Universidade Federal do Maranhão
Centro de Ciências Exatas e Tecnologia
Programa de Pós-Graduação em Engenharia de Eletricidade

DEFINIÇÃO DE UMA ARQUITETURA P2P BASEADA EM REPUTAÇÃO E ORIENTADA A SERVIÇOS

Flávio Marcílio Paiva Ramos

São Luís
2009

Universidade Federal do Maranhão
Centro de Ciências Exatas e Tecnologia
Programa de Pós-Graduação em Engenharia de Eletricidade

DEFINIÇÃO DE UMA ARQUITETURA P2P BASEADA EM REPUTAÇÃO E ORIENTADA A SERVIÇOS

Flávio Marcílio Paiva Ramos

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia de Eletricidade da Universidade Federal do Maranhão, como parte dos requisitos necessários para obtenção do grau de Mestre em Engenharia de Eletricidade.

São Luís
2009

Ramos, Flávio Marcílio Paiva

Definição de uma Arquitetura P2P Baseada em Reputação e Orientada a Serviços / Flávio Marcílio Paiva Ramos. - São Luís, 2009.

144f.:il.

Dissertação (Mestrado em Engenharia de Eletricidade) - Centro de Ciências Exatas e Tecnologia, Universidade Federal do Maranhão, 2009.

1. Redes Peer-to-Peer. 2. Computação Orientada a Serviços. 3. Protocolos. 4. Reputação. I.Título.

CDU 004.722.2

DEFINIÇÃO DE UMA ARQUITETURA P2P BASEADA EM REPUTAÇÃO E ORIENTADA A SERVIÇOS

Flávio Marcílio Paiva Ramos

Dissertação aprovada em 04 de maio de 2009.

Prof. Zair Abdelouahab, Ph.D.
(Orientador)

Prof. Mario Antonio Meireles Teixeira, Dr.
(Co-orientador)

Prof. Miguel Franklin de Castro, Dr.
(Membro da Banca Examinadora)

Prof. Omar Andres Carmona Cortes, Dr.
(Membro da Banca Examinadora)

"Quando uma porta se fecha outra se abre;
mas nós quase sempre olhamos tanto e de
maneira tão arrependida para a que se fechou,
que não vemos aquelas que foram abertas para nós".

Alexander Graham Bell

À minha esposa e filhos.

À minha mãe, meu pai e minha irmã.

Agradecimentos

A Deus pelo milagre da vida.

À minha mãe Zulmira por seu amor e sacrifícios para proporcionar-me, através da educação, a busca do conhecimento.

A meu pai Antônio pelos seus exemplos de superação.

À minha esposa Gabryela pelo seu amor e apoio incondicional em todas as horas.

Aos meus filhos João Gabriel e Pedro Flávio que aceitaram meus momentos de ausência.

À minha irmã Cláudia e minha sobrinha Lisa, que apesar da distância, sempre estiveram torcendo por mim.

À meus sogros Gilmar e Conceição pelo reconhecimento do meu esforço.

À minha vó Rosa (*in memoria*m) que deve estar muito feliz onde quer que ela esteja.

Ao meu orientador, professor Zair Abdelouhab por acreditar na minha capacidade e por seus valiosos ensinamentos.

Ao meu co-orientador, professor Mário Meireles, por suas importantes opiniões, críticas e pelas horas em que esteve disponível para guiar-me neste trabalho.

Ao Coordenador do Programa de Pós-Graduação de Engenharia de Eletricidade, Sebastian Yuri Catunda, pela oportunidade.

Aos professores Miguel Franklin e Omar Cortes por aceitarem participar desta banca.

A professora Maria da Guia pelos conselhos dados para maior empenho na conclusão deste mestrado.

Aos professores Anselmo, Auxiliadora, Alexandre, Aristófanes, Carlos Sales, Francisco Silva e Ivo; Rosa, Lucinete ("Baixinha") e todos que compõem o Departamento de Informática da UFMA, com os quais tive oportunidade de trabalhar na área docente, como professor substituto.

A todos os colegas de mestrado, Aline, Helaine, Johneth, Falkner e Gilberto, pela amizade, força, incentivo e união durante todo o tempo que passamos juntos, além dos momentos de confraternização que realizamos.

A todos os colegas do LAWS (*Laboratory of Advanced Web Systems*) pela contribuição para este trabalho.

A Alcides por sua atenção dispensada a todos os alunos do programa de mestrado.

Aos colegas de trabalho (TRT - 16ª Região) pelo incentivo e apoio para que esta meta fosse atingida.

E a todos que direta e indiretamente contribuíram nesta importante jornada.

Resumo

As redes *peer-to-peer* (P2P) são muito populares atualmente, principalmente quando se deseja buscar ou compartilhar uma grande quantidade de informações e recursos entre os seus participantes. Uma das grandes dificuldades desse tipo de tecnologia é evitar que esses participantes mantenham, ao mesmo tempo, um número considerável de arquivos compartilhados, e ainda garantir que esses arquivos não sejam conteúdo poluído ou corrompido. Outro problema bem comum nas redes P2P, é a falta de interoperabilidade entre as diversas redes existentes, principalmente devido às incompatibilidades dos metadados e das interfaces das operações utilizadas na comunicação entre os nós. Este trabalho descreve o P2PWSRep, um protocolo de gerenciamento de reputação para identificar nós que não desejam cooperar ou que podem prejudicar o desempenho da rede pelo compartilhamento de arquivos corrompidos, infectados ou inexistentes. A infraestrutura da rede do P2PWSRep foi baseada em serviços web para contornar problemas de interoperabilidade e facilitar sua extensibilidade, tornando-o fácil de ser utilizado por diversas aplicações de redes P2P. O protocolo P2PWSRep possui um cálculo de reputação distribuído, utilizando-se uma média ponderada exponencial, que considera o valor da reputação anterior do nó e o valor atual, obtido dos demais nós da rede, e regulado por um parâmetro de ajuste, para obter a reputação final, de forma que o histórico do comportamento do nó seja considerado. O protocolo P2PWSRep é validado por meio de simulação e os resultados obtidos mostram que o mesmo é capaz de apontar os nós ou recursos mais confiáveis da rede, ao mesmo tempo em que isola aqueles nós que não são íntegros ou pouco cooperativos, além de não impor uma sobrecarga desnecessária à rede P2PWS.

Palavras-Chave: Redes Peer-to-peer. Serviços Web. Protocolos. Reputação.

Abstract

Nowadays, peer-to-peer networks are very popular mainly, when we wish to search or share a considerable amount of information and resources among various participants. One of the main difficulties of this technology is how to avoid that those participants maintain, at the same time, a considerable number of shared resources and to guarantee that those resources are not corrupted or polluted content. Another problem commonly found in P2P networks is the lack of interoperability among existing P2P solutions especially because the inconsistencies of metadata and operation interfaces used in node communication. This work describes P2PWSRep, a reputation management protocol that identifies non-cooperative nodes or that can hinder network performance by sharing corrupted, infected or non-existent files. P2PWSRep infrastructure was based on web services in order to tackle interoperability problems and to facilitate its extensibility, making it feasible to be accessed by several other P2P applications. The P2PWSRep protocol employs a distributed reputation computation using an exponentially weighted average that takes into account the current and previous node reputation and which is tuned by an adjustment parameter in order to obtain the final reputation, thus considering the node's behavior. The P2PWSRep protocol is validated by means of simulation and our results show that it is able to point out the more trustable nodes in the network as well as to insulate those which are not reliable or cooperative. Besides, the protocol does not unnecessarily impacts on the network load P2PWS.

Keywords: Peer-to-peer Networks. Web Services. Protocols. Reputation.

Lista de Figuras

Lista de Tabelas

Lista de Listagens

Lista de Abreviaturas

ATC	Approximate Trust Computation
ATM	Asynchronous Transfer Mode
BFS	Breath First Search
DDoS	Distributed Denial of Service
DHT	Distributed Hash Table
DTD	Document Type Definitions
DTM	Dynamic Trust Computation
FTP	File Transfer Protocol
GPL	General Public License
HTTP	Hypertext Transfer Protocol
IP	Internet Protocolo
IPS	Intrusion Prevent System
IRTF	Internet Research Task Force
LAN	Local Area Network
MPLS	Multiprotocol Label Switching
NAT	Network Address Translation
OASIS	Organization for the Advancement of Structured Information Standards
P2P	Peer-to-peer
P2PWSRep	Peer-to-peer Web Service Reputation Protocol
RPC	Remote Procedure Call
SGML	Standard Generalized Markup Language
SHA	Secure Hash Algorithm
SMS	Short Message Service
SMTP	Simple Mail Transfer Protocol
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
TTL	Time to Live

UDDI	Universal Description, Discovery and Integration
URI	Uniform Resource Identifiers
URL	Universal Resource Locators
W3C	World Wide Web Consortium
WSDL	Web Services Definition Language
WWW	World Wide Web
XML	Extensible Markup Language

Sumário

Capítulo 1

Introdução

A Internet é o resultado do desenvolvimento de uma filosofia de interligação de redes de computadores, cuja característica mais relevante é a total transparência aos usuários dos detalhes relativos às tecnologias e a forma como a interligação e comunicação é feita (Chiozzotto and Mauro, 1999). No final da década de 90, milhões de usuários já utilizavam a rede mundial de computadores, sendo que, no princípio, o modelo de interação era através de uma divisão hierárquica clara, onde os serviços ficavam disponíveis em máquinas chamadas de servidores e os clientes, para acessá-los, utilizavam conexões ainda bem lentas e de baixa qualidade. Esse modelo, bem conhecido e utilizado até hoje, é chamado de Modelo Cliente/Servidor. É um modelo hierárquico, sendo que há claramente a divisão de papéis entre os elementos da rede, onde quem fornece o serviço é o elemento servidor e quem consome o serviço é o elemento cliente. Com o crescimento acelerado da Internet, logo surgiram preocupações a respeito de sua infraestrutura, já que os mais pessimistas apostavam no seu colapso, afirmando que esta não suportaria a demanda de milhares de computadores que ingressavam na rede por dia, enquanto outros não concordavam com as restrições de provedores para armazenamento ou troca de arquivos e estavam em busca de alguma tecnologia que possibilitasse uma interação *ad-hoc*, sem dependência de uma entidade ou elemento de rede centralizado. A Figura 1.1 mostra o cenário, ainda na década de 80, da Internet. Percebe-se que já havia uma cobertura bem grande no território americano e conexões com outros

continentes.

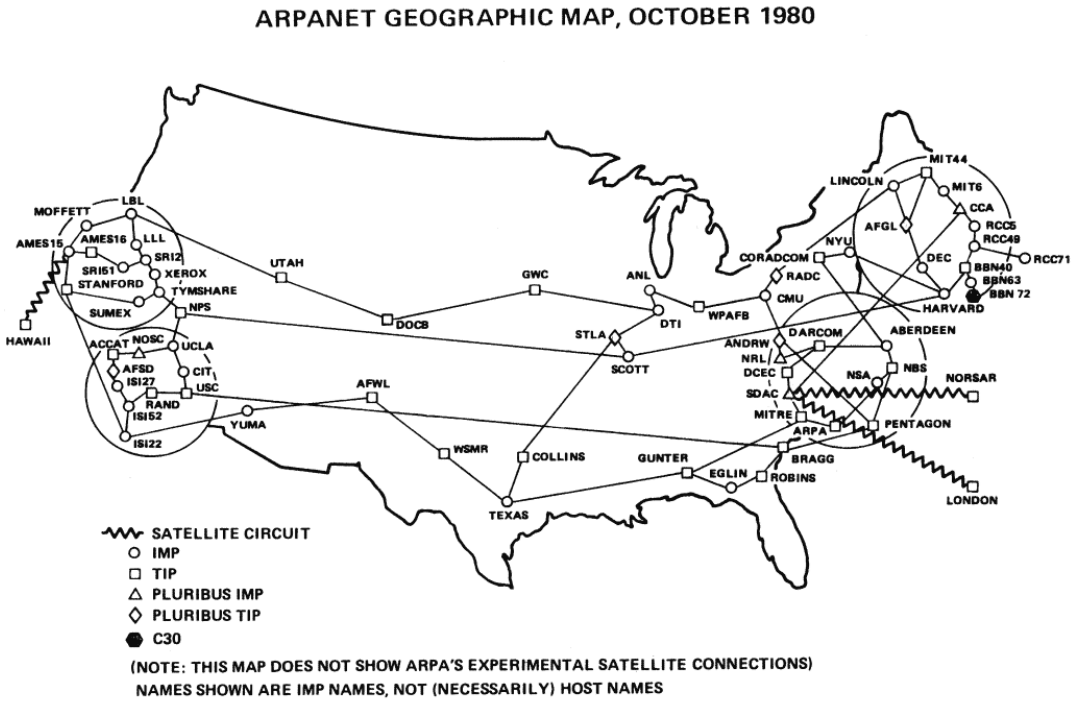

Figura 1.1: Cenário da Internet na década de 80. (Computer Museum History)

Na década de 90, os usuários da Internet já contavam com máquinas que possuíam recursos que há uma década eram inexistentes e a um custo relativamente baixo, assim estavam em busca de se livrar das restrições dos provedores, como mencionado anteriormente. Foi então que um novo modelo de troca de informações surgiu na Internet: as redes de conexão ponto-a-ponto (*pee-to-peer*). As redes P2P distinguiam-se do modelo Cliente/Servidor por utilizar uma comunicação não hierárquica, onde as partes que constituem o grupo podem se comunicar livremente umas com as outras, numa comunicação *ad-hoc*, compartilhando seus recursos e utilizando os dos demais, isto é, não há rotulação de papéis do tipo cliente ou servidor dentro da rede. Nessas redes, os integrantes são chamados de nós (*peers*) e interagem diretamente, sem necessidade de um elemento para intermediar a comunicação, daí a denominação dessa rede de *peer-to-peer*.

As redes P2P alcançaram seu auge por volta do ano 2000, com um serviço chamado de Napster (Napster, 2008), que no seu melhor momento, teve mais de 50

milhões de usuários trocando arquivos de músicas, o que representou a maior violação de direitos autorais em toda a história da indústria fonográfica. As empresas fonográficas então se reuniram e começaram uma batalha judicial para por fim ao Napster, e finalmente conseguiram. Mas, logo após a interrupção do funcionamento do Napster, surgiram outras redes P2P, não necessariamente para compartilhar músicas, mas qualquer tipo de arquivo, sendo que as novas redes criadas surgiram com a preocupação de preservar o anonimato de cada integrante e garantir a descentralização das informações e recursos compartilhados na rede. Atualmente, além de compartilhamento de arquivos e recursos, as redes P2P estão presentes nas mais diversas áreas, tais como, jogos *on-line*, comunicação instantânea, sites de relacionamento, telefonia sobre Internet, etc.

1.1 Motivação e Justificativa

As redes P2P são o meio mais utilizado atualmente quando se deseja compartilhar uma grande quantidade de informações e recursos, já que é difícil concentrar o mesmo volume de dados em poucos servidores de arquivos. A principal dificuldade é preservar o desempenho da rede, sem a utilização de um elemento que centralize as informações a respeito dos nós que integram a rede ou a qualidade e confiabilidade do que é compartilhado entre mesmos. A inserção de elementos moderadores, sejam eles autônomos ou controlados, facilita o gerenciamento da rede, mas também introduz outros problemas, tais como confiabilidade, desempenho e segurança, já que este elemento representa um ponto de falha e sua capacidade depende diretamente do número de nós que o utilizam, tornando-se um alvo fácil a ataques direcionados. A designação de um nó da rede para executar operações de gerenciamento (supernó) também induz o usuário a retirar-se da rede ou alterar o protocolo a fim de evitar ter qualquer sobrecarga em sua máquina.

A baixa interoperabilidade entre as diversas redes existentes também afeta as redes P2P, principalmente devido às incompatibilidades dos metadados e das interfaces das operações utilizadas na comunicação entre os nós. Os usuários, às

vezes, têm em suas máquinas vários clientes de rede P2P, o que muitas vezes os forçam a ter várias cópias do mesmo arquivo armazenado em seu disco, devido a configurações exigidas para cada rede. Este problema de interoperabilidade pode ser resolvido colocando-se entre as redes elementos chamados de *gateways*, que fazem interconexão entre redes P2P diferentes. Mas é bom lembrar que esses elementos podem impor limitações de desempenho e escalabilidade.

Como mencionado anteriormente, as redes P2P visam o compartilhamento e troca de arquivos entre seus integrantes. Nesta curta definição do seu objetivo, identificam-se dois elementos cruciais, o primeiro deles, são os arquivos (ou recursos, como são chamados nas redes P2P), principal alvo ou meio de ataque à rede e aos seus integrantes. O segundo são os usuários, que podem agir de duas formas, sendo uma menos prejudicial em termos de segurança, mas que afeta diretamente desempenho da rede, já que eles não costumam compartilhar os dados que obtêm dos demais; e outra mais prejudicial, onde os mesmos utilizam a rede para distribuir arquivos contaminados por vírus e outros códigos maliciosos, disponibilizar arquivos corrompidos e espalhar na rede informações inconsistentes.

A justificativa para realização deste trabalho é propor um protocolo de reputação que seja aplicado a qualquer rede P2P e que possa, de forma eficiente, identificar os usuários que tentam prejudicar o desempenho e conteúdo das mesmas. Além disso, apresentar a definição da infraestrutura orientada a serviços de uma rede P2P para proporcionar interoperabilidade, em nível de operações e de recursos disponibilizados, incluindo o protocolo de reputação.

1.2 Objetivo Geral

Apesar de muitos pensarem que as redes P2P já esgotaram seus temas de pesquisa, ainda há inúmeros problemas a serem solucionados. Este trabalho trata da identificação de nós mal intencionados e a interoperabilidade entre redes P2P, através da utilização de um protocolo de gerenciamento de reputação para identificar quais os melhores e piores nós da rede, não necessariamente por métricas

já consideradas ineficazes, tais como, largura de banda, tempo de conexão do nó, número de recursos, volume de dados armazenados, etc., pois essas métricas são características obtidas diretamente do nó, sem nenhuma participação dos demais. O gerenciamento de reputação através do protocolo P2PWSRep pretende diminuir o número de nós que possam comprometer o desempenho da rede, bem como, o conteúdo corrompido. A interoperabilidade é tratada através da definição de uma arquitetura orientada a serviços, que está presente na infraestrutura básica da rede e em operações como ingresso na rede, busca de recursos, anotação dos dados, gerenciamento de reputação e transferência de arquivos.

1.3 Objetivos Específicos

Este trabalho tem dois objetivos bem definidos. Decerto, é sabido que o gerenciamento de reputação traz a grande contribuição para as tecnologias de redes P2P, mas a busca de uma interoperabilidade perfeita também é muito importante. Para atingir os dois objetivos gerais deste trabalho, alguns objetivos específicos tiveram que ser atingidos:

- Estudo da viabilidade de utilização de tecnologias de serviços web na construção de infraestruturas P2P, para possibilitar as interações entre os componentes da rede;
- Definição das interfaces servidor e cliente de cada nó, incluindo a definição dos dados contidos nas interações entre os serviços;
- Definição de repositórios de metadados para anotação dos nós, recursos e outras informações armazenadas nos mesmos;
- Definição dos mecanismos de busca de nós e recursos na rede;
- Estudo de algoritmos de reputação aplicados em redes P2P;
- Definição de um protocolo de reputação apto a tratar os problemas de poluição e interoperabilidade na rede;

- Estudo comparativo do protocolo proposto com outros protocolos existentes;
- Simulação e validação do protocolo de reputação.

1.4 Metodologia

Este trabalho foi realizado em três etapas, baseadas nos procedimentos metodológicos que orientam a realização de uma pesquisa científica. A primeira etapa consistiu no levantamento bibliográfico, através da leitura artigos, dissertações, livros e sites da web, a fim de obter conhecimento das áreas que foram estudadas durante a pesquisa. As seguintes áreas foram destacadas como importantes para o trabalho:

- Arquiteturas Orientadas a Serviços;
- Serviços Web;
- Redes *Peer-to-Peer*;
- Reputação;
- Ferramentas de Simulação de Redes.

A segunda etapa do trabalho consistiu na definição do protocolo de reputação P2PWSRep, com uma infraestrutura orientada a serviços, para identificar a reputação de nós e recursos na rede P2P, baseando-se em informações fornecidas pelos demais nós da rede e utilizando-se de valores anteriores da reputação dos nós procurados. Nesta etapa, foram realizados estudos dos protocolos de reputação bem aceitos pela comunidade científica. A terceira etapa abordou a validação do protocolo através de um modelo de simulação, onde foram estudadas várias ferramentas de simulação, de tal forma que se pudesse utilizar uma que fosse mais adequada ao propósito do trabalho. A última etapa concentrou-se em definir uma rede orientada a serviços utilizando o protocolo de reputação P2PWSRep, definindo as interfaces dos serviços, métodos, estruturas de dados para armazenamento e manipulação dos dados e outros detalhes da arquitetura.

1.5 Estrutura da Dissertação

Esta dissertação esta organizada em 7 capítulos. O **Capítulo 1** traz uma visão geral do trabalho e está dividido em contextualização, motivação, objetivos, justificativa e metodologia. O **Capítulo 2** apresenta os conceitos relacionados com serviços web, apresentando o embasamento teórico importante para o entendimento deste trabalho, onde são descritos, a arquitetura orientada a serviços e seus componentes básicos. O **Capítulo 3** traz os conceitos referentes às redes P2P, aplicações de redes P2P, suas arquiteturas, tipos de busca, segurança e exemplos de redes P2P, sendo apresentados neste trabalho, o Gnutella e Napster. No **Capítulo 4** são apresentados os conceitos de reputação e apresentados trabalhos relacionados, abordando vantagens e problemas encontrados em cada um deles. O **Capítulo 5** apresenta o protocolo de reputação P2PWSRep, resultado deste trabalho de mestrado, onde é descrita a sua arquitetura e algoritmo. O **Capítulo 6** descreve a validação do protocolo P2PWSRep por simulação, onde é apresentado cada experimento e análise dos resultados obtidos. O **Capítulo 7** apresenta as conclusões do presente estudo e trabalhos futuros que podem ser desenvolvidos a partir do mesmo.

Capítulo 2

Serviços Web

Neste capítulo são apresentados os conceitos da Arquitetura Orientada a Serviços (SOA) e do elemento essencial dessa arquitetura, o serviço. Além de apresentar uma visão geral sobre serviços web e das tecnologias fundamentais formadoras de sua base, XML, SOAP, WSDL e UDDI.

2.1 Introdução

As organizações estão adotando cada vez mais a Arquitetura Orientada a Serviços com a finalidade de suportar suas aplicações. SOA (*Service Oriented Architecture*) é um paradigma computacional que emergiu como consequência direta da necessidade de colaboração e integração de componentes funcionais, disponibilizados nesse contexto como serviços, entre empresas, de forma interoperável e com fraco acoplamento. Serviços web são uma implementação poderosa dessa arquitetura, que podem ser basicamente definidos como uma interface para recursos computacionais, acessível através da troca de mensagens baseadas em XML (*Extensible Markup Language*) e construída com a utilização de protocolos padrões da Internet.

Segundo a definição do W3C (*World Wide Web Consortium*), Serviço Web é um sistema de software projetado para suportar interoperabilidade máquina a máquina através de interações sobre uma rede de computadores. Sua interface é

descrita em um formato que é processado em ambos os lados, especificamente em WSDL (*Web Services Definition Language*). As bases para a construção de um serviço web são os padrões XML e SOAP (*Simple Object Access Protocol*).

O SOAP é um protocolo leve para troca de informação estruturada em um ambiente descentralizado e distribuído. É baseado na linguagem XML e consiste em três partes: um envelope, isto é, seu cabeçalho, que descreve a mensagem, seu conteúdo e como processá-la, um conjunto de regras para codificar instâncias de tipos de dados da aplicação e uma convenção para representar chamadas de procedimentos remotos e suas respostas. O transporte dos dados é realizado normalmente via protocolo HTTP (*Hypertext Transfer Protocol*), mas também podem ser utilizados outros protocolos, tais como SMTP (*Simple Mail Transfer Protocol*) e FTP (*File Transfer Protocol*). Os dados são transferidos no formato XML, encapsulados pelo protocolo SOAP.

Um serviço web é uma solução utilizada na integração de sistemas e na comunicação entre aplicações diferentes. Com esta tecnologia, é possível que novas aplicações possam interagir com aquelas já existentes e que sistemas desenvolvidos em plataformas diferentes sejam integrados. Em uma definição mais objetiva, serviços web são componentes que permitem as aplicações enviar e receber dados em formato XML. Cada aplicação pode ter a sua própria linguagem, que é traduzida para uma linguagem universal, o formato XML. O processo de publicação, pesquisa e descoberta de serviços web utiliza o protocolo UDDI (*Universal Description, Discovery and Integration*).

2.2 Serviço

Um serviço está disponível em um ponto particular de uma rede de computadores, recebendo e enviando mensagens; e comporta-se de acordo com sua especificação. Os aspectos funcionais de um serviço são descritos usando metadados, e as limitações e condições que estão associadas com o uso do serviço são especificadas através de políticas que podem ser adicionadas a esses metadados. Inter-

faces e políticas que descrevem os termos e as condições que guiam a utilização do serviço são publicadas de forma que os usuários possam descobrir e receber toda a informação de que eles precisam para se associar, muitas vezes dinamicamente, àquele serviço.

A informação que é publicada sobre o serviço fornece detalhes daquilo que o serviço oferece. O serviço também oferece toda a informação necessária, para que, no ambiente onde está hospedado, um cliente do serviço possa acessá-lo e interagir com ele de forma bem-sucedida. Embora o ambiente precise dessa informação, o cliente não necessita ficar ciente desse tipo de informação de acesso, já que não está interessado em entender os detalhes sobre a implementação do serviço. Além de permitir que um consumidor potencial concentre-se especificamente na funcionalidade que está sendo ofertada, outra vantagem do modelo de serviço está na capacidade de criação de novos serviços a partir daqueles já existentes sem abandonar o paradigma (W3C, 2008).

2.3 Arquitetura Orientada a Serviços

A Arquitetura Orientada a Serviços (SOA) é um paradigma para organização e utilização de competências distribuídas que estão sob controle de diferentes domínios proprietários. A SOA utiliza serviços como elementos fundamentais para desenvolver aplicações ou soluções (OASIS, 2008).

Na arquitetura SOA, todos os componentes de software são modelados como serviços, possibilitando que processos de negócio sejam projetados para serem utilizados remotamente e acessados através de protocolos Internet. A Figura 2.1 mostra os papéis e as principais operações desta arquitetura. Qualquer serviço contém três papéis principais: o serviço requisitante (*service requestor*), o provedor do serviço (*service provider*) e um registro de serviços (*service registry*) (Ferris and Farrell, 2003).

Um provedor de serviço é comparável com o "servidor" do modelo cliente/servidor. Ele é responsável por criar uma descrição do serviço, disponibilizando-o

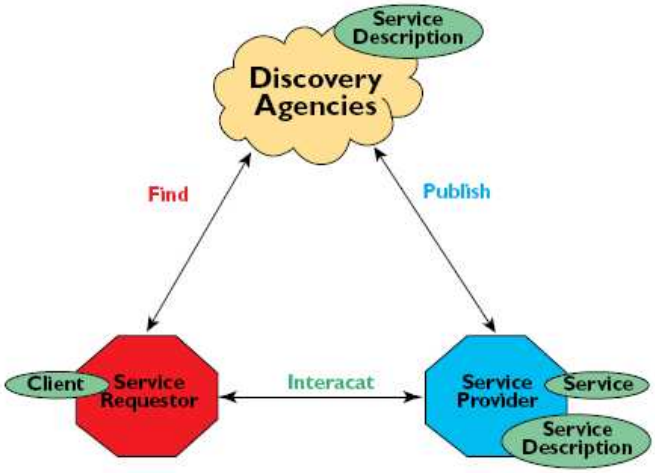

Figura 2.1: Arquitetura SOA. (W3C, 2008)

no ambiente distribuído e tornando-o acessível por outras entidades da rede. Ele também é responsável por publicar o serviço em um ou mais serviço de registros (UDDI) e receber requisições de um ou mais serviços requisitantes, encaminhando-as para o serviço adequado de acordo com a descrição contida na requisição. Assim, pode-se visualizar o provedor de serviços como uma entidade que disponibiliza seus serviços na Internet.

O serviço requisitante é comparável com a figura do "cliente" da arquitetura cliente-servidor. Ele é responsável por achar um serviço através de uma descrição, que submetida a um registro de serviços retorna sua localização. Uma vez localizado o serviço, ele faz a requisição ao serviço hospedado no provedor de serviços. O registro de serviços é responsável por manter as descrições dos serviços publicados pelos provedores de serviços e por permitir que requisitantes de serviços pesquisem em sua coleção de serviços publicados. A pesquisa e publicação dos serviços no catálogo de serviços não é obrigatória, pois o provedor pode dar a descrição dos serviços que oferece diretamente ao requisitante, desde que este último saiba a localização do provedor.

As operações SOA são:

- **Publicação (*publish*)** - É a ação da publicação, pelo provedor de serviços, dos seus serviços em um registro;

- **Busca (*find*)** - É a ação de pesquisar em um registro quais provedores de serviços oferecem determinados serviços, de acordo com a descrição fornecida;
- **Interação (*bind*)** - É a operação de comunicação entre o requisitante (cliente) e o provedor do serviço (servidor).

2.4 Linguagem XML

Serviços web é uma das concepções mais bem sucedidas da arquitetura orientada a serviços. Podem ser definidos como um método de integração de dados e aplicações via padrões XML através de plataformas computacionais e sistemas operacionais (Benz and Durant, 2003). Aplicações habilitadas para serviços web fazem chamadas e enviam respostas umas às outras por meio de um formato XML chamado de SOAP. Serviços web são descritos aos clientes e a outras aplicações servidoras através do uso de outro formato XML chamado de WSDL. Informação de registro e localização de um serviço web pode ser publicada em um diretório UDDI.

O XML desempenha papel vital no núcleo da tecnologia, por ser facilmente transportável, compatível e comum com a camada de transporte pela qual é transmitida em todos os formatos de comunicação. A Figura 2.2 ilustra como XML posiciona-se na base dos três principais padrões adotados para formar a tecnologia de serviço web. Outros padrões também compatíveis com serviço web empregam XML como sua linguagem básica.

A Linguagem XML é derivado da SGML (*Standard Generalized Markup Language*) (ISO 8879), um padrão internacional para definir documentos eletrônicos. SGML é uma linguagem de definição de documentos auto-explicativos usada para descrever os mais variados tipos de documentos. Ela especifica formas de como descrever porções de um documento com marcadores de identificação. XML é uma versão especializada de SGML usada para descrever documentos eletrônicos disponíveis na Internet. Ela não é na prática uma linguagem, mas uma metalinguagem para definir novas linguagens. A definição de XML é independente de

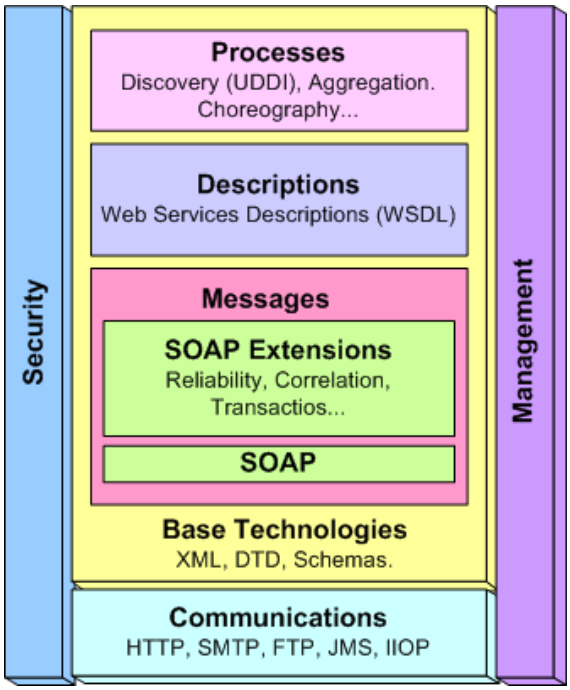

Figura 2.2: Linguagem XML como base de serviços web. (W3C, 2008)

plataforma e é especificada usando *Unicode*, permitindo-a representar conteúdo de várias linguagens naturais. Devido a estes fatores e do largo suporte de XML por parte da maioria dos fornecedores de *software*, XML tem se tornado o formato de fato para a troca de dados entre entidades diferentes.

