PONTIFÍCIA UNIVERSIDADE CATÓLICA DO PARANÁ
CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

NEANDER LARSEN BRISOLA

UM MODELO BASEADO EM CHAVES PÚBLICAS PARA PROVER AUTENTICIDADE DE CONTEÚDO, CONTROLE DE ACESSO E REPUTAÇÃO EM REDES PEER-TO-PEER

Dissertação apresentada ao Programa de Pós-Graduação em Informática da Pontifícia Universidade Católica do Paraná como requisito parcial para obtenção do título de Mestre em Informática.

CURITIBA
2007

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO PARANÁ
CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

NEANDER LARSEN BRISOLA

UM MODELO BASEADO EM CHAVES PÚBLICAS PARA PROVER AUTENTICIDADE DE CONTEÚDO, CONTROLE DE ACESSO E REPUTAÇÃO EM REDES PEER-TO-PEER

Dissertação apresentada ao Programa de Pós-Graduação em Informática da Pontifícia Universidade Católica do Paraná como requisito parcial para obtenção do título de Mestre em Informática.

Área de Concentração: Ciência da Computação

Orientador: Prof. Dr. Altair Olivo Santin

Co-orientador: Prof. Dr. Lau Cheuk Lung

CURITIBA
2007

BRISOLA, Neander Larsen

Um Modelo Baseado em Chaves Públicas para prover Autenticidade de Conteúdo, Controle de Acesso e Reputação em redes Peer-to-Peer. Curitiba, 2007.

Dissertação de Mestrado – Pontifícia Universidade Católica do Paraná.
Programa de Pós-Graduação em Informática.

1. Redes Peer-to-Peer 2. Protocolos JXTA 3. SPKI/SDSI. 4. Sistemas Distribuídos.
I.Pontifícia Universidade Católica do Paraná. Centro de Ciências Exatas e de Tecnologia. Programa de Pós-Graduação em Informática II-t

Esta página deve ser reservada à ata de defesa e termo de aprovação que serão fornecidos pela secretaria após a defesa da dissertação e efetuadas as correções solicitadas.

Dedico este trabalho ao meu Deus, à minha linda esposa Andréia e à minha família.

Agradecimentos

Primeiramente, agradeço a Deus por ter me concedido este privilégio de estar concluindo mais uma fase em minha vida, e pela força a qual me susteve durante este processo de mestrado, que a honra seja dada a Ele.

A minha preciosa esposa Andréia, que por muito tempo precisou adiar nossos planos pessoais para que eu pudesse concluir este curso com êxito. Além de todo o acompanhamento em minha caminhada com muita paciência e companheirismo, obrigado por ser uma mulher virtuosa. Quero agradecer ao meu filho Davi, que por muitas vezes deixamos de desfrutar do nosso convívio de pai e filho pelas circunstâncias desta atividade acadêmica.

Agradeço pelo amor, carinho e suporte espiritual dado por meus pais Elias e Zulméia Brisola e aos meus sogros Rubens e Trancilita Monteiro, para que eu pudesse terminar bem esta carreira que me foi proposta. A minha tia Jussemy Monteiro Steinmann, pela sua ajuda dedicando um tempo na leitura desse trabalho.

Não menos importante faço aqui uma menção honrosa a um verdadeiro amigo, Heverson Borba Ribeiro que como colega esteve acima da média, e demonstrou um espírito de luta durante a pesquisa e as barreiras que foram surgindo, sem desistir.

Ao meu amigo Jim Lau sempre prestativo a todo instante e a todos os companheiros de laboratório, com os quais pude aprender e pesquisar além da troca de conhecimento.

Ao Professor Doutor Carlos Maziero, que nunca deixou de me atender e responder as dúvidas que porventura eu tivesse, sempre com muita paciência e principalmente respeitando os meus limites de entendimento, demonstrando uma excelente didática.

Aos professores, que me orientaram em todo o tempo na faculdade, com idéias, correções e apoio para que o trabalho pudesse ser reconhecido nos eventos internacionais a que foram submetidos e aprovados com êxito. Em especial ao meu orientador Professor Doutor Altair Olivo Santin e meu Co-Orientador Professor Doutor Lau Cheuk Lung, por procurar me direcionar nos caminhos da pesquisa, para que o trabalho alcançasse seu objetivo o qual é uma contribuição no meio científico. E agradeço ao colega Emerson Mello e aos demais membros do grupo do laboratório DAS da UFSC pelo apoio.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

ACL	Access Control List
API	Application Programming Interface
CA	Certificate Authority
CERT	Computer Emergency Response Team
CMS	Content Management System
CORC	Credit Only Reputation Computation
CPU	Central Process Unit
DCRC	Debit-Credit Reputation Computation
DHT	Distributed Hash Table
ESD	Eletronic Software Distribution
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
ID	Identification
IETF	Internet Engineering Task Force
IP	Internet Protocol
ISP	Internet Solution Provider
J2ME	Java 2 Mobile Environment
JXTA	Acrônimo para a Palavra Justapor, no inglês Juxtapose.
MD	Message Digest
MIT	Massachusetts Institute of Technology
OverQoS	Overlay Quality of Service
P2P	Peer-to-Peer
PBP	Pipe Binding Protocol
PDP	Peer Discovery Protocol
PEP	Peer EndPoint Protocol
PIP	Peer Information Protocol
PKI	Public Key Infrastructure

PRP	Peer Resolver Protocol
RBAC	Role-based Access Control
RCA	Reputation Computation Agent
RON	Resilient Overlay Network
RVP	Rendezvous Protocol
SSH	Secure Shell
SSL	Secure Socket Layer
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TTL	Time To Live
TTP	Trusted Third Party
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
WSDL	Web Services Description Languages
XML	Extensible Markup Language

Resumo

No uso clássico de redes *peer-to-peer* (par-a-par / P2P) para compartilhamento de arquivo, principalmente música e filmes, geralmente não há uma preocupação com autenticidade e controle de acesso a conteúdo. Propostas de segurança encontradas atualmente na literatura técnica tentam adaptar técnicas de arquitetura cliente-servidor para o ambiente P2P, o que não parece ser a abordagem mais apropriada. Este trabalho propõe o uso de uma abordagem mais flexível, segura e apropriada para o ambiente P2P, baseada no SDSI/SPKI (*Simple Distributed Security Infrastructure/Simple Public Key Infrastructure*). Na proposta está demonstrado que o uso de chaves públicas para identificar um *peer* permite a criação de um esquema de identificação persistente, sem perder o anonimato, até mesmo em um ambiente auto-gerenciado como o P2P. Além disso, é adotado o uso da assinatura digital para prover autenticidade para o conteúdo P2P. Para prover credibilidade para as chaves públicas usadas pelo SDSI/SPKI, é aplicado um esquema de reputação dos *peers*. A reputação adota uma abordagem baseada no esquema de votação para qualificar autores e provedores de conteúdo. Um esquema também é proposto para garantir irretratabilidade na transferência de conteúdos P2P. A implementação de um protótipo como prova de conceito mostrou a viabilidade da proposta.

Palavras-Chave: Segurança, redes *peer-to-peer*, autenticação, autorização, reputação.

Abstract

In the classic use of P2P network for file sharing, mainly music and movies, there is not the same concern with authenticity and content access control. Security proposals currently found in technical literature attempt to adapt techniques of client-server architecture to the P2P environment, which does not seem to be the most appropriate approach. This work proposes the usage of a more flexible, secure and appropriate approach to the P2P environment, the SDSI/SPKI (Simple Distributed Security Infrastructure/Simple Public Key Infrastructure). It is shown in the proposal that the use of public keys to identify peer allows the creation of a persistent identification scheme, without loosing anonymity, even in a self-managed environment as P2P. In addition, the usage of digital signature to provide authenticity to P2P content is adopted. In order to provide credibility to the public keys used by the SDSI/SPKI, a reputation based approach is applied. The reputation adopts an approach based on the voting scheme to qualify authors and peer-to-peer content servers. A scheme is also proposed in order to guarantee non-repudiation in the transfer of P2P contents. The implementation of a prototype as a proof of concept showed the viability of the proposal.

Keywords: Security, *peer-to-peer*, authentication, authorization, reputation.

Capítulo 1

Introdução

1.1 Motivação

Nos últimos anos, os sistemas computacionais baseados em redes *peer-to-peer* (par-a-par) vêm adquirindo considerável aceitação não somente entre os usuários domésticos, mas também no meio acadêmico e corporativo. O principal atrativo destas redes é seu suporte distribuído a aplicações para atender demandas destes ambientes.

Nesses sistemas, os *peers* presentes na rede podem atuar ao mesmo tempo como cliente (consumidor) ou servidor (provedor) de recursos, diferentemente da arquitetura cliente-servidor clássica, no qual um nó só pode desempenhar nativamente um dos dois papéis ao mesmo tempo. Algumas das características apresentadas pelas redes *peer-to-peer* são: auto-organização, dinamismo na composição da rede, balanceamento de carga, tolerância a faltas, disponibilidade, replicação e fácil escalabilidade [1].

Devido ao seu objetivo inicial, compartilhamento de conteúdos (geralmente arquivos de música e filmes), o tema segurança não foi considerado na maioria das propostas de rede *peer-to-peer*. Por exemplo, um *peer* mal intencionado poderia facilmente **disseminar conteúdos maliciosos, falsos ou corrompidos** pela rede com o intuito de **tirar proveito da situação ou tornar os recursos indisponíveis** aos demais *peers*[1].

Outro comportamento indesejado nas redes *peer-to-peer* é o de *peers* **que consomem os recursos da rede sem fornecer qualquer tipo de recurso em troca**. É comum que um *peer* cliente, ao realizar uma busca por algum conteúdo específico, receba como resultado várias fontes (servidores). Neste caso, a escolha do servidor para baixar o conteúdo vai ser feita de maneira arbitrária, pois não existe nenhum critério plausível que auxilie esta decisão.

Em última análise, qualquer *peer* da rede pode atribuir palavras chave de interesse comum e atrativas para um conteúdo falso apenas com o intuito de atrair *peers* clientes para o seu repositório de conteúdos.

Em redes *peer-to-peer*, o **conteúdo falso (corrompido ou poluído)** propagado por *peers* mal intencionados podem representar mais de 60% do número total de cópias [2], de alguns conteúdos bastante requisitados na rede. Além do inconveniente recebimento desse tipo de conteúdo indesejado, o tráfego de conteúdos poluídos diminui a quantidade de largura de banda útil disponível para os *peers* da rede. Assim, faz-se necessário o uso de mecanismos que assegurem a autenticidade dos conteúdos disponibilizados na rede *peer-to-peer*.

Uma abordagem para minimizar este problema é o uso de mecanismos criptográficos, tais como certificados digitais. Porém, como a rede *peer-to-peer* é muito dinâmica e a infra-estrutura de chaves públicas X.509 tem uma organização rígida e inflexível, normalmente não se adotam os certificados assinados por uma CA (*Certification Authority*), usando em seu lugar os certificados auto-assinados.

A desvantagem dos certificados auto-assinados é que não há uma entidade bem conhecida que os endossem, como a CA da infra-estrutura de chaves públicas [5], por exemplo. Ou seja, o certificado auto-assinado carece de credibilidade. Neste caso, pode-se considerar o histórico do *peer* em comunicações anteriores para dar credibilidade ao emissor de certificados auto-assinados. O problema desta abordagem é que na prática, os *peers* mudam constantemente de identificação para darem suporte ao anonimato [6]. Então, é necessário criar um mecanismo persistente de identificação para os *peers*, considerando que a rede é orientada a conteúdo e não a cliente/servidor.

Propor abordagens robustas de segurança em redes *peer-to-peer é* um grande desafio, tendo em vista que algumas das principais características dessas redes não vão ao encontro dos requisitos necessários para o correto funcionamento dos mecanismos de segurança. Alguns exemplos dos desafios encontrados quando se almeja fornecer soluções de segurança para redes *peer-to-peer* são: criação de mecanismos de identificação, garantindo o anonimato dos *peers* e o provimento da confidencialidade, integridade, disponibilidade e da autenticidade de conteúdos que não estão confinados a um servidor, mas distribuídos em uma rede sob a qual não há controle e nem sempre seus nós estão ativos.

Desenvolver uma arquitetura que forneça tais características de forma confiável pode ampliar ainda mais a utilização das redes *peer-to-peer* em ambientes profissionais, como o corporativo e o acadêmico.

1.2 Objetivos

Este trabalho propõe um modelo baseado em uma infra-estrutura de chaves públicas para prover integridade e autenticidade aos conteúdos compartilhados na rede *peer-to-peer*. Tal infra-estrutura deve ser a mais adequada às características de auto-gerenciamento, descentralização, escalabilidade, flexibilidade e dinamismo das redes P2P.

Propõe-se também a utilização de uma infra-estrutura de chaves públicas para fazer controles sobre os *peer*s, sem ferir o anonimato, garantindo assim a identificação persistente do *peer*/*principal*. Adicionalmente, objetiva-se impor um mecanismo de controle de acesso aos conteúdos, sem inibir o compartilhamento dos mesmos. Além disso, adota-se um esquema de reputação para obter escores que possam ser utilizados em uma abordagem de troca justa que auxilie o controle de acesso. O esquema de reputação visa dar credibilidade aos *peers*, auxiliando o esquema de identificação persistente baseado na infra-estrutura de chaves.

Todos os controles citados acima devem corroborar no sentido de diminuir a discriminação de conteúdo poluído na rede *peer-to-peer*.

1.3 Organização do Trabalho

Nesse capítulo foi descrito, de uma maneira geral, a motivação para este trabalho, bem como os objetivos traçados para o projeto. Os capítulos seguintes estão dispostos na seguinte ordem:

- No capítulo 2, são apresentados os conceitos fundamentais de segurança, suas políticas e violações, além dos modelos de segurança propostos pela literatura e os mecanismos de segurança. Neste capítulo também são abordados os conceitos de redes *overlay* e *peer-to-peer*.
- No capítulo 3, são apresentados os conceitos relacionados às tecnologias utilizadas neste projeto, sendo a arquitetura JXTA, o sistema de infra-estrutura de chaves SDSI/SPKI e a infra-estrutura de tabelas de *hash* distribuídas, o Bamboo.

- No capítulo 4, são apresentados alguns trabalhos relacionados a esta proposta, encontrados na literatura. Estes envolvem soluções que abordam autenticidade, reputação, controle de acesso, integridade e anonimato.
- No Capítulo 5, é apresentado o modelo proposto, onde são detalhados todos os aspectos conceituais.
- O capítulo 6 aborda os aspectos técnicos relacionados ao desenvolvimento do protótipo, bem como a sua implementação, sua avaliação e os resultados obtidos.
- No capítulo 7, são feitas as conclusões finais, pertinentes ao projeto.

Capítulo 2

2 Conceitos de Segurança, Redes *Overlay* e *Peer-to-Peer*

2.1 Introdução

Este capítulo tem como objetivo apresentar os conceitos de segurança, redes *overlay* e P2P. Tais temas são de suma importância para esta dissertação, pois são a base conceitual de suporte para a proposta.

2.2 Segurança

Com o aumento significativo das aplicações distribuídas, tais como: o comércio eletrônico, o vídeo sob demanda e a videoconferência, nota-se o aumento da complexidade no que se refere ao gerenciamento dos recursos fornecidos por esse ambiente de natureza distribuída.

Neste contexto, denomina-se *principal* (sujeito) toda entidade que possua participação num sistema e execute ações sobre objetos do sistema. O *principal* pode ser um usuário ou até mesmo um processo, e também pode ser visto como outra máquina em rede. Toda entidade com acesso não autorizado ao sistema que tente utilizá-lo recebe o nome de intruso.

Descrevem-se aqui alguns aspectos de segurança baseados em conceitos, metodologias e técnicas, os quais tentam assegurar certas propriedades em um sistema computacional. Há várias definições para segurança na literatura técnica, em geral segurança é caracterizada como a necessidade de se garantir várias propriedades, independente do

sistema, para que as informações se mantenham íntegras, confidenciais e sempre disponíveis [3], [4] e [5]:

a) confidencialidade: somente sujeitos autorizados do sistema devem ter acesso à informação;

b) integridade: toda e qualquer informação deve estar protegida de modificações, sejam intencionais ou acidentais, feitas por aqueles sujeitos que não possuem permissão para tal;

c) disponibilidade: assegura a disponibilidade das informações a sujeitos legítimos, evitando a negação do serviço através de ações maliciosas;

Outros autores acrescentam mais duas propriedades [6]:

d) autenticidade: garante que o sujeito, portador de uma identificação, pode provar ser o detentor da mesma;

e) irretratabilidade: evita que um sujeito negue falsamente sua responsabilidade por ações que tenha executado.

2.2.1 Política de segurança

A gestão de políticas de segurança em sistemas distribuídos é uma tarefa árdua, devido à natureza heterogênea dos sistemas. Poucos são os mecanismos que a implementam de forma integral [7], [8], [9] e [10].

Dependendo de como aplicada, uma política de segurança pode ter significados diferentes. Dentro de uma empresa, pode ter uma conotação puramente administrativa. Neste caso a política é um conjunto de leis e práticas, servindo para regulamentar a forma como a instituição faz o gerenciamento, a proteção e a distribuição de suas informações e recursos. A implementação da visão administrativa da política leva à segurança da informação em ambientes computacionais. Nessa perspectiva, tais regras passam a vigorar a fim de garantir que o sistema provedor de um serviço tenha as propriedades de segurança asseguradas [11].

2.2.2 Violações de segurança

Uma vez que tenham sido estabelecidas as regras de uma política de segurança, estas serão aplicadas a entidades que poderão ter autorização a recursos e sofrerão as responsabilidades que tal autorização impõe.

Toda ação cujo objetivo é contornar os controles de segurança de um sistema, de forma que viole a política de segurança, é considerada uma violação.

Segundo [11], vulnerabilidade é uma falha ou fraqueza no projeto, na implementação ou na operação e gerenciamento, que poderia ser usada para violar a política de segurança do sistema.

Vulnerabilidades existem em todos os sistemas operacionais comerciais atuais, assim como novas vulnerabilidades são anunciadas regularmente em alertas de segurança e de organizações de resposta a incidentes, tais como CERT (*Computer Emergency Response Team*) em todo o mundo [6].

Uma ameaça a um computador pode ser definida como qualquer ocorrência que, potencialmente, exponha ao risco os recursos do sistema, ou que possa ter um efeito não desejado sobre o mesmo [4].

Ainda segundo [11], um ataque é uma tentativa deliberada (especialmente no sentido de um método ou de uma técnica) de violar a política da segurança de um sistema, ou seja, efetivar uma ameaça.

2.2.3 Modelos de Segurança

Pode-se considerar que uma descrição formal do comportamento de um sistema de segurança, que atua segundo regras de uma política, pode ser descrita ou chamada de um modelo de segurança.

Segundo [12], a forma com que são representados os modelos é através de um conjunto de entidades e relacionamentos.

Conforme visto em [13] e [14] existem três tipos básicos de modelos que a literatura apresenta: aqueles que se baseiam em identidade[1] ou discricionários (*discricionary*), os baseados em regras ou obrigatórios (*mandatory*), e por fim os que têm base em papéis desempenhados no sistema (*role*).

[1] OSI Security Architecture Standard ISO/IEC 7498-2.

2.2.4 Mecanismos de Segurança

A implementação propriamente dita das políticas é feita pelos chamados mecanismos de segurança. Para realizar isso são utilizados os mecanismos de controle de acesso e os controles criptográficos [15].

A seguir serão descritos alguns destes mecanismos e controles de segurança [13]:

- **Autenticação**: é o mecanismo utilizado para garantir que um sujeito possa provar a um processo que o mesmo é o detentor verdadeiro de uma identidade;
- **Controle de Acesso**: define quais sujeitos ou processos têm acesso a quais recursos do sistema computacional e com que direitos. Pode ser empregado em diferentes níveis da aplicação, até aos sistemas básicos;
- **Criptografia**: é uma técnica utilizada para trocar o conteúdo compreensível (texto em claro ou texto plano) de uma mensagem por um conteúdo incompreensível (texto cifrado), através de uma função de transformação, chamado método de criptografia.

Com a criptografia pode-se obter a confidencialidade na troca de mensagens, especialmente quando as mensagens percorrem um canal de comunicação público e/ou não confiável. Inicialmente, surgiram os métodos criptográficos onde a confidencialidade da mensagem era feita usando o segredo do próprio método. Porém, esta técnica evoluiu para a criptografia com o uso de chaves. Um texto em claro, para ser transformado num texto cifrado ou vice-versa, requer um método criptográfico e uma chave. O método criptográfico é considerado confiável e seguro quando é de domínio público, podendo ser exaustivamente testado por todos [16].

Existem dois tipos de criptografia usando chaves:

- simétrica: nessa forma de criptografia, a mesma chave é utilizada tanto para cifrar quanto para decifrar o texto;
- assimétrica: nessa abordagem existem duas chaves (privada e pública) e a chave utilizada para cifrar é diferente da usada para decifrar.

***Secure Hashes e Message Digests*:** *Hashes* são funções de resumo/sumarização. Quando são irreversíveis são chamadas de *Secure Hashes* ou algoritmos de *Message Digest* (MD). Um algoritmo MD resume um texto em um bloco de tamanho fixo, que normalmente tem 128 ou 160 bits. As propriedades básicas de uma função de *Secure Hash* são:

- a partir de um *hash* de uma mensagem, garantir a não colisão (dois textos diferentes produzindo o mesmo *hash*);
- a partir de um *hash* de uma mensagem, garantir a integridade.

O MD5 e SHA1 são as duas funções de *Message Digest* mais comumente utilizadas atualmente [17].

Assinatura Digital: esta técnica consiste de duas etapas: a assinatura da mensagem em si e a verificação. O objetivo desse mecanismo é assegurar a autenticidade e a integridade de uma mensagem a tal ponto que ninguém a altere ou ainda faça uso total ou parcial da mesma. Também, que seu remetente não possa posteriormente negar que a tenha enviado, ou seja, garantir o não-repúdio da origem da mensagem.

A assinatura digital tem por base a criptografia assimétrica e funções *hash*. Assim sendo, utiliza-se de sua chave privada para assinar um resumo de mensagem, e o receptor verifica a assinatura e autenticidade com a chave pública do emissor [16].

2.3 Redes *Overlay*

Segundo Andersen e outros [18], uma rede *overlay* é uma rede computacional "virtual" criada sobre uma rede existente. Os *peer*s na rede *overlay* são conectados por ligações virtuais ou lógicas, cada uma correspondente a um caminho ou ainda apontando para muitas ligações físicas da rede subjacente. Um exemplo que pode esclarecer este conceito são as redes *peer-to-peer*, pois estão sobrepostas à Internet, e os *peer*s da rede usam uns aos outros como roteadores para enviar dados. Um exemplo pode ser encontrado em [19].

As redes *overlay* são muito usadas para permitir roteamento de mensagens para destinos que não são especificados por um endereço IP (*Internet Protocol*). Por exemplo, o Freenet [20] e *Distributed Hash Tables* – DHT (tabelas de *hash* distribuídas) podem ser usados para rotear mensagens para um *peer* que armazena um arquivo específico, cujo endereço IP não é previamente conhecido, só os endereços em nível *overlay*.

As redes *overlay* não controlam a maneira como os pacotes de dados são roteados entre *peer*s da rede subjacente, mas pode controlar a seqüência dos *peer*s da rede *overlay* através dos quais mensagens vão passar antes de alcançar seu destino final.

Uma rede real (física) pode ter seus componentes fazendo parte de várias redes *overlay*, com funções diferentes em cada uma das redes as quais estejam participando. Um *host* em uma rede física pode comportar-se como um *host* em uma rede *overlay* e como um

roteador em outra, e pode também ter mais de uma função na mesma rede *overlay*. A Figura 2-1 e a Figura 2-2 ilustram alguns exemplos.

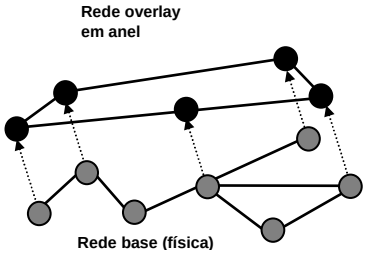


Figura 2-1 Uma rede overlay em anel

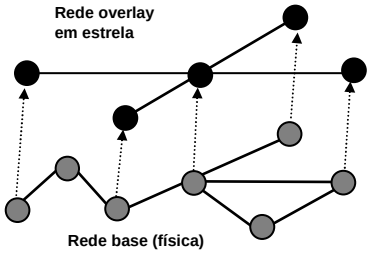


Figura 2-2 Uma rede overlay em estrela

Outras aplicações das redes *overlay* podem ser encontradas em serviços *multicast* [21], Redes de Distribuição de Conteúdo (CDNs) [22], *Resilient Overlay Networks* (RONs) [23] e OverQoS para assegurar a melhoria e garantia de serviços [24].

2.4 Redes *Peer-to-Peer* - (P2P)

A terminologia exata de redes *peer-to-peer* é sempre bem controversa, e existem várias definições. Mas isso não impede de se observar a diferença entre o paradigma cliente-servidor e o paradigma *peer-to-peer*. No primeiro caso, nota-se a distinção entre a entidade que provê (servidor) e a que consome (cliente) um determinado serviço. Por outro lado, tal distinção não ocorre em uma rede P2P, pois todos os *peers* (*peer*s) possuem funcionalidades equivalentes, isto é, tanto podem ser provedores (servidores) quanto consumidores (clientes) ao mesmo tempo.

A seguir serão expostas algumas definições citadas na literatura para uma melhor compreensão deste tema.

2.4.1 Definição de Redes *Peer-to-Peer*

Segundo [25], redes *peer-to-peer* são redes virtuais cujo os *peer*s podem atuar como servidores e clientes ao mesmo tempo. A maioria dos pesquisadores concorda que esta afirmação não está errada, mas acreditam que ela não está completa.

Para Sha [26] *peer-to-peer* são redes de computadores distribuídos de grande escala, e sem uma autoridade central. Cada computador ou máquina é conhecido como um *peer* (nó). Esses computadores são heterogêneos, isto é, tem poder computacional variado.

O grupo de trabalho *peer-to-peer* [27] define P2P como sendo: uma rede de compartilhamento de recursos e serviços computacionais por meio de troca direta entre sistemas. Os *peer*s podem atuar como servidores e clientes, assumindo o papel que é mais eficiente para a rede.

Stoica e Morris [28], por exemplo, dizem que redes *peer-to-peer* são sistemas distribuídos sem qualquer forma de controle centralizado ou hierarquia organizacional, no qual o *software* que está sendo executado em cada *peer* é equivalente em funcionalidade.

No trabalho de Rowstron e Druschel são apresentados vários aspectos interessantes de redes *peert-to-peer* tais como: auto-organização, adaptabilidade e escalabilidade. Sistemas *peer-to-peer* podem ser caracterizados como sistemas distribuídos nos quais todos os *peer*s possuem capacidades e responsabilidades idênticas e toda comunicação é simétrica [29].

Nesse trabalho é adotada a definição de Androutsellis-Theotokis e Spinellis [30], onde *peer-to-peer* são sistemas distribuídos que consistem de *peer*s interconectados, com capacidade de se auto-organizarem em topologias de rede, com o objetivo de compartilhar recursos como ciclos de CPU, armazenamento e largura de banda. São ainda capazes de se adaptarem a falhas e acomodarem populações itinerantes de *peer*s, enquanto mantém conectividade e performance aceitáveis, sem depender da intermediação ou suporte de uma autoridade (servidor) central.

A idéia de usar o conceito de redes P2P é fortalecida devido à maneira escalável e auto-organizada com que estas trabalham com grandes populações de *peer*s itinerantes. Além disso não precisam de uma entidade central que naturalmente pode se tornar crítica quanto à performance, ponto central de falhas e vulnerabilidades e motivo de demanda de administração do mesmo.

Uma expressão que está atrelada ao conceito de redes *peer-to-peer*, é o termo *Servent* que em seu conceito mostra as diferenças entre a arquitetura P2P e Cliente/Servidor. Esta expressão tem origem nos termos *Server* (Serv) e *Client* (ent). Um *peer* é denominado *Servent* em uma rede P2P pois atua como cliente ou servidor, ou os dois ao mesmo tempo. Isso é completamente diferente no modelo Cliente/Servidor, no qual cada entidade pode ser exclusivamente cliente ou servidor, mas nunca os dois ao mesmo tempo.

O que trouxe notoriedade às redes *peer-to-peer*, sem dúvida, são alguns de seus sistemas, tais como: Napster, que surgiu nos anos 90 [31] cuja característica foi a de fazer o

compartilhamento de arquivos de áudio (no formato MP3[2]). Outros exemplos são os sistemas de troca de mensagens: AIM[3], MSN Messenger[4], Yahoo! Messenger[5] e computação distribuída como SETI@home[6].

A topologia, estrutura e grau de centralização da rede *peer-to-peer*, e os mecanismos de busca e roteamento que emprega são cruciais para sua operação. Alguns exemplos clássicos em redes P2P são vistos: Freenet [20], Gnutella [19].

2.4.2 Topologias *Peer-to-Peer*

Sistemas P2P podem ser diferentes no que se refere à centralização ou não. Mesmo existindo o entendimento que as redes P2P são descentralizadas em sua essência, na prática isso nem sempre ocorre. Isto é, existem vários sistemas com diferentes níveis de centralização.

Nessa seção, será apresentada uma visão geral dessas topologias e alguns de seus aspectos mais relevantes.

2.4.2.1 Arquitetura Puramente Descentralizada

Nessa arquitetura, todos os *peer*s na rede executam as mesmas tarefas, ou seja, tanto são clientes quanto servidores. Não existe uma entidade central (servidor) para coordenar suas atividades [30] nem para desempenhar funções especiais ou específicas.

Num sistema P2P puro, os *peer*s são interconectados diretamente e enviam mensagens de pesquisa para seus vizinhos[7] (*peer*s que conseguem alcançar). Em algumas redes P2P, cada *peer* recebe uma requisição de busca e procura em sua base de dados local, retorna o resultado e em paralelo encaminha a busca para um ou alguns, ou até mesmo todos os seus vizinhos (dependendo da heurística adotada). Uma vez que não existe uma unidade organizacional central, os *peer*s têm que manter por si só informações sobre a conectividade dos *peer*s vizinhos.

---

[2] MP3 (MPEG-1/2 Audio Layer 3) foi um dos primeiros tipos de compressão de áudio com perdas quase imperceptíveis ao ouvido humano)
[3] AIM (Aol Instant Messenger – http://www.aim.com)
[4] MSN Menssenger (http://mesenger.msn.com)
[5] Yahoo! Messenger (http://messenger.yahoo.com)
[6] SETI@home (http://setiathome.ssl.berkeley.edu)
[7] A noção de vizinhança pode ser definida por várias funções diferentes e depende da implementação, porém um vizinho necessariamente deve ser um *peer* alcançável.

A vantagem de sistemas P2P puros é o fato de que não existe um ponto único de faltas (ex: um servidor central) que resulta em uma maior capacidade de sobrevivência da rede P2P. Uma desvantagem de uma rede P2P completamente descentralizada é a dificuldade de realizar uma atualização do sistema completamente. É possível construir um protocolo e o *software* que permita fazer atualizações gerenciadas por servidor central, mas atualmente ainda nenhum sistema P2P possui essa característica.

O maior problema dos sistemas P2P puros é o processo de *bootstrapping* (inicialização). Quando o *software* P2P é o primeiro iniciado na máquina, ao menos um *peer* dos existentes no sistema precisa ser conhecido para habilitar a entrada na rede do *peer* em inicialização. Na maioria dos casos, isso é feito fornecendo um servidor fixo e bem conhecido que possui a lista de alguns outros *peer*s P2P. Outra possibilidade seria empregar conexões aleatórias em certas faixas de endereços IP. Porém, esta solução é considerada muito hostil e conseqüentemente deve ser considerada somente em último caso. Um sistema P2P, que contate um servidor central somente uma vez para obter uma lista de outros pares, pode ainda ser considerado como um sistema P2P puro porque uma interação maior ou adicional com o usuário não é necessária.

A Figura 2-3 mostra um exemplo típico de uma rede puramente descentralizada, onde a comunicação e troca de mensagens é feita independentemente de um ponto centralizador.

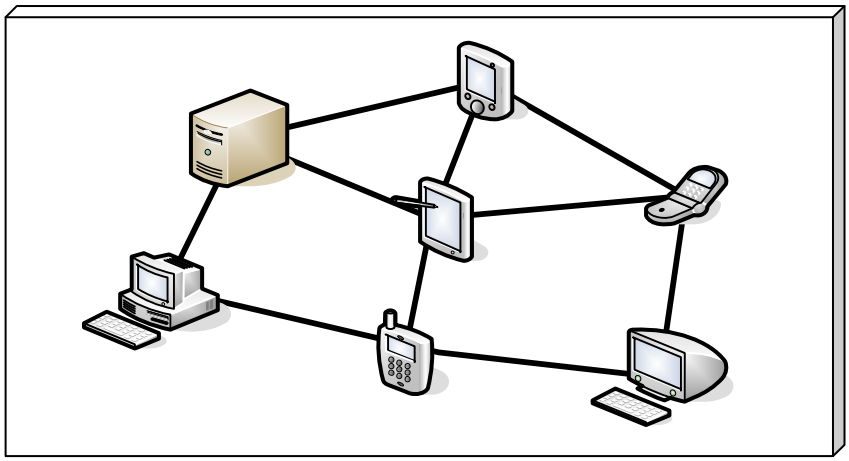


Figura 2-3 Arquitetura Puramente Descentralizada

2.4.2.2 Arquitetura Descentralizada Híbrida

A abordagem aqui apresentada retrata um sistema que possui um servidor para facilitar a interação entre os *peer*s. Existem diferentes maneiras de uso do servidor.

O servidor pode fornecer facilidades de busca e indexação (ex: Napster [31]), autenticação de usuários e distribuição de chaves criptográficas (ex: Grokster [32]) ou coordenação do trabalho dos *peer*s (ex: SETI@home [33]).

