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

FELIPE JUNG VILANOVA

Uma Ferramenta Peer-to-Peer para Gerenciamento Cooperativo de Redes

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

Profa. Dra. Maria Janilce Bosquiroli Almeida
Orientadora

Porto Alegre, agosto de 2006.

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Vilanova, Felipe Jung

Uma Ferramenta Peer-to-Peer para Gerenciamento Cooperativo de Redes / Felipe Jung Vilanova – Porto Alegre: Programa de Pós-Graduação em Computação, 2005.

63 f.:il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação. Porto Alegre, BR – RS, 2005. Orientadora: Maria Janilce Bosquiroli Almeida.

1. Gerenciamento de redes distribuído. 2. Gerenciamento de redes cooperativo. 3. Peer-to-Peer. 4. JXTA. I. Almeida, Maria Janilce B. II. Título.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

AWT	Abstract Window Toolkit
CD	Compact Disk
CLI	Command Line Interface
CMSPro	Cooperative Management Service Provider
CPU	Central Processing Unit
DLL	Dynamic Link Library
DNS	Domain Name System
ERP	Endpoint Router Protocol
FTP	File Transfer Protocol
HTTP	HiperText Transfer Protocol
ICQ	I seek you
IP	Internet Protocol
JDK	Java Development Kit
JXTA	Juxtapose
MP3	MPEG Audio Layer-3
MSN	Microsoft Network
NAT	Network Address Translator
P2P	Peer-to-Peer
PBP	Pipe Binding Protocol
PDA	Personal Digital Assistant
PDP	Peer Discovery Protocol
PRP	Peer Resolver Protocol
RSVP	Resource Reservation Protocol
RVP	Rendezvous Protocol
SETI	Search for Extraterrestrial Intelligence
SNMP	Simple Network Management Protocol

SSH	Secure Shell
TCP	Transport Control Protocol
XML	Extencible Markup Language
XORP	Extensilble Open Router Platform

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Com o crescimento em número e diversidade dos componentes das redes de computadores, surge a necessidade de buscar uma maneira consistente de realizar seu gerenciamento para, com isso, manter toda sua estrutura funcionando de forma suave e atendendo às necessidades de seus usuários e às expectativas de seus administradores. Em aspectos gerais, as abordagens centralizadas têm se mostrado inadequadas para o gerenciamento de redes de computadores com um grande número de nós ou com grande diversidade de dispositivos. A necessidade de distribuição da gerência torna-se, assim, evidente, onde vários operadores administram cooperativamente a rede, cada um sendo responsável por um segmento da mesma, mas que precisam poder inspecionar os demais segmentos para poder resolver problemas distintos. Por isso a necessidade de um sistema de suporte ao gerenciamento cooperativo, que forneça suporte à interação dos administradores, independente de tempo e da localização dos participantes. É esse sistema que determinará como os administradores vão se comunicar, distribuir responsabilidades, compartilhar informações, e utilizar as ferramentas disponíveis. Nesse contexto, esta dissertação de mestrando apresenta a proposta de um ambiente de gerenciamento distribuído e cooperativo, baseado na tecnologia P2P. Esse ambiente oferece quatro serviços: compartilhamento de arquivos de configuração de dispositivos, compartilhamento de registros de dispositivos, configuração de dispositivos e solicitação de reserva de banda. A partir da proposta do ambiente foi implementada em Java, utilizando a plataforma de desenvolvimento JXTA, uma aplicação para a realização de testes, com o objetivo de confirmar a possibilidade de utilização da aplicação para realizar o gerenciamento de redes. Os parâmetros observados nas avaliações de desempenho foram o tráfego gerado e o tempo de resposta. Os resultados dos testes comprovaram a possibilidade da utilização de sistemas P2P para facilitar o gerenciamento cooperativo de redes e foram bastante satisfatórios com relação aos parâmetros avaliados.

Palavras-Chave: Gerenciamento de redes distribuído, Gerenciamento de redes cooperativo, Peer-to-Peer, JXTA.

A Peer-to-Peer Tool for Cooperative Network Management

ABSTRACT

With the increasing number and diversity of components of the computer networks, comes the necessity to find a consistent way to manage modern networks. This is required in order to keep all network infrastructures working in a proper and smooth way, while taking care of to the network users' necessities and filling the network administrators' expectations. In general aspects, the centralized management approaches are inadequate for the management of networks with a great number of nodes or great diversity of devices. The necessity of a distributed management thus becomes evident. In such distributed management, a group of administrators manages a single network in a cooperative fashion, each administrator being responsible for a segment of the managed network but at the same time being able to inspect other remote segments in order to solve non-overlapping problems. That is the motivation for having management systems with proper cooperative management support that allows interactions among administrators independent of time and location. Such a system will determine how administrators will communicate to delegate responsibilities, share information, and use the available tools. In this context, this dissertation presents the proposal of a distributed and cooperative management environment based on P2P technology. This environment offers four services: sharing of device configuration files, sharing of registered devices, configuration of devices, and bandwidth reservation. Based on the proposed environment, a Java application, using the JXTA development platform, has been implemented. This application allowed us to test and confirm the possibility of using P2P technologies for network management. In addition, the performance of the implemented solution has been measured considering the generated management traffic and response time. The results from the evaluation tests had proven the possibility of the use of P2P systems to facilitate the cooperative network management and had been sufficiently satisfactory in relation to the evaluated parameters.

Keywords: Distributed network management, cooperative network management, Peer-to-Peer, JXTA.

1 INTRODUÇÃO

O grande avanço nas áreas de telecomunicações e de redes de computadores, aliado à grande redução de custos dos recursos computacionais, motivou a proliferação das redes por todos os segmentos da sociedade. Isso trouxe consigo um aumento na diversidade de recursos e serviços oferecidos, o que, por sua vez, tem aumentando a complexidade das redes. Não bastassem esses fatos, os sistemas computacionais ainda apresentam grande heterogeneidade dos padrões de redes, sistemas operacionais, equipamentos, etc.

Atualmente as redes de computadores e os seus recursos associados têm tornado-se tão importantes para uma organização, que elas basicamente "não podem falhar". Além disso, com o crescimento em número e diversidade dos componentes dessas redes, surge a necessidade de buscar uma maneira consistente de realizar seu gerenciamento para, com isso, manter toda sua estrutura funcionando de forma suave e atendendo às necessidades de seus usuários e às expectativas de seus administradores.

As abordagens de gerência de rede no fim dos anos 80s eram baseadas em uma arquitetura estritamente centralizada, que consiste em uma estação central de gerência e em um número de agentes fornecendo a informação necessária (SCHÖNWÄLDER; QUITTEK; KAPPLER, 2000). Em aspectos gerais, as abordagens centralizadas de gerenciamento têm se mostrado inadequadas para o gerenciamento de redes com um grande número de nós ou com grande diversidade de dispositivos, devido ao grande volume de informações a serem tratadas e à dispersão geográfica das mesmas. Segundo Schönwälder, Quittek e Kappler (2000), o crescimento da heterogeneidade de tecnologias e equipamentos tem tornado a paradigma centralizado ineficiente, além de barrar a escalabilidade do sistema.

De acordo com Goldszmidt e Yemini (1995), os sistemas centralizados estabelecem diversas barreiras à gerência efetiva. A alocação rígida de funções e responsabilidades de gerenciamento conduz a muitos problemas de escalabilidade, e um sistema centralizado gera “gargalos” suscetíveis a falhas na comunicação. Durante instantes de falhas, em particular, tendem a aumentar as taxas de acesso a dados, justamente em um momento em que a rede está menos capaz de suportar. Como o ponto central é responsável pela maioria das funções de gerenciamento, fica mais vulnerável às falhas de rede, e se esse ponto central estiver inativo ou sobrecarregado, os dispositivos não podem realizar a recuperação, pois devem esperar instruções do próprio. Logo, mesmo um pequeno problema pode potencialmente conduzir a uma avalanche de falhas em todo sistema de gerenciamento da rede, levando-o a uma parada completa.

A necessidade de distribuição da gerência torna-se, assim, evidente, onde vários operadores administram cooperativamente a rede, cada um sendo responsável por um segmento da mesma, mas que precisam poder inspecionar os demais segmentos para

poder resolver problemas distintos. O gerenciamento de redes de grande porte geralmente é feito em várias localidades, e essas precisam trabalhar em conjunto para resolver os problemas. Cada uma dessas localidades deve ser capaz de compartilhar informações com os administradores externos para resolver problemas entre os usuários e a infra-estrutura da rede. Por isso a necessidade de um sistema de suporte ao gerenciamento cooperativo, que forneça suporte à interação dos administradores, independente de tempo e da localização dos participantes. É esse sistema que determinará como os administradores vão se comunicar, distribuir responsabilidades, compartilhar informações, e utilizar as ferramentas disponíveis.

A maior parte dos serviços de Internet são distribuídos utilizando o tradicional modelo cliente/servidor, ilustrado na figura 1.1. Nesse modelo os clientes utilizam um protocolo de comunicação específico para acessar um recurso especifico e grande parte do processamento envolvido no serviço ocorre no servidor. Esse modelo tem a grande desvantagem de possuir um ponto central de falhas, além do fato de que com o crescimento do número de clientes o servidor pode ficar sobrecarregado.

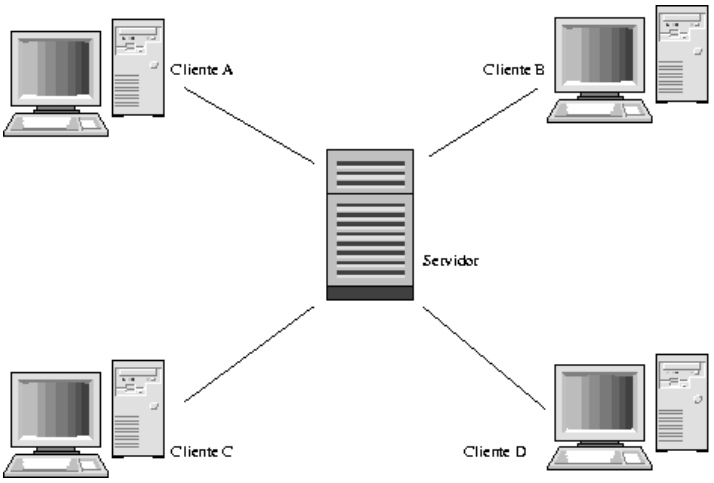

Figura 1.1: Modelo Cliente/Sevidor

O cliente em um modelo cliente/servidor tem um papel passivo, ou seja, pode efetuar pedidos a serviços mas não pode disponibilizar serviços a outros clientes. Uma outra abordagem para serviços distribuídos é o modelo *peer-to-peer* (P2P), o qual dá a máquinas individuais a capacidade de fornecer serviços umas às outras. Ao contrário de uma rede cliente/servidor, redes P2P não dependem de servidores centrais, disponibilizando uma rede plana e interconectada, como apresentado na figura 1.2. O modelo P2P permite que as máquinas ajam como clientes e servidores ao mesmo tempo.

As barreiras para tais sistemas iniciarem e crescerem são baixas, geralmente não requerem recursos administrativos ou financeiros especiais, ao contrário dos sistemas centralizados. Os sistemas P2P sugerem uma maneira agregar e empregar os recursos computacionais e de armazenamento que estariam disponíveis em computadores através da Internet. A natureza descentralizada e distribuída desses sistemas dá-lhes o potencial de ser tolerante às falhas ou aos ataques intencionais, tornando-os ideais para o armazenamento e processamento de longo prazo (BALAKRISHNAN, 2003).

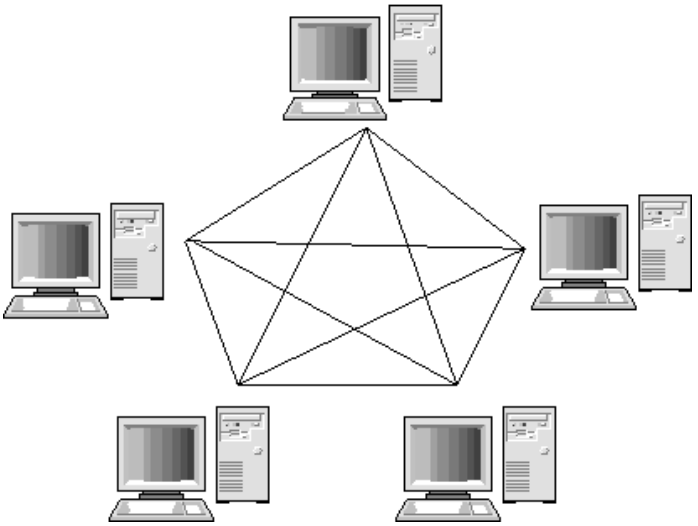

Figura 1.2: Modelo P2P

Segundo Granville et al. (2005), os serviços introduzidos pelos sistemas P2P apresentam características inovadoras, e sua utilização vai além de simples compartilhamento de arquivos ou computação distribuída, eles também podem ser utilizados para ajudar a resolver problemas de outras áreas críticas, como por exemplo, a gerência de redes.

Este trabalho tem como objetivo investigar as plataformas para desenvolvimento de sistemas *peer-to-peer* (P2P), especificar e implementar um ambiente P2P com funções básicas de cooperação e definir um conjunto de serviços de gerenciamento de redes a ser incluído no ambiente implementado.

O resultado concreto final é um ambiente de gerenciamento distribuído e cooperativo, baseado na tecnologia P2P. Esse ambiente oferece quatro serviços: compartilhamento de arquivos de configuração de dispositivos, compartilhamento de registros de dispositivos, configuração de dispositivos e solicitação de reserva de banda. A proposta de utilização sistemas P2P para gerenciamento de redes é avaliada através da implementação do ambiente, e da análise dos resultados obtidos.

O restante desta dissertação está organizado da seguinte forma: No capítulo 2 são apresentados o conceito e objetivos dos sistemas P2P, suas características e componentes. O capítulo 3 descreve a plataforma JXTA e alguns conceitos a seu respeito necessários à compreensão deste trabalho. No capítulo 4 é apresentada a arquitetura proposta para o ambiente de gerenciamento, seus componentes, inter-relacionamentos e características. A implementação dessa arquitetura está documentada no capítulo 5. No capítulo 6 são descritos os cenários de testes utilizado para avaliar a ferramenta e apresentados os resultados obtidos. Por fim, o capítulo 7 apresenta as conclusões e sugestões de trabalhos futuros.

2 SISTEMAS PEER-TO-PEER

> Sistemas e aplicações peer-to-peer são sistemas distribuídos sem qualquer controle centralizado ou organização hierárquica, onde todos os nodos executam programas com funcionalidades equivalentes (STOICAY et al., 2001).

> O termo "peer-to-peer" (P2P) refere-se a uma classe de sistemas e aplicações que utilizam recursos distribuídos para executar funções críticas de um modo descentralizado. Os recursos incluem poder de processamento, dados, banda, e presença. A função crítica pode ser processamento distribuído, troca de arquivos, comunicação e colaboração, ou serviços de plataforma (MILOJICIC et al., 2002).

Sistemas *peer-to-peer* (P2P) são sistemas distribuídos, com uma estrutura extremamente dinâmica e descentralizada, e com o objetivo de compartilhar recursos computacionais através de comunicação direta entre seus componentes. Um sistema onde a capacidade de armazenamento, os ciclos de processamento e as informações de cada integrante podem ser aproveitados por todos. Pode ser visto como um sistema operacional que oferece serviços para o compartilhamento de recursos distribuídos, utilizando uma estrutura física já existente.

Esses sistemas têm como objetivo compartilhar os custos de manutenção das aplicações entre os usuários (por exemplo, nas aplicações de troca de arquivos, o espaço de armazenamento é fornecido pelos usuários), possibilitar a agregação de recursos e a interoperabilidade, aumentar a autonomia dos sistemas, garantir o anonimato e a privacidade dos usuários e oferecer suporte a ambientes dinâmicos, onde os dispositivos entram e saem constantemente. A independência de cada nodo da rede traz características muito desejadas nas redes atuais, como a descentralização, a escalabilidade e a tolerância à falhas.

Uma das idéias da descentralização é que os usuários mantenham e controlem os dados e recursos. Como benefícios imediatos da descentralização temos a capacidade de suportar um grande número de usuários, e eliminar pontos centrais de falhas. Mas isso dificulta a implementação de modelos P2P, pois não há um servidor central com uma noção geral de todos os dispositivos da rede e os dados que eles disponibilizam. Essas informações têm que ser conhecidas por todos os nodos, e quando um nodo entra na rede, deve executar mecanismos para descobrir os outros dispositivos, e guardar informações sobre seus endereços. Por essa razão, muitos sistemas *peer-to-peer* são implementados de uma maneira híbrida, como, por exemplo, Napster, onde há um diretório central para os arquivos, mas os nodos fazem o *download* desse arquivo diretamente do dispositivo a que ele pertence.

Antes do surgimento das aplicações P2P, a utilização da Internet por usuários comuns consistia em uma rede praticamente cliente/servidor, com diversos clientes requisitando conteúdo e serviços publicados por servidores com endereços fixos registrados no DNS (*Domain Name System*). Com a inovação tecnológica e a popularização de diversos dispositivos com acesso a Internet, passamos a contar com uma rede na qual esses dispositivos também são capazes de fornecer recursos, porém nem sempre estarão conectados ou utilizando os mesmos endereços. O aumento da largura de banda, e a maior disponibilidade de acesso fizeram com que os usuários sentissem a necessidade de uma rede mais colaborativa, na qual a busca de conteúdo e serviços, além da interação com outros usuários, deixa de ser privilégio de alguns servidores. Essas talvez tenham sido as motivações para o constante crescimento das redes e aplicativos P2P (TRUELOVE, 2001).

2.1 Componentes Básicos de um Sistema P2P

A figura 2.1 mostra uma arquitetura P2P informal, proposta por Milojicic et al. (2002), com os componentes básicos de um sistema. Nos itens a seguir serão explicadas as funções e particularidades de cada uma das camadas componentes.

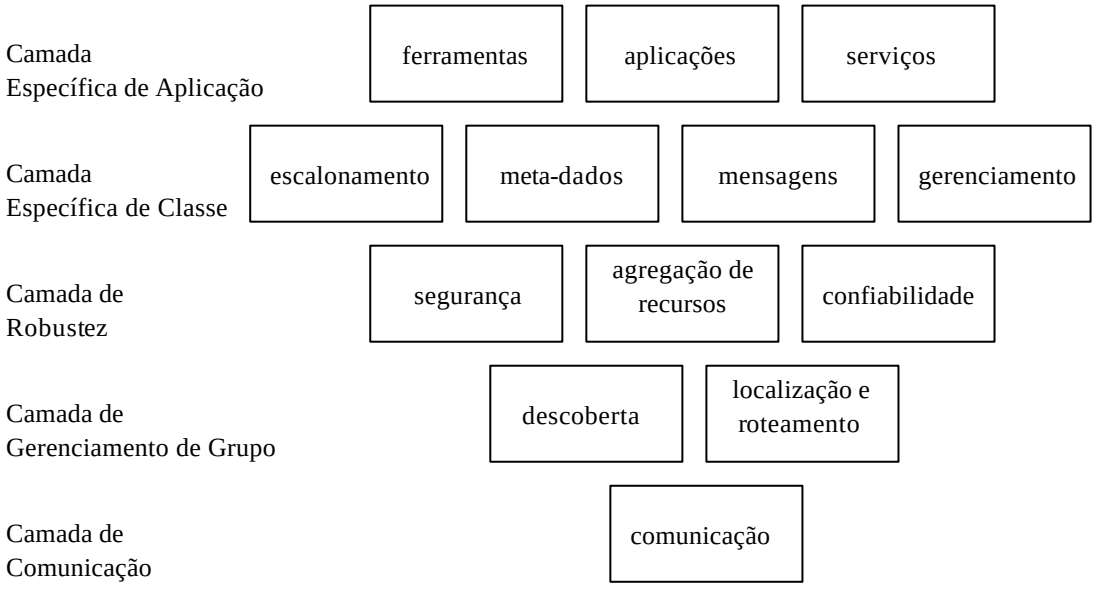

Figura 2.1: Arquitetura P2P

2.1.1 Camada de Comunicação

O modelo P2P cobre uma vasta variedade de paradigmas de comunicação, desde computadores pessoais, conectados na Internet, em sua maioria por canais estáveis e de elevadas velocidades (SAROIU; GUMMADI; GRIBBLE, 2002), até dispositivos portáteis, como PDAs, conectados de uma maneira ad hoc através de um meio sem fio. A camada de comunicação é responsável pela primeira tarefa que um sistema P2P deve executar, e que é o principal desafio dessa tecnologia: o estabelecimento das conexões entre os vários e diferentes dispositivos da rede. O problema em estabelecer e manter essas conexões se deve à natureza dinâmica dos dispositivos participantes.

