços de gerenciamento a outros *peers*. Nesse caso, um serviço de gerenciamento, ao invés de ser oferecido por apenas um *Broker*, é oferecido por um grupo de *Brokers*. A existência deste grupo aumenta a disponibilidade das funções do *Broker*, porque enquanto houver um *peer* ativo no grupo, as funções deste *Broker* estarão disponíveis ao restante do *overlay*. Além disso, este grupo possibilita que a carga de processamento sobre cada *Broker* seja balanceada, evitando assim possíveis instabilidades quando o *Broker* executa suas funções e muitos pedidos de registro são gerados. A existência deste grupo também possibilita que as mensagens de registro possam ultrapassar barreiras como *firewalls*, permitindo que os administradores listem e se registrem para receber informações de presença de dispositivos ou serviços (*i.e.*, *Principals*) localizados em outros domínios administrativos.

3.1.2 Watcher

Os *Watchers*, assim como no modelo de serviço de presença do IETF, são as entidades interessadas em receber as informações de presença dos *Principals*. Para isso, estas entidades podem se registrar no serviço de presença e a partir daí recebem as notificações de estado (periodicamente ou ocasionalmente). Os *Watchers* também podem consultar o estado atual dos *Principals* sem necessariamente ter que se registrar no serviço de presença. Supõe-se que em uma situação normal, os *Watchers* sejam os administradores, ou seja, os operadores humanos que visualizam as notificações por meio da interface gráfica disponível nos TLMs.

Na arquitetura proposta, cada *Watcher* possui em sua estrutura interna, receptores que implementam a comunicação com cada um dos serviços de presença selecionados na seção 2.2.8, inclusive do protótipo de serviço de presença desenvolvido. O *Watcher*, depois de enviar o registro ao grupo de *Brokers*, aguarda pelas mensagens de configuração que informam qual serviço de presença deve ser utilizado, para então, iniciar o receptor relacionado a este serviço. Por fim, o *Watcher* aguarda pelas informações de presença que contêm o estado dos *Principals* em que se registrou.

Os *Watchers* foram projetados como serviços de gerenciamento do ManP2P, sendo que suas funções podem ser disponibilizadas tanto por *peers* TLMs como por MLMs. Este fato possibilita que as entidades que recebem as notificações de estado não sejam apenas os administradores, mas que sejam também os próprios *peers* do *overlay* P2P. Por exemplo, um *peer* que opera como MLM pode receber as notificações e sem encaminhá-la para administradores humanos, pode processá-la e tomar decisões sobre a rede, caracterizando um gerenciamento pró-ativo.

Este cenário em que as notificações não são encaminhadas para administradores humanos, mas sim para outros *peers* já foi utilizado por Marquezan *et al.*, (MARQUEZAN et al., 2007), onde os *peers* que recebem as notificações são entidades autonômicas do *overlay*, que são notificados quando um serviço de gerenciamento deixou de operar e, a

partir daí, tentam reiniciá-lo em outro *peer* do *overlay* P2P. Outro cenário possível em que os interessados em receber as informações de presença não são humanos é quando os *Watchers* operam como *gateways*. Neste cenário, uma informação de presença é entregue ao MLM que opera como *Watcher* e este pode enviá-la, por exemplo, para outros sistemas de gerenciamento.

3.1.3 Controlador do Sistema

A escolha de qual serviço de presença deve ser utilizado para informar o estado dos *Principals* é realizada com base em um conjunto de parâmetros definidos em uma etapa inicial pelo controlador do sistema de gerenciamento. A função do controlador, assim como o *Broker*, não faz parte do modelo de serviço de presença proposto pelo IETF e foi sugerido para que os controladores tenham a flexibilidade de adaptar o sistema de gerenciamento de acordo com as suas necessidades.

O controlador interage com o sistema de gerenciamento P2P utilizando a interface gráfica disponível nos *peers* TLMs. Nesta interface gráfica, cada controlador visualiza quais são os serviços de presença disponíveis no sistema de gerenciamento (*e.g.*, XMPP e SIMPLE) e define em quais situações cada serviço será utilizado como mecanismo de notificação. As situações são representadas por um conjunto de parâmetros que os controladores têm à sua disposição para edição. Por exemplo, um dos parâmetros disponíveis, o "Log", possibilita que o controlador escolha qual serviço de presença deve ser utilizado quando a informação de presença tiver que ser armazenada para consultas futuras. Outro exemplo é quando o controlador informa qual serviço de presença deve ser utilizado quando o intervalo de tempo entre duas mensagens é curto, utilizando para isto o parâmetro "Periodicidade". O controlador também pode utilizar mais de um parâmetro para definir uma situação. Por exemplo, o controlador pode definir uma situação em que o intervalo de tempo entre duas notificações é curto e que a informação de presença contida nesta notificação contém apenas o estado do recurso gerenciado.

Depois que o controlador define as situações e quais serviços de presença devem ser utilizados em cada uma, uma mensagem é enviada para o grupo de *Brokers* contendo os parâmetros escolhidos e o serviço de presença a ser utilizado. Os *Brokers*, por sua vez, armazenam estas configurações em um repositório que será acessado sempre que receberem um pedido de registro vindo dos *Watchers* para então executarem o mecanismo de escolha, como explicado anteriormente. Os parâmetros que o controlador tem à sua disposição para definir as situações em que cada serviço de presença deve utilizado são apresentados a seguir.

1. Log: se as informações de presença devem ser armazenadas para futuras consultas;
2. Periodicidade: intervalo de tempo entre o envio de duas notificações;
3. Mudança no estado: se as informações devem ser enviadas apenas quando houver mudança no estado dos *Principals* ou se devem ser enviadas de acordo com a periodicidade;
4. Tempo de duração: por quanto tempo as notificações devem ser enviadas;
5. Tipo de mensagem: estado, disponibilidade ou todas as informações disponíveis;
6. *Principal*: usado quando o controlador quer informar que um serviço de presença específico deve ser utilizado sempre que uma informação de um determinado *Principal* for requisitada pelos administradores.

O controlador também tem a possibilidade de especificar um serviço de presença padrão, que será utilizado quando um pedido de registro feito por um administrador não possuir uma situação correspondente às definidas pelos controladores. Nesse caso, o controlador seleciona o serviço de presença disponível e então especifica que este serviço irá ser utilizado como mecanismo de notificação padrão, não sendo necessário especificar qualquer um dos parâmetros apresentados acima.

3.1.4 *Presence User Agents* (PUAs) & Monitores (*Gateways*)

Mais próximos aos dispositivos e serviços gerenciados estão os PUAs, que assim como no modelo de serviço de presença do IETF, têm como função enviar aos *Watchers* os estados dos *Principals*. A forma como é feito o acesso aos *Principals* não faz parte do escopo desta dissertação, uma vez que existem diferentes dispositivos e serviços nas modernas redes de computadores e que podem ser acessados de diversas maneiras. Desta forma, nesta dissertação se introduziu o componente denominado monitor, que é o encarregado de implementar a comunicação com os dispositivos e serviços. Estes monitores operam então como *gateways*, obtendo o estado dos *Principals* e encaminhando esta informação para dentro do sistema de gerenciamento.

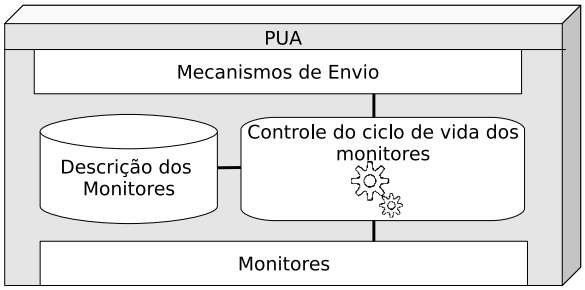

Figura 3.3: Estrutura interna do PUA

Na Figura 3.3, pode-se observar a estrutura interna do *peer* que opera como PUA. Dentro deste *peer*, podem ser disponibilizados vários monitores, cada um acessando informações de presença de diferentes *Principals*. Dentro de cada PUA existe uma lista de dados que contém informações sobre os monitores disponíveis e que é acessada e publicada para os *Brokers* sempre que o PUA é iniciado. Desta forma, todos os *Principals* disponíveis no sistema de gerenciamento P2P, podem ser listados pelos administradores interessados em serem notificados sobre o estado dos mesmos.

Os monitores possuem outra função, que é a de converter as informações de presença obtidas dos *Principals* (que podem ser representadas de diversas formas) para os formatos de mensagens utilizados pelos serviços de presença selecionados para serem integrados à implementação da arquitetura proposta. Por exemplo, um monitor que tenha sido desenvolvido para obter o estado de uma impressora enviando mensagens *Internet Control Message Protocol* (ICMP) é capaz de transformar a resposta desta mensagem em um *XML Stanza* em uma situação em que o XMPP tenha sido escolhido como mecanismo de notificação. O desenvolvimento destes monitores é feito com base em uma API definida nesta dissertação e que será apresentada em detalhes na próxima seção.

Ao receber a configuração encaminhada pelos *Brokers* e que contém o pedido de registro de um *Watcher*, o PUA verifica qual serviço de presença deve ser utilizado para encaminhar as informações de estado, inicia o mecanismo de envio selecionado e então

instancia o monitor responsável pelo *Principal* selecionado. A partir daí, o PUA controla o ciclo de vida deste processo, informando ao monitor qual mecanismo de envio deve ser utilizado, o momento em que deve enviar as notificações e por quanto tempo este processo deve executar.

3.1.5 *Presence Agents* (PAs)

O modelo de serviço de presença proposto pelo IETF possui um componente denominado *Presence Agent* (PA), que é responsável por controlar algumas funções dos serviços de presença. Por exemplo, no XMPP o PA é utilizado para controlar as conexões dos usuários, fazer o roteamento das mensagens e armazenar as listas de contatos dos usuários. No SIMPLE, o PA é responsável por aceitar os pedidos de inscrição e gerar notificações que são entregues aos *Watchers* quando há mudança no estado dos *Principals*. Para que os usuários (*i.e.*, entidades interessadas em estabelecer uma comunicação) possam enviar mensagens ou informações de presença para outros usuários utilizando os dois serviços de presença citados, é necessário que suas aplicações (*i.e.*, componentes de *software* que implementam a comunicação) estejam corretamente configuradas para se conectar aos PAs. Para isto, detalhes sobre o funcionamento de cada PA devem ser conhecidos previamente, tais como endereço de rede e portas TCP/UDP.

Estes PAs, no entanto, são componentes que não fazem parte do *overlay* de gerenciamento P2P, mas que no contexto desta dissertação precisam ser utilizados quando o mecanismo de notificação escolhido for o SIMPLE ou o XMPP. Naturalmente, os detalhes de funcionamento destes PAs não podem ser acessados pelos *peers* do *overlay*, tornando necessário que uma das seguintes possíveis abordagens seja utilizada. A primeira é informando explicitamente aos *Brokers* estas informações e a segunda é utilizando um *peer* associado a cada PA e que é responsável por repassar tais informações aos *Brokers* quando estes requisitam. Nesta dissertação, utilizou-se a segunda abordagem, que possibilita um cenário mais elaborado, com mais de um PA. A escolha de qual PA deve ser utilizado para estabelecer a conexão entre um PUA e um *Watcher* é feita com base nos algoritmos de balanceamento de carga existentes no ManP2P. Desta forma, quando o número de PUAs enviando notificações aos *Watchers* aumenta, pode-se ter um ambiente mais escalável, não sobrecarregando excessivamente cada PA.

O funcionamento destes *peers*, denominados PAs, da segunda abordagem ocorre da seguinte forma: quando um *Watcher* se registra para ser notificado sobre o estado de um *Principal* e o serviço de presença escolhido é o SIMPLE ou o XMPP, um *Broker* faz uma chamada ao serviço de gerenciamento disponibilizado nestes *peers* PAs, onde um *peer* eleito por balanceamento de carga responde informando os detalhes de operação do seu PA associado. O *Broker* então constrói as mensagens de configuração que são enviadas aos *peers* envolvidos na comunicação, para que os receptores dos *Watchers* e os mecanismos de envio dos PUAs possam se conectar ao PA escolhido.

3.1.6 Interação Entre os Componentes da Arquitetura

Todos os passos executados pelos componentes da arquitetura proposta podem ser separados em três etapas: controle, registro e notificação. A etapa de controle envolve a seqüência de passos executados pelo sistema de gerenciamento quando o controlador define quais serviços de presença devem ser utilizados como mecanismo de notificação de acordo com cada situação. A etapa de registro ocorre quando o *Watcher* se registra no sistema para ser notificado sobre o estado de algum *Principal*. Durante esta etapa, ocorre também o envio de mensagens de configuração, geradas pelo *Broker*, aos *peers*

envolvidos na comunicação. Por fim, a etapa de notificação ocorre quando as informações de presença são enviadas aos *Watchers*. Um resumo dos passos executados nas três etapas são apresentados na Figura 3.4 e são listados a seguir:

1. Ao iniciar, cada PUA envia para o grupo de *Brokers* sua lista de dados que contém informações sobre os monitores disponíveis;

2. O controlador define parâmetros que representam as situações possíveis em que cada serviço de presença deve ser utilizado para informar o estado dos *Principals*, estas regras são então enviadas para o grupo de *Brokers*;

3. O *Watcher* se registra no sistema de gerenciamento para receber as notificações de estado dos recursos gerenciados;

4. O *Broker* gera uma mensagem de configuração que é encaminhada aos *peers* relacionados com a comunicação;

5. O PUA, no momento definido pelo administrador, envia a informação de presença sobre os dispositivos e serviços gerenciados.

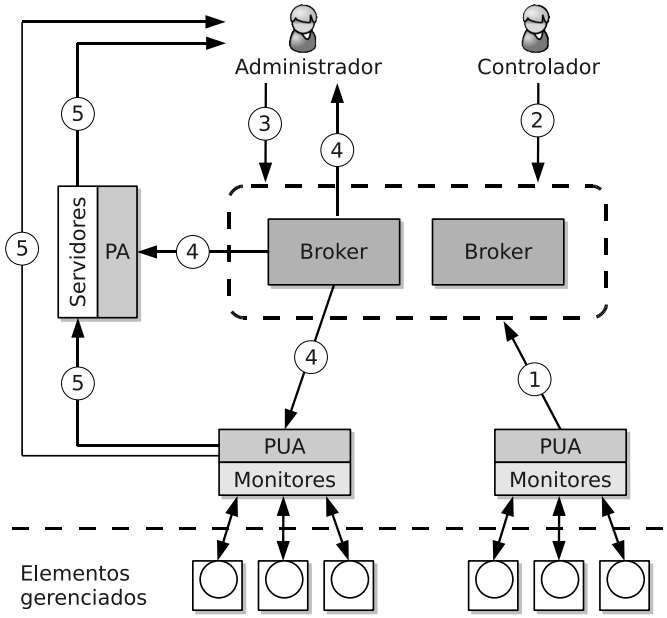

Figura 3.4: Resumo da seqüência de passos executados nas etapas de controle, registro e notificação

No entanto, dependendo do serviço de presença escolhido pelos *Brokers* como mecanismo de notificação, os passos executadas nas etapas de registro e notificação podem mudar, por exemplo, não sendo necessário buscar informações sobre os PAs.

Na Figura 3.5 são apresentados os passos da etapa de registro quando o serviço de presença escolhido é o SIMPLE ou o XMPP. Em um primeiro instante, (1) o *Watcher* se registra no sistema de gerenciamento para ser notificado sobre o estado de algum dispositivo ou serviço (*i.e.*, *Principals*). Em seguida, (2) o *Broker* obtém as informações de localização e de acesso do PA do serviço de presença escolhido. Com base nestas

informações e nas configurações já registradas na etapa de controle, o *Broker* cria uma mensagem de configuração que é enviada ao PUA (3) e ao *Watcher* (4). Esta mensagem indica a estes componentes como seus receptores (no caso dos *Watchers*) e mecanismos de envio (usados pelos PUAs) podem se conectar aos PAs. Por fim, o *Broker* envia uma confirmação ao *Watcher* informando que seu registro foi feito com sucesso.

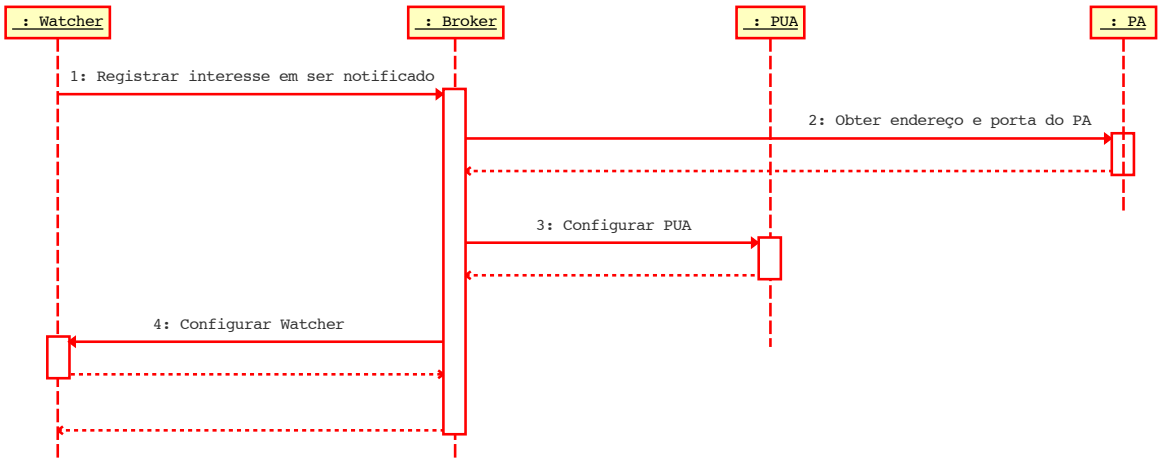

Figura 3.5: Diagrama de seqüência da etapa de registro do SIMPLE ou do XMPP

Quando o serviço de presença escolhido como mecanismo de notificação for o serviço de presença desenvolvido (com qualquer um dos seus meios de comunicação), não há a necessidade de fazer com que os clientes desta comunicação (*i.e.*, PUA e *Watcher*) se registrem em algum ponto do sistema de gerenciamento. Como apresentado na Figura 3.6, o *Watcher* envia seu pedido de registro ao grupo de *Brokers* (1), onde um *peer* recebe este pedido e cria uma mensagem de configuração que é encaminhada ao PUA (2) e ao próprio *Watcher* (3). Por fim, é retornada ao *Watcher* uma confirmação de seu registro.

