ROBERT PEREIRA PINTO

UM MODELO DE FILTROS COMPOSTOS PARA DETECÇÃO DE MENSAGENS MALICIOSAS

Belo Horizonte
04 de maio de 2006

Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Programa de Pós-Graduação em Ciência da Computação

UM MODELO DE FILTROS COMPOSTOS PARA DETECÇÃO DE MENSAGENS MALICIOSAS

Dissertação apresentada ao Curso de Pós-Graduação em Ciência da Computação da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

ROBERT PEREIRA PINTO

Belo Horizonte
04 de maio de 2006

UNIVERSIDADE FEDERAL DE MINAS GERAIS

FOLHA DE APROVAÇÃO

Um Modelo de Filtros Compostos para Detecção de Mensagens Maliciosas

ROBERT PEREIRA PINTO

Dissertação defendida e aprovada pela banca examinadora constituída por:

Ph. D. Wagner Meira Júnior – Orientador
Universidade Federal de Minas Gerais

Ph. D. Jussara Marques de Almeida – Co-orientador
Universidade Federal de Minas Gerais

Ph. D. Dorgival Olavo Guedes
Universidade Federal de Minas Gerais

Ph. D. Virgílio Augusto Fernandes Almeida
Universidade Federal de Minas Gerais

Ph. D. Lucila Ishitani
Pontifícia Universidade Católica de Minas Gerais

Belo Horizonte, 04 de maio de 2006

Resumo

O crescimento da Internet criou toda uma nova gama de aplicações, muitas das quais se caracterizam pela alta interatividade e comodidade, como é o caso das mensagens eletrônicas. Entretanto, esses novos modelos de interação também permitiram o surgimento de práticas ruins, onde os mecanismos são utilizados de forma abusiva. Um exemplo típico de tais práticas são as mensagens maliciosas, as quais se tornaram uma verdadeira praga virtual, dado o seu volume e variedade. Com o objetivo de minimizar essa situação diversas propostas de combate têm sido colocadas, entre as quais destacamos os filtros. Filtros são programas que analisam as mensagens sendo recebidas pelo servidor de correio eletrônico e assinalam as mensagens que parecem ser maliciosas. Esse processo de assinalamento pode empregar uma grande variedade de critérios, cada qual mensurando um aspecto que pode identificar a mensagem como maliciosa ou não. A completeza dos critérios ainda é um problema em aberto, ou seja, nenhuma solução corrente possui uma precisão de 100%. Mais ainda, é importante ressaltar que esses filtros são computacionalmente intensivos, e o custo de filtragem é proporcional à complexidade dos critérios, ao tamanho da mensagem e, obviamente, ao número de critérios empregados. Uma mensagem maliciosa pode satisfazer a vários critérios, o que caracteriza um desperdício de recursos computacionais, já escassos em grande parte dos serviços de correio eletrônico. Uma opção nesse caso é empregar apenas alguns dos critérios, reduzindo o custo de filtragem, mas com o desafio de manter a efetividade de detecção. Neste trabalho investigamos os compromissos entre o custo e o benefício desta abordagem através do modelo de filtros compostos. Também apresentamos uma estratégia gulosa de construção de tais filtros, além de avaliá-los utilizando cargas de trabalho realistas, onde pudemos verificar que essa é uma proposta promissora.

Abstract

The growth of the Internet created a whole new set of applications, most of them being characterized by the ease of use and interactivity. However, these new interaction models also allowed the raise of bad practices, where those novel mechanisms are used abusively. A typical example of such practices are the malicious messages or spams, which became a virtual prague, given their volume and variety. In order to minimize the spam-related problems, several proposals have been presented, including filters. Filters are programs that analyze the incoming messages to the mail server and identify those that seem to be malicious. This process may employ a large variety of criteria, each of them quantifying an aspect that may identify the message as being malicious or not. The completeness of these criteria is still an open problem, that is, no current solution has 100% precision. Further, we should note that filtering is a computationally expensive process, and the filtering cost is a function of the criterion complexity, the message lenght, and, obviously, the number of criteria employed. A malicious message may be detected by several criteria, characterizing a waste of computational resources, which are already escarce in most of the mail services. An alternate approach is to employ just some of the criteria, reducing the filtering cost, but facing the challenge of maintaining the detection effectiveness. In this work we investigate the trade offs of this last proposal through the composite filter model. We also present a greedy strategy for building such filters, and evaluated it using actual workloads, where we were able to verify that this is a promising solution.

À minha esposa, Elisângela, às minhas duas mães, Aparecida e Native, ao meu filho Robert e à Letícia

Agradecimentos

Agradeço às minhas duas mães, Aparecida e dona Native, pelos princípios, ensinamentos, cuidados, preocupações, carinhos e orações. Vocês foram fundamentais nesta minha jornada que vem de longa data. Minha tia e mãe Native me acolheu quando eu ainda não sabia nem mesmo falar e fez de mim uma pessoa determinada a vencer. Minha mãe Aparecida que nunca me abandonou, mesmo nos momentos mais difíceis, esteve ao meu lado, torcendo por mim e me confortando com seu grande amor de mãe.

Agradeço aos meus familiares por estarem sempre torcendo por mim. A vocês da casa do meu Tio Orlando. Agradeço a vocês, Timô, tia Lenir, Júnior, Lucilene, Robson, Xandão, Sérgio, tio Eustáquio, tia Neusir, Gustavo, Paola, Léo, Ramon e Fabrício pela força durante este período e por me mostrarem o verdadeiro valor de uma família.

Agradeço ao meu grande amigo Valter, pelos bons conselhos e paciência em me ouvir quando estava angustiado. Aos meus grandes amigos de graduação, Lucas Issa, Gisele, Isabela, Stênio, Lcrocha, Pablo, Juliano e Diego. Fico muito feliz em ter tido a oportunidade de fazer tão boas, importantes e, certamente, duradouras amizades.

Aos meus amigos do laboratório e-speed que me deram muita força e ajuda desde a iniciação científica. Agradeço ao Fabrício, Barroca, Coutinho, Macambira, Bruno Diniz, Bruno Grossi, Andrec, Zeniel, Tassni, Brut, Adrianoc, que de uma maneira ou de outra contribuíram para a minha caminhada. À Raquelcm pela torcida e preces. Ao Adrianov pelos conselhos e direcionamentos que foram fundamentais para a definição deste trabalho. Ao Luiz Henrique por ter me dado um empurrão nesta área de pesquisa. À dupla Fernando e Fabiano, que estiveram diretamente envolvidos no início deste meu mestrado.

Agradeço ao professor Virgílio, que me orientou no início do mestrado e me direcionou para esta frente de pesquisa. À Lucila, quem adimiro muito, e que teve grande responsabilidade pela minha aprovação no mestrado. À professora Jussara, por ter participado e feito questionamentos importantíssimos para o o fechamento deste trabalho. Um agradecimento especial ao professor Wagner Meira. Você esteve presente em todos os momentos durante a minha vida acadêmica. Não posso considerá-lo apenas um orientador, sua participação foi muito além disto. Sou extremamente grato ao Frota

pelo apoio no Cecom, o qual foi fundamental para a conclusão deste trabalho.

Um agradecimento aos meus sócios da Base2: Hugo, Juliano e Diego. A participação de vocês, a colaboração e a paciência durante os períodos mais complicados jamais serão esquecidos.

Ao meu filho, por ter trazido tantas alegrias para minha vida. Um agradecimento especial é para minha esposa e amada Elisângela. Sempre acreditou no meu pontencial, mesmo quando eu estava desacreditado você estava lá, me encorajando, sendo paciente, me dando apoio, me confortando, sendo simplesmente fundamental para o meu sucesso. Dedico esta vitória a você. Sem a sua presença em minha vida, este momento não teria o mesmo valor.

E agradeço acima de tudo à Deus, por ter me dado força e coragem para enfrentar todos os obstáculos que surgiram nesta minha caminhada. Foram os momentos difíceis que ele colocou à minha frente que me transformaram no que sou hoje. Esses momentos em que ele mesmo me ajudou a vencer e a provar, para mim mesmo, que sou capaz de superar meus próprios limites.

Sumário

1 Introdução **1**

1.1 Arquitetura de Sistemas de Correio Eletrônico 2

1.2 Mensagens Maliciosas 5

1.2.1 Impacto das Mensagens Maliciosas 7

1.3 Motivação e Contribuição do Trabalho 8

1.3.1 Contribuições 8

1.4 Organização do Trabalho 9

2 Trabalhos Relacionados **10**

2.1 Introdução 10

2.2 Técnicas de Combate 10

2.2.1 Técnicas de Combate *Pre-Acceptance* 10

2.2.2 Técnicas de Combate *Post-Acceptance* 13

2.3 Trabalhos de Caracterização 16

2.4 Considerações 17

3 O Modelo de Filtro Composto **18**

3.1 Introdução 18

3.2 Modelo Tradicional 18

3.3 Modelo com Priorização 20

3.4 Modelo de Filtro Composto 22

3.4.1 Representação do MFC 24

3.5 Exemplificação 26

3.6 Considerações 28

4 Estratégia de Montagem do Filtro Composto **30**

4.1 Introdução 30

4.2 Problema de Classificação 30

4.3 Algoritmo 34

4.3.1 Escolha de Componentes para Filtro 34

4.3.2 Escolha das Mensagens de Treinamento 35
4.3.3 Preparação da Base de Treinamento 36
4.3.4 Geração da Árvore de Detecção 36
4.3.5 Função de Seleção de Componentes 39
4.4 Considerações 41

5 Estudo de Caso **43**
5.1 Introdução 43
5.2 O *SpamAssassin* 44
5.2.1 Testes Implementados no *SpamAssassin* 45
5.3 Componentes do Filtro Composto 47
5.4 Estudo dos Componentes 48
5.5 Considerações 51

6 Avaliação Experimental **52**
6.1 Introdução 52
6.2 Métricas de Avaliação 53
6.3 Carga de Trabalho 53
6.4 Metodologia 55
6.5 Resultados 56
6.5.1 Filtro Composto X *SpamAssassin* 57
6.5.2 Variações do Filtro Composto 63
6.6 Limitações da Abordagem 71
6.7 Considerações 72

7 Conclusões e Trabalhos Futuros **74**
7.1 Conclusões 74
7.2 Trabalhos Futuros 75

Referências Bibliográficas **77**

Lista de Figuras

1.1 Arquitetura típica de um sistema de correio eletrônico 3
1.2 Diagrama de Tempo de uma Conexão SMTP 5

3.1 Arquitetura Padrão para Transferência e Filtragem de Mensagens de Correio Eletrônico 19
3.2 Arquitetura baseada em mecanismo de priorização 21
3.3 Grafo de um Sistema de Transferência de Mensagens de Correio Eletrônico Com Filtros Baseados em Priorização 24
3.4 1ºExemplo de Configuração: Detecção Conjuntiva 27
3.5 2ºExemplo de Configuração: Detecção Disjuntiva 27
3.6 3ºExemplo de Configuração: Detecção Distribuída 27

4.1 Exemplo de árvore de decisão 33
4.2 Geração do filtro composto de acordo com o modelo proposto 34

5.1 Variação da Nota de Corte para o Componente *body* 50
5.2 Variação da Nota de Corte para o Componente *header* 50
5.3 Variação da Nota de Corte para o Componente *rawbody* 50
5.4 Variação da Nota de Corte para o Componente *uri* 50
5.5 Tempo de processamento(acumulado) das mensagens por componente . . . 51

6.1 Função de densidade de probabilidade do volume de dados das cargas utilizadas 54
6.2 Árvore de detecção gerada para a carga de trabalho do projeto *SpamAssassin* 60
6.3 Percentual de mensagens processadas por componente 61
6.4 Precisão na detecção de *spams* em cada componente 61
6.5 Filtro Composto 62
6.6 *SpamAssassin* 62
6.7 Percentual de Mensagens Processadas dentro do Intervalo Específico 62
6.8 Árvore de detecção Nº1 67
6.9 Árvore de detecção Nº2 67

6.10 Árvore de detecção Nº3 67
6.11 Árvore de detecção Nº4 67
6.12 Árvore de detecção Nº5 67
6.13 Árvore de detecção Nº6 67

Lista de Tabelas

1.1 Principais Comandos SMTP 4

3.1 Base de Mensagens de Exemplo 26
3.2 Resultados do Exemplo 28

5.1 Tabela de Exemplo de Avaliação de Mensagens no *SpamAssassin* 44
5.2 Notas médias/CV de cada filtro em cada carga 48
5.3 Totais de regras 49
5.4 Dados referentes aos componentes 50
5.5 Tempo médio de processamento das mensagens por componente 51

6.1 Comparativo resumido das cargas de trabalho utilizadas 54
6.2 Distribuição das mensagens para os cenários da primeira etapa 57
6.3 Sumário da base de treinamento nos três cenários 58
6.4 Componente *body*: resultados por cenário 58
6.5 Comparativo: Filtro Gerado X *SpamAssassin* (Média entre os três cenários) 60
6.6 Tempo médio de processamento das mensagens entregues em cada componente X Tempo médio de processamento dessas no *SpamAssassin* (segundos) 63
6.7 Distribuição das mensagens nos cenários gerados para a carga de trabalho da UFMG 64
6.8 Sumário da base de treinamento nos três cenários (carga de trabalho da UFMG) 64
6.9 Comparativo dos Filtros Compostos gerados para a carga da UFMG . . . 68
6.10 Percentual de mensagens entregues por componente nos filtros compostos . 69
6.11 Efetividade de cada componente na entrega das mensagens 70

Capítulo 1

Introdução

O problema abordado neste trabalho é fruto direto do surgimento de uma era na qual não existem limites para a troca de informações entre os usuários de um amplo sistema computacional, criado exatamente para vencer as barreiras físicas, culturais e sociais, permitindo que pessoas colaborem umas com as outras e que comunidades inteiras possam evoluir através do compartilhamento de informações.