Seja intencionalmente, como por exemplo, porque um usuário desliga seu computador, ou involuntariamente, devido a uma falha de conexão, os grupos de *peers* mudam freqüentemente, e isso dificulta um diagnóstico da situação corrente da rede, é quase impossível manter um conjunto de informações atualizadas sobre toda a rede. Manter a conectividade em tal ambiente é um dos maiores desafios que enfrentam os desenvolvedores P2P (MILOJICIC et al., 2002).

2.1.2 Camada de Gerenciamento de Grupo

Depois de conectados, os dispositivos devem ter um meio de encontrar os serviços, recursos e outros participantes da rede (PEERMETRICS, 2003). A camada de gerenciamento de grupo inclui funcionalidades para descoberta, localização e roteamento entre esses dispositivos. A descoberta pode ser totalmente centralizada como, por exemplo, no Napster, totalmente distribuída, como Gnutella (RIPEANU; IAMNITCHI; FOSTER, 2002), ou qualquer combinação entre as duas abordagens.

Diversos fatores influenciam a arquitetura de um algoritmo de descoberta, por exemplo, dispositivos móveis sem fio podem descobrir outros participantes baseados em sua área de comunicação (ROMAN; HUANG; HAZEMI, 2001). Já os protocolos para máquinas *desktop* usam freqüentemente outras abordagens, como diretórios centralizados. Os algoritmos de localização e roteamento tentam otimizar o caminho de uma mensagem transportada de um dispositivo a outro, enquanto sistemas como Napster e Gnutella tentam otimizar fatores como latência da rede.

2.1.3 Camada de Robustez

A robustez em sistema P2P é garantida por três componentes principais: segurança, agregação de recursos e confiabilidade. A segurança, outro dos maiores desafios dessa tecnologia, é essencial em qualquer sistema. O fato de os dispositivos poderem atuar como clientes e servidores, ao mesmo tempo, coloca o sistema em posição de risco. Cada participante em um sistema P2P deve proteger seus recursos e serviços da intrusão, tanto dos outros participantes da mesma rede quanto de acessos externos não autorizados (PEERMETRICS, 2003).

Essa necessidade de segurança exige um constante controle da parte de cada usuário, ou a interação desse usuário com uma terceira parte, capaz de validar a identidade dos usuários. Mas centralizar os mecanismos de segurança é uma solução que anula os benefícios de uma estrutura descentralizada. Outro aspecto relativo à segurança é a utilização de sistemas de criptografia para a transmissão dos dados.

O modelo P2P fornece uma base para que os dispositivos agreguem recursos como arquivos, informações, o poder de processamento de suas CPUs , banda, energia e espaço de armazenamento. Classificar a arquitetura de um componente P2P de agregação de recursos é difícil devido à grande variedade dos recursos que podem ser agregados através dos *peers*.

A confiabilidade, por sua vez, é também outro grande problema dos sistemas P2P. A natureza distribuída desses sistemas torna difícil garantir que tenham um comportamento confiável. A solução mais comum para esse problema é a redundância, por exemplo, no caso de aplicações de processamento distribuído, após a detecção de uma falha em um dispositivo, a tarefa que ele estava executando pode ser reiniciada em outras máquinas. Ou a mesma tarefa pode ser executada em mais de um dispositivo. Em

aplicações de troca de arquivos, os dados podem ser replicados em diversos dispositivos.

2.1.4 Camadas de Serviços Específicos de Cada Classe e de Cada Aplicação

Os serviços discutidos anteriormente são aplicáveis a qualquer sistema P2P, mas também existem serviços aplicáveis somente a determinadas classes desses sistemas. Em sistemas de processamento paralelo ou distribuído são necessários serviços de escalonamento, para controlar a execução das pequenas tarefas distribuídas pela rede.

Os serviços de meta-dados descrevem as informações armazenadas pelos dispositivos da rede, e são utilizados pelos sistemas de troca de arquivos para determinar a localização dos dados desejados. Os serviços de troca de mensagens são utilizados pelas aplicações de colaboração para permitir a comunicação. E os serviços de gerenciamento servem de interface entre as aplicações e a infra-estrutura P2P.

A última camada engloba as ferramentas, aplicações e serviços com funcionalidades específicas. Correspondem a casos específicos de aplicações de processamento distribuído (financeiro, bio-tecnológico,...), troca de arquivos (MP3, documentos,...), ou aplicações para colaboração e sistemas de comunicação, como *chat*, mensagens instantâneas, calendário, etc.

2.2 Categorias de Sistemas P2P

Esta seção descreve as categorias de sistemas P2P, ilustradas na figura 2.2. Os sistemas P2P podem ser divididos nas seguintes categorias: protocolos, plataformas e aplicações, descritas a seguir.

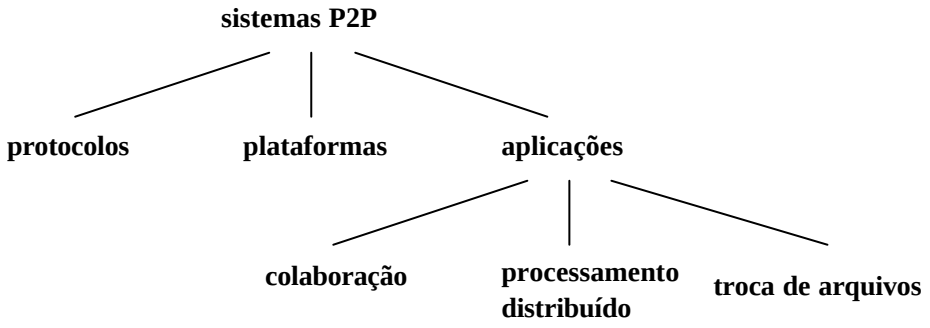

Figura 2.2: Categorias de Sistemas P2P

2.2.1 Protocolos

Na categoria dos protocolos estão relacionadas tecnologias que definem, ou tentam definir, padrões para comunicação em redes P2P. Esses padrões definidos pelos protocolos são utilizados como base para o desenvolvimento das aplicações ou plataformas P2P.

Um exemplo de protocolo é Gnutella (RIPEANU; IAMNITCHI; FOSTER, 2002), provavelmente o mais conhecido, o qual é utilizado para pesquisa e compartilhamento distribuídos de informação. Esse protocolo define de que modo os *peers* se comunicam através da rede. Consiste em um conjunto de descritores para a comunicação de dados e um conjunto de regras que regulam a troca desses descritores entre os participantes da

rede. O sistema P2P definido por Gnutella é totalmente descentralizado, sem alguma autoridade central.

2.2.2 Plataformas

As plataformas são sistemas que oferecem os componentes P2P básicos, como descoberta, comunicação, segurança e agregação de recursos. Servem como base para o desenvolvimento e utilização das aplicações. As plataformas permitem que desenvolvedores sem conhecimentos avançados dos padrões de comunicação em redes P2P possam implementar aplicações dessa natureza.

Como exemplos bem populares dessas plataformas podem ser citados .NET (MICROSOFT 2003), Bluetooth (HAARTSEN et al, 1998) e JXTA (TRAVERSAT et al., 2003). .NET é um conjunto de aplicações, baseadas em Web Sevices, para estabelecer conexão entre pessoas, sistemas e dispositivos através da Internet. Fornece serviços de armazenamento de arquivos, gerenciamento das preferências dos usuários, calendário, entre outros. Bluetooth é utilizada para estabelecer conexão entre dispositivos móveis e JXTA é um conjunto de protocolos P2P baseados em mensagens XML para o desenvolvimento de aplicativos distribuídos. A plataforma JXTA está descrita com mais detalhes no capítulo seguinte.

2.2.3 Aplicações

As aplicações são os sistemas com funcionalidades específicas e podem, por sua vez, ser subdivididas em outras três categorias: processamento distribuído, colaboração e troca de arquivos. As aplicações de processamento distribuído utilizam o poder computacional disponível dos seus usuários para formar supercomputadores. Esses sistemas dividem grandes tarefas em pequenos pedaços, e distribuem esses pedaços para serem processados pelos dispositivos da rede (ANDROUTSELLIS-THEOTOKIS; SPINELLIS, 2004). Um exemplo desse tipo de aplicação é SETI@home (ANDERSON et al., 2002), um experimento científico para detectar a existência de inteligência extraterrestre que usa o poder de processamento livre dos computadores conectados para analisar sinais de radio capturados, em tempo real.

As aplicações de colaboração permitem que os usuários colaborem em tempo real. Permitem a troca de informações através de mensagens instantâneas, *chat*, calendários, entre outros. Também são muito usadas para o desenvolvimento de jogos *multiplayer*. ICQ (ICQ, 2003) e MSN (MICROSOFT, 2005) são exemplos conhecidos desse tipo de aplicação.

E as aplicações de troca de arquivos têm como objetivo armazenar e distribuir arquivos pela rede. Disponibilizam funcionalidades de busca e catálogo de arquivos, assim como mecanismos para otimizar e acelerar as transferências. BearShare (BEARSHARE, 2003) é uma aplicação P2P de troca de arquivos, baseada no protocolo Gnutella. É muito comum às aplicações de colaboração permitirem a troca de arquivos, e vice-versa.

3 PLATAFORMA JXTA

JXTA (Juxtapose) foi a plataforma escolhida para implementação do ambiente de gerenciamento por ser é a mais difundida dentre as encontradas, apresentar uma ampla comunidade de usuários e desenvolvedores, disponibilizar vasta documentação e ser de código aberto. O projeto da arquitetura foi desenvolvido sobre esta plataforma, de modo que a aplicação utilize os protocolos e serviços para estabelecer uma rede P2P independente, sobre a Internet.

JXTA é um projeto de código aberto, um conjunto de protocolos P2P baseados em mensagens XML para o desenvolvimento de aplicativos distribuídos. O nome vem de *juxtapose* (justapor), uma referência ao modelo oposto de aplicações P2P em relação aos modelos tradicionais utilizados como cliente/servidor. Não é uma linguagem de programação, é uma especificação que tem implementações em diversas linguagens como C e Java (VERBEKE et al., 2002). É uma plataforma para construção de sistemas P2P, permitindo que qualquer dispositivo conectado em uma rede, independente de sua plataforma, natureza, ou protocolo de rede possa interagir, compartilhar recursos, e formar uma rede distribuída, descentralizada e cooperativa (LI, 2001). Os protocolos que compõem essa plataforma permitem que os *peers* localizem outros *peers*, serviços, e recursos, enviem mensagens entre si através de canais virtuais chamados *pipes*, obtenham informações sobre outros *peers*, compartilhem serviços e recursos, entrem e saiam de grupos e interajam utilizando uma topologia P2P sobre uma topologia física diversa.

O projeto foi iniciado pela Sun Microsystem em abril de 2001 e teve como arquiteto líder o chefe do departamento de computação da Sun, Billy Joy. O objetivo do JXTA é facilitar o desenvolvimento de aplicativos P2P, encapsulando funcionalidades e serviços comuns, escondendo do desenvolvedor a complexidade das implementações, podendo esse preocupar-se somente com a lógica e os detalhes pertinentes à sua aplicação (SUN MICROSYSTEMS 2001).

A plataforma estabelece uma rede virtual que opera sobre a Internet ou outras redes, permitindo que os *peers* interajam diretamente e auto-organizem-se, independente de sua conectividade física (TRAVERSAT et al., 2003). Em outras palavras, ela fornece um modo completamente descentralizado para que os *peers* comuniquem-se e compartilhem recursos uns com os outros, sem considerar um sistema centralizado para serviços de endereçamento ou armazenamento. Em uma rede JXTA *peers* podem ser ligados, desligados, reiniciados, removidos ou substituídos constantemente (LI, 2001). A figura 3.1 ilustra como a plataforma JXTA forma uma rede virtual, permitindo a comunicação transparente entre *peers* que utilizam protocolos diferentes ou pertencem a redes diferentes.

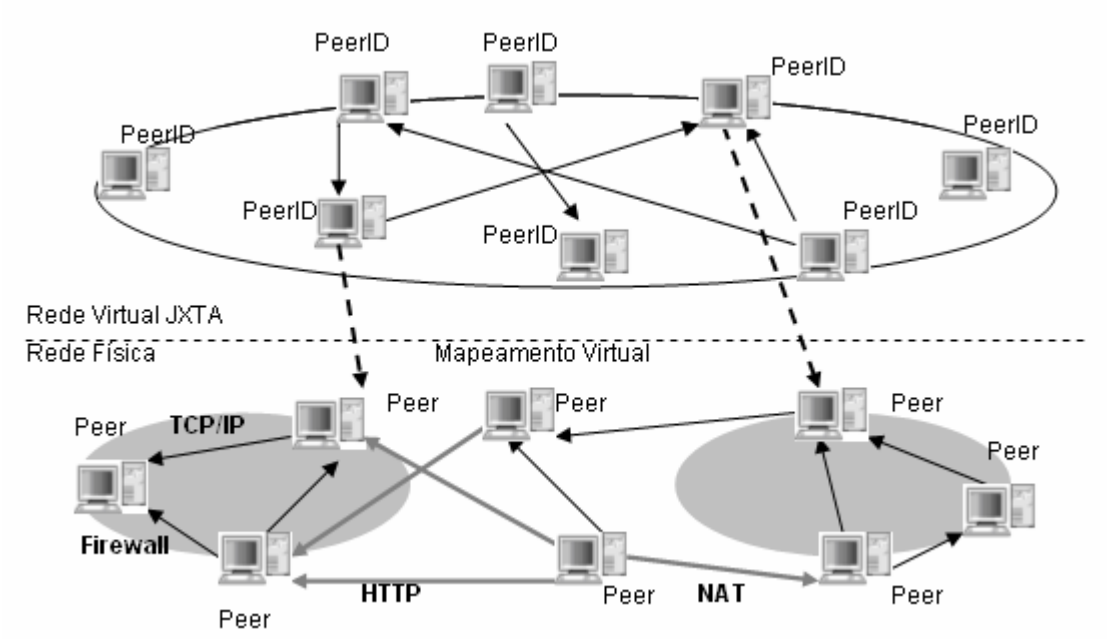

Figura 3.1: Rede virtual JXTA (TRAVERSAT et al., 2003)

3.1 Conceitos JXTA

JXTA define algumas abstrações importantes que permitem a formação de uma camada de rede virtual. Primeiramente um endereço lógico para cada *peer* é definido em toda rede, através de um *peer ID*. Os *peers* auto-organizam-se em grupos chamados *peer groups*, formando domínios virtuais com características pré-definidas, e são responsáveis também pela criação de recursos e serviços, como por exemplo os *peers groups*, que precisam ser publicados para serem válidos para os outros participantes da rede JXTA. Cada recurso ou serviço possui um tipo específico de *advertisement*. As operações de *bind* necessárias em ambientes distribuídos, como traduções de nomes em endereços de rede, são implementadas por mecanismos denominados *resolvers*, e *pipes* são utilizados para comunicação entre *peers* de uma forma transparente (TRAVERSAT et al., 2003). As principais abstrações e componentes característicos da rede JXTA estão definidas nos itens a seguir.

3.1.1 Peers

Um *peer* é qualquer entidade que possa participar e interagir com a rede, seja ela um computador, um processo, um processador ou até mesmo um usuário, e é fundamental em um sistema JXTA. Eles são tanto consumidores quanto fornecedores de serviços em uma rede. Podem ser associados a um ou mais pontos físicos de uma rede e esses pontos podem mudar dinamicamente. Isso significa que um *peer* pode "navegar" por diferentes redes físicas mantendo a mesma identificação. Em JXTA os *peers* são representados por um identificador único, o *Peer ID* (LI, 2001).

Para ser considerado um *peer*, uma entidade deve ao menos entender os protocolos de comunicação básicos definidos pela tecnologia, como o *Peer Resolver Protocol* e o *Endpoint Router Protocol*, explicados mais adiante neste capítulo.

3.1.2 Peer Groups

Um *peer group* é um conjunto de *peers* com serviços comuns, identificado por um *peer group ID* único. Os *peers* auto-organizam-se nesses grupos, e cada *peer* pode pertencer a um ou mais grupos simultaneamente. Por padrão, o primeiro *peer group* instanciado é o *Net Peer Group*, do qual todos os *peers* fazem parte. Os protocolos JXTA descrevem como os *peers* devem publicar, descobrir, monitorar e juntar-se a *peer groups*. JXTA define o seguinte conjunto de serviços que são disponibilizados por um *peer group* (SUN MICROSYSTEMS, 2003):

- Serviço de Descoberta (*Discovery Service*) - utilizado pelos *peers* para procurar pelos recursos do *peer group*, como por exemplo, outros *peers*, *peers groups*, *pipes* e serviços.
- Serviço de Membros (*Membership Service*) - utilizado pelos membros do grupo para aceitar ou rejeitar um novo *peer*. *Peers* que desejam juntar-se a um *peer group* devem localizar um membro desse grupo, e então pedir permissão. O pedido é aceito ou negado por todos membros do grupo, por votação ou eleição de um representante para tomar a decisão.
- Serviço de Acesso (*Access Service*) - utilizado para validar requisições feitas de um *peer* para outro.
- Serviço de Canal (*Pipe Service*) - utilizado para criar e gerenciar canais de comunicação entre os membros do *peer group*.
- Serviço de Resolução (*Resolver Service*) - utilizado para enviar consultas genéricas a outros *peers*. Os *peers* podem definir e trocar consultas para encontrar qualquer informação necessária, como por exemplo, o estado de um serviço.
- Serviço de Monitoramento (*Monitoring Service*) - permite que um *peer* monitore outros do mesmo *peer group*.

3.1.3 JXTA IDs

Peers, *peer groups*, *pipes* e outros recursos JXTA recebem um identificador único. Esses identificadores contêm 128 bits e são gerados randomicamente por cada entidade (TRAVERSAT et al., 2003). Abaixo podem ser vistos dois exemplos de JXTA ID.

JXTA ID de um *peer*:

urn:jxta:uuid-
59616261646162614A7874615032503 3F3BC76FF13C2414CBC0AB663666DA53903

JXTA ID de um pipe:

urn:jxta:uuid-
59616261646162614E504720503250338E3E786229EA460DADC1A176B69B731504

3.1.4 Serviços de Rede

Os *peers* cooperam e se comunicam para publicar, descobrir e invocar *serviços de rede (network services)*. Eles podem publicar múltiplos serviços, os quais são

descobertos via *Peer Discovery Protocol*. Os protocolos JXTA reconhecem dois níveis de serviços de rede:

- Serviços de *Peer* (*Peer Services*)

Serviço acessível somente no *peer* que está publicando esse serviço. Se o *peer* falhar o serviço também falhará.

- Serviços de *Peer Group* (*Peer Group Services*)

É uma coleção de instâncias de um serviço operando em múltiplos membros do *peer group*. Se um *peer* falhar, o serviço não é afetado, desde que esteja disponível em outro membro do grupo. Os serviços de *peer group são* publicados como parte do *peer group advertisement*.

3.1.5 Módulos JXTA

Módulo JXTA (*JXTA module*) é uma abstração usada para representar um trecho de "código" utilizado para implementar um comportamento na plataforma JXTA. Esse "código" pode ser uma classe Java, uma DLL, um conjunto de mensagens XML ou um script. Os *serviços de rede* são o exemplo mais comum de um comportamento que pode ser instanciado em um *peer*.

A estrutura de um módulo permite a representação e divulgação de comportamentos independentes de plataforma, e permite que *peers* descrevam e instanciem qualquer tipo de implementação de um comportamento. A abstração de módulo inclui três partes: *Module Class, Module Specification* e *Module Implementation*.

A *Module Class* é usada para divulgar a existência de um comportamento, e sua definição representa esse comportamento. A especificação (*Module Specification*) é usada para acessar o módulo, contendo todas informações necessárias para esse fim. E *Module Implementation* é a implementação propriamente dita de um *Module Specification*.

Os módulos são utilizados pelos serviços de *peer groups*, para identificar a existência do serviço (seu *Module Class*), da especificação (seu *Module Specification*) ou da implementação do serviço (um *Module Implementation*). Cada um desses componentes tem um *advertisement* associado, o qual pode ser publicado e descoberto por outros *peers* (SUN MICROSYSTEMS, 2003).

3.1.6 Pipes

Pipes são canais de comunicação assíncronos e unidirecionais utilizados pelos *peers* como mecanismo para transmissão de mensagens. Não estão associados fisicamente à localização ou ao endereço de um *peer* e são identificados por um *pipe ID*. Por não estarem associados ao endereço físico de um *peer*, aplicações e serviços que se comunicam através de *pipes* não são afetados pela mudança de localização física dos *peers* e eventuais mudanças de rota que possam acontecer durante a transmissão de dados. Os *pipes* podem restringir o conteúdo a ser trafegado e prover comunicação segura entre os *peers* (LI, 2001). A figura 3.2 ilustra a comunicação entre *peers* realizada através de *pipes*.