Existem várias vantagens no uso de um servidor central. O protocolo pode ser facilmente mudado e melhorado sem ter o problema de convencer todos os clientes a suportar esse protocolo (embora isto não seja considerado sempre como uma vantagem, porque um protocolo deve sempre ser projetado para ser o mais flexível possível). Além disso, o uso de um servidor central ajuda na segurança. Mecanismos de autenticação e autorização podem ser empregados, pois todo o tráfego significativo (ex: requisições de busca, *login*, etc) está fluindo através do servidor. Isso pode não ser significante em relação ao compartilhamento de arquivos distribuídos, mas segurança definitivamente é uma questão importante quando sistemas P2P são empregados em ambientes corporativos, por exemplo.

Por outro lado, existem desvantagens em relação à redes P2P baseadas em servidor. Obviamente, o servidor central representa um ponto único de falha, vulnerabilidades e comprometimento do desempenho. Isso não é somente um problema técnico, mas também legal. No entanto segundo Milojici [34] o uso de um servidor centralizado como o Napster, não é um problema, uma vez que traz eficiência e estabilidade. O grande problema foi a questão legal que trouxe o Napster à queda, ou seja infringindo leis de direitos autorais [34]. Mas isso não é um problema geral de sistemas P2P – somente quando usados para compartilhar conteúdos legais ou com direitos autorais. Ataques de negação de serviço são potencialmente grandes ameaças para servidores centrais (ex: grandes servidores como www.cnn.com). De acordo com a definição acima, sistemas P2P podem ter vantagens usando um servidor central, mas devem também funcionar sem esse serviço. Um exemplo para esta característica é o *Groove* [35]. O *Groove* pode conectar-se a um servidor central para armazenar informações permanentemente, mas na ausência de tal servidor, computadores que tenham o *Groove* sendo executado podem ainda interagir entre si e prover a maioria das funcionalidades.

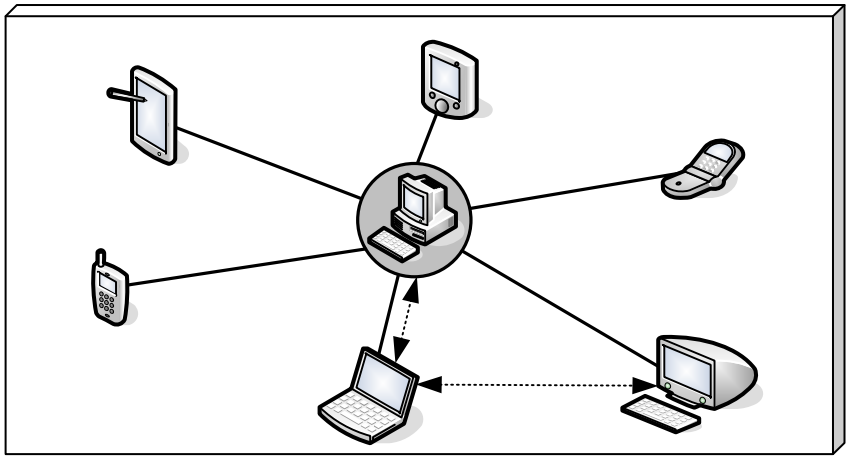


Figura 2-4 Arquitetura Descentralizada Híbrida

2.4.2.3 Arquitetura Parcialmente Centralizada

Mesmo tendo as mesmas bases que os sistemas puramente descentralizados, nesta arquitetura alguns *peer*s possuem uma função ou papel diferenciado, tendo um *peer* central para fornecer índices de conteúdos compartilhados por outros pares [30].

Um exemplo dessa abordagem na escolha desses *peer*s centrais foi usado primeiramente pela rede *FastTrack* [36]. Certos critérios, tais como largura de banda, latência e poder de CPU, são usados para eleger o servidor mais apropriado (no *FastTrack*, tais servidores são chamados de Super-*Peer*s). Um *peer* normal conecta-se em um super-*peer* e envia uma lista dos seus recursos compartilhados. Requisições de busca são enviadas somente para o super-*peer* que direciona as mensagens para outros super-*peer*s. Cada super-*peer* que recebe uma consulta, realiza primeiro uma busca em seu banco de dados local de arquivos compartilhados (aqueles que foram anunciados pelos seus *peer*s clientes).

A vantagem dessa abordagem é que combina as vantagens da topologia centralizada (buscas rápidas) com a descentralizada pura (não possui um ponto único de falhas). Esta característica torna o super-*peer* um ponto forte deste modelo P2P, que tem sido aprovado pela sua ampla utilização em ferramentas de compartilhamento de arquivos P2P (ex: [37] e [38].

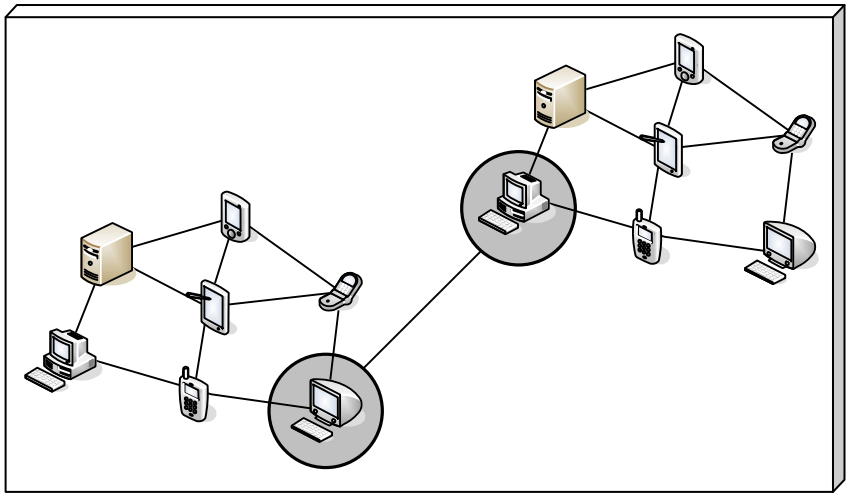


Figura 2-5 Arquitetura Parcialmente Centralizada

2.4.3 Estruturas *Peer-to-Peer*

Segundo Androutsellis-Theotokis [30] as estruturas das redes P2P são divididas em redes **não-estruturadas** e **estruturadas**.

Na abordagem das **redes não estruturadas (de formação *ad hoc*)**, a localização do conteúdo compartilhado não possui nenhum vínculo com a topologia *overlay* propriamente dita. Faz-se então necessário a busca do conteúdo diretamente nos pares. Alguns mecanismos usam o método de *flooding* (inundação), que propaga consultas na rede até que o conteúdo seja alcançado.

Os sistemas não-estruturados são, geralmente, usados em situações que existe alta taxa de entrada e saída de *peer*s na rede. Alguns exemplos tradicionais destes sistemas P2P são:, Publius [39], Gnutella [19], Kazaa [37], Edutella [40], FreeHaven [41], etc.

As **redes estruturadas** são determinísticas. Dessa forma, todo o conteúdo está indexado, isto é, a cada consulta, pode-se saber exatamente em que ponto da rede um determinado conteúdo está.

Todo o conteúdo (arquivos) possui ponteiros para si informando sua localização na rede P2P. Assim, é feito um mapeamento completo dos conteúdos da rede P2P criando uma relação entre identificador e endereço do conteúdo. A disposição dessas informações está em uma tabela de roteamento distribuída.

Exemplos típicos de sistemas estruturados baseados em DHT: CHORD [28], CAN [42], PAST [43], Tapestry [44].

2.4.4 Mecanismos de Busca

A localização de maneira eficiente dos pares e seus serviços em um sistema *peer-to-peer* é uma tarefa complexa, especialmente em sistemas descentralizados e em grande escala. Tradicionalmente, mecanismos de busca são apenas eficientes em sistemas centralizados, onde um servidor mantém um histórico de todos os conteúdos disponíveis no sistema. Dessa maneira, os *peers* clientes usam o servidor para localizar conteúdos oferecidos por outros *peers*. Uma alternativa é que os *peers* conectem-se no servidor e registrem seus conteúdos. Então, sempre que um *peer* cliente localizar algum conteúdo, o seu proprietário será avisado para que então o forneça.

Nos sistemas descentralizados, a indexação do conteúdo é feita diretamente no *peer* fornecedor ao invés de ser feita em um servidor. Nesse caso, a resposta às solicitações por conteúdos é feita pelo próprio *peer* fornecedor, sempre que o mesmo perceber que seu conteúdo corresponde ao solicitado pelo *peer* cliente [45]

Para otimizar o envio de mensagens de um *peer* para outro, são utilizados algoritmos de busca e roteamento. Existem três algoritmos mais comuns que visam melhorar a busca e o roteamento, que são: **Busca Centralizada**, **Busca por Inundação** e **Busca por DHT**.

Busca Centralizada: baseia-se em um ponto centralizador (podem até parecer vários, mas isso normalmente é feito utilizando espelhamento) de busca e *peer*s clientes que consultam este ponto centralizador para então realizar trocas de informações com outros *peers* de forma direta; um exemplo popular é o Napster, onde os *peers* fazem conexão diretamente a um diretório central e registram informações sobre seus conteúdos oferecidos de maneira compartilhada.

Na ilustração da Figura 2-6, nota-se que ao receber uma requisição de um *peer* (*peer* 2), o servidor central, que é o índice (diretório), escolhe o *peer* no diretório que for mais adequado. Isto é, o *peer* escolhido pode ser o mais rápido e disponível, dependendo das necessidades do usuário, nesse exemplo o *peer* (nó 3). Então, a troca de arquivos será realizada diretamente entre os dois *peers*. Naturalmente, nesta abordagem é requerida uma infra-estrutura para gerenciar o servidor de diretório, que armazena informações sobre todos os participantes da comunidade. Conforme a necessidade de escalabilidade se fizer presente, mais servidores para a infra-estrutura serão necessários, limitando o modelo que se tornará dependente da tecnologia. Contudo, a experiência do Napster mostrou que, exceto por questões legais, esse modelo era bastante robusto e eficiente.

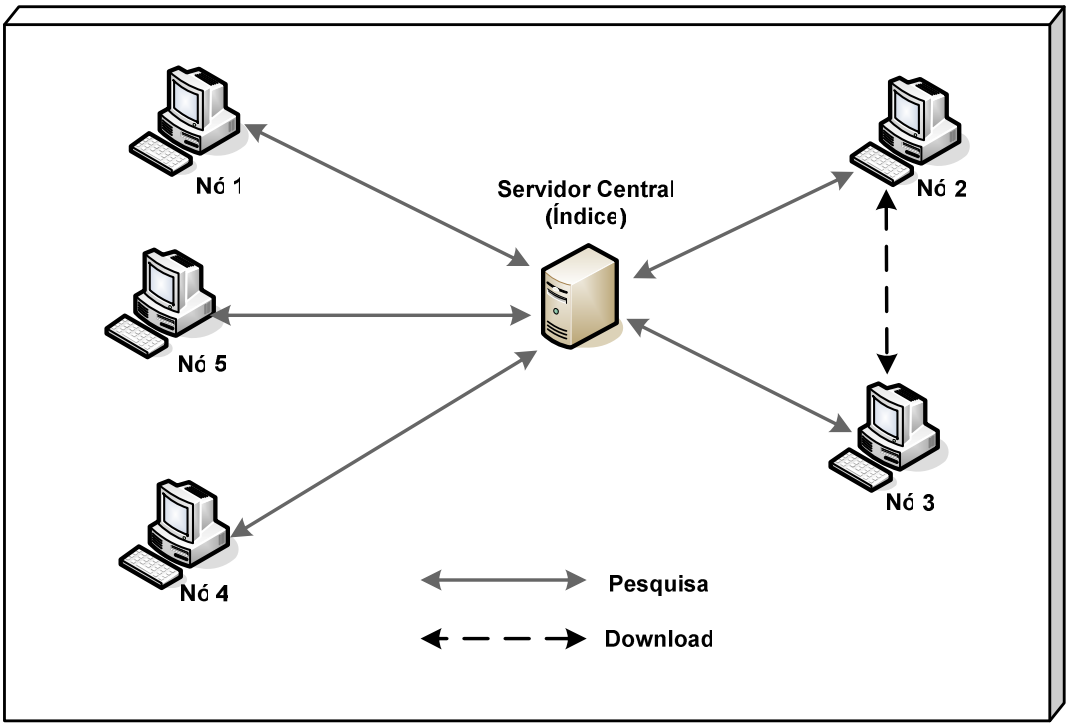


Figura 2-6 Modelo Centralizado

Busca por Inundação: É feita em uma rede com *peers* independentes. A busca é feita pelos *peers* vizinhos e suas respectivas proximidades, isto é, o vizinho mais próximo do *peer* solicitante. Quando se trata dessa abordagem de busca por inundação, tal modelo se diferencia do modelo anteriormente citado, pois não se baseia na publicação dos recursos compartilhados.

Ao contrário do modelo centralizado, cada requisição de um *peer* é enviada para todos os *peers* diretamente conectados (nesse exemplo o *peer* 2 faz uma pesquisa). Estes, por sua vez reenviam a requisição aos *peers* que também estão diretamente conectados a eles, e assim sucessivamente até que a requisição seja respondida (*peer* 5 devolve resposta ao *peer* 2) ou que ocorra o número máximo de encaminhamentos (valores normalmente usados de 5 a 9).

Como pode ser visto na Figura 2-7, essa abordagem de busca é utilizada pelo Gnutella [19] e requer alta capacidade dos enlaces de comunicação para proporcionar desempenho razoável. No entanto, problemas relacionados à escalabilidade surgem quando se pretende alcançar todos os *peers* em uma rede, mas é eficiente em comunidades limitadas, tais como em uma rede corporativa.

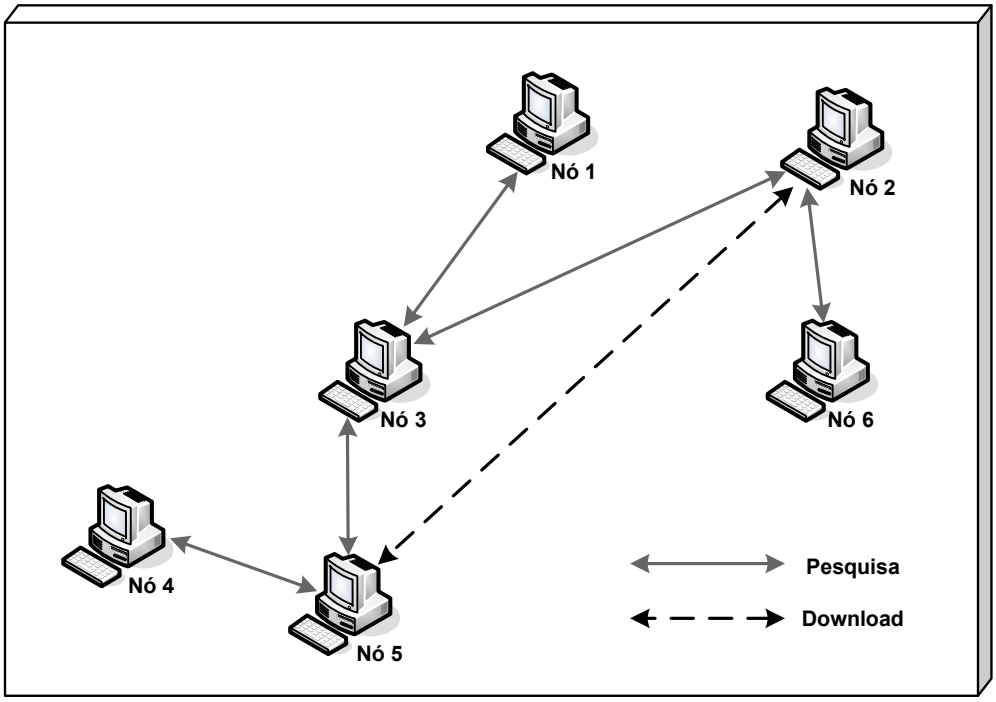


Figura 2-7 Modelo de Inundação

Algumas providências têm sido tomadas por algumas corporações, para minimizar os problemas relacionados ao consumo dos enlaces de comunicação. Tais companhias têm desenvolvido uma técnica chamada de *super-peer,* onde esse peer concentra as requisições, levando a uma diminuição dos recursos de rede e o alto custo com consumo de CPU. As buscas recentes que ficam nos *caches* desses *super-peers* também ajudam a diminuir o tempo de resposta.

Busca por DHT (*Distributed Hash Table*): A busca usando tabelas de *hash* distribuídas (DHT) é o método mais recente. Isso é observado na Figura 2-8. Quando um documento é publicado (compartilhado) em tal sistema, um ID é associado ao documento baseado em um *hash* do seu conteúdo e de seu nome (exemplo *peer* 4). Cada *peer* então encaminha o documento ao *peer* cujo ID é mais próximo do ID do documento (é feito o armazenamento no *peer* 6). Esse processo é repetido até que o ID do *peer* atual seja o mais próximo do ID do documento. Cada operação de roteamento também garante que uma cópia local do documento seja mantida. Quando um *peer* solicita o documento de um sistema P2P, a requisição irá até o *peer* com ID mais semelhante ao ID do documento. Esse processo continua até que uma cópia do documento seja encontrada. Então o documento é transferido

ao *peer* que originou a requisição, enquanto cada *peer* que participou do roteamento permanecerá com uma cópia local do documento [28].

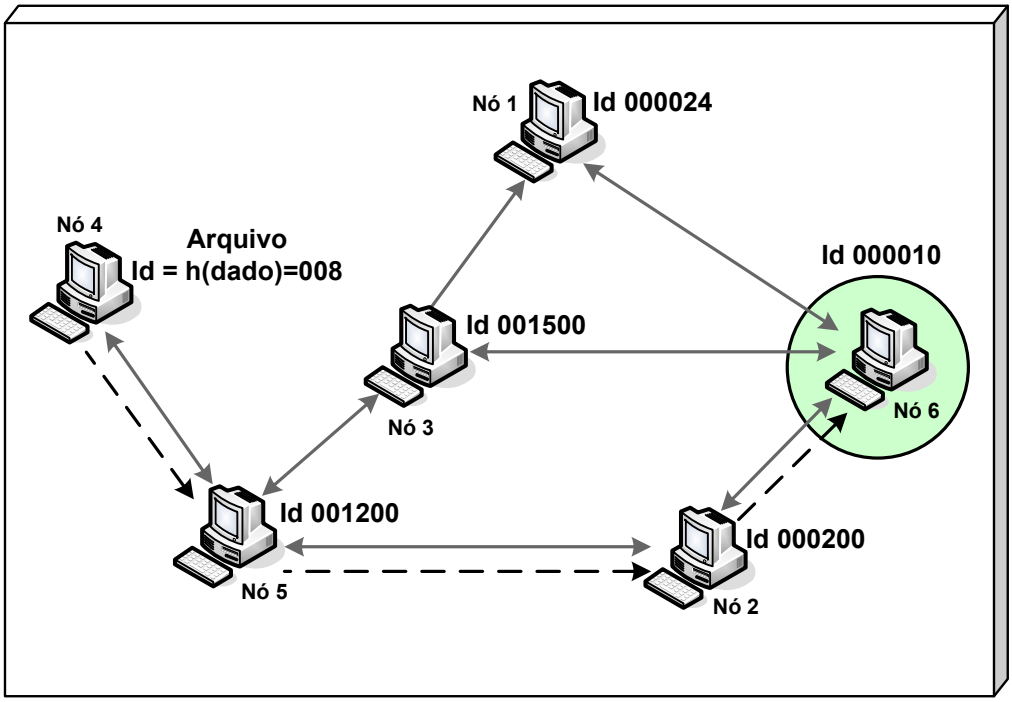


Figura 2-8 Modelo DHT

Existem quatro sistemas principais que implementam esse modelo: Chord [28], CAN [42], PAST [43], Tapestry [44].

2.4.5 Áreas de Aplicação

Nessa seção, uma visão geral do uso de aplicações e protocolos das redes *peer-to-peer* é exposta.

Mensagens Instantâneas: Uma forma conveniente de comunicação com um grupo pequeno de pessoas selecionadas (ex: amigos, família, etc). Nesse caso, geralmente, um servidor central armazena os perfis dos usuários e a lista dos usuários registrados. Enquanto a comunicação é feita entre os *peer*s, a busca por outras pessoas é feita por meio do servidor. Uma das razões pelo qual o servidor é requerido é a habilidade de envio de mensagens a outras pessoas (ex: *peer*s). No caso em que o *peer* alvo não esteja on-line, o sistema armazenará as mensagens até que ele fique on-line novamente. Isso pode ser possível em um sistema *peer-to-peer* sem servidor, mas o preço disso seria a complexidade de implementação

e a possível perda de mensagens. Exemplos de sistemas com servidor central são: Napster [31], [46], Threedegrees [47] e Jabber [48];

Troca de Arquivo: Nesse tipo de aplicação existe certa disputa sobre a utilidade de aplicações P2P de compartilhamento de arquivos. Enquanto o *download* de arquivos é sempre realizado diretamente entre os *peer*s, a forma de localizar tais arquivos difere muito nas aplicações P2P. Alguns usam um servidor central (ex: Napster) enquanto outros enviam requisições de consultas diretamente a outros *peer*s (ex: Gnutella [49], Freenet [20], e FastTrack [36];

Colaboração: é um ambiente onde existem aplicações que permitem aos seus usuários a comunicação de dados e voz (VoIP), mensagens de texto, imagens, entre outras formas de informação de maneira direta, ou seja, sem um servidor. Um exemplo P2P é o Groove [35], que armazena a informação compartilhada e fornece serviços de colaboração sem a existência de um servidor.

Processamento Compartilhado: Um dos maiores benefícios de P2P é o do processamento compartilhado das máquinas na Internet, isto é, a combinação do poder computacional que está ocioso nas máquinas na rede. Para utilizar estes recursos, os usuários baixam e instalam programas capazes de realizar uma pequena parte de uma complexa tarefa computacional enquanto o computador não é usado. Exemplos de programas P2P são: SETI@home [33] e Genome@home [50].

Serviços de Localização: A maioria da pesquisa científica em P2P está sendo feita na área de serviços de busca. Isso não é surpreendente, porque busca é um dos maiores desafios em redes P2P. Dentre os sistemas P2P que são otimizados para serviços de busca estão aqueles que usam tabelas de *hash* distribuídas (DHT), que são capazes de localizar usando algoritmos de busca complexos. Porém, o inconveniente da maioria desses sistemas é o fato de que eles somente fazem busca por números (no caso de busca por *strings*, elas são localizadas por meio de representações numéricas de tais *strings*). Exemplos são sistemas como PAST [29], Chord [28], e P-Grid [51].

Comunicação Móvel *Ad Hoc*: ocorre, especialmente, entre dispositivos móveis, por exemplo, dispositivos conectados via *wireless.* Alguns dos exemplos da utilidade de P2P nessa área são: GnuNet [49] e JXME [52] (JXTA para J2ME – Java 2 Mobile Environment);

Distribuição de Conteúdo: sistemas P2P podem ser usados para a distribuição de informação ou arquivos (algumas vezes chamado de ESD – *eletronic software distribution*).

Ao invés de ter uma fonte central que envia arquivos para computadores de destino diretamente, uma rede P2P pode disseminar arquivos, de maneira anônima. A carga (por exemplo, largura de banda, poder de CPU, *throughput*, etc) é distribuída sobre toda a rede. Esse conceito é usado com sucesso, por exemplo, pela Intel [53], onde o *software* é distribuído para vários fabricantes usando P2P. Esse sistema pode ser comparado a um sistema *push*, mas a vantagem é que não precisa de um servidor ou *proxies*.

Middleware: O caso que mais demanda o uso de um sistema P2P é seu uso como uma plataforma de *middleware*. Sistemas de *middleware* P2P fornecem serviços tais como busca distribuída ou aplicações de alto nível para descoberta de *peer*s. Dependendo da estrutura (ou topologia) da rede P2P, vários casos de uso podem ser aplicáveis ou não. Se um sistema P2P é necessário como *middleware*, a sua aplicação tenta se ajustar da melhor maneira aos requisitos. Alguns poucos sistemas P2P podem ser usados como uma plataforma P2P. Entre eles estão JXTA [54] e Omnix [25].

2.4.6 Plataformas de Desenvolvimento *Peer-to-Peer*

Atualmente, das inúmeras aplicações *peer-to-peer*, muitas não utilizam nenhuma infra-estrutura (*framework*) para desenvolver suas aplicações. Mas já existem algumas tecnologias utilizadas para esse fim, com o objetivo de facilitar a vida dos desenvolvedores.

O projeto **JXTA** é um conjunto de protocolos P2P simples e abertos e que permitem a construção de *softwares* para rede P2P. Esse projeto foi criado pela *Sun Microsystems*. E está disponível na Internet para uso livre [55].

A *Microsoft* oferece recursos na plataforma **.NET** para desenvolvimento de aplicações P2P [56].

Outro exemplo de plataforma de desenvolvimento é o **GDK** [57], que foi desenvolvido usando a tecnologia *Microsoft Componnent Object Model* (COM) e que pode ser obtida livremente. Porém, para todo produto criado usando esta plataforma, deve-se adquirir uma licença.

2.5 Considerações Finais

Neste capítulo foram apresentados os conceitos básicos da segurança de sistemas onde há as políticas, modelos e mecanismos de segurança. Além disso, abordaram-se também definições de redes P2P e *Overlay*, detalhando os aspectos de topologia, estrutura e

mecanismos de busca. As áreas de aplicação de redes P2P e plataformas de desenvolvimento também foram abordadas.

No capítulo seguinte, serão apresentadas as tecnologias usadas no desenvolvimento do protótipo deste projeto: **JXTA**, **SDSI/PKI** e **Bamboo**.

Capítulo 3

3 Tecnologias P2P, DHT e de Segurança

3.1 Introdução

Neste capítulo, são apresentadas algumas tecnologias de interesse para a proposta. Será descrita em detalhes a infra-estrutura de desenvolvimento JXTA em seus aspectos funcionais e componentes, bem como sua arquitetura. Na seqüência, abordam-se a tecnologia de segurança SDSI/SPKI e, por fim, uma visão do projeto Bamboo, utilizado como repositório distribuído de índice.

3.2 JXTA

O JXTA é um projeto idealizado pela *Sun Microsystems*. Esta nova tecnologia é um conjunto de protocolos abertos, que permitem que todos os dispositivos conectados na rede se comuniquem e colaborem entre si como em uma rede P2P [55].

A arquitetura do JXTA é composta por seis protocolos diferentes [58]:

- ***Peer EndPoint Protocol*** **- PEP**: é o protocolo com o qual um *peer* pode descobrir uma rota, usada para enviar mensagens a outro *peer*;
- ***Rendezvous Protocol*** **- RVP**: é um protocolo de propagação através do qual um *peer* pode difundir uma mensagem para um grupo de outros *peer*s.
- ***Peer Resolver Protocol*** **- PRP**: é o protocolo através do qual é possível encaminhar consultas através dos demais *peer*s;
- ***Peer Discovery Protocol*** **- PDP**: é o protocolo através do qual um *peer* pode publicar as suas características na rede e descobrir características dos demais *peer*s;

- ***Peer Information Protocol* - PIP**: é o protocolo através do qual um *peer* pode obter informações sobre o estado de outro *peer*. Um estado pode ser: a carga, tráfego, funcionalidades, etc.;
- ***Pipe Binding Protocol* - PBP**: é o protocolo usado para estabelecer a comunicação virtual (*pipe*) entre dois ou mais *peer*s.

Uma característica inovadora na plataforma JXTA é sua intenção de tornar todos os sistemas P2P compatíveis. Por isso, JXTA tenta padronizar seus protocolos não sendo exigido que todos eles sejam implementados, apesar de recomendável. Outro aspecto é que o JXTA permite a criação de uma rede virtual no topo de redes existentes, ficando, dessa maneira, transparente à rede física, mascarando a complexidade existente nas camadas mais baixas da pilha de protocolos.

A rede JXTA permite que *peer*s interajam entre si de maneira independente de localização, tipo de serviço ou ambiente operacional – ainda que certos *peer*s ou recursos estejam protegidos por um *firewall* ou usem diferentes tecnologias de transporte de rede [59]. Por conta disto, o acesso aos recursos da rede não têm limitação, devido às divergências e incompatibilidades comuns de plataformas, ou ainda às restrições que a arquitetura cliente-servidor impõe. Um exemplo pode ser visto na Figura 3-1.

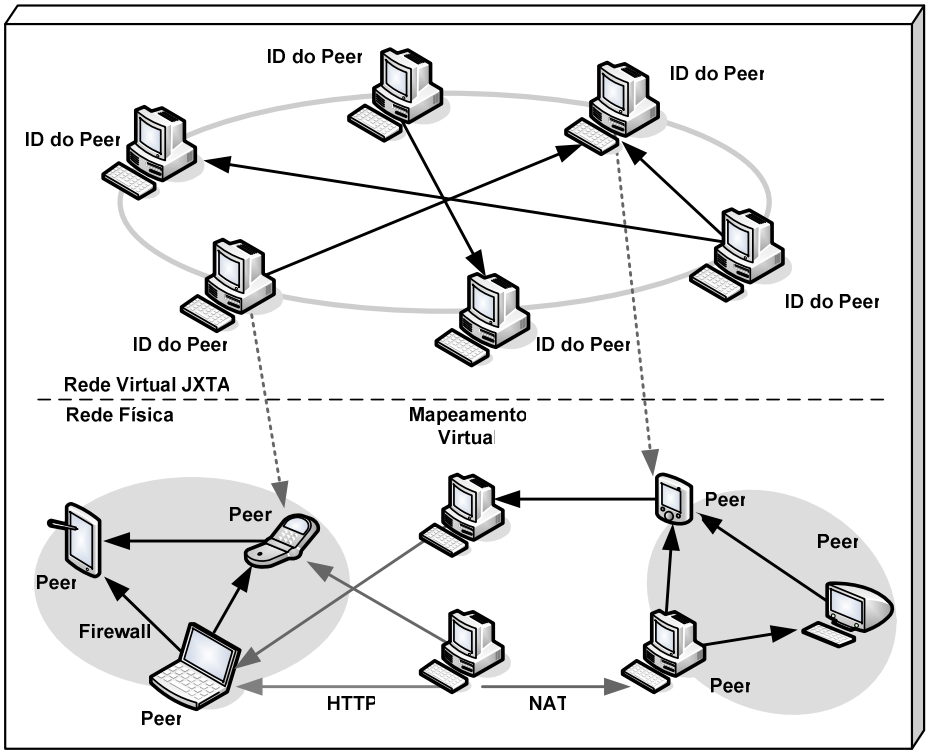


Figura 3-1 Rede virtual JXTA [55]

O JXTA utiliza recursos tecnológicos como HTTP, TCP/IP e XML. Por não estar preso a uma linguagem de programação específica, sistema de rede, ou à plataforma de sistema operacional, isto o qualifica para trabalhar com a combinação dos mesmos. Existem algumas implementações de referência do JXTA (Java, C/C++, Perl, entre outros) [60].

3.3 Arquitetura JXTA

O que assegura o funcionamento adequado dos protocolos JXTA é sua arquitetura. A divisão dessa arquitetura compreende três camadas distintas, como mostrado na Figura 3-2.

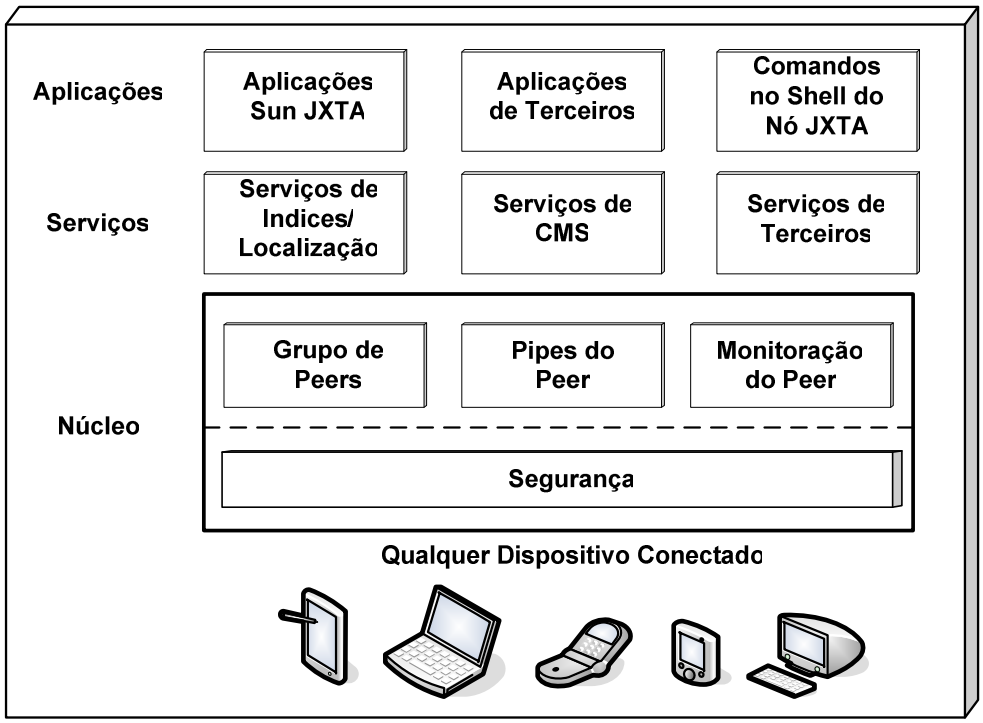


Figura 3-2 Arquitetura de camadas JXTA [55]

Segundo [61], no núcleo estão intrínsecas as operações básicas comuns a toda as camadas e a efetivação da rede P2P. Ou seja, estão presentes todos os mecanismos e a infra-estrutura necessários para criar aplicações *peer-to-peer*. Já na camada de serviço, por sua vez, há serviços de rede para a operação da rede P2P. Porém, nem todas as camadas são, necessariamente, requeridas. Há uma série de funcionalidades adicionais, desenvolvidas pela comunidade JXTA que podem ser agregadas aos serviços básicos. Alguns exemplos desses serviços de rede podem ser: indexação e localização de arquivos, sistema de armazenamento (CMS – *Content Management System*) e compartilhamento de arquivos. Na camada de

aplicação, o usuário pode ter o controle dos mais variados serviços. Um exemplo relevante é da aplicação sala de bate-papo, que se utiliza tanto dos serviços, quanto do núcleo para troca de mensagens [60].

