# Um Sistema de Reputação Para Redes Peer-to-Peer Estruturado Baseado na Reputação de Arquivos, com Verificação Pela Reputação dos Nós

Este exemplar corresponde à redação final da Dissertação devidamente corrigida e defendida por Douglas Gielo Quinellato e aprovada pela Banca Examinadora.

Campinas, 31 de março de 2009.

Paulo Lício de Geus (Orientador)

Dissertação apresentada ao Instituto de Computação, UNICAMP, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

**FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA DO IMECC DA UNICAMP**
Bibliotecária: Crisllene Queiroz Custódio – CRB8 / 7966

Quinellato, Douglas Gielo
Q44s Um sistema de reputação para redes Peer-to-Peer estruturado baseado na reputação de arquivos, com verificação pela reputação dos nós / Douglas Gielo Quinellato -- Campinas, [S.P. : s.n.], 2009.

Orientador : Paulo Lício de Geus
Dissertação (Mestrado) - Universidade Estadual de Campinas, Instituto de Computação.

1. Arquitetura Peer-to-peer (Redes de Computação). 2. Redes de computação - Sistemas de segurança. 3. Sistemas de Reputação. 4. Redes estruturadas. I. Geus, Paulo Lício de. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

(cqc/imecc)

Título em inglês: A structured Peer-to-Peer reputation system based on file reputation, with verification by the nodes reputation.

Palavras-chave em inglês (Keywords): 1. Peer-to-peer architecture (Computer networks). 2. Computer network security. 3. Reputation Systems. 4. Structured Networks.

Área de concentração: Segurança de redes

Titulação: Mestre em Ciência da Computação

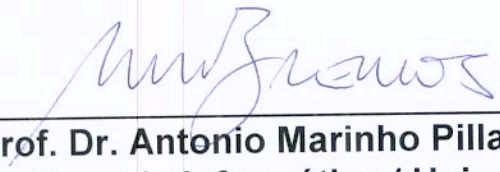
Banca examinadora: Prof. Dr. Paulo Lício de Geus (IC-Unicamp)
Prof. Dr. Marinho Pilla Barcellos (UFGRS)
Profa. Dra. Islene Calciolari Garcia (IC-Unicamp)
Prof. Dr. Ricardo Dahab (Suplente) (IC-Unicamp)
Prof. Dr. Paulo Henrique de Aguiar Rodrigues (Suplente) (DCC/IM-UFRJ)

Data da defesa: 27/02/2009

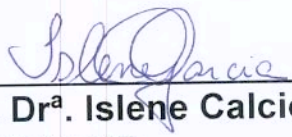
Programa de Pós-Graduação: Mestrado em Ciência da Computação

# TERMO DE APROVAÇÃO

Dissertação Defendida e Aprovada em 27 de fevereiro de 2009, pela Banca examinadora composta pelos Professores Doutores:



**Prof. Dr. Antonio Marinho Pilla Barcellos**
**Instituto de Informática / Universidade Federal do Rio Grande do Sul.**



**Profª. Drª. Islene Calciolari Garcia**
**IC / UNICAMP.**

Instituto de Computação
Universidade Estadual de Campinas

# Um Sistema de Reputação Para Redes Peer-to-Peer Estruturado Baseado na Reputação de Arquivos, com Verificação Pela Reputação dos Nós

**Douglas Gielo Quinellato**[1]

Março de 2009

**Banca Examinadora:**

- Paulo Lício de Geus (Orientador)
- Marinho Pilla Barcellos - II - UFRGS
- Islene Calciolari Garcia - IC - Unicamp
- Ricardo Dahab - IC - Unicamp (Suplente)
- Paulo Henrique de Aguiar Rodrigues - DCC - IM - UFRJ (Suplente)

[1]Suporte financeiro de: Bolsa do CNPq 03/2007–02/2008

# Resumo

As redes P2P ganharam bastante popularidade na última década, consolidando-se como um dos serviços mais populares da internet, provendo uma arquitetura distribuída para o fornecimento de serviços sem a necessidade de um *host* assumir o papel de servidor. A popularidade trouxe, entretanto, a necessidade de se desenvolver mecanismos para garantir o funcionamento perante os crescentes ataques à rede. Com a estabilidade dos algoritmos relacionados ao funcionamento das redes P2P foi possível um aumento no desenvolvimento destes mecanismos de segurança.

Nesta dissertação é proposto um sistema de reputação para redes P2P de compartilhamento de arquivos, um mecanismo de segurança que visa impedir a proliferação de arquivos corrompidos. Tais sistemas funcionam gerenciando as opiniões emitidas pelos nós participantes da rede sobre os serviços prestados pelos outros nós. Estas opiniões podem ser sobre o nó que prestou o serviço ou sobre a qualidade do serviço prestado. As opiniões sobre um mesmo nó ou serviço avaliado são armazenadas e posteriormente agregadas através de uma função, formando a reputação destes.

O mecanismo proposto baseia-se nas opiniões emitidas sobre a autenticidade os arquivos, utilizando a reputação dos nós para indicar a qualidade da opinião sendo emitida por eles. Essa verificação da qualidade da opinião visa aumentar a confiança na opinião utilizada com a adição de um nível de verificação por motivos de eficiência, visto que implementar uma rede de confiança inteira é custosa.

Foram realizadas simulações para a verificação da eficácia da rede, realizando comparações tanto com uma rede sem nenhum sistema de reputação quanto com outros sistemas de reputação.

# Abstract

P2P networks have earned a great deal of popularity over the last decade, consolidating itself as one of the most popular internet service, providing a distributed architecture for the furnishing of services without the need of a centralized server host. However, such popularity brought the necessity for security mechanisms in order to assure the network availability in spite of the attacks on the network. Stability in the algorithms related to the basic operation of the P2P networks made possible the rise on the development of security systems.

In this dissertation it's proposed a reputation system for file sharing P2P networks, a security mechanism aimed at lowering the spread of corrupted files in the network. Such systems work by managing the opinions issued by the participants of the network about the received services from the other nodes. These opinions can be about the nodes, or about the quality of the services themselves. Opinions about the same service or node are them joined through the use of a mathematical model (function), calculating their reputation.

The proposed reputation system is based on the reputation of the files, using the node reputation as a means to assess the quality of the opinion being issued. This check is made with the purpose of improving trust in the used opinion by adding one level of opinion checking. Only one level is used for efficiency, since implementing a full trust network is expensive.

Simulations were used in order to assess the effectiveness of the proposed reputation system. The results are used in comparisons with the same simulation without the use of any reputation system, and with the results of other reputation systems found in the literature.

# Agradecimentos

Eu gostaria de agradecer primeiramente a minha família, por todo o suporte dado durante toda a minha vida, por ter aguentado a distância que estivemos durante este mestrado.

Aos amigos do LAS, Ricardo Saffi, Miguel, Felipe, Luciano, Aysyane e todos os outros, por todo o conhecimento que foi e ainda será trocado.

Aos amigos de Campinas, Leonardo Juliano, Danilo, Murilo, Rudson, Ana Zeferino, Daiane, Fabiane, Maurilio, Leonardo e todos os outros pela companhia durante todo esse tempo no qual estive em Campinas, pelas festas e pelos shows que fomos. Aos que moraram comigo, pela convivência quase sempre harmoniosa, e pelo aprendizado da convivência conjunta em uma república.

A todos os amigos com os quais realizei as matérias do mestrado, pelas horas de estudo conjunto, os dias realizando trabalhos e as comemorações realizadas.

Aos amigos que ficaram no rio durante a minha ida para Campinas, Liana, Vinícius, Bruno Nicolau, Diogo, Daniel, Lynton, Nathalia, Rodrigo Silveira, Rodrigo Lemos, Luiz Valmont, e todos os outros amigos formados durante os anos de graduação e estágio na UFRJ. Foram anos de estudos, festas, shows, cinema, conversas, coisas que a distância não pôde separar.

Ao meu orientador Paulo Lício de Geus, pela ajuda fundamental nos momentos onde eu me encontrava mais perdido, fornecendo a ajuda necessária para que o meu trabalho seguisse o rumo certo.

A todo o pessoal o LAS, do IC e da Unicamp, pela infra-estrutura que permitiu que este trabalho fosse feito.

# Sumário

# Lista de Tabelas

# Lista de Figuras

# Capítulo 1

# Introdução

As redes Peer to Peer (P2P) são um tipo de arquitetura de rede distribuída para a provisão de serviços na internet, onde os participantes formam uma rede na camada de aplicação estabelecendo conexões entre si. Estas redes ganharam fama em 1999 com a criação do Napster, e desde então têm conquistado cada vez mais popularidade, já tendo se tornado um serviço estabelecido na rede. Esta revolução foi causada pois as redes P2P permitem que os usuários forneçam serviços uns aos outros, tornando qualquer pessoa conectada na internet um provedor de serviços.

Existem diversos serviços que a rede pode prover. Os mais usuais são o compartilhamento de arquivos, o de distribuição de processamento ou de armazenamento.

Esta distribuição da provisão do serviço foi possível devido a uma grande mudança em relação aos serviços outrora existentes: a descentralização dos elementos responsáveis pelo funcionamento da rede. Todos os nós (participantes) que formam a rede são responsáveis pelo seu funcionamento. Todas as funcionalidades providas pela rede (entrada de participantes, busca de serviços, etc.) são executadas pelos participantes, ao contrário do modelo cliente/servidor habitual do provimento de serviços na internet, onde um grupo de participantes é responsável integralmente pelo fornecimento dos serviços, restando aos outros apenas utilizá-los.

A idéia de descentralização no provimento de serviços não é nova na internet. A internet em si foi criada com o objetivo de funcionar de forma distribuída, mantendo o funcionamento mesmo que uma parte da rede seja retirada do ar. Diversos serviços de suporte a rede (DNS, roteamento, email) funcionam de forma descentralizada. Apenas os serviços que os usuários utilizam de forma direta são centralizados, como o WWW. Ainda assim, estes serviços são providos por diversos servidores. A centralização nestes casos ocorre pois um serviço específico (por exemplo, o email do Yahoo!, ou a página de notícias da UOL) é provido por um único fornecedor, ficando indisponível caso este saia do ar. Nas redes P2P, um mesmo serviço pode ser fornecido por diversos participantes,

logo a saída de um nó não significa necessariamente o término do fornecimento do serviço que ele fornecia.

Para o funcionamento da rede, os nós formam conexões entre si, criando uma rede de conexões. Através dela, os nós participantes podem trocar mensagens entre si e requisitar serviços. para que haja comunicação entre nós que nõo possuasm conexão direta, envia-se uma mensagem para um nó vizinho, que a roteia para um outro vizinho seu, até que se alcance o nó desejado.

Entretanto, a distribuição do funcionamento da rede de forma indiscriminada entre os participantes gera alguns problemas. Como os serviços essenciais ao funcionamento da rede são realizados por participantes dos quais não se tem conhecimento, a rede se torna mais vulnerável a ataques, visto que um atacante pode entrar na rede, assumir a responsabilidade por um serviço importante e manipular os resultados, causando um mal funcionamento. Este problema se agravou com a popularização da rede, pois o aumento da utilização causa um aumento no benefício de um ataque.

Os constantes ataques às redes P2P, junto com a necessidade de criar métodos para garantir o funcionamento das redes para o uso em redes corporativas, levou os pesquisadores a desenvolverem métodos para manter a segurança da rede. Diversos aspectos da segurança foram abordados. Os problemas enfrentados pelas redes P2P incluem segurança no roteamento das mensagens, a integridade dos dados transmitidos, a identidade dos nós, entre outros.

Um destes aspectos, de grande importância, é o da qualidade dos serviços prestados. Uma grande dificuldade de se garantir a qualidade do serviço prestado se dá pela dificuldade de verificar o provimento correto do serviço. Considere, por exemplo, o compartilhamento de arquivos. Verificar se o arquivo baixado realmente possui o conteúdo anunciado é uma tarefa que não é possível ser feita automaticamente. Uma das formas de se fazer esta verificação é coletando a opinião das pessoas que o avaliaram. Esta opinião pode ser sobre a qualidade do arquivo, ou sobre a confiabilidade do arquivo que o possui. Geralmente estas opiniões são consolidadas num valor único, calculado através da agregação das opiniões dos outros participantes. Este valor calculado é chamado de *reputação*, e os sistemas desenvolvidos para o gerenciamento de reputação (armazenamento, cálculo, etc.) é chamado de *Sistemas de Reputação*.

Nesta dissertação é proposto um sistema de reputação para sistemas P2P de compartilhamento de arquivos. No Capítulo 2 é feita uma revisão das características principais das redes P2P. Os principais aspectos de segurança em redes P2P são discutidos no Capítulo 3. Um foco maior em sistemas de reputação é dado no Capítulo 4, seguido de uma pesquisa dos principais sistemas no Capítulo 5. O sistema de reputação proposto é exposto no Capitulo 6, e testado através de simulação no Capítulo 7. Finalmente, as conclusões são apresentadas no Capítulo 8.

# Capítulo 2

# Redes Peer to Peer

As redes P2P (Peer to Peer) são um tipo de sistema distribuído formado por nós conectados entre si, tendo como finalidade o compartilhamento de recursos entre os participantes da rede. Os recursos compartilhados podem ser ciclos de CPU, arquivos, armazenamento, etc. Estas redes podem ser utilizadas como base para o desenvolvimento de aplicações distribuídas.

Este modelo de rede se tornou bastante popular a partir de 1999, com a criação do Napster, uma rede para compartilhamento de arquivos que, apesar de ser P2P, possuía um servidor centralizado para a realização de buscas de arquivos, sendo apenas a troca dos arquivos feita diretamente entre os nós. Depois de sua desativação através do desligamento dos servidores de busca por motivos legais, diversas redes apareceram para substituí-la, com mudanças no funcionamento de forma a ficarem menos suscetíveis a problemas como os que o Napster teve. As arquiteturas das redes P2P serão discutidas na seção 2.2.

Androutsellis-Theotokis em [3] define redes P2P como:

> "Peer-to-peer systems are distributed systems consisting of interconnected nodes able to self organize into network topologies with the purpose of sharing resources such as content, CPU cycles, storage and bandwidth, capable of adapting to failures and accommodating transient populations of nodes while maintaining acceptable connectivity and performance, without requiring the intermediation or support of a global centralized server or authority."

Nas redes P2P, os participantes, denominados *nós* ou *peers*, conectam-se entre si, criando uma rede sobre a camada TCP/IP onde todos os nós podem prover ou receber serviços. Esta rede é denominada *overlay*. A rede overlay é utilizada para a troca de mensagens relacionadas ao funcionamento da rede, como o registro de um nó novo, busca por serviços e, dependendo da rede, para a reorganização dos nós na topologia. A prestação do

serviço pode ser feita através do roteamento pela própria rede, ou por uma conexão direta entre os nós envolvidos, sendo a última opção a mais comum por questões de eficiência.

Uma rede P2P pode ser utilizada para o provimento de diversos recursos. O serviço mais comumente provido por redes P2P é o compartilhamento de arquivos, mas existem outros serviços populares, como o compartilhamento de processamento ou de armazenamento (que é diferente de compartilhamento de arquivo: no compartilhamento de armazenamento os nós fornecem espaço em disco para um arquivo ser armazenado; no compartilhamento de arquivos é fornecido um arquivo para os outros copiarem). Cada recurso sendo compartilhado necessita de um suporte específico na rede, o que leva cada rede a se especializar no compartilhamento de um recurso. Como exemplo, o projeto seti@home[1] é uma rede P2P de compartilhamento de processamento, que utiliza a capacidade de processamento não utilizada dos computadores para tentar encontrar sinais de vida inteligente fora da Terra. Nesta dissertação, será focado o serviço de compartilhamento de arquivos, visto que este será o tipo de rede P2P sobre o qual o sistema de reputação proposto atuará.

O modelo de sistema P2P contrapõe-se ao modelo cliente/servidor. O modelo cliente/servidor divide os *hosts* em dois tipos: o servidor, responsável pelo fornecimento do serviço, e o cliente, a quem cabe apenas a função de acessar o serviço prestado. Com essa divisão há um controle maior sobre o que é posto na rede, e um aumento na eficiência da busca. Entretanto, há um custo maior de operação para os mantenedores do servidor, que deve lidar sozinho com a carga de pedidos. O modelo P2P acaba com esta diferença, tornando todos os participantes iguais na capacidade de prover ou de receber serviços, dividindo os custos e a carga entre todos os participantes.

As redes P2P provêem uma série de vantagens: como todos os nós possuem a capacidade de prestar o serviço, a rede não possui um ponto único de falha. Caso um nó saia do ar, apenas a parte do serviço do qual ele era responsável fica indisponível. No modelo cliente/servidor, caso o servidor saia do ar, nenhum serviço é acessível. Os serviços mais populares são naturalmente replicados em diversos nós participantes, o que permite a distribuição de carga entre estes nós, e a replicação destes serviços, aumentando a disponibilidade dos mesmos.

## 2.1 Características

Existem algumas características básicas para a classificação de uma rede como sendo P2P. Aberer [1] descreve três princípios que uma rede P2P deve seguir:

**O princípio dos recursos compartilhados:** deve haver algum recurso (serviço) a ser

[1] setiathome.berkeley.edu/

compartilhado, algo que não possa ser feito somente por um nó;

**o princípio da descentralização:** decorrente do primeiro princípio, se todos possuem algum serviço a ser compartilhado, e todos podem solicitar algum serviço, então todos possuem o mesmo grau de importância e de responsabilidade;

**o princípio da auto-organização:** como não existe um nó para coordenar a rede, a rede como um todo deve ser capaz de se auto-adaptar às mudanças.

Outra característica importante das redes P2P é a autonomia dos nós participantes, que podem entrar e sair da rede quando quiserem. A entrada e saída constante de nós nas redes P2P é chamada de *churn*. Em [2], esta autonomia é ressaltada através das seguintes perguntas para determinar se um sistema é P2P:

- O sistema dá aos nós na borda da rede autonomia significativa?
- O sistema permite conectividade variável e endereços de rede variáveis?

Estas duas perguntas visam auferir a distribuição de responsabilidades entre os nós e a capacidade da rede se auto-organizar mesmo com a entrada e saída dos nós, duas características importantes.

Para o funcionamento das redes P2P, são utilizadas algumas operações básicas. Estas operações são implementadas através de mensagens que são enviadas aos nós responsáveis pela execução, ou a nós que as redirecionarão aos responsáveis. As operações básicas são: entrar na rede, se manter na rede[2] (*keep alive*), procurar um arquivo e requisitar um arquivo. Dependendo da rede, outras mensagens podem ser necessárias.

## 2.2 Classificação

Existem vários tipos diferentes de redes P2P, mas a maioria delas seguem alguns princípios, o que permite classificá-las. Esta classificação pode ser feita de acordo com o grau de centralização, em *centralizadas*, *descentralizadas* e *híbridas*, ou em relação ao grau de organização, em *estruturadas* ou *desestruturadas*.

### 2.2.1 Grau de Centralização

O grau de centralização indica o quanto as operações da rede estão distribuídas entre os nós da rede. Por motivos de eficiência, algumas operações podem ser concentradas em alguns nós com maior capacidade. As operações mais comuns para a centralização são as de entrar na rede e de busca.

[2] assim como a maioria dos protocolos de rede, as redes P2P implementam *soft state*, sendo necessário o envio de mensagens para manter viva a conexão.

### Centralizadas

As redes centralizadas possuem um nó que funciona como um servidor central, indexando o conteúdo dos outros participantes e autenticando os nós caso necessário. A busca pelos objetos é feita no servidor, que retorna uma lista de nós que o possuem. A transferência dos arquivos é feita diretamente entre o requisitante e o requisitado. Este modelo não se distancia muito do modelo cliente/servidor, retirando do servidor apenas a carga relativa ao armazenamento e transferência dos arquivos, sendo apenas parcialmente P2P. Como vantagens deste modelos destacamos a busca rápida, pois o servidor possui o índice de todos os arquivos. Porém, assim como no modelo cliente/servidor, o nó central é um ponto único de falha, e sua saída causa a queda da rede. O único exemplo popular deste tipo de rede é o Napster, que saiu do ar após uma decisão judicial ordenar o desligamento do servidor central, evidenciando a gravidade de um ponto único de falha. Uma rede que se encaixa parcialmente como centralizada é o BitTorrent[3].

Este foi um modelo inicial, sendo abandonado quase que inteiramente após o desenvolvimento dos outros tipos de rede, mais resilientes à saída dos nós.

### Descentralizadas

Neste tipo de rede todos os nós possuem o mesmo papel, sendo igualmente responsáveis pelo armazenamento do conteúdo, e por responder ou reencaminhar as buscas realizadas pelos outros nós. Como nenhum nó possui atribuições especiais, não há um ponto único de falha, sendo necessário a retirada de praticamente todos os nós para que a rede saia do ar. São exemplos deste tipo de rede as redes Gnutella, Chord [46], Kademlia [32] e CAN (*Content Addressable Network*) [36].

As redes descentralizadas são as mais aderentes às características das redes P2P, tendo todas as operações distribuídas igualmente entre todos os nós. Os métodos de busca utilizados dependem do grau de estruturação da rede.

### Híbridas

As redes híbridas visam obter um meio termo entre os dois modelos anteriores, procurando um equilíbrio entre consumo de banda e eficiência de busca. Algumas das operações, como a de indexação e busca, são atribuídas a alguns nós especiais, chamados de *superpeers*. Os nós comuns conectam-se a um ou mais destes nós, enquanto eles realizam conexões entre si formando uma subrede.

Cada superpeer armazena um índice contendo os arquivos pertencentes aos nós comuns conectados a ele. Um nó comum realiza uma busca enviando a mensagem para o seu

[3]http://www.bittorrent.org/beps/bep_0003.html

superpeer correspondente, que a reenvia somente para os outros nós da subrede. Esta solução obtém uma quantidade de resultados tão boa quanto um *flooding* (ver seção 2.2.2), utilizando menos mensagens por propagar as buscas para menos nós. Os superpeers não constituem um ponto de falha, pois, além de haver vários, novos podem ser eleitos automaticamente para substituir os que saírem. Como desvantagem temos o aumento do *overhead* nos superpeers, tanto por armazenar os índices quanto por ter de lidar com as mensagens de manutenção e busca.

Entre as redes híbridas mais famosas estão a KaZaA, a eDonkey. As versões mais novas do Gnutella (a partir da versão 0.6) também são híbridas.

### 2.2.2 Grau de Estruturação

Outra forma de se classificar as redes P2P é pela topologia da rede. Os nós podem se organizar aleatoriamente ou numa estrutura planejada, visando uma maior eficiência. O posicionamento dos arquivos também é determinado por esta decisão [28].

Independente do grau de estruturação da rede, elas podem ser divididas em camadas [28]. Cada camada provê um serviço à superior, de forma análoga ao funcionamento dos diferentes protocolos de rede do padrão OSI. Apesar de todos os tipos de redes poderem ser divididas desta forma, esta divisão se faz mais útil nas redes estruturadas, onde há uma complexidade maior.

#### Não Estruturadas

As redes não estruturadas são caracterizadas por não terem nenhuma estrutura pré-definida para a topologia overlay. Os nós, ao entrarem, estabelecem conexões aleatoriamente entre si. Estas redes possuem como vantagens o baixo custo de manutenção (entrada, saída, organização de nós) e maior simplicidade dos algoritmos. Porém, buscas neste modelo tendem a ser mais custosas, devido ao fato de que todos os nós devem ser consultados para achar todas as respostas possíveis. A maior parte das redes populares são não estruturadas: Gnutella, KaZaA, eDonkey, etc.

Métodos de busca para este tipo de rede incluem o *flooding* de mensagens, que consiste em um nó enviar a mensagem de busca para todos os seus vizinhos, e estes enviarem para todos os seus vizinhos, até um determinado número de saltos (*hops*), realizando uma busca em largura. Um outro método básico é o *random walker*, no qual apenas uma mensagem é enviada, e esta percorre individualmente cada nó até uma determinada quantidade de saltos, numa busca em profundidade. No primeiro caso, os arquivos são encontrados mais facilmente, e o tempo de resposta é menor. Porém, a quantidade de mensagens enviada é muito grande, o que faz a rede ser pouco escalável. No segundo caso, os custos de consumo de banda são diminuídos, mas o número de respostas encontradas e o tempo de resposta

são bastante prejudicados. Outros métodos para a busca existem, geralmente sendo uma variação dos dois métodos básicos descritos.

Como as buscas são encaminhadas aos nós independente de posicionamento, as elas podem ser feitas com palavras chave e critérios de pesquisa complexos. Arquivos raros, entretanto, são difíceis de serem encontrados, mesmo no *flooding* [38].

### Estruturadas

Neste caso os nós organizam-se numa estrutura rígida, visando facilitar as buscas. O conteúdo armazenado na rede é organizado através do seu posicionamento em nós específicos. A forma mais comum de se implementar uma rede estruturada é associando identificadores (*nodeId*) aos nós e aos objetos. A estrutura da rede é determinada pela organização desses identificadores, determinando com quais nós serão abertas conexões para formar a camada de *overlay*. A regra de organização e o número de conexões são o que diferenciam as diferentes redes estruturadas. Esta camada da rede permite que se encontre um nó através do seu *nodeId*. Pela complexidade envolvida na manutenção da rede e pelo relativo pouco tempo de existência, a maior parte das redes estruturadas são redes acadêmicas, como a Chord e a CAN.

A distribuição dos arquivos entre os nós é feita através da construção de uma *Distributed Hash Table (DHT)* [28]. As DHT's funcionam como uma tabela hash, posicionando os arquivos em locais determinados por um identificador do arquivo. Este identificador é calculado utilizando uma função de hash baseado nos seus metadados. Porém, esta tabela se encontra distribuída em diversos nós, cada um responsável por um intervalo dos identificadores. Os objetos são posicionados na rede de acordo com alguma relação entre o seu identificador e o do nó responsável por ele (por exemplo, o objeto com *nodeId* menor que o do nó atual e maior que o do nó anterior). Os identificadores dos objetos são mapeados para os nós de forma que a quantidade de objetos realocados no caso de entrada ou saída de um nó seja mínimo. Esta técnica é denominada de *hashing consistente (consistent hashing)* [25]. Esta estruturação da rede permite que qualquer objeto possa ser encontrado de forma eficiente e com garantia de resposta, com a desvantagem de um aumento do custo de manutenção relacionado ao posicionamento dos nós na estrutura, reorganização dos objetos na saída de um nó, entre outros. Outra desvantagem das buscas em redes estruturadas é a dificuldade de se realizar buscas por palavras-chaves: para se encontrar um arquivo é necessário o identificador, geralmente calculado pelo nome completo do arquivo, logo buscas por parte do nome não são possíveis. Existem algumas propostas para permitir buscas por palavras chave [11].

As redes estruturadas geralmente são organizadas em camadas, cada uma com parte das responsabilidades da rede. A camada inferior, que corresponde a rede P2P propriamente dita, fornece o serviço de encontrar, dado um identificador, o nó responsável por

ele. A camada imediatamente superior é a DHT, que utiliza a camada inferior para determinar qual nó é o responsável por qual intervalo de objetos. A DHT fornece como interface as funções de adicionar um par (chave, arquivo), e recuperar um arquivo dado uma chave. Por fim, temos a camada de aplicação da rede P2P, que implementa o serviço fornecido pela rede. Nesta camada são determinados as estruturas de dados da aplicação, estruturas estas que serão armazenados utilizando a camada que implementa a DHT.

O funcionamento de uma busca dentro de um nó é mostrado na Figura 2.1. O usuário do sistema inicia uma busca por um determinado arquivo(1), fazendo uma requisição à camada superior da rede(2). Esta camada calcula o *hash* do arquivo(3), e repassa para a DHT(4) uma requisição para que seja localizado o nodeId do nó responsável pelo hash daquele arquivo(5). Uma vez determinado o nodeId, a camada inferior é consultada para determinar o IP deste nó (6, 7).

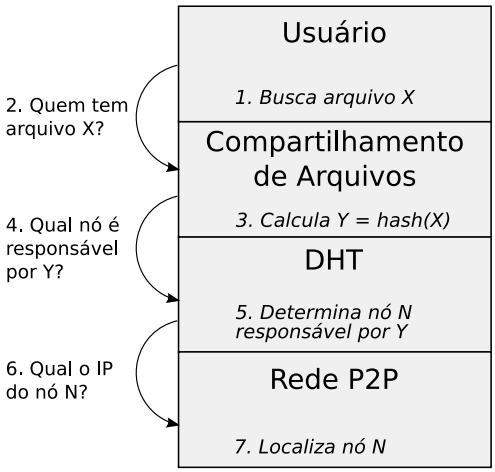



Figura 2.1: Busca por arquivo em redes estruturadas.

## 2.3 Chord [46]

A rede Chord é uma rede estruturada descentralizada que organiza os nós num círculo ordenados pelos *nodeId's*. Ele foi projetado visando prover um protocolo cuja correção e eficiência pudesse ser provado, mesmo com a ocorrência de *churn* e falhas de nós. Estas características foram obtidas pela simplicidade dos algoritmos utilizados. O sistema de reputação proposto é construido sobre esta rede.

Cada nó na rede, assim como os arquivos, possuem um identificador, denominado *nodeId*, representado por um número de $m$ bits. A rede é organizada em ordem crescente de identificador, com os nós estabelecendo ligações com os nós sucessores e mantendo informação sobre quem é o predecessor. Esta rede é circular, ou seja, o último nó está

conectado com o primeiro. Cada nó é responsável pelo intervalo de identificadores situado entre os *nodeIds* do predecessor e o seu.

Um nó, ao entrar na rede, realiza uma busca pelo seu identificador para descobrir o seu sucessor. Feito isso, ele comunica ao seu sucessor a sua entrada na rede, atualizando as informações sobre predecessores e sucessores de ambos. O intervalo de identificadores do sucessor é dividido com o nó entrante, repassando a este nó a parte do intervalo de sua responsabilidade. Esta divisão ocorre somente no intervalo no qual está ocorrendo a entrada do nó, minimizando a movimentação de arquivos, o que funciona de acordo com o hashing consistente.