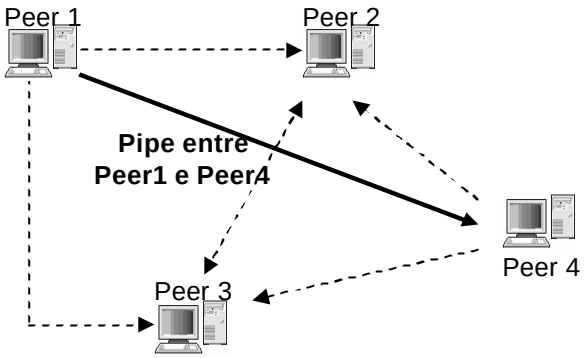

Figura 3.2: Comunicação através de *pipes*

3.1.7 EndPoints

EndPoint é o endereço de rede ao qual um *peer* está associado. Ao se estabelecer uma conexão entre *peers* os *pipes* se encarregam de saber o *endPoint* dos participantes, abstraindo os *peers* de um endereçamento físico (SUN MICROSYSTEMS, 2003).

Um e*ndPoint* é associado a cada *PeerID* e encapsula todos os endereços de rede físicos disponíveis para um *peer*. O *endPoint* de um *peer* é como se fosse um cartão de apresentação, listando muitas maneiras de contatar uma pessoa (telefone, fax, email, etc.). Ao receber o *advertisement* do *endPoint* de um *peer*, um outro *peer* pode selecionar a maneira mais eficiente para comunicar-se com esse, a partir da lista de endereços disponíveis, usando TCP/IP diretamente quando possível, ou HTTP sobre TCP/IP se for necessário passar por um *firewa*ll (TRAVERSAT et al., 2003).

3.1.8 Advertisements

A forma básica de comunicação entre os participantes de uma rede JXTA é através de mensagens XML padronizadas, chamadas *advertisements*, respeitando os protocolos definidos nas especificações do projeto JXTA (LI, 2002). Todos os recursos dessa rede, como *peers*, *peer groups*, *pipes*, e serviços são representados por *advertisements*. Os *peers* armazenam, publicam e trocam *advertisements* para descobrir e encontrar recursos disponíveis na rede. Todos *advertisements* são publicados com um "tempo de vida", que especifica a sua duração na rede. Após expirar essa duração, o *advertisement* pode ser republicado (TRAVERSAT et al., 2003). Os protocolos JXTA definem os seguintes tipos de *advertisements*:

- *Peer Advertisement* – contém informações sobre o *peer*, como seu nome, *peer ID*, endereços e outros atributos.
- *Peer Group Advertisement* – descreve os recursos específicos de um grupo, como nome, *peer group ID*, descrição, especificação, e parâmetros de serviços. A figura 3.3 apresenta um exemplo de *peer group advertisement*.
- *Pipe Advertisement* – descreve um canal de comunicação e é utilizado para criar os pontos de entrada e saída de um *pipe*. Cada *pipe advertisement* contém um ID simbólico opcional, um tipo (*point-to-point*, *propagate*, *secure*, etc.) e um *pipe ID* único.
- *Module Class Advertisement* - descreve um *module class* e documenta formalmente a sua existência. Contém um nome, descrição e um

identificador único (*ModuleClassID*). Esse *advertisement* apenas divulga a existência de um serviço, para acessar o serviço o *peer* precisa encontrar o module *spec advertisement* correspondente.

- *Module Spec Advertisement* - define um *module specification*. Sua finalidade principal é fornecer referências à documentação necessária para criar implementações dessa especificação. Uma outra utilidade é tornar instancias executáveis acessíveis remotamente, publicando informações tal como o *pipe advertisement*. Um *module specification* inclui o nome, a descrição, um identificador único (*ModuleSpecID*), o *pipe advertisement*, e um campo com parâmetros arbitrários.
- *Module Impl Advertisement* - define a implementação de um *module specification*. Inclui nome, *ModuleSpecID* associado, assim como o código, pacote, e campos que permitam que o *peer* recupere os dados necessários para executar a implementação.
- *Rendezvous Advertisement* - descreve um *peer* que age como *rendezvous peer* para um determinado *peer group*.
- *Peer Info Advertisement* - esse *advertisement* contém informação específica sobre o estado atual de um *peer*, tal como instante de início de atividade, contagem de mensagens enviadas e recebidas, tempo da última mensagem recebida, e da última enviada (SUN MICROSYSTEMS, 2003).

```
<?xml version=”1.0”?>
<!DOCTYPE jxta:PGA>
<jxta:PGA xmlns:jxta=”http://jxta.org”>
    <GID>
          urn:jxta:jxta-NetGroup
    </GID>
    <MSID>
          urn:jxta:uuid-DEADBEEFDEAFBABAFEEDBABE000000010206
    </MSID>
    <Name>
          NepPeerGroup
    </Name>
    <Desc>
    NetPeerGroup by default
    </Desc>
</jxta:PGA>
```

Figura 3.3: Peer Group Advertisement

3.1.9 Mensagens

Uma mensagem é um objeto enviado entre os *peers*, é a unidade básica de troca de dados entre eles. Essas mensagens podem ser representadas por um documento XML ou

por um código binário e são formadas por uma seqüência ordenada de elementos (*message elements*). O conteúdo de cada um desses elementos pode ser um tipo arbitrário, como uma *string* ou um vetor de *bytes*.

Os protocolos JXTA são especificados como um conjunto de mensagens XML trocadas entre os *peers*, o que permite que diferentes tipos de *peers* participem em um protocolo (SUN MICROSYSTEMS, 2003).

3.1.10 Relay Peers

Muito dos *peers* participantes da rede JXTA possuem conexões temporárias, e freqüentemente têm o acesso direto à Internet restrito por um *firewall* ou por uma conexão a ser realizada através de um NAT (*Network Address Translator*). Para conseguir acesso aos recursos da rede JXTA esses *peers* necessitam do auxílio de um *relay peer* (TRAVERSAT et al., 2003).

Um *relay peer* mantém informações sobre as rotas para outros *peers* e faz o roteamento de mensagens para os *peers*. Um *peer*, ao tentar comunicar-se com outro, procura em sua memória local informações sobre a rota. Se não encontrar essa informação, envia um pedido a um *relay peer* perguntando sobre a rota. *Relay peers* também encaminham mensagens de *peers* que não podem endereçar diretamente um outro, como por exemplo, *peers* de diferentes redes (SUN MICROSYSTEMS, 2003). Qualquer *peer*, desde que tenha as devidas permissões em seu grupo, pode tornar-se um *relay peer*. A divulgação de *relay peers* é feita pela publicação de um *advertisement* específico para esta finalidade. A figura 3.4 exemplifica a utilização de um *relay peer*.

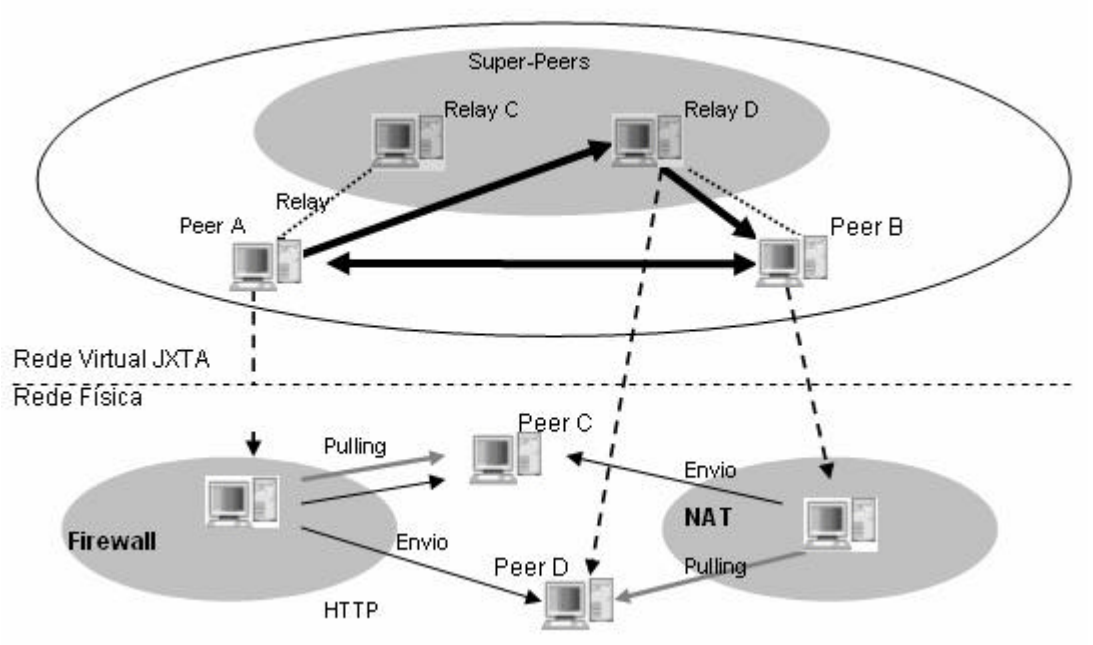

Figura 3.4: *Relay Peer* (TRAVERSAT et al., 2003)

3.1.11 Rendezvous Peers

Um *rendezvous peer* é um *peer* como qualquer outro, e mantém um *cache* de *advertisements* que indexam informações sobre outros *peers* e recursos que ele tem

conhecimento, além de encaminhar requisições de descoberta para ajudar outros *peers* a encontrarem recursos. Quando um *peer* entra em um grupo, automaticamente procura por um *rendezvous peer*, e se não houver um ele torna-se o *rendezvous peer* daquele *peer group* (SUN MICROSYSTEMS, 2003).

Caso não haja nenhuma informação disponível no índice de um *rendezvous peer*, a requisição de descoberta é passada adiante para outro *rendezvous peer*, e assim por diante. *Rendezvous peers* são de fundamental importância na rede para diminuir o tráfego de mensagens enviadas e para otimizar o processo de busca de recursos, evitando que a busca propague-se pela a rede inteira. A divulgação de um *rendezvous peers* é feita pela publicação de um *advertisements* específico para este propósito. (TRAVERSAT et al., 2003). A figura 3.5 exemplifica a utilização de um *rendezvous peer*.

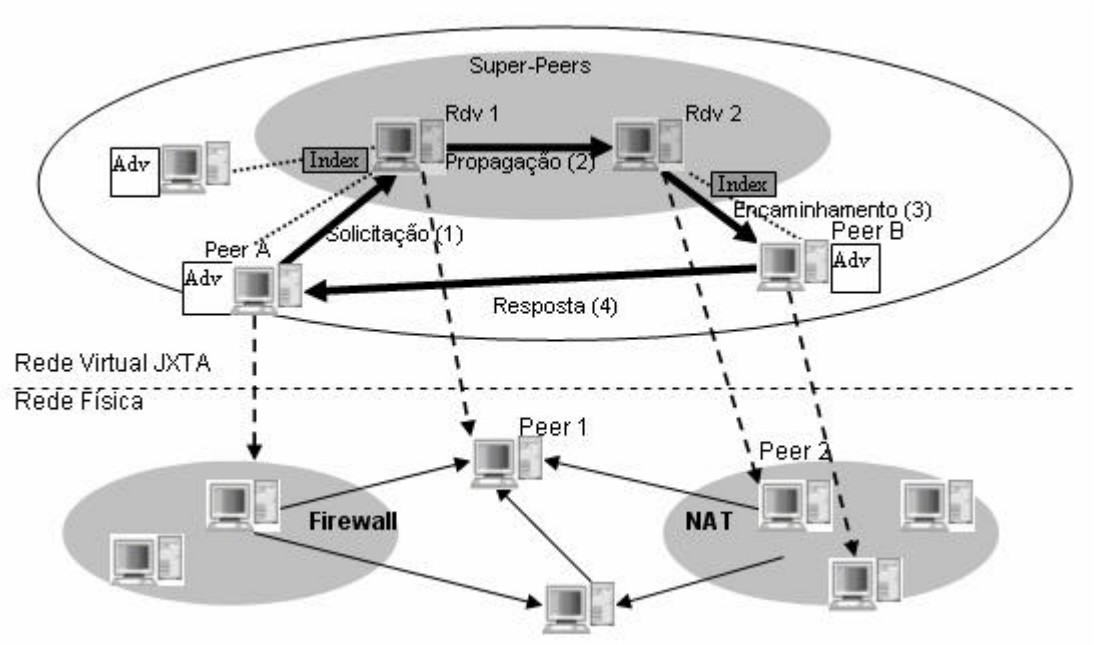

Figura 3.5: R*endezvous Peer* (TRAVERSAT et al., 2003)

3.2 Protocolos JXTA

JXTA define diversos protocolos para a comunicação entre os *peers.* Esses protocolos são definidos por mensagens no formato XML trocadas entre os *peers* participantes da rede. Os *peers* utilizam esses protocolos para descobrir uns aos outros, divulgar e descobrir recursos, realizar o roteamento de mensagens e possibilitar a comunicação entre eles. Atualmente estão definidos seis protocolos com suas funcionalidades explicadas a seguir (SUN MICROSYSTEMS, 2003):

- *Peer Discovery Protocol* (PDP) - como o nome sugere, o PDP permite que um *peer* descubra outros *peers* na rede. Mas o protocolo não se restringe à descoberta de *peers*, pode ser utilizado também para descobrir qualquer recurso relacionado a um *peer*, incluindo *peer groups*, *pipes*, serviços e conteúdo. Cada um dos recursos associados ao *peer* é descrito e publicado por um *advertisement*. Os *peers* geram mensagens de busca para descobrir *advertisements* em um *peer group*, e essas podem ser enviadas a todos os

peers de um local ou a um *rendezvous peer*. Esse *peer* pode receber zero, uma, ou mais respostas a um pedido de descoberta, e essa resposta retorna um ou mais *advertisements*.

- *Peer Information Protocol* (PIP) - utilizado pelos *peers* para obter informações sobre o estado de outros *peers*. Uma vez que um *peer* é localizado, seu estado e suas capacidades podem ser consultados, através de um conjunto de mensagens fornecido pelo protocolo. O PIP envia uma mensagem, chamada *ping message*, para um *peer*, para checar se ele está ativo e para coletar informações a seu respeito. Essa mensagem especifica se deve ser retornada uma resposta completa, contendo o *peer advertisement* do *peer* consultado ou se deve ser retornado um simples aviso de que o *peer* está ativo. A mensagem utilizada para responder uma *ping message* é a *peerinfo message*, que contém a credencial do remetente, o *peer* ID da fonte e do destino, o instante em que o *peer* foi ativado e o *peer advertisement*.
- *Peer Resolver Protocol* (PRP) - permite que os *peers* enviem consultas genéricas a um ou mais *peers* e identifiquem as respostas a essas consultas. As consultas podem ser enviadas a um *peer* especifico ou propagadas por meio de um serviço *rendezvous* para um *peer group*. PRP utiliza o serviço *rendezvous* para disseminar a consulta para múltiplos *peers*, e utiliza mensagens *unicast* para enviar consultas para *peers* específicos. Ao contrário de PDP e PIP, que são usados para consultar informações específicas, esse protocolo permite a definição e troca de informações arbitrárias.
- *Pipe Binding Protocol* (PBP) - utilizado por *peers* para estabelecer um canal de comunicação virtual, ou *pipe*, entre um ou mais *peers*. O link virtual do *pipe* pode ser estabelecido sobre qualquer número de canais de transporte físico. Cada extremo do *pipe* trabalha para manter o link virtual e re-estabelecer-lo, se necessário.
- *Endpoint Routing Protocol* (ERP) - utilizado pelos *peers* para encontrar rotas para portas de destino em outros *peers*. O *Endpoint Routing Protocol* (ERP) define um conjunto de mensagens de requisição/consulta usadas para obter informações de roteamento. Quando um *peer* é solicitado para enviar uma mensagem para um dado endereço de rede associado a outro *peer*, ele procura em sua *cache* local para determinar se possui uma rota para esse *peer*, e se não encontrar essa rota envia uma requisição de resolução de rota para os *relays peers* disponíveis pedindo por informação de roteamento. Quando um *relay peer* recebe uma requisição de rota, verifica se a conhece. Caso positivo, retorna essa informação ao *peer*.
- *Rendezvous Protocol* (RVP) - mecanismo pelo qual os *peers* podem inscrever-se para um serviço de propagação, utilizado pelo PRP e pelo PBP para propagar mensagens. Em um *peer group*, os *peers* podem ser *rendezvous peers* ou *peers* que estão "escutando" *rendezvous peers*. O RVP permite que um *peer* envie mensagens a todos "ouvintes" de um determinado serviço.

Os *peers* JXTA não precisam implementar todos os seis protocolos, eles apenas implementam os que serão utilizados. A Figura 3.6 mostra quais são os protocolos obrigatórios e quais os aleatórios. Esses protocolos são a base da interoperabilidade

JXTA. Quaisquer plataformas que suportem um transporte padrão, como TCP/IP, e tenham a habilidade de transmitir e receber mensagens XML sobre esse transporte podem interagir. Isso significa que clientes e serviços JXTA podem ser criados em quase todos os sistemas operacionais atuais, utilizando praticamente qualquer linguagem de programação (LI, 2001).

Todos protocolos JXTA são assíncronos, e baseados em um modelo de consultas e respostas. Os *peers* utilizam esses protocolos para comunicar-se com os participantes de um *peer group*. Por exemplo, um *peer* pode usar PDP para enviar uma requisição de descoberta para todos os *peers* do grupo padrão (*Net Peer Group*). Nesse caso, múltiplos *peers* enviarão resposta a essa consulta. Em outro exemplo, um *peer* pode enviar uma requisição de descoberta por um *pipe* específico, nomeado "pipeex". Nesse caso, o *peer* só receberá resposta se o *pipe* for encontrado (SUN MICROSYSTEMS, 2003).

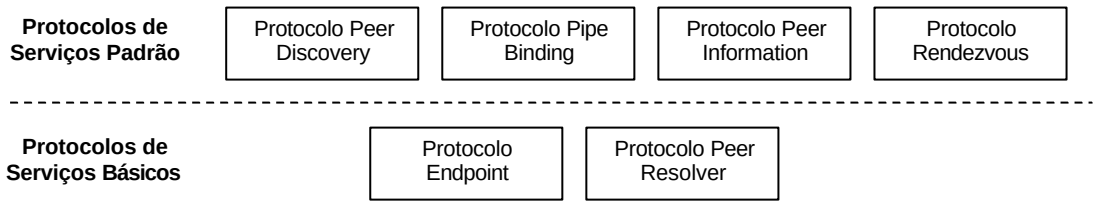

Figura 3.6: Protocolos definidos na tecnologia JXTA (TRAVERSAT et al., 2003)

4 ARQUITETURA PROPOSTA

Neste capítulo é apresentada a arquitetura proposta para o ambiente de gerenciamento distribuído e cooperativo implementado. Na modelagem dessa arquitetura foram definidos os principais componentes do ambiente, e como é feita sua interface com a plataforma JXTA e com os usuários.

A proposta da arquitetura estende o ambiente CMSPro (*Cooperative Management Service Provider*) (SPELLMEIER, 2003), que é uma aplicação P2P que oferece um serviço de busca e armazenamento de informações sobre configurações de equipamentos em uma rede.

4.1 Rede de Gerenciamento

A ferramenta de gerenciamento forma, como pode ser observado na figura 4.1, uma rede formada pelos *peers*, os quais funcionam tanto como cliente, quanto como servidor. Considera-se que cada *peer* é responsável por um domínio administrativo, e através desse o administrador gerencia os dispositivos que fazem parte do domínio. Pode ocorrer também de um *peer* ser responsável por mais de um domínio, ou um domínio possuir mais de um *peer* responsável.

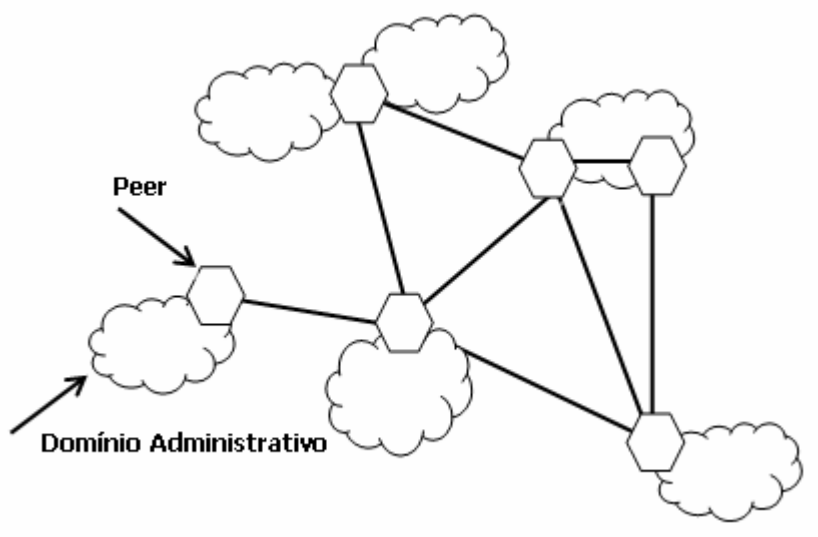

Figura 4.1: Rede P2P de gerenciamento

Através dos *peers* os administradores podem, além de gerenciar seus domínios, interagir entre si. Eles utilizam os serviços de gerenciamento oferecidos pela rede para

compartilhar recursos e informação de gerenciamento. A seguir estão explicados os serviços de gerenciamento oferecidos pela aplicação, a estrutura interna dos *peers* e as mensagens trocadas entre eles.