3.4 Principais Componentes da Plataforma JXTA

Alguns componentes do JXTA têm um destaque maior por contribuir para a abstração de rede, criada pelo próprio JXTA com o aspecto de uma camada. Para isto, o JXTA vale-se de um endereço lógico na rede para cada *peer*, isto é, um *peer* ID. Outro componente é o grupo de *peer*s ou *peer groups*, os quais formam um conjunto de *peer*s que se auto-organizam, formando domínios virtuais com características pré-definidas. Outro papel que um *peer* pode assumir é a responsabilidade pela criação de recursos e serviços como, por exemplo, os *peers groups*, que precisam ser publicados para estarem disponíveis à outros *peer*s na rede JXTA. Todo recurso precisa ser anunciado/publicado através de *advertisement* específico. A rede JXTA também possui os *resolvers*, mecanismos que fazem todas as operações de *bind* como traduções de nomes em endereços de rede. Os *pipes*, que são os mecanismos mais importantes da arquitetura, são utilizados no processo de comunicação entre *peer* de maneira transparente [60]. Algumas das abstrações principais, e componentes característicos da rede JXTA, serão apresentados a seguir.

3.4.1 Peer

Para [62] um *peer* é: qualquer entidade capaz de executar algum trabalho útil e passível de comunicar os resultados desse trabalho para outras entidades da rede P2P, podendo ser direta ou indiretamente. Ou seja, é uma simples aplicação sendo executada em um dispositivo que tenha a capacidade de se comunicar com outros *peers*. Os *peers* possuem a característica de serem autônomos, operarem independentemente e de forma assíncrona: Podem, também, serem classificados de acordo com suas funcionalidades, como: *peer* simples, *peer* de encontro e *peer* de roteamento [62].

- ***Peer* simples (*Simple Peer*)**: somente serve o usuário, permitindo fornecimento e consumo de serviços por outros *peer*s;
- ***Peer* de encontro (*Rendezvous Peer*)**: provê um ponto para localizar outros *peers* e seus respectivos serviços para um determinado segmento da rede. O

rendezvous é usualmente utilizado como "ponte" para comunicação com outros *peers* da rede, mas não possui a capacidade de atravessar *firewalls*;

- ***Peer* de roteamento (*Relay Peer*)**: possui mecanismo para comunicação com outros *peers* separados por *firewalls* ou mecanismo de NAT (*Network Address Translation*).

3.4.2 Grupo de *Peers*

Um *peer group* é uma coleção de *peers* que possuem em comum algum tipo de interesse. Os *peers* organizam-se em *peer groups* por afinidade, sendo cada um deles identificado por um ID único [58].

Os *peers* possuem a capacidade de pertencer a mais de um *peer group* simultaneamente. Por padrão, todos os *peer*s fazem parte de um grupo de *peers* global, denominado *Net Peer Group* [58].

3.4.3 Pipes

Pipe é um mecanismo que estabelece canais de comunicação virtuais, usados por serviços ou aplicativos para enviar e receber dados. Desta forma, os *peers* não precisam se preocupar com a localização dos outros *peers* e a topologia de rede para enviar as mensagens. Os *pipes* podem transferir qualquer objeto, seja este código binário, data *strings* ou objetos baseados na tecnologia JAVA [60].

Sendo os *pipes* unidirecionais, para que dois *peers* se comuniquem, são necessários dois deles. Um para o *peer* que envia e outro para o *peer* que recebe, sendo chamados respectivamente de *output pipe* e *input pipe* [60].

3.4.4 Serviços

São entidades que provém funcionalidades para a rede, das quais os outros *peers* podem usufruir. Alguns exemplos são: serviço de transferência de arquivos, serviço de processamento, ou qualquer outra atividade que possa ser desempenhada por um *peer* da rede P2P [58].

3.4.5 Anúncios

Todas as entidades, serviços ou recursos do JXTA são publicados para a rede através de anúncios (*advertisements*). Os anúncios são feitos sob a forma de documentos XML, contendo estrutura de meta-dados que se referem ao documento WSDL (*Web services description language*). Por se tratarem de uma abstração para os recursos, visam principalmente identificar o *peer* para as outras entidades da rede, como os grupos de *peer*s ou ainda outros *peer*s.

No JXTA são definidos cinco tipos de anúncios: *Peer*, *Peer Group*, *Module Class*, *Module Specification* e *Module Implementation*, os quais possuem um período de validade predeterminado, evitando que recursos que não estão disponíveis sejam propagados como disponíveis pela rede [62].

3.4.6 Mensagens

Mensagens são a unidade básica de comunicação entre *peers*, que representadas por documentos XML. Tais mensagens são objetos enviados e recebidos pelos *peers* e utilizadas por serviços de *pipes*.

3.5 Segurança

JXTA oferece apenas recursos para proteção de mensagens em trânsito, principalmente com base no TLS e suas implicações.

Protocolo de transporte seguro TLS (*Transport Layer Security*): Também conhecido como SSL (*Secure Socket Layer*), é baseado na infra-estrutura de chave pública. O JXTA fornece, como meio de comunicação de mensagens, aplicações que exploram estes recursos criando *pipes* seguros ao usar o TLS para se proteger de ataques como interceptação de mensagens.

Certificados nos *peer*s: A camada TLS obriga o uso de certificados X.509. Cada *peer* gera seu próprio certificado e atua como sua própria CA (*Certificate Authority*). Esse certificado é chamado certificado raiz e é usado para assinar os certificados de serviços que os *peers* emitem para cada serviço que oferecem e distribuem através de anúncios de *peers*.

Ambiente de segurança pessoal: Cada *peer* possui um identificador (ID) e uma senha, que são usados para decifrar a chave privada no ambiente pessoal do usuário do *peer*.

O JXTA é compatível com o *Java Cryptography Extension* (JCE), para fornecer recursos de privacidade, criptografia e autenticação nativos do Java [55].

3.6 Comunicação

A seguir será mostrado como é o mecanismo de comunicação entre *peers* da rede JXTA.

3.6.1 Localização de Anúncios

Os anúncios (*advertisements*) são importantes na rede P2P, porque é através deste mecanismo que um *peer* publica o que tem disponível e assim outros *peers* podem localizá-los. Na plataforma JXTA os *peer*s conectados utilizam três métodos para a descoberta dos anúncios: Descoberta em cache, Descoberta Direta e Descoberta Indireta.

3.6.1.1 Descoberta em Cache

Considerada a maneira mais simples e rápida de localizar um anúncio, é feita de maneira passiva, ou seja, parte do princípio que existe uma lista que contém o endereço e portas de *peers*. Desta forma, elimina-se totalmente o processo de descoberta, diminuindo assim o tráfego na rede, além de ser de fácil implementação [58].

Um problema nesse método é que pode fazer referências a serviços que não estão mais disponíveis na rede, fazendo com que o *peer* gere tráfego desnecessário na rede. A solução encontrada é determinar um TTL (*time to live*) para cada anúncio [62].

A seguir, na Figura 3-3, pode-se observar um exemplo disso.

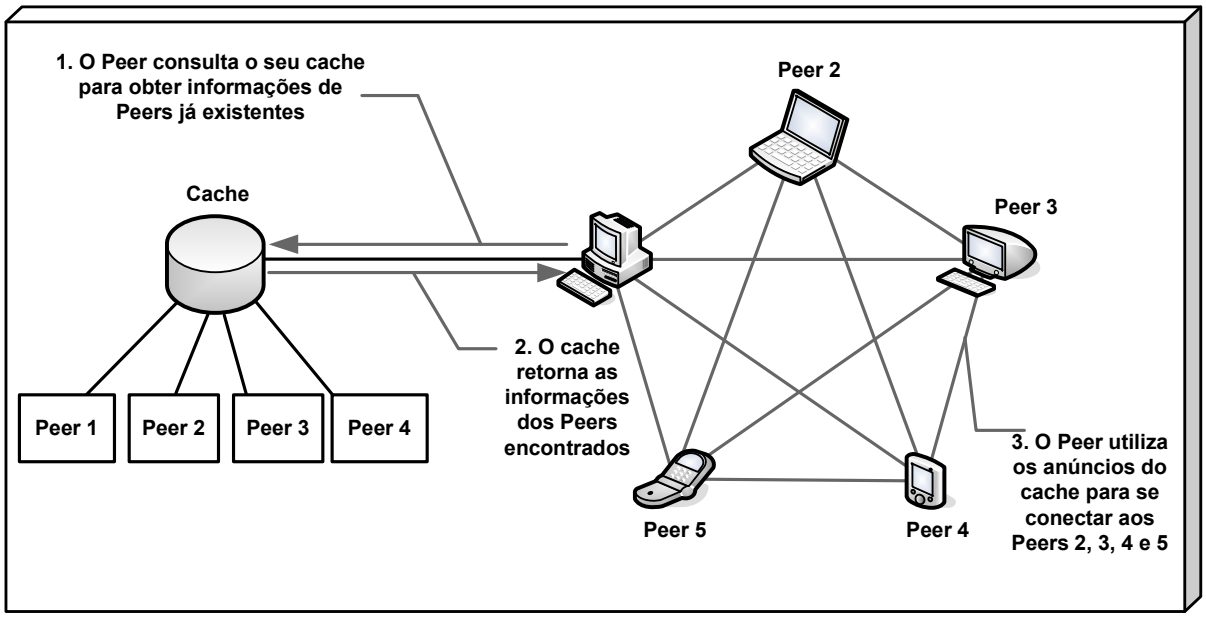


Figura 3-3 Funcionamento da descoberta em cache [58]

3.6.1.2 Descoberta Direta

Se os *peers* estiverem na mesma LAN, a descoberta dos *peers* vizinhos é feita por meio de um *broadcast* ou *multicast* [62].

Uma vez descobertos os *peers* da rede local, utilizando-se dessa abordagem de localização, o *peer* poderá se comunicar diretamente com outros *peer*s, sem a necessidade de realizar novamente o *broadcast* ou *multicast*. Porém, esta abordagem é exclusiva para LAN. Se outros *peers* estiverem fora da rede local, é necessária a utilização da abordagem de descoberta indireta [62].

A Figura 3-4 ilustra um exemplo de como a descoberta direta pode ser feita.

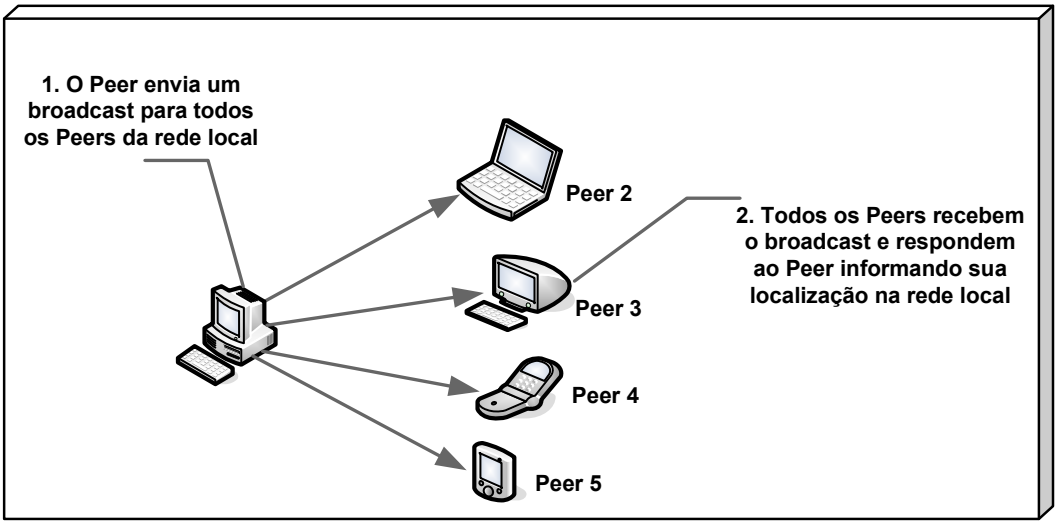


Figura 3-4 Funcionamento da descoberta direta [58].

3.6.1.3 Descoberta Indireta

A abordagem de descoberta indireta é utilizada por *peers* que não estão aptos a fazer *broadcast* ou *multicast*, ou ainda para acessar um *peer* externo à rede local. O diferencial dessa técnica é o uso de um *peer rendezvous* que atuará com uma central de informações de outros *peers* ou anúncios (*advertisements*).

Para localizar os anúncios usando o *peer rendezvous*, a propagação pode ser usada. Isto significa que o *peer rendezvous* repassa a solicitação aos *peers* que conhece, os quais, por sua vez, também poderiam passar a solicitação adiante aos *peers* por eles conhecidos até chegar ao destino. Outra maneira é o *peer rendezvous* consultar seu próprio cache para responder [62] ao *peer* solicitante.

Na descoberta indireta, existe o problema que os *peers rendezvous* podem propagar indefinidamente suas solicitações, criando um tráfego desnecessário na rede. A solução proposta é criar um TTL (*time to live).* Desta forma, o anúncio é descartado após circular, por um determinado “tempo”, na rede [62].

Existe ainda uma solução que é guardar informações do caminho na solicitação. Dessa forma, se um pedido passar duas vezes pelo mesmo local, será descartado.

Abaixo, na Figura 3-5, pode-se observar o funcionamento da descoberta indireta.

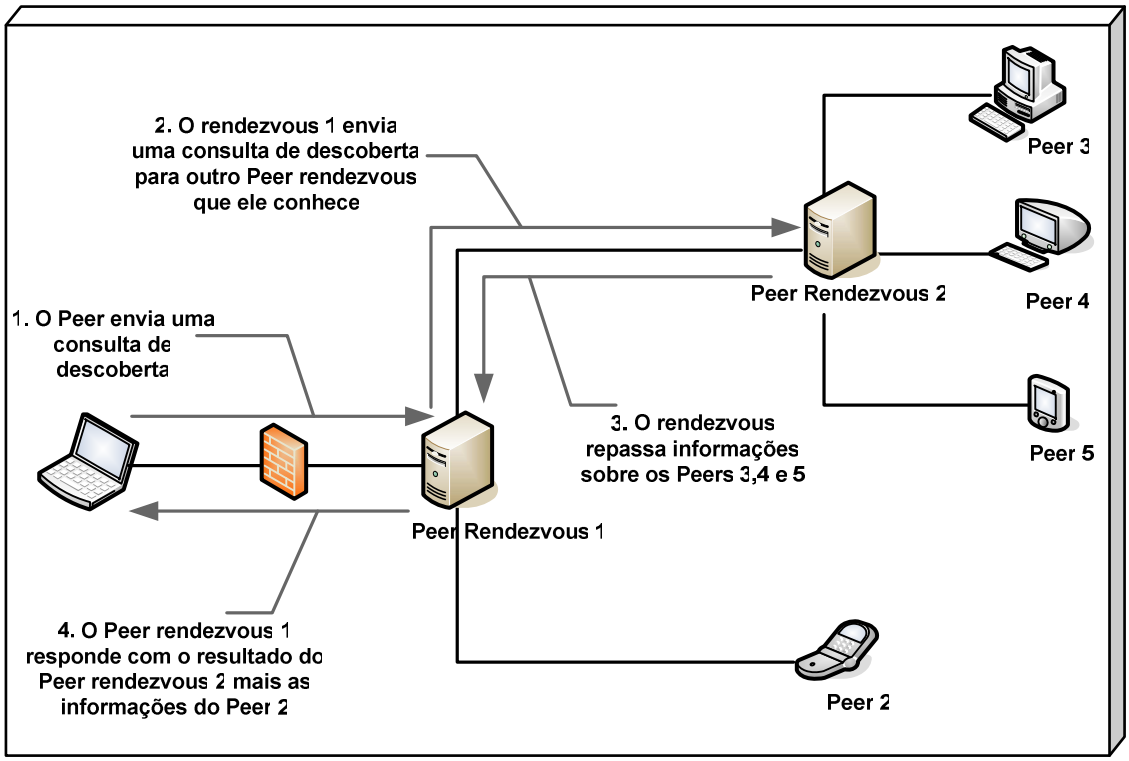


Figura 3-5 Funcionamento da descoberta indireta[58].

3.6.2 Rotas e *Peers* de Encontro (*Rendezvous*)

Uma rede P2P é funcional, se um *peer*, a partir de uma rede local, conseguir acessar outro *peer* que está posicionado em outra rede distinta. Porém, muitas vezes há obstáculos que dificultam essa comunicação, dentre eles o *firewall*. A alternativa, na forma indireta, é utilizar os *peers rendezvous* e os *peers router* da rede [62].

3.6.3 Atravessando *Firewall* e NAT através de *Relay*

Para transpor os *firewalls* e criar conexões com *hosts* fora do *Firewall*/NAT *Gateway é* utilizado um protocolo permitido pelo *Firewall* sob o qual se cria a conexão externa. Freqüentemente utiliza-se o protocolo HTTP, protocolo de pedido e resposta. No entanto, este protocolo pode abrir uma conexão para fora da rede, mas não permite que outros *peers* atravessem o *Firewall*. Então, a conexão é realizada da seguinte maneira: o *peer* externo conecta-se a um *peer relay*, enquanto o *peer* que está atrás do *Firewall*, de tempos em tempos, estabelece uma conexão com o *peer relay* (usando HTTP) para obter todas as mensagens que estão chegando como resposta ao HTTP. Desta forma, os dois *peers* podem se comunicar através do *relay* [62]. A Figura 3-6 ilustra o uso do *peer relay*.

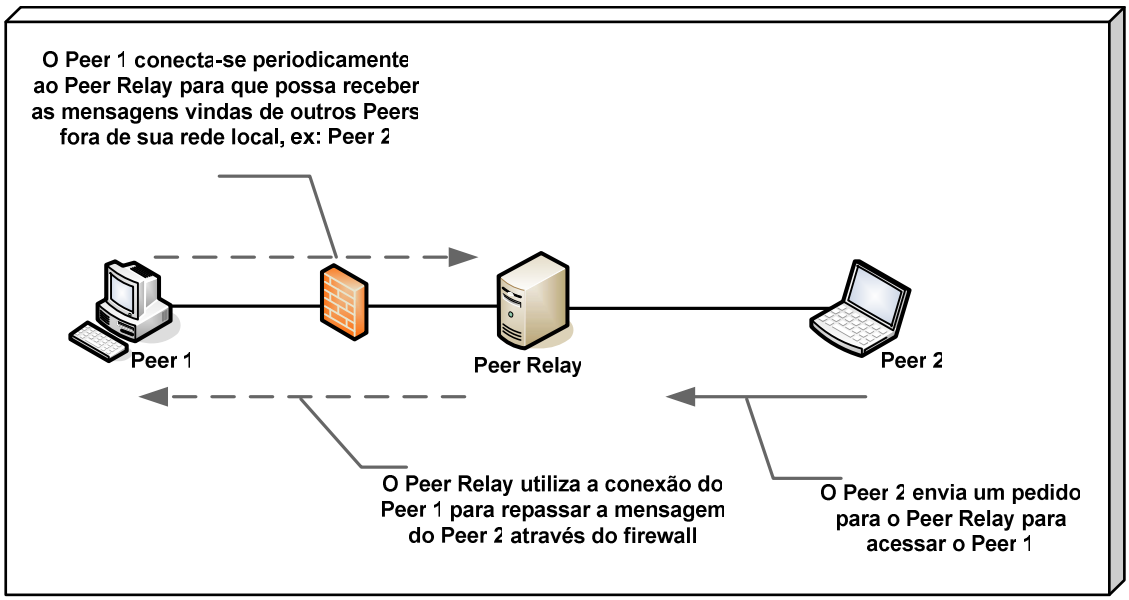


Figura 3-6 Atravessando *Firewalls* usando um *peer Relay [58]*

3.7 SDSI/SPKI

Na Internet, muitos de seus protocolos empregam chaves públicas para propósitos de segurança e requerem uma infra-estrutura de chaves públicas, ou PKI, para o gerenciamento dessas chaves. Por isso, um grupo do IETF[8] ficou responsável pela criação de padrões para o formato dos certificados, assinaturas associadas e outros formatos, e ainda pelos protocolos de aquisição de chaves. Nessa seção descreveremos a infra-estrutura de chaves públicas SDSI/SPKI, uma PKI que não se baseia em entidades centralizadoras para certificação e nem constrói cadeias de autenticação, mas sim de autorização.

3.7.1 Fundamentos de SDSI/SPKI

A criação do SDSI (*Simple Distributed Security Infrastructure*) e do SPKI (*Simple Public Key Infrastructure*) facilitou a construção de sistemas distribuídos escaláveis e seguros. O SDSI/SPKI foi motivado pela limitação e pela complexidade da infra-estrutura de chaves públicas X.509, derivada da hierarquia global de nomes do X.500. O SPKI [63] é uma infra-estrutura de chaves públicas que tem como características ser descentralizada e ter o uso de espaço de nome local. O SPKI [64] foi desenhado com a intenção de ser um modelo de

[8] IETF: The Internet Engineering Task Force. http://www.ietf.org/

autorização simples e flexível, muito bem definido e de fácil implementação. A união do SPKI e SDSI resultou em um sistema de autenticação e autorização para aplicações distribuídas [65]. Alguns autores fazem menção a esta união apenas como SPKI.

O emissor de um certificado é um *principal* (sujeito, administrador ou guardião de um serviço) que cria um certificado para delegar permissões de acesso a outros *principais* no sistema [15].

Na infra-estrutura SDSI/SPKI, existem dois tipos distintos de certificados: um para nomes e outro para autorizações. Os certificados de nomes fazem uma ligação de nomes a chaves públicas ou, ainda, a outros nomes. A forma de nomeação é adotada do SDSI e, ao invés de se utilizar nomes globais, cria-se um espaço de nome local correspondendo ao espaço de nomes do emissor do certificado. O emissor do certificado é sempre identificado pela sua chave pública. A combinação: chave pública mais nome local forma um identificador global único [66].

Tabela 3-1. Certificado de Nomes SPKI

Campos	**Descrição**
Emissor (*Issuer*)	Chave pública da entidade (chave emissora) que está definindo o “Nome” em seu espaço de nomes local.
Nome (*Name*)	Nome local que está sendo atribuído ao sujeito.
Sujeito (*Subject*)	Uma chave pública ou um nome definido em outro espaço de nomes que será redefinido (referenciado) no espaço de nomes local ao emissor.
Validade (*Validity dates*)	Especificação do período de validade do certificado – em formato ‘data-hora’.

Por ser um modelo igualitário, o SPKI e seus *principais* são chaves públicas que podem assinar e divulgar certificados, como as CAs do X.509. Assim, qualquer *principal* pode criar seu par de chaves (privada e pública), associar a chave pública do par a um nome no seu espaço local de nomes e divulgá-la através de um certificado. Não há uma entidade centralizadora para registro de chaves públicas e emissão de certificados como a CA da PKI X.509. Assim, cada *principal* define da maneira que lhe parecer mais intuitiva em seu espaço de nomes, os nomes que deseja atribuir aos outros *principais* de seu relacionamento. Só o *principal* que emite um certificado pode revogá-lo [15].

O SPKI contempla o uso do certificado de nome fazendo uma referência a um nome de um outro certificado publicado que está no espaço de nomes de outro *principal* e, assim, sucessivamente. Isso forma uma seqüência de nomes ligados (*linked names*). Esta seqüência de nomes deve ser reduzida a uma chave pública, que representa o *principal*, sendo referenciado quando se deseja a sua identificação.

Os certificados de autorização SPKI (Tabela 3-2) ligam autorizações a um nome ou a uma chave. Através destes certificados, o emissor delega permissões de acesso a outros *principais* no sistema.

Na infra-estrutura SPKI os certificados de autorização são construídos a partir das ACL's do guardião. Na verdade, no que se refere aos certificados e ACL's, o SPKI define um formato único de representação, como ilustrado na Figura 3-8, facilitando as atribuições e verificações de autorização. O conteúdo do certificado pode ser o mesmo da ACL. Porém, é acrescido ao certificado o campo do emissor assinando o certificado – a ACL não possui este campo porque é local ao guardião do serviço.

Tabela 3-2. Certificado de Autorização

Campos	Descrição
Emissor (*Issuer*)	Chave pública do emissor do certificado.
Sujeito (*Subject*)	Chave pública (ou *hash* dessa) ou nome identificando o *principal* que receberá a autorização.
Delegação (*Delegation*)	Valor lógico (*True/False*), indicando se o sujeito pode (*True*) ou não (*False*) propagar a autorização que lhe foi delegada pelo emissor [67].
Autorização (*Authorization*)	Contém as permissões concedidas pelo emissor – representadas como S-*expression*.
Validade (*Validity dates*)	Especificação do período de validade do certificado – em formato 'data-hora'.

3.7.2 Processo de Verificação de Autorização e Autenticação

No momento em que um *principal* (cliente) quiser fazer acesso a um determinado serviço, será necessário que o mesmo apresente a cadeia de certificados de autorização. A função da cadeia é provar que o cliente está autorizado a acessar o serviço desejado. Como mencionado anteriormente, o guardião do serviço têm como responsabilidade verificar se a cadeia fornecida é válida e em seguida confrontar o certificado com a política na ACL do serviço. Dessa maneira, pode-se verificar se o *principal* realmente tem os direitos exigidos. No caso das duas verificações ocorrerem com sucesso, a operação solicitada é executada [15].

O processo de autenticação no SPKI não é muito complexo, isto é, envolve apenas a comprovação de posse da chave privada – através da assinatura digital – que corresponde à chave pública autorizada a realizar o acesso [15].

Uma visão geral de todo processo de solicitação de recurso até sua obtenção é ilustrado na Figura 3-7.

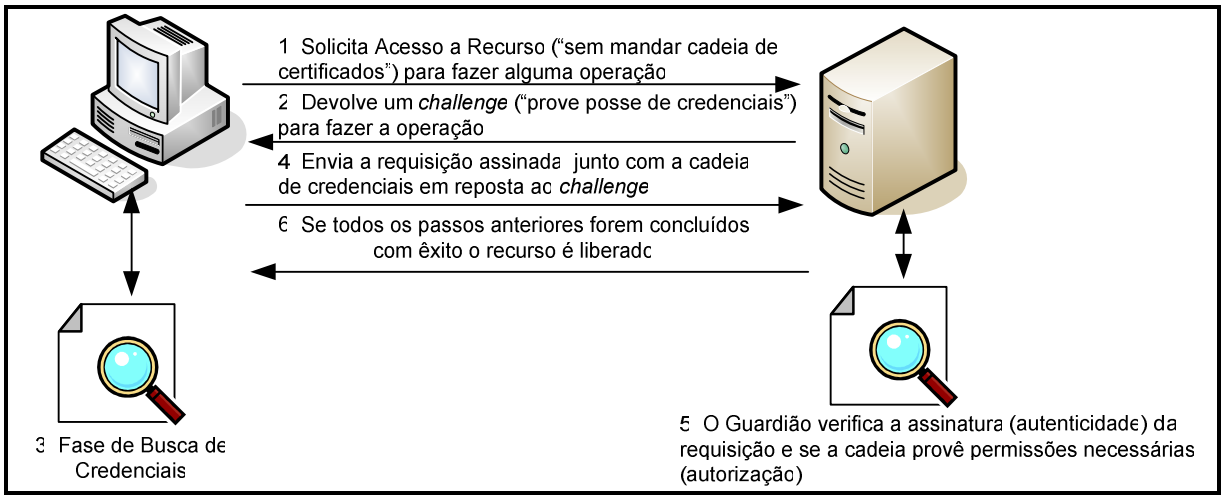


Figura 3-7 Cliente acessando um objeto protegido por SDSI/SPKI

No caso do passo 5, se for bem sucedido, significa que há um caminho de confiança. Em outras palavras, "uma seqüência de entidades confiáveis" por onde foram propagados os certificados até o cliente. Este caminho de confiança tem uma função similar a hierarquia de CAs de X.509 [15].

Na seqüência um detalhamento do passo 5 (autenticação e autorização) é ilustrado na Figura 3-8.

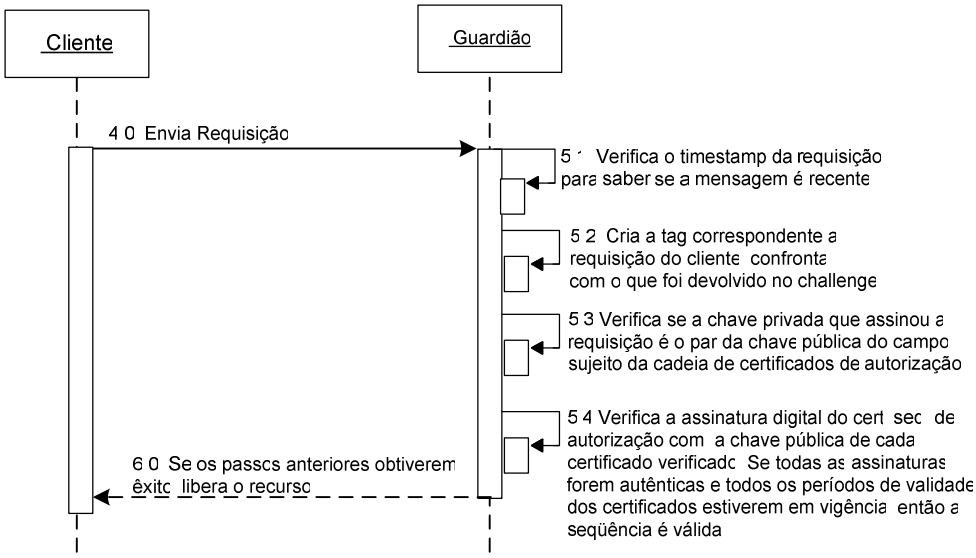


Figura 3-8 Efetivação de controle de acesso no SDSI/SPKI

Se todos os passos acima na Figura 3-8 forem bem sucedidos, significa que toda a autenticação e o controle de acesso ocorreram de maneira satisfatória. O acesso é então garantido.

3.8 –DHT - (*Distributed Hash Tables*)

As DHTs são tabelas hash distribuídas [68], que possuem a função de fazer um mapeamento de chaves em valores como as tabelas hash tradicionais, mas o que difere estas tabelas das tradicionais é que é feito um particionamento da tabela e então dividido entre os nós da rede. As chaves são obtidas semelhante à uma tabela hash tradicional, isto é, obtidas a partir da aplicação de uma função hash no dado que se deseja armazenar. O resultado dessa função é um par <chave, valor> que é armazenado na tabela.

É de responsabilidade de cada nó o fornecimento do mapeamento para um grupo de chaves e estas chaves são distribuídas na DHT de acordo com o identificador dos nós. A identificação única (ID) do nó é obtida através do mesmo processo usando a função hash aplicada para criar as chaves da DHT. Os nós possuem também uma tabela de roteamento com informações de outros nós, e serve para localizar nós adjacentes ou também chamados, nós vizinhos [69].

Segundo Rowstron e Druschel [29], uma tabela de *hash* distribuída, ou DHT, pode estruturar uma rede *peer-to-peer*. Essa estruturação pode ser usada para facilitar a

implementação de várias aplicações *peer-to-peer*, tais como serviço de compartilhamento de arquivos, DNS, caches Web, etc. [70].

As DHTs foram inicialmente introduzidas na comunidade científica em 2001, onde nessa mesma época foram propostas simultaneamente quatro arquiteturas: Chord [28], Can [42], Pastry [43] e Tapestry [44].

3.8.1 Bamboo

O Bamboo [71], é baseado no Pastry [43], na verdade uma re-engenharia dos protocolos do Pastry, tornando-se quase uma nova DHT. O Bamboo foi criado por Sean Rhea, da Universidade de Berkeley na Califórnia [72]. Seu código foi escrito em Java e está disponível sob a licença GPL [73].

O termo geometria é usado para se referir ao padrão de ligações entre vizinhos em uma DHT, independente dos algoritmos de roteamento ou gerenciamento de vizinhos usados. O Bamboo usa a geometria do Pastry. Isso não significa, entretanto, que utiliza os mesmos algoritmos de entrada na rede ou gerenciamento de vizinhos.

Comparado ao Pastry, o Bamboo e seus algoritmos possuem melhorias que lhe dão uma robustez maior quando há grandes incrementos no número de *peer* na DHT e em momentos de alto grau de *churn* (entrada e saída) na rede, principalmente em ambientes com largura de banda limitados.

O Bamboo associa a cada um de seus *peer*s um ID único de 160 bits (algoritmo de *hash* SHA1). O conjunto de IDs de *peer*s existentes estão distribuídos de maneira não uniforme, e estão baseados nos valores gerados pela função *hash* que pode ser aplicada a qualquer dado (endereço IP e número de porta ou chave pública, por exemplo). Na Figura 3-9 é ilustrada uma “tabela de roteamento” por IDs. Uma mensagem endereçada a uma certa chave *k* é roteada para o *peer* que possua um ID numericamente mais próximo do ID daquela chave entre os *peer*s ativos na rede responsável por aquela faixa de IDs [43] e [74].

O Bamboo também oferece um controle de acesso aos pares armazenados na tabela, sendo que sua utilização é opcional. Esse controle possibilita que somente os *peer*s que inserem determinados valores na rede tenham a capacidade de alterá-los. Esse controle é realizado através do fornecimento de uma senha durante a inserção do par (chave, valor).