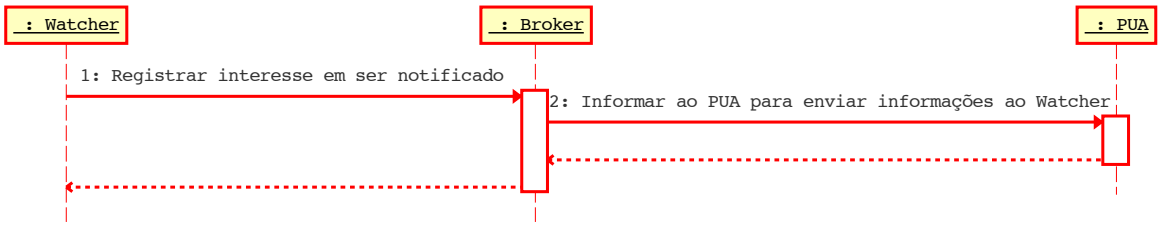

Figura 3.6: Diagrama de seqüência da etapa de registro do serviço de presença proposto

Durante a etapa de notificação, quando o serviço de presença selecionado é o SIMPLE ou o XMPP, o PUA envia a informação de presença ao PA (1), que por sua vez, reencaminha esta notificação para o *Watcher* (2). Ambos os serviços de presença confirmam o envio de suas mensagens. Esta seqüência de passos é apresentada na Figura 3.7.

Quando o serviço de presença escolhido como mecanismo de notificação for o protótipo do serviço de presença desenvolvido, podem ser utilizados três métodos de comunicação disponíveis no ManP2P para enviar as informações de presença. Em qualquer um dos casos, a notificação contendo o estado dos *Principals* é enviada diretamente do PUA ao *Watcher* (1), como apresentado na Figura 3.8. A única mudança existente entre estes

três métodos de comunicação, que serão apresentados em detalhes no próximo capítulo, é a existência ou não de confirmação.

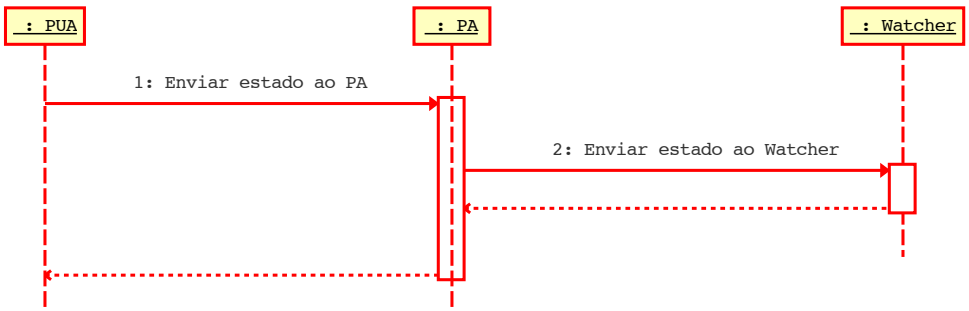

Figura 3.7: Diagrama de seqüência da etapa de notificação do SIMPLE ou do XMPP

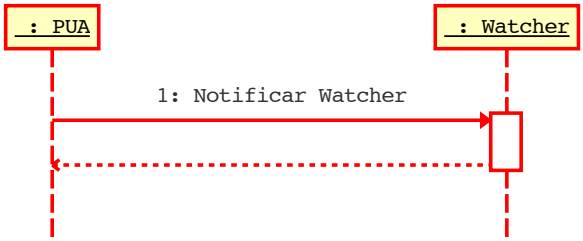

Figura 3.8: Diagrama de seqüência da etapa de notificação do serviço de presença proposto

3.2 Serviço de Presença Proposto

Normalmente, os serviços de presença tradicionais utilizam técnicas que não condizem mais com a realidade encontrada nas modernas redes de computadores. Por exemplo, verificou-se na subseção 2.2.8 que a maioria dos serviços de presença avaliados utilizam uma abordagem centralizada, suscetível há falhas e não escalável. Além disso, estes serviços de presença muitas vezes exigem configurações extras nos *firewalls* e NATs das organizações para que usuários localizados em domínios administrativos diferentes possam se comunicar. Por outro lado, as modernas tecnologias P2P apresentam interessantes características que as candidatam para serem utilizadas neste contexto. Tais tecnologias, por exemplo, possuem meios para ultrapassar as barreiras dos domínios administrativos. Além disso, estas tecnologias não são suscetíveis à problemas decorrentes da centralização de funções em um único ponto.

Neste contexto e com o objetivo de observar o comportamento de um serviço de presença que utiliza exclusivamente tecnologias P2P para possibilitar a troca de mensagens entre os participantes de uma comunicação, é proposto um serviço de presença que se baseia em tecnologias P2P para notificar o estado dos recursos gerenciados. Este serviço de presença desenvolvido foi então integrado à implementação da arquitetura proposta, da mesma forma que os dois serviços de presença selecionados no capítulo anterior. Desta forma, foi possível observar o comportamento dos dois serviços de presença selecionados (*i.e.*, SIMPLE e XMPP) e comparar este comportamento com o do serviço de presença proposto, o que será apresentado no capítulo 5.

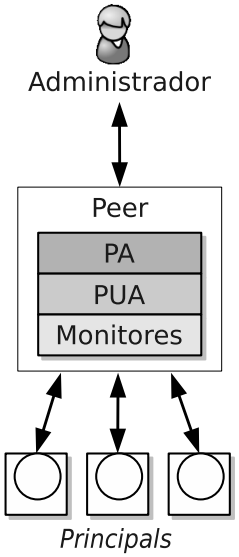

Figura 3.9: Arquitetura do serviço de presença proposto

No serviço de presença proposto, apresentado na figura 3.9, cada participante de uma comunicação (*i.e.*, PUAs e *Watchers*), de fato é um *peer* do sistema de gerenciamento P2P. Com o objetivo de diminuir o tempo de envio de uma notificação e diminuir o consumo de banda, decidiu-se por disponibilizar as funções que normalmente são atribuídas aos PAs no mesmo *peer* onde estão as funções dos PUAs. Desta forma, um único *peer* pode receber as informações de presença dos *Principals*, processar tais informações, gerar as notificações que serão entregues diretamente aos *Watchers* e armazenar as listas de contatos dos usuários (*i.e.*, *Watchers* interessados). Contudo, para evitar problemas com a centralização de tarefas em um único ponto, este *peer* pode ser replicado, obtendo-se um cenário em que dois ou mais *peers* recebem o pedido de registro vindo dos *Brokers*, obtêm o estado dos *Principals* e comunicam-se entre si para verificar qual *peer* deve ser o responsável por enviar as notificações. As funções de reencaminhamento das notificações ficam a cargo do próprio *overlay* P2P.

4 IMPLEMENTAÇÃO

Neste capítulo serão apresentados detalhes de implementação do protótipo da arquitetura de gerenciamento apresentada no capítulo 3 e também do serviço de presença proposto. Em seguida, será apresentado como é realizada a integração de serviços de presença à estrutura do protótipo desenvolvido e como é mantido o suporte aos dispositivos e serviços encontrados nas modernas redes de computadores. Ao final, será apresentada a interação entre o operador humano (desempenhando os papéis de administrador e de controlador) com o protótipo desenvolvido que foi integrado ao sistema de gerenciamento de redes ManP2P (GRANVILLE et al., 2005).

4.1 Protótipo da Arquitetura de Gerenciamento Proposta

O protótipo da arquitetura de gerenciamento apresentada no capítulo 3 foi desenvolvido utilizando a API para criação de serviços de gerenciamento disponível no ManP2P, que é um sistema de gerenciamento de redes de computadores desenvolvido pelo Grupo de Redes da UFRGS e que combina funcionalidades de gerenciamento de redes e características de sistemas P2P, tais como distribuição de tarefas em vários *peers*, que podem estar localizados em diferentes domínios administrativos. O ManP2P, por sua vez, foi desenvolvido utilizando JXTA (*Juxtapose* – Justaposto), que é um *framework* para o desenvolvimento de aplicações P2P.

O JXTA foi desenvolvido inicialmente pela Sun Microsystems, Inc. e, atualmente, é mantido por um grupo independente. O JXTA compreende um total de 6 protocolos que entre suas principais funções estão: troca de informações de estado (*e.g.*, quantidade de conexões e tempo ativo) dos *peers*, descoberta de rotas, e envio de consultas (*queries*) entre os *peers*. Além dos protocolos, o JXTA possui *peers* especiais em sua infra-estrutura, que são necessários para manter o *overlay* P2P. Por exemplo, o *Rendezvous Peer* tem como função armazenar anúncios (*Advertisements*) enviados pelos *peers* convencionais, denominados *Edge Peers*. Estes anúncios representam quaisquer recursos disponibilizados no *overlay*, tais como, canais de comunicação, grupos e os próprios *peers*. Outro componente existente na infra-estrutura do JXTA é o *Relay Peer*, que tem como função facilitar a conexão de *peers* localizados em diferentes domínios administrativos, permitindo que as mensagens ultrapassem barreiras como *firewalls* e NATs.

Apesar de existirem implementações dos protocolos do JXTA em outras linguagens de programação (*i.e.*, C) e plataformas de desenvolvimento (*i.e.*, Java ME – *Java Micro Edition*), o desenvolvimento do protótipo da arquitetura proposta nesta dissertação utilizou a implementação em Java SE (*Standard Edition*). Esta escolha deve-se principalmente ao fato de que a implementação em Java SE foi a utilizada pelo ManP2P. O fato de utilizar a linguagem de programação Java, possibilita que a implementação do protótipo proposto

possa ser executada em diferentes plataformas de *hardware* e *software*.

Para possibilitar a troca de mensagens entre os serviços de gerenciamento, o ManP2P utiliza o *framework* JXTA-SOAP (AMORETTI; ZANICHELLI; CONTE, 2005) que é uma implementação do *Simple Object Access Protocol* (SOAP) (MITRA, 2003) sobre a infra-estrutura P2P do JXTA, permitindo assim o desenvolvimento de Web Services que executam acima dos protocolos JXTA. Desta forma, na Figura 4.1 pode-se observar as camadas de *software* utilizadas no desenvolvimento deste trabalho.

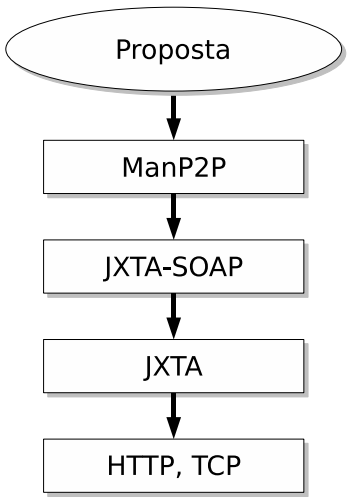

Figura 4.1: Camadas de *software* utilizadas no desenvolvimento do protótipo

Na camada inferior da Figura 4.1 estão os protocolos que o JXTA pode utilizar para propagar suas mensagens na rede. Na camada seguinte, estão os protocolos JXTA que são utilizados para criar o *overlay* P2P. Mais acima, há o JXTA-SOAP, que possibilita a troca de mensagens utilizando chamada remota de procedimentos. Na camada ManP2P, está a API para criação de serviços de gerenciamento disponível no ManP2P. Esta camada abstrai a utilização do JXTA-SOAP e facilita ainda mais a utilização dos serviços de comunicação do JXTA. Por fim, está a implementação do protótipo proposto, que possibilita que serviços de presença sejam integrados à sua estrutura para serem utilizados como mecanismo de notificação.

4.1.1 Comunicação entre os componentes

O desenvolvimento dos componentes da arquitetura proposta nesta dissertação foi realizado com base na API de criação de serviços de gerenciamento existente no ManP2P. Desta forma, em cada componente da arquitetura, há um serviço de gerenciamento responsável por receber as chamadas feitas por outros *peers* do *overlay* P2P. A comunicação entre estes componentes ocorre através da troca de mensagens SOAP, que carregam em seu conteúdo as mensagens definidas nesta dissertação e que serão apresentadas no decorrer desta subseção.

Entre as facilidades que o ManP2P possui em sua estrutura para aprimorar a tarefa de gerenciamento de redes, está o balanceamento de carga das tarefas de gerenciamento utilizando grupos de *peers*. Para possibilitar este balanceamento, o ManP2P possui cinco algoritmos de distribuição de carga que são usados para decidir qual *peer* deve ser utilizado para processar uma determinada requisição. Contudo, o estudo destes algorítimos não faz parte do escopo desta dissertação, uma vez que foram implementados e avaliados

por André Panisson (PANISSON, 2007), integrante do Grupo de Redes da UFRGS. O ManP2P também possibilita que uma chamada a um serviço de gerenciamento seja enviada a todos os *peers* que possuem este serviço, caracterizando assim, um mecanismo de *multicast* em nível de aplicação para o envio de mensagens. Este mecanismo de *multicast* também foi utilizado para a implementação de algumas comunicações entre os componentes da arquitetura proposta e que serão apresentadas a seguir.

Em um primeiro momento de execução do protótipo, os PUAs devem enviar a todos os *Brokers* uma relação de quais são os *Principals* disponíveis no sistema de gerenciamento. Este envio ocorre utilizando o mecanismo de *multicast* do ManP2P, onde um PUA faz uma chamada à operação *SetPrincipalList* disponível no serviço *BrokerService* que é implementado por todos os *Brokers*. Esta lista é formada a partir das informações descritas em um arquivo XML (*monitorsList.xml*) existente em cada PUA (a ser apresentado em maiores detalhes na subseção 4.3.3) e contém o nome e a descrição de todos os dispositivos e serviços que podem ser acessados pelos monitores. A Figura 4.2 apresenta um exemplo desta mensagem que é enviada pelos PUAs a todo o grupo de *Brokers*.

```
1 <puaDescriptor>
2         <name>PresenceUserAgent1 </name>
3         <element name="RouterIF0" description="Router interface 0" />
4         <element name="RouterIF1" description="Router interface 1" />
5         <element name="RouterIF2" description="Router interface 2" />
6 </puaDescriptor>
```

Figura 4.2: Mensagem enviada pelos PUAs descrevendo quais são os recursos disponíveis

Na Figura 4.2 percebe-se a existência de um atributo *<name>*, que é responsável por informar aos *Brokers* qual é o nome do serviço de gerenciamento disponível nos PUAs e que deverá ser invocado na etapa de registro. Como os PUAs podem formar grupos para balancear a carga sobre cada *peer*, deve-se ter o cuidado de fazer com que este atributo *<name>* seja relacionado a um conjunto específico de *Principals*. Os demais parâmetros são as informações que serão exibidas na interface gráfica disponível nos TLMs, para que os *Watchers* possam escolher sobre quais *Principals* querem ser notificados.

Em um segundo momento de execução do protótipo, correspondente à etapa de controle, o operador humano desempenhando o papel de controlador do sistema visualiza, em sua interface gráfica, quais são os serviços de presença que podem ser utilizados como mecanismo de notificação. A obtenção de quais são os serviços de presença disponíveis é realizada por uma chamada a operação *GetPresenceServiceList* disponível no serviço *BrokerService*, utilizando o balanceamento de carga do ManP2P. Após o controlador definir as situações em que um determinado serviço de presença deve ser utilizado, é feita uma chamada a operação *RegisterRule* disponível também no serviço *BrokerService*. Esta chamada é realizada utilizando *multicast*, onde todos os *Brokers* a recebem e a processam. A Figura 4.3 apresenta um exemplo da mensagem que é enviada aos *Brokers* pelos controladores. Nesta mensagem, estão representados o serviço de presença que deve ser utilizado para notificar os *Watchers* e em quais situações.

Em um terceiro momento de execução do protótipo, correspondente a etapa de registro, o administrador interessado em ser notificado sobre o estado de algum recurso, visualiza na interface gráfica os nomes e as descrições dos *Principals* disponíveis no sistema de gerenciamento. Estas são as mesmas informações enviadas pelos PUAs ao grupo de *Brokers*. Esta solicitação ocorre através de uma chamada a operação *GetPrincipal-*

```
<ruleDescriptor>
        <rule>
                <mechanism>xmpp</mechanism>
                <options messageType="*" principalType="*"
                        durationTime="3600000" log="true"
                        periodicity="60000" onChange="true" />
        </rule>
</ruleDescriptor>
```

Figura 4.3: Mensagem enviada pelos controladores definindo qual serviço de presença deve ser utilizado em determinada situação

List, também disponível no *BrokerService*, porém, apenas um *Broker* é o responsável por processá-la. Logo após, o administrador escolhe sobre quais recursos quer receber informações de presença e em quais situações. Em seguida, é realizada uma chamada à operação *Register* disponível no *BrokerService* e que é processada por apenas um *Broker*. Este *Broker*, escolhido por balanceamento de carga, será o responsável por decidir qual serviço de presença deve ser utilizado e é responsável também por enviar as mensagens de configuração aos *peers* envolvidos na comunicação.

```
<PresenceDescriptor>
        <watcher value="WatcherService1" />
        <presence>
                <from>RouterIF0</from>
                <from>RouterIF1</from>
        </presence>
        <options messageType="*"
                durationTime="3600000" log="true"
                periodicity="60000" onChange="true" />
        <options/>
</PresenceDescriptor>
```

Figura 4.4: Mensagem de registro enviada pelos adminitradores

A mensagem de registro que é enviada aos *Brokers* pelos administradores, apresentada na Figura 4.4, contém um atributo <*watcher*> que é responsável por informar aos *Brokers* qual é o nome do serviço de gerenciamento disponível nos *Watchers* e que deverá ser invocado quando o *Broker* estiver enviando as mensagens de configuração. O valor deste atributo <*watcher*> deve ser único no *overlay* P2P, uma vez que identifica qual *peer* deve receber as informações de presença. Os demais parâmetros representam sobre quais recursos o administrador quer ser notificado e em qual situação.

Com base nas mensagens de controle enviadas pelos controladores e nas mensagens de registro enviadas pelos administradores, o mecanismo de seleção interno em cada *Broker* é capaz de decidir qual é o serviço de presença que deve ser utilizado para notificar os administradores interessados no estado dos recursos disponibilizados. Após tomar esta decisão, o *Broker* cria uma mensagem de configuração que é enviada aos PUAs e aos *Watchers* através de uma chamada a operação *Configure*, disponível nos serviços de gerenciamento de ambos os *peers*. Caso o serviço de presença escolhido seja o XMPP ou o SIMPLE, torna-se necessário obter informações sobre o funcionamento dos respectivos PAs, que é realizado através de uma chamada a operação *GetPAinfo*, disponível no *peer* PA.

A Figura 4.5 representa uma mensagem de configuração enviada ao *Watcher* em que o

```
<presenceConfiguration>
        <mechanism>xmpp</mechanism>
        <specificConfig>143.54.12.160:5222</specificConfig>
</presenceConfiguration>
```

Figura 4.5: Mensagem de configuração enviada aos *Watchers*

```
<presenceConfiguration>
        <watcher>WatcherService1@143.54.12.160/smack</watcher>
        <presence>
                <from>RouterIF0</from>
                <from>RouterIF1</from>
        </presence>
        <options messageType="*"
                durationTime="3600000" log="true"
                periodicity="60000" onChange="true" />
        <mechanism>xmpp</mechanism>
        <specificConfig>143.54.12.160:5222</specificConfig>
</presenceConfiguration>
```

Figura 4.6: Mensagem de configuração enviada aos PUAs

serviço de presença escolhido é o XMPP. Neste exemplo, o atributo <*mechanism*> indica qual receptor o *Watcher* deve iniciar e o atributo <*specificConfig*> indica em qual PA este receptor deve se registrar. Na Figura 4.6, enviada ao PUA, o atributo <*watcher*> é responsável por informar ao PUA para qual cliente XMPP as mensagens devem ser enviadas. O atributo <*from*> informa quais são os monitores que devem ser utilizados para capturar as informações de presença. O atributo <*specificConfig*> também informa em qual PA o PUA deve se registrar. Por fim, os demais atributos representam em quais situações o PUA deve enviar as notificações.