XML descreve a estrutura de documentos ao especificar marcadores que identificam e delimitam porções de documentos. Cada uma dessas porções é chamada de elemento. Elementos podem aparecer aninhados, sendo que o elemento mais externo é chamado de elemento raiz, e os contidos no elemento raiz são seus elementos filhos. Cada um desses elementos podem, por sua vez, ter seus próprios elementos filhos. Além disso, XML fornece uma forma de associar pares nome e valor, chamados de atributos, junto aos elementos.

Elementos XML começam com um marcador inicial e terminam com um marcador final. Cada tipo de documento possui um conjunto de marcadores válidos. Marcadores iniciais consistem de um nome colocado entre os sinais de menor e maior,

como em $< inicio >$. O marcador final correspondente é similar ao marcador inicial, só diferenciando deste pelo acréscimo da barra (/) depois do sinal de menor, como em $< /inicio >$. Atributos podem ser usados para caracterizar elementos entre os marcadores inicial e final. As restrições sintáticas, por exemplo, a exigência de que um marcador final complemente um marcador inicial, é definido por XML. Essas restrições capacitam o uso de analisadores sintáticos (*parsers*) de XML, que devem ser flexíveis o bastante para trabalhar com qualquer documento especificado por XML.

Uma propriedade importante dos documentos XML é a capacidade de serem unidos para criar um novo documento, resultado da combinação de elementos e atributos de diversas fontes, cada uma trabalhando independentemente das outras. Essa capacidade de reutilização de documentos XML para compor novos documentos produz dois problemas: reconhecimento e conflito. O problema de reconhecimento diz respeito à possibilidade do processador XML reconhecer e separar quais elementos do novo documento pertencem aos documentos que o originaram. O problema do conflito ocorre quando existem elementos repetidos e recorrentes em mais de um documento usado para construir o documento final.

2.4.1 *Document Type Definitions* (DTD)

Na maioria das aplicações que utilizam documentos XML, o fato de ser bem formado não garante que os mesmos possam ser processados e cumpram a função para a qual foram criados. Nesse contexto, tem-se a noção de validade. O DTD (W3C, 2008) constitui-se em mecanismo declarativo que pode automatizar a validação de conteúdos XML quando estes são processados por analisadores sintáticos. Para manipular a checagem de validação, o mecanismo DTD pode implementar regras nos seguintes aspectos de um documento XML: identificação de elementos que podem ou devem estar presentes em um documento, identificação da ordem e da relação entre elementos e identificação de atributos de todos os elementos, além da determinação de quais destes atributos são obrigatórios ou opcionais.

Apesar de representarem um significativo avanço na validação de documentos

XML, deficiências podem ser encontradas em DTDs, o que limita sua utilização em um grande número de aplicações, principalmente naquelas orientadas a dados. Entre estas limitações pode-se citar: os arquivos de validação DTDs não são escritos em XML, tornando sua manipulação e processamento complexos; DTDs foram projetados antes dos *XML Namespaces*, que são fundamentais para aplicações orientadas a dados; DTDs não comportam noções sobre tipos de dados, o que impossibilita a representação de estruturas de dados provenientes de linguagens de programação.

2.4.2 XML *Schemas*

XML Schema (W3C, 2008) é uma metalinguagem usada com especificações XML para descrever estrutura de dados, limitações sobre o conteúdo, e tipos de dados. Foi desenvolvido para fornecer um controle mais poderoso sobre os dados do que aquele proporcionado por DTDs. Diferentemente dos DTDs, *XML Schema* é expresso em XML, o que elimina a necessidade dos analisadores sintáticos manipularem outra sintaxe, além de produzir o ganho do poder de XML. O próprio vocabulário do *XML Schema* também é autodefinido empregando XML.

Além de especificar estruturas XML, *XML Schema* define um conjunto de tipos de dados simples que pode ser usado para estabelecer valores para atributos e elementos, e construtores recursivos de tipo, para definir arbitrariamente estruturas complexas de tipo. Uma vez que o esquema tenha sido definido, processadores de esquema são capazes de validar um documento para assegurar que o mesmo corresponda à estrutura do esquema e aos valores permitidos. Essa verificação pode eliminar a origem de várias das vulnerabilidades que prejudicam sistemas baseados na *Web*.

2.5 *Simple Object Access Protocol* (SOAP)

Na arquitetura de serviços web, o protocolo padrão de fato para comunicação entre duas partes é o SOAP. SOAP é um protocolo baseado em XML que fornece um mecanismo simples para a troca de informação tipada e estruturada

entre serviços. Foi projetado para reduzir o custo e a complexidade de se integrar aplicações construídas em diferentes plataformas de *hardware* e *software*.

Uma das características mais importantes do SOAP é a separação entre o formato do dado a ser transmitido e o protocolo de nível inferior que irá transportar o dado, garantindo a independência de plataforma e linguagem, pois emprega-se XML para descrever os dados e protocolos bem estabelecidos na Internet, como HTTP, para transportar as mensagens (Abinaer and Lins, 2006). Geralmente, os dados transportados em uma mensagem SOAP representam chamadas remotas de procedimentos (RPC) que invocam procedimentos específicos em um provedor de serviço. A especificação do protocolo SOAP limita-se em formalizar questões básicas que informam precisamente como aplicações diferentes, estabelecidas em ambientes de comunicação distintos, devem proceder para trocar mensagens SOAP. Projetado para manter a compatibilidade, a capacidade de expansão e adequação ao ambiente da Internet, SOAP pode expandir-se para acomodar tecnologias e padrões emergentes, bem como acomodar padrões já estabelecidos.

2.5.1 Mensagem SOAP

Uma mensagem SOAP é a unidade básica de comunicação entre nós SOAP, representada por um documento XML bem formado que contém três elementos distintos: um envelope (*Envelope*), um cabeçalho (*Header*) e um corpo (*Body*). A Figura 2.3 mostra como os três principais elementos de uma mensagem SOAP são estruturados. Além destes elementos, o protocolo SOAP define o elemento *Fault* para manipulação de erros que devem ser reportados ao emissor da mensagem.

O elemento envelope funciona como *container* para os elementos cabeçalho e corpo. É o elemento raiz da mensagem SOAP, sua função principal é indicar ao receptor da mensagem onde começa e termina a mensagem. Ao encontrar o marcador $< Envelope >$, o receptor sabe que pode processar a mensagem de acordo com a especificação SOAP.

Opcionalmente, uma mensagem SOAP pode conter um cabeçalho, designado pelo marcador $< Header >$. Caso esteja presente na mensagem, ele deve ser

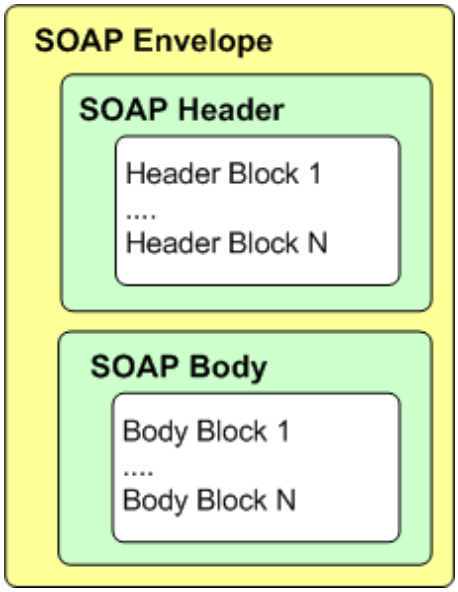

Figura 2.3: Estrutura da mensagem SOAP. (W3C, 2008)

o primeiro imediatamente após o elemento *envelope*. O elemento cabeçalho possibilita uma seção na qual podem ser adicionados metadados relativos ao conteúdo do elemento corpo e/ou instruções para processamento específico da mensagem SOAP.

Apesar de não obrigatório, o cabeçalho constitui-se no mecanismo chave para a capacidade de expansão do protocolo SOAP, já que com a adição apropriada de entradas no cabeçalho, nós de processamento são instruídos a se comportarem de modo específico em determinados contextos como autenticação, autorização, roteamento, etc. Ao receber a mensagem SOAP com este atributo, o nó deverá executar o que determina a entrada ou, em caso de falha, enviar uma mensagem padrão de erro.

O elemento corpo, indicado pelo marcador $< Body >$, deve obrigatoriamente estar presente em todas as mensagens SOAP. Este elemento carrega a carga útil destinada ao receptor final da mensagem. A carga útil pode conter uma chamada RPC, uma resposta RPC ou documentos XML. O elemento $< Fault >$ é um mecanismo sofisticado e preciso para informar ao emissor a ocorrência de algum erro ou falha no processamento de mensagens, também pode aparecer no corpo da mensagem.

O código apresentado na Listagem 2.1 mostra uma mensagem SOAP simples com alguns elementos possíveis: envelope, cabeçalho e corpo.

Listagem 2.1: Exemplo de mensagem SOAP

```
1  <?xml version="1.0" enconding="uft-8"?>
2  <SOAP–ENV:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-
3    instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
4    xmlns:SOAP–ENV="http://schemas.xmlsoap.org/soap/envelope">
5    <SOAP–ENV:Header>
6      <a:authetication xmlns:a="http://www.wrox.com/soap/authetication"
7        SOAP–ENV:mustUnderstand="1">
8        <a:username>usuario</a:username>
9        <a:password>senha</a:password>
10     <a:authetication>
11   </SOAP–ENV:Header>
12   <SOAP–ENV:Body>
13     <cmd:processReboot xmlns:cmd="http://www.wrox.com/soap/cmd">
14       <ip xsi:type="xsd:string">192.168.1.3</ip>
15     </cmd:processReboot>
16   </SOAP–ENV:Body>
17 </SOAP–ENV:Envelope>
```

2.5.2 Modelo de Processamento SOAP

O comportamento de um nó SOAP, ao manusear mensagens SOAP, é determinado pelo modelo de processamento. Um nó SOAP é uma implementação das regras de processamento descritas na especificação SOAP e pode enviar, receber e processar mensagens SOAP. Três categorias de nós de processamento são definidas por SOAP, emissor, receptor e intermediário. Se um nó SOAP envia uma mensagem, ele é chamado de emissor SOAP. Diferentemente do receptor SOAP, responsável por receber uma mensagem. Alguns nós se situam entre o consumidor e o provedor do serviço, atuando tanto como emissor quanto como receptor. Nesse caso, eles são chamados de intermediários. Geralmente, um nó intermediário também é um serviço web capaz de adicionar funcionalidades às transações que passam por

ele.

Uma mensagem SOAP pode destinar cabeçalhos para nós específicos, de modo a garantir que a mensagem seja roteada passo a passo, obedecendo à sequência de processamento pré-estabelecida. Cada entrada no cabeçalho deve conter o nome do nó a qual se destina, informação identificada através do atributo *actor* na entrada do cabeçalho.

Essa sequência de processamento faz com que uma mensagem SOAP em algumas ocasiões atravesse vários nós SOAP. Um emissor SOAP inicial cria a mensagem; a mensagem por sua vez passa por vários intermediários antes de chegar ao receptor SOAP final, cuja função principal é processar a carga útil do corpo da mensagem. Esse conjunto de nós SOAP é chamado de caminho da mensagem SOAP.

2.5.3 SOAP sobre Protocolos de Transporte

Uma mensagem SOAP pode ser transmitida sobre diferentes protocolos de transporte: HTTP, protocolo de correio eletrônico (SMTP) ou até mesmo protocolos proprietários de transporte. O protocolo selecionado talvez forneça recursos adicionais como garantia de entrega, correlação de uma resposta para uma solicitação, ou detecção e correção de erro, mecanismos esses que estendem as funcionalidades básicas oferecidas pelo protocolo SOAP. Além disso, o protocolo base de transmissão talvez suporte padrões de troca de mensagem mais complexos do que a troca de mensagem em sentido único especificada por SOAP.

Devido a sua grande utilização na Internet, HTTP é de longe o protocolo de transporte mais utilizado na troca de mensagens SOAP. Até mesmo a especificação SOAP oferece tratamento especial para HTTP, descrevendo com detalhes como a semântica do modelo de troca de mensagens SOAP é mapeado para HTTP. SOAP sobre HTTP é naturalmente compatível com as convenções RPC do SOAP (solicitação/resposta), já que HTTP é um protocolo baseado neste modelo. Uma mensagem de solicitação SOAP é enviada a um servidor HTTP junto a uma solicitação HTTP, e o servidor retorna a mensagem de resposta SOAP em uma resposta HTTP.

2.6 *Web Services Description Language* (WSDL)

WSDL é um formato XML para descrever serviços de rede como um conjunto de pontos de acesso que funcionam através de mensagens contendo informação orientada ao documento ou orientada ao procedimento. As operações e as mensagens são descritas de forma abstrata e ligadas ao protocolo de transporte e ao formato da mensagem para definir um ponto de acesso. WSDL é extensível para permitir que a descrição de pontos de acesso e suas mensagens não levem em consideração quais formatos de mensagem ou protocolos de transporte estão sendo usados na comunicação (Weerawarana et al, 2005).

O provedor de um serviço web utiliza a WSDL para construir um documento que descreve detalhes necessários na invocação do serviço por um cliente. Detalhes como: a interface do serviço, quais métodos estão disponíveis, as assinaturas dos métodos e os valores retornados, o endereço onde o serviço está localizado e o protocolo que o serviço é capaz de processar para efetuar a comunicação, são disponibilizados para a captura independente de plataforma e linguagem. O uso da WSDL proporciona algumas vantagens:

- Torna mais fácil a criação e a manutenção de serviços através do fornecimento de uma abordagem mais estruturada para definir interfaces de serviços web;
- Facilita o consumo de serviços web ao reduzir a quantidade de código (e erros potenciais) que uma aplicação cliente precisa implementar;
- Facilita a implementação de mudanças que provavelmente serão menos impactantes às aplicações clientes SOAP. A descoberta dinâmica de descrições WSDL permite que tais mudanças sejam repassadas automaticamente aos clientes que estejam usando WSDL, de forma que modificações custosas no código do cliente não tenham que ser feitas a cada vez que uma mudança ocorra.

2.6.1 Estrutura do Documento WSDL

Um documento WSDL organiza-se em duas seções lógicas: uma descrição abstrata e uma descrição concreta. A parte abstrata, composta pelos elementos $< types >$, $< message >$ e $< portType >$, descreve o comportamento de um serviço web em relação às mensagens que ele consome e produz. A parte concreta se ocupa em informar como e onde acessar a implementação de um serviço. Essas informações são cobertas pelos elementos $< binding >$ e $< service >$. A estrutura sintática do documento WSDL é mostrada na Figura 2.4.

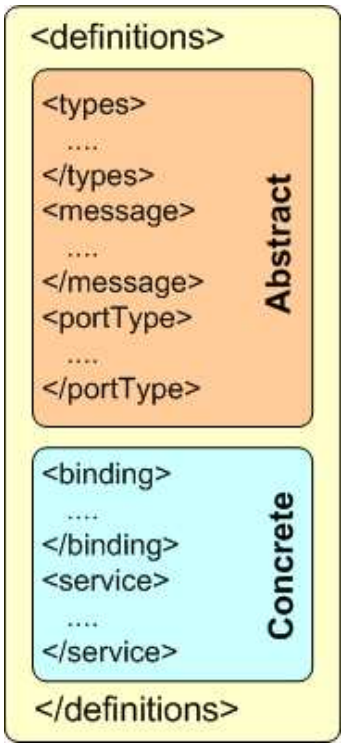

Figura 2.4: Estrutura do documento WSDL. (W3C, 2008)

Arquivos WSDL iniciam com um elemento raiz $< definitions >$, que contém geralmente atributos definindo *namespaces* que descrevem quais XML *Schemas* são usados ao longo do documento WSDL. O elemento $< types >$ engloba a definição dos tipos de dados relevantes para as mensagens trocadas entre o serviço e o cliente.

Todas as mensagens que possam vir a ser trocadas entre o consumidor do serviço e o serviço são listadas no documento WSDL, através do emprego do elemento $< message >$. Esse elemento é responsável por definir de forma abstrata o conteúdo das mensagens. O elemento $< portType >$ define um conjunto de operações relacionadas que um serviço web suporta. Uma operação é um agrupamento das mensagens que podem ser trocadas em uma interação particular com o serviço e é indicado pelo elemento $< operation >$. Quatro tipos de operações são suportados:

- **Unidirecional** - Uma mensagem chega ao serviço e não é produzido nada como resposta;
- **Solicitação/Resposta** - Uma mensagem chega ao serviço e é produzida uma mensagem como resposta;
- **Pedido/Resposta** - O serviço envia uma mensagem e obtém uma resposta de volta;
- **Notificação** - O serviço envia uma mensagem e não recebe nada como resposta.

O protocolo usado para transportar as mensagens é especificado na descrição concreta pelo elemento $< binding >$. Esse elemento associa um protocolo específico (SOAP, HTTP, SMTP, etc) a um elemento $< portType >$ particular. O elemento $< service >$ é a parte final da descrição de um serviço. Através de seu elemento filho $< port >$ ele indica o endereço de rede onde o serviço web está localizado, aguardando para ser acessado.

2.7 *Universal Description, Discovery and Integration* (UDDI)

Um consórcio de companhias, incluindo Ariba, IBM e Microsoft, começou o desenvolvimento do conceito de um diretório de negócios na Internet. Esse desen-

volvimento produziu como resultado o projeto UDDI, uma infraestrutura integrada para descrição, publicação e descoberta de informações sobre serviços web oferecidos na Internet. Como se trata de uma iniciativa aberta da indústria para oferecer um serviço de diretório, UDDI foi estabelecido como um mecanismo aberto, padronizado e independente de plataforma, cujo foco é facilitar a localização de serviços disponíveis.

Basicamente, UDDI disponibiliza uma forma estruturada de descrever uma organização, os serviços que são oferecidos por essa organização e uma interface para os serviços. A necessidade do uso de UDDI justifica-se cada vez mais à medida que o número de serviços web considerados para a escolha e utilização cresce muito. Nesse cenário, UDDI garante que pelo menos grande parte dos potenciais parceiros que oferecem serviços web, aderentes aos requisitos procurados, fossem realmente considerados nessa escolha.

Antes do aparecimento do projeto UDDI, não havia na indústria uma forma padronizada de oferecer a clientes e parceiros maiores informações sobre serviços web disponibilizados pelas organizações. Não havia método uniforme e padrão que detalhasse como integrar os sistemas e processos já estabelecidos e disponíveis, e como esta integração se daria entre parceiros de negócios que se interessassem pelos serviços web oferecidos.

2.7.1 Registro UDDI

Um registro UDDI é um repositório de dados sobre os negócios e os serviços oferecidos por eles. Possibilita a qualquer um recuperar dados UDDI existentes, bem como a qualquer empresa registrar-se a si própria e seus respectivos serviços. A implementação do registro UDDI é semelhante a uma lista telefônica constituída de páginas brancas, amarelas e verdes.

Nas páginas brancas estão disponíveis informações sobre contato e identificadores da empresa, incluindo nome da empresa, endereço, números de telefone, além de outras informações sobre a empresa. A função das páginas brancas é permitir a descoberta de serviços web oferecidos através da identificação de cada em-

presa. As páginas amarelas descrevem serviços web usando diferentes categorias, permitindo sua descoberta, segundo um grupo, uma classificação que reúne serviços similares. As páginas verdes são usadas para indicar os serviços web oferecidos pelas empresas, incluindo todas as informações técnicas necessárias na interação com o serviço, ou com sua utilização, como parâmetros, localização e outras.

2.8 Considerações Finais

Nos últimos anos, os serviços web têm chamado a atenção de muitas empresas e pesquisadores que buscam soluções para criar e integrar aplicações sem altos investimentos, utilizando-se de padrões da Internet. Serviços web foram criados para construir aplicações deste tipo ou possibilitar tal integração. A facilidade que serviços web trouxeram para desenvolvedores, devido à utilizaçao de documentos XML para troca de informações, os tornam independente de plataforma, sendo possível que novas aplicações possam interagir com aquelas já existentes e que sistemas desenvolvidos em plataformas diferentes sejam compatíveis. Cada aplicação pode ter a sua própria "linguagem", que é traduzida para uma linguagem universal, o formato XML. Assim, serviços web, atualmente, se apresentam como a melhor escolha para integrar e desenvolver soluções.

Capítulo 3

Redes P2P

3.1 Introdução

A rede mundial de computadores sempre teve como proposta principal promover a liberdade, que deve se traduzir no acesso irrestrito a todos os recursos da rede, de qualquer lugar e a qualquer hora. Apesar disso, a Web ainda está presa ao modelo cliente/servidor, no qual servidores centralizados executam tarefas para clientes distribuídos, ou seja, a maior parte das máquinas participam da Web apenas como coadjuvantes, acessando recursos providos pela minoria (Rocha et al., 2004).

A tecnologia P2P (*peer-to-peer*) surge para mudar o paradigma existente, à medida que não depende de uma organização central ou hierárquica, além de dispor aos seus integrantes as mesmas capacidades e responsabilidades (Parameswaran et al., 2001). Através dessa tecnologia qualquer dispositivo pode acessar diretamente os recursos de outro, sem nenhum controle centralizado.

3.2 Evolução das Redes P2P

As redes P2P são sistemas que estão emergindo como uma nova forma de computação distribuída, principalmente quando se fala de uma organização independente de restrições, estrutura descentralizada, autônoma e com participação irrestrita dos integrantes (Ting and Deters, 2003). Essa evolução começou na década

de 90, com o ganho de capacidade de computadores *desktop* e a desmotivação de usuários da Internet com as restrições para troca e armazenamento de dados nos provedores de acesso. O surgimento das redes P2P representou a primeira experiência de uma rede totalmente livre de cláusulas contratuais e restrições tecnológicas impostas pelos provedores. Esse surgimento se deu com a popularização da Internet, melhores equipamentos e aumento da largura de banda das conexões de acesso. A Figura 3.1 mostra o modelo básico de uma rede P2P. Percebe-se que todos os nós podem ser interconectados entre si. Cada participamente tem um identificador único dentro da rede e as mesmas funcionalidades estão presentes em todos os nós.

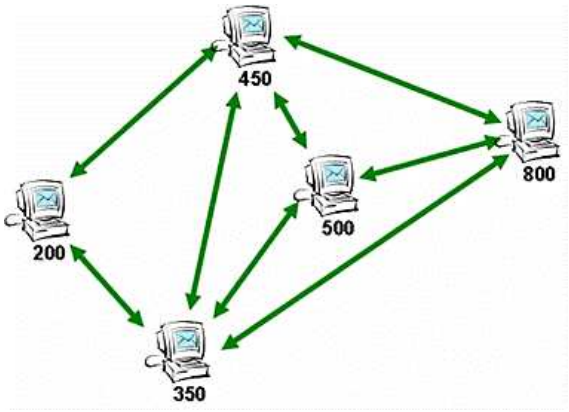

Figura 3.1: Exemplo de uma rede P2P.

As redes P2P são tidas como sistemas colaborativos (GT-P2P, 2008) que emergiram a partir de 2000, principalmente com as aplicações para distribuição de arquivos. Os nós conectados ao sistema colaborativo formam uma rede virtual sobre a rede de dados subjacente, baseada no protocolo IP, no caso da Internet. Os sistemas P2P trazem a conectividade para as bordas da rede, permitindo que qualquer equipamento conectado se comunique e colabore com os demais (Sadok, 2003).

Segundo (Loo, 2003), o modelo de rede P2P colaborativo tem sido apontado como uma das maiores transformações da Internet nos últimos anos e com tendência de expansão. Esse modelo é atrativo por várias razões: os sistemas P2P oferecem

meios para agregar e utilizar recursos geograficamente distribuídos; o custo para a criação dos ambientes colaborativos é baixo e não requer investimento maciço em equipamentos; a natureza descentralizada desses sistemas torna-os inerentemente resistentes a falhas ou ataques intencionais; e, alta escalabilidade para tratar do crescimento de elementos que se juntam à rede P2P (Righi et al., 2004).

Atualmente, as redes P2P respondem pela maior parte do tráfego da Internet. Considerando que muitas das trocas de recursos são falhas, deve-se empregar cada vez mais mecanismos para evitar que isto ocorra, possibilitando melhor desempenho das redes P2P e por sua vez, da Internet. A Figura 3.2 mostra um gráfico do aumento do tráfego das aplicações P2P na Internet de 1993 a 2006. Percebe-se que o tráfego de dados gerado pelas redes P2P, em 2006, já superava o dos demais protocolos, como o FTP, HTTP (Web) e o SMTP (e-mail). Isto mostra que estes protocolos já não são tão utilizados para troca de dados como outrora.

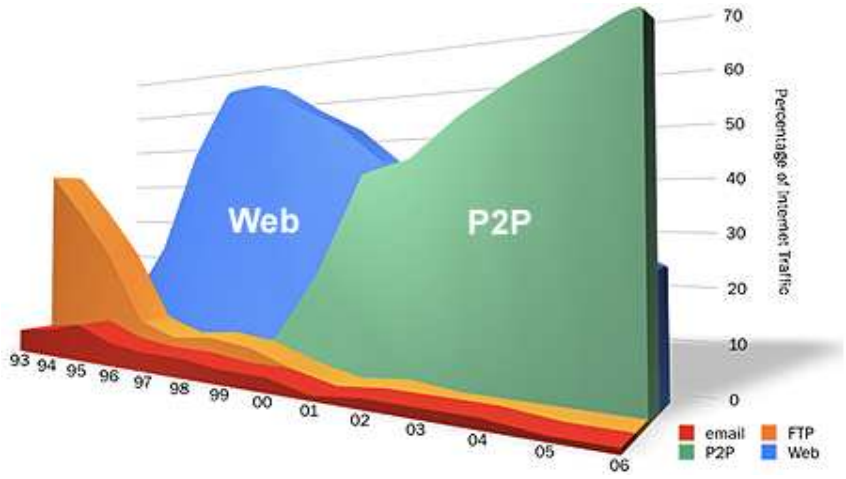

Figura 3.2: Tráfego das redes P2P entre 1993 e 2006. (Cache-Logic Reseach, 2006)

3.3 Características das Redes P2P

Redes P2P são redes virtuais que funcionam sobre a infraestrutura da Internet com o objetivo de compartilhar recursos entre seus participantes, sendo que

por princípio não há diferenciação entre os mesmos. O grupo de pesquisa sobre redes P2P da *Internet Research Task Force* (IRTF-P2P-GROUP, 2008) as define como compartilhamento de recursos e serviços computacionais diretamente entre sistemas. Em geral, é aceito pela comunidade que sistemas P2P devem suportar os seguintes requisitos:

- Nós podem estar localizados nas bordas da rede;
- Nós com conectividade variável ou temporária e endereços também temporários;
- A capacidade de lidar com diferentes taxas de transmissão entre nós;
- Nós com autonomia parcial ou total com relação a um servidor centralizado;
- Assegurar que os nós possuam capacidades iguais de fornecer e consumir recursos de seus nós;
- A rede deve ser escalável;
- A capacidade dos nós se comunicarem diretamente uns com os outros.

Tendo todas estas características, uma rede pode ser dita P2P, mesmo que algumas das funções de controle da rede estejam localizadas em um servidor central.

3.4 Aplicações P2P

As redes P2P têm se colocado como a melhor solução para muitas aplicações que necessitam de uma ampla participação e interação dos usuários no seu funcionamento. Nesta seção serão apresentadas as principais aplicações que fazem uso deste tipo de rede (Da Silva, 2007).

3.4.1 Compartilhamento de Arquivos

São as aplicações mais populares e foram responsáveis pelo cenário atual das redes P2P. Normalmente não há restrições de publicação e obtenção do conteúdo

disponibilizado. Porém, essas redes têm sido um dos principais motivos de estudos para melhoria do desempenho, já que elas são responsáveis por grande parte do conteúdo que trafega na Internet. As redes P2P deste tipo mais populares são Napster (Napster, 2008), Gnutella (Gnutella, 2008), eDonkey (eDonkey, 2008), Kazaa (Kazaa, 2008) e BitTorrent (BitTorrent, 2008).

3.4.2 Computação Distribuída

São aplicações que visam utilizar os recursos de milhares de computadores para prover um poder de processamento intensivo através da formação de uma grade. Recentes projetos têm estimulado o interesse, o projeto SETI@home (Seti@Home, 2008), por exemplo, possui um poder computacional de aproximadamente 25 TFlops/s (trilhões de operações de ponto flutuante por segundo), coletado de mais de três milhões de computadores conectados à Internet. Esse projeto visa utilizar essa grande capacidade de processamento para analisar os sinais obtidos a partir do rádio-telescópio do observatório de Arecibo à procura de algum sinal inteligente de vida extraterrestre. Outros exemplos são o OurGrid (Andrade et al., 2008), InteGrade (InteGrade, 2008) e Genome@Home (Genome@Home, 2008).

3.4.3 Troca de Mensagens

A necessidade de troca de mensagens instantâneas é muito grande atualmente. Muitos sistemas P2P de compartilhamento de arquivos também possuem o recurso dos usuários se comunicarem, mesmo não sabendo sua identidade. Outros sistemas permitem mensagens de voz, vídeo, troca de arquivos e integração dos sistemas com sistemas de mensagens SMS (*Short Message Service*). Exemplos de aplicações desse tipo são *MSN Messenger Live* (MSN, 2008), *Google Talk* (GoogleTalk, 2008), *Yahoo Messenger* (YahooMessenger, 2008), *Jabber* (Jabber, 2008), ICQ (ICQ, 2008), Skype (Skype, 2008) e LawsTalk (LawsTalk, 2008).

3.4.4 Transmissão de Dados

Também conhecidos como sistemas *overlay multicast*, formam uma infraestrutura de comunicação baseada em *multicast* em nível de aplicação. Possibilitam que conteúdo seja distribuído a um número potencialmente grande de nós dispersos geograficamente. Normalmente essa tecnologia é empregada para a transmissão de eventos ao vivo (*live streaming*). Pode-se citar como exemplo o *End System Multicast* (ESM, 2008).

3.5 Redes P2P *versus* Redes *Overlay*

Uma rede *overlay* é uma rede "virtual" criada sobre uma rede existente, por exemplo, a Internet. A rede *overlay* cria uma arquitetura com nível mais alto de abstração, de modo a poder solucionar vários problemas que, em geral, são difíceis de serem tratados no nível dos roteadores da rede subjacente (Andersen et al., 2001). A própria Internet pratica o paradigma *overlay* quando usa o protocolo IP como solução de *internetworking* sobre tecnologias de redes de várias tecnologias, tais como ATM (*Asynchronous Transfer Mode*), *Frame Relay*, MPLS (*Multiprotocol Label Switching*), *Ethernet*, etc.

Numa perspectiva de alto nível, uma rede P2P pode ser considerada uma rede *overlay*, uma vez que funciona como uma rede virtual, formada pela interconexão dos nós, executando sobre a infraestrutura de uma rede física. A Figura 3.3 mostra a topologia de uma Rede *Overlay*. Percebe-se que as conexões entre os nós da rede *Overlay* sobrepõem as interconexões da rede IP, onde a simples comunicação direta entre dois nós, é viabilizada por um complexo mecanismo de roteamento IP.

3.6 Arquitetura das Redes P2P

As redes P2P podem ser organizadas sob diversos aspectos, e embora todas elas possuam problemas de segurança em comum, cada arquitetura possui seus próprios problemas. Aqui são apresentados os modelos arquiteturais comuns das

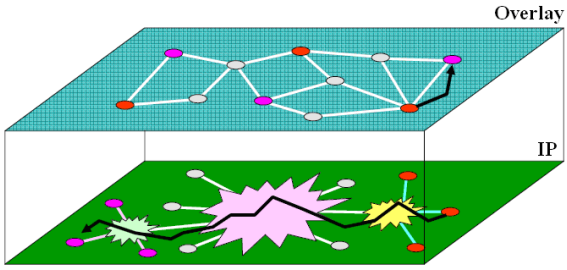

Figura 3.3: Rede *overlay*.

redes P2P, mostrando as vantagens e os problemas normalmente encontrados. Esses tipos de redes podem ser divididos basicamente de duas formas: com relação à centralização ou não da rede, e com relação à estruturação da busca de recursos (Do Carmo et al., 2004).