4.2 Serviços de Gerenciamento

A aplicação oferece quatro serviços: compartilhamento de arquivos de configuração de dispositivos, compartilhamento de registros de dispositivos, configuração de dispositivos e solicitação de reserva de banda. Cada nodo da rede é capaz de interagir com os outros através desses serviços, de maneira que cada um funcione como cliente, permitindo ao usuário buscar informações e requisitar operações, e como servidor, já que oferece à rede um serviço que responde a essas buscas e requisições.

O compartilhamento de arquivos de configuração de dispositivos inclui as funcionalidades de armazenamento, disponibilização e consulta de arquivos de configuração de equipamentos de comunicação de dados. Esse serviço permite o cadastramento e a disponibilização na rede de registros de configurações dos equipamentos gerenciados, como roteadores, multiplexadores, switches, entre outros. Os *peers* poderão consultar os registros de configurações armazenados pelos outros *peers* da rede e fazer *download* dos arquivos escolhidos para serem utilizados nos dispositivos do domínio local.

Já o compartilhamento de registros de dispositivos permite que cada *peer* disponibilize para os outros uma relação dos dispositivos pelos quais é responsável, para que possam ter conhecimento dos recursos disponíveis em cada um dos domínios administrativos que fazem parte da rede virtual de gerenciamento.

Através de serviço de configuração de dispositivos, os *peers* são capazes de carregar as configurações de um determinado arquivo de configuração em um determinado dispositivo. Esse serviço cria uma interface entre o administrador e os dispositivos pelos quais ele é responsável, permitindo que a configuração desses seja feita de forma automática.

O último serviço permite que seja feita reserva de banda em um determinado conjunto de roteadores, para, por exemplo, possibilitar a realização de vídeo conferências. O *peer* que desejar agendar uma reserva de banda deve solicitar aos *peers* responsáveis por cada um dos roteadores envolvidos informações sobre a disponibilidade desses dispositivos no instante desejado, e após receber a resposta de todos os consultados define se a reserva será efetuada ou cancelada.

4.3 Estrutura Interna dos *Peers*

Cada um dos *peers* pode funcionar tanto como cliente, quanto como servidor, e é capaz de executar todos os serviços de gerenciamento disponíveis para a aplicação. A figura 4.2 mostra a estrutura interna do *peer*, e como é feita a interface com seu administrador e com a rede. Como pode ser visto na figura, a aplicação foi dividia em oito módulos distintos, representando os serviços disponibilizados, o cliente, o servidor, a base de armazenamento de dados e a plataforma JXTA. Na seqüência cada um desses módulos será descrito.

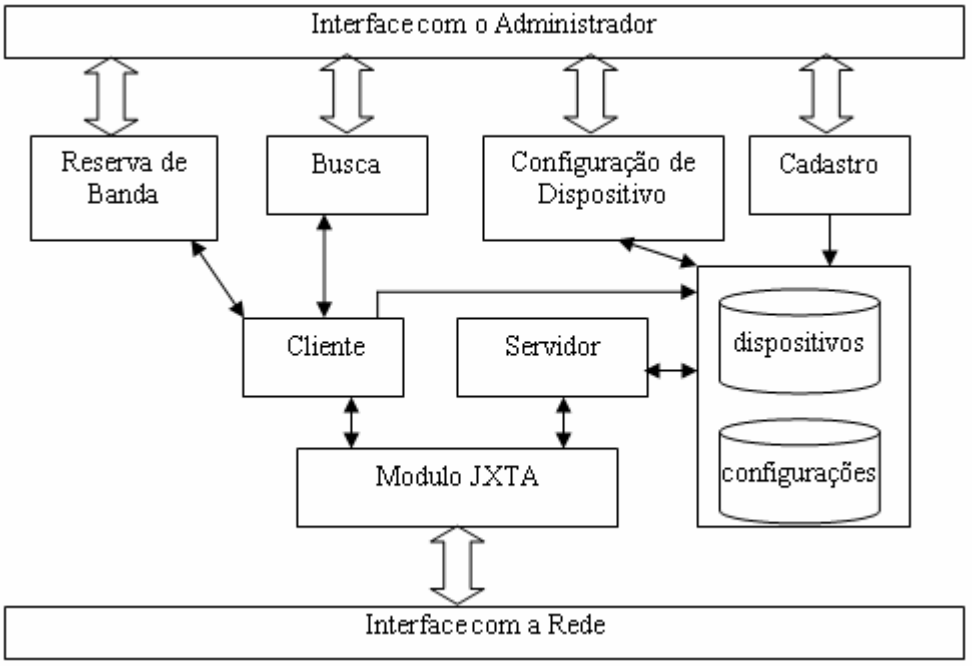

Figura 4.2: Arquitetura do *peer*

4.3.1 Cadastro

O módulo de cadastro é responsável por oferecer ao usuário a possibilidade de cadastrar novos registros de configurações e de dispositivos em sua base de dados local. Como pode ser visto na figura 4.3, através da interface com o usuário, o módulo recebe os parâmetros do novo registro a ser criado, e armazena esse registro no Banco de Dados de dispositivos e Configurações.

Os parâmetros para registro de uma configuração são o tipo do dispositivo ao qual a configuração se aplica, o fabricante, o modelo, o sistema operacional e uma descrição desse dispositivo, além de uma referência para o arquivo de configuração. E os parâmetros para registro de dispositivos são o tipo, o fabricante, o modelo, uma descrição e o endereço IP desse dispositivo.

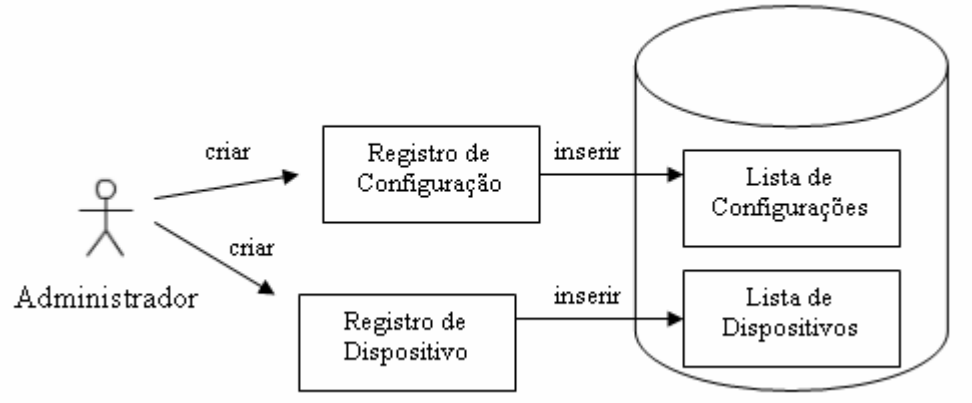

Figura 4.3: Cadastro

4.3.2 Banco de Dados de Dispositivos e Configurações

Todas as informações cadastradas pelo usuário, assim como os registros obtidos como resultado de buscas, são armazenados e controlados através desse módulo. Os registros são cadastrados em uma estrutura que contém as informações básicas necessárias para sua identificação, além de referências aos arquivos de configuração ou dispositivos relacionados aos mesmos. As estruturas de dados que formam o banco estão representadas no diagrama de classes da figura 4.4.

Esse banco de dados fica disponível para ser consultado pelo módulo Servidor local do *peer*, para que o mesmo possa responder às requisições de outros *peers* que façam parte da rede de gerenciamento. O módulo também fica disponível para que o Cliente cadastre registros de configuração originados de *downloads* de arquivos de configuração de outros *peers*.

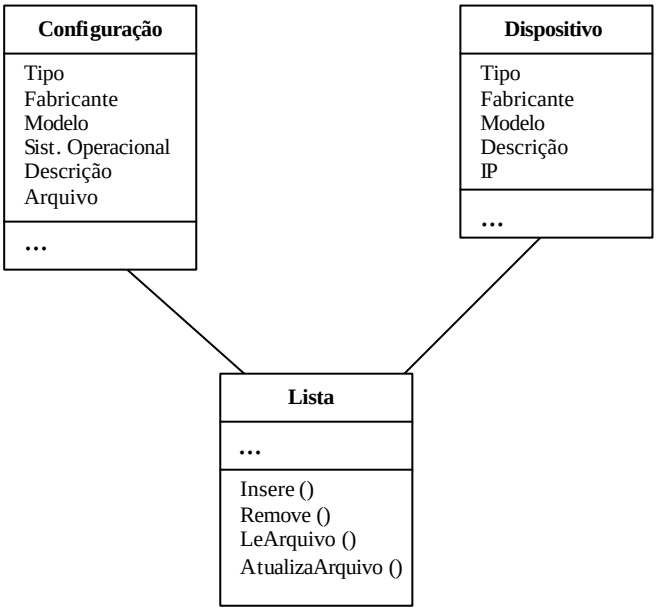

Figura 4.4: Diagrama de Classes da Base de Dados

4.3.3 Busca

O módulo de busca fornece aos usuários definições de buscas por arquivos de configuração ou dispositivos. Esse módulo comunica-se com a interface do usuário, de onde obtém os parâmetros da busca, e então, envia os dados ao módulo cliente, para que este possa executar a busca nos outros *peers*.

O módulo recebe do Cliente os resultados das buscas realizadas e os exibe para o usuário, e possibilita que, após uma busca por arquivos de configuração, o usuário faça uma requisição de *download* dos arquivos que desejar. As mensagens trocadas em uma busca podem ser vistas no diagrama da figura 4.5.

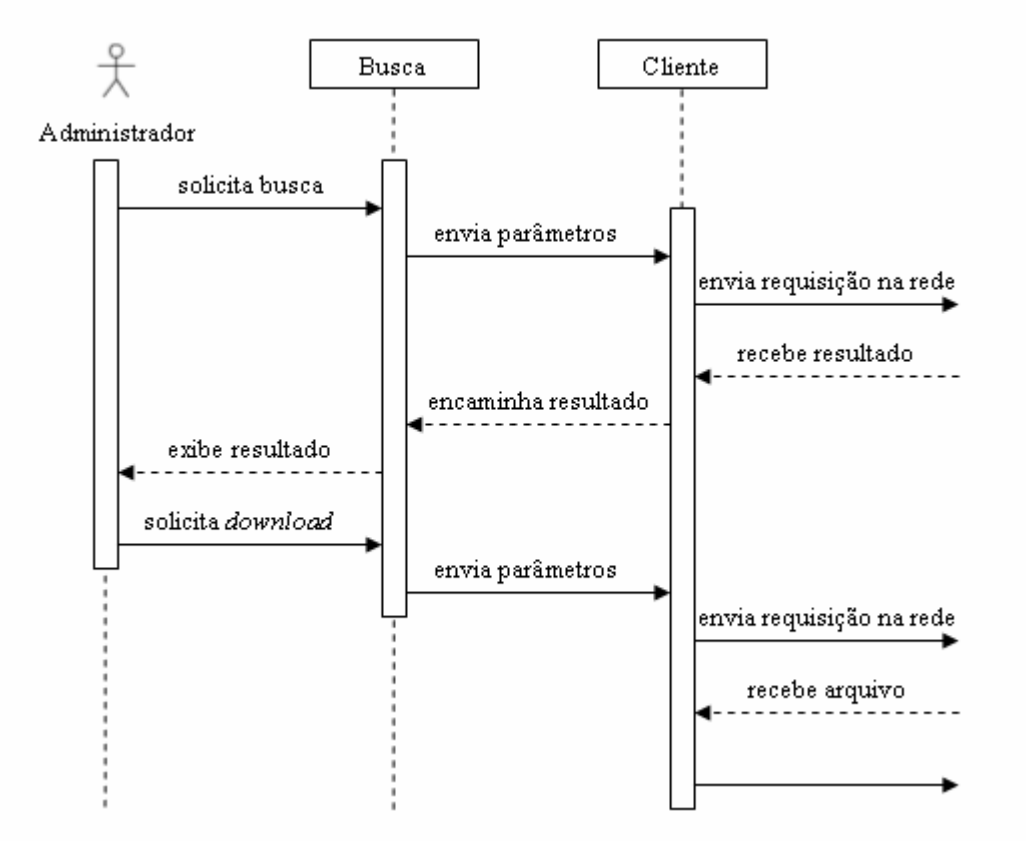

Figura 4.5: Mensagens trocadas em uma busca

4.3.4 Configuração de Dispositivo

É o módulo responsável por fazer com que um determinado dispositivo gerenciado pelo *peer* utilize as configurações contidas no arquivo a ser aplicado. O módulo fará a interface entre a aplicação e o dispositivo gerenciado, para que o arquivo de configuração possa ser transferido e carregado pelo dispositivo.

A partir da definição do dispositivo a ser configurado e do arquivo a ser aplicado, o módulo deve fazer a transferência do arquivo para o dispositivo e logo após fazer com que esse dispositivo carregue as configurações contidas no arquivo.

Essa abordagem de configuração funciona para dispositivos que utilizam arquivos de configuração. Para abranger dispositivos que utilizam protocolos de configuração como SNMP (*Simple Network Management Protocol*) (HARRINGTON; PRESUHN; WIJNEN, 2002), CLI (*Command Line Interface*) ou HTTP, a arquitetura pode ser estendida, de forma a obter-se uma abstração das particularidades de acesso aos equipamentos.

4.3.5 Reserva de Banda

Quando um administrador precisa reservar banda para transmitir dados em um conjunto de roteadores gerenciados pela rede P2P, ele deve usar este módulo. Através da Interface com o Administrador os dados necessários para reserva (IPs dos dispositivos envolvidos, data e hora de inicio, e duração da reserva) são passados para o módulo, o qual passa esses dados para o módulo Cliente, para que esse, por sua vez,

encontre os *peers* responsáveis por cada dispositivo envolvido na reserva e envie a eles uma requisição de reserva.

Quando a requisição de reserva é recebida pelo módulo Servidor dos *peers* remotos, eles devem consultar os dispositivos e informar ao *peer* requisitante sobre disponibilidade para reserva no intervalo determinado.

Como a solicitação de reserva é encaminhada a todos os *peers* ao mesmo tempo, os *peers* consultam os roteadores em seus domínios em paralelo, ao contrário do que acontece em uma reserva utilizando RSVP (*Resource Reservation Protocol*) (BRADEN et al., 1997), onde a consulta aos roteadores é feita de forma seqüencial, partindo da máquina que deve receber os dados e percorrendo a rota reversa, até a origem dos dados. A figura 4.6 ilustra essa diferença entre as abordagens para reserva.

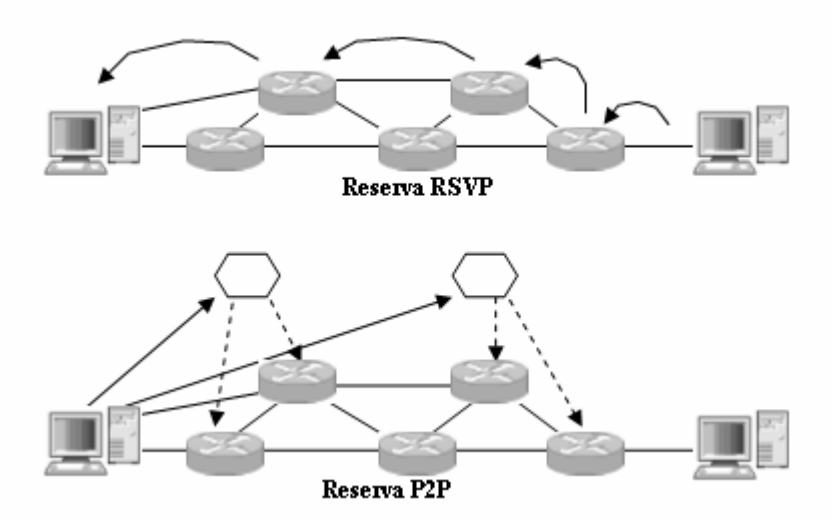

Figura 4.6: Reserva de Banda

4.3.6 Cliente

O módulo cliente atende às requisições dos módulos locais e, através da Plataforma JXTA, envia para os outros *peers* da rede essas requisições. Ele deve encontrar os *peers* que ofereçam o serviço desejado e estabelecer um canal de comunicação para troca de mensagens com os mesmos.

É também sua responsabilidade aguardar as respostas das requisições, coordenar a entrega das respostas aos respectivos módulos e registrar, na base de dados, os registros de configuração originados de *downloads* de arquivos de outros de *peers*.

4.3.7 Servidor

O servidor é responsável por publicar os serviços disponíveis pela aplicação, divulgar um canal de comunicação que permita que outros *peers* enviem mensagens para o mesmo e aguardar as requisições de clientes remotos.

No caso uma requisição de busca por dispositivos ou arquivos de configuração, o módulo deve acessar as informações contidas no Banco de Dados de Dispositivos e Configurações para enviá-las ao cliente remoto, e quando necessário, enviar também o arquivo de configuração solicitado. Esse módulo também é responsável por oferecer

ferramentas de pesquisa local nos registros para, no caso de buscas por arquivos de configuração, retornar ao Cliente remoto apenas os registros que satisfaçam as definições da busca.

4.3.8 Módulo JXTA

O módulo é utilizado pelos módulos cliente e servidor para realizar a comunicação com a rede, através dos protocolos *Peer Discovery Protocol* (PDP) e *Pipe Binding Protocol* (PBP), apresentados no capítulo 3. O PDP é utilizado pelo cliente para encontrar na rede *peers* que ofereçam o serviço de busca definido pela aplicação, e pelo servidor para divulgar os serviços que disponibiliza (através de *advertisements*).

Ao encontrar um serviço na rede, o módulo cliente recebe as informações necessárias para criar um canal de comunicação entre ele e o serviço encontrado, através do PBP. É através desse canal que são trocadas informações entre o cliente local e o servidor remoto.

4.4 Mensagens

A comunicação na rede de gerenciamento é feita através de mensagens trocadas entre os módulos Cliente e Servidor dos *peers* envolvidos. Essas mensagens são utilizadas pelo módulo Cliente para fazer as solicitações de serviços e passar parâmetros ao *peer* remoto, e pelo modulo do Servidor do *peer* remoto para responder às requisições e enviar arquivos.

Na tabela 4.1 estão apresentadas as mensagens podem ser trocadas entre o Cliente e o Servidor de *peers* distintos.

Tabela 4.1: Troca de Mensagens entre Cliente e Servidor

Mensagem	Direção	Conteúdo
Busca	Cliente → Servidor	Parâmetros da busca e identificação do canal de comunicação (endereço do cliente).
Resposta da Busca	Servidor → Cliente	Resultados da pesquisa na forma de registros (de configuração ou de dispositivos) presentes no servidor.
Requisição de arquivo	Cliente → Servidor	Nome do arquivo desejado e identificação do canal de comunicação
Envio de arquivo	Servidor → Cliente	Arquivo requisitado pelo cliente.
Reserva de Banda	Cliente → Servidor	Parâmetros da reserva e identificação do canal de comunicação
Resposta da Reserva	Servidor → Cliente	Disponibilidade do dispositivo

Este capítulo apresentou a modelagem da arquitetura do ambiente de gerenciamento proposto, seus aspectos e características. No próximo capítulo é descrito como foi feita a implementação de uma aplicação baseada nessa arquitetura.

5 IMPLEMENTAÇÃO

A partir da arquitetura apresentada foi implementada uma aplicação para a realização de testes e validação da proposta. A aplicação foi escrita inteiramente em Java, utilizando a plataforma JXTA e neste capítulo estão descritos os principais aspectos relacionados a essa implementação. A figura 5.1 apresenta a arquitetura proposta no capítulo anterior, contendo referências às classes implementadas, as quais serão descritas nos itens a seguir.