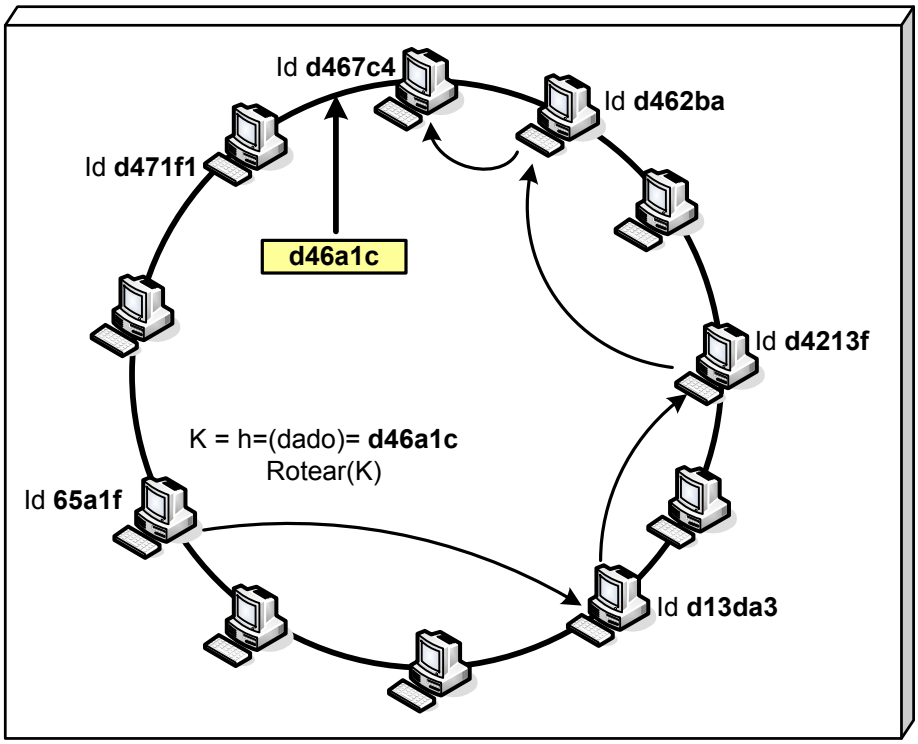


Figura 3-9 Estrutura do Bamboo

3.9 Considerações Finais

Nesse capítulo, foram apresentados os detalhes das tecnologias envolvidas, o JXTA e seu funcionamento, bem como características e funcionamento do SDSI/SPKI e seus processos de autenticação e controle de acesso. Ao final do capítulo, foi apresentada uma visão geral do Bamboo, que nesse projeto é utilizado como repositório distribuído (DHT) para o armazenamento das informações, tais como: chaves, certificados e votos atribuídos aos *peer*s e aos recursos que os compartilham.

No capítulo 4 serão apresentados trabalhos que têm alguma relação com a idéia proposta nesse projeto, a consideração de detalhes mais importantes do seu funcionamento e alguns comentários críticos.

Capítulo 4

4 Trabalhos Relacionados

4.1 Introdução

Na literatura técnica, muitos trabalhos propõem métodos para assegurar propriedades de segurança aos conteúdos distribuídos na rede P2P.

Nas seções subseqüentes, são abordados trabalhos nos quais aspectos de segurança, tais como autenticidade, reputação, controle de acesso, integridade e anonimato tentam ser garantidos.

4.2 Autenticidade

A autenticidade do conteúdo (documento) pode ser definida de várias formas. De acordo com [1], pelo documento mais antigo (assume-se que o primeiro postado na rede é autêntico), baseado em especialistas (o conteúdo é avaliado por um especialista que emite sua opinião sobre a autenticidade do conteúdo), ou baseados em voto (esta abordagem se diferencia da baseada em especialistas uma vez que um conjunto de especialistas votam para decidir a autenticidade de um conteúdo).

Além disso, de acordo com [1], a autenticidade de conteúdo pode ser avaliada por mecanismos baseados em reputação, que objetivam coletar e compartilhar opiniões desenvolvidas entre *peer*s sobre a credibilidade de um compartilhamento. Não é difícil rastrear reputações de *peer*s em uma rede P2P centralizada como o Napster, porque em um servidor central a busca por conteúdo é facilitada pelo servidor. Porém, em uma rede P2P descentralizada isso pode ser um problema; é necessário “rastrear” a reputação de um *peer* a partir dos outros *peers* da rede.

Para assegurar a autenticidade de conteúdo no projeto Poblano, baseado na plataforma JXTA [75], são aplicados o certificado de nome X.509 emitido por uma CA ou são aplicados certificados auto-assinados para fazer assinaturas digitais baseadas em chaves públicas obtidas do certificado. O Poblano também pode gerenciar o certificado baseado em uma rede parecida com o modelo *Web of Trust* do PGP (*Pretty Good Privacy*), fazendo, deste modo, uma rede de confiança [76].

Em outro trabalho [77] é utilizada uma infra-estrutura de chaves públicas (SPKI/SDSI) para garantir a autenticidade, sem ferir a propriedade do anonimato em redes *peer-to-peer*.

4.3 Reputação

Dependendo do valor que um usuário atribui a um conteúdo, o usuário considera os riscos de baixá-lo ou não de uma rede P2P. Se o usuário realmente quer o conteúdo terá que se sujeitar ao risco de baixar o conteúdo de um *peer* desconhecido. Para tentar minimizar o trauma da situação, foram criados os sistemas de reputação, que tentam prover um mecanismo que referencia o comportamento de um determinado *peer* para o usuário (cliente da rede P2P).

Segundo Damiani [76] os sistemas de reputação estão a mercê de algumas situações, devido a abordagem adotada:

a) ***Pseudonyms***: muitos sistemas *peer-to-peer*, na identificação dos usuários, utilizam pseudônimos. Com está técnica de identificação, um *peer* pode obter uma nova identidade, criando um novo pseudônimo. Ingressando novamente na rede com uma nova identidade, não há nenhum mecanismo que vincule o novo pseudônimo a identidade antiga que estava com má reputação, por exemplo.
b) ***Cold-start***: ocorre quando os *peers* que entram na rede nunca são selecionados para fornecer conteúdo, pois sempre as primeiras opções são os *peers* com grau de reputação maior. Dessa forma fica difícil para um novo *peer* aumentar sua reputação.
c) ***Pseudospoofing***: já que não é difícil a criação de outras identidades, um *peer* pode criar várias e usá-las como falsas testemunhas para aumentar a sua reputação.
d) ***Shilling***: esse problema é muito parecido com o anterior (*Pseudospoofing*). A diferença é que cria múltiplas identidades com diferentes endereços IP.
e) ***Load problems***: se arquivos são compartilhados tanto por *peers* com alta reputação quanto com baixa, os *peers* com alta reputação sempre serão os escolhidos para

fornecer o conteúdo, o que com o passar to tempo pode acarretar sobrecarga a estes *peers*.

Os sistemas de reputação tentam fornecer algum grau de credibilidade na tomada de decisão dos *peers* consumidores (clientes). A seguir, serão apresentados e avaliados alguns sistemas de reputação, baseando-se nas seguintes categorias descritas em [78]:

a) ***Self-policing***: um *peer* deve ser capaz de utilizar o sistema de reputação sem a ajuda ou supervisão de uma autoridade central;
b) ***Anonymity***: o sistema de reputação deve usar identidades indiretas, tais como pseudônimos ou nomes de usuários, e em hipótese alguma usar endereços IP para identificar usuários;
c) ***Overhead***: o sistema deve produzir um mínimo de custo na computação, infra-estrutura, armazenamento e complexidade de mensagens;
d) ***Malicious collectives***: o sistema deve ser robusto o suficiente para resistir a um conjunto de *peer*s que possuem conhecimento mútuo e que coletivamente tentam subverter o sistema.

4.3.1 Sistema de Reputação para Redes P2P

O trabalho de Gupta [79] é composto por duas abordagens presentes em seu sistema de reputação. Na primeira abordagem é utilizada a modalidade de débito-crédito (DCRC: *Debit-Credit Reputation Computation*) e na outra abordagem somente a de crédito (CORC: *Credit Only Reputation Computation*).

No caso do mecanismo DCRC, toda vez que um *peer* provê conteúdo, são acrescidos créditos no seu escore e quando o mesmo faz *downloads*, são feitos decréscimos dos créditos que possui baixando o seu escore.

Por outro lado o mecanismo CORC, somente faz créditos no escore do *peer* quando o conteúdo é fornecido pelo mesmo, mas não há nenhum desconto dos créditos quando o *peer* realiza *download.* Ao invés disso, utiliza-se a expiração do escore, servindo assim como uma forma de débito.

Os escores de reputação dos *peers* são armazenados localmente. E para assegurar que os *escores* de reputação sejam sempre confiáveis, nesse sistema é utilizado um agente de computação de reputação central (RCA: *Reputation Computation Agent*), que faz a pontuação na reputação dos *peer*.

Tanto no primeiro como no segundo mecanismo, um *peer* pode optar por não ter sua reputação rastreada. Porém, isso implicará em sempre receber um valor de escore com valor igual a 0, isto é, é criado um valor mínimo de reputação para os *peers* dentro do sistema.

Quando quer entrar nesse sistema, um *peer* gera um par de chaves (pública, privada) e as registra no RCA central, que por sua vez também possui um par de chaves (pública, privada) e sua chave pública é de conhecimento de todos os *peer*s. Da mesma forma, todos os *peer*s podem ter as chaves uns dos outros.

Para que os *peers* obtenham os seus escores por sua contribuição na rede, entram em contato com o RCA a cada período de tempo. Através da interação entre os *peers* são gerados os PPs (*Proof of Processing*), que são parâmetros armazenando informações como a identidade do *peer* que baixou um conteúdo, o *time stamp* e outras informações sobre sua interação. A partir desses valores de PPs, o RCA calcula os escores dos *peers* e os assina com sua chave privada. A seguir, são abordados os aspectos das categorias anteriormente citadas para o trabalho de [79]:

Self-policing: na avaliação esse item não é contemplado por esse sistema de reputação, uma vez que usa um agente central para auxiliar os *peers*;

Anonymity: este item é satisfatoriamente atendido, pois utiliza o pseudônimo para a identificação dos *peers*;

Overhead: como os valores de confiança ficam armazenados localmente e quem calcula os escores com base nos PPs é o RCA, presume-se que o *overhead* computacional no *peer* seja baixo. Em relação à troca de mensagens das chaves entre o *peer* e o RCA, segundo Gupta [79] o sistema tenta otimizar as trocas para garantir o mínimo de *overhead*;

Malicious collectives: na abordagem do CORC não existe débito permitindo que um *peer* ajude ao outro a obter maior reputação na rede, mas o sistema pode evitar esse problema analisando os PPs. Entretanto, a abordagem do DCRC tem débitos quando o *peer* faz *download*, mas o mesmo pode criar várias identidades, não sendo possível evitar *Malicious collectives* .

4.3.2 EigenTrust

Esse é um sistema de reputação projetado por Kamvar e outros [78], que usa o conceito de confiança transitiva. Ou seja, um *peer* pode dar mais importância às opiniões de outros *peers* que confia mais. Dessa maneira, para determinar quanto um *peer* A confia em

um *peer* B, dependerá dos valores de confiança que o *peer* B recebe dos *peers* que fazem parte do círculo de confiança do *peer* A.

No EigenTrust, cada *peer* possui um conjunto de *peer*s que mantém seus valores de confiança, chamados gerenciadores de escore. Estes valores são guardados em uma tabela de *hash* distribuída (DHT). Estes gerenciadores de escore são associados à identidade (ID) de cada *peer*. Os IDs dos *peer*s são compostos pelo endereços de IP e porta TCP. Quando um *peer* cliente deseja conhecer o valor de confiança de outro *peer* (servidor), simplesmente calcula o *hash* do ID do servidor e obtém os IDs dos gerenciadores de escore do mesmo. Então, o *peer* cliente pode consultar os gerenciadores de escore e obter os valores de confiança do servidor que deseja baixar algum conteúdo. Baseado nas categorias de um sistema de reputação anteriormente citadas [76], a seguir são descritos as informações sobre o EigenTrust:

Self-policing: os *peer*s podem executar o sistema sem a ajuda ou supervisão de uma terceira parte, então o sistema faz o *self-policing*.

Anonymity: a identificação de um *peer* no EigenTrust, é feita através do *hash* do endereço de IP. A identidade dos gerenciadores de escore do *peer* é encontrada através do cálculo do *hash* de um endereço IP.

Overhead: o escore de reputação de um *peer* é armazenado em um número limitado de gerenciadores. Quando um *peer* quer conhecer o escore de reputação de outro, pode facilmente contatar o gerenciador de escore do qual deseja tal informação. Assim sendo, neste sistema, o *overhead* no sistema total é mais baixo do que no sistema XRep, por exemplo, onde várias mensagens devem ser trocadas para obter tal informação.

Malicious collectives: no sistema de reputação EigenTrust, a opinião de um *peer* malicioso é menos importante do que a opinião de um *peer* confiável (confiança transitiva). Portanto, é difícil para os *peers* colaboradores subverterem o sistema, aumentando a reputação uns dos outros através da troca de arquivos para computar votos positivos nos gerenciadores de escore.

4.3.3 Xrep

O projeto Xrep é uma extensão do protocolo Gnutella [31]. Diferente dos sistemas de reputação descritos previamente, o Xrep não se baseia somente na reputação do *peer*, mas

também na reputação do conteúdo, como mencionado em [76]. Por esta razão, o Xrep é um sistema de reputação distribuído e completo.

Quando um *peer* solicita um conteúdo, todos os *peer*s que possuem a palavra chave combinando com tal conteúdo respondem à consulta informando um *hash* do conteúdo.

Para selecionar o *peer* apropriado para baixar o conteúdo, uma nova consulta é feita para outros *peer*s, perguntando pela reputação dos candidatos a *peer* servidor e seus conteúdos. Os *peers* que respondem a essa solicitação, enviam seus endereços IPs e suas opiniões sobre os *peers* servidores e seus respectivos conteúdos.

A seguir, o *peer* cliente avalia as opiniões e julga a reputação tanto do conteúdo quanto de sua fonte (*peer* servidor). Essa tarefa é feita verificando os votos ou opiniões diretamente com os *peers* que opinaram (através dos seus endereços IPs – enviados no passo anterior). Isso é feito para prevenir *shilling*.

Na seqüência, o *peer* cliente escolhe o *peer* servidor o qual julga ser o mais confiável e verifica se realmente possui o conteúdo que veio na resposta de sua busca inicial, validando a existência do conteúdo.

Por fim, o conteúdo é baixado do *peer* servidor escolhido. Quando o *download* é completado, verifica-se a integridade usando uma função de *hash*. O *peer* cliente então emite seu parecer sobre o conteúdo e o respectivo servidor, atualizando seus repositórios. Baseado nas categorias de sistemas de reputação anteriormente citadas [76] o Xrep será classificado:

Self-policing: Os *peers* podem utilizar o sistema de reputação sem a ajuda ou supervisão de uma autoridade central. Dessa forma atende a este requisito.

Anonymity: A identificação dos *peers* é feita através de pseudônimos, porém, o sistema tem limitações, pois quando um *peer* emite sua opinião sobre a reputação de outro *peer* envia juntamente seu endereço IP, deixando de ser anônimo nesse momento. Os *peers* maliciosos podem, por exemplo, perguntar por seus próprios valores de confiança e atacar os *peers* que respondam com valores ruins de reputação.

Overhead: Para cada arquivo a ser baixado, o *peer* cliente tem que perguntar a reputação a outros *peers*, calcular os valores de confiança dos conteúdos e seus servidores. Este processo de votação leva muito tempo e usa vários recursos de computação e infra-estrutura, causando *overhead*.

Malicious collectives: Neste quesito, o sistema adota uma abordagem adequada, pois todas as respostas às consultas por reputação vêm com o endereço IP dos *peers* que opinaram

e que posteriormente serão verificados. Esta estratégia previne o uso de pseudônimos múltiplos no mesmo endereço de IP. O mecanismo de reputação desencoraja o conteúdo malicioso, pois *peers* com má reputação não serão selecionados para serem servidores de conteúdo.

4.4 Controle de Acesso

Como a grande maioria das aplicações *peer-to-peer* apenas se preocupam em compartilhar recursos, não existe preocupação com relação ao controle de acesso, no sentido de definir regras (políticas) para definir quem, quando e o que pode ser acessado [80]. Considerando as redes P2P para aplicações além do simples compartilhamento de conteúdos multimídia, faz sentido a adoção de mecanismos de controle de acesso.

Como o ambiente *peer-to-peer* é distribuído e requer anonimato, a tarefa de controle de acesso para conteúdos compartilhados se torna mais difícil, logo, não parece muito adequado usar os métodos clássicos. Segundo [81] um modelo de controle de acesso em *peer-to-peer* deve possuir as seguintes características:

- **Controle e Suporte não centralizado**: modelos de controle de acesso clássico tais como, ACL e RBAC [82] baseiam-se em servidores centrais para realizar as operações de autorização, uma única autoridade que identifica usuários, e define papéis ou grupos e controla os direitos de acesso. Entretanto, em redes *peer-to-peer* tais mecanismos centralizados de autorização não devem existir. Na verdade, um *peer* deve possuir alto grau de autonomia bem como gerenciar o acesso aos conteúdos neles armazenados. Um modelo de controle de acesso para redes *peer-to-peer* certamente precisa observar este requisito.
- **Classificação de *peer***: o anonimato de um *peer* é outro requisito em redes *peer-to-peer*, diferente do ambiente cliente/servidor onde há forte acoplamento entre os mecanismos de identificação e controle de acesso. Em P2P na maioria das vezes os *peers* não têm conhecimento uns dos outros. O controle de acesso em P2P deve fornecer mecanismos que permitam de alguma maneira classificar o *peer* cliente e designar-lhes diferentes direitos de acesso, ainda que os usuários sejam desconhecidos.

- **Encorajar compartilhamento de arquivos**: os motivos que levam os usuários à rede *peer-to-peer* é a disponibilidade e a diversidade de conteúdos que o sistema provê. Porém, com a adoção do controle de acesso, determinados *peers* podem não conseguir os conteúdos que desejavam. O controle de acesso não deve desestimular o compartilhamento por limitar o acesso de alguns *peers*. Os *peers* precisam estar convencidos de que com o compartilhamento de seus conteúdos, obterão os conteúdos que desejam.
- **Limitar a disseminação de conteúdo malicioso**: as características abertas e o anonimato nas redes *peer-to-peer* fazem com que *peers* maliciosos a usem como ambiente de disseminação de conteúdo danoso/malicioso, tal como pornografia, etc. Seguindo essa classificação, [81] propõe um **mecanismo de controle de acesso para sistemas de compartilhamento de arquivos**, ciente dos desafios de não comprometer a escalabilidade, não usar serviços centralizados e tratar o anonimato.

Na proposta de [81], são abordados aspectos de modelos de reputação e recomendação com esquemas de troca-justa (*fairness*) para fazer o controle de acesso. Nessa abordagem, o *peer* é considerado como um sistema *standalone* em que os arquivos compartilhados são tratados como objetos que precisam ser protegidos, e os *peers* que solicitam tais objetos são sujeitos que possuem ou precisam ganhar direitos de acesso. Existe um repositório de escores de onde são obtidos os valores-base para o cálculo que irá permitir o acesso ou não aos recursos da seguinte maneira: *direct trust* (quando o servidor P2P acredita diretamente no cliente P2P e em sua honestidade e confiabilidade, baseado em experiências anteriores), *indirect trust* (semelhante ao anterior, mas baseia-se em recomendações feitas por outros *peers*), *direct contribuiton* (características de contribuição do consumidor ao fornecedor em termos de volume de *download* e *upload* entre os dois) e *indirect contribuiton* (semelhante ao *direct contribuiton*, mas as informações são entre o cliente P2P e outros *peers*) [81].

Uma vez que uma transação entre *peers* se encerra, os valores *direct trust* e *direct contribution* são atualizados de acordo com o grau de satisfação do *peer* em relação a transação. Tais valores atualizados irão influenciar as novas avaliações para a tomada de decisão do controle de acesso [80].

Nessa abordagem, cabe ao *peer* cliente coletar – junto a outros *peer*s – recomendações que o habilitem a calcular seus valores de escore perante um outro *peer*.

Em [83] os autores propõem uma abordagem, baseada no modelo RBAC (*Role-based Access Control*) para controle de acesso em aplicações P2P. É definido um *middleware* que atua como um *broker* (intermediário) na comunicação. Neste caso, diferente do exemplo anterior, uma entidade central (o servidor de políticas) é utilizada para armazenar políticas de acesso aos recursos. As políticas são transferidas periodicamente dos servidores para os *peers* para que as decisões sejam autônomas.

4.5 Integridade

Diversos mecanismos baseados em criptografia vêm sendo empregados para prover integridade às aplicações P2P [80]. Em [84] é proposto um mecanismo que permite que a integridade de um objeto possa ser verificada pelo *peer* que o recuperou. O *peer* que desejar inserir um objeto na rede calcula o *hash* de seu conteúdo (usando uma função conhecida) para produzir a chave que identifica o objeto. Ao recuperar um objeto da rede é feito o cálculo do *hash* do objeto e comparado à chave usada na busca. O objeto será íntegro se os valores forem iguais.

4.6 Anonimato

Uma maneira simples de identificar um *peer* é pelo seu endereço IP. Entretanto, este método é severamente limitado porque é vulnerável a *IP-Spoofing*. Além disto, os *peers* podem ter endereços IP dinamicamente atribuídos por seus provedores. No modelo *Identity Crisis* [85] é assumido que todos os *peers* usam a mesma identidade por toda a sua vida de *peer*. Este modelo usa chaves oriundas de certificados auto-assinados permitindo que *peers* bem comportados criem uma relação de confiança entre si durante uma série de desconexões e reconexões com diferentes endereços IP.

De acordo com [86], o anonimato pode ser visto em muitos aspectos, tornando difícil para os *peer*s descobrirem quem criou o arquivo, quem armazenou o arquivo, quem acessou o arquivo e quais documentos estão armazenados em um *peer*.

Os autores Marti e Garcia-Molina [85] também consideram que anonimato é visto como uma desvantagem em P2P, uma vez que pode abrir portas para vários problemas de segurança que preocupam os clientes P2P. É perfeitamente plausível confiar em um único

serviço centralizado, mas obviamente não é sábio confiar em uma multidão de provedores quaisquer de recursos anônimos em toda a rede P2P. Um *peer* malicioso pode facilmente iludir outros *peer*s e os *hackers*, assim como os *worms* podem fazer um *spoof* de uma identidade de *peer* para danificar todo um sistema P2P. O autor sugere a adoção de chaves assimétricas que preservam o anonimato e identificam os *peers* de maneira independente de tecnologia. Desta forma, a identificação do *peer* é feita usando sua própria chave ou *hash*.

No trabalho de Wierzbicki e outros [87], autenticação e anonimato são consideradas na construção de protocolo de autenticação com anonimato controlado para sistemas P2P. O protocolo é baseado em *Merkle's Puzzles* [88] e em *Zero-knowledge proofs* [89] e foi projetado para ser usando em sistemas de armazenamento P2P. Com essas técnicas o protocolo permite que um *peer* que deseje armazenar um conteúdo faça isso de maneira anônima, utilizando-se dos serviços de um *super-peer* ou *bootstrap* que gere os IDs para os *peers* da rede. Dessa forma somente o *super- peer* é que realmente conhece os IDs dos *peer* da rede.

4.7 Considerações Finais

As propostas encontradas na literatura técnica baseiam a segurança P2P na arquitetura segura de cliente-servidor adaptadas às limitações do P2P.

Como visto nesse capítulo, existem alguns trabalhos na literatura que tentam cobrir partes dessas necessidades. O projeto **Poblano** [76] utiliza certificados através de uma PKI hierárquica tradicional X.509 para garantir a autenticidade. Nessa visão clássica da autenticação, existe o endosso do certificado de nomes por uma CA. Esta estrutura é rígida e inflexível sendo oposta ao modelo *peer-to-peer*, no qual a identificação de um *peer* pode estar em constante mudança. Uma adaptação desse modelo é a utilização de certificados auto-assinados, mas que não vincula quem assina com o conteúdo fornecido. Dessa maneira o certificado não tem credibilidade. Por estas razões, torna-se mais difícil o seu gerenciamento em um ambiento global, como, por exemplo, a Internet [66].

O trabalho [79] não apresenta as características necessárias para compartilhamento de conteúdo malicioso. Este sistema não pode ser considerado mecanismo de reputação com limitações. Porém, serve como um bom exemplo de mecanismos de incentivo a colaboração.

O **Xrep** apresenta uma boa abordagem para reputação de *servent* e seu respectivo conteúdo, evitando *cold-start e pseudonym*. Se um sistema não é afetado pelo problema de

cold-start, pode facilmente espalhar conteúdos maliciosos. Entretanto, o Xrep faz a avaliação do conteúdo dos *peers* presentes na rede, logo o problema em questão não o afeta. Outra vantagem da avaliação por conteúdo é que proporciona balanceamento de carga, pois, mesmo que um *peer* não tenha uma das melhores reputações por ser novo na rede, por exemplo, pode fornecer conteúdo bem reputado. Outros *peers* podem requisitar *download* destes conteúdos bem reputados e consequentemente aumentarem a credibilidade do *peer* em questão. Como resultado positivo desta situação, temos que os *peer* com maior reputação ficam menos demandados por *download* e o balanceamento de carga efetivamente acontece naturalmente na rede P2P.

O mecanismo de reputação **EingenTrust** viola o anonimato para o serviço de escore, podendo ser alvo de *peers* maliciosos que descobrem os seus *peers* de escore, que podem sofrer ataques de negação de serviços, por exemplo. Dessa maneira o Eingentrust não se apresenta como a solução mais adequada para a reputação em redes *peer-to-peer*.

Nesse trabalho podemos destacar alguns pontos onde este trabalho atende as características expostas anteriormente. A chave pública pode ser usada como um Pseudônimo para identificação do *peer*. Como a identificação é baseada na criação de chaves públicas a tentativa de recriá-las várias vezes para tentar burlar o sistema de reputação se mostraria inapropriada devido ao custo computacional para gerar as chaves.

Quanto a questão do *Cold-start*, o usuário do sistema proposto neste trabalho vai poder escolher em suas pesquisas os critérios que melhor se encaixar as suas necessidades, dessa forma mesmo que um *peer* não tenha um grau de reputação muito alto poderá estar presente na lista apresentada ao usuário, caso o mesmo tenha os critérios determinados na busca do usuário.

Para inibir o *Pseudospoofing* este trabalho utiliza um sistema de reputação. Mesmo que o *peer* venha a criar identidades falsas, todas terão que ter uma boa reputação, caso contrário não serão utilizadas nas trocas de arquivos com outros *peer*. E sua opinião terá um valor insignificante na votação da credibilidade de outros *peers*.

O protótipo desenvolvido neste projeto, atende ao item de *Self-policing*, pois não precisa da ajuda ou supervisão de uma entidade central para funcionar na rede.

No que se refere ao anonimato, o sistema possui tal característica, pois sua identificação é baseada em chaves, diferente de outros sistemas que utilizam o endereço IP para a identificação dos *peers*.

O sistema possui algum *Overhead*, entretanto o custo-benefício torna-se satisfatório, pois se o conteúdo for grande e após baixá-lo o usuário perceber que o mesmo for malicioso ou ruim, isso irá gerar uma frustração por parte do usuário. Dessa maneira, a medida que o tamanho do conteúdo aumentar, maior também será a vantagem de utilizar as verificações de segurança e integridade do sistema para evitar o *download* de conteúdo indesejado ou ruim.

Uma idéia mais detalhada sobre esse custo-benefício será apresentada na seção de avaliação do protótipo.

Como as informações de reputação dos conteúdos e de seus fornecedores ficam armazenados de maneira distribuída, torna-se mais difícil o surgimento de *Malicious collectives*, tornando o sistema mais robusto para resistir a um conjunto de *peers* que possivelmente tentarão subverter o sistema.

Capítulo 5

5 Modelo Proposto

5.1 Introdução

Nesse capítulo é descrito o modelo que introduz uma camada de segurança intermediária entre a camada de aplicação e a infra-estrutura *peer-to-peer*.

5.2 Motivação

As redes *peer-to-peer*, apesar de sua escalabilidade, tolerância a faltas e auto-organização, mostram deficiências em segurança talvez por serem tão dinâmicas, distribuídas e porque não foram concebidas com tal preocupação.

As maiores restrições para o uso intensivo de redes *peer-to-peer*, destinada a fins que não sejam o mero compartilhamento de arquivos (principalmente de músicas e filmes), como por exemplo, em ambientes corporativos ou comerciais, são relacionadas aos aspectos de segurança. Há riscos envolvidos no uso profissional de redes P2P: o compartilhamento (intencional ou acidental) de informações de domínio privado, pode, por exemplo, causar prejuízos incalculáveis.

Muitos trabalhos vêm sendo realizados na área de segurança em redes *peer-to-peer* para tentar assegurar a **integridade,** a **autenticidade** e a **disponibilidade** dos conteúdos fornecidos pela rede [1]. Além disso, alguns trabalhos procuram criar meios para que os conteúdos disponibilizados tenham credibilidade, bem como seus provedores (usando para tal, mecanismos de **reputação e confiança**) [76], [78]. Porém, há a possibilidade de surgimento de *peers* mal intencionados que se unem em conluio entre *peers*, visando subverter o sistema de reputação.

Para redes *peer-to-peer* em ambiente mais profissional (restrito), o **controle de acesso** sobre conteúdos deve ser adotado, porém sem ferir o **anonimato**.

A identificação de um *peer* pode ser gerada de várias maneiras, como baseada em IP [28], UUID [55], Pseudônimo [76] etc. Entretanto esses métodos de identificação são passíveis de ataques por associação múltipla de identidades ao mesmo *peer*, por exemplo, ataque conhecido como *Sybil* [90]. Outro tipo de ataque à forma de identificação de *peer* baseada em IP é o *Spoofing* [85].

Nos ataques relacionados à identificação por IP mencionados acima, a vulnerabilidade existe porque não há uma relação forte entre a identificação e a prova da identificação que daria autenticidade à mesma. Além disso, este tipo de identificação é baseado em dados não persistentes. Por exemplo, um endereço IP dinâmico atribuído por provedor a um *peer* na rede. Dessa forma torna-se difícil criar um mecanismo de **irretratabilidade** que tente evitar que um *peer* negue falsamente alguma troca de mensagem realizada na rede *peer-to-peer*.

Outro problema inerente às redes *peer-to-peer* é conhecido como *free-rider* (nós/*peers* carona), que são *peers* que consomem conteúdos da rede, mas se negam a fornecer/compartilhar qualquer tipo de conteúdo em troca [91]. Para evitar este comportamento nas redes *peer-to-peer* são adotados mecanismos de compensação ou **troca justa**, que recompensam os *peers* de acordo com a sua colaboração com a rede. Esta recompensa é efetivada de várias maneiras, por exemplo, atribuindo maior prioridade em filas de espera para obtenção de conteúdos, ou só fornecer x *bytes* de conteúdo ao *peer* que compartilha x *bytes* com a rede [30].

Como efeito colateral dos mecanismos de troca justa, as redes *peer-to-peer* sofrem com uma quantidade muito significativa de **lixo P2P** (conteúdo corrompido ou poluído), pois para ganharem acesso à conteúdos alguns *peers* maliciosos compartilham grandes arquivos com conteúdos inúteis. Tal situação cresce em grande escala e em alguns casos chega a alcançar mais de 60% do número total de cópias de um conteúdo que é lixo P2P [2]. Se for considerado o fato inconveniente do armazenamento desse tipo de conteúdo indesejado, a rede perde também em disponibilidade de largura de banda, trafegando estes conteúdos poluídos. Além disso, a replicação passiva [30] de conteúdos poluídos faz com que a quantidade destes seja multiplicada de forma acelerada na rede *peer-to-peer*.

5.3 Objetivos

O modelo proposto visa oferecer os aspectos de segurança à rede *peer-to-peer* baseado na utilização de chaves públicas e de um mecanismo de escore (que calcula os valores de votos dados ao *peer* e seu conteúdo, conferindo ao mesmo um grau de reputação). Esses aspectos dizem respeito à autenticidade de conteúdo na rede, a sua integridade e em alguns casos a confidencialidade do mesmo.

Para atingir tais objetivos, na proposta almeja-se os seguintes objetivos específicos:

- Basear a infra-estrutura de chaves públicas (PKI) em SDSI/SPKI, que se mostra mais adequada às características de auto-gerenciamento, descentralização, escalabilidade e flexibilidade das redes P2P;
- Oferecer autenticidade, integridade, confidencialidade e um mecanismo de irretratabilidade aos conteúdos providos pelo *peers* na rede com base na ICP;
- Propor um mecanismo de reputação de *peers* e conteúdos baseado em um serviço de escore, visando reduzir comportamentos maliciosos dos *peers* da rede;
- Propor um mecanismo para prover credibilidade aos certificados auto-assinados emitidos pelo SDSI/SPKI;
- Propor um mecanismo distribuído e eficiente de armazenamento e recuperação de certificados e cadeias SDSI/SPKI;
- Propor um mecanismo distribuído de controle de troca justa que possa ser utilizado nos provedores/servidores de conteúdo P2P;
- Utilizar um esquema que minimize o comportamento de nós/*peers* carona e ataques *Sybil* no modelo proposto.
- Utilizar um esquema de gerenciamento de escores que possibilite ao interessado acesso direto ao mesmo.

5.4 Modelo

O modelo proposto está baseado numa arquitetura que interpõe segurança entre a camada de aplicação e a infra-estrutura *peer-to-peer*. A Figura 5-1 mostra a arquitetura geral do modelo proposto, sendo que o repositório é baseado em um modelo distribuído e a outra parte da arquitetura se encontra em cada *peer* da rede.

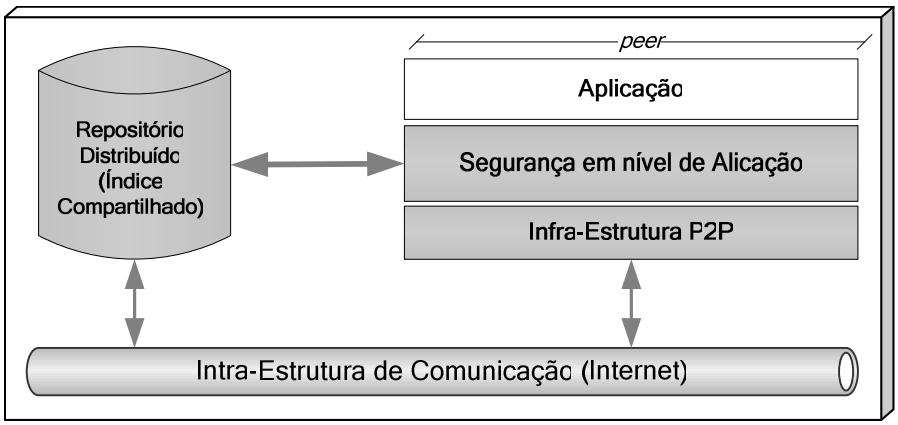


Figura 5-1 Arquitetura Geral do Modelo Proposto

Na camada de aplicação, estão os *softwares* que fazem uso da rede *peer-to-peer*. A camada da infra-estrutura representa a tecnologia, ou conjunto de protocolos, escolhidos para prover as funcionalidades para acesso à rede *peer-to-peer*.