Além de uma conexão com o nó sucessor, o Chord mantém uma tabela para nós distantes, denominada *finger table*. Esta tabela inicialmente teria $m$ apontadores, mas segundo o artigo apenas $log(n)$ entradas são diferentes, pois as primeiras entradas da tabela apontam para o sucessor com alta probabilidade, de acordo com a teoria do *balls into bins* [35]. A $i^a$ entrada da *finger table* de um nó de identificador $id_n$ aponta para o primeiro nó maior que $id_n + 2^i$ (sucessor de $id_n + 2^i$). Esta tabela é utilizada para a realização de buscas na rede de forma mais eficiente, não sendo necessária para a prova de funcionamento do Chord. A Figura 2.2(a) mostra a *finger table* do nó 8. Nesta rede, $m = 6$. Observe que, das 6 entradas, as 3 primeiras apontam para o mesmo nó, o sucessor do 8.

As buscas feitas pela *finger table* seguem o seguinte algoritmo: se o identificador $id_o$ procurado for de responsabilidade do sucessor $id_s$ (isto é, $id_n < id_o \leq id_s$), então a busca retorna $id_s$ como responsável pelo $id_o$; senão, a mensagem de busca é encaminhada para o nó com maior identificador menor que $id_o$, que executa este algoritmo, até chegar ao predecessor do identificador procurado, que retornará o seu sucessor. Este algoritmo encontra o nó responsável pelo objeto em $O(log(n))$ mensagens em média. O funcionamento deste algoritmo é mostrado na Figura 2.2(b). O nó 8 inicia uma busca por um arquivo de identificador 54. O nó na *finger table* de 8 mais próximo do resultado é 42, logo a mensagem de busca é enviada a este nó. O nó 42, por sua vez, envia para o nó mais próximo de 54, 51 neste caso. Como o sucessor de 51 é 56, e o identificador 54 está neste intervalo, o nó 51 retorna 56 como o *nodeId* do nó responsável.

## 2.4 Conclusões

As redes P2P surgiram como uma alternativa ao modelo cliente/servidor de prover serviços na internet, com o objetivo de distribuir as responsabilidades pelo fornecimento do conteúdo entre os participantes da rede. Elas vêm fazendo bastante sucesso deste então, principalmente para o compartilhamento de arquivos.

Apesar de inicialmente serem utilizadas mais para o compartilhamento de arquivos

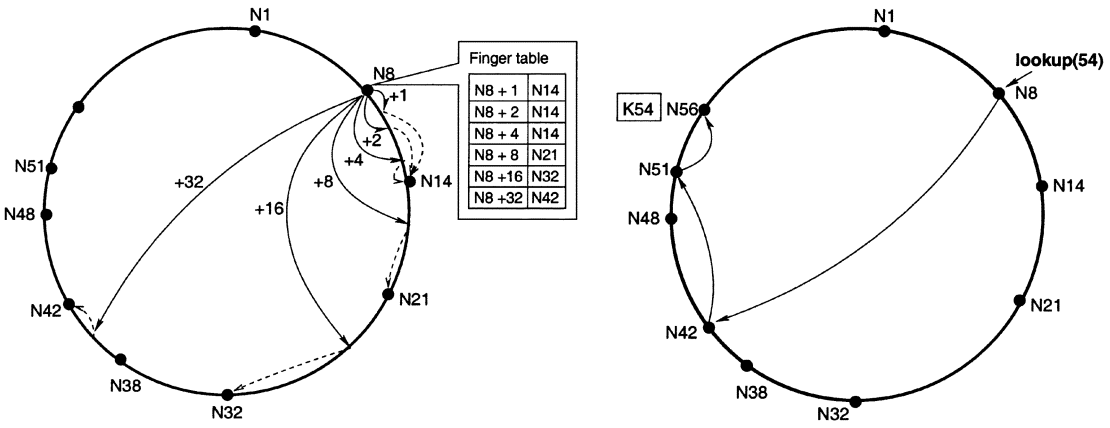



Figura 2.2: Funcionamento da rede Chord: (a) *finger table* do nó 8; (b) busca pela chave 54 iniciada pelo nó 8. Figuras retiradas de [46]

entre usuários domésticos, a resiliência apresentadas por estas redes levou-as a serem utilizadas em projetos de redes acadêmicas e empresariais como forma de distribuir o fornecimento de serviços. O interesse destes setores pelas redes P2P levou-os a desenvolverem protocolos novos com o objetivo de garantir o funcionamento das redes de forma eficiente.

Pelo fato de diversos nós colaborarem sem um ponto de coordenação, algumas tarefas tornaram-se mais difíceis de serem realizadas de forma eficiente. Diversas arquiteturas foram desenvolvidas para otimizar estas tarefas. Estas otimizações conseguiram aumentar grandemente a eficiência em relação aos protocolos iniciais, aumentando um pouco o *overhead* de manutenção.

O aparecimento de redes como as estruturadas mostram que é possível obter uma qualidade de serviço garantida, com um tempo médio de resposta aceitável.

# Capítulo 3

# Segurança em Peer to Peer

A crescente popularização dos sistemas P2P levou estes a se tornarem alvo freqüente de ataques. Como os protocolos inicialmente preocupavam-se mais com o funcionamento eficiente da rede, elas ficaram vulneráveis a diversos tipos de ataques. A distribuição de responsabilidades cruciais ao funcionamento da rede, como roteamento e buscas, à usuários desconhecidos provê aos atacantes um poder grande para afetar o funcionamento de uma boa parte da rede. Como a maioria das redes populares de P2P são utilizadas primariamente para a troca de arquivos de música muitas vezes protegidas por leis de propriedade intelectual, as gravadoras donas destes direitos têm realizado ataques freqüentes às redes, como a retirada do Napster do ar por meios legais (possível pela sua arquitetura centralizada), e a inserção de arquivos ilegítimos no eMule e o KaZaA. Desenvolvedores de vírus podem também se utilizar das mesmas táticas de arquivos falsos para disseminar vírus.

Com um maior estabelecimento nas áreas infra-estruturais, um foco maior para a área de segurança pôde ser dado, para garantir o funcionamento das redes mesmo com a presença de participantes maliciosos. Os protocolos desenvolvidos até então consideravam que todos os nós agiam de boa fé, seguindo estritamente a especificação. Porém, por funcionar em ambientes sem controle, os nós podem funcionar de forma arbitrária, deixando de rotear mensagens, ou modificando resultados de buscas, por exemplo.

Neste capítulo serão discutidas as questões de segurança em redes P2P de um modo geral. Inicialmente serão apresentados os diversos aspectos de segurança na Seção 3.1. Na Seção 3.2 abordaremos os ataques mais comuns, e as soluções propostas na literatura.

## 3.1 Aspectos de Segurança em P2P

Para um melhor estudo, a área de segurança é dividida de acordo com os aspectos atingidos pelos ataques. No artigo [17], são propostos 4 aspectos que devem ser cobertos para se

garantir um funcionamento correto da rede, a saber: *disponibilidade*, *autenticidade de arquivo*, *anonimato* e *controle de acesso*. Nessas áreas, deve-se *"desenvolver técnicas para a prevenção, detecção, gerenciamento e recuperação de ataques"* (tradução livre).

### 3.1.1 Disponibilidade

O aspecto da disponibilidade foca na garantia de acesso aos nós e aos serviços por eles prestados, ou seja, garantir que os arquivos sendo oferecidos pela rede possam ser alcançados, e que seja possível enviar uma mensagem a qualquer um dos nós participantes na rede. Segundo Marinho[6], disponibilidade é definida em relação à parcela de tempo na qual um serviço ou objeto está disponível aos outros nós.

Os ataques a este aspecto procuram impedir o acesso a um grupo de arquivos, ou impedir que um grupo de nós acessem a rede. Para atingir este objetivo, os atacantes podem inundar a rede com requisições falsas de modo que as mensagens legítimas não cheguem ao destinatário (negação de serviço), alterar as tabelas de roteamento de forma a impedir o acesso a alguns nós (ataques de roteamento) ou adulterar as mensagens removendo alguns resultados. Estes ataques serão detalhados na Seção 3.2.

A forma mais básica de se impedir o bloqueio de acesso aos arquivos é através da replicação dos arquivos [20]. Replicar um determinado arquivo garante que em caso de falha de (ou ataque em) um dos nós responsáveis, os outros possam servir tal arquivo sem prejuízo. Porém, segundo [45], é possível um ataque à disponibilidade em redes estruturadas mesmo com replicação, utilizando-se de um ataque *Sybil* (um nó ter várias identidades; detalhes na Seção 3.2.3) de forma a se conseguir as diversas identidades necessárias para negar acesso ao arquivo determinado, em tempo $O(G)$, onde $G$ é o número de nós comuns (não maliciosos) no sistema. Um ataque deste tipo é mais barato que o ataque mais direto, de inversão de *hash*[1], que possui complexidade $O(\sqrt{S})$, onde $S$ é o tamanho da imagem da função de *hash*. Mas, apesar de $G \ll S$, a obtenção destas identidades ainda é cara em termos práticos, pois o ataque possui complexidade proporcional ao tamanho da rede.

### 3.1.2 Autenticidade de Arquivo e de Nós

A autenticidade de arquivos aborda a coerência entre a descrição e o conteúdo de um arquivo. Como qualquer usuário pode prover serviço, usuários maliciosos podem divulgar um determinado conteúdo mas fornecer outro no lugar. Os usuários que copiarem esta versão não terão acesso ao conteúdo desejado, tendo que realizar o download novamente.

[1] tentar descobrir ou gerar algum conteúdo cujo *hash* coincida com um *hash* dado

Uma grande quantidade de arquivos ilegítimos numa rede diminui a confiança na mesma, o que leva muitos usuários a desistirem do seu uso.

Considera-se como ataques à autenticidade a adulteração proposital dos arquivos. A modificação acidental é tratada de forma eficaz através de checagens de integridade, utilizando hashes criptográficos como o *md5*, ou códigos de correção de erros como o *CRC*.

Existem diversas formas de se realizar um ataque deste tipo. A forma mais simples é anunciar um arquivo e colocar outro com o conteúdo totalmente diferente, ou até mesmo fornecer um arquivo corrompido (sem conteúdo nenhum). Um ataque mais sutil é a alteração do conteúdo original. Esta alteração pode ser, por exemplo, mudar um trecho de um livro ou adicionando algum ruído no meio da música, ou pela adição de *malware* (vírus, cavalo de Tróia, etc.). A maioria destes casos a adulteração não é perceptível (uma mudança num livro não é notável a não ser que a versão original seja conhecida), o que pode fazer com que se leve um grande tempo até a detecção do problema.

A autenticidade dos arquivos têm uma relação próxima com a disponibilidade de arquivos na rede. Uma grande quantidade de arquivos ilegítimos diminui a disponibilidade dos arquivos autênticos, pois ao baixar um arquivo, a probabilidade de selecionar um arquivo ilegítimo é maior, bloqueando o acesso aos autênticos.

#### Identidade

A determinação da identidade dos nós é um fator importante, não só para sistemas de reputação, mas também para os outros aspectos de segurança. Idealmente deve-se garantir que um usuário acesse o sistema sempre com o mesmo identificador, impedindo-o de mudar de identidade durante o uso da rede, ou até mesmo de utilizar mais de um identificador ao mesmo tempo, aparecendo como mais de um nó na rede. Em relação a este último caso o atacante pode realizar certos tipos de ataque, como o *sybil* (Seção 3.2.3), ou impedir o acesso a um determinado arquivo escolhendo identidades de forma a adquirir controle de todas as cópias deste.

Porém, devido à natureza distribuída das redes P2P, a atribuição de identidades conforme descrito acima é uma tarefa complicada. Segundo [20], isso não é possível sem alguma forma de estrutura centralizada. Esta estrutura não precisa ser necessariamente um servidor centralizado explícito, já que isso seria incoerente com a arquitetura distribuída da rede, tornando-se um ponto único de falha. Um esquema para gerar identidade baseado no IP do computador já provê alguma proteção contra este tipo de ataque, possuindo na distribuição de IPs um esquema centralizado implícito. Porém, conforme ressalta [29], IPs podem ser mudados facilmente (mudados automaticamente pelo provedor de acesso, por exemplo), possibilitando mudança de identidade. No caso da mudança de IP pelo provedor demonstra que este método é pouco confiável para a manutenção de identidade, pois as informações associadas à identidade podem ser perdidas inclusive acidentalmente.

Os identificadores não podem ser escolhidos pelos nós, entretanto [8]. Se os nós puderem escolher seus próprios identificadores, então os atacantes poderão fazer ataques como o *eclipse* (Seção 3.2.4) muito facilmente, simplesmente escolhendo os identificadores adequados. A identificação deve ser atribuída ao nó, seja diretamente por um servidor responsável pela identificação, ou indiretamente através do cálculo do identificador baseado em algum dado específico ao nó. Este dado pode ser o IP, com as vantagens e desvantagens já vistas, ou as chaves criptográficas, nos sistemas que as utilizarem.

Uma forma possível de se lidar com identidades facilmente trocáveis é impor um custo para a entrada no sistema [22, 37]. Este custo pode ser uma taxa em dinheiro, o oferecimento de um serviço limitado, ou uma reputação inicial baixa. Esta estratégia visa tornar pouco rentável a troca de identidade diminuindo o ganho obtido na troca. Por outro lado, os nós honestos que entrarem no sistema também sofrerão a punição, logo ela não pode ser tão rigorosa a ponto de desestimular os novos usuários de utilizarem o sistema.

Diversos artigos [18, 29, 31, 8] comentam sobre o uso de formas de identificação através da identidade real do usuário, utilizando-se de algum documento ou o número do cartão de crédito. Estes esquemas funcionariam melhor com uma entidade centralizada responsável pela verificação das identidades. Esta abordagem, apesar ser bastante efetiva, apresenta como grande problema a invasão da privacidade dos usuários. Muitos potenciais usuários podem se mostrar resistentes ao armazenamento de dados pessoais. A seção 3.1.3 discute mais profundamente as questões de anonimato.

Outra possibilidade é a identificação através de criptografia de chaves públicas. A utilização de técnicas de criptografia possui diversas vantagens nas redes P2P: elas previnem contra o roubo de identidade ou a perda acidental, impedem que as mensagens sejam alteradas ou falsificadas durante o roteamento, etc. Além disso, a geração de chaves novas dá um trabalho extra, atrasando a geração de novas identidades.

A falta de uma autoridade certificadora poderia permitir a realização de ataques *man-in-the-middle*. Porém, este é um problema que não ocorre nas redes P2P, já que não há uma identidade associada às chaves: as chaves são a identidade. As informações de reputação são associadas diretamente às chaves, logo um nó que queira personificar outro teria que ter acesso às chaves privadas deste, ou o nó vítima acessaria a reputação do nó atacante, evitando o ataque.

O uso de criptografia de chaves públicas por si só não impede os ataques de *Sybil*, pois um nó pode gerar várias identidades. Para evitar isto, seria necessário um servidor central distribuidor de identidades, ou de uma rede de confiança, como a que foi proposta pelo Philip Zimmermann para o PGP (Pretty Good Privacy) [51]. No PGP, forma-se um grafo, onde cada pessoa é um nó, e as arestas são as ligações entre as pessoas. Cada nó mantém uma lista com as chaves públicas de seus amigos (vizinhos), adquiridos de alguma forma

segura. Dependendo da forma que a chave é adquirida, ela ganha um grau de confiança, que indicada os pesos das arestas. A partir da lista original, pode-se adicionar novos nós, com a confiança nestes novos nós calculada baseando-se no caminho de ligações entre os nós no grafo.

### 3.1.3 Anonimato

Saindo do aspecto do conteúdo, o anonimato visa garantir o direito dos usuários do sistema de o utilizar sem terem a sua identidade real exposta. Este aspecto é importante para evitar censura e repressão em países onde a liberdade de expressão é restrita, por exemplo.

Dentro do aspecto do anonimato, existem diversos subaspectos a serem tratados: devemos garantir o anonimato de *acesso aos arquivos*, de *publicação dos arquivos* e de *armazenamento dos arquivos*.

Anonimato de acesso e de publicação de arquivo refere-se à capacidade de acessar ou publicar um arquivo na rede sem ser identificado.

O subaspecto de anonimato de armazenamento é importante para complementar o anonimato de publicação, protegendo o nó que armazena os dados de terceiros. Em redes que visam o anonimato, as informações não podem ser armazenadas no mesmo nó que o publicou, pois isto o identificaria. Por causa disso, os dados são armazenados em outros nós, o que pode ser um problema, já que o usuário poderia ser considerado responsável pelo conteúdo armazenado em seu computador. O anonimato de armazenamento visa garantir que os dados armazenados num computador não possam ser associados ao seu dono, provendo o que é chamado de *negabilidade plausível (pausible deniability)* [19]. Um método para se atingir isso é através do uso de criptografia, de forma que o nó que irá armazenar os dados não tenha acesso à chave. Este método é utilizado pela rede FreeNet [10].

A forma básica de se garantir o anonimato de acesso e publicação aos arquivos é através do roteamento por *relays*. Este tipo de roteamento consiste em enviar a mensagem de comunicação aleatoriamente entre um conjunto de nós, até que um deles, definido arbitrariamente, envia a mensagem para o destinatário final. O objetivo deste comportamento é evitar a identificação da origem e do destino da mensagem. O envio da mensagem deve ser feito de forma a evitar que seja possível determinar um nó como o emissor inicial da mensagem, distinguindo-o de um *relay*. O mesmo vale para o nó destinatário: não deve ser possível determinar se o próximo nó da cadeia é o destinatário ou apenas mais um nó intermediário.

Os requisitos de anonimato geram um conflito com os sistemas de reputação: tais sistemas geralmente mantêm estado sobre os usuários e os arquivos, ação incompatível com a idéia de anonimato.

Conforme veremos no Capítulo 4, é necessário associar os valores de reputação ao usuário de alguma forma. Por esse motivo, é necessário ter no sistema algum mecanismo de identidade, através da qual pode-se registrar os valores de reputação de cada usuário. Para evitar *whitewashing* (usuário abandonar uma identidade com reputação ruim para recomeçar com outra de reputação melhor), é importante assegurar que um usuário possua apenas um identificador. Por ter que garantir uma identidade exclusiva por usuário, é possível associá-la ao indivíduo por trás dela, e conseqüentemente às suas ações. Uma possibilidade para evitar essa associação ao usuário é utilizar um esquema de identidade fraca por pseudônimos (identificador opaco [12]), evitando a associação com o usuário, mas tornando mais fácil o *whitewashing*. A relação anonimato/identidade deve ser balanceada para cada caso, de acordo com a necessidade.

### 3.1.4 Controle de Acesso

Em algumas redes, pode ser requerido que os arquivos tenham alguma forma de controle de acesso, para impedir que os arquivos sejam acessados irrestritamente. Este requisito é mais interessante em sistemas de arquivos em P2P, como o CFS [15].

De acordo com [6], existem três requisitos básicos para os mecanismos de controle de acesso poderem ser utilizados em redes P2P: não comprometer a escalabilidade, sendo preferencialmente distribuído, para evitar o aparecimento de ponto único de falha; lidar com a dificuldade de se obter identidades fortemente acopladas aos usuários, e evitar que usuários maliciosos tenham acesso indevido a arquivos protegidos, ou que usuários autênticos percam acesso devido; e manter o incentivo ao compartilhamento de arquivos apesar da restrição de acesso.

## 3.2 Ataques em Redes P2P

Existe uma grande quantidade de ataques em redes P2P. Estes ataques podem variar do mais genérico, efetivo em qualquer tipo de rede, aos mais específicos, que exploram alguma vulnerabilidade específica da rede alvo. Cada ataque visa um dos aspectos de segurança descritos anteriormente, podendo eventualmente abranger mais de um ao mesmo tempo.

### 3.2.1 Arquivos Ilegítimos

Este tipo de ataque é bem simples, consistindo em disponibilizar na rede um arquivo com o conteúdo diferente do anunciado. As possibilidades mais comuns são [9]:

**arquivo adulterado:** o arquivo disponibilizado possui o conteúdo original modificado, faltando partes, com vírus ou algum outro tipo de *malware*. Exemplo: colocar uma

música substituindo trechos com barulhos aleatórios, ou colocando programas com vírus;

**arquivo trocado:** quando o arquivo disponível é anunciado com um nome, mas na verdade é um outro arquivo;

**arquivo corrompido:** o arquivo publicado não é um arquivo válido, consistindo apenas de dados aleatórios.

Enquanto os dois últimos ataques podem ser percebidos tão logo o arquivo seja baixado, o primeiro pode demorar para ser detectado, ou nem ser. Isso gera dois problemas: primeiro, nós considerados como honestos passam a hospedar arquivos ilegítimos, o que pode enganar os nós que tinham confiança nestes; segundo, aumenta a replicação dos arquivos ruins, o que aumenta a probabilidade deles serem baixados.

Independente da forma de execução deste ataque, o objetivo é diminuir a disponibilidade de um determinado arquivo, dificultando o acesso ao arquivo original. Isso ocorre porque a existência de arquivos ilegítimos na rede cria uma probabilidade de se baixar um arquivo que não é o esperado, forçando os usuários a tentarem realizar o download novamente. Dado uma probabilidade alta de se baixar um arquivo corrompido, é possível que os usuários desistam de tentar baixar o arquivo pretendido, ocorrendo uma negação de serviço.

Outro dano causado por este tipo de ataque é a diminuição da credibilidade da rede. Se muitos arquivos numa determinada rede são falsos, a confiança nesta vai declinando, a ponto das pessoas eventualmente desistirem de utilizá-la.

### 3.2.2 Free-Riders

*Free Riding* não é um ataque propriamente dito, mas sim um comportamento danoso dos usuários. Este comportamento consiste em utilizar os recursos do sistema sem fornecer nada em troca. Exemplificando com o caso de troca de arquivos, o free-rider não compartilha nenhum arquivo, apenas copia dos outros participantes. Este tipo de comportamento, além de ir contra o comportamento esperado das redes P2P, diminui a quantidade de arquivos disponíveis, reduzindo a utilidade da rede. Outro problema causado é a sobrecarga dos nós que disponibilizam arquivos, o que pode levar a um efeito cascata, no qual os nós que compartilham arquivos param de disponibilizá-los para se livrarem da sobrecarga, piorando a distribuição de carga nos que continuam fornecendo, e assim em diante.

Devido ao seu efeito danoso, foram desenvolvidas táticas para evitar tal comportamento. Sistemas de reputação podem ser eficientes nestes casos, exigindo que os usuários forneçam arquivos à rede antes de baixarem. Usuários que não compartilham arquivos, ou compartilham arquivos falsos possuem baixa reputação, enquanto quem compartilha

recebe uma reputação maior. Pode-se utilizar esta reputação para definir prioridade para acesso, ou até mesmo bloquear acesso aos nós com um nível de reputação abaixo de um certo limiar. Os usuários que não compartilharem arquivos terão dificuldades em utilizar os serviços da rede, liberando recursos para os usuários que colaboram.

### 3.2.3 Ataques Sybil

Este ataque, descrito em [20], é um ataque no qual um único atacante se passa por diversos nós. Uma vez controlando diversos nós na rede, o atacante pode realizar sozinho algum ataque para o qual seria necessário uma grande quantidade de nós.

A viabilidade deste ataque ocorre devido às fracas verificações de identidade que ocorrem na maioria das redes. Se não houver verificação de identidade, ou a verificação for em série, um atacante qualquer pode simular quantas identidades desejar. Mas mesmo que a verificação seja feita paralelamente, um atacante pode falsificar tantas identidades quantas vezes forem maiores a capacidade de processamento dele em relação à vítima. Considera-se nestes casos uma verificação na qual o nó identificador envia algum tipo de desafio para o nó a ser identificado, de tal forma que supostamente um nó seja incapaz de resolver dois destes de uma única vez, forçando que cada computador/usuário corresponda a apenas um nó.

A solução apontada para evitar este tipo de ataque é a utilização de alguma forma de controle centralizado, mesmo que indiretamente, conforme previamente discutido na seção3.1.2.

Exemplos de ataques possíveis a partir do *sybil* são: ataque de Eclipse (ver seção 3.2.4) e ataque de inflação de reputação[2].

No artigo [45], Srivastra discute diversos ataques possíveis baseados no ataque *sybil*, focando-se em redes estruturadas utilizando DHT. Entre estes ataques encontram-se ataques de roteamento, onde os diversos nós maliciosos controlados pelo atacante divulgam rotas falsas, impedindo o acesso a um ou vários nós; corrupção dos dados, com os nós entrando no sistema em diversos pontos diferentes através do uso de ids diferentes de forma a obter o controle de todas as cópias de um arquivo, anulando os benefícios da replicação de dados.

### 3.2.4 Eclipse

Um outro ataque possível contra a disponibilidade de arquivos e nós é o chamado ataque de *Eclipse*, onde um grupo de nós maliciosos agindo em conjunto pode isolar parcial ou totalmente um pedaço da rede, impedindo a comunicação entre ambos, filtrando o

[2]falsificar votos para melhorar ou piorar a imagem em sistemas de reputação, recomendação ou qualquer sistema baseado em votos dos outros pares - ver seção 4.5.2

conteúdo que é permitido ser acessado[43]. Este ataque pode ser feito tanto por diversos nós maliciosos diferentes, atacando em conluio, ou por um nó utilizando-se de múltiplas identidades.

Em [8], é descrito melhor o funcionamento destes ataques: suponha uma rede estruturada com replicação de dados. Um grupo de nós pode escolher um grupo de ids específico, de forma que cada um destes nós fique responsável por uma cópia do arquivo cujo controle se deseja obter. Assim, todas as requisições para este arquivo necessariamente passam por um dos nós maliciosos, que determina se o conteúdo original é fornecido, ou um conteúdo ilegítimo é repassado, ou até mesmo a existência do arquivo pode ser negada.

De forma semelhante pode-se isolar um nó na rede. Em redes estruturadas, onde os vizinhos de um determinado nó são previsíveis, é possível obter os identificadores necessários para intermediar todas as comunicações entre o nó vítima e o resto na rede. Feito o isolamento, pode-se controlar totalmente o acesso deste nó ao resto da rede, restringindo buscas, comunicação com os outros nós, etc.

Uma outra forma de realizar este ataque, isolando um subgrupo de nós, é criando uma subrede com os nós maliciosos, e induzir os nós que estão entrando na rede a utilizarem um destes nós para realizar o *bootstrap* (entrada na rede). Em caso de sucesso, os nós vítimas recebem informações de roteamento incluindo apenas os nós maliciosos, ficando presos na subrede.

Para evitar tais ataques, uma possibilidade é a utilização de ids seguros, que impossibilitem que um único atacante adquira vários ids e que garanta a distribuição aleatória de ids para os nós. Este último requisito dificulta consideravelmente que um grupo de atacantes maliciosos consiga os ids necessários para a execução dos ataques. Outra possibilidade é a utilização de caminhos alternativos aos determinados pela rede P2P utilizada, como forma de confirmação: se o resultado da utilização de ambos não forem iguais, então houve uma adulteração do resultado, possivelmente por um nó malicioso. Com estas técnicas diminui-se a possibilidade dos primeiros tipos de ataques descritos.

O ataque de isolamento de rede pode ser evitado através da utilização de um conjunto de nós padrão para o *bootstrap*. O conhecimento destes pode se dar através de uma lista inserida previamente nos programas que rodam nos nós. Porém, esta abordagem possui como desvantagem o fato de que os nós responsáveis tornam-se pontos de falha, impedindo a entrada de novos nós caso todos falhem, não sendo uma solução escalável para uma grande entrada de nós na rede.

### 3.2.5 Negação de Serviço

No ataque de *Negação de Serviço (DoS - Denial of Service)* o atacante inunda um nó com mensagens inúteis, de modo que o nó vítima não consiga receber nenhuma comunicação

efetiva, afetando a disponibilidade do sistema. Isso pode ser feito com diversos objetivos: impedir que este nó responda a uma mensagem de busca, ou impedindo o acesso a um arquivo (mais efetivo se for um arquivo raro); pode-se impedir a participação total de um nó na rede desta forma.

Além de inundar o nó, pode-se inundar a rede, causando um retardamento no roteamento das mensagens ou até uma parada total na rede.

A inundação para o ataque pode ser feita tanto por um DoS tradicional (inundando os links físicos) quanto na camada de aplicação, utilizando-se de falhas no protocolo ou de efeitos de ampliação. Outra possibilidade de ataque pela rede *overlay*, sem ser por inundação, é alterar as mensagens sendo roteadas, para remover os resultados apresentados, de modo a dar ao nó que realizou a pesquisa a impressão de que o arquivo encontra-se indisponível na rede. Este último tipo de ataque pode ser difícil de ser realizado, caso haja a necessidade de interceptar a mensagem na camada IP por meio de um ataque *man-in-the-middle* ou domínio de roteador, mas caso o atacante se encontre no caminho do roteamento interno da rede P2P, esse ataque é bastante factível.

### 3.2.6 Ataques de Roteamento

Ataques de roteamento visam impedir o funcionamento correto da rede interferindo no roteamento das mensagens. Nas redes não-estruturadas este tipo de ataque consiste basicamente em não rotear uma mensagem, tendo em vista que não há nenhuma estrutura no posicionamento dos nós na rede. Este ataque pode ser bastante ineficiente no caso das buscas em *broadcast*, tendo em vista que apenas um ramo da busca será interrompido; os outros seguirão normalmente[3]. Nas redes estruturadas, onde geralmente é enviada apenas uma mensagem, este ataque pode efetivamente impedir o funcionamento da busca. Existem três categorias deste tipo de ataque [44]:

**roteamento incorreto:** um nó malicioso simplesmente deixa de rotear a mensagem, ou roteia para um nó incorreto, distante do nó ao qual ele deveria realmente ter roteado;

**atualização incorreta de rota:** o(s) nó(s) maliciosos enviam atualizações de rota erradas, induzindo os outros nós ao erro;

**particionamento de overlay:** um grupo de nós maliciosos podem formar uma rede paralela à original, e inserir nesta rede os nós que tentarem entrar através dos nós maliciosos.

[3] supondo que haja apenas um adversário; mais de um atacante permitiria o bloqueio de mais de um ramo.

## 3.3 Conclusões

A popularidade dos sistemas de P2P para o compartilhamento de arquivos, junto com as pesquisas por redes mais eficientes têm tornado este tipo de rede forte candidato a aplicações mais críticas, substituindo o tradicional modelo cliente/servidor. Mas, com a popularidade, vem a exposição maior das falhas de segurança. Este capítulo teve como objetivo apresentar uma visão geral do estado da segurança em redes P2P, discutindo os principais problemas e as possíveis soluções.

Os principais aspectos de segurança em redes P2P foram descritos, mostrando a importância de cada um para o funcionamento da rede. Finalmente foram descritos os principais ataques encontrados na literatura sobre segurança em P2P, mostrando os aspectos que eles visam, e os principais métodos de resolução. A maioria dos ataques é possível devido à dificuldade de se obter identidades fortes sem comprometer a usabilidade da rede e/ou a identidade real do usuário.