Com relação à sua distribuição, podem ser organizadas em três tipos: centralizadas, descentralizadas ou descentralizadas com uso de supernós. As redes centralizadas possuem um nó central que realiza alguma função especial, como controle de acesso ou busca de recursos. Redes descentralizadas são as chamadas redes P2P puras, nas quais cada nó tem exatamente as mesmas funcionalidades que os outros, isto é, tem um papel de cliente e servidor. As redes com supernós possuem nós com maior capacidade de armazenamento e processamento, que são eleitos para realizar funções especiais, como repositório para busca de recursos. A diferença em relação às redes centralizadas é que esses nós podem se tornar supernós somente por um determinado período de tempo e a rede deve ser capaz de funcionar mesmo que nenhum supernó esteja presente. A Figura 3.4 ilustra bem essas redes. Veem-se nós conectados diretamente e nós conectados e gerenciados por um ou mais nós centrais.

3.6.1 Redes P2P Centralizadas

Do ponto de vista de segurança, as redes P2P centralizadas são as que possuem melhores condições de se tornarem seguras. Isto porque a entidade centralizadora se responsabiliza pela maior parte da segurança da rede. Embora possa parecer que redes P2P centralizadas não possuam riscos grandes à segurança, alguns pontos devem ser observados para que se obtenha uma rede P2P segura.

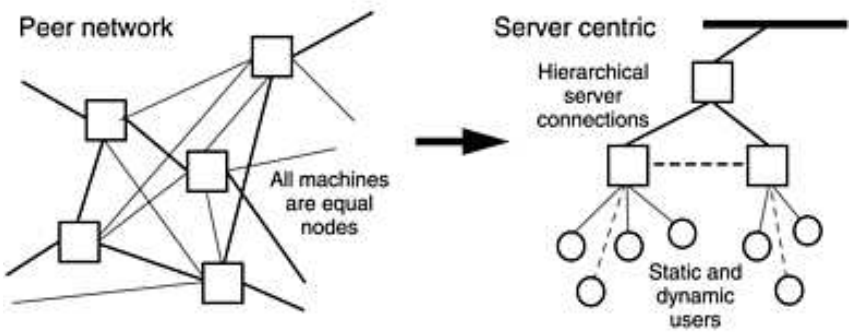

Figura 3.4: Rede P2P descentralizada *versus* centralizada. (Addison, 2002)

Normalmente, uma rede P2P centralizada pressupõe um nó central que é responsável por algumas funcionalidades ausentes nos outros nós, sendo assim, redes centralizadas possuem uma organização hierárquica onde todos os nós dependem de um ou mais servidores. Por exemplo, no caso do Napster, a rede possui um nó central que realiza as buscas de conteúdo, controle de ingresso e conexão à rede. Sem esse nó, a rede não funciona. É interessante destacar que o nó central pode ser representado por um ou mais servidores em *cluster*. O nó central é o ponto alto da hierarquia do sistema, sendo todas as pesquisas feitas ao servidor central, porém os recursos ficam presentes nos nós. Assim, o servidor passa a referência de onde está o recurso e o nó requisitante se conecta diretamente para fazer a transferência. A Figura 3.5 (a) mostra esta estrutura.

Em redes P2P, o nó central é responsável pela distribuição de identificadores únicos, distribuição de chaves públicas, cálculo de reputação de nós, entre outros serviços que ajudam a tornar a rede segura. Entretanto, um nó central não pode assegurar algumas questões, como garantir a inexistência de conteúdo malicioso contido em informações supostamente confiáveis. Como as informações não trafegam pelo servidor, ele não é capaz de garantir a qualidade dos dados. Porém, apesar da maior segurança em redes P2P centralizadas, o fator que mais dificulta o crescimento dessa arquitetura é sua escalabilidade, já que a rede fica restrita à capacidade do nó central. Além disso, a rede fica totalmente dependente da operabilidade desse

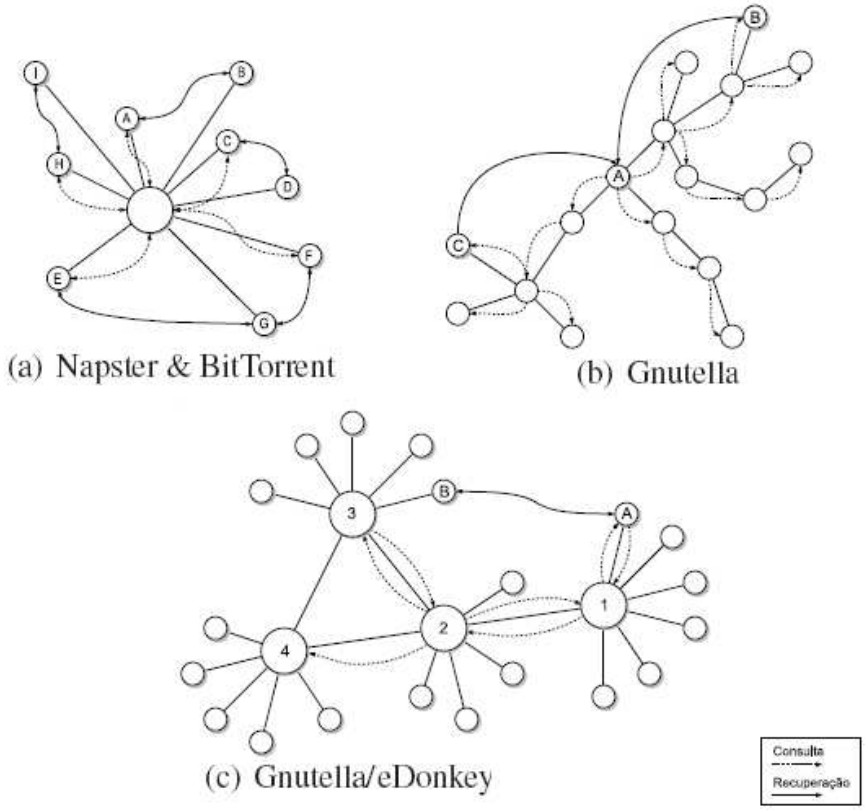

Figura 3.5: Arquiteturas de redes P2P: **(a)** Centralizada com busca centralizada; **(b)** Descentralizada com busca desestruturada; **(c)** Híbrida com busca descentralizada estruturada (Barcellos and Gaspary, 2006).

nó. Mesmo que nós auxiliares do nó central sejam implantados, a rede ainda terá problemas sérios de escalabilidade, já que a capacidade de provimento de recursos de um nó é limitada. Com vários nós auxiliares a rede se descaracteriza e deixa de ser considerada uma rede centralizada. A utilização de *cluster* de servidores, como no Napster, é uma ótima solução, mas torna a estrutura cara e com necessidade de uma equipe para administrá-los.

3.6.2 Redes P2P Descentralizadas

Redes P2P descentralizadas não devem possuir nenhum nó com qualquer diferença de funcionalidade dos demais, ou seja, cada nó é igual ao outro do ponto de vista de suas funções no sistema. Este fato ajuda na escalabilidade do sistema, que não necessita de pontos realizando funções especiais como controle de acesso ou busca de recursos. Outro ponto positivo é a tolerância a falhas que esse modelo possui, já que nenhum nó é essencial ao sistema. Pode-se notar nesse esquema que não há qualquer forma de hierarquia entre os nós, já que um modelo hierárquico pressupõe controle dos nós inferiores pelos nós superiores. Modelos hierárquicos normalmente possuem facilidades de segurança. Em modelos hierárquicos, os pontos superiores realizam certas funcionalidades de segurança responsáveis por garantir a segurança de toda a rede. Além disso, os nós superiores podem monitorar as ações dos nós inferiores para que eles se comportem de maneira aceitável dentro do sistema. As redes P2P descentralizadas têm sua segurança muito mais complexa que redes hierárquicas, já que, nessas redes, nenhum nó pode ser capaz de garantir a segurança. Esse modelo é chamado de rede P2P pura. A Figura 3.5 (b) mostra este tipo de arquitetura.

Podem-se destacar dois modelos de segurança para redes P2P descentralizadas. No primeiro, a rede como um todo tenta se manter segura com a ajuda de todos os nós colaborando. Esse é um bom modelo, desde que a rede consiga minimizar ações de nós que, porventura, tentem corromper ou desvirtuar esse trabalho, ou seja, esse modelo normalmente é vulnerável a ataques internos, promovidos principalmente por nós inseridos na rede com o código do protocolo alterado para gerar diversos ataques, tais como, inundação de requisições, respostas falsas, adulteração de pacotes para que eles percorram mais nós, etc. Para que o modelo de segurança funcione, a rede deve ser capaz de identificar e ignorar nós internos maliciosos. Já contra ataques externos à rede, esse modelo funciona bem, já que ataques à segurança de um único nó na rede deve notificar toda a rede contra a ameaça.

Outro modelo para segurança de redes P2P descentralizadas é quando cada nó tem sua própria política de segurança. Um bom exemplo disso é o trabalho

(Kamvar et al., 2003). Nesse trabalho, cada nó define o quanto pode e deseja confiar nos outros nós. Esse nó só será afetado por um problema de segurança se confiar em um nó malicioso. O problema desse modelo com relação ao anterior é que ele é menos eficiente contra ataques externos, embora seja bem mais eficiente contra ataques internos.

3.6.3 Redes P2P com Supernós

Supernós podem ser uma solução ao problema da escalabilidade das redes P2P centralizadas. Se forem usados diversos supernós em pontos estratégicos, uma rede P2P pode crescer sem maiores problemas de escalabilidade. Apesar disso ser um ponto positivo, existem fatores que comprometem esse modelo. Um fator importante com relação ao uso de supernós é o controle desse uso. Na rede Kazaa, qualquer nó pode se tornar um supernó. Essa liberdade dada aos nós da rede tem diversas consequências, tanto positivas quanto negativas. Um ponto positivo é que os nós se tornam mais autônomos para decidir sobre suas obrigações com a rede. Outra vantagem é um ganho de desempenho no sentido de que, se existe controle, necessariamente deve existir também uma troca de mensagens para que o controle funcione. Já um ponto negativo é que alguns nós pouco capazes podem encarregar-se de obrigações e diminuir o desempenho da rede, pois nós com pouco poder de processamento não conseguiriam atender às obrigações em um tempo aceitável. Quando existe um controle de acesso ao grupo de supernós, uma série de ações devem ser tomadas. Uma delas é estabelecer as condições que são submetidas aos nós para se tornar um supernó. Isso demanda troca de mensagens entre os nós, consequentemente aumentando o tráfego da rede. Outro ponto que pode ser levado em conta é a localização geográfica dos nós. Se o controle levar em conta esse fator, já estará em vantagem em relação ao modelo anterior, pois se os nós se tornam supernós por uma simples autoproclamação, podem ocorrer regiões da rede que estejam superlotadas de supernós e outras sem nenhum supernó ao redor. A Figura 3.5 (c) ilustra este tipo de rede.

3.7 Tipos de Busca em Redes P2P

A busca de recursos nas redes P2P pode ser feita de três modos (Gupta, 2003). No primeiro, a busca centralizada, acontece quando um nó central realiza as buscas e armazena os dados necessários. Este modelo não existe em redes P2P puras. O segundo é a desestruturada, a qual um nó que busca determinado recurso faz uma pergunta aos nós mais próximos e estes ou respondem afirmativamente ou repassam a pergunta a outros nós. Nesse modelo a busca é mais simples de ser implementada, porém pode se tornar menos efetiva e mais lenta, já que nós muito distantes têm dificuldades de se comunicarem. O outro modelo, chamado de estruturado, ocorre quando uma rede P2P como um todo mantém uma tabela *hash* distribuída, ou DHT (*Distributed Hash Table*), com a localidade dos recursos. Nessa arquitetura, cada nó armazena uma parte da DHT. Os nós também são responsáveis por manter a DHT atualizada. Essa arquitetura é mais elaborada e exige maiores cuidados com segurança. Apesar disso, se for bem utilizada, leva a uma melhor utilização dos recursos de uma rede P2P, caso não exceda na utilização da largura de banda para operações de manutenção da tabela.

3.7.1 Busca Centralizada

A busca centralizada caracteriza-se pela existência de um nó responsável pelas buscas de todo o sistema. Quando algum nó da rede deseja procurar por um recurso ele encaminha um pedido de busca ao nó central e este pesquisa se o recurso existe e onde se encontra o recurso. É importante diferenciar esse modelo de busca com a arquitetura centralizada. Na arquitetura centralizada o nó central não é responsável apenas por realizar as buscas do sistema. Por exemplo, em uma rede P2P centralizada o nó central presente pode realizar o controle de acesso ao sistema e a rede pode realizar uma busca desestruturada. Nesse exemplo, a arquitetura é centralizada, mas a busca não. Um exemplo de rede P2P que utiliza este mecanismo de busca é o Napster.

3.7.2 Busca Descentralizada Desestruturada

A busca descentralizada desestruturada se caracteriza por, quando algum ponto do sistema deseja procurar algum dado na rede, o nó faz uma pergunta a seus nós vizinhos. Se algum deles tiver o dado, responde ao requisitante. Se não tiver, os nós repassam a requisição a outros nós, sendo que esse pedido tem um valor finito de nós que deve percorrer. Esse valor finito serve para não propagar indefinidamente pedidos pela rede. Como uma rede P2P descentralizada normalmente não é capaz de saber seu tamanho e organização estrutural, esse valor finito é necessário. Porém, isto acarreta numa perda de funcionalidade, pois dois nós podem estar separados por um valor muito alto de saltos que se tornam inalcançáveis um ao outro. Exemplo de rede que utiliza esse tipo de busca é Gnutella v0.4.

3.7.3 Busca Descentralizada Estruturada

O outro tipo de busca descentralizada é a busca estruturada, feita pelo uso de uma tabela *hash* distribuída. Essa tabela *hash* é dividida pela rede de forma que, cada nó, dependendo de sua colocação no sistema e de seu identificador, fica responsável por uma parte dessa tabela. Quando um determinado recurso deve ser colocado na tabela, o nó possuidor do recurso deve notificar o nó responsável por armazenar a posição em que esse recurso deve ser indexado. Se algum outro nó deseja requisitar o recurso, ele faz uma pesquisa na DHT para saber quem possui o recurso. Como o algoritmo de busca na DHT é conhecido por todos os nós, é mais rápido para o requisitante encontrar o nó que armazena a porção da DHT na qual se deve encontrar as informações do possuidor do recurso, já que a busca é direcionada. Assim, o nó requisitante descobre onde está localizada a informação que deseja dentro da DHT. Este modelo tem a vantagem de proporcionar uma busca direcionada aos recursos de uma rede P2P. Apesar disso, deve haver uma colaboração total de cada nó da rede para o bom funcionamento do sistema. Se, por exemplo, um nó se recusa a responder às requisições sobre sua parte cabível da DHT, os recursos nela listados estarão inalcançáveis mesmo para os nós vizinhos ao detentor da informação. O uso de DHT pode ser também encontrado em redes

P2P desestruturadas com supernós. Os supernós seriam responsáveis pela DHT, enquanto que os outros nós apenas se encarregam de enviar as informações à DHT e a utilizarem para buscas. Embora o uso de supernós diminua a quantidade de nós em que a DHT está presente, isso não acarreta na mudança da política de segurança da DHT. A rede eDonkey, Gnutella v0.6 e Kazaa utilizam este mecanismo de busca.

3.8 *Firewalls versus* Redes P2P

A escolha de serviços web para o protótipo da rede P2PWSRep foi motivada por dos fatores. O primeiro é a facilidade de implementar a interoperabilidade com outras redes P2P, além de facilitar a extensibilidade do protocolo, sem muitos impactos em versões anteriores. O segundo é o fato das redes P2P utilizando serviços web poderem atravessar elementos de proteção como *Firewalls* e *Proxies*. Um nó em uma rede interna não pode ser acessado por um nó da rede externa. Para isso, as conexões têm que ser criadas de dentro para fora da intranet, devido à necessidade de realização de NAT (*Network Address Translation*) para que os pacotes gerados por esses nós acessem a Internet. Assim, o uso combinado de *firewalls* e NAT acaba resultando em uma especial dificuldade para a comunicação P2P. Segundo (Wilson, 2004), na maioria das redes internas, o protocolo HTTP é geralmente habilitado para iniciar conexões. Infelizmente, cada conexão HTTP envia um pedido e aguarda a resposta, sendo assim quando uma conexão faz um pedido inicial de comunicação terá que ficar aberta até receber uma resposta. Embora o HTTP possa prover mecanismos para enviar pedidos da rede interna, não provê a capacidade para os nós externos cruzarem o limite dos *firewalls*.

A única forma que um nó dentro de uma LAN (*Local Area Network*) possui para resolver alguns destes problemas é sua capacidade para criar conexões de rede em um nó fora da proteção de *firewalls*. Nós podem usar protocolos permitidos pelo *firewall* para criar um túnel para a rede externa (*HTTP Tunneling*). Porém, se um *firewall* é configurado para negar todas as conexões que partem da rede interna, a comunicação do nó se torna impossível, mas, com um protocolo de rede que usa a

porta 80, já que um *firewall*, por padrão, não tem como fazer o bloqueio.

Para solucionar este problema, um nó de rota, como é chamado, dentro de uma rede interna protegido por um *firewall* utiliza um nó localizado fora da proteção do mesmo, como mostrado na Figura 3.6. Qualquer nó da Internet ao tentar se comunicar com um nó da LAN não conseguirá, por restrições do *firewall*. Para resolver o problema, o nó 1 faz uma conexão ao nó 4, para manter uma conexão. Assim, as requisições podem ser enviadas para qualquer nó da LAN utilizando o nó 1 como rota de entrada e todas as mensagens direcionadas para os nós internos são liberadas para comunicação.

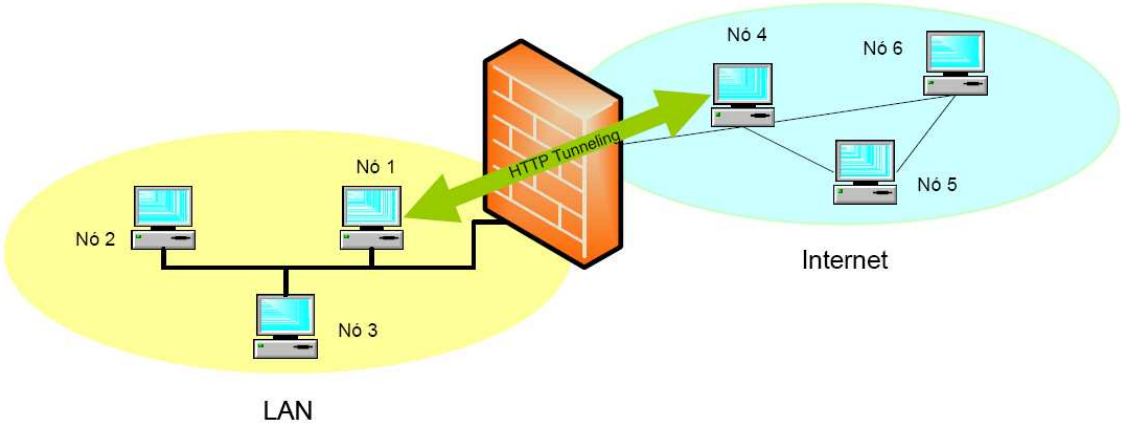

Figura 3.6: Travessia de *firewall* com *HTTP Tunneling*.

3.9 Segurança em Redes P2P com Reputação

Nesta seção, trataremos de alguns ataques a que as redes P2P são suscetíveis. O primeiro ataque é conhecido como *whitewashing* e apenas ocorre quando nós podem trocar sua identidade facilmente, o que é o caso de muitas redes P2P. Um nó pode deixar a rede e voltar em seguida com uma nova identidade, em uma tentativa de se livrar de qualquer reputação ruim que ele tenha acumulado. Se um nó não consegue distinguir um novo nó de um antigo, então *whitewhashers* podem causar o colapso do sistema se nenhuma contramedida for tomada (Feldman et al., 2004). Além disso, o nó pode adquirir múltiplos identificadores para si, caracterizando

um segundo tipo de ataque, chamado de *Sybil*, conforme (Douceur, 2002). Isso possibilita o nó utilizar as várias identidades para agir de acordo com seus interesses.

O terceiro tipo de ataque é o de conluio contra sistemas de reputação. Este tipo de ataque é frequentemente efetivo, porque em sistemas de reputação típicos, um nó deve consultar outros nós sobre a reputação de um terceiro. Se muitos nós estão comprometidos, então este grupo pode fornecer falso testemunho, no sentido de aumentar a reputação de um nó malicioso ou de atacar um nó íntegro, diminuindo a sua reputação.

O quarto tipo de ataque é o de nó traidor (Marti and Garcia-Molina, 2006). Em tal ataque, um nó se comporta adequadamente por um tempo de forma a construir uma boa reputação, e então explora o sistema valendo-se da mesma. Este ataque é especialmente efetivo quando os nós ganham privilégios à medida que conquistam reputação. Um exemplo do ataque de traidor é quando um usuário no eBay (eBay, 2006) constrói uma reputação com muitas transações de pequeno valor, e então lesa alguém em uma transação de grande valor. Em termos sistêmicos, um nó traidor pode surgir não de uma mudança comportamental de um usuário, mas de uma mudança no ambiente: por exemplo, uma máquina cliente perfeitamente correta pode ser infectada com um vírus estilo Cavalo de Tróia, que então poderia aleatoriamente abusar da boa reputação do nó. A resistência a esse tipo de ataque pode ser aumentada usando a análise da história recente de um nó (Feldman et al., 2004).

Por fim, a reputação de um nó é tipicamente usada na política de seleção de um nó com quem interagir. Segundo (Dumitriu et al., 2005), é caracterizado o impacto de políticas de seleção entre respostas adotadas por nós P2P que originam buscas. São identificadas diversas políticas, dentre as quais incluem-se escolher o nó que anuncia a melhor capacidade. Sob o ponto de vista de ataques, a pior política de escolha de um nó é aquela que seleciona o nó que se anuncia como o melhor. Por exemplo, se uma informação de desempenho é facilmente falsificável, uma busca só terá sucesso quando nenhuma resposta for de um nó malicioso, pois um nó malicioso sempre será escolhido. Sistemas de reputação não conseguem resolver esse problema

mesmo quando os erros do sistema de reputação são mínimos. Técnicas baseadas em aleatoriedade são efetivas para aumentar a resistência de um sistema P2P a ataques. Entretanto, aleatoriedade impacta negativamente no desempenho quando atacantes não estão presentes.

3.10 Soluções de Redes P2P

Nas próximas seções serão apresentadas duas redes P2P bem populares que reúnem as principais características dos tópicos abordados nas seções anteriores: a rede Gnutella, que até hoje serve como base para a criação de outras redes que desejam características de redes P2P puras, e a rede Napster, que atingiu uma marca de milhões de usuários no 2000.

3.10.1 Gnutella

É um sistema de compartilhamento de arquivos de topologia *ad-hoc*. Sua arquitetura é descentralizada, isto é, pura. Seu mecanismo de busca é descentralizado e desestruturado, sendo todos os nós funcionalmente idênticos. Os nós nessas redes também são chamados *servents*, porque são servidores e clientes ao mesmo tempo e as buscas por arquivos são realizadas através de uma inundação de escopo limitado, utilizando um campo no cabeçalho, chamado de TTL, que limita o número de vezes que o pacote é encaminhado (Buford et al., 2009). Os nós que possuem o arquivo solicitado respondem ao nó que encaminhou a solicitação, sendo que a resposta à busca também segue o mesmo caminho da solicitação. O nó requisitante então escolhe um dos nós que retornaram a resposta e faz o *download* diretamente deste nó. O Gnutella, pelo seu bom desempenho, e apesar de utilizar mecanismos de busca por inundação, consegue atender satisfatoriamente às características de desempenho das redes que seguem este modelo, considerando-se o tempo de resposta das buscas de recursos.

A Figura 3.7 mostra um exemplo de arquitetura Gnutella, onde um nó faz uma inundação para localizar um arquivo. Os nós que não possuem a informação

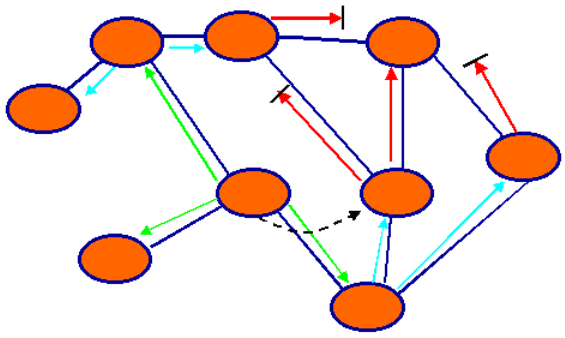

Figura 3.7: Rede Gnutella.

buscada reencaminham a busca para seus vizinhos. Perceba que há setas com uma barra, representando que a busca já atingiu a profundidade especificada. Uma vez encontrado o recurso, o nó requisitante faz *download* diretamente de um dos nós que responderam positivamente.

Visão Geral do Protocolo Gnutella v0.4

Um nó ***X*** estabelece uma conexão com o nó ***Y***, que já está conectado à rede. A conexão se dá da seguinte forma, o nó ***X*** envia uma requisição contendo a *string* "GNUTELLA CONNECT/0.4/n/n" para estabelecer uma conexão com o nó ***Y***. O nó ***Y*** pode aceitar a conexão, enviando para o nó ***X*** a *string* "GNUTELLA OK/n/n", ou recusar a conexão, enviando qualquer resposta que não seja a que contenha a *string* de aceitação de conexão. Uma vez que o nó ***X*** está conectado a rede, ele pode se comunicar com outros nós enviando e recebendo mensagens gnutella, também chamadas de descritores. Cada descritor possui 22 *bytes* de cabeçalho consistindo dos seguintes campos: *Descriptor ID* (*byte* 0 a 15), *Payload Descriptor* (*byte* 16), TTL (*byte* 17), *Hops* (*byte* 18) e *Payload Length* (*byte* 19 a 22). O *descriptor ID* é um identificador único da mensagem na rede e tipicamente criado usando um gerador randômico. O *time to live* (TTL) é o número de vezes que a mensagem será encaminhada pelos nós antes de ser excluída. *Hops* representa o número de vezes que a mensagem será encaminhada entre os nós. Cada vez que a mensagem é encaminhada o TTL da mensagem é decrementado em um e o *hops* é incrementado

em um. O *payload descriptor* contém o código do tipo da mensagem. Os código são **0x00** para *Ping message*, **0x01** para *Pong message*, **0x40** para *Push message*, **0x80** para *Query message*, **0x81** para *Query_Hit message*. O *payload length* contém o tamanho da mensagem que segue o cabeçalho, isto é a carga útil. Então, o total da mensagem é 22 *bytes* + o *payload length*.

Um nó pode descobrir a informação (endereço IP e número de porta, número de recursos, etc.) sobre os outros nós enviando uma mensagem *Ping*. Uma mensagem *Ping* tem um *payload length* zero. Ao receber uma *Ping message*, um nó irá encaminhá-la para todos os nós diretamente ligados a ele, exceto o nó que enviou a mensagem. Ao mesmo tempo o nó irá enviar uma mensagem *Pong*, com o mesmo *descriptor ID* correspondente à mensagem *Ping*, em resposta ao nó que enviou a mensagem. A mensagem *Pong* tem como conteúdo o número de arquivos compartilhados, o número de kilobytes compartilhados, o endereço IP e o número da porta de conexão.

Enquanto as mensagens *Ping* e *Pong* são usadas para descobrir outros nós na rede, mensagens *Query* e *Query_Hit* são usadas para localizar recursos. Para localizar um recurso, um nó envia uma mensagem *Query* que contém como conteúdo a velocidade de rede mínima exigida dos nós, que possam conter o recurso e a *string* que descreve o recurso requisitado. Similar ao processo de mensagem *Ping*, uma mensagem *Query* é recebida por um nó é encaminhada em *broadcasting* para todos os nós conectados diretamente a ele, exceto ao nó que enviou a mensagem. As mensagens *Query* são também sujeitas ao incremento do *hop* e decremento do TTL para limitar a profundidade da busca. Cada mensagem será replicada e propagada na rede até que um nó atenda a requisição ou que o valor do TTL seja excedido para todos os pacotes. Os nós que são capazes de oferecer o recurso respondem com a mensagem *Query_Hit*. A mensagem *Query_Hit* contém a informação sobre o nó que compartilha o recurso buscado.

Mensagens *Pong* e mensagens *Query_Hit* são roteadas pelo mesmo caminho de volta por onde as mensagens *Ping* e *Query* passaram. Isto é feito através de um registro mantido nos nós por onde essas mensagens passaram através do *descriptor*

ID da mensagem e o *payload descriptor* do cabeçalho correspondente. Quando um nó recebe uma mensagem de resposta, ele irá checar se ela foi gerada de um *Ping* ou *Query* que ele encaminhou. Se o nó não é o destino, ele irá checar se foi encaminhado um *Ping* ou *Query* que tem o *descriptor ID*. Se não houve um *Ping* ou *Query* passando por ele, a mensagem é excluída, senão ela será enviada para o destino. Uma vez que um nó destino recebe uma mensagem *Query_Hit*, gerada por outro nó, em resposta a sua mensagem *Query*, ele irá acessar o recurso usando o protocolo HTTP.

3.10.2 Napster

Precursor em termos de compartilhamento de arquivos, o Napster foi a rede P2P mais conhecida, principalmente devido a problemas legais, sendo que até hoje é tido como o principal disseminador do ideal das redes P2P. Apesar disso, o Napster vai contra os princípios das redes P2P pura, porque depende de um servidor central para seu funcionamento. O Napster obteve enorme sucesso ao permitir o compartilhamento de arquivos de música, porém o conteúdo publicado no Napster era protegido por *copyright* e não poderia ser copiado por outros usuários, sendo este o principal motivo de interrupção da sua rede P2P de uso aberto. Usuários que ingressam na rede Napster oferecem conteúdo enviando informações sobre arquivos locais ao servidor central; operações de busca são resolvidas no servidor, que retorna ao nó requisitante uma lista de endereços dos nós disponibilizando o arquivo procurado. O *download* do arquivo então se dá diretamente entre os nós envolvidos, sem sobrecarregar o servidor. A arquitetura do Napster está ilustrada na Figura 3.8 (Addison, 2002). Computadores de usuários buscam informações sobre arquivos no diretório central e então se comunicam com outros nós diretamente para obter os arquivos desejados. De acordo com estudos, a rede Napster chegou a contar com mais de 50 milhões de usuários em 2001 (OpenNap, 2008).

A arquitetura da rede Napster, apesar de usar um modelo centralizado, demonstrou que é possível atingir um grau de escalabilidade grande, comprovado pelo grande número de usuários que a rede atingiu. Isso se deve principalmente ao tipo

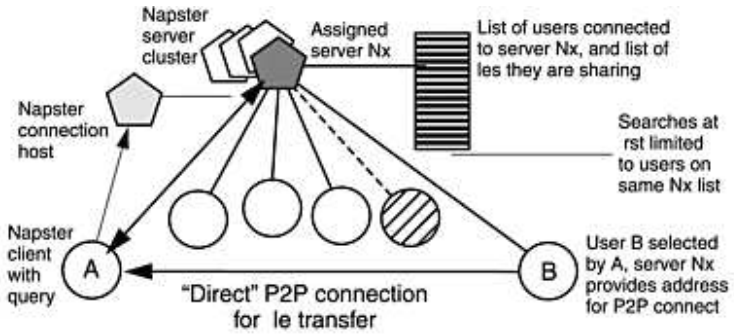

Figura 3.8: Arquitetura do Napster (Addison, 2002).

de infraestrutura tecnológica adotada na parte que se refere ao servidor central. Na verdade o Napster possuía vários servidores, configurados em *cluster*. Isso dá ao usuário a impressão que existe uma única entidade servidora, mas na verdade as requisições são distribuídas para um dos membros do *cluster*. Isso torna um modelo de rede P2P complexo de administrar e consideravelmente caro. Porém, mostra a simplicidade do protocolo de registro de nós, de busca de recursos e *download* de arquivos.

O funcionamento da rede Napster é bem simples. Tudo começa com o registro do nó na rede. O registro é feito em um dos servidores de metadados. Cada servidor tem capacidade para atender até 15.000 nós simultaneamente. O nó, ao se registrar no servidor, recebe uma identidade e passa os dados dos recursos que compartilha para o servidor. Cada vez que o cliente entra na rede, ele se conecta ao servidor e atualiza vários dados, tais como, os recursos disponíveis e o endereço IP corrente. Um nó quando necessita de um recurso, abre uma conexão com o servidor e informa o nome do recurso que está buscando. O servidor varre suas tabelas de metadados em busca do recurso, acha os nós que dispõem do mesmo e responde ao nó requisitante. O nó recebe os dados que identificam os nós que contêm o recurso e os endereços IP correspondentes, então se conecta diretamente ao nó destino e faz o *download* do arquivo. As redes P2P com este tipo de arquitetura têm uma baixa latência nas buscas, no caso do Napster o tempo de resposta dificilmente ultrapassava 15 segundos (Addison, 2002).