A camada de infra-estrutura P2P oferece os recursos de localização, armazenamento e transporte de objetos P2P às camadas superiores, abstraindo a rede física (infra-estrutura de comunicação). Além disto, a camada P2P deve oferece canais seguros de comunicação entre *peers*, além dos recursos básicos para o funcionamento de um *peer* em uma rede P2P [92].

No repositório compartilhado, serão armazenadas as informações referentes aos aspectos de segurança e reputação. Já a infra-estrutura de comunicação depende da arquitetura de rede utilizada. Na Figura 5.1 está sendo indicada a arquitetura de rede mais comum, a Internet. A escolha por repositório distribuído visa substituir o método de busca distribuída mais comumente utilizado em P2P, o *flooding* (inundação), por seu método de busca determinística (utilizando DHT).

Interposta à camada de aplicação e à infra-estrutura P2P há a camada de segurança, que oferecerá as funcionalidades e propriedades de segurança mencionadas nos objetivos. A camada de segurança está baseada numa infra-estrutura de chaves (sistema criptográfico de chave pública e certificados) com os quais um *principal* poderá assinar seus documentos e delegar direitos de acesso a outros *peer*s da rede P2P. A chave pública também é utilizada para manter a identificação persistente do *peer*, dado que na rede P2P a identificação pode mudar a cada reinício do *peer*. A seguir a dinâmica de funcionamento do modelo é mostrada (Figura 5-2).

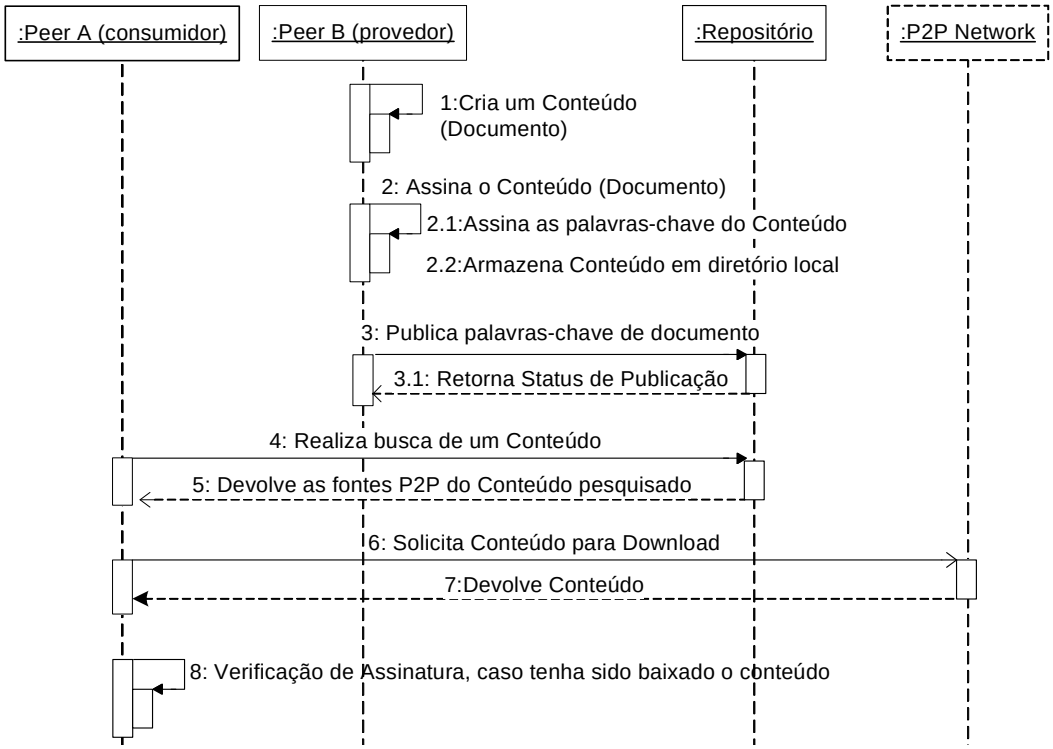


Figura 5-2 Dinâmica do Modelo

Na Figura 5-2, um *peer* B (autor/provedor do conteúdo), após gerar o conteúdo, (evento 1), gera as palavras chave que descrevem tal conteúdo e as assina com sua chave pública, armazenando o conteúdo localmente (eventos 2, 2.1 e 2.2). A seguir, (evento 3) o *peer* B faz a publicação das palavras chave no repositório.

No evento 4, considera-se uma busca por um conteúdo feita pelo *peer* A, o qual recebe do repositório (evento 5, Figura 5-2) uma lista contendo a(s) palavra (s) chave consultada (s) e seus respectivos provedores.

Uma vez que o *peer* A decide de onde baixar o conteúdo (qual será o *peer* provedor) solicita o conteúdo (evento 6, Figura 5-2), que é baixado do provedor da rede, *peer* B (evento 7). No evento 8 é feita a verificação da assinatura e da integridade do conteúdo.

Os eventos 6 e 7 da Figura 5-2 representam o mecanismo de troca justa (que será abordado em detalhes na seção 5.4.7).

A Figura 5-3 mostra os principais módulos que compõem a camada de segurança, que serão abordados a seguir.

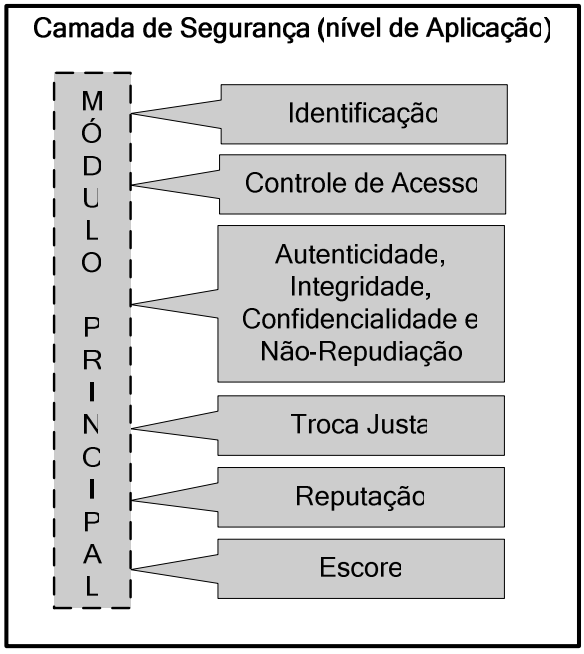


Figura 5-3 Módulos do Modelo Proposto

5.4.1 Identificação de *Peer* e de Objetos P2P

No modelo, a identificação dos *peers* ocorre através da utilização de chaves públicas SPKI/SDSI na camada de aplicação *peer-to-peer* – esta identificação é independente do *peer* ID que é utilizado na rede *overlay* para roteamento. Cada *peer* deve gerar um par de chaves (pública, privada), mantendo a chave privada protegida de algum modo. O *peer* deve divulgar a chave pública que o identifica. A chave pública é utilizada como um identificador único para identificar o *peer* na Internet sem que o mesmo perca o anonimato, pois conhecendo a chave pública não se tem acesso físico ao *peer* pois não há uma relação desse identificador com o endereço IP do *peer*. Para acessar fisicamente o *peer* é necessário o seu *peer ID*, que muda constantemente. Além disto, a chave pública SPKI/SDSI não vincula um *principal* no mundo real a si mesma, e o SDSI/SPKI é independente de tecnologia.

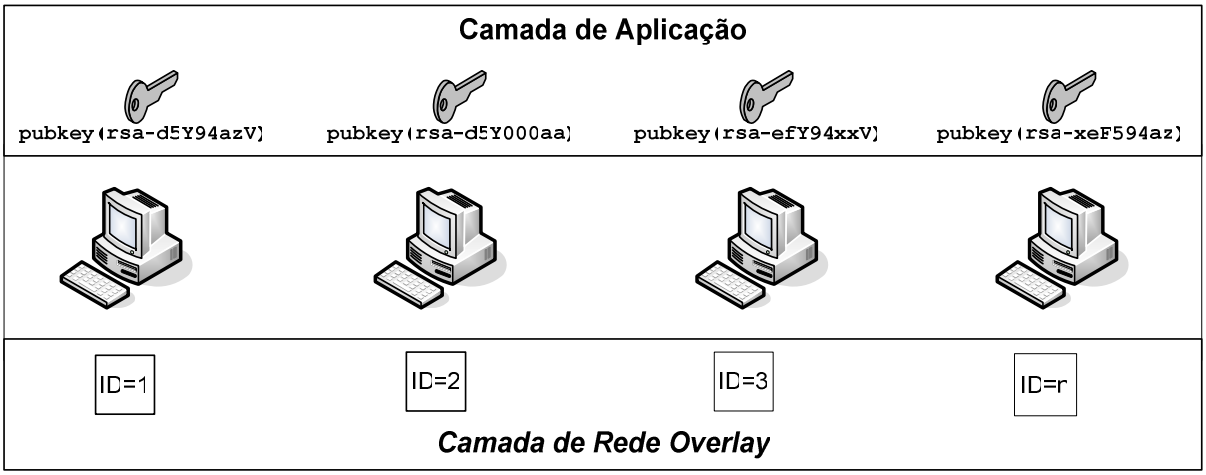


Figura 5-4 Identificação Única em Nível de Aplicação

Se um *peer* deseja conhecer quem é o dono da chave pública que está fazendo uma publicação no repositório compartilhado, deve buscar o certificado de nome associando a essa chave também no repositório. Se o certificado existe no repositório, o *peer* pode identificar o autor da publicação. Caso contrário o anonimato fica preservado, pois se um *peer* não publicar seu certificado de nome pode fazer suas publicações e gerar conteúdos autênticos sem se identificar, pois sem o certificado de nome não será possível identificá-lo como dono da chave.

O conteúdo autoral de um *peer* é identificado pelo formato: AuthorKey@DocumentName (identificação do documento). Ou seja, para manter uma forma de identificação única de conteúdo numa rede P2P adota-se um esquema que associa o nome do arquivo à chave pública do autor que a gerou, produzindo assim um identificador global único.

Quando um *peer* tem uma cópia de um conteúdo publicado por outro *peer*, este irá anunciar uma nova identificação no momento da republicação da seguinte forma: ServerKey@AuthorKey@DocumentName (identificação de documento replicado). Ou seja, a identificação do documento e do seu autor são mantidos, apenas acresce-se a chave do provedor/servidor do conteúdo.

5.4.2 Mecanismo de Autenticidade, Integridade, Confidencialidade e Irretratabilidade (*Non-Repudiation*)

Toda vez que um *peer* faz uma publicação de um conteúdo, deve assinar a publicação (inserida no repositório) e o conteúdo (depositado localmente num diretório compartilhado), como ilustra a Figura 5-4. Assim, o *peer* que consulta a rede *peer-to-peer* tem

como saber se o outro *peer* que está fazendo a publicação é um *Servent* confiável. Logo, obtém-se a autenticidade do conteúdo a partir do histórico da chave que faz a publicação.

As publicações compartilhadas por todos no repositório também podem ser aplicadas para manter a autenticidade cronológica de uma publicação [1], prevenindo um conteúdo já publicado de ser ilegalmente republicado como novo por um *peer* malicioso.

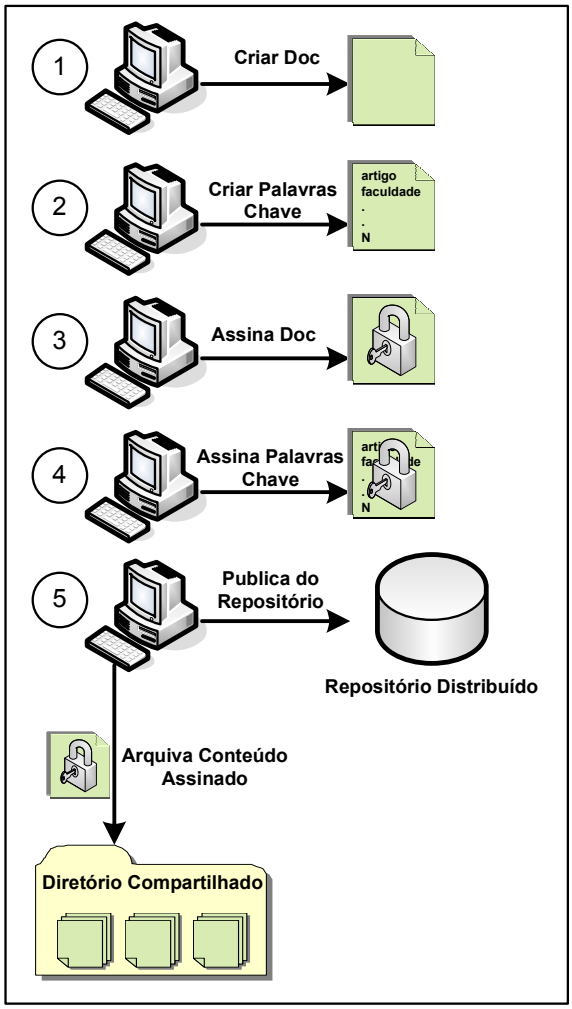


Figura 5-5 Identificação Única em Nível de Aplicação

Como no modelo tudo deve ser assinado (publicações, conteúdos, certificados, pendências, votos, etc.), as chaves dos envolvidos nas trocas de mensagens ou publicações podem ser registradas em um arquivo de registros (log), por exemplo. Este arquivo pode servir para auditoria e para o mecanismo de irretratabilidade, pois publicações assinadas evitam a falsa negação da origem (irretratabilidade da origem) e o *peer* que serve conteúdos registra o *peer* que baixou tal conteúdo através da publicação assinada da pendência de qualificação (irretratabilidade do destino). Se houver a necessidade de arbitramento devido à

irretratabilidade, o anonimato da chave poderá ser quebrado no servidor de onde o conteúdo foi baixado. Neste caso, são respeitados os outros tipos de anonimato relatados em [41].

As assinaturas são feitas em *hash* de conteúdos, logo a integridade é obtida a partir da assinatura digital, pois a assinatura só é conferida se o *hash* estiver correto e a chave que assinou puder ser verificada no destino.

O mecanismo de chaves públicas permite que uma mensagem seja cifrada usando a chave pública do destinatário. Com isto obtém-se a confidencialidade de um conteúdo, que é contemplada no modelo a partir das chaves públicas adotadas do SDSI/SKPI.

5.4.3 Controle de Acesso

O controle de acesso a conteúdos P2P não é muito comum em aplicações convencionais de compartilhamento de mídia (música, filmes, etc.). Porém, em aplicações mais profissionais que usam a infra-estrutura P2P como suporte para ambiente distribuído, e escalável, este se faz necessário. Para implementar o controle de acesso, o modelo utiliza mecanismos criptográficos e o mecanismo de troca justa.

No uso de redes *peer-to-peer* por padrão o acesso de leitura aos conteúdos compartilhados deve ser permitido, também para não inibir o compartilhamento estimulado pelo mecanismo de troca justa [81]. Entretanto, há casos onde se deseja restringir o acesso de alteração dos conteúdos P2P, mantendo a sua autenticidade.

Para controlar a modificação do conteúdo mantendo sua autenticidade e preservando o autor do mesmo, foi aplicado o modelo de autorização do SDSI/SPKI. Assim, assume-se que o conteúdo está sempre disponível para leitura, mas sua alteração somente poderá ser efetivada quando esta operação for explicitamente permitida, através de uma cadeia de certificados de autorização SDSI/SPKI.

Nos casos que o *peer* deseja controlar também o acesso de leitura a um conteúdo P2P, o mesmo deve publicar o conteúdo cifrado usando a chave pública do destinatário. Se for permitida a alteração do conteúdo, o *peer* de origem publica também os certificados de autorização, atribuindo direito de modificação diretamente para a chave do *peer* de destino. Assim, o *peer* de destino poderá alterar o conteúdo P2P, mantendo sua autenticidade se após a alteração publicar também a cadeia que concedeu o direito de alteração. Ou seja, o *peer* que recebeu o direito de alterar o conteúdo, após realizar a modificação do mesmo, quando for republicar o mesmo na rede, assina o conteúdo e publica o conteúdo assinado, junto com a cadeia de certificado de autorização que recebeu indicando que a alteração foi legítima [93].

Em SDSI/SPKI, no campo de identificação do objeto para o qual está se delegando direito de acesso, é utilizada uma URI. Porém, em P2P, uma URI não faz sentido. Como comentado na seção 5.4.1, um objeto será identificado a partir da concatenação do nome do objeto à chave pública do autor do conteúdo (KeyAuthor@DocumentName). Deste modo a substituição da URI será feita pelo identificador de objeto que fornece uma forma persistente de identificação do mesmo na rede P2P. Isto significa que no certificado de autorização SDSI/SPKI, no lugar de uma URI, será colocada a identificação do objeto proposto neste trabalho.

Na camada de segurança há também a implementação do *enforcement* (aplicação) SDSI/SPKI feito através da verificação da cadeia de certificados, para o momento que o *peer* realizar a função de provedor/servidor P2P.

Uma avaliação para verificar se as políticas de troca justa combinam com as informações sobre o *peer* consumidor é feita quando o *peer* tenta acesso a um conteúdo compartilhado. Se não combinarem, o acesso ao conteúdo é negado, mas se o resultado for positivo o acesso é permitido. Para melhor entendimento deste esquema outros elementos da proposta serão introduzidos e este assunto será tratado em maior detalhe na seção 5.4.7.

5.4.4 Reputação

No resultado de uma busca, um *peer* cliente recebe uma lista de possíveis *peers* provedores do conteúdo desejado. O *peer* cliente pode escolher fazer o *download* de conteúdos somente de outros *peer*s cuja chave pública já conheça (através de um histórico positivo de *download* de conteúdo) ou então a escolha pode ser feita através da consulta ao serviço de escores (Seção 5.4.5). Ou seja, em ambos os casos o *peer* cliente tenta se valer de esquemas que lhe confirmem a boa reputação do *peer* provedor de conteúdo. Dessa forma a probabilidade de obter um conteúdo falso de uma publicação autêntica tende a ser muito baixa. Portanto, a credibilidade de uma chave pública é baseada em sua reputação positiva, obtida do bom relacionamento com outros *peers* da rede. A reputação positiva de uma chave é obtida com base na oferta de conteúdo autêntico para a rede e quando o *peer* se comporta de maneira esperada.

A avaliação da autenticidade de um conteúdo somente pode ser feita por um ser humano [1], pois mesmo que a integridade e a origem de um conteúdo possam ser verificados isso não impede que *peers* maliciosos tentem publicar conteúdos maliciosos. Com o passar do tempo, tal comportamento logo será detectado. Um motivo que pode levar um *peer* bem

reputado a produzir conteúdos indesejados é o comprometimento (violação) do mesmo. Ou a comercialização da chave pública bem reputada que passa para o controle de um outro indivíduo.

O esquema de reputação proposto é baseado em qualificação, tanto do provedor/servidor do conteúdo quanto do autor do mesmo, por meio de um mecanismo de votação. Quando um *peer* solicita um conteúdo (documento) para fazer *download*, o *peer* provedor envia ao *peer* cliente um pedido de qualificação, isto é, uma pendência de voto. Tal pedido deve ser assinado com a chave privada do *peer* cliente, como requisito para obtenção do conteúdo. Quando a pendência assinada é devolvida ao provedor, esse a publica no repositório compartilhado e o conteúdo é então fornecido.

Após o *download* do documento, o *peer* consumidor, utilizando interação do usuário do *peer*, avalia o conteúdo e atribui uma "nota" para o conteúdo e seu provedor através do mecanismo de votação, que é mantido no repositório compartilhado. A nota pode ser neutra, positiva ou negativa. No repositório compartilhado, um pedido de qualificação é respondido pelo respectivo votante expressando a nota associada pelo cliente ao autor e ao servidor de conteúdo.

Quando um *peer* cliente atribui uma nota positiva para o provedor/conteúdo, isto significa que o conteúdo será compartilhado. Portanto, o voto positivo indica que o cliente passará a ser também um provedor daquele conteúdo, inserindo seu registro no repositório distribuído (DHT) e tornando-se um *peer* provedor que realiza a replicação do conteúdo baixado. Se a nota for negativa ou neutra a replicação não é feita. Dessa maneira, a replicação de conteúdo ruim tende a minimizar, assim como a proliferação de lixo P2P.

5.4.5 Serviço de Escores

Baseado no esquema de reputação, o serviço de escore (pontuação ou contagem) P2P foi definido conforme ilustrado na Figura 5-6. O serviço, executado em vários períodos determinados (por exemplo, hora em hora, um determinado horário do dia, ou dia da semana) consulta o repositório compartilhado para coletar dados sobre os votos e pendências, gerando relatórios estatísticos baseado em tais informações (número de pendências, votos positivos, negativos e/ou neutros). Com essas informações é possível gerar uma reputação dos *peers* na rede.

Além da geração de estatísticas, o relatório também avalia o possível comportamento de *free-rider* (nós-carona). Por exemplo, um *peer* que emite mais qualificações negativas ou neutras do que positivas, para evitar o compartilhamento de recursos com a rede.

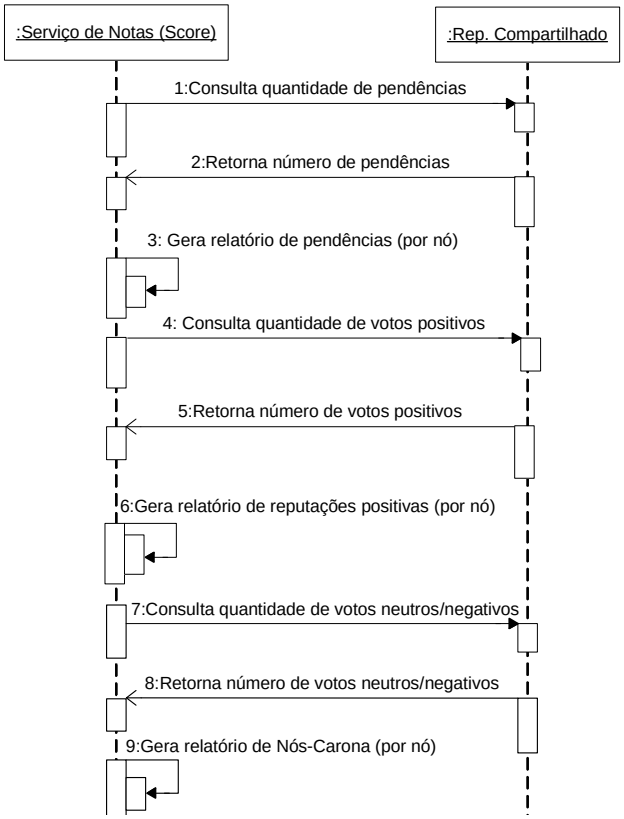


Figura 5-6 Serviço Gerador de Escores P2P

5.4.6 Repositório

O modelo utiliza uma DHT como repositório distribuído para armazenar os dados que são manipulados pelo sistema. Essa abordagem tenta evitar que os *peers* de maneira maliciosa possam alterar tais informações.

Nessa seção, são descritos os tipos de registros que são armazenados no campo "valor" do par (chave, valor). Os registros são inseridos na DHT (repositório). Estes tipos diferentes de registros na DHT são devido aos diversos tipos de informações que são armazenadas na mesma.

A seguir estes tipos de registros serão apresentados com uma breve descrição do que cada campo armazena para os casos de: publicação de conteúdo, publicação de valores de segurança (certificado de nomes e de autorizações), votação, geração de pendências e geração dos escores.

Tabela 5-1. Registro de Conteúdos

Tipo	PP	PA	IDO	AS	H	C
Conteúdo	PrvKp	AutKp	AutKp@doc	AutKp(H(doc))	H(doc)	Boolean

Tipo: Identifica o Conteúdo

PP: Chave Pública do *Peer* Provedor

PA: Chave Pública Autor do Conteúdo

IDO: Identificador do Objeto (Chave Pública do Autor@Nome do Documento)

AS: Assinatura do Documento com a Chave do Autor

H: *Hash* do Documento

C: Indica se o Conteúdo está ou não Cifrado

Tabela 5-2. Registro para Certificado de Nome

Tipo	PP	PA	IDO	AS	H	C
CertNome	PrvKp	AutKp	AutKp@doc	AutKp(H(doc))	H(doc)	Boolean

Tipo: Identifica um Certificado de Nomes

PP: Chave Pública do *Peer* Provedor

PA: Chave Pública Autor do Conteúdo

IDO: Identificador do Objeto (Chave Pública do Autor@Nome do Documento)

AS: Assinatura do Documento com a Chave do Autor

H: *Hash* do Documento

C: Indica se o Conteúdo está ou não Cifrado

Tabela 5-3. Registro para Certificado de Autorização

Tipo	PP	PA	IDO	AS	H
CertAuto	PrvKp	AutKp	AutKp@doc	AutKp(H(doc))	H(doc)

Tipo: Identifica um Certificado de Autorização

PP: Chave Pública do *Peer* Provedor

PA: Chave Pública Autor do Conteúdo

IDO: Identificador do Objeto (Chave Pública do Autor@Nome do Documento)

AS: Assinatura do Documento com a Chave do Autor

H: *Hash* do Documento

Tabela 5-4. Registro para Voto

Tipo	PP	PA	IDO	AS-VF	AS-VP	VP	VC
Voto	PrvKp	AutKp	AutKp@doc	VotKp(H(vf))	VotKp (H(vp))	Voto	Voto

Tipo: Identifica o Voto dado por um consumidor a um Cliente

PP: Chave Pública do *Peer* Provedor

PA: Chave Pública Autor do Conteúdo

IDO: Identificador do Objeto (Chave Pública do Autor@Nome do Documento)

AS-VP: Assinatura do Voto dado ao Provedor

AS-VC: Assinatura do Voto dado ao Conteúdo

VP: Voto dado ao Provedor (Positivo, Neutro, Negativo)

VC: Voto dado ao Conteúdo (Positivo, Neutro, Negativo)

Tabela 5-5. Registro para Pendência

Tipo	PP	PA	DH	AP	HP
Pendência	PrvKp	AutKp	Data/Hora	ConKp (H(pd))	H(pd)

Tipo: Identifica a Pendência (valor gerado pelo fornecedor de conteúdo)

PP: Chave Pública do *Peer* Provedor

PA: Chave Pública Autor do Conteúdo

DH: Data e Hora que foi inserida Pendência

AP: Assinatura da Pendência pelo Consumidor

HP: *Hash* da Pendência

Tabela 5-6. Registro para Escore

Tipo	PP	PA	IDO	AS	H	C
Score	SrvKp	AutKp	AutKp@doc	AutKp(H(doc))	H(doc)	Boolean

Tipo: Identifica um Escore

PP: *Peer* Provedor

PA: Chave Pública Autor do Conteúdo

IDO: Identificador do Objeto

AS: Assinatura do Documento com a Chave do Autor

H: *Hash*

C: Cifrado

5.4.7 Troca Justa

Esse mecanismo tem um papel importante na tomada de decisão de acesso em um *peer* provedor/servidor de conteúdo. Ou seja, o esquema de troca justa permite que o *peer* provedor possa impor regras para permitir o acesso aos seus recursos, tais como, permitir acesso quando um *peer* consumidor se encaixar em um perfil pré-determinado baseando-se em quantidade de votos positivos, negativos e neutros, por exemplo, fornecidos pelo serviço de escore.

O mecanismo de troca justa não depende do mecanismo de controle de acesso, mas o complementa. Assim, a união desses dois esquemas permitirá que um *peer* provedor só forneça conteúdo para *peer*s clientes que possuam escores compatíveis com os requisitos da política de troca justa. Um *peer* servidor pode ou não adotar o esquema de troca justa.

O objetivo do esquema é a estimulação de compartilhamento de conteúdo (documentos) para reduzir o comportamento de *free-rider* (nó-carona) e premiar os *peers* que mais colaboram com a rede. A Figura 5.6 ilustra a dinâmica do esquema de troca justa, que representa os passos 6 a 7 ilustrados na Figura 5-2, na seção 5.4 deste capítulo.

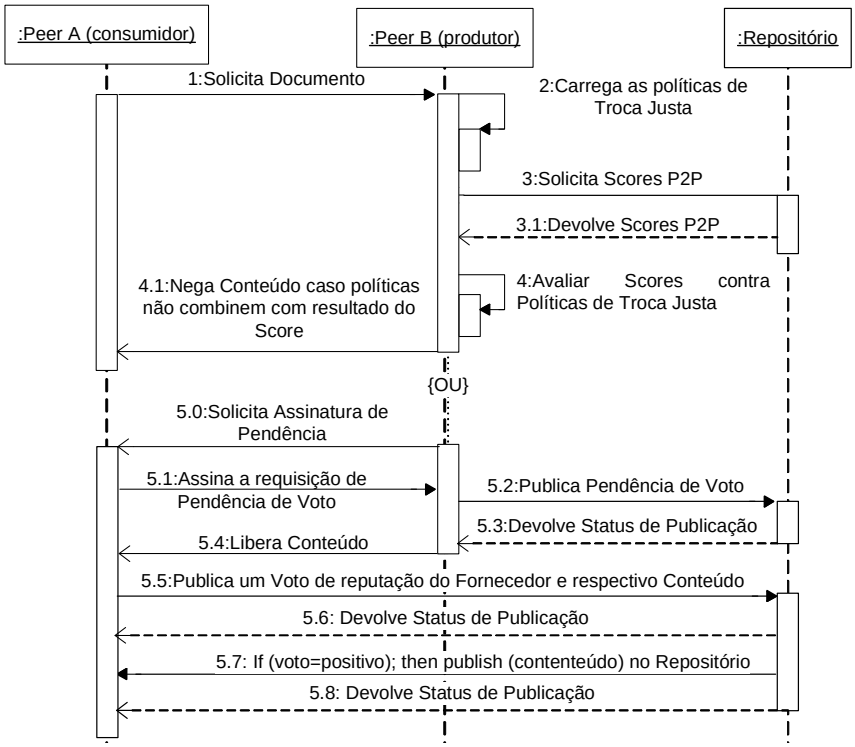


Figura 5-7 Dinâmica do Esquema de Troca Justa

Um *peer* A (cliente) solicita o *download* de um conteúdo a um *peer* B (provedor), (evento 1 da Figura 5-6). O *peer* B, por sua vez carrega suas políticas de troca justa pré-configuradas, a fim de filtrar somente os pedidos que se ajustem às suas regras, por exemplo, somente fornece o conteúdo a *peers* que tenha um número determinado de votos positivos, no máximo x votos negativos (evento 2).

Subsequentemente, o *peer* B solicita os escores armazenados no repositório referentes ao *peer* consumidor (*peer* A), os quais são devolvidos no evento 3. Os escores são então confrontados com as políticas do esquema de troca justa (evento 4). Caso as regras não combinem com o esperado, o acesso ao conteúdo é negado, (evento 4.1). Caso contrário, o *peer* B (provedor) solicita que o *peer* A (consumidor) assine a pendência de qualificação (evento 5).

No evento 5.1 da Figura 5-6, o *peer* A faz a assinatura da pendência com sua chave privada e devolve ao *peer* B, que o publica no repositório (evento 5.2). Uma vez que esse requisito foi atendido, o conteúdo de *peer* B é liberado para *download* ao *peer* A (evento 5.4).

De posse do conteúdo, o *peer* A faz a avaliação do mesmo e atribui o seu voto ao provedor e autor, publicando-os no repositório (evento 5.5). Se o voto é positivo, o *peer* A passa a ser também mais um provedor do conteúdo, fazendo a replicação do mesmo e criando uma nova entrada no repositório (evento 5.7).

5.5 Considerações Finais

Como mencionado anteriormente neste capítulo, a quantidade de conteúdo falso ou corrompido (poluído) cresce exponencialmente e representa mais de 50% do conteúdo disponível na rede P2P. O tráfego de conteúdo poluído diminui a quantidade de largura de banda disponível para um uso apropriado da rede.

No modelo proposto tudo é digitalmente assinado. Assim, com um esquema de chaves públicas confiáveis, é esperado que somente o conteúdo autenticado e assinado seja compartilhado. As chaves públicas que não tem credibilidade (boa reputação) não serão acessadas; portanto, não haverá benefícios no compartilhamento de conteúdo poluído.

Atualmente, o sistema P2P tenta minimizar o comportamento de *free-rider* (nós-carona) requerendo compensação do *peer*. Por exemplo, para que o *peer* obtenha conteúdo da rede P2P, precisa compartilhar quantidades similares de conteúdo em número de bytes. Neste caso, muitos *peers* maliciosos tornam disponíveis grandes arquivos com conteúdo falso para ganhar uma quantidade similar de conteúdo da rede P2P. No modelo proposto, o conteúdo de

um *peer* maldoso não será baixado, porque um *peer* que demonstra este comportamento é detectado pelo mecanismo de escore e não estará bem reputado por nenhum *peer* da rede.

Quanto ao ataque s*ybil,* na proposta, não há beneficio nenhum em um *peer* ficar gerando identidades para votar positivamente em uma de suas identidades falsas para tentar elevar a reputação de alguém, porque o mecanismo de escore detecta a falsa publicação de conteúdo disponível e também detecta a irrelevância dos votos positivos gerados a partir de identidades falsas de *peer*s que são bem reputados. Portanto, o esquema de reputação tende a diminuir o efeito *free-rider* (nó-carona) [19] e desestimular o ataque *sybil* [1].

O mecanismo de troca-justa (baseado em DHT) funciona como mecanismo de compensação confiável, devido à ausência de ponto centralizador na gestão dos escores.

A identificação através de chaves públicas garante a identificação persistente de *peer*/*principal* preservando o anonimato daqueles que não desejam ser identificados, gerando um mecanismo robusto para a identidade, ou seja, a assinatura digital.

O mecanismo de identificação de objetos, independente do caminho (*path*) do servidor, baseando-se na concatenação da chave pública do *peer* com o nome do documento, apresenta-se como uma alternativa interessante de utilização do SDSI/SPKI no ambiente P2P.

