EDSON OSSAMU KAGEYAMA

UM SISTEMA DE CORREIO ELETRÔNICO BASEADO NO PARADIGMA PEER-TO-PEER

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

Curitiba PR
Junho de 2008

EDSON OSSAMU KAGEYAMA

UM SISTEMA DE CORREIO ELETRÔNICO BASEADO NO PARADIGMA PEER-TO-PEER

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

Área de concentração: *Ciência da Computação*

Orientador: Prof. Dr. Carlos Alberto Maziero

Curitiba PR
Junho de 2008

Dados da Catalogação na Publicação
Pontifícia Universidade Católica do Paraná
Sistema Integrado de Bibliotecas – SIBI/PUCPR

Kageyama, Edson Ossamu
K11s Um sistema de correio eletrônico baseado no paradigma peer-to-peer /
2008 Edson Ossamu Kageyama ; orientador, Carlos Alberto Maziero. – 2008.
xxii, 61 f. : il. ; 30 cm

Dissertação (mestrado) – Pontifícia Universidade Católica do Paraná, Curitiba, 2008
Bibliografia: f. 59–63

1. Sistemas de transmissão de dados. 2. Correio eletrônico. 3. Redes de computação. 4. Arquitetura de redes de computadores. I. Maziero, Carlos Alberto. II. Pontifícia Universidade Católica do Paraná. Programa de Pós-Graduação em Informática. III. Título.

CDD 20. ed. – 004.66

Biblioteca Central

Pontifícia Universidade Católica do Paraná
Centro de Ciências Exatas e de Tecnologia
Programa de Pós-Graduação em Informática

ATA DE DEFESA DE DISSERTAÇÃO DE MESTRADO
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

DEFESA DE DISSERTAÇÃO Nº 05/2008

Aos 07 dias do mês de março de 2008 realizou-se a sessão pública de Defesa da Dissertação **"Um Sistema de Correio Eletrônico Baseado no Paradigma *Peer-to-Peer*"**, apresentada pelo aluno **Edson Ossamu Kageyama** como requisito parcial para a obtenção do título de **Mestre em Informática**, perante uma Banca Examinadora composta pelos seguintes membros:

Prof. Dr. Carlos Alberto Maziero PUCPR (Orientador)	(assinatura)	aprov (aprov/reprov.)
Prof. Dr. Altair Olivo Santin PUCPR		aprov.
Prof. Dr. Frank Augusto Siqueira UFSC		aprov.

Conforme as normas regimentais do PPGIa e da PUCPR, o trabalho apresentado foi considerado – aprovado – (*aprovado/reprovado*), segundo avaliação da maioria dos membros desta Banca Examinadora. Este resultado está condicionado ao cumprimento integral das solicitações da Banca Examinadora registradas no Livro de Defesas do programa.

Prof. Dr. Alessandro Lameiras Koerich
Diretor do Programa de Pós-Graduação em Informática

Programa de Pós-Graduação em Informática
PPGIa
PUCPR

Rua Imaculada Conceição, 1155 Prado Velho CEP 80215 901 Curitiba Paraná Brasil
Tel.: (41) 3271 1669 Fax: (41) 3271 2121 www.ppgia.pucpr.br

A minha família...

Agradecimentos

Expresso aqui minha gratidão ao Prof. Dr. Carlos Alberto Maziero que, com sabedoria e dedicação mostrou-me caminhos, apontou-me erros, encorajou-me nos momentos difíceis, acreditou e minha capacidade, orientando e participando da jornada e realização deste trabalho.

Agradeço, com carinho, a minha família (pais, irmãos, esposa e filhos). Eles souberam compreender minhas ausências, meus silêncios e minhas angústias. Um obrigado especial a minha sogra que colaborou na revisão do texto desta dissertação.

Um agradecimento sincero aos verdadeiros amigos que sempre estiveram presentes para me ajudar, em especial ao Juan Andrés Mussini que foi um grande companheiro de estudo nas salas de aula e nos laboratórios de pesquisa.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviações

CA	*Certificate Authority*
CAN	*Content-Addressable Network*
COTS	*Commercial Off-The-Shelf*
DHT	*Distributed Hash Table*
DKIM	*Domains Keys Identified Mail*
DNS	*Domain Name Service*
DoS	*Denial of Service*
E-Mail	*Electronic Mail*
ERP	*Endpoint Routing Protocol*
HTL	*Hope-To-Live*
HTML	*HyperText Markup Language*
IMAP	*Internet Mail Access Protocol*
IP	*Internet Protocol*
JES	*Java E-mail Server*
MDA	*Mail Delivery Agent*
MIME	*Multi-purpose Internet Mail Extensions*
MTA	*Mail Transport Agent*
MUA	*Mail User Agent*
MX	*Mail eXchanger*
NAT	*Network Address Translation*
OpenNap	*Open Source Napster Server*
P2P	*Peer-to-Peer*
PBP	*Pipe Binding Protocol*
PDP	*Peer Discovery Protocol*
PEM	*Privacy Enhanced Mail*
PGP	*Pretty Good Privacy*
PIP	*Peer Information Protocol*
PKI	*Public Key Infrastructure*
POP3	*Post Office Protocol*
PRA	*Purported Responsible Address*
PRP	*Peer Resolver Protocol*
RBAC	*Role-Based Access Control*
RBL	*Realtime Blackhole List*
RFC	*Requests for Comments*
RVP	*RendezVous Protocol*

SIDF	*Sender ID Framework*
S/MIME	*Secure/Multipurpose Internet Mail Extensions*
SMTP	*Simple Mail Transfer Protocol*
SPF	*Sender Policy Framework*
UA	*User Agents*
UCE	*Unsolicited Commercial E-mail*
URL	*Uniform Resource Locator*
WWW	*World Wide Web*
XML	*eXtensible Markup Language*

Resumo

O sistema de correio eletrônico é um dos serviços mais utilizados na Internet. O sistema convencional é baseado no paradigma cliente/servidor: no servidor ficam as mensagens e os dados para autenticação. Esta centralização causa alguns problemas como ponto único de falha e a exigência de grande capacidade de armazenamento em disco. Além disso, o sistema de e-mail atual sofre com alguns problemas, os mais comuns são os *spams* e a propagação de vírus. Vários estudos já foram realizados para diminuir esses problemas. Este trabalho propõe uma nova arquitetura e-mail, criando um modelo descentralizado utilizando o conceito de sistemas *peer-to-peer*. Na arquitetura proposta não há um servidor central para armazenamento das mensagens e nem para dados de usuários. As mensagens são armazenadas no disco local de cada usuário e somente uma notificação é enviada para cada destinatário. Os usuários receptores podem escolher entre buscar as mensagens ou simplesmente ignorá-las. Caso o receptor faça a opção pela leitura, a busca é realizada diretamente da máquina do emissor. Uma das vantagens deste sistema é usar o disco local do usuário emissor para armazenar as mensagens de e-mail. Para demonstrar a arquitetura foi implementado um protótipo e realizados alguns testes para comparação com o sistema convencional atualmente em uso.

Palavras-chave: e-mail, *peer-to-peer*, sistemas distribuídos.

Abstract

Electronic mailing system is one of the most used Internet services. Nowadays, this system is based on the client/server paradigm: the server keeps the messages and the authentication data. This centralization causes some problems such as a unique failure point and the demand for larger storage capacity. Besides, the current e-mail system suffers some other problems, the most common are spams and virus dissemination. Several studies were made to reduce such problems. This work proposes a new email infra-structure, creating a decentralized model using the peer-to-peer systems concept. The proposed architecture has no central server for message storage or users data. Message are kept on each users' local disk and only a notification is sent to each recipient. Receiver users can choose between downloading the messages or simply ignore them. Should the receiver choose to download the message, the search is made directly from the issuer machine. One of the advantages of this system is the use of the issuer local disk to store the e-mails. To demonstrate the architecture a prototype has been developed and some tests were made, in order to compare it with the current convencional system.

Keywords: e-mail, peer-to-peer, distributed systems.

Capítulo 1

Introdução

1.1 Contexto

O serviço de correio eletrônico ou *Electronic Mail* (E-Mail) é um dos serviços mais antigos e mais utilizados atualmente na rede. A facilidade e a simplicidade do sistema fazem com que as pessoas utilizem o e-mail como ferramenta de comunicação pessoal e profissional. Os servidores de e-mail são baseados no modelo criado desde seu início, que mantém um servidor centralizado para o armazenamento de mensagens e dados dos usuários, e também serve para a autenticação dos usuários.

Apesar de ser um meio de comunicação muito utilizado, o protocolo de comunicação continua quase o mesmo desde seu início, por este motivo vem sofrendo com vários problemas decorrentes de sua simplicidade. Por existirem fragilidades na infra-estrutura de e-mail, usuários mal intencionados procuram possíveis falhas nos sistemas de e-mail para a prática de atos ilícitos, como envio de mensagens não-solicitadas (*spam*) ou mensagens com programas maliciosos (*vírus)* em anexo.

Para diminuir o problema de *spam* várias técnicas são implementadas, como filtros de conteúdos que procuram padrões para classificação das mensagens, autenticação em servidores de e-mail e uso de chaves de criptografia para cifrar e decifrar o conteúdo de uma mensagem. Todas essas implementações são sempre realizadas nos servidores receptores, causando assim tráfego de mensagens que pode ser desnecessário e produzindo uma demanda de processamento para execução das técnicas.

As redes *peer-to-peer* foram criadas, inicialmente, para compartilhamento de arquivos, porém, com a sua popularização, passaram a ter outras aplicações, inclusive o serviço de correio eletrônico. Usando este conceito de sistemas distribuídos, vários estudos foram realizados com o objetivo de resolver os problemas que ocorrem, principalmente, com a centralização.

1.2 Objetivo

Analisando algumas deficiências no sistema de e-mail, que é baseado em uma abordagem *PUSH* de envio e recepção de mensagens, este trabalho propõe uma infra-estrutura de e-mail descentralizada, baseada em uma abordagem *PULL*, ou seja, os clientes buscam as mensagens em que estão interessados.

1.2.1 Objetivos específicos

Os objetivos específicos deste trabalho são:

1. Elaborar uma arquitetura baseada em um modelo de transferência de e-mails usando uma abordagem *PULL*;
2. Construir uma arquitetura de e-mail descentralizada, sem um servidor central para armazenamento de mensagens e sem autenticação centralizada;
3. Avaliar o sistema proposto para verificar sua viabilidade.

1.2.2 Justificativas

A utilização da abordagem *PULL* permite que o usuário leia (faça o *download* das mensagens) somente as mensagens escolhidas por ele, evitando assim, que todas as mensagens trafeguem pela rede. A utilização de algumas características dos sistemas P2P permite ao serviço de e-mail, a descentralização do armazenamento e dos dados dos usuários. O sistema de e-mail convencional sempre envia as mensagens de um servidor remetente para o servidor destinatário, ou seja, o sistema funciona utilizando uma abordagem *PUSH* no qual os remetentes "empurram" os e-mails aos destinatários.

Na arquitetura proposta, que é baseada em uma arquitetura P2P, somente é publicada uma notificação, isto faz com que as mensagens trafeguem pela rede somente após a decisão do destinatário. Com isso haverá uma economia de banda de rede e, com a utilização do disco local para armazenamento, não há a necessidade de servidor intermediário. Como somente uma notificação é recebida, em um primeiro momento, o usuário pode ignorar uma mensagem caso haja suspeita desta ser *spam* ou ter conteúdo malicioso. Assim pode-se diminuir o tráfego de mensagens não solicitadas. A segurança deste sistema faz o uso do PGP, onde cada par de chaves é associado a um endereço de e-mail.

1.3 Estrutura do trabalho

O trabalho está dividido e organizado na seguinte forma: o capítulo 2 apresenta um estudo sobre os sistemas de e-mail, mostrando suas características, protocolos utilizados e seus problemas; o capítulo 3 trata dos sistemas de autenticação no serviço e-mail para uma possível diminuição dos problemas; o capítulo 4 descreve os conceitos fundamentais e principais características das redes *peer-to-peer*; o capítulo 5 apresenta a proposta, descrevendo as características da infra-estrutura; o capítulo 7 mostra os detalhes da implementação e os resultados de alguns testes realizados; finalmente, o capítulo 8 traz uma conclusão sobre a infra-estrutura, analisando alguns benefícios e trabalhos futuros a serem realizados para a melhoria da arquitetura.

Capítulo 2

Sistemas de correio eletrônico

2.1 Introdução

O sistema de e-mail é muito utilizado pela sua facilidade e simplicidade. Estas características fazem com que o e-mail se torne um dos serviços de comunicação mais utilizados na Internet. O sistema de e-mail convencional é baseado na arquitetura cliente/servidor. No **servidor** ficam as mensagens e os dados dos usuários que são utilizados para sua autenticação. Os usuários utilizam uma interface (**cliente**) para criar, enviar e receber novas mensagens.

Para o envio de mensagens entre os servidores, o protocolo utilizado é o *Simple Mail Transfer Protocol* (SMTP) [Klensin, 2001] e para um cliente ler as mensagens podem ser usados os protocolos *Internet Mail Access Protocol* (IMAP) [Crispin, 2003] ou *Post Office Protocol* (POP3) [Myers and Rose, 2001]. Este capítulo apresenta a infra-estrutura de e-mail convencional, explorando suas principais características e os problemas mais relevantes presentes na mesma.

2.2 Infra-estrutura

O sistema de e-mail convencional é composto por diversos agentes e protocolos que são definidos por várias *Requests for Comments* (RFC). A infra-estrutura de transporte de e-mail definida pela RFC 2821 [Klensin, 2001] é constituída por agentes que funcionam como emissores ou receptores de mensagens, podendo um e-mail ser enviado, recebido ou encaminhado. O modelo definido pela RFC 2821 apresenta uma estrutura ilustrada na figura 2.1.

Os agentes básicos para o funcionamento são o *Mail User Agent* (MUA), que é o programa utilizado pelo usuário para enviar e receber mensagens, e o *Mail Transport Agent* (MTA), que é o agente responsável pelo envio das mensagens recebidas do MUA. O MTA também é responsável por comunicar-se com outros MTAs para o envio das mensagens. Os sistemas de e-mails implementam também o agente de entrega de e-mail conhecido como *Mail Delivery Agent* (MDA) que são aplicativos responsáveis por receber uma mensagem de um MTA e fazer com que esta mensagem seja recebida pelo sistema local. O *procmail* (`www.procmail.org`) é muito utilizado como um MDA em sistemas *UNIX*.

2.2.1 Principais Protocolos

Para o envio e busca de mensagens podem ser utilizados os seguintes protocolos:

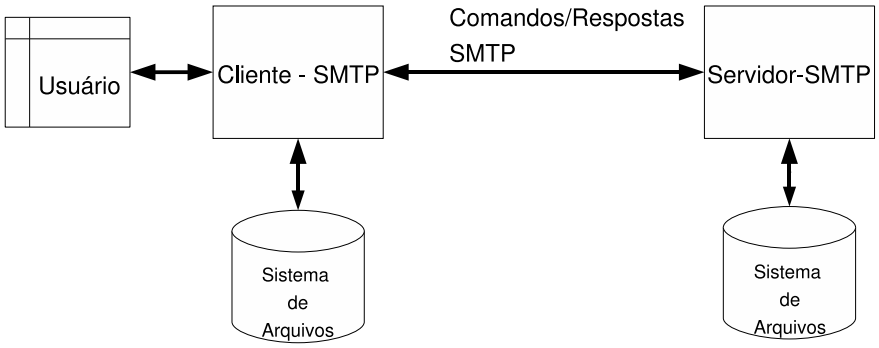

Figura 2.1: Modelo SMTP [Klensin, 2001]

SMTP

Definido pela RFC 2821 [Klensin, 2001] é responsável pela troca de mensagens entre os servidores de mensagens (MTA) e utilizado pelos usuários para enviar mensagens, isto é, a conexão entre o MUA e o servidor de envio é feita pelo protocolo SMTP. A porta que serve para a comunicação é a 25. A comunicação SMTP com um cliente, para proteger as credenciais para autenticação, pode ser feita de modo seguro, realizada sobre TLS [Hoffman, 1999], através da porta 465.

Os **comandos** SMTP são constituídos por uma linha de texto iniciada por uma "palavra especial" de quatro letras, que poderá ser seguida de argumentos. Na maioria dos casos as respostas a comandos são constituídas de uma única linha, contudo, múltiplas linhas de resposta também são possíveis.

A Tabela 2.1 apresenta os principais comandos que devem ser implementados pelo SMTP.

Comando	**Descrição**
HELO <SP> <domain> <CRLF>	Identificador do Envio
MAIL <SP> FROM:<reverse-path> <CRLF>	Identifica o criador do e-mail
RCPT <SP> TO:<forward-path> <CRLF>	Indentifica o destinatário do e-mail
DATA <CRLF>	Transferência de mensagem texto
RSET <CRLF>	Aborta a transação de e-mail atual
NOOP <CRLF>	Sem operação
QUIT <CRLF>	Fecha a conexão TCP

Tabela 2.1: Comandos SMTP

As **respostas** SMTP são formadas de três digitos seguidos por uma informação adicional, onde o primeiro digito indica a categoria da resposta; os demais dígitos indicam o erro em si. Essas respostas são classificadas como:

- **Resposta de conclusão positiva:** ações de requisições foram completadas com sucesso.
  - **211:** status do sistema ou resposta de ajuda do sistema

  - **214:** mensagem de ajuda
  - **220:** <domain> serviço pronto
  - **221:** <domain> fechando serviço de canal de transmissão
  - **250:** requisição de ação de e-mail ***OK***, completada
  - **251:** usuário não faz parte do domínio; a mensagem será reencaminhada para <forward-path>

- **Resposta intermediária positiva:** o comando foi aceito, mas estão faltando informações para a ação requisitada. O **emissor** deve enviar um outro comando especificando esta informação. Esta resposta é utilizada quando é necessária uma sequência de comandos.

  - **301:** inicio do envio do e-mail; finalizar com <CRLF>.<CRLF>

- **Resposta de conclusão temporariamente negativa:** o comando não foi aceito e a ação requisitada não ocorreu. Contudo, a condição de erro é temporária e pode ser requisitada novamente.

  - **421:** <domain> serviço indisponível: perdendo canal de transmissão
  - **450:** requisição negada: caixa de correio indisponível
  - **451:** requisição abortada: erro local no processo
  - **452:** requisição negada: espaço insuficiente

- **Resposta de conclusão permanentemente negativa:** o comando não foi aceito e a requisição não poderá ser feita novamente.

  - **500:** erro de sintaxe, comando não reconhecido
  - **501:** erro de sintaxe nos parâmetros ou argumentos
  - **502:** comando não implementado
  - **503:** sequência errada de comandos
  - **504:** parâmetro não implementado
  - **550:** requisição negada: caixa de correio indisponível
  - **551:** usuário não faz parte do domínio; tente <forward-path>
  - **552:** requisição de e-mail abortada: excedido espaço de alocação
  - **553:** requisição negada: caixa de correio indisponível
  - **554:** erro na transação

Os comandos e respostas são gerados e trocados entre servidores e clientes de e-mail até que a mensagem seja entregue ao seu destino final.

POP3

A versão 3 é definida pela RFC 1939 [Myers and Rose, 2001] e empregada pelo MUA para acessar as mensagens armazenadas no servidor. Utiliza a porta 110 e no modo TLS/SSL a porta 995, que é definida pela RFC 2595 [Newman, 1999]. Considera apenas uma pasta no servidor e, por padrão, descarrega as mensagens do servidor no computador do cliente.

O POP3 tem como uma das facilidades, o acesso às mensagens quando se está desconectado da rede, pois neste sistema o cliente faz o *download* das mensagens e estas são armazenadas em seu disco local.

Alguns **comandos** do protocolo POP3:

Principais comandos do POP3	
USER $nome$	Identificador do cliente
PASS $senha$	Senha para acesso
DELE msg	Elimina a mensagem identificada pelo némero msg
LIST [msg]	Lista assunto do e-mail de número msg
RETR msg	Busca no servidor o conteúdo do e-mail de número msg
RSET	Retira marcas de eliminação de e-mail
STAT	Indica o número de e-mail destinados ao usuário e o espaço usado
QUIT	Muda para atualização e o servidor elimina todos os e-mails marcados

Tabela 2.2: Comandos POP3

As **respostas** para os comandos enviados pelo clientes podem ser:

- +OK mensagem
- -ERR

Nos comandos, como LIST e RETR, a informação solicitada termina com uma linha formada pelo par ponto e $CRLF.$.