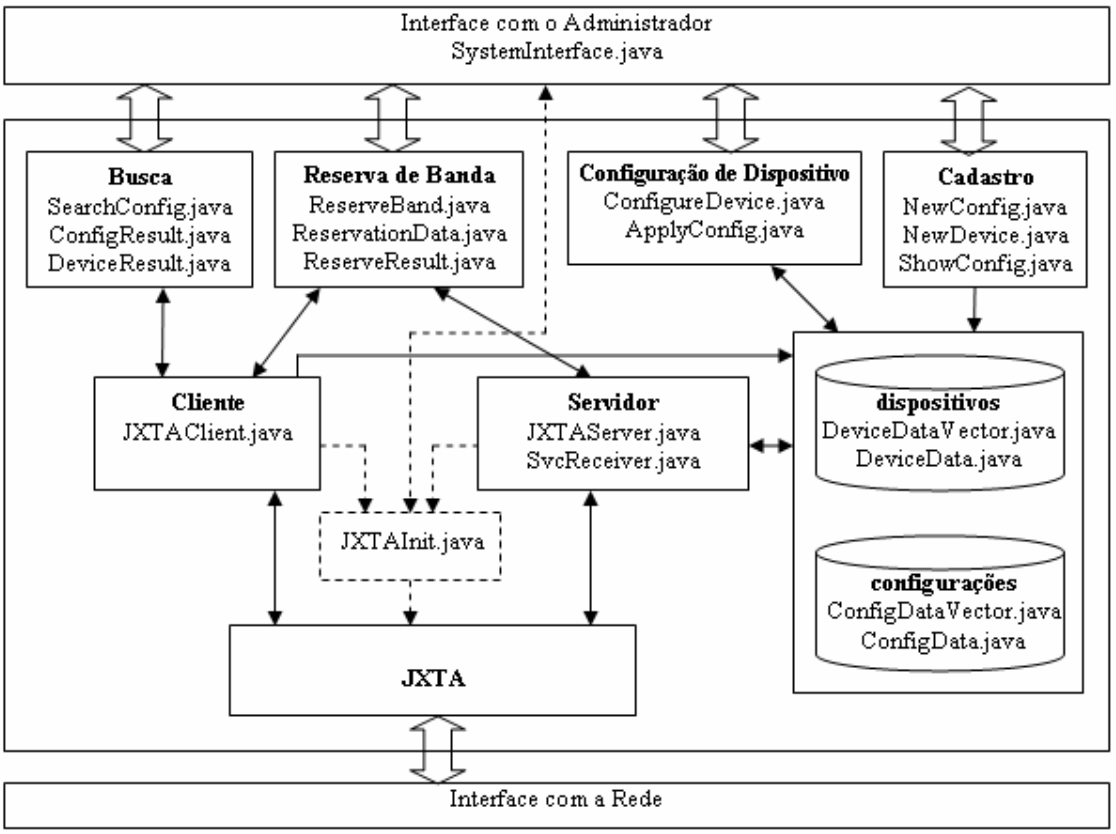

Figura 5.1: Arquitetura de Classes

5.1 Interface com o Administrador

A interface com o administrador é feita pela classe principal da implementação, *SystemInterface.java*, e também pelos módulos de Cadastro, Busca, Configuração de Dispositivo e Reserva de Banda. As telas da interface foram implementadas com a utilização dos pacotes AWT e SWING, disponibilizados no ambiente de desenvolvimento JAVA (JDK).

A classe principal é responsável por iniciar a aplicação, carregar um arquivo com as informações de configuração do sistema, carregar os registros com os cadastros feitos pelo usuário e disponibilizar acesso aos serviços da aplicação, através da tela principal do sistema, ilustrada na figura 5.2. Essa classe também mantém uma instância da classe *JXTAInit.java*, responsável pela inicialização da plataforma JXTA.

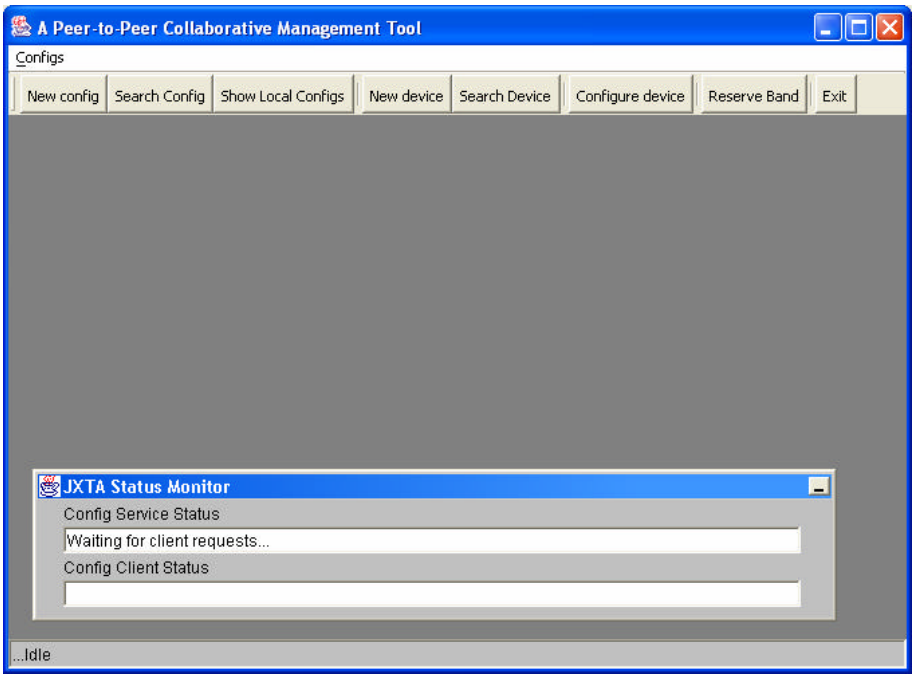

Figura 5.2: Interface Principal

A classe *NewConfig.java*, pertencente ao módulo de cadastro, é responsável pela exibição da interface para inserção de novos registros de configuração, ilustrada na figura 5.3. A interface para cadastro de novos dispositivos, figura 5.4, é feita pela classe *NewDevice.java* também pertencente ao módulo de cadastro. Com os dados preenchidos nos campos de cada interface é criado um registro, o qual é inserido na base de dados de dispositivos e configurações. A classe *NewDevice.java* também é responsável por enviar ao servidor um pedido de publicação dos novos dispositivos registrados. Outra classe que faz parte do módulo de cadastro é *ShowConfig.java*, que exibe uma relação com as configurações armazenadas localmente no *peer*.

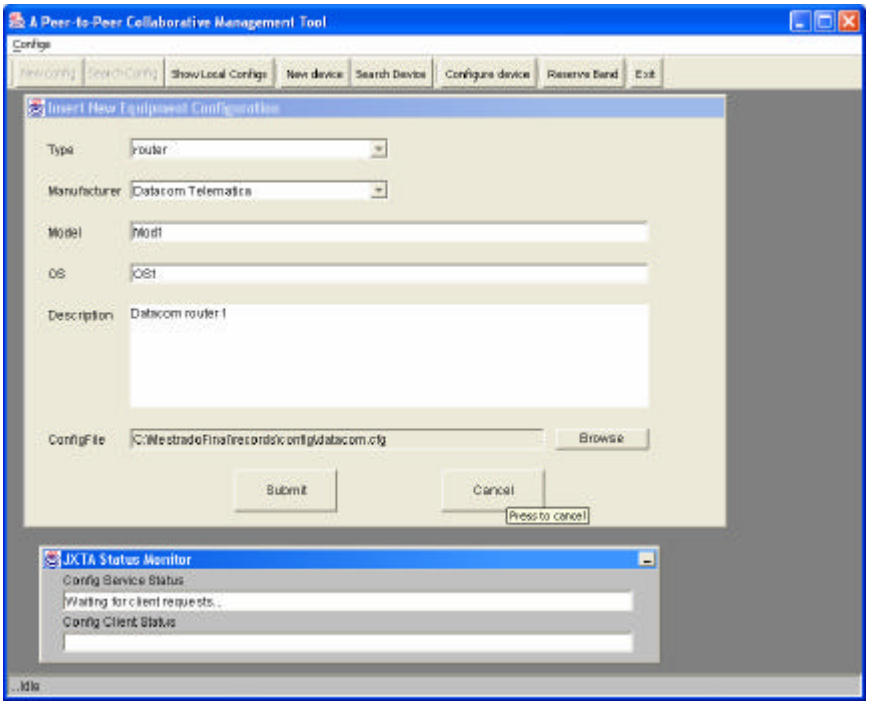

Figura 5.3: Interface para Cadastro de Configuração

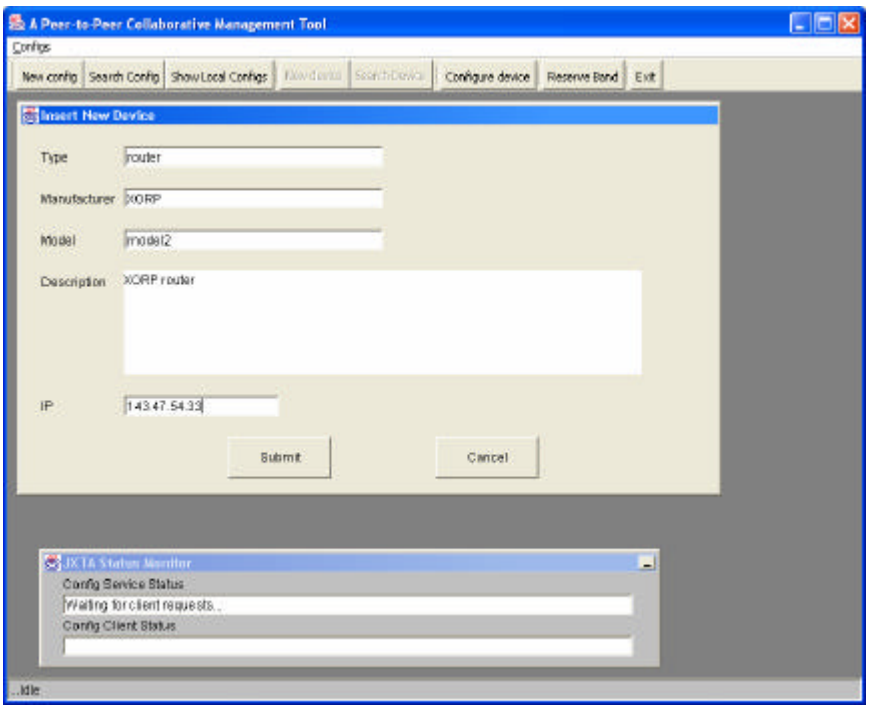

Figura 5.4: Interface para Cadastro de Dispositivo

A classe *SearchConfig.java* implementa a interface para a definição dos parâmetros de busca de configurações, ilustrada na figuras 5.5. Após o usuário solicitar uma busca a classe verifica os parâmetros definidos e cria um registro de configuração para ser utilizado nessa busca. Então é criada uma instância da classe *JXTAClient.java*, que vai iniciar a busca na rede pelas configurações que satisfaçam os parâmetros definidos.

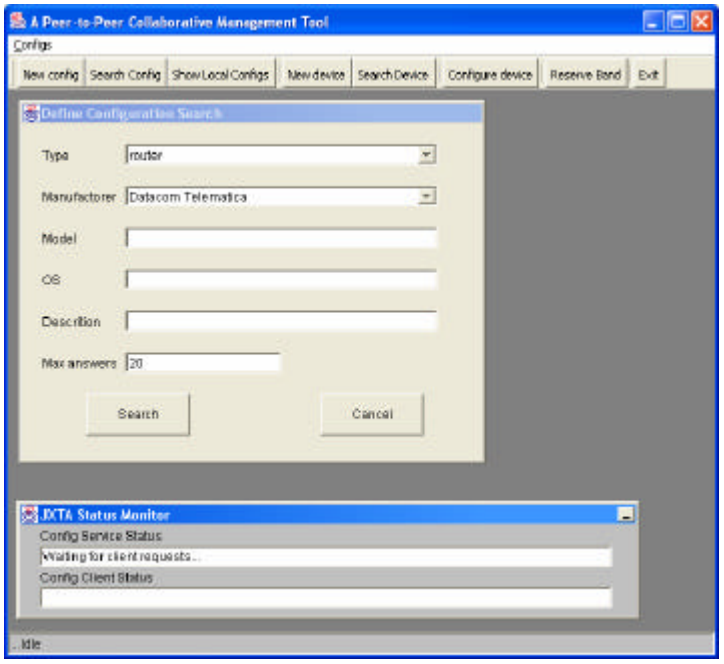

Figura 5.5: Interface para Busca de Configurações

O resultado da busca por configurações é exibido pela classe *ConfigResult.java*, através da tela ilustrada na figura 5.6. A medida que o cliente repassa os resultados obtidos, através da função *addResult*, eles são exibidos na interface. É a partir dessa tela que o administrador pode fazer uma requisição de *download* de um arquivo de configuração, selecionado uma ou mais das configurações exibidas e acionando o botão "Download". Essa requisição de download é repassada ao módulo cliente, que a executa. A classe *DeviceResult.java* é responsável pela exibição do resultado de buscas por dispositivos, como mostrado na figura 5.7.

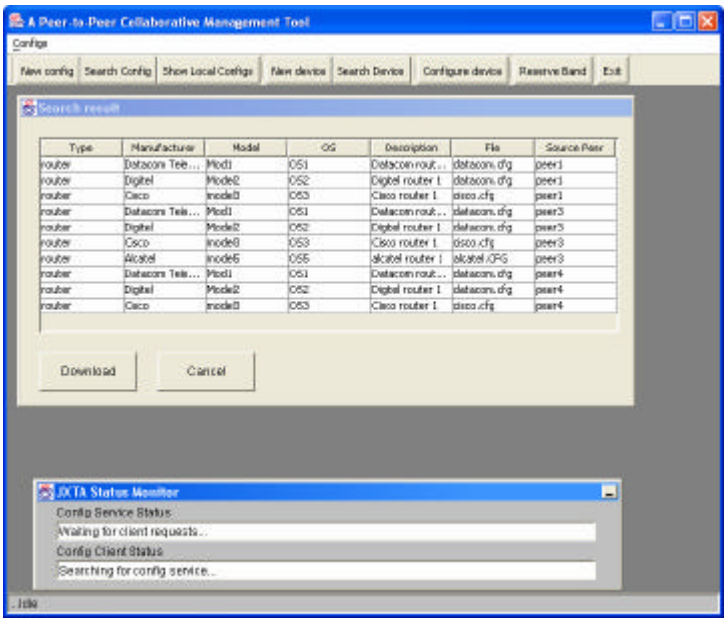

Figura 5.6: Interface para Resultado de Busca por Configurações

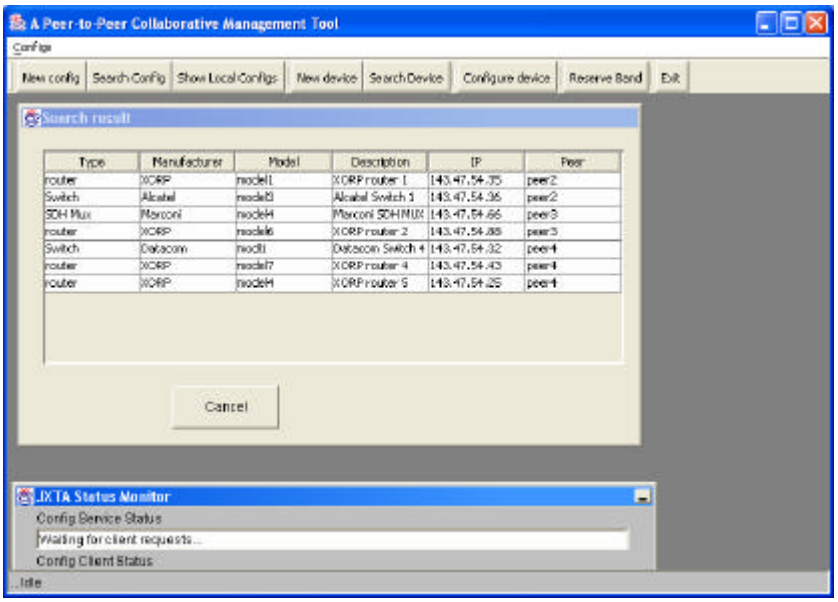

Figura 5.7: Interface para Resultado de Busca por Dispositivos

A interface para o serviço de configuração de dispositivos é feita pela classe *ConfigureDevice.java.* Como pode ser visto na figura 5.8, essa classe exibe uma lista com os dispositivos gerenciados pelo *peer*. Ao selecionar um dispositivo e acionar o botão "Configure" o administrador poderá selecionar o arquivo de configuração que será aplicado ao dispositivo, e esses serão passados como parâmetros para a classe *ApplyConfig.java,* que fará a configuração do dispositivo.

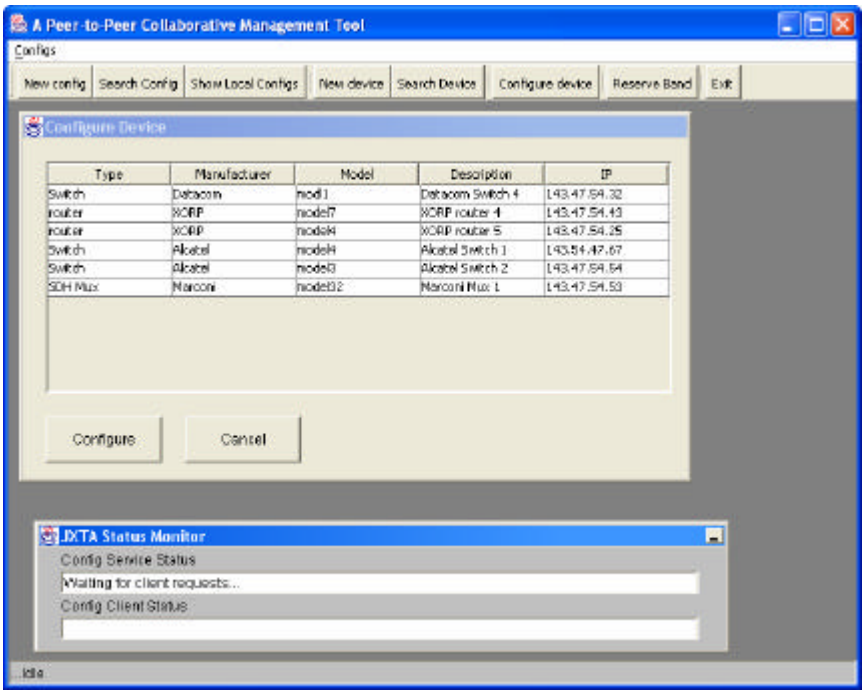

Figura 5.8: Interface para Configuração de Dispositivos

Para realizar uma reserva de banda o administrador utiliza a interface fornecida pela classe *ReserveBand.java*, ilustrada na figura 5.9. Ele fornece a data e hora de início, a duração e os endereços IPs dos roteadores envolvidos. Esses parâmetros são passados para o módulo cliente para que consulte os *peers* responsáveis por cada roteador.

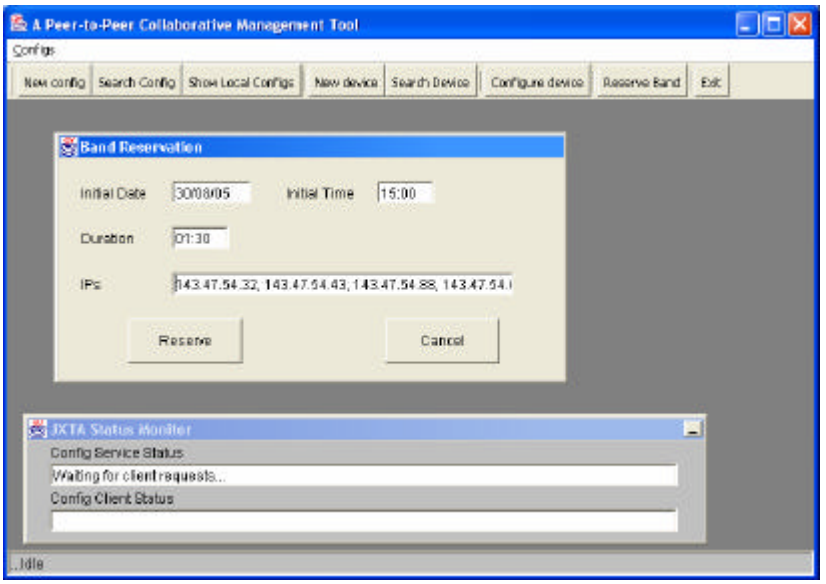

Figura 5.9: Interface para Reserva de Banda

O resultado da reserva será exibido pela classe *ReserveResult.java*, como pode ser visto na figura 5.10. À medida que o cliente for descobrindo os *peers* responsáveis por cada dispositivo e recebendo a resposta sobre sua disponibilidade ele atualiza a listagem com o resultado da solicitação referente ao dispositivo. Quando concluída a solicitação de reserva para todos os dispositivos envolvidos o cliente informa se a reserva foi realizada ou cancelada.