3.11 Considerações Finais

O pontencial das redes P2P foi revelado nos últimos anos pela grande demanda a esta tecnologia, principalmente na área de compartilhamento de recursos. Esse sucesso se deve a diversas formas de projetar uma rede P2P, pela variedade de arquiteturas, mecanismos de busca, transmissão de dados e aplicações que se adequam a este modelo, permitindo que uma rede P2P seja desenvolvida considerando as características da aplicação. O crescimento e popularização dessas redes gera um tráfego intenso na rede mundial de computadores ou nas redes locais onde elas estão presentes, e como qualquer sistema que se sobressai, desperta para criação de outras tecnologias que buscam prejudicar o funcionamento natural para o qual foi proposto, seja para tirar maior vantagem ou simplesmente causar um desequilíbrio, explorando falhas e criando mecanismos avançados de ataques. Porém, as redes P2P têm evoluído para superar estes problemas. Assim, projetar sistemas que tenham como base as redes P2P exige que sejam revistas várias caractarísticas destas redes e inseridos novos mecanismos que possam torná-las mais seguras e confiáveis, mantendo seu desempenho.

Capítulo 4

Reputação em Redes P2P

4.1 Introdução

Estudos (Adar and Huberman, 2000) demonstraram que 70% dos usuários integrantes de redes P2P só estão interessados em utilizar o sistema sem cooperar, isto é, não compartilham os recursos que possuem ou que obtiveram na rede. Outro dado que foi mostrado na pesquisa é que 50% das respostas às buscas são retornadas por apenas 1% dos nós integrantes da rede. Estudos também recentes de (Liang et al., 2005) mostraram que 50% dos arquivos mais populares disponibilizados na rede FastTrack (FastTrack, 2008) são corrompidos. Esses dados demonstram que devem existir mecanismos que possam ajudar a manter um nível de colaboração entre os nós participantes da rede e informações sobre os recursos que estão disponibilizados pelos mesmos.

4.2 Tipos de Reputação

4.2.1 Reputação de Nós

Os mecanismos de reputação aumentam a confiabilidade das redes P2P porque permitem que os nós possam avaliar o comportamento dos demais e então definir de onde obter informações confiáveis. Os mecanismos propostos na lite-

ratura sugerem construir e manter a reputação de um nó através de avaliações enviadas por outros nós à medida que o mesmo interage na rede, bem como validar a informação para saber se a mesma é integra. Este tipo de reputação é muito semelhante à utilizada para classificação de usuários de sites de comércio eletrônico como eBay (Ebay, 2008) e Mercado Livre (MercadoLivre, 2008), onde os membros são classificados levando em consideração o número de produtos vendidos, número de produtos comprados, prazo de entrega, qualificação dos compradores, qualidade dos produtos, etc. Todos os membros podem ser vendedores e compradores, sendo que as transações são feitas diretamente entre eles. O sistema gera um percentual que permite aos clientes escolherem os vendedores pelo escore, mas mesmo assim ficam livres para negociar com novos vendedores, fazendo com que estes vendedores elevem seus escores. Os vendedores também podem, com base na reputação do comprador, rejeitar o pedido de transação de compra devido a uma reputação negativa. A Figura 4.1 mostra a reputação utilizada pelo site Mercado Livre. Perceba que existem várias métricas para avaliar um vendedor: o número de transações, qualificações positivas e negativas e tempo que o vendedor é membro do sistema. Também são utilizados estereótipos para permitir que os usuários assimilem visualmente e rapidamente a reputação de um vendedor.

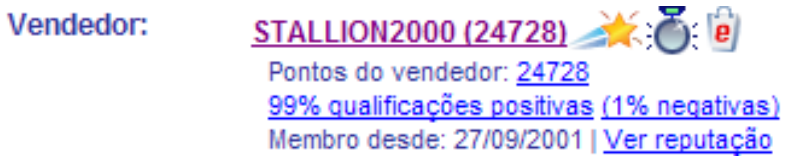

Figura 4.1: Reputação de um vendedor do site Mercado Livre.

O site de vendas americano eBay tem um mecanismo igual ao do site brasileiro Mercado Livre. Ao final de uma transação de compra e venda são realizadas avaliações das partes. O produto também pode ser avaliado da mesma forma que o vendedor e comprador. A Figura 4.2 mostra como é realizado o cálculo de reputação do eBay. O número de transações positivas é dividido pelo somatório de todas as transações ocorridas em um período de um ano, desconsiderando-se as avaliações de

um mesmo membro na mesma semana. O resultado em percentual é a reputação do participante.

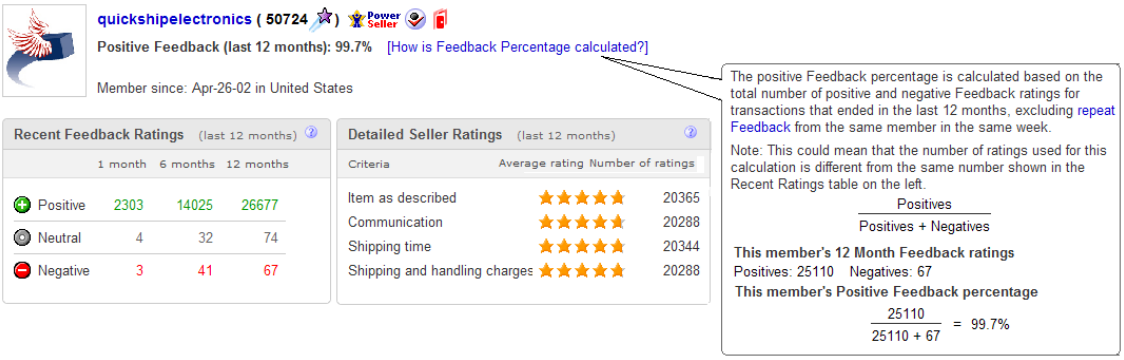

Figura 4.2: Mecanismo de cálculo de reputação do eBay.

A reputação de nós em uma rede P2P pode ser gerenciada de três formas. A primeira é com os nós executando o algoritmo de reputação e mantendo as métricas de todos os outros nós que ele já interagiu ou pretende interagir. Esse modelo é descentralizado. A segunda abordagem é através de um elemento central mantendo uma base de informação de todos os nós. Os nós individualmente enviam seus dados para este servidor e o mesmo executa um algoritmo para identificar a reputação do nó. A terceira e última abordagem é um modelo híbrido, onde as informações são processadas e armazenadas localmente. Neste caso, o servidor central também recebe dados dos nós e os disponibiliza. É um modelo mais complexo, pois envolve uma gerência de reputação tanto nos nós, quanto em um servidor central, havendo a necessidade de manter uma integridade dos dados em ambos os locais.

Os sistemas de reputação permitem manter um histórico de transações praticadas por ou com um nó da rede P2P. Esse histórico possibilita que o nó possa manter um grau de confiança no nó com o qual deseja interagir e que possa responder as requisições de reputação a respeito deste nó com base na sua experiência. Sabe-se que a maioria dos nós que integram uma rede P2P usufruem mais dos recursos do que os disponibilizam. Nós de uma rede P2P que não colaboram podem ser divididos em egoístas e maliciosos. O objetivo de egoístas é usufruir o máximo do sistema P2P contribuindo com o mínimo de recursos compartilhados. Esse comportamento é

muito comum nas redes P2P que não têm uma gerência de reputação. Em contraste, o objetivo dos maliciosos é causar mal a um dos nós ou ao sistema como um todo. Nós maliciosos estão dispostos a empregar recursos para ataque, tal como injetar arquivos corrompidos, o que não ocorre com nós egoístas, responder requisições com adulteração da resposta original, simular juntamente com outros nós, através de interações que não existiram, uma reputação forjada, dentre muitas outras.

Existem diferentes abordagens para se incentivar nós a colaborarem. Destacam-se duas classes gerais de mecanismos de incentivo. O primeiro são os baseados em confiança e reputação, amparados na reciprocidade, isto é, cada nó possui uma reputação associada, que parte de um estado inicial e é construída ao longo da vida do mesmo, com base em suas interações com os outros. Essa informação de reputação pode influenciar a decisão de um nó sobre os parceiros de sua próxima interação, buscando reciprocidade. Por exemplo, um nó pode preferir solicitar um serviço a outro que tem lhe respondido correta e eficientemente as últimas requisições, desconsiderando certos nós que têm lhe retornado arquivos com conteúdo corrompido. Reciprocidade é obtida em um sistema P2P através de um sistema de gerenciamento de reputação. Os termos reputação e confiança estão intimamente ligados (Grandison and Sloman, 2000).

Confiança é a base para permitir que um nó use ou manipule recursos do outro. O grau de confiança em uma operação tem relação inversamente proporcional ao seu risco. Sobre a relação entre os termos, um sistema de gerência de reputação determina a reputação de nós baseado no histórico das ações dos mesmos e permite que sejam formadas opiniões sobre o grau de confiança acerca de outros. As relações de confiança podem ser de um para outro (confiar em um nó para troca de recursos), de um para vários (confiar em um conjunto de nós com os quais recursos podem seguramente ser trocados), de vários para um (nesse caso, todos os nós devem se reportar a um só nó, em que confiam) e de vários para vários (quando um grupo confia em outro). Uma das propriedades mais importantes de confiança é a transitividade: se A confia em B e B confia em C, então é possível que A confie (pelo menos limitadamente) em C.

A confiança de um nó em outro é baseada, naturalmente, na visão do nó sobre a reputação do outro. Os sistemas de reputação têm em comum três partes principais: coleta de informações, determinação de escore e ranqueamento e ações de resposta (Marti and Garcia-Molina, 2006). A coleta de informações está relacionada ao esquema de identificação usado, às fontes de informação, à agregação de informações e à política adotada para nós que ingressam no sistema sem histórico associado. Um dos problemas fundamentais de sistemas de reputação é garantir a validade das informações prestadas por outros nós. Portanto, é natural que na determinação do escore de reputação de um estranho, experiências anteriores do próprio nó sejam valorizadas em relação à opinião de outros.

O segundo esquema de incentivo, baseado em comércio, inclui mecanismos de micropagamento e esquemas de troca de recursos. Para isso, os nós, para obter recursos compartilhados, têm que pagar por eles através de uma moeda virtual que existe na rede P2P ou oferecer seus recursos em troca dos que ele obteve ou pretende obter, isto é, uma espécie de escambo (Righi et al., 2004).

4.2.2 Reputação de Recursos

Com o objetivo de minimizar as ações dos nós egoístas e de nós maliciosos nas redes P2P, será abordado outro problema mais relacionado aos arquivos compartilhados pelos nós. Para um nó, quando um recurso é encontrado e baixado, é feita uma verificação do seu conteúdo. Se não houve detecção de alteração no *hash* do arquivo, então ele é dado como íntegro e a reputação do nó que disponibilizou o arquivo é incrementada. Porém, este arquivo pode conter conteúdo perigoso, tal como *spyware*, cavalo de tróia ou outros tipos de ameaças. Assim, a reputação aplicada somente a nós da rede P2P não é totalmente segura, o ideal é fazer o uso de reputação tanto em nível dos nós quanto dos recursos presentes na rede. Esse problema relacionado aos recursos é conhecido por disseminação de conteúdo poluído em redes P2P. Fazendo uma comparação com sistemas reais, utilizados nos sites de comércio eletrônico, isso equivale à avaliação do produto pelos compradores. Assim, antes de fechar a compra, o interessado pode fazer uma verificação se há opiniões

negativas sobre o produto que o mesmo está pretendendo comprar. A presença de recursos corrompidos nas redes P2P causa um grande desperdício de largura de banda, já que atualmente o tráfego gerado pelos sistemas P2P responde por grande parte do tráfego na Internet.

Cada um dos vários sistemas P2P utilizados atualmente possui um determinado nível de poluição de recursos. Esses níveis podem levar essas redes ao desuso, assim como aconteceu com muitas outras que foram substituídas por redes P2P que ofereciam melhores resultados na obtenção de recursos íntegros. No trabalho de (Costa et al., 2005) foram avaliados dois tipos de redes: moderadas e não moderadas. Nas redes moderadas, existe a presença de um agente moderador humano capaz de interferir nos arquivos que estão sendo compartilhados na rede, como no caso da rede BitTorrent (BitTorrent, 2008). As redes não moderadas constituem os demais tipos de redes. Além disso, foi avaliado qual o efeito da poluição da rede, no espalhamento, quando os usuários do sistema recebem incentivos para apagar arquivos poluídos.

A rede BitTorrent é uma das redes P2P moderadas mais utilizadas atualmente. Percebe-se que quando uma rede P2P tem a participação dos usuários para identificar os recursos corrompidos, a mesma mantém um nível de poluição aceitável pelos usuários. Porém, nota-se que a maioria dos usuários são negligentes, pois apesar de detectar os recursos corrompidos, não os apagam. Isso acaba contribuindo para a queda do desempenho da rede. O mecanismo utilizado pelo BitTorrent consiste na publicação de informações sobre os arquivos disponibilizados na rede. Os usuários que decidem obter este arquivo receberão pedaços uns dos outros, seguindo uma política "olho por olho, dente por dente" (*tit-for-tat*), o que implica que os nós só irão ceder recursos para outros nós que têm tendência a cooperar. Os moderadores consistem de usuários que disponibilizam o arquivo de metadados que contém informações suficientes para um participamente da rede obter o arquivo buscado. Esse moderador pode apagar o metadado caso perceba que aquele arquivo está corrompido (Costa et al., 2005). Existem sites na Internet que disponibilizam arquivos bem recentes de metadados para fontes (pedaços) de arquivos disponibilizados na

rede BitTorrent.

4.3 Sistemas Baseados em Comércio e Reputação

Neste trabalho, serão apresentadas cinco abordagens para diminuir problemas das redes P2P segundo (Barcellos and Gaspary, 2006). Vários desses trabalhos foram utilizados como base para a definição do protocolo P2PWSRep. Os dois primeiros são baseados em comércio, o sistema PPay e redes de Escambo. Porém, esses trabalhos diminuem apenas o alastramento de nós caronas (*freeriders*), mas não evitam que a qualidade dos recursos trocados seja considerada no que se refere à corrupção de dados e infecção por vírus. Os três últimos trabalhos são baseados em reputação. São os sistemas EingenTrust, XRep e PeerTrust. Esses sistemas utilizam algoritmos que controlam o grau de confiança de um nó ou recurso compartilhado através de métricas baseadas nas transações dos nós, que refletem seu comportamento.

4.3.1 PPay

Sistemas baseados em comércio também podem ser chamados de modelo de micropagamentos. São responsáveis por controlar o fluxo de recursos entre os participantes da rede P2P. Baseiam-se na ideia de compra de recursos, onde um nó apenas adquire acesso aos recursos que deseja, se pagar por ele. Para isso, existe uma espécie de moeda, um "dinheiro virtual" adquirido pelos participantes da rede P2P. Essa moeda tem um único motivo: evitar que nós caronas possam usufruir dos recursos disponíveis sem pagar por eles. Isso estimula nós que não queiram investir a deixar a rede. Os recursos podem ter valores diferentes, dependendo do tipo e até mesmo do nó que o hospeda. Porém, isso pode gerar uma "inflação virtual", quando poucos nós possuem o recurso e passam a cobrar caro por eles. Percebe-se que não há uma preocupação em determinar a qualidade do recurso compartilhado e também não há como o participante que adquiriu um recurso reclamar ou divulgar uma negociação frustrada, afim de evitar que outros participantes possam ser vítimas de

novos "golpes".

Muitos esquemas de micropagamentos se baseiam em confiança e são gerenciados por um nó central, chamado de intermediário. Ele é responsável por gerenciar contas de usuários, distribuir e trocar dinheiro envolvido nas transações e manter a segurança do sistema. Além disso, para que os pagamentos ocorram, o intermediário deverá estar disponível para participar em cada transação. A desvantagem desse esquema é a limitação que este nó pode impor ao sistema, comprometendo sua escalabilidade e desempenho, no caso de grande número de participantes e transações.

O protocolo PPay (Yang and Garcia-Molina, 2003) tem um esquema de micropagamento onde os usuários obtém "dinheiro virtual" para utilizar em transações com outros participantes da rede. Cada recurso que um nó deseja adquirir tem um valor estipulado por seu proprietário. O PPay requer somente um intermediário envolvido quando nós desejam abrir ou fechar suas contas, e em alguns casos, para realizar operações em nome de outros nós. Neste protocolo, os nós são responsáveis pelo seu próprio dinheiro, sem necessidade do intermediário.

Para garantir a segurança, o sistema utiliza certificados associados ao dinheiro virtual. Esses certificados são obtidos quando um usuário obtém um crédito virtual e limitam quanto de dinheiro ele está autorizado a utilizar. O esquema de micropagamento consiste em que as transações sejam sustentadas por baixos valores trocados por usuários bem intencionados, mas, se um usuário mal-intencionado desejar fazer um ataque, por mais que ele possa ter adquirido créditos de forma normal, ele gastará rapidamente seu dinheiro virtual, no caso de um ataque de DDoS (*Distributed Denial of Service*), por exemplo.

4.3.2 Sistemas de Escambo

Os sistemas baseados em Escambo, assim como o modelo de micropagamentos, também são responsáveis por controlar o fluxo de recursos entre os participantes da rede P2P, porém ele se baseia na ideia de troca de recursos, onde um nó apenas adquire acesso aos recursos que deseja se possuir outros recursos para disponibilizar no ambiente colaborativo. Dessa forma, participantes que são parasitas (aqueles que

não colaboram com os demais) terão acesso restrito às vantagens que a rede proporciona e, então, são encorajados a saírem da condição de caronas para desfrutarem integralmente a comunidade P2P.

A diminuição dos nós caronas propiciada pelo Escambo gera uma série de vantagens aos sistemas colaborativos. O número reduzido de nós caronas possibilita uma melhor distribuição do tráfego da rede, pois os recursos ficam mais espalhados na topologia. A disponibilidade dos recursos na rede melhora, já que um maior número de informações está disponível para os usuários legítimos. Em alguns protocolos P2P, obtêm-se vantagens também no tempo de resposta das solicitações. Além desses benefícios técnicos, o baixo número de participantes caronas consegue tornar a comunidade P2P mais justa e harmoniosa, colaborando socialmente para o progresso do sistema.

O modelo de escambo modifica a estrutura normal das redes P2P para alcançar os seus objetivos. Nele, um nó que recebe uma solicitação por um recurso que detém, deve antes de permitir o acesso, verificar se o nó requisitante também dispõe de recursos na rede P2P. O Escambo deve oferecer uma sistemática que possibilite reconhecer quais nós são caronas e quais não o são. A política de controle de acesso aos recursos informa quais as condições que o nó cliente deve suprir para acessar os recursos do nó servidor. Divide-se em três categorias:

- Tamanho total dos recursos compartilhados (medido em bytes);
- Número de arquivos oferecidos;
- Tipo de recursos disponibilizados.

O Escambo utiliza um modo simplificado de micropagamento. O micropagamento introduz o conceito de pagamento pelo acesso a um recurso ou pedido de atividade. Esse pagamento pode ser de dois tipos:

- O nó servidor não recebe nenhum valor e o cliente paga-o geralmente com trabalho (cálculo de uma tarefa computacional complexa);
- O cliente utiliza algum sistema de pagamento, por exemplo, dinheiro virtual, para pagar o detentor dos recursos, o qual tem acesso ao montante recebido.

No caso específico do Escambo, o nó cliente paga o servidor através da oferta de recursos na rede P2P. A reputação nos sistemas colaborativos P2P tradicionais informa quais participantes da rede são honestos ou bons servidores de recursos. A reputação dos nós é adquirida através das experiências dos próprios membros da rede e das trocas de informações de reputação entre os pares que confiam um no outro (Damiani et al., 2002). O conceito de reputação encontrado no Escambo distancia-se do mencionado anteriormente, pois a reputação de cada nó está associada à quantidade e qualidade do material que ele disponibiliza na rede P2P e não depende da opinião de outros participantes do sistema (o nó é o único responsável por sua reputação). As arquiteturas P2P, e nesse caso, especialmente o Escambo, devem possuir mecanismos para evitar que o sistema de reputação seja enganado. O Escambo deve estar precavido contra usuários que alteram as mensagens passadas entre os nós em benefício próprio.

4.3.3 EigenTrust

EigenTrust (Kamvar et al., 2003) é um algoritmo para gerência de reputação para sistemas de compartilhamento de arquivos. Cada nó possui associada uma reputação global, que é baseada no histórico de *uploads* de arquivos. A reputação global de um nó $\boldsymbol{i}$ é calculada com base nos índices de reputação atribuídos localmente a $\boldsymbol{i}$ por cada nó $\boldsymbol{j}$, $\boldsymbol{k}$, $\boldsymbol{l}$, etc. e pesados de acordo com a própria reputação daqueles nós. No estudo realizado, a abordagem ajudou a reduzir o número de arquivos corrompidos compartilhados. Os valores de confiança atribuídos por um nó são normalizados. Isso é necessário para evitar que um nó subverta o sistema atribuindo reputações arbitrariamente altas a outros nós maliciosos, influenciando o valor de reputação global em um ataque combinado. Em um nó $\boldsymbol{i}$, a reputação do nó $\boldsymbol{j}$ é normalizada dividindo-se o valor de reputação de $\boldsymbol{j}$ em $\boldsymbol{i}$ pelo somatório de todos os valores de reputação que $\boldsymbol{i}$ mantém. Ou seja, todos os valores de reputação atribuídos por um nó situam-se entre 0 e 1. As desvantagens dessa normalização são duas. Primeiro, não há distinção entre nós novos e má reputação. Segundo, os valores são relativos e, portanto, não podem ser interpretados de forma absoluta.

Por exemplo, se em $\boldsymbol{i}$ dois nós $\boldsymbol{j}$ e $\boldsymbol{k}$ possuem o mesmo valor de reputação $\boldsymbol{r}$, então se sabe que aos olhos de $\boldsymbol{i}$, $\boldsymbol{j}$ e $\boldsymbol{k}$ são igualmente reputáveis, mas não se sabe se ambos têm boa ou má reputação. Para agregar os valores normalizados computados por cada nó, um nó $\boldsymbol{i}$ pergunta aos seus nós vizinhos, contidos em sua tabela de nós, sobre os valores de reputação que eles atribuíram a um nó $\boldsymbol{k}$, e usa uma média ponderada com as reputações deles para calcular a confiança de $\boldsymbol{i}$ em $\boldsymbol{k}$.

Para aumentar o conhecimento, um nó pode pedir a opinião dos vizinhos dos vizinhos, e assim por diante, recursivamente, até atingir a rede inteira. Não é possível permitir que cada nó seja responsável por calcular e reportar sua própria reputação. Portanto, a reputação de um nó é calculada por mais de um nó na rede, e armazenada em outro nó. Múltiplos nós computam o escore de um nó, e uma pesquisa DHT é usada para encontrar tais nós. O algoritmo proposto evita que um nó saiba a identidade do nó para o qual ele está calculando a confiança, de forma que um nó malicioso não possa aumentar artificialmente a reputação de outro nó malicioso. Nós que entram no sistema não podem escolher qual a posição em que entram no espaço de $\boldsymbol{IDs}$, evitando que um nó retorne exatamente na posição do nó responsável por calcular a sua reputação.

Valores globais de reputação podem ser usados então para isolamento de nós maliciosos. Um nó usa a reputação dos candidatos como política de seleção de onde irá baixar um arquivo. Entretanto, uma escolha desse tipo concentra as requisições nos nós com reputação mais alta e não permite que outros nós corretos adquiram reputação. A proposta é usar um esquema onde o nó é selecionado de forma semi-aleatória, com influência da reputação. Segundo (Marti and Garcia-Molina, 2006), EigenTrust oferece uma solução puramente descentralizada, mas usa uma identidade fraca, tornando-o suscetível a ataques de *whitewashing*, que é um tipo de artifício de que um nó se utiliza, saindo da rede e entrando novamente para obter um novo identificador, pois isso permite que a má reputação anterior seja perdida. Além disso, (Cheng and Friedman, 2005) indicam que EigenTrust é suscetível ao ataque *Sybil* (Douceur, 2002), uma vez que um nó malicioso pode criar uma subporção inteira do *grafo* da rede. O ataque *Sybil* permite que um único nó

falsifique a existência de várias identidades dentro da rede. Este tipo de ataque é complexo de ser resolvido em um sistema amplamente distribuído, mas pode ser controlado através da existência de autoridades certificadoras.

4.3.4 XRep

Em (Damiani et al., 2002), propõe-se um esquema para gerência de reputação para a rede Gnutella. Diferentemente de outras abordagens, a reputação de nós é combinada com a reputação de recursos, aumentando a resistência contra ataques do tipo *Sybil*. Reputações são cooperativamente gerenciadas através de um algoritmo distribuído de *polling* de maneira a refletir a visão da comunidade sobre o risco de *download* e uso de um recurso. O protocolo proposto pelos autores estende o protocolo de busca do Gnutella com passos e mensagens adicionais, facilitando a atribuição, o compartilhamento e a combinação de reputações de nós e recursos. O princípio básico do esquema é o protocolo XRep, uma extensão ao protocolo de busca do Gnutella. No Gnutella, um nó inicia uma busca enviando mensagens ($QUERY$) aos seus vizinhos; todos os nós que encontram o objeto requisitado retornam mensagens de resposta através do mesmo caminho pelo qual vieram. Após receber múltiplas respostas positivas (mensagens $QUERY\_HIT$), o nó pergunta a outros nós opiniões sobre os nós que ofereceram o recurso buscado. O processo de *polling* é composto de cinco fases, mostradas na Figura 4.3 e descritas abaixo:

1. **Busca de recursos** - Nesta fase, nas mensagens $QUERY\_HIT$ que são retornadas pelos nós que detêm um ou mais objetos que satisfazem a busca, acrescenta-se um *digest* para cada objeto referenciado na mensagem;

2. **Seleção de recurso e voto por *polling*** - O nó requisitante escolhe o melhor nó dentre aqueles que parecem satisfazer sua busca (ou seja, que responderam). Para tanto, o nó envia uma mensagem a seus vizinhos contendo uma requisição de voto sobre a reputação dos objetos oferecidos e dos nós que os oferecem. Estas mensagens são implementadas usando mensagens convencionais $QUERY$ e contêm uma chave pública a ser usada na cifragem da resposta, para proteger

a integridade e a confidencialidade das respostas. Nós que recebem a pergunta verificam seus repositórios de experiência e respondem;

3. **Avaliação dos votos** - O nó descarta mensagens adulteradas e os autores indicam que um nó faz também um agrupamento e combinação de votos que são oriundos de um mesmo nó para evitar ataques *Sybil*. Após o descarte, o nó seleciona um conjunto de votantes e envia uma outra mensagem de *polling* (*TRUEVOTE*) direto a cada um para que respondam confirmando seus votos, obrigando nós atacantes a usarem IPs reais;

4. **Verificação de melhor nó** - O nó mais confiável é contactado para verificar se ele realmente dispõe do recurso;

5. **Download do recurso** - O nó requisitante contacta o nó que dispõe do recurso e solicita o *download* do mesmo, depois disso ele verifica a integridade do recurso recebido através do *digest* e atualiza seu repositório de experiências.

4.3.5 PeerTrust

PeerTrust (Xiong and Liu, 2004) é um *framework* de reputação que inclui um modelo de confiança adaptativo para quantificar e comparar a confiança de nós baseado em um sistema de transações com *feedback*, e uma implementação descentralizada de tal modelo em uma rede P2P. As duas características principais do PeerTrust são a definição de três parâmetros básicos de confiança e dois fatores adaptativos na computação do grau de confiança em um nó e a definição de uma métrica geral de confiança para combinar esses parâmetros. Os fatores que um nó leva em consideração no cálculo de nível de confiança no PeerTrust são:

- *Feedback* recebido de outros nós, na forma de um valor;
- Escopo do *feedback*, tal como o número de transações que um nó possui com outro;

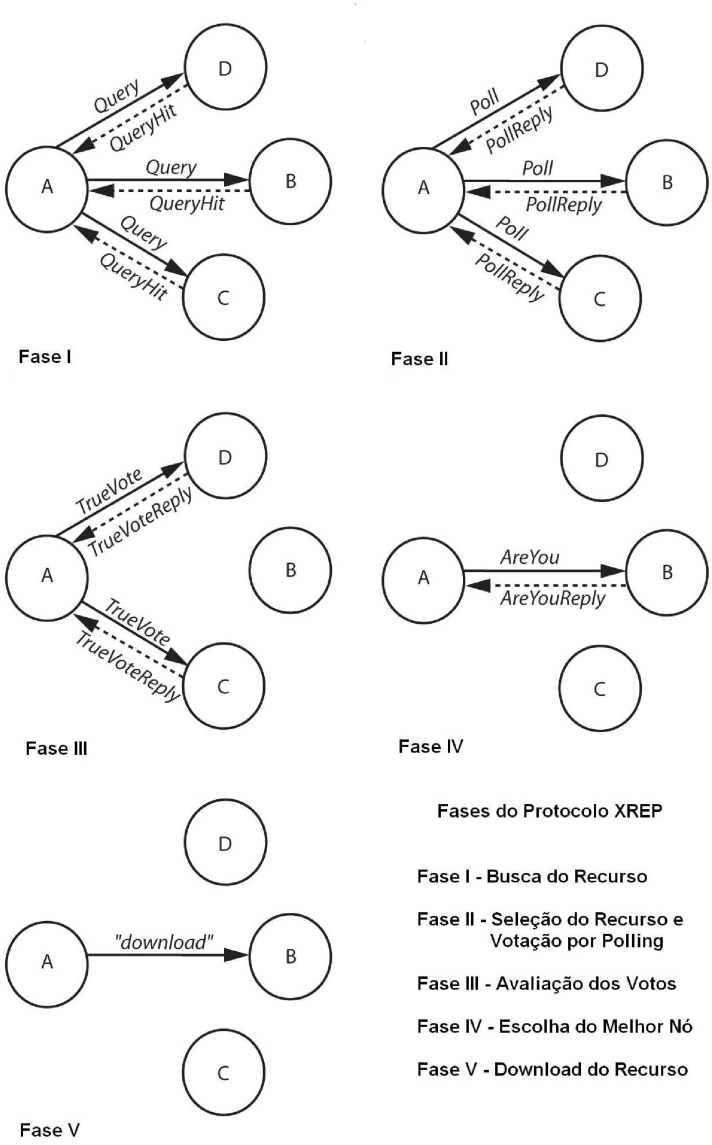

Figura 4.3: Fases do protocolo XREP.

- Fator de credibilidade para um nó que forneceu o *feedback*, diferenciando a qualidade do *feedback* recebido de outros nós de acordo com a confiança nos mesmos;

- Fator de contexto de transação para diferenciar as mais importantes das menos importantes, associando pesos às transações;

- Fator de contexto da comunidade para tratar características relacionadas à comunidade e vulnerabilidades peculiares, por exemplo, criar um incentivo à submissão de *feedback* sobre outros nós.

Para implementação desse modelo de confiança, cada nó possui um gerente de confiança e um localizador de dados. O primeiro é responsável pela submissão de *feedback* e avaliação de confiança através de um banco de dados com um segmento da base global. Já o localizador de dados serve para alocação e localização de dados de confiança na rede P2P (*overlay*).

O gerente executa duas funções principais. A primeira é a submissão do *feedback* para rede através do localizador, que roteia a informação para outros nós. A segunda função é a avaliação do nível de confiança de um determinado nó, que é executado em dois passos. Primeiro, ele coleta informações de confiança sobre o nó em questão através do localizador, e então calcula o valor de confiança. Dois métodos são propostos para cálculo do nível de confiança: cálculo dinâmico de confiança (DTM, *Dynamic Trust Computation*), que emprega informações recentes obtidas sob demanda de todos os outros nós da rede e a computação aproximada de confiança (ATC, *Approximate Trust Computation*), que é mais eficiente, porém menos precisa, pois calcula a confiança por informações presentes em uma *cache*. Cada nó mantém uma *cache* contendo os valores de confiança fornecidos por outros nós com que ele recentemente se comunicou, sendo que isso só é preciso quando ele não encontra um nó em sua *cache*. Essa é uma característica interessante do protoloco PeerTrust, pois diminui bastante o tráfego de requisições na rede.