4.1.2 Mecanismo de Seleção

Quando um *Broker* recebe uma mensagem de controle, ele a armazena em arquivo XML (*rules.xml*) e também a mantém em memória para diminuir o tempo de acesso às mensagens de controle, que são chamadas de regras nesta dissertação. O mecanismo de seleção utiliza estas regras definidas pelos controladores e as informações contidas nas mensagens de registro para fazer a escolha de qual serviço de presença deve ser utilizado para enviar as informações de presença.

Entre as opções que o controlador possui à sua disposição para criar as regras, existe a possibilidade de definir um serviço de presença padrão que será utilizado sempre que houver uma requisição por informações de presença de um determinado *Principal*. Isto ocorre por meio do parâmetro *Principal*, disponível na interface gráfica e que é representado pelo atributo <*principalType*>, como pode ser visto na Figura 4.3. Quando o *Broker* recebe uma regra com o valor deste atributo diferente de "*", desconsidera os demais parâmetros para tomar a decisão de qual serviço de presença deve ser utilizado. Nas demais situações, o mecanismo de seleção analisa cada um dos 5 parâmetros existentes nas mensagens de registro. A seleção ocorre basicamente em três etapas, que são apresentadas a seguir:

1. São analisados os parâmetros que podem apresentar valores previamente conhecidos pelos *Brokers*, tais como: <*log*>, <*onChange*> <*messageType*>;

2. O mecanismo de seleção, escolhe dentre as regras selecionadas na primeira etapa, quais possuem o valor de periodicidade (<*periodicity*>) mais próximos aos definidos pelos administradores;

3. Por fim, é escolhida a regra que possui o valor para tempo de duração do registro (<*durationTime*>) mais próximo ao definido pelo administrador.

Estas etapas, cujo algoritmo é apresentado na Figura 4.7, podem originar situações em que mais de uma regra pode ser utilizada para um determinado pedido de registro, porém, a implementação do controlador evita tais situações escolhendo a primeira regra que se enquadra nos parâmetros informados. Na figura, o atributo *Registry* corresponde à mensagem de registro enviada pelos administradores, o atributo *Rules* representa todas as regras que já foram definidas pelos controladores, por fim, os atributos *rulesToApply** representam o conjunto de regras que vão sendo selecionadas em cada etapa e ao final apenas uma regra é obtida.

```
1  private Rule getRuleToApply(Registry registry) {
2          List rulesToApply1, rulesToApply2, rulesToApply3;
3          Rule ruleTmp; long timeTmp;
4
5          for(Rules rule: rules) {
6                  if (rule.onChange == registry.onChange &&
7                  rule.messageType == registry.messageTypes &&
8                  rule.log == registry.log) {
9                          rulesToapply.add(registry);
10                 }
11         }
12         for(Rules rule: rulesToapply) {
13                 long timeDelta = Math.abs(rule.periodicity - registry.periodicity);
14                 if (timeDelta < timeTmp) {
15                         ruleToApply2.erase(); ruleToApply2.add(rule);
16                 } else if (timeDelta = timeTmp) {
17                         ruleToApply2.add(rule);
18                 }
19         }
20         for(Rules rule: ruleToApply2) {
21                 long timeDelta = Math.abs(rule.periodicity - registry.periodicity);
22                 if (timeDelta < timeTmp){
23                         ruleToApply3.erase(); ruleToApply3.add(rule);
24                 } else if (timeDelta = timeTmp) {
25                         ruleToApply3.add(rule);
26                 }
27         }
28         if (rulesToApply3.size == 1) {
29                 return rulesToApply3.get(0);
30         } else {
31                 return null;
32         }
33 }
```

Figura 4.7: Algoritmo de seleção

Caso nenhuma regra atenda às necessidades dos administradores, o mecanismo de seleção utiliza uma regra padrão, armazenada no arquivo *defaultRule.xml* e que é definida pelo controlador em sua interface gráfica. A Figura 4.8 apresenta um exemplo de regra definida pelo controlador em que o serviço de presença padrão é o XMPP.

4.1.3 Controle do Ciclo de Vida dos Monitores

Para possibilitar o acesso às informações de presença dos recursos gerenciados, definiu-se nesta dissertação componentes denominados monitores, que são os responsáveis por

```
<ruleDescriptor>
        <defaultRule>
                <mechanism>xmpp<mechanism>
        </defaultRule>
</ruleDescriptor>
```

Figura 4.8: Arquivo *defaultRule_xml* que contém o serviço de presença padrão

implementar a comunicação com os dispositivos e serviços. Estes monitores possuem um ciclo de vida bem definido, em que é especificado como cada monitor é carregado, instanciado, inicializado, como e quando suas operações são chamadas e como é finalizado. As fases de carga e finalização são expressas através dos métodos *init()* e *destroy()* da interface *MonitorLifecycle* e o controle deste ciclo de vida é de responsabilidade do PUA.

Primeiramente, o PUA carrega a classe (*i.e.*, o monitor) que implementa a comunicação com o *Principal* definido na mensagem de registro. Esta carga ocorre utilizando a API *Reflection*, disponível na plataforma Java, para obter em tempo de execução meta-informações a respeito de classes, como, por exemplo, quais interfaces implementa, quais são seus métodos e quais são seus atributos. Depois de carregar as classes dos monitores, o PUA instancia os objetos destas classes para serem utilizados. Após a instanciação, o PUA inicializa cada objeto monitor antes deste começar a acessar as informações de estado dos *Principals*. A inicialização ocorre para possibilitar que os monitores localizem dados de configuração, iniciem recursos custosos e executem outras atividades de preparação para sua execução.

O PUA inicializa o objeto monitor invocando o método *init()* da interface *MonitorLifecycle*, passando como parâmetro um objeto *Configuration* que representa a mensagem de configuração recebida dos *Brokers*. Este objeto de configuração permite que os monitores saibam em qual momento devem buscar pelas informações de presença dos *Principals*. Esse objeto de configuração também possibilita que os monitores saibam quais informações devem obter para enviar aos *Watchers*, por exemplo, se o interessado na informação quer receber apenas informações de estado, apenas informações de disponibilidade, uma informação específica, ou então se querem receber todas as informações possíveis.

Após ter iniciado o objeto monitor, o PUA utiliza a informação contida no atributo *<durationTime>* da mensagem de configuração para saber em qual momento deve invocar o método *destroy()* do monitor, para que os recursos possam ser liberados e o monitor pare de enviar as informações de presença.

4.2 Serviço de Presença Proposto

O desenvolvimento do serviço de presença proposto teve como base o *framework* JXTA para implementar a comunicação entre os PUAs e os *Watchers*. Foram utilizados três meios de comunicação disponíveis neste *framework* e que serão explicados a seguir.

O mecanismo de *pipes* (MICROSYSTEMS, 2005) é a principal forma de comunicação disponível no JXTA. Os *pipes* podem ser vistos como canais de comunicação virtuais, unidirecionais, assíncronos e não confiáveis que conectam dois ou mais *peers* do *overlay* P2P. O mecanismo de *pipes* fornece dois modos de comunicação: ponto-a-ponto e por propagação. No primeiro, utilizado no desenvolvimento do serviço de presença proposto, um *peer* se comunica diretamente com apenas um outro *peer*. No modo de propagação um *pipe* de saída é conectado a vários *pipes* de entrada e a mensagem flui do *pipe* de

saída para todos os *pipes* de entrada e toda a comunicação é feita num escopo de grupo de *peers*. Este modo de propagação, pode ser usado por exemplo, para implementar uma comunicação *multicast*.

O segundo meio de comunicação utilizado no desenvolvimento do serviço de presença proposto foi o de *sockets* do JXTA. Estes *sockets* são canais de comunicação bi-direcionais e que são construídos utilizando *pipes*. O JXTA *Socket* é uma implementação de *sockets* tradicionais em cima da infra-estrutura JXTA. O terceiro e último meio de comunicação utilizado foi o JXTA-SOAP, explicado anteriormente e que é o mecanismo utilizado pelo ManP2P para trocar mensagens entre os *peers*. Sendo assim, a Figura 4.9 apresenta a infra-estrutura de comunicação que foi utilizada para o desenvolvimento do serviço de presença proposto nesta dissertação.

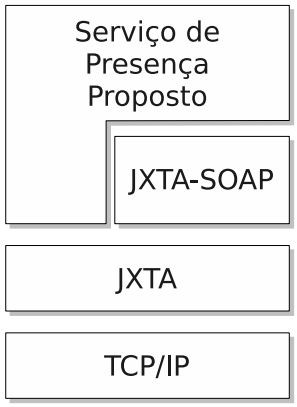

Figura 4.9: Infra-estrutura de comunicação utilizada no serviço de presença proposto

O serviço de presença proposto, possui a capacidade de repassar duas informações de presença: estado e disponibilidade. Além destas duas informações, há um campo adicional (<*specific*>) disponível nas mensagens de notificação que pode ser utilizado para enviar informações definidas pelos desenvolvedores dos monitores e que não fazem parte das definições desta dissertação. Por exemplo, os desenvolvedores de monitores, podem utilizar este campo para enviar informações específicas sobre o funcionamento de cada *Principal*, tais como carga de processamento e número de conexões. Cabe lembrar que estas informações serão exibidas aos administradores da mesma forma como elas forem formatadas pelos desenvolvedores dos monitores.

4.3 Integração e Extensão

Na implementação da arquitetura proposta (feita em Java), foi desenvolvida uma (API) que os desenvolvedores devem utilizar para adicionar novos serviços de presença ao sistema de gerenciamento apresentado. Também foi definida uma API para o desenvolvimento dos monitores e que permite aos PUAs controlar seus ciclos de vida. Sendo assim, nesta seção serão apresentadas detalhadamente cada uma destas APIs e como elas podem ser utilizadas para ampliar as funcionalidades do protótipo desenvolvido nesta dissertação.

4.3.1 Adição de Novos Serviços de Presença

Para adicionar um novo serviço de presença à estrutura do protótipo desenvolvido, duas etapas devem ser seguidas: criação dos receptores e dos mecanismos de envio. Os receptores são os componentes de *software* responsáveis por implementar a recepção, por parte dos *Watchers*, das informações de presença, enquanto que os mecanismos de envio, são utilizados pelos PUAs para enviar tais informações. Nos *Watchers*, há elementos denominados *listeners*, que são responsáveis por processar as informações de presença recebidas pelos receptores. Um exemplo de um *listener* é o que foi utilizado no desenvolvimento do protótipo desta dissertação e que é responsável por exibir as informações de presença na interface gráfica.

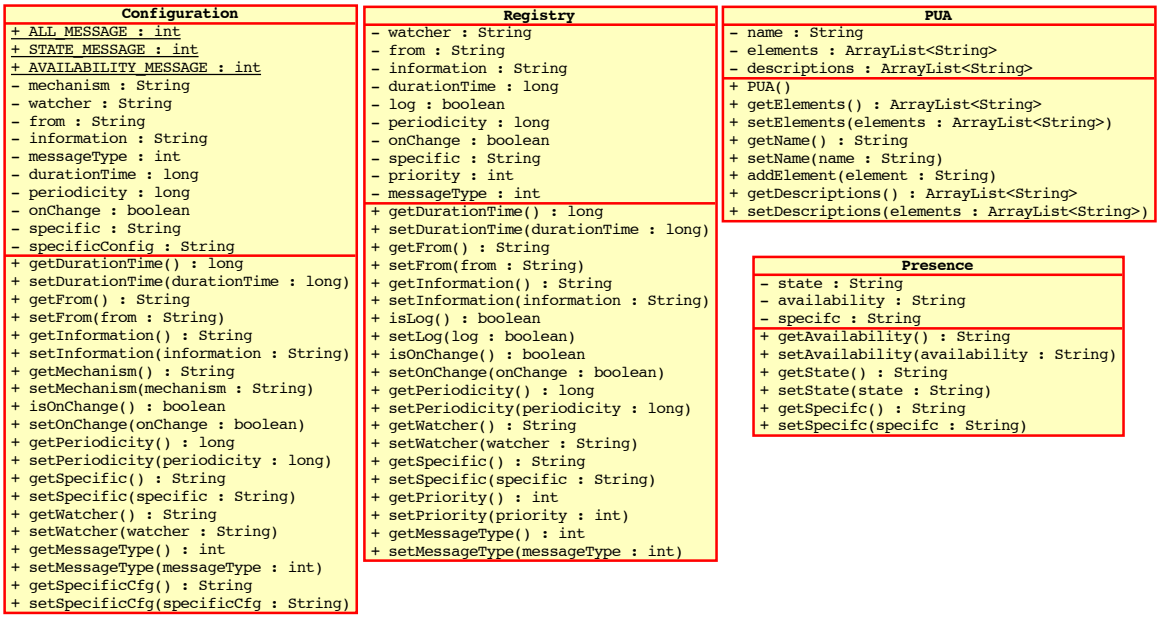

Figura 4.10: Diagrama de classes VOs (*Value Objects*) utilizadas no protótipo desenvolvido

Para criar um receptor, um desenvolvedor deve inicialmente implementar a interface *Receiver*. Esta interface possui dois métodos que devem ser implementados: o método *getPresence()* e o método *fireProcessPresence()*. Estes dois métodos são utilizados pelo *Watcher* para notificar os *listeners* que uma informação de presença foi recebida e que deve ser processada. Para instanciar um receptor, o desenvolvedor deve fazer uma comparação dentro do método *createReceiver()*, da classe *WatcherService*, para verificar se o mecanismo definido na mensagem de configuração é o mesmo que o desenvolvedor está adicionando. Esta classe *WatcherService* é a implementação do serviço de gerenciamento disponível nos *Watchers* e que possui a operação *configure()*, que será invocada pelos *Brokers* para enviar a mensagem de configuração. Ainda na classe que implementa a interface *Receiver*, o desenvolvedor deve instanciar um objeto do tipo *Presence*, que possui três atributos: *state, availability* e *specific*. Esta classe *Presence* pode ser vista na Figura 4.10, onde também são apresentadas: a classe PUA, que representa as mensagens enviadas pelos PUAs aos *Brokers* contendo informações sobre os *Principals*, a classe *Configuration*, que representa a mensagem de configuração enviada pelos *Brokers* e a classe *Registry*, que representa a mensagem de registro enviada pelo *Watcher*.

Para criar um novo *listener*, ou seja, a classe que de fato vai tratar a informação de

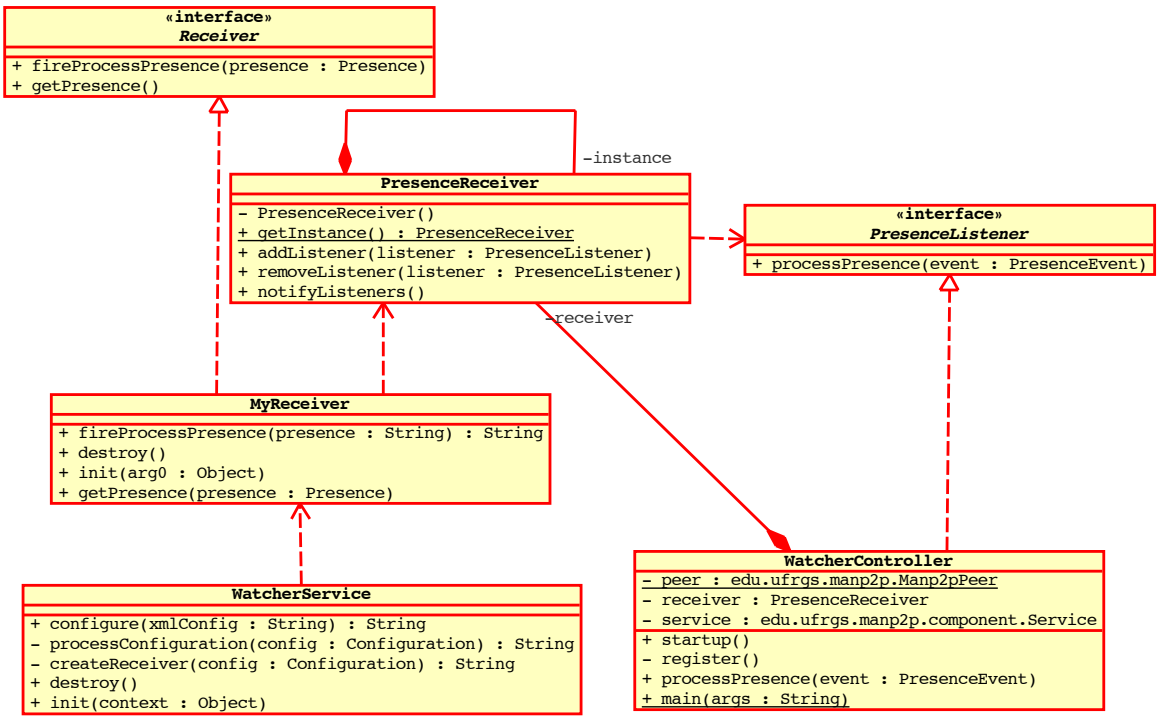

Figura 4.11: API para desenvolvimento dos receptores

presença, um desenvolvedor deve criar uma classe que implementa a interface *PresenceListener*, que obriga a implementação do método *processPresence()*. É neste método que o desenvolvedor recebe um objeto do tipo *Presence* e a partir daí, pode processá-lo. Na implementação do protótipo, quando o objeto da classe *WatcherController* recebe este objeto *Presence*, é realizada uma chamada aos métodos que implementam a interface gráfica, passando como parâmetro este objeto *Presence*. Todas estas classes e seus respectivos métodos podem ser visualizados na Figura 4.11.

Para adicionar um novo mecanismo de envio nos PUAs, um desenvolvedor deve implementar a classe *Sender*. Esta classe possui dois métodos que são relacionados com o envio das informações de presença aos *Watchers*: o método *send()* e o método *setConfiguration()*. O primeiro é responsável por implementar de fato a comunicação com o PA, no caso do XMPP e do SIMPLE, ou diretamente com o *Watcher*, no caso do serviço de presença desenvolvido. O método *setConfiguration()*, por sua vez, é o responsável por informar ao *Sender* os parâmetros necessários para ele se conectar ao PA, quando for o caso. Em seguida, o desenvolvedor deve adicionar uma comparação dentro método *getSender()* da classe *ConcreteSenderFactory*, onde o retorno é uma instância da classe que o desenvolvedor criou e que corresponde ao mecanismo definido na mensagem de configuração. A forma como foi projetado o serviço de envio segue o padrão de projeto criacional *Factory Method* (GAMMA et al., 1995). A Figura 4.11 apresenta estas classes e os métodos que devem ser utilizados.