Esta rede mundial de computadores, conhecida como Internet, já pode ser considerada uma ferramenta imprescindível para a evolução global. Sua presença é impactante no meio acadêmico, profissional e pessoal da população. Mesmo aqueles que ainda não têm acesso a esta tecnologia já conseguem perceber que a presença da Internet vem causando alterações na realidade em que vivem. Seu surgimento abriu espaço para a criação de um mundo paralelo, que é comumente chamado de mundo virtual.

Uma infinidade de produtos e serviços foram estendidos ou mesmo só puderam ser criados neste mundo virtual. Um exemplo que pode ser citado são os serviços voltados para as redes sociais de relacionamento. A efetividade desses só pode se tornar uma realidade por causa da facilidade do compartilhamento de informações pela Internet, o que aproximou mais as pessoas. Essas redes fazem parte de um conjunto maior de serviços dedicados à comunicação entre os participantes da rede mundial de computadores. Os meios de comunicação sempre foram grandes aliados à evolução de qualquer sociedade. Em se tratando de Internet, a abrangência desses meios ultrapassa qualquer distância, seja ela física, cultural ou social.

Dentre os diversos meios de comunicação existentes na Internet, destacamos neste trabalho o correio eletrônico. Sua popularidade é tão grande que é impossível pensar nos dias de hoje em estar conectado à Internet e não possuir um endereço eletrônico. O funcionamento de sistemas de correio eletrônico é bastante simples e é apresentado com maiores detalhes na seção 1.1. Esta simplicidade, além de facilitar a popularização desses sistemas, também permite que esses possam ser utilizados em prejuízo dos seus

usuários, uma vez que permitem vários tipos de abusos. Por exemplo, é possível a disseminação de mensagens de qualquer natureza para qualquer destinatário sem que o remetente tenha que informar sua identidade real.

Essas mensagens, também chamadas de mensagens maliciosas, são exatamente a manifestação de grupos que, por diversos motivos, muitas vezes transgridem os princípios básicos dos indivíduos. Um grande alimentador deste problema são as pessoas que se interessam pelas mensagens enviadas sem solicitação. Além daquelas que transmitem, existem os compradores dos produtos anunciados, o que garante, muitas vezes, a lucratividade desta prática. Em 2003, foi apresentado um estudo no qual foi levantado que 7% dos usuários de correio eletrônico já realizaram alguma compra de produto ofertado em mensagens não solicitadas [29].

A disseminação dessas pragas virtuais é de difícil solução podendo, a princípio, ser reduzida e não contida plenamente. Em nenhum momento até então foi apresentado um estudo que viabilizou a eliminação completa deste problema. Até mesmo a classificação realizada por seres humanos apresenta um limite máximo de 99.9% de efetividade [71]. Existem mensagens bastante características, o que facilita a sua detecção por parte de uma diversa quantidade de mecanismos, porém, existem aquelas que são geradas sem nenhuma característica particular distinguindo-as das mensagens legítimas. Essas criam um desafio maior que é a proposta de técnicas que, muitas vezes são complexas e apresentam alto custo de processamento, pois têm de ser implementados mecanismos que sejam capazes de evitar que mensagens legítimas sejam classificadas como se fossem maliciosas.

Dada esta problemática, o objetivo deste trabalho é propor um modelo de detecção de mensagens maliciosas que seja mais eficiente que os modelos existentes, tanto do ponto de vista de custo quanto de eficácia.

1.1 Arquitetura de Sistemas de Correio Eletrônico

Um sistema de correio eletrônico é um conjunto de componentes com funções específicas, entre os quais trafegam as mensagens dos usuários. Neste sistema, cada elemento pode ser visualizado como um componente que recebe mensagens, realiza algum tipo de processamento e pode ou não repassá-las para outros componentes.

A arquitetura desses sistemas tem como base servidores nos quais são disponibilizados espaço de armazenamento de dados, bem como um endereço para seus usuários. Para enviar uma mensagem, um usuário deve conhecer o endereço eletrônico do destinatário. Para efetivar o envio, o remetente deve abrir uma conexão com um servidor. Nesta conexão a comunicação é realizada através de um protocolo de transferência de

mensagens eletrônicas, o SMTP [56] (*Simple Mail Transfer Protocol*). Essas mensagens eletrônicas são também conhecidas como *e-mails*. A conexão não é feita diretamente entre o remetente e o destinatário. Conforme pode ser observado na figura 1.1, a comunicação é realizada entre o cliente remetente e um servidor de correio eletrônico. Esse por sua vez comunica-se com um servidor intermediário ou diretamente com o servidor destino onde está a área de armazenamento de mensagens do usuário destinatário. Esta área de armazenamento é comumente conhecida como caixa de mensagens de correio eletrônico.

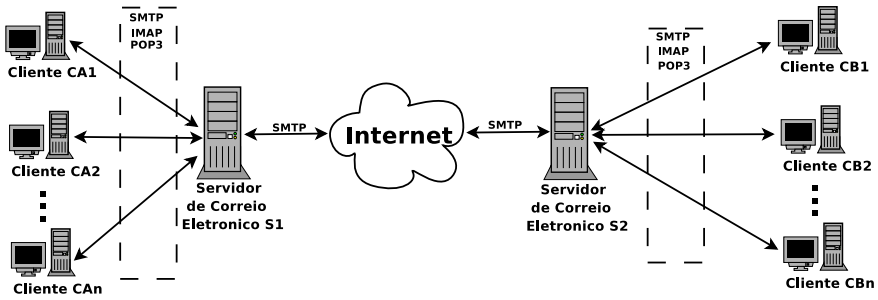

Figura 1.1: Arquitetura típica de um sistema de correio eletrônico

O recebimento das mensagens não necessariamente ocorre junto com o processo de transferência. A manipulação (leitura, cópia ou remoção) dessas pelo destinatário pode ocorrer a qualquer momento após a transmissão ter sido finalizada. Esta manipulação é feita por meio de protocolos distintos do SMTP: POP3 ou IMAP. O POP3 [53] (*Post Office Protocol* versão 3) permite que todas as mensagens contidas na caixa de correio eletrônico possam ser transferidas sequencialmente para um computador do recipiente. Desta forma, o usuário pode ler as mensagens recebidas, apagá-las, respondê-lás, armazená-las etc. O IMAP [10] (*Internet Message Access Protocol*) é um protocolo mais avançado que o POP3, uma vez que este permite que o usuário possa manipular suas mensagens diretamente no servidor, sem que as mesmas tenham de ser recuperadas. A seguir, descrevemos de maneira sucinta o protocolo SMTP.

A conexão SMTP não depende de um tipo particular de meio de transporte. Porém, é comumente realizada sob o TCP. O protocolo SMTP é bem simples, baseado em um conjunto de comandos enviados pelo remetente da mensagem e por respostas geradas pelo destinatário a partir dos comandos recebidos. Esses comandos e respostas são trafegados em formato texto sobre a camada de transporte utilizada. O fim de cada comando e resposta é determinado pelos caracteres de controle de retorno de carro e linha (*Carriage Return*-CR/*Line Feed*-LF).

Um conjunto mínimo de comandos para a transferência de mensagens pode ser encontrado na tabela 1.1. Os tipos de respostas estão apresentados com os rótulos S,

E, F e I. Esses representam, respectivamente, respostas de sucesso, erro, falha ou de inicialização. Esta última é enviada somente em resposta ao comando DATA indicando que os blocos de dados da mensagem podem ser enviados pelo remetente.

Comando	Descrição	Tipos de Respostas
HELO	Identifica o computador que dará origem a mensagem. Este nome é formado pelo nome do dominio do computador ao qual o usuário remetente está ligado. A sintaxe deste comando é `HELO <nome_computador>`	S: 250; E: 500, 501, 504, 421
MAIL	Este comando é utilizado para envio da identificação do remetente da mensagem. A sintaxe deste comando é `MAIL FROM:<e-mail do remetente>`	S: 250; F:552, 451, 452; E: 500, 501, 421
RCPT	Neste comando segue o endereço do destinatário da mensagem. Sua sintaxe é `RCPT TO:<e-mail do destinatário>`	S: 250, 251; F: 550, 551, 552,553, 450, 451, 452; E: 500, 501, 503, 421
DATA	Este indica que os próximos blocos de dados a serem enviados formam o conteúdo da mensagem. Estes são finalizados somente após o envio do caracter . no final. Este caracter não deve ser precedido de nenhum outro e após o mesmo devem ser enviados os caracteres `<CR><LF>`	I: 354; S: 250; F: 552, 554, 451, 452, 451, 554; E: 500, 501, 503, 421
RSET	Este comando indica que a mensagem que estava sendo transmitida foi cancelada. O servidor ao recebê-la deve descartar todas as informações referentes à mensagem em andamento, limpar todas as regiões de memória ocupadas e limpar as tabelas de estados inicializadas.	S: 250; E: 500, 501, 504, 421
NOOP	O único objetivo deste comando é receber do servidor uma resposta indicando se o mesmo está disponível ou não	S: 250; E: 500, 501, 504, 421
QUIT	Este comando é enviado para a finalização da conexão SMTP. Uma conexão só pode ser finalizada pelo transmissor se este enviar o comando `QUIT` e receber uma resposta positiva. Já o receptor só deve finalizar a conexão se receber o mesmo comando.	S: 221; E: 500

Tabela 1.1: Principais Comandos SMTP

O diagrama da figura 1.2 apresenta um exemplo simples do processo de envio de mensagem eletrônica. Neste, o cliente $CA1$ abre uma conexão SMTP com o servidor $S1$ para enviar uma mensagem para o cliente $CB1$. Este servidor por sua vez abre uma conexão com o servidor $S2$. Uma grande fragilidade deste sistema encontra-se na fase de identificação do remetente e dos destinatários. Apesar de já não ser uma prática comum nos servidores implementados atualmente, muitos ainda não fazem qualquer tipo de verificação do remetente, o que permite que qualquer usuário possa utilizar esses servidores para enviar mensagens para qualquer destinatário. Esses servidores são conhecidos como servidores abertos.

Neste diagrama é possível perceber que as mensagens são enviadas para o servidor destino somente quando o cliente informa que a mesma já foi finalizada. Desta forma, é possível que a mensagem seja cancelada antes que ocorra um tráfego desnecessário entre os servidores. Após indicar que a mensagem foi finalizada, a responsabilidade pela entrega da mensagem passa a ser do servidor remetente $S1$. Quando este finaliza o envio da mensagem, esta responsabilidade passa para o servidor destino $S2$. O usuário pode acessar a mensagem recebida a qualquer momento t após a entrega da mesma na sua caixa de entrada.