Quando a reputação de um nó é baseada na média cumulativa de suas transações de seu tempo de vida, e um nó adquiriu uma sólida reputação, as transações atuais têm pouco impacto na reputação e, portanto um nó possui menor incentivo para se comportar de forma honesta. Um nó pode, mesmo assim, oscilar entre um comportamento honesto e desonesto de forma a manter uma reputação razoável, apesar de agir incorretamente em determinadas transações. Os autores propõem um algoritmo simples de janela de tempo deslizante. Os valores de confiança são computados de forma global e recente, e então são comparados. A ideia

é que uma boa reputação seja difícil de atingir, leve tempo para construir, porém possa ser destruída rapidamente após poucas transações incorretas. Para evitar problemas de segurança relativos ao armazenamento e transmissão das informações de confiança, o PeerTrust emprega criptografia com chaves públicas e privadas. Cada nó é obrigado a ter um par de chaves e assinar suas mensagens de *feedback* com sua chave privada, e fornecer a chave pública, garantido a integridade e autenticidade. O identificador de cada nó é um *digest* de sua chave pública, ou sua própria chave pública.

4.4 Considerações Finais

Neste capítulo, foram apresentados os conceitos básicos de reputação em redes P2P e os principais sistemas que aplicam essas técnicas. Sabe-se que quanto mais são criados sistemas refinados para controle de reputação, mais os atacantes se adaptam a estes ambientes. Reunir todas as soluções que possam tornar a rede segura e garantir recursos íntegros é um grande desafio, além de adicionar risco no comprometimento do desempenho da rede com excesso de execução de mecanismos de controle. Apesar das contribuições apresentadas por cada um dos sistemas serem validas, a análise da natureza da aplicação no qual serão aplicadados também deve ser consideradas.

Capítulo 5

Protocolo de Reputação P2PWSRep

Os protocolos de reputação em redes P2P têm um papel importante na segurança da rede. Esses protocolos são responsáveis por auxiliar os nós a encontrar a fonte mais confiável de um determinado recurso que desejam obter. Normalmente os protocolos baseiam-se em experiências anteriores para calcular um valor numérico que represente um valor de confiança com relação a um determinado nó possuidor do recurso ou ao próprio recurso. Dentro da rede, o protocolo P2PWSRep, proposto neste trabalho, fará o gerenciamento de reputação de nós e recursos, sendo projetado com a tecnologia de serviços web para possibilitar uma melhor integração com outras tecnologias e a extensibilidade do seu código, além de permitir que o tráfego da rede P2P possa passar por *firewalls* e outros elementos de proteção.

5.1 Introdução

O P2PWSRep é um protocolo de gerenciamento de reputação dos nós da rede P2PWS[1] e de seus recursos compartilhados. Os recursos podem ser de diversos

[1]Neste trabalho, o termo P2PWS refere-se à rede que segue a definição da arquitetura de serviços web para redes P2P e o termo P2PWSRep é referente ao protocolo de reputação utilizado por redes P2P, orientadas a serviços ou não.

tipos, dependendo da natureza da aplicação P2P. Assim, para uma rede de compartilhamento de arquivos, existem arquivos de dados, músicas, vídeos; para uma rede de relacionamento, existe o perfil do integrante; para uma grade de computadores, tem-se a capacidade de processamento, etc. O P2PWSRep será capaz de controlar qualquer um destes recursos, devido ao seu projeto permitir sua extensibilidade e portabilidade para diversas aplicações P2P.

O P2PWSRep tem uma arquitetura descentralizada, isto é, não há um servidor central para autenticar usuários ou auxiliar na busca de recursos na rede. Essas duas características adicionam dois problemas, que são: como um nó irá ingressar na rede e como um recurso compartilhado será localizado. O primeiro problema é causado pela característica dinâmica das redes P2P, já que os nós entram e saem da rede *overlay* de forma não previsível e pode ser solucionado por uma estratégia de *pong caching*, onde uma resposta ao *Ping* leva consigo uma lista de nós vizinhos com endereços pré-configurados para que o nó possa ingressar na rede com um conhecimento mínimo de alguns nós. O segundo problema pode ser contornado utilizando uma estratégia chamada cega, onde a consulta é encaminhada para o maior número de nós possíveis, sem levar em consideração consultas prévias.

O protocolo P2PWSRep se baseia no protocolo Gnutella versão 0.4, mas suas interfaces foram definidas para funcionar em qualquer arquitetura de rede P2P. O motivo desta escolha é validar seu funcionamento em uma arquitetura totalmente desestruturada, sem qualquer mecanismo de otimização de ingresso e busca de recursos. Assim, pode-se observar como o protocolo irá funcionar com problemas já conhecidos dessas redes, tais como, comprometimento da largura de banda e escalabilidade, quando a rede possui milhares de nós.

O mecanismo de roteamento das mensagens é similar ao Gnutella, conhecido como inundação, onde uma mensagem é enviada para todos os vizinhos exceto o que originou a mensagem. Esta forma de encaminhamento é chamada de cega e gera um itenso tráfego na rede. O número de vezes que as mensagens serão encaminhadas depende do valor de um campo de cabeçalho TTL (*time to live*), que impede que uma mensagem seja encaminhada na rede indefinidamente. Em uma estratégia

guiada uma consulta é encaminhada com base em similaridade, isto é, toda vez que uma mensagem for enviada, o nó analisa e procura todos os nós que já responderam a consultas parecidas, então envia a mensagem para estes nós. Esta estratégia apresenta uma redução considerável do número de mensagens na rede.

A proposta de definir um protocolo de reputação para redes P2P foi motivado pelo fato da maioria das redes P2P existentes não possuírem mecanismos efetivos para evitar a presença de nós egoístas e maliciosos, bem como restrições para barrar ou detectar a inserção de conteúdo corrompido. A eliminação desses problemas e controle de uma série de ataques, que foram mencionados neste trabalho, sendo os mais comuns o *whitewashing*, *sybil* e ataque do traidor, além de incompatibilidades de operações e metadados dessas redes, fez com que o protótipo da rede P2PWSRep fosse totalmente portável para todas as aplicações que desejem integrá-lo. Por exemplo, caso um site de vendas queira utilizar o controle de reputação do protocolo P2PWSRep, então poderá, através da interface de serviços web do protocolo fazê-lo, já que o protocolo não é voltado especificamente ao conteúdo que é disponibilizado, mas à qualidade de quem oferece e do que é oferecido. Para possibilitar que o protocolo possa ser facilmente adaptado para suportar novos ataques, e utilizado por diversos tipos de aplicações P2P, a tecnologia de serviços web foi utilizada para facilitar tal integração.

A incompatibilidade refere-se aos metadados das diferentes redes P2P, dificultando a reutilização de recursos, e por vezes, a comunicação entre estas redes. Por isso, os metadados foram propostos de tal forma que facilitem a integração, sem necessidade de elementos intermediários, como *gateways*, diminuindo assim o volume de arquivos duplicados na rede, mas mantendo os identificadores de cada um, já que utilizam mecanismos diferentes de geração de identificadores.

Dois fatores contribuíram para a escolha de uma infraestrutura baseada em serviços web: o crescente número de aplicações que utilizam essa tecnologia, facilidade de configuração sem que haja muitas intervenções por parte dos administradores de rede, já que o tráfego das requisições passa através de elementos de proteção, tais como *firewalls*, *proxies*, IPS (*Intrusion Prevent System*), etc., pois

os pacotes HTTP estão liberados na maioria dos mesmos, além da simplicidade da infraestrutura orientada a serviços e de suas requisições. Além disso, a utilização desta tecnologia proporciona a interoperabilidade com outros serviços, oferecendo portabilidade para a aplicação e uma flexibilidade para que a mesma seja estendida, apenas agregando-se novos serviços web. Os metadados da infraestrutura serão escritos em XML. Isso facilitará a integração com outras ferramentas de compartilhamento e até mesmo com outras infraestruturas orientadas a serviços.

5.2 Descrição da Arquitetura P2PWSRep

5.2.1 Modelagem dos Casos de Uso

As funcionalidades básicas do protocolo P2PWSRep estão representadas nos casos de uso que se seguem. Através dos mesmos, tem-se uma visão geral do funcionamento e o papel desempenhado por cada nó no sistema. Os casos de uso estão divididos em três categorias: *Caso de Uso Ingressar na Rede*, *Caso de Uso Realizar Pesquisa de Recursos*, *Caso de Uso de Atualizar Reputação*.

O primeiro caso de uso, *Ingressar na Rede P2P*, está apresentado na Figura 5.1. Nota-se que ele executa três outros casos de uso para realizar toda a operação. Tudo começa com um nó que pretende entrar na rede. Assim, ele deve contactar um nó que já faz parte da mesma, chamado de anfitrião, passando seus dados. Os dados podem incluir informações de capacidade (que serão verificadas), disponibilidade de recursos compartilhados, como quantidade e volume em kilobytes.

Ao receber os dados do nó interessado em participar da rede, é gerado um identificador através de um algoritmo de geração de números randômicos, SHA-1 (*Secure Hash Algorithm*). O identificador é chamado de *peerID* e passa a servir como uma identidade única do nó em toda rede. O identificador é transferido para o novo nó, juntamente com a tabela de nós vizinhos do nó anfitrião. A tabela é chamada de *TabPeer* e seus campos são basicamente o identificador de cada nó conhecido e seu endereço corrente. Como são utilizados serviços web para comunicação, então este endereço é representado por uma *url* para o serviço. Ao *peerID* é associado um

carimbo de tempo para permitir que se conheça quando o nó ingressou na rede. Ao final das etapas, o nó já faz parte da rede P2P e todas as vezes que o nó se conectar à rede, ele envia para seus vizinhos seu *peerID* para infomar que está ativo. Isso possibilita a seus vizinhos medir disponibilidade do nó. Segundo (Stallings, 2007), a possibilidade de duplicação de um identificador gerado por um algoritmo de números aleatórios com chave de 160 bits é muito improvável, sendo suficientemente seguro para uma rede P2P.

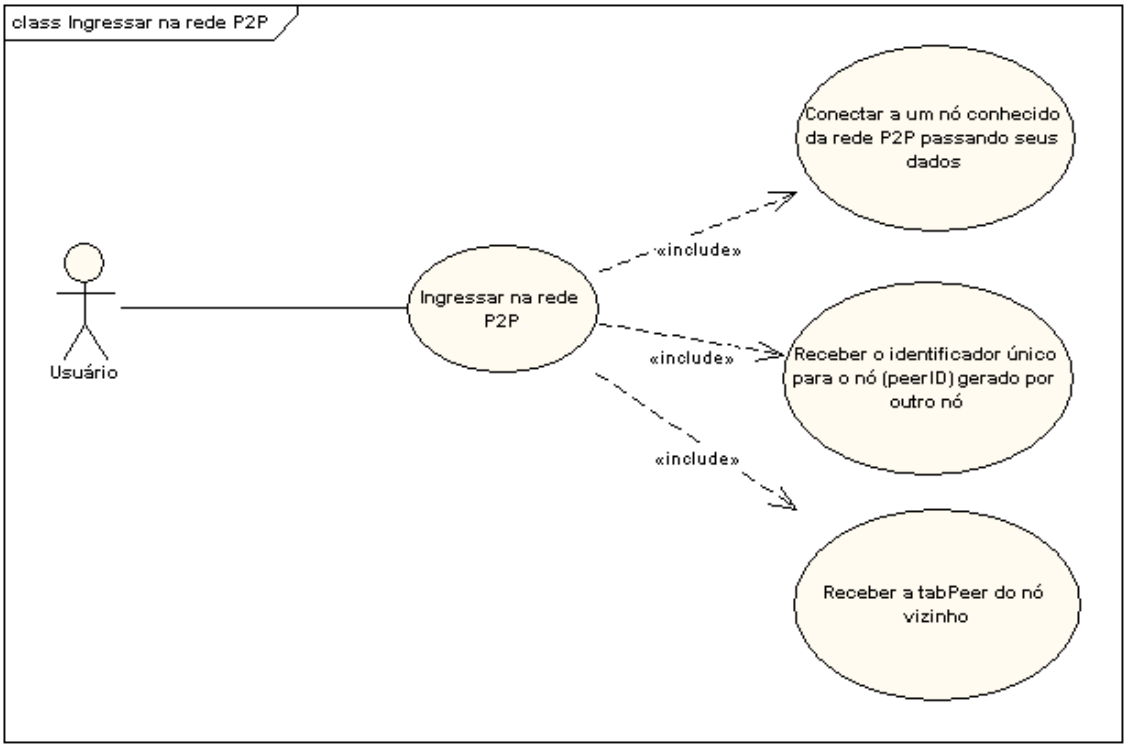

Figura 5.1: Caso de uso de ingresso na rede

O *Caso de Uso Realizar Pesquisa de Recursos* é o que executa mais operações, e é mostrado na Figura 5.2. Inicialmente, o nó interessado em um recurso faz uma pesquisa na rede P2P. A pesquisa é enviada para todos os nós vizinhos contidos em sua *TabPeer* que estão *on-line*, através de uma mensagem que contém a descrição do recurso buscado (*string*) em seu cabeçalho, além de outros dados do protocolo que serão detalhados nas próximas seções.

A Figura 5.3 mostra o roteamento das mensagens da rede P2PWS. As setas

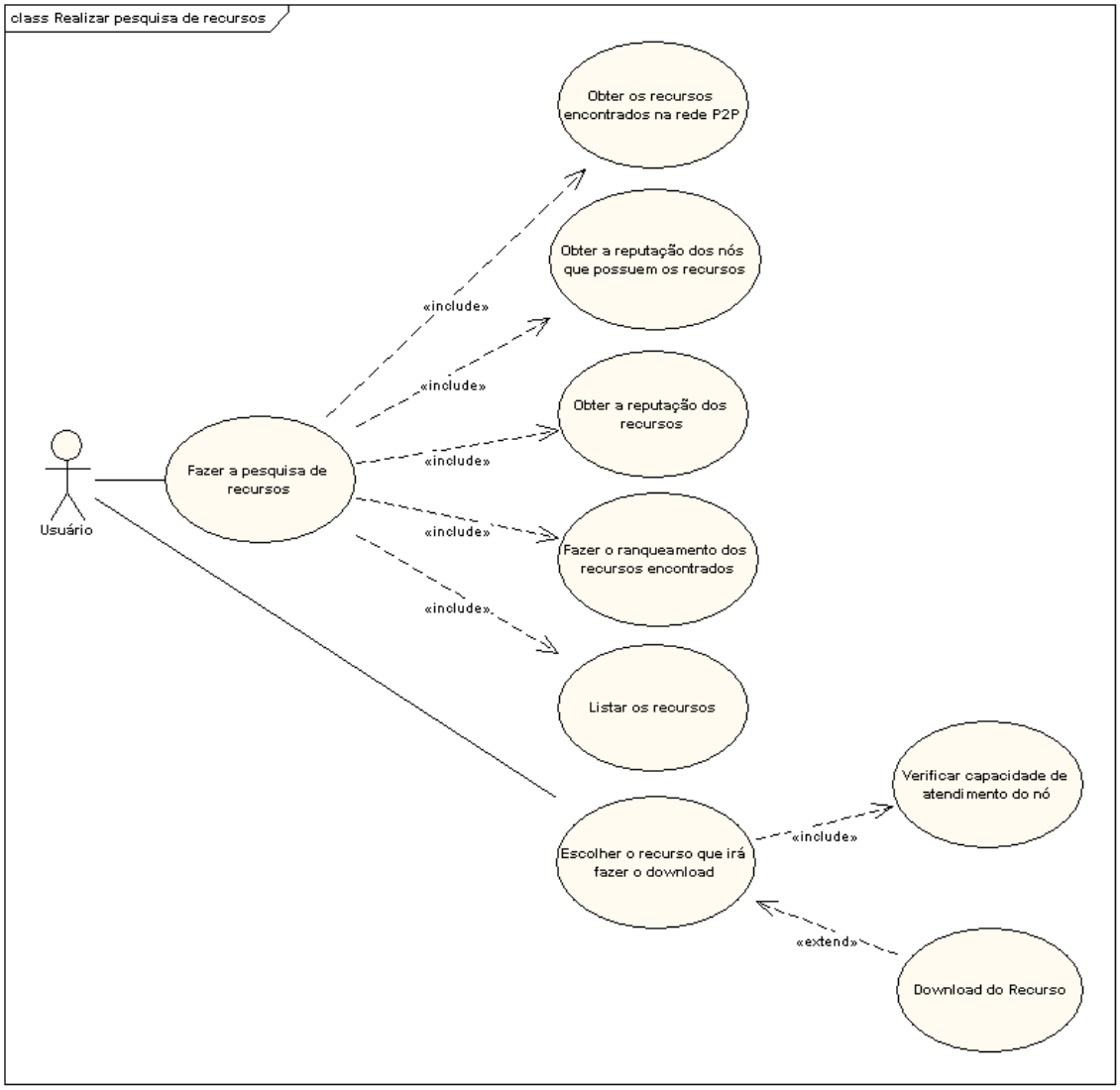

Figura 5.2: Caso de uso de busca de recursos

contínuas representam as requisições de busca e as setas tracejadas a resposta da requisição de busca. O nó E inicia uma busca pelo arquivo *rock*. Uma mensagem é gerada com a *string* de busca e TTL igual a 3, além de identificação do nó de origem, *timestamp*, id da mensagem e outras informações. A mensagem é enviada a todos os nós ligados diretamente a ele. Os nós B e C respondem positivamente. Note-se que o nó B envia a mensagem de resposta pelo mesmo caminho que recebeu a requisição, isto é, através do nó A. As mensagens de A-D, C-D e D-B são descartadas por cópia duplicada.

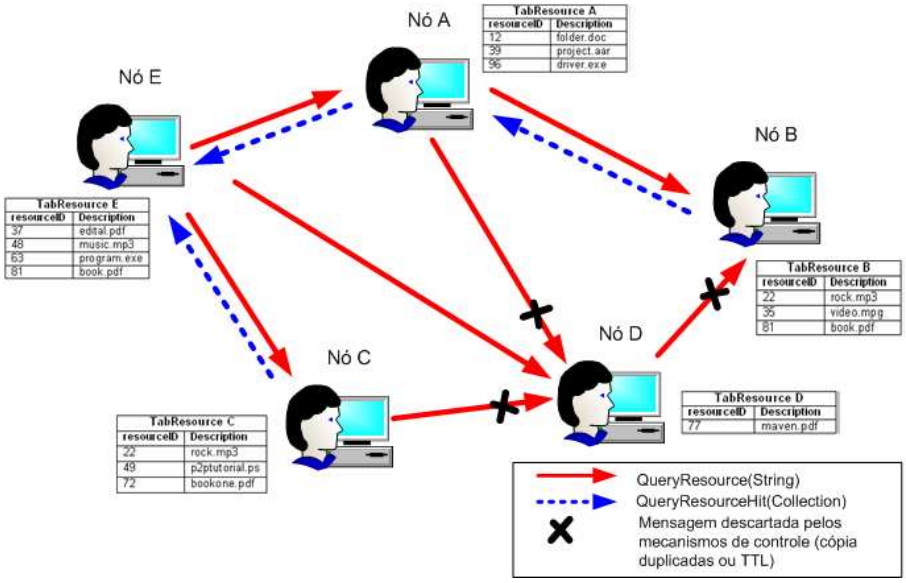

Figura 5.3: Roteamento das mensagens na rede P2PWS.

Ao receber a resposta dos nós que contêm os recursos, o nó que iniciou a busca também iniciará a obtenção da reputação dos nós e recursos disponíveis. Após ter a reputação dos nós e dos recursos, é realizado um ranqueamento dos recursos encontrados, o qual é mostrado ao usuário, como se vê na Figura 5.4. O usuário, ao escolher o recurso que pretende fazer o *download*, é feita uma verificação de capacidade do nó fornecedor. Essa verificação tem como objetivo avaliar a disponibilidade do nó em atender a requisição, para que não venha a sobrecarregá-lo. O nó estando apto a atender, o *download* do arquivo então é iniciado. Percebe-se, no Caso de Uso da Figura 5.2, que há um estereótipo << *extend* >> mostrando que nem sempre o *download* é realizado por problemas de capacidade do nó fornecedor, no entanto, estatégias de enfileiramento podem ser empregadas para diminuir a sobrecarga do fornecedor e atender futuramente a requisição.

Por último, após o *download* do recurso, é realizado o *Caso de Uso Atualizar Reputação*, apresentado na Figura 5.6. A atualização é realizada positivamente caso o *download* ocorra com sucesso, isto é, sem interrupção, mas mesmo que o arquivo baixado não tenha tido problemas na transmissão, é verificada a integridade do arquivo. Para isso, é gerado um *hash* do mesmo e iniciada uma consulta à rede,

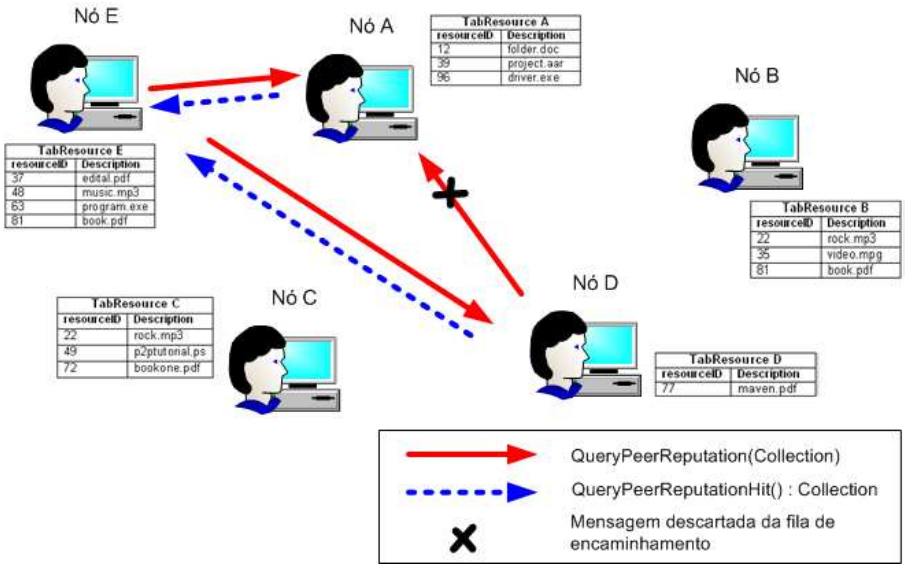

Figura 5.4: Obtenção da reputação de um nó da rede.

similar à busca de reputação de nós e recursos. Logo os nós que conhecem o recurso, irão responder com o código do arquivo para aquele nó. A Figura 5.5 mostra o *download* sendo realizado diretamente do nó escolhido. Porém, antes do *download* é realizada uma requisição ao nó, que verifica sua capacidade de atendimento da requisição.

Existe uma tabela específica para reputação dos recursos (*tabResourceReputation*). Isso possibilita que nas próximas buscas de reputação do recurso, o mesmo seja identificado como conteúdo poluído. Para cada entrada desta tabela, irá existir o identificador do nó, o identificador do recurso e o código de erro. Esse código de erro informa se o arquivo é corrompido, não autêntico, com vírus ou verme, etc. Uma notificação é enviada ao nó que forneceu o arquivo para que ele possa remover o arquivo de seu repositório de arquivos compartilhados. A reputação do nó que forneceu o recurso também é decrementada, já que o mesmo foi negligente em deixar um recurso não íntegro na rede.

Até o momento, a identificação dos recursos não havia sido mencionada. Cada recurso possui um identificador único, chamado *resourceID*, gerado por um algoritmo de *hash*, SHA-1, com um tamanho de chave 160 *bits*. O mecanismo de geração é o mesmo utilizado para gerar o *peerID*.

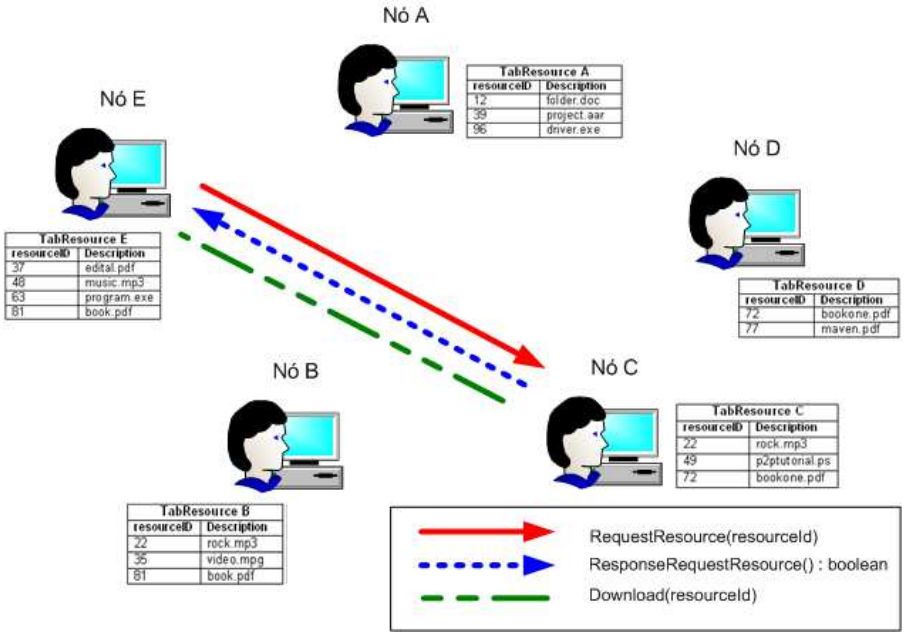

Figura 5.5: Download realizado diretamente do nó escolhido

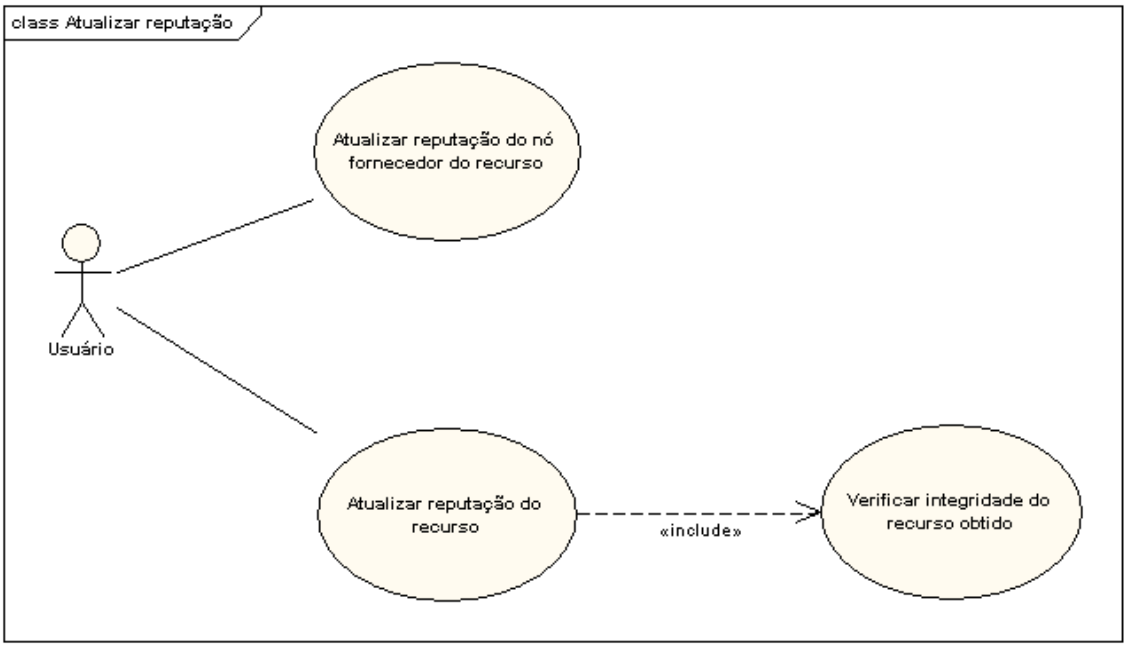

Figura 5.6: Caso de uso de gerenciamento de reputação

5.3 Metadados

Para a realização do gerenciamento de reputação, dados referentes aos nós, recursos e outras informações são necessários. Agora, será apresentada a modelagem

dos dados necessários ao funcionamento do protocolo P2PWSRep. Cada nó contém cinco tabelas, sendo duas para controle dos nós e recursos compartilhados, duas para controle da reputação dos nós e controle da reputação dos recursos presentes na rede e uma quinta tabela é usada para guardar informações pertinentes ao nó, tais como seu identificador, descrição, localização, informações para controle de valor de *hops* adaptativo, reputação conhecida através de outros nós na rede, localização dos metadados de nós e recursos, data do ingresso na rede, etc.

5.3.1 Metadados de Nós

Os metadados dos nós são informações para controle dos nós e da reputação. São tabelas que não possuem um número muito grande de dados, somente de seus vizinhos, para que as buscas realizadas localmente pelo nó nestas tabelas não possam comprometer o desempenho do protocolo. São três as tabelas que armazenam dados dos nós. A Tabela 5.1 - Tabela de Nós (*tabPeer*), armazena informações sobre os nós vizinhos, isto é, os nós que se ligam diretamente ao proprietário da tabela.

Nome do Campo	Descrição
peerID	Identificador gerado por uma função *hash*. O algoritmo utilizado para geração da chave é o SHA-1, com um *hash* de 160 *bits*.
peerUrl	Endereço de conexão do serviço. É dado através do IP corrente e o *port type* do serviço. O endereço IP utiliza a versão IPv4.
joiningDate	Data do ingresso na rede. Utilizado para verificar desde quando o nó está presente na rede. É importante para determinação da confiabilidade do nó, bem como sua disponibilidade.
protoVersion	Versão do protocolo P2PWSRep.

Tabela 5.1: Tabela de nós.

A Tabela 5.2, Tabela de Reputação de Nós, é referente aos metadados dos

nós, é chamada de *tabPeerReputation*, armazena as informações necessárias para controle de reputação dos nós presentes na rede. A tabela contém apenas nós que já interagiram com o proprietário da tabela. Cada nó possui sua tabela de reputação.

Nome do Campo	Descrição
peerID	Identificador do nó conhecido.
satTrans	Número de transações satisfatórias com este nó.
unsatTrans	Número de transações insatisfatórias com este nó.
previousRep	Valor da reputação, para o nó buscado, obtida anteriormente. Isto possibilita que este valor seja utilizado para considerar a evolução do nó com relação à reputação atual.

Tabela 5.2: Tabela de reputação dos nós.

A Tabela 5.3, Tabela de Configurações (*tabSettings*), armazena as informações de configuração dos nós, contendo dados importantes para o funcionamento dos mesmos e da própria rede.

Nome do Campo	Descrição
peerID	Identificador do nó.
nodeName	Nome do nó (32 caracteres).
description	Descrição do nó. (64 caracteres).
location	localização do nó (país).
reputation	Reputação recente do nó. Este dado é obtido de outros nós e não pode ser atribuído ou calculado pelo próprio nó.

Tabela 5.3: Tabela de configuração do nó.

O campo *reputation* tem como finalidade fazer com que o usuário conheça sua reputação na rede. O processo de obtenção da reputação para o próprio nó é o mesmo que é realizado para busca de reputação de qualquer outro nó, podendo este valor ser atualizado toda vez que o nó se conecta na rede ou periodicamente.

5.3.2 Metadados de Recursos

Os metadados dos recursos, assim como dos nós, armazenam dados referentes aos recuros compartilhados na rede e dados necessários para controle da reputação pelo protocolo P2PWSRep. São duas as tabelas, a primeira é a *tabResource*, que armazena informações sobre os recursos que o nó compartilha, sendo sua apresentada na Tabela 5.4.

Nome do Campo	Descrição
resourceID	Identificador do recurso gerado por uma função hash. O tamanho do hash é de 160 bits, gerado através do algoritmo SHA-1.
name	Nome do recurso na rede.
description	Descrição do recurso.
resourceCount	Contador de acesso para o recurso. Pode ser utilizado para fornecer estatísticas dos recursos mais buscados na rede.

Tabela 5.4: Tabela de recursos.

A Tabela 5.5 - Tabela de Reputação de Recursos (*tabResourceReputation*) armazena a reputação dos recursos. Ela é semelhante a uma lista negra de recursos presentes na rede, só que de forma descentralizada, onde cada nó mantém a reputação de recursos.

É importante salientar que o protocolo P2PWSRep é distribuído, assim o gerenciamento da reputação de nós e recursos é realizado por cada nó, tendo em suas tabelas valores que não necessariamente são iguais em toda rede, pois não há uma sincronização de dados das tabelas entre os nós.