4.3.2 Integração com os serviços de presença selecionados

Com o objetivo de integrar os dois serviços de presença selecionados na seção 2.2.8, utilizou-se *frameworks* específicos para o desenvolvimento dos receptores e dos mecanismos de envio. Para o desenvolvimento dos receptores e dos mecanismos de envio do

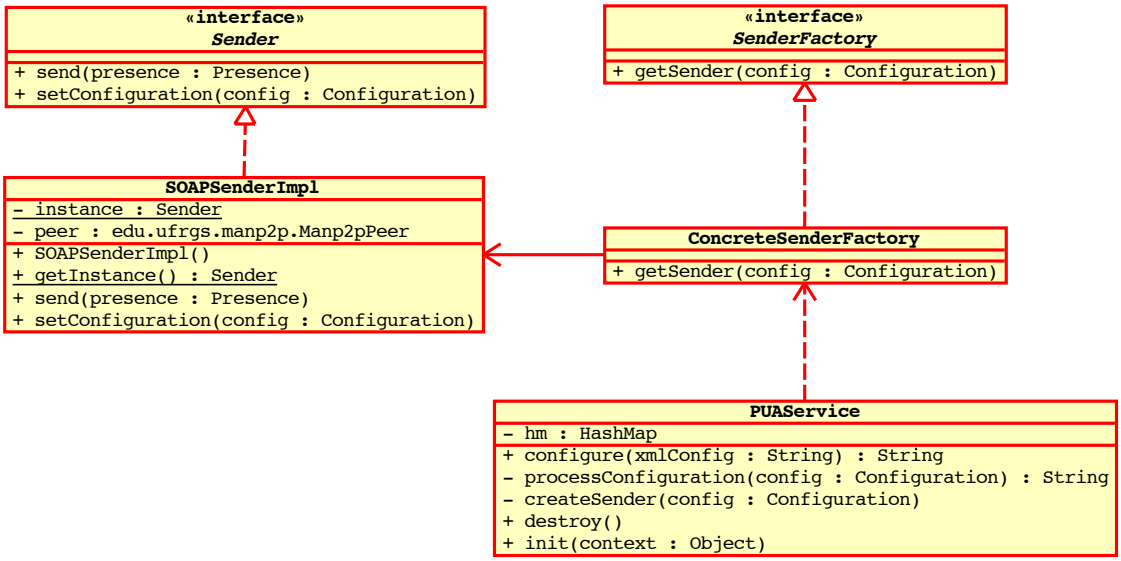

Figura 4.12: API para desenvolvimento dos mecanismos de envio

XMPP, por exemplo, utilizou-se o *framework* Smack (SOFTWARE, 2006), que é um *framework* de código aberto e implementado em Java. Como solução de PA utilizou-se o projeto *Tigase Server* (TIGASE.ORG, 2007), desenvolvido em Java e que possui seu código aberto e livre. O fato de utilizar um PA desenvolvido em Java dá a possibilidade de executá-lo em diferentes plataformas de *hardware* e *software*. A Figura 4.13 apresenta as classes que foram utilizadas no desenvolvimento do receptor e do mecanismo de envio do XMPP.

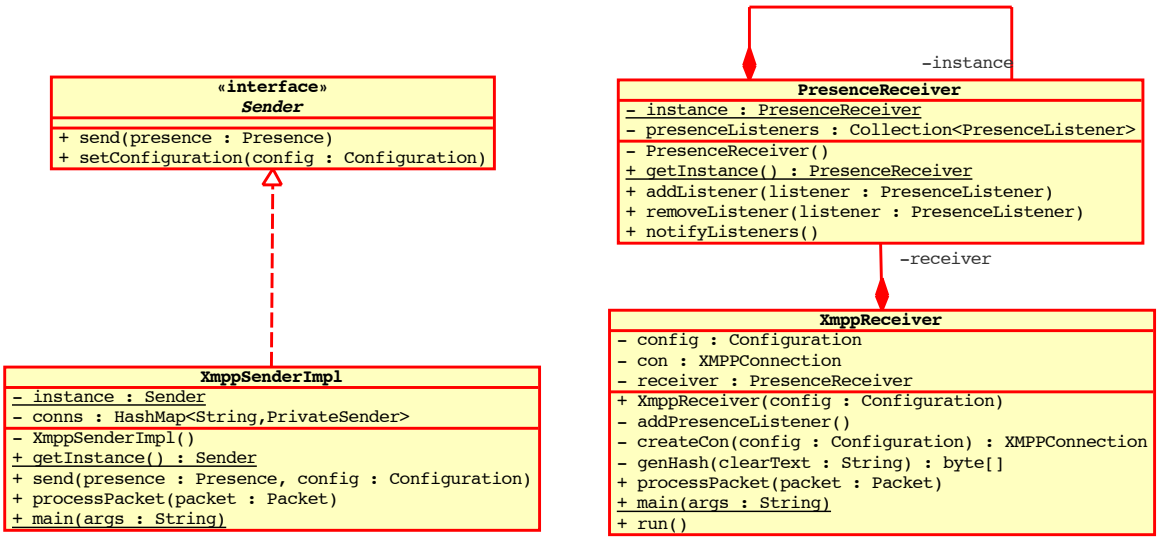

Figura 4.13: Classes utilizadas no receptor e no mecanismo de envio XMPP

Quanto ao SIMPLE, não foram encontrados *frameworks* desenvolvidos em Java como solução para o PA (com funções de *Registrar* + *Proxy*), portanto, utilizou-se o *Sip Express Router* (SER) (IPTEL.ORG, 2007), que é um servidor SIP de código livre e sob a licença GNU. O SER, foi desenvolvido utilizando a linguagem de programação C e, atualmente, está disponível apenas para sistemas Unix. Para o desenvolvimento do receptor e do me-

canismo de envio do SIMPLE, utilizou-se o Jain-SIP (STANDARDS; NIST, 2007), que é um *framework* em Java desenvolvido pelo *National Institute of Standards and Technology* (NIST), para desenvolver aplicações que se conectem com servidores SIMPLE. Na Figura 4.14, pode-se observar as classes que foram necessárias para desenvolver os receptores e os mecanismos de envio.

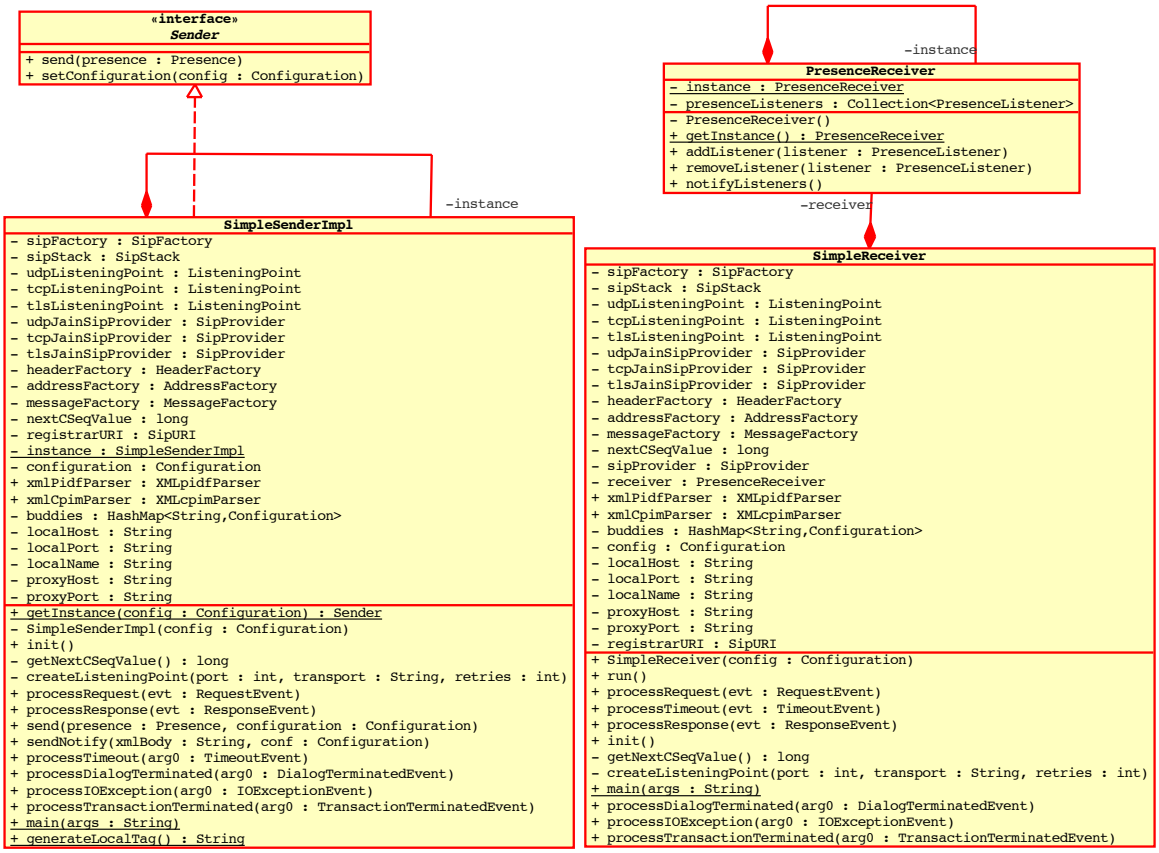

Figura 4.14: Classes utilizadas no receptor e no mecanismo de envio SIMPLE

4.3.3 Criação de Novos Monitores

Com o objetivo de manter o suporte a diferentes dispositivos e serviços que podem ser encontrados nas modernas redes de computadores, foi desenvolvida uma API que deve ser utilizada para que novos monitores possam ser criados e integrados ao protótipo desenvolvido e assim, as informações de estado de novos dispositivos e serviços possam ser encaminhadas aos *Watchers*.

Para criar um novo monitor, um desenvolvedor deve primeiro criar uma classe que implementa a classe abstrata *Monitor* que possui três métodos abstratos que devem ser implementados: os métodos *walkPresence()*, *init()* e *destroy()*. A função dos métodos *init()* e *destroy()* já foi explicada anteriormente, já o método *walkPresence()* é o método que é executado quando a informação de presença tiver que ser obtida dos *Principals*. O momento em que o método *walk* é executado depende do valor definido no parâmetro <*periodicity*> da mensagem de configuração. Este método retorna um objeto do tipo *Presence*, que é utilizado para gerar as mensagens que serão enviadas aos *Watchers* pelos mecanismos de envio. Na Figura 4.15 é apresentada a classe *Monitor* e três exemplos de monitores que foram desenvolvidos nesta dissertação.

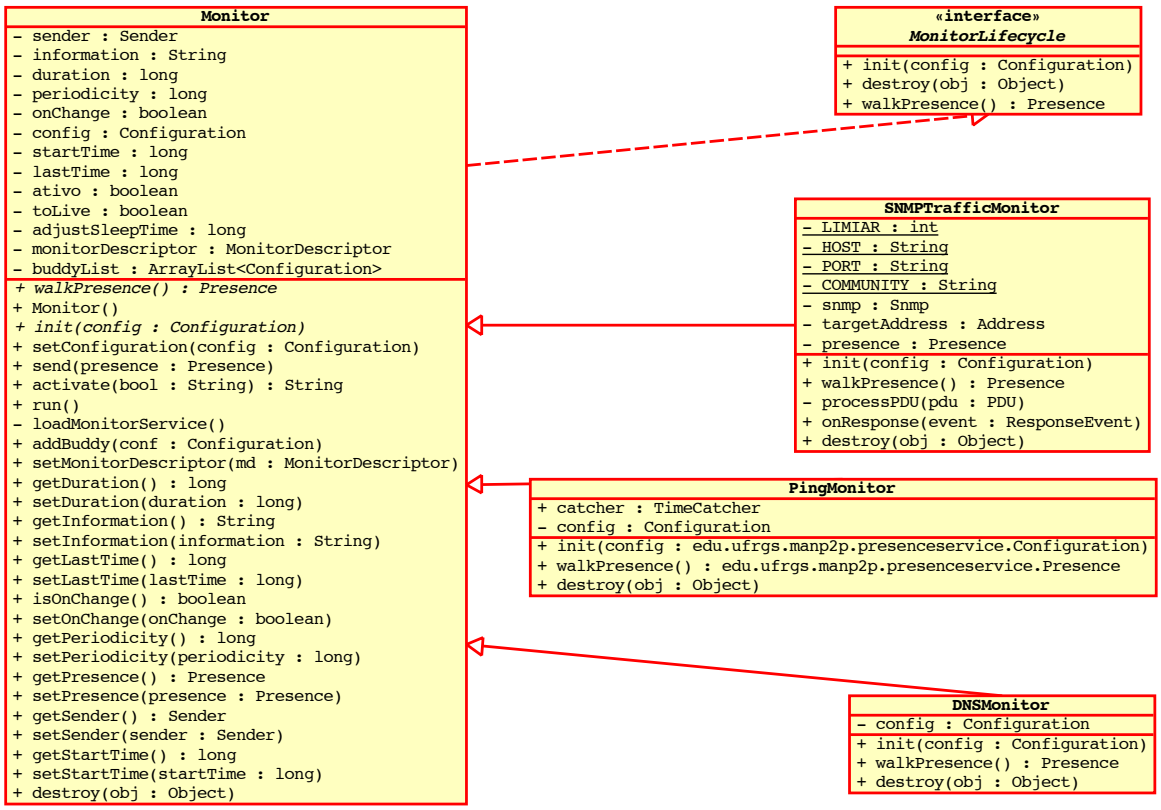

Figura 4.15: Diagrama de classes para desenvolvimento de monitores com três exemplos de monitores desenvolvidos

Após criar o novo monitor, o desenvolvedor deve disponibilizá-lo para ser utilizado pelo protótipo da arquitetura desenvolvido. Para isto, deve ser utilizado o arquivo *MonitorDescriptor.xml*, que será lido pelos PUAs ao iniciarem. Este arquivo define o nome do monitor (de preferência um que lembre o nome do dispositivo ou serviço a ele associado), uma breve descrição do elemento gerenciado e, por fim, a classe Java que o implementa. O conteúdo deste arquivo, é o mesmo que os PUAs irão enviar aos *Brokers* informando quais são os *Principals* disponíveis e que, por fim, será visualizado pelos administradores e pelos controladores na interface gráfica. Também podem ser definidos neste arquivo parâmetros que serão passados aos monitores no momento em que estes estiverem sendo inicializados pelos PUAs. Estes parâmetros podem, por exemplo, informar qual o endereço de rede do dispositivo ou serviço que será acessado. A Figura 4.16 apresenta um exemplo de um arquivo *MonitorDescriptor.xml* em que é definido um monitor que irá enviar mensagens do tipo ICMP para um endereço IP de rede.

```
<monitors>
        <monitor>
                <name>PingADSLRouterMonitor</name>
                <description>Monitor do Modem ADSL D-Link 502G</description>
                <classname>
                        edu.ufrgs.manp2p.presenceservice.pua.monitors.PingMonitor
                </classname>
                <init-param>
                        <param-name>address</param-name>
                        <param-value>10.1.1.1</param-value>
                </init-param>
        </monitor>
</monitors>
```

Figura 4.16: Arquivo de descrição dos monitores

4.4 Interface com o Usuário no ManP2P

A interface com os administradores e com os controladores foi implementada utilizando-se a API *Swing* do Java. Essa API se caracteriza como a melhor opção para desenvolver interfaces gráficas para aplicações Java independentes de plataforma. Esta interface gráfica foi então integrada a já existente no ManP2P e suas telas são apresentadas a seguir.

4.4.1 Visão do Administrador

Após logar-se no sistema de gerenciamento de redes ManP2P, o administrador já encontra a esquerda da tela a área de notificações, como apresentado na Figura 4.17. Nesta área são exibidos os nomes e as descrições dos *Principals* que o administrador já registrou interesse. Com base no estado de cada recurso gerenciado, sua representação pode ser exibida sob as cores: verde, vermelho e amarelo. A cor verde ocorre quando o estado do recurso gerenciado é *online*, a cor vermelha é para o caso de estar *offline*. A cor amarela foi implementada utilizando a informação <*specific*> definida no serviço de presença proposto. Ela é utilizada para informar que o recurso gerenciado está ativo, porém, merece alguma atenção. Por exemplo, o monitor *SNMPTrafficMonitor* faz o acesso por SNMP a um roteador, busca pelas informações *ifInOctets*, *ifOutOctets* e *SysUpTime*. Quando a diferença entre a quantidade de dados que entraram e a quantidade de dados que saíram em um determinado espaço de tempo ultrapassa um limiar (definido como parâmetro de inicialização no arquivo *MonitorDescriptor.xml*), é enviada uma notificação, com o campo <*specific*> informando que o limiar foi ultrapassado.

A segunda interface gráfica disponível aos administradores é a de registro, apresentada na Figura 4.18. Para acessar esta interface gráfica o administrador deve selecionar a segunda aba da área de notificações. Nesta interface, o administrador pode fazer uma busca por *Principals* específicos através do campo *Resource*. Abaixo do campo de busca, são listados todos os *Principals* disponíveis e que foram encontrados. O administrador pode então selecionar em quais quer se registrar para ser notificado e então especificar em quais situações e como ele quer ser notificado, utilizando para isto, os campos: *Duration Time, Information, Log, On change* e *Periodicity*. Logo após, o administrador deve clicar no botão *Subscribe* e quando o registro for efetuado, será exibida uma mensagem informando-o do sucesso da operação.

4.4.2 Visão do Controlador

A última visão disponível na área de notificações, apresentada na Figura 4.19, é a visão que o controlador tem para definir as regras. O primeiro campo que o controlador tem a sua disposição é o *Presence Service*, onde poderá definir qual é o serviço de presença que será utilizado como mecanismo de notificação nas situações que ele irá definir com os demais parâmetros. O segundo campo é o *Resource*, com o qual o controlador pode associar um serviço de presença a um *Principal* específico, tornando desnecessário o preenchimento dos demais campos. Contudo, se este não for o caso, o controlador pode definir as situações em que o serviço de presença escolhido será utilizado. Por fim, o controlador deve clicar no botão *Send* para enviar as regras a todo o grupo de *Brokers*.