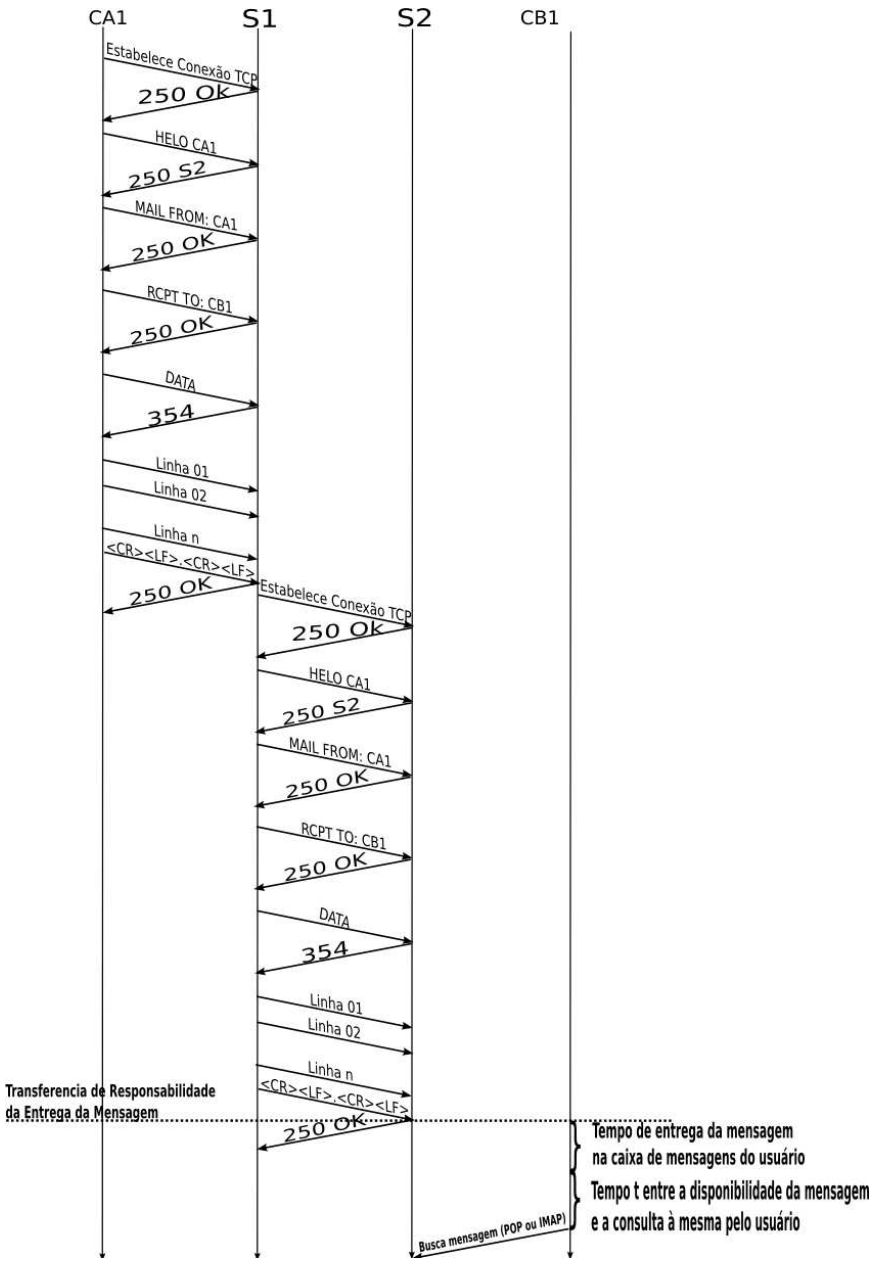

Figura 1.2: Diagrama de Tempo de uma Conexão SMTP

1.2 Mensagens Maliciosas

Mensagens comerciais não solicitadas, também definidas como mensagens indesejadas e comumente conhecidas como *spam*, são apresentadas como um problema reconhecido desde 1975 [55]. Até a primeira metade dos anos 90 este problema estava plenamente sob controle [23]. Neste período, essas mensagens consistiam basicamente das enviadas por pessoas que queriam fazer algum tipo de brincadeira ou criação de correntes. Após esse período, o acesso a Internet tornou-se bastante popular e junto com este as práticas comerciais também se tornaram mais frequentes. Em 1998 foi

observado que, em um grande provedor de serviços de Internet, 50% das mensagens de correio eletrônico que chegavam eram, na verdade, *spams*.

Existem dois fatores que contribuem muito para a disseminação dos *spams*: a simplicidade na transferência das mensagens de correio e a existência de servidores que permitem o envio das mesmas por qualquer usuário remetente. Qualquer programa de computador que consiga abrir uma conexão através de uma camada de transporte, como por exemplo o TCP, pode iniciar uma conexão SMTP e enviar diversas mensagens por meio desta conexão. Isto garante que os meios de transmissão de *spams* estejam em conformidade com os padrões utilizados.

Diversas técnicas e ferramentas de combate aos *spams* já foram propostas e desenvolvidas, dentre as quais destacam-se: utilização de listas negras para bloqueios de mensagens vindas de servidores abertos, filtros automáticos [36], ferramentas de filtros baseadas em regras previamente definidas por usuários (como exemplo podemos citar o *SpamAssassin* [8]), utilização de mecanismos de priorização [70] e até mesmo utilização de mecanismos de cobrança. Um detalhamento dessas técnicas pode ser encontrado no capítulo 2.

No entanto, dois fatores principais impedem que tais técnicas sejam plenamente efetivas, contribuíndo para a proliferação destas mensagens indesejadas [34]: o primeiro refere-se à evolução das técnicas de disseminação dos *spams*. A cada nova técnica de combate, são desenvolvidos novos métodos para burlá-las. Os responsáveis pela transmissão de *spams*, também conhecidos como *spammers*, contam ainda com uma grande variedade de softwares de baixo custo que facilitam o disparo de uma grande quantidade de mensagens por hora, chegando a taxas superiores a 250.000 mensagens/hora [23]. A simplicidade do protocolo utilizado pelos sistemas de correio eletrônico permite que essas ferramentas enviem mensagens com identificações forjadas, o que dificulta a obtenção da origem das mesmas, permitindo desta forma que o remetente real trabalhe de forma oculta.

O segundo fator é referente aos falsos positivos, ou seja, a classificação de mensagens legítimas como maliciosas. Por mais que os filtros evoluam, atualmente não se conhece um método 100% eficaz para a classificação de mensagens de correio eletrônico [71]. Um dos grandes desafios para a obtenção desses filtros é a quantificação do valor da informação para os usuários, ou seja, filtrar mensagens que conceitualmente para uns sejam *spam* e para outros não. Tal quantificação é uma preocupação recorrente e estudos voltados para a interação homem máquina já foram realizados sobre o assunto [6].

Os baixos custos de transmissão, a conformidade com os padrões adotados na Internet e as diferenças quase inexistentes entre mensagens legítimas e maliciosas tornam essas um grande desafio para os fornecedores de serviços de Internet. Aliado a isto

ainda existe o problema do conceito do que é ou não *spam*.

Um avanço nesta área de pesquisa está na caracterização do tráfego dessas mensagens [34]. Nesse trabalho foi apresentado que existem diferenças significativas entre o tráfego gerado pelas mensagens maliciosas e pelas mensagens legítimas. Essas informações podem ser utilizadas para a criação de um filtro mais inteligente e dinâmico, que adapta seus resultados à natureza do tráfego atual do servidor. Esse tipo de filtro pode agir de forma colaborativa com outros filtros, gerando resultados mais precisos.

1.2.1 Impacto das Mensagens Maliciosas

O grande aumento destas mensagens indesejáveis vem criando em muitos usuários uma certa resistência ao uso do correio eletrônico, bem como uma redução da confiabilidade e integridade deste serviço. Pesquisas realizadas em 2003 e 2004 [29, 61] apontam que cerca de 29% dos usuários de correio eletrônico afirmavam ter reduzido o uso deste serviço por causa de tais mensagens. Outro resultado desta pesquisa é que aproximadamente 63% afirmavam que os *spams* estariam causando uma redução da confiabilidade das mensagens eletrônicas. Em geral, mais de 80% dos usuários deste serviço se dizem aflitos com a existência dessas mensagens, seja pelo aborrecimento em receber pequena quantidade de mensagens de *spam* cujo conteúdo seja constrangedor, ou pelo grande volume dessas mensagens, o que consome um considerável tempo de trabalho.

Essas mensagens, além de consumirem grandes quantidades de recursos do sistema computacional dos servidores de correio eletrônico (por exemplo, espaço de armazenamento e banda de rede), forçam a criação de filtros mais inteligentes, que acabam consumindo muito processamento desses servidores. O tráfego intenso gerado por esses *spams*, muitas vezes, causam o atraso na entrega das mensagens legítimas uma vez que essas, por não se distinguirem diretamente das maliciosas, também devem ser processadas [70].

Um impacto criado não exatamente pelas mensagens, mas sim pelos mecanismos de detecção é o erro de classificação. Como ainda não se conhece uma solução para obtenção de 100% de precisão de classificação [71], esses mecanismos erram e seus erros causam um desconforto ainda maior em seus usuários. Na verdade, esses erros tornam o sistema não confiável. Quando o filtro não consegue detectar um *spam*, este chega à caixa de entrada do usuário, gerando o esforço de ter que removê-lo ou mesmo o constrangimento com algum conteúdo não apropriado. Esse erro também recebe o nome de geração de falsos negativos. O segundo erro de classificação é quando o filtro descarta uma mensagem legítima como se fosse maliciosa. Neste caso, o transtorno pode ser ainda maior para o usuário pois o mesmo não tem acesso a informação útil e

muitas vezes de extrema importância para o desenvolvimento de seu trabalho. Neste caso, temos ocorrências de falsos positivos.

1.3 Motivação e Contribuição do Trabalho

A principal motivação de nosso trabalho é a dificuldade em detectar de forma eficiente as mensagens maliciosas sem que os custos sejam afetados significativamente. Neste trabalho distinguimos dois tipos distintos:

Custo de processamento: Este é referente ao consumo de recursos do sistema computacional do servidor de correio eletrônico, podendo ser memória, cpu, I/O, banda de rede ou tempo de processamento.

Custo de Erro de classificação: Este refere-se a um conceito mais subjetivo, que depende do que o usuário final considera como sendo ou não um *spam*. O resultado deste depende diretamente do custo associado à classificação incorreta de mensagens maliciosas e legítimas.

As diversas técnicas de filtragem de mensagens maliciosas são, em geral, redundantes, ou seja, atuam de forma eficiente sobre os mesmos conjuntos de mensagens. Porém, quando se trata da detecção de *spams* mais elaborados, possuindo características que os tornam similares às mensagens legítimas a eficiência dessas técnicas fica comprometida.

A nossa proposta busca combinar diversos filtros para que trabalhem de forma colaborativa onde as mensagens redundantes, que sejam facilmente detectáveis por diversas técnicas distintas, possam ser capturadas antecipadamente sem que ocorra processamento desnecessário e aquelas que apresentam um maior grau de dificuldade de detecção sejam processadas por um conjunto maior de filtros.

Esses filtros podem ser combinados de diversas formas, no entanto, algumas configurações apresentam maior desempenho, tanto com relação à efetividade, quanto em relação aos custos. No capítulo 3 é feito um detalhamento referente à combinação de diversos filtros de mensagens maliciosas.

1.3.1 Contribuições

A principal contribuição deste trabalho está na proposta de um modelo diferenciado, no qual podem ser combinadas diversas técnicas de combate de mensagens maliciosas levando em consideração a efetividade e o custo de processamento. Com inspiração

em técnicas específicas de mineração de dados, definimos uma metodologia para a montagem de um filtro real seguindo este modelo proposto.

Existem propostas de filtros compostos por técnicas complementares que, em conjunto, detectam mensagens maliciosas. No entanto, não se conhece trabalhos que relacionam conjunto de filtros tradicionais ou parte desses de forma a gerar um filtro composto capaz de tomar a decisão antecipada da detecção de mensagens maliciosas.

Apresentamos neste trabalho um estudo de caso baseado em um filtro real, o *SpamAssassin* [8]. Esse estudo foi realizado para demonstrar que o modelo pode ser adotado na prática, desde que a relação entre as classificações realizadas pelos filtros atue de forma favorável à classificação final de cada mensagem.

1.4 Organização do Trabalho

O restante deste trabalho está organizado em mais seis capítulos. No capítulo 2 é feito um levantamento dos trabalhos referentes aos filtros de *spams* que vêm sendo propostos e adotados. Também fazemos uma breve referência a trabalhos sobre caracterização. Esses podem auxiliar no processo de escolha de componentes para o filtro composto e fornecer conhecimento necessário para descrição da carga de treinamento e de testes a serem utilizadas na montagem desse filtro composto. No capítulo 3 fazemos um estudo com relação aos modelos tradicionais de entrega de mensagens de correio eletrônico, além de uma referência a um modelo composto no qual é adotado o conceito de priorização. Nesse mesmo capítulo, apresentamos o modelo composto proposto neste trabalho. O capítulo 4 é dedicado à estratégia adotada para a montagem de um filtro que esteja de acordo com nosso modelo. Apresentamos todos os passos que devem ser seguidos e como deve ser feito o mapeamento com a técnica de classificação adotada na mineração de dados para obtermos um filtro composto mais eficiente. Já no capítulo 5 mostramos um estudo de caso no qual um filtro real é quebrado em diversos componentes de forma a possibilitar a montagem de um filtro composto. No capítulo 6 serão apresentados os resultados referentes ao estudo de caso, comparando o filtro gerado com um filtro real, e avaliando os efeitos das alterações nos fatores que ajustam a importância de cada critério na montagem do filtro. O fechamento deste trabalho é realizado no capítulo 7, onde apresentamos nossas conclusões, bem como direcionamentos para possíveis trabalhos futuros.