5.4 O Protocolo P2PWSRep

Nesta seção será descrito o protocolo P2PWSRep, com detalhes do roteamento de mensagens e do cálculo de reputação. Cada nó possui uma tabela do

Nome do Campo	Descrição
resourceID	Identificador do recurso.
peerID	Identificador do nó que contém o recurso.
errorType	Tipo de erro do recurso: 1 - recurso corrompido; 2 - recurso contém vírus ou verme; 3 - recurso adulterado; 4 - recurso inexistente na origem.

Tabela 5.5: Tabela de reputação de recursos.

histórico das transações satisfatórias (*satTrans*) e insatisfatórias (*unsatTrans*) com os demais nós. Na infraestrutura proposta, a tabela armazena apenas o somatório dessas transações, sendo que não há informação temporal de cada uma delas. Para que o comportamento da reputação do nó com relação ao tempo seja considerado, será armazenada para cada nó a última reputação obtida. Esse valor será importante para o método que será aplicado para o cálculo da reputação, já que será utilizada uma série temporal, a média ponderada exponencial.

A Tabela 5.2 (Reputação de Nós) é chamada de *tabPeerReputation*. Cada linha da tabela mostra o nó que já forneceu dados compartilhados e o número de transações que foram completadas com sucesso e as que falharam por diversos motivos, tais como: interrupção da conexão, baixa qualidade de conexão, arquivo não encontrado, etc. O protocolo P2PWSRep não considera o contexto da transação, apenas avalia se ela foi completada ou não. Para cada nó também é guardado o valor da última reputação obtida da rede, no campo *previousRep*. Esse dado é interessante no cálculo da reputação atual em relação com sua média histórica. Caso não exista este valor, isto é, o nó nunca teve troca de dados com o requisitante, então ele não será considerado no cálculo.

Quando um nó faz a pesquisa de um recurso, recebe a listagem de todos os nós que contém o recurso e sua localização. A rede segue o princípio do anonimato dos nós, já que é utilizado um identificador específico, que não revela quem é quem

dentro da rede. Essa identificação é feita através do identificador *peerID*, obtido uma única vez, no momento do ingresso na rede. É interessante destacar que o anonimato do nó não é real, sendo também chamado de pseudoanonimato, já que alguns dados são impossíveis de ocultar. Um exemplo é o endereço IP que o nó está utilizando na conexão.

A busca é desestruturada, sendo baseada no protocolo Gnutella v0.4. As requisições de buscas são enviadas para todos os vizinhos. Cada requisição leva consigo uma *string* com a descrição do recurso buscado e um número de vezes que a requisição deve ser encaminhada. Em cada nó que a requisição chega, este valor é decrementado e, quando atingir o valor zero, a requisição é removida da rede. A resposta é enviada pelo mesmo caminho que a requisição de busca percorreu. Isto é possível devido a forma de roteamento das mensagens, que é realizada da mesma forma do protocolo Gnutella v0.4. Cada nó guarda o identificador da requisição de busca. Quando uma resposta chega ao nó, ele verifica se houve a passagem dessa requisição; se houve, a requisição é encaminhada ao próximo nó em direção ao destino, caso contrário ela é excluída. Isso evita que requisições falsas sejam inseridas na rede e que haja um aumento do número de respostas devido à inundação natural de redes com estratégia cega.

Localizados os recursos, o próximo passo é obter a reputação dos nós que possuem os recursos. Assim, o nó requisitante envia para todos os nós que ele possui na sua tabela de nós uma requisição da reputação dos nós que possuem o recurso desejado. Essa requisição é encaminhada para todos os nós em profundidade e por inundação. O número de encaminhamentos é controlado através dos campos *hops* e TTL, contidos no cabeçalho da requisição.

Uma alternativa para o mecanismo de envio de mensagens do protocolo P2PWSRep é fazer com que o valor TTL seja adaptativo. Para isso baseou-se no protocolo de inicialização lenta (*slow start*) (Jacobson, 1988). Assim, a mensagem de busca é enviada com um valor de TTL baixo, se não existir nenhuma resposta à busca, então o valor do TTL é incrementado em 1. Será estabelecida uma janela de tempo para a espera máxima das respostas. Caso as respostas às

buscas estejam demorando muito, isso pode significar que a profundidade é grande demais para causar uma latência na rede. Este valor fica armazenado temporariamente, até que o nó se desconecte, para que nas próximas mensagens de busca seja utilizado o melhor valor de profundidade de busca.

Antes de enviar o resultado, é feito o cálculo da reputação. O cálculo é realizado pela Fórmula 5.1:

$$reputation_{i,j} = \frac{sat\,(i,j) - unsat\,(i,j)}{sat\,(i,j)}$$

O cálculo da reputação é realizado em todos os nós que receberam o pedido de reputação sobre um determinado nó, sendo $\boldsymbol{i}$ o nó local e $\boldsymbol{j}$ o nó que está sendo pesquisada a reputação. Então $\boldsymbol{sat}(\boldsymbol{i},\boldsymbol{j})$ corresponde ao número de transações satisfatórias que ocorreram entre $\boldsymbol{i}$ e $\boldsymbol{j}$ e $\boldsymbol{unsat}(\boldsymbol{i},\ \boldsymbol{j})$ corresponde ao número de transações insatisfatórias que ocorreram entre $\boldsymbol{i}$ e $\boldsymbol{j}$. O valor obtido através da fórmula é enviado ao nó que solicitou a reputação de $\boldsymbol{j}$. Os nós que obtiverem como valor de reputação para o nó $\boldsymbol{j}$ igual a zero não necessitam enviar a resposta da requisição para o nó requisitante. Isso faz com que ocorra uma redução do número de requisições desnecessárias na rede, principalmente quando o nó pesquisado é um nó novo.

As respostas, ao chegarem ao nó requisitante, todas as requisições provenientes dos nós consultados acerca da reputação do nó $\boldsymbol{j}$, são armazenadas para que a reputação global seja calculada. Este armazenamento é realizado em uma fila de mensagens, que será processada ao fim da janela de tempo. O cálculo é realizado da seguinte forma: todos os valores são somados e é feita uma média aritmética. O valor da média dos valores de reputação recebidos dos $\boldsymbol{k}$ nós consultados na rede a respeito do nó $\boldsymbol{j}$, é chamado de reputação atual para $\boldsymbol{j}$ (*currentRep*). A Fórmula 5.2 mostra este cálculo:

$$currentRep_{i,j} = \frac{\sum Reputation_{k,j}}{totalResponses}$$

Com a reputação atual e a reputação anterior do nó, outrora obtida e contida na *tabPeer*, será feita uma média ponderada exponencial. Dessa forma, o protocolo

P2PWSRep considera que dados pontuais mais recentes têm maior peso, com este peso declinando exponencialmente à medida que esses dados tornam-se ultrapassados.

A constante *alfa* de ajuste aplicada determinará o nível de ajuste e a velocidade de transição da reputação, para a diferença entre as previsões e as ocorrências reais. A determinação da constante pode até ser arbitrária, mas de acordo com simulações realizadas neste trabalho, optou-se por utilizar o valor de *alfa* como 0.4. Analisando a Fórmula 5.3, pode-se observar que quanto maior o valor de *alfa*, há uma valorização da reputação anterior (o histórico da reputação). Caso o *alfa* esteja tendendo a um valor mais baixo, então há uma valorização da reputação atual. A consideração para a escolha desse valor é que a reputação atual tem um maior peso, muito embora o reflexo da reputação anterior é também considerado. Além disso, definição do valor de *alfa*, leva a entender que a reputação pode ser facilmente ser modificada em poucas interações, mas isso se deve a necessidade de obter um conhecimento do comportamento dos nós da rede na simulação, com uma quantidade de ciclos menores.

Ao fazer o cálculo da reputação dos nós que contêm os recursos, então os mesmos são mostrados em ordem da maior reputação. Por enquanto, não estão sendo incluídos outros dados para fazer o ranqueamento dos recursos encontrados, somente o valor da reputação. A obtenção da reputação se dá pela Fórmula 5.3:

$$reputation = (1 - \alpha) \cdot currentRep + previousRep \cdot \alpha$$

O gráfico da Figura 5.7 pode mostrar melhor o que foi considerado. Nesta simulação, o nó possuía uma reputação de 0.8, armazenado na *tabPeer* do nó interessado. Após consultados os nós na rede, o resultado da reputação atual para o nó foi de 0.655. Isto significa que o nó não cometeu muitas transações insatisfatórias na rede. O protocolo de P2PWSRep não implica em que a reputação seja considerada sem que sejam levado em consideração o fator temporal. A partir do gráfico, pode-se ver como o *alfa* influencia na reputação final. Para *alfa* igual a 1, a reputação atual não afeta o valor anterior, o que não é interessante para o protocolo. Para *alfa* igual

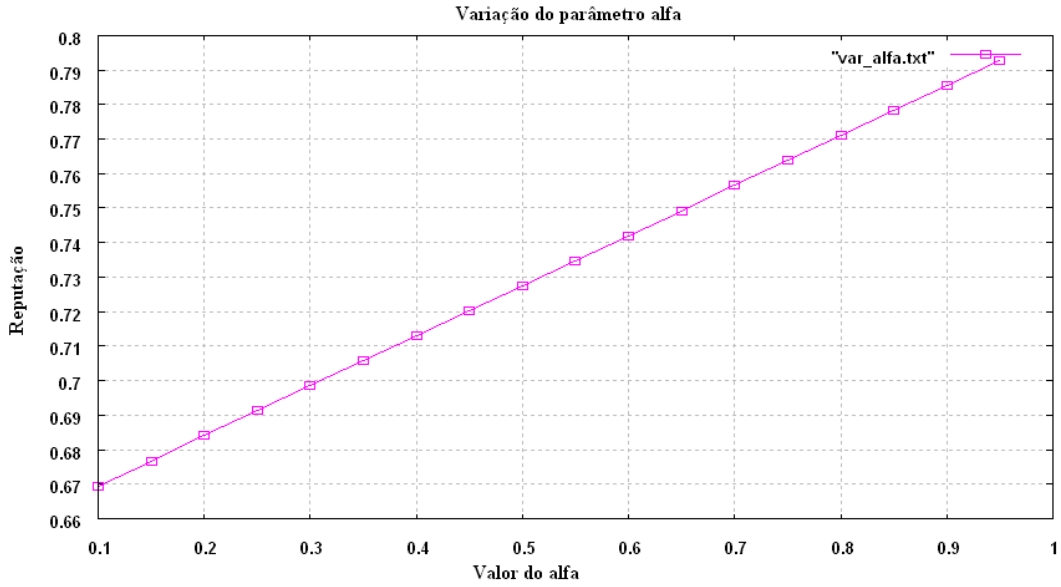

Figura 5.7: Influência do parâmetro alfa no cálculo da reputação

a zero, o valor da reputação anterior não é considerado na reputação final. Para *alfa* igual a 0.5, o valor da reputação sofre uma queda para o valor de 0.7275, uma perda de 9,06% da reputação. Para o valor de *alfa* de 0.4, o valor final da reputação sofre uma queda para 0.713, uma perda de 10,87%, o requerido para o protocolo. O protocolo foi projetado para que as perdas não sejam tão abruptas, já que o nó pode ser prejudicado por fatores em que ele não esteja envolvido diretamente, tal como, falhas na rede. Imagine um cliente de um banco há 10 anos, que nunca deixou de pagar suas contas em dia, mas, por um problema de comunicação entre sua empresa empregadora e o banco, seu salário não foi creditado, assim todas as operações de débito automático foram recusadas e cheques foram devolvidos. Este cliente ficaria prejudicado se perdesse sua reputação abruptamente. O protocolo P2PWSRep tenta contornar esse tipo de problema, num cenário de compartilhamento de recursos em redes P2P.

Quando todos os valores de reputação finais são obtidos para os nós que possuem o recurso buscado, o nó requisitante iniciará a busca da reputação desses recursos. O processo é muito semelhante à busca de reputação de nós. Para isso, o requisitante envia para seus vizinhos uma mensagem que contém a lista dos recursos

pesquisados e os nós que os compartilham. Os nós recebem a requisição e respondem ao requisitante com a reputação dos recursos, enviando uma lista, com o *peerID*, *resourceID* e o código de erro (*errorType*). O mecanismo de roteamento, controle de mensagens duplicadas, TTL adaptativo e janela de processamento das respostas são os mesmos para busca de reputação de nós. Após o término da busca da reputação dos recursos, uma lista é exibida ao usuário com os recursos disponíveis, ranqueados de acordo com os dados da reputação de nós e recursos. O usuário fica livre para escolher qualquer recurso para fazer o *download*.

Ao escolher a fonte, o nó que possui o recurso irá fazer testes de capacidade de atendimento da requisição e retornar ao nó que está solicitando uma resposta, que pode ser uma requisição aceita, requisição recusada ou requisição enfileirada. Isso evita que um nó de alta reputação e com um grande número de recursos compartilhados seja sobrecarregado, pois será natural que o mesmo sempre seja bem ranqueado nas buscas. Caso ocorram falhas de conexão, queda de conexão ou outro problema de comunicação, o número de transações insatisfatórias (*unsatTrans*) é incrementado em 1 (um).

Para gerenciar a reputação dos recursos, cada recurso tem um identificador único gerado através de uma função *hash*. Isso possibilita que os recursos compartilhados, por mais que tenham nomes diferentes, sejam identificados de forma única dentro da rede P2P. Isso possibilita a utilização de reputação para os recursos ou uso de *blacklist* de recursos maliciosos ou não íntegros.

Após o término da transmissão do recurso, o mesmo é avaliado através de uma função de verificação de integridade. Caso o arquivo seja íntegro, então o número de transações satisfatórias com o nó fornecedor do recurso é incrementado em 1 (um); caso contrário, o número de transações insatisfatórias é incrementado em 1 (um) e também é atualizada a tabela de reputação de recursos *tabResourceReputation*, informando o identificador do nó, o identificador do recurso e o código de erro do recurso. Nas próximas consultas de reputação do recurso, este dado será repassado para os requisitantes.

Além disso, em melhorias futuras do protocolo, pode-se colocar como peso,

dados a respeito da velocidade da conexão e largura de banda real do nó fornecedor. Nesse caso, o valor de incremento não será um valor 1, mas seguindo um intervalo entre 0 e 1, já que o peso irá influenciar na opinião do requisitante.

5.5 Descrição das Interfaces dos Serviços do Protocolo P2PWSRep

A arquitetura da rede P2PWS é baseada na rede Gnutella, sendo descentralizada, com mecanismo de busca realizado através de encaminhamentos sucessivos. O protocolo P2PWSRep insere mecanismos de otimização da busca e diminuição de sobrecarga de mensagens geradas na rede. As mensagens enviadas entre os nós têm roteamento semelhante a da rede Gnutella, mas são realizadas por meio de serviços web. As requisições aos serviços são realizadas utilizando chamadas síncronas ou assíncronas (*callback*), o que significa que o nó não ficará bloqueado esperando a resposta.

A Tabela 5.6 mostra as principais operações do protocolo P2PWSRep. No nome da operação, é destacado o tipo de mensagem, que pode ser síncrona ou assíncrona. Na descrição, são detalhadas as funcionalidades das operações.

Além das operações mostradas na Tabela 5.6, os nós executam algumas operações durante o processamento das mensagens. As operações são implementadas na interface cliente ou servidor dos nós, através de métodos que são invocados ou controlados por meio de *threads*. O diagrama de sequência da Figura 5.8 mostra as interações das operações do protocolo P2PWSRep.

- **Geração do identificador único do nó** - Cada nó da rede P2PWSRep tem um indentificador único, chamado de *peerID*. Este identificador é gerado através do método *GeneratePeerID()*, da interface servidor, que utiliza uma função SHA-1 para geração de *hash*.

- **Busca de recursos** - Quando um nó recebe uma requisição, a *string* de consulta, contida na carga útil da requisição, é usada para pesquisar no repositório

Nome da Operação	Descrição
Login (Síncrona)	Solicita ingresso na rede P2P. O nó anfitrião gera um peerID e o envia junto com a sua tabela de nós para o novo nó.
Connect (Síncrona)	Operação realizada cada vez que o nó se conecta à rede. Ele atualiza seu IP corrente na rede e faz a verificação do seu peerID.
QueryResource (Assíncrona)	Envia a descrição do recurso (string) a ser pesquisado na rede.
QueryResourceHit (Assíncrona)	Resposta para um QueryResource. Traz consigo os endereços (urls) dos recursos pesquisados.
QueryPeerReputation (Assíncrona)	Busca a reputação dos nós que possuem o recurso, através QueryResourceHit.
QueryPeerReputationHit (Assíncrona)	Resposta para o QueryPeerReputation. Traz como resultado os valores de reputação dos nós, obtidos dos demais nós da rede.
QueryResourceReputation (Assíncrona)	Busca da reputação dos recursos encontrados nos nós.
QueryResourceReputationHit (Assíncrona)	Resposta para o QueryResourceReputation, trazendo como resultado a reputação dos recursos.
Download (Síncrona)	Solicitação do download do recurso.
QueryMyReputation (Assíncrona)	O nó faz uma busca de como está sua reputação na rede.
NotifyDeletion (Assíncrona)	Informa ao nó que contém um recurso corrompido, que o mesmo deve ser removido do seu repositório.

Tabela 5.6: Operações do protocolo P2PWSRep

de dados do nó. O método *SearchRepository()*, da interface servidor, é utilizado para obter os recursos que o nó contém. A operação *QueryResourceHit* envia a resposta ao nó requisitante.

- **Verificação do número de hops** - A cada nó que a mensagem é encaminhada, o número de hops é decrementado, quando o valor chega a zero, a mensagem é removida da rede. Isto possibilita que as mensagens não fiquem trafegando indefinidamente na rede. Os métodos que realizam estes controles são o *Foward()* e o *DropMessage()*, da interface servidor.

- **Verificação de mensagem duplicada** - Cada mensagem possui em seu cabeçalho um identificador e um *timestamp*, além do *peerID* do nó que a gerou, que possibilita que a mensagem, ao passar por um nó, tenha seus dados gravados em uma tabela. Caso uma outra mensagem chegue ao nó, ele pesquisa na tabela, se encontrar, então é uma mensagem que já foi roteada, sendo a mesma removida da rede. Esse processo de verificação de mensagens duplicadas é realizado pelo método *VerifyDuplicateMessage()* e *DropMessage()*, da interface do servidor.

- **Controle do TTL adaptativo** - O valor do TTL inicia baixo (com valor 2) e vai sendo incrementado (até o valor 6) quando as buscas de recursos ou reputação à rede não retornam resultados dentro da janela de tempo. Isso possibilita uma diminuição do número de mensagens na rede e uma adaptação dinâmica do valor do TTL. O método que realiza esse mecanismo é o *AdaptativeTTL()*, da interface do cliente.

- **Controle da janela para processamento de mensagens** - Este mecanismo é realizado pela interface cliente, em vários métodos, tais como, *ProcessResourceList()*, *CalculateCurrentRep()*, *CalculateReputation()* e *ProcessReputationResource()*. Quando uma pesquisa é iniciada, uma janela de tempo é criada, controlada por uma *thread*, sendo as mensagens que chegam dentro da janela enfileiradas. Ao fim da janela, as mensagens são processadas. As mensagens que chegam fora da janela são descartadas.

- **Cálculo da reputação** - Existem três métodos para processamento de dados de reputação. O primeiro método é realizado pela interface servidor, *CalculateReputation()*, que cálcula a reputação de um nó, pelo seu histórico de transações satifatórias e insatisfatórias. O valor obtido é transferido para o nó requisitante através da operação *QueryPeerReputationHit*. O segundo e terceiro método, *CalculateCurrentRep()* e *CalculateReputation()*, são realizados pela interface cliente, com os dados de reputação recebidos dos nós.

- **Ranqueamento dos recursos** - Após obter os dados da reputação dos nós e dos recursos, é realizado um ranqueamento dos melhores recursos para que o usuário possa escolher para *download*. O método *Ranking()* da interface cliente realiza está tarefa.

- **Teste da capacidade de atendimento do download** - Quando o usuário seleciona um recurso para ser baixado, o nó destino, que compartilha o recurso, faz uma verificação de capacidade para atendimento do nó. O método responsável por este teste é o *TestDownload()*, da interface servidor. Caso, esteja com uma sobrecarga de conexões, o nó pode negar a requisição ou enfileirar, para atendê-la mais tarde.

- **Verificação de integridade do recurso** - Ao baixar um recurso, o nó realiza uma verificação de integridade do mesmo, aplicando uma função *hash*, para saber se o *hash* obtido condiz com o original. Esta função é realizada pelo método *VerifyResourceHash()*, da interface cliente. Caso o recurso esteja corrompido, adulterado ou infectado por vírus, o nó usa a operação *NotifyDeletion* para informar o nó de origem. Além de adicionar o recurso na tabela *tabResourceReputation*, com o código de erro obtido pela função de validação.

- **Atribuição valor de transação** - Ao final da transação, um valor de reputação é atribuído à mesma. Se foi satisfatória, o campo *satTrans* do nó, na tabela tabPeer é incrementado, caso contrário o campo *unsatTrans* é incrementado. Esta função é realizada pelo método *SetReputation()*, da interface cliente.

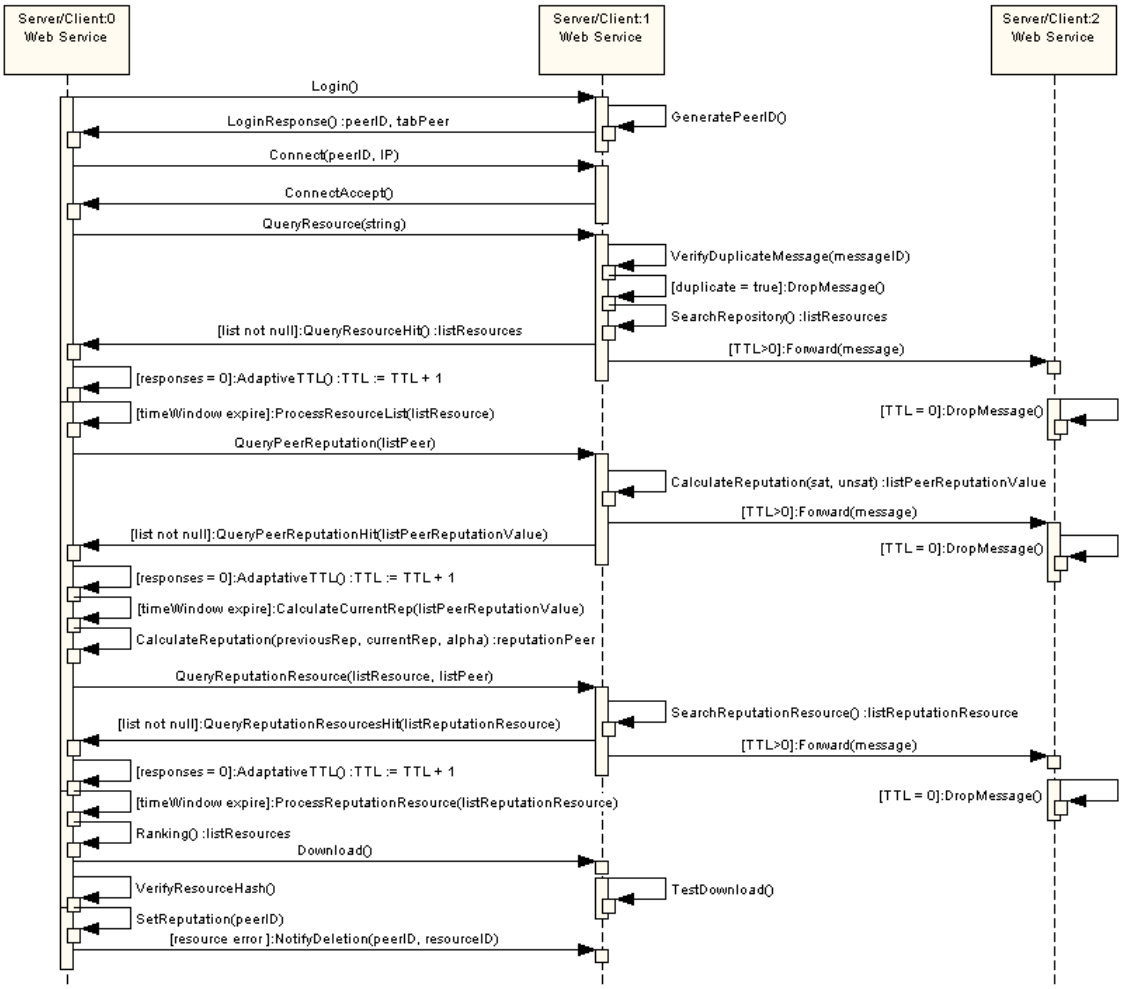

Figura 5.8: Diagrama de sequência do protocolo P2PWSRep

No diagrama de sequência da Figura 5.8, estão representados apenas três nós. Cada nó possui uma interface para as operações do cliente e servidor. As operações são realizadas através de serviços web. A operação se inicia com um nó ingressando na rede. Para isto, existe uma operação chamada de *Login()*, onde o nó solicita a um nó já conectado a rede, chamado de anfitrião, o seu ingresso. Ao receber a solicitação, o nó anfitrião gera um *peerID* para o novo nó e o envia juntamente com sua tabela de nós vizinhos. Isto possibilita que o novo nó passe a conhecer seus vizinhos. Todas as vezes que o nó ingressar na rede, ele executa a operação *Connect()*, passando seu *peerID* e seu endereço IP atual, já que pelo fator dinâmico das redes P2P, há necessidade de identificação do nó apenas pelo seu

peerID, preservando seu anonimato. Porém há necessidade de atualizar o endereço atual utilizado pelo nó. Todas as operações estão detalhadas na Tabela 5.6 e os principais métodos foram descritos nesta seção.

5.6 Considerações Finais

Neste capítulo, foram apresentadas as principais características do projeto do protocolo P2PWSRep, incluindo a arquitetura desta rede P2P, topologia e mecanismo de roteamento de mensagens. Seu mecanismo de controle de reputação é baseado nos protocolos de reputação apresentados neste trabalho, adicionando-se mecanismos para torná-lo simples e sem sobrecargas de processamento nos nós. A principal finalidade deste protocolo é controlar a reputação dos nós presentes na rede e dos recursos compartilhados de forma distribuída, sem depender de nenhum elemento central, utilizando como base a tecnologia de serviços web para possibilitar a interoperabilidade com outras redes P2P e extensibilidade de código para suportar adições de melhorias e contornar problemas de desempenho e segurança da rede.

Capítulo 6

Simulação do Protocolo P2PWSRep

6.1 Introdução

As aplicações que utilizam serviços web têm um desempenho dito inferior em relação às tecnologias convencionais de desenvolvimento de software distribuído. Assim, adicionar mecanismos complexos de controle de segurança a essas redes as deixam mais lentas. O P2PWSRep é um protocolo sem excessos de mecanismos de segurança, mas que ofereçe meios de combater as principais vulnerabilidades das redes P2P, contribuindo para que ocorra a adição de sucessivas melhorias, por este ser extensível e portável. A etapa de simulação foi importante para este trabalho, por comprovar que o protocolo P2PWSRep pode ser viável para boa parte das aplicações de redes P2P, orientadas ou não a serviços.

A simulação do protocolo P2PWSRep se ateve ao mecanismo de reputação aqui proposto, independentemente da forma com que as mensagens são transportadas, até porque o núcleo do protocolo pode ser utilizado por qualquer outra rede, mesmo que não seja orientada a serviços. A ferramenta de simulação utilizada neste trabalho foi o PeerSim (PeerSim, 2009), desenvolvido e mantido pelo Projeto BISON da Universidade de Bolonha, na Itália. Alguns critérios foram empregados na escolha do simulador, tais como, atividade atual do projeto, documentação

disponível, linguagem em que o simulador foi escrito e tipo de licenciamento. O projeto PeerSim está atualmente ativo e tem uma vasta documentação disponível, com vários exemplos de cenários de simulação. Seu código é escrito em Java e tem licença GPL (*General Public License*).

Outro fator prepoderante na escolha do simulador foi a possibilidade de fazer simulações para uma grande quantidade de nós, bem como utilizar tanto uma arquitetura de simulação estruturada, como desestruturada. O PeerSim possui escalabilidade, podendo chegar a 1 milhão de nós, sendo que na simulação do protocolo P2PWSRep foram utilizados 10 mil nós para obtenção dos resultados. Além disso, o PeerSim possibilita implentação de simulação dirigida a ciclos (*cycle-based model*) e dirigida a eventos (*event-based model*), sendo necessária a utilização dos dois tipos de simulação para a validação deste trabalho, especialmente devido às particularidades do protocolo P2PWSRep.

6.2 Descrição do Ambiente de Simulação

Para facilitar o entendimento da rede simulada, é apresentada uma pequena rede P2P com apenas 10 nós, baseada na topologia do P2PWSRep simulada com 10.000 nós. O objetivo de mostrar esta pequena rede é fazer com que se possa visualizar a forma com que os nós se interligam, isto é, a topologia da rede.

Na topologia ilustrada na Figura 6.1, percebem-se algumas características do projeto P2PWSRep. Ele é baseado na rede Gnutella versão 0.4, assim não existem nós que centralizam funções específicas da rede, tais como ingresso na rede, busca de recursos e controle de reputação; e todas as interações entre os nós são realizadas por meio de encaminhamentos sucessivos.

Para configurar o ambiente de simulação no PeerSim, foram implementadas as classes base do protocolo P2PWSRep. O elemento principal da simulação é o nó (*peer*), ao qual estão associadas todas as operações, recursos que o nó compartilha e informações de reputação dos próprios nós e recursos. O simulador PeerSim tem uma interface chamada *Node* que é implementada pela classe *GeneralNode*, ambas

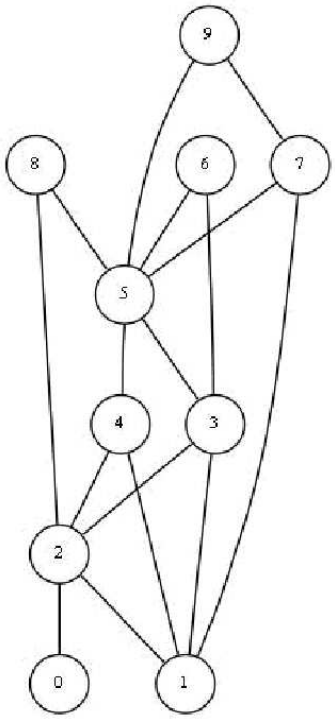

Figura 6.1: Exemplo da topologia da rede P2PWS.

do pacote *peersim.core*, que contém as principais operações para simular uma rede P2P, tais como: criação do nó, obtenção de identificador, ingresso e saída de nós da rede e implentação do protocolo núcleo da rede simulada. O nó do protocolo P2PWSRep foi criado extendendo-se a classe *GeneralNode*, sendo a classe resultante a *P2PWSRepNode*. A seguir, na Listagem 6.1, é mostrada uma parte do código da classe escrito em linguagem Java:

Listagem 6.1: Trecho da classe P2PWSRepNode

```
1 package p2pwsrep;
2
3 import java.util.Hashtable;
4 import peersim.core.GeneralNode;
5
6 public class P2PWSRepNode extends GeneralNode {
7   /* identificador do nó */
```

```
8           private long peerId;
9           private boolean strong;
10
11          /* tabela de nos */
12          private Hashtable tabPeer;
13          /* tabela de recursos */
14          private Hashtable tabResource;
15          /* tabela de reputacao de nos */
16          private Hashtable tabPeerReputation;
17          /* tabela de reputacao de recurso */
18          private Hashtable tabResourceReputation;
19          /* tabela de configuracao */
20          private Hashtable tabSettings;
21
22          /* usado em redes estruturadas */
23          private boolean supernode = false;
24
25          /* Constructor */
26          /** Cria uma nova instancia de P2PWSRepNode */
27          public P2PWSRepNode(String prefix) {
28                  super(prefix);
29                  supernode = false;
30
31                  tabPeer = new Hashtable();
32                  tabResource = new Hashtable();
33                  tabPeerReputation = new Hashtable();
34                  tabResourceReputation    = new Hashtable();
35                  tabSettings      = new Hashtable();
36          }
37  [...]
```

No código, pode-se notar que cada nó tem como principais atributos seu identificador (*peerID*) e as tabelas de nós vizinhos (*tabPeer*), tabela de recursos (*tabResource*), tabela de reputação de nós (*tabPeerReputation*) e de recursos (*tabResourceReputation*) e uma tabela que contém os dados de configuração do nó (*tabSettings*). Essas tabelas têm os campos descritos na seção de metadados, apresentada

no Capítulo 5. Fisicamente, os dados são armazenadas em arquivos XML e são manipulados por classes que fazem o *parsing* destes arquivos.