IMAP

A versão 4-rev1 é definida pela RFC 3501 [Crispin, 2003]. Trata-se de um método de acesso a mensagens armazenadas em um servidor local ou remoto. Usa normalmente a porta 143 e no modo TLS/SSL (RFC 2595 [Newman, 1999]) a porta 993. O usuário pode manipular suas mensagens e pastas a partir de computadores diferentes em diversas localidades sem que seja necessária a transferência das mesmas do servidor para o computador. Assim, as mensagens podem ser acessadas de um computador portátil durante uma viagem, no micro de casa ou do trabalho. O protocolo IMAP também permite o acesso das mensagens armazenadas localmente, desde que o usuário tenha feito o sincronismo das mensagens armazenadas no servidor com o seu disco local.

Algumas características foram comparadas [Gray, 1995] entre os protocolos POP e IMAP:

- Características comuns aos dois protocolos:

  - Ambos podem operar em modo *offline*;
  - Mensagens novas podem ser lidas a partir de vários clientes em diferentes plataformas seja *Unix*, *MacOS*, entre outros;
  - Mensagens podem ser lidas de qualquer lugar da rede;
  - Os protocolos são abertos e definidos pelas RFCs;
  - As implementações são livres, qualquer pessoa pode desenvolver um servidor POP ou IMAP;
  - Esses protocolos só servem para leitura, para envio dependem do protocolo SMTP;

- Vantagens do POP3:
  - Protocolo é simplificado e com isso é mais fácil de implementar;
  - Mais clientes estão disponíveis atualmente.

- Vantagens do IMAP:
  - Pode alterar o estado (*flag*) das mensagens;
  - Pode armazenar mensagem, bem como fazer sua busca;
  - Pode acessar e gerenciar múltiplas caixas postais;
  - Suporta atualizações e acessam caixas postais compartilhadas;
  - Construído para permitir o acesso *online*, especialmente em banda de rede com velocidade reduzida.

Além destes, outros protocolos podem ser usados, como o HTTP (em sistemas de *webmail*), *WebDAV* ou protocolos proprietários em sistemas de correio eletrônico corporativo com o *Lotus Notes* (`www-306.ibm.com/software/lotus`) e *Microsoft Exchange* (`www.microsoft.com/exchange`).

2.2.2 Funcionamento

Após o usuário escrever sua mensagem em um cliente de e-mail (também denominado MUA), os dados são formatados no padrão RFC 2822 [Resnick, 2001] e em seguida enviados ao MTA que usa o protocolo SMTP para fazer a comunicação com o próximo servidor. Para prosseguir a transmissão faz-se uma consulta na entrada *Mail eXchanger* (MX) no *Domain Name Service* (DNS) do domínio, e após ter recebido o endereço *Internet Protocol* (IP) do servidor destino, estabelece-se a comunicação com este servidor e faz-se a entrega da mensagem. Um domínio pode ter várias entradas MX que servem para balanceamento de carga ou apenas como redundância.

A figura 2.2 detalha a seqüência para envio de uma mensagem. Após o MUA remetente (1) escrever o e-mail, envia o mesmo para o MTA que responde pelo seu domínio (2). Esta mensagem pode ser encaminhada por uma seqüência de um ou mais MTAs. A comunicação entre os MTAs é feita através do protocolo SMTP. Quando o servidor MTA destinatário recebe a mensagem (3), o serviço de entrega *Mail Delivery Agent* (MDA) envia a mensagem para a caixa de entrada do receptor. Para a leitura da mensagem, o MUA receptor (4) pode se comunicar através do protocolo POP3 ou IMAP.

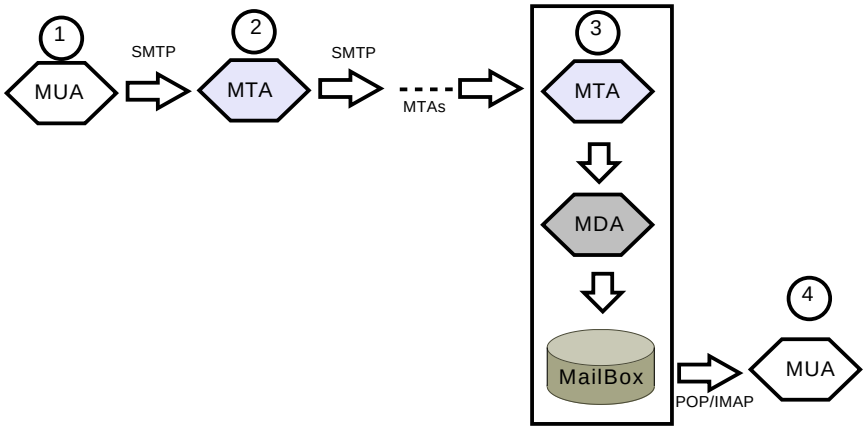

Figura 2.2: Fluxo do E-mail no Modelo SMTP

2.3 Formato de uma mensagem

A seguir é mostrado o código fonte que forma uma mensagem. As linhas 1 a 13 formam o cabeçalho, onde ficam alguns campos como *From*, *To*, *Subject* que são visíveis em um MUA. A linha 14, que é uma linha em branco, separa o cabeçalho do corpo da mensagem. As linhas 15 a 18 fazem parte do corpo de uma mensagem.

```
1  From - Thu Dec 13 18:12:25 2007
2  X-DeliveredTo: edson@three.p2pmail.com
3  X-RecievedDate: Thu Dec 13 18:12:18 BRST 2007
4  Received: by EricDaugherty's JES SMTP two.p2pmail.com from client:
       127.0.0.1
5  Message-ID: <47619222.8010909@two.p2pmail.com>
6  Date: Thu, 13 Dec 2007 18:12:18 -0200
7  From: edson <edson@two.p2pmail.com>
8  User-Agent: Thunderbird 2.0.0.6 (X11/20071022)
9  MIME-Version: 1.0
10 To: edson@three.p2pmail.com
11 Subject: Ola
12 Content-Type: text/plain; charset=ISO-8859-1; format=flowed
13 Content-Transfer-Encoding: 7bit
14
15 Ola,
16 Tchau
17
18 User2
```

2.4 MIME

O padrão *Multi-purpose Internet Mail Extensions* (MIME) foi proposto para suprir a necessidade de enviar arquivos anexados junto a mensagens de e-mail. São definidos pelas RFC 2045 [Freed and Borenstein, 1996a], RFC 2046 [Freed and Borenstein, 1996b], RFC 2047 [Moore, 1996], RFC 2048 [Freed, 1996b] e RFC 2049 [Freed, 1996a]. O padrão MIME codi-

fica os arquivos anexados a uma mensagem, formatando-as em modo texto e inserindo-os na mensagem. Com essa abordagem não foram necessárias alterações no protocolo SMTP.

Foram definidas novas variáveis de cabeçalhos, que podem ser inclusas no cabeçalho de mensagens. Estes campos possuem informações sobre o corpo da mensagem, conforme descrito:

- *MIME-Version*: informa que a mensagem tem o formato MIME;
- *Content-ID*: identifica o conteúdo do corpo da mensagem;
- *Content-Type*: especifica o tipo de arquivo ou subtipo de dados incluído na mensagem;
- *Content-Transfer-Encoding*: define vários métodos para a representação de dados binários em formato texto ASCII;
- *Content-Description*: descreve o conteúdo do corpo da mensagem, decifrando quando o objeto não pode ser lido, como por exemplo, arquivo de música.

Também são definidas as codificações de conteúdo de transferência (*Content-Transfer-Encoding*) onde são apresentados vários métodos para conversão de um dado binário em ASCII. Podem ser:

- *7bit*: indica que o texto é codificado em 7 bits (conjunto de caracteres US-ASCII);
- *8bit*: mostra que a mensagem contém texto com alguns caracteres que necessitam de 8 bits para serem codificados (conjunto de caracteres não US-ASCII). Caso uma mensagem deste tipo passe por uma zona da rede que permite transportar somente caracteres de 7 bits, todos os caracteres que necessitam de 8 bits chegarão ao destino com erro;
- *binary*: refere-se a dados em que qualquer seqüência de octetos é permitida;
- *quoted-printable*: codifica textos simples com caracteres US-ASCII, transformando, por exemplo, a palavra *Avião* para *Avi=E3o*;
- *base64*: demonstra a existência de binários codificados. Todos os caracteres são codificados como grupos de caracteres de 7 bits, de tal modo que o binário não será alterado ao trafegar pela rede.

Através do padrão MIME, um usuário pode incluir arquivos em um e-mail de diferentes formatos com diferentes codificações. Grande parte do sucesso do padrão MIME deve-se ao fato do mesmo ser transparente aos servidores de e-mail, pois o corpo de cada e-mail é visualizado em formato ASCII, conforme a definição da RFC 2822. Todo o processamento de codificação e decodificação das mensagens de e-mail é realizado pelos programas clientes (MUA), durante o envio e a recepção. Um exemplo de mensagem com arquivo anexo, pode ser visualizado a seguir:

```
1  From - Wed Dec 12 20:53:17 2007
2  X-DeliveredTo: edson@three.p2pmail.com
3  X-RecievedDate: Wed Dec 12 20:50:04 BRST 2007
4  Received: by EricDaugherty's JES SMTP p2pmail.two from client: 127.0.0.1
5  Message-ID: <4760659C.20104@two.p2pmail.com>
6  Date: Wed, 12 Dec 2007 20:50:04 -0200
7  From: edson <edson@two.p2pmail.com>
8  User-Agent: Thunderbird 2.0.0.6 (X11/20071022)
9  MIME-Version: 1.0
10 To: edson@three.p2pmail.com
11 Subject: Teste de envio com anexo =?ISO-8859-1?Q?bin=E1rio?=
12 Content-Type: multipart/mixed;
13  boundary="------------090607080100080908090907"
14
15 This is a multi-part message in MIME format.
16 --------------090607080100080908090907
17 Content-Type: text/plain; charset=ISO-8859-1; format=flowed
18 Content-Transfer-Encoding: 8bit
19
20 Teste de envio com anexo binario.
21 <>s;
22 User
23
24 --------------090607080100080908090907
25 Content-Type: chemical/x-mopac-input;
26  name="msg-bin-1k.dat"
27 Content-Transfer-Encoding: base64
28 Content-Disposition: inline;
29  filename="msg-bin-1k.dat"
30
31 FMDg5y9fes5+czySUX2SRdHj/M8pzqwdwdTD7I5er4ehmMd8ZNvkeZ8Qu9vjML6SI6UQ38Jc
32 0PAM50HA7HDk4Issf5TJ/Pw8c/y63S6mrcnJSWYiucogNIotNeFkHA3AM9stMMm9+Lz9dbQ0
33 yStDyyHZBJqNsJF2jIxKxxjKB4XGTC22hMjVCoIV5JlchNYbKriUd0iNmfHo8u58dCSkkYp0
34 20iyH8oNJQx3yBYaEWNCrPMm6wE1UZxkB4OsOb0jEbczXa2MOKbt0XeN8J6jZpmHiW9h9M6X
35 OX6LWeQ5JDEYScJdR25n4Ye4neX4Zi5bXFy0fXL56XwGNx7Pi6fGs6Ox0f+nhgceYWJigjnC
36 tl57FSShBeEjFxQ3W/SCfU1FLxRF4DP7kK7XeakhQgUwykU9nlUpwITojC6ryH0KhQBdSxuM
37 2Fjwxr+LSdxZwp3Tx6u4bSfCMeF9r7De3vVHEp8uBSSQzDH3BaZAckdb5QOPDj2FrgNzh+gX
38 9tFYzxhfSwhNYPwu2wtCEy3Iz0NSyvh5EIM8uTVKqbmImzfSAcDpwDjm2ZL7eDz47quIYBuK
39 LtExrm0ARKvwDBSEoSrr3AyhCITPWTCP1CMlRskWoTgLzCH7hDFWARotXzCSJ35ZjKIkko27
40 wWfWjnROLbhq/HtsffHw0YPP+7vTkZEBayjcPIOrGX9ut6LvvvuWwcM+ZRUxBh6gn+JJ8Q==
41 --------------090607080100080908090907--
```

Analisando este exemplo, é possível verificar a inclusão de novas variáveis de cabeçalho na mensagem e informações adicionais no corpo da mensagem que informam o tipo de codificação (linhas 25 a 29), o nome do arquivo anexado e outras informações pertencentes ao padrão MIME. O arquivo codificado está entre as linhas 24 e 41.

2.5 Avaliação do sistema de e-mail

A estrutura de e-mail convencional atualmente em uso, descrita na seção anterior, apresenta algumas deficiências apontadas por vários autores [Bouwsma and Visser, 2004, Kangasharju et al., 2003]. Estes problemas são relacionados a algumas características, e são elas:

- **Falta de confidencialidade**: as mensagens trafegam na rede em aberto (sem criptografia), de forma que qualquer pessoa que recebe pode ler a mensagem, isto é, ainda não utiliza a cifragem de forma efetiva.
- **Fragilidade de autenticação**: no caso dos protocolos POP3 e IMAP, para ler as mensagens necessita-se de um usuário e senha para autenticação, ou seja, há a necessidade de armazenamento destas informações em algum repositório.
- **Possibilidade de repúdio**: é possível enviar mensagens a determinadas pessoas como emitidas por outra pessoa e também podem-se enviar mensagens com remetentes falsos.
- **Baixa confiabilidade**: as mensagens são armazenadas em um servidor central, e caso este fique indisponível, os usuários não poderão ler as mensagens.
- **Tratamento ineficiente de anexos e vários destinatários**: quando uma mensagem é enviada a vários usuários, a mensagem é destinada individualmente a cada receptor, causando tráfego de rede desnecessário e podendo resultar em mensagens duplicadas no disco do servidor.

2.6 Técnicas que exploram limitações do sistema de e-mail

2.6.1 *Spam*

Spam [Hambridge and Lunde, 1999] é o termo usado para se referir aos e-mails não-solicitados, que geralmente são enviados para um grande número de pessoas. Quando o conteúdo é exclusivamente comercial, este tipo de mensagem também é referenciado como *Unsolicited Commercial E-mail* (UCE). Mensagens não solicitadas representam a maior parte do tráfego de mensagens atuais, que além de propagandas, enviam muitas mensagens maliciosas que levam os usuários a acessarem páginas falsas em que solicitam dados confidenciais, incluindo senhas, informações bancárias, etc. Esta técnica é mais conhecida como *phishing scam* (Seção 2.6.2). Os *spams* têm causado muitos prejuízos para empresas e provedores devido ao alto tráfego na rede e custo de armazenamento, provocando atraso e, às vezes, indisponibilidade do sistema. Uma entidade que envia mensagens de *spams* normalmente é conhecida como *spammer*.

2.6.2 *Phishing Scam*

O *scam* ou *phishing scam* é um subconjunto de *spam* que faz uso de engenharia social em conjunto com fragilidades dos sistemas de e-mails, visando enganar os destinatários, convencendo-os a informar dados sigilosos como informações bancárias, números dos cartões de crédito e outras informações confidenciais ou instalar *spywares*[1], que é o objetivo mais freqüente.

[1]Programa que se auto-instala no sistema, de forma camuflada e sem o consentimento do usuário, para coletar informações locais (como senhas) e enviar a uma entidade externa na Internet

2.6.3 Vírus de computador

Vírus de computador [Cohen, 1987] é um programa que se propaga inserindo cópias de si mesmo e se tornando parte de outros programas e arquivos de um computador. Uma das formas mais usadas para propagação é através do sistema de e-mail. Um vírus propagado por e-mail, normalmente é recebido como um arquivo em anexo e o conteúdo dessa mensagem procura induzir o usuário a clicar sobre o arquivo para executar o vírus, podendo causar danos no computador e enviar cópias de si mesmo pelo sistema de e-mail.

2.6.4 *Worm*

Um *worm* é semelhante a um vírus, cria cópias de si mesmo de um computador para outro, mas ao invés de ser iniciado por um usuário, faz isto de forma automática. Normalmente, no primeiro passo, ele tenta realizar o controle dos recursos de um computador que permitem a replicação de arquivos ou de informações, em seguida ele contamina o sistema fazendo o deslocamento de um computador para outro automaticamente. A propagação ocorre pela rede decorrente, em geral, de uma falha em um aplicativo, no sistema operacional ou no serviço de e-mail.

2.7 Técnicas de combate ao *Spam*

Para tentar minimizar os problemas com *spams* algumas recomendações foram descritas na RFC 2505 [Lindberg, 1999], e têm como objetivo a redução dos *spams*, mas também há orientações para não utilizar algumas técnicas que podem impedir que mensagens lícitas cheguem ao destino. Algumas delas:

- deve restringir o uso não autorizado do servidor como *relay*, ou seja, o servidor não deve aceitar conexões de qualquer usuário da Internet para envio de e-mails através dele;
- deve manter o domínio registrado no DNS do provedor, cadastrando o *reverso* e o MX;
- deve gerar a linha do cabeçalho *received:* com informação suficiente para permitir o rastreamento do caminho utilizado para a distribuição de uma mensagem;
- gerar registros locais para um possível rastreamento de uma mensagem;
- possibilitar a geração de registros de todas as ocorrências geradas pelo *anti-spam*;
- não deve recusar mensagens com endereços vazios, isto é, com o campo no formato *Mail From: <>*;
- deve ser capaz de rejeitar e-mails de um determinado servidor ou de um grupo de servidores, baseado em listas de IP ou nomes autorizados e não autorizados;
- deve ser capaz de limitar a taxa de envio e recebimento de e-mails.

Outras técnicas como filtros também são utilizadas, algumas delas são:

Listas Negras (*Black Lists*). Servidores *Realtime Blackhole List* (RBL) distribuídos na Internet mantêm listas de endereços IP de geradores ou propagadores de *spams*, que podem ser consultadas por meio de DNS pelos servidores de e-mail para verificar a confiabilidade de um remetente; essas listas são mantidas por meio de vários mecanismos, como contribuições de usuários ou resultados de varreduras automáticas.

Listas Brancas (*White Lists*). Cada servidor de e-mail pode manter uma lista de remetentes nos quais confia; essa lista é normalmente mantida por meio de pedidos de confirmação de envio que remetem a um formulário em *HyperText Markup Language* (HTML) ou outra abordagem similar [Hall, 1998]. Alguns sistemas alimentam suas listas brancas com outras técnicas, como forçar o remetente a tentar o mesmo envio várias vezes como no caso do sistema *Greylist* [Harris, 2003].

Filtros *anti-spam*. Programas de filtragem que classificam os e-mails de acordo com seu conteúdo; podem ser usadas técnicas estatísticas, de classificação *bayesiana* [Sahami et al., 1998], redes neurais, entre outras. Muitos *spammers* tentam burlar tais filtros por meio de técnicas que permitem inserir textos em imagens e usar estruturas HTML para confundir os filtros.

A RFC 2505 recomenda cuidado no uso dos filtros, pois as regras podem causar atrasos ou indisponibilidade do serviço gerados por grande quantidade de mensagens, normalmente executadas por técnicas de *Denial of Service* (DoS)[2].

2.8 Conclusão

Neste capítulo foi apresentada uma visão geral sobre o sistema de e-mail, detalhando sua arquitetura e seus principais componentes e protocolos usados em sua construção. Também foram estudados os principais problemas apresentados pelos sistemas de e-mail, que podem comprometer seu funcionamento e segurança. Diversas técnicas podem ser usadas para melhorar a segurança dos sistemas de e-mail, algumas das quais foram expostas neste capítulo. Na seqüência, destacam-se as principais técnicas usadas para a autenticação de mensagens em sistemas de e-mail. Estas técnicas são úteis para identificar os remetentes das mensagens e também na verificação do conteúdo transmitido, e assim combater *spams* e *scams*. A seguir, no próximo capítulo, serão apresentados alguns métodos de autenticação utilizados em clientes e servidores.

[2]Técnica utilizada para indisponibilizar um serviço através de envio de grande quantidade de requisições simultaneamente a um servidor.

Capítulo 3

Autenticação em Sistemas de E-mail

3.1 Introdução

A autenticação nos serviços de e-mail pode ser implementada nos clientes e/ou nos servidores para validar os usuários dos sistemas de mensagens. Autenticar os usuários é um dos passos para minimizar problemas como os *spams*, *phishing scams* e propagação de programas maliciosos. As principais propostas de autenticação de remetentes são PEM [Linn, 1993, Kent, 1993, Balenson, 1993, Kaliski, 1993], S/MIME [B. Ramsdell, 2004a, B. Ramsdell, 2004b], SPF [Wong and Schlitt, 2006], PGP [Zimmermann, 1995], SIDF [Lyon and Wong, 2006] e *DKIM* [Allman et al., 2007].