Capítulo 2

Trabalhos Relacionados

2.1 Introdução

Este capítulo é dedicado ao levantamento de trabalhos relacionados aos mecanismos voltados ao combate de mensagens maliciosas, além de um breve apanhado sobre trabalhos de caracterização neste contexto. Primeiramente, apresentaremos trabalhos voltados para o combate aos *spams*. Serão levantadas diversas técnicas de combate, com destaque para técnicas que utilizam mineração de dados. Em seguida descrevemos o trabalho desenvolvido em [34], no qual são avaliadas diversas caraterísticas do tráfego de mensagens maliciosas, entre outros trabalhos relevantes ao nosso escopo.

2.2 Técnicas de Combate

Existem diversas técnicas de combate propostas e em estudos. Muitas delas já são amplamente utilizadas em sistemas reais como, por exemplo, os filtros bayesianos. Essas técnicas podem ser divididas em dois grupos maiores. A definição desses grupos é baseada no ponto onde a responsabilidade pela entrega da mensagem é passada do servidor SMTP transmissor para o servidor SMTP receptor. As técnicas que atuam antes da transferência de responsabilidade são chamadas de técnicas *pre-acceptance*. Aquelas que atuam após este momento pertencem ao grupo das técnicas *post-acceptance* [70].

2.2.1 Técnicas de Combate *Pre-Acceptance*

Técnicas pertencentes a este grupo impedem que uma transferência de mensagens eletrônicas seja iniciada ou mesmo cancelada durante o processo. Um exemplo são técnicas baseadas em listas negras. Servidores que tenham seus endereços nessas listas não conseguem nem mesmo estabelecer uma conexão com o servidor SMTP destino.

O receptor, ao receber uma requisição de conexão, verifica se o endereço do servidor de origem encontra-se nessas listas negras e, em caso positivo, cancela a conexão antes mesmo que os primeiros comandos SMTP sejam transmitidos.

Atualmente, os principais tipos de técnicas *pre-acceptance* são:

Blocante: São técnicas baseadas na recusa de mensagens de origem duvidosa. Um mecanismo muito comum é a utilização de listas negras. Essas listas podem ser públicas [1] ou privadas. O servidor de correio eletrônico, ao receber uma solicitação de conexão a partir de um endereço presente em alguma dessas listas, recusa imediatamente a conexão impedindo que suas mensagens cheguem até seus destinatários. Existem listas negras que também armazenam endereços de MTAs, agentes transferidores de mensagens, que são utilizados como servidores abertos pelos usuários que enviam mensagens maliciosas [1, 3]. Existe também um mecanismo alternativo que visa incrementar listas negras privadas. Este consiste na determinação de, para cada remetente, um limite de mensagens para destinatários desconhecidos. Acima deste limite o sistema considera que está ocorrendo um ataque no qual estão sendo buscados endereços válidos. Uma vez detectado o ataque, o endereço do remetente é adicionado à lista negra e suas mensagens passam a ser recusadas [5]. Uma desvantagem clara dessas técnicas é que, na ocorrência de falsos positivos, as mensagens legítimas são eliminadas e não existe a possibilidade de recuperação uma vez que a mesma sequer chega ao ambiente destino [39].

Atraso: O mecanismo utilizado em técnicas baseadas em atraso posterga o envio das respostas aos comandos SMTP. Tais técnicas buscam consumir recursos dos servidores de envio de mensagens maliciosas. Dado que esses enviam mensagens para muitos destinatários diferentes, o atraso gerado pode consumir recursos suficientes de forma a tornar o processo de distribuição de mensagens inviável [45]. A grande desvantagem de técnicas de atraso é que recursos dos servidores que recebem as mensagens acabam sendo consumidos também, o que pode causar um grande atraso na recepção das mensagens legítimas.

Falha temporária: O servidor que utiliza esta técnica, ao receber mensagens de remetentes desconhecidos, envia uma resposta de falha temporária [56]. Quando a resposta de falha for enviada para remetentes legítimos, esses irão reenviar as mensagens que falharam. No entanto, isto não deve ocorrer quando o remetente for um *spammer* [44].

Combate em Servidores Remetentes: Essa técnica visa reduzir o número de mensagens maliciosas enviadas pelos ISP (Provedores de Serviços da Internet). A

responsabilidade por bloquear ou filtrar os remetentes é dos próprios servidores de origem dessas mensagens [19, 20, 73]. Por meio de análises de *logs*, tanto de chegada quanto de saída, é possível encontrar problemas gerados pelos usuários locais do ISP [19, 20]. Em [73] é proposto um sistema no qual busca-se agregar custos nas mensagens enviadas pelos remetentes baseados em seus comportamentos e qualidade das mensagens enviadas. A qualidade é obtida por meio de filtros instalados nesses servidores. O custo associado, neste caso, é um custo computacional que interfere diretamente no tempo de envio da suposta mensagem não solicidada. A desvantagem desta abordagem é que o custo associado às mensagens consideradas de baixa qualidade podem causar um consumo de recursos nos servidores que pode comprometer o envio das mensagens legítimas.

Autenticação: Nesta abordagem deve existir a figura de um cartório virtual que garanta a autenticidade dos remetentes das mensagens [15, 18]. Além disto, os MTAs receptores das mesmas devem verificar a origem dessas antes de aceitar os pedidos de conexão. A clara desvantagem desta técnica é a exigência de modificações estruturais na rede dos servidores de correio eletrônico existentes.

Legislação: O desenvolvimento de mecanismos legais é um trabalho que vem sendo realizado em diversos países e muitos trabalhos já abordaram essa questão [49, 52, 47, 58]. Em [49] é feito um estudo sobre as leis e a evolução da legislação vigente na União Européia voltada ao combate das mensagens maliciosas. Já em [52] é feito um paralelo entre a legislação que combate mensagens indesejadas da União Européia e dos Estados Unidos, além de fazer um levantamento geral de diversas leis existentes em outros países, como Canadá e Austrália. No entanto, mecanismos baseados em legislação são falhos por vários motivos [47]. O primeiro deles é a dificuldade para implantação. A obtenção da identidade real do infrator também representa um grande problema para ser resolvido. Além desses, ainda existe a subjetividade na determinação do que é uma mensagem realmente maliciosa ou apenas uma mensagem de interesse do destinatário. Apesar das críticas, em [58] é apontado como diferencial das medidas legais o custo criado pelo risco das punições jurídicas.

Ainda podem ser apontadas como técnicas *pre-acceptance*: as listas brancas, que são listas que armazenam o endereço de usuários confiáveis [39]; técnicas baseadas em reputação, a qual pode ser construída por meio de redes de relacionamento [33] e armazenada em listas brancas [32]; e técnicas baseadas em tarifação, nas quais devem ser agregados mecanismos como selos eletrônicos, que associariam custos tanto para enviar quanto para receber mensagens de correio eletrônico [42, 39, 42]. Dada a linha

de trabalhos baseados em custo, em [25] é apresentado um modelo econômico, no qual as mensagens maliciosas não são impedidas de serem entregues, mas é feito um balanceamento entre o custo de envio das mensagens e as preferências dos destinatários.

2.2.2 Técnicas de Combate *Post-Acceptance*

Técnicas desse grupo buscam impedir que as mensagens maliciosas já recebidas no servidor SMTP cheguem às caixas de entradas dos usuários do sistema. Essas técnicas atuam dentro do sistema receptor das mensagens e, geralmente, ao identificar um *spam*, geram uma marca neste ou mesmo o elimina do sistema.

Existem diversas propostas de ferramentas e mecanismos voltados para a contenção de mensagens maliciosas [67, 37, 51, 41, 62, 12, 65, 54, 64, 8] do lado do servidor destino. Essas técnicas funcionam como mecanismos que são comumente conhecidos como filtros de correio eletrônico. As diversas propostas também podem ser divididas em 5 grupos específicos, sendo eles:

Filtros Baseados em Regras: São filtros que realizam avaliação direta do conteúdo das mensagens buscando padrões específicos, tanto de mensagens maliciosas quanto de mensagens legítimas. Esses padrões são definidos por meio de regras que são normalmente implementadas como expressões regulares. Um importante representante deste tipo de filtro é o *SpamAssassin* [8].

Filtros Colaborativos: São filtros que visam identificar *spams* através da troca de informações entre diversos servidores ou mesmo entre os usuários dos sistemas [37, 41, 57, 68]. Em [37] é proposto a arquitetura *CASSANDRA*, onde o conceito de mensagem maliciosa pode variar de usuário para usuário. É um filtro colaborativo com personalização onde, quando uma mensagem *spam* é encontrada, essa recebe uma identificação que é compartilhada entre os diversos participantes do sistema. Desta forma, para cada usuário ou grupo de usuários serão marcadas como mensagens ilegítimas somente aquelas que são assim consideradas por estes. Assim como em [37, 41] apresenta uma proposta de um filtro colaborativo baseado em grupos de usuários, porém nesse é montada uma rede baseada nas relações sociais de seus participantes. Os sistemas apresentados por [57, 68] são sistemas colaborativos distribuídos e também levam em consideração que uma mesma mensagem maliciosa é enviada para diversos usuários [39]. A informação de mensagens *spams* é distribuída entre diversos servidores que devem reportar a seus usuários que as mensagens avaliadas estão ou não presentes em suas bases de dados.

Filtro com Priorização: É um filtro no qual é realizada uma avaliação prévia das mensagens ou da origem dessas. Essa avaliação busca antecipar a classificação das mensagens. Aquelas que são consideradas mensagens legítimas recebem uma prioridade maior no processamento. Uma proposta baseada simplesmente no histórico dos remetentes é apresentada em [70].

Filtros com Correção de Palavras: Com a evolução dos mecanismos de filtragem de *e-mails*, os usuários remetentes de mensagens maliciosas também evoluiram suas técnicas. Uma forma que vem sendo muito utilizada por esses usuários é a modificação de palavras [40]. Esta modificação cria um desafio maior uma vez que as palavras presentes nas mensagens indesejadas não pertencem a um vocabulário conhecido. Para auxiliar os filtros, em [11] é apresentada uma técnica que busca remover caracteres que não sejam alfa-numéricos das palavras presentes no texto das mensagens (por exemplo, quando for encontrada a palavra V.i-a.g*r.a no texto original, esta é modificada para Viagra). Uma outra alternativa é substituir combinações de caracteres que possam parecer com letras (por exemplo, \/iagra, onde o \/ seria substituído por V).

Mineração de Dados: A tarefa de identificação de mensagens maliciosas pode ser diretamente mapeada como um problema de mineração de dados, mais especificamente um problema de classificação. A classe define se a mensagem é legítima ou não, os atributos são características das mensagens e as transações são os próprios *e-mails*. Sendo assim, muitos trabalhos já foram realizados na tentativa de solucionar este problema através de técnicas de classificação [66, 51, 63, 69, 26].

Muitos dos trabalhos nesta linha já abordam a questão do custo de classificação incorreta [66, 65]. É fato que, quando um filtro classifica incorretamente uma mensagem legítima, este pode estar causando grandes danos para o destinatário da mensagem. Quando o erro de classificação ocorre com uma mensagem maliciosa, os danos, em geral, não chegam a ser considerados traumáticos. O custo neste caso é do usuário ter de ler a mensagem para saber se a mesma é ou não um *spam* e apagá-la manualmente. Sendo assim, é de extrema importância a existência de trabalhos que procuram agregar estes custos aos seus objetivos.

No trabalho realizado em [27] é feita uma combinação entre o algoritmo bastante utilizado em classificadores, o *Naive Bayes* e o algoritmo *K-Nearest Neighbor (kNN)*. Nesse trabalho os autores conseguiram uma considerável melhoria na precisão da classificação com um menor custo. Como mostrado em [28], para aumentar a precisão do *Naive Bayes*, é necessário aumentar o tamanho da dimensão

dos vetores representantes das mensagens. Com a combinação, o tamanho ótimo foi reduzido, o que permitiu também uma economia de recursos computacionais.

Dois subconjuntos muito comuns de técnicas baseadas em mineração de dados são:

Filtros Bayesianos: A utilização de modelos bayesianos para identificação de mensagens *spam* foi proposto inicialmente em 2002 [35]. Filtros bayesianos são basicamente classificadores que utilizam redes Bayesianas para realização da classificação das mensagens [64, 13]. Essas redes são montadas com a utilização de um conjunto pré-classificado de mensagens que servem como base de dados de treinamento. Dessa forma, é definido um nó da rede como sendo a classe binária C(a mensagem é legítima ou não) e outros vários nós, representando as variáveis independentes V_i, são as palavras presentes nas mensagens. As arestas entre os nós da rede são definidos como a influência probabilística que as palavras exercem sobre a classificação das mensagens. Essa técnica de combate também é utilizada em conjunto com outras técnicas em ferramentas como o *SpamAssassin* [8]. Nesse tipo de ferramenta é definido um número mínimo de mensagens legítimas e não legítimas que devem ser utilizadas como base de dados de treinamento [4, 54].