Outra parte fundamental para a simulação é a definição da estrutura da mensagem. O simulador oferece uma classe chamada de *SMessage*. Esta classe implementa a interface *cloneable*, que permite criar múltiplas cópias das mensagens. A seguir, na Listagem 6.2, há uma parte do código da classe *SMessage*, já com as adições referentes ao protocolo P2PWSRep.

Listagem 6.2: Classe SMessage

```
1  package p2pwsrep;
2
3  import java.sql.Timestamp;
4
5  import peersim.cdsim.CDState;
6  import peersim.core.Node;
7
8  public class SMessage implements Cloneable {
9      /* definicao dos tipos de mensagens do protocolo de reputacao */
10     public static final int QRY = 0;
11     public static final int FWD = 1;
12     public static final int HIT = 2;
13
14     /* busca da reputacao do nos que contem o recurso */
15     public static final int QRYREP = 3;
16     /* resposta da busca da reputacao */
17     public static final int HITREP = 4;
18     /* busca da reputacao do recurso encontrado */
19     public static final int QRYREPRESOURCE = 5;
20     /* resposta da busca da reputacao do recurso */
21     public static final int HITREPRESOURCE = 6;
22
23     private static int seq_generator = 0;
24
25     public int hops, type, seq, start;
26     public Node originator;
27     public String payload;  /* array de strings */
```

```
28      public Timestamp time;
29
30      public SMessage(Node originator, int type, int hops, Timestamp
            timestamp, String payload) {
31          this.originator = originator;
32          this.type       = type;
33          this.hops       = hops;
34          this.timestamp  = timestamp;
35          this.payload    = payload;
36          this.seq        = ++seq_generator;
37          this.start      = CDState.getCycle();
38      }
39  [...]
```

Destacam-se no código alguns pontos importantes, tais como os atributos *hops*, *type* e *timestamp*. O campo TTL indica o número de vezes que a mensagem será encaminhada antes de ser removida da rede; a cada nó por onde a mensagem passa, o valor de TTL é decrementado em 1 e o *hops* é incrementado em 1. Quando o valor chega a zero, a mensagem é descartada da rede. O Campo *type* determina o tipo de mensagem, podendo ser uma mensagem de pesquisa de recurso, pesquisa de reputação de nó, pesquisa de reputação do recurso, mensagem de resposta de busca a recursos, resposta de busca da reputação de um nó ou resposta de busca de reputaçao de um recurso. O campo *timestamp* armazena uma marca de tempo, indicando o exato momento em que a mensagem foi gerada. O *timestamp* permite controlar o momento em que as mensagens chegam ao destino. Existe um intervalo máximo de espera das respostas pelas mensagens de busca, chamado de intervalo de janela. Caso as mensagens cheguem fora da janela de tempo, isto é, após o intervalo de espera, elas serão descartadas pelo nó que iniciou a busca. Caso contrário, são armazenadas em uma fila para posterior processamento, ou seja, quando a janela de tempo se esgotar.

O núcleo do protocolo P2PWSRep foi implementado no simulador na classe *P2PWSRepProtocol* a partir de três interfaces. A *CDProtocol* (*Cycle Driven*), que possibilita a simulação de operações que devem se repetir a cada ciclo de execução

do simulador, de acordo com a configuração do experimento, realizada por meio de um arquivo de entrada de dados. O *EDProtocol* (*Event Driven*) utilizado por operações simuladas a partir da geração de um evento, tal como o recebimento de uma mensagem, solicitação de um *download*, etc.; e, por último a *Linkable* que é responsável por controlar as conexões entre o nó e seus vizinhos. Alguns métodos desta classe são destacados a seguir:

- ***send(Node n, SMessage mes)*** - Ao iniciar uma busca, uma mensagem com a *string* de busca para o recurso é gerada e enviada através do método *send()*. Uma fila é utilizada para controlar as buscas pendentes. Assim, para cada busca iniciada, é adicionada, em uma tabela, seu identificador. Essa tabela é implementada através de uma *hashtable*, tendo como campos chave o identificador da busca e seu *timestamp*. A cada mensagem que chega, o tipo da mensagem é checado e é verificado na tabela se a busca foi realizada pelo nó. Para cada tipo de mensagem, existe uma tabela específica para que não ocorra uma sobrecarga de processamento caso todas as mensagens estivessem em uma só tabela. Se a busca foi iniciada pelo nó, então a resposta é enfileirada em uma tabela, aguardando o término da janela de tempo. Quando a janela de tempo se esgota, todas as mensagens são processadas. Uma *thread* fica responsável por notificar o momento do processamento das mensagens enfileiradas. Todas as mensagens que chegarem após o fim da janela de tempo são descartadas. O tamanho da janela é associado ao TTL: para um TTL maior, uma janela maior; caso contrário, uma janela menor é utilizada. Este controle é realizado no nó que iniciou a busca de recursos e busca de reputação. As mensagens de resposta não possuem o controle de janela de tempo. Assim, quando um nó responde a uma requisição, seja ela qual for, a mensagem com o conteúdo da resposta é colocada no campo *payload* e o valor do campo *hops* é utilizado dependendo da profundidade em que este nó se encontra.

- ***updateRoutingTable(Node n, SMessage mes)*** - A cada mensagem que é enviada ou recebida, a tabela de roteamento é atualizada. Em cada nó, existe uma *hashtable* que controla o roteamento das mensanges seguindo o

mecanismo de roteamento do Gnutella, isto é, quando uma mensagem é enviada ou encaminhada, seu identificador é armazenado em uma tabela para que, quando uma mensagem de reposta chegue, seja feita uma verificação se o nó participou do envio ou encaminhamento da mesma.

- ***forward(Node n, Smessage mes)*** - Caso a mensagem tenha passado pelo nó, ele cópia (clona) a mensagem e a encaminha, através deste método, para seus vizinhos, exceto para o vizinho que lhe enviou. Caso contrário, a mensagem é descartada. Esse mecanismo permite que as mensagens retornem pelo mesmo caminho que percorreram para atingir um alvo. Ao clonar a mensagem, é realizada a alteração do valor do campo TTL e *hops*.

- ***process(SMessage mes)*** - Ao esgotar a janela de tempo, as mensagens são processadas através deste método. No processamento, é verificado o tipo de mensagem e, de acordo com o tipo, a mensagem é enviada para o método específico de processamento, seja para uma simples pesquisa a um recurso ou para uma busca de reputação. No caso da busca de recursos, o nó que recebe a mensagem extrai a *string* e pesquisa, em seu repositório de arquivos compartilhados, se há algum arquivo que possua a *string* buscada. Caso encontre, ele responde para o nó que originou a busca, enviando a mensagem por encaminhamento sucessivos. Caso a mensagem seja uma busca de reputação, então o nó que recebeu o pedido verifica em sua tabela de reputação de nós se ele conhece o nó procurado. Caso seja um nó com que ele já interagiu, então reúne as informações para o cálculo da reputação local, que são o número de transações satisfatórias e o número de transações insatisfatórias, realizando o cálculo de acordo com a Fórmula 5.1. Então, o resultado é enviado para o nó interessado. Caso a reputação do nó procurado seja zero, a resposta não é enviada.

A topologia e o mecanismo de roteamento das mensagens foram implementadas na classe é *GnutellaProtocol*, mostrada na Listagem 6.3, que implementa o protocolo Gnutella versão 0.4.

Listagem 6.3: Classe GnutellaProtocol

```
1  public class GnutellaProtocol extends P2PWSRepProtocol{
2      /** Creates a new instance of GnutellaProtocol */
3      public GnutellaProtocol(String prefix) {
4          super(prefix);
5      }
6
7      public void nextCycle(Node node, int protocolID) {
8          super.nextCycle(node,protocolID);
9          /**If we have to produce a query*/
10         String data = this.pickQueryData();
11         if (data != null) {
12
13         SMessage m = new SMessage(node, SMessage.QRY, 0, new Timestamp
               (System.currentTimeMillis()), data);
14
15         /** This node already is knows */
16         if (!this.match(m)){
17           //If I don't
18           for (int i = 0; i < this.view.size() &&
19             i < this.degree(); i++){
20              System.out.println("Node: "+node.getID()+" sending to "+
21              ((Node) this.getNeighbor(i)).getID()");
22              this.send((Node) this.getNeighbor(i) , m);}
23         } else {
24            /* I know this node, so I don't need to propagate */
25            this.messageTable.put(m, new Integer(0));
26            [...]
27         }
28     }
29 [...]
```

Todas as mensagens são armazenadas em uma tabela quando chegam no nó que solicitou a reputação, esperando pelo fim da janela de tempo. Quando o tempo se esgota, todas as mensagens são processadas. É realizado um cálculo com todos os valores recebidos, aplicando uma média aritmética para obter a reputação recebida

da rede, pela Fórmula 5.2. Com este valor, é realizado outro cálculo, utilizando a Fórmula 5.3 para se obter a reputação final. O novo valor será armazenado no campo *previousRep*, sobrescrevendo o valor anterior, contido na tabela *tabReputation* do nó que iniciou a busca.

A próxima etapa do protocolo é a busca da reputação dos recursos. Esta operação é mais rápida, pois não envolve cálculo distribuído. Somente é realizada uma consulta na rede para verificar se o recurso tem alguma restrição no nó em que ele está disponível. Assim, é realizada uma consulta aos nós da rede através de uma mensagem *QUERYREPRESOURCE*, contendo na sua carga útil uma lista dos recursos a serem pesquisados, para que os que já obtiveram o arquivo do nó possam fornecer a informação sobre os recursos. Isso é importante para diminuir o número de mensagens do protocolo P2PWSRep na rede, caso fosse utilizada uma mensagem para consultar a reputação de cada recurso.

Ao final deste processo, os recursos são ranqueados de acordo com a reputação dos nós que os compartilham e a lista é exibida ao usuário, excluindo-se os recursos com reputação duvidosa. Havendo empate, é considerado o nó com maior número de transações satisfatórias. Caso persista o empate, o nó mais antigo na rede é melhor ranqueado. Cada nó possui uma tabela de reputação de recursos, sendo que a estrutura da tabela é o identificador do recurso, identificador do nó e um *array* de códigos de reputação do recurso. Para a tabela não crescer muito e comprometer o desempenho do protocolo, somente são armazenados os códigos de recursos corrompidos, infectados ou inexistentes. Outros códigos podem ser criados e incluídos no protocolo.

Todos os processos acima são iniciados pelo método *main()* da classe *Simulator*. Para carregar os dados dos metadados de arquivos e reputação, são utilizadas classes de inicialização, para fornecer a carga de trabalho da simulação e os dados iniciais para configuração de toda a rede P2P.

- **P2PWSRepInitializerFiles** - Instancia aleatoriamente um conjunto de arquivos para os nós da rede.
- **P2PWSRepInitializerReputation** - Atribui valores de reputação aos nós

da rede.

- **P2PWSRepInitializerReputationFiles** - Atribui valores aleatórios de reputação aos recursos, de forma que se possa ter parte dos arquivos da rede íntegros, corrompidos, infectados ou inexistentes, de acordo com um percentual estabelecido no cenário do experimento.

- **P2PWSRepInitializer** - Faz a inicialização da simulação, onde é definida a arquitetura da rede, número de nós, protocolo da rede e execução dos experimentos.

A primeira inicialização dos dados é o preenchimento aleatório da tabela de recursos, realizada pela classe *P2PWSRepInitializerReputationFiles*. A segunda inicialização é a de reputação, onde valores de reputação são atribuídos aleatoriamente aos nós. Esses valores dividem os nós em basicamente quatro grupos: ótimos, bons, regulares, ruins e péssimos, sendo que os grupos estão nos seguintes intervalos, respectivamente, 1 e 0.8; 0.79 e 0.6; 0,59 e 4, 0.39 e 2; 0.19 e 0. Os valores atribuídos aos nós são transações satisfatórias, insatisfatórias e última reputação conhecida do nó *previousRep*. A classe responsável por esta inicialização é a *P2PWSRepInitializerReputation*. A terceira inicialização é a da reputação dos arquivos, implementada pela classe *P2PWSRepInitializerReputationFiles*, caracterizando os arquivos somente por valores associados a um status: 1 - corrompido, 2 - infectado, 3 - adulterado, 4 - inexistente.

Todas as inicializações são realizadas durante a própria inicialização do simulador. Cada nó da rede é instanciado com base em uma classe padrão para o nó. A arquitetura P2PWSRep define um nó que é instanciado na inicialização. Esse nó recebe um identificador e é inserido aleatoriamente na rede através de uma distribuição bayesíana para ligação com os demais nós da rede. A classe que define um nó da rede P2PWS é *P2PWSRepNode*.

A configuração do ambiente de simulação, tais como, tamanho da rede, inicialização, logs de saída é realizada pelo arquivo de configuração do simulador, mostrado no Anexo 1.

Para que se possa visualizar o mecanismo de inicialização da simulação e encaminhamento das mensagens, é apresentado um trecho do log de saída de uma simulação, no Anexo 2.

Na saída mostrada no Anexo 2, foram omitidos os pacotes referentes às buscas de reputação. Pela característica do protocolo Gnutella, as mensagens serão encaminhadas para os nós vizinhos até que seu TTL seja esgotado. Para as mensagens de reputação, bem como outros tipos de mensagens, é verificado se a mensagem já passou pelo nó. O teste sendo positivo, a mesma não será processada, nem encaminhada novamente. Isso conseguiu reduzir o número de mensagens de reputação trafegando na rede.

A validação do protocolo P2PWSRep foi realizada em vários cenários, para que fossem obtidos resultados que demonstrem o comportamento do protocolo em apontar corretamente a reputação de um nó ou recurso e o impacto do protocolo no geração de tráfego na rede. Nas próximas seções são apresentados os resultados para cenários distintos. São eles: Análise do Protocolo de Reputação dos Nós, Análise do Protocolo de Reputação dos Recursos e Análise do Protocolo Quanto à Carga na Rede.

6.3 Cenário 1 - Análise do Protocolo de Reputação dos Nós

O objetivo da simulação para o protocolo P2PWSRep, em relação a reputação dos nós, é validar seu mecanismo de obtenção de dados e cálculo da reputação, sem que haja comprometimento do funcionamento da rede. A importância da reputação na rede P2PWS é fazer com que nós novos ou de baixa reputação possam melhorar sua reputação caso cooperem de forma satisfatória na rede, isto é, compartilhando seus recursos e mantendo somente recursos íntegros em seu repositório. Além disso, mostrar que, quando um nó tem boa reputação, para mantê-la o mesmo tem que continuar cooperando de forma positiva, caso contrário sua reputação será decrementada pelo protocolo P2PWSRep.

6.3.1 Descrição do Cenário

Neste cenário, uma rede com 10.000 nós foi "povoada" com recursos íntegros e corrompidos e com valores de reputação aleatórios. Estes dados foram inseridos nas tabelas de reputação de nós e recursos. Foram elaborados para este primeiro cenário quatro experimentos, onde um nó $\boldsymbol{i}$ será o nó observador, isto é, o nó que fará as requisições de reputação ao nó observado, chamado de $\boldsymbol{j}$. Os demais nós da rede fornecerão dados relativos aos valores de reputação de $\boldsymbol{j}$, baseando-se nas transações satisfatórias e insatisfatórias ocorridas com o nó $\boldsymbol{j}$, contidas em sua tabela de reputação. O nó $\boldsymbol{i}$ tem um valor de reputação do nó $\boldsymbol{j}$, em sua tabela de reputação, no campo *previousRep*, igual a 0.8. O valor da reputação do nó $\boldsymbol{j}$ obtido da rede foi de 0.655.

Foi considerado *alfa* igual a 0.4 para o protocolo P2PWSRep, o que possibilita que haja um ganho de reputação e uma perda gradual, fazendo com que a curva de perda seja mais suave, valorizando mais as transações recentes. Cada situação foi simulada de forma independente, sendo os resultados apresentados em um mesmo gráfico (Figura 6.2) para que se possa fazer uma comparação entre as três abordagens e verificar como o protocolo funciona e age em diferentes situações na rede P2P. A ferramenta utilizada para construir os gráficos foi o GnuPlot [1].

6.3.2 Resultados

Na primeira simulação, um nó não irá cooperar na rede, assim ele não terá realizado ao final da simulação transações satisfatórias ou insatisfatórias. O objetivo deste primeiro cenário é mostrar que, um nó que não realize os dois tipos de transações citados anteriormente, não pode manter sua reputação atual, já que este comportamento caracteriza um nó carona ou egoísta, um nó que só consome os recursos da rede, mas não compartilha seus recursos com os demais. O protocolo de reputação P2PWSRep age nesse nó reduzindo sua reputação, porém mais lentamente que a de um nó com um histórico negativo. Percebe-se que a reputação, após várias consultas diminuem, isso retrata que um nó que não é muito acessado

[1] Página Inicial: http://www.gnuplot.info

perde reputação. Este comportamento é mostrado no gráfico da Figura 6.2, onde se percebe que, ao final da simulação, que durou 100 ciclos, o valor da reputação é igual a 0.653, que representa o valor do campo *previousRep* do nó observado para o nó que finalizou esta simulação em busca da reputação. Isso representa uma perda de reputação em termos percentuais de 18.37%. Em termos reais, a simulação equivale a estressar o nó pelo número de transações. O resultado da simulação equivale ao esperado na definição do protocolo P2PWSRep, já que, em relação a reputação, houve uma queda de 0.655 para 0.653. Isto mostra que apesar de não existir uma centralização das informações de reputação na rede, os valores de reputação sempre são bem próximos quando consideramos uma amostragem geral da reputação de um nó qualquer na rede, validando a idéia do protocolo P2PWSRep, que, mesmo descentralizado, pode apontar para um valor de reputação que seja bem semelhante em toda a rede.

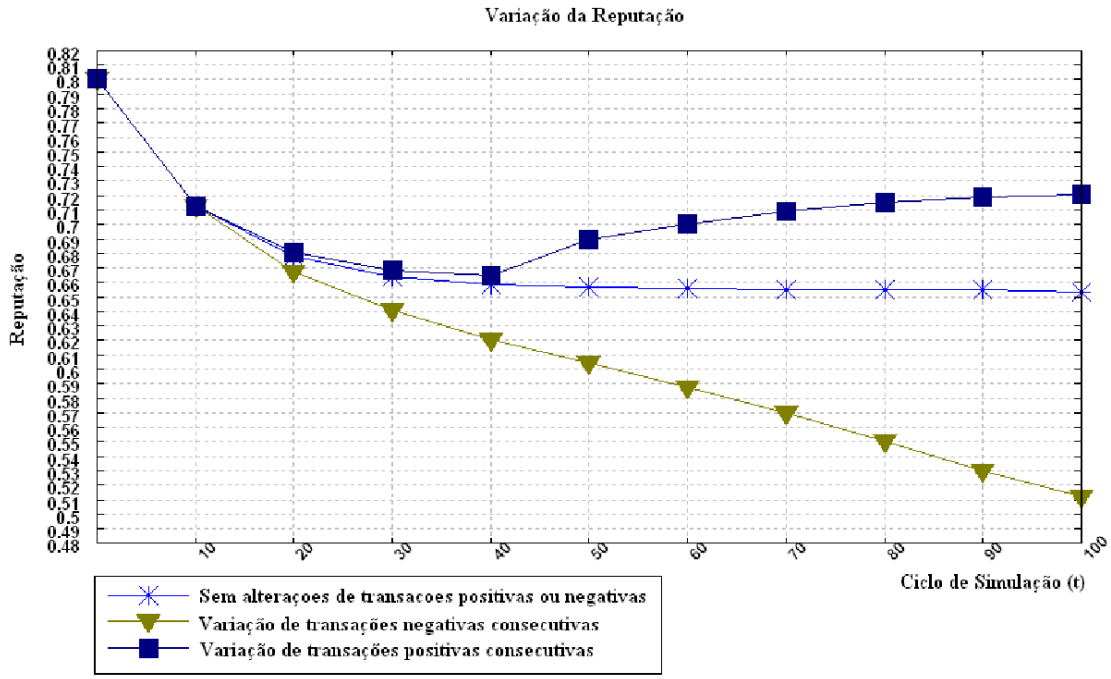

Figura 6.2: Gráfico de variação da reputação.

A segunda simulação, utilizou a mesma caracterização da carga de trabalho, 10.000 nós na rede e valor de *alfa* de 0.4, também com uma reputação inicial de 0.8

para o nó observado. O comportamento atribuído para o nó é que o diferencia neste cenário, onde o nó realizará várias transações positivas, enquanto sua reputação é pesquisada na rede pelo nó observador. Neste caso, espera-se que, ao final da simulação, a reputação apresente um ganho. Porém, teremos que considerar o peso de sua reputação obtida na rede. O nó interessado pela reputação pode ter um valor no seu campo *previousRep* que não caracterize o comportamento do nó ultimamente, isso só pode ser obtido na rede através de uma consulta à reputação. Após a busca, o nó obteve uma reputação de 0.655.

Observando o gráfico da Figura 6.2 em relação à curva que mostra a variação das transações positivas consecutivas, faz-se com que a baixa reputação do nó obtida da rede provoque uma queda na primeira metade da simulação, fazendo com que na segunda metade, a partir do ciclo 50 a reputação começe a ter um ganho, finalizando em 0.721, o que representou um acréscimo de 9.87%. Pode parecer estranho que um nó realizando somente transações satisfatórias tenha uma queda de reputação no início da simulação, mas isto é justificado pela natureza do protocolo de reputação P2PWSRep, que leva em consideração as experiências anteriores dos demais nós da rede com este nó, bem como a experiência anterior do nó que está negociando uma troca de recursos com ele. O que era esperado como resultado ao fim da simulação, pela concepção do projeto P2PWSRep, foi obtido.

O próximo experimento considera o nó $\boldsymbol{j}$, o nó observado, realizando transações insatisfatórias durante a simulação, onde o nó i faz sucessivas requisições de sua reputação. Considerando que transações insatisfatórias são prejudiciais à reputação de um nó, o gráfico da Figura 6.2 deixa bem claro que o protocolo P2PWSRep penaliza o nó com um decréscimo de reputação considerável, como mostrado na curva de reputação mais acentuada após estas transações insatisfatórias ao final dos 100 ciclos de simulação. Assim, tendo a reputação de $\boldsymbol{j}$ de 0.8 na tabela de reputação do nó $\boldsymbol{i}$, ao final da reputação seu valor equivale a 0.512, representando uma perda de reputação de 36% em relação à sua reputação inicial. Após a ocorrência de várias transações negativas, a reputação do nó cai, influenciada pela reputação anterior, também baixa, e pelo comportamento negativo do nó na rede P2P.

6.3.3 Análise dos Resultados

Os resultados mostrados no gráfico da Figura 6.2 demonstram que o protocolo P2PWSRep conseguiu apresentar valores esperados para o comportamento dos nós, sendo que seria interessante considerar que o usuário pode ter em sua interface gráficos gerados pelo protocolo P2PWSRep que mostram o risco ou benefício de interagir com determinados nós na rede. É bom ressaltar que o P2PWSRep não toma decisões autônomas para restringir o acesso a um nó que tenha baixa reputação, até porque uma característica de redes que utilizam mecanismos de reputação é que os nós novos tenham reputação mais baixa que os nós mais velhos, e para ganhar reputação devem oferecer recursos. O protocolo P2PWSRep mantém associado ao *peerID* à idade do nó. Assim, quando o usuário tomar conhecimento da reputação de um nó, poderá identificar se ele é um nó antigo ou novo na rede. Controlar a reputação dos nós pode ser algo inclusive independente do controle de reputação de recursos. Porém, desaconselha-se que a reputação de nós e recursos do protocolo P2PWSRep sejam trabalhadas de forma independente, pois a definição do protocolo considera mais segura a aplicação das duas abordagens.

Para ajudar na compreensão do protocolo de reputação P2PWSRep, o gráfico da Figura 6.3 apresenta a evolução da reputação de um nó na rede. Este nó foi observado na simulação enquanto as transações satisfatórias e insatisfatórias ocorriam aleatoriamente com vários nós da rede, que interagiam com ele em busca de seus recursos. Alguns recursos do nó foram assinalados como corrompidos para que nas requisições sejam geradas ações negativas e possam reflitir a reputação do nó. Em vinte ciclos de simulação, o nó parte de uma reputação de 0.8 para 0.51, perdendo 36.25% de sua reputação inicial. Percebe-se no gráfico que, nos momentos em que o nó realiza interações positivas, ele começa a ganhar reputação, como observado entre os ciclos 9 e 11, porém as transações insatisfatórias se tornam novamente a maioria, fazendo com que a reputação descresça novamente.

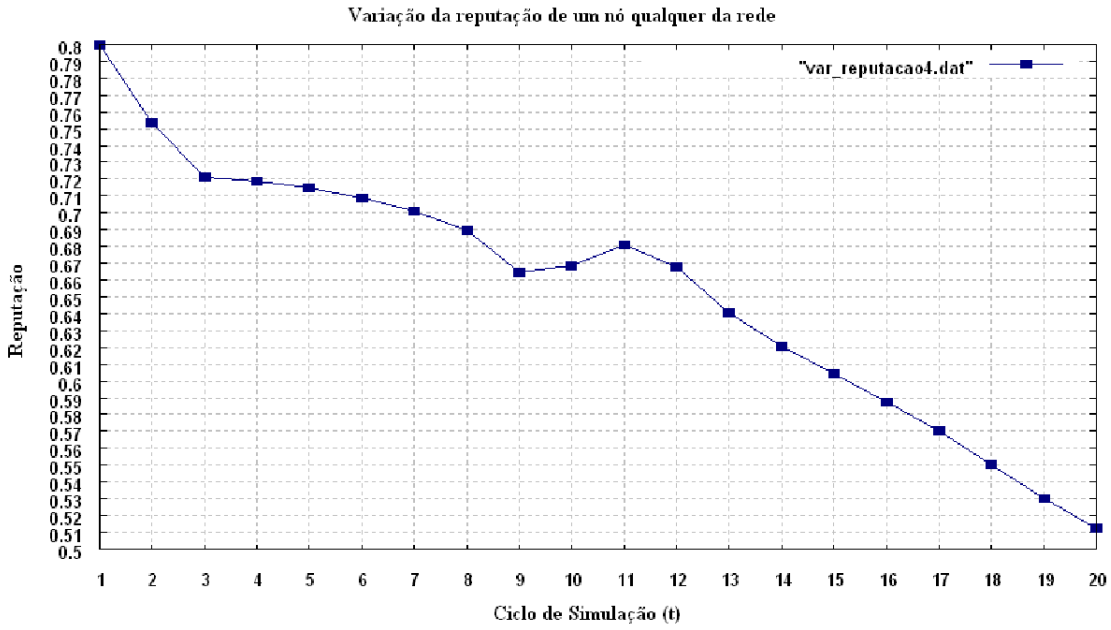

Figura 6.3: Gráfico de variação da reputação de um nó da rede.

6.4 Cenário 2 - Análise do Protocolo de Reputação dos Recursos

O objetivo da simulação aplicada nesta seção é comprovar a eficiência do protocolo P2PWSRep em relação ao controle de recursos corrompidos, infectados ou com alterações de metadados a fim de indisponibilizar o acesso aos mesmos na rede P2P. O protocolo P2PWSRep foi elaborado com o objetivo de fazer também o controle da reputação dos recursos, já que o prejuízo provocado por um nó egoísta é o desperdício de banda e interações na rede, tornando-a mais lenta, porém o prejuízo causado por recursos corrompidos na rede é bem maior, pois as taxas de transmissão desperdiçadas por arquivos corrompidos são bem maiores que os causados por nós egoístas, sendo isso uma ameaça à rede. Além disso, um arquivo infectado é uma ameaça ao nó participante da rede, e por conseguinte, à rede. Evitar que estes recursos permaneçam na rede ou apontar para os nós que os compartilham é um dos objetivos do protocolo P2PWSRep.

6.4.1 Descrição do Cenário

Neste cenário, uma rede com 10.000 nós foi populada com recursos íntegros e corrompidos. Os arquivos corrompidos foram disponibilizados em diversos nós da rede. Vários nós foram utilizados para fazer requisições aos recursos contidos nestes nós que compartilham estes arquivos, e foi observada a evolução das requisições aos nós que possuem estes arquivos danificados. A simulação durou 25 ciclos e foram iniciadas as requisições em número de mil por ciclo. A cada requisição, o mecanismo de obtenção de reputação de recurso foi utilizado, fazendo com que os nós da rede obtivessem os valores da reputação dos recursos. Os mesmos recursos foram espalhados na rede de forma que uma parte seja corrompida e outra parte, seja íntegra. Para realização da simulação foi utilizado um mecanismo de inicialização de recursos aleatoriamente na rede, onde os nós recebem recursos que podem ser íntegros ou não. Alguns nós que possuem recursos corrompidos foram assinalados para serem observados durante a simulação. Apesar do protocolo P2PWSRep possuir um mecanismo adaptativo para controle de TTL, nesta simulação utilizou-se um valor fixo de 4.

6.4.2 Resultados

Os resultados da simulação são apresentados no gráfico da Figura 6.4, onde se percebe que inicialmente o número de requisições era de 997 requisições por ciclo aos nós observados. A cada ciclo, dos 25 considerados, os nós executaram o algoritmo de reputação para detecção dos arquivos corrompidos. Quando uma reputação obtida esta "corrompida", o identificador do recurso e do nó são adicionados à tabela de reputação do nó que iniciou a busca. Isso permite que, nas próximas pesquisas de busca na rede, da reputação deste recurso, seja notificado ao nó requisitante sobre a reputação ruim, fazendo que o mesmo busque outra fonte. O protocolo P2PWSRep também notifica o nó que possui o recurso corrompido para que o mesmo seja apagado, além de incrementar o número de transações insatisfatórias com o nó, pois o mesmo está disponibilizando conteúdo poluído na rede.

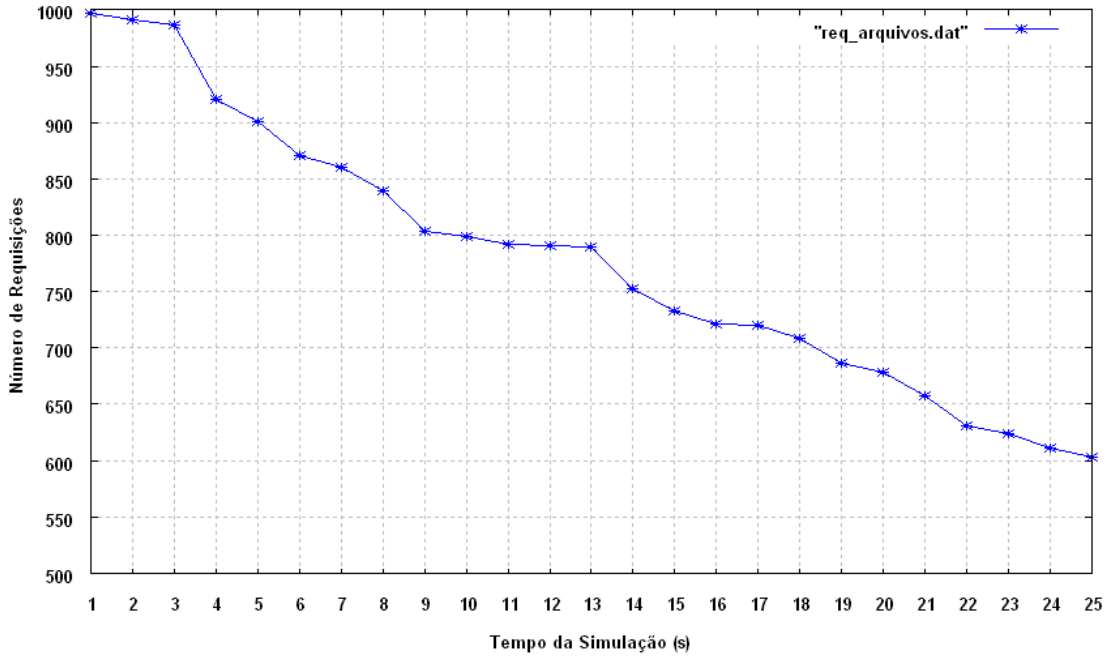

Figura 6.4: Gráfico do número de requisições de arquivos corrompidos.