A confidencialidade também é alcançada com esta proposta, porque o conteúdo pode ser cifrado na chave pública do (*peer*/*principal*) destinatário e só o próprio poderá decifrar o conteúdo. Além da confidencialidade, a integridade é obtida pelo uso de mecanismos de *hash* na assinatura digital.

A adoção do mecanismo de irretratabilidade evita que um autor negue falsamente sua participação em uma transação com um outro *peer*.

É importante notar que o índice compartilhado não é um índice baseado em servidor. É um repositório distribuído e tolerante a faltas (uma DHT), não figurando como um possível ponto central de falhas ou vulnerabilidades.

O esquema de credibilidade baseado em escores produz níveis de reputação. Se um *peer* tem uma reputação positiva, isto endossa sua chave pública. Tal esquema pode ser comparado à credibilidade dada, quando um certificado é assinado por uma CA em PKI X.509, por exemplo.

Capítulo 6

6 Aspectos de Implementação

6.1 Introdução

Este capítulo apresentada os detalhes de implementação da proposta, bem como os resultados obtidos com a avaliação do protótipo.

6.2 Arquitetura do Protótipo

O protótipo proposto é composto por várias camadas que juntas implementam o modelo apresentado no capítulo 5. Em cada camada, como ilustra a Figura 6-1, foi empregada uma tecnologia que pudesse atender aos requisitos do modelo.

Inicialmente, na camada de aplicação existem dois componentes importantes: o primeiro é uma aplicação P2P (um cliente DHT) construído a partir da tecnologia Java; o segundo componente permite que um conteúdo seja disponibilizado usando a Web a partir dos recursos do Jetty [94].

Para o repositório foi utilizada uma implementação de *Distributed Hash Table* (DHT), o Bamboo [72], baseada no Tapestry [44]. O Bamboo foi escrito usando um estilo de programação *event-driven* (dirigido a evento) e *single-thread*. Muitos programadores possuem familiaridade com este estilo de programação através da experiência com bibliotecas de interface gráfica (GUI – *Graphical User Interface*), tais com o *Java Swing*.

A programação dirigida a eventos facilita muito a programação usando *threads*, principalmente, no que diz respeito à sincronização. Existem basicamente duas abordagens para se estruturar um programa dirigido a eventos: o SEDA (*Staged, Event-Driven Architecture*) e o SFS (*Self-Certifying File System*) [95]. No SEDA, as unidades funcionais de

um programa são divididas em estágios separados (*stages*), os quais se comunicam com objetos Java chamados eventos (*events*).

O SFS é um sistema de arquivos distribuído que foi inicialmente desenvolvido por David Mazieres [96]. O SFS se difere do SEDA basicamente em dois pontos. Primeiro, ao invés de estágios e eventos, o SFS é organizado em *functions* e *callbacks*. E em segundo lugar, é projetado para *single-threaded*.

Devido ao fato de possuir uma documentação mais acessível e ser mais flexível no uso com o Bamboo, foi adotado o SEDA para o desenvolvimento do protótipo.

A camada de segurança contempla vários mecanismos para atender os objetivos do trabalho. O SDSI/SPKI foi utilizado para servir como infra-estrutura de segurança para prover identificação e controle de acesso baseado nos certificados de nome e autorização. O uso de chaves traz outras implicações diretas: autenticidade, irretratabilidade, confidencialidade e controle de acesso fino.

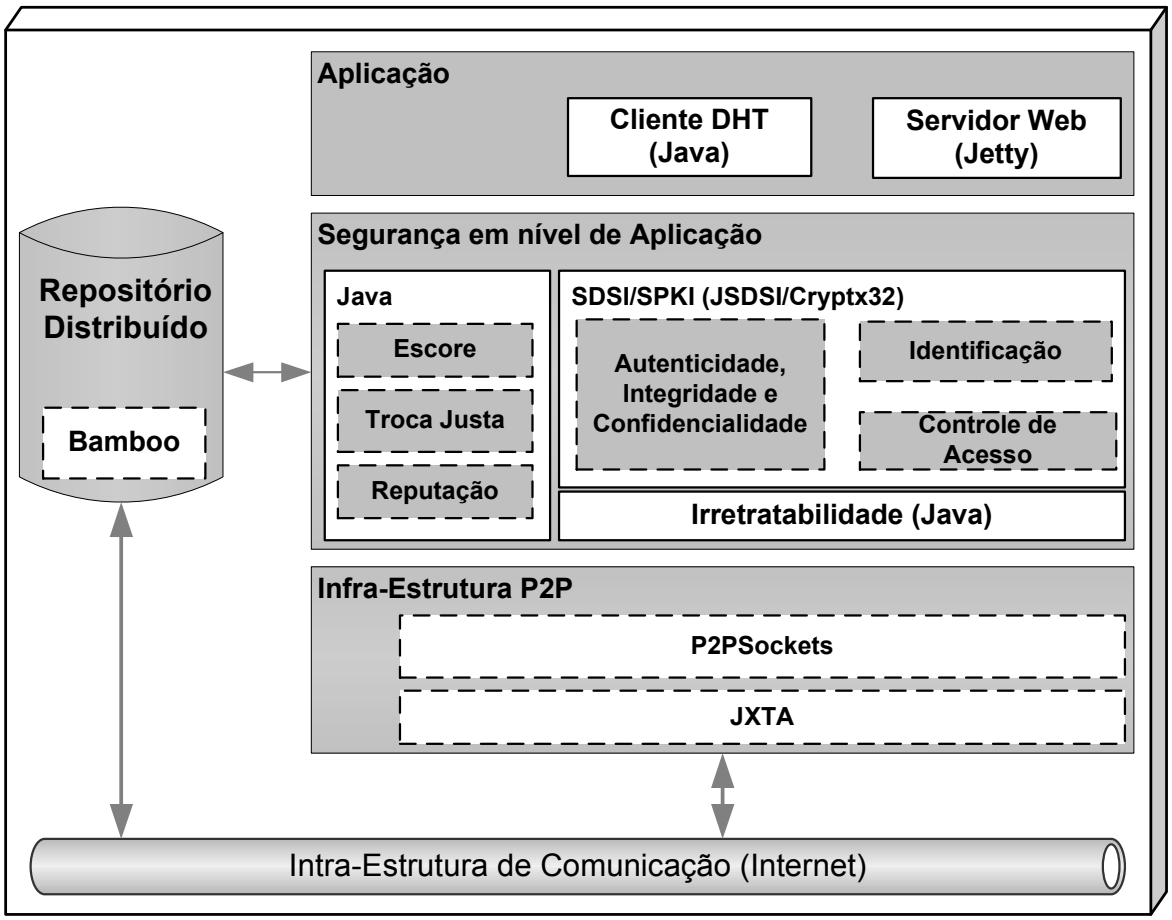


Figura 6-1 Arquitetura do Protótipo

A irretratabilidade está fortemente ligada à assinatura digital, visto que uma mensagem assinada garante a origem da mesma e uma mensagem cifrada na chave do destinatário evita a negação de recebimento no destino.

Além da adoção de chaves, outro mecanismo importante na proposta é o esquema de reputação baseado em votação, que dá suporte ao mecanismo de escore (gerador de relatório utilizado pelo mecanismo de troca justa e para detecção de *peer* com tendência a comportamentos maliciosos). O mecanismo de troca justa, que complementa o controle de acesso, permite que os *peers* servidores possuam políticas que definem “perfis” de *peers* que poderão baixar conteúdos do servidor. Todos estes módulos foram implementados na linguagem Java.

Foi utilizada a infra-estrutura P2P do JXTA para alcançar a interdependência de plataformas e ambientes de rede, assim como para efetivar o transporte de objetos P2P. O *p2psockets* oferece um API que adapta a implementação clássica de *socket* Java para fazer uso da infra-estrutura JXTA.

A seguir, serão apresentados detalhes de cada camada da arquitetura da proposta e um protótipo para um caso de uso.

6.3 Camada de Aplicação

Nesta camada, o funcionamento é feito da seguinte forma: todas as consultas e inserções feitas no repositório utilizam-se do cliente DHT.

Todas as mensagens enviadas são definidas no formato XML, tal como no exemplo ilustrado na Figura 6-2:

```
<?xml version=”1.0” enconding=”ISSO-8859-1”?>
<conteudo>
  <pp>Provedor</pp>
  <pa>KpAutor</pa>
  <ido>KpAutor@doc</ido>
  <as>KpAutor(H(doc))</as>
  <h>H(doc)</h>
  <c>boolean</c>
</conteudo>
```

Figura 6-2 Fragmento de Documento DHT

Na figura 6-3 é mostrado um fragmento de código, que mostra como é feita a criação das estruturas XML no sistema.

```
public void CriarEstrutura(String rootName, String[] listaElementos, String destino){
              Element root = new Element(rootName);
              System.out.println("Gerando root XML : " + rootName);

              for(int i =0; i < listaElementos.length;i++){
                     Element elemento = new Element(listaElementos[i]);
                     root.addContent("\n");
                     root.addContent(elemento);
                     System.out.println("Gerando Elemento XML : " + listaElementos[i]);
              }
              root.addContent("\n");
              Document doc = new Document(root);
              Gravar(doc, destino);
       }
       public  void Gravar(Document doc, String file){
              XMLOutputter outputter = new XMLOutputter();
              FileOutputStream out;
              PrintStream p = null;
              try {
                     out = new FileOutputStream (file);
                     p   = new  PrintStream (out);
                     try {
                           outputter.output(doc, p);
                         }
                         catch (IOException e) {
                           System.err.println(e);
                         }
                     p.close();
              } catch (FileNotFoundException e1) {
                     e1.printStackTrace();
              }
       }
```

Figura 6-3 Fragmento de código de criação das estruturas XML

Os conteúdos compartilhados com a rede P2P são oferecidos pelo servidor Jetty e transportados sobre o protocolo HTTP. Foi implementado o mecanismo de troca justa da proposta onde para que os próprios *peer* façam as regras de quem pode ou não baixar seus conteúdos.

6.4 Camada de Segurança

Nessa camada, estão implementados os mecanismos de identificação e controle de acesso que dão suporte à autenticidade, confidencialidade, integridade e irretratabilidade dos conteúdos. Também estão contemplados os mecanismos de escore e troca justa, baseados no esquema de reputação.

6.4.1 Módulo de Identificação

Foi utilizada neste projeto a biblioteca JSDSI [97], implementação das funcionalidades do SPKI/SDSI (desenvolvidas em Java). Por ser de código livre, foi empacotada e distribuída como **SDSI**. Em 2002, Sameer Ajmani reimplementou está API com várias melhorias e mudou seu nome para **JSDSI**. As melhorias feitas incluem suporte completo as *tags* SPKI, mais eficiência no *parsing* e *marshalling* de objetos SPKI/SDSI, além de melhoria na busca de certificados [98].

Foi utilizada a biblioteca JSDSI 2.0 que contém classes que provêem meios de converter e criar *S-expressions* e implementa os objetos fundamentais do SPKI/SDSI, como certificados e chaves.

Ainda no que se refere ao uso de criptografia, a biblioteca JSDSI 2.0 usa os recursos de outra biblioteca desenvolvida no MIT, a **Cryptix**. Nesse trabalho, utilizou-se uma versão implementada em Java pela *Sun Microsystems*. A biblioteca **Cryptix** oferece as seguintes funcionalidades:

Cifradores: Blowfish, CAST5, DES, IDEA, MARS, RC2, RC4, RC6, Rijndael, Serpent, SKIPJACK, Square, TripleDES, Twofish;

Acordo de Chaves: Diffie-Hellman;

Modos: CBC, CFB-(8, 16, 24, ..., tamanho de bloco), ECB, OFB-(tamanho de bloco), openpgpCFB;

Hashes: MD2, MD4, MD5, RIPEMD-128, RIPEMD-160, SHA-0, SHA-1, SHA-256/384/512, Tiger;

MACs: HMAC-MD2, HMAC-MD4, HMAC-MD5, HMAC-RIPEMD-128, HMAC-RIPEMD-160, HMAC-SHA-0, HMAC-SHA-1, HMAC-Tiger;

Assinaturas: RawDSA, RSASSA-PKCS1, RSASSA-PSS;

Cifradores Assimétricos: RSA/PKCS#1, ElGamal/PKCS#1.

Em termos de linguagem de desenvolvimento da camada de segurança, foi adotada a linguagem Java da *Sun Microsystems.* Os motivos foram: a portabilidade da linguagem, a compatibilidade com as bibliotecas acima citadas, além da infra-estrutura JXTA, que está também implementada nesta linguagem, bem como o navegador *Web2Peer*.

6.4.1.1 Gerando a Identificação dos *Principals*/*Peers*

Este módulo é responsável pela criação do par de chaves (Pública/Privada) para cada *peer/principal*.

Durante o processo de geração das chaves são criados objetos definidos pela linguagem *S-expression* (linguagem baseada em Lisp) proposta por [65] que serve para descrever os campos dos certificados SDSI/SPKI. Na Figura 6-4 e Figura 6-5 é ilustrado um fragmento de código *S-expression* que descreve respectivamente a chave pública e privada de um *principal*.

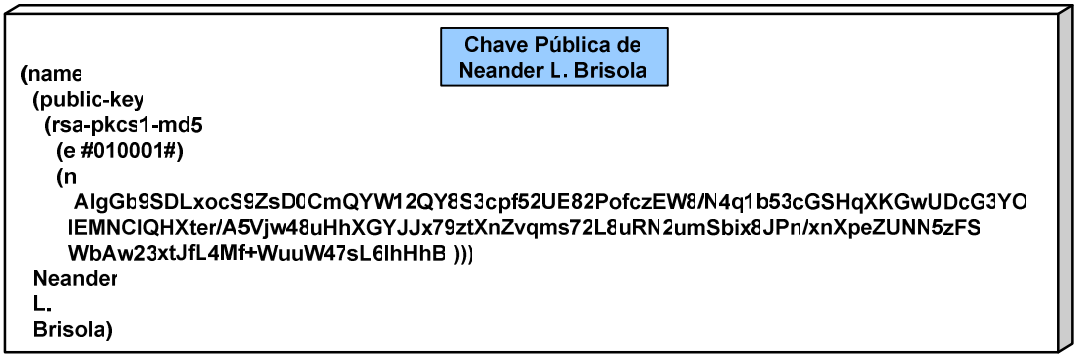


Figura 6-4 Fragmento de código *S-expression* para descrever Chave Pública

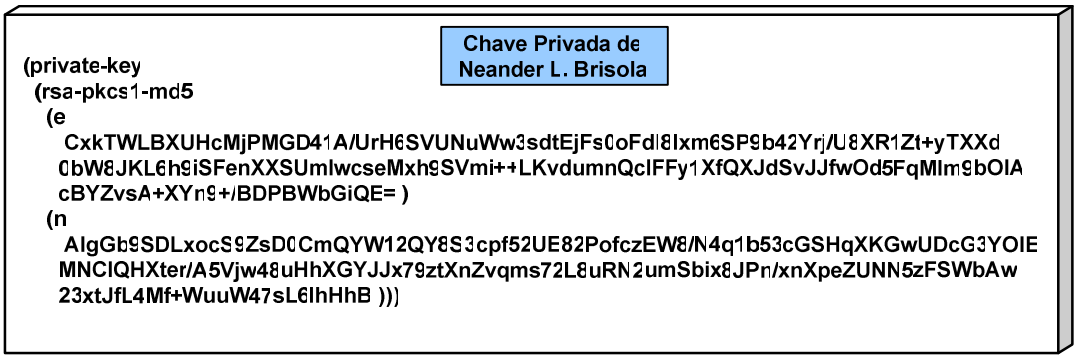


Figura 6-5 Fragmento de código *S-expression* para descrever Chave Privada

Na Figura 6-6 é mostrado um trecho de código exemplificando a criação das chaves a partir da API Java.

```
public void KeyGen(String NameSpace){

    KeyPairGenerator kpg = null;
    try {
        kpg = KeyPairGenerator.getInstance("RSA");
        kpg.initialize(1024,new SecureRandom());
    } catch (NoSuchAlgorithmException e) {
        e.printStackTrace();
    }
    KeyPair kp = kpg.generateKeyPair();
    sdsiPrivKey = new SDSIRSAPrivateKey((RSAPrivateKey)kp.getPrivate());
    sdsiPubKey = new SDSIRSAPublicKey((RSAPublicKey)kp.getPublic());
    this.SaveKeys(sdsiPubKey, sdsiPrivKey);
    this.SaveNSpace(NameSpace, sdsiPubKey);
}

public SDSIRSAPublicKey getMyPubKey(){
    return(sdsiPubKey);
}

public SDSIRSAPrivateKey getMyPrivKey(){
    return(sdsiPrivKey);
}
```

Figura 6-6 Fragmento de Código usado na Criação do Par de Chaves

Esta funcionalidade permite criar o certificado de nome do usuário de um *peer*, permitindo vincular o nome que se deseja publicar a sua chave pública correspondente. A publicação do certificado é feita no repositório. Este módulo envolve a utilização da biblioteca **JSDSI**.

Na Figura 6-7, pode ser visto como foi montada a interface de inserção de dados para a criação do certificado de nome. Para facilitar o processo de criação, as chaves assimétricas e o certificado de nome usam a mesma interface.

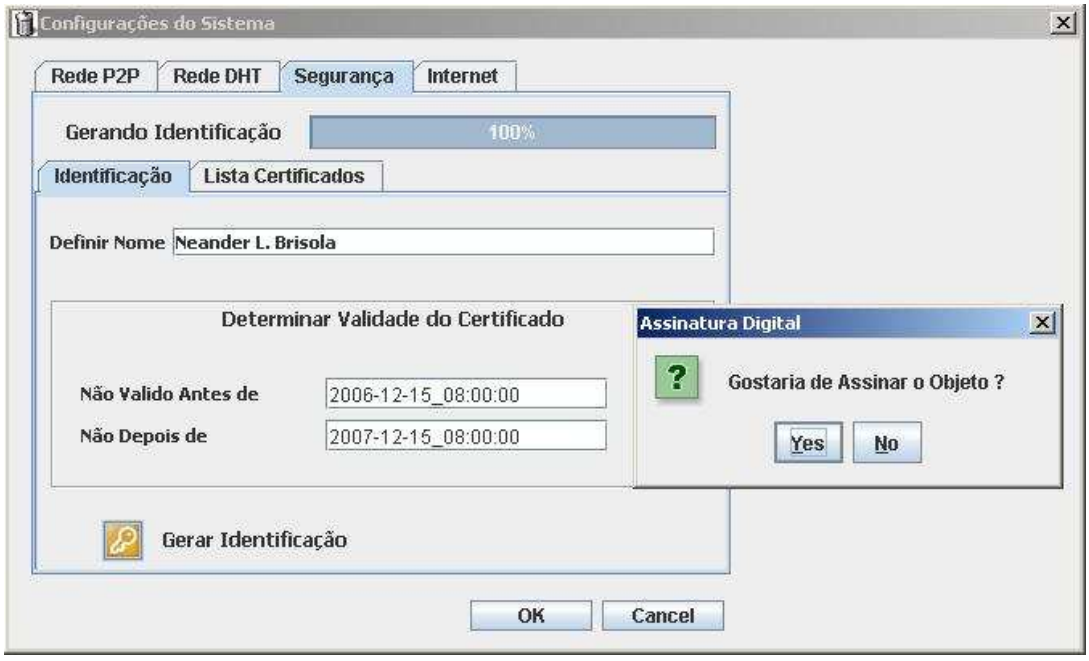


Figura 6-7 Interface para criação do par de Chaves/Certificado de Nome

O processo de geração da descrição *S-expression* do Certificado de Nome e da Assinatura pode ser visto a seguir na Figura 6-8.

```
(cert
  (issuer
    (name
      (public-key
        (rsa-pkcs1-md5
          (e #010001#)
          (n
            |AKJ29vsxRlyiejTZONdXBPiJDVa1GJyyJBtbhoKMvydcKtIf4UfVzc9mP2S
            rZCLIY7bUXYl3ucFl/eH/WvZ9WIMbOQH9yIjmwxzLKe/nCHHWDghJvQpP9cj
            wiBxeRodO5IF/6wWy+ECMwRokE/qvf4Lv3RmNgZagVDrQtTXnhhv5|)))
      "Neander L. Brisola"))
  (subject
    (public-key
      (rsa-pkcs1-md5
        (e #010001#)
        (n
          |AKJ29vsxRlyiejTZONdXBPiJDVa1GJyyJBtbhoKMvydcKtIf4UfVzc9mP2SrZ
          CLIY7bUXYl3ucFl/eH/WvZ9WIMbOQH9yIjmwxzLKe/nCHHWDghJvQpP9cjwiBx
          eRodO5IF/6wWy+ECMwRokE/qvf4Lv3RmNgZagVDrQtTXnhhv5|)))))
  (not-before "2006-12-15_08:00:00")
  (not-after "2007-12-15_08:00:00"))

(signature
  (hash md5 |e60pFdZtz03X/FKDNSMtyA==|)
  (public-key
    (rsa-pkcs1-md5
      (e #010001#)
      (n
        |AKJ29vsxRlyiejTZONdXBPiJDVa1GJyyJBtbhoKMvydcKtIf4UfVzc9mP2SrZCL
        IY7bUXYl3ucFl/eH/WvZ9WIMbOQH9yIjmwxzLKe/nCHHWDghJvQpP9cjwiBxeRod
        O5IF/6wWy+ECMwRokE/qvf4Lv3RmNgZagVDrQtTXnhhv5|)))
  (MD5withRSA
    |eD1e8rF9iyzoXj0ZqUSWJQdGmM2x7WOkzMTg5wXC1YtwhoGUEyuj93CS4RlaERZKSLH
    UQ7fGe7QyChX+ect8N6s4OeQLDWbrzGEskShmxXeK1J1XmbA3TxalhINj/04QK7ra9lj
    /GlS2StJFbVzGk+A+q1Zg/fQ3WVOfRWSYu5w=|))
```

Figura 6-8 Certificado de Nome e sua Assinatura (em *S-expression*)

6.4.2 Mecanismo de Controle de Acesso

Como comentado anteriormente, o controle de acesso foi implementado como controle de acesso à modificação de conteúdos P2P, pois por padrão é assumido que a leitura dos conteúdos é sempre permitida. Quando o autor do conteúdo não deseja que o conteúdo seja acessado por todos, cifra o mesmo na chave pública do *principal* ao qual disponibilizará tal conteúdo.

O módulo de controle de acesso realiza o processo de *enforcement* (confronto das políticas com a cadeia de certificados para decidir se o acesso de alteração ao conteúdo é permitido). Se a cadeia dá direitos de alteração do conteúdo ao *principal* que está desejando fazer a mudança, este precisa publicar a cadeia posteriormente à alteração para manter o conteúdo autêntico.

O módulo verificador de integridade e verificador de autenticidade que avaliam a assinatura de um *peer* antes de fornecer um conteúdo também estão acessíveis a partir deste mecanismo. Além dos controles criptográficos e de acesso presentes neste mecanismo, há também o mecanismo de troca justa.

O mecanismo de troca justa está baseado nos escores provenientes da reputação. Ou seja, o servidor de conteúdo previamente verifica se o cliente P2P possui bom relacionamento com a rede, confrontando os escores com as suas políticas locais. Caso afirmativo, o conteúdo é fornecido, do contrário é negado.

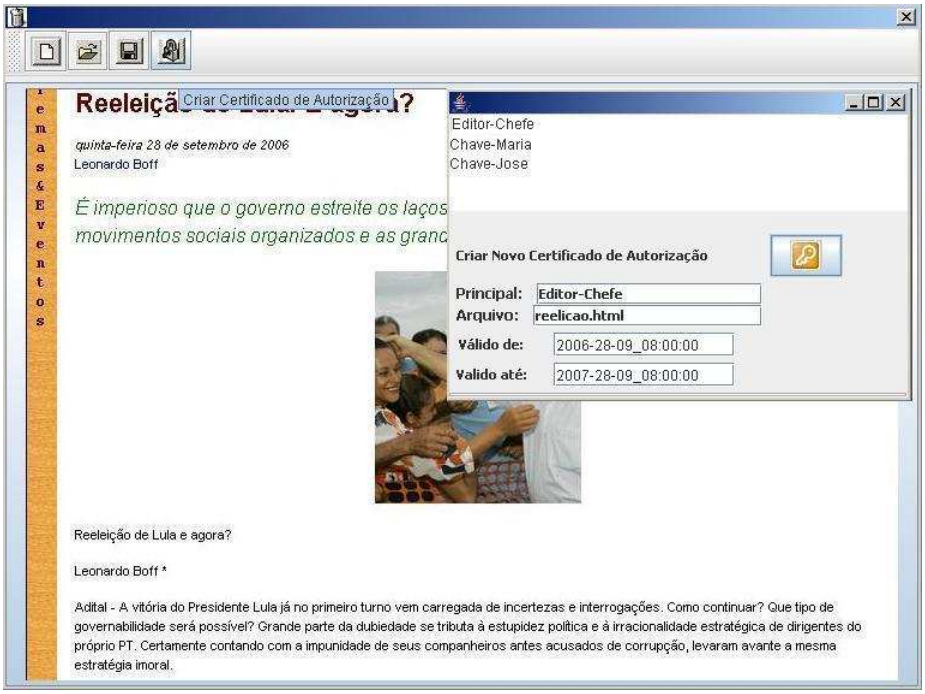


Figura 6-9 Aplicação de um Certificado de Autorização

Na Figura 6-10 é mostrado um certificado de autorização gerado a partir da tela da Figura 6-9, emitido em favor de um *principal* (editor-chefe) – na prática a delegação é feita para a chave pública vinculada ao nome local editor-chefe, concedendo-lhe direito de gravação no arquivo (reeleiçao.html).

```
(cert
  (issuer
    (public-key
      (rsa-pkcs1-md5
       (e #010001#)
       (n
         |AMhMF5tkowUb4S8+RwNKuiX+vtoCeW1aGLohzRDPFZUx4SAJ3UjRs0HaK4cgph
          8F+UBcYP70en1qKDRh+55I/mQNQIH5oBah/aa4UTQze6N7eG1ddjFfznFNWdete
          IybyeggjkmVkRetzRl9xlW8v/Jwwl64lSrs5s9a4QYCoUZv|))))
  (subject
    (name
      (public-key
       (rsa-pkcs1-md5
         (e #010001#)
         (n
          |AKJ29vsxRlyiejTZONdXBPiJDVa1GJyyJBtbhoKMvydcKtIf4UfVzc9mP2S
           rZCLIY7bUXYl3ucF|/eH/WvZ9WIMbOQH9yIjmwxzLKe/nCHHWDghJvQpP9cj
           wiBxeRodO5IF/6wWy+ECMwRokE/qvf4Lv3RmNgZagVDrQtTXnhhv5|)))
      Editor-Chefe))
  (propagate)
  (tag rw-reeleicao.html))
```

Figura 6-10 Resultado *S-expression* da criação do Cert. de Autorização

No exemplo acima, foi emitido um certificado de autorização ao **Editor-Chefe** para que o mesmo possa gravar no arquivo **reeleicao.html**. Usando a cadeia de certificados, é possível saber se o Editor-Chefe pode ou não alterar o arquivo, checando a cadeia. Caso não tenha permissão, o editor dentro do navegador não permitirá a edição do arquivo.

6.4.3 Módulo de Autenticidade, Integridade e Confidencialidade

Este módulo utiliza-se dos recursos criptográficos de funções *hash* para fazer/verificar a integridade de arquivos, assinatura digital para fazer/verificar autenticidade e cifragem/decifragem para prover/obter confidencialidade.

6.4.3.1 Autenticidade

Todas as transações executadas na arquitetura proposta são assinadas e precisam ter sua assinatura verificada pelo interessado na autenticidade dos objetos em questão (documento, publicação, certificado, etc.).

Na figura 6-11, é mostrado um fragmento de código que é utilizado no protótipo para verificar a assinatura digital de um conteúdo (documento).

```
public void verifySignatureFile () throws SignatureException, IOException, SDSIException,
    Signature sign = Signature.getInstance("MD5withRSA");
    SDSIPrivateKey oPrivKey = null;
    SDSIPrivateKey priv = (SDSIPrivateKey)oPrivKey;
    sign.initSign(priv);
    File file = new File(FilePath+FileName);
    FileInputStream fis = new FileInputStream(file);
    while (fis.available() != 0) {
        sign.update((byte) fis.read());
    }
    SDSISignature signature;
    try {
        // read public key from file
        fis = new FileInputStream(FilePath+FileName);
        signature = (SDSISignature)SDSIObject.readFrom(fis);
        fis.close();
        ) catch(IOException e){
            System.out.println("Erro Lendo Arquivo");
        } catch (SDSIException e) {
            e.printStackTrace();
        }

        if (!(signature instanceof SDSISignature)) {
            System.out.println("O arquivo precisa ter uma assinatura válida");
        }
    if (sign.verify(signature)) {
        System.out.println("A Assinatura Confere");
    } else {
        System.out.println("A Assinatura não Confere");
    }
```

Figura 6-11 Fragmento de Código para verificação de Assinaturas Digitais

6.4.3.2 Assinar Conteúdos

A Figura 6-11 mostra a interface utilizada para proceder a uma assinatura digital, no caso, de um conteúdo.

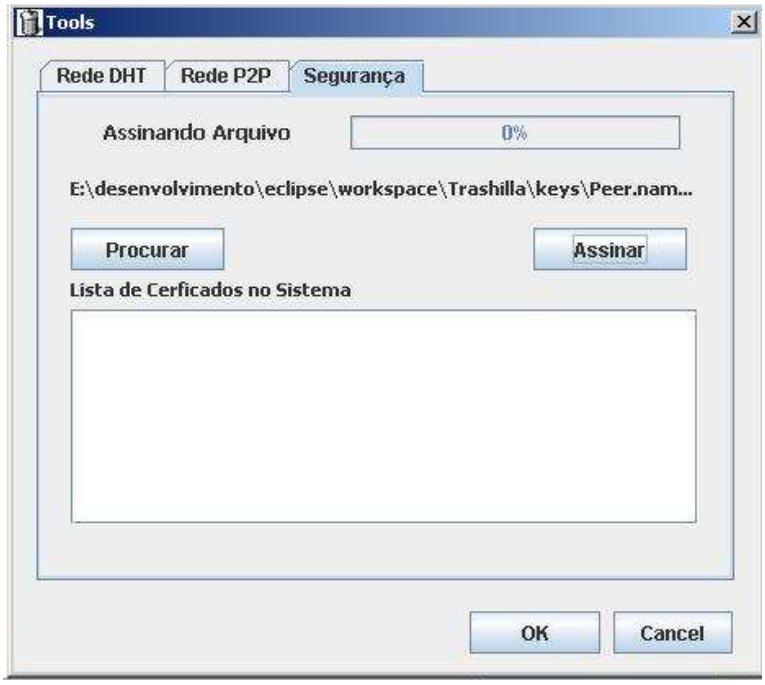


Figura 6-12 Interface para Assinatura de Conteúdo

Na seqüência, é apresentado um fragmento de código na Figura 6-12 que mostra a implementação do mecanismo da assinatura no protótipo.

```
public void SaveSignature (SDSIObject signature, String fName){
    System.out.println("Salvando a Assinatura do arquivo : " + fName);
    try {

        FileOutputStream fos = new FileOutputStream(fName);
        signature.writeReadable(fos);
        fos.close();

    } catch (IOException e) {
        System.out.println("Erro ao Criar chave de Assinatura do Arquivo");
        return;
    }
}

public void signObject(SDSIObject obj, SDSIPublicKey kpub, SDSIPrivateKey kpriv){
    try {

      SDSISignature newSig = new SDSISignature(obj, kpriv, kpub);
      System.out.println("Objeto Assinado\n");
      this.SaveSignature(newSig, "nCertSign");

    } catch (SignatureException e) {
      e.printStackTrace();
      System.out.println("Impossível Assinar!");
    }
  }
```

Figura 6-13 Fragmento de Código para Assinatura Digital de Arquivos

6.4.3.3 Integridade

A integridade é obtida através da implementação de função de *hash* (resumo/sumarização) sobre o mesmo. Na Figura 6-13 é ilustrado um fragmento de código no protótipo que calcula o *hash* com base na função MD5.

```
Hash fileHash(String sFile)
  {
    try {
          File dFile = new File(sFile);
          if (!dFile.exists())
            throw new FileNotFoundException("File not Found");

          byte[] data = new byte[(int)dFile.length()];
          FileInputStream input = new FileInputStream(dFile);
          input.read(data);
          MessageDigest digest = MessageDigest.getInstance ("md5");
          return new Hash("md5", digest.digest(data));
        } catch (IOException e) {
          System.out.println("Erro1");
        } catch (NoSuchAlgorithmException e) {
            System.out.println("Erro2");
        } catch (SexpParseException e) {
            System.out.println("Erro3");
        }
        return null;
  }
```

Figura 6-14 Fragmento de Código para gerar o Hash do Arquivo

6.4.4 Mecanismo de Reputação

No módulo de reputação, o objetivo principal é catalogar informações que permitam reputar *peer*s e seus conteúdos. Uma vez registrado isso no repositório, o mecanismo de escore pode fazer o levantamento de quais *peer*s que estão atuando na rede são bem ou mal comportados.

Na Figura 6-15, é ilustrada uma busca e o resultado obtido, trazendo os valores dos votos e a reputação das respectivas fontes (servidores P2P).