# Capítulo 4

# Sistemas de Reputação

Um dos principais tipos de ataques que as redes P2P têm sofrido é o de arquivos ilegítimos. A freqüência destes ataques se dá pela facilidade de sua execução, e pela eficiência. A presença de arquivos ilegítimos, tanto corrompidos quanto adulterados, causa um aumento no consumo dos recursos da rede, pois os nós podem ter que baixar diversas cópias do mesmo arquivo até conseguirem encontrar uma cópia legítima. Além do aumento da utilização da rede, uma grande quantidade de cópias ruins pode esconder as cópias legítimas, tornando o arquivo indisponível. Ambos problemas causam uma diminuição na confiança da rede, podendo inclusive a levar ao seu desuso.

Os sistemas de reputação visam a resolução deste problema, atribuindo aos nós e/ou aos arquivos um valor que indica o nível de confiança que se tem no avaliado. As opiniões de reputação são atualizadas pelos usuários da rede a medida que eles vão a utilizando, aumentando a reputação dos nós bem comportados e diminuindo a dos maliciosos, de forma a discriminá-los.

A abordagem por sistemas de reputação não elimina totalmente o download de arquivos corrompidos. Para que um nó ou arquivo seja classificado como ruim, deve-se baixar um arquivo corrompido para que um ser humano avalie o seu conteúdo, visto que este não é um trabalho que os computadores consigam fazer automaticamente. Porém, estes sistemas provêem uma boa redução na quantidade de arquivos ruins baixados. Segundo [29], o principal objetivo do sistema de reputação é reduzir o número de arquivos que um usuário deve examinar até achar o arquivo correto, sendo esta a medida da eficiência do sistema.

## 4.1 Conceito de Confiança, Reputação e Contexto

Os sistemas de reputação procuram gerenciar as opiniões sobre o comportamento dos nós e/ou a qualidade dos arquivos. Estas opiniões são combinadas para formar a reputação,

que é utilizada posteriormente para determinar a confiança do nó. Para uma melhor definição dos sistemas de reputação, é interessante definir primeiro os conceitos de confiança e de reputação. Jøsang[41] define confiança (*trust*) como:

> "Trust is the extent to which one party is willing to depend on something or somebody in a given situation with a feeling of relative security, even though negative consequences are possible."

Esta definição mostra os aspectos principais em relação à confiança: dependência em quem realizará alguma ação por você, o risco de conseqüências negativas caso os resultados não sejam os esperados, e a expectativa de que tal risco não se concretizará. Esta expectativa é a confiança em si. A confiança, segundo [41], é algo pessoal, único para cada indivíduo. Essa confiança é baseada na experiência direta. Comparando com a definição de reputação, dada pelo mesmo artigo de acordo com o *Concise Oxford Dictionary*:

> "Reputation is what is generally said or believed about a person's or thing's character or standing"

Termos portanto a reputação de alguém ou algo como a agregação das opiniões de diversas fontes, sendo ligado a um grupo de pessoas. Para os indivíduos pertencentes ao grupo, a reputação de alguém pode ou não estar de acordo com a sua confiança nesta pessoa. A reputação pode ser considerada menos precisa que a confiança, por ser baseada na confiança de outras pessoas, cuja opinião pode ou não ser confiável sob o seu ponto de vista. Porém, para sistemas peer-to-peer, onde a quantidade de nós e muito grande para se ter uma opinião própria sobre cada nó, o compartilhamento de opiniões é bastante vantajoso, servindo de base para a determinação da confiança pelos nós.

O cálculo da reputação, como visto, leva em consideração a opinião de diversos nós. Mas, assim como não se confia previamente no nó sendo avaliado, também não se confia necessariamente nos nós fornecedores de opiniões. Um nó malicioso pode fornecer opiniões erradas, indicando outros nós maliciosos como honestos e vice-versa. Uma das formas possíveis de se evitar isso é através da verificação da reputação do nó que está fornecendo a opinião, de forma a se ter o quanto esta opinião é confiável. A confiança final seria a opinião dada ajustada pela reputação do nó avaliador. Caso a reputação do avaliador seja obtida de um outro nó, pode-se obter também a reputação deste, e assim sucessivamente até que se obtenha uma reputação confiável, geralmente uma avaliação própria. Este caminho de indicações forma uma *rede de confiança*. Esta rede de confiança segue o mesmo principio de funcionamento da descrita na seção 3.1.2.

Outro ponto importante em relação à confiança e reputação é o conceito de contexto [1]. A confiança ou a reputação que depositamos em alguém refere-se a alguma situação: você

pode confiar numa pessoa para dirigir um carro, mas não para consertar um computador, por exemplo.

No caso de sistemas de reputação, temos geralmente dois contextos [41]: o de fornecimento de arquivos e o de fornecimento de opiniões. Dependendo das funcionalidades da rede P2P, podemos ter também o contexto de fornecimento de índices. Em diversos sistemas, porém, esta divisão em contextos não é realizada, pois estes sistemas focam-se prioritariamente em detectar comportamento malicioso de um modo geral.

## 4.2 Base Econômica da Reputação

Por ser formado por diversos agentes racionais independentes interagindo entre si e interferindo nos resultados dos outros, é possível realizar uma análise econômica do funcionamento da rede. Diversos artigos realizam esta análise, a maioria abordando questões relativas a free-riders, mas alguns, como [26] também abordam a economia das redes P2P levando em consideração nós maliciosos.

Para o estudo econômico dos sistemas de reputação, são utilizados modelos de teoria dos jogos. A teoria dos jogos procura predizer o comportamento de um sistema onde as decisões dos participantes interferem nos resultados dos outros, até a situação de equilíbrio do sistema. Cada jogador toma decisões de acordo com uma estratégia, podendo mudá-la de acordo com os resultados obtidos. A situação de equilíbrio ocorre quando se torna improvável que os jogadores mudem de estratégia, o que faz com que o jogo atinja uma situação estável. O sucesso dos jogadores é determinado pela *função utilidade*, um valor numérico que indica o grau de sucesso do participante no jogo. O jogo em si geralmente é descrito por uma tabela que indica os ganhos para todas as combinações possíveis de decisões dos jogadores.

O estudo econômico das redes P2P é feito baseado no *Dilema do Prisioneiro Iterado (Iterated Prisoner's Dilemma)* [4]. O Dilema do Prisioneiro descreve um jogo econômico onde dois participantes devem escolher entre cooperar ou não cooperar entre si. Caso ambos cooperem, ambos ganham, com a soma máxima dos ganhos; se ambos não cooperarem, os dois são punidos; finalmente, se um coopera e outro não, o traidor recebe o benefício máximo, e o outro a punição máxima. No dilema do prisioneiro iterado os participantes repetem diversas vezes este jogo.

O Dilema do Prisioneiro é geralmente representado por uma tabela de ganhos/perdas dos participantes. A Tabela 4.1 mostra um exemplo de matriz de pagamentos para o Dilema do Prisioneiro. Um dos participantes tem as suas decisões na vertical, e o outro na horizontal, com os valores nas células da tabela representando os ganhos do jogador na vertical/horizontal. Podemos ver que, quando ambos colaboram, cada um tem um ganho de 5, com os ganhos totais da rede somando 10; caso apenas um colabore, este perde 2 (-2),

Tabela 4.1: Exemplo de uma matriz de pagamentos para o Dilema do Prisioneiro

| | Colaborar | Não colaborar |
|---|---|---|
| Colaborar | 5\5 | -2\8 |
| Não colaborar | 8\-2 | 0\0 |

enquanto o que não colaborou possui um ganho maior (8). Porém, neste caso a rede tem um ganho total menor que no caso da colaboração mútua. Caso ambos não colaborem, ninguém ganha nada. Esta matriz é simétrica, pois nos casos onde um coopera e o outro não, os ganhos e as perdas são iguais, independente de qual lado colaborou.

Para a modelagem das redes P2P, é utilizada uma variação denominada *Dilema do Prisioneiro Generalizado (Generalized Prisoner's Dilemma)* [21]. Esse jogo difere do dilema original pela assimetria na matriz de pagamentos. Tal assimetria é causada pela diferença de papel entre os participantes: a não cooperação pela parte interessada em receber o arquivo possui pagamentos diferentes da não cooperação do fornecedor de arquivos. Como exemplo deste tipo de matriz, temos a Tabela 4.2, obtida de [21]. Nela podemos observar que a não colaboração por parte do interessado em baixar o arquivo (Não pede serviço) não implica em nenhum custo ou benefício para a rede. Apenas quando este nó requisita serviço, há a possibilidade de ganhos na rede, caso o fornecedor de arquivo coopere.

A modelagem através do Dilema do Prisioneiro é utilizada para o combate dos free-riders. No caso dos nós maliciosos, este modelo não é válido, pois eles possuem uma função utilidade diferente da dos nós normais. O objetivo dos nós maliciosos não é obter o benefício máximo da rede, mas sim impedir seu o bom funcionamento. Os motivos para tal comportamento podem variar de criadores de vírus desejando espalhar suas criações ou *SPAM* a empresas proprietárias de direitos autorais sendo violados pela rede.

Em [26], Krishnan descreve a seguinte motivação para o comportamento malicioso, no caso de um possuidor de direitos autorais sendo violado: a utilidade obtida por um nó ao adquirir uma música, comprando ($c$) o álbum, por exemplo, é $U - p_c - sc_c$, onde $U$ é o benefício de se ter a música, $p$ é o preço da música e $sc$ é o custo associado à busca pela música; a utilidade de baixar ($b$) a música por uma rede P2P é $U - sc_b$, visto que o custo é zero. Para que se faça valer a pena comprar o álbum pela gravadora, é necessário que $U - p_c - sc_c > U - sc_b$. Para fazer com que baixar o arquivo seja menos

Tabela 4.2: Exemplo de uma matriz de pagamentos para o Dilema do Prisioneiro Generalizado

| | Fornece serviço | Ignora requisição |
|---|---|---|
| Pede serviço | 7\-1 | 0\0 |
| Não pede serviço | 0\0 | 0\0 |

vantajoso, é necessário ou abaixar os custos para a compra, ou aumentar os custos de baixar. O custo de baixar o arquivo pode ser aumentado tornando a busca mais custosa. Colocando arquivos corrompidos na rede faz com que os nós que queiram baixar tais arquivos precisem baixar o arquivo diversas vezes até encontrar o arquivo autêntico é uma forma de se aumentar o custo de baixar a música.

Os sistemas de reputação funcionam discriminando os nós maliciosos dos bons. Para tal, ele agrupa opiniões sobre o comportamento dos nós durante as transações, provendo estas informações para impedir um mal serviço nas iterações futuras. No caso dos free riders, uma reputação ruim pode fazer com que serviço seja negado aos nós que não compartilhem arquivos, tornando interessante para os nós a colaboração com a rede para que a reputação seja mantida alta. A possibilidade que as suas atitudes no presente influenciem a recepção de serviços no futuro é chamado de *shadow of the future* [4, 37]. Esta preocupação é a responsável pelo incentivo aos nós de cooperarem. Como os nós maliciosos não têm nenhuma expectativa de receberem serviços da rede, os sistemas de reputação simplesmente servem para indicar quais nós são confiáveis.

## 4.3 Características de um Sistema de Reputação

Existem algumas características que um sistema de reputação requer para que ele funcione bem. Resnik [37] define três propriedades básicas para o funcionamento destes sistemas:

- grande tempo de vida dos nós, pois isso gera expectativa de interações futuras (*shadow of the future*);
- captura e distribuição de informações sobre as interações passadas;
- uso das informações acumuladas para guiar decisões futuras.

Estas três propriedades abordam três características básicas de sistemas de reputação [31]: *identidade*, *métrica* e *resposta*.

Um esquema de identidade forte é importante para que se associe de forma confiável a reputação ao nó devido. Identidades fracas permitem que os atacantes possam se livrar da reputação ruim, simplesmente trocando de identidade e reentrando na rede com a identidade nova, ou até roubem a identidade de um outro nó, se apoderando de sua reputação.

Coleta de informações das interações passadas é a funcionalidade principal de um sistema de reputação. É pelo comportamento passado que calculamos a reputação dos nós. Este cálculo é feito através de uma função que mapeia um histórico de comportamentos a um valor, pelo qual podemos comparar os nós e decidir sobre a sua índole. À função de mapeamento e dado o nome de *métrica*, e à imagem desta função de *reputação*.

Finalmente, de posse dos valores de reputação, deve-se utilizá-los para determinar se um arquivo deve ou não ser baixado nas interações futuras. Se a métrica mapear corretamente o comportamento dos nós, então será possível evitar o download dos arquivos ilegítimos.

Karmvar [24] descreve também algumas questões importantes a serem tratadas para um bom funcionamento destes sistemas:

**auto-policiado:** como as redes P2P dispensam servidores para o seu funcionamento, seria incoerente o desenvolvimento de um sistema de proteção que necessitasse de um, pois isto adicionaria um ponto único de falha. O sistema de reputação deve funcionar de forma distribuída, com as responsabilidades atribuídas aos nós;

**anonimato:** deve-se manter o anonimato dos nós, na medida do possível. O anonimato completo não é possível com o uso de um sistema de reputação, pois o funcionamento destes implica na coleta e distribuição de dados sobre o comportamento dos nós, além da necessidade de se ter identidades fortes. Porém, é possível ter algum nível de anonimato evitando a associação da identidade real do usuário a um usuário do sistema, através do uso de *pseudônimos*;

**desconfiar dos novatos:** pela facilidade que a rede permite a troca de identidades, reforçada pela necessidade de anonimato para os nós, não temos como saber se um nó entrante é realmente um participante novo da rede, ou algum nó malicioso fazendo *whitewashing*. A solução para isso é ou reforçar as identidades, tarefa extremamente difícil em redes descentralizadas, ou punir os novatos, de forma a tornar desinteressante a troca de identidade;

**resistência a conluios:** a rede de ser capaz de lidar com um grupo de nós maliciosos agindo em grupo para subverter a rede.

## 4.4 Componentes de um Sistema de Reputação

Baseado nas características descritas, os sistemas de reputação podem ser divididos em diversos componentes. Essa divisão é interessante para que se estude mais profundamente o funcionamento das partes. Marti [31] define três componentes: *coleta de informação*, *métrica* e *resposta*. Eles se assemelham com as características descritas pelo Resnik [37], exceto pela questão da identidade.

Ooi [34] também fornece uma divisão para os sistemas de reputação, mas com uma visão mais voltada para a implementação. Os componentes descritos são: *armazenamento*, *integridade*, *métrica* e *identidade*.

As duas divisões possuem focos diferentes, e são de certa forma, complementares. Podemos juntá-las, adicionando outras características importantes, obtendo uma divisão mais abrangente:

- sistema base;
- objetivo;
- acúmulo de informação;
  - armazenamento;
  - integridade;
- scoring ou métrica;
- resposta;
- identidade.

O sistema base refere-se à rede P2P sobre a qual o sistema de reputação é implementado. Vários parâmetros do sistema de reputação são influenciados pelo sistema base, daí sua inclusão na lista de características. O grau de estruturação da rede, por exemplo, influi no armazenamento da reputação, pois uma rede estruturada possibilita o armazenamento centralizado dos valores de reputação através da DHT, enquanto as redes não estruturadas favorecem as redes de confiança, com cada nó armazenando as próprias opiniões.

As outras características serão descritas nas próximas seções, exceto pela identidade, já descrita na Seção3.1.2.

### 4.4.1 Objetivo

Os sistemas de reputação geralmente se especializam em um tipo de atacante [31]. Isso acontece porque as medidas para combater um tipo de atacante não funcionam para os outros. Como exemplo, as medidas para combater os free-riders, como sistema de crédito/débito, não funcionam contra nós maliciosos, visto que os últimos não têm intenção de copiar arquivos do sistema, tornando esta resposta inútil.

Os dois principais tipos de adversários nos quais os sistemas de reputação se especializam são os nós *egoístas* e os *maliciosos* [31].

Os nós egoístas visam obter os benefícios da rede sem contribuir com ela. São os chamados free-riders. O combate aos nós egoístas é feito através de incentivos para a cooperação, negando o download a quem não tenha realizado uma determinada quantidade de upload, ou dando privilégios a quem compartilha arquivos.

Já os nós maliciosos têm como objetivo impedir o bom funcionamento do sistema. As táticas utilizadas por estes nós incluem fornecer arquivos falsos, ou arquivos com vírus[1], etc. de forma a dificultar o acesso aos arquivos legítimos.

As estratégias utilizadas para minimizar os efeitos dos ataques de nós maliciosos envolvem manter um valor de reputação indicando a confiabilidade do nó, atualizando-o de acordo com a qualidade do arquivo fornecido por este nó. Ao selecionar um arquivo para download, essa informação de reputação é utilizada para escolher o arquivo proveniente de um nó cuja reputação seja confiável o suficiente para garantir uma que uma boa cópia seja entregue. Outra possibilidade de uso de reputação se dá com a atribuição de reputação aos arquivos ao invés dos nós, ou a ambos. Neste último caso, deve-se combinar as métricas para se tomar a decisão sobre baixar ou não o arquivo.

### 4.4.2 Acúmulo de Informação

#### Armazenamento

O armazenamento dos dados de reputação pode ser feita de diversas formas. Geralmente ele é feito de acordo com a estrutura de armazenamento da rede, mas não obrigatoriamente. As principais formas de armazenamento são: no nó que deu a avaliação, no nó avaliado ou utilizando a infra-estrutura de DHT.

O armazenamento no nó que deu a opinião é geralmente utilizado em redes não-estruturadas, e consiste em cada nó manter as suas avaliações. Como ponto positivo, temos uma confiança maior na integridade do valor, dado que a opinião é obtida com o nó que teve a experiência, logo não há risco de adulteração da opinião. Além disso, temos junto com a opinião a identidade do nó que a emitiu, logo podemos verificar a reputação deste nó para saber o quão confiável as suas opiniões são. Como pontos negativos temos a dificuldade de se encontrar as opiniões sobre um determinado nó na rede, pois é necessária uma busca para encontrar as opiniões sobre um nó, com as mesmas características das buscas em redes não estruturadas discutidas na Seção 2.2.2. Apesar de nem todas as opiniões sobre um nó/arquivo serem encontradas com esta forma de armazenamento, espera-se que a capacidade de verificar a reputação dos nós que as forneceram torne a reputação calculada mais confiável [31].

Uma outra possibilidade, que não exige o uso de uma rede estruturada mas ainda assim provê uma grande eficiência na obtenção das opiniões, é armazenar as opiniões ou a reputação no nó avaliado. Assim, ao contatar um nó para obter o arquivo, pode-se pedir a ele as opiniões emitidas pelos outros nós sobre si, sem a necessidade de realizar uma

[1]Outros tipos de ataques para impedir o funcionamento da rede também são possíveis, como ataques às tabelas de roteamento ou negação de serviço. Estes ataques, portanto, não são evitados por sistemas de reputação

busca custosa. Por outro lado, essa abordagem deve lidar com o problema da integridade, pois é de interesse direto do nó adulterar as opiniões por ele sendo armazenadas. Veremos mais como lidar com problemas de integridade no próximo tópico.

Para redes estruturadas, é possível armazenar os dados na DHT. Ao fim de uma transação, o nó que baixou o arquivo envia ao nó responsável pelo armazenamento da reputação do nó que forneceu o arquivo uma mensagem com a sua opinião. A reputação do nó sendo avaliado é atualizada de acordo com a nova opinião. O nó responsável é escolhido através do *hash* do id do nó, da mesma forma que os outros dados armazenados. Como todos os nós enviam as suas opiniões para um único nó, e por esse estar localizado em uma posição definida, consegue-se encontrar facilmente as reputações. Entretanto, como todas as opiniões já estão agregadas, não temos como verificar a confiabilidade das opiniões individuais, sendo possível minar a reputação de um nó com opiniões falsas. Por outro lado, o fato de se ter acesso a todas as opiniões emitidas sobre um nó torna mais provável que a soma das opiniões seja confiável [31, 37]. Uma outra medida que pode auxiliar no problema da qualidade das informações é o uso de *prova de transação* [18]: o nó, ao enviar uma opinião, envia também uma prova de que a transferência realmente ocorreu. Dessa forma é evitado que nós maliciosos inundem a rede com opiniões sobre transações inexistentes.

### Integridade

Dependendo do local de armazenamento da reputação, é necessário garantir a integridade dos dados. Caso o próprio nó armazene a sua reputação, não há problema, mas nos casos onde a reputação é armazenada em outros nós, existe a possibilidade de que haja adulteração das opiniões. Segundo as possibilidades de armazenamento consideradas acima, há duas situações nas quais é necessário alguma estratégia de manutenção de integridade: armazenamento em rede estruturada e no nó avaliado.

No primeiro caso, como o nó responsável é escolhido aleatoriamente, é pouca a possibilidade de conluio com algum nó de quem ele seja responsável pelo armazenamento. Ainda assim, é possível que um nó malicioso modifique as reputações armazenadas, invertendo as reputações para induzir outros nós a baixarem algum arquivo corrompido qualquer. Desta forma, este nó conseguiria subverter o funcionamento da rede, mesmo não estando em conluio com nenhum nó de quem ele armazene a reputação. A solução comumente empregada nestes casos é a replicação: a reputação de um nó é armazenada em $n$ lugares diferentes, cada um escolhido com uma função de hash distinta. A opinião que figurar na maioria dos índices é considerada a autêntica. Como todos os índices são escolhidos aleatoriamente, é bem pequena a probabilidade de que a maioria dos nós responsáveis por um nó sejam maliciosos.

O segundo caso apresenta um problema maior, visto que o nó tem interesse direto

em adulterar a sua reputação. Nestes casos, a solução usualmente adotada é assinar a reputação, de modo que qualquer modificação seja percebida. Dois sistemas de reputação que utilizam este esquema adotam abordagens diferentes para a assinatura: Gupta em [23] descreve um sistema no qual a assinatura é emitida por um servidor de cálculo de reputação. Os nós trocam mensagens assinadas, que funcionam como prova de transação. Estas mensagens são enviadas posteriormente ao servidor, que atualiza a reputação no nó e envia um certificado assinado novo. Já no RCertP/RCertPX [34, 27] é mantida uma lista com as opiniões de todos os nós a quem foi provido serviço, cada uma assinada pelo nó que forneceu a opinião.

Mesmo com estas medidas, ainda é possível que o nó adultere a sua reputação, descartando a opinião nova caso ela seja negativa. no RCertPX, este problema é resolvido através da revogação do certificado antigo. Ao emitir uma opinião, o nó entra em contato com o último a emitir opinião para revogar o certificado emitido por ele, e envia o certificado substituo ao nó avaliado. Um nó que queira checar a reputação de um nó contata o último a enviar uma opinião para garantir que o certificado ainda é válido. Caso o certificado não seja válido, então houve uma tentativa de adulteração, logo o nó não é confiável. Gupta resolve este problema para o crédito/débito mantendo no servidor a contagem de débitos, deduzindo-os da próxima requisição de crédito.

### 4.4.3 Scoring ou Métrica

A métrica define como os diversos parâmetros referentes às transações são reunidos num valor indicando a reputação do objeto avaliado, assim como quais parâmetros são utilizados. Ela pode ser vista como uma função, mapeando o conjunto de parâmetros (domínio) a uma imagem, na qual se pode comparar os nós pela sua reputação.

A reputação geralmente é calculada para um nó ou um arquivo. A reputação para nós tenta indicar a qualidade dos serviços prestados pelo no, seja fornecendo arquivos ou emitindo opiniões. Como o comportamento dos nós pode variar com o passar do tempo, a reputação pode não mais condizer com o comportamento atual. Já a reputação dos arquivos indica se este arquivo é legítimo ou corrompido. Uma das vantagens da reputação por arquivo é imutabilidade do conteúdo do arquivo[2], o que faz com que não se tenha problemas com mudança de comportamento. Por outro lado, a grande quantidade de versões diferentes de arquivos podem aumentar em muito a quantidade de reputação a ser armazenada.

Diversos parâmetros podem ser utilizados para o cálculo da reputação de um nó/arquivo, dependendo do objetivo a ser alcançado. O mais habitual é analisar o comportamento

[2]realizando-se a associação da reputação com o conteúdo do arquivo, uma mudança no conteúdo resultaria numa nova versão.

dos nós em termos de cooperação ou não-cooperação. Porém, apesar de ser um parâmetro mais relevante que o número de cooperações, em alguns casos é difícil detectar uma não-cooperação. Um exemplo de cooperação não detectável ocorre quando um nó se recusa a responder a uma busca apesar de ter o arquivo procurado [31]. No caso dos arquivos temos como parâmetros mais usuais o número de opiniões positivas ou negativas.

Outros fatores abordam características mais específicas. Com a finalidade de equilibrar a quantidade de bytes enviada e recebida, pode-se considerar na reputação o tamanho dos arquivos e a banda disponibilizada por um nó. A reputação de quem forneceu uma opinião pode ser considerada, aumentando a confiança na reputação obtida. Ainda na questão segurança, é comum o uso do tamanho da transação realizada (em bytes) e de algum fator de envelhecimento no cálculo da métrica para evitar alguns comportamentos de traição. O uso do tamanho da transação evita traições do tipo ganhar reputação com transações pequenas para aplicar um golpe em uma transação grande, e o fator de envelhecimento evita a mudança do comportamento para malicioso depois de acumular uma boa reputação.

Existem algumas características que definem uma boa métrica. Dingledline [18] descreve um conjunto de características que as métricas devem possuir. Nem todas são possíveis de serem possuídas simultaneamente, pois algumas são contraditórias entre si. Os sistemas devem, portanto, possuir o maior número possível destas características.

**Bom desempenho em longo prazo:** o sistema deve manter o nível de confiança na reputação calculada dos nós durante o passar do tempo. Ele deve ser capaz de distinguir também entre um nó entrante e um com um longo histórico de atos ruins;

**priorizar as interações recentes:** as últimas avaliações devem possuir um peso maior na reputação. Isso permite que um nó que mude de comportamento, passando a agir de forma maliciosa, possa ser detectado rapidamente;

**eficiente:** deve ser possível calcular a métrica de forma rápida;

**resiliente à ataques:** nós maliciosos vão tentar subverter o sistema de reputação de forma a obter uma boa reputação, ou para sujar a reputação dos nós honestos. O sistema deve ser capaz de resistir a estes tipos de ataques;

**passível de avaliação estatística:** deve ser possível determinar facilmente quais fatores influenciam a métrica, e como;

**privada:** as opiniões emitidas não devem ser visíveis a todos;

**variação suave:** o valor de reputação não deve sofrer grandes variações como conseqüência de um voto;

**compreensível:** os usuários do sistema devem entender de forma fácil como um voto influencia o valor de reputação;

**verificável:** deve ser possível verificar um valor questionável com dados que comprovem a sua veracidade.

Algumas destas características são bastante importantes para o funcionamento do sistema. Considera-se que a métrica deve ter uma subida lenta (variação suave), mas uma descida rápida, de forma que seja difícil criar uma boa reputação, mas que seja fácil perder em caso de mudança de comportamento (priorizar interações recentes).

Além disso, devido à facilidade de troca de identidade, é recomendado que os nós entrem na rede com reputação baixa [22]. Tal medida se dá devido ao fato de ser impossível determinar se um nó entrante é realmente um usuário novo ou um malicioso reentrando na rede para obter uma reputação nova. Os nós novos são recebidos com desconfiança, devendo construir a reputação de baixo como uma forma de "prova de esforço", demonstrando boa vontade com a rede. Uma vez conquistado uma boa reputação, é mais interessante manter a identidade do que trocar e ter que reconquistá-la. Iniciando com uma reputação baixa, os nós maliciosos ou têm a sua capacidade de causar danos na rede limitada pela reputação baixa inicial, ou têm que colaborar com a rede para obter uma reputação maior e poder causar mais dano. Porém, no processo de aumento de reputação, ele estará colaborando com a rede.

### 4.4.4 Resposta

Assim como a métrica, a resposta dada pelo sistema de reputação deve ser de acordo com o objetivo.

Para o caso dos nós egoístas, deve-se impedir que estes baixem arquivos, ou reduzir a qualidade do serviço prestado. Desta forma, os nós serão incentivados a cooperarem, caso contrário não poderão obter os benefícios desejados da rede.

Já para os nós maliciosos, como eles não esperam nada da rede, restringir o uso da rede é ineficiente. Deve-se, portanto, identificá-los de modo que os nós honestos possam evitar estes nós.

## 4.5 Problemas em Sistemas de Reputação

O funcionamento do sistema de reputação pode ser prejudicado de duas formas. Os problemas enfrentados podem ser decorrentes do mau uso do sistema ou provir de um ataque. A diferença acontece, basicamente na intenção do nó em causar problemas. No primeiro caso, não há intenção do nó em causar dano na rede, que apenas adota uma

atitude egoísta. No segundo caso a intenção é a de realmente prejudicar o funcionamento da rede.

### 4.5.1 Dificuldades Enfrentadas

Os problemas aqui descritos ocorrem não por ataques maliciosos, mas por mau uso do sistema. Estes problemas podem aparecer devido a dificuldade de utilizar o sistema, mau *design*, conflitos decorrentes de diferença de interesses, entre outros. Jøsang [41] apresenta diversos destes problemas, para sistemas de reputação para serviços online, onde ambos os lados de uma transação (o provedor e o receptor do serviço) são avaliados. Apresentaremos aqui apenas os que também são relevantes para sistemas de reputação.

**Pouco incentivo para fornecer avaliações:** as avaliações devem ser feitas após a obtenção dos arquivos e a verificação da autenticidade, sendo que este último é feito fora da rede. Isso pode fazer com que muitos usuários não realizem a avaliação do download. Essa falta de retorno ao sistema pode diminuir a precisão das informações. Outro motivo que leva os usuários a não realizarem avaliações é a falta de benefício direto para si. Se todos realizarem as avaliações, então todos ganhariam, mas ninguém ganha diretamente com a sua própria avaliação. Xiong em [49] incentiva a realização de avaliações embutindo na métrica um componente relacionado à porcentagem de downloads avaliados, de forma que uma taxa maior de avaliações gera um aumento na reputação no nó;

**avaliações injustas:** não temos como saber com certeza absoluta se uma avaliação dada por um nó é sincera. Nós maliciosos podem dar opiniões erradas intencionalmente, com o objetivo de diminuir a reputação de um nó honesto. Algumas soluções são possíveis para minimizar este problema. Uma delas é utilizar a reputação do nó como peso para ajustar a opinião do nó. Opiniões fornecidas por um nó com reputação baixa, portanto, teriam um peso menor na avaliação final.