Filtros Baseados em Memória: Assim como os filtros bayesianos, este também exige a utilização de bases de dados de treinamento. No entanto, todas as bases são armazenadas em memória para que suas instâncias sejam utilizadas diretamente na classificação [14, 65]. A estrutura em memória utilizada para armazenamento é um espaço de múltiplas dimensões onde cada instância é um ponto neste espaço. A classificação é feita através de cálculos que ajudam estimar a similaridade entre a mensagem avaliada e as instâncias armazenadas [65]. Normalmente é utilizado o algoritmo *K-Nearest-Neighbor (k-NN)* [22].

Além dos filtros com funcionalidades específicas, propostas compostas por diversas camadas também vêm sendo trabalhadas. Estas levam em consideração várias propostas já feitas, de medidas legais até técnicas computacionais [43, 67, 12, 72]. Em [43] é apresentado um considerável conjunto de técnicas e como elas podem ser combinadas. Um fator que pode contribuir para a combinação de várias estratégias é a utilização de padrões abertos. Os projetos *SpamGuru* [67] e *Spamato* [12] são exemplos de plataformas de filtragem que permitem a adoção conjunta de diversas técnicas. Um exemplo de componente envolvido na plataforma *SpamGuru*, é o classificador *Chung-Kwei* [62].

Esse é um sistema capaz de aprender, cujo protótipo desenvolvido apresentou ótimos resultados com relação à precisão de classificação de mensagens maliciosas.

2.3 Trabalhos de Caracterização

Trabalhos de caracterização ajudam, muitas vezes, a entender o comportamento dos usuários de serviços de correio eletrônico e, principalmente, as características do tráfego gerado pelas mensagens maliciosas. Os resultados dessas caracterizações podem auxiliar na especificação de técnicas de combate que estejam mais adequadas ao contexto de determinados servidores de correio eletrônico.

Os autores do trabalho apresentado em [34] realizaram um amplo levantamento de características presentes no tráfego de mensagens *spams*. Foram avaliadas características como número de destinatários por mensagem, distribuição do tamanho das mensagens, processo de chegada e a localidade temporal dos remetentes destinatários das mensagens. Os autores realizam uma comparação entre o tráfego de *e-mails* legítimos e o tráfego das mensagens maliciosas. Em [17, 16] é feito um estudo similar, porém com escopo limitado aos *e-mails* legítimos.

Seguindo esta linha comparativa entre o tráfego de mensagens legítimas e mensagens maliciosas, os trabalhos realizados em [33] demonstram que existe uma clara diferença entre as redes de relacionamentos criadas pelos dois tráfegos.

Um último estudo de caracterização que deve ser apresentado é com relação aos usuários que enviam essas mensagens indesejadas. Em [59], são identificados dois tipos básicos desses usuários: o primeiro envia por um longo espaço de tempo, no mínimo 1 mês, mensagens com ofertas de produtos. Este é chamado de vendedor. O segundo tipo de usuário é o fraudador. Este envia em pouco espaço de tempo, em torno de 12 horas, mensagens com informações falsas com o objetivo de enganar um maior número possível de destinatários das mensagens enviadas. Em [48] também é realizado um estudo sobre a utilização de boas palavras escondidas em mensagens maliciosas para enganar os filtros. Um exemplo de filtro burlável por meio desta técnica é o baseado em regras. Como esse busca padrões de mensagens, quando encontra palavras normalmente utilizadas em mensagens legítimas, realiza uma classificação incorreta dos *spams*. No trabalho apresentado em [48], os autores apresentam o impacto nos filtros, sendo demonstrando o quanto esses degradam seus resultados quando essas técnicas são utilizadas.

2.4 Considerações

Todos os trabalhos aqui apresentados possuem um objetivo único, reduzir o número de mensagens maliciosas causando o menor impacto possível aos usuários de correio eletrônico. Porém, cada um atua em determinado momento do processo de transferência de mensagens e apresentam suas respectivas vantagens e desvantagens. O nosso trabalho busca uma otimização entre a precisão de classificação, com baixo custo para o usuários, tanto para processamento quanto de erros de classificação. Este balanceamento não é otimizado plenamente em nenhuma das propostas já levantadas. Identificar corretamente mensagens comerciais típicas, que possuem uma grande concentração de mensagens e jargões próprios de mensagens *spams*, como por exemplo divulgação de material pornográfico, é relativamente fácil e é uma tarefa realizada por praticamente todas as propostas levantadas. O problema encontra-se em detectar mensagens maliciosas que não possuem maiores distinções das mensagens legítimas. São aquelas mensagens transmitidas como se fossem realmente direcionadas para o usuário destino.

No modelo proposto neste trabalho buscamos antecipar a identificação das mensagens que são tradicionalmente classificadas corretamente deixando um processamento mais rigoroso para aquelas que, a princípio, não se distinguem diretamente das mensagens legítimas. Desta forma, busca-se com um menor custo separar as mensagens fáceis de serem detectadas das mensagens difíceis.

No próximo capítulo, será apresentado o modelo proposto, no qual é feita uma abertura para a combinação das diversas técnicas, assim como proposto em outros trabalhos [43, 67, 12, 72], porém buscando uma forma diferenciada de organização. Lembrando que nosso objetivo é obter um modelo de detecção eficiente tanto em questão de custos quanto de efetividade de detecção. Como uma grande parte dos mecanismos que podem ser facilmente combinados atuam após o recebimento da mensagem do lado do servidor destino, focaremos nosso trabalho em técnicas *post-acceptance*.

Capítulo 3

O Modelo de Filtro Composto

3.1 Introdução

Neste capítulo, faremos uma breve revisão do modelo de entrega de mensagens de correio eletrônico considerando apenas um filtro na estrutura. Para introdução do conceito de filtro composto, apresentaremos como exemplo o modelo com priorização [70], no qual existe a figura de um componente auxiliar que busca antecipar a identificação de mensagens legítimas. Em seguida, será apresentado o modelo proposto neste trabalho, no qual vários componentes associados formando um filtro composto com o objetivo de antecipar a classificação das mensagens, buscando reduzir o custo. Levantaremos também a problemática da disposição ótima dos componentes levando em consideração a efetividade, o custo de processamento em cada componente e o custo do erro de classificação.

3.2 Modelo Tradicional

A estrutura convencional de transmissão e filtragem de mensagens eletrônicas amplamente implementada é baseada em um modelo simples. Conforme apresentado no capítulo 1, as mensagens são enviadas para seus destinatários por meio de agentes que utilizam um protocolo bastante simplificado [56]. Nesta seção apresentamos como o filtro de mensagens está posicionado na arquitetura de sistemas de correio eletrônico.

As mensagens de correio eletrônico são endereçadas a um agente de transferência de mensagens, conhecido como MTA (*Mail Transfer Agent*), destino a partir de um MTA de origem. Um MTA bastante conhecido é o *Postfix* [2]. Este é um servidor SMTP [56] de código aberto que é amplamente utilizado nos provedores de serviços de Internet.

Uma vez estabelecida a conexão, o MTA repassa as mensagens recebidas para um filtro de *spam*. Em geral, esse filtro marca as mensagens como legítimas ou não. O destino das mensagens é determinado conforme a classificação gerada. Quando o agente usuário de mensagens MUA (*Mail User Agent*) requisita ao servidor de correio as mensagens, esse deve retornar tanto aquelas armazenadas no local de entrega de mensagens maliciosas quanto aquelas armazenadas no local de entrega de mensagens legítimas. A requisição realizada pelo MUA deve ocorrer através de protocolos como o IMAP ou o POP3 [10, 53].

A figura 3.1 apresenta uma arquitetura simples de um sistema de correio eletrônico com um filtro associado. Nessa, o filtro faz a avaliação das mensagens e estas são retornadas para o MTA. O agente de transferência é quem decide, a partir da classificação, efetivamente o destino dos *e-mails*. Isto é o que ocorre em ambientes que utilizam como filtro apenas o *SpamAssassin* [8]. Filtros como este não fazem nada além de criarem marcações nas mensagens quando essas são identificadas como maliciosas.

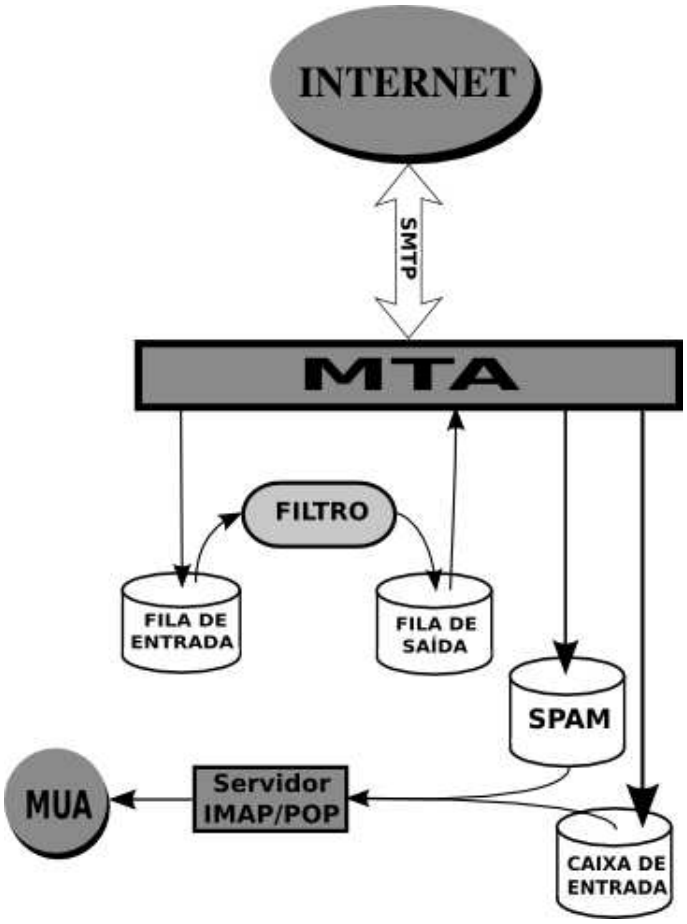

Figura 3.1: Arquitetura Padrão para Transferência e Filtragem de Mensagens de Correio Eletrônico

No *SpamAssassin*, cada mensagem é submetida a diversas operações, todas base-

adas em regras, onde cada regra aplicável é associada a uma nota. Ao final, todas as mensagens recebem uma nota total que é comparada com um limiar que separa os *e-mails* legítimos dos *spams*. Se a nota estiver acima deste limiar, a mensagem é marcada como sendo maliciosa. Esta marcação é utilizada pelo MTA para tomada de decisão. No capítulo 5 o *SpamAssassin* será apresentado com maiores detalhes.

Em geral, no modelo simples não existe motivação para analisar o custo de processamento das mensagens, uma vez que todas passam pelo mesmo processo, independente da classificação. O tempo final de entrega das mensagens legítimas acaba sendo dependente do tráfego total do servidor e do tamanho das mensagens transmitidas.

Outro custo que não é abordado neste modelo tradicional é o de classificação incorreta. No entanto, é fato que os danos causados na classificação incorreta de mensagens legítimas é, normalmente, muito maior do que a classificação incorreta de mensagens maliciosas [65].

3.3 Modelo com Priorização

Uma alternativa proposta por [70] envolve um componente cuja função é antecipar a identificação de mensagens legítimas de forma a priorizar o processamento das mesmas. Este componente tem a tarefa de realizar uma operação de previsão e não de uma definitiva avaliação de conteúdo.

É apresentado na figura 3.2 a representação esquemática do sistema com o componente preditivo utilizado em [70]. Os autores utilizam um componente que funciona como um filtro probabilístico que, baseado no histórico dos servidores remetentes, calcula a probabilidade da mensagem ser ou não legítima. Uma vez feito o cálculo, as mensagens são colocadas nas filas de alta ou baixa prioridade e são escalonadas de forma que as mensagens provenientes de servidores que provavelmente enviam somente *spams* são direcionadas para um servidor com um filtro mais robusto que consome uma grande quantidade dos recursos computacionais. As demais mensagens são repassadas para um servidor que possui um maior poder computacional, podendo ser, por exemplo, uma máquina com maior capacidade de processamento. Neste cenário, as mensagens previamente classificadas como maliciosas irão demorar mais a serem entregues ou definitivamente descartadas. Nesta abordagem foi obtido uma taxa de acerto na previsão maior do que 90%. No entanto, outros métodos de previsibilidade poderiam ser utilizados, como o histórico dos destinatários ou mesmo a taxa de entrega das mensagens, uma vez que o tráfego dos *spams* distingue-se do tráfego das mensagens legítimas [34].