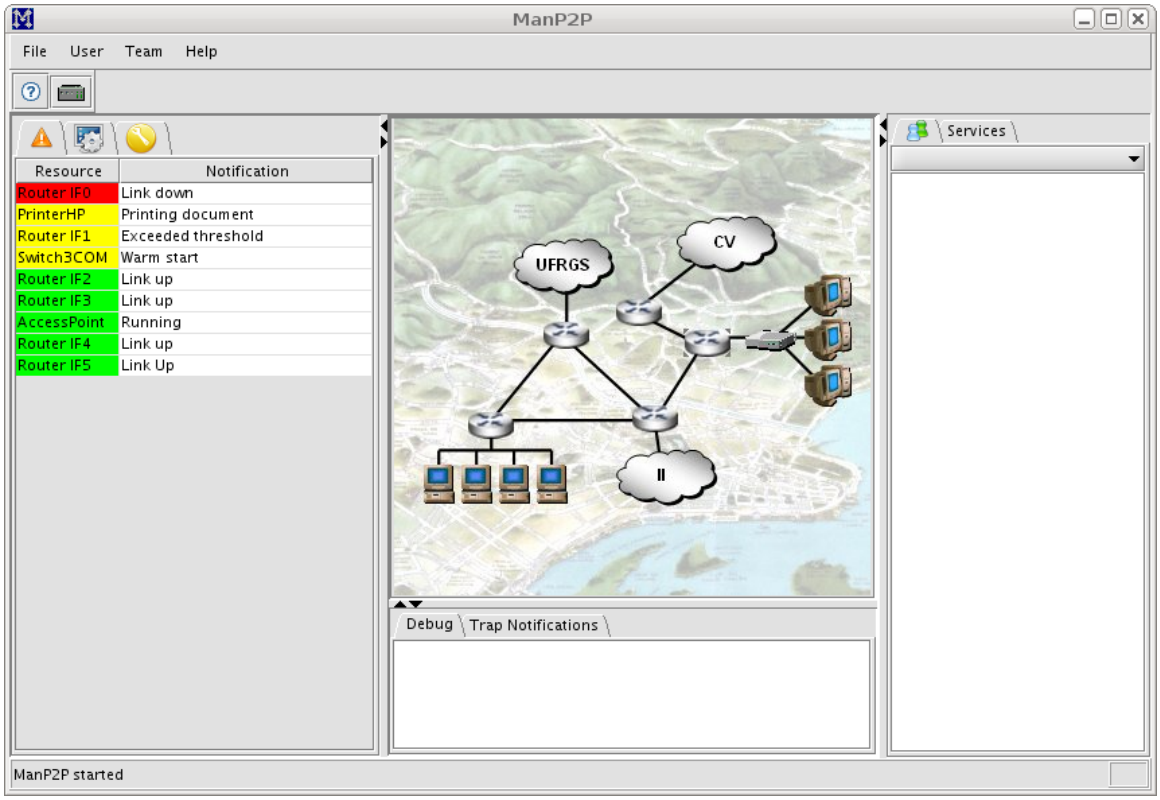

Figura 4.17: Visão das notificações

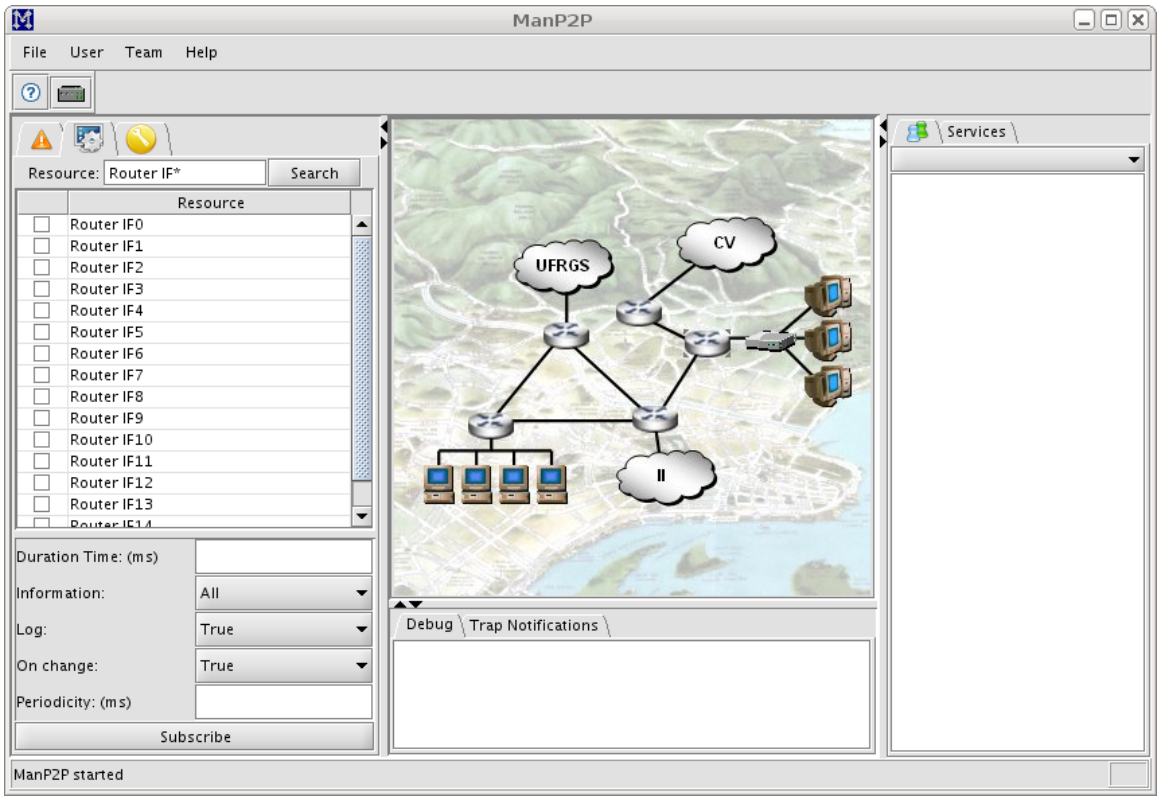

Figura 4.18: Visão de registro

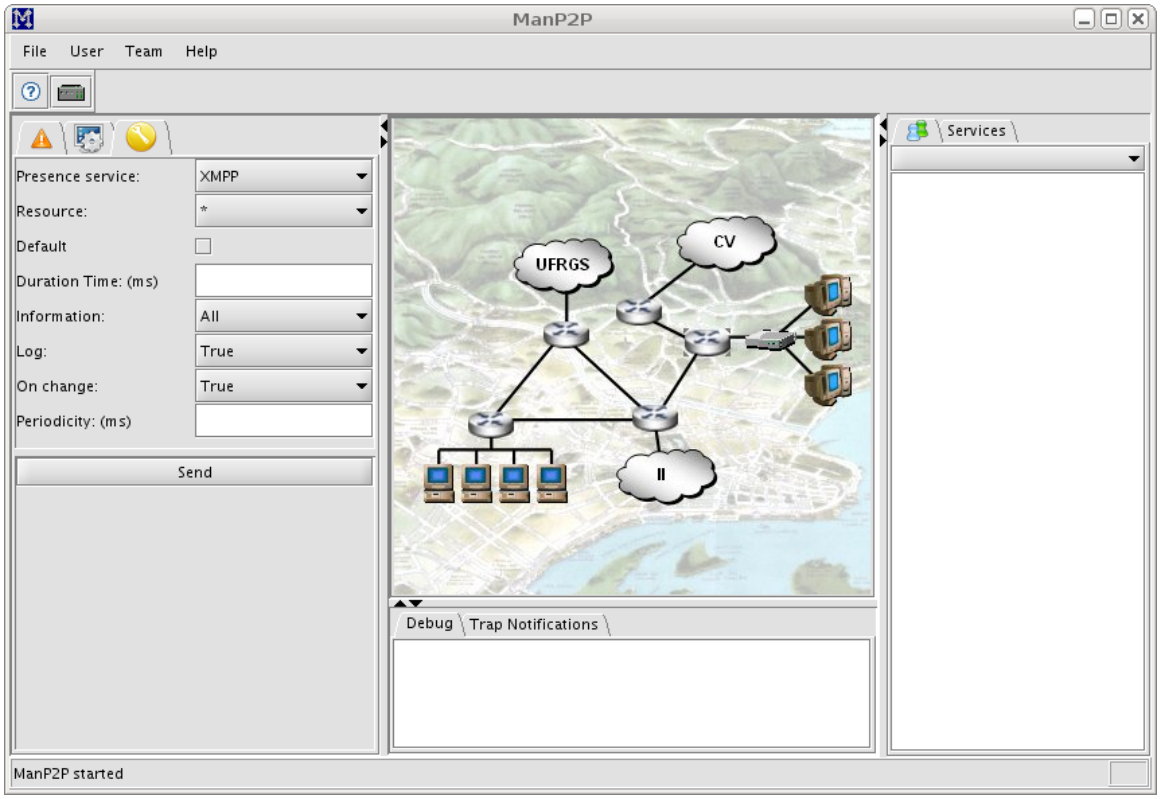

Figura 4.19: Visão de controle

5 AVALIAÇÃO DE DESEMPENHO

Neste capítulo é apresentada a avaliação de desempenho do protótipo desenvolvido, junto com os serviços de presença que foram integrados à sua estrutura. Esta avaliação foi realizada em um cenário de teste real formado por um conjunto de computadores, onde foram avaliadas as seguintes métricas: tempo da etapa de registro e da etapa de notificação (SANTOS et al., 2008); consumo total de banda (*i.e.*, tráfego de dados e de controle), consumo útil de banda (*i.e.*, tráfego de dados); taxa máxima de envio de notificações e perda de mensagens.

5.1 Ambiente de Testes

Os testes foram realizados em um dos laboratórios de graduação do Instituto de Informática da Universidade Federal do Rio Grande do Sul (UFRGS). Durante a execução dos testes, o laboratório não foi utilizado para outros fins, ou seja, não haviam outras aplicações fazendo uso da rede. Desta forma, obteu-se um ambiente estável e previsível. Os testes foram realizados utilizando até 19 computadores (Dell Optiplex GX270) conectados por uma rede *FastEthernet* de 100Mbps via um HUB (3Com 3c16611). As configurações de *hardware* e *software* utilizadas podem ser vistas na Tabela 5.1.

Tabela 5.1: Configuração dos computadores

Processador	Pentium IV 2.8 GHz
Memória RAM	512 MB
Sistema Operacional	Gnu/Linux Ubuntu (Kernel 2.6.20-16)
Java	J2SE (build 1.6.0_03-b05)

Cada computador hospedou um único *peer* do *overlay*. Cada *peer* por sua vez era responsável por disponibilizar apenas um único serviço de gerenciamento aos *peers* remotos, estes serviços de gerenciamento são os mesmos apresentados no capítulo 4 e são responsáveis por implementar as funções dos *Brokers*, *Watchers*, PUAs e PAs. Além disso, um computador ficou responsável por hospedar o *Rendezvous Peer*, necessário para manter a infra-estrutura P2P do JXTA.

5.2 Metodologia de Testes

Foram utilizadas três abordagens diferentes para avaliar as métricas apresentadas: análise do tráfego gerado pelo protótipo e pelos serviços de presença utilizados; medição do tempo decorrido de um procedimento executado em um mesmo computador e medição do tempo decorrido em uma interação entre dois computadores.

A primeira abordagem foi utilizada para avaliar o consumo de banda (total e útil) gerada na etapa de notificação. Por consumo de banda total entende-se o somatório de dados que continham as informações de presença, mais os dados de controle, gerados por cada serviço de presença utilizado e pelo protótipo. Nesta abordagem, um computador, além dos utilizados para executar os *peers*, foi utilizado como *sniffer* para capturar todos os pacotes que foram enviados. Estes pacotes foram capturados utilizando o *software* Wireshark 0.99.6, que é uma evolução do *Ethereal*. Com este *software* foram criados dois filtros distintos de pacotes: um responsável por selecionar todos os pacotes capturados desde a primeira informação de presença até a última, enquanto que o segundo era responsável por selecionar apenas os pacotes que correspondiam às mensagens que carregavam as informações de presença.

A segunda abordagem foi utilizada para avaliar o tempo da etapa de registro, onde um pedido de registro era enviado ao grupo de *Brokers* que após configurar corretamente o sistema, retornava uma mensagem de confirmação. Desta forma, o código fonte foi instrumentado para registrar *timestamps* no momento em que a mensagem de registro era enviada e no momento que sua confirmação era retornada. Estes tempos foram capturados utilizando o método *currentTimeMillis()* da classe *System*, disponível na plataforma Java. O impacto dessa instrumentação nos tempos de execução foi considerado irrelevante e foi ignorado.

Para avaliar as situações que envolviam a captura de tempos em dois computadores diferentes (*i.e.*, etapa de notificação e taxa máxima de envio), percebeu-se que mesmo com o *Network Time Protocol* (NTP) corretamente configurado, ainda poderiam haver inconsistências nos tempos registrados. Portanto, utilizou-se uma abordagem que além de registrar o *timestamp* no momento de envio e recepção das notificações, eram enviadas mensagens ICMP para um ponto específico da rede. Apesar de fugir do escopo desta dissertação, esta técnica foi avaliada e os resultados mostraram uma precisão de 99,2% em relação ao tempo real e que, portanto, poderia ser empregada sem grandes impactos nos tempos avaliados. O computador que recebia as mensagens ICMP era o mesmo responsável por fazer a captura de todo o tráfego gerado utilizando o *software* Wireshark. Este *software* também foi utilizado para ler e exportar os dados capturados para arquivos CSV (*Comma Separated Values*), que posteriormente foram analisados utilizando o *software* OpenOffice.org, que também foi utilizado para gerar os gráficos.

Por fim, dependendo da métrica que se queria avaliar, um número de execuções (*i.e.*, pedidos de registro ou notificações enviadas) diferente foi utilizado. Por exemplo, para avaliar o tempo de registro e o atraso da notificação, foram realizadas 58 execuções, o que garantiu um intervalo de confiança de 90%. Por outro lado, para avaliar a taxa máxima de envio de notificações, observou-se uma alta variabilidade, tornando necessário que 200 execuções fossem realizadas para atingir o mesmo intervalo de confiança. Não foram avaliados os tempos de obtenção das informações de presença dos *Principals*, uma vez que o acesso aos dispositivos ou serviços não faz parte do escopo desta dissertação.

Os cenários que foram avaliados são apresentados na próxima seção.

5.3 Cenários Avaliados

Na execução dos testes, foram utilizados diferentes cenários de acordo com a métrica que se queria avaliar. No entanto, todos os cenários foram elaborados com base em um mais simples composto por 5 computadores, apresentado na Figura 5.1, que possuía exatamente um *peer Watcher*, um *peer Broker*, um *peer* PUA e um *peer* PA. Além destes há o *Rendesvouz Peer* necessário para manter a infra-estrutura P2P do JXTA. Além destes computadores, havia um outro responsável por fazer a captura dos pacotes.

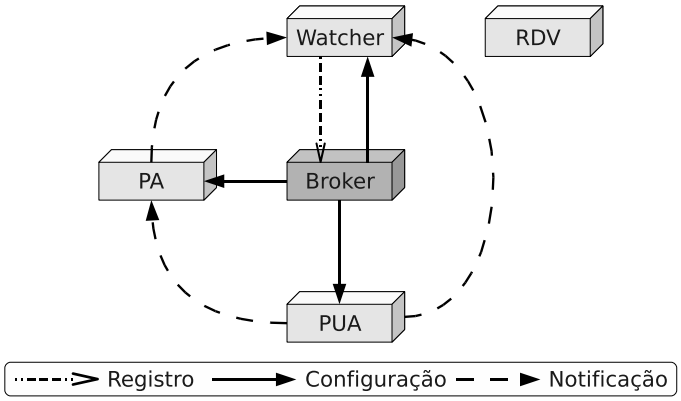

Figura 5.1: Cenário básico

Este cenário mais simples foi utilizado na avaliação de todas as métricas, contudo, para medir o tempo de registro e o tempo de notificação foram utilizados dois outros cenários em que o número de *peers* foi aumentado. Desta forma, no cenário intermediário utilizou-se 4 *Watchers* e 2 PUAs, enquanto o número dos outros *peers* permaneceu inalterado. No maior cenário utilizado, o número de *peers* foi multiplicado por 2: 2 *Brokers*, 4 PUAs e 8 *Watchers*. Mesmo nestes cenários maiores, ainda se manteve um único *Rendesvouz Peer*.

5.4 Avaliação do Tempo de Registro e de Notificação

Como já citado, a avaliação do tempo de registro e do tempo de notificação foi realizada utilizando três cenários com números diferentes de *peers*. Desta forma, foi possível observar a escalabilidade de alguns *peers* do protótipo, quando expostos a situações em que o número de requisições cresce. No cenário intermediário, os 4 *Watchers* se registravam ao mesmo tempo no único *Broker* disponível, sendo que dois *Watchers* requisitavam por informações de presença geradas pelo primeiro PUA, enquanto que os outros dois requisitavam por informações de presença do segundo PUA. No maior cenário utilizado, a mesma divisão de pedidos de registro foi mantida, contudo, com o dobro do número de *peers*. Em cada cenário, as informações (*i.e.*, mensagens de registro ou notificações), foram enviadas periodicamente, até atingir o número de 58 execuções. Os resultados destas avaliações são apresentados em gráficos de barra, uma vez que existem cenários intermediários (com variações no número de *peers*) entre os utilizados e que não foram avaliados.

Na Figura 5.2 observa-se o tempo médio gasto durante a etapa de registro, que envolve o envio do pedido de registro pelos administradores, a seleção do serviço de presença a ser utilizado, o envio das mensagens de configuração aos *peers* envolvidos na comunica-

ção e a confirmação da execução de todos estes passos. Pode-se observar que o serviço de presença mais eficiente foi o desenvolvido nesta dissertação quando os mecanismos de comunicação utilizados eram o JXTA *Pipes* ou JXTA *Socket*. Isto ocorre porque o serviço de presença proposto não precisa registrar-se em um ponto único do sistema (*e.g.*, servidores). Esta característica pode ser utilizada quando os administradores precisam receber imediatamente as informações de presença. Percebe-se que o mecanismo de comunicação JXTA-SOAP não é apresentado na Figura 5.2, uma vez que apresentou o pior tempo dentre os avaliados (aproximadamente 5 vezes maior que o maior tempo encontrado) prejudicando assim a visualização dos demais valores.