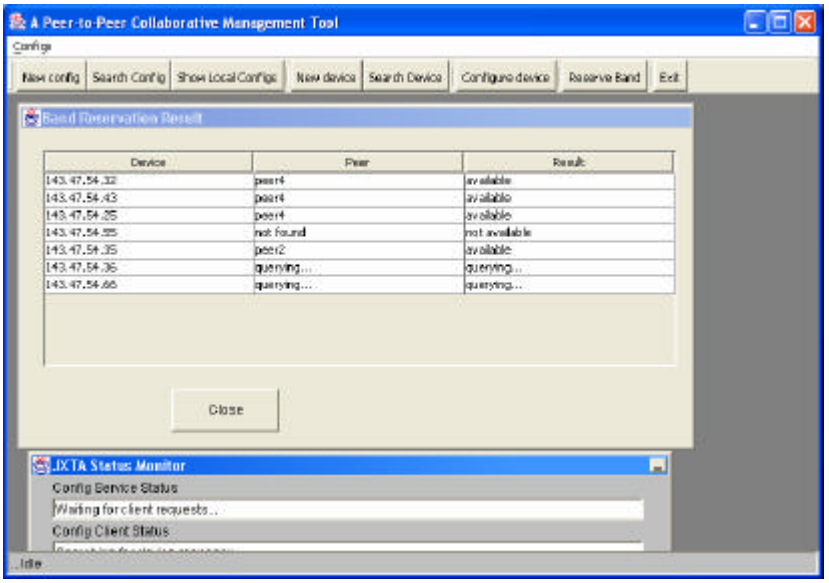

Figura 5.10: Interface para Resultado de Reserva de Banda

5.2 Configuração de Dispositivo

A configuração de um dispositivo a partir de um arquivo de configuração é feita pela classe *ApplyConfig.java*, após receber da classe *ConfigureDevice.java* o endereço IP do dispositivo e a referência para o arquivo de configuração. O sistema comunica-se com o dispositivo através de uma conexão SSH. Para implementar as funcionalidades de um cliente SSH foi utilizado o pacote "j2ssh" (SOURCEFORGE, 2005). O primeiro passo dessa comunicação é estabelecer a conexão com o dispositivo, da forma que mostra o trecho de código abaixo:

```
SshClient ssh = new SshClient();
ssh.connect(<ip_do_dispositivo>);
PasswordAuthenticationClient pwd = new PasswordAuthenticationClient();
pwd.setUsername("<usuário>");
System.out.print("Password: ");
String password = reader.readLine();
pwd.setPassword(<senha>);
int result = ssh.authenticate(pwd);
```

Para estabelecer a conexão é necessário criar um cliente SSH, instância da classe *SshClient*, e utilizar a função *connect*. A autenticação no dispositivo remoto é feita por uma instância da classe *PasswordAuthenticationClient*, que recebe a identificação e senha do usuário. A senha é passada pelo administrador através de um terminal da aplicação.

Depois de realizada a autenticação, o sistema cria uma seção e inicia um terminal para manipular o dispositivo remoto por linha de comando. O arquivo de configuração é transferido para o dispositivo através de FTP e então carregado pelo dispositivo. O código utilizado para criar a seção, o terminal e passar as linhas de comandos é como o seguinte:

```
SessionChannelClient session = ssh.openSessionChannel();
if(!session.requestPseudoTerminal("vt100", 80, 24, 0, 0, ""))
    System.out.println("Failed to allocate a pseudo terminal");
if (session.startShell())
{
    ...
    OutputStream out = session.getOutputStream();
    String cmd = "<comando>";
    out.write(cmd.getBytes());
    ...
}
```

A implementação dessa funcionalidade foi feita de forma a ser compatível com um roteador XORP (HANDLEY; HODSON; KOHLER, 2002) e pode ser facilmente estendida a outros tipos roteadores que utilizam arquivo de configuração. XORP é um roteador totalmente implementado em software, e está disponível em uma versão que pode ser carregada em CD. Pela falta de roteadores disponíveis para testes e trocas de configuração e também de computadores disponíveis para a instalação de um roteador, XORP foi escolhido para a realização dos testes, pela possibilidade de ser carregado em CD, podendo assim ser utilizado em qualquer computador sem precisar de instalação.

Outra abordagem para a configuração de dispositivos poderia ser a utilização de SNMP, porém, não foi utilizada devido a não disponibilidade desse serviço na versão do roteador XORP em CD.

5.3 Banco de Dados de Dispositivos e Configurações

O banco de dados de dispositivos e configurações é implementado pelas classes *DeviceData.java, ConfigData.java, DeviceDataVector.java* e *ConfigDataVector.java.* A primeira representa a estrutura básica do registro de um dispositivo e armazena os dados do dispositivo registrados pelo administrador, e a segunda representa a estrutura básica do registro de uma configuração, que armazena, além das informações sobre a configuração, uma referência ao arquivo de configuração propriamente dito.

Os registros de dispositivos e configurações cadastrados pelo usuário, ou, no caso de configurações, resultados de uma busca na rede, são armazenados em vetores de registros. O vetor de registros de dispositivos é implementado pela classe e *DeviceDataVector.java* e o vetor de registros de configurações pela classe *ConfigDataVector.java.* As classes que implementam os vetores oferecem funcionalidades para inserção e exclusão dos registros e também mantêm arquivos com o conteúdo dos registros, um para os dispositivos e um para as configurações, armazenados no disco do usuário. Esses arquivos são utilizados para carregar os dados dos vetores na inicialização do sistema.

5.4 Cliente

O módulo cliente é representado pela classe *JXTAClient.java.* Essa classe implementa uma *thread* que é iniciada toda vez que o usuário inicia uma busca ou uma reserva de banda. Para encontrar os *peers* que contêm arquivos de configuração ou que gerenciam os dispositivos envolvidos em uma reserva o cliente utiliza os serviços de descoberta da plataforma JXTA, mais especificamente a classe *DiscoveryService*, que localiza esses *peers* através dos *advertisements* publicados por eles. O trecho de código abaixo mostra como as funções *getRemoteAdvertisements* e *getLocalAdvertisements* podem ser utilizadas para descobrir *peers*.

```
private DiscoveryService discSvc;
...
discSvc.getRemoteAdvertisements(<peer> , <tipo> , <atributo>,<valor>,
                                                  <thershold>, <retorno>);

Enumeration  enum = discSvc.getLocalAdvertisements(<tipo>, <atributo>,
                                                              <valor>);
```

A função *getRemoteAdvertisements* é utilizada para procurar *advertisements* na rede, para isso faz uma consulta aos *redezenvous peers*. O *advertisements* resultados dessa busca são armazenados localmente no *peer*. E a função *getLocalAdvertisements* faz uma busca no próprio *peer*, para localizar *advertisements* cadastrados localmente. Os parâmetros dessas funções são:

- *peer* - especifica o *peer* onde a busca será feita. Se o valor for null a busca é feita em todo *peer group*;

- tipo - especifica o tipo de busca. Pode ser uma busca por *peers* (PEER), *peer groups* (GROUP) ou *advertisements* (ADV);
- atributo - identifica o atributo que será utilizado como parâmetro na busca;
- valor - determina o valor que o atributo especificado deve conter para satisfazer a condição de busca;
- *thershold* - limite máximo de respostas e
- retorno – variável onde serão armazenados os *advertisements* encontrados.

Ao localizar um *advertisement* o cliente recebe o *pipe advertisement* do *peer* remoto, permitindo que seja criado um canal de comunicação entre os *peers*, por onde serão enviadas as mensagens do cliente para o servidor do *peer* remoto. O canal é criado pela função *createOutputPipe*, da classe *PipeService*, e os parâmetros dessa função são o *pipe advertisement* do *peer* de destino e o *timeout* da conexão. O resultado da execução dessa função, que pode ser vista no trecho de código abaixo, é uma variável da classe *OutputPipe* representando o canal de comunicação entre os *peers*.

```
OutputPipe myPipe = pipeSvc.createOutputPipe(pipeadv, 5000);
```

No caso de uma busca por arquivos de configuração ou dispositivos, o cliente localiza todos os *peers* ativos que fazem parte da rede de gerenciamento e envia a cada um uma mensagem de busca, e se a busca for por configuração, essa mensagem conterá também os parâmetros de busca definidos pelo usuário. No caso de uma reserva de banda, o cliente localiza apenas os *peers* responsáveis pelos dispositivos envolvidos na reserva, e envia a esses uma mensagem contendo os parâmetros definidos. Para enviar uma mensagem pelo canal de comunicação é utilizado o método *send* da classe *OutputPipe*:

```
myPipe.send(<mensagem>);
```

Juntamente às mensagens, tanto de busca quanto de reserva, o cliente envia seu p*ipe advertisement*, para que seja criado um outro canal de conexão, por onde serão enviadas as respostas do servidor do *peer* remoto. Ficando assim estabelecidos dois canais de comunicação entre os *peers*, um em cada direção. Os *pipe adversments* são armazenados em arquivos, no disco do usuário, e é nessa forma que eles são enviados pelas mensagens. A figura 5.11 ilustra o conteúdo de um arquivo que representa um *pipe adversment.*

Após enviar a(s) mensagen(s) a *thread* monitora o *pipe* criado, aguardando respostas dos *peers* remotos. Quando uma resposta é recebida, é tratada de acordo com sua finalidade. Se for o resultado de uma busca por configuração a mensagem conterá um vetor de arquivos de configuração, a identificação do *peer* que armazena esses arquivos, e o seu p*ipe advertisement.* O vetor e a identificação do *peer* são enviados para a classe *ConfigResult.java*, para serem exibidos ao administrador. À medida que as respostas dos outros *peers* envolvidos na busca são recebidas, vão sendo enviadas para a classe que exibe o resultado. No caso de o administrador solicitar o *download* de um arquivo de configuração, o cliente envia novamente uma mensagem ao *peer* remoto fazendo a solicitação, e a *thread* volta a monitorar o *pipe* no aguardo do arquivo.

Se a mensagem recebida for uma resposta de busca por dispositivos, conterá um vetor com dispositivos e a identificação de *peer* responsável por eles, os quais serão enviados para a classe *DeviceResult.java* e exibidos ao administrador. E se for uma resposta de uma solicitação de *download* de arquivo de configuração, a mensagem conterá esse arquivo, o qual será gravado no disco do usuário. Após gravar o arquivo, o cliente registra no banco de dados de dispositivos e configurações a nova configuração.

Quando estiver fazendo uma reserva de banda o cliente, depois de receber as respostas de todos os *peers* remotos consultados, verifica se todos roteadores estão disponíveis e comunica esse resultado ao usuário. No caso de um resultado negativo, o usuário é informado sobre quais roteadores não estavam disponíveis.

```
<?xml version="1.0"?>
<!DOCTYPE jxta:PipeAdvertisement>
<jxta:PipeAdvertisement xmlns:jxta="http://jxta.org">
    <Id>
        urn:jxta:uuid-59616261646162614E504720A8B22AC88CAE9056A04
    </Id>
   <Type>   JxtaUnicast    </Type>
    <Name>
        SvcPipeAdvertisement
    </Name>
</jxta:PipeAdvertisement>
```

Figura 5.11: Pipe Advertisement

5.5 Servidor

O servidor é implementado na classe *Server.java.* A primeira função do servidor é a criação e publicação dos serviços disponíveis. Para isso ele cria um *module class advertisement* associado ao o serviço e o publica no *Peer Group*. Depois cria um *module spec advertisement* para o serviço e o associa a um identificador, para que um cliente remoto possa encontrá-lo na rede. Cria também um *pipe advertisement* para o serviço, que será usado para estabelecer o canal de comunicação com o cliente e associa esse *pipe* ao *module spec advertisement*, que também será publicado no *Peer Group. O pipe advertisement* é lido de um arquivo, para garantir que o *peer* seja identificado sempre pelo mesmo *pipe*. O servidor também publica no mesmo *module class advertisement* um *module spec advertisement* para cada dispositivo gerenciado pelo *peer*, e o identificador desse *advertisement* é formado pelo nome do serviço acrescido do endereço IP do dispositivo. É através desses *advertisements* que os clientes identificam os *peers* responsáveis por determinado dispositivo.

Depois de publicados os serviços disponibilizados pelo *peer*, o servidor monitora o *pipe* criado, aguardando por requisições remotas, através do método *waitForMessage* da classe InputPipe:

```
msg = inPipe.waitForMessage();
```

Ao receber uma mensagem, a primeira providencia é identificar o *pipe* do cliente que a enviou. Para isso a mensagem é enviada para a classe *SvcReceiver.java*, que é responsável por extrair os elementos contidos nas mensagens, além de executar a busca por configurações nos vetores locais e verificar a disponibilidade dos dispositivos para uma reserva de banda.

Depois de identificado o *pipe* da mensagem, o servidor identifica qual o seu propósito. Se a mensagem for uma busca por dispositivos o servidor envia ao requisitante o seu vetor de dispositivos e a identificação do *peer*. Se for uma busca por configurações, a classe *SvcReceiver.java* extrai da mensagem os parâmetros da busca e pesquisa no vetor de configurações do *peer*, retornando o resultado na forma de um vetor de configurações para a classe *Server.java*, que envia esse vetor para o cliente, junto com a identificação do *peer* e o seu pipe *advertisement*.

Se a mensagem for uma requisição de *download* de arquivo de configuração, o servidor responde com uma mensagem contendo esse arquivo, e se for um pedido de reserva de banda a classe *SvcReceiver.java* consulta o dispositivo para determinar sua disponibilidade, retornando a resposta à classe *Server.java*, que por sua vez, comunica o resultado ao cliente. Novamente devido à falta de roteadores disponíveis para testes e à limitação dos serviços disponíveis na versão do roteador XORP em CD, a classe *ReservationData.java* não implementa a consulta aos dispositivos. Essa consulta é simulada através da geração de um número randômico, e o resultado tem 90% de chances de ser positivo para a disponibilidade do roteador. O trecho de código que representa o método que executa essa geração do número randômico está reproduzido abaixo.

```
public static Boolean getReservation(String ip)
{
    if(Math.random() < 0.9) return new Boolean(true);
    return new Boolean(false);
}
```

5.6 Classe JXTAInit.java

Essa classe é responsável pela inicialização da plataforma JXTA, pela criação do *Peer Group*, pela instanciação dos serviços de descoberta e publicação de recursos e pelo serviço de criação e utilização dos *pipes*. È nessa classe que é feita a inicialização do servidor.

Outra funcionalidade da classe é criar uma janela chamada “JXTA Status Monitor”, que apresenta ao usuário o estado atual do Cliente e do Servidor. A figura 5.12 mostra essa janela no momento de uma busca por configurações.

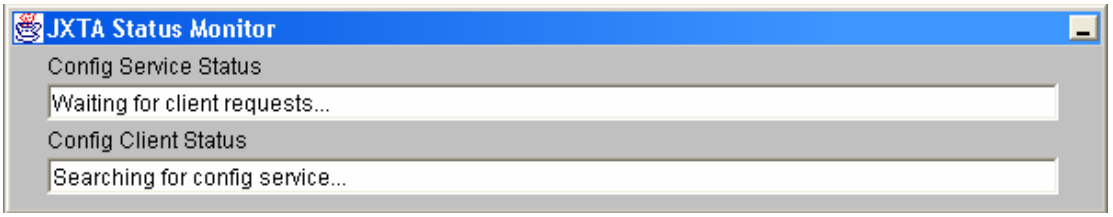

Figura 5.12: Monitor de Status

Neste capítulo foi apresentada a implementação da aplicação. Com a realização dessa implementação foi possível a realização de testes com o ambiente de gerenciamento, os quais estão descritos no capítulo seguinte.

6 AVALIAÇÃO DA SOLUÇÃO

Nos capítulos anteriores foram apresentadas a arquitetura proposta para uma aplicação de gerenciamento cooperativo de redes e a implementação dessa aplicação. Neste capítulo será apresentada uma avaliação dessa aplicação. A partir da implementação da arquitetura foi possível a realização de testes para validar a utilização de sistemas P2P para o gerenciamento cooperativo de redes.

O objetivo desses testes, além de confirmar a possibilidade de utilização da aplicação para realizar o gerenciamento, foi medir a variação do tráfego gerado e do tempo de resposta de acordo com o número configurações e dispositivos envolvidos.

Os quatro serviços implementados foram executados em um ambiente de testes, descrito na seção 6.1. Na seção 6.2, será descrita a metodologia que foi utilizada na realização dos testes. As seções 6.3, 6.4 e 6.5 apresentam a execução dos testes e seus resultados. Por fim, na seção 6.6 são feitas algumas considerações sobre os testes.

6.1 Ambiente de Testes

O ambiente de testes, representado na figura 6.1, foi composto por quatro computadores distribuídos em duas redes locais diferentes, conectadas através da Internet, sendo dois computadores em cada rede. Na primeira rede, em um dos computadores foi executada a aplicação de gerenciamento, sendo esse então um *peer* da rede, e o outro computador foi utilizado para executar o roteador XORP, utilizado para testar o serviço de configuração de dispositivos. Na outra rede os dois computadores executaram a aplicação de gerenciamento.

No mesmo computador em que foi executado o “Peer1” foi instalado um *sniffer* de rede, para capturar o fluxo de dados gerado pela execução dos serviços. O programa utilizado foi Ethereal (OREBAUGH et al., 2004), que possui um filtro para análise de pacotes JXTA.

Os testes foram realizados em redes locais diferentes com o objetivo de verificar, além do funcionamento dos serviços implementados, a capacidade da plataforma JXTA de comunicar-se pela Internet através de NAT e *firewalls.* Para isso “Peer1” e “Peer2” foram configurados para agir como *relay peers*. O “Peer1” também foi configurado como *rendezvous peer*, para servir como índice de informações sobre os recursos disponíveis na rede, reduzindo a quantidade de troca de mensagens para descoberta de recursos entre os *peers*.

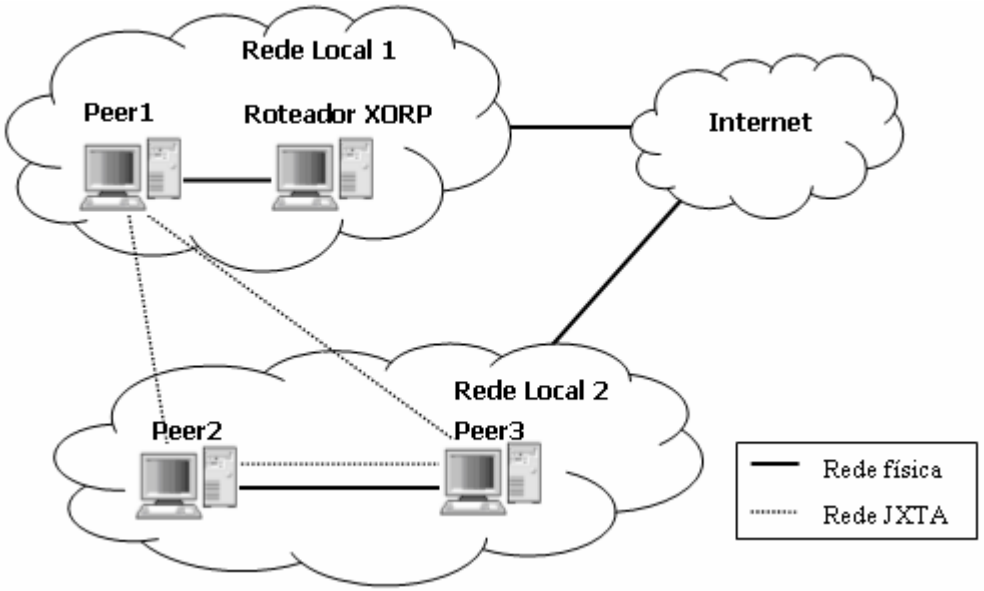

Figura 6.1: Ambiente de Testes

6.2 Metodologia de Testes

Os testes realizados no ambiente apresentado consistiram em três cenários distintos: configuração de dispositivos, localização de dispositivos na rede e tentativas de reserva de banda. Para gerar o tráfego necessário às medições, foram executadas cinco rodadas para cada cenário, variando exponencialmente (de 4 a 64) o número de dispositivos ou configurações envolvidos.

Para o cenário de configuração de dispositivos foi variada a quantidade total de configurações cadastradas nos *peers*. Cada configuração cadastrada é um registro na base de dados do *peer* contendo as informações sobre a configuração e uma referência ao arquivo de configuração.

Para o cenário de localização de dispositivos na rede foi variada a quantidade total de dispositivos cadastrados e para o cenário de reserva de banda foi variado o número de dispositivos envolvidos na reserva. Em cada uma das rodadas foram extraídas trinta medições e dessas foram calculados a média e o desvio padrão.

Nas amostras coletadas, dois parâmetros foram avaliados: tráfego gerado e tempo de resposta percebido. O primeiro refere-se à quantidade de bytes gerada pela troca de mensagens entre os *peers*. De acordo com Sen e Wang (2004), o tráfego P2P pode ser classificado em duas categorias: sinalização e transferência de dados, e para compreender o comportamento de um sistema P2P ambos devem ser considerados. Sendo assim, os dois tipos de tráfego foram medidos e avaliados. O segundo parâmetro refere-se ao tempo decorrido desde a solicitação de um serviço até a chegada da resposta. As medições de tráfego gerado e tempo de resposta foram feitas pelo *sniffer* Ethereal, instalado no “Peer1”.

Foi implementada uma variação do serviço de reserva de banda, na qual o cliente espera a resposta da solicitação de um roteador para executar a solicitação seguinte, para fazer uma comparação entre a reserva de banda implementada, onde a consulta aos roteadores é feita em paralelo, e o modelo de reserva seqüencial. O resultado dessa comparação é apresentado no item 6.5.