O mecanismo de priorização pode ser visto como um modelo composto por três

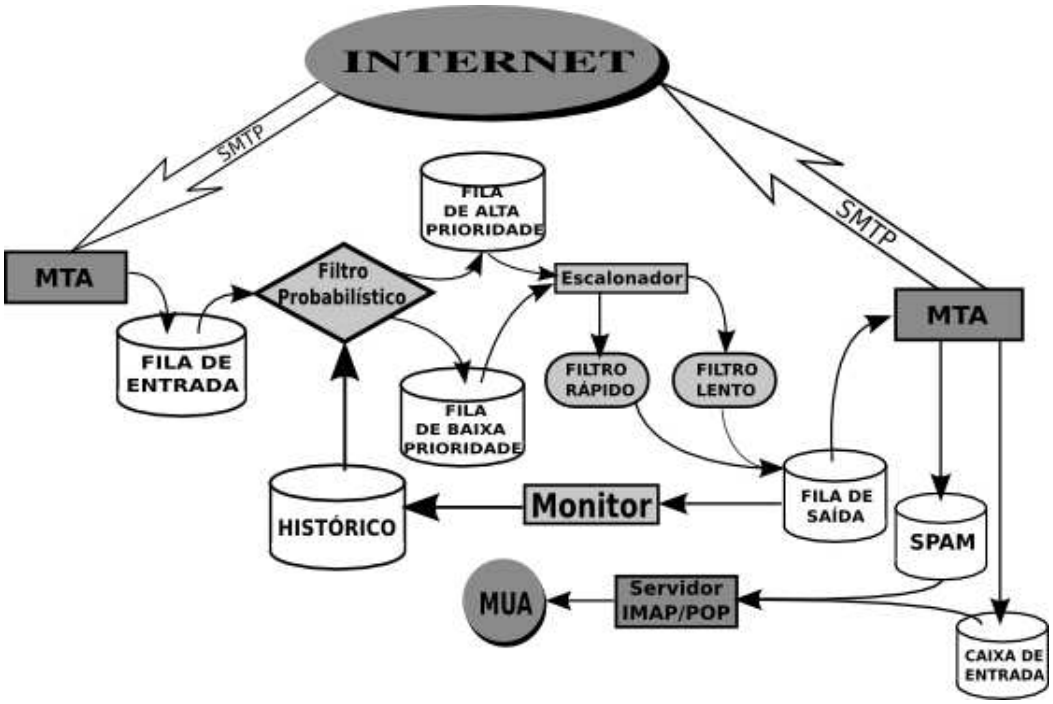

Figura 3.2: Arquitetura baseada em mecanismo de priorização

componentes. O objetivo neste caso é a redução do tempo de processamento das mensagens legítimas gerado pelo processamento das mensagens maliciosas. No entanto, aqui também não é diferenciado o custo com o erro de classificação. Como já foi relatado, o custo em classificar erroneamente uma mensagem legítima como maliciosa é maior do que quando o oposto ocorre. Neste modelo, a penalização é ainda maior, uma vez que, quando identificada como *spam*, a mensagem legítima receberá um tratamento com baixa prioridade e será entregue a um custo muito maior enquanto que, quando o oposto ocorrer, a mensagem maliciosa terá alta prioridade e um baixo custo de entrega. O tempo total de entrega das mensagens nesse tipo de filtro agora depende também do tempo da realização da pré-classificação e do tempo de escalonamento. Apesar do primeiro tempo citado sofrer a mesma influência do filtro simples (tráfego e tamanho das mensagens), o filtro com priorização apresenta o escalonamento com um ponto de diferenciação. Mensagens com pré-classificações distintas são escalonadas e entregues em intervalos distintos. Obviamente, a eficiência desse mecanismo dependerá diretamente do algoritmo de escalonamento utilizado.

Nessa abordagem, as mensagens legítimas classificadas corretamente serão entregues com um baixo custo, no entanto, o custo do erro de classificação é maior do que no modelo tradicional, uma vez que, além do custo do erro propriamente dito [65], a mensagem legítima será penalizada com a baixa priorização, como se fosse maliciosa.

O filtro baseado em priorização pode ser visualizado como motivador para o desen-

volvimento deste trabalho. Ele pode facilmente ser mapeado em um filtro composto por três componentes, onde o primeiro visa antecipar a classificação e os demais visam efetivar a classificação com diferentes prioridades. Na próxima seção apresentaremos o modelo proposto e como ele pode interferir positivamente nos custos de processamento em cada componente e no custo de erro de classificação.

3.4 Modelo de Filtro Composto

O modelo de filtro composto, MFC, é um modelo baseado em um conjunto de componentes. Esses pode ser filtros tradicionais ou apenas filtros simplificados. Tais componentes realizam uma classificação sobre as mensagens de correio eletrônico e decidem repassá-las para outros componentes ou entregá-las com uma marcação definitiva.

Como já apresentado no capítulo 2, existem diversas propostas onde são empregadas, em conjunto, várias técnicas para o combate às mensagens maliciosas. Essas técnicas são implementadas como etapas, ou camadas, que processam todas as mensagens e somente o resultado final é considerado. Muitos desses trabalhos adicionam camadas que não fazem avaliação da mensagem propriamente dita, como mecanismos de cobrança ou mesmo medidas não tecnológicas como a aplicação de leis.

Tais estudos adotam estratégias para que o custo de processamento associado a cada camada seja reduzido. Um exemplo é o trabalho apresentado em [67]. Nesse, o pré-processamento das informações é realizado uma única vez, e os elementos gerados são compartilhados por todas as camadas que realizam algum tipo de processamento que necessite dos dados. No entanto, não é correto negligenciar que todas as mensagens, sendo elas legítimas ou não, são avaliadas e processadas de alguma maneira por todos os níveis nesse filtro. Desta forma os custos associados a cada etapa acabam sendo somados, gerando um custo resultante ainda maior. Existe um conjunto de mensagens que são detectáveis pelas diversas técnicas. Um exemplo a ser citado são mensagens com textos sobre sexo. Essas apresentam padrões que são facilmente capturados pelas diversas técnicas utilizadas em filtros com múltiplas camadas. Se essas mensagens fossem descartadas logo nas primeiras camadas, o custo final de processamento seria diretamente reduzido e, consequentemente, o impacto na entrega das mensagens legítimas seria menor.

Em [67] não existe uma composição de filtros propriamente dita e sim uma composição de mecanismos de tratamento de mensagens. Porém, se forem adotados filtros intermediários, que chamaremos de componentes, certamente existirão mensagens que serão detectadas por uma grande parte destes e poderão receber uma classificação definitiva logo no início do processo sem que tenham de ser avaliadas todos os componentes.

Com relação ao custo do erro de classificação, mais uma vez é negligenciado o fato dos dois tipos de erros apresentarem custos distintos. Nos trabalhos apresentados no capítulo 2, a maior preocupação é com a efetividade. Foram listados mecanismos com múltiplas camadas onde o resultado final são classificadores mais rigorosos. Com o aumento da efetividade, o número total de mensagens classificadas incorretamente é reduzido, mas isto não implica que o custo total do erro de classificação também será mais satisfatório. Esta implicação seria verdadeira se os dois tipos de erros (classificação errônea de *spams* e classificação errônea de mensagens legítimas) tivessem o mesmo custo.

Para melhor esclarecer este problema, tomemos as equações 3.1 e 3.2, nas quais $efetiv$ é a efetividade geral, $M_{S \to S}$ é o número de mensagens maliciosas classificadas corretamente, $M_{L \to L}$ é o número de mensagens legítimas classificadas corretamente, $M_{S \to L}$ é o número de mensagens maliciosas classificadas como legítimas, $M_{L \to S}$ é o número de mensagens legítimas classificadas como maliciosas, M é o total de mensagens, C_{erro} o custo total do erro de classificação, α é o custo de classificar incorretamente uma mensagem legítima como *spam* e β é custo de classificar incorretamente uma mensagem maliciosa como legítima.

$$efetiv = \frac{M_{S \to S} + M_{L \to L}}{M} \tag{3.1}$$

$$C_{erro} = M_{L \to S} * \alpha + M_{S \to L} * \beta \tag{3.2}$$

Conforme já relatado, o custo de classificar incorretamente uma mensagem legítima é maior do que da classificação incorreta de mensagens maliciosas. Logo, normalmente, $\alpha > \beta$. Podem existir situações nas quais um filtro **A** apresente uma efetividade geral maior do que um filtro **B**, porém realiza uma classificação incorreta de um número maior de mensagens legítimas. Nessa situação teremos um filtro mais efetivo, porém com maior custo de erro de classificação.

A abordagem apresentada em trabalhos com múltiplos filtros busca explorar a cooperação entre os componentes de forma que os erros cometidos por uns possam ser corrigidos por outros de forma que a efetividade seja ajustada [21]. Isto é um fator positivo e motivador para a busca de um modelo que aproveite esta cooperação mas com o compromisso de otimizar a relação existente entre efetividade e custo, sendo este custo o de processamento realizado nos componentes e também o custo do erro de classificação. Na próxima seção apresentamos o modelo proposto considerando os efeitos em sua configuração. Para escolha da melhor configuração, foi adotada uma estratégia inspirada em técnicas de mineração de dados. Esta estratégia encontra-se detalhada no capítulo 4.

3.4.1 Representação do MFC

Conforme descrito no capítulo de introdução deste trabalho, um sistema de correio eletrônico pode ser visualizado como um conjunto de componentes no qual as mensagens são repassadas de um componente para outro. Em cada um desses componentes ocorre algum tipo de processamento. Na seção 3.2 foi apresentada uma arquitetura simples, na qual as mensagens são processadas por um filtro de mensagens maliciosas. Considerando esta estruturação de componentes e troca de mensagens, podemos representar este sistema como um grafo direcionado acíclico $G(V, A)$, onde V é o conjunto de vértices do grafo G e representam todos os elementos do sistema e A é o conjunto de todas as aretas $a(i, j)$ pertencentes a este mesmo grafo e que representam as mensagens que trafegam do elemento i para o elemento j deste mesmo sistema. A título de ilustração, a figura 3.3 apresenta como o sistema apresentado pela figura 3.2 pode ser representado.

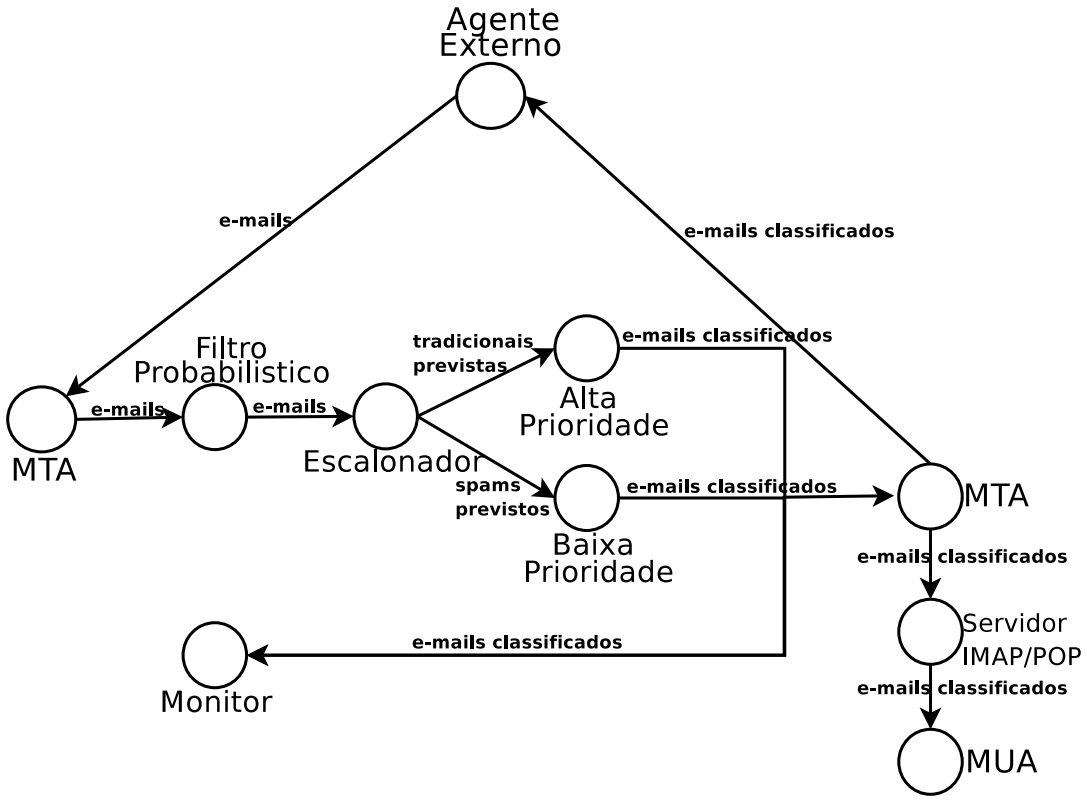

Figura 3.3: Grafo de um Sistema de Transferência de Mensagens de Correio Eletrônico Com Filtros Baseados em Priorização

Se o único filtro for substituído por um filtro composto, a representação em grafo do sistema mantém-se válida. No entanto, no novo grafo existirá um maior número de vértices e arestas. Dado o foco deste trabalho, para uma maior simplificação, a partir

deste ponto será considerado apenas o sub-grafo $G_1(V_1, A_1)$, onde $V_1 \subset V$ são todos os componentes do filtro composto e A_1 são as mensagens trocadas entre eles.