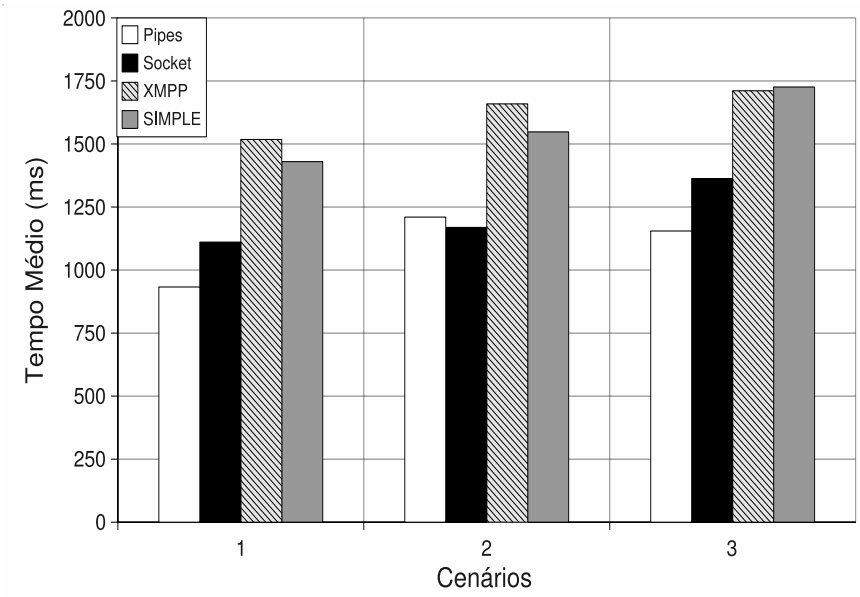

Figura 5.2: Tempo médio da etapa de registro

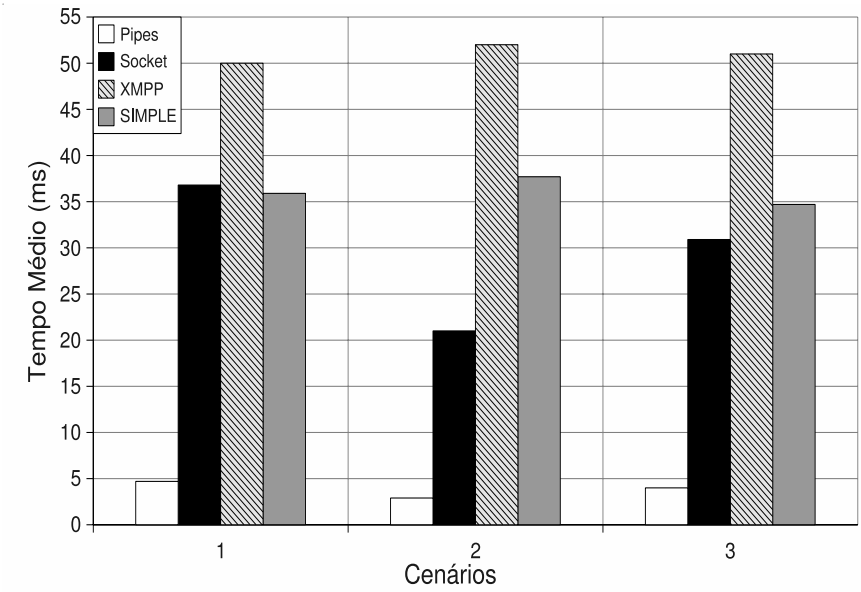

Figura 5.3: Tempo médio da etapa de notificação

A Figura 5.3 por sua vez, apresenta o tempo médio da etapa de notificação. Mais uma vez o mecanismo de comunicação JXTA-SOAP não foi apresentado já que seus tempos são os maiores entre os avaliados. Entre os apresentados na Figura 5.3, o XMPP foi o que apresentou o pior desempenho, que pode ser explicado pelo fato de que todas as suas notificações obrigatoriamente passavam pelo servidor XMPP (*i.e.*, PA), que por sua vez foi implementado em Java. Apesar deste desempenho inferior, o uso do XMPP pode ser interessante em situações em que seja necessário armazenar as informações de presença para futuras consultas. Os tempos obtidos com o serviço de presença proposto (utilizando JXTA *Pipes* ou JXTA *Socket*) mais uma vez foram os melhores encontrados, seguidos pelo SIMPLE. No entanto, os dois mecanismos de comunicação utilizados no serviço de presença proposto não possuem confirmação de suas mensagens, ao contrário do SIMPLE. Esta confirmação pode ser usada para garantir o correto envio das informações de presença aos *Watchers*.

5.5 Avaliação do Consumo de Banda

Com o objetivo de avaliar o consumo de banda, foi utilizado o cenário de testes mais simples, apresentado na Figura 5.1. Nesta figura, um PUA envia informações de presença para um *Watcher*. Essas informações são enviadas na forma de mensagens periódicas de notificação e, para fins das medições, tiveram o seu período variado de 1 até 80 milisegundos (mais precisamente, foram usados os valores 1, 2, 3, 4, 5, 10, 20, 40 e 80 milissegundos). Cada ponto apresentado nos gráficos, que corresponde a um serviço de presença e um dos intervalos entre mensagens, correspondem a média de um total de 200 medições.

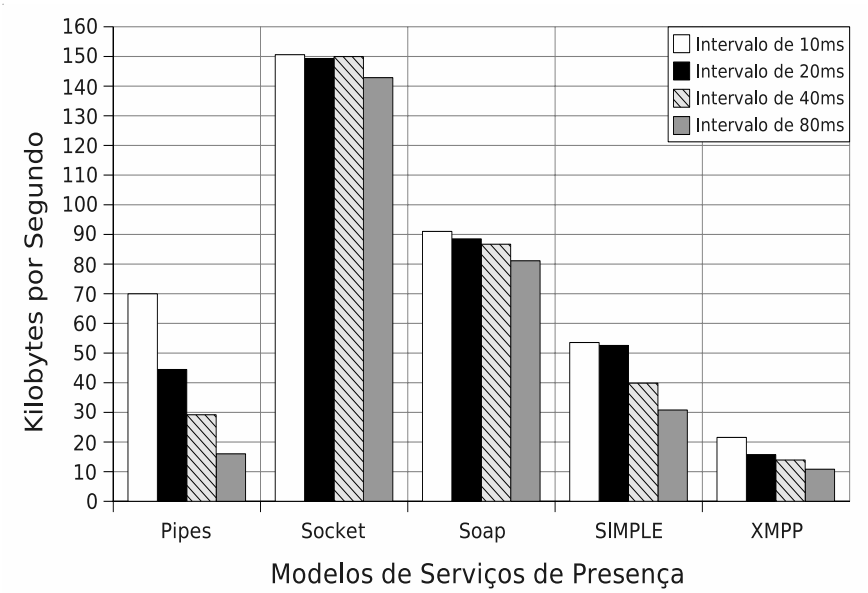

Figura 5.4: Consumo total de banda

A Figura 5.4 apresenta o total da banda consumida por cada serviço de presença enquanto que a Figura 5.5 apresenta a banda útil (*i.e.*, apenas mensagens contendo informações de presença) em relação à banda total. Nestas figuras, não foram utilizados intervalos entre notificações menores do que 10 milisegundos, uma vez que, nessas taxas, podem

ocorrer perdas de mensagens que interfeririam nas medições.

Uma primeira observação que pode ser feita com base na Figura 5.4 é que a implementação de JXTA *Sockets* é a que mais consome recursos da rede enquanto que o XMPP é o menos custoso, dentre os 5 avaliados. No entanto, avaliando os resultados apresentados na Figura 5.5, observa-se que o grande consumo do JXTA *Sockets* não é devido às notificações em si, mas sim ao intenso tráfego de mensagens de controle: provavelmente mensagens JXTA usadas para manter a comunicação JXTA *Sockets*. Conclusão semelhante pode ser tirada da observação da implementação do JXTA-SOAP utilizada no serviço de presença proposto: um alto consumo de banda da rede com uma baixa utilização.

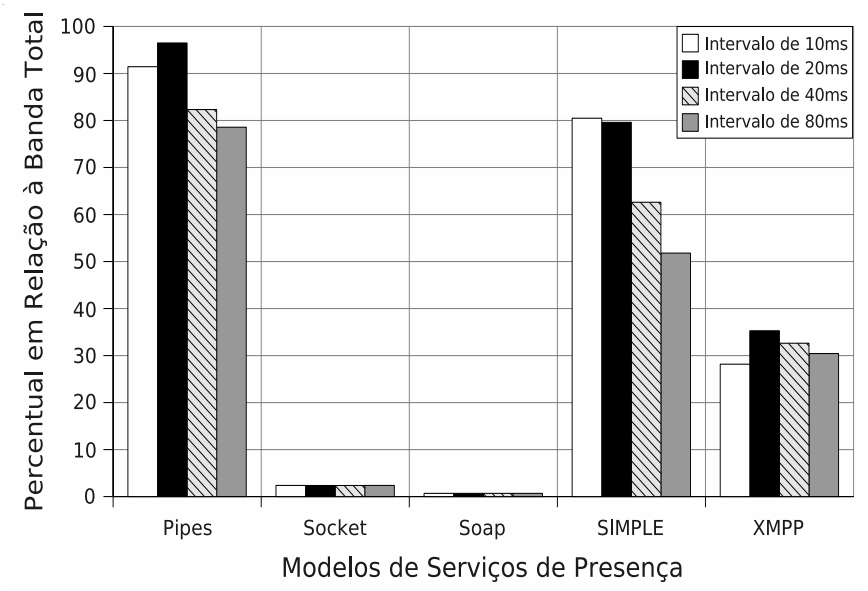

Figura 5.5: Ocupação efetiva da banda

Por outro lado, observa-se a implementação de JXTA *Pipes* para notificar é a mais efetiva, pois apresenta um baixo consumo de banda (quando comparado com os outros avaliados) aliado a um fator de utilização muito bom, chegando próximo a 95%. Esse alto fator de utilização da banda faz com que variações na taxa de envio de informações de presença sejam acompanhadas de variações mais ou menos proporcionais ao uso total da banda, conforme pode ser observado na Figura 5.4. O mesmo não se observa com o JXTA *Sockets*, ficando o consumo de banda inalterado com a variação da taxa de envio de mensagens de notificação.

5.6 Avaliação da Capacidade de Envio de Notificações

A terceira métrica avaliada representa a capacidade dos *peers* em enviar um grande número de notificações em um curto espaço de tempo, ou seja, a taxa de envio de informações de presença. Na Figura 5.6, percebe-se que existe um ponto a partir do qual o aumento na taxa de envio não se reflete em aumento na taxa de recepção: esse é o ponto de saturação do serviço. No caso das implementações XMPP e JXTA *Pipes*, que são as mais eficientes, essa saturação é atingida em torno de 250 mensagens enviadas por segundo (o que corresponde a uma periodicidade de 4 milisegundos entre envio de notificações).

Por outro lado, o SIMPLE, menos eficiente, atingiu a saturação da sua capacidade em 50 mensagens enviadas por segundo (20 milisegundos entre mensagens enviadas).

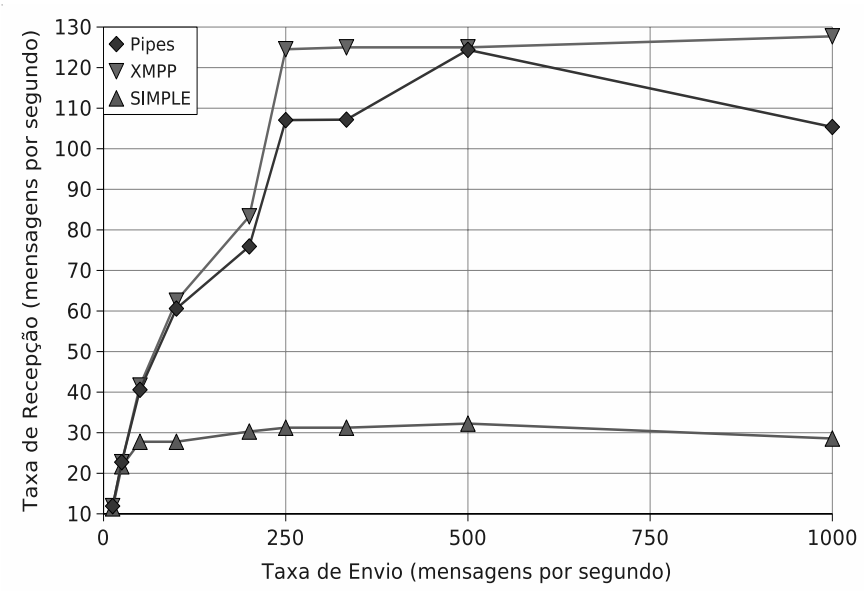

Figura 5.6: Taxa efetiva de envio (JXTA *Pipes*, XMPP e SIMPLE)

Os dois modelos que apresentaram o pior desempenho estão representados na Figura 5.7, pois seus resultados são quase inexpressivos quando comparados aos três primeiros. Percebe-se que estes modelos chegam aos seus limites quando são enviadas de 25 a 100 mensagens por segundo. Entretanto, o receptor só consegue perceber uma taxa em torno de duas e 3,8 mensagens por segundo para o JXTA-SOAP e JXTA *Sockets*, respectivamente.

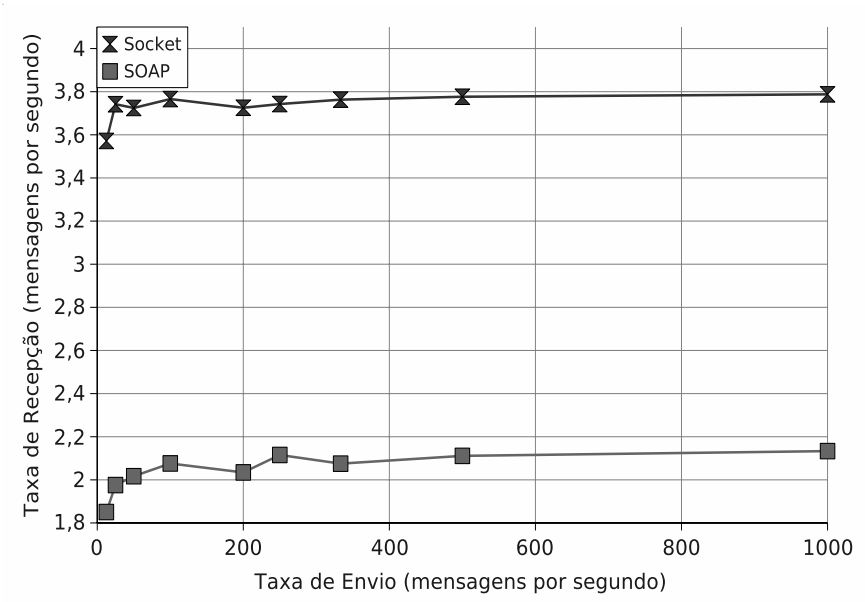

Figura 5.7: Taxa efetiva de envio (JXTA *Socket* e JXTA-SOAP)

5.7 Perdas de Mensagens

Além destes resultados, observou-se a perda de mensagens (*i.e.*, informações de presença) encaminhadas aos *Watchers*. Dos cinco métodos de entrega de notificações analisados, constatou-se perdas significativas de mensagens apenas no SIMPLE e no JXTA-SOAP. Entretanto, essas perdas ocorreram em cenários com altas taxas de envio de notificação (superior a 100 mensagens por segundo). Estes resultados podem ser observados na Tabela 5.2.

Tabela 5.2: Quantidade de Mensagens Perdidas

Modelo	Intervalo entre notificações				
	1ms	2ms	3ms	4ms	5ms
SIMPLE	14	11	3	1	0
SOAP	3	4	1	0	0

Embora nestes dois métodos de entrega de notificações tenham ocorrido perdas, ambos possuem a característica de confirmar o envio de mensagens. Este fato, pode ser utilizado para construir um mecanismo que verifique o envio correto das mensagens. Este mecanismo, ao perceber falha no envio das informações de presença, pode reenviar a notificação perdida.

6 CONCLUSÕES E TRABALHOS FUTUROS

Nesta dissertação foi proposta uma arquitetura de gerenciamento de redes, baseada em tecnologias P2P, que possibilita a integração de serviços de presença em sua estrutura para serem utilizados como mecanismo de notificação. Esta arquitetura permite que os administradores interessados no estado de determinados dispositivos ou serviços da rede, possam se registrar no sistema de gerenciamento para receber as informações de presença da melhor forma possível. Além desta arquitetura, foram revistos alguns dos principais serviços de presença existentes na literatura, dentre os quais, dois (*i.e.*, SIMPLE e XMPP) foram selecionados para serem integrados à arquitetura proposta. Por fim, foi desenvolvido um serviço de presença, também integrado à arquitetura, que utiliza exclusivamente os mecanismos de comunicação existentes no *framework* JXTA, para encaminhar as informações de presença aos interessados.

A arquitetura de gerenciamento proposta é capaz de decidir qual é o serviço de presença mais adequado para ser utilizado como mecanismo de notificação em diferentes situações. Estas situações, por sua vez, representam quando, como e sobre quais recursos (*i.e.*, dispositivos e serviços) os administradores querem ser notificados. Para escolher qual serviço de presença é o mais apropriado em cada situação foram definidos componentes, denominados *Brokers*, que utilizam regras definidas por operadores humanos (que desempenham o papel de controladores) e que informam qual é o serviço de presença que melhor atende às necessidades dos administradores interessados nas notificações. Os *Brokers* também são responsáveis por configurar e controlar os *peers* do *overlay* de gerenciamento para que possam se comunicar devidamente.

O acesso aos diferentes dispositivos e serviços que podem ser encontrados nas atuais redes de computadores não foi discutido nesta dissertação, uma vez que foge do escopo deste trabalho. Contudo, foi apresentada uma API que possibilita a criação de *gateways*, denominados monitores, que implementam a comunicação com tais recursos para obter as informações de presença. Uma vez que estes monitores sabem o estado dos recursos gerenciados, são geradas as mensagens que são enviadas aos administradores interessados utilizando o *overlay* de gerenciamento. O uso desta API também permite que o próprio protótipo controle o funcionamento individual de cada novo monitor.

Um protótipo baseado na arquitetura de gerenciamento proposta foi desenvolvido com o objetivo de validar os componentes definidos e também avaliar o desempenho dos serviços de presença selecionados. Esta avaliação, em conjunto com as características observadas na seção 2.2, possibilitou entender melhor o comportamento que tais serviços de presença apresentam quando utilizados em processos de gerenciamento de redes. Este protótipo foi incorporado no sistema de gerenciamento de redes ManP2P, adicionando funcionalidades para o envio de notificações. É importante ressaltar que os *frameworks* utilizados na implementação do SIMPLE e do XMPP, ainda não possuem suporte à men-

sagens CPIM, que é uma proposta do IETF com o objetivo de padronizar o formato das informações de presença e das mensagens instantâneas. Caso mensagens CPIM fossem suportadas, a geração das mensagens de notificação seria facilitada, uma vez que não seria necessário mapear as informações de estado para os diferentes formatos utilizados pelos serviços de presença, o que facilitaria também na integração de novos serviços no protótipo desenvolvido.

Para avaliar o desempenho do protótipo desenvolvido e, principalmente, dos serviços de presença que foram adicionados à sua estrutura, realizou-se uma série de experimentos em um cenário de testes real. Nestes experimentos, foram avaliados os seguintes critérios: tempo médio da etapa de registro e da etapa de notificação, consumo total e útil da banda, capacidade de envio de notificações e quantidade de mensagens perdidas. Os resultados obtidos indicaram que o serviço de presença proposto nesta dissertação, quando utilizado com o JXTA *Pipes*, é o que possui o melhor desempenho para enviar as informações de presença. Contudo, os demais serviços de presença ainda apresentam características que os candidatam para serem utilizados. Por exemplo, o XMPP pode ser utilizado em situações em que seja necessário armazenar o histórico das informações de presença enviadas, ou então quando for necessário consumir poucos recursos da rede (*e.g.*, banda). A utilização do SIMPLE, por exemplo, pode ser interessante em ambientes que já possuem sistemas de comunicação baseados em SIP. Apesar de a implementação JXTA-SOAP ter sido a mais custosa, ela pode ser interessante em ambientes que já utilizam Web Services para receber informações de estado, tornando desnecessário a reimplementação de tais funções. O JXTA *Sockets*, por sua vez, pode ser utilizado quando for necessário enviar além das informações de presença, dados não textuais, por exemplo, em situações de transferência de *scripts*.

Finalmente, com base nas conclusões obtidas com a comparação das características dos serviços de presença e com os resultados da avaliação de desempenho realizada, pode-se concluir que de fato, serviços de presença tradicionais podem ser utilizados como mecanismo de notificação em processos de gerenciamento de redes. Contudo, não se pode definir um serviço de presença definitivo, uma vez que as soluções avaliadas possuem características próprias que as candidatam para serem utilizadas em diferentes situações. Neste cenário, uma solução que possua as principais funcionalidades dos atuais serviços de presença e que seja projetada utilizando tecnologias P2P, apresenta-se como a mais indicada para ser utilizada nas modernas redes de computadores.