### 4.5.2 Ataques

Ataques são tentativas intencionais de subverter o sistema. Existem alguns ataques que são comuns a diversos tipos de sistemas de reputação, pois exploram características comuns a estes sistemas. Além dos problemas de identidade previamente discutidos, alguns problemas comuns são: nós traidores, inflação de reputação (*ballot stuffing*) e front nodes.

### Traidores

Este tipo de ataque consiste em um nó malicioso se comportar bem, até obter uma reputação alta. De posse de uma boa reputação, o nó muda de comportamento e passa a enviar arquivos corrompidos para a rede. Quando a reputação diminui novamente, o nó muda de identidade, ou recomeça o processo. Uma outra possibilidade é o roubo da identidade de um nó honesto.

Este tipo de ataque é possível devido ao fato de se utilizar as interações anteriores para definir a índole do nó, logo um nó pode se passar por honesto simplesmente agindo como tal. Por ser algo inerente ao funcionamento, não há como evitar o aparecimento de traidores. As soluções usualmente adotadas envolvem reduzir o valor de reputação rapidamente, de forma que mesmo alguém com boa reputação não consiga enviar muitos arquivos maliciosos. Duas abordagens encontradas na literatura são: calcular a reputação utilizando somente as últimas iterações (técnica conhecida como *aging*), ou diminuir a reputação mais rapidamente que a taxa de aumento.

### Inflação da Reputação

Inflação da reputação (também chamado de *ballot stuffing* ou *shilling* [18]) significa aumentar a reputação de um arquivo (ou nó) artificialmente. Isso é feito adicionando no sistema opiniões sobre transações que não ocorreram. Este ataque se aproveita da facilidade de falsificar a ocorrência de uma interação para inflar a reputação. Analogamente, é possível destruir a reputação de um nó, enviando várias opiniões negativas.

Uma forma de lidar com este problema é através da prova de transação [27]. Um voto só é válido se for apresentada uma prova de transação, assinada pelo nó avaliado. Na proposta apresentada no RCertP [27], a prova de transação para o fornecedor do arquivo é a requisição do mesmo, enquanto para o requisitante a prova de transação é a mensagem de resposta da requisição acima, contendo o *timestamp* da transação, para evitar que uma mesma mensagem seja utilizada como prova para mais de um voto.

A prova de transação evita a maioria dos ataques de inflação de reputação. O único que se mantêm efetivo é o ataque no qual diversos nós maliciosos diferentes interagem com o nó vítima, conseguem uma prova de transação, para finalmente emitirem a avaliação. Mas ainda assim, é um procedimento mais custoso.

### Front Nodes

Um dos ataques mais elaborados e eficientes contra sistemas de reputação que funcionam por redes de confiança, ele consiste em um grupo de atacantes maliciosos agindo em dois grupos: um fornece arquivos bons, de modo a ganhar boa reputação; este grupo então indica os nós do outro grupo como confiável, que aproveita as boas indicações para

fornecer arquivos corrompidos. Os nós que fornecem arquivos corrompidos passam a ter, após certo tempo, avaliações negativas, mas se o número de avaliações positivas do outro grupo de nós maliciosos forem maiores então pode demorar para que o sistema reconheça os nós/arquivos como maliciosos.

Este ataque visa especificamente os sistemas de reputação com redes de confiança, sendo indiferente nos outros. O motivo disso é que ele tem como objetivo subverter justamente o esquema de funcionamento das redes de confiança, criando artificialmente um grupo de nós com boa reputação para inflar a reputação de um nó malicioso.

## 4.6 Conclusões

Os sistemas de reputação lidam com o grave problema de arquivos ilegítimos nas redes P2P, que é um dos ataques mais comuns devido à facilidade de realização é ao dano causado. A bibliografia existente utiliza-se de conceitos econômicos para adotar estratégias efetivas para a solução dos problemas enfrentados.

Para um melhor estudo dos sistemas de reputação, eles foram divididos em diversos componentes. Estes componentes foram descritos em termos de características e problemas potenciais, e soluções adotadas para cada situação.

Por também serem expostos junto com as redes P2P que eles protegem, os sistemas de reputação também podem sofrer com ataques e mau uso. Os problemas mais comuns foram descritos, junto com as soluções usualmente adotadas.

# Capítulo 5

# Survey em Sistemas de Reputação Para Redes Peer-to-Peer

Serão estudados neste capítulo alguns dos sistemas de reputação para P2P existentes na literatura, com as soluções adotadas para os aspectos discutidos no capítulo anterior. Estes sistemas serviram de base para a definição das características dos sistemas de reputação, tanto diretamente, quanto através dos surveys existentes [41, 40, 6].

## 5.1 Marti [30]

Este sistema de reputação tem como objetivo combater os nós maliciosos. Entende-se como nós maliciosos os nós que entram no sistema com o objetivo de corromper a confiança e a funcionalidade da rede introduzindo arquivos ilegítimos. O sistema não visa lidar, por exemplo, com os nós egoístas, que utilizam a rede sem contribuir com recursos.

O sistema proposto é implementado sobre o Gnutella. Cada nó armazena suas opiniões sobre os nós com quem ele interagiu. A métrica utilizada deve ser escalar, para facilitar a comparação, sendo utilizada a taxa de arquivos autênticos que o nó forneceu sobre o número total de arquivos. Baseado nessa métrica, a seleção dos nós pode ser feita de duas formas:

**Select Best:** escolhe o nó com a maior avaliação. Corre o risco de sobrecarregar um nó, sem dar chances para outros nós ganharem reputação;

**Weighted Select Best:** cada nó que possui o arquivo desejado possui uma probabilidade de ser escolhido proporcional ao seu valor de reputação.

Além de utilizar somente a própria métrica, pode-se também levar em consideração as opiniões de outros nós, compartilhando os valores de reputação. A avaliação compartilhada funciona como uma extensão do modelo local. Nesta, após o recebimento dos

resultados da busca pelo arquivo, o nó requisitante entra em contato com outros nós para obter as avaliações sobre os que apresentaram resultado. O cálculo da avaliação final possui um componente com a nota local e outro com a dos outros nós, com um peso indicando a importância de cada parte. A parte compartilhada é calculada como uma média ponderada, utilizando como peso a opinião que o próprio nó tem dos avaliantes. A avaliação do no $r$, feita pelo nó $q$, levando em consideração as opiniões dos nós $v \in Q$ é:

$$\rho_r = (1 - \omega_Q)R(q, r) + \omega_Q \frac{\sum_{v \in Q} R(q, v)R(v, r)}{\sum_{v \in Q} R(q, v)},$$

onde $R(a, b)$ é a reputação de $b$ segundo $a$, e $\omega_q$ é o peso dado às avaliações dos outros nós. A escolha dos nós que farão parte do conjunto $Q$ pode ser feita de duas formas: escolhendo entre os vizinhos ou entre os conhecidos (nós com quem houveram interações) . Experimentos feitos mostram que a utilização de nós conhecidos diminui o número de arquivos inautênticos baixados, o que é esperado, já que o cálculo leva em consideração a opinião acerca dos nós $v$ que estão fornecendo opiniões sobre $r$. Para melhorar o desempenho é utilizado um *cache* dos nós conhecidos com maiores reputações.

Quando o arquivo é pouco popular, pode ocorrer de nenhuma cópia autêntica ser encontrada. Nestes casos, o algoritmo selecionaria a cópia inautêntica com a maior reputação, mesmo que esta não seja alta. Para evitar estes casos, é definido um limite para a reputação, $\rho_T$, abaixo do qual as respostas são desconsideradas, por serem pouco confiáveis.

Em relação à identidade, são considerados dois modelos, baseados nos apresentados em [29]: um modelo equivalente à existência de um servidor central, onde troca de identidade é difícil; e um onde a troca de identidade é mais fácil, mas mais próximo da realidade. Não são especificados, porém, detalhes sobre tais esquemas, provavelmente sendo descritos apenas para a comparação nos experimentos.

Para a realização dos experimentos, os nós maliciosos foram divididos em 3 categorias:

- os nós apenas propagam arquivos inválidos, mas dando opiniões sinceras;
- além de espalhar arquivos falsos, dão opinião ruim a todos os nós;
- Nós maliciosos agem em grupo, dando avaliações boas para os do conluio, e ruins para os outros. Também é considerado um ataque mais elaborado no qual um subgrupo fornece arquivos bons, ganhando uma boa reputação, e dando uma avaliação boa para os nós maliciosos fora do subgrupo, dando a estes uma avaliação positiva proveniente de nós com boa reputação. Este tipo de ataque é chamado de *front nodes*.

As métricas utilizadas para a validação do método foram: a taxa de verificação, definido como o número de verificações de autenticidade por download bem-sucedido; carga do sistema, definido como o número de downloads de arquivo pelo número de buscas na qual houve algum resultado válido; e o tráfego de mensagens do sistema de reputação, medido como uma porcentagem do número de mensagens enviadas normalmente num flood. Nas duas primeiras métricas, quanto menor o valor, melhor, sendo 1 o valor mínimo.

Os experimentos mostram que a melhor estratégia é a de compartilhamento de opiniões com amigos. No pior caso, com troca de identidade e conluio, a taxa de verificação foi 4 verificações por busca, contra 28 do caso sem uso de reputação. Porém, o sistema tem uma perda de eficiência quando utilizados *front nodes*, por um fator de 3.

## 5.2 Aberer [1]

O sistema proposto por Aberer visa combater os nós maliciosos, provendo acesso rápido às informações de reputação. Para isso, ele utiliza a infra-estrutura de DHT proporcionada pelas redes estruturadas para armazenar as informações de reputação. Apesar de poder ser qualquer rede estruturada, rede escolhida foi a P-Grid, do mesmo autor.

O armazenamento é feito seguindo as regras da DHT: as informações de reputação sobre um nó $x$ são armazenadas no nó responsável pela chave $hash(x)$. Para evitar que um nó corrompa as reputações que ele armazena, a reputação é replicada em diversos nós, prevalecendo a reputação que aparecer mais vezes.

As avaliações dos nós, dada a reputação, é binária: ou o nó é considerado confiável ou não. Armazena-se sobre os nós apenas as reclamações, baseado da suposição de que o comportamento malicioso seja exceção. O cálculo do valor de reputação de um determinado nó é feito multiplicando o número de reclamações recebidas pelo número de reclamações feitas. Para facilitar a busca destes valores, as reclamações são armazenadas indexadas tanto pelo criticado quanto pelo crítico.

Para decidir se um determinado nó é confiável ou não, inicialmente procura-se na rede por diversas testemunhas. Como a estrutura da rede P-Grid não é uniforme, as diversas opiniões sobre o nó desejado podem ser encontradas com freqüências diferentes. Por conta disso os valores são normalizados levando em consideração a freqüência com a qual os valores foram encontrados nas buscas. A decisão ou não pela confiança no nó é feita de forma heurística, comparando este valor com uma média aproximada destes valores considerando diversos nós.

A normalização das reclamações pela freqüência é feita da seguinte forma: seja $cr_i(q)$ o número de reclamações feita pela testemunha $i$ sobre o nó $q$, e $f_i$ a freqüência com a qual esta opinião é encontrada (em número de de respostas nas $s$ buscas). Então

$$cr_i^{norm}(q) = cr_i(q)(1 - (\frac{s - f_i}{s})^s.$$

O mesmo cálculo é feito sobre as reclamações realizadas ($cf_i(q)$). Com base nestes valores, determina-se a confiabilidade comparando com os valores médios da rede. Se o valor calculado for muito maior que a média, então o nó é considerado não confiável.

Os experimentos feitos visaram medir a eficiência do método avaliando o número de falsos positivos. Foram feitas simulações, com uma certa porcentagem dos nós maliciosos. O resultado do algoritmo era comparado com a informação da construção da rede. Os resultados mostraram uma grande taxa de acerto, com identidade estática (não é considerada troca de identidades nem entrada e saída de nós).

## 5.3 P2PRep [12]

P2PRep é um sistema desenvolvido para funcionar em redes Gnutella, para gerência de reputação com o objetivo de diminuir o número de downloads de arquivos corrompidos.

Neste sistema, cada nó armazena as suas opiniões sobre os nós com quem interagiu. As opiniões dos outros nós são levadas em consideração, aumentando a quantidade de informações disponível para a tomada de decisão.

O sistema utiliza o conceito de identificadores opacos com o objetivo de preservar a identidade real do usuário. Este identificador no P2PRep é o hash da chave pública do nó. Este identificador é denominado *servent_id*.

O sistema de reputação não procura desenvolver um método para evitar a troca de identidade. Ele é feito, porém, de forma a torná-la desinteressante, penalizando os usuários novos. Espera-se com isso fazer com que aqueles que tiverem uma reputação boa não queriam perdê-la, e quem tiver má reputação não tenha um grande ganho trocando de identidade.

Ao fazer uma busca por um determinado arquivo, diversos nós respondem com informações sobre o arquivo e o id do nó. O método básico funciona assim: após a recepção das respostas, o nó requisitante envia mensagens com o conjunto de ids dos nós que responderam às buscas. Nesta mensagem é enviada uma chave pública temporária específica para esta transação para garantir a integridade e a confidencialidade da opinião, sem que o nó requisitante seja identificado. Depois de coletada as opiniões de reputação, pode-se consultar diretamente os nós que deram as opiniões para evitar o uso de ip's falsos, confirmando as opiniões através de mensagens *TrueVote*. Os nós que forneceram opinião não precisam se identificar neste método.

O método acima descrito permite que um nó vote em nome do outro, justamente por não verificar a identidade do opinador. Uma melhoria no método é apresentada na qual

é adicionada uma verificação semelhante ao *TrueVote*, com a finalidade de se verificar a identidade do nó. Para isso às mensagens com a opinião do nó são adicionadas informações acerca da identidade do nó: IP, porta e o *servent_id*, o identificador do nó. A identidade do nó é verificada através de uma mensagem *AreYou*, na qual se confirma se o *servent_id* pertence realmente àquele endereço (IP e porta). Além desta verificação, mantém-se informações sobre a credibilidade dos nós, medindo a confiança das opiniões.

Mesmo o método melhorado não impede que um atacante crie diversos nós com IPs válidos para subverter o sistema. Para evitar que um agente malicioso utilize um ataque de múltiplas identidades a partir de um IP (ou uma rede), o sistema propõe um esquema de voto por cluster. Este esquema organiza os votos por proximidade de IP, por exemplo, subredes do tipo C. Múltiplos votos de uma região próxima são amortizados pelo número de votos, de forma que se uma subrede emitir muitos votos, esses votos terão um peso menor no cálculo final.

As avaliações que cada nó armazena consistem em um número de avaliações positivas e negativas relacionada às experiências com os nós. O cálculo para a escolha do nó é feito da forma preferida pelo nó requisitante, não existindo um algoritmo geral.

## 5.4 XRep [16]

O XRep é uma extensão do sistema P2PRep descrito na seção 5.3. Ele herda todas as características presentes no sistema base, adicionando informações de reputação para os arquivos compartilhados. Os arquivos são identificados pelo *digest* do conteúdo, calculado utilizando uma função de *hash* criptográfico, por exemplo.

A verificação da confiabilidade de um recurso pode ser feita verificando a reputação do nó e/ou do recurso, num protocolo de acesso aos dados parecido com o descrito no P2PRep.

A reputação por arquivos, junto com a reputação do nó e as outras características destes dois sistemas de reputação provêem algumas propriedades interessantes. Algumas também valem para o P2PRep [12], apesar de só terem sido discutidas em [16].

Ambos os sistemas são capazes de resistir a certos tipos de ataques de *pseudospoofing* (também chamado de *Sybil*), ataque no qual um nó simula uma grande quantidade de nós. Esse ataque é neutralizado com a utilização da votação por cluster (ver [12]), e pela reputação por recurso, onde, mesmo que as diversas identidades falsas criem uma falsa boa reputação para si mesmas, o arquivo adulterado pode ser detectado pela sua má reputação.

Outro ataque evitado, apenas pelo XRep, é o de roubo de identidade. Este ataque consiste em roubar a identidade de um nó com boa reputação, e responder mensagens de busca com duas respostas, utilizando o próprio id em uma delas e o roubado em outra.

Espera-se que, ao avaliar a reputação dos nós que fornecem o arquivo, possivelmente sem informação de reputação, a reputação da identidade roubada dê ao arquivo adulterado uma boa reputação. Este ataque é evitado num passo do algoritmo que verifica se algum nó com boa reputação realmente possui o arquivo.

O terceiro ataque descrito é o de *shilling*, ou *ballot stuffing*. Este ataque não é detectado eficientemente pelo XRep, exceto se houver diversas avaliações de nós honestos para ofuscar as maliciosas. Este ataque é bastante parecido como o de *front nodes*, onde alguns nós fornecem arquivos bons para ganhar uma reputação positiva, posteriormente utilizando esta boa avaliação para indicar nós maliciosos e induzir outros a baixarem arquivos adulterados.

## 5.5 RCertP [27, 34]

O RCertP é um protocolo desenvolvido com a finalidade de fornecer armazenamento e acesso eficiente de avaliações de reputação. Não é especificado uma métrica para a avaliação dos nós, sendo descrito somente o protocolo.

A abordagem dada por este protocolo é bem diferente das usualmente adotadas, por delegar aos nós a responsabilidade de armazenar as opiniões sobre si. Esta decisão, embora facilite a busca por informações sobre o nó (estão todas com o avaliado), cria um grande problema de segurança, pois é de interesse direto do nó adulterar as opiniões que ele armazena. Este problema é resolvido através do uso de uma infra-estrutura de chave pública. Cada nó possui um par de chaves (pública e privada), e esta chave é utilizada para assinar a opinião emitida pelo nó. Assim evita-se que o nó mude as opiniões emitidas sobre si, ou adicione opiniões falsas. Outra característica interessante deste protocolo é o uso de prova de transações, que pode ser usado para garantir que os votos sejam dados apenas após uma transação, evitando *ballot stuffing*.

O armazenamento das opiniões é feito numa estrutura denominada RCert, que armazena as opiniões dos diversos nós em estruturas denominadas RCertUnit. Cada RCertUnit contém a assinatura do avaliador abrangendo todo o certificado, de forma que basta conferir a última assinatura para se certificar da validade do certificado todo.

O protocolo básico (RCertP) consiste numa negociação em 6 passos com o nó. Inicialmente procura-se candidatos para o fornecimento de recursos utilizando o protocolo da rede P2P. Os nós respondem enviando o certificado com as avaliações dos outros nós. De posse dos certificados, o requisitante escolhe um dos nós candidatos enviando um ACK assinado pela sua chave privada. Este ACK funciona como uma prova de transação para o nó oferecendo o serviço. Este nó, para confirmar a aceitação do ACK, retorna com um timestamp assinado, que serve de prova de transação para o requisitante. Finalmente a transação é realizada, e o requisitante envia o certificado RCert atualizado com a sua

opinião.

Este protocolo, porém, tem uma falha: ele permite que o nó avaliado descarte o certificado novo, caso ele possua uma avaliação negativa. Isso é possível porque não existe nenhuma forma de se determinar se o certificado entregue no início do protocolo é realmente o último emitido. Devido a este problema, é proposto um protocolo melhorado, chamado RCertPX.

A diferença entre os protocolos ocorre na verificação do certificado (RCert). No protocolo melhorado, a verificação se dá contatando o último nó que realizou uma transação com o nó, obtendo dele uma estrutura contendo o último timestamp, o status dele (válido/revogado) e a identidade do nó que revogou. Baseados nestes dados, o nó compara os timestamps, e confirma se este é realmente o último certificado emitido para o fornecedor. Assim, se o nó tentar utilizar algum certificado antigo (com menos avaliações negativas, por exemplo), a verificação do timestamp vai dar este certificado como revogado. No fim do protocolo, após o envio do novo certificado, o nó requisitante envia uma mensagem de revogação para último avaliador.

Ainda assim, o avaliado pode apresentar um certificado vazio. Isto é equivalente a adquirir uma nova identidade, o que o protocolo não impede, logo não é uma ameaça nova. Além disso, nós sem reputação podem ser considerados de pouca confiança, logo apagar o certificado não é um problema grave.

No protocolo RCertPX, a avaliação do certificado pode ser feita utilizando os $n$ últimos avaliadores para aumentar a disponibilidade.

## 5.6 EigenTrust [24]

O EigenTrust é um sistema de reputação com o objetivo de detectar nós maliciosos através do uso de redes de confiança. Para tal, ele monta uma matriz com as opiniões de todos os nós sobre os outros, e calcula a reputação dos nós como o autovetor desta matriz. Este cálculo é feito de forma distribuída por todos os nós.

Neste sistema, o armazenamento da reputação local (opinião do nó) é feito no nó que emitiu a opinião. A reputação global (o resultado do cálculo do autovetor) é armazenado numa DHT, por eficiência de acesso e segurança. Complementando a questão de segurança, utiliza-se replicação para evitar que os valores de reputação que ficarem sob a responsabilidade de um nó malicioso sejam perdidas.

Não é definido um intervalo para os valores de reputação. Cada nó pode utilizar o intervalo que desejar para a atribuição de notas para o serviço. Para que se torne possível realizar comparações, a métrica utilizada é normalizada dividindo-a pela soma de todas as notas dadas pelo nó. Dessa forma, todas as avaliações ficam no intervalo $[0, 1]$. Feita a normalização, os valores globais podem ser calculados sem a necessidade de

normalização em cada iteração, além de se ter um modelo estatístico elegante. Porém, devido à normalização, os valores perdem sentido absoluto, ou seja, se dois valores são iguais não sabemos se ambos são igualmente confiáveis ou igualmente maliciosos. Esta relatividade das notas não atrapalha o funcionamento do sistema, mas o torna menos intuitivo.

A agregação da opinião sobre um nó $j$ é feita levando em consideração a opinião local dos nós conhecidos, ajustada pela confiança nestes. Ou seja: $t_{ij} = \sum_k c_{ik}c_{kj}$, onde $t_{ij}$ é a reputação agregada do nó $j$ de acordo com a visão de $i$, $c_{ij}$ é a opinião de $i$ sobre $j$, e $k$ são os conhecidos utilizados para obter opiniões sobre $j$.

Desconsiderando o fator distribuído do algoritmo, podemos considerar tais opiniões como parte de uma matriz $C$ de elementos $[c_{ij}]$, de avaliações feitas pelo nó $i$ sobre o nó $j$. O cálculo do valor global de reputação pode ser visto, então, como uma multiplicação de matrizes: o vetor $\overrightarrow{t_i}$ das opiniões de i sobre os outros nós é calculado como $\overrightarrow{t_i} = C^T c_i$. Para levar em consideração as opiniões dos amigos dos amigos, teríamos $\overrightarrow{t_i} = (C^T)^2 c_i$. Considerando a rede toda, teremos $\overrightarrow{t_i} = (C^T)^n c_i$, algo caro para se calcular, dado a magnitude de $n$. Porém, segundo o artigo, o vetor convergido $\overrightarrow{t_i}$ é o mesmo independente de $i$, sendo este o *autovetor próprio esquerdo de C (left principal eigenvector).* Logo, o cálculo da reputação global dos nós pode ser realizado calculando-se o autovetor de forma distribuída pela rede, e distribuindo o resultado, de forma que todos os nós da rede tenham o mesmo vetor $\overrightarrow{t}$.

Uma otimização para o algoritmo é a utilização é um vetor $\overrightarrow{p}$ de nós que são previamente confiáveis. Exemplos destes nós podem ser os desenvolvedores da rede, e os *early adopters*. Este vetor pode ser utilizado no lugar de $c_i$ para acelerar a convergência de $\overrightarrow{t}$. Outro uso previsto deste vetor é para lidar com conluio de nós maliciosos, "realimentando" este vetor a cada passo do algoritmo.

Por medidas de segurança, um nó não calcula a parte do vetor responsável pelas opiniões sobre si mesmo, e cada linha da matriz é calculada por mais de um nó. A consulta aos dados de reputação é feita consultando os diversos nós responsáveis, escolhendo o valor de reputação que possuir maioria para evitar dados corrompidos (caso alguns dos portadores do valor de reputação faça parte de um conluio para melhorar a reputação do pesquisado).

Os valores de reputação são atribuídos aos seus gerenciadores utilizando alguma rede DHT, como CAN ou Chord. A localização das informações de um nó é calculada como o hash de um identificador único, como o IP e a porta TCP. Os gerenciadores de reputação diferentes são escolhidos utilizando-se de funções hash distintas. Esta distribuição por hashes possui como vantagens o anonimato acerca do nó do qual se calcula o valor de reputação. A DHT pode ser construída apenas para o EigenTrust, caso a rede original não suporte (nos experimentos supõe-se que a rede para download dos arquivos é a Gnutella.

Os experimentos realizados mostraram que o sistema forneceu uma melhora significativa mesmo com 70% de nós maliciosos. Foram considerados os seguintes tipos de ataque: A - sempre fornece arquivos falsos e dá notas boas a nós ruins e vice-versa; B - sempre fornece arquivos ruins, e dá notas boas aos nós do seu conluio; C - fornece arquivos ruins $f\%$ das vezes; D - comportam-se como nós bons, mas dão notas boas a nós do tipo B (*front nodes*). Os piores resultados foram contra o tipo C com $f = 50\%$, e o tipo D. Mas mesmo nestes casos, a proporção de arquivos autênticos por arquivos falsos fornecidos por nós maliciosos foi grande. O EigenTrust não é capaz, porém, de resistir a ataques *Sybil*.

## 5.7 Secure Score Management [5]

Protocolo para gerenciamento seguro de valores (scores) em P2P, onde o valor armazenado pode ser de reputação, micropagamentos ou qualquer informação que se queira armazenar sobre um nó de forma segura. Considera-se que este valor deva ser protegido de todos os nós, inclusive do nó armazenando as informações.

O armazenamento dos valores de um nó $N$ é feito num nó $M$, determinado pelo hash do identificador de $N$. O uso do hash provê uma aleatoriedade na escolha do nó responsável, o que diminui as chances de cair num nó que esteja em conluio com o dono das informações ($N$).

O protocolo tem a participação de 4 nós: o requisitante ($N_q$), o requisitado ($N_r$) e os responsáveis pelos valores dos mesmos ($M_q$ e $M_r$). A descrição do funcionamento do protocolo é descrita no relatório técnico, e envolve enviar a resposta através dos de $M_q$ e $M_r$. Uma grande desvantagem do protocolo é o *overhead* causado pelas adições de mensagens em comparação com o acesso direto ao serviço. Enquanto o acesso normal levaria duas mensagens (a requisição e a resposta), o protocolo descrito precisa de 7 mensagens, algumas delas envolvendo a busca por um nó dado o identificador, o que leva $O(log(n))$.

O relatório mostra que todos os passos apresentados mostram-se necessários, mas somente para o caso no qual o requisitante precise pagar pelo serviço, como no caso de micropagamentos. Para outros casos, como o de reputação, alguns passos podem ser removidos, evitando parte do *overhead*.

## 5.8 Feldman [21]

O sistema de reputação proposto neste artigo se propõe a combater os free-riders, nós que não compartilham arquivos. Estes nós são considerados ruins porque uma grande quantidade deles pode comprometer a rede, visto que isso fará com que os downloads se

concentrem em uma quantidade pequena de nós, causando nestes uma grande carga.

A modelagem do sistema é feita levando em consideração as seguintes características das redes P2P:

**Dilema social:** cooperação possui um benefício para todos, mas se aproveitar dos outros têm um benefício próprio maior. Mas se ninguém cooperar os ganhos da rede são pequenos ou nulos;

**transações assimétricas:** os serviços fornecidos e recebidos por um nó não precisam ser necessariamente para o mesmo nó. Pode-se receber serviço de um e prover para outro. O sistema deve ser capaz de lidar com este tipo de assincronia;

**não oferecimento de serviços indetectável:** Não é possível saber se alguém tem um determinado arquivo e simplesmente não está respondendo às buscas;

**população dinâmica:** nós podem entrar e sair quando quiserem.

Com o objetivo de facilitar a discussão, o artigo denomina como servidores os nós que, numa determinada transação, estão provendo um serviço, e como clientes os nós que estão recebendo.

O sistema foi projetado com as seguintes características:

**Discriminação da seleção de servidor:** cada nó deve manter um histórico de ações que outros nós fizeram para ou receberam dele, e utilizar esta informação para a escolha de qual nó utilizar como servidor. A idéia é utilizar um servidor já confiável, ou dar chance aos que utilizaram seus serviços de retribuírem. Esta utilização de nós com quem já houveram interações, segundo [4], estimula a cooperação, por aumentar a possibilidade de interações futuras, criando o que é chamado de *sombra do futuro (shadow of the future)*. Para evitar que os nós se confinem em pequenos grupos, ignorando o resto da rede, assume-se uma probabilidade de 0,3 de um nó desconhecido contribuir. Esta característica lida bem com o problema de grandes populações, mas não lida bem com grandes taxas de saída de nós;

**compartilhamento de histórico:** visa aumentar a eficiência do uso do histórico através do compartilhamento das opiniões sobre os nós, obtendo dessa forma opiniões sobre nós com os quais ainda não haviam tido interações. As opiniões compartilhadas aumentam a quantidade de informações sobre os nós, melhorando a confiança na escolha dos servidores e permitindo transações assíncronas (retribuir serviço prestado a outro nó). O compartilhamento de históricos provê uma melhora na eficiência para grandes populações (utilizado junto com a seleção de servidor), mas o grande benefício se dá com o aumento da taxa de saída de nós, mantendo a utilidade do

sistema alta até a taxa de 10% de saída, contra 1% da estratégia privada. Porém, algumas dificuldades aparecem: é necessário algum mecanismo para acesso eficiente dos dados de outros nós, como uma DHT, e deve-se preocupar também com a possibilidade de que alguns nós juntem-se para inflar as notas positivas uns dos outros;

**reputação baseada no algoritmo de fluxo máximo:** evita que um grupo de nós formem um conluio, sabotando o sistema de reputação compartilhado fornecendo notas altas para membros maliciosos do grupo. Uma forma de se lidar com este problema é a verificação da reputação do nó que esta dando a avaliação, e a avaliação de quem deu esta avaliação, etc. num caminho de confiança entre o nó cliente e o servidor. O uso do fluxo máximo realiza este cálculo, com a vantagem de agregar a informação de diversos caminhos diferentes entre os dois nós. Como o algoritmo de fluxo máximo é caro ($O(V)$, onde $V$ é o número de nós da rede), utiliza-se um algoritmo aproximado, mais rápido. Comparações com os mecanismos anteriores mostram um aumento na eficiência até um certo número de nós na rede. Esta abordagem também tem o problema de nós que tem comportamento bom mas dão más avaliações (*front nodes*)

**política adaptativa para estranhos (recém entrados no sistema):** procura combater o *whitewashing* tornando a reentrada no sistema algo custoso. Isso é feito de forma adaptativa, respondendo aos novos usuários de acordo com o comportamento dos últimos estranhos. O cálculo é feito considerando o número de contribuições e de utilizações dos novatos. A proporção de serviços consumidos sobre os fornecidos provê a taxa de confiança nos novatos;

**histórico de curto prazo:** procura combater os traidores, detectando rapidamente um nó que passou a se comportar mal. Para isso, utiliza-se apenas uma memória de curto prazo.