6.3 Cenário 1: Configuração de Dispositivos

No primeiro cenário, ilustrado na figura 6.2, o “Peer1” busca pela rede por um arquivo de configuração para configurar um dispositivo sob seu controle, o “Roteador XORP”. Esse arquivo encontra-se, inicialmente, cadastrado no “Peer3”.

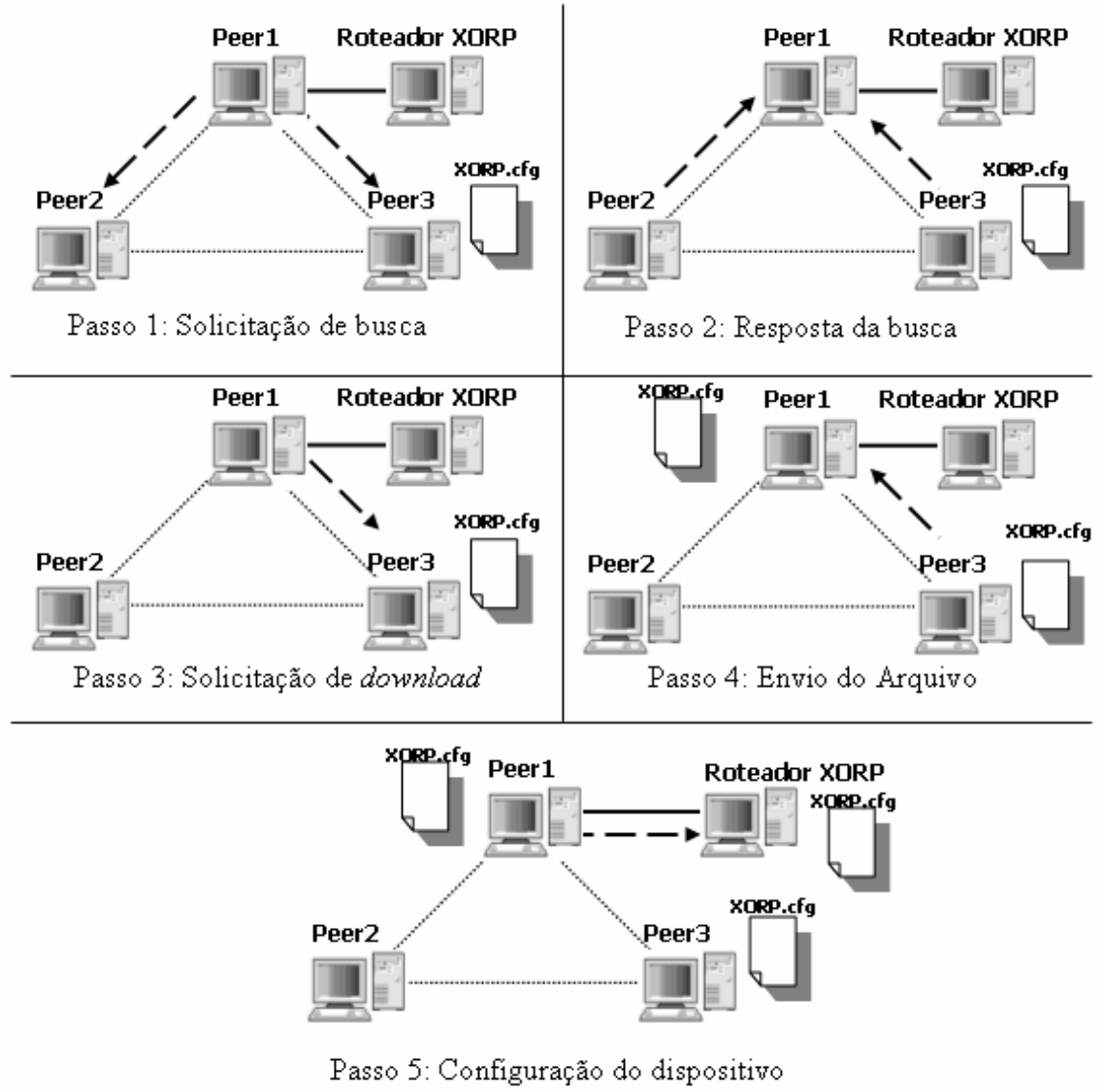

Figura 6.2: Cenário de Busca e Configuração

6.3.1 Execução do Teste

O primeiro passo executado foi uma solicitação do “Peer1” aos outros dois *peers* por seus arquivos de configuração para roteadores XORP. Na seqüência, “Peer2” e “Peer3” respondem com uma lista de configurações que satisfazem os parâmetros de busca, no caso somente o “Peer3” possui tal configuração.

O terceiro passo foi a solicitação do “Peer1” para fazer o *download* do arquivo encontrado no “Peer3”. Quando o arquivo foi armazenado no “Peer1”, o mesmo pode utilizá-lo para configurar o dispositivo. O arquivo foi transferido para o roteador, que por sua vez carregou as configurações nele contidas.

As medições de tráfego gerado e tempo de resposta envolveram somente os passos 1 e 2, ou seja, envolveram somente a busca pelos arquivos de configuração. A aplicação do arquivo de configuração no roteador não foi considerada na avaliação de desempenho, e não foi repetida em todas as rodadas, foi executada somente para testar a capacidade da aplicação de configurar o roteador.

6.3.2 Resultados

A busca por arquivos de configuração, o *download* do arquivo de configuração e a configuração do roteador com esse arquivo foram realizados com sucesso. Devido à corrupção e perda de pacotes, e às periódicas trocas de mensagens de controle entre os *peers*, houve variação no tráfego gerado. A média e o desvio padrão dos valores medidos estão apresentados na tabela 6.1 e os gráficos das figuras 6.3 e 6.4 ilustram a variação das médias de acordo com a quantidade de configurações cadastradas.

Tabela 6.1: Resultados do Cenário de Busca e Configuração

Configurações Cadastradas	Tráfego Gerado (bytes)		Tempo de Resposta (s)	
	Média	Desvio Padrão	Média	Desvio Padrão
4	57924,7	3137,401	3,885	0,372
8	59426,17	3585,822	3,894	0,341
16	65159,93	4632,058	4,240	0,323
32	75129,1	2415,334	4,444	0,230
64	80993,07	3336,457	5,202	0,695

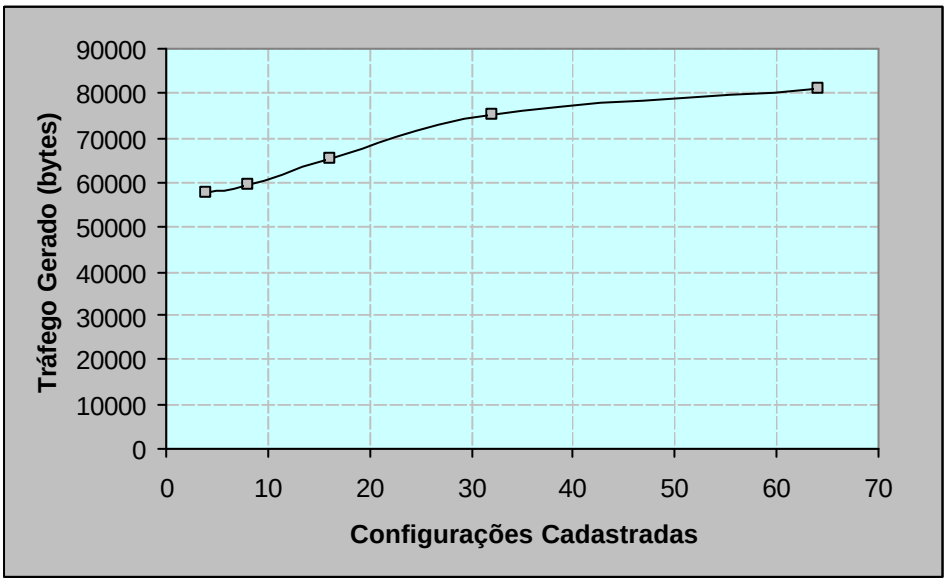

Figura 6.3: Cenário de Busca e Configuração: Tráfego Gerado

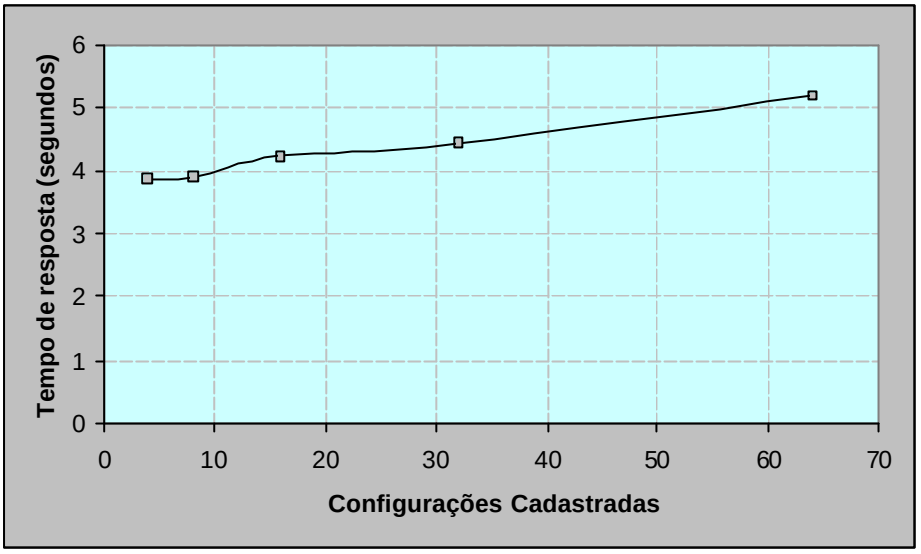

Figura 6.4: Cenário de Busca e Configuração: Tempo de Resposta

6.4 Cenário 2: Localização de Dispositivos na Rede

No cenário de localização de dispositivos, ilustrado na figura 6.5, o "Peer1" consulta os outros *peers* para descobrir quais dispositivos eles gerenciam. Cada um dos outros dois *peers* contem uma lista de dispositivos cadastrados. Esses dispositivos não existem fisicamente, são fictícios.

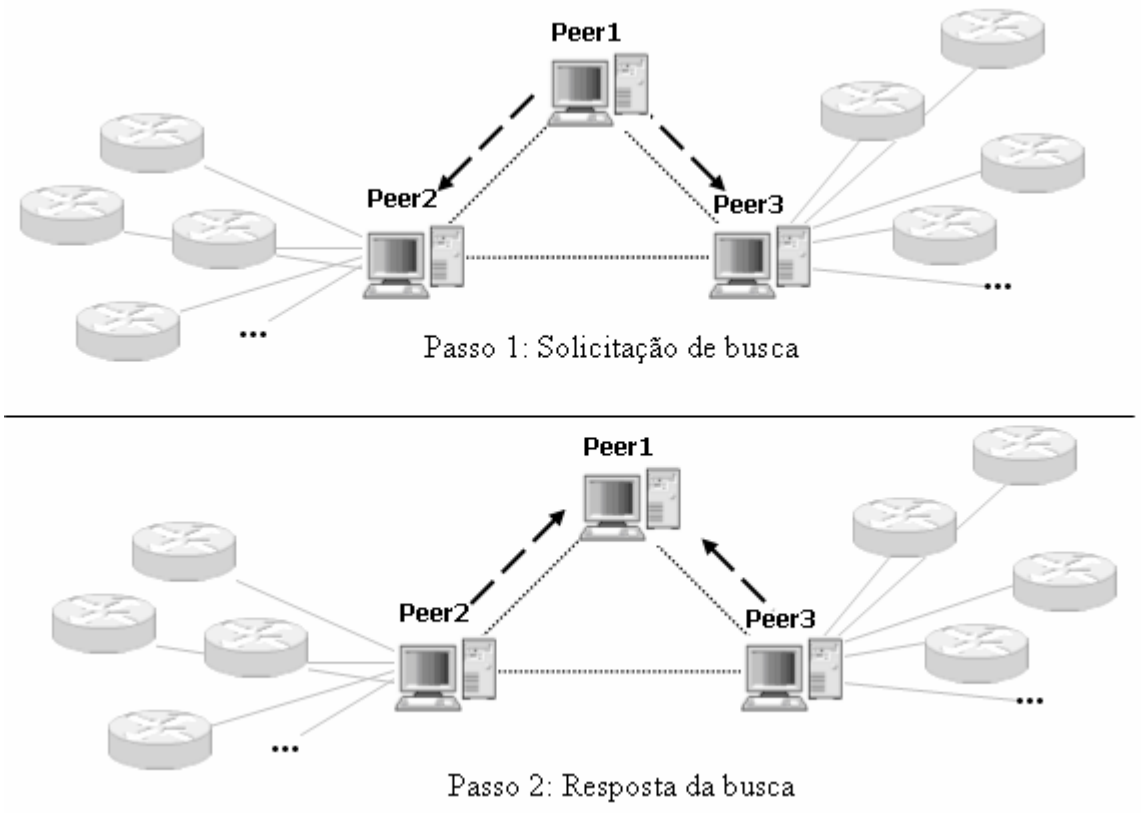

Figura 6.5: Cenário de Localização de Dispositivos na Rede

6.4.1 Execução do Teste

O primeiro passo executado foi uma solicitação do “Peer1” aos outros dois *peers* por seus registros de dispositivos. Na seqüência, “Peer2” e “Peer3” respondem com uma lista contendo todos os dispositivos neles cadastrados.

A diferença entre a busca por configurações, realizada no cenário anterior, e a busca por dispositivos é que na primeira é realizada uma busca interna no *peer* para encontrar e retornar somente alguns registros específicos, e na segunda todos os registros são retornados. A comparação entre os resultados obtidos nos dois cenários pode quantificar a influência da busca interna no *peer* no tempo de resposta.

6.4.2 Resultados

A busca pelos dispositivos cadastrados nos *peers* foi realizada com sucesso. A média e o desvio padrão dos valores medidos estão apresentados na tabela 6.2 e os gráficos das figuras 6.6 e 6.7 ilustram a variação das médias de acordo com a quantidade de dispositivos cadastrados.

Tabela 6.2: Resultados do Cenário de Localização de Dispositivos na Rede

Configurações Cadastradas	Tráfego Gerado (bytes)		Tempo de Resposta (s)	
	Média	Desvio Padrão	Média	Desvio Padrão
4	56237,57	3057,73	3,94	0,31
8	56455,07	3421,94	3,96	0,31
16	61108,9	3690,67	4,29	0,29
32	75037,37	2846,71	4,54	0,28
64	78222,13	4572,34	4,83	0,64

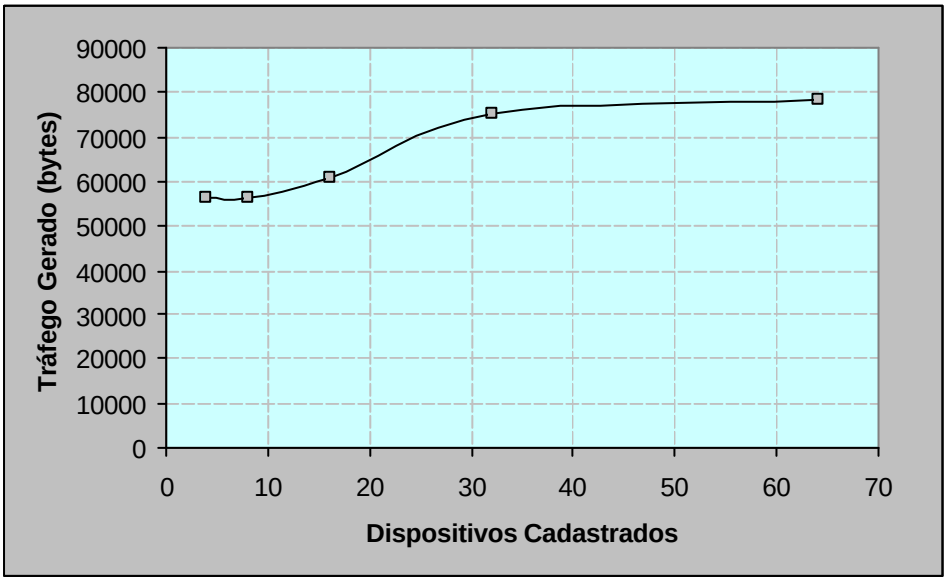

Figura 6.6: Cenário de Localização de Dispositivos na Rede: Tráfego Gerado

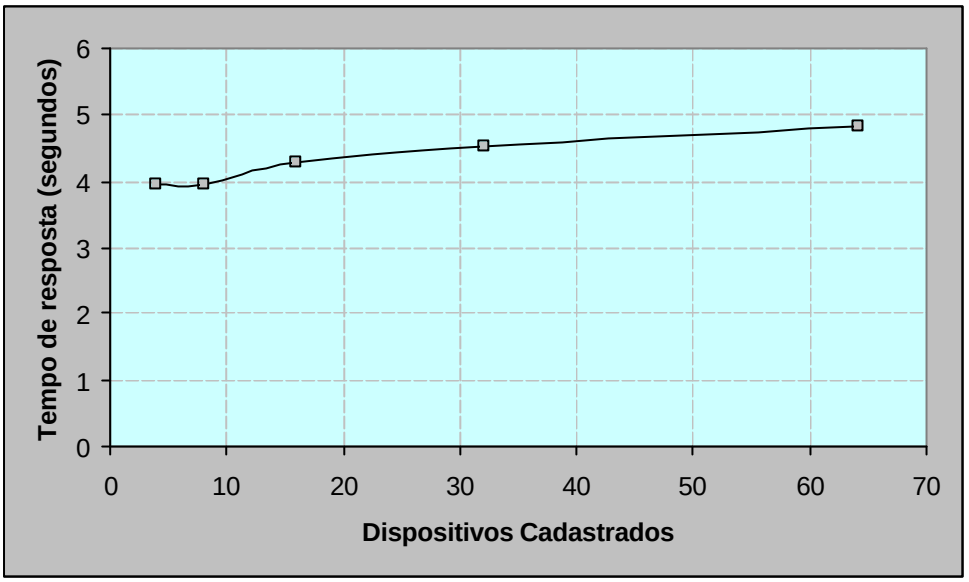

Figura 6.7: Cenário de Localização de Dispositivos na Rede: Tempo de Resposta

6.5 Cenário 3: Tentativas de Reserva de Banda

Nesse cenário foram realizadas diversas tentativas de reserva de banda, solicitadas pelo “Peer1”. As solicitações de reserva foram realizadas de forma a envolver dispositivos cadastrados no “Peer2” e no “Peer3”. O cenário está ilustrado na figura 6.8.

6.5.1 Execução do Teste

No primeiro passo o “Peer1” identifica o *peer* responsável pelo dispositivo solicitado (Peer2 ou Peer3) e encaminha a esse *peer* o pedido de reserva. Ao receberem uma solicitação, “Peer2” e “Peer3” consultam a disponibilidade do dispositivo solicitado e informam a disponibilidade desse roteador ao “Peer1”.

Quando a reserva é feita de forma paralela o “Peer1” encaminha o pedido de reserva de todos dispositivos de uma vez só, e aguarda as respostas de “Peer2” e “Peer3”. Quando a reserva é feita de forma seqüencial o “Peer1” aguarda a resposta de um pedido para então encaminhar o próximo pedido, até que a disponibilidade de todos os dispositivos seja consultada.

As medições foram feitas para a reserva em paralelo e para a reserva seqüencial. O resultado dessas medições e a comparação entre as duas formas de reserva estão apresentadas nos itens a seguir.