Como trabalhos futuros pretende-se integrar novos serviços de presença (e.g., *Wireless Village* e APEX) à estrutura da arquitetura proposta. Com estes novos serviços de presença, outros experimentos deverão ser realizados, envolvendo novos cenários e um maior número de *peers*. Também pretende-se aprimorar o serviço de presença proposto, desenvolvendo novas funcionalidades, tais como a possibilidade de se armazenar as informações de presença para futuras consultas, assim como no XMPP, para que os administradores possam receber todas as alterações que determinado recurso apresentou durante o tempo em que não estiveram conectados no sistema. Ainda que a implementação efetuada nesta dissertação seja apenas um protótipo, pode-se afirmar que a mesma está operacional e pode ser adotada prontamente para notificar os administradores sobre o estado de dispositivos e serviços utilizando o ManP2P.

REFERÊNCIAS

AMORETTI, M.; ZANICHELLI, F.; CONTE, G. SP2A: a service-oriented framework for p2p-based grids. In: MGC '05: PROCEEDINGS OF THE 3RD INTERNATIONAL WORKSHOP ON MIDDLEWARE FOR GRID COMPUTING, 2005, New York, NY, USA. **Anais...** ACM Press, 2005. p.1–6.

ANDROUTSELLIS-THEOTOKIS, S.; SPINELLIS, D. A Survey of Peer-to-Peer Content Distribution Technologies. **ACM Comput. Surv.**, New York, NY, USA, v.36, n.4, p.335–371, 2004.

B. CAMPBELL, E.; ROSENBERG, J.; SCHULZRINNE, H.; HUITEMA, C.; GURLE, D. **Session Initiation Protocol (SIP) Extension for Instant Messaging**. 2002. n.3428.

CARZANIGA, A.; ROSENBLUM, D. S.; WOLF, A. L. Design and Evaluation of a Wide-Area Event Notification Service. **ACM Transactions on Computer Systems (TOCS)**, New York, NY, USA, v.19, n.3, p.332–383, 2001.

CASE, J.; FEDOR, M.; SCHOFFSTALL, M.; DAVIN, J. **RFC 1157 - Simple Network Management Protocol (SNMP)**. Disponível em <ftp://ftp.rfc-editor.org/in-notes/rfc1157.txt>. Acesso em maio de 2006.

Cisco. **Cisco IOS NetFlow**. Disponível em <http://www.cisco.com/warp/public/732/Tech/netflow>. Acesso em mai. de 2007.

CURBERA, F.; DUFTLER, M.; KHALAF, R.; NAGY, W.; MUKHI, N.; WEERAWARANA, S. Unraveling the Web Services Web - An Introdution to SOAP, WSDL,and UDDI. **IEEE Internet Computing**, [S.l.], v.6, n.2, p.86–93, 2002.

DAY, M.; AGGARWAL, S.; MOHR, G.; VINCENT, J. **Instant Messaging / Presence Protocol Requirements**. 2000. n.2779.

DAY, M.; ROSENBERG, J.; SUGANO, H. **A Model for Presence and Instant Messaging**. 2000. n.2778.

ERICSSON; MOTOROLA; NOKIA. **Wireless Village The Mobile IMPS Initiative, System Architecuture Model**. Disponível em: <http://www.openmobilealliance.org/tech/affiliates/LicenseAgreement.asp?DocName=/wv/wv_architecture_v1.0.pdf>. Acesso em: nov. 2007.

FELBER, P. A.; GUERRAOUI, R.; KERMARREC, A.-M. The Many Faces of Publish/-Subscribe. **ACM Computing Surveys (CSUR)**, New York, NY, USA, v.35, n.2, p.114–131, June 2003.

FOSTER, I.; KESSELMAN, C.; TUECKE, S. The Anatomy of the Grid: enabling scalable virtual organizations. **International Journal of High Performance Computing Applications**, Thousand Oaks, CA, USA, v.15, n.3, p.200–222, 2001.

GAMMA, E.; HELM, R.; JOHNSON, R.; VLISSIDES, J. **Design Patterns**. [S.l.]: Addison-Wesley Professional, 1995.

GOLDSZMIDT, G.; YEMINI, Y. Distributed management by delegation. In: INTERNATIONAL CONFERENCE ON DISTRIBUTED COMPUTING SYSTEMS, 1995, 15., 1995, Vancouver, BC, Canada. **Proceedings...** [S.l.: s.n.], 1995. p.333–340.

GONG, L. JXTA: a network programming environment. **IEEE Communications Magazine**, [S.l.], v.5, n.3, p.88–95, May 2005.

GRANVILLE, L. Z.; ROSA, D. M. da; PANISSON, A.; MELCHIORS, C.; ALMEIDA, M. J. B.; TAROUCO, L. M. R. Managing Computer Networks Using Peer-to-Peer Technologies. **IEEE Communications Magazine**, [S.l.], v.43, n.10, p.62–68, 2005.

GROUP, P. **PAM Working group Progress**. Disponível em: <http://www.parlay.org/en/docs/may2001mm/11_pamstatus.ppt>. Acesso em: nov. 2007.

HARRINGTON, D.; PRESUHN, R.; WIJNEN, B. **RFC 3411 - An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks**. Disponível em: <http://www.ietf.org/rfc/rfc3411.txt?number=3411>. Acesso em: nov. 2007.

IPTEL.ORG. **Sip Express Router - SER**. Disponível em: <http://www.iptel.org/ser>. Acesso em: nov. 2007.

ISOMAKI, M.; LEPPANEN, E.; NOKIA. **RFC 4827 - An Extensible Markup Language (XML) Configuration Access Protocol (XCAP) Usage for Manipulating Presence Document Contents**. Disponível em: <http://www.ietf.org/rfc/rfc4827.txt>. Acesso em: set. 2007.

ITU-T Rec. X.700. **Management Framework for Open Systems Interconnection (OSI)**. Disponível em <ftp://ftp.rfc-editor.org/in-notes/rfc3198.txt>. Acesso em jun. de 2007.

KLYNE, G.; CORPORATION, C.; ROSE, M.; DOVER BEACH CONSULTING, I.; SCHWARTZ, M.; LLC, C. O. T. R.; DIXON, E.; FRANKLIN, H.; KINT, J.; NEW, D.; PEAD, S. **RFC 3342 - The Application Exchange (APEX) Option Party Pack, Part Deux!** Disponível em: <http://www.ietf.org/rfc/rfc3342.txt>. Acesso em: set. 2007.

KLYNE, G.; NINE, N. by; ATKINS, D.; CONSULTING, I. **RFC 3862 - Common Presence and Instant Messaging (CPIM): message format**. Disponível em: <http://www.ietf.org/rfc/rfc3862.txt>. Acesso em: set. 2007.

KLYNE, G.; NINE, N. by; NOTTINGHAM, M.; BEA; MOGUL, J.; LABS, H. **RFC 3864 - Presence Information Data Format (PIDF)**. Disponível em: <http://www.ietf.org/rfc/rfc3864.txt>. Acesso em: set. 2007.

LAKSHMI, R. OSI systems and network management. **IEEE Communications Magazine**, [S.l.], v.36, n.3, p.46–53, 1998.

LEINWAND, A.; CONDROY, K. F. **Network Management**: a practical perspective. 2.ed. [S.l.]: Menlo Park, 1996.

LOBO, J. **JSR 123: jain presence and availability management (pam) api**. [S.l.: s.n.], 2003. JSR. (123).

MARQUEZAN, C. C.; GRANVILLE, L. Z.; VIANNA, R. L.; ALVES, R. S. QAME Support for Policy-Based Management of Country-wide Networks. In: IFIP/IEEE INTERNATIONAL SYMPOSIUM ON INTEGRATED NETWORK MANAGEMENT, 2005. IM 2005., 2005., 2005, Nice, France. **Anais...** [S.l.: s.n.], 2005. Aceito no Application Section do IM 2005.

MARQUEZAN, C. C.; SANTOS, C. R. P. dos; NOBRE, J. C.; ALMEIDA, M. J. B.; TAROUCO, L. M. R.; GRANVILLE, L. Z. Self-managed services over a P2P-based Network Management Overlay. In: LATIN AMERICAN AUTONOMIC COMPUTING SYMPOSIUM (LAACS 2007), 2., 2007, Petrópolis, Brazil. **Proceedings...** [S.l.: s.n.], 2007. p.–. To appear.

MARQUEZAN, C. C.; SANTOS, C. R. P. dos; SALVADOR, E. M.; ALMEIDA, M. J. B.; CECHIN, S. L.; GRANVILLE, L. Z. Performance Evaluation of Notifications in a Web Services and P2P-Based Network Management Overlay. In: INTERNATIONAL COMPUTER SOFTWARE AND APPLICATIONS CONFERENCE (COMPSAC'07), 2007. **Proceedings...** [S.l.: s.n.], 2007.

MARTIN-FLATIN, J.-P.; ZNATY, S.; HABAUX, J.-P. A Survey of Distributed Entreprise Network and Systems Management Paradigms. **Journal of Network and Systems Management**, [S.l.], v.7, n.1, p.9–26, 1999.

MICROSYSTEMS, S. **JXTA v2.3.x: java programmer's guide**. Disponível em: <http://www.jxta.org/docs/JxtaProgGuide_v2.3.pdf>. Acesso em: nov. 2007.

MITRA, N. **SOAP Version 1.2 Part 0: primer**. Disponível em: <http://www.w3.org/TR/2003/REC-soap12-part0-20030624/>. Acesso em: nov. 2007.

MOURA, G. C. M.; SILVESTRIN, G.; SANCHEZ, R. N.; GASPARY, L. P.; GRANVILLE, L. Z. On the Performance of Web Services Management Standards - An Evaluation of MUWS and WS-Management for Network Management. In: INTEGRATED NETWORK MANAGEMENT, 2007. **Anais...** IEEE, 2007. p.459–468.

PANISSON, A. **Aplicação de Técnicas de Distribuição de Carga em Sistemas de Gerenciamento de Redes Baseados em P2P**. 2007. Dissertação (Mestrado em Ciência da Computação) — Programa de Pós Graduação em Ciências da Computação - UFRGS.

PANISSON, A.; MELCHIORS, C.; GRANVILLE, L. Z.; ALMEIDA, M. J. B.; TAROUCO, L. M. R. Designing the Architecture of P2P-Based network Management Systems. In: IEEE SYMPOSIUM ON COMPUTERS AND COMMUNICATIONS (ISCC'06), 2006, Los Alamitos, CA, USA. **Proceedings...** IEEE Computer Society, 2006. p.69–75.

PELTZ, C. Web Services Orchestration and Choreography. **Computer**, Los Alamitos, CA, USA, v.36, n.10, p.46–52, 2003.

ROSE, M. **The Blocks Extensible Exchange Protocol Core**. 2001. n.3080.

ROSE, M.; CONSULTING, D. B.; KLYNE, G.; CORPORATION, C.; CROCKER, D.; INTERNETWORKING, B. **RFC 3340 - The Application Exchange Core**. Disponível em: <http://www.ietf.org/rfc/rfc3340.txt>. Acesso em: set. 2007.

ROSE, M.; CONSULTING, D. B.; KLYNE, G.; CORPORATION, C.; CROCKER, D.; INTERNETWORKING, B. **RFC 3341 - The Application Exchange (APEX) Access Service**. Disponível em: <http://www.ietf.org/rfc/rfc3341.txt>. Acesso em: set. 2007.

ROSE, M.; KLYNE, G.; CROCKER, D. **The Application Exchange (APEX) Presence Service**. 2003. n.3343.

ROSENBERG, J.; SCHULZRINNE, H.; CAMARILLO, G.; JOHNSTON, A.; PETERSON, J.; SPARKS, R.; HANDLEY, M.; SCHOOLER, E. **SIP: session initiation protocol**. 2002. n.3261.

SAINT-ANDRE, E. P.; FOUNDATION, J. S. **RFC 3920 - Extensible Messaging and Presence Protocol (XMPP): core**. Disponível em: <http://www.ietf.org/rfc/rfc3920.txt>. Acesso em: set. 2007.

SAINT-ANDRE, E. P.; FOUNDATION, J. S. **RFC 3921 - Extensible Messaging and Presence Protocol (XMPP): instant messaging and presence**. Disponível em: <http://www.ietf.org/rfc/rfc3921.txt>. Acesso em: set. 2007.

SAINT-ANDRE, E. P.; FOUNDATION, J. S. **RFC 3922 - Mapping the Extensible Messaging and Presence Protocol (XMPP) to Common Presence and Instant Messaging (CPIM)**. Disponível em: <http://www.ietf.org/rfc/rfc3922.txt>. Acesso em: set. 2007.

SAINT-ANDRE, E. P.; FOUNDATION, J. S. **RFC 3923 - End-to-End Signing and Object Encryption for the Extensible Messaging and Presence Protocol (XMPP)**. Disponível em: <http://www.ietf.org/rfc/rfc3923.txt>. Acesso em: set. 2007.

SANTOS, C. R. P. dos; CECHIN, S. L.; ALMEIDA, M. J. B.; CECHIN, S. L.; GRANVILLE, L. Z.; TAROUCO, L. M. R. On the Performance of Employing Presence Services in P2P-Based Network Management Systems. In: ACM SYMPOSIUM ON APPLIED COMPUTING (SAC 2008), 23., 2008, Fortaleza, Brazil. **Anais...** [S.l.: s.n.], 2008. p.–. To appear.

SANTOS, C. R. P. dos; MARQUEZAN, C. C.; SALVADOR, E. M.; SANTA, L. F. D.; CECHIN, S. L.; ALMEIDA, M. J. B.; CECHIN, S. L.; GRANVILLE, L. Z.; TAROUCO, L. M. R. On the Design and Performance Evaluation of Notification Support for P2P-Based Network Management. In: ACM SYMPOSIUM ON APPLIED COMPUTING (SAC 2008), 23., 2008, Fortaleza, Brazil. **Anais...** [S.l.: s.n.], 2008. p.–. To appear.

SCHULZRINNE, H.; U., C.; GURBANI, V.; LUCENT; KYZIVAT, P.; ROSENBERG, J.; CISCO. **RFC 4480 - RPID: rich presence extensions to the presence information data format (pidf)**. Disponível em: <http://www.ietf.org/rfc/rfc4480.txt>. Acesso em: set. 2007.

SCHöNWALDER, J.; QUITTEK, J.; KAPPLER, C. Building Distributed Management Applications with the IETF ScriptMIB. **IEEE Journal on Selected Areas in Communications**, [S.l.], v.18, n.5, p.702–714, 2000.

SOFTWARE, J. **Smack API v3.0.4**. Disponível em: <http://www.igniterealtime.org/move.jsp>. Acesso em: nov. 2007.

SOLDATOS, J.; ALEXOPOULOS, D. Web services-based network management: approaches and the wsnet system. **Int. J. Netw. Manag.**, New York, NY, USA, v.17, n.1, p.33–50, 2007.

STALLINGS, W. **SNMP, SNMPv2, and CMIP**: The Practical Guide to Network-Management Standards. [S.l.]: Addison-Wesley Professional, 1993.

STALLINGS, W. **SNMP, SNMPv2, SNMPv3, and RMON 1 and 2**. [S.l.]: Addison-Wesley Professional, 1999.

STANDARDS, N. I. of; NIST, T. **Jain-Sip**. Disponível em: <https://jain-sip.dev.java.net/>. Acesso em: nov. 2007.

SUGANO, H.; MAZZOLDI, F.; DIACAKIS, A.; FUJIMOTO, S.; HUDSON, G.; RAMSDELL, J. D. Presence and Instant Messaging Protocol (PRIM). **Internet Draft**, [S.l.], 2001.

TIGASE.ORG. **Tigase**. Disponível em: <http://www.tigase.org/>. Acesso em: nov. 2007.

WESTERINEN, A.; SCHNIZLEIN, J.; STRASSNER, J.; SCHERLING, M.; QUINN, B.; HERZOG, S.; HUYNH, A.; CARLSON, M.; PERRY, J.; WALDBUSSER, S. **RFC 3198 - Terminology for Policy-Based Management**. Disponível em <ftp://ftp.rfc-editor.org/in-notes/rfc3198.txt>. Acesso em jun. de 2007.

APÊNDICE A ARTIGO PUBLICADO

Neste anexo é apresentado o artigo "On the Performance of Employing Presence Services in P2P-Based Network Management Systems"desenvolvido durante o mestrado. O artigo é resultado do segundo ano do mestrado e apresenta a utilização de serviços de presença tradicionais como mecanismo de notificação em sistemas P2P de gerenciamento de redes. O artigo apresenta também uma arquitetura para o tema, mostrada nesta dissertação, que possibilita a integração de tais serviços de presença em sua estrutura. Por fim, são apresentados os resultados da avaliação de desempenho realizadas entre os serviços de presença SIMPLE, XMPP e o serviço de presença desenvolvido que utiliza exclusivamente os meios de comunicação disponíveis no JTXA para enviar as notificações de estado aos interessados.

- Título: On the Performance of Employing Presence Services in P2P-Based Network Management Systems
- Nome: ACM Symposium on Applied Computing (SAC 2008)
- URL: http://www.acm.org/conferences/sac/
- Data: De 16 a 20 de março de 2008
- Local: Hotel Vila Galé, Fortaleza, Brazil

On the Performance of Employing Presence Services in P2P-Based Network Management Systems

Carlos R. P. dos Santos
crpsantos@inf.ufrgs.br

Sérgio Luis Cechin
cechin@inf.ufrgs.br

Lisandro Z. Granville
granville@inf.ufrgs.br

Maria J. B. Almeida
janilce@inf.ufrgs.br

Liane M. R. Tarouco
liane@penta.ufrgs.br

Federal University of Rio Grande do Sul - Institute of Informatics
Av. Bento Goncalves, 9500
Porto Alegre, RS - Brazil

ABSTRACT

The use of notifications to report the status of underlying communication networks has a crucial impact on the performance of the managed network itself. Presence services, which are implemented using notification messages, have been designed with the objective to provide ways of delivering accurate presence information to interested parties. However, up to now, the use of presence services in the network management discipline has not been properly addressed. Is this paper, we propose an architecture that introduces presence services into traditional network management processes. We evaluated, through a system prototype implementation, the feasibility of using diverse presence services solutions as management tools and compare their performance against to each other. We present the results of a set of experiments regarding the propagation delay of the two main steps of a presence service, i.e., registration and notification. Those results allows a network administrator to choose which presence service solution is more adequate for the administrator's necessities, thus helping to improve the notification process.

Categories and Subject Descriptors

C.2.3 [**Network Operations**]: Network Management

General Terms

Network Management

Keywords

Peer-to-Peer, Notifications, Presence Services

SAC'08 March 16-20, 2008, Fortaleza, Ceará, Brazil

1. INTRODUCTION

The increasing complexity of computer networks has always emphasized the importance of network management as a tool to reduce communication costs. Usually, the more complex the managed network, the more sophisticated is its management system. However, the evolution of management solutions has been less effective than the introduction of new networking technologies. As a consequence, traditional management is unable to properly address the necessities of modern networks. For example, the actual usage of the Simple Network Management Protocol (SNMP) [8] - defined in the 1980s and reviewed in the 1990s - does not couple with the distributed management approach that avoids the centralized approach's problems (e.g., a single station controlling an increasing number of devices does not scale in terms of processing power and bandwidth consumption). Since firewalls and NAT confine SNMP traffic to a domain's boundaries, it cannot be used in multi-domain environments (e.g., virtual organizations) where autonomous human administrators need to cooperate to accomplish a common task. In this context, new management solutions become clearly required.