6.4.3 Análise dos Resultados

O gráfico da Figura 6.4 mostra que, no final da simulação, os números de requisições aos nós que possuem os recursos corrompidos diminuiram para 603 requisições por ciclo, o que comprova que a reputação dos recursos corrompidos foi propagada na rede e contribuiu para redução do tráfego da rede e aumento da segurança, já que fez com que as fontes de arquivos corrompidos fossem identificadas e que, requisições que seriam realizadas para estas fontes fossem redirecionadas a outros nós. Uma opção do protocolo P2PWSRep é apagar o arquivo da origem. Porém, isso abre para um novo tipo de ataque. O que pode ser feito é uma pré-negociação da remoção do arquivo, fazendo com que o nó, mesmo com o arquivo corrompido, tenha um ganho de reputação por contribuir na eliminação do conteúdo poluído da rede P2P. A redução de requisições a arquivos corrompidos foi considerável, 39.51%, em uma simulação de 25 ciclos. Claro que deve ser considerado o volume de conteúdo poluído, sendo que o mesmo permaneceu constante durante a simulação, não tendo sido injetados novos arquivos corrompidos.

6.5 Cenário 3 - Análise do Protocolo quanto à Carga na Rede

A análise do protocolo P2PWSRep em relação à carga gerada na rede tem como objetivo ter conhecimento e comprovar que é viável utilizar o protocolo, sem que seja adicionado um número grande de interações que possam aumentar o fluxo de dados de controle na rede, além de aumentar o tempo de resposta das operações, como busca de recursos, *download* de recursos, dentre outros, já que essas operações envolvem a ativação do algoritmo de reputação.

6.5.1 Descrição do Cenário

Na simulação, foi utilizada uma rede de 10.000 nós, no mesmo ambiente computacional das simulações anteriores. Os dados observados na simulação foram o número de mensagens trafegando na rede, considerando a variação do TTL para verificar o alcançe da busca, isto é, o número de nós consultados na rede. Foram injetadas várias consultas na rede, geradas por nós que fizeram operações de buscas de arquivos, de reputação de nós e de recursos. Medir o tempo de resposta das operações e o número de nós consultados na rede e os nós que efetivamente responderam por alguma das requisições foram os pontos principais observados nesta simulação.

6.5.2 Resultados

O primeiro gráfico, ilustrado na Figura 6.5, mostra que, para um TTL de valor 1 e 2, o alcance da rede é muito pequeno em relação ao tamanho da mesma. O valor de TTL a partir de 3 mostrou-se bom, já que se tem como resultado quase 50% de nós da rede consultados. Para o TTL igual 4, o número de mensagem na rede gerado foi de 12.053, sendo que o número de nós que foram consultados foi de 7.807, representando 78% dos nós da rede. Para o TTL igual a 5, o número de mensagens teve um aumento de pouco mais de 5.000 mensagens, para um aumento de alcance de pouco mais de 1.600 nós. Nos TTLs 6 e 7, o alcance é praticamente total na

rede, porém o número de mensagens geradas gira em torno de 20.000 trafegando na rede, com a quantidade de nós consultados bem próxima de 10.000, isto é, quase a totalidade da rede.

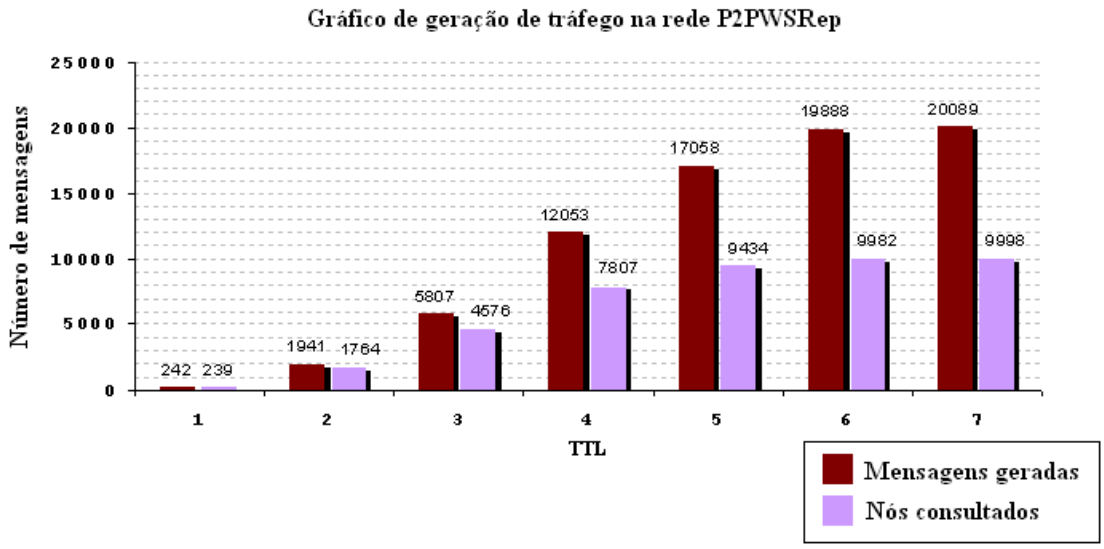

Figura 6.5: Gráfico do tráfego de mensagens *versus* alcance de busca.

A Figura 6.6 traz os resultados de uma simulação para o tempo de resposta de consultas de reputação. Os dados avaliados foram o número de mensagens de reputação recebidas como resposta, o número de nós consultados, de um total de 100 nós que possuem informação de reputação para o nó buscado e o tempo de resposta. Os 100 nós foram assinalados na inicialização do experimento. No gráfico, vê-se uma análise mais específica do tempo de resposta de uma consulta de reputação *versus* a quantidade de nós que respondem, dependendo do valor do TTL associado à consulta. A simulação aqui considerada populou 100 nós da rede em relação à reputação de um nó que será pesquisada. O tamanho da rede é de 10.000 nós. Para TTL 1 não houve resposta em relação a reputação do nó buscado. Isso significa que a profundidade da busca foi pequena demais para obter dados de reputação. Com TTL 2 e 3, o volume de nós que responderam foi suficiente para que o protocolo P2PWSRep tivesse dados para o cálculo da reputação, com um tempo de resposta bom e dentro do esperado, entre 3 e 5 segundos. Para TTL 4 e 5, o número de nós

consultados foi bem próximo: 69 e 72 nós responderam, respectivamente, ao pedido de reputação. Porém, o tempo de resposta foi alto, entre 8 e 11 segundos. Para TTL 6 e 7, houve um aumento significativo no tempo de reposta, chegando a ser quase o dobro em relação aos TTLs desses valores, porém a resposta dos nós foi quase 100% dos nós que têm dados de reputação a respeito do nó pesquisado.

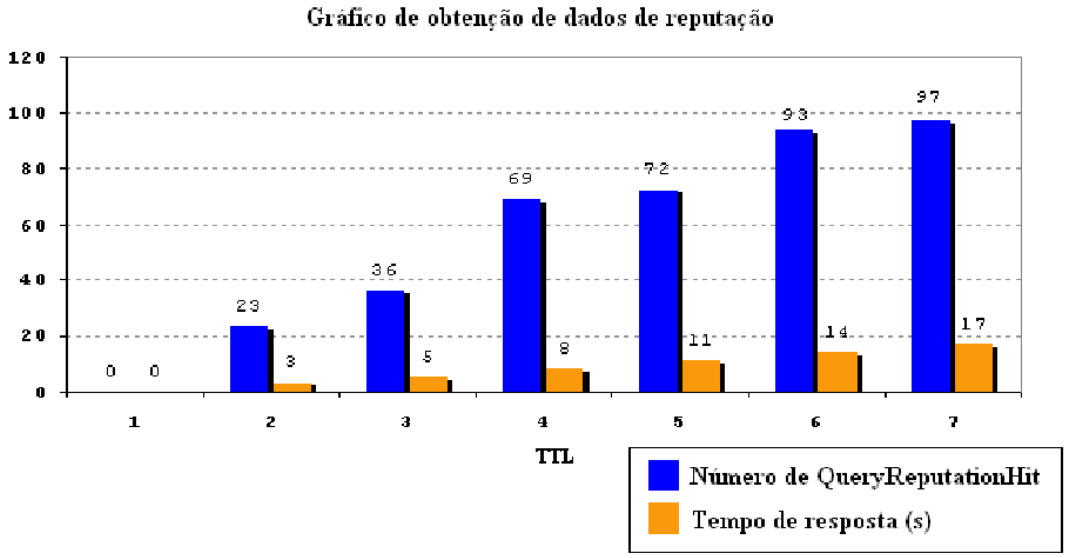

Figura 6.6: Gráfico do número de respostas de reputação versus tempo.

6.5.3 Análise dos Resultados

Diante dos dados, pode-se verificar que quanto maior o TTL, maior a profundidade da consulta. Porém, o volume de mensagens trafegando na rede pode sobrecarregar os nós, já que os mecanismos de processamento de roteamento e reputação utilizam filas (tabelas *hashtable*) para fazer esses controles, acarretando em um número muito grande de mensagens armazenadas nas filas e um número muito grande de réplicas trafegando na rede. O TTL usual para a rede P2PWS é 3. O protocolo P2PWSRep possui um mecanismo adaptativo que ajusta o TTL para um número de respostas que pode ser aceito pelo protocolo ou usuário. Assim, o protocolo P2PWSRep pode variar seu alcance de 2 a 6, fazendo que a maioria das consultas possa obter respostas sem uma profundidade muito grande, logo nos

primeiros valores de TTL.

Isso nos permite avaliar que o atraso na resposta da reputação pode ser prejudicial pelo tamanho da janela de tempo definida no nó que iniciou a busca. O protocolo P2PWSRep utiliza um mecanismo adaptativo para regular o TTL, de forma que isso possa ser de acordo com a posição do nó na rede e dos recursos que ele busca, mas isso não significa que haja uma limitação da profundidade da rede, mas sim uma diminuição do número de mensagens devido à obtenção de recursos e dados de reputação em fontes mais próximas. O valor da janela de tempo para a pesquisa, então, será regulada de acordo com o TTL. Assim, o protocolo fica livre para fazer controle de TTL adaptativo até o TTL de valor 6, sendo que o valor mínimo que o protocolo P2PWSRep trabalha é 2.

6.6 Considerações Finais

Pelos resultados apresentados nos três cenários de simulação realizados, conclui-se que o protocolo P2PWSRep teve um comportamento esperado para o que foi proposto e, em comparação a outros protocolos, um desempenho bastante razoável, já que foi definido em uma arquitetura descentralizada e sem qualquer mecanismo de otimização de buscas e consultas na rede. O número de mensagens geradas na rede foi melhorado pela janela de tempo estabelecida no nó requisitante, de buscas a recursos e reputação, pelo TTL adaptativo que possibilita uma redução no número de requisições para TTLs menores e pela forma com que a reputação de nós e recursos é consultada, isto é, não há uma mensagem de consulta para cada nó ou recurso, mas uma mensagem que contém um *array* de nós ou recursos que serão consultados.

O tempo de resposta em relação às consultas de reputação também mostrou que o protocolo tem um mecanismo de cálculo simples, sem envolver um número muito grande de interações ou controles mais seguros de troca de mensagens, utilizando criptografia, como ocorre em outros protocolos.

O protocolo P2PWSRep foi projetado para controlar a reputação de nós e

recursos, sendo que os dados obtidos da reputação mostraram que o número de requisições a arquivos corrompidos foi reduzido em 40%, levando à conclusão que o protocolo consegue eliminar da rede as interações com nós que detêm estes recursos espúrios. A evolução da reputação de nós também foi como esperada, onde no gráfico da Figura 6.2 são mostradas três curvas de variação de reputação, que permitem visualizar um comportamento de reputação bem comum em redes sociais ou P2P.

Uma análise geral do resultado da aplicação do protocolo P2PWSRep mostra que o nó que não compartilha seus dados perde reputação naturalmente, devido à influência histórica de sua reputação ser inferior à atual, além do que o cálculo da reputação utilizado pelo protocolo P2PWSRep preza pela reputação mais recente, dada a parametrização escolhida. Quanto à reputação de recursos, a diminuição do número de requisições a fontes comprometidas mostra mais um ponto positivo no protocolo, além dos mecanismos inseridos para ajuste de busca na rede, diminuindo o grau de inundação, número de mensagens e tempos de reposta.

Capítulo 7

Conclusão

A definição do protocolo de reputação P2PWSRep possibilita o emprego de técnicas para identificar e evitar que nós mal-intencionados prejudiquem o desempenho da infraestrutura da rede P2P, bem como a diminuição do conteúdo poluído nessas redes. O trabalho também busca demonstrar que a interoperabilidade e extensibilidade das redes P2P é viável, com a utilização de serviços web.

7.1 Contribuições

Reduzir o número de nós mal-intencionados e conteúdo poluído das redes P2P não é uma tarefa trivial, principalmente devido às particularidades dessas redes, que prezam por anonimato e uma independência em relação a elementos moderadores. Propor um protocolo de reputação e apresentar um modelo baseado em serviços foi um desafio, pois a alta escalabilidade das redes P2P e o volume do tráfego gerado por esse tipo de rede exige que as interações do protocolo não adicionem sobrecargas à rede. O protocolo P2PWSRep prezou, desde sua concepção, pela definição de mecanismos para redução do *overhead* de processamento de mensagens, mas sem deixar de lado a segurança contra os principais ataques a que estas redes estão susceptíveis. Dentre as contribuições deste trabalho, podemos enumerar:

- Arquitetura simples e que proporciona escalabilidade, uma vez que utiliza uma arquitetura de redes descentralizada e um modelo orientado a serviços,

que reduz as limitações impostas por *firewalls*, *proxies*, *IPS* e outros tipos de ferramentas de segurança de redes, já que as mensagens trafegam sobre o protocolo HTTP;

- Protocolo genérico, que pode ser utilizado por vários tipos de aplicações, pois o mesmo faz a avaliação da reputação em relação a um recurso qualquer, considerando características da aplicação e não do recurso compartilhado;

- Cálculo de reputação distribuído realizado em três etapas. A primeira é a obtenção da reputação do nó (Fórmula 5.1) através de uma busca na rede. Na segunda etapa, é realizada uma média aritmética (Fórmula 5.2) no nó que busca a reputação a fim de normalizar os valores obtidos da rede. Por último, é realizado o cálculo da reputação utilizando uma média ponderada exponencial (Fórmula 5.3), considerando o último valor da reputação obtido da rede, contido no campo *previousRep* e o valor atual obtido. O parâmetro *alfa* da Fórmula 5.3, da média ponderada exponencial é utilizado para dar mais ênfase à reputação atual ou a anterior. No caso do P2PWSRep, o valor de *alfa* escolhido foi de 0.4, dando mais ênfase à reputação atual, porém sem desconsiderar a reputação anterior do próprio nó, já que considerar somente a reputação dada por outros nós poderia comprometer a transação;

- Proporciona interoperabilidade e extensibilidade ao seguir a metodologia de orientação a serviços e a definição de metadados em formato XML. A definição de suas interfaces possibilita o protocolo ser extensível, podendo ser utilizado por vários tipos de aplicações de redes P2P, tais como, compartilhamento de arquivos, grades de computadores que compartilham recursos, sites de vendas de mercadorias, análise de risco de mercado financeiro, sites de relacionamentos, dentre muitas outras. A extensibilidade possibilita a adição de mecanismos de segurança, alterações de mecanismos de busca ou até mesmo a topologia da rede, de forma mais simples e modular, além da possibilidade de integração com redes P2P já existentes, pela definição de *gateways*.

- Redução na geração do número de mensagens, em relação a outros protocolos,

utilizando vários mecanismos. O primeiro deles é a busca de reputação de nós e recursos utilizando um *array* com vários nós ou recursos a serem consultados, ao invés de gerar uma mensagem para cada nó ou recurso pesquisado. O *array* para as buscas é bem estruturado, contendo somente o identificador de cada nó ou recurso. A resposta adiciona um campo ao *array* que é o valor de reputação para os nós ou código de avaliação do recurso. O segundo mecanismo é o TTL adaptativo associado a uma janela de tempo para processamento das mensagens, que possibilita que as buscas utilizem uma profundidade de acordo com a localização dos nós na rede, em relação aos recursos ou reputações buscadas. Esses mecanismos contribuem diretamente para a redução do número de mensagens na rede, apresentando um bom desempenho e tempo de resposta, como mostrado nas simulações realizadas, mesmo para uma rede P2P descentralizada como a aqui proposta.

7.2 Trabalhos Futuros

As redes P2P têm sido muito utilizadas em diversos tipos de aplicações. Contudo, um fator negativo em relação ao seu crescimento é o surgimento de novas técnicas para comprometer o funcionamento das mesmas. Algoritmos de reputação têm sido uma boa solução para diminuir o impacto desses ataques e reduzir o número de nós mal-intencionados e a poluição dessas redes, causadas pelo grande volume de recursos corrompidos ou infectados. Todavia, as redes P2P não estão livres desses ataques, pois muitos deles trabalham abaixo da camada do protocolo de reputação, atingindo o mecanismo de ingresso na rede, geração de identificadores, roteamento de mensagens, etc. Assim, durante a elaboração do protocolo P2PWSRep foram identificados algumas contribuições que podem ser adicionadas por trabalhos futuros, tais como:

- Adição de mecanismos de segurança ao protocolo, através da utilização de criptografia segundo o padrão *WS-Security* (OASIS), proporcionando suporte à integridade e confidencialidade das mensagens, ao permitir que os integrantes

da rede se comuniquem trocando mensagens criptografadas e assinadas digitalmente. Além disso, validação de identificadores de nós e recursos para evitar ataques do tipo *sybil* ou *whitewashing*;

- A implementação do protótipo em si, da rede P2PWSRep, pois foram definidos o modelo de serviços e realizadas simulações do algoritmo de reputação. Obter resultados de uma aplicação real será muito interessante para consolidar o protocolo P2PWSRep como uma boa solução para aplicações P2P;

- Adaptação do protocolo P2PWSRep para utilizar uma arquitetura centralizada ou com supernós;

- Introdução de mecanismos de resposta automática a ataques, tais como detecção e remoção de nós da rede identificados como maliciosos e detecção e remoção de recursos da rede corrompidos ou infectados;

- Utilização de outras métricas, tais como volume de dados compartilhados, número de respostas a buscas de reputação de nós e recursos, diponibilidade do nó, etc., para avaliar a reputação dos nós, além de transações efetivamente concluídas.

Referências Bibliográficas

[Abinader and Lins] , Abinader, Jorge Abílio; Lins, Rafael Dueire. *Web Services em Java*. Rio de Janeiro. Brazport, 2006.

[Adar and Huberman, 2000] , E. Adar e B. Huberman, *Free Riding on Gnutella*. Xerox Palo Alto Research Center, Technical Report, August - 2000. Disponível: http://citeseer.ist.psu.edu/adar00free.html.

[Addison, 2002] , Addison Wesley, *Peer to Peer: Collaboration and Sharing over the Internet*, 2002.

[Andersen et al., 2001] , Andersen, D., Balakrishnan, H., Kaashoek, F. Morris, R., *Resilient Overlay Networks*, 18th ACM Symposium on Operating Systems Principles, Banff/Canada, Outubro de 2001.

[Andrade et al., 2008] , Andrade, N., Brasileiro, F., Cirne, W., and Mowbray, M. (2004). *Discouraging free riding in a peer-to-peer cpu-sharing grid*. In 13th IEEE Symposium on High Performance Distributed Computing (HPDC'04).

[Barcellos and Gaspary, 2006] , *Segurança em Redes P2P: Princípios, Tecnologias e Desafios*, 2006.

[Benz and Durant, 2003] , Benz, Brian., Durant, John R., *XML Programming Bible*, 2003.

[BitTorrent, 2008] , *BitTorrent*. http://bitconjurer.org/bittorrent/.

[Buford et al., 2009] , Buford, John F., Yu, Heather., Lua, Eng Keong. *P2P Networking and Applications*, Morgan Kaufmann, 2009.

[CacheLogic Reseach, 2006] , *http://www.cachelogic.com/research.*, 2006.

[Cheng and Friedman, 2005] , Cheng, A. and Friedman, E. (2005). *Sybilproof reputation mechanisms*. In P2PECON '05: Proceeding of the 2005 ACM SIGCOMM workshop on Economics of peer-to-peer systems, pages 128-132, New York, NY, USA. ACM Press.

[Costa et al., 2005] , Costa, Cristiano., Soares, Vanessa., Benevenuto, Fabricio., Vasconcelos, Marisa., Almeida, Jussara., Almeida, Virgilio., Mowbray, Miranda. *Disseminação de Conteúdo Poluído em redes P2P*, UFMG, 2005.

[Chiozzotto and Silva, 1999] , Chiozzotto, Mauro., Silva, Luís Antonio Pinto da. *TCP/IP: tecnologia e implementação*, Ed. Érica, São Paulo, 1999.

[Da Silva, 2007] , Da Silva, Juliano Freitas., *Métodos para Contenção de Poluição em Redes P2P*, UNISINOS, 2007.

[Damiani et al., 2002] , Damiani, E., Vimercati, S. C., Paraboschi, S., Samarati, P. and Violante, F. (2002). *A Reputation-Based Approach for Choosing Reliable Resources in Peer-to-Peer Networks*. In: Computer and Communications Security (CCS'02), November 18-22, Washington DC, USA.

[Do Carmo et al., 2006] , Do Carmo, S. B., Barbar, G. S., Coelho, Barbosa, J. F., *A New Classification for Peer-to-peer Networks Architectures*. IEEE, 2006.

[Douceur, 2002] , Douceur, J. R. (2002). *The sybil attack*. In 1st International Workshop on Peer-to-Peer Systems.

[Dumitriu et al., 2005] , Dumitriu, D., Knightly, E., Kuzmanovic, A., Stoica, I., and Zwaenepoel, W. (2005). *Denial-of-service resilience in peer-to-peer file sharing systems*. In ACM SIGMETRICS, 2005.

[eBay, 2008] , *eBay* (2008). Ebay website. http://www.ebay.com/.

[eDonkey, 2008] , *eDonkey Overnet*(2008). http://edonkey2000.com/.

[ESM, 2008] , ESM (2008), *End System Multicast*. http://esm.cs.cmu.edu/.

[FastTrack, 2008] , *Fasttrack*. http://www.fasttrack.com.

[Feldman et al., 2004] , Feldman, M., Lai, K., Stoica, I., and Chuang, J. . *Robust incentive techniques for peer-to-peer networks*. Proceedings of the 5th ACM conference on Electronic commerce, pages 102 111, New York, NY, USA. ACM Press, 2004.

[Ferris and Farrell, 2003] , *What are Web Services?*, Communications of the ACM, Vol. 46, No. 62003, June 2003.

[Genome@Home, 2008] , *Genome* (2006). Genome@home. http://genomeathome.stanford.edu/.

[Gnutella, 2008] , *Gnutella* (2008). Gnutella website. http://www.gnutella.com/.

[GoogleTalk, 2008] , *Google Inc.* (2008), www.google.com/talk/intl/pt-BR/.

[Grandison and Sloman, 2000] , Grandison, T. and Sloman, M. (2000). *A survey of trust in internet applications*. IEEE Communications Surveys, 3(4):2-16.

[GT-P2P, 2008] , *Grupo de Trabalho em Computação Colaborativa* (GT-P2P) - RNP, www.gprt.ufpe.br/gtp2p, 2008.

[Gupta, 2003] , Gupta, M., Judge, P., and Ammar, M. (2003). *A Reputation System for Peer-to-peer Networks*. In NOSSDAV '03: Proceedings of the 13th international workshop on Network and operating systems support for digital audio and video, pages 144-152, New York, NY, USA. ACM Press.

[ICQ, 2008] , *ICQ* (2008). ICQ.com website. http://www.icq.com/.

[InteGrade, 2008] , *Integrade*, www.integrade.org.br, 2008.

[IRTF-P2P-GROUP, 2008] , *Peer-to-Peer Research Group*. http://www.irtf.org/charters/p2prg.html.

[Jabber, 2008] , Jabber (2008). *Jabber: Open instant messaging and a whole lot more, powered by xmpp*. http://www.jabber.org/.

[Jacobson, 1988] , Jacobson. *Congestion avoidance and control*, 1988.

[(Kamvar et al., 2003] , Kamvar, S. D., Schlosser, M. T., and Garcia-Molina, H. (2003). *The eigentrust algorithm for reputation management in p2p networks*. In Proceedings of the 12th international conference on World Wide Web (WWW '03), pages 640-651, New York, NY, USA. ACM Press.

[Kazaa, 2008] , *Kazaa* (2008). Kazza web site. http://www.kazaa.com.

[LawsTalk, 2008] , *Lawstalk*, http://laws.deinf.ufma.br, 2008.

[Liang et al., 2005] , Liang, J., Kumar, R., Xi, Y., and Ross, K. W. (2005). *Pollution in p2p file sharing systems*. In Proc. of IEEE Infocom, Miami, FL, USA.

[Loo, 2003] , A. W. Loo, *The Future of The Peer-to-Peer Computing*, Communications of the ACM, 46(9), pp.57-61, 2003.

[Marti and Garcia-Molina, 2006] , Marti, S. and Garcia-Molina, H. (2006). *Taxonomy of trust: Categorizing p2p reputation systems*. Computer Networks, 50(4):472 484.

[MercadoLivre, 2008] , *Mercado Livre web site*. http://www.mercadolivre.com.br.

[MSN, 2008] , *MSN* (2008). MSN.com website. http://www.msn.com/.

[Napster, 2008] , Napster, *www.napster.com/.*, 2008.

[Oasis, 2008] , *Modelo de Referência para Arquitetura Orientada a Serviço 1.0*. OASIS Committee Specification Business Transaction Protocol, Versão 1.0, Julho 2006.

[OpenNap, 2008] , OpenNap (2006). *OpenNap: Open Source Napster Server website*. http://opennap.sourceforge.net/.

[PeerSim, 2009] , Jelasity., Márk , Montresor., Alberto , Jesi, Gian Paolo., Voulgaris., Spyros. *The PeerSim Simulator*. http://peersim.sf.net/.

[Parameswaran et al., 2001] , Parameswaran, M., Susarla, A., Whinston, A., *P2P Networking: An Information-Sharing Alternative*. IEEE Computer, Julho de 2001.

[Righi et al., 2004] , RIGHI, Rafael R., Pellissari, Felipe R., Westphall, Carla M. *Escambo: Um Modelo de Reputação e Micropagamentos para Sistemas Peer-to-Peer*, CBComp. 2004.

[Rocha et al., 2004] , *Peer-to-Peer: Computação Colaborativa na Internet*. Minicurso, SBRC, 2004.

[Sadok, 2003] , *Computação Colaborativa (P2P)*, Technical Work, Grupo de Trabalho da Rede Nacional de Pesquisa - março 2003. Disponível: http://www.rnp.br/arquivo/gt/2003/p2p.pdf.

[Seti@Home, 2008] , *SETI* (2006). SETI@home website. http://setiathome.ssl.berkeley.edu/.

[Skype, 2008] , *Skype* (2008). Skype. http://www.skype.com/.

[Stallings, 2007] , Stallings, William., *Criptografia e Segurança de Redes: Princípios e Práticas*. Prentice-Hall, 2007.

[Ting and Deters, 2003] , Ting, Nyik San, Deters, Ralph. *A Generic Peer-to-Peer Network Simulator*, technical report, Department of Computer Science, University of Saskatchewan, Junho de 2003.

[W3C, 2008] , *Web services definition from W3C Web Services Architecture Working Group, Web Services Architecture Requirements, W3C Working Draft*, (Aug. 19, 2002); www.w3.org/TR/2002/WD-wsa-reqs-20020819

[Weerawarana et al, 2005] , Weerawarana, Sanjiva; Curbera, Francisco; Leymann, Frank; Storey, Tony; Ferguson, Donald F. *Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging, and More*. Prentice Hall, 2005.

[Wilson, 2004] , Brendon J. *Project Jxta book*, Vancouver, Canada, 2004.

[Xiong and Liu, 2004] , L. Xiong and L. Liu. *Peertrust: Supporting reputation-based trust for peer-to-peer electronic communities*. IEEE TKDE,16(7):843-857, July 2004.

[YahooMessenger, 2008] , *Yahoo Inc.*, http://br.messenger.yahoo.com, 2008.

[Yang and Molina, 2003] , B. Yang e H. Garcia-Molina, *Ppay: Micropayments for Peer-to-Peer Systems*, in Proc. 2003 CCS, pp. 27-31.

Apêndice A

Listagens de códigos de arquivos fontes e arquivos de configuração do ambiente de simulação.

A.1 Anexo 1

Este anexo mostra o arquivo utilizado para configuração do simulador PeerSim.

Listagem A.1: Arquivo de configuração da simulação

```
1  simulation.cycles 200
2  simulation.experiments 1
3
4  control.0 peersim.cdsim.Shuffle
5
6  network.size 10000
7  network.maxSize 15000
8  network.node P2PWSRepNode
9
10 include.protocol gnu
11
12 protocol.gnu p2pwsrep.GnutellaProtocol
13 protocol.gnu.ttl 4
14
15 init.0 peersim.dynamics.WireScaleFreeBA
16 init.0.protocol gnu
17 init.0.k 2
18 init.0.undir
```

```
19
20 init.1 p2pwsrep.P2PWSRepInitializerFiles
21 init.2 p2pwsrep.P2PWSRepInitializerReputation
22 init.3 p2pwsrep.P2PWSRepInitializerReputationFiles
23
24 init.4 p2pwsrep.P2PWSRepInitializer
25 init.4.protocol gnu
26 init.4.keyword rock
27 init.4.query_nodes 1
28 init.4.query_interval 4
29 init.4.max_queries 1
30
31 control.0 peersim.reports.GraphStats
32 control.0.protocol p2pwsrep
33 control.0.undir
34 control.0.nl 1
35 control.0.nc -1
36
37 control.1 p2pwsrep.P2PWSRepGraphStats
38 control.1.protocol p2pwsrep
39 control.1.outf graph-stats
40 control.1.onlyonecycle
41 control.1.from 0
```

A.2 Anexo 2

Este anexo mostra um log de saída de uma simulação.

Listagem A.2: Log de saída da simulação da rede P2PWS

```
1 Simulator: loading configuration
2 ConfigProperties: File template/p2wsprep-simulator.txt loaded.
3 Simulator: Number of experiments 1
4 Simulator: starting experiment 0 invoking peersim.cdsim.CDSimulator
5 Random seed: 123456789
6 CDSimulator: resetting
7 tamanho da rede: 10
```

```
8  CDSimulator: running initializers
9  Configuration files to nodes
10 The simulation will search for: rock
11 - Running initializer init.0: class peersim.dynamics.WireScaleFreeBA
12 - Running initializer init.1: class p2pwsrep.P2PWSRepInitializerFiles
13 <<< File create in node 8 >>>
14 <<< File create in node 3 >>>
15 - Running initializer init.2: class p2pwsrep.
       P2PWSRepInitializerReputation
16 - Running initializer init.4: class p2pwsrep.P2PWSRepInitializer
17 CDSimulator: starting simulation
18 It's searching in nodeId: 0
19 Node: 0 sending to: 2
20 It's searching in nodeId: 2
21 2 forwarding to node: 8 ttl: 1 type: 0
22 2 forwarding to node: 3 ttl: 1 type: 0
23 2 forwarding to node: 4 ttl: 1 type: 0
24 2 forwarding to node: 1 ttl: 1 type: 0
25 It's searching in nodeId: 3
26 File found in nodeId 3 --->  File name:  rock music
27 It's searching in nodeId: 4
28 4 forwarding to node: 1 ttl: 2 type: 0
29 4 forwarding to node: 5 ttl: 2 type: 0
30 5 forwarding to node: 9 ttl: 1 type: 3
31 5 forwarding to node: 6 ttl: 1 type: 3
32 5 forwarding to node: 7 ttl: 1 type: 3
33 5 forwarding to node: 8 ttl: 1 type: 3
34 [...]
35
36 7 forwarding to node: 9 ttl: 2 type: 3
37 7 forwarding to node: 1 ttl: 2 type: 3
38 It's searching in nodeId: 7
39 7 forwarding to node: 9 ttl: 4 type: 0
40 7 forwarding to node: 1 ttl: 4 type: 0
41 It's searching in nodeId: 8
42 File found in nodeId 8 ---->  File name:  rock music
43 8 forwarding to node: 2 ttl: 1 type: 4
```

```
44 8 forwarding to node: 2 ttl: 2 type: 3
45 It's searching in nodeId: 9
46 9 forwarding to node: 5 ttl: 1 type: 4
47 It's searching in nodeId: 1
48 1 forwarding to node: 3 ttl: 2 type: 0
49 1 forwarding to node: 4 ttl: 1 type: 3
50 1 forwarding to node: 2 ttl: 1 type: 3
51 [...]
```