Seja um conjunto de N componentes distintos e independentes, com características próprias e até mesmo complementares. No nosso modelo, cada componente deve repassar, a partir da classificação realizada, as mensagens para um próximo componente ou mesmo gerar uma classificação definitiva. A estratégia de classificação depende da natureza do componente, como um limiar probabilístico ou um conjunto de regras, cada uma com um peso, e um limiar para definir a natureza da mensagem. Sem perda de generalidade, vamos assumir que o componente insere uma anotação na mensagem (*flag*) com o resultado da sua avaliação. Uma vez que essa mensagem esteja anotada, ela é enviada para outro filtro ou é gerada uma classificação definitiva, dependendo da natureza da anotação.

A obtenção do melhor grafo não necessariamente exigirá a utilização de todos os N componentes considerados. É possível a existência de um sub-conjunto de componentes que apresente o melhor ganho possível dentro do conjunto total. Isto pode ocorrer quando a maior efetividade possível para o conjunto inteiro for obtida com apenas um sub-conjunto de componentes. Caso isto ocorra, acrescentar componentes ao sistema acarretará em processamento adicional que pode degradar a otimização do sistema. Como já relatado, o que se busca aqui é a otimização da efetividade com relação aos custos de processamento e de erro de classificação.

Dado o conjunto V_1 composto por todos os componentes considerados passíveis de serem utilizados no sistema, seja $\Im = \{\forall H(V', A') | V' \subset V_1 \; e \; A' = \{\forall s'(i,j) | i \neq j \; e \; componente \; i \; envia \; mensagens \; maliciosas \; para \; componente \; j\} \cup \{\forall h'(i,j) | i \neq j \; e \; componente \; i \; envia \; mensagens \; legítimas \; para \; componente \; j\}\}$. É fato que, para qualquer grafo $H(V', A') \in \Im$ os totais $M_{S \to S}$, $M_{L \to L}$, $M_{S \to L}$ e $M_{L \to S}$ podem sofrer alguma alteração. Isto ocorre porque em cada combinação espera-se um comportamento diferente. No exemplo apresentado na próxima seção é possível observar este comportamento.

O exemplo apresentado na próxima seção é uma boa demonstração. Neste temos três grafos distintos representando filtros compostos pelos mesmos tipos de componentes. Em todas as três situações apresentadas os valores referentes à detecção de mensagens foram diferentes. Neste caso, para cada um desses grafos temos efetividade e custos distintos. Seja $\Gamma_K(efetiv_K, C_{proc_k}, C_{erro_K})$ a função de retorno do valor custo/benefício do grafo $K \in \Im$, onde C_{proc_K} é o custo de processamento total do grafo K. O resultado ótimo é obtido para o sub-conjunto de grafos $\xi \subset \Im$, onde $\xi = \{\forall H(V', A') \; \exists K(V'', A'') | \Gamma_K \leq \Gamma_H\}$.

3.5 Exemplificação

Sejam os filtros A, B e C passíveis de serem adotados como componentes de um filtro composto hipotético. Sem perda de generalidade, os custos de processamento por mensagem são os mesmos para os três. Na tabela 3.1 é apresentada uma pequena base de mensagens, onde estão dispostas as classificações geradas pelos componentes. Também estão disponibilizadas as classificações reais das mensagens. No conjunto apresentado existe um total de 10 mensagens, das quais 7 são legítimas e 3 são *spams*. Nas figuras 3.4, 3.5 e 3.6 podem ser observadas três configurações geradas aleatoriamente para o filtro composto. Quando um componente recebe uma mensagem, baseado na classificação da mesma, ou ele a repassa para outro ou finaliza o processo de classificação.

	Classificação			
	Filtro A	**Filtro B**	**Filtro C**	**Real**
Mensagem 1	Spam	Spam	Spam	Spam
Mensagem 2	Spam	Legítima	Legítima	Legítima
Mensagem 3	Spam	Legítima	Legítima	Legítima
Mensagem 4	Legítima	Legítima	Spam	Legítima
Mensagem 5	Spam	Spam	Spam	Spam
Mensagem 6	Legítima	Spam	Legítima	Legítima
Mensagem 7	Legítima	Legítima	Legítima	Legítima
Mensagem 8	Spam	Legítima	Legítima	Spam
Mensagem 9	Spam	Spam	Spam	Legítima
Mensagem 10	Spam	Legítima	Spam	Legítima

Tabela 3.1: Base de Mensagens de Exemplo

Na tabela 3.2 está contida uma breve análise das configurações de exemplo. A primeira configuração realiza uma detecção conjuntiva de mensagens *spam*, ou seja, uma mensagem é efetivamente classificada como maliciosa somente quando for classificada dessa forma por todos os componentes. Esta configuração, neste contexto, possui uma boa efetividade, uma vez que 80% das mensagens são classificadas corretamente. No entanto, esta apresenta um custo relativamente alto, uma vez que 30% das mensagens são processadas por todos os filtros e dessas, uma foi classificada erroneamente como *spam*. Nesse caso, a mensagem é penalizada com o tempo de processamento e com a classificação incorreta. O tempo de processamento dessa mensagem é a soma dos tempos gastos em cada componente e somente no final, a mesma recebe uma classificação incorreta.

A segunda alternativa pode ser apontada como a pior. Nessa é realizada uma detecção disjuntiva, ou seja, as mensagens são detectadas como *spams* se em qualquer um dos componentes do filtro composto a mesma for classificada dessa forma. Esta con-

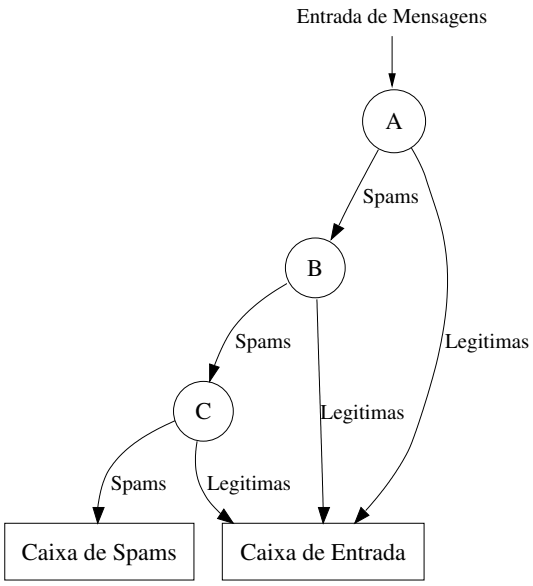

Figura 3.4: 1ºExemplo de Configuração: Detecção Conjuntiva

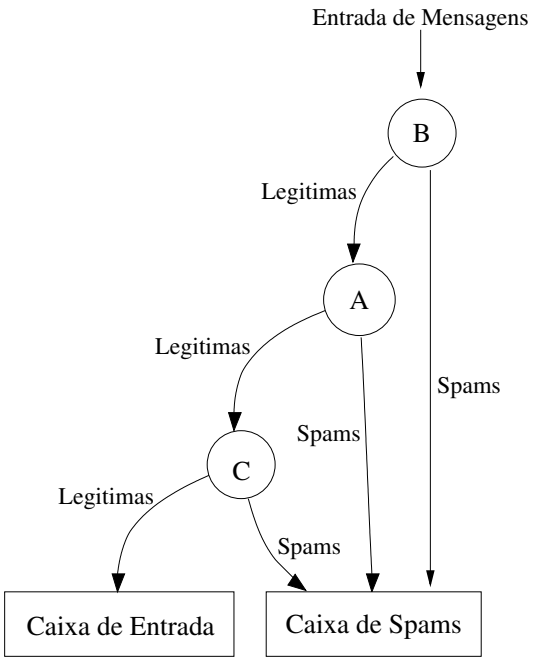

Figura 3.5: 2ºExemplo de Configuração: Detecção Disjuntiva

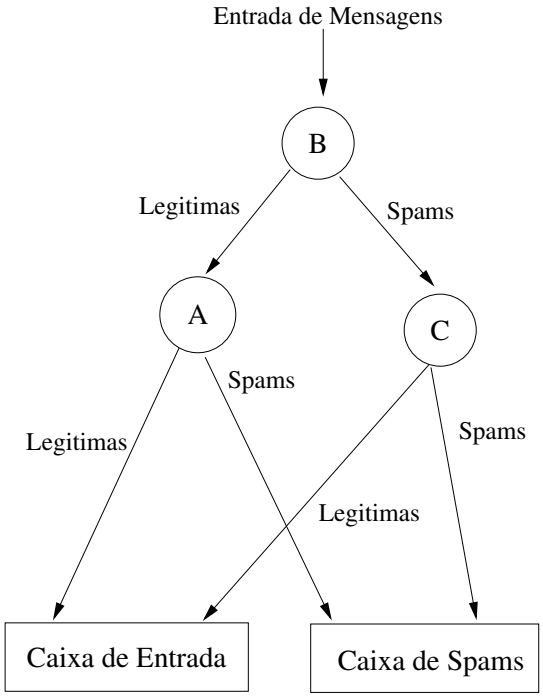

Figura 3.6: 3ºExemplo de Configuração: Detecção Distribuída

figuração apresenta o mesmo problema com relação ao custo de processamento. 20% das mensagens são processadas por todos os filtros e dessas, 50% é classificada incorretamente. Sua efetividade é a metade da primeira opção, apenas 40% das mensagens são classificadas corretamente. Já a terceira configuração é uma opção intermediária. A efetividade dessa é de 60%, porém o custo de processamento é menor, uma vez que nenhuma das mensagens são processadas por todos os filtros (60% são processadas pelos filtros A e B e 40% são processadas pelos filtros B e C).

Estatísticas	1ª Configuração	2ª Configuração	3ª Configuração
# de mensagens spams classificadas corretamente	1	3	3
# de mensagens legítimas classificadas corretamente	7	1	3
# de mensagens spams classificadas incorretamente	1	0	0
# de mensagens legítimas classificadas incorretamente	1	6	4
# de mensagens processadas pelo filtro A	10	6	6
# de mensagens processadas pelo filtro B	7	10	10
# de mensagens processadas pelo filtro C	3	2	4

Tabela 3.2: Resultados do Exemplo

3.6 Considerações

Conforme apresentado neste capítulo, existem muitas propostas de combate a mensagens maliciosas, onde a melhor efetividade é o objetivo principal. Há propostas que visam reduzir os erros, principalmente o número de falsos positivos, porém nem sempre esses dois objetivos são abordados em conjunto. Conforme apresentado em [24], a disseminação de mensagens maliciosas possui características similares aos problemas sociais e como tal apresenta-se impossível de ser eliminada. No entanto, a combinação de várias propostas surge como uma boa oportunidade para a redução deste problema, uma vez que cada técnica possui suas respectivas vantagens e desvantagens. Combinadas podem apresentar um considerável ganho, dado que podem realizar um trabalho cooperado onde os erros de uma podem ser corrigidos pelas demais.

No entanto, não podem ser deixados de lado os custos. Dado que estão sendo adotadas diversas técnicas, a soma dos custos da execução de cada uma pode comprometer a eficiência do sistema como um todo. Além disto, a combinação pode até mesmo gerar um resultado onde os erros de cada técnica são somados gerando um modelo de péssima qualidade.

Dado este cenário, neste trabalho propomos um modelo no qual várias técnicas são combinadas, mas esta combinação não deve ser aleatória. Deve-se buscar critérios que forcem uma combinação específica, aproveitando o melhor que cada técnica tem a oferecer de maneira que o sistema não se torne sobrecarregado e nem comprometa a efetividade da classificação.

A principal vantagem do modelo é a classificação antecipada, ou seja, a possibilidade de se realizar a detecção de mensagens maliciosas a um menor custo. Por outro lado, o principal desafio do modelo é fazer isso sem perder a precisão. Este modelo na verdade incorpora outros existentes. Por exemplo, o *SpamAssassin* pode ser visto como uma lista encadeada de filtros, onde a saída de um filtro é a entrada de outro. A questão fundamental é então determinar quais são os filtros e como eles devem ser estruturados, a qual discutimos a seguir.

Uma proposta inicial para obtenção de uma boa configuração é assunto do capítulo 4, onde apresentamos as etapas que devem ser realizadas bem como uma proposta de algoritmo para a formação de um ou mais grafos.

Capítulo 4

Estratégia de Montagem do Filtro Composto

4.1 Introdução

Neste capítulo apresentamos a estratégia de montagem de um filtro composto que esteja de acordo com o modelo proposto neste trabalho. A principal motivação para definição de uma estratégia de montagem é a possível ineficácia do filtro composto gerado arbitrariamente. Se a combinação dos componentes for feita incorretamente, os erros gerados por cada um podem ser somados ao invés de corrigidos.