A métrica utilizada para a escolha sobre cooperar ou não é a *generosidade normalizada*. Define-se como generosidade de um nó $i$ a taxa de serviço prestado sobre a de serviços consumidos ($g(i) = \frac{p_i}{c_i}$). Esta métrica, por si só poderia ser utilizada, mas algumas políticas (como a de negar serviços a estranhos) podem fazer com que nós cooperativos neguem serviço um ao outro. Problemas deste tipo são evitados normalizando o nível de generosidade do nó provedor do serviço ($i$) pelo próprio nível ($j$): ($g_j(i) = \frac{g(i)}{g(j)}$).

## 5.9 Gupta [23]

Gupta em [23] propõe um sistema de reputação, capaz de lidar tanto com crédito-apenas quanto com crédito e débito (reputação e microcrédito, respectivamente). O sistema é desenvolvido sobre o Gnutella, podendo ser adaptado para outros sistemas, segundo o autor.

Assim como em [27, 34], o armazenamento dos créditos é feito no nó avaliado. Porém, neste sistema, o cálculo dos créditos é feito num servidor central. Para evitar que o nó adultere o valor de seus créditos (aumentando o número de créditos, por exemplo), estes são assinados pelo nó central. Pelo fato dos créditos serem armazenados fora do servidor central, ele não é necessário para as transações, sendo contatado apenas periodicamente para a consolidação dos créditos, diminuindo os problemas causados por sua falha.

Os nós, ao realizarem operações que gerem créditos, contatam o servidor central, fornecendo a prova de realização da operação para o recebimento dos créditos. Essa troca pode acontecer após certo número de operações, para reduzir o tráfego. O servidor, após receber as provas, calcula a quantidade de crédito devida e retorna o número de créditos do nó atualizado, num certificado assinado. A forma de se lidar com os débitos deve ser diferente da forma de se lidar com os créditos, pois caso o servidor envie um certificado com o número de créditos menor que o atualmente possuído por um nó este irá descartá-lo, mantendo o antigo. Ao receber uma requisição de débito, o servidor mantêm o estado do débito para o nó, deduzindo-o do próximo crédito a ser contabilizado.

As provas de transação são geradas através de mensagens assinadas, assim como em [27, 34]. Cada nó gera um conjunto de chaves pública e privada, sendo o hash da chave pública o identificador do nó. O nó pode mudar de identidade simplesmente criando novas chaves. Argumenta-se no artigo, porém, que esta mudança não é vantajosa para o nó, visto que isto fará com que ele perca todos os seus créditos, ficando com a quantidade mínima de créditos, ou seja, zero.

São propostos dois métodos de utilização de métrica: débito-crédito, e crédito-apenas. No primeiro, um nó ganha créditos por upload, e os gasta realizando downloads. Estes valores são proporcionais à transferência realizada. No segundo os downloads não são debitados, e a reputação serve apenas para indicar nós confiáveis.

A métrica é composta de dois componentes: um referente ao comportamento e outro à capacidade do nó. A capacidade também é levada em consideração pois nós com mais banda disponível proporcionam uma experiência melhor (download mais rápido).

O cálculo da reputação de um nó é feita levando em consideração três parâmetros: banda disponível ($b$, número real), tamanho do arquivo compartilhado ($f$, inteiro) e tempo online/compartilhando ($t$, em horas). Estes parâmetros são convertidos em valores de reputação com a utilização de um *fator de conversão*. Os parâmetros são convertidos

em créditos sendo divididos por estes fatores. Assim, nós que disponibilizam arquivos grandes e/ou bastante banda ganham mais pontos de reputação em comparação com quem disponibiliza apenas arquivos pequenos, compensando pelo custo maior gasto a favor da rede. Isso é mais importante para o sistema de crédito-débito.

No modelo débito-crédito, a reputação é calculada pela seguinte fórmula:

$$ReputationScore_k = a * QR + \sum_l b * UC_l - \sum_m c * DC_m + d * SC$$

onde $QR$ é o número de pontos ganhos por cada uma das $a$ mensagens de busca respondidas, $UC_l$ o crédito ganho por cada um dos $b$ uploads feitos do arquivo $l$, $DC_m$ os débitos feitos pelos $c$ downloads do arquivo $m$, e $SC$ os pontos ganhos por disponibilizar arquivos para compartilhamento por $d$ unidades de tempo calculadas de acordo com o fator de tempo acima descrito. Descreveremos agora como calcular os valores de crédito/débito utilizados.

O crédito referente as mensagens de busca respondidas ($QR$) é calculado pelo tamanho das mensagens de resposta.

As transferências de arquivos ($UC$ e $DC$) são computadas multiplicando o tamanho do arquivo transferido ($s$) pela banda utilizada ($bw$), ambos divididos pelos seus fatores: $\frac{s}{f} * \frac{bw}{b}$.

Por fim, o crédito de compartilhamento ($SC$) é calculado levando em consideração o tamanho dos arquivos compartilhados (dividido pelo fator $f$). Este crédito é custoso para se calcular de forma confiável.

No modelo crédito-apenas, retira-se o débito por download, e acrescenta-se um *timestamp* nos créditos para evitar que os nós acumulem uma grande reputação no início e posteriormente parem de compartilhar arquivos.

Adesão ao sistema de reputação é voluntário. O nó pode escolher não participar do sistema, para preservar o seu anonimato, visto que o sistema de reputação armazena informações sobre o nó — com quem o nó andou prestando/recebendo serviços). Caso um dos nós não esteja participando do sistema, é atribuído a ele a reputação de 0, a mais baixa possível, e não há atualização dos valores de crédito do nó participante.

O sistema prevê a possibilidade da rede permitir download simultâneo de diversas partes de nós diferentes. Neste caso, é dado a cada um dos nós créditos relativos ao pedaço fornecido.

## 5.10 Yu [50]

O mecanismo de reputação descrito neste artigo foi desenvolvido para aumentar a confiabilidade das opiniões em um sistema para troca de informações criado pelos autores.

Neste sistema é construído uma rede social, onde os participantes se dizem especialistas em uma determinada área. Cada usuário é representado na rede por um nó. Ao receber uma busca sobre determinado assunto, o nó mostra a busca sendo realizada para o seu usuário, que decide por enviar uma resposta, caso a pergunta esteja em sua área de conhecimento, ou envia um grupo de referências de outros usuários que possam ser capazes de responder à pergunta. Observe que a geração da resposta é feita manualmente pelo usuário, e não automaticamente pelo programa que implementa o nó.

Em relação ao sistema de reputação desenvolvido, os autores definem que ele deve lidar com 3 desafios: dar controle total aos usuários sobre quando revelar suas opiniões, ajudar um agente a achar um nó confiável mesmo que ele nunca tenha interagido com ele antes e acelerar a propagação de informações na rede. O sistema descrito lida com estas 3 características.

O sistema definido utiliza a rede social da rede base para definir a reputação dos nós com os quais ainda não houveram interações. Para se definir a reputação de um nó com o qual não houve interações, é utilizado a opinião de um vizinho deste nó, ajustada pela reputação do vizinho.

O valor de reputação é definido como um número real entre -1 e 1. Inicialmente os nós começam com 0 de reputação. Atualizações nos valores de reputação seguem uma regra relativamente complexa, levando em consideração se houve ou não um bom serviço prestado e o sinal da reputação anterior. Basicamente podemos caracterizar as alterações como sendo ajustadas por um valor $\alpha \geq 0$ caso o nó colabore, ou $\beta \leq 0$ caso negativo. É recomendado que $|\alpha| < |\beta|$, para que o aumento da reputação seja gradual, mas a queda seja rápida, fazendo com que os nós traidores sejam rapidamente identificados.

A definição sobre o nível de confiança de um nó é feito baseado em dois limites: um limite inferior para confiança ($\omega$), e um limite superior para a desconfiança ($\Omega$), sendo que $\omega \geq \Omega$. Entre os dois valores não se pode concluir nada acerca da confiança do nó.

A rede permite que se contate um nó a partir de um caminho de nós vizinhos. O cálculo da reputação do nó contatado segundo este caminho é feito utilizando a seguinte função: sejam dois nós vizinhos, de reputação $x$ e $y$. Definimos $x \otimes y = se(x \geq 0\, e\, y \geq 0)\; x \times y\; senão\; -|x \times y|$. Aplicando esta função para os nós seguidos do caminho, dois a dois, temos o valor da reputação desejado. O penúltimo nó da cadeia é denominado de *testemunha*, por possuir evidência direta do nó que se está calculando a reputação. Para que a opinião desta testemunha seja levada em consideração, é requerido que ela seja maior que 0. Caso exista mais de uma testemunha, combina-se as opiniões destas.

O para ajudar a detectar os nós que provêem mau serviço, o sistema determina que, ao receber um serviço ruim, além de atualizar a sua noção sobre o nó, a parte prejudicada pode enviar uma mensagem aos vizinhos informando do mal serviço recebido.

Os experimentos foram feitos apenas para verificar o comportamento da métrica, visto

que o artigo não especifica detalhes do funcionamento do protocolo em si. Foram realizados testes com diferentes valores de $\alpha$ e $\beta$, com o objetivo de mostrar como estes parâmetros interferem na variação da reputação de acordo com as proporções de cooperações por não-cooperações. Os resultados mostram que o cálculo da métrica com os parâmetros configurados de forma que $|\alpha| < |\beta|$ fornecem o comportamento desejado de subida lenta e queda rápida dos valores de reputação.

## 5.11 LOCKSS [39]

Este sistema de reputação aproveita um sistema base diferente dos demais para definir um sistema baseado em premissas diferentes das usualmente utilizadas.

O sistema base tem como objetivo o armazenamento de artigos em muito longo prazo, estilo biblioteca física. Cada nó neste sistema armazena os arquivos que ele quer preservado, utilizando as cópias dos outros nós apenas para checagem de erros e aumento de disponibilidade.

Devido à natureza do problema (armazenamento a muito longo prazo), foram escolhidos requisitos diferentes dos habitualmente utilizados em redes P2P:

**Limitar a taxa de operação:** não fazer nada mais rápido que o necessário. O objetivo é evitar a degradação por muito uso, fazendo com que um nó malicioso não tenha muito tempo para agir;

**assumir adversário poderoso:** não impor limites arbitrários aos adversários;

**não guardar segredos por muito tempo:** como o sistema prevê a manutenção dos dados por muito tempo, segredos como chaves privadas não resistiriam a ataques de adversários poderosos. Por isso evita-se guardar os segredos por mais que alguns dias;

**não depender da identidade dos nós:** devido à facilidade de falsificar identidade;

**evitar reputação de terceiros:** para evitar que opiniões corrompidas estraguem o seu armazenamento;

**não realizar acumulo de credito:** sem identidades fortes nem autoridade central, não podemos confiar nos créditos de operações anteriores;

**minimizar o número estados:** como os períodos de tempo são grandes, as informações na memória podem ser apagadas ou perdidas, logo quanto menos estados forem mantidos, e menor o tempo, melhor;

**fazer detecção de forma inerente:** utilizar alarmes para avisar os administradores de que algum ataque está ocorrendo, antes de que ele seja efetivo.

Como o sistema evita armazenar uma grande quantidade de estados, ele utiliza prova de esforço ao invés de utilizar valores de reputação. Neste sistema, um nó deve executar uma função limitada pela memória (*MBF - Memory Bound Function*) como uma forma de mostrar comprometimento com a operação, e votação por maioria. A prova de esforço torna custoso para um nó simular diversos nós, já que a resolução de diversas provas de esforço é muito custosa.

Cada nó mantém uma lista com um grupo de nós que ele conhece. Periodicamente, os nós fazem uma votação pra checar se os seus arquivos ainda são válidos, ou seja, não foram corrompidos por um ataque ou falha de hardware. A votação inicia com o nó escolhendo um subconjunto de nós de sua lista para enviar mensagens pedindo votos. Estes nós, caso aceitem participar da votação, enviam uma mensagem para o chamador, especificando uma prova de esforço, pedindo para o chamador executar uma MBF. Uma vez recebido a resposta deste desafio, os nós chamados escolhem um subgrupo de sua lista de conhecidos para enviar ao chamador para participar da votação, e calculam o seu voto e a prova de esforço, enviando o resultado de ambos ao chamador da votação. Este voto pode ser, por exemplo, o hash do conteúdo do arquivo.

Ao receber os votos, o chamador da votação confere a prova de esforço e contabiliza o voto. No fim da votação, se o arquivo que o nó possui receber uma quantidade esmagadora de votos, ele é considerado como válido. Se ele receber uma quantidade esmagadora de votos contrários, ele corrige o arquivo, possivelmente baixando um novo. Se nenhum dos dois resultados for alcançado, considera-se que houve danos significativos neste arquivo, disparando os alarmes de detecção de intrusão solicitando que algum administrador humano intervenha no sistema. O artigo considera que não é possível ter um sistema totalmente automático.

Caso a votação chegue a um resultado conclusivo, o nó que chamou a votação atualiza a sua lista de conhecidos, removendo os nós utilizados na votação, e adicionando os nós cujo voto foi favorável à decisão final.

As características deste sistema são avaliadas de acordo com três métricas: autonomia, memória e esforço.

**Limite de danos:** o sistema tenta limitar a velocidade com a qual os atacantes podem causar danos no sistema. A forma escolhida de se obter isso foi limitando a janela de tempo na qual um atacante pode tentar algum ataque, fazendo com que cada nó decida independentemente quando avaliar os seus arquivos, sem levar em consideração nenhum aspecto da rede. Isso faz com que o atacante não possa causar uma votação precoce, tendo que esperar o nó decidir realizar uma votação antes de

poder agir. Durante esta janela de tempo, ainda sim a possibilidade de ataque é limitado, devido à prova de esforço necessária para o voto ser válido. Ganha-se com isso uma grande autonomia, ao custo de mais esforço;

**sinalização custosa:** utilizado para limitar os danos causados durante uma votação. Utiliza-se para isso alguma função custosa para calcular, mas rápida para verificar o resultado. Experimentos feitos mostram que esta tática é efetiva para um conluio de até 60 nós. A partir deste ponto, o atacante consegue aumentar consideravelmente o intervalo entre votações;

**reputação:** evita-se de compartilhar informações de reputação entre os nós, utilizando-se somente os valores do nó. Esta opção foi feita para evitar a possibilidade de que um nó malicioso envie opiniões falsas, induzindo o nó a confiar em nós maliciosos;

**expiração de memória:** com o objetivo de evitar que um grupo de nós maliciosos infiltrem-se lentamente na lista de nós conhecidos, é utilizado um tempo curto de vida para os dados armazenados na memória, retirando os nós da lista após algumas votações, além de retirar os nós participantes da votação;

## 5.12 PeerTrust [49]

O sistema PeerTrust, desenvolvido pela Xiong, prima pelo cálculo da métrica, composta de 5 elementos. Este sistema foi projetado para funcionar com sistemas que implementem DHT, sendo utilizado o P-Grid.

A métrica do PeerTrust é composta de 5 elementos: a opinião do nó, o "escopo" (número de transações anteriores, por exemplo) do nó que forneceu a opinião, a credibilidade da opinião, o contexto da transação (envolvendo valores altos ou baixos, por exemplo) e o contexto da comunidade. Por contexto da comunidade entende-se como ações tomadas em prol do bom funcionamento da comunidade, como sempre fornecer opiniões sobre as transações, servindo de estímulo para tais atos.

O cálculo da reputação de um nó é dividida em duas partes: uma parte relativa às interações e outra relativa ao contexto da comunidade, cada um com um peso ajustável de acordo com a importância de cada um. Na parte das interações, as opiniões dos nós são ponderadas pela credibilidade da reputação e pelo fator referente ao contexto da transação.

Dos elementos para a computação do fator de interação, dois (a opinião e o número de transações) são facilmente conseguidos, pois são valores diretamente medidos pelo sistema. Já o fator credibilidade da opinião é mais difícil de se obter, dado que este valor não é armazenado pelo sistema. Uma possibilidade considerada é dividir a reputação de um

nó em reputação para serviço e outra para opiniões, mas isto tornaria o sistema mais complexo. A abordagem adotada foi a de calcular este valor baseado em informações já existentes. Duas possibilidades são estudadas:

- utilizar a reputação do nó dando a avaliação. Esta abordagem é simples, mas nem sempre há relação entre o comportamento em transações e o comportamento nas avaliações, visto que um nó pode ser honesto nas transações que faz, mas fornecer opiniões desonestas para os nós rivais;

- realizar um cálculo para determinar a similaridade entre os nós (o avaliado e o avaliador). Para tal, analisa-se as notas dadas a nós que ambos tenham interagido. A similaridade é determinada pela *raiz média quadrática (root mean square)* ou pelo desvio padrão. Este cálculo é relativamente complexo, porém.

Finalizando o cálculo da métrica, são definidos o cálculo dos fatores de transação e de comunidade. O primeiro é definido pelo tamanho da transação; o segundo é definido como a razão entre o número de opiniões emitidas e o número de interações realizadas. O fator de transação serve para evitar que os nós acumulem reputação colaborando em transações pequenas, para posteriormente agir maliciosamente numa transação grande.

Para lidar com traidores, além do cálculo feito com todas as interações, calcula-se também a reputação com as interações realizadas numa janela de tempo. Caso a diferença entre as reputações seja menor que um certo limite de erro, reputação final passa a ser a calculada no intervalo. Os experimentos mostram que esta tática faz com que a reputação caia drasticamente na medida em que um nó comece a agir de forma maliciosa.

A rede utilizada para a realização de testes e discussão de questões de implementação foi a P-Grid, mas o sistema poderia ter sido implementado em outra rede. Para ambas as formas de se calcular a credibilidade da reputação, são consideradas duas possibilidades: obter os valores dinamicamente, ou utilizar cache para armazenar os valores, trocando precisão por velocidade. Experimentos mostram, porém, que a precisão não é penalizada com o armazenamento das opiniões. O uso de cache diminui o *overhead* de troca de mensagem.

Complementando as questões de implementação, são considerados os problemas referentes a segurança das informações de reputação. Para tais problemas são utilizadas chaves públicas para garantir a integridade dos dados sendo transmitidos, e replicação para a segurança dos dados armazenados.

## 5.13 Credence [48, 47]

Credence é um sistema de reputação desenvolvido para Gnutella, que funciona armazenando reputação apenas para os arquivos. Não é armazenado reputação dos nós porque

o comportamento dos nós são dinâmicos, podendo mudar de acordo com o tempo. Já a qualidade dos arquivos é imutável. Os arquivos são identificados através do *hash* dos metadados utilizados para a identificação.

A avaliação de um arquivo é feita após este ter sido baixado e verificado pelo usuário. A nota pode ser *+1* caso o arquivo seja válido ou *-1* caso contrário. É possível utilizar as avaliações feitas por outros nós. Por não haver reputação sobre os nós, essas avaliações são ponderadas por um coeficiente de correlação, que indica o quanto o nó fornecendo as avaliações é parecido com o pedinte, em termos de avaliações iguais sobre os mesmos arquivos. Utiliza-se a função estatística *correlação phi* para realizar este cálculo, obtendo-se um valor entre $[-1, 1]$.

O funcionamento básico do protocolo é o seguinte: o cliente realizando a busca envia uma mensagem requisitando os votos para um determinado arquivo. Os nós que responderem enviam um subgrupo das opiniões que ele tem armazenado com as maiores notas (nós armazenados na tabela de correlação), numa mensagem assinada com a sua chave privada. Os votos recebidos têm a sua integridade verificada e são armazenados numa *cache* para uso futuro. Finalmente, eles são ajustados de acordo com o coeficiente de correlação do nó que o forneceu, já calculado nas mensagens anteriores do protocolo utilizando os votos armazenados.

O cálculo do limiar é feito utilizando-se os votos armazenados na *cache*. Periodicamente pega-se os votos fornecidos por outros para os quais também existam uma avaliação própria para o cálculo de correlação com o nó emissor do voto. Caso a correlação esteja acima de um limiar, este nó é adicionado à lista de correlação.

Para aumentar o número de nós na lista de correlação, o sistema provê um mecanismo para a transitividade dos nós na lista de correlação. Periodicamente, pede-se aos nós mais confiáveis a sua lista de correlação, adicionando-a na sua própria. O cálculo do nível de confiança é feito multiplicando a confiança no nó que forneceu a indicação pela confiança do indicado. É possível confirmar a qualidade da indicação realizando uma auditoria no valor de confiança fornecido. Nessa verificação, obtém-se os votos dados pelo nó suspeito aos diversos arquivos, e calcula-se a correlação com este nó diretamente.

## 5.14 Scrubber [14]

O Scrubber é um sistema de reputação que visa diminuir a disseminação de arquivos ilegítimos e estimular que nós honestos apaguem os arquivos corrompidos que eles venham a baixar. Para tal é utilizado reputação por nós. Não são discutidos questões sobre a rede base, nem a respeito da identidade dos nós, sendo o foco mantido somente no sistema de reputação.

A métrica é composta de dois componentes: a *experiência individual*, que representa a

opinião do nó, baseada na avaliação dos downloads feitos por ele, e no *testemunho*, que é a opinião coletada de outros nós. A experiência individual do nó $i$ sobre o nó $j$ é calculada pela seguinte fórmula:

$$I_i(j) = \begin{cases} min(0, I_i(j) - \alpha_d n^2) \; se \; o \; objeto \; e \; corrompido \\ max(1, I_i(j) - \alpha_i) \qquad\quad caso \; contrario \end{cases}$$

onde $n$ é o número de downloads consecutivos de objetos poluídos de $j$, $\alpha_d$ e $\alpha_i$ são os parâmetros de decrescimento e crescimento da métrica. O decrescimento é incrementado pelo número de downloads de objetos poluídos, fazendo com que a métrica decresça de forma mais rápida que o crescimento. O testemunho sobre um nó $j$ é calculado como a média da experiência individual dos outros nós ponderada pela reputação dos nós consultados. Para o cálculo da reputação final são utilizados tanto a experiência individual quanto o testemunho, utilizando-se de um peso ($\beta$) para indicar o quanto o testemunho de outros nós é relevante. É suposto que todas as opiniões encontram-se acessível para o cálculo do testemunho sobre qualquer nó.

Em [13], o Scrubber é estendido para levar em consideração a reputação de arquivos. Para isso, também é utilizado o testemunho da comunidade sobre os arquivos, além do testemunho dos nós. O parâmetro para decrescimento é divido em dois, um para indicar a queda no caso de fornecimento de arquivo corrompido, e outro para fornecimento de opinião ruim (mentir sobre a qualidade do arquivo). O cálculo do testemunho de um arquivo é feito de forma análoga ao testemunho dos nós.

O decrescimento da reputação do nó é calculada de uma forma ligeiramente modificada: caso o nó tenha mentido, a métrica é decrescida da forma descrita anteriormente, porém utilizando o parâmetro para mentirosos; caso o nó tenha fornecido um arquivo corrompido, utiliza-se o parâmetro para fornecimento de arquivos corrompidos; caso ambos aconteçam, calcula-se a reputação das duas formas, utilizando-se a de maior valor[13]. O crescimento é feito da forma anterior.

## 5.15 Bauermann [7]

Este método visa impedir o download de peças de arquivos corrompidos em redes BitTorrent. Nessas redes, os arquivos são divididos em peças, e estas em blocos. Consegue-se com isso realizar o download do arquivo de diversas fontes.

A cada nó é associada uma reputação, que é aumentada ou diminuída de acordo com a cooperação do nó em uma peça corrompida ou não corrompida, respectivamente. Quando a reputação de um nó é diminuída além de um limite, ele é posto em quarentena, sendo desconsiderado para o fornecimento de mais peças. Tenta-se recuperar a peça corrompida baixando blocos considerados corrompidos até a recuperação da peça.

Um outro algoritmo proposto, com o objetivo de impedir a mentira de peças (enviar blocos incorretos com o objetivo de corromper a peça), funciona sem o uso de reputação: os nós monitoram a quantidade de blocos enviados e recebidos dos outros membros do enxame (grupo de nós compartilhando um arquivo). Caso esta quantidade num determinado nó esteja abaixo de um limiar, supõe-se que ele seja malicioso, sendo posto em quarentena. Assume-se que os nós maliciosos realizem o esforço mínimo para corromper uma peça, ou seja, só envie um bloco corrompido.

## 5.16 Conclusões

Neste capítulo foram analisados diversos sistemas de reputação distintos. A maioria está focada no problema de lidar com nós maliciosos, com alguns lidando com os nós que não compartilham arquivos (*free-riders*). Para o armazenamento/busca dos dados foi dada preferência para a utilização de redes estruturadas, devido a sua eficiência e pela segurança causada pela escolha aleatória do local de armazenamento, seguido das redes Gnutella, devido à popularidade destas. Destacam-se também na questão armazenamento os sistemas que armazenam as informações nos próprios nós avaliados, com o devido suporte para evitar a corrupção dos dados. Nas redes DHT, o armazenamento dos dados é feito no nó determinado pelo *hash* do identificador. Replicação é feita escolhendo outros nós através de outras funções de *hash*. Em sistemas baseados em Gnutella cada nó armazena suas próprias opiniões, e o compartilhamento geralmente é feito através de mensagens de consulta aos outros nós, formando-se redes de confiança (*web of trust*).

Na questão das métricas, diversas alternativas foram apresentadas, com diversos níveis de complexidade. A maioria, de algum modo, utiliza as opiniões de outros nós, com algum mecanismo para checar a confiabilidade destas opiniões. Enquanto sistemas que utilizam DHT possuem uma grande variedade de métricas, sistemas baseados em redes de confiança calculam a confiança de uma forma parecida: ajustando o valor de reputação obtido através de um caminho pela reputação dos nós intermediários. O sistema de reputação EigenTrust [24] implementa uma idéia parecida utilizando DHT. Outras formas de agregação de opiniões consideradas nos artigos foram fluxo máximo, soma das opiniões dos diversos nós ou alguma outra função de agregação além da soma. PeerTrust [49] adota uma métrica composta, utilizando-se de diversos fatores para o cálculo.

A determinação da identidade dos nós é algo cujo tratamento foi negligenciado por diversos trabalhos. De fato, este é um problema de tratamento difícil num ambiente distribuído, no qual não há uma autoridade central para gerenciar a criação de identidades, o que faz com que a troca de identidade seja extremamente difícil de ser evitada. Diversos artigos lidam com o problema da troca de identidade tornando-a indesejável, atribuindo pouca confiança aos recém entrados no sistema. Alguns poucos sistemas adotaram um

nível de confiança para desconhecido adaptativo, ajustando o nível de acordo com a proporção de desconhecidos maliciosos num determinado intervalo de tempo.

As principais características dos sistemas discutidos foram agrupadas na Tabela 5.1.

Tabela 5.1: Características dos sistemas de reputação.

| Nome<br>*Objetivo* | Sistema Base | Acúmulo de Informações | Métrica | Identidade | Extras |
|---|---|---|---|---|---|
| Marti [30]<br>*Detectar nós maliciosos* | Gnutella | Cada nó armazena próprias opiniões e cuida da integridade | Local: $\frac{autênticos}{total}$<br>global: média ponderada da avaliação dos nós pela sua opinião sobre estes | Não trata | Cache dos amigos de maior reputação |
| Aberer [1]<br>*Evitar nós maliciosos* | P-Grid, ou outra rede estruturada | Acumula informações em nó escolhido pela arquitetura da rede. Mantém integridade com replicação. | Armazena reclamações, acima de um nível o nó não é confiável. | Não trata | |
| P2PRep [12]<br>*Evitar nós maliciosos* | Gnutella | Armazenamento no próprio nó, compartilha opiniões. | Número de opiniões positivas e negativas, sem função de cálculo específica | Tenta evitar ganhos com troca de identidade. | Hash de arquivos permite baixar de qualquer nó. Voto por cluster evita ataques sybil. |
| XRep [16]<br>*Evitar arquivos corrompidos* | Gnutella, mas pode ser qualquer um. | Mesmo que P2PRep, com acúmulo de informações por arquivos também, utilizando hash do arquivo. | Nós armazenam suas próprias opiniões. Sobre nós armazena como no P2PRep; sobre arquivos armazena se é autêntico ou não. Compartilha-se opiniões. | Não lida | Resiste a ataques de Sybil, roubo de identidade, mas não ballot stuffing. |
| RCertP e RCertPX [27, 34]<br>*Sistema de armazenamento eficiente* | Qualquer um. | Armazena as opiniões no nó avaliado, assinada pelo avaliador para evitar corrupção. | Não define métrica. | Utiliza chaves públicas, sem autoridade central. | RCertPX contata últimos avaliadores para evitar descarte de opiniões negativas. |
| EigenTrust [24]<br>*evitar downloads corrompidos.* | Qualquer, mas preferência por algum que implemente DHT. | Cada nó mantém suas opiniões, mas reputação global é armazenada na DHT. Integridade é obtida com replicação. | Considera-se as opiniões numa matriz $[a_{ij}]$ das notas de i sobre j. As reputações são o auto-vetor desta matriz. | Não fala. | Cálculo por autovetor considera a reputação de quem deu opinião no cálculo. |
| Secure Score Management [5]<br>*gerenciamento de reputação* | Sistemas estruturados (implementem DHT). | Informações do nó N são armazenadas no nó de id = hash(N). | Não tem. | Não fala. | Lida com microcrédito também. Parecido com protocolo usado em [24]. |

| nome<br>*objetivo* | Sistema Base | Acúmulo de Informações | Métrica | Identidade | Extras |
|---|---|---|---|---|---|
| Feldman [21]<br>*Combate os free-riders* | Não cita. | Não especifica, sugere DHT. | Taxa entre serviços prestados e consumidos. Opinião de outros são consideradas com algoritmo de fluxo máximo | Considera possibilidade de troca de identidade. | Baseado em mecanismos econômicos. Usa política adaptativa para novatos. |
| Gupta [23]<br>*Combater nós maliciosos ou free-riders* | Gnutella | Armazenado no nó avaliado, assinado por um servidor central para evitar adulteração. | Credita-se pontos por upload, resposta de busca, e compartilhar arquivo. Debita em downloads se aplicável. | Troca de identidade implica perda de crédito, logo não vantajoso. | Cálculo de créditos é feita num servidor central, que retorna os créditos assinados. |
| Yu [50]<br>*Determinar se um nó é confiável.* | Rede social de troca de informações. | Cada nó armazena suas opiniões. | Valores entre -1 e 1. Atualização com pesos diferentes para cooperação e não cooperação. | Não cita. | Limites para determinar confiança ou desconfiança, com intervalo de incerteza |
| LOCKSS [39]<br>*Manter dados em longuíssimo prazo.* | Sistema de armazenamento artigos, estilo biblioteca. | Nó mantém lista de nós conhecidos. | Prova de esforço no lugar de métricas de reputação. Validade dos arquivos é verificada por votação. | Não depende de identidade dos nós. | Escolhas de projeto bem diferentes das comuns. |
| PeerTrust [49]<br>*Combater nós maliciosos.* | P-Grid. | Dados armazenados na DHT. | 5 elementos: a opinião, o escopo do nó, a credibilidade, e os contextos da transação e da comunidade. | Não trata. | Contexto da comunidade visa estimular a emissão de votos após as interações. |
| Credence [48, 47]<br>*Combater nós maliciosos.* | Gnutella. | Dados armazenados nó próprio nó. | Reputação apenas para arquivos, agregação de notas +1 ou -1. | Utiliza PKI, prova de esforço para entrega de chaves. | "reputação" dos nós calculadas por coeficiente de correlação dos votos. |
| Scrubber [14, 13]<br>*Combater nós maliciosos.* | Não especifica. | Dados armazenados nó próprio nó. | Reputação de nós e arquivos, com uso de opiniões de outros nós. | Utiliza PKI, prova de esforço para entrega de chaves. | Descrescimento da reputação proporcional ao quadrado do número de downloads poluídos consecutivos |
| Bauermann [7]<br>*Combater nós maliciosos.* | BitTorrent. | Dados armazenados nó próprio nó. | Valores entre $[0:1]$, crescimento de $\delta_i$ e decrescimento de $2 * \delta_d$ | Não fala. | Favorece nós que transmitem blocos legítimos. |

# Capítulo 6

# O Sistema de Reputação Proposto

O sistema de reputação proposto visa combater os nós maliciosos, que entram no sistema com o objetivo de disseminar arquivos ilegítimos. Ele foi desenvolvido para funcionar sobre o Chord, porém pode ser utilizado em qualquer rede estruturada que implemente uma DHT. A opção por estas redes se deu pela eficiência de acesso aos dados.