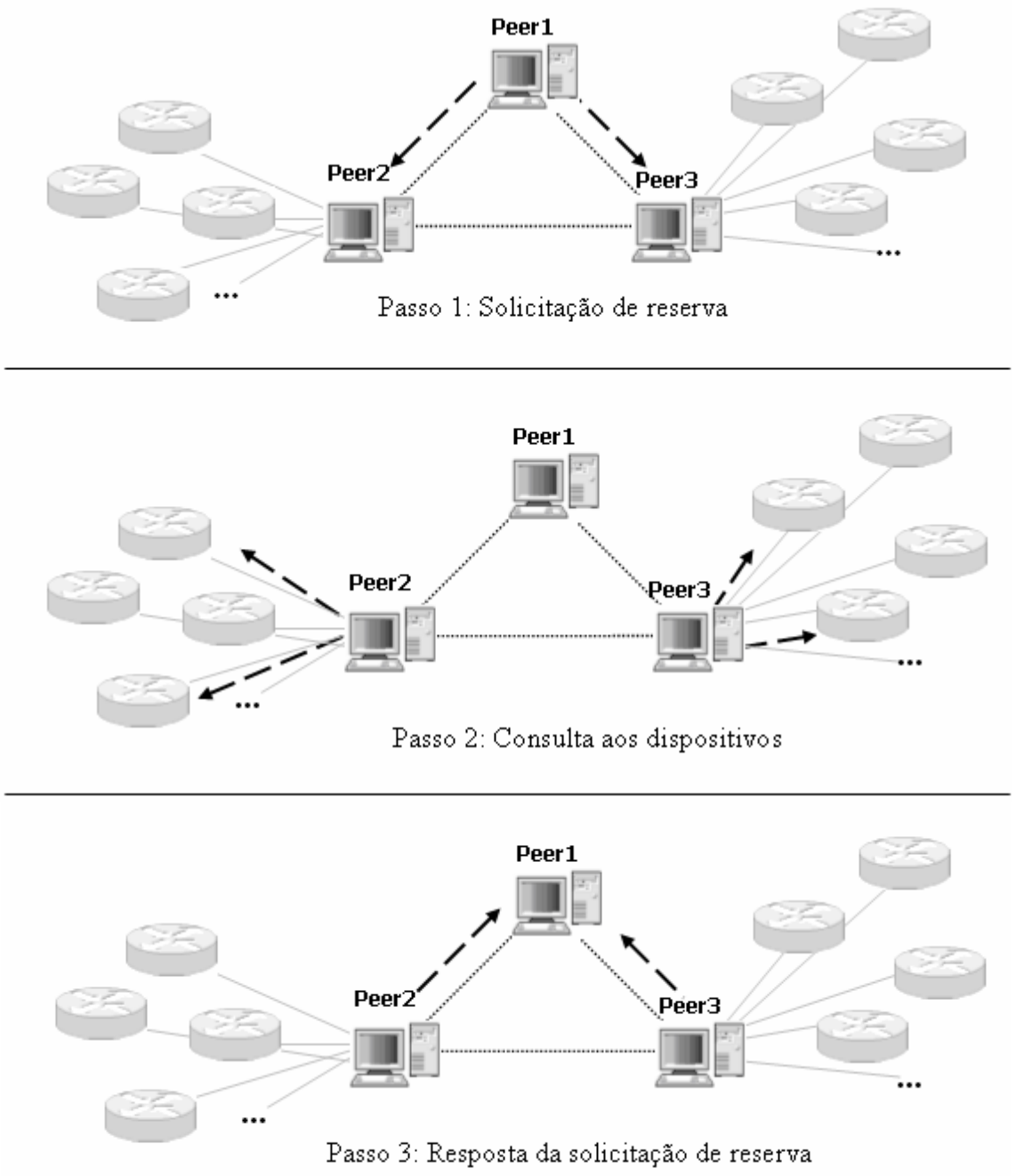

Figura 6.8: Cenário de Tentativas de Reserva de Banda

6.5.2 Resultados

A descoberta dos *peers* responsáveis por cada dispositivo ocorreu de forma correta. Como os dispositivos envolvidos não existem fisicamente, são apenas registros nas bases de dados dos *peers*, a reserva foi simulada, foi possível somente verificar a sua disponibilidade, não foi possível efetivar uma reserva real.

A média e o desvio padrão dos valores medidos para reserva paralela e para seqüencial estão apresentados nas tabelas 6.3 e 6.4, respectivamente, e os gráficos das figuras 6.9 e 6.10 ilustram as variações das médias de acordo com a quantidade de dispositivos envolvidos.

Tabela 6.3: Resultados do Cenário de Tentativas de Reserva de Banda (paralela)

Configurações Cadastradas	Tráfego Gerado (bytes)		Tempo de Resposta (s)	
	Média	Desvio Padrão	Média	Desvio Padrão
4	159706,2	6463,66	8,76	0,36
8	339229,4	9032,47	12,84	1,46
16	621136	13438,57	15,14	0,58
32	1327036	29209,77	19,96	0,78
64	2576367	81557,52	25,51	1,17

Tabela 6.4: Resultados do Cenário de Tentativas de Reserva de Banda (sequencial)

Configurações Cadastradas	Tráfego Gerado (bytes)		Tempo de Resposta (s)	
	Média	Desvio Padrão	Média	Desvio Padrão
4	169158,7	544,16	13,34	0,76
8	268161,7	6729,92	24,9	0,44
16	610313,7	43183,52	54,43	2,72
32	1471557	33781,23	110,13	3,1
64	2737326	129342,9	230,93	27,74

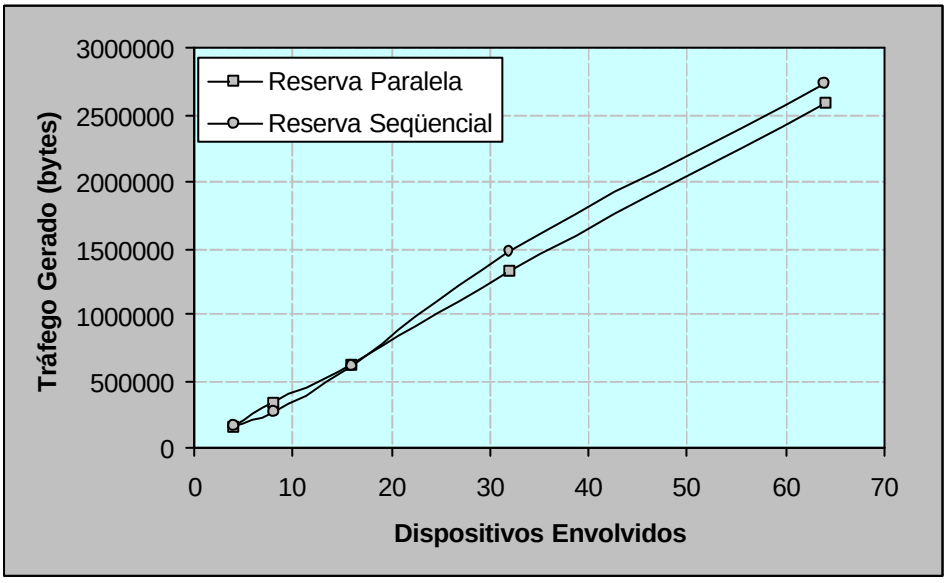

Figura 6.9: Tentativas de Reserva de Banda: Tráfego Gerado

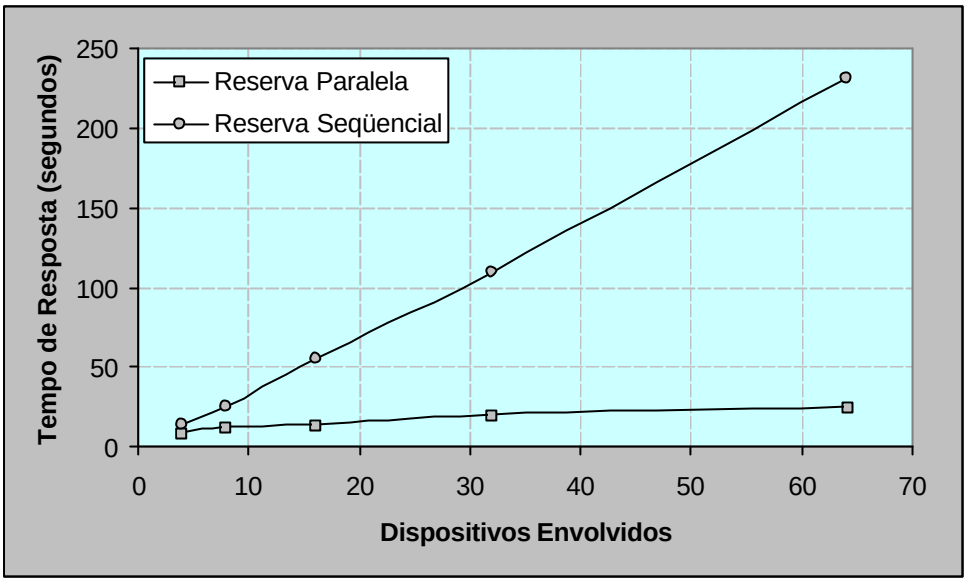

Figura 6.10: Tentativas de Reserva de Banda: Tempo de Resposta

6.6 Considerações sobre os Testes

Os ambientes de testes descritos acima englobaram a utilização de todas as funcionalidades da aplicação: o cadastro de dispositivos e de configurações, a busca por configurações, o *download* de arquivos de configuração, a aplicação dos arquivos em um dispositivo, a busca por dispositivos e a reserva de banda.

A execução dos testes mostrou que os mecanismos de descoberta de serviços da plataforma JXTA funcionaram corretamente, visto que o cliente de cada *peer* encontrou os dois outros *peers* da rede, assim como todos os serviços e dispositivos divulgados. Os *pipes* foram estabelecidos corretamente. Os serviços de *redezenvous peer* e *relay peers* também funcionaram com correção, sendo possível o estabelecimento da comunicação transparente entre *peers* em domínios isolados por *firewalls* e NATs, permitindo que gerentes cooperem em domínios administrativos diferentes.

Pela análise dos dados gerados nos dois primeiros cenários foi possível observar que o aumento do tráfego gerado e do tempo de resposta foi pequeno em relação ao aumento do número de dispositivos ou configurações cadastrados. Os valores obtidos nas duas primeiras rodadas foram praticamente iguais. Isso indica que a maior parte do tráfego gerado é de sinalização, e que o tráfego gerado pela transferência de dados tem pouca influência no desempenho dos serviços.

Comparando os valores obtidos no primeiro cenário com os obtidos no segundo foi possível observar que o tempo de busca interna no *peer* para encontrar e retornar somente alguns registros específicos não tem influência significativa no tempo de resposta do serviço. Essa comparação também reforçou a afirmação de que a maior parte do tráfego gerado é de sinalização.

No terceiro cenário foi observado que o tráfego gerado cresce proporcionalmente ao número de dispositivos envolvidos na reserva, tanto na reserva paralela quanto na seqüencial. A diferença de desempenho entre as duas abordagens de reserva ficou evidente pelos valores medidos para o tempo de resposta. Na reserva seqüencial o

tempo de resposta cresceu proporcionalmente ao número de dispositivos envolvidos, enquanto na reserva paralela o crescimento foi consideravelmente menor.

Com a implementação e os resultados obtidos fica constatada a possibilidade de utilização sistemas P2P para facilitar o gerenciamento cooperativo de redes. As redes P2P permitem que vários tipos de serviços sejam espalhados pela rede de modo que os recursos fiquem mais acessíveis e não sobrecarreguem as máquinas.

7 CONCLUSÃO

A necessidade de um tipo de computação descentralizada, mais colaborativa, e adaptativa aumenta à medida que surgem novos dispositivos e plataformas. A evolução nos sistemas de comunicação tem tornado os serviços e as tecnologias cada vez mais heterogêneas. Surge a necessidade de buscar uma maneira consistente de realizar o gerenciamento das redes para manter toda sua estrutura funcionando de forma suave e atendendo as necessidades de seus usuários e as expectativas de seus administradores. Comparado aos paradigmas tradicionais de gerenciamento, o paradigma distribuído fornece mais escalabilidade, flexibilidade e confiabilidade, mas traz também alguns desafios novos, como por exemplo, a comunicação entre os dispositivos.

Apesar das dificuldades existentes, o desenvolvimento e o amadurecimento na implementação de aplicações P2P vêm sendo notados, bastando observar o sucesso na utilização de aplicativos para compartilhar arquivos e trocar mensagens instantâneas. A adoção de aplicações P2P por empresas privadas deve popularizar a tecnologia, e conseqüentemente disponibilizar e capacitar mão de obra para o desenvolvimento de novas aplicações P2P.

Embora a tecnologia P2P tenha ganhado fama pela distribuição ilegal de arquivos com direitos de cópia e propriedade intelectual, ela tem mais a oferecer que o acesso fácil a música ou vídeos. Os serviços introduzidos pelos sistemas P2P apresentam características inovadoras, e também podem ser utilizados para ajudar a resolver problemas de outras áreas críticas, como por exemplo, a área de gerência de redes.

A iniciativa do projeto JXTA, com o objetivo de especificar protocolos independentes de plataforma, dispositivo ou linguagem de programação, torna-se fundamental para a construção de novas aplicações P2P, além da possibilidade de integrar, sem grandes dificuldades, as novas aplicações que serão construídas seguindo as especificações definidas em JXTA. A implementação para linguagem Java da plataforma JXTA, mostrou-se bastante avançada, permitindo a construção de aplicações P2P sem grandes dificuldades e com resultados satisfatórios.

Aliando-se a necessidade de ferramentas que auxiliem no gerenciamento cooperativo de redes com o crescimento da tecnologia P2P, foi proposta nesta dissertação uma arquitetura de aplicação capaz de fornecer aos administradores alguns recursos para a execução de tarefas de gerenciamento distribuídas. Para elaboração dessa arquitetura foi realizado um estudo sobre a tecnologia P2P e sobre a plataforma de desenvolvimento JXTA.

A partir da arquitetura proposta foi implementada uma aplicação oferecendo quatro serviços de gerenciamento: compartilhamento de arquivos de configuração de dispositivos, compartilhamento de registros de dispositivos, configuração de dispositivos e solicitação de reserva de banda. A aplicação foi escrita inteiramente em Java, utilizando a plataforma JXTA e foi utilizada para realização de testes e validação da proposta.

Os quatro serviços implementados foram executados em um ambiente de testes, com o objetivo de confirmar a possibilidade de utilização da aplicação para realizar o gerenciamento e também de avaliar o seu desempenho. Foi implementada também uma variação do serviço de reserva de banda de forma seqüencial, para fazer uma comparação com a reserva da arquitetura proposta.

Os testes englobaram a utilização de todas as funcionalidades da aplicação, e sua execução mostrou que tanto os serviços implementados quanto os mecanismos da plataforma JXTA funcionaram corretamente. Pela análise dos dados coletados foi possível concluir que a maior parte do tráfego gerado pela aplicação é de sinalização, e que o tráfego gerado pela transferência de dados é consideravelmente menor. Isso significa que o volume de registros armazenados nos *peers* tem pouca influência no desempenho dos serviços. Também foi constatada uma diferença de desempenho entre as duas abordagens de reserva de banda. A reserva paralela apresentou vantagem sobre a reserva seqüencial em relação ao tempo de resposta.

Com a implementação e validação dessa arquitetura ficou constatada a possibilidade da utilização de sistemas P2P para facilitar o gerenciamento cooperativo de redes. As redes P2P permitem que vários tipos de serviços sejam espalhados pela rede de modo que os recursos fiquem mais acessíveis. A tecnologia também possibilita o estabelecimento de comunicação entre *peers* em domínios isolados por *firewalls* e NATs, permitindo que gerentes cooperem mesmo em domínios administrativos diferentes.

A aplicação desenvolvida neste trabalho pode servir de ponto de partida para o desenvolvimento de outros serviços de gerenciamento. Outra possibilidade de trabalho futuro é a extensão do serviço de configuração de dispositivos para outros tipos de roteadores, além do XORP.

REFERÊNCIAS

ANDERSON, D. et al. SETI@home: An Experiment in Public-Resource Computing. **Communications of the ACM**, New York, v.45, n.11, p. 56-61, Nov. 2002.

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

BALAKRISHNAN, H. et al. Looking Up Data in P2P Systems. **Communications of the ACM**, New York, v.46, n.2, p. 43-48, Nov. 2003.

BEARSHARE. **BearShare**. Disponível em: <http://www.bearshare.com>. Acesso em: jul. 2003.

BRADEN, R. et al. **Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification**: RFC 2205. [S.l.]: Internet Engineering Task Force, Network Working Group, 1997.

GOLDSZMIDT, G.; YEMINI, Y. Distributed Management by Delegation. In: INTERNATIONAL CONFERENCE ON DISTRIBUTED COMPUTING SYSTEMS, 15., ICDCS, 1995. **Proceedings…** Vancouver, British Columbia, Canada: IEEE Computer Society, 1995. p. 333-340.

GRANVILLE, L.; DA ROSA, D.; PANISSON, A.; MELCHIORS, C.; ALMEIDA, M.J.; TAROUCO, L. Managing Computer Networks Using Peer-to-Peer Tecnologies. **IEEE Communications Magazine**, New York, v.43 n.10, p. 62-68, Oct. 2005.

HANDLEY, M.; HODSON, O.; KOHLER, E. Xorp: An open platform for network research. In: WORKSHOP ON HOT TOPICS IN NETWORKS, HOTNETS-I, 2002, USA. **Proceedings…**Princeton, NJ, USA: ACM, 2002. p. 53-57.

HARRINGTON, D.; PRESUHN, R.; WIJNEN, B. **An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks**: RFC 3411. [S.l.]: Internet Engineering Task Force, Network Working Group, 2002.

ICQ. Disponível em: <http://web.icq.com>. Acesso em: jul. 2003.

HAARTSEN, J. et al. Bluetooth: Vision, Goals, and Architecture. **Mobile Computing and Communications Review**, [S.l.], v.2 n.4, p. 38-45, Oct. 1998.

LI, G. **Project JXTA:** A Technology Overview. [S.l.]: Sun Microsystems, 2002. Disponível em: <http://www.jxta.org/project/www/docs/jxtaview_01nov02.pdf>. Acesso em: jul. de 2005.

LI, G. JXTA: A Network Programming Environment. **IEEE Internet Computing**, New York, v.5 n.3, p. 88-95, May. 2001.

MICROSOFT. **.NET:** Driving Business Value with the Microsoft Plataform. [S.l.]: Microsoft. Disponível em: <http://www.microsoft.com/net>. Acesso em: ago. 2003.

MICROSOFT. **MSN**. Disponível em: <http://www.msn.com>. Acesso em: ago. 2005.

MILOJICIC, S.; KALOGERAKI, V.; LUKOSE, R. **Peer-to-peer Computing**. Palo Alto: HP Laboratories, 2002. Technical Report.

OREBAUGH, A. et al. **Ethereal Packet Sniffing**. [S.l.]: Syngress, 2004.

PEERMETRICS. **PeerMetrics Peer System**. Draft. Disponível em: <http://www.peermetrics.com>. Acesso em: jul. 2003.

RIPEANU, M.; IAMNITCHI, A.; FOSTER, I. Mapping the Gnutella network: Properties of large- scale peer-to-peer systems and implications for system design. **IEEE Internet Computing Journal,** New York, v.6, n.1, p. 50-57, 2002.

ROMAN, G.; HUANG, Q.; HAZEMI, A. Consistent Group Membership in Ad Hoc Networks. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, ICSE, 2001. **Proceedings…**Toronto, Canada: IEEE Computer Society: ACM, 2001.

SAROIU, S.; GUMMADI, P.; GRIBBLE, S. Measurement Study of Peer-to-Peer File Sharing Systems. In: MULTIMEDIA COMPUTING AND NETWORKING, MMCN, 2002. **Proceedings…**[S.l.:s.n.], 2002.

SCHÖNWÄLDER, J.; QUITTEK, J.; KAPPLER, C. Building Distributed Management Applications with the IETF Script MIB. **IEEE Journal on Selected Areas in Communications**, New York, v.18 n.5, 2000.

SEN, S.; WANG, J. Analyzing Peer-To-Peer Traffic Across Large Networks. **IEEE/ACM Transactions on Networking**, New York, v.12 n.2, p. 219-232, 2004.

SOURCEFORGE TEAM. **J2SSH**. [S.l.]: SourceForge Team, 2003. Disponível em: <http://sourceforge.net/projects/sshtools>. Acesso em: ago. 2005.

SPELLMEIER, G. **Estudo e Implementação de uma Rede Peer-to-Peer para suporte ao Gerenciamento Cooperativo de Redes de Computadores**. 2004. 35 f. Trabalho Diplomação (Bacharelado em Engenharia da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

STOICAY, I. et al. Chord: A Scalable Peer-to-peer Lookup Protocol for Internet Applications. **IEEE/ACM Transactions on Networking**, New York, v.11 n.1, p. 17-32, 2003.

SUN MICROSYSTEMS. **Project JXTA V2.0:** Java™ Programmer's Guide. [S.l.], 2003. Documentação do projeto JXTA.

SUN MICROSYSTEMS. **Project JXTA:** An open, Innovative Collaboration. [S.l.], 2001. Documentação do projeto JXTA. Disponível em: <http://www.jxta.org/project/www/docs/OpenInnovative.pdf>. Acesso em: jul. 2005.

TRAVERSAT, B. et al. **Project JXTA 2.0 Super-Peer Virtual Network.** [S.l.], 2003. Documentação do projeto JXTA. Disponível em: <http://www.jxta.org/project/www/docs/JXTA2.0protocols1.pdf>. Acesso em: ago. 2005.

TRUELOVE, K. **Gnutella and the Transient Web**. [S.l.]: O'Reilly P2P, 2001. Disponível em: <http://www.openP2P.com/lpt/a/705>. Acesso em: mar. 2005.

VERBEKE, J. et al. Framework for Peer-to-Peer Distributed Computing in a Heterogeneous, Decentralized Environment. In: INTERNATIONAL WORKSHOP ON GRID COMPUTING, GRID, 3., 2002. **Proceedings…**[S.l.:s.n.], 2002. p. 1-12. (Lecture Notes in Computer Science, v.2536).