Dadas as características do problema, buscamos suporte nas técnicas de mineração de dados. Na seção 4.2 apresentamos uma justificativa para essa decisão, fazemos um paralelo com o problema de classificação, amplamente estudado em mineração de dados. Isto permite apontar as reais contribuições deste mapeamento no contexto deste trabalho. A seção 4.3 pode ser considerada o ponto chave deste capítulo uma vez que é onde serão definidas as restrições e os passos a serem realizados para a obtenção do modelo de filtros compostos de detecção de mensagens maliciosas. Um resumo da metodologia é apresentado na seção 4.4.

4.2 Problema de Classificação

A mineração de dados pode ser definida como um processo não-trivial de identificação em uma base de dados de padrões válidos, implícitos, previamente desconhecidos, potencialmente úteis e compreensíveis [31]. Este processo só é útil em grandes massas de dados, onde a descoberta de informações úteis é uma tarefa extremamente difícil. Esta faz parte de um processo maior, conhecido como Descoberta de Conhecimentos

em Banco de Dados (KDD - *Knowledge Discovery in Databases*) [38]. Este processo pode ser dividido em três etapas:

Preparação de Dados: Nesta etapa os dados são selecionados e limpos (remoção de inconsistências, ruídos e tratamento dos valores ausentes). Também ocorre a integração de informações de diversas fontes, são selecionados atributos que trarão maior ganho para a mineração e também ocorre a transformação dos atributos selecionados para que os mesmos tomem uma forma mais conveniente. Exemplos de transformação são a discretização, normalização ou agregação dos dados.

Mineração de Dados: Nesta etapa são buscados padrões implícitos na base de dados. Esta etapa é composta, de uma maneira geral, por atividades de determinação de regras de associação que visam obter as correlações entre os atributos; técnicas de agrupamento, que visam criar conjuntos de dados com características similares; e técnicas de classificação, que visam criar modelos que descrevam características de classes de dados de forma que, caso não se conheça a classe de uma determinada instância, possa inferir a mesma baseada em algum modelo gerado.

Visualização dos Resultados: Nesta etapa os resultados obtidos com a mineração são apresentados de forma inteligível para os usuários. O objetivo aqui é possibilitar que os resultados sejam interpretados de forma correta. Esta etapa também deve permitir que sejam selecionados os dados mais relevantes para análise.

Conforme já levantado no capítulo 2, a mineração de dados é bastante explorada para o combate às mensagens maliciosas. Uma atenção especial pode ser dada aos trabalhos relacionados às técnicas de classificação [66, 27, 36, 35, 51, 62, 64] baseadas na seleção de atributos. Essas técnicas buscam identificar o menor número de atributos que permitam classificar entidades desconhecidas, ou seja, determinar os atributos com melhor capacidade de discriminação. Nos trabalhos voltados para filtragem de mensagens de correio que utilizam classificadores, em geral, são considerados como atributos válidos as palavras encontradas no vocabulário do sistema [13]. Os valores desses atributos representam a presença ou ausência dessas palavras em uma determinada mensagem e as classes possíveis são apenas duas: maliciosa ou legítima. Para que sejam criados modelos efetivos, são utilizados conjuntos de mensagens previamente classificadas.

Assim como as técnicas tradicionais de mineração de dados, nossa estratégia visa gerar modelos que apresentem a melhor eficácia possível para classificação dos *e-mails* maliciosos ou legítimos. Porém, busca-se também o balanceamento entre eficácia e

custos associados à filtragem de das mensagens. O ideal é que mensagens de fácil classificação sejam identificadas logo nos primeiros filtros que as processarem dentro do filtro composto.

Uma diferença para os mecanismos tradicionais que utilizam técnicas de mineração de dados é que neste trabalho o modelo não está sendo gerado diretamente a partir das palavras do conteúdo das mensagens e sim a partir de um conjunto de componentes que realizam uma classificação prévia das mesmas.

Para fornecer um embasamento para nossa proposta, tomemos a técnica de classificação baseada em árvores de decisão [60, 46]. Nessa, o modelo a ser considerado é uma árvore onde cada vértice representa um atributo da base de dados a ser utilizado na classificação e as arestas representam a decisão a ser tomada. Os vértices folha representam a classificação final. A avaliação de cada instância da base a ser classificada é bastante simples. Partindo da raiz, considerando o valor do atributo raiz, decide-se qual deve ser o próximo atributo a ser avaliado. Quando o valor do atributo na instância avaliada estiver direcionado para algum vértice folha, significa que é o momento de marcá-la como sendo da classe referente ao vértice folha.

Para melhor compreensão, tomemos como exemplo a árvore de decisão representada pela figura 4.1. Esta árvore deve ser utilizada para classificar instâncias de uma base de dados com os atributos $< A, B, C, D, CL >$, onde o atributo CL representa a classe. As possíveis classes consideradas nesta base são C_1, C_2, C_3, C_4, C_5. Sejam $I_A =< 1, 1, 2, 1, ? >$ e $I_B =< 2, 1, 2, 2, ? >$ duas instâncias da base de dados cuja classe é desconhecida. Para a instância I_A, o sistema avalia o atributo A e decide seguir e avaliar o atributo B uma vez que $A = 1$; avalia B e decide que a instância é da classe C_1 uma vez que $B = 1$. Os valores dos demais atributos neste caso não são relevantes e, portanto, a avaliação dos mesmos nem chega a ser realizada, evitando, desta forma, o consumo desnecessário de recursos. Para a instância I_B, o sistema avalia o atributo A e decide seguir e avaliar o atributo C, uma vez que $A = 2$; avalia o atributo C e decide seguir e avaliar o atributo D, dado que $C = 2$; avalia D e decide classificar I_B como C_5. Neste caso, apenas o atributo B não é avaliado.

Sendo uma árvore um grafo direcionado, podemos fazer um paralelo do nosso modelo proposto com este tipo de estrutura. Sendo assim, é possível utilizar técnicas baseadas na geração de árvores de decisão para obtenção de nosso modelo.

Para fazer este mapeamento, podemos considerar que as instâncias da base de dados sejam as mensagens de correio eletrônico, onde os atributos são os componentes e seus valores representam a classificação realizada: mensagem maliciosa/mensagem legítima. Além disto, deve ser considerada como classe das instâncias desta base a classificação final de cada mensagem: mensagem maliciosa/mensagem legítima. Como temos, neste contexto, apenas dois valores possíveis para cada atributo, podemos gerar

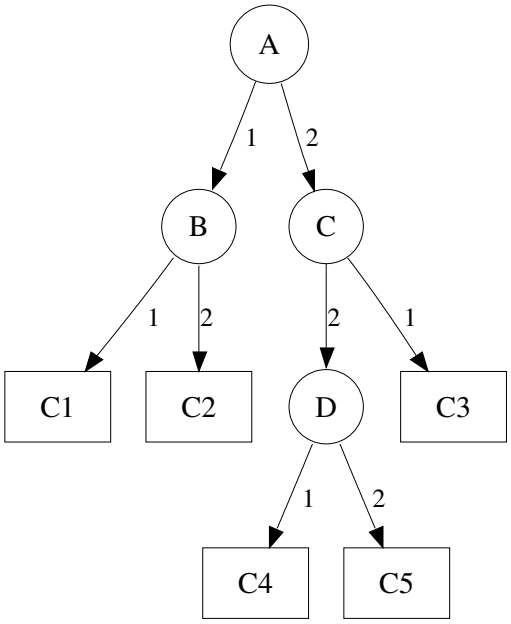

Figura 4.1: Exemplo de árvore de decisão

uma árvore binária de decisão, onde a avaliação de cada atributo representa realmente a classificação realizada pelo mesmo sobre cada mensagem.

Conforme apresentado no capítulo 3, o modelo proposto pode ser representado como um grafo direcionado $G(V, A)$, onde V é o conjunto de vértices do grafo G e representam todos os componentes do filtro composto e A é o conjunto de todas as arestas $a(i, j)$ pertencentes a este mesmo grafo e representam as mensagens que trafegam do componente i para o componente j. Esta representação viabiliza a utilização de técnica de classificação baseada em árvores de decisão para a obtenção de um grafo com maior eficiência para detecção de mensagens maliciosas. Esse grafo será funcionalmente equivalente a uma árvore binária de decisão.

Dado o exemplo acima, dentre as várias alternativas, podemos mapear as classes $\{C_1, C_2, C_3, C_4, C_5\}$ como $\{LEGIT, SPAM, LEGIT, LEGIT, SPAM\}$. Consequentemente, teremos apenas duas classes efetivamente: $\{LEGIT, SPAM\}$, onde $LEGIT$ é a classe das mensagens legítimas, e $SPAM$ é a classe das mensagens maliciosas. Já os valores $1, 2$ podem ser mapeados como L_i, S_i, onde L_i é a identificação de uma mensagem legítima pelo componente i e S_i é a identificação de uma mensagem maliciosa pelo mesmo componente i. Sendo assim, as instâncias I_A e I_B são mensagens classificadas como legítima e não-legítima respectivamente pelo filtro composto.

A montagem da árvore de classificação é assunto abordado na seção 4.3, onde é apresentado um roteiro para a geração do modelo eficiente de detecção de mensagens maliciosas.

4.3 Algoritmo

Esta seção será dedicada ao mecanismo utilizado para geração do filtro composto para detecção de mensagens maliciosas. Para tal, será gerado um grafo direcionado semelhante a uma árvore de decisão através de técnicas de classificação de dados. Neste trabalho, nós referenciaremos esta árvore como uma árvore de detecção de mensagens maliciosas. Uma árvore de detecção, a exemplo de outras árvores, é um grafo acíclico direcionado, mas com a restrição de não haver dois caminhos entre dois vértices alcançáveis. Os passos principais para a sua construção devem ser: selecionar os componentes a serem utilizados no modelo; levantar um conjunto de mensagens previamente classificadas para servirem de base de treinamento; para cada componente considerado, obter a classificação de todas as mensagens de treinamento; e gerar a árvore de detecção. O processo está representado na figura 4.2 e pode ser dividido em quatro etapas distintas. Essas serão discutidas nas próximas seções.

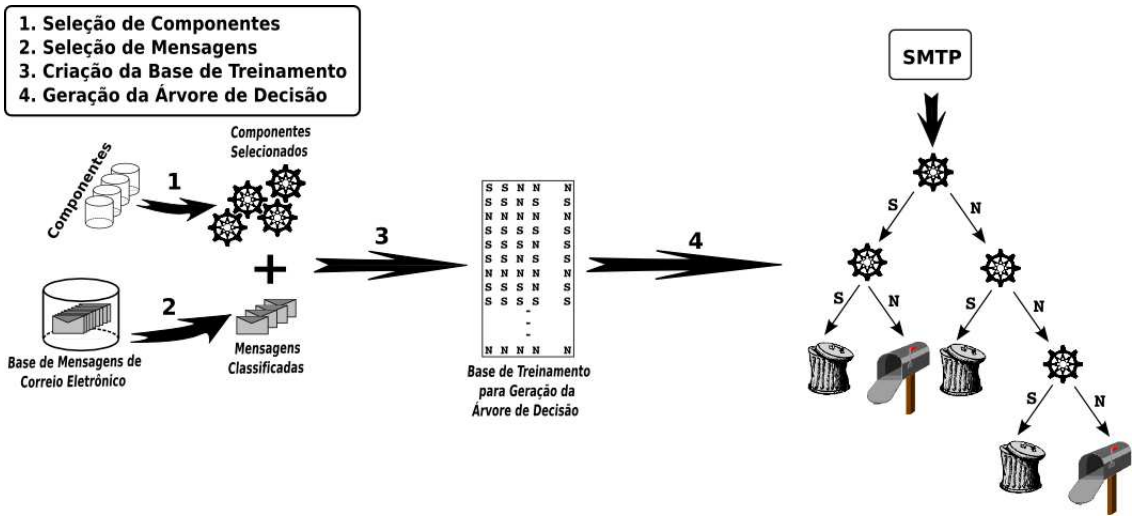

Figura 4.2: Geração do filtro composto de acordo com o modelo proposto

4.3.1 Escolha de Componentes para Filtro

A escolha dos componentes pode ser considerada a etapa mais simples, porém de extrema importância para o filtro como um todo. As demais etapas são diretamente dependentes do resultado desta e o filtro composto pode ter sua efetividade comprometida quando esta seleção é realizada sem o cuidado necessário.

Em primeiro lugar, deve-se definir um ou mais critérios de seleção de tais componentes. Os principais critérios que devem ser atendidos são:

Eficácia do componente: Os componentes selecionados devem ser capazes de realizar uma classificação com precisão mínima, caso contrário provavelmente não

farão diferença ou mesmo aumentarão o número de erros do filtro composto gerado.

Custo de processamento: Este custo pode ser tempo de classificação por mensagem, consumo de CPU, consumo de memória etc. O componente a ser selecionado deve estar abaixo de um valor limiar em termos de custo e este deve ser determinado com base na capacidade computacional do servidor de correio eletrônico no qual este será adotado. Caso o componente não atenda este critério, o mesmo pode comprometer a entrega das mensagens pelo