Além do uso do Chord, também é utilizado um mecanismo de indexação de arquivos denominado Arpeggio [11]. Este sistema provê à rede uma forma de procurar os arquivos através de busca por metadados. Ele cria sobre o Chord um índice distribuído associando aos metadados existentes uma lista dos arquivos que possuem tais metadados, e, para cada arquivo, uma lista dos nós fornecedores.

O sistema de reputação baseia-se na reputação dos arquivos, indicando a qualidade destes. Os nós também possuem reputação, indicando a confiabilidade das opiniões emitidas pelo nó. Dessa forma consegue-se obter uma confiança maior nas opiniões dos nós.

A integridade das mensagens e dos votos é mantida com o uso de criptografia. Os nós são identificados por um par de chaves pública/privada. O identificador dos nós é calculado como o *hash* da chave pública. Não é utilizado uma autoridade certificadora para garantir a associação de identidade entre a chave e um usuário, pois não é desejável a associação da identidade real do usuário. Para o sistema, a identificação somente pela chave já é o suficiente, conforme discutido na seção 3.1.2.

## 6.1 Motivação

O que foi buscado na abordagem do sistema de reputação proposto é limitar o problema da confiabilidade da opinião emitida por um nó, problema este que aparece nos sistemas no qual a reputação é armazenada de forma centralizada.

O armazenamento centralizado mantém todas as opiniões já agregadas. Essa forma de armazenamento ocupa menos espaço, mas não permite que se saiba quem foram os nós

que forneceram opinião. Logo é possível adulterar a reputação de um nó para mais ou para menos através de diversas avaliações falsas. Como não é possível verificar a reputação dos nós que emitiram as opiniões, não dá pra filtrar as opiniões maliciosas.

Por outro lado, implementar uma verificação da reputação dos nós no estilo redes de confiança, num caminho completo de referências entre o nó que pediu um arquivo e o que fornecerá, é muito custoso. É necessário armazenar separadamente todas as opiniões emitidas por todos os nós, e contatar todos os nós no caminho para obter todas os valores de reputação necessários. Esta última tarefa é a mais complicada, pois encontrar todos os nós necessários é custoso mesmo considerando redes estruturadas, que possuem um custo menor para encontrar um nó.

Um outro método utilizado nos sistemas de reputação pesquisados é o armazenamento no nó avaliado, com verificação de integridade para evitar que o nó adultere as opiniões sendo armazenadas por ele. Este método é bastante eficiente, mas é necessário verificar a integridade das opiniões sempre que elas forem utilizadas. Geralmente isso é feito através da assinatura do último nó que emitiu uma opinião, mas para tal é necessário que este nó esteja conectado. Caso contrário não é possível atestar a qualidade da informação fornecida.

A verificação da reputação por rede de confiança é a mais eficaz, mas o custo de implementar uma rede completa é alto. Por conta disso, foi optado por fazer a verificação em um nível somente, verificando apenas a reputação dos nós que fornecerão as opiniões sobre os arquivos candidatos a serem baixados. Essa verificação permite filtrar as opiniões oriundas de nós com baixa reputação, apesar de não proteger totalmente contra inflação de reputação.

## 6.2 Acúmulo de Informações

### 6.2.1 Armazenamento

Existem 3 informações a serem armazenadas: o índice dos arquivos, a reputação dos nós e a opinião sobre os arquivos. A primeira é armazenada na DHT segundo o artigo do Arpeggio [11]; A reputação do nó também é armazenada na DHT, utilizando como chave o hash da chave pública. A opinião sobre os arquivos é armazenada no nó que a forneceu, indexada pelo hash do conteúdo do arquivo.

A rede suporta a existência de mais de uma versão do mesmo arquivo. Por conta deste suporte, o índice de arquivos possui, para cada entrada de arquivo, uma lista das diferentes versões do arquivo, listadas pelo hash do conteúdo do arquivo. Para cada versão do arquivo é mantida uma lista dos nós que a possuem, e uma lista de quem deu opinião.

### 6.2.2 Integridade

A integridade do sistema é obtida através da replicação dos dados. Cada dado da DHT é armazenado em 3 lugares diferentes, cada lugar determinado pelo uso de uma função diferente de hash para o cálculo da chave. Ao acessar um dado, pode-se optar por verificar a autenticidade do mesmo, acessando as três cópias e verificando qual versão possui o maior número de cópias (maior ou igual a 2). Os dados de reputação do arquivo, por serem armazenados no nó de quem deu a opinião, não necessitam de replicação.

Para evitar ataques de *ballot stuffing*, é utilizado prova de transação para a emissão do voto: as mensagens trocadas para o download do arquivo são assinadas. Ao enviar o voto, o nó envia junto uma mensagem assinada enviada pelo nó que forneceu o arquivo, como prova de transação. Assim evita-se que diversos nós votem sem nunca terem contatado o nó, destruindo a sua reputação. Este tipo de ataque não é totalmente evitado, mas é controlado, visto que o nó malicioso deve enviar uma mensagem assinada para cada voto que o nó malicioso quiser emitir. Após a primeira requisição falsa de arquivo, pode-se negar download a este nó, impedindo novos votos ruins.

## 6.3 Métrica

A definição da métrica é feita baseada em três conceitos: a *opinião sobre um arquivo*, que indica a qualidade do arquivo segundo um nó, como boa (valor 1) ou ruim (valor 0); a *reputação do nó*, que indica o quanto a opinião emitida por um nó é confiável, definida como um número real no intervalo $[0, 1]$; e a *reputação do arquivo*, que é calculada baseada nas opiniões sobre o arquivo e a reputação dos nós que forneceram as opiniões utilizadas, sendo também um numero real no intervalo $[0, 1]$. Não é mantida a reputação dos nós no contexto de fornecimento de arquivo, pois já é possível medir a qualidade do arquivo pela reputação que o arquivo tem. Mais interessante é medir a qualidade da opinião que o nó fornece. Desta forma, uma vez escolhido o arquivo, é possível baixá-lo de qualquer nó, determinando a autenticidade do arquivo utilizando um código de verificação de integridade, neste caso o hash do conteúdo. Caso seja detectado que o arquivo é ruim, a reputação dos nós de bons indicadores é diminuída.

A Figura 6.1(a) mostra de forma simplificada o processo de obtenção da opinião dos outros nós. O nó **A** quer baixar o arquivo **X**, possuído pelos nós **B** e **D**. Para isso, ele obtém a reputação dos nós que possuem o arquivo ($\mathbf{O}_B$ e $\mathbf{O}_D$), entrando em contato com **C** e **E**. Num segundo passo, **A** obtém a opinião dos nós contatados que possuírem reputação menor que a mínima, contatando-os diretamente. De posse das opiniões dos nós e de suas reputações, **A** calcula a reputação do arquivo e decide baixar o mesmo (passo 3), caso a reputação do arquivo seja maior que a reputação mínima. Terminado o

download, **A** está apto a avaliar a qualidade do arquivo (passo 4).

As opiniões são dadas no final do download, após a verificação da autenticidade do arquivo, conforme mostrado na Figura 6.1(b). Ao emitir uma opinião, o nó que baixou o arquivo (**A**) armazena a sua opinião sobre o arquivo, e atualiza a reputação dos nós que lhe enviaram uma opinião (**B** e **D**), enviando uma mensagem para os nós responsáveis pelo armazenamento destas reputações (respectivamente **C** e **E**) indicando se a reputação armazenada deve ser aumentada ou diminuída, de acordo com a concordância ou não com a opinião do nó que baixou. No exemplo, a opinião de **B** foi igual à emitida por **A**, logo ela foi aumentada; o contrário ocorreu com a opinião de **D**.

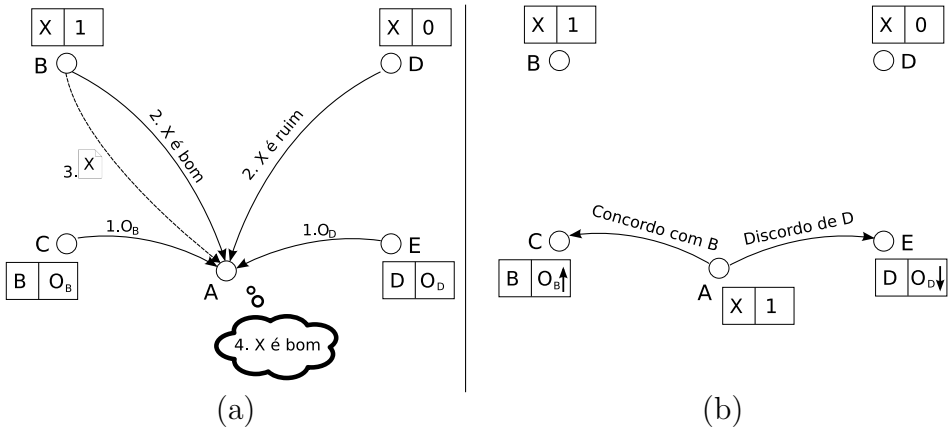



Figura 6.1: Funcionamento da métrica (a) antes de baixar o arquivo (b) depois de baixar o arquivo.

### 6.3.1 Reputação do Nó

A reputação do nó é representada por um valor real entre $[0, 1]$. Este valor é atualizado de acordo com uma função que indica o comportamento da reputação para aumento ou diminuição dos valores.

Para a atualização da reputação do nó foram consideradas três funções. As simulações foram rodadas com as três, com o objetivo de se comparar os resultados e escolher a mais adequada. A seleção das funções se deu de acordo com as características descritas na seção 4.4.3. As funções utilizadas foram: *percentual*, *exponencial*, e *normal*.

A atualização descrita acima serve para o crescimento da métrica. Para a descida foi decidido pela diminuição de uma porcentagem do valor atual. Esta fórmula para o decaimento foi considerada boa, pois ela prevê uma queda maior para nós com reputação alta, revelando traidores rapidamente. Uma desvantagem desta queda mais rápida é uma vulnerabilidade maior a ataques, pois um nó malicioso, ao dar uma avaliação ruim a um

nó honesto com boa reputação, diminui bastante a reputação. A equação que descreve este funcionamento é

$$R(t) = R(t-1)q$$

onde $R(t)$ é a reputação do nó no tempo $t$ e $q$ é a taxa de decaimento em caso de não cooperação.

Serão descritas agora as funções consideradas para o crescimento da métrica.

### Percentual

A função percentual aumenta a reputação do nó em uma porcentagem do que falta para a reputação chegar a 1. Isso é expresso na seguinte relação de recorrência:

$$\begin{aligned} R(0) &= R_0 \\ R(t) &= R(t-1) + (1 - R(t-1))p \end{aligned}$$

onde $R_0$ é a reputação inicial e $p$ é a taxa de crescimento da reputação em caso de colaboração. Esta relação de recorrência, em sua forma fechada (considerando apenas colaborações), se torna

$$R(t) = 1 + (R_0 - 1)(1-p)^t = 1 - (1 - R_0)(1-p)^t. \tag{6.1}$$

Como $1 - R_0 \geq 0$ e $(1-p)^t$ é exponencial, temos uma função exponencial invertida em torno do eixo $x$, o que a faz possuir uma curva estilo logarítmica. Escolhendo os valores de $p$ (subida) e $q$ (queda) adequadamente podemos fazer a função assumir um crescimento mais lento e uma queda mais rápida. Porém a natureza da função força um crescimento rápido no caso de reputação baixa. Este comportamento é facilmente visto na relação de recorrência, onde o crescimento é realizado como uma porcentagem do complemento de 1 da reputação anterior, logo as reputações mais próximas de 0 apresentarão crescimento maior que reputações mais próximas de 1. Pelo fato da reputação ser calculada passo a passo, foi preferido utilizar a relação de recorrência.

Esta função é baseada em partes da métrica utilizada em [50], mais precisamente na parte de crescimento do valor da métrica para casos positivos (a métrica do artigo varia entre $[-1, 1]$).

### Exponencial

A função percentual apresenta o funcionamento inverso ao esperado, com crescimento rápido para os nós com reputação baixa, e crescimento devagar para os nós com reputação

alta. Portanto, nada mais natural que considerar também a utilização de sua função inversa, de forma exponencial. Partindo da Equação 6.1, temos como a função exponencial

$$R(t) = log_{(1-p)}\left(\frac{t-1}{R(0)-1}\right).$$

Esta curva possui um comportamento parecido com a exponencial pois a base do logaritmo é menor que zero. A função exponencial possui um comportamento mais conforme com as características desejadas para a métrica que a percentual.

Por ser a função inversa da percentual, alguns problemas precisaram ser tratados: um deles foi o problema de domínio e imagem da função. A função percentual realiza o mapeamento $[0, \infty] \rightarrow [0, 1]$, logo a função aqui denominada exponencial faz o mapeamento inverso: $[0, 1] \rightarrow [0, \infty]$. Foi necessário, portanto, a realização de ajustes para que os intervalos se tornassem adequados. Este ajuste foi feito estabelecendo um número de opiniões positivas seguidas necessárias para atingir a reputação 1. Seja $n$ este parâmetro. Para termos um domínio no intervalo $[0, 1]$, definimos $t' = \frac{t}{n}$. Para limitar a imagem no intervalo $[0, 1]$, dividimos a função pelo valor máximo possível, ou seja, $R(1)$. Assim temos a função final

$$R'(t) = \frac{R(\frac{t}{n})}{R(1)}$$

que possui as características desejadas. O parâmetro $n$ é utilizado nas outras funções de forma indireta, influenciando na escolha dos valores dos parâmetros de modo a que todos alcancem a reputação máxima com aproximadamente o mesmo número de avaliações, conforme veremos no Capítulo 7.

### Normal

A função normal é baseada na distribuição estatística normal, ou Gaussiana. Esta distribuição é definida por dois parâmetros, a média ($\mu$) e a variância ($\sigma$). O uso da normal como métrica se dá por diversas características que se mostraram interessantes: por ser uma função estatística, ela varia naturalmente no intervalo $[0, 1]$; além disso, ela tem um crescimento lento para valores baixos, uma subida rápida para valores intermediários e para valores altos ela tende suavemente a 1.

A função normal é:

$$f(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{(x-\mu)^2}{2\sigma^2}} dx$$

As três métricas consideradas possuem comportamento diferente uma das outras. Essa diferença pode ser observada na Figura 6.2.

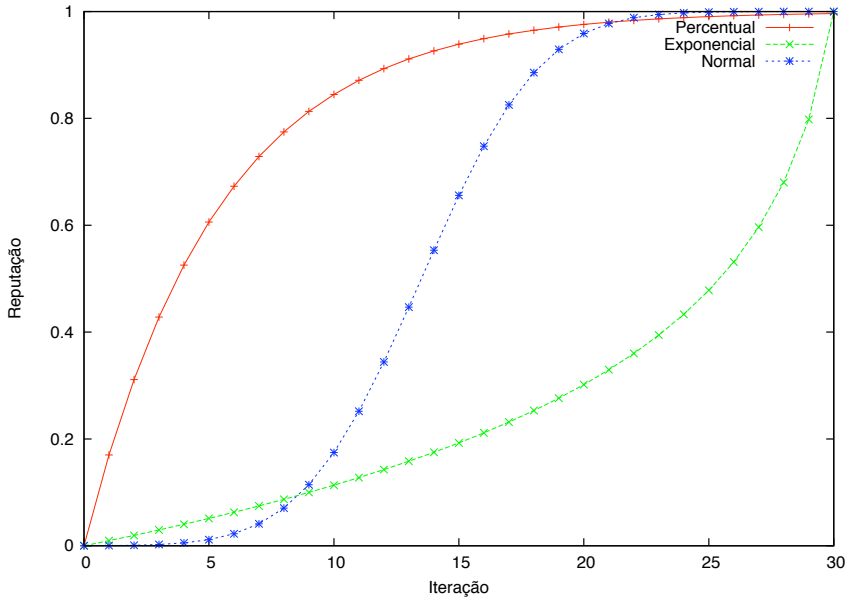



Figura 6.2: Evolução das métricas consideradas.

### 6.3.2 Reputação do Arquivo

O cálculo da reputação final de um arquivo é dado pela soma das opiniões dos nós consultados multiplicados pela sua reputação, dividido pelo número de nós. Seja $o_i$ a opinião do nó $i$ sobre um determinado arquivo $a$, e $r_i$ a sua reputação. A reputação de um arquivo $R_a$ levando em consideração a opinião de $n$ nós é calculada como:

$$R_a = \frac{\sum_{i=0}^{n} r_i.o_i}{n} \tag{6.2}$$

Este cálculo foi utilizado no lugar de uma média ponderada porque esta última trata a reputação dos nós de forma relativa, e não de forma absoluta como desejado. Como exemplo, considere que há apenas um nó, e que este nó tenha reputação de $r_i = 0,1$, e que ele diga que o arquivo é bom ($o_i = 1$). Pela média ponderada, teríamos $R_a = \frac{0,1.1}{0,1} = 1$, ou seja, o arquivo teria reputação máxima mesmo vindo de um nó com reputação baixa. No cálculo utilizado, temos $R_a = \frac{0,1.1}{1} = 0,1$

A reputação calculada dessa forma fornece valores no intervalo $[0, 1]$. Ela atinge o valor 0 quando todos os valores de reputação e/ou opinião são zero, e 1 com todos os valores de reputação e de opinião iguais a um. O fato de de ter no denominador o número de opiniões consideradas, aliado com o fato das opiniões poderem ser apenas 0 ou 1, faz com que a função se comporte de forma parecida com uma média ponderada pelas opiniões. Porém, ao contrário desta média, a função utilizada não desconsidera opiniões negativas

no cálculo. Na média ponderada pelas opiniões, uma opinião de valor zero adiciona zero na soma do numerador e no denominador, sendo ignorada. Na função utilizada ela é contada no denominador, efetivamente diminuindo o valor da reputação, como desejado.

## 6.4 Protocolo

Segue-se agora a descrição do protocolo, desde a busca pelo arquivo utilizando um conjunto de palavras chave que permitam identificar o arquivo, até o envio das avaliações dos nós aos responsáveis.

A partir dos metadados de identificação do arquivo consulta-se o índice do arquivo, obtendo uma estrutura de dados que contém a lista de arquivos que satisfazem os critérios de busca, junto com a lista dos nós que possuem cada versão do arquivo e da lista dos nós que realizaram avaliações destes. A existência de duas listas se faz necessária pois nem todos os nós que possuem o arquivo enviaram avaliação, assim como nem todos que enviaram avaliação irão manter o arquivo. No caso de um nó honesto baixar um arquivo ilegítimo, por exemplo, ele enviará uma avaliação negativa, mas apagará o arquivo, não se incluindo na lista de nós que o possui.

De posse da lista de nós que possuem cada arquivo, procura-se a reputação destes, calculando o *hash* dos seus identificadores para descobrir o nó responsável pelo armazenamento da reputação. Como podem haver uma quantidade grande de nós, pode-se optar por consultar uma amostragem deles, e não todos.

Obtidas as reputações dos nós, os mais confiáveis são escolhidos a partir de algum algoritmo de seleção. Este algoritmo pode ser, por exemplo, os $N$ nós mais confiáveis ou os todos os nós com reputação acima de um limiar. Os nós escolhidos são então contatados diretamente para serem requisitados do envio da sua opinião sobre o arquivo.

Com a reputação dos nós e a reputação do arquivo, é realizado o cálculo da reputação final dos arquivos de acordo com o algoritmo descrito na Seção6.3.2.

Baseado nesse resultado final, é decidido qual versão do arquivo deve ser baixada. Um algoritmo simples seria baixar a versão de maior reputação, porém se todas tiverem reputação ruim, isso pode indicar que todas as versões são corrompidas, se fazendo necessário o uso de um limite inferior de reputação para o download. Por outro lado, um limite muito alto pode ser excludente demais, impedindo os arquivos novos de serem baixados devido à baixa reputação inicial que eles possam apresentar.

Uma vez escolhido o arquivo, escolhe-se aleatoriamente um nó para baixar. O nó escolhido neste ponto não é importante, dado que a reputação é associada ao arquivo, e não aos nós. É possível inclusive baixar o arquivo de diversos nós simultaneamente.

Depois de baixar o arquivo, ele é avaliado para confirmar a autenticidade. Nesta etapa é feita uma verificação de integridade pelo *hash* do conteúdo do arquivo e uma verificação

de conteúdo pelo usuário. A opinião sobre o arquivo é armazenada no próprio nó. Para cada nó que forneceu uma avaliação sobre o arquivo, verifica-se a concordância entre a opinião fornecida e a avaliação dada, e envia-se mensagens para atualizar a reputação dos nós que lhe indicaram o arquivo, aumentando a reputação dos nós cujas opiniões foram concordantes, e diminuindo dos discordantes.

## 6.5 Identidade

A manutenção da identidade dos nós é feita através de chaves públicas sem autoridade certificadora, com cada nó gerando as suas próprias chaves. O uso de chaves públicas permite que se utilize provas de transação, além de prevenir perda de identidade por troca de IP, ou por roubo simples de identidade. Mudança de identidade continua sendo possível, mas agora há um trabalho adicional de ser gerar as chaves novas.

Não é utilizado um servidor central de distribuição por diversos motivos: a rede toda é descentralizada, não sendo interessante ter um servidor centralizado. Além disso, os possíveis problemas criados pela falta de servidor central não se concretizam.

Os problemas causados pela falta da autoridade certificadora são: possibilidade de troca de identidade e ataques *man-in-the-middle*. O primeiro é possível mesmo sem o uso de chaves, não sendo realmente um problema novo; o *man-in-the-middle* não é problemático, pois este ataque visa o roubo de identidade pela associação da chave do atacante com a identidade roubada. Porém, os dados referentes a uma identidade são endereçados pela chave. Um atacante que tente realizar este ataque, ao passar a sua chave para a vítima, não poderá utilizar-se da reputação do nó do que ele estiver sendo personificado, pois a reputação a ser consultada será a associada a sua chave.

## 6.6 Ataques

Os sistemas de reputação devem ser resilientes a ataques que visam minar o seu bom funcionamento. Além de serem testados contra diversos tipos de ataques através de simulação, cujos resultados serão mostrados no próximo capítulo, é interessante realizar previamente uma análise do comportamento do sistema de reputação frente a algumas ameaças mais comuns. Serão considerados aqui os ataques de *cooperação em r% das vezes*, os ataques *sybil* e os de conluio. Estes ataques visam de um modo geral inflar a reputação de nós maliciosos, para posteriormente utilizar-se dela para induzir outros nós a acreditarem que um determinado arquivo malicioso é bom.

No ataque no qual os atacantes cooperam r% das vezes, não cooperando nas outras, os nós maliciosos têm como objetivo aumentar a sua reputação, de forma a aumentar a

reputação dos arquivos maliciosos a serem oferecidos. O uso de métricas de subida lenta e queda rápida para a reputação impõe que a taxa de colaboração seja alta para que este ataque seja efetivo, o que diminui a quantidade de dano que um nó implementando este tipo de ataque consiga causar.

A realização de um ataque *sybil* ainda é possível, porém é imposto um custo maior para o seu funcionamento. Para que um nó crie diversas identidades, é necessário que ele gere várias chaves; além disso, cada identidade tem que aumentar a sua reputação, para que o nó seja considerado confiável. Como é utilizado prova de transação nas avaliações, estes nós têm que falsificar transações também. Todas estas barreiras impõem um certo custo, que apesar de não impedir que os ataques *sybil* ocorram, pode impedir que um único nó possa se passar por uma quantidade muito grande de nós.

No ataque de conluio, um grupo de nós maliciosos se une para inflar a reputação um dos outros realizando transações entre si. Por estas transações falsas são conseguidas provas de transação para que os atacantes possam realizar avaliações entre si, inflando rapidamente as suas reputações. Este é um tipo de ataque bastante difícil de se combater, pois as transações entre os nós maliciosos são aparentemente legítimas. A forma mais eficiente de se lidar com este tipo de ataque é através de redes de confiança, pois por ela as opiniões do grupo de nós maliciosos seriam consideradas pouco confiáveis. Redes de confiança, porém, são custosas de serem implantadas.

A taxa de sucesso destes ataques depende da diferença entre a quantidade de boas ações necessárias para construir a reputação e a quantidade de ações ruins que é possível realizar. Esta razão é regida pelo funcionamento da métrica. Mesmo que ainda seja possível propagar arquivos ruins, os nós que o quiserem fazer terão antes que propagar arquivos bons, possivelmente numa quantidade maior que a de arquivos ruins. Outro fator de limite de ataque é o número de atacantes por nós normais. Caso a quantidade de atacantes seja muito menor, a influência destes será pequena. Com mais atacantes que nós honestos, porém, a situação se inverte, e os nós honestos passam a ter menos influência que os nós atacantes efetivamente tomando a rede. A partir deste ponto, o sistema de reputação passa a ser de pouca valia, pois as opiniões maliciosas dos atacantes passam a contribuir mais para o cálculo das reputações.

## 6.7 Conclusões

O sistema de reputação proposto visa adicionar verificabilidade nas opiniões armazenadas da DHT, estilo rede de confiança. Como implementar uma rede de confiança integralmente é custoso, optou-se por adicionar um nível de verificação.

São utilizados dois valores de reputação: para os arquivos e para os nós. A reputação dos arquivos refere-se à qualidade dos mesmos, enquanto a reputação dos nós julga a

qualidade da opinião fornecida pelos nós sobre os arquivos. Estes valores correspondem aos dois níveis de reputação descritos acima.

Alguns ataques populares foram considerados. Uma análise breve mostra que as características da rede devem diminuir os efeitos destes ataques. Uma análise mais profunda sobre a resistência da rede será feita através de simulações no próximo capítulo.

# Capítulo 7

# Simulação

O funcionamento de um sistema pode ser verificado de diversas formas. O método mais completo de se comprovar o funcionamento é através de uma prova analítica formal do funcionamento do protocolo. Porém, para protocolos mais complexos, este método se torna extremamente difícil de ser aplicado.

Outra possibilidade é a implementação do protocolo, e a implantação numa rede real, preferencialmente numa rede controlada para um melhor controle dos resultados. Este método permite que se realize medições no ambiente no qual o protocolo será utilizado, o que provê medições precisas. A implementação de protocolos distribuídos, como os de sistemas P2P, enfrentam grandes dificuldades, por precisar de uma grande quantidade de nós para a execução. Estas dificuldades decorrem da complexidade de se comandar uma grande quantidade de computadores, e distribuir as versões novas do protocolo para os nós, o que pode tornar a execução impraticável. Algumas iniciativas visam resolver estes problemas relacionados ao teste das redes P2P facilitando a distribuição e o comando de diversos computadores, como o PlanetLab[1].

Uma forma mais fácil, porém ainda eficaz de verificar o funcionamento de um sistema é através da simulação do mesmo. Para tal é criado um programa que emula a rede e os protocolos envolvidos no funcionamento do sistema, simulando o funcionamento de todos os nós da rede. Dessa forma, os problemas de se lidar com nós distribuídos é eliminado, já que as alterações no simulador afetam automaticamente todos os nós.

A simulação pode ser feita com diversos níveis de detalhamento. Pode-se simular somente os protocolos relevantes ao sistema sendo testado, ou então simular a rede inteira, incluindo os protocolos mais básicos, como o TCP e o IP. No primeiro caso, a simulação fica mais rápida, pois não é gasto tempo de processamento para simulação dos protocolos inferiores da rede. Por outro lado, a simulação perde precisão nas métricas relacionadas a tempo. No segundo caso consegue-se uma precisão maior, ao custo de uma simulação mais

[1] www.planet-lab.org/

demorada. Deve-se escolher o nível de detalhamento da simulação baseado na necessidade: caso não sejam necessárias medições baseadas no tempo, uma simulação sem os protocolos de rede é mais eficiente.