PEM, S/MIME e o PGP são ferramentas implantadas em computadores de usuários de e-mail, sendo transparentes aos servidores de e-mail, para assinar e cifrar arquivos e/ou documentos a serem enviados pela rede. Essas ferramentas garantem a privacidade, integridade e autenticidade de mensagens, valendo-se de sistemas baseados em chaves assimétricas para cifrar e assinar mensagens.

O SPF cria uma infra-estrutura de verificação de autenticidade entre servidores de e-mail, transparente aos clientes. O DKIM utiliza uma infra-estrutura de chaves públicas/privadas para validar a autenticidade de um servidor de e-mail. O SIDF é uma implementação baseada no SPF que visa resolver problemas de autenticação de mensagens encaminhadas.

3.2 O Padrão X.509

O padrão X.509 [Adams and S.Farrell, 1999] é um padrão para infra-estrutura de chave pública (*Public Key Infrastructure* (PKI)) que especifica o formato dos certificados digitais. O X.509 utiliza **autoridades certificadoras** ou *Certificate Autority* (CA) para gerar e validar certificados digitais na rede. O modelo X.509 é baseado em uma estrutura em árvore hierárquica cuja raiz representa o principal fornecedor de certificado chamado de CA raiz.

O padrão X.509 garante a confiança de um certificado a partir da aceitação do certificado da CA raiz como confiável. Por causa da estrutura de árvore do X.509, um usuário deverá confiar automaticamente em todos os certificados criados pela CA. Nos aplicativos dos serviços mais conhecidos na Internet como *World Wide Web* (WWW) e clientes de e-mail, os serviços de certificado digital já são implementados, e algumas das CA mais conhecidas para validar um certificado já vêm instaladas e configuradas.

3.3 PEM

Privacy Enhanced Mail (PEM) é um padrão da *Internet* para sistemas de e-mail que emprega técnicas de criptografia para garantir a privacidade e a segurança das mensagens. A RFC 1421 [Linn, 1993] define procedimentos para criptografia e autenticação de e-mails, utilizando criptografia simétrica e assimétrica. A RFC 1422 [Kent, 1993] especifica suporte a mecanismos de gerenciamento de chaves baseado em criptografia de chave pública. A RFC 1423 [Balenson, 1993] propõe os algoritmos de criptografia usados nas RFC 1421 e RFC 1422 e a RFC 1424 [Kaliski, 1993] propõe os procedimentos para o gerenciamento de chaves e infraestrutura para o suporte a esses serviços.

As chaves públicas são assinadas por uma Autoridade Central e deve ser criada uma certificação, e somente as chaves oficialmente assinadas podem ser usadas. Um certificado é uma estrutura de dados que contém o nome do usuário, a chave pública e o nome da entidade organizacional à qual o usuário pertence. A CA age como a raiz da hierarquia de certificados para a rede.

Com uma possível necessidade de pagamento pelo uso do recurso para o funcionamento e para que todas as chaves sejam certificadas, demanda em custos financeiros e de administração do sistema. A complexidade imposta pela certificação de chaves, sem dúvida, torna mais difícil a possibilidade de acesso ao sistema por usuários não autorizados. Entretanto, é também mais difícil de implementar e de utilizar; o esquema de distribuição de chaves do PEM é tão complexo e demanda tanto esforço para ser instalado, que até o presente momento não está operacional. Além disso, o PEM não provê nenhum mecanismo para evitar o acesso indevido às mensagens armazenadas nos servidores de e-mail (sejam elas mensagens recebidas ou por enviar).

3.4 S/MIME

O *Secure/Multipurpose Internet Mail Extensions* (S/MIME) [B. Ramsdell, 2004a, B. Ramsdell, 2004b] é um modelo que possibilita enviar mensagens através do sistema de e-mail de uma maneira segura. Clientes de e-mail que tenham suporte do S/MIME são capazes de adicionar facilmente assinaturas digitais em mensagens. O S/MIME foi baseado na especificação MIME. O S/MIME disponibiliza um serviço de cifragem que possui algoritmos de criptografia especificados pelo usuário; autenticação por meio de chaves públicas/privadas X.509 e o não repúdio por meio de mensagens assinadas criptograficamente. Com o S/MIME é possível:

- cifrar uma mensagem, quando interessa somente sigilo ou confidencialidade;
- assinar uma mensagem, quando interessa somente autenticidade e integridade;
- cifrar e assinar uma mensagem, quando interessa confidencialidade, autenticidade e integridade.

A confidencialidade das mensagens que utilizam o S/MIME é garantida pelo uso de algoritmos de criptografia simétricos. O S/MIME utiliza assinaturas digitais que são certificadas por CAs a partir do padrão X.509. A grande falha no X.509 é que se de alguma forma a raiz CA for comprometida, nenhum dos certificados gerados poderá ser considerado confiável. Além disso, a estrutura de árvore não é muito flexível.

3.5 PGP

O *Pretty Good Privacy* (PGP), descrito em [Zimmermann, 1995], é uma ferramenta de criptografia e certificação digital que utiliza criptografia assimétrica (chaves pública e privada) para cifrar/decifrar e assinar mensagens que trafegam pela rede. Esta ferramenta foi desenvolvida como alternativa ao X.509 que possui falha em sua estrutura hierárquica. Ao contrário do X.509, o PGP tem um modelo de confiança similar aos sistemas *peer-to-peer* (Seção 4), descentralizando os certificados digitais. Isto significa que cada usuário é sua própria CA raiz. A confiança é expressa através da assinatura da chave pública de um usuário com a chave assimétrica correspondente à chave pública ou por inferência de confiança através de uma base de chaves local.

O PGP usa chaves de sessão que são criadas a partir de um número gerado aleatoriamente a cada cifragem. As chaves de sessão são destinadas para cifrar os dados, utilizando algoritmos de criptografia simétricos. A chave de sessão é cifrada através da chave pública do destinatário. Ao receber uma mensagem cifrada, o destinatário utiliza a chave privada para recuperar a chave de sessão e decifrar o texto. O motivo pelo qual os autores optaram por chaves de sessão se dá pelo fato de os algoritmos de criptografia simétrica serem muito mais rápidos (tanto na cifragem quanto na decifragem) que os algoritmos assimétricos. Mesmo que o dado tenha sido gerado por uma chave não segura, esta chave é cifrada pela chave pública do usuário e enviada pela rede, garantindo a privacidade, velocidade de decifragem e segurança da informação.

As assinaturas digitais são geradas de maneira similar à cifragem dos dados. A diferença é que o PGP usa a chave privada para gerar a assinatura digital. Para comprovar a autenticidade da assinatura, um usuário deverá usar a chave pública do possível emissor para decifrar a assinatura. Se a assinatura for corretamente decifrada, isto indica que a chave foi gerada pelo seu par (privada/pública). Do contrário, a assinatura foi gerada por um emissor malicioso. As abordagens baseadas em P2P são mais flexíveis que a abordagem hierárquica X.509.

3.6 SPF

O *Sender Policy Framework* (SPF) [Wong and Schlitt, 2006] é uma especificação de autenticação de servidores de e-mail que visa diminuir o tráfego de mensagens não-solicitadas em servidores de e-mail (SMTP), através de políticas e filtros em um ambiente que funciona juntamente com servidores DNS.

As regras dos filtros e políticas do SPF são definidas a partir de uma linguagem simples que descreve o domínio de um determinado servidor de e-mail. Com essas informações, servidores serão capazes de avaliar se uma mensagem foi originada por um determinado domínio. Caso seja comprovada uma fraude de domínio, os servidores de e-mail serão capazes de descartar, analisar ou então manipular o e-mail fraudulento de acordo com as políticas determinadas nos serviços instalados nos servidores de e-mail.

Servidores baseados na especificação SPF deverão possuir atributos únicos que serão utilizados por outros servidores de e-mail para validar a autenticidade de mensagens. Esses atributos fazem parte de um registro SPF que permitirá aos servidores de e-mail destinatários verificar a origem de uma mensagem, descartando as mensagens com remetentes falsos.

O armazenamento dos registros disponibilizados para consulta dos servidores de e-mail é realizado no campo *TXT* de uma entrada DNS de um determinado domínio que envia men-

sagens de e-mail. Os registros DNS armazenam informações sobre o domínio de e-mail como endereços IP, servidores MX, IP de servidores MX, que indicam quais entidades possuem permissão para utilizar o nome de um domínio. Os registros SPF são publicados em servidores DNS de seu domínio, que deverão ser consultados toda vez que existir a necessidade de validação da autenticidade de uma mensagem. Os campos de um registro SPF são: versão SPF, mecanismos, prefixos e modificadores. Como exemplo de registro SPF:

$v = spf1 + mxa : teste.spf.com/28 - all$

onde:

- $v = spf1$: versão do SPF;
- mx, a, all: mecanismos do SPF.

Os mecanismos SPF são os responsáveis para identificar endereços IP autorizados a enviar mensagens a partir de um determinado domínio. Existem dois mecanismos básicos definidos no SPF relacionados às categorias de IPs (internos e externos ao domínio): *all* e *include*. Os demais mecanismos são responsáveis por autorizar e designar emissores. Os mecanismos do SPF são:

- a: verifica se o endereço de envio é um endereço IP válido;
- mx: verifica se o endereço de envio é o endereço MX do domínio;
- ptr: testa se o nome do endereço de envio encontra-se em um determinado domínio;
- all: testa se todas as informações coincidem;
- $include$: executa uma consulta recursiva ao SPF;
- $ip4/ip6$: verifica qual a versão do IP;
- $exists$: constrói um nome de endereço arbitrário que será utilizado numa consulta de registro DNS A. Isto possibilita um esquema complicado envolvendo partes arbitrárias de um e-mail para determinar a legalidade.

Outros mecanismos podem ser definidos, pois o SPF possibilita a criação de novos mecanismos para serem utilizados no futuro. Desta forma, os mecanismos podem coincidir, não coincidir ou então gerar uma exceção. Caso os mecanismos coincidam, seus valores de prefixo deverão ser retornados; se os mecanismos não coincidirem, o processamento deverá continuar; caso seja lançada uma exceção, retorna um valor da exceção. Alguns mecanismos possuem suporte para argumentos opcionais. Esses argumentos são responsáveis por definir nomes de domínios, conjuntos de IP e outras informações pertinentes para a validação de endereços/domínios. O SPF possui algumas limitações e problemas que não foram originalmente abordados, que são:

- servidores que possuem múltiplos domínios podem ser utilizados por usuários maliciosos para envio de mensagens, pois usuários do domínio *dominio1.com* são capazes de se passar por um determinado usuário do domínio *dominio2.com* que se encontram no mesmo servidor;

- o encaminhamento de e-mail pode não ser preciso, pois um agente é capaz de repassar uma mensagem sem alterar o endereço em *from*. Como o SPF valida o conteúdo da mensagem, o retorno para encaminhamento será sempre de uma falha (*fail*), pois as informações da consulta não irão coincidir com o servidor utilizado para fazer o encaminhamento da mensagem;

- se um servidor DNS for invadido, as informações referentes a um domínio poderão ser alteradas.

3.7 DKIM

O *Domains Keys Identified Mail* (DKIM) [Allman et al., 2007] é uma tecnologia desenvolvida em conjunto pelas empresas *Yahoo* (`www.yahoo.com`) e *Cisco* (`www.cisco.com`). Assim como o SPF, tem como objetivo autenticar o domínio e validar a integridade das mensagens enviadas. Além disso, o DKIM foi projetado para ser transparente e compatível com a infra-estrutura atual e independente de clientes, além de não requerer uma autoridade de certificação centralizada. O DKIM cria um par de chaves pública/privada para cada domínio e subdomínio. A chave pública é armazenada em um registro do DNS no campo *TXT*. A chave privada é armazenada em um diretório ou em outra forma de armazenamento local, acessado pelo sistema de saída do servidor de e-mail para gerar uma assinatura no cabeçalho de mensagens que serão enviadas pelo sistema através da chave privada.

Quando a mensagem for enviada por um servidor com suporte a DKIM, o servidor de destino deverá submeter a mensagem recebida às seguintes etapas de autenticação:

1. extrair a assinatura cifrada com a chave privada do emissor do e-mail;

2. solicitar a chave pública do domínio em questão;

3. utilizar a chave pública para verificar se a assinatura foi gerada a partir da chave privada do servidor emissor. Se a chave pública for capaz de decifrar a assinatura, significa que o e-mail foi originado pelo domínio descrito na mensagem. Do contrário, o emissor é falso e a mensagem poderá ser descartada.

Para evitar conflito com o nome do campo no registro DNS, a especificação do DKIM propõe o uso de um identificador especial para os **registros**: *domainkey*. Este *identificador* é reservado para armazenar as chaves públicas de um domínio no formato: *domainkey*.dominio1.com.

Por padrão, os algoritmos de criptografia utilizados pelo DKIM são os RSA [Kalisk and Staddon, 1998] e SHA1 [Eastlake and Jones, 2001], contudo, novos algoritmos podem ser utilizados, tal como o PGP.

A chave pública no DKIM é representada da seguinte forma:

```
mydomain.domainkey IN TXT "k=RSA; p=StDyyHZBJqNsJF2jIxKxxjKB4X
GTC22hMjVCoIV5JlchNYbKriUd0iNmfHo8u58dCSkkYp0"
```

onde:

- g: granularidade da chave;

- k: tipo da chave;
- n: notas que podem ser de interesse humano (nenhuma interpretação é feita pelos programas);
- p: chave pública codificada em *Base64*: se o valor for vazio, significa que a chave foi revogada;
- t: modo de teste. Indica se esta chave é apenas para teste.

A assinatura digital gerada pelo DKIM para autenticar a mensagem deve ser armazenada como uma **linha do cabeçalho** da mensagem a ser transmitida; o campo que armazena esta assinatura é o *DomainKey-Signature*:. Um exemplo do campo pode ser visto a seguir:

```
DomainKey-Signature: a=rsa-sha1 s=teste; d=dominio1.com.br;
c=simple; q=dns; b=StDyyHZBJqNsJF2jIxKxxjKB4XGTC22hMjVCoIV5;
```

onde:

- a: algoritmo utilizado para gerar a assinatura;
- b: assinatura codificada em Base64;
- c: forma em que o cabeçalho/conteúdo são apresentados;
- d: nome do domínio que gerou a assinatura;
- q: tipo de consulta para requisitar a chave pública no servidor DNS;
- s: *selector* que será consultado para requisição da chave pública.

O DKIM disponibiliza um status para os clientes de e-mail que é armazenado no campo *DomainKey-Status*. Os **resultados das consultas** podem ser:

- *good*: assinatura válida;
- *bad*: assinatura inválida;
- *no key*: chave inexistente;
- *revoked*: chave revogada;
- *bad format*: dados da chave inválidos;
- *non-participant*: não faz parte do domínio.

A grande maioria dos clientes de e-mail possuem como recurso a configuração de regras de cabeçalho. Servindo-se desta funcionalidade, é possível que usuários apliquem regras para manipular mensagens de acordo com o status retornado pelo *DKIM*.

As principais **vantagens** de se utilizar o *DKIM* são:

- um par de chaves pode ser criado para cada domínio da Internet, tornando possível um sistema de autenticação de servidores de e-mail global;

- qualquer domínio pode gerar suas chaves e disponibilizá-las no servidor DNS, não havendo necessidade de um repositório de chaves global;
- as informações que vão ao cabeçalho SMTP não influenciam o e-mail, isto é, se um determinado servidor não suportar *DKIM*, a assinatura será ignorada.

As **limitações** desta implementação são:

- para o funcionamento do DKIM é necessário que as configurações sejam realizadas nos servidores remetentes e nos receptores;
- pode haver atraso por problemas de desempenho do processo de recebimento de mensagens, pois o *DKIM* deverá consultar o DNS, recuperar a chave, decifrar a assinatura e depois decidir receber ou rejeitar a mensagem.

3.8 SIDF

Sender ID Framework (SIDF) [Lyon and Wong, 2006] é um serviço definido pela *Microsoft* baseado no SPF (Seção3.6), que valida (faz teste de autenticidade) o campo *FROM* da mensagem no cabeçalho do e-mail. A validação é feita de acordo com a reputação do remetente e segue um algoritmo chamado *Purported Responsible Address* (PRA) que foi definido pela RFC 4407 [Lyon, 2006]. O SIDF é um complemento do SPF, pois com ele é possível autenticar mensagens encaminhadas de outros servidores.

O algoritmo certifica se o campo do cabeçalho com o endereço do e-mail é responsável por enviar a mensagem. Como ele deriva do SPF, pode também validar o *MAIL FROM*, mas ele irá definir uma nova identidade, PRA e novos campos de política de envio, substituindo o *MAIL FROM* (MFROM by Sender ID), PRA, ou os dois.

Sender ID é licenciado pela *Microsoft*, a despeito de projetos de licença pública, e precisa da especificação para PRA na entrada *TXT* do DNS. Como isso pode causar erros de sintaxe no padrão SPF, que é mais usado, tornou-se uma política pouco implementada.

3.9 Conclusão

Neste capítulo foram apresentados os principais métodos de autenticação nos sistemas de e-mails. Alguns dos métodos são baseados em clientes como: PEM, S/MIME e PGP, enquanto outros são baseados no servidores, como: SPF, DKIM e SDIF. Os modelos baseados em clientes são pouco escaláveis e de difícil manutenção, principalmente de chaves. Os modelos baseados em servidores podem ser eficientes em detecção e rastreamento de mensagens de *spams*, mas não impedem o tráfego destes pela rede.

No próximo capítulo serão apresentadas as definições de sistemas *peer-to-peer*, algumas de suas aplicações e problemas relacionados à segurança nesse tipo de ambiente.

Capítulo 4

Sistemas *Peer-to-Peer*

4.1 Introdução

As arquiteturas de computação distribuída conhecidas como *Peer-to-Peer* (P2P) têm promovido mudanças de paradigmas nos serviços de rede em grande escala. Um dos fatores que alavancou os sistemas P2P foi a descentralização dos serviços baseados no sistema cliente/servidor. Mesmo não existindo uma definição padronizada, vários autores [Androutsellis-Theotokis and Spinellis, 2004, Rocha et al., 2004, Barcellos and Gaspary, 2006] detalham algumas das características para formar um conceito que defina uma rede P2P. São apresentados vários conceitos de sistemas distribuídos como a DHT e também alguns problemas relacionados à sua segurança.

4.2 Sistemas P2P - Definição

Nas literaturas [Androutsellis-Theotokis and Spinellis, 2004, Rocha et al., 2004] não existe um consenso sobre a definição ou caracterização de sistemas P2P. Normalmente, a definição mais superficial e aceita é:

> *Peer-to-peer é um sistema distribuído descentralizado em que todos os nós participantes têm as funcionalidades equivalentes de cliente/servidor.*

Em [Androutsellis-Theotokis and Spinellis, 2004] são apresentadas duas características para definir sistemas P2P:

- compartilhamento de recursos computacionais de forma direta sem o intermédio de um servidor central. Aceita-se o uso de um servidor central para tarefas específicas como localização de nós ou busca de informações em outros nós.
- capacidade de funcionar em uma rede instável, adaptando automaticamente a variação de conexões em caso de falhas de redes ou dos computadores.

Baseado nessas características foi proposta a seguinte definição:

> *Sistemas peer-to-peer são sistemas distribuídos compostos de nós interconectados, habilitados para se auto-organizar em uma topologia de rede com o propósito de*

compartilhar recursos como conteúdo, ciclos de CPU, armazenamento e largura da banda, capazes de se adaptar a falhas e acomodar-se à variação de números de nós, mantendo a conectividade e desempenho, sem a necessidade de intermediação ou suporte de um servidor global centralizado.

4.3 Classificações de Sistemas Peer-to-Peer

O funcionamento de um sistema P2P depende de uma rede de nós com ligações lógicas que interagem, formando uma rede chamada de *overlay* ou sobreposta. As redes *overlay* são redes lógicas criadas sobre uma rede física, isto é, uma rede com uma topologia virtual. Os sistemas P2P [Lua et al., 2005] podem ser classificadas com relação a sua estrutura e topologia em redes virtuais, conforme segue:

***Overlay* Não-estruturados.** A topologia não tem uma estrutura própria e as redes são formadas de acordo com os nós que entram na rede de forma independente. Esta independência dificulta a busca de uma informação, pois não se sabe a localização de uma informação ou de um nó. Para efetuar a busca, a forma mais comum é por **inundação**, onde uma informação a ser localizada é solicitada a todos os nós.

***Overlay* Estruturados.** A topologia caracteriza-se pela existência de um índice que facilita a localização. Este identificador situa os nós que contêm certo conteúdo. Desta forma, a localização e consultas são dirigidas diretamente ao nó de uma informação solicitada. Estes sistemas são mais complexos, sendo assim demandam manutenção tanto da estrutura quanto no dos índices dos nós, mas a viabilidade está na rapidez e precisão na busca de uma informação.