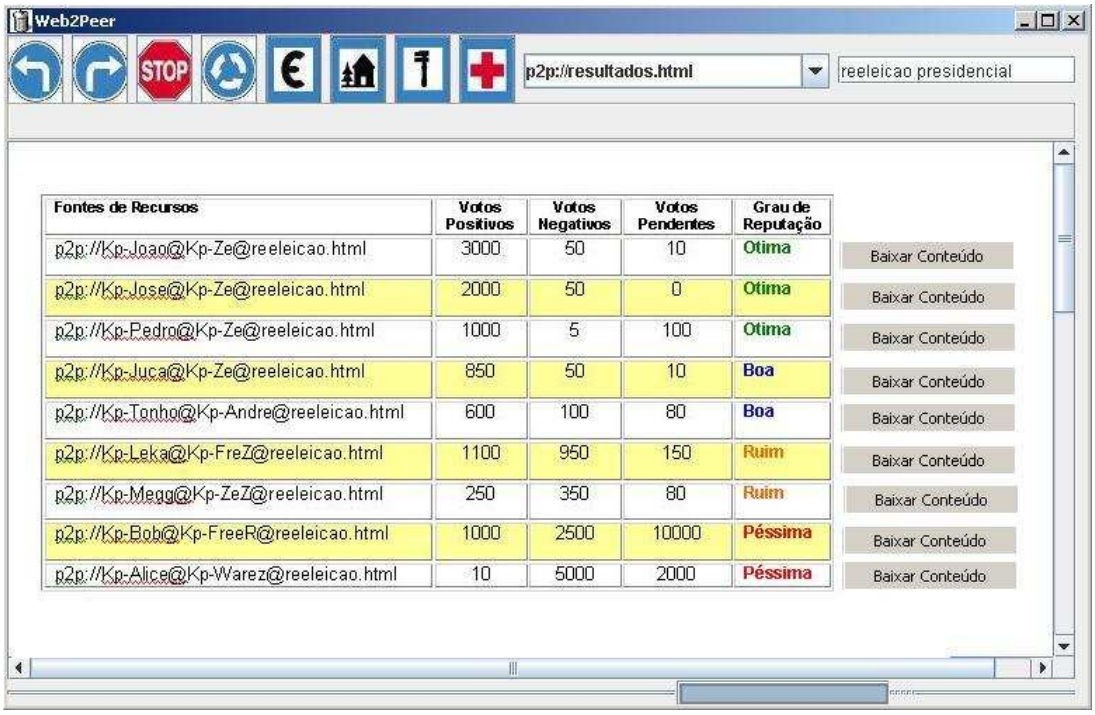


Figura 6-15 Lista de Provedores e suas Reputações

Uma vez que é escolhido um conteúdo para baixar, no momento que se escolhe qual a fonte (provedor/servidor), imediatamente gera-se uma pendência de qualificação para o conteúdo solicitado, para que o peer cliente assine e a mesma é publicada na DHT pelo servidor P2P.

Após baixar o conteúdo e avaliá-lo o *peer* cliente precisa qualificar o o servidor e o conteúdo. Na Figura 6-16 é possível visualizar alguns exemplos de pendências a serem votadas na interface, onde um *peer* consumidor expressa sua opinião tanto para o *peer* fornecedor quanto para o seu conteúdo que faz solicitação do voto ao *peer* cliente.

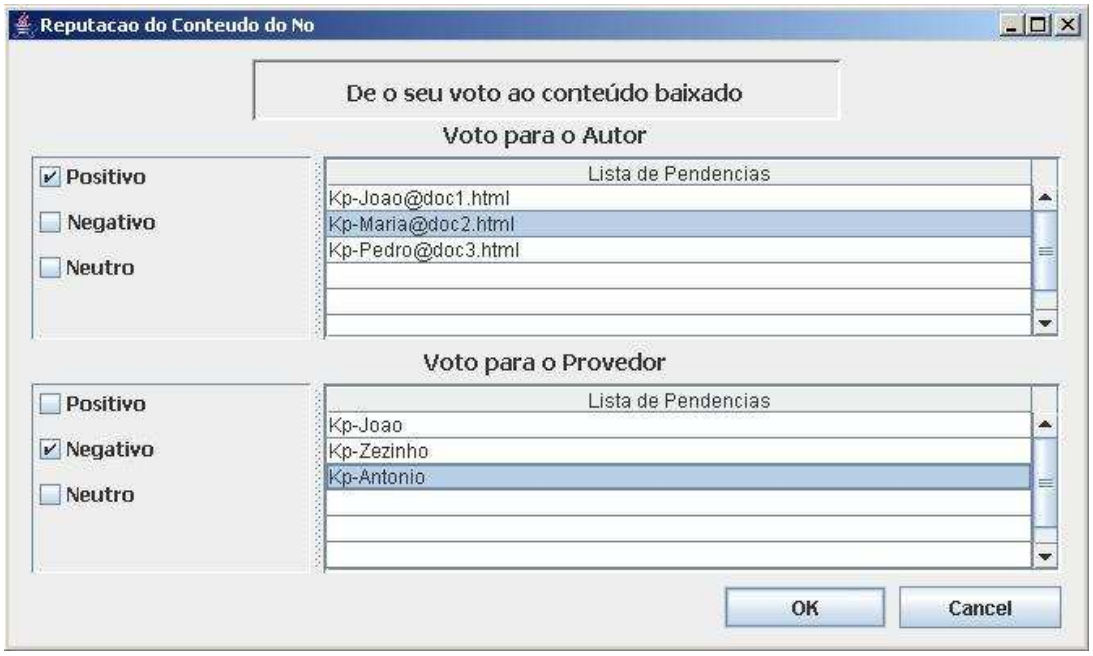


Figura 6-16 Mecanismo de Votação

Os votos inseridos no repositório são os dados para o mecanismo de escore gerar novas estatísticas utilizadas para compor a reputação de um peer.

6.4.5 Mecanismo de Escore

Esse módulo da camada de segurança é responsável por habilitar o serviço (status) de escore no *peer*. Além disso, possui uma lista que é atualizada periodicamente, formando assim um ranking dos *peers* da rede, conforme seus relacionamentos nas trocas de conteúdo. Isto é baseado no mecanismo de reputação, que é outro esquema implementado na camada de segurança.

Como todos os dados das qualificações e pendências estão no repositório, foi implementado o módulo do escore dentro de alguns *peers* que fazem parte da DHT. Esse módulo é acionado no momento da inicialização do Bamboo e é feito por meio dos arquivos de configuração do SEDA. Um fragmento do arquivo pode ser visto na Figura 6-17, a seguir.

```
<sandstorm>
<global>
  <initargs>
   node id locahost: 3200
  </initargs>
</global>
<stages>
  <Network>
    class bamboo.network.Network
    <initargs>
    </initargs>
  </Network>
  <Router>
     class bamboo.router.Router
       <initargs>
       gateway count 1
       gateway 0 localhost: 3200
       </initargs>
  </Router>

       <Escore>
              class bamboo.Escore
              <initargs>
              debug delevell
              mode sender
              </initargs>
      </Escore>
       </stages>
</sandstorm>
```

Figura 6-17 Configuração do Escore no Formato SEDA

Depois de carregado, o mecanismo de escores varre periodicamente a DHT buscando qualificações e pendências para contabilizá-las para cada peer (chave pública).

Além da reputação o mecanismo de escores também faz a relação entre votos positivos, negativos e pendências, para saber se um nó está comportando-se como *free-rider* (nó carona), por exemplo. A estratégia utilizada é um voto positivo anular um negativo ou uma pendência, isto é, se um *peer* possui 10 votos negativos e 7 pendências, mas possui 20 votos positivos, terá um saldo de 3 votos positivos. Quando os valores de votos negativos e pendências de um *peer* forem maiores que 50% em relação aos seus votos positivos, o *peer* automaticamente será indicado pelo mecanismo de escore como um possível nó carona, pois é visto pela rede negativamente. Porém, quem efetivamente define quais valores representam um nó carona são as políticas de troca justa escritas em cada peer provedor/servidor P2P.

Estes critérios podem ser usados de duas maneiras, sendo que na primeira, o *peer* (cliente) verifica qual *peer* (provedor/servidor) possui uma reputação melhor antes de baixar qualquer conteúdo. Já o *peer* que estiver atuando como servidor poderá fazer um controle de troca justa como mencionado anteriormente, definindo critérios que permitirão ou não que um *peer* (cliente) baixe seus conteúdos. Um exemplo seria o *peer* servidor somente permitir que

baixe seus recursos os *peer*s (clientes) que tiverem votos positivos acima de 25 e pendências abaixo de 50.

Com base nos escores, se um *peer* malicioso tentar aumentar sua reputação forjando várias identidades com o intuito de reputar uma de suas identidades falsas, será penalizado. Isso ocorre da seguinte forma: toda vez que um *peer* vota positivamente deve também replicar o conteúdo. Após o voto ser considerado positivo, o mecanismo de reputação verifica se o conteúdo pode ser baixado. Caso isso não ocorra durante um certo período, assume-se que o *peer* cliente está agindo de má fé e tentando simular o comportamento de um peer autêntico, isto pode dar indícios de ataque *sybil*.

6.5 Configuração da DHT

Nesta seção, é apresentada a configuração do repositório que facilmente pode ser feita utilizando-se a interface oferecida pela aplicação.

Para que o cliente DHT saiba qual repositório deve se conectar, o nome do servidor e porta TCP devem ser informados.

Uma vez feita esta configuração, a aplicação já estará apta a publicar, consultar e recuperar informações do repositório. Na Figura 6-18 é mostrada a interface do usuário para esta funcionalidade.

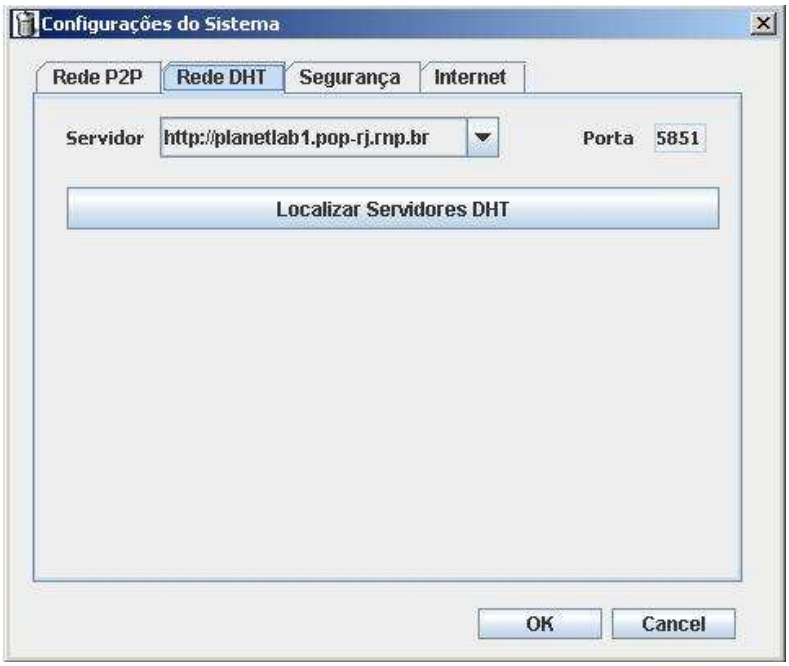


Figura 6-18 Interface de Configuração do Repositório (DHT)

6.6 Caso de Uso (Exemplo)

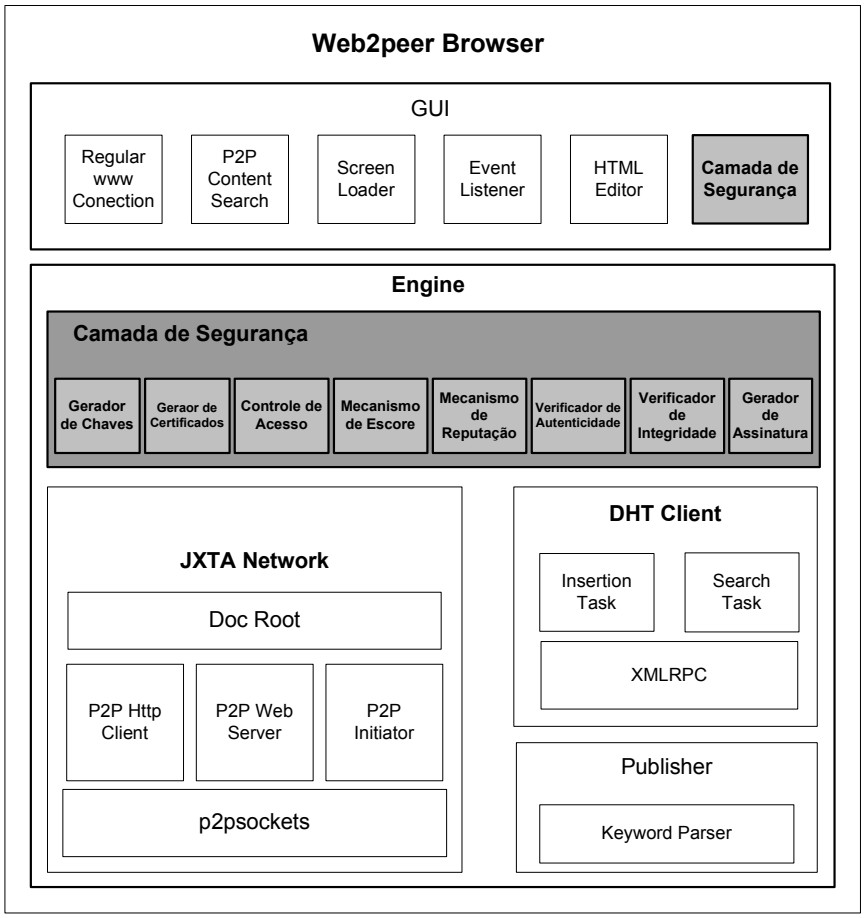
A seguir é mostrada a aplicação desenvolvida para servir de prova de conceito na avaliação do protótipo a partir do modelo.

6.6.1 Navegador Web2Peer

Seguindo o conceito *peer-to-peer*, onde os *peer*s são tanto provedores (servidores) quanto consumidores (clientes) foi implementado o Web2Peer [99] para fazer a parte de servidor nos *peers* do protótipo. Ou seja, cada *peer* possui um servidor de páginas http interno e um cliente (navegador Web) que acessa outros *peer*s para obter os conteúdos disponibilizados como páginas html, por exemplo.

O Web2Peer oferece um servidor Web e um navegador Web dentro de uma rede P2P. O navegador possui um componente chamado *publisher* que efetua a publicação de conteúdos na rede *peer-to-peer*. No Web2Peer, a publicação é feita na DHT implementada pelo Bamboo.

Na Figura 6-19, é mostrada a arquitetura do Web2Peer onde foi inserida a proposta da camada de segurança.



Fonte: [99]

Figura 6-19 Arquitetura em Blocos

Considerando-se uma agência de notícias, como exemplo, onde todas as publicações são disponibilizadas via Internet. Para evitar o gasto com sistemas de alta-disponibilidade, além do ponto único de falhas (servidor Web da agência) e a dependência de um *webdesigner* a agência optou por utilizar a rede P2P para veicular suas notícias.

Os integrantes da agência (jornalistas e editores) disponibilizam reportagens em seus próprios computadores ao invés de enviarem os conteúdos para um *webdesigner* para publicação no servidor Web. Um jornalista pode estar em um local remoto ao produzir uma notícia e ao disponibilizá-la em seu computador estará permitindo imediatamente a sua leitura, sem precisar enviar a notícia até a central de notícias para ser diagramada e disponibilizada em uma página Web.

Se o jornalista precisar do aval/revisão do editor ou algo assim, a notícia (documento) pode ser cifrada na chave pública do editor, o conteúdo é então armazenado localmente e as palavras-chave e a cadeia de autorização SDKI/SPKI podem ser publicadas na

DHT (repositório compartilhado). O editor pode obter o documento, revisá-lo, armazená-lo localmente e publicar as mesmas palavras-chave e a cadeia de autorização no repositório.

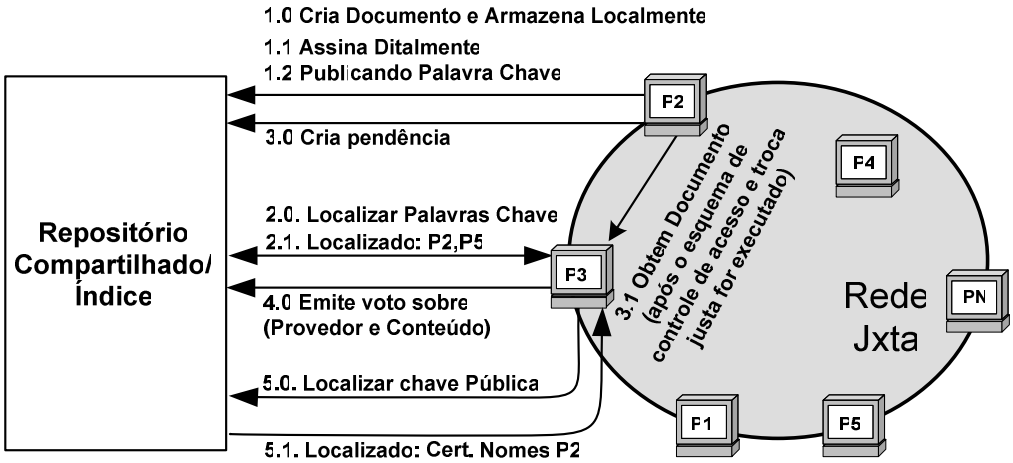


Figura 6-20 Cenário de uma Agência de Notícias

Na Figura 6-20, observa-se em mais detalhes os passos desse cenário. Inicialmente, deve-se considerar no cenário que todos os jornalistas que queriam publicar suas notícias obrigatoriamente devem possuir um par de chaves assimétricas, e preferencialmente um certificado de nomes SPKI/SDSI publicado na DHT – o certificado facilita a identificação do jornalista.

A partir do navegador adaptado com a camada de segurança, é possível editar páginas html. Após a edição da matéria no editor html, o jornalista assina digitalmente a notícia. Em seguida o jornalista deposita a notícia no diretório compartilhado (diretório padrão do servidor de páginas http), gera as palavras-chave, assinando o conteúdo e as palavras-chave digitalmente e em seguida gerando a identificação do objeto (notícia), seguindo de sua publicação na DHT, (Figura 6-20, passo 1).

Supondo que em dado momento, um *peer*, P3 por exemplo, faça uma busca por aquelas palavras-chave anteriormente publicadas e como resultado receba várias *fontes* provedoras (servidores) de onde baixar as publicações (Figura 6-20, passo 2) e suas respectivas reputações.

Dentre as respostas está a publicação feita pelo *peer* P2. Então, P3 requisita o conteúdo ao servidor Web executando no *peer* P2 (supondo que o mesmo possua um grau de reputação maior gerado pelo mecanismo de escore), através da rede JXTA, é então gerada um pendência por parte do *peer* provedor P2. A seguir, é feito o confronto das regras de controle de acesso e troca justa (Figura 6-20, passo 3).

Uma vez baixado o conteúdo do *peer* P2, o consumidor *peer* P3 verifica a assinatura do arquivo, avalia o conteúdo e da sua opinião (voto positivo/neutro/negativo) sobre o provedor e o seu respectivo conteúdo (Figura 6-20, passo 4). Se o voto for positivo o mesmo irá replicar o conteúdo.

É importante que P3 registre a chave pública de origem (chave do *peer* P2) do conteúdo, pois se o conteúdo for replicado, a informação do servidor P2P de origem do conteúdo replicado será perdida. O registro de origem do conteúdo serve para construir o histórico e para fins de irretratabilidade.

Após verificar a assinatura do conteúdo do *peer* P2 e avaliá-lo, P3 pode desejar saber a quem pertence a chave pública que assinou o arquivo. Neste caso, P3 deve recorrer a DHT novamente para recuperar o certificado de nomes correspondente à chave de P2 (Figura 6-20, passo 5).

O cenário foi implementado usando a tecnologia citada no capítulo 3 e os aspectos de implementação do Capítulo 6, mas os casos de uso não se limitam a aplicação citada e sim abrangem qualquer cenário onde a proposta seja viável.

6.7 Ambiente de Testes

Para realizar os testes necessários do protótipo deste trabalho, foi criado um ambiente de testes. Toda a estrutura física e lógica é apresentada a seguir, bem como serviços auxiliares que foram necessários para o funcionamento do protótipo.

Foram testados dois *peers* executados em máquinas fisicamente diferentes e separadas pela Internet nas plataformas Windows e Linux, os serviços de DHT e de *peer* Relay/Rendezvous foram executados em ambiente Linux, a ilustração desse ambiente é vista na Figura 6-21.

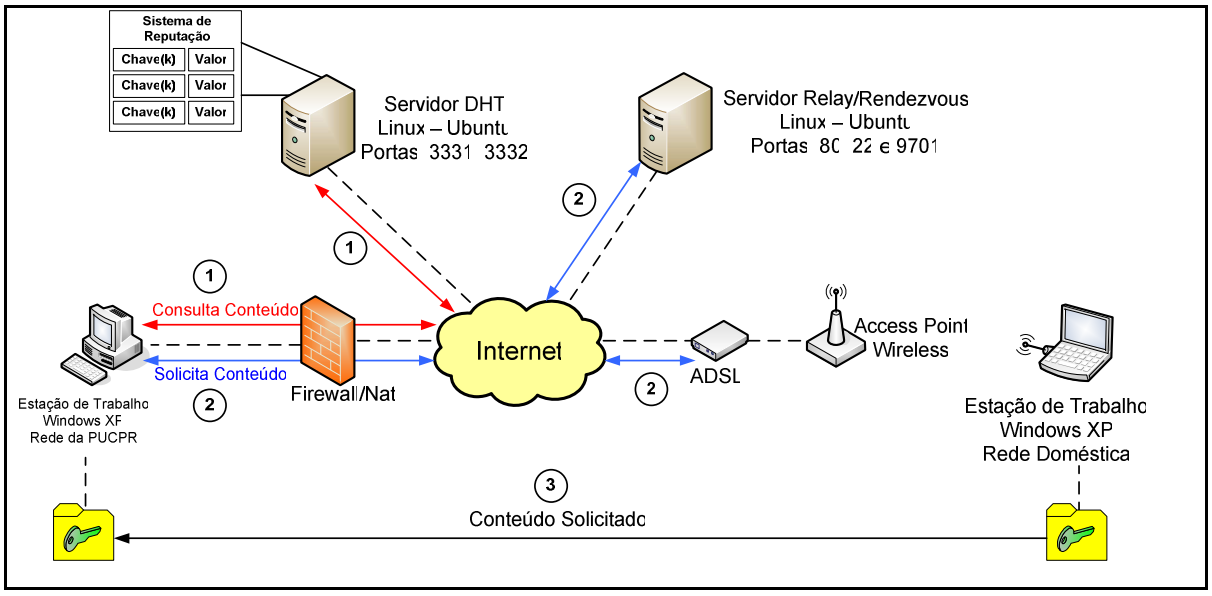


Figura 6-21 Ambiente de Testes

Na fase de testes os *peers* tinham comunicação por meio da Internet, sendo que um estava na rede da PUCPR e outro fazia parte de uma rede doméstica, os dois *peers* tinham acesso aos servidores de DHT e Relay/Rendezvous na Internet.

O *peer* na rede da PUCPR tinha acesso por uma rede Ethernet de 10/100Mbps, e o mesmo estava atrás de um Firewall. Do outro lado foi utilizado um ADSL com conexão de 2Mbps e uma rede local Wireless de 10Mbps.

Tanto o servidor DHT, quanto o Relay/Rendezvous foram hospedados no data center da PUCPR com endereços de Internet válidos. O servidor DHT estava aguardando conexões nas portas 333 e 3332, enquanto que o Relay/Rendezvous aguardava por conexões nas portas 80, 22 e 9701.

A validação do funcionamento do protótipo foi feita da seguinte maneira, utilizando a estrutura do navegador Web2Peer, criou-se páginas HTML com os seguintes tamanhos: 1Kb, 10Kb, 100Kb, 1Mb, 10Mb, 100Mb e as páginas passaram pelo processo de assinatura com chaves públicas de (1024 e 20048 bits), bem como uma associação de palavras chave referentes ao seu conteúdo.

Por fim é feita a consulta pelas palavras previamente associadas usando o serviço do servidor DHT, e posteriormente a transferência dos arquivos através do JXTA com base no serviço do servidor de Relay/Rendezvous. Após a validação do conteúdo é emitido um parecer ou voto, reputando o conteúdo e seu fornecedor, caso o conteúdo passe na avaliação o mesmo é então replicado.

6.8 Avaliação do Protótipo

Nessa seção são descritas as médias feitas para validar o protótipo. As tomadas de tempo foram feitas levando em conta as fases do protótipo, no que se refere à publicação e busca de conteúdos.

Toda a medição foi feita em um ambiente inicialmente sem a camada de segurança implementada no protótipo e posteriormente com a utilização da camada, para que se pudesse mensurar o impacto no uso e seu custo benefício.

Quanto à publicação de conteúdos (páginas html), foram publicadas vinte páginas tomando-se o tempo gasto para a publicação sem disponibilizar qualquer recurso para prover a autenticidade e integridade do conteúdo. Foram refeitos os mesmos passos, mas com a inserção da camada de segurança, como ilustrado na Figura 6-22.

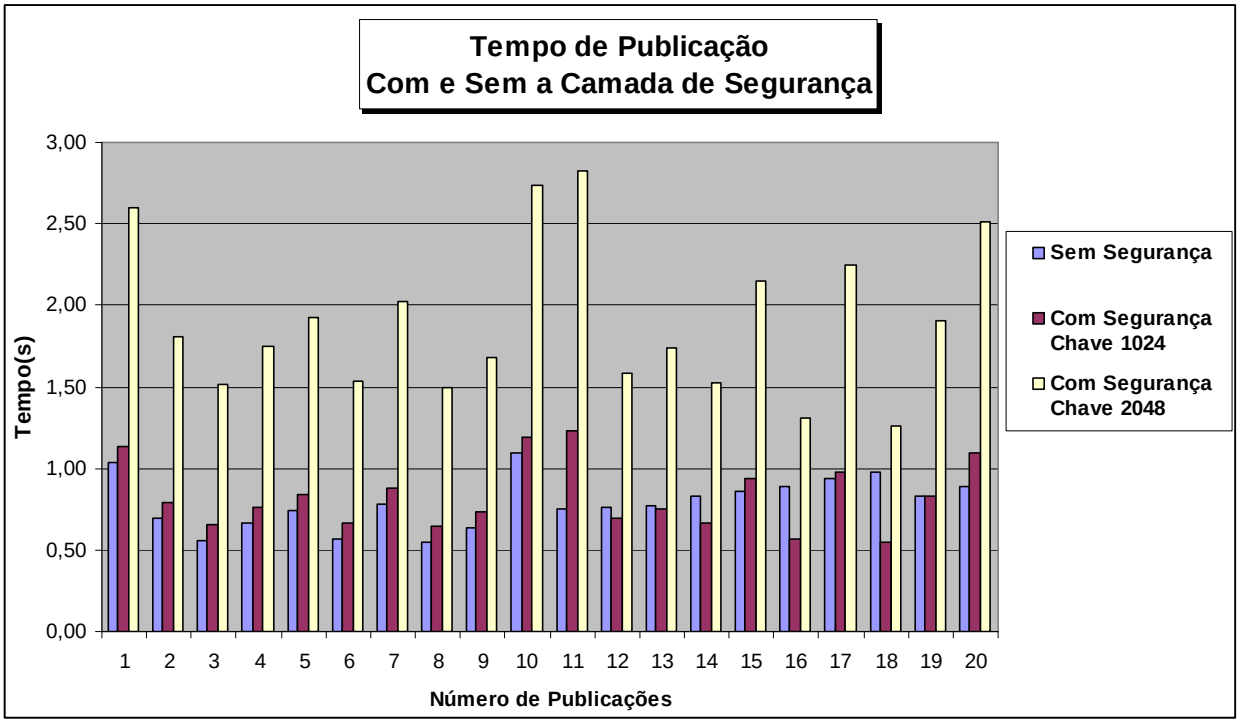


Figura 6-22 Tempo de Publicação de Conteúdo

Foi observado que a inserção da camada de segurança (medindo o tempo para prover autenticidade e integridade) acresceu treze por cento (13%) no tempo médio de publicação com uma chave criptográfica de 1024 bits e de vinte e oito por cento (28%), com uma chave de 2048 bits. Porém, o custo benefício torna-se claro se, por exemplo, um *peer* cliente baixar um conteúdo e o mesmo não for autêntico, isso gera um alto grau de insatisfação.

O mesmo procedimento de medida de tempo para a pesquisa de conteúdo foi adotado. Tais valores podem ser vistos na Figura 6-23.

Foram realizadas vinte (20) consultas variando-se o número de palavras-chave entre (1, 3, 5 e 7). Como resultado da busca, foram obtidas dez (10) fontes (servidores P2P) do conteúdo pesquisado. Os resultados comparam a busca com e sem a camada de segurança, filtrando as respostas segundo os critérios previamente configurados pelo usuário do *peer* cliente, juntamente com a lista ordenada por reputação baseada no escore.

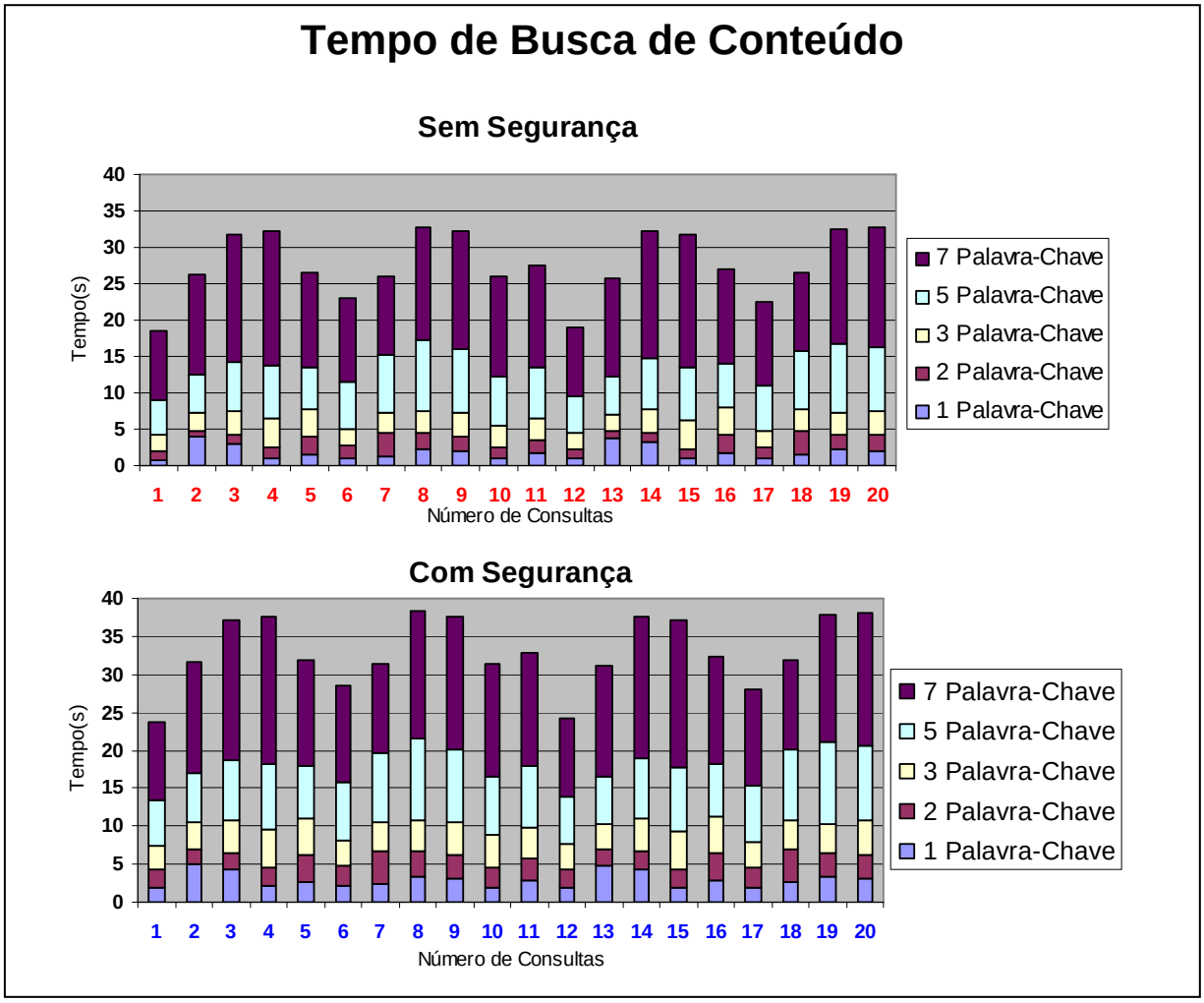


Figura 6-23 Tempo de Busca de Conteúdo

Como resultado dessa avaliação, percebeu-se um aumento em dezenove por cento (19%) no tempo de busca de conteúdo, pois como mencionado anteriormente neste trabalho, a camada de segurança utiliza os critérios de seleção adicionais para reduzir a possibilidade de baixar um conteúdo indesejado de *peers* fornecedores que não tenham uma boa reputação na rede.

Uma vez localizado o conteúdo e servidor P2P, foi avaliado o tempo gasto para baixar um conteúdo sem nenhum critério do ponto de vista de segurança, bem como

utilizando dos mecanismos propostos de reputação e escore para avaliar a viabilidade do sistema como ilustram as seguintes figuras: Figura 6-24 a Figura 6-29.

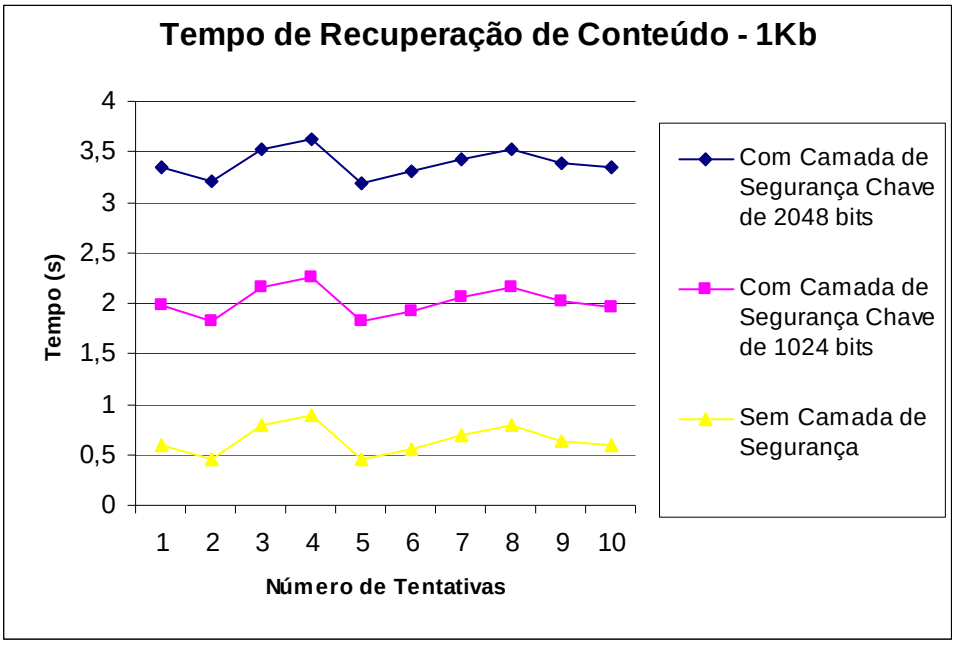


Figura 6-24 Tempo de Recuperação de Conteúdo 1Kb

Como ilustrado na Figura 6-24, utilizando-se de um conteúdo de tamanho 1Kb o qual foi recuperado dez vezes (10) observou-se que o tempo médio não sofreu grande variação.