Ainda na questão do detalhamento da simulação, estes podem ser divididos em dois tipos de simuladores: os simuladores por *ciclos* ou por *eventos*.

Nos simuladores por ciclo são realizados todos os passos da busca em cada ciclo [33]. Cada nó pode estar em um dos estados de: *realizando uma busca*, *inativo*, ou *desconectado da rede* [42]. Este tipo de simulador desconsidera totalmente o tempo. As simulações deste tipo são mais rápidas de serem feitas e executadas, ao custo de exatidão das medidas.

Os simuladores por evento funcionam através do agendamento e execução de eventos. Os eventos representam acontecimentos na rede, como a recepção de uma mensagem, ou a expiração de algum *time to live*. A base destes simuladores é o escalonador, que é responsável pelo controle da passagem do tempo, e pela execução dos eventos no momento certo. Como exemplo, quando uma mensagem é enviada, é criado um evento "recepção de mensagem", agendado para o tempo no qual a mensagem deve ser recebido. Este evento é inserido no escalonador, que mantém a fila de eventos ordenadas pelo tempo de execução. Ao terminar a execução de um evento, o escalonador pega o próximo da lista, avança o tempo da simulação ajustando-o para o tempo deste evento e o executa. Estas simulações provêem mais precisão, ao custo de uma necessidade maior de processamento.

Por serem mais simples, os simuladores por ciclo são geralmente utilizados para uma validação preliminar do protocolo, detectando falhas na especificação do protocolo. Para uma validação mais completa, é recomendado o uso da simulação por evento, visto que este tipo de simulação também detecta problemas de sincronização das mensagens.

Para a validação do sistema de reputação proposto foi realizada a simulação por eventos. Na Seção 7.1 serão discutidas as características com as quais a simulação foi realizada, os valores utilizados nos parâmetros e os modelos de atacantes considerados. Na Seção 7.2 os resultados da simulação são apresentados.

## 7.1 Características

A simulação foi feita utilizando-se o *peersim*[2], um simulador em Java para simulação de redes. A escolha foi feita baseado nos simuladores de rede apresentados em [33]. Este simulador foi escolhido por já ter uma implementação do protocolo Chord, além de ser facilmente extensível.

O peersim provê uma infra-estrutura básica para a simulação, permitindo a sua extensão através do uso de interfaces para a implementação de protocolos. As estatísticas

[2]http://peersim.sourceforge.net/

acerca da simulação podem ser coletadas pela criação de controles (*Control*), classes[3] que são executadas periodicamente. A configuração da simulação é feita a partir de um arquivo, que pode ser estendido para acomodar as configurações dos protocolos e controles adicionados.

O peersim permite dois tipos de simulação: simulação por ciclos e por eventos. Ele não provê uma implementação para as camadas abaixo do Chord, emulando estes protocolos com um modelo de atraso.

Em cima do protocolo Chord, foram implementados os detalhes referentes ao Arpeggio [11], e do sistema de reputação. Entretanto, nem todos os detalhes do Arpeggio foram utilizados, por não serem interessantes à simulação do sistema de reputação. Foi implementado do Arpeggio somente o índice de arquivos por metadados, não sendo implementados os índices por conjunto de metadados, a replicação de índices e as subredes Chord para a cópia dos arquivos, dados que estas características não interferem no funcionamento dos sistema de reputação.

Para a modelagem da topologia do sistema de reputação, foram utilizados os dados apresentados em [42, 26]. Nestes artigos são descritas as configurações de diversos parâmetros de redes P2P, utilizando como referência diversas medições de redes reais. As configurações utilizadas foram:

**Arquivos:** os arquivos foram divididos em categorias, que representam os grupos de interesse dos usuários, como gênero musical, por exemplo. Dentro de um grupo, os arquivos possuem distribuição Zipf[4], que define a popularidade do arquivo dentro da categoria. Os arquivos são identificados pelo par (categoria, popularidade), que representam os metadados do arquivo. Por exemplo, o arquivo *"(3,2)"* representa o segundo arquivo mais popular da terceira categoria. Adicionalmente à modelagem do artigo, foi adicionado o atributo *hash do conteúdo* representando o conteúdo do arquivo, diferenciando as diversas versões que um mesmo arquivo pode ter. Este *hash* é um valor aleatório atribuído na criação do arquivo.

**Distribuição dos arquivos:** as categorias são organizadas por popularidade, através da distribuição Zipf, de forma parecida com os arquivos numa categoria. A cada nó é atribuído um determinado número mínimo $C_{min}$ de categorias. Para cada categoria, é atribuído um peso que denota o grau de interesse na categoria. No artigo do Schlosser [42], este peso é escolhido aleatoriamente e de forma uniforme no intervalo $[0 : 1]$. Porém, na implementação do simulador, foi preferido distribuir os pesos baseados na distribuição Zipf, de forma a garantir que a soma dos pesos dê

[3]No sentido utilizado em Orientação a Objetos

[4]distribuição de probabilidade segundo a qual, numa lista, um elemento aparece com freqüência inversamente proporcional a sua popularidade.

1. O motivo desta escolha foi para que fosse possível determinar a quantidade de arquivos por nó. Observe que a atribuição de pesos para as categorias é independente da popularidade, logo para um nó a categoria com o maior peso pode ser a quinta categoria mais popular, enquanto a a primeira categoria pode ter o menor peso.

**Realização de buscas:** Cada nó inicia uma nova busca de acordo com distribuição de Poisson, que determina o número de buscas iniciadas num determinado período de tempo. Para a implementação no simulador, foi utilizado o fato de que o tempo entre duas chegadas de Poisson com parâmetro $\frac{1}{\lambda}$ segue a distribuição exponencial com parâmetro $\lambda$ para calcular o tempo entre o inicio de duas buscas. Este parâmetro é escolhido uniformemente num intervalo de valores $[\lambda_{min}, \lambda_{max}]$ definido no arquivo de configuração.

A determinação do arquivo a ser baixado é feita escolhendo-se inicialmente a categoria, e posteriormente um arquivo da categoria selecionada. Em ambos os casos, a probabilidade de um elemento ser escolhido é determinado pelo peso deste elemento em relação ao total, sendo utilizado para a categoria o peso atribuído pelo nó, e para os arquivos a popularidade dentro da categoria.

Além das configurações para a simulação da rede P2P, também é necessário definir as configurações para o sistema de reputação. As principais configurações utilizadas são:

**useReputation:** indica se o sistema de reputação será utilizado ou não;

**reputation:** qual a métrica a ser utilizada para o cálculo da reputação dos nos. Cada métrica é configurada de acordo com os parâmetros necessários para o seu funcionamento, conforme descrito na Seção 6.3;

**initialNodeReputation:** a reputação inicial do nó, ao entrar na rede;

**numNodesToQuery:** indica o número de nós cujas opiniões vão ser consultadas, ou 0 para utilizar todas a opiniões disponíveis.

**freeRiders:** indica a porcentagem de *free-riders* na rede.

**goodNodeKind:** determina o comportamento dos nós honestos. Inclui configurações sobre a reputação mínima dos arquivos e dos nós.

**fractionOfBadNodes:** indica a porcentagem de nós maliciosos na rede.

**badNodeKind:** seleciona a estratégia utilizada pelos nós maliciosos.

Todos estes parâmetros são configurados no arquivo de configuração do peersim.

### 7.1.1 Configurações

Para a realização das simulações, foram utilizados alguns valores padrão. A simulação foi executada com 500 nós, durante 30000 unidades de tempo. As métricas foram configuradas de forma que a reputação máxima seja atingida com 20 avaliações boas, e a mínima em 16 avaliações ruins, contando a partir da reputação máxima. Para isso tivemos as seguintes configurações[5]: reputação normal (baseada na distribuição estatística normal, crescimento lento no inicio e no fim, rápido no meio): $\sigma = 3,3$, $\mu = 10^6$; percentual (crescimento da métrica em uma porcentagem do complemento do valor atual): $p = raiseRate = 0,08$; exponencial (inversa à percentual, crescimento lento no inicio e rápido no fim): $n = iterationsToMaxRep = 20$. Em todas as métricas, $q = lowerRate = 0,75$.

A reputação inicial dos nós foi ajustada para $0,2$. Este foi considerado um bom valor, baixo o suficiente para que o nó iniciante seja considerado pouco confiável, mas ainda deixando espaço para que um mau comportamento inicial seja punido.

A quantidade de arquivos foi planejada de forma a se ter uma probabilidade alta de haver todos os arquivos na rede, mantendo ainda uma boa quantidade de arquivos para serem baixados. Não é possível garantir a presença de todos os arquivos pois a escolha dos arquivos é feita de forma aleatória, logo pode ocorrer de algum arquivo nunca ser selecionado. Os arquivos foram divididos em 60 categorias, cada uma com 30 arquivos. Cada arquivo possui 10 versões, 5 ruins e 5 boas. Cada nó possui 300 arquivos, divididos em 15 categorias.

Com esta distribuição de arquivos, temos um total de $60 * 30 = 1800$ versões de arquivos diferentes, e $1800 * 5 = 9000$ versões legítimas e ilegítimas, num total de 18000 arquivos na rede.

O número de cópias de arquivos armazenados em todos os nós é de $500 * 300 = 150000$ arquivos. Logo cada arquivo tem em média $\frac{150000}{1800} \approx 83$ cópias, e cada versão do arquivo possui uma média de $\frac{83}{10} \approx 8$ cópias.

De acordo com [42], a porcentagem de free-riders numa rede é de 25%. Porém, para o funcionamento do sistema de reputação, os *free-riders* não são interessantes, visto que eles contribuem pouco, não alterando significativamente as reputações dos outros nós, nem as suas. Além disso, o fato deles terem pouca participação na rede faz com que a reputação dos nós *free-riders* destoe da reputação dos outros nós, adulterando a média da reputação e tornando-a menos significativa. Foi optado, portanto, por não utilizar free-riders.

Para os nós bons, as reputações mínimas para arquivos e nós são, respectivamente, $0,12$ e $0,1$. Estes valores permitem que um nó seja mau avaliado duas vezes antes de ter a sua opinião ignorada.

[5] nas equações, a primeira variável refere-se à descrita no Capítulo 6, e a segunda variável ao nome utilizado no arquivo de configuração.

[6] considerando que $f_{normal}(\mu + 3 * \sigma) \approx 1$

### 7.1.2 Modelos de Atacantes

Com o objetivo de se testar diversos aspectos do sistema de reputação, foram considerados quatro modelos de atacantes. Estes modelos utilizam técnicas distintas para induzir os nós a baixarem arquivos corrompidos.

Os atacantes variam seu comportamento nos seguintes aspectos: inicio de download (escolha de arquivo a ser baixado), avaliação de arquivo (a opinião do nó sobre o arquivo), o tipo de arquivo que o nó possui no inicio da simulação (legítimos ou ilegítimos), a avaliação dada aos nós que forneceram opinião, se o nó mantém o arquivo baixado ou não e a escolha da cópia do arquivo a ser baixada.

Os quatro modelos de atacantes são:

**Nunca Coopera:** todos os arquivos que este nó fornece são ruins, e as opiniões são invertidas (avalia bem nós que fornecem avaliação ruim e vice-versa). Ele baixa arquivos de qualquer nó. Este tipo de ataque é utilizado como um teste básico, visto que o nó sempre tem um comportamento ruim;

**Coopera r% das vezes:** em r% dos casos, comporta-se como um nó honesto, e nos outros (1-r)% dos casos age como o atacante do tipo *Nunca coopera*. r% dos arquivos armazenados são bons. Espera-se que as vezes na qual foi fornecido um bom comportamento aumente a reputação do nó, de forma a induzir os nós honestos a baixarem os arquivos corrompidos;

**Sempre Provê Boas Avaliações:** todos os arquivos armazenados por este nó são ruins, e todas as avaliações são positivas, independente da qualidade do arquivo. Funciona de forma parecida com o *coopera r% das vezes*, mas tenta aumentar a sua reputação enviando avaliações boas;

**Aumenta a Reputação do Grupo:** os nós maliciosos atuam em conluio. Os nós maliciosos possuem apenas arquivos ruins. A reputação dos nós do conluio é aumentada com os nós baixando arquivos entre si, e provendo avaliações positivas para os arquivos ruins baixados.

## 7.2 Resultados

Com base na configuração descrita acima, foram feitos diversos experimentos para determinar o funcionamento do sistema de reputação. Inicialmente foram realizados experimentos sem nenhum sistema de reputação atuando. Estes valores serão utilizados como base para determinar a eficiência das diferentes métricas.

A avaliação da eficiência é feita utilizando-se 4 medidas: reputação nos nós, reputação dos arquivos, porcentagem de downloads ruins e número médio de arquivos ruins até um bom.

A *reputação dos nós* é calculada como dois gráficos, uma para os nós honestos e outro para os maliciosos. Os valores nos gráficos são a média dos nós que possuem aquele tipo de comportamento. Assim como os valores de reputação do qual é média, esta métrica varia entre [0:1]. Quanto melhor for o sistema, maiores devem ser a média dos nós bons, e menores a dos nós maliciosos, e consequentemente maior a distância entre as médias.

De forma semelhante, a *reputação dos arquivos* contabiliza as médias separadas entre arquivos bons (legítimos) e ruins (ilegítimos). O cálculo da reputação dos arquivos é feito da mesma forma que um nó faria, utilizando-se todos os nós que possuem opinião. Esta métrica pode não ser muito precisa, devido ao fato dos nós honestos filtrarem as opiniões de nós cuja reputação esteja abaixo de um nível, o que não ocorre no cálculo desta métrica.

A *porcentagem de arquivos ruins* e o *número médio de arquivos ruins até um bom* é medida somente dos nós bons. Na primeira medida, temos a média da porcentagem de arquivos ruins baixados por cada nó. No segundo caso, temos o número médio de arquivos ruins que são baixados até se achar um arquivo bom, numa determinada busca. Para ambos os casos, quanto menor o valor melhor.

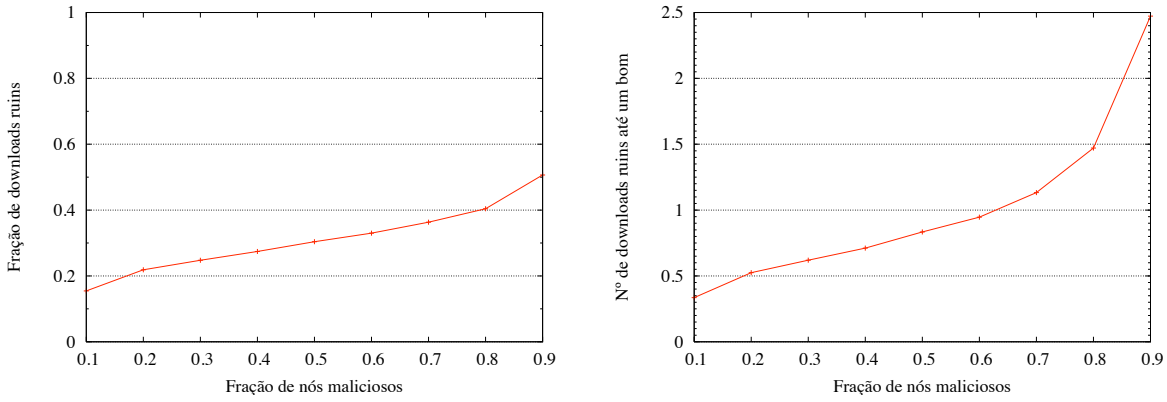
Para melhorar a precisão dos resultados, cada experimento foi executado 10 vezes, com sementes diferentes, e a média destas execuções foi utilizada para o cálculo das medidas. Desta forma, variações específicas de um experimento não interferem no resultado. Cada execução gera como saída diversos arquivos, cada um com os resultados de uma das métricas em função do tempo.

A consolidação destas medições foi feita em duas etapas. Na primeira, calcula-se a média das 10 execuções da mesma configuração, tomando os valores por tempo, obtendo uma média por tempo para a configuração. Na segunda etapa, esta média por tempo é consolidada em um valor para a configuração, calculando a média de todos os valores outrora discriminados por tempo. Na segunda etapa, são desconsiderados os valores das primeiras 5000 unidades de tempo, de forma a desprezar a fase transiente da simulação.

### 7.2.1 Sem Reputação

Os testes sem o uso de reputação foram feitos apenas com o ataque do tipo *Never Cooperate*. Isso foi feito pois, como não há detecção de nós maliciosos, as estratégias utilizadas para enganar os nós honestos não têm efeito. Como não há reputação para discriminar entre os arquivos, escolhe-se a versão do arquivo de forma aleatória. Os resultados são apresentados na Figura 7.1.

Com o sistema de reputação desativado, e s% de nós maliciosos, teoricamente de-




(a) (b)

Figura 7.1: Resultados da simulação sem sistema de reputação

veríamos ter s% de downloads ruins. Porém, algumas características da rede impedem isso:

Os nós bons baixam quantos arquivos forem necessários para achar um bom, e só mantêm este último. Dessa forma, cada início de download termina com o download de um arquivo bom (exceto nos raros casos onde não houver um arquivo bom), gerando um número maior de downloads positivos. Por exemplo, se $X$ arquivos forem pedidos, e em todos os downloads a primeira versão for ruim, a porcentagem de downloads ruins será $\frac{X_{ruins}}{X_{ruins}+X_{bons}} = 0,5$, e não $1,0$ como seria esperado. Além disso, no fim dos downloads a rede possuirá $X$ arquivos bons a mais, e o número de arquivos ruins permaneceria o mesmo, diminuindo a porcentagem.

Os nós maliciosos, por outro lado, não possuem um comportamento análogo; eles simplesmente ficam com o primeiro arquivo, independente dele ser bom ou ruim. Logo o número de arquivos ruins não aumenta na mesma proporção que o de bons. Por conta destes dois motivos, a porcentagem de downloads ruins na Figura 7.1(a) não cresce na mesma proporção que a porcentagem de nós ruins, ficando ligeiramente abaixo.

A Figura 7.1(b) mostra a evolução no número médio de arquivos ruins baixados por download. Com 10% de nós maliciosos, temos aproximadamente 0,35 arquivos ruins por download, o que quer dizer que na média os nós honestos baixam um arquivo ruim a cada três downloads bons, aproximadamente. A marca de 1 download ruim para cada bom é atingida com 60% de nós maliciosos, e com 90% de nós maliciosos temos 2,5 downloads ruins por arquivo.

### 7.2.2 Com Reputação

Serão examinadas agora as simulações realizadas com o sistema de reputação ativo. Para cada um dos 4 modelos de atacante foram rodadas simulações com as 3 métricas a serem consideradas, num total de 12 casos diferentes. Para cada um destes casos, variou-se a porcentagem de nós maliciosos entre 10% e 90%, com passo de 10%, sendo realizadas 9 execuções diferentes para os 12 casos, resultando em 108 configurações diferentes. Cada uma dessas configurações foi rodada 10 vezes, cada uma com uma semente diferente para os números pseudo-aleatórios.

Os resultados foram agrupados por ataque, mostrando as três métricas referentes ao mesmo ataque em um gráfico. Cada gráfico mostra, para um determinado ataque, a variação da métrica pela porcentagem de nós maliciosos. Apesar de se perder os detalhes da variação das métricas pelo tempo com essa abordagem, ganha-se com a redução no número de gráficos para a análise.

#### Nunca Coopera

Serão analisados os resultados do ataque mais básico, onde os nós maliciosos são mais facilmente identificáveis.

Observando a variação da reputação dos nós (Figura 7.2(a)), percebe-se que com até 30% de nós maliciosos para as métricas percentual e exponencial, e 40% para a normal ocorre uma queda grande na reputação dos nós honestos, seguida de uma estabilização a partir daí. A reputação dos nós maliciosos sempre se mantêm baixa, aumentando ligeiramente nas proximidades de 90% de nós maliciosos.

Apesar da reputação dos honestos se aproximar bastante da reputação dos maliciosos para uma grande quantidade de atacantes, ela se mantém sempre acima, sendo possível diferenciar entre os nós. A reputação dos arquivos (Figura 7.2(b)), porém, decai a ponto de não ser mais distinguível. Este valor de reputação, entretanto, não reflete necessariamente a visão individual dos nós, visto que estes filtram as notas provenientes dos nós abaixo de uma determinada reputação. Isto explica o porque de, apesar da suposta não diferenciação entre os arquivos bons e ruins, a porcentagem de downloads ruins (Figura 7.2(c)) é consideravelmente mais baixa quando comparado com a simulação sem uso de reputação, mantendo-se próximo de 0% até uns 60%, contra até 50% do pior caso sem reputação. O número de downloads ruins até um bom (Figura 7.2(d)) cai de 2,5 no pior caso (Figura 7.1(b)) para valores entre 2 (com métrica exponencial) e 1,3 (percentual), sendo que para um número de nós maliciosos abaixo de 80% o sistema de reputação manteve o número médio de downloads ruins até um bom abaixo de 0,1, ou seja, são baixados em média 10 arquivos bons até baixar um ruim.

Comparando as métricas entre si, vê-se que a função normal obtém um resultado

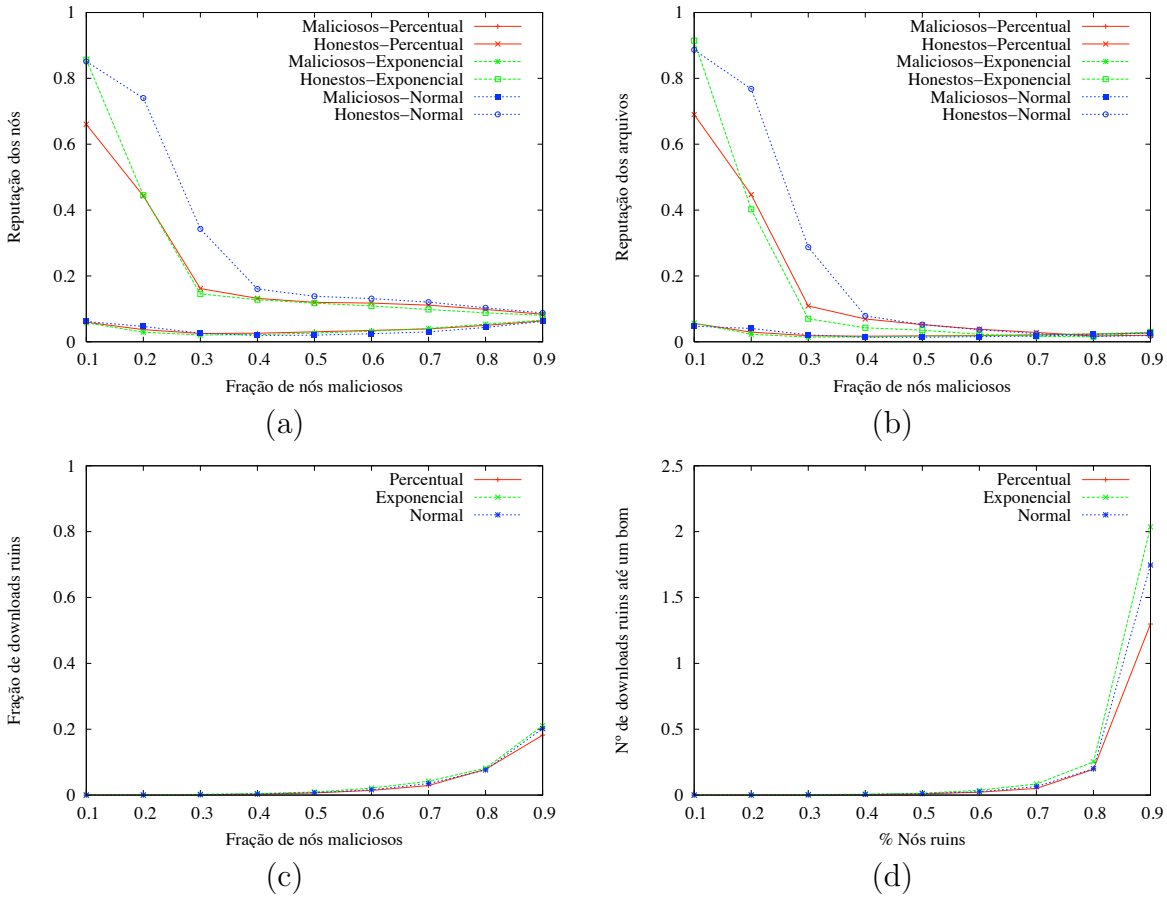



Figura 7.2: Resultados da simulação para o ataque Nunca Coopera

melhor que as outras métricas para até 40% de nós maliciosos. As outras funções tem um comportamento bastante parecido entre si.

### Coopera R% das Vezes

Este ataque é mais elaborado que o anterior, visto que os nós maliciosos tentam enganar os nós honestos fornecendo arquivos bons algumas vezes. Para estas simulações, foi utilizado $p = 50\%$.

Este caso mostrou uma maior diferença entre a reputação dos nós e a dos arquivos, assim como uma maior variação entre as métricas. Na Figura 7.3(a) nota-se que a diferença de reputação entre os nós honestos e maliciosos se mantém alta, mesmo com 90% de nós maliciosos. A reputação dos arquivos também provê uma boa distinção entre os tipos de arquivos, conforme a Figura 7.3(b). A função normal mais uma vez apresenta um resultado melhor que as outras métricas.

A porcentagem de downloads ruins (Figura 7.3(c)) é menor que a registrada no ataque

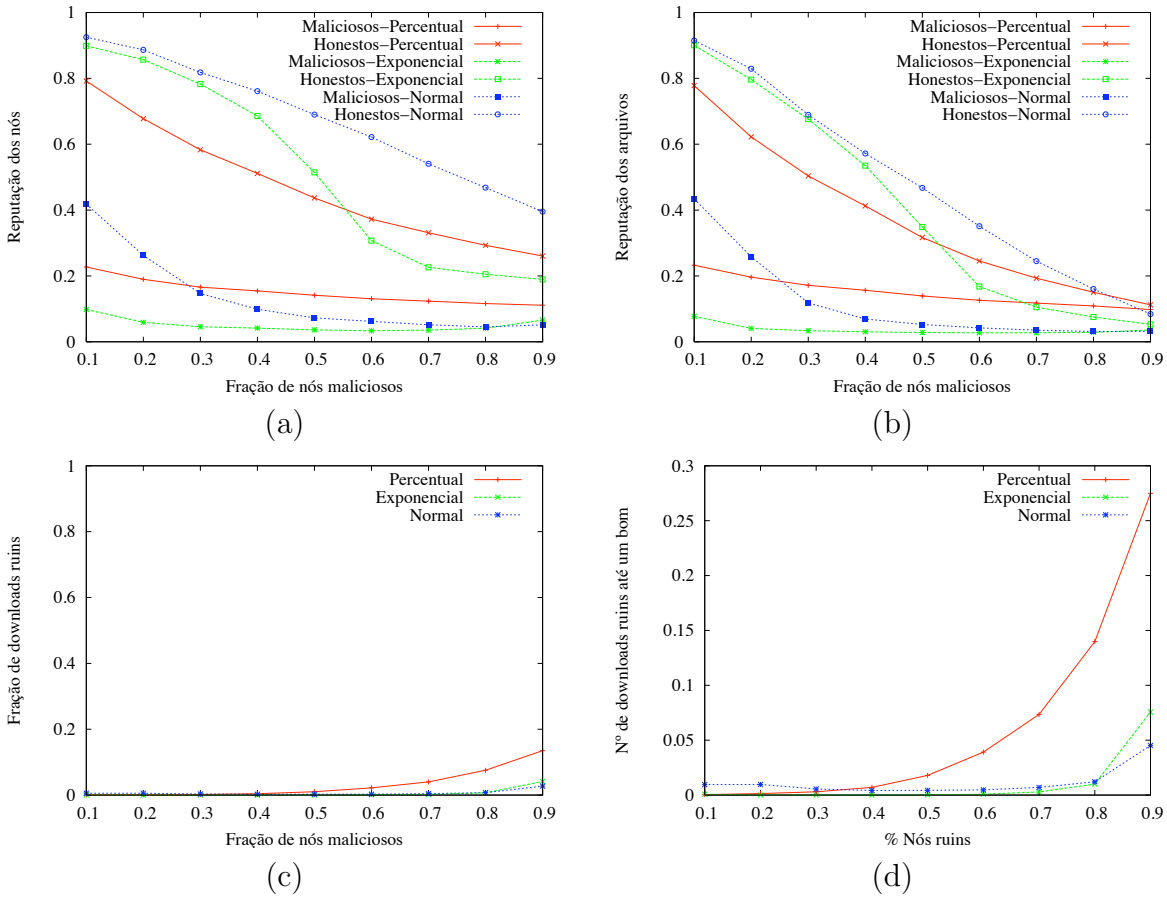



Figura 7.3: Resultados da simulação para o ataque Coopera r% das vezes

*Nunca Coopera*, o que também pode ser visto na figura 7.3(d). Este ataque apresenta um número médio de arquivos ruins até um bom bem abaixo do ataque anterior. De acordo com estes dois gráficos, a métrica *log* mostrou um desempenho pior que as outras.

Apesar de ser um ataque mais elaborado, este ataque se mostrou menos danoso ao sistema. Pode-se creditar isso a dois fatos: em primeiro lugar, a perda de reputação se dá numa velocidade maior que o ganho de reputação, logo a reputação dos nós maliciosos tende a cair; em segundo lugar, colaborando 50% das vezes, os nós maliciosos fornecem naturalmente menos arquivos ruins à rede.

### Sempre Boas Avaliações

Este ataque funciona no estilo do *Coopera R% da Vezes*, cooperando algumas vezes na emissão de opiniões positivas para os arquivos bons de modo a aumentar a reputação. Entretanto, não se tem uma porcentagem fixa para a quantidade de cooperação.