4.4 Modelo de Arquiteturas de Sistemas P2P

Diversos modelos de arquiteturas são apresentados em [Ding et al., 2005, Androutsellis-Theotokis and Spinellis, 2004, Rocha et al., 2004] quanto à centralização, cada qual diferenciado pelo grau de centralização. Há três categorias mais definidas pelos autores para classificar uma arquitetura:

Totalmente Descentralizada. É uma rede onde não há um nó central, isto é, cada nó tem autonomia, sendo cada um responsável pelo controle do próprio recurso. Os nós agem executando funções de clientes e/ou servidores, mais conhecidos como *servents*. A figura 4.1 mostra um *peer* $Peer1$ enviando uma consulta c para todos os nós de uma rede. Somente o nó $Peer2$ que contém a informação solicitada envia uma resposta r. Em seguida, depois de conhecido o nó detentor, a comunicação e a troca de dados d são realizadas diretamente entre o nó requisitante e nó detentor dos dados.

Parcialmente Centralizada. A estrutura é parecida com a arquitetura totalmente descentralizada, mas neste caso, certos nós recebem algumas funções como: armazenar a localização de um outro *peer* ou a localização de uma informação em um determinado nó. Os nós que executam este papel, normalmente são chamados de **super-nós** ou **super *peers***, sendo eleitos dinamicamente.

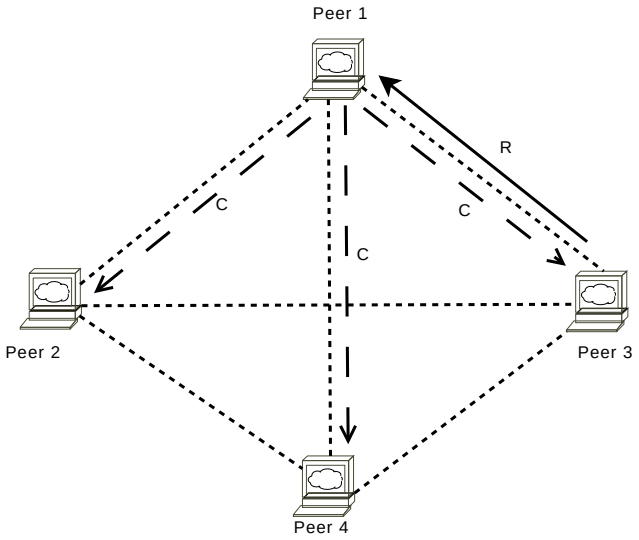

Figura 4.1: P2P Totalmente Descentralizada

A figura 4.2 mostra um nó $A1$ fazendo uma consulta c diretamente ao super-nó $SuperPeerA$, e este responde com a resposta r com a informação e localização do nó $A2$ que detém a informação. Em seguida, a troca de dados d é realizada diretamente entre o nó que fez a requisição $A1$ e o nó detentor dos dados $A2$.

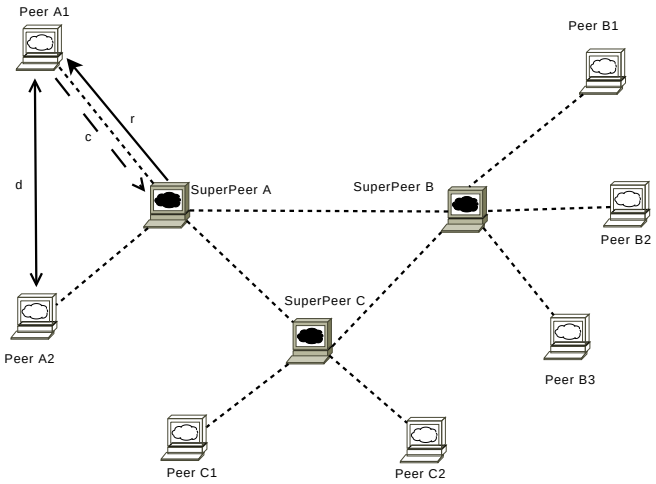

Figura 4.2: P2P Parcialmente Centralizada

Descentralizada Híbrida. Nesta arquitetura há um servidor central que facilita a localização e busca de uma informação. A consulta é sempre feita no servidor central e a troca de arquivos é realizada diretamente de um *peer* para outro. A principal desvantagem desta estrutura é a existência do próprio servidor, que pode se constituir em um ponto único de falha, dificultando a escalabilidade e constituindo-se um alvo fácil para ataques maliciosos.

A figura 4.3 mostra o funcionamento de uma rede P2P com arquitetura híbrida. É possível verificar que, para qualquer comunicação entre os nós, é necessário antes fazer uma

consulta ao super-nó ou a um servidor central. Na figura que ilustra um exemplo de consulta, um *peer* $A1$ faz uma requisição c para um servidor central, e este responde r com a localização da informação solicitada. O *download* d é realizado diretamente entre o nó requisitante e o nó detentor do arquivo.

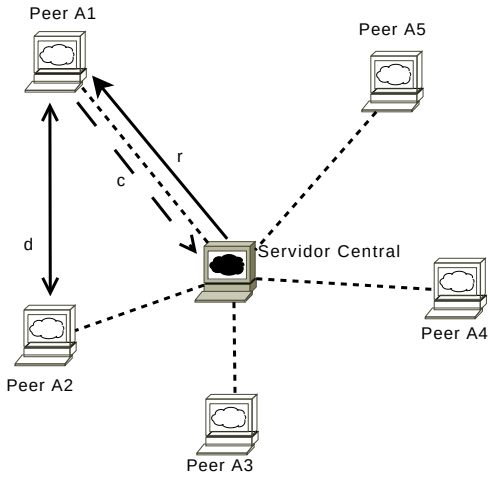

Figura 4.3: P2P Descentralizada Híbrida

4.5 Aplicações

Após o surgimento do *Napster* [Fanning, 2007], várias aplicações foram desenvolvidas utilizando o conceito P2P. A seguir, são apresentadas algumas aplicações que contêm características, diferentes como *Gnutella* [Ripeanu, 2001, Gnutella, 2007] e *Freenet* [Clarke et al., 2001]. Também é apresentado o JXTA, que é uma plataforma para desenvolvimento de aplicações P2P.

4.5.1 Napster

O *Napster* [Fanning, 2007, Ding et al., 2005] foi o aplicativo pioneiro em compartilhamento de arquivos, principalmente os arquivos de músicas. Foi através do *Napster* que o compartilhamento de dados utilizando sistemas P2P ganhou repercussão de pesquisadores e usuários. O *Napster* utiliza um servidor central, ao qual cada nó participante informa seus arquivos locais.

As operações de busca são realizadas pelo servidor, mas o *download* é requisitada diretamente ao nó detentor do arquivo. Sua arquitetura é considerada como Descentralizada Híbrida. Uma extensão do protocolo chamada *Open Source Napster Server* (OpenNap) [OpenNap, 2007] foi criada e disponibilizada para os usuários. O *OpenNap* em si, é somente um servidor e não um cliente. Para utilizar este serviço é necessária a ajuda de um cliente como o original *Napster* ou outro cliente disponível na Internet.

4.5.2 *Gnutella*

As redes *Gnutella* [Ripeanu, 2001, Gnutella, 2007, Ding et al., 2005] são classificadas como totalmente descentralizadas, pois todos os nós funcionam como cliente e/ou servidores.

A busca é realizada por inundação, ou seja, quando uma requisição é realizada, todos os nós são consultados e somente o nó detentor da informação retorna a informação. Em seguida, a troca de dados é realizada diretamente entre o nó que fez a requisição e os nós que responderam.

4.5.3 Freenet

Freenet [Clarke et al., 2001] é um sistema de armazenamento distribuído que foi projetado para proporcionar privacidade e disponibilidade, criando uma rede anônima e sem censuras. O *Freenet* foi construído para alcançar os seguintes objetivos:

- anonimato para os produtores e consumidores de informações;
- capacidade de negar acesso a informação para quem disponibiliza o recurso para armazenamento;
- não permitir a negação do acesso à informação por outros usuários;
- dinâmica eficiente de armazenamento e encaminhamento de informações;
- descentralização de todas as funções de rede.

A identificação dos objetos é baseada em DHT (Seção 4.6), utiliza palavras chaves e texto descritivo. A busca é realizada, primeiramente em seu repositório local; caso não encontre, o nó determina um valor para o *Hops-To-Live* (HTL), que é um valor que determina a quantidade máxima de nós pelos quais a mensagem de busca será propagada, impedindo uma busca infinita. Em cada nó o HTL é decrementado. Ao chegar a 0, a mensagem de busca não será mais passada adiante.

4.5.4 JXTA

O JXTA [Waterhouse, 2001, Wilson, 2002, Brookshier et al., 2002] é uma especificação independente de plataforma e linguagem para comunicação entre dispositivos não importando a localização física e tecnologia de rede utilizada. Primeiramente foi desenvolvida pela *Infrasearch,* que foi adquirida pela *SUN Microsystems*. O JXTA surgiu decorrente da palavra *juxtapose*, que significa "unir-se e caminhar lado a lado".

A plataforma JXTA foi especificada para definir algumas abstrações de rede que permitem a construção de uma rede virtual P2P, ou seja, muitos dos serviços básicos estão implementados como módulos: serviços de descoberta de *peers* ou de recursos na rede, comunicação entre dois ou mais *peers*.

Uma rede virtual JXTA é composta por *peers* que operam de forma assíncrona em relação a outros *peers* e são identificados unicamente por seu *peerID*. *Peer* é qualquer entidade que possa participar e interagir com a rede. A rede consiste de alguns tipos de *peers*, que podem assumir algumas das funções descritas a seguir:

***Edge Peers*.** São os *peers* simples, podendo tanto ser computadores ou outros dispositivos computacionais, conectados à rede. Os *Edge peers* ainda podem ser classificados em *Minimal Peers* que são dispositivos com restrições de recursos, como celulares e computadores de mãos; e *Proxy Peers* que são computadores que realizam funções de *proxy* para os *peers* que não possuem endereço IP público ou que estão protegidos por um *firewall*.

Rendezvous Peers. São *peers* que têm a função de atuar como armazenamento (*cache*) de informação sobre os *peers* conectados, facilitando a descoberta de recursos; normalmente realizado com computadores destinados somente para este serviço.

Relay Peers. São os *peers* que mantêm as informações de roteamento, realizam passagem de mensagens para outros *peers* que ficam atrás de um roteador, *firewall* ou um *Network Address Translation* (NAT).

Os *peers* se organizam em grupos que no sistema são conhecidos como *Peer Groups*, e são identificados por seus *peerGroupID*. São usados para definir um conjunto de serviços e recursos utilizados por vários *peers*. As entidades do JXTA, incluindo *peers*, *peers groups*, *pipes* e serviços são representadas usando *advertisements*, que são documentos *eXtensible Markup Language* (XML) que contêm a informação. Quando um recurso é disponibilizado na rede, significa que o seu *advertisement* foi criado e publicado, ou seja, foi enviado para os demais *peers*.

Os *peers* transmitem mensagens através de *pipes*, que são canais virtuais de comunicação usados para enviar e receber mensagens entre os serviços e aplicações. Possuem ID únicos, e não são associados a nenhum dispositivo de rede real. As mensagens são documentos em XML, que possuem roteamento baseado no ID da fonte, carregando em seu cabeçalho a informação de roteamento necessária, tal como a seqüência de *peers* a ser percorrida. O *endpoint* é o ponto extremo em uma comunicação na rede JXTA. Qualquer informação transmitida terá uma origem e o destino *endpoint*. O *endpoint* corresponde à interface de rede utilizada para enviar e receber dados.

O JXTA possui um conjunto de protocolos [Waterhouse, , Brookshier et al., 2002] distintos, providos por qualquer *peer group* criado. São eles:

- ***Peer Discovery Protocol* (PDP)**: utilizados pelos *peers* para descobrir e publicar recursos dinamicamente;
- ***Peer Resolver Protocol* (PRP)**: permite o envio de uma consulta genérica a outros *peers*;
- ***Peer Information Protocol* (PIP)**: possibilita que um *peer* obtenha informações sobre outros *peers*;
- ***RendezVous Protocol* (RVP)**: responsável por propagar e controlar mensagens dentro de um grupo. O RVP é base para os protocolos: PRP e PBP;
- ***Pipe Binding Protocol* (PBP)**: disponibiliza um mecanismo para ligar um canal de comunicação entre um *pipe* e um *endpoint*;
- ***Endpoint Routing Protocol* (ERP)**: fornece um conjunto de mensagens usadas para possibilitar o roteamento de mensagens de um nó de origem até um nó destino.

4.6 DHT

Distributed Hash Table (DHT) [Lua et al., 2005] são classes de sistemas distribuídos descentralizados que fragmentam um conjunto de chaves entre nós participantes, e podem rotear mensagens eficientes para um único dono de uma chave. Em uma tabela de índices, cada nó

da DHT é geralmente elaborada para receber um grande número de nós e suportar a entrada e saída ou falha constantes de nós. Esta infra-estrutura pode ser usada para construir sistemas mais complexos, como sistemas de arquivos distribuídos, sistema de compartilhamento de arquivos *peer-to-peer*, armazenamento cooperativo, sistemas de compartilhamento de conteúdo, entre outros serviços distribuídos.

4.6.1 Características

As características de um DHT se concentram nas seguintes propriedades:

- Descentralização: os nós coletivamente formam o sistema, sem coordenação central.
- Escalabilidade: o sistema deve funcionar eficientemente, mesmo com centenas ou milhares de nós.
- Tolerância a falhas: o sistema deve funcionar (até certo momento) mesmo com nós constantemente entrando, saindo e falhando.

Uma técnica usada para alcançar estas propriedades é que qualquer nó se coordene com alguns outros nós do sistema - na maioria das vezes, com $log\ n$ participantes - para que assim, somente uma quantidade de processamento seja necessária para cada mudança de membros.

4.6.2 Implementações de DHT

Muitas implementações utilizam o conceito de DHT para fornecer uma infra-estrutura distribuída. Na seqüência serão apresentadas algumas das mais conhecidas, como CAN [Ratnasamy et al., 2001], *Tapestry* [Zhao et al., 2001], *Pastry* [Rowstron and Druschel, 2001], *Chord* [Stoica et al., 2001] e OpenDHT [Rhea et al., 2005].

CAN

O *Content-Addressable Network* (CAN) [Ratnasamy et al., 2001] é uma infra-estrutura cujo princípio está no uso de um espaço de coordenadas cartesianas virtual de n dimensões. O espaço é dinamicamente fracionado entre os nós presentes na rede e estas partições são subdivididas ou reagrupadas de acordo com a entrada e saída dos nós.

Para a entrada de um nó na rede, primeiramente faz-se a conexão a um nó que já está conectado e em seguida faz-se a escolha de um ponto aleatório no espaço cartesiano definido pelo sistema. Este nó que acabou de acessar a rede envia uma requisição (através do nó ao qual conectou) ao nó responsável pela partição do referido espaço. Ao receber a requisição, o nó responsável fraciona o espaço em duas partes e designa uma delas ao nó requisitante, informando as chaves que serão de sua responsabilidade.

Antes da saída de um nó na rede, ele deve enviar aos nós vizinhos a tabela de sua responsabilidade. Caso haja algum problema e o nó saia antes de enviar a comunicação de saída, os nós vizinhos detectarão sua ausência, através de mensagens de verificação, com isso um dos vizinhos assume o controle da partição pertencente ao nó com problema.

Tapestry

O *Tapestry* [Zhao et al., 2001] é uma infra-estrutura que permite o roteamento de mensagens a objetos (ou a cópia mais próxima a eles, se mais de uma cópia existir) de uma forma distribuída, auto-organizável e tolerante a falhas. As informações de roteamento e localização são distribuídas entre os nós da rede. A consistência da topologia é verificada dinamicamente e pode ser reconstruída em caso de perda; a localização é baseada nos mecanismos de roteamento em malha. Esta estrutura distribuída permite que os nós localizem objetos em uma rede de tamanho arbitrário, usando mapas de roteamento pequenos e de tamanho constante.

Os nós podem assumir o papel de servidores (que armazenam objetos), roteadores (que encaminham mensagens) e clientes (que originam requisições). Cada nó mantém um mapa de vizinhos, cada mapa possui múltiplos níveis, cada nível n contendo apontadores para nós cujo id deve coincidir em n dígitos. Cada entrada no mapa de vizinhos corresponde a um apontador para o nó mais próximo na rede cujo id confere com o número no mapa, até uma posição de dígitos. Mensagens roteadas são incrementadas através dos nós dígito por dígito, da direita para a esquerda, por exemplo, $xx3 \rightarrow x23 \rightarrow 123$.

A malha utiliza um nó raiz para cada objeto, que serve como uma garantia a partir do qual um objeto pode ser localizado. Quando um objeto o é inserido na rede no nó ns, um nó raiz nr é associado ao objeto usando um algoritmo determinístico global. Uma mensagem é então roteada de ns para nr, armazenando dados na forma de um mapeamento $[o, ns]$ em todos os nós ao longo do caminho. Durante uma operação de busca, mensagens destinadas a o são inicialmente roteadas com destino nr, até que um nó seja encontrado contendo o mapeamento $[o, ns]$.

Pastry

O *Pastry* [Rowstron and Druschel, 2001] é uma rede completamente descentralizada, escalável e auto-organizável. No *Pastry*, cada nó tem um identificador único chamado de $nodeId$. Esse identificador, bem como as chaves dos pares (chave, valor), são representados por números de 128 bits, gerados randomicamente quando o nó entra na rede a partir do endereço IP ou da chave pública do nó, e dispostos num espaço circular. Como as demais implementações de DHT, os nós numa rede *Pastry* armazenam informações referentes às chaves distribuídas nos nós, procurando colocar sempre o par (chave, valor) no nó com identificador mais próximo ao valor da chave requisitada.

Para armazenar e localizar estes pares, o *Pastry* utiliza uma tabela de roteamento com ponteiros para os demais nós da rede. Esta tabela armazena um conjunto de nós vizinhos e nós chamados folhas. Os nós vizinhos são os nós mais próximos do nó em questão e são utilizados para controlar a entrada e saída dos nós na rede; já os nós folhas são usados com a intenção de roteamento de mensagens. O *Pastry* é utilizado pelo PAST [Druschel and Rowstron, 2001] que é um sistema de armazenamento P2P que objetiva fornecer alta confiabilidade e alta disponibilidade dos dados, escalabilidade e segurança.

Chord

O *Chord* [Stoica et al., 2001] é um algoritmo de DHT baseado no par chave/valor. A topologia lembra um anel virtual. Na rede *overlay*, o id do cliente da rede representa a chave, e

o valor correspondente são os dados que são armazenados nos nós. No *Chord*, cada nó armazena somente um subconjunto de todas as chaves, o que aumenta a confiabilidade da rede.

O *Chord* pode ser utilizado para implementar diferentes serviços, como a busca de informação distribuída. O *Chord* foi desenvolvido com alguns objetivos:

- **Escalabilidade**: o sistema deve continuar funcionando de forma eficiente mesmo quando a rede crescer.
- **Disponibilidade**: o sistema deve funcionar mesmo quando a rede se dividir ou quando alguns nós falharem.
- **Balanceamento de carga**: os pares chave/valor devem ser distribuídos igualmente no sistema.
- **Flexibilidade**: o sistema deve suportar mudanças rápidas na topologia da rede.

Para a localização de uma chave, o nó solicitante necessita apenas conhecer seu nó sucessor no anel, assim numa pesquisa procura-se somente este nó sucessor; caso este nó não possua a informação, a mensagem será propagada para o próximo nó sucessor, e assim por diante até encontrar a chave desejada. Ao encontrar a chave, o nó detentor responde com informação diretamente ao nó solicitante. Para não ocorrer a consulta em todos os nós para localização de uma informação, alguns dados dos nós são armazenados em uma tabela chamada de *finger table*.

OpenDHT

É um projeto desenvolvido para criar uma DHT de acesso público em forma de serviço. Esta rede possui uma interface simples, com funções básicas de inserção, recuperação e remoção das entradas contendo chave e valor e que, ao mesmo tempo, dispõe de todos os recursos oferecidos pelas tabelas *hash* distribuídas.

O *OpenDHT* [Rhea et al., 2005] é uma implementação utilizando a DHT *Bamboo* [Rhea, 2007] que tem como base a rede *PlanetLab* [Peterson et al., 2006]. O *Bamboo* é uma DHT baseada nas características do *Tapestry*. Para armazenar informações na *OpenDHT* são necessários os seguintes dados, mostrados na tabela 4.1.