Recently, the network management community has initiated the investigation of two new technologies defined originally in other areas: Web services and peer-to-peer (P2P). The main motivation for this researches is the fact that features of these new technologies could improve the management processes. For example, Web services traffic can cross administrative boundaries easier than SNMP because it uses XML documents transported on top of Web protocols, such as HTTP, SMTP, and FTP. Similarly, P2P technologies present advantages too; for example, they enable distributed and cooperative management through different administrative domains. However, despite to present interesting advantages, both Web service and P2P tend to require more resources (e.g., bandwidth, processing power) to operate when compared to traditional management solutions.

In an effort to prevent the extra consumption of the network resources, mechanisms used in the network management activity can be employed together with these two technologies in order to not damage the user data traffic as a consequence to the then intense management traffic. Notifications are one of these mechanisms, where traditional band-

width consuming polling processes are replaced by asynchronous communications triggered only when strictly necessary. Support for notifications can be built following the Publish/Subscribe model [6] in order to report to interested manager the status of the underlaying network and its devices and services. The use of notifications help to decrease the bandwidth consumption because just one single request is made by the administrator (i.e., a subscription message informing that a particular manager is interested in receiving future notifications), and the notification messages are sent only when they are really necessary. Some researchers has been using notifications in Web services and P2P-based network management systems [9], and the general current conclusion is that they are a feasible solution to maintain, at the management stations, a consistent picture of the managed network resources.

Besides the choice of a feasible mechanism to report the network status, management systems should support features to guarantee that the correct information are sent to interested managers. Presence Services is a mechanism that can be used for this purpose. Presence services have been initially conceived in the context of the Instant Messaging (IM) applications that became quite popular among Internet users. Presence services are used to notify the user status (e.g., online, offline, talking at the phone, etc.) to a group of interested remote persons, usually referenced as the user's buddy list. In addition to the IM applications context, the concept of notification can be founded in other contexts, like in general P2P systems. In this case, these services are used to report, for example, information regarding the underlaying operation of a P2P overlay (e.g., peers connectivity) to the peers that compose the overlay. Up to now, the use of P2P-based notifications and, more specifically, presence services has not been investigated as a network management tool in a systematic way. We believe that these services can be used in network management because they provide ways to deliver correct presence information to interested parties. For example, they can be used to report a simple on/off status of network entities (e.g., devices, softwares, people) or more specific information, such as when and how network entities can be accessed. In face of these facts, the main objective of this work is to show how these services can be used to improve the efficiency of network management solutions. To accomplish this objective, we implemented a system prototype and executed a set of measurements that will be discussed in further sections of this document.

The remainder of this paper is structured as follows. Section 2 presents current P2P-based network management work and depicts the concept of presence services. In section 3 we propose a management architecture based on P2P notification services. The evaluation results of a set of measurements carried out using our prototype system is presented in Section 4. Finally, in Section 5, conclusions and the future work close this paper.

2. BACKGROUND

Presence services at one side, and network management based on P2P technologies at another side are independent research topics being investigated by some research projects and groups. However, up to now, there are no reports about their use in an integrated fashion. In this section we review the main related work that cover these still disconnected areas.

2.1 Network Management Based on P2P Technologies

One of the first work related to network management based on P2P technologies was proposed by State et al. [12]. They worked with P2P integrated to traditional management technologies such as SNMP. Their solution uses Java Management Extensions (JMX) for the management of agents created with this technology and the JXTA [7] framework to development of P2P overlay. In such work, the authors present a new management mechanism in which the entities to be managed publish their management interfaces to remote peers. The architecture proposed is not completely integrated with the native management agents located inside the current network devices (e.g., routers, etc.) since it is currently composed only by JMX agents.

The work of Binzenhofer et al. [2] suggests the usage of a P2P overlay for fault and performance management tasks. The architecture proposal is based on agents, called of *Distributed Network Agents* (DNAs), that compose the management overlay and can execute connectivity and QoS monitoring tests in a distributed fashion. However, the proposed architecture allows only fault and performance management. It does not treat other critical functions such as configuration management or notification support.

Kamienski et al. propose the framework P4MI (P2P Policy Management Infrastructure) that employs policies and P2P technologies for network management purposes. The authors present the PBMAN, a P4MI application for the management of Ambient Networks (AN). It provides a scalable mechanism to compose networks inside an AN and provides distribution and recovery policies. The P4MI is composed of Policy Decision Networks (PDNs), Policy Enforcement Points (PEPs), and user agents. Throughout this approach one can establishes policies to manage devices and services while keeping the compatibility with the SNMP.

2.2 Presence Services

The concept of presence services was used initially in communication systems that are normally related to Instant Messaging applications, originating the acronym IM&P (Instant Messaging and Presence). The first effort to produce standards for Presence Services has been made by the Internet Engineering Task Force (IETF); they produced RFC 2778 [4] and RFC 2779 [3] that present an abstract model for Presence and Instant Messaging systems, the involved entities, the associated terminology, the operational requirements, and some minimally required functionalities. RFC-2778 defines the function of a presence service to allow an user to be notified about the status change in another network element (e.g., person, device, application). Presence information can include the status and/or availability of the notifying elements. The status information reports whether the element is active or not, whereas the availability information reports whether the element can be accessed. Other data can also be included in a presence information, for example, the location and the activity of an user.

The standard defines two main elements: *Presentities* and *Watchers*. Presentities are responsible for storing and distributing presence information. Watchers are interested in receiving such information and can be divided in two groups: *Fetchers*, which search for the presence information, and *Subscribers*, that are registered to receive the notifications from the presence service. Additionally, there is a special

type of Fetcher, called *Poller*, which searches regularly for presence information. The standard also defines *Principal* entities (e.g., people, systems, devices): which are entities that do not belong to the system but holds presence information. Finally, there is Presence User Agents (PUAs) and Presence Agents (PAs). PUAs are responsible for collecting the presence information from Principals, while PAs are responsible for retrieving information from several PUAs and generate events in the Presentities. Figure 1 presents these elements and their relationship.

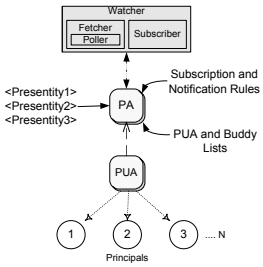

Figure 1: Elements defined in RFC 2778

Several specifications for presence service systems are under development by different working groups, based on the previous IETF standards. The two most sophisticated and important specifications are SIMPLE and XMPP, and due to that we have chosen them to be used in our solution. SIP for Instant Messaging and Presence Leveraging Extensions (SIMPLE) [1] was initially developed as an extension of the Session Initiation Protocol (SIP) to allow the exchange of messages and presence information. Its messages and presence information are sent using the SIP infrastructure for registration and notification tasks. The communication between two participants is carried out directly using the P2P approach. The Extensive Message and Presence Protocol (XMPP) [11] is an open protocol based on XML messages. It was developed initially by the Jabber community and, it was accepted as an IETF standard in the RFC 3920. Its messages are called XML Stanzas and all communications are carried out by intermediary servers, an approach that can generate problems of scalability if the employed servers are unable to handle a great number of communications.

3. INTEGRATING PRESENCE SERVICES IN A P2P-BASED MANAGEMENT SYSTEMS

We propose a management architecture that integrates presence services and P2P-based management using the ManP2P management overlay [10]. In this section we will present the technologies used to build the proposed architecture, their elements and the relashionship between them.

3.1 Underlying Technologies

ManP2P is a P2P-based management system whose objective is to ease the development of management services that will be available to remote peers. We selected ManP2P as the basis system of our investigation because it offers important network management features, such as allowing load balancing of network management tasks using peer groups, support for inter-domain management, and the possibility of integration with traditional management standards like SNMP. In ManP2P, the administrator interacts with the system through a peer called TLM (Top Level Management), which is mainly characterized by exposing a Graphical User Interface (GUI) to administrators.

In our architecture, we integrate SIMPLE and XMPP in the whole management process. Besides SIMPLE and XMPP, an additional basic presence service has been developed. This additional service uses three available JXTA communication mechanisms: JXTA-SOAP, which is an implementation of Simple Object Access Protocol (SOAP) on top of the JXTA; JXTA pipes, which are asynchronous communication channels with well defined end-points; and JXTA sockets, which are implementations of traditional sockets over of the JXTA infrastructure. It will allow us to compare the two presence services with the additional presence model built over a P2P infrastructure (i.e., over the ManP2P overlay) and foresee the behavior of traditional models working on top of a P2P infrastructure.

3.2 Architecture

The Figure 2 presents the architecture proposal and how its elements are related to each other. In the architecture, the administrator represents the human interested in receiving notifications from the management system. Since the (human) receiver uses a TLM peer, the proposal structure allows the notification receiver to be either a machine or another system.

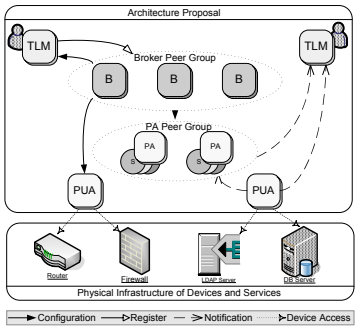

Figure 2: Architecture Proposal

Another element of the architecture is the *monitors*. They are peers that access devices or services (Principals) and propagate information about their status to other peers of the overlay. RFC 2778 defines PUAs as having almost the same function as the monitors. Thus, from now on, we will use the term PUA when referring to these monitors.

The main element considered in the architecture is the *broker*, responsible for controlling and configuring the whole system. One broker function is to hide from the TLMs the details of the overlay infrastructure (e.g., number and location of remote peers), in order to prevent the TLMs of being overloaded with unnecessary information. Another function of this broker is to control which peers are connected to the system, to guarantee that messages are not delivered to administrators that have left the overlay. In addition to

these basic functions, the main broker's responsibility is to choose what is the best way to notify the administrators, in accordance with some administrator's requirements specified in "RegisterDescriptor.xml" files. In a initial stage, this file is provided by the administrator and contain the configurations defined by a human system controller of the system. After receiving these parameters, the broker will decide which model is the best and will deliver the configurations to the corresponding elements.

In fact, brokers can be placed in more than one peer, creating a group of peer with brokerage duties. That is specially useful to guarantee the availability of the broker services to decrease the load processing in each peer acting as a broker. This is also useful when management traffic needs to cross the network boundaries, allowing the administrator to have access to information of elements which are located in other administrative domains.

The last element in our architecture is PA, that is responsible for collecting information about one or more PUAs and, based on the collected information, generating notifications that will be delivered to the Watchers (typically TLMs) that are interested in it. It is important to stand out that in the basic built presence service, PAs are not isolated because they are located in the same node that PUAs are located.

3.3 Notification Process

The whole notification process is in fact divided in two main steps: the registration step, where interested TLMs express their wish to receive notification messages, and the notification itself, where the P2P overlay manage to delivery the notification messages issued at one point of the network to the interested TLMs. The sequence steps of the register and notification acts are:

1. The network administrator defines configuration parameters to be stored on the brokers;

2. The administrator sends a "RegisterDescriptor.xml" file to the brokers peergroup;

3. All brokers receive this "RegisterDescriptor.xml" and one of them, chosen by load balancing, sends configuration parameters for the related peers;

4. PUA sense the status change of its associated elements (e.g., device or software), and converts this information for the data format of each model;

5. PUA sends notifications to the interested parties, directly or not, depending of the chosen model.

The parameters contained in "RegisterDescriptor.xml" file can represent, for example, what and when notifications should be sent. The set of informations that can be informed by administrators to brokers are: registration life time, polling time, necessity for storing notification messages in logs, necessity for message delivery confirmation, bandwidth consumption, and notification delivery time.

In the next section we will provide an evaluation of our proposal considering the architecture presented here. As mentioned before, the main objective is to observe which models are more appropriate for each situation.

4. EVALUATION

In this section we will present the scenarios used to evaluate the architecture proposed and the configuration used for this purpose. Still in this section, we analize the results obtained from these scenarios. The metrics used to evaluate the scenarios are the delay time to register and to send a notification message. These results will characterize the behavior of each presence service, and the administrator will be able to choose the best configuration to satisfy the requirements of the system.

4.1 Evaluation Scenarios

Our experiments have been carried out through 18 homogeneous machines where the peers have been deployed. Each machine has a 2.8GHz Pentium IV with 512Mb of RAM. In order to evaluate the register delay time and notification delay time three different scenarios have been used. In the simplest one – we numbered this scenario as 1 – 3, only one PUA send presence information of a device for just one Watcher. During the register step, only one Broker and one PA (when needed) was used.

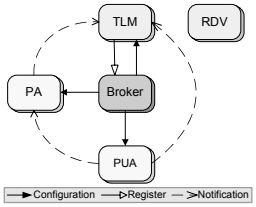

Figure 3: Scenario 1

In scenarios 2 (a medium load scenario) and 3 (a big scenario), the number of peers are greater. In scenario 2 there are 4 Watchers, one in a different peer. These Watchers request presence information to two PUAs and all messages are processed by just one Broker. In the biggest scenario the number os peers are doubly: there are 2 Brokers, 2 PAs, 4 PUAs and 8 Watchers. In all scenarios one Rendezvous was used to keep the JXTA infrastructure and to allow that all peers have an initial meeting point. It is necessary because all peers have to obtain information about all other peers and services of the overlay.

4.2 Analysis Results

We made 58 runs for each model and scenario to reach a confidence interval of 90%. Figure 4 presents the delay times for the register step. The registration and notification delay of the JXTA-SOAP communication is sensible greater than the others. Thus, it was not plotted on the graphics as it is not comparable to other models measurements.

It is possible to observe that the two most efficient models are JXTA Pipes and JXTA Sockets. It happens because this models do not need a registration in a server. This characteristic will be used by the administrators when they need a very fast register.

Figure 5 shows the delay mean times of the notification step. There, once more the JXTA-SOAP mechanism have a bigger delay time, approximately 5 times than the XMPP. XMPP has the biggest delay time among the 4 showed in the figure. The reason for this low performance is the need

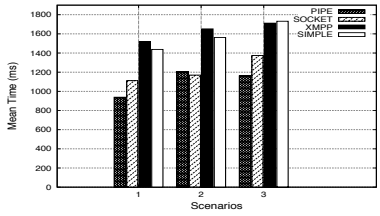

Figure 4: Register Mean Time

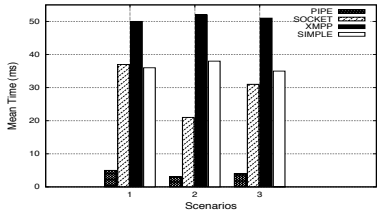

Figure 5: Notification Delay

for notification messages pass through XMPP server (PA). Despite that, its usage can be interesting in cases where logs are necessary. However, this server is a single point of failure and safer architecture would need duplicate this peer.

JXTA Pipe is the most efficient and is followed by SIMPLE model. Nevertheless, JXTA-pipes does not issue confirmation messages while SIMPLE do. This confirmation message can be used to assure that the notification was delivered. This mechanism can be used to replicate a lost notification message.

5. CONCLUSIONS AND FUTURE WORK

In this paper we evaluated two presence services specifications with three communication mechanisms available in the JXTA P2P overlay, used in the ManP2P management system. We carried out a set of experiments evaluating the registration and notification delay times to observe how each mechanism behaves and, with this, have the opportunity to define which are the best choices based on the administrators' requirements.

Given the architecture implementation and the associated evaluation, we can first observe that the communication mechanisms provided by JXTA, with exception of the JXTA-SOAP, present an interesting performance for the notification task. That helps to conclude that such mechanisms can be used when administrators required prompt notifications about events on the managed network. Despite the fact that SIMPLE and XMPP have a greater delay time, they have features that candidate them to be used in specific cases, for example, when administrators have the necessity of storing logs of notifications for future retrieval, or when the messages have to be cyphered for security reasons. The results showed in Section 4 suggests that regardless the increasing number of peers in the management overlay, the notification delay tends to remain approximately constant.

For future work we plan to execute more evaluations in other scenarios with a larger number of peers. We also intend to carry out other tests with other presence services specifications, for example, Wireless Village [5]. Finally, based on the results of this work, we plan to build a complete P2P based presence service to explore complementary aspects not originally covered in the current research.

6. REFERENCES

[1] E. B. Campbell, J. Rosenberg, H. Schulzrinne, C. Huitema, and D. Gurle. Session Initiation Protocol (SIP) Extension for Instant Messaging. RFC 3428, December 2002.

[2] A. Binzenhofer, K. Tutschku, B. auf dem Grabem, M. Fiedler, and P. Carlsson. A p2p-based framework for distributed network management. In *Proceedings. Wireless Systems and Network Architectures in Next Generation Internet*, volume 3883 of *Lecture Notes in Computer Science*, pages 198–210. Heidelberg, Springer-Berlim, 2006.

[3] M. Day, S. Aggarwal, G. Mohr, and J. Vincent. Instant Messaging / Presence Protocol Requirements. RFC 2779, February 2000.

[4] M. Day, J. Rosenberg, and H. Sugano. A Model for Presence and Instant Messaging. RFC 2778, February 2000.

[5] Ericsson, Motorola, and Nokia. Wireless Village The Mobile IMPS Initiative, System Architecuture Model, April 2002. Available at http://www.openmobilealliance.org/tech/affiliates /wv/wv_architecture_v1.0.pdf.

[6] P. A. Felber, R. Guerraoui, and A.-M. Kermarrec. The Many Faces of Publish/Subscribe. *ACM Computing Surveys (CSUR)*, 35(2):114–131, June 2003.

[7] L. Gong. JXTA: A Network Programming Environment. *IEEE Communications Magazine*, 5(3):88–95, May 2005.

[8] D. Harrington, R. Presuhn, and B. Wijnen. RFC 3411 - An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks, December 2002. Available at: http://www.ietf.org/rfc/rfc3411.txt?number=3411.

[9] C. C. Marquezan, C. R. P. dos Santos, E. M. Salvador, M. J. B. Almeida, S. L. Cechin, and L. Z. Granville. Performance evaluation of notifications in a web services and p2p-based network management overlay. In *Proceedings. IEEE International Computer Software and Applications Conference (COMPSAC 2007)*, pages 241–250. IEEE Computer Society, 2007.

[10] A. Panisson, C. Melchiors, L. Z. Granville, M. J. B. Almeida, and L. M. R. Tarouco. Designing the Architecture of P2P-Based network Management Systems. In *Proceedings. IEEE Symposium on Computers and Communications (ISCC'06)*, pages 69–75, Los Alamitos, CA, USA, 2006. IEEE Computer Society.

[11] P. Saint-Andre. Extensible Messaging and Presence Protocol (XMPP): Core. RFC 3920, October 2004.

[12] R. State and O. Festor. A Management Platform Over Peer-to-Peer Service Infrastructure. In *Proceeding. 10th International Conference on Telecommunications, 2003. ICT 2003*, pages 124–131, Vancouver, BC, Canada, 2003.