Conforme pode ser notado nos gráficos de reputação dos nós e dos arquivos (Figuras

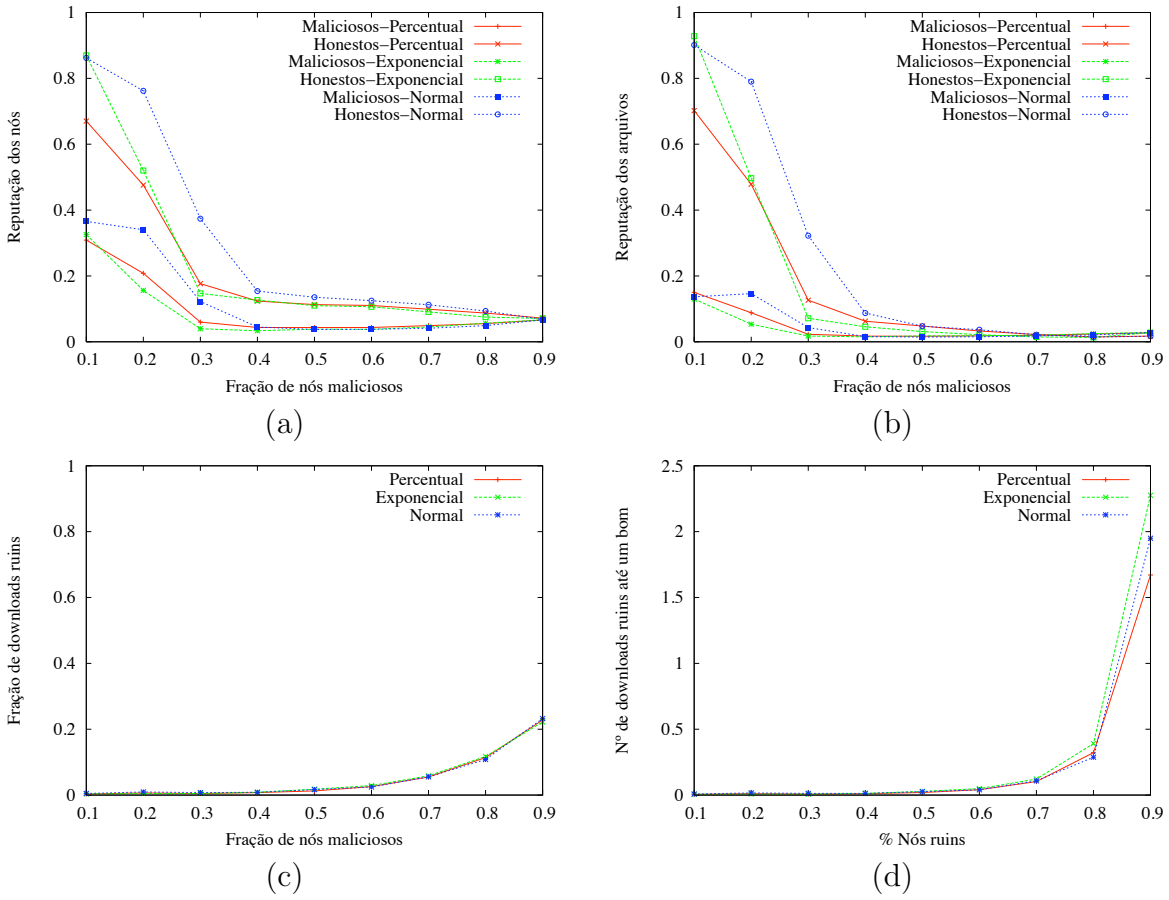



Figura 7.4: Resultados da simulação para o ataque Sempre Boas Avaliações

7.4(a) e (b)), sempre fornecer boas avaliações é um ataque mais danoso que cooperar r% das vezes. Em ambos os gráficos, a diferença de reputação entre os nós decai consideravelmente já com 40% de nós maliciosos. A diferença de reputação dos nós se mantém para porcentagens maiores.

A porcentagem de downloads ruins (Figura 7.4(c)) mantém-se próxima de 0% com até 50% de nós maliciosos subindo para até 20% com 90% de nós ruins, um número próximo aos resultados dos outros ataques. O número médio de downloads ruins até um bom (Figura 7.4(d)), entretanto, atinge valores mais altos, chegando a passar de 1 para 90% de nós maliciosos.

Em todos os gráficos, as métricas comportam-se de forma bastante parecida entre si, exceto pela métrica normal, que apresenta um resultado melhor nos gráficos de reputação dos nós e arquivos (Figuras 7.4(a) e (b)).

### Aumenta a Reputação do Grupo

Este ataque de conluio é o mais elaborado dos ataques considerados. Neste, os nós maliciosos atuam em grupos, realizando trocas de arquivos entre si para espalhar os arquivos ruins e ter uma prova de transação para que seja possível inflar suas reputações. A simulação foi realizada com 2 grupos de nós[7].

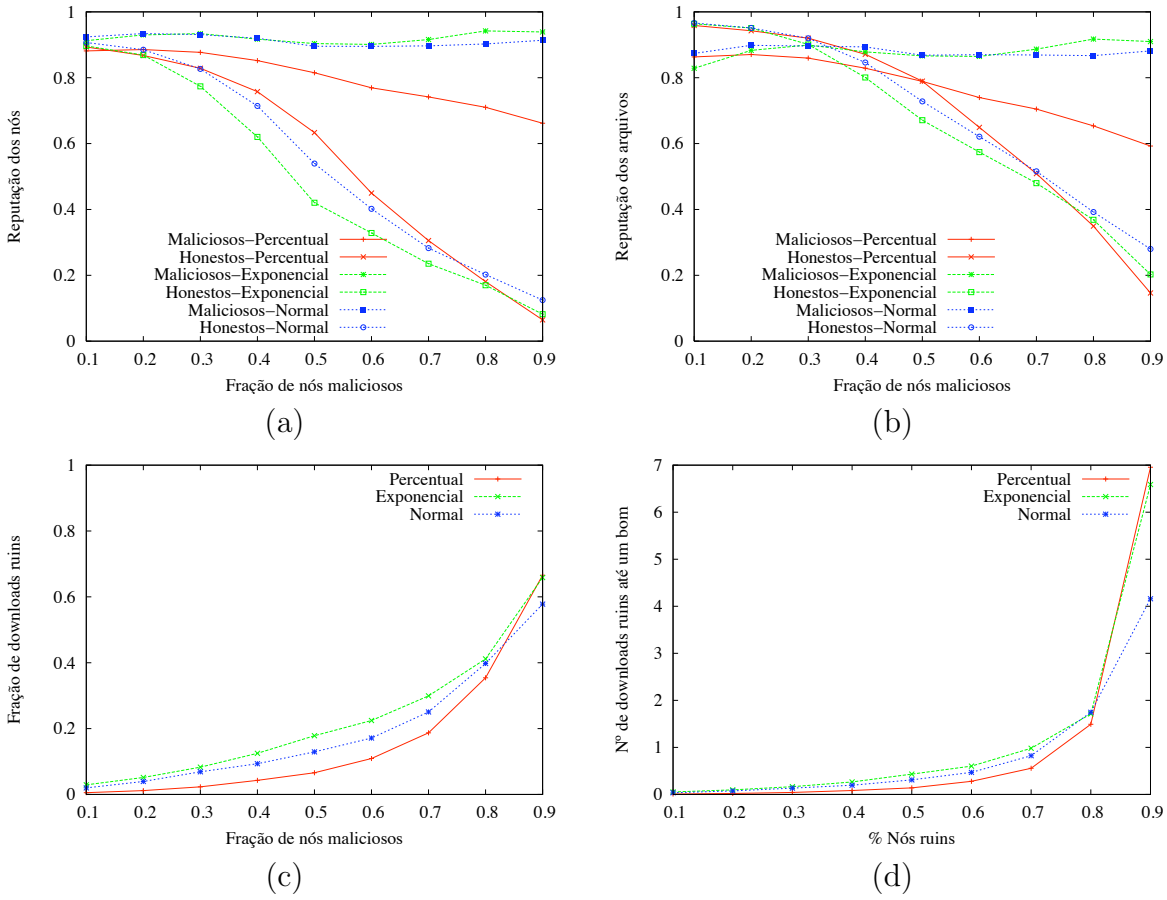



Figura 7.5: Resultados da simulação para o ataque Aumenta a Reputação do Grupo

O ataque de conluio é o ataque mais devastador dentre os considerados. O conluio consegue igualar a reputação dos nós honestos já com 10% de nós maliciosos (Figura 7.5(a)) em 0,9, e a partir dai a reputação dos nós honestos cai constantemente até atingir 0,1, enquanto a reputação dos nós maliciosos se mantém acima de 0,6, mesmo no pior caso.

A reputação dos arquivos resiste mais ao ataque, com a reputação dos arquivos bons sendo superada pela dos ruins apenas com 30% de nós maliciosos (exceto para a métrica

[7]os grupos de nós não atuam em conjunto; apenas os nós do mesmo grupo que colaboram entre si.

percentual, onde isso ocorre aos 50%). Após este valores os arquivos ruins passam a ter uma reputação melhor.

Tal comportamento nos valores de reputação tem efeito na quantidade de arquivos ruins baixados. A porcentagem de arquivos ruins (Figura 7.5(c)), que nos outros ataques chegavam no máximo a 20%, agora chega a este valor com aproximadamente 50% de nós maliciosos. Com 90%, a quantidade de arquivos ruins chega a mais de 60%. O número médio de arquivos ruins baixados (Figura 7.5(d)) chega a 6, ultrapassando em muito os valores atingidos nos outros ataques, de no máximo 2,2.

Comparando o ataque de conluio com os outros, verifica-se que este é o mais danoso, subvertendo o sistema. Comparando com o sistema sem reputação, entretanto, conseguimos verificar uma melhora discreta na porcentagem de arquivos ruins baixados. Em compensação, o número médio de downloads ruins até um bom situa-se bem acima do caso sem reputação. Esse aumento ocorre pois há uma diferença de comportamento entre o ataque realizado no caso sem reputação e no conluio: os atacantes em conluio copiam arquivos maliciosos entre si, o que aumenta o número de cópias destes arquivos, assim como o número de avaliações positivas destes. Esses aumentos são os responsáveis pela maior quantidade de cópias ruins na rede, e ao consequente aumento no número de downloads ruins.

A métrica que se saiu melhor neste ataque foi a percentual, abaixando mais a nota dos nós maliciosos.

## 7.3 Comparação Com Outros Sistemas

Conforme visto na seção anterior, o sistema de reputação proposto diminui a eficiência da maioria dos ataques, obtendo um resultado ruim apenas em ataques de conluio. Para uma melhor avaliação, convém comparar os resultados obtidos com os resultados dos sistemas de reputação estudados. Tal comparação sofre uma limitação pois cada sistema de reputação utiliza um conjunto diferente de métricas. Algumas métricas, entretanto podem ser comparadas, por medirem valores passíveis de conversão.

O sistema proposto por Marti [30] possui a métrica *taxa de verificação (verification ratio)*, que realiza uma medição equivalente ao número médio de downloads ruins até um arquivo bom, diferindo por uma unidade. Analisando os gráficos considerando *whitewashing* e as opiniões dos outros nós, a taxa de verificação varia entre 2 e 4, para a porcentagem de nós maliciosos variando de 10% a 50%. Comparando com os resultados obtidos pelo sistema proposto, temos numa métrica equivalente, uma variação entre 1 e 2, obtendo um desempenho ligeiramente melhor.

O EigenTrust [24] fornece métricas sobre o número de downloads inautênticos (ruins) para os casos de atacante individual (estilo *Nunca coopera*), conluio e Coopera p% das

vezes com 37% de nós maliciosos (cálculo aproximado). No caso dos atacantes individuais, os dois sistemas possuem uma eficiência similar, com o sistema proposto com uma porcentagem menor de downloads ruins. Para atacantes em conluio, porém, o *EigenTrust* possui uma grande vantagem em relação ao sistema proposto, sendo capaz de manter a quantidade de downloads ruins num nível baixo, enquanto o mesmo ataque no sistema proposto subverte a rede. Entretanto, no ataque de *Coopera r% das vezes*, o sistema proposto foi capaz de impedir o ataque de forma mais eficaz que o *EigenTrust*, visto que este último teve até 30% de downloads ruins.

Os resultados obtidos pelo sistema PeerTrust [49] foram próximos aos obtidos pelo sistema proposto. A métrica utilizada pelo Xiong é a de porcentagem de transações com sucesso, que é a inversa da porcentagem de downloads ruins. Para os casos sem conluio, a taxa de sucesso foi de praticamente 100%, resultado parecido com os obtidos no sistema proposto, de quase 0% de downloads ruins. Para os casos de conluio, o comportamento do PeerTrust foi dependente da variação do sistema utilizado: para o caso de utilizar diretamente as reputações dos outros nós, o sistema obteve uma taxa de sucesso de 0%; mas na variação onde as opiniões dos outros nós eram ajustadas pela opinião pessoal do nó, a taxa de sucesso foi de 100%. O sistema proposto obteve desempenho intermediário em relação às duas variações do PeerTrust analisadas, com a taxa de sucesso decaindo gradativamente com o aumento do número de nós maliciosos, até um mínimo de 40% de sucesso.

## 7.4 Conclusões

Neste capítulo foi feita a avaliação do sistema de reputação. Diversos métodos foram considerados, sendo escolhida a simulação, pela facilidade de execução e resultados confiáveis. O modelo de simulação utilizado foi o de simulação por eventos, sem emulação das camadas inferiores da arquitetura de rede.

A simulação foi feita de acordo com modelos de redes descritos em artigos, artigos estes baseados em medições realizadas em redes reais. Dos artigos foram obtidas especificações sobre caracterização e distribuição dos arquivos entre os nós, buscas entre outros.

Os resultados mostram que o sistema de reputação proposto é capaz de diminuir a quantidade de arquivos ruins baixados em quase todos os casos, exceto nos casos de conluio. Comparando com os resultados obtidos em outros artigos, nota-se que os resultados foram próximos aos obtidos pelos outros sistemas, sendo ligeiramente melhor em alguns casos. De um modo geral, os resutados foram melhores quando comparados a sistemas de reputação que não utilizam redes de confiança, como o do Marti e o PeerTrust, sendo piores que os apresentados pelo EingenTrust, que implementa redes de confiança. Ainda assim, O EigenTrust apresentou resultados melhores apenas no caso de conluio, tendo

inclusive apresentado resultados piores que o sistema proposto no ataque de *Coopera r% das vezes*.

Analisando estes resultados, é possível ver que as redes de confiança provêem uma eficácia maior contra ataques de conluio. A verificação em um nível utilizada no sistema proposto provê uma melhora no funcionamento nos casos sem conluio, mas não é suficiente para ser eficiente nos casos sem. Uma potencial linha de pesquisa futura seria sobre os efeitos da adição de níveis adicionais de verificação nos ataques de conluio, se mais alguns níveis seriam suficientes ou se é realmente preciso a rede de confiança completa.

Entre as métricas testadas, a que obteve o melhor comportamento foi a Normal, com resultados melhores em quase todos os ataques, exceto no ataque de conluio, onde a Percentual resistiu mais aos ataques.

# Capítulo 8

# Conclusões

As redes P2P tiveram um crescimento grande nos últimos anos, se consolidando como um dos serviços mais populares da rede. Seu principal uso é para o compartilhamento de arquivos, mas o crescimento da sua aceitação como uma forma confiável de fornecimento de serviços está levando o seu uso a ser considerado inclusive em redes empresariais. Entretanto, a popularidade trouxe também diversos problemas de segurança, problemas estes agravados pelo fato de seu funcionamento ser delegado a nós que podem potencialmente ser maliciosos.

Um dos problemas com os quais as redes P2P de compartilhamento de arquivos têm que lidar é com a presença de arquivos ilegítimos. Estes arquivos se fazem passar por arquivos autênticos, induzindo os usuários a os baixarem erroneamente. A presença destes arquivos gera algumas consequências negativas. Uma delas é a diminuição da disponibilidade dos arquivos legítimos, devido à demora em se encontrá-los. Outra é a desistência por parte dos usuários de utilizar a rede, caso o tempo para realizar um download com sucesso fique muito grande.

A forma usualmente adotada para a detecção dos arquivos ilegítimos são os sistemas de reputação. Estes sistemas utilizam-se das opiniões dos usuários para decidir se um determinado arquivo é bom ou não. As avaliações podem ser sobre o arquivo em si ou sobre o nó que estiver fornecendo-o.

Existem na literatura diversas abordagens para a implementação dos sistemas de reputação. Para o seu estudo, os sistemas de reputação foram divididos em diversos componentes, cada um abordando uma parte do problema, com diversas soluções possíveis. Algumas destas partes são dependentes da rede P2P sob a qual o sistema de reputação está construído.

A solução apresentada foi construída sobre redes estruturadas que implementem DHT. A rede escolhida com esta característica foi a Chord, e foi adicionada as funcionalidades do Arpeggio para a possibilidade de buscas por palavras-chave. Foram utilizados dois

tipos de reputação: a reputação dos arquivos, que indica a qualidade deles e a reputação dos nós, que indica a qualidade da opinião dos mesmos. A infra-estrutura de DHT é utilizada para armazenar a reputação. A reputação dos arquivos é calculada baseada na reputação dos nós e nas opiniões que eles fornecem sobre os arquivos, com esta última sendo armazenada nos próprios nós. Foram utilizadas chaves públicas para a determinação da identidade dos nós.

A validação do sistema proposto foi feita através de simulação. A rede e o sistema de reputação foram implementados utilizando o simulador *peersim*. Diversos tipos de ataques foram simulados, de forma a testar o funcionamento do sistema frente às diversas formas de subversão da rede. A simulação foi feita com a utilização de diversas métricas para o cálculo da reputação dos nós, com o objetivo de de determinar a mais eficiente. Foi realizada também a simulação sem o uso do sistema de reputação, obtendo valores base para a verificação da eficiência do sistema.

Os resultados mostraram que o sistema é capaz de resistir a diversos tipos de ataques, sendo capaz de distinguir entre um nó malicioso e um honesto com até aproximadamente 60% de nós maliciosos nos ataques de *Nunca Coopera*, *Coopera r% das Vezes* e *Sempre Boas Avaliações*. A porcentagem de arquivos maliciosos baixados se manteve em praticamente 0% nestes ataques com até 70% de nós maliciosos. O ataque mais efetivo foi o de *Conluio*, que foi capaz de subverter a rede mesmo com uma pequena porcentagem de nós. Ainda assim, para uma pequena quantidade de nós maliciosos, foi possível diminuir a porcentagem de arquivos ruins baixados.

A comparação com outros sistemas mostrou que o sistema proposto desempenha ligeiramente melhor que os sistemas de reputação sem redes de confiança, mas o desempenho e bem pior quando comparado com um sistema de reputação, principalmente no caso de conluio. Um dos possíveis fatores determinantes para tal diferença de desempenho é a rede de confiança, que realiza a verificação a reputação de todos os nós que forneceram alguma opinião. Essas verificações impedem o mecanismo de funcionamento do conluio, pois a reputação dos nós que dão avaliações boas aos nós ruins também é considerada, impedindo a inflação da reputação.

A investigação sobre como melhorar a detecção de ataques de conluio é um caminho natural a seguir em trabalhos futuros. Redes de confiança são custosas de operar, e um nível de verificação melhorou o funcionamento, mas não o suficiente para o sistema de reputação resistir à ataques de conluio. A adição de alguns níveis de verificação pode melhorar o funcionamento sem aumentar os custos demasiadamente.

Além dos casos de conluio, outras melhorias podem ser feitas no sistema proposto. Diversos parâmetros utilizados nos testes foram definidos de forma empírica, baseado em informações obtidas na bibliografia. Exemplos incluem os parâmetros de funcionamento das métricas, e as reputações mínima para os arquivos e os nós. Nestes últimos, pode-

se estudar a possibilidade de se utilizar algoritmos adaptativos, ajustando os limites de acordo com as experiências positivas ou negativas dos nós. A inclusão de componentes ao cálculo da métrica, como o tamanho da transação realizada, pode contribuir também para a melhoria da detecção de comportamento malicioso.

Outro trabalho futuro possível é a de verificação da utilização de recursos para a manutenção do sistema, como o uso de banda, número de mensagens trocadas e o tempo necessário para obter as informações para o cálculo da reputação (latência adicionada ao sistema). Estas métricas podem se mostrar bastante interessantes para analisar os custos que a implantação do sistema de reputação adicionaria ao tráfego. As métricas de desempenho também poderiam ser utilizadas na avaliação do *overhead* do uso de uma rede de confiança.

As redes P2P se estabilizaram como mais um serviço na rede, com seu uso sendo considerado para outras finalidades além da troca de arquivos. Mas essa expansão para ambientes onde há mais necessidade de garantia de funcionamento é dependente de mecanismos que garantam o funcionamento da rede. O desenvolvimento de mecanismos de segurança tornam a rede mais segura, tornando a rede mais resiliente a ataques e a mau funcionamento, provendo a garantia de funcionamento requerida para futuras expansões.

# Referências Bibliográficas


[1] Karl Aberer and Zoran Despotovic. Managing trust in a peer-2-peer information system. In *CIKM '01: Proceedings of the tenth international conference on Information and knowledge management*, pages 310–317, New York, NY, USA, 2001. ACM.

[2] Karl Aberer and Manfred Hauswirth. Overview on peer-to-peer information systems, 2002.

[3] Stephanos Androutsellis-Theotokis and Diomidis Spinellis. A survey of peer-to-peer content distribution technologies. *ACM Comput. Surv.*, 36(4):335–371, 2004.

[4] Robert M. Axelrod. *The Evolution of Cooperation*. Basic Books, 1984.

[5] H Garcia-Molina B Yang, SD Kamvar. Secure score management for p2p systems. Technical report, Stanford University, 2003.

[6] Marinho P. Barcellos and Luciano P. Gaspary. *Segurança em Redes P2P: princípios, tecnologias e desafios*, chapter 5. SBRC, 2006.

[7] Daniel Bauermann, Matheus Lehmann1, Rodrigo Mansilha, and Marinho P. Barcellos. Protegendo bittorrent: projeto e avaliação de contra-medidas eficazes para ataques dos. In *VIII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais*, pages 215–228, 2008.

[8] Miguel Castro, Peter Druschel, Ayalvadi Ganesh, Antony Rowstron, and Dan S. Wallach. Secure routing for structured peer-to-peer overlay networks. *SIGOPS Oper. Syst. Rev.*, 36(SI):299–314, 2002.

[9] Nicolas Christin, Andreas S. Weigend, and John Chuang. Content availability, pollution and poisoning in file sharing peer-to-peer networks. In *EC '05: Proceedings of the 6th ACM conference on Electronic commerce*, pages 68–77, New York, NY, USA, 2005. ACM.

[10] I. Clarke, O. Sandberg, B. Wiley, and T.W. Hong. Freenet: A distributed anonymous information storage and retrieval system. In *International Workshop on Design Issues in Anonymity and Unobservability*, pages 46–66, 2000.

[11] Austin T. Clements, Dan R. K. Ports, and David R. Karger. Arpeggio: Metadata searching and content sharing with chord. In *Peer-to-Peer Systems IV*, 2005.

[12] Fabrizio Cornelli, Ernesto Damiani, Sabrina De Capitani di Vimercati, Stefano Paraboschi, and Pierangela Samarati. Choosing reputable servents in a p2p network. In *WWW '02: Proceedings of the 11th international conference on World Wide Web*, pages 376–386, New York, NY, USA, 2002. ACM.

[13] C. Costa and J. Almeida. Reputation systems for fighting pollution in peer-to-peer file sharing systems. In *Seventh IEEE International Conference on Peer-to-Peer Computing*, pages 53–60, 2007.

[14] Cristiano Costa, Vanessa Soares, Jussara Almeida, and Virgilio Almeida. Combatendo a disseminação de conteúdo poluído em redes par-a-par para compartilhamento de arquivos. In *25o SBRC Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pages 869–881, 2007.

[15] Frank Dabek, M. Frans Kaashoek, David Karger, Robert Morris, and Ion Stoica. Wide-area cooperative storage with cfs. *SIGOPS Oper. Syst. Rev.*, 35(5):202–215, 2001.

[16] Ernesto Damiani, De Capitani di Vimercati, Stefano Paraboschi, Pierangela Samarati, and Fabio Violante. A reputation-based approach for choosing reliable resources in peer-to-peer networks. In *CCS '02: Proceedings of the 9th ACM conference on Computer and communications security*, pages 207–216, New York, NY, USA, 2002. ACM.

[17] Neil Daswani, Hector Garcia-Molina, and Beverly Yang. Open problems in data-sharing peer-to-peer systems. In *ICDT '03: Proceedings of the 9th International Conference on Database Theory*, pages 1–15, London, UK, 2002. Springer-Verlag.

[18] Roger Dingledine. *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*, chapter 16: Accountability. O'Reilly & Associates, Inc., 2001.

[19] Roger Dingledine, Michael J. Freedman, and David Molnar. The free haven project: Distributed anonymous storage service. In *Proceedings of the Workshop on Design Issues in Anonymity and Unobservability*, pages 67–95. Springer-Verlag, 2000.

[20] John R. Douceur. The sybil attack. In *IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems*, pages 251–260, London, UK, 2002. Springer-Verlag.

[21] Michal Feldman, Kevin Lai, Ion Stoica, and John Chuang. Robust incentive techniques for peer-to-peer networks. In *EC '04: Proceedings of the 5th ACM conference on Electronic commerce*, pages 102–111, New York, NY, USA, 2004. ACM.

[22] Eric Friedman and Paul Resnick. The social cost of cheap pseudonyms, 2001.

[23] Minaxi Gupta, Paul Judge, and Mostafa Ammar. 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, 2003. ACM.

[24] Sepandar D. Kamvar, Mario T. Schlosser, and Hector Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pages 640–651, New York, NY, USA, 2003. ACM.

[25] David Karger, Eric Lehman, Tom Leighton, Rina Panigrahy, Matthew Levine, and Daniel Lewin. Consistent hashing and random trees: distributed caching protocols for relieving hot spots on the world wide web. In *STOC '97: Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 654–663, New York, NY, USA, 1997. ACM.

[26] Ramayya Krishnan, Michael D Smith, and Rahul Telang. The economics of peer-to-peer networks. *Journal of Information Technology Theory and Application*, 5(3):31–44, 2003.

[27] Chu Yee Liau, Xuan Zhou, Stéphane Bressan, and Kian-Lee Tan. Efficient distributed reputation scheme for peer-to-peer systems. In *Web and Communication Technologies and Internet-Related Social Issues - HSI*, pages 54–63. Springer Berlin / Heidelberg, 2003.

[28] Eng Keong Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim. A survey and comparison of peer-to-peer overlay network schemes. *Communications Surveys & Tutorials, IEEE*, 7(2):72–93, Quarter 2005.

[29] Sergio Marti and Hector Garcia-Molina. Identity crisis: Anonymity vs. reputation in p2p systems. In *P2P '03: Proceedings of the 3rd International Conference on*

*Peer-to-Peer Computing*, page 134, Washington, DC, USA, 2003. IEEE Computer Society.

[30] Sergio Marti and Hector Garcia-Molina. Limited reputation sharing in p2p systems. In *EC '04: Proceedings of the 5th ACM conference on Electronic commerce*, pages 91–101, New York, NY, USA, 2004. ACM.

[31] Sergio Marti and Hector Garcia-Molina. Taxonomy of trust: categorizing p2p reputation systems. *Comput. Networks*, 50(4):472–484, 2006.

[32] Petar Maymounkov and David Mazières. Kademlia: A peer-to-peer information system based on the xor metric. In *Peer-to-Peer Systems*, pages 53–65. Springer-Verlag, 2002.

[33] Stephen Naicken, Anirban Basu, Barnaby Livingston, and Sethalat Rodhetbhai. A survey of peer-to-peer network simulators. *Proceedings of the 7th Annual Postgraduate Symposium (PGNet '06)*, 2006.

[34] B. OOI and C. LIAU. Managing trust in peer-to-peer systems using reputation-based techniques, 2003.

[35] Martin Raab and Angelika Steger. "balls into bins- a simple and tight analysis. In *Randomization and Approximation Techniques in Computer Science*, pages 159–170. Springer Berlin / Heidelberg, 1998.

[36] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Schenker. A scalable content-addressable network. *SIGCOMM Comput. Commun. Rev.*, 31(4):161–172, 2001.

[37] Paul Resnick, Ko Kuwabara, Richard Zeckhauser, and Eric Friedman. Reputation systems. *Commun. ACM*, 43(12):45–48, 2000.

[38] John Risson and Tim Moors. Survey of research towards robust peer-to-peer networks: Search methods. *Computer Networks*, 15(17):3485–3521, 5 December 2006.

[39] D. Rosenthal, M. Roussopoulos, P. Maniatis, and M. Baker. Economic measures to resist attacks on a peer-to-peer network, 2003.

[40] Sini Ruohomaa and Lea Kutvonen. Trust management survey. In *Proceedings of the iTrust 3rd International Conference on Trust Management*, volume 3477/2005, pages 77–92. Springer Berlin / Heidelberg, 2005.

[41] Audun Jøsang, Roslan Ismail, and Colin Boyd. A survey of trust and reputation systems for online service provision. *Decis. Support Syst.*, 43(2):618–644, 2007.

[42] Mario Schlosser and Sepandar Kamvar. Simulating a file-sharing p2p network. Technical report, Stanford University, 2002.

[43] A. Singh, T.-W. Ngan, P. Druschel, and D. S. Wallach. Eclipse attacks on overlay networks: Threats and defenses. *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, pages 1–12, April 2006.

[44] Emil Sit and Robert Morris. Security considerations for peer-to-peer distributed hash tables. In *IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems*, pages 261–269, London, UK, 2002. Springer-Verlag.

[45] Mudhakar Srivatsa and Ling Liu. Vulnerabilities and security threats in structured overlay networks: A quantitative analysis. In *ACSAC '04: Proceedings of the 20th Annual Computer Security Applications Conference (ACSAC'04)*, pages 252–261, Washington, DC, USA, 2004. IEEE Computer Society.

[46] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. *SIGCOMM Comput. Commun. Rev.*, 31(4):149–160, 2001.

[47] Kevin Walsh and Emin Gün Sirer. Fighting peer-to-peer spam and decoys with object reputation. In *P2PECON '05: Proceeding of the 2005 ACM SIGCOMM workshop on Economics of peer-to-peer systems*, pages 138–143, New York, NY, USA, 2005. ACM.

[48] Kevin Walsh and Emin Gün Sirer. Thwarting p2p pollution using object reputation. Technical report, Cornell University, 2005.

[49] Li Xiong and Ling Liu. Peertrust: Supporting reputation-based trust for peer-to-peer electronic communities. *IEEE Transactions on Knowledge and Data Engineering*, 16(7):843–857, 2004.

[50] Bin Yu and Munindar P. Singh. A social mechanism of reputation management in electronic communities. In *CIA '00: Proceedings of the 4th International Workshop on Cooperative Information Agents IV, The Future of Information Agents in Cyberspace*, pages 154–165, London, UK, 2000. Springer-Verlag.

[51] Philip R. Zimmermann. *The Official PGP User's Guide*. MIT Press, 1995.