OpenDHT	
chave	String para localizar uma informação
valor	Onde são colocadas as informações (tamanho máximo 1024 bytes)
senha	Necessário para remoção da informação
TTL	Tempo que permanecerá na *OpenDHT* (*default*=1 semana)

Tabela 4.1: Dados da *OpenDHT*

As principais funções oferecidas pela *OpenDHT* são listadas na tabela 4.2

Funções	
$put(k, v, H(s), t)$	Grava (k, v) para um determinado tempo t (TTL) com uma senha $H(s)$, onde k é a chave e v o valor
$get(k)$ returns $\{(v, H(s), t)\}$	Retorna todos os valores referentes a k
$remove(k, H(v), s, t)$	Remove (k, v) com a senha s

Tabela 4.2: Funções da *OpenDHT*

4.7 Segurança em Sistemas P2P

Os sistemas P2P trazem muitos benefícios, porém, como em qualquer outro serviço, também existem problemas como informações distorcidas e corrompidas. Um exemplo é o usuário disponibilizar um filme com um título e o conteúdo ser de outro filme. Alguns aspectos em relação à segurança em sistemas P2P [Barcellos and Gaspary, 2006] são abordados para analisar a disponibilidade, autenticidade, autorização, integridade, reputação e anonimato.

4.7.1 Disponibilidade

A disponibilidade de uma rede P2P pode ser afetada através de ataques de DoS (negação de serviço) [Dumitriu et al., 2005] ao sistema de roteamento das redes. Este tipo de ataque pode comprometer além do nó local, o sistema como um todo, pois um nó pode ser responsável pelo roteamento a outro nó.

O objetivo do ataque DoS é fazer com que um ou mais nós maliciosos enviem muitas requisições, tornando a rede P2P sobrecarregada. Esta sobrecarga pode fazer com que um nó fique incomunicável e até seja excluído da rede por outros nós. O outro ataque é de roteamento; normalmente, os nós maliciosos desviam as mensagens para longe dos destinos ou para os nós maliciosos. Este tipo de ataque pode fazer com que ocorram problemas na busca, podendo incorrer uma resposta errada e/ou maliciosa.

4.7.2 Autenticidade

No sistema P2P, uma das formas de autenticar um nó é através de nós conhecidos, isto é, um nó aceita somente conexões de nós que já validou anteriormente. A validação pode ocorrer de forma direta, que são os nós validados pelo próprio nó, ou indiretamente, que são os nós recomendados pelos nós conhecidos e já validados. Um dos problemas que podem ocorrer é a falsificação de várias identidades, este ataque é mais conhecido como *Sybil* [Douceur, 2002]. Em uma rede de larga escala, como o sistema P2P, é quase impossível evitar este tipo de ataque.

Se um mesmo nó pode assumir várias identidades, então a replicação pode ser comprometida, visto que múltiplas réplicas de um objeto podem ficar sob controle de um nó malicioso. Uma das formas para impedir este ataque é o gerenciamento de identidades através de certificados digitais que podem ser assinados por uma autoridade certificadora, só que a centralização deste serviço em um sistema fortemente distribuído com a rede P2P não é aceitável.

4.7.3 Reputação

Uma das características de uma rede P2P é a colaboração. Quando algum nó de um sistema P2P não colabora, os sistemas ficam comprometidos, pois normalmente os aplicativos assumem que um nó é capaz de aderir e participar do serviço. Os nós que utilizam mais recursos do que disponibilizam são conhecidos como *free-riders* [Hughes et al., 2005]. Para evitar que isto aconteça, algumas ações punitivas podem ser tomadas:

- no nível mínimo de punição, pode-se montar um esquema que limita a propagação de mensagens enviadas a partir dos *peers* que fazem mais *download* do que *upload*;
- no segundo nível, o sistema poderá ignorar pesquisas geradas pelos *free riding peers*;
- no terceiro nível, o sistema pode desconectar *peers* maliciosos ou improdutivos da rede.

Em alguns sistemas são utilizados os esquemas de confiança e reputação, onde os nós que participam há mais tempo da rede e têm um nível de reputação elevado quanto à colaboração e participação serão os recomendáveis para a busca de uma informação. Uma outra forma é usar uma moeda de troca, quanto mais um nó colaborar mais eficiente será sua busca e *download* de um arquivo.

4.7.4 Autorização

A autorização em uma rede P2P consiste em prover um mecanismo de controle de acesso entre os nós. Normalmente as aplicações que compartilham recursos abertamente não definem regras como quem, quando e o que pode ser acessado. O controle de acesso em uma rede P2P é complexo por ser um sistema muito dinâmico, onde os nós entram e saem da rede frequentemente.

Uma forma de criar um mecanismo de autorização confiável é a utilização de SPKI/SDSI no sistema P2P [Mello et al., 2005]. O SPKI/SDSI é uma infra-estrutura de chaves públicas/privadas completamente descentralizada definida para a autorização e autenticação. Cada *principal* (nó) é caracterizado por um par de chaves que pode emitir e assinar certificados, sem o aval de uma CA como no padrão X.509. Funciona como uma cadeia de confiança, onde certificados *principal* podem ser delegados de um *principal* a outro, criando assim uma cadeia de certificados SPKI/SDSI.

Existe outra proposta [Park and Hwang, 2003] que é baseada em *Role-Based Access Control* (RBAC) [Sandhu et al., 1996]. O RBAC é aplicado para descrever mecanismos de segurança que controlam o acesso de usuários a recursos computacionais, baseado na construção de papéis. Os papéis definem um conjunto de atividades concedidas para usuários autorizados. Neste caso, é criada uma entidade central, como servidor de políticas de acesso ao recurso dos nós. As políticas são transferidas em um determinado período para os nós, para que as decisões possam ser tomadas individualmente por cada nó.

4.7.5 Integridade

Nos sistemas P2P, os serviços mais utilizados são os de armazenamento e compartilhamento de arquivos. Os arquivos disponibilizados não devem estar corrompidos, seja de forma

intencional ou por problemas no armazenamento. A forma mais comum de verificar a integridade é através da geração de um *hash* do arquivo. Outra forma é a utilização de um método de criptografia, como o uso dos pares de chaves públicas e privadas.

4.7.6 Anonimato

É um dos aspectos que muitos aplicativos disponibilizam para oferecer resistência à censura, confidencialidade e privacidade aos usuários. O anonimato [Clarke et al., 2001] consiste em não permitir a identificação:

- do autor ou daquele que publicou o arquivo ou objeto;
- da identidade do nó que armazena o arquivo ou objeto;
- da identidade do conteúdo do arquivo ou objeto;
- dos dados para requisição e recuperação do arquivo ou objeto.

A centralização de uma informação, como no modelo de rede P2P onde há um nó central para armazenamento de algumas informações, facilita a identificação, como no caso do *Napster*. Para evitar, os objetos são distribuídos e identificados por chaves como no *Freenet* [Clarke et al., 2001], que é um dos aplicativos que cumpre os aspectos de anonimato.

4.8 Conclusão

Neste capítulo foram apresentadas algumas das possíveis definições e classificações de redes *peer-to-peer*. Apesar de ser um conceito bastante debatido, ainda não há um consenso entre os pesquisadores. Também foram descritos alguns dos principais aplicativos que utilizam o conceito de sistemas P2P, bem como alguns conceitos de tabelas *hash* distribuídas e algumas implementações de DHT e certos problemas relevantes em sistemas P2P, principalmente segurança.

O próximo capítulo tratará da proposta do trabalho que visa construir uma nova infraestrutura de e-mail descentralizada, com o propósito de diminuir alguns problemas no sistema de e-mail.

Capítulo 5

Um sistema de correio eletrônico baseado no paradigma peer-to-peer

5.1 Introdução

Este capítulo visa apresentar uma nova infra-estrutura de sistema de e-mail baseada em sistemas *peer-to-peer*, utilizando uma abordagem onde as mensagens só serão enviadas por decisão do destinatário, após o recebimento de uma notificação. As mensagens são transportadas, quando solicitadas, diretamente de um repositório dos remetentes para os destinatários. Primeiramente, apresenta-se a arquitetura, onde são detalhados os componentes da infra-estrutura e em seguida são apresentados o funcionamento de envio e recepção de uma mensagem a um receptor ou a vários receptores, bem como aspectos relativos à replicação de mensagens e eliminação de mensagens já lidas.

5.2 Objetivo

Visualizando alguns problemas relacionados a algumas características do sistema convencional de e-mail, como: centralização dos dados dos usuários, armazenamento de e-mails e a forma como as mensagens são trafegadas, o objetivo principal é apresentar uma infra-estrutura baseada nas características de um sistema P2P onde os usuários de compartilhamento de arquivos nunca enviam o arquivo para outro usuário, simplesmente disponibilizam em seu repositório local e somente as informações relativas aos arquivos são publicadas.

5.2.1 Objetivos específicos

- Criar um sistema de e-mail baseado na abordagem *PULL* em vez de uma arquitetura tradicional que funciona na abordagem *PUSH*;
- Criar uma arquitetura baseada em um sistema distribuída P2P totalmente descentralizada. O sistema P2P é conhecido por ser um sistema colaborativo, aproveitando dos recursos dos computadores dos usuários como disco local, processamento e a banda da rede.
- Criar um sistema onde não terá um servidor central para armazenamento de mensagens de e-mail, pois as mensagens serão armazenados utilizando o disco local dos *peers*;

- Elaborar um sistema onde não haverá um servidor para autenticação; a autenticação será baseada em chaves PGP;
- Criar um sistema de notificação baseada em um sistema DHT, onde serão postados os dados dos *peers* e as notificações de novas mensagens e de mensagens lidas.

5.3 Arquitetura

A arquitetura descrita na figura 5.1 demonstra uma rede formada por clientes em um sistema de mensagens utilizando a abordagem proposta. Analisando a figura, podemos visualizar os clientes (MUA) conectados a um *p2pMTA*, que é responsável por postar e buscar as notificações, depositar e receber as mensagens e também cifrar e decifrar as mensagens e notificações.

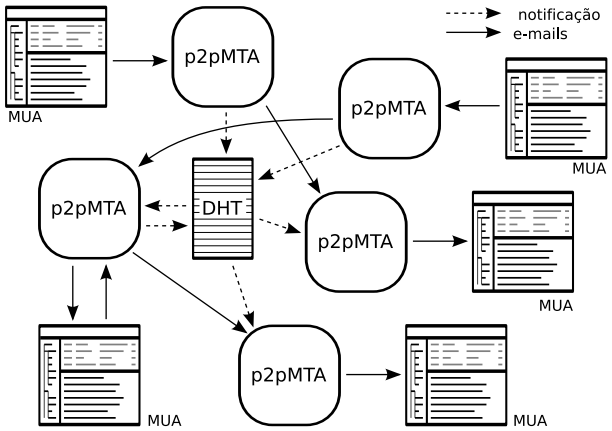

Figura 5.1: Arquitetura do sistema de e-mail proposto

A arquitetura proposta usa uma DHT como canal de comunicação para a troca de notificações entre os *peers*. Assume-se que a DHT provê três serviços básicos:

- *put(x,v,p,t)*: registra uma entrada definida por uma chave x, o valor v, a senha p e o tempo de validade t; a DHT suporta colisões de chaves: duas ou mais valores podem ser armazenadas usando a mesma chave x;
- *get(x)*: busca todas as entradas $e_i = [x_i, v_i]$ em que $x_i = x$;
- *del(x,v,p)*: apaga a entrada definida por $[x, v]$, protegida por uma senha p.

Deve ser notado que a senha p somente protege a entrada conta a operação de *del*, mas não impede as operações de consultas realizado pelo *get*. Foi considerado que a infraestrutura da DHT apaga todas as entradas quando expira o tempo de validade de acordo com as características da *OpenDHT* [Rhea et al., 2005]. A utilização do *hash* SHA1 é viável por verificar a integridade da mensagem após a realização do *download* e também por ser quase impossível uma duplicação do *hash* gerado.

5.4 Um nó *p2pMTA*

Os componentes de um *p2pMTA* podem ser visualizados na figura 5.2. O MUA é o cliente de e-mail escolhido pelo usuário; o *inbox* é onde são armazenadas as mensagens recebidas pelo usuário; o *spool* é a área que recebe as mensagens a enviar; o *key cache* é o repositório de chaves PGP já conhecidas, isto é, são chaves públicas dos usuários aos quais já foram enviadas mensagens anteriormente; o *outbox* é onde são disponibilizadas as mensagens que estão aguardando a leitura pelos seus destinatários e o *p2pMTA* é o responsável pelas transações existentes entres as áreas. Os protocolos utilizados para a comunicação entre o MUA e o nó *p2pMTA* é o SMTP (para envio) e o POP3 (para a leitura).

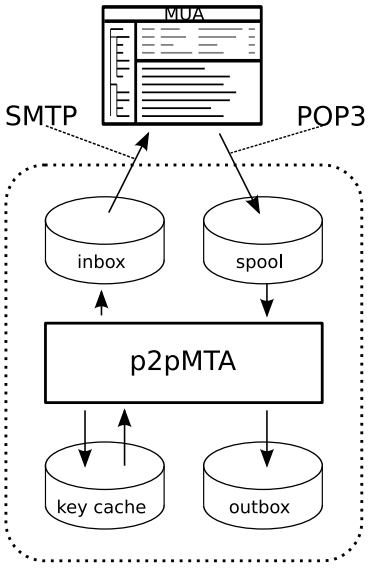

Figura 5.2: Componentes de um *p2pMTA*

Quando inicializado, um *peer* i posta na DHT uma notificação denominada de *descritor do peer* d_i, utilizando o endereço de e-mail como **chave**. O descritor d_i contém em seu campo **valor** o par *IP:Porta* do *peer*, uma senha que é determinada aleatoriamente pelo *peer* e um **tempo de validade** t de um dia (86400 segundos) arbitrariamente escolhido. O descritor ficará na DHT podendo ser renovado ou atualizado quando necessário ou até mesmo removido, caso o usuário desligue o seu nó. Se o *peer* não remover o descritor, este será automaticamente removido da DHT por causa do tempo t estipulado na sua publicação.

5.5 Funcionamento

Para manter a compatibilidade com um MUA convencional, recorreu-se à utilização dos serviços SMTP e POP3 no seu modo nativo de acordo com as RFCs. Para que a formatação das mensagens fosse entendida pelo sistema não houve alterações no formato da mensagem de acordo com padrão definido pela RFC 2822. O *p2pMTA* é a aplicação responsável pela transformação das mensagens recebidas pelo SMTP para a infra-estrutura apresentada.

O endereço de e-mail de um *peer* tem o formato *name@group*, onde *name* define a identidade de um *peer* e o *group* é o grupo ao qual ele pertence. Assumimos que cada *peer* tem um par de chaves PGP públicos/privados; a chave pública k é disponibilizada para outros

peers, enquanto a chave privada k' é armazenada em um lugar seguro. O nome do arquivo disponibilizado é um *hash* SHA1 do próprio arquivo de mensagem.

O endereço de e-mail de um usuário *p2pMTA* é qualquer e-mail no formato usado no sistema convencional **nome@domínio**, onde o domínio define o grupo (*group*) ao qual pertence, por exemplo: *edson@p2pmail.com*.

5.5.1 Envio e Recepção de Mensagens

O procedimento necessário para o **envio** de uma mensagem de e-mail pode ser visualizado no diagrama apresentado na figura 5.3. É possível verificar todo o procedimento executado pelo *p2pMTA* desde a recuperação da mensagem do *spool* até a colocação da mensagem cifrada no *outbox* e a postagem da notificação na DHT.

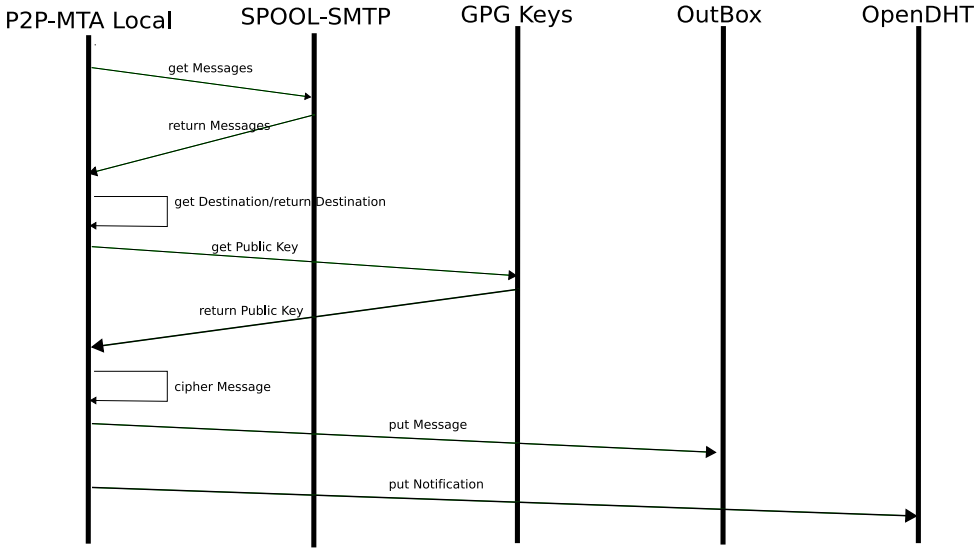

Figura 5.3: Processo para envio de um e-mail

O procedimento de recepção de uma mensagem de e-mail é apresentado no diagrama que é aparce na figura 5.4.

Os passos necessários para transferir um e-mail m_i de um *peer* remetente S para um *peer* receptor R são apresentados na figura 5.5:

1. o remetente S envia uma mensagem m_i utilizando um cliente (MUA) para um *p2pMTA* local através do protocolo SMTP; a mensagem é armazenada em uma área de *spool* de S;

2. S procura a chave pública do destinatário k_r em seu repositório local de chaves, ou no *peer* destinatário R, através da *Uniform Resource Locator* (URL) [Berners-Lee et al., 1994] `http://IP:PORTA/pubkey` (IP e $PORTA$ obtidos do descritor d_r). Caso não encontre a chave, retorna um erro e uma nova tentativa de envio será realizada mais tarde;

3. a mensagem m_i é cifrada usando k_r: $m'_i = k_r\{m_i\}$, e um *hash SHA1* h_i obtido de m'_i: $h_i = SHA_1(m'_i)$;

4. o arquivo m'_i é armazenado no *outbox* de S, usando h_i como nome do arquivo;

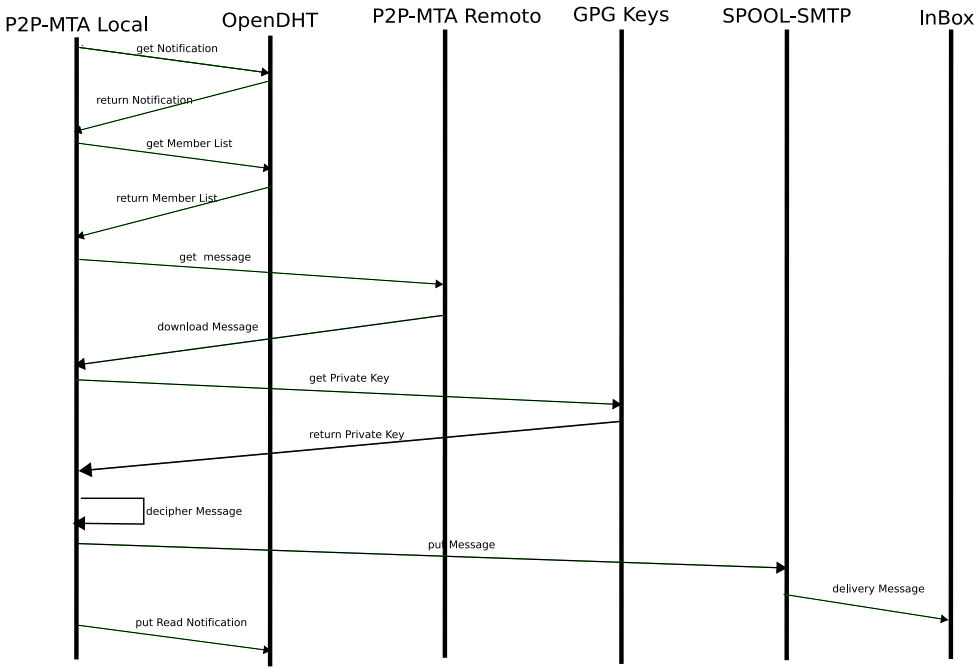

Figura 5.4: Processo para recebimento de um e-mail

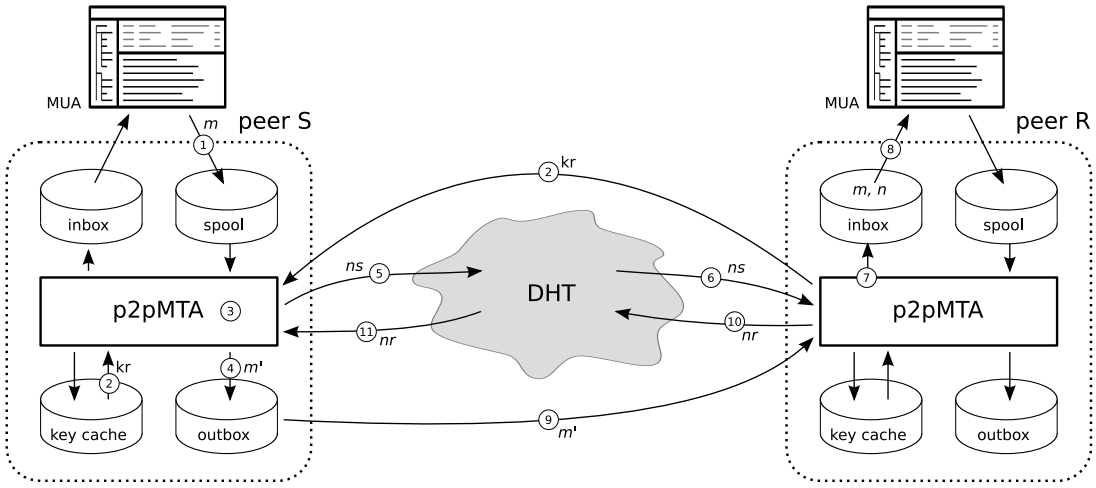

Figura 5.5: Envio e Recebimento de um e-mail

5. S publica a notificação de envio ns_i para R na DHT, usando como chave *ns:peer@group* (onde *peer@group* é o endereço de R). O valor da notificação ns_i contém alguns dados do cabeçalho da mensagem m_i (*sender@group*, *assunto*, *data*, *tamanho*, *nome/tipo/tamanho do anexo*), h_i, e a senha p_i (um valor aleatório) para a entrada na DHT; o valor de ns_i é cifrado usando k_r. O tempo de validade t foi arbitrariamente definido como uma semana. A notificação ns_i é copiado localmente em S;