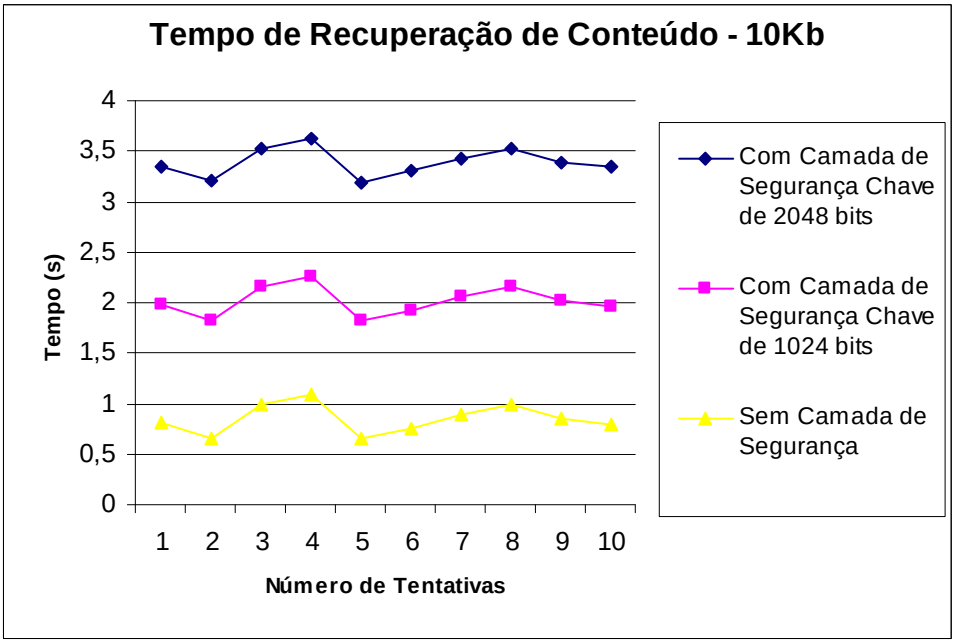


Figura 6-25 Tempo de Recuperação de Conteúdo 10Kb

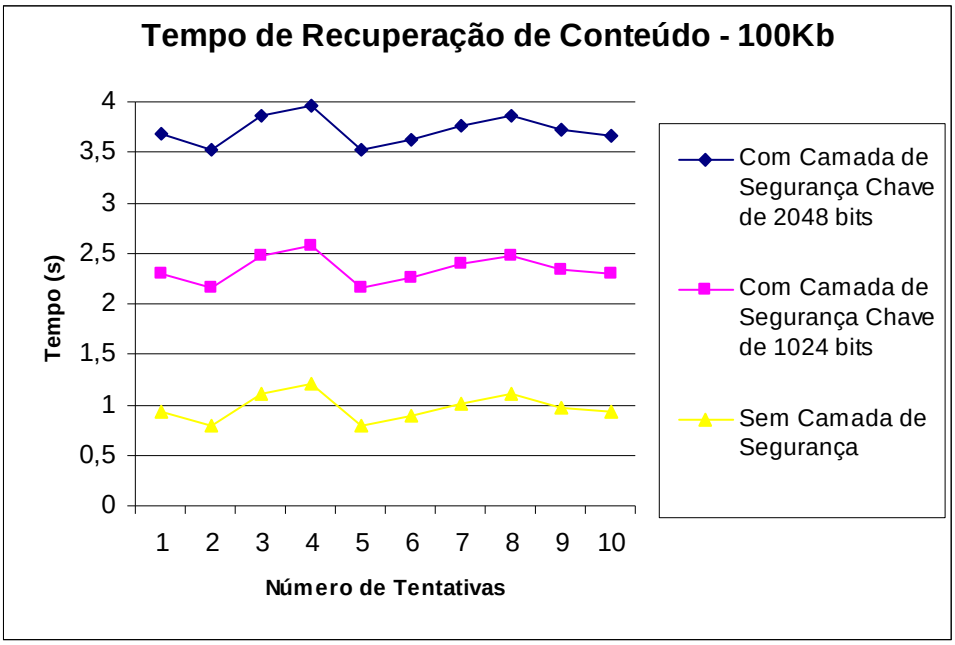


Figura 6-26 Tempo de Recuperação de Conteúdo 100Kb

Com os arquivos variando de 1Kb até 100Kb de tamanho, o tempo médio entre o uso convencional e com a camada de segurança (com chaves de 1024 e 2048) observou-se que os tempos se mantiveram estáveis sem muitas diferenças significativas.

Mas a medida que os arquivos foram aumentando o tamanho o tempo se aproximou mais, isto é, o tempo gasto com a aplicação da camada de segurança tornou-se muito pequeno em relação ao tempo gasto para baixar os arquivos, compensando o uso da camada de segurança, para evitar conteúdo ruim.

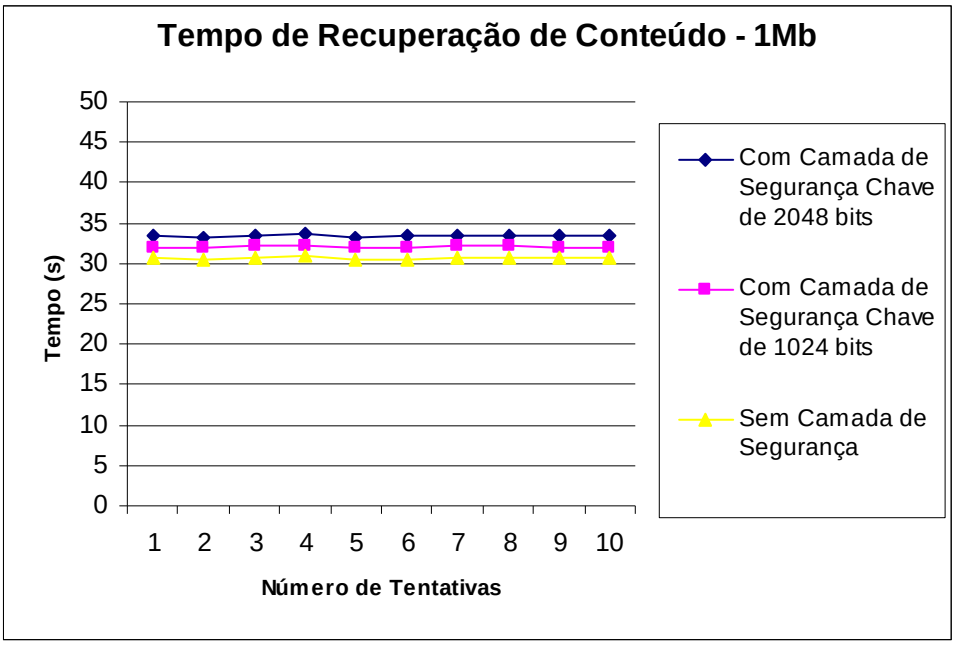


Figura 6-27 Tempo de Recuperação de Conteúdo 1Mb

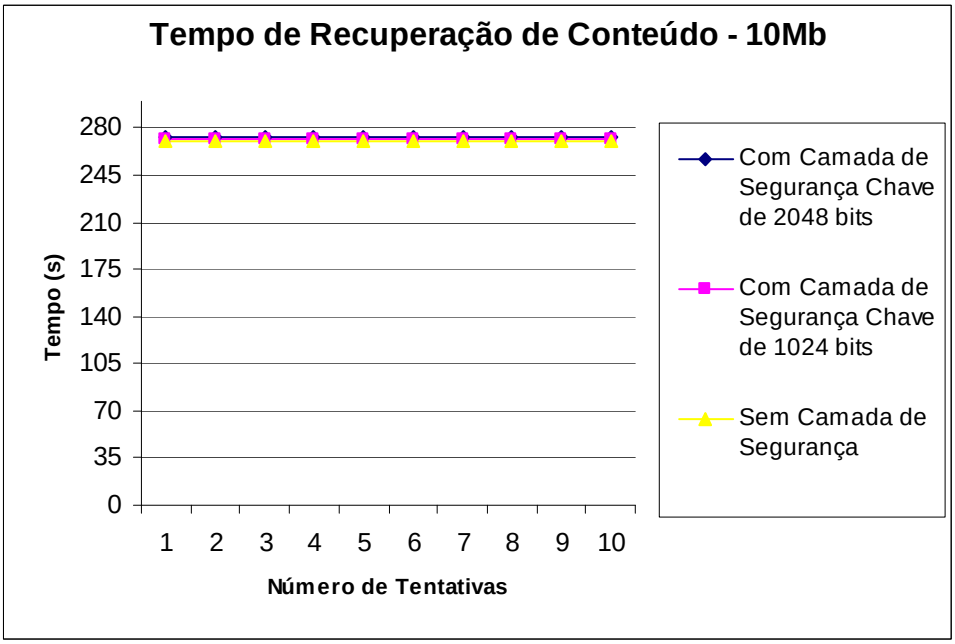


Figura 6-28 Tempo de Recuperação de Conteúdo 10Mb

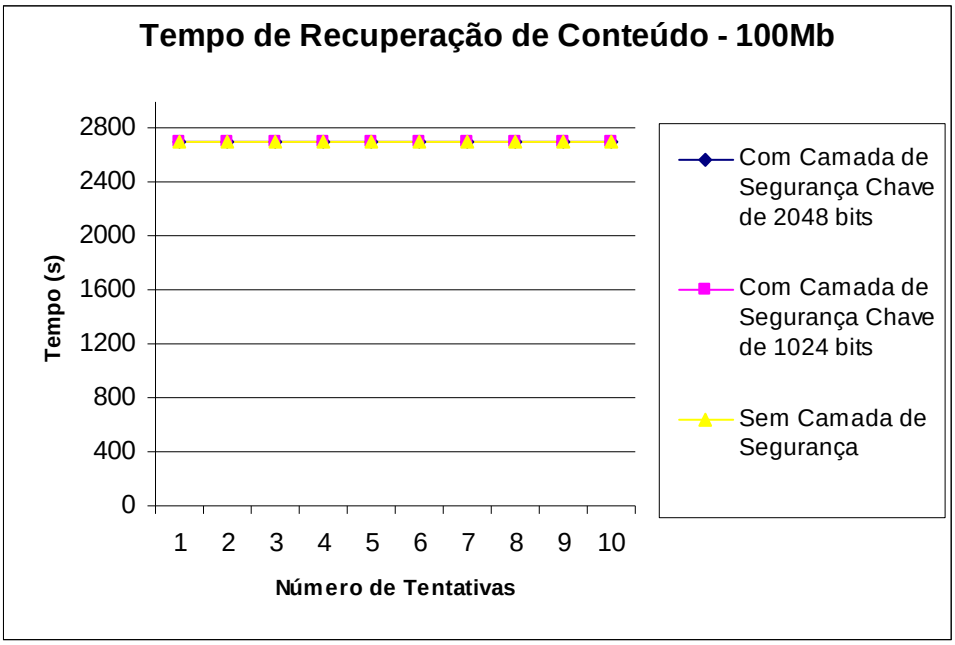


Figura 6-29 Tempo de Recuperação de Conteúdo 100Mb

Mesmo como o aumento do tempo de recuperação do conteúdo da rede, devido aos mecanismos que provêm a autenticidade e integridade do conteúdo, bem como a reputação do mesmo e a do seu fornecedor, Esse tempo pode ter pouca relevância se comparado ao tempo perdido quando se obtém um conteúdo ruim com tamanho significativo e que levou horas para baixar.

As medidas de tempo mostram que a camada de segurança onera a troca de conteúdos, mas como prova de conceito, a qualidade das trocas justifica a sua inserção.

6.9 Conclusão do Capítulo

Este capítulo apresentou a implementação da camada de segurança do modelo proposto, abrangendo os mecanismos que visam garantir integridade, confidencialidade, autenticidade. Foi descrito também o controle de acesso e o mecanismo de troca justa, que objetivam dar um controle ao usuário do *peer* sobre o conteúdo que o mesmo compartilha na rede.

Detalharam-se os métodos empregados para se garantir a reputação de um *peer* provedor bem como o seu conteúdo, utilizando o mecanismo de reputação e escore, que utilizam um repositório para armazenar as informações de pendências e votos bem como toda a publicação de conteúdo envolvida no sistema.

Nesse capítulo foram abordadas as técnicas utilizadas para garantir a persistência da identificação de um *peer* e ao mesmo tempo garantir o anonimato. Por fim, a camada de segurança implementada integrada ao navegador Web2Peer [99]. Foram detalhados os módulos da camada da segurança bem como suas funcionalidades, códigos e interfaces.

Além da aplicação prática, isto é, o uso da camada implementada no Web2Peer, foi apresentado um estudo de caso, com a seqüência de passos necessários para uso da ferramenta, baseando-se em uma agência de notícias.

Pode-se observar que comparado ao sistema de Grupta [76] onde há uma maneira de dar créditos para o uso do sistema de troca de conteúdo, neste projeto de maneira semelhante é apresentado um conceito onde o usuário aumenta sua reputação e diferentemente do trabalho mencionado não perde créditos ou neste caso reputação pelo uso e sim quando comete alguma ação contra o uso normal na rede.

Também a forma de guardar estes créditos que em Grupta [76] é feita de maneira local, neste trabalho utiliza-se uma forma distribuída (DHT), tanto no sistema de créditos como aqui apresentado usa-se o conceito de Scores, mas o primeiro utiliza de maneira centralizada como visto na seção de trabalhos relacionados, diferente do exposto neste documento.

Nesse trabalho procurou-se aproveitar as idéias de alguns trabalhos como o Eingtrust, a qual foi o uso de DHT para Scores, mas evitou-se utilizar identificação por endereço IP e porta, ao contrário utilizou-se chaves públicas como identificadores.

Outra idéia aproveitada veio do Xrep [80]que avalia tanto a reputação do *peer*, quanto do seu conteúdo, isto também foi aplicado no modelo descrito neste documento. Mas

ao contrário do Xrep [80] quando um *peer* emite seu voto não é enviado o IP do *peer,* para evitar o comprometimento do anonimato.

Por fim utilizou-se as técnicas tradicionais de verificação de assinaturas para assegurar a integridade dos arquivos e o controle de acesso, baseou-se me certificados de autorização SDSI/SPKI e não nas técnicas centralizadoras tais como ACLs e RBAC.

Capítulo 7

7 Conclusão e Trabalhos Futuros

No inicio das redes *peer-to-peer, havia* apenas a preocupação na disponibilização do conteúdo, entretanto com o passar do tempo percebeu-se que a rede sofria de vários problemas de segurança que precisavam ser tratados.

Nesse trabalho, foram abordados alguns problemas no que se refere à integridade e autenticidade dos documentos e sua confidencialidade. Foi constatada a necessidade de um mecanismo que permita o compartilhamento de conteúdos de forma controlada, para que as aplicações *peer-to-peer* pudessem ser usadas em ambientes mais profissionais tais como corporativo e acadêmico. O trabalho propôs um mecanismo que fornece um controle de acesso utilizando uma infra-estrutura de chaves públicas utilizando o SDSI/SPKI.

Utilizaram-se os certificados auto-assinados, devido à natureza distribuída e dinâmica das redes *peer-to-peer*, para adequar o modelo de chaves públicas a esta arquitetura. Porém diferentemente das propostas relacionadas a este trabalho, foi proposto um mecanismo que dá credibilidade as chaves e certificados auto-assinados.

Tal mecanismo de reputação também serviu para reputar os conteúdos. Um mecanismo de escore foi proposto para que a reputação pudesse ser obtida, a partir de votos (positivos, neutros e negativos) dados aos *peers* provedores de conteúdo. Com esse serviço, sempre que um *peer* for baixar um conteúdo pode verificar previamente a reputação do conteúdo e do seu provedor.

Utilizou-se ainda a abordagem de chave pública como um identificador de *peer* persistente. Isso permitiu o controle efetivo contra a disseminação de conteúdo poluído,

reduzindo também os efeitos negativos que os nós-caronas trazem para a rede. A identificação baseada em chaves e o mecanismo de reputação da chave inibem também o ataque *sybil*.

O protótipo mostrou que o cenário com a agência de notícias, baseado em P2P, é vantajoso se comparado ao convencional (baseado em Web). As principais vantagens são à disponibilidade imediata de conteúdo sem a intermediação de *webdesigners* e principalmente a disponibilidade de conteúdo autêntico mesmo fora do site da agência.

Como trabalhos futuros são destacados os seguintes temas:

- Por tar estes recursos de segurança utilizados no Web2Peer para o Navegador Mozilla na forma de *plug-in*;
- Utilizar outros tipos de DHTs para alcançar um ganho de performance e flexibilidade de inserção de valores;
- Criar novos filtros para consulta com base em reputação de *peers*;
- Criar novos serviços associados a DHT usando programação com o SEDA.

Por fim outro fator relevante com a evolução desse trabalho, foi que outros trabalhos de mestrado estão sendo realizados com base nas contribuições feitas nesse documento.

Estão sendo trabalhados focando no uso de dispositivos móveis, utilizando DHT e JXTA-ME.

Referências Bibliográficas

[1] N. Daswani, H. Garcia-Molina, and B. Yang, "Open Problems in Data-Sharing Peer-to-Peer Systems," *Proceedings of the 9th International Conference on Database Theory*, pp. 1-15, 2003.

[2] J. Liang, R. Kumar, Y. Xi, and K. Ross, "Pollution in p2p file sharing systems," *IEEE Infocom*, 2005.

[3] D. E. R. Denning, *Cryptography and Data Security*: Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 1982.

[4] E. G. Amoroso, *Fundamentals of computer security technology*: Prentice-Hall, Inc. Upper Saddle River, NJ, USA, 1994.

[5] D. Russell and G. T. Gangemi, *Computer security basics*: O'Reilly & Associates Sebastopol, CA, 1991.

[6] C. E. Landwehr, "Computer security," *International Journal of Information Security*, vol. 1, pp. 3-13, 2001.

[7] P. Ashley, "Authorization For a Large Heterogeneous Multi-Domain System," *Australian Unix and Open Systems Group National Conference*, pp. 159-169, 1997.

[8] N. C. Damianou, "A Policy Framework for Management of Distributed Systems," Imperial College, 2002.

[9] T. Y. C. Woo and S. S. Lam, "A framework for distributed authorization," *Proceedings of the 1st ACM conference on Computer and communications security*, pp. 112-118, 1993.

[10] T. Ryutov and C. Neuman, "Representation and evaluation of security policies for distributedsystem services," *DARPA Information Survivability Conference and Exposition, 2000. DISCEX'00. Proceedings*, vol. 2, 2000.

[11] R. Shirley, "Internet Security Glossary," *IETF Newtork Working Group, RFC-2828. May*, 2000.

[12] J. A. Goguen and J. Meseguer, "Security policies and security models," *Proc. IEEE Symposium on Security and Privacy*, pp. 11–20, 1982.

[13] R. Sandhu and P. Samarati, "Authentication, access control, and audit," *ACM Computing Surveys (CSUR)*, vol. 28, pp. 241-243, 1996.

[14] S. Osborn, R. Sandhu, and Q. Munawer, "Configuring Role-Based Access Control to Enforce Mandatory and Discretionary Access Control Policies," *ACM Transactions on Information and System Security*, vol. 3, pp. 85-106, 2000.

[15] A. Santin, J. Fraga, E. Mello, and F. Siqueira, "Teias de federações como extensões ao modelo de autenticação e autorização SDSI/SPKI," *Anais XXI Simposio Brasileiro de Redes de Computadores*, pp. 553–568, 2003.

[16] B. Schneier, "Applied Cryptografy," *John Willey*, Novembro de1995.

[17] F. Nist, "180-1: Secure Hash Standard," Abril de 1995.

[18] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris, *Resilient overlay networks*: ACM Press New York, NY, USA, 2001.

[19] Gnutella, "Gnutella http://www.gnutella.com" Acesso: [20 de Dezembro de 2004].

[20] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong, "Freenet: A distributed anonymous information storage and retrieval system," *Workshop on Design Issues in Anonymity and Unobservability*, vol. 320, 2000.

[21] S. Y. Shi and J. S. Turner, "Routing in overlay multicast networks," *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 3, 2002.

[22] K. Seada, M. Zuniga, A. Helmy, and B. Krishnamachari, "Energy-efficient forwarding strategies for geographic routing in lossy wireless sensor networks," *Proceedings of the 2nd international conference on Embedded networked sensor systems*, pp. 108-121, 2004.

[23] D. G. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. Morris, "Resilient overlay networks. 18th ACM SOSP," *Banff, Canada, Oct*, 2001.

[24] L. Subramanian, I. Stoica, H. Balakrishnan, and R. Katz, "OverQoS: An Overlay Based Architecture for Enhancing Internet QoS," *Proceedings of NSDI*, 2004.

[25] D. I. R. Kurmanowytsch, *Omnix: An Open Peer-to-Peer Middleware Framework*: PhD thesis, Vienna University of Technology, Feb 2004, 2004.

[26] G. Shah, "Distributed Data Structures for Peer-to-Peer Systems," Yale University, 2003.

[27] PWG, "Peer-to-Peer Working Group" http://www.p2pwg.org/, Acesso: [03 de Maio de 2005].

[28] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," *Proceedings of the 2001 SIGCOMM conference*, vol. 31, pp. 149-160, 2001.

[29] A. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems," *Lecture Notes in Computer Science*, vol. 2218, pp. 329–350, 2001.

[30] S. Androutsellis-Theotokis and D. Spinellis, "A survey of peer-to-peer content distribution technologies," *ACM Computing Surveys (CSUR)*, vol. 36, pp. 335-371, 2004.

[31] L. L. C. Napster, "Napster Online," http://www.napster.com Acesso: [20 de Março de 2004].

[32] E. F. Foundation-Eff, "Napster Cases, Grokster Case," presented at Retrieved March, 2005.

[33] SETI@Home, "SETI@Home", http://setiathome.berkeley.edu, Acesso: [09 de Julho de 2006].

[34] D. S. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, B. Richard, S. Rollins, and Z. Xu, "Peer-to-Peer Computing," *HP Laboratories Palo Alto, March*, 2002.

[35] Groove, "Groove: Introduction to Groove" Acesso: [10 de Setembro de 2006].

[36] FastTrack, "The FastTrack Protocol," http://www.fasttrack.com Acesso: [06 de Outubro de 2005].

[37] Kazza, "Kazaa Media Desktop. Kazaa homepage, http://www.kazaa.com, Acesso: [01 de Dezembro de 2005].

[38] Grokster, "Grokster homepage", http://www.grokster.com/, Acesso: [25 de Novembro de 2005].

[39] M. Waldman and D. Mazières, "Tangler: a censorship-resistant publishing system based on document entanglements," *Proceedings of the 8th ACM conference on Computer and Communications Security*, pp. 126-135, 2001.

[40] W. Nejdl, B. Wolf, C. Qu, S. Decker, M. Sintek, A. Naeve, M. Nilsson, M. Palmér, and T. Risch, "EDUTELLA: a P2P networking infrastructure based on RDF," *Proceedings of the eleventh international conference on World Wide Web*, pp. 604-615, 2002.

[41] R. Dingledine, M. Freedman, and D. Molnar, "Peer-to-peer: Harnessing the power of disruptive technology," O'Reilly, 2001.

[42] S. Ratsanamy, P. Francis, M. Handley, and R. Karp, "A Scalable Content-Addressable Network," *ACM SIGCOMM Conference*, pp. 161–172, 2001.

[43] P. Druschel and A. Rowstron, "PAST: A large-scale, persistent peer-to-peer storage utility," *Proc. HotOS VIII*, 2001.

[44] B. Y. Zhao, J. Kubiatowicz, and A. D. Joseph, "Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and Routing," Computer Science Division University of California, Berkeley, 2001.

[45] V. Kalogeraki, D. Gunopulos, and D. Zeinalipour-Yazti, "A local search mechanism for peer-to-peer networks," *Proceedings of the eleventh international conference on Information and knowledge management*, pp. 300-307, 2002.

[46] ICQ, "ICQ homepage," http://www.icq.com, Acesso: [20 de Março de 2005].

[47] Threedegrees, "Threedegrees homepage," http://www.kolabora.com/news/2004/12/06/p2p_desktop_social_interaction_for.htm, Acesso: [10 de Dezembro de 2006].

[48] Jabber, "Jabber Software Foundation," http://www.jabber.org, Acesso: [05 de Fevereiro de 2005].

[49] Gnunet, "Gnunet homepage", http://www.ovmj.org/GNUnet, Acesso: [03 de Outubro de 2005].

[50] genome@HOME, "genome@HOME homepage", http://genomeathome.stanford.edu, Acesso: [17 de Agosto de 2005].

[51] K. Aberer, "P-Grid: A self-organizing access structure for P2P information systems," *Lecture Notes in Computer Science*, vol. 2172, pp. 179-185, 2001.

[52] A. Arora, C. Haywood, and K. S. Pabla, "JXTA for J2ME–Extending the Reach of Wireless with JXTA Technology," *Sun Microsystems, March*, 2002.

[53] I. Gupta, K. Birman, P. Linga, A. Demers, and R. van Renesse, "Kelips: Building an Efficient and Stable P2P DHT Through Increased Memory and Background Overhead," *Group*, vol. 30, pp. 23ms, 2003.

[54] L. Gong, "Project JXTA: A Technology Overview," Sun Whitepaper, 2001.

[55] JXTA, "JXTA Project", http://www.jxta.org, Acesso [01 de Julho de 2004].

[56] L. Olson, ".NET P2P: Writing Peer-to-Peer Networked Apps with the Microsoft .NET Framework," *MSDN Magazine*, vol. 16, 2001.

[57] J. Edwards, *Peer-to-peer Programming with Groove*: Addison-Wesley Professional, 2002.

[58] J. Gradecki, *Mastering JXTA*: Wiley Publishing Indianapolis, Indiana, 2002.

[59] B. Traversat, A. Arora, M. Abdelaziz, M. Duigou, C. Haywood, J. C. Hugly, E. Pouyoul, and B. Yeager, "Project JXTA 2.0 Super-Peer Virtual Network," *Sun Microsystem White Paper. www. jxta. org/project/www/docs,* Acesso: [18 de Maio de 2003].

[60] I. Sun Microsystems, "“Project JXTA v2. 0: Java™ Programmer’s Guide," Sun Microsystems Inc. Available: http://www. jxta. org/docs/jxtaprogguide_final. pdf, Acesso: [30 de Maio 2005].

[61] S. Microsystems, "JXTA v2. 0 Protocols Specification," *Sun Microsystems-White Papers*, 2004.

[62] B. J. Wilson, "*JXTA"*. New Riders Boston, 2002.

[63] D. E. Clarke, "SPKI/SDSI HTTP Server/Certificate Chain Discovery in SPKI/SDSI," Massachusetts Institute Of Technology, 2001.

[64] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen, "SPKI Certificate Theory," RFC 2693, September 1999.

[65] R. L. Rivest and B. Lampson, "SDSI-A Simple Distributed Security Infrastructure," *Manuscript*, 1996.

[66] A. Santin, J. Fraga, E. Mello, and F. Siqueira, "Extending the SPKI/SDSI model through federation webs," *Proc. 7th IFIP Conference on Communications and Multimedia Security)*, 2003.

[67] K. Truelove, "Gnutella: Alive, well, and changing fast," http://www. openp2p. com/pub/a/p2p/2001/01/25/truelove0101.html, Acesso: [22 deAgosto de 2004].

[68] J. A. Mussini, "Distributed Hash Table: The State of the Art," 2006.

[69] F. a. Favarim, "Espaço de Tuplas como Suporte para Serviços em Grids Computacionais," Universidade Federal de Santa Catarina, 2005.

[70] M. Dischinger, "A flexible and scalable peer-to-peer multicast application using Bamboo," *Report of the University of Cambridge Computer Laboratory*, 2004.

[71] Bamboo, "Bamboo DHT http://www.bamboo-dht.org/", Acesso: [13 de Setembro de 2006].

[72] S. Rhea, B. Godfrey, B. Karp, J. Kubiatowicz, S. Ratnasamy, S. Shenker, I. Stoica, and H. Yu, "OpenDHT: a public DHT service and its uses," *Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications*, pp. 73-84, 2005.

[73] R. M. Stallman, "GNU General Public License," *GNU Project--Free Software Foundation*, 1991.

[74] F. Dabek and M. Frans, "Kaashoek, David Karger," *Robert Morris, and Ion Stoica," Widearea Cooperative Storage with CFS," In SOSP*, 2001.

[75] K. Walsh and E. G. Sirer, "Experience with an Object Reputation System for Peer-to-Peer Filesharing," *USENIX NSDI*, 2006.

[76] E. Damiani, S. Paraboschi, P. Samarati, and F. Violante, "A reputation-based approach for choosing reliable resources in peer-to-peer networks," *Proceedings of the 9th ACM conference on Computer and communications security*, pp. 207-216, 2002.

[77] E. Mello, J. Fraga, and A. Santin, "O uso do SPKI/SDSI em redes P2P," *I Workshop de Redes Peer-to-Peer - WP2P 2005*, 2005.

[78] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina, "The Eigentrust algorithm for reputation management in P2P networks," *Proceedings of the twelfth international conference on World Wide Web*, pp. 640-651, 2003.

[79] M. Gupta, P. Judge, and M. Ammar, "A reputation system for peer-to-peer networks," *Proceedings of the 13th international workshop on Network and operating systems support for digital audio and video*, pp. 144-152, 2003.

[80] M. P. Barcellos, L. P. Gaspary, and E. Madeira, "Segurança em Redes P2P: Princípios, Tecnologias e Desafios," in *Livro de Minicursos, SBRC 2006*, vol. 1, 2006, pp. 211-260.

[81] H. Tran, M. Hitchens, V. Varadharajan, and P. Watters, "A Trust based Access Control Framework for P2P File-Sharing Systems," *System Sciences, 2005. HICSS'05. Proceedings of the 38th Annual Hawaii International Conference on*, pp. 302c-302c, 2005.

[82] M. Bishop and M. A. Bishop, *Computer Security: Art and Science*: Addison-Wesley Professional, 2003.

[83] J. S. Park and J. Hwang, "Role-based access control for collaborative enterprise in peer-to-peer computing environments," *Proceedings of the eighth ACM symposium on Access control models and technologies*, pp. 93-99, 2003.

[84] F. Dabek, E. Brunskill, M. F. Kaashoek, D. Karger, R. Morris, I. Stoica, and H. Balakrishnan, "Building peer-to-peer systems with Chord, a distributed lookup service," *Proceedings of the 8th Workshop on Hot Topics in Operating Systems (HotOS-VIII)*, pp. 81–86, 2001.

[85] S. Marti and H. Garcia-Molina, "Identity crisis: anonymity vs reputation in P2P systems," *Peer-to-Peer Computing, 2003.(P2P 2003). Proceedings. Third International Conference on*, pp. 134-141, 2003.

[86] L. Tang, "Identifying Resource Authenticity in P2P Networks", Department of Automation, Tsinghua University, Beijing, China, 2004.

[87] A. Wierzbicki, A. Zwierko, and Z. Kotulski, "Authentication with controlled anonymity in P2P systems," *Parallel and Distributed Computing, Applications and Technologies, 2005. PDCAT 2005. Sixth International Conference on*, pp. 871-875, 2005.

[88] R. C. Merkle, "Secure communications over insecure channels," *Communications of the ACM*, vol. 21, pp. 294-299, 1978.

[89] J. Brandt, "Zero-Knowledge Authentication Scheme with Secret Key Exchange," *Journal of Cryptology*, vol. 11, pp. 147-159, 1998.

[90] A. Cheng and E. Friedman, "Sybilproof reputation mechanisms," *Applications, Technologies, Architectures, and Protocols for Computer Communication*, pp. 128-132, 2005.

[91] R. Krishnan, M. D. Smith, Z. Tang, and R. Telang, "The Impact of Free-Riding on Peer-to-Peer Networks."

[92] C. Schmidt and M. Parashar, "Enabling flexible queries with guarantees in P2P systems," *Internet Computing, IEEE*, vol. 8, pp. 19-26, 2004.

[93] J. E. Elien, "Certificate Discovery Using SPKI/SDSI 2.0 Certificates," Massachusetts Institute of Technology, 1998.

[94] J. J. WebServer, "jetty.mortbay.org", http://jetty.mortbay.org/ Acesso: [23 Julho de 2006].

[95] M. Welsh, "The Staged Event-Driven Architecture for Highly-Concurrent Server Applications," *University of California, Berkeley*, 2000.

[96] D. Mazieres, "Self-certifying File System," Massachusetts Institute of Technology, 2000.

[97] S. Ajmani, "JSDSI: A Java SDSI/SPKI implementation", http://jdsi.sourceforge.net Acesso: [02 de Abril de 2005].

[98] A. Morcos, "A Java Implementation of Simple Distributed Security Infrastructure," Massachusetts Institute of Technology, 1998.

[99] L. C. L. H. B. Ribeiro, A. O.Santin, N. L. Brisola, "Implementing a Peer-to-Peer Web Browser for Publishing and Searching Web Pages on Internet. 21st International IEEE Conference on Advanced Information Networking and Applications, AINA-07," presented at AINA-07, Niagara Falls, Canada, 2007.