6. periodicamente, R verifica a entrada ns na DHT; cada ns_i é decifrada usando a chave privada k'_r, para visualizar o cabeçalho da mensagem, o hash de h_i e a senha p_i;

7. para cada ns_i, R cria uma mensagem n_i na área (*Inbox*) local, usando cabeçalho m_i e sem conteúdo no corpo;

8. o cliente R busca todas as mensagens e as apresenta para o usuário;

9. se o usuário decide ler n_i, R busca a mensagem m'_i usando a URL `http://IP:PORTA/outbox/`h_i (IP e $PORTA$ que são obtidas de d_s); m'_i é decifrada utilizando k'_r para obter m_i. De outra forma, se o usuário decide apagar a mensagem, o arquivo correspondente à mensagem é apagado do *inbox* de R;

10. após a mensagem m'_i ser lida ou a notificação n_i ser apagada; a notificação ns_i é removida da DHT; e uma notificação de mensagem lida nr_i é postada na DHT, usando *nr:peer@group* como chave (onde *peer@group* é o endereço de S). O valor nr_i é o hash h_i, e a mesma senha p_i de ns_i é usada para proteger a entrada na DHT;

11. periodicamente, S procura notificações nr; para cada nr_i, é removida a entrada na DHT, e o arquivo correspondente à mensagem é removido do *outbox*.

5.5.2 Envio a vários destinatários

Os passos indicados na seção anterior apontam o comportamento do sistema em um caso simples, isto é, em que um *peer* envia uma mensagem a um único destinatário. Num cenário onde se envia uma mensagem para vários destinatários, o que é muito comum em listas de discussão, ocorrem alterações no cenário anterior visto que uma mensagem m_i é enviada para vários receptores R_x:

- m_i deverá ser cifrada com a chave pública k_x de cada receptor R_x. O PGP faz isso de forma versátil: ele cifra m_i usando uma chave aleatória de sessão ks ($m'_i = ks\{m_i\}$), então cifra ks usando k_x para cada R_x para obter as chaves cifradas kc ($kc_x = k_x\{ks\} \forall R_x$); então todas as kc_x são anexadas à m'_i;

- no próximo passo, S deverá postar uma notificação ns_x para cada receptor R_x;

- somente quando todas as notificações de mensagens lidas ns_x forem respondidas com nr_x ou as notificações expiradas, o arquivo de m_i armazenado em S será removido.

O processo de repeticão segue a partir da operação 3 da figura 5.3.

É fácil perceber que cada mensagem é armazenada apenas uma vez, mesmo que a mensagem tenha vários receptores.

5.5.3 Replicação de Mensagens

Em um sistema e-mail convencional, o servidor de correio eletrônico atua como um armazenamento temporário para mensagens de e-mail, se o servidor destinatário estiver fora do ar. Esta arquitetura propõe a não-existência de um servidor centralizado de e-mail para armazenamento, o que pode causar problemas, porque tanto o remetente quanto o destinatário deverão estar disponíveis para permitir que um e-mail possa ser transmitido. Para contornar essa limitação, os *peers* são organizados em grupos (*peer groups*).

Para proporcionar maior disponibilidade dos e-mails que estão aguardando a leitura pelos destinatários, *peers* de um mesmo grupo replicam o conteúdo dos seus *outboxes*. Um grupo é formado por *peers* que tenham alguma relação de confiança em si. Cada grupo tem um nome *gname* e um descritor gd_i registrado na DHT localizado pela chave *grupo:gname*.

O valor do descritor de um grupo contém uma lista com os endereços dos membros do grupo. Para cada grupo, um *peer* chamado de *group master* é o responsável por registrar e manter o descritor do grupo. Atualmente, o *group master* é definido manualmente, bem como membros do grupo; técnicas mais sofisticadas de criação e manutenção de grupos estão sendo estudadas e serão citadas como trabalhos futuros.

O metódo de replicação do *outbox* é bastante simples: um determinado *peer* de um grupo verifica as listas de mensagens nos *outboxes* contidas em cada um dos *peers* pertencentes a um grupo, a partir das URLs `http://IP:PORTA/outbox/Msglist`, onde o IP e $PORTA$ são recuperadas do descritor do *peer* publicado na DHT e o $Msglist$ é o arquivo que contém a lista de mensagens disponíveis para leitura de seus destinatários. Em seguida, ele compara o conteúdo da lista com os e-mails presentes em seu *outbox* local; mensagens de e-mail que não estão mais na lista devem ser removidas, porque elas foram apagadas pelo seu proprietário; mensagens de e-mail que não estão no *outbox* devem ser copiadas localmente, para serem replicadas.

Durante a transferência de mensagens, se o *peer* remetente S estiver indisponível, o *peer* receptor R poderá pegar a lista de outros *peers* nos descritores do grupo, para descobrir outros *peers* de onde as mensagens m'_i poderão ser replicadas, e em seguida, tentará copiar a mensagem de um deles.

O processo de replicação de mensagem pode ser visualizado na figura 5.6

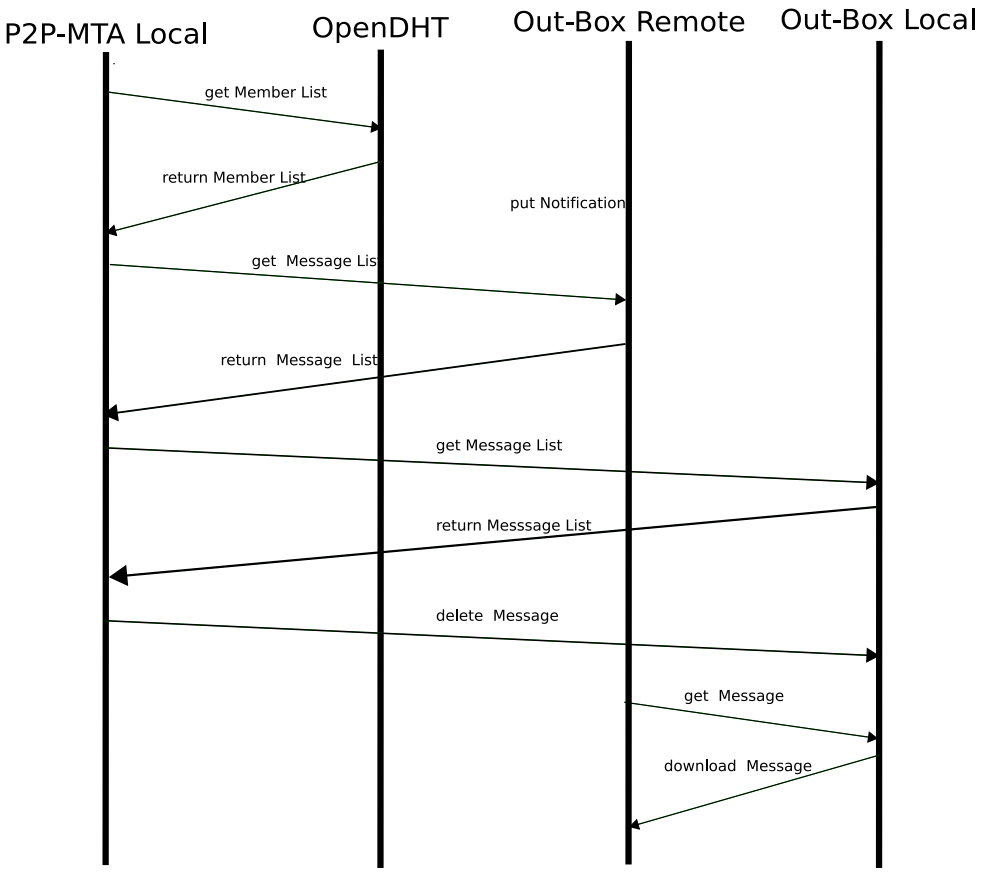

Figura 5.6: Processo para replicação de mensagens

5.5.4 Manutenção e Limpeza

Há algumas informações que são armazenadas localmente (em cada *peer*) ou na DHT. Estas informações devem ser excluídas quando não são mais necessárias. As informações só serão removidas quando:

- todas as entradas na DHT têm uma validade (tempo de vida) associada; quando o tempo de vida expira, a infra-estrutura da *OpenDHT* as remove de forma transparente [Rhea et al., 2005];
- e-mails armazenados no *outbox* do remetente são apagados quando:
  1. o receptor ler a mensagem e postar a notificação de mensagem lida;
  2. quando o tempo de validade da entrada correspondente na DHT expirar.
- e-mails replicados entre os membros do grupo são apagados durante o procedimento de replicação (seção 5.5.3), se os proprietários das mensagens as apagarem de seus respectivos *outboxes*.

5.6 Limitações

Na elaboração da arquitetura proposta não foram considerados alguns ítens, como:

- Por ser totalmente descentralizado há problema de comunicação quando um *peer* estiver utilizando NAT para acesso a Internet;
- Os arquivos são disponibilizados através do protocolo HTTP e, por padrão, utiliza a porta 80, pode ser que o *peer* fique inacessível caso esteja protegido por um *firewall* e a porta não esteja liberada.

5.7 Conclusão

Neste capítulo foi apresentada a proposta de uma infra-estrutura de e-mail fundamentada em um sistema descentralizado, sem um servidor central de armazenamento de mensagens e dados dos usuários, os quais são autenticados utilizando uma estrutura de chaves pública/-privada disponibilizados pela rede PGP. As mensagens são enviadas obedecendo o conceito *PULL*, onde os transporte dos arquivos de e-mails são realizados diretamente entre o remetente e o receptor. Uma das vantagens do sistema proposto é o uso do disco local do remetente para armazenamento, evitando tráfego de mensagens desnecessárias, pois ela só é transmitida quando solicitada pelo destinatário.

Uma das vantagens do sistema de e-mail baseado no conceito *PULL* é que a mensagem só trafega na rede caso o usuário faça a solicitação do *download*, sendo que na abordagem *PUSH* todas as mensagens são trafegadas e recebidas pelo servidor receptor. Uma outra vantagem é o uso do disco local do remetente para armazenamento.

No *p2pMTA* há a necessidade do recurso da DHT que pode não ser acessível em um determinado momento caso ocorra um problema na rede, mesmo que os serviços da *OpenDHT* tenham uma rede com vários servidores na Internet.

Em relação ao sistema convencional, nesta arquitetura não há a centralização dos dados dos usuários e nem um servidor central para armazenar as mensagens. A verificação do conteúdo do corpo de uma mensagem só pode ser realizado após o *download* e decifragem da mesma, sendo assim algumas técnicas de filtros de vírus e *spam* somente poderão ser realizadas após o recebimento da mensagem.

A seguir, no próximo capítulo, é descrito e comentado alguns trabalhos relacionados, citando algumas características e comparando com o sistema proposto.

Capítulo 6

Trabalhos correlatos

6.1 Introdução

Algumas pesquisas e estudos foram realizados sobre elaboração de infra-estrutura de e-mail utilizando redes P2P, visando melhorar a centralização de dados e mensagens, ponto único de falha, facilidade em escalabilidade, entre outras características. Serão apresentados alguns estudos sobre viabilidade de uso das redes P2P num sistema de e-mail para melhoria de alguns problemas.

6.2 *Secure and Resilient Peer-to-Peer E-Mail*

Esta pesquisa [Kangasharju et al., 2003] propõe mudanças na arquitetura dos sistemas de e-mail, devido a alguns problemas relacionados pelo autor:

- ponto único de falha: quando um servidor de mensagens falha, o usuário não poderá ler suas mensagens;
- armazenamento: com o crescimento de mensagens com arquivos anexados com imagens gráficas, arquivos multimídias, documentos; a média de armazenamento de mensagens cresce exponencialmente a cada ano;
- servidor sobrecarregado: muitos servidores estão sobrecarregados pelo processamento de um grande número de mensagens;
- listas de e-mails (*Mailing Lists*): listas de e-mails com milhares de destinatários resultam em grande número de mensagens duplicadas.

Explora-se a implementação de um serviço de e-mail, sem um servidor, sobre uma estrutura *peer-to-peer* baseada em DHT. O sistema requer um serviço externo baseado em DHT que consiste em um grande número de computadores chamados de nós ou *peers*. O sistema é composto por um conjunto de nós e *User Agents* (UA). O sistema de nós são os computadores que utilizam um substrato DHT. A função dos nós do sistema é fornecer uma persistência para as mensagens que estão no trânsito do remetente ao receptor. Os UAs são os aplicativos de leitura de mensagens usados pelos clientes.

Cria-se um certificado através de um serviço externo para associar o endereço de e-mail a uma chave pública. O envio é realizado criando uma chave de sessão, que é cifrada utilizando a chave pública do destinatário. A cifragem é realizada somente no corpo da mensagem e juntamente com cabeçalho é postada em um substrato da DHT. Quando o usuário for verificar o *inbox*, ele somente verá o cabeçalho das mensagens não lidas desde o último acesso. Inicialmente, é parecido com o sistema POP: quando o usuário solicitar a visualização do corpo da mensagem e fizer o *download*, esta mensagem será apagada do *peer* que a mantém. O substrato da DHT armazena 3 tipos de objetos: certificado do endereço de e-mail, corpo da mensagem do e-mail e os *inboxes* (caixas postais).

Alguns problemas nos serviços de mensagens convencionais são resolvidos da seguinte forma:

- **Lista de e-mails** (*Mailing Lists*): o usuário solicita a inscrição na lista e recebe uma chave pública. Da mesma forma que no envio para um único destinatário, usa-se a chave de sessão que é cifrada com a chave pública da lista e depositada no *inbox* do destinatário.

- **Segurança** (Vírus e *Spam*): a checagem é feita após o *download* da mensagem, pois somente depois de ser decifrado é que se tem acesso ao corpo da mensagem. No caso do *spam*, faz-se o mesmo processo, consultando uma lista de *spammers* conhecidos. Também pode ser feito o controle recebendo a mensagem somente de chaves autorizadas.

A solução baseada na DHT elimina o ponto único de falha dos servidores atuais e reduz o processamento, bem como minimiza os espaços armazenados no caso de envio para uma lista. Como a mensagem armazenada é cifrada, somente o destinatário pode fazer a leitura desta mensagem. As mensagens são enviadas para uma estrutura DHT e a checagem de vírus é realizada apenas quando a mensagem é recebida, gerando assim tráfego no sistema, mesmo no caso de mensagens indesejadas.

6.3 *E-Mail Services on Hybrid P2P Networks*

Neste estudo [Zhao et al., 2004] foi utilizado um modelo híbrido de redes P2P, isto é, tendo um servidor central para autenticar e localizar outros *peers*.

Para oferecer o serviço de e-mail baseado em um sistema P2P, algumas características devem ser consideradas:

- emitir e receber mensagens no estilo P2P, ao invés de um único servidor de e-mail centralizado;

- fornecer um protocolo apropriado para suportar o serviço de correio entre usuários da rede P2P e da parte externa à rede, isto é, compatível ao protocolo de e-mail com servidor centralizado convencional.

A proposta consiste em desenvolver uma solução através de um sistema P2P híbrido, pois alguns recursos e informações estão centralizados. O sistema híbrido foi escolhido pelo fato de ser mais popular e maduro em alguns sistemas como o JXTA (Seção 4.5.4) e o *OpenNap* (Seção 4.5.1). O ambiente de rede proposto tem três serviços de e-mail diferentes: serviço inter-comunidade, serviço intra-comunidade e o serviço fora do ambiente P2P.

Os serviços inter-comunidade e o intra-comunidade são prestados dentro da comunidade P2P. Para comunicar com destinatários fora da comunidade P2P, usa-se um nó especial dentro da rede denominado *broker node*. Para enviar uma mensagem, primeiramente envia-se um $SENDING\_REQUEST$ (incluindo o remetente, o destinatário e um *timestamp*) para o seu super-nó. Se a mensagem for para fora da rede P2P, ele a formata no padrão RFC 2822 [Resnick, 2001] e a encaminha para o *broker node*. Se for para a rede P2P e fora da sua comunidade, o super-nó encaminha para outros super-nós. Se for local (da mesma comunidade) ele faz a entrega imediata, caso o destinatário esteja disponível.

A implementação segue alguns esquemas:

- pseudo-cooperativo: um super-nó serve como servidor centralizado como no caso dos sistemas atuais cliente/servidor, exceto no caso de existirem vários super-nós. As mensagens são armazenadas nestes super-nós.

- cooperativo simples: os super-nós não armazenam cópias e nem transferem mensagens para os nós destinatários. Quando um usuário envia uma mensagem, ele envia primeiramente um $SENDING\_REQUEST$ para o super-nó. Se o nó destino estiver disponível, o super-nó informa o IP do destinatário ao remetente, e este entrega a mensagem diretamente. Se estiver indisponível, o super-nó calcula um *hash* $H(s, r)$ onde s é o emissor, r é o receptor e atribui um tempo que ficará disponível e replica a outro nó escolhido pelo super-nó, e que é chamado de *transferrer*. Quando o receptor acessa a comunidade o super-nó informa o *transferrer* para enviar a mensagem ao destinatário, caso o *transferrer* falhe, o super-nó escolhe outro; quando é concluído, envia-se uma notificação de $SENDER\_OK$.

- cooperativo avançado: neste esquema o super-nó mantém uma cópia da lista do *inbox e outbox* das mensagens enviadas e recebidas.

Em uma rede P2P híbrida os super-nós são relativamente estáveis, porém passíveis de ficarem indisponíveis. Quando isto acontece, uma possível escolha de um novo super-nó é através de votação, só que este não conterá nenhuma informação, para isso algumas recuperações são executadas:

- **Recuperação dos nós locais:** ao replicar um *inbox*, o nó armazena também os metadados, incluindo a informação do identificador do *inbox*, do remetente e do receptor, o *timestamp* e a marca de leitura. Na recuperação, o novo super-nó reúne todos estes metadados dos nós regulares em sua comunidade, reconstruindo toda a informação necessária. Para o esquema cooperativo simples, o super-nó reconstruirá as tabelas de replicação e a tabela de emissão da notificação. Para esquema cooperativo avançado, os super-nós tem que reconstruir a informação do *inbox* e do *outbox* para cada nó.

- **Recuperação dos outros super-nós:** para o serviço da inter-comunidade, há trocas de informações entre super-nós. No esquema cooperativo simples, os novos super-nós solicitarão as informações a outros super-nós da inter-comunidade em sua comunidade local.

A pesquisa aborda uma proposta de um sistema de e-mail baseado numa rede P2P híbrida, utilizando um super-nó como se fosse um servidor central e também como um ponto de ligação para a estrutura convencional de e-mail.

6.4 *Experiences in Building and Operating ePOST*

Em outra pesquisa, o *ePOST* [Mislove et al., 2006, Mislove and Post, 2003] apresenta um projeto de e-mail sem um servidor central de e-mail. Os principais objetivos são alavancar um projeto seguro, escalável e altamente maleável. A implementação é sobre o *POST* [Mislove and Post, 2003, Mislove and Post, 2003] que é um sistema P2P colaborativo de mensagens que utiliza recursos das estações de trabalho dos usuários. O *POST* é um sistema implementado sobre o *PAST* [Druschel and Rowstron, 2001], um serviço baseado no serviço de DHT do *Pastry* (Seção 4.6).

Cada cliente *ePOST* funciona como um programa (*daemon*) rodando na estação de trabalho do usuário, habilitando os serviços SMTP e IMAP, permitindo assim que o usuário utilize programas convencionais de cliente de e-mail. Para participar do sistema é necessário cadastrar um usuário em uma página *web* específica. As mensagens recebidas de um cliente de e-mail são analisadas gramaticalmente e os componentes MIME da mensagem (corpo de mensagem e alguns anexos) são armazenados como mensagens separadas no POST, com isso cada anexo é armazenado somente uma vez.

Os componentes das mensagens são primeiramente inseridos no ambiente *POST* pelo *ePOST*, depois é enviada uma notificação ao receptor. Se as mensagens forem enviadas ou reenviadas por usuários diferentes, os dados originais da mensagem não necessitam ser armazenado outra vez. A entrega do e-mail é realizada usando o serviço de notificação *POST*. O remetente constrói inicialmente uma mensagem de notificação que contém a informação básica do cabeçalho, tal como os nomes do remetente e do receptor, do assunto e de uma referência ao corpo e aos anexos da mensagem; só depois solicita ao serviço *POST* local para entregar essa notificação a cada um dos receptores.

As mensagens são armazenadas como *metadados*; cada pasta de e-mail é representada por um registro (*log*) *POST*. Cada entrada no *log* representa mudanças no estado da pasta associada, como adição e deleção de mensagens. Além disso, o registro pode somente ser escrito pelo seu proprietário e seu índice pode estar cifrado. Em seguida, descreve-se um registro do *log* que representa uma inserção de uma mensagem de e-mail na pasta *inbox* de um usuário. Um registro de inserção do e-mail contém o índice do cabeçalho MIME da mensagem, do identificador da mensagem e de sua chave de decifragem, e de uma assinatura do remetente, que cifra com a chave pública do receptor.

6.5 Conclusão

Nota-se que as pesquisas buscam o conceito de sempre enviar as mensagens para armazenamento seja num sistema como no *ePost* [Mislove et al., 2006] que armazena na estrutura *POST* que é baseada numa estrutura DHT do *Pastry* ou na pesquisa de *Kangasharju* [Kangasharju et al., 2003] onde são enviados e alocados numa estrutura DHT. Na pesquisa de *Zhao* [Zhao et al., 2004], antes de enviar uma mensagem, o nó remetente faz uma consulta ao super-nó para verificar se o destinatário está disponível, e caso esteja *online*, envia a mensagem direto ao destinatário; caso contrário, pode enviar a um super-nó que é responsável por fazer a entrega no futuro.

Pode-se verificar que nas pesquisas do *Kangasharju* [Kangasharju et al., 2003] e do *ePOST* [Mislove et al., 2006, Mislove and Post, 2003] utiliza-se a DHT como base de armazenamento das mensagens. A maioria das DHTs atuais suporta apenas o registro de informações

pequenas, tornando inviável o registro de anexos de grandes dimensões, e para armazenar os arquivos na DHT tem que fracionar em várias partes pequenas e a mesmo tempo cifrá-las individualmente, o que pode causar alto processamento de envio e recebimento de um e-mail. O uso da DHT no *p2pMTA* é somente para envio de notificações e para armazenamento dos descritores.

No *ePOST* há a necessidade de registrar um usuário em uma página *web* específica, que é usado para a autenticação. No *p2pMTA* é necessária a criação de um endereço de e-mail qualquer associado a um par de chaves gerado pelo PGP. Na proposta de *Kangasharju* também há uma associação do endereço de e-mail a um par de chaves público/privado para cifrar as mensagens e na pesquisa de *Zhao*, os dados dos usuários são centralizados nos super-nós tanto para autenticação quanto para a sua localização. Neste caso, a existência de um super-nó facilita a integração com o sistema de mensagem convencional, mas cria um ponto frágil para o sistema.

No próximo capítulo é demostrada a implementação e alguns testes do *p2pMTA*, onde são apresentadas algumas medidas e suas avaliações dos dados fornecidos pelo sistema convencional de e-mail e pelo aplicativo do modelo proposto.

Capítulo 7

Implementação e Avaliação

7.1 Introdução

A arquitetura proposta no capítulo 5 foi implementada em um protótipo usando componentes existentes, conhecidos como *Commercial Off-The-Shelf* (COTS) [Dean and Vigder, 1997] e a linguagem *Perl*. Este capítulo trata das tecnologias e recursos utilizados para desenvolvimento do protótipo e também apresenta alguns resultados dos testes realizados.

7.2 Tecnologias utilizadas

Uma das funcionalidades preservadas no desenvolvimento do protótipo foi manter a compatibilidade com os clientes de e-mail convencional como *Mozilla Thunderbird* (`www.mozilla.com`) ou *Microsoft Outlook Express* (`www.microsoft.com`). Assim, o usuário não necessita instalar um novo aplicativo para ler ou enviar e-mails. Para o funcionamento, ele terá que inicializar o serviço *p2pMTA* (aplicativo da proposta). Outro aspecto previsto é que o sistema seja compatível com várias plataformas.

Plataforma. A plataforma escolhida para o desenvolvimento do protótipo foi o *Linux* devido à facilidade de instalação e configuração de aplicativos e ferramentas para desenvolvimento. Os sistemas operacionais baseados no *Linux*, como o *Ubuntu*, têm como facilidade a disponibilização de muitos pacotes de aplicativos e serviços. O serviço proposto funciona em qualquer plataforma, desde que as ferramentas, serviços e aplicativos necessários estejam instalados.

Linguagem de Programação. A linguagem escolhida para desenvolvimento dos aplicativos foi o *Perl* [Wall et al., 2000], por ser uma linguagem versátil, pois permite trabalhar interagindo com comandos e funções do sistema operacional e tem as funcionalidades para desenvolver aplicativos e interfaces para o protocolo HTTP, além de ser uma linguagem multi-plataforma.

Cliente de E-mail (MUA). Para realização dos testes foram utilizados os clientes de e-mail tradicionais como o *Mozilla Thunderbird* e o *Microsoft Outlook Express*, mas pode-se usar qualquer cliente que obedeça aos padrões definidos pelas RFCs 2821, 2822, 1939 e 3501.

OpenPGP e GnuPG. Em 1998 foi criado um padrão aberto para PGP, o *OpenPGP* que é definido pela RFC 2440 [Callas et al., 1998]. Para executar este serviço foi utilizado o *GnuPG* (`www.gnupg.org`) que é um projeto baseado no GNU (`www.gnu.org`), isto é, de distribuição e utilização gratuita do OpenPGP.

Hash SHA1. O *SHA1SUM* foi o aplicativo escolhido para gerar o *hash* SHA1 [Eastlake and Jones, 2001] do arquivo da mensagem e dos endereços de e-mail. O *hash* apresentado foi utilizado para criar um endereço e nome do arquivo único e também para checar a integridade das mensagens disponibilizadas pelo remetente.

Serviço HTTP. Para a disponibilização dos arquivos foram utilizados os serviços do protocolo HTTP, devido a sua simplicidade e por ser um dos serviços mais comuns na Internet. Na implementação do protótipo foi instalado o *Lighttpd* (`www.lighttpd.net`), um servidor simples, porém com todos os recursos necessários para criação do protótipo.

Servidores de E-mail SMTP e POP. O *Java E-mail Server* (JES) (`www.ericdaugherty.com/java/mailserver`) é um servidor de e-mail que disponibiliza os protocolos SMTP e POP3, respondendo nas portas 25 e 110 respectivamente. Para que somente o usuário local possa ter acesso, o serviço foi disponibilizado apenas para *localhost*. Para manter a integração com o cliente de e-mail convencional, utilizado atualmente, não foi realizada nenhuma alteração nos protocolos SMTP e POP3.

Rede DHT. Para viabilização dos serviços DHT foi utilizado o *OpenDHT* (Seção 4.6.2) e para as funções de *put, get, remove* foram empregadas os aplicativos disponibilizados pelos desenvolvedores do DHT *Bamboo* [Rhea, 2007].

7.3 Testes

Para comparar o sistema da proposta ao sistema convencional foram realizados alguns testes, que consistiram no envio de 1000 mensagens para um único receptor, utilizando mensagens com o corpo vazio e um arquivo em anexo. Dois tipos de arquivos e quatro tamanhos foram utilizados: textos puros (ASCII) ou com dados binários, com tamanhos de anexo de 1K, 10K, 100K, ou 1Mbyte.

Os primeiros ensaios mostraram uma grande variabilidade nos tempos de acesso ao serviço remoto *OpenDHT*. A fim de assegurar a estabilidade de tempo de acesso necessária, os experimentos foram efetuados usando uma DHT instalada localmente (servidor Bamboo instalado na rede local). Os mesmos testes foram repetidos em um sistema convencional SMTP, para comparação.

7.3.1 Ambiente dos Testes

Para realizar os testes foram necessários vários equipamentos utilizados como clientes e/ou como servidores. Para os clientes foram usados computadores com processadores *Pentium IV* de 1,6 Ghz com 512 MB de memória e 40 GB de disco rígido. A instalação do serviço SMTP foi feita em um computador com processador *Pentium D* 2,8 Ghz com 2,0 GB de memória e 100 GB de disco rígido.

Para resolver o problema de latência do serviço DHT, foi instalada a DHT *Bamboo* em um computador com processador Pentium IV 2,0 Ghz com 1,0 GB de memória e 100 GB de disco rígido. O sistema operacional utilizado em todos os computadores foi *Linux Ubuntu Desktop* versão 7.04. Todos os computadores estavam conectados em uma rede local através de um *switch*.

Ambiente SMTP

A figura 7.1 ilustra o ambiente configurado para realização dos testes com o protocolo convencional SMTP. Em (1) ocorre a comunicação realizada no envio da mensagem de um cliente para um servidor SMTP; em (2) o SMTP remetente faz uma consulta no DNS para localizar o IP servidor do domínio do destinatário e em (3) o SMTP remetente estabelece a comunicação com o servidor destino, e entrega a mensagem.

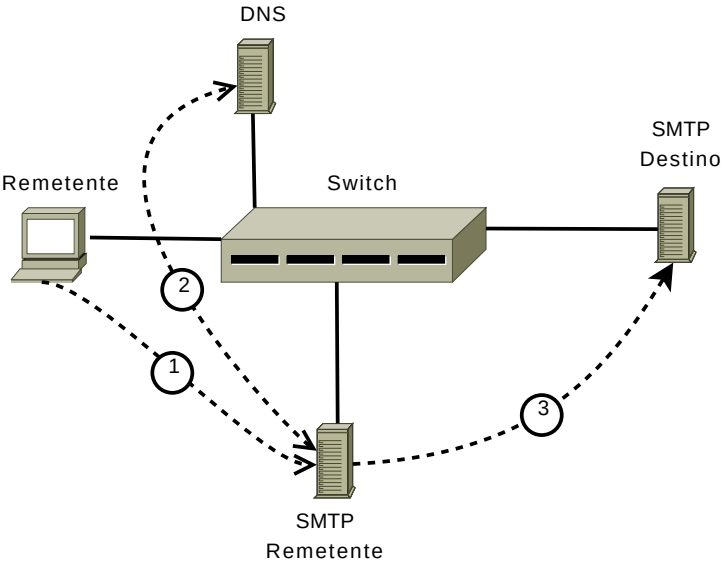

Figura 7.1: Ambiente SMTP para testes

Ambiente p2pMTA

A figura 7.2 apresenta o ambiente usando a infra-estrutura *p2pMTA*. Em (1) e (3) estão as comunicações realizadas pelos clientes remetentes e destinatários com a estrutura DHT. São realizadas algumas operações de postagem, consulta e remoção de notificações. Em (2) está a busca da chave pública do destinatário; o item 4 representa o *download* da mensagem.

7.4 Avaliação

A figura 7.3 mostra o tempo necessário para receber as 1000 mensagens no *p2pMTA*, usando o protocolo convencional SMTP. O sistema SMTP tem um resultado melhor para pequenas mensagens, mas tem um aumento de tempo sensível para mensagens maiores. No *p2pMTA* o aumento do tempo é mais suave.

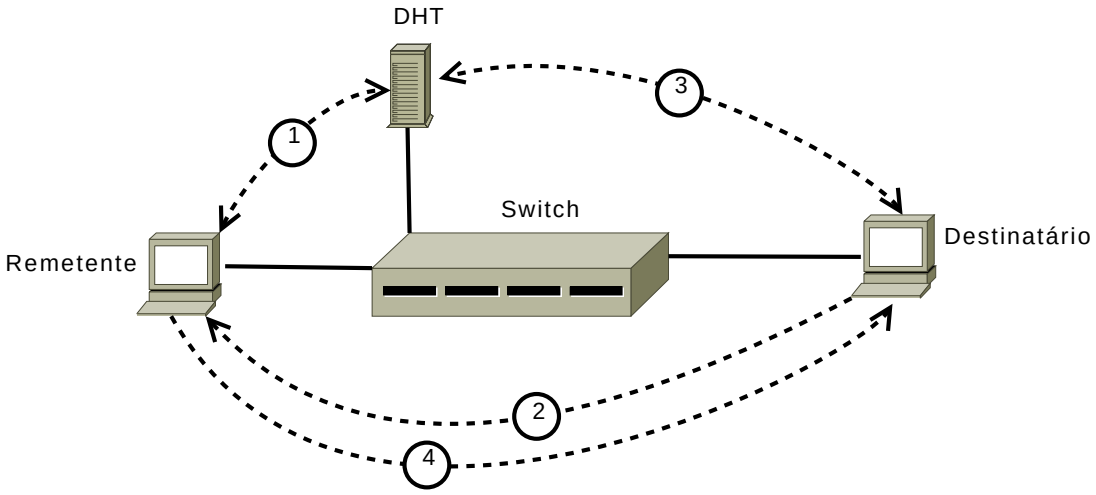

Figura 7.2: Ambiente p2pMTA para testes

O tempo elevado no *p2pMTA* deve-se principalmente às consultas na DHT e ao uso do processamento para a criptografia e *hash*, procedimentos aplicados a cada mensagem. Além disso, *p2pMTA* tem os efeitos da compressão proporcionada pelo serviço GPG: os tempos para transferência para mensagens de texto são inferiores aos tempos de transferência para mensagens binárias, porque as mensagens com anexo em formato texto são mais compressíveis.

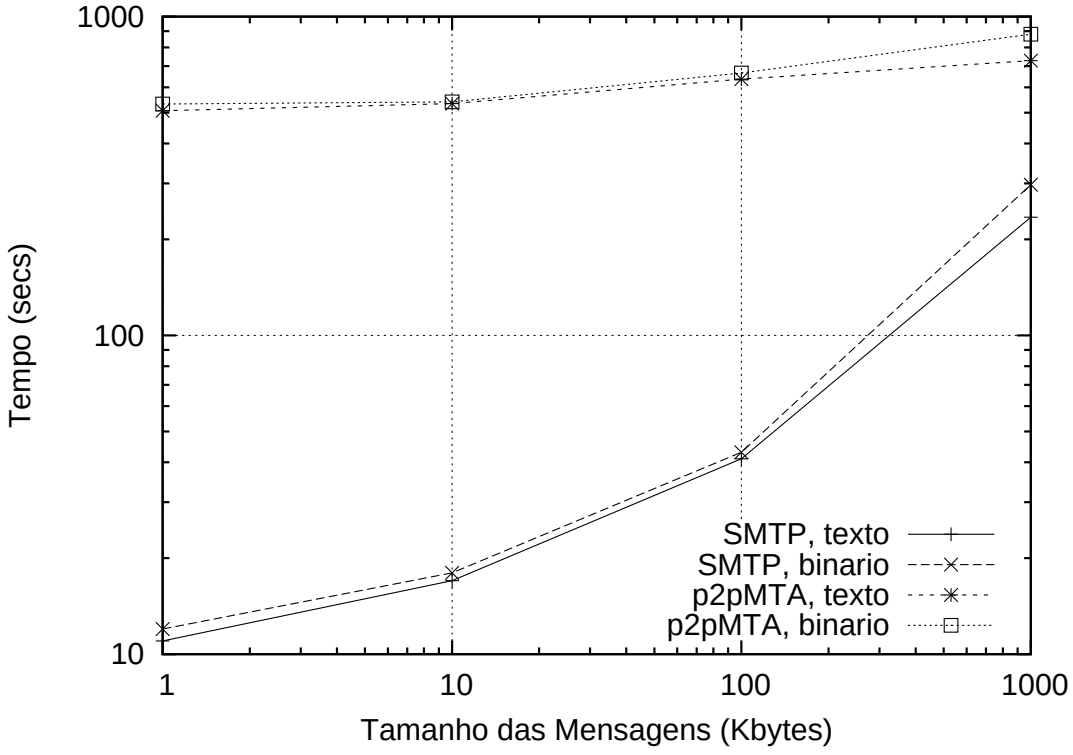

Figura 7.3: Tempo para receber 1000 mensagens

A figura 7.4 compara o tráfego de rede entre os sistemas *p2pMTA* e o SMTP. O gráfico mostra as somas dos tráfegos que entram e saem de um receptor. Os tráfegos de rede usados

pelo *p2pMTA* são menores que os sistemas convencionais, pois as mensagens são cifradas e compactadas. Os efeitos são mais visíveis em arquivos de mensagens de texto maiores, porque são mais compressíveis.

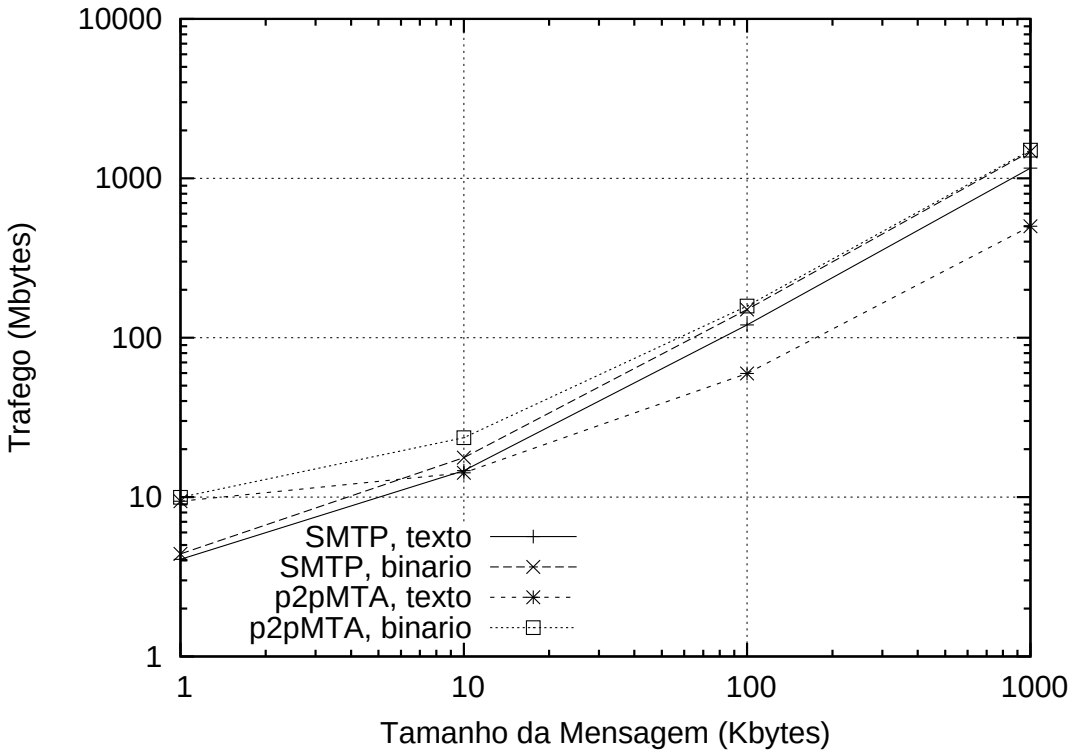

Figura 7.4: Tráfego de Rede para receber 1000 mensagens

Nos testes anteriores, o receptor aceita todas as mensagens enviadas a ele. Se este decidir ignorar algumas delas (porque podem ser *spams*), o seu tráfego de rede deve reduzir proporcionalmente. Essa avaliação foi feita usando uma experiência na qual o remetente produz 1000 mensagens contendo um arquivo binário de 10K anexado.

O receptor aceita apenas uma determinada percentagem p das mensagens enviadas a ele. A figura 7.5 apresenta o tráfego observado no cliente, de acordo com p, e mostra também o tráfego correspondente ao receptor no sistema SMTP equivalente. A redução é bem visível: se o receptor aceita menos de 80% das mensagens recebidas, o seu tráfego de rede será inferior ao equivalente receptor SMTP. Note-se que o tráfego no receptor nunca é nulo, porque ele sempre consulta a DHT para receber/enviar notificações.

7.5 Conclusão

Neste capítulo foi apresentada a implementação do *p2pMTA*. Alguns detalhes para criação do protótipo da arquitetura com linguagem de programação, serviços para armazenamento de transferências das mensagens, aplicativos para cifragem/decifragem foram descritas. Em seguida, foram realizados alguns testes comparando a arquitetura *p2pMTA* e o protocolo convencional SMTP.

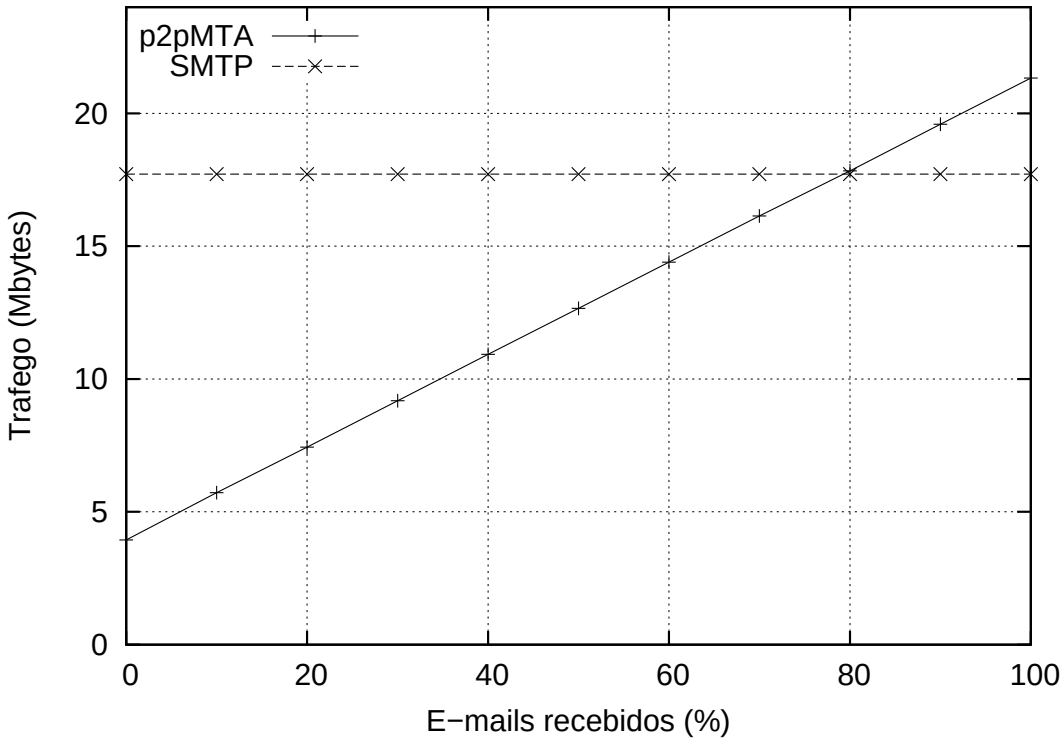

Figura 7.5: Tráfego de Rede recebido pelo receptor

Capítulo 8

Conclusão

Este trabalho apresenta uma arquitetura alternativa para sistemas de e-mail distribuído para a Internet. Nesta proposta, as mensagens de e-mails são armazenadas nos remetentes e somente a notificação é postada em um serviço global disponibilizado pela DHT. Os receptores checam periodicamente a DHT para verificar novas notificações; as mensagens existentes podem ser ignoradas, ou quando solicitadas, são trazidas diretamente do remetente correspondente ou de algum outro membro do *peer group* do remetente.

As mensagens são replicadas entre os membros de um *peer group* para disponibilizar as mensagens quando o remetente estiver indisponível. Esta arquitetura foi inspirada nos programas de troca de arquivos na rede *peer-to-peer*, onde um usuário localiza uma informação e faz o *download* somente se a informação for o que está procurando.

Uma das contribuições desta arquitetura é manter o armazenamento das mensagens no disco local do remetente e dos membros de um grupo, constituindo um sistema colaborativo, que é uma das características dos sistemas *peer-to-peer*. Além disso, se um destinatário decide não receber um e-mail, este não será transferido do remetente, economizando consequentemente, a banda de rede. A arquitetura foi implementada em um protótipo utilizando componentes COTS, e alguns experimentos realizados para estimar alguns dados, em comparação com um sistema convencional.

O trabalho possibilitou as publicações dos artigos completo no *23rd Annual ACM Symposium on Applied Computing - SAC'08* [Kageyama et al., 2008a] e no *11th IEEE International Conference on Computational Science and Engineering - CSE'08* [Kageyama et al., 2008b].

Alguns **trabalhos futuros** podem ser realizados para melhoria e complementos da arquitetura proposta, como:

1. **Usuários protegidos por um *Firewall***: para o uso deste serviço é necessário o acesso à porta 80 das máquinas que disponibilizam os arquivos de mensagens. Caso um usuário esteja atrás de um *firewall* é possível que o receptor não consiga fazer o *download* de uma mensagem, devido à filtragem dessa porta. O mesmo se aplica a outras portas.

2. **IP não público**: as mensagens que estiverem em um *peer* que tiver um IP não público não serão alcançadas, devido ao mesmo problema.

3. **Integração com Sistemas de E-mails Atual**: para que as mensagens possam ser roteadas para um sistema convencional é necessária a criação de um servidor que diferencie as mensagens de uma sistema P2P daquelas direcionadas a um sistema de e-mail convencional.

4. **Fragmentar busca**: para agilizar a busca de uma mensagem, pode-se criar um método de *download* de fragmentos entre os vários *peers* que mantêm réplicas da mesma, tal qual ocorre nos serviços de compartilhamentos de arquivos.

5. **Relação de confiança dos membros**: criar mecanismos dinâmicos para relacionar os membros de um grupo de forma automática.

Para a implementação dos itens 1, 2 e 3 pode-se criar um servidor central (*relay*) que será responsável por rotear mensagens de um sistema convencional para o *p2pMTA*. Ele também serviria com um *proxy* para os *peers* que estiverem protegidos por um *firewall* e não tiverem um IP público.

Referências Bibliográficas

[Adams and S.Farrell, 1999] Adams, C. and S.Farrell (1999). Internet X.509 Public Key Infrastructure - Certificate Management Protocols. IETF RFC-2510.

[Allman et al., 2007] Allman, E., Callas, J., Delany, M., Libbey, M., Fenton, J., and Thomas, M. (2007). DomainKeys Identified Mail (DKIM) Signatures. IETF RFC-4871.

[Androutsellis-Theotokis and Spinellis, 2004] Androutsellis-Theotokis, S. and Spinellis, D. (2004). A Survey of Peer-to-Peer Content Distribution Technologies. *ACM Computing Surveys*, 36(4).

[B. Ramsdell, 2004a] B. Ramsdell, E. (2004a). Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.1 - Certificate Handling. IETF RFC-3850.

[B. Ramsdell, 2004b] B. Ramsdell, E. (2004b). Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.1 - Message Specification. IETF RFC-3851.

[Balenson, 1993] Balenson, D. (1993). Privacy Enhancement for Internet Electronic Mail: Part III: Algorithms, Modes, and Identifiers. IETF RFC-1423.

[Barcellos and Gaspary, 2006] Barcellos, M. P. and Gaspary, L. P. (2006). *Segurança em Redes P2P: Princípios, Tecnologias e Desafios*, volume 1, pages 211–260. SBC.

[Berners-Lee et al., 1994] Berners-Lee, T., Masinter, L., and McCahill, M. (1994). Uniform Resource Locators (URL). IETF RFC-1738.

[Bouwsma and Visser, 2004] Bouwsma, J. and Visser, R. (2004). Distributed E-mail - *A distributed asynchronous message service, is that possible? Master Education SNB*.

[Brookshier et al., 2002] Brookshier, D., Govoni, D., Soto, J., and Krishnan, N. (2002). *JXTA: Java P2P Programming*. Sams Publishing.

[Callas et al., 1998] Callas, J., Donnerhacke, L., Finney, H., and Thayer, R. (1998). OpenPGP message format. IETF RFC-2440.

[Clarke et al., 2001] Clarke, I., Sandberg, O., Wiley, B., and Hong, T. W. (2001). Freenet: A Distributed Anonymous Information Storage and Retrieval System. In *International Workshop on Designing Privacy Enhancing Technologies*, pages 46–66, New York, NY, USA. Springer-Verlag New York, Inc.

[Cohen, 1987] Cohen, F. (1987). *Computer Viruses: Theory and Experiments*, volume 6, pages 22–35. Elsevier Advanced Technology Publications, Oxford, UK, UK.

[Crispin, 2003] Crispin, M. . (2003). Internet Message Access Protocol Version 4 rev1. IETF RFC-3501.

[Dean and Vigder, 1997] Dean, J. C. and Vigder, M. R. (1997). System Implementation Using Commercial Off-The-Shelf (COTS) Software. Technical Report NRC No. 40173, Communications Office, Institute for Information Technology, National Research Council of Canada, Ottawa, Ontario, Canada K1A 0R6.

[Ding et al., 2005] Ding, C., Nutanong, S., and Buyya, R. (2005). *Peer-to-Peer Networks for Content Sharing*. Idea Group Pub.

[Douceur, 2002] Douceur, J. R. (2002). The Sybil Attack. In *Peer-to-Peer Systems: First International Workshop, IPTPS 2002. Revised Papers*, pages 251–260. Springer Berlin / Heidelberg.

[Druschel and Rowstron, 2001] Druschel, P. and Rowstron, A. (2001). PAST: A Large-scale, Persistent Peer-to-peer Storage Utility. In *Proc. HotOS VIII*, pages 75–80.

[Dumitriu et al., 2005] Dumitriu, D., Knightly, E., Kuzmanovic, A., Stoica, I., and Zwaenepoel, W. (2005). Denial-of-service Resilience in Peer-to-peer File Sharing Systems. In *SIGMETRICS '05: Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 38–49, New York, NY, USA. ACM.

[Eastlake and Jones, 2001] Eastlake, D. and Jones, P. (2001). US Secure Hash Algorithm 1 (SHA1). IETF RFC-3174.

[Fanning, 2007] Fanning, S. (2007). Napster. Website. `http://www.napster.com`.

[Freed, 1996a] Freed, N. (1996a). Multipurpose Internet Mail Extensions (MIME) Part Five: Conformance Criteria and Examples. IETF RFC-2049.

[Freed, 1996b] Freed, N. (1996b). Multipurpose Internet Mail Extensions (MIME) Part Four: Registration Procedures. IETF RFC-2048.

[Freed and Borenstein, 1996a] Freed, N. and Borenstein, N. (1996a). Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies. IETF RFC-2045.

[Freed and Borenstein, 1996b] Freed, N. and Borenstein, N. (1996b). Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types. IETF RFC-2046.

[Gnutella, 2007] Gnutella (2007). Gnutella. Website. `http://www.gnutella.com`.

[Gray, 1995] Gray, T. (1995). Comparing Two Approaches to Remote Mailbox Access: IMAP vs. POP. Technical report. `http://www.imap.org/papers/imap.vs.pop.brief.html`.

[Hall, 1998] Hall, R. J. (1998). How to Avoid Unwanted Email. In *Communications of the ACM*, volume 41, pages 88–95, New York, NY, USA. ACM.

[Hambridge and Lunde, 1999] Hambridge, S. and Lunde, A. (1999). DON'T SPEW A Set of Guidelines for Mass Unsolicited Mailings and Postings (spam*). IETF RFC-2635.

[Harris, 2003] Harris, E. (2003). The Next Step in the Spam Control War: Greylisting. Technical report. `http://projects.puremagic.com/greylisting/whitepaper.html`.

[Hoffman, 1999] Hoffman, P. (1999). SMTP Service Extension for Secure SMTP over TLS. IETF RFC-2487.

[Hughes et al., 2005] Hughes, D., Coulson, G., and Walkerdine, J. (2005). Free Riding on Gnutella Revisited: The Bell Tolls? *IEEE Distributed Systems Online*, 06(6).

[Kageyama et al., 2008a] Kageyama, E., Maziero, C., and Santin, A. (2008a). A Pull-based E-mail Architecture. In *SAC'08: Proceedings of the 23rd Annual ACM Symposium on Applied Computing*, pages 468–472, New York, NY, USA. ACM.

[Kageyama et al., 2008b] Kageyama, E., Maziero, C., and Santin, A. (2008b). An experimental peer-to-peer e-mail system. In *CSE'08: Proceedings of the 11th IEEE International Conference on Computational Science and Engineering*, pages 0–0, Los Alamitos CA USA. IEEE Computer Society.

[Kalisk and Staddon, 1998] Kalisk, B. and Staddon, J. (1998). PKCS #1: RSA Cryptography Specifications Version 2.0. IETF RFC-2437.

[Kaliski, 1993] Kaliski, B. (1993). Privacy Enhancement for Internet Electronic Mail: Part IV: Key Certification and Related Services. IETF RFC-1424.

[Kangasharju et al., 2003] Kangasharju, J., Ross, K. W., and Turner, D. A. (2003). Secure and Resilient Peer-to-Peer E-Mail: Design and Implementation. In *P2P '03: Proceedings of the 3rd International Conference on Peer-to-Peer Computing*, page 184, Washington, DC, USA. IEEE Computer Society.

[Kent, 1993] Kent, S. (1993). Privacy Enhancement for Internet Electronic Mail: Part II: Certificate-Based Key Management. IETF RFC-1422.

[Klensin, 2001] Klensin, J. (2001). Simple Mail Transfer Protocol. IETF RFC-2821.

[Lindberg, 1999] Lindberg, G. (1999). Anti-Spam Recommendations for SMTP MTAs. IETF RFC-2505.

[Linn, 1993] Linn, J. (1993). Privacy Enhancement for Internet Electronic Mail: Part I: Message Encryption and Authentication Procedures. IETF RFC-1421.

[Lua et al., 2005] Lua, K., Crowcroft, J., Pias, M., Sharma, R., and Lim, S. (2005). A Survey and Comparison of Peer-to-peer Overlay Network Schemes. *Communications Surveys & Tutorials, IEEE*, pages 72–93.

[Lyon, 2006] Lyon, J. (2006). Purported Responsible Address in E-Mail Messages. IETF RFC-4407.

[Lyon and Wong, 2006] Lyon, J. and Wong, M. (2006). Sender ID: Authenticating E-Mail. IETF RFC-4406.

[Mello et al., 2005] Mello, E. R., Fraga, J. S., and Santin, A. (2005). O uso do SPKI/SDSI em redes P2P. In *Simpósio Brasileiro de Redes de Computadores Anais I Workshop de Peer-to-Peer - WP2P*, pages 49–60. SBC.

[Mislove and Post, 2003] Mislove, A. and Post, A. (2003). POST: A Secure, Resilient, Cooperative Messaging System. In *USENIX HotOS*.

[Mislove et al., 2006] Mislove, A., Post, A., HaeberRlen, A., and Druschel, P. (2006). Experiences in Building and Operating ePOST, a Reliable Peer-to-peer Application. In *EuroSys '06: Proceedings of the ACM SIGOPS/EuroSys European Conference on Computer Systems 2006*, pages 147–159, New York, NY, USA. ACM.

[Moore, 1996] Moore, K. (1996). MIME (Multipurpose Internet Mail Extensions) Part Three: Message Header Extensions for Non-ASCII Text. IETF RFC-2047.

[Myers and Rose, 2001] Myers, J. and Rose, M. (2001). Post Office Protocol - version 3. IETF RFC-1939.

[Newman, 1999] Newman, C. (1999). Using TLS with IMAP, POP3 and ACAP. IETF RFC-2595.

[OpenNap, 2007] OpenNap (2007). OpenNap: Open Source Napster Server. Website. `http://opennap.sourceforge.net`.

[Park and Hwang, 2003] Park, J. S. and Hwang, J. (2003). Role-based Access Control for Collaborative Enterprise in Peer-to-peer Computing Environments. In *SACMAT '03: Proceedings of the eighth ACM symposium on Access control models and technologies*, pages 93–99, New York, NY, USA. ACM.

[Peterson et al., 2006] Peterson, L., Bavier, A., Fiuczynski, M. E., and Muir, S. (2006). Experiences Building PlanetLab. In *Proceedings of the 7th USENIX Symposium on Operating System Design and Implementation (OSDI '06)*, Seattle, WA.

[Ratnasamy et al., 2001] Ratnasamy, S., Francis, P., Handley, M., Karp, R., and Schenker, S. (2001). A Scalable Content-Addressable Network. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 161–172, New York, NY, USA. ACM.

[Resnick, 2001] Resnick, P. (2001). Internet Message Format. IETF RFC-2822.

[Rhea, 2007] Rhea, S. (2007). The Bamboo Distributed Hash Table: A Robust, Open-Source DHT. Website. `http://bamboo-dht.org`.

[Rhea et al., 2005] Rhea, S., Godfrey, B., Karp, B., Kubiatowicz, J., Ratnasamy, S., Shenker, S., Stoica, I., and Yu, H. (2005). OpenDHT: a Public DHT Service and its Uses. In *SIGCOMM '05: Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 73–84, New York, NY, USA. ACM.

[Ripeanu, 2001] Ripeanu, M. (2001). Peer-to-Peer Architecture Case Study: Gnutella Network. In *Peer-to-Peer Computing, 2001. Proceedings. First International Conference on*, pages 99–100, Los Alamitos, CA, USA. IEEE Computer Society.

[Rocha et al., 2004] Rocha, J., Domingues, M., Callado, A., Souto, E., Silvestre, G., Kamienski, C., and Sadok, D. (2004). *Peer-to-Peer: Computação Colaborativa na Internet*, pages 3–46. SBC.

[Rowstron and Druschel, 2001] Rowstron, A. and Druschel, P. (2001). Pastry: Scalable, Distributed Object Location and Routing for Large-scale Peer-to-peer Systems. *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, 11:329–350.

[Sahami et al., 1998] Sahami, M., Dumais, S., Heckerman, D., and Horvitz, E. (1998). A Bayesian Approach to Filtering Junk E-Mail. In *Learning for Text Categorization: Papers from the 1998 Workshop*, volume 62. Madison, Wisconsin: AAAI Technical Report WS-98-05.

[Sandhu et al., 1996] Sandhu, R. S., Coyne, E. J., Feinstein, H. L., and Youman, C. E. (1996). Role-Based Access Control Models. volume 29, pages 38–47, Los Alamitos, CA, USA. IEEE Computer Society.

[Stoica et al., 2001] Stoica, I., Morris, R., Karger, D., Kaashoek, M., and Balakrishnan, H. (2001). Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. In *Proceedings of the 2001 SIGCOMM conference*, volume 31, pages 149–160. ACM Press New York, NY, USA.

[Wall et al., 2000] Wall, L., Christiansen, T., and Orwant, J. (2000). *Programming Perl*. O'Reilly.

[Waterhouse,] Waterhouse, S. JXTA Search Protocol Specification. Technical report, tech. report, Sun Microsystems, Palo Alto, Calif., 2001.

[Waterhouse, 2001] Waterhouse, S. (2001). JXTA Search: Distributed Search for Distributed Networks. *Sun Microsystems, Mai*.

[Wilson, 2002] Wilson, B. (2002). *JXTA*. New Riders Publishing, 1st edition.

[Wong and Schlitt, 2006] Wong, M. and Schlitt, W. (2006). Sender Policy Framework (SPF) for Authorizing Use of Domains in E-Mail - Version 1. IETF RFC-4408.

[Zhao et al., 2001] Zhao, B., Kubiatowicz, J., and Joseph, A. (2001). Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and Routing. *Computer Science Division*, 74.

[Zhao et al., 2004] Zhao, Y., Zhou, S., and Zhou, A. (2004). E-Mail Services on Hybrid P2P Networks. In *Grid and Cooperative Computing Conference - GCC*, pages 610–617.

[Zimmermann, 1995] Zimmermann, P. (1995). *The Official PGP User's Guide*. The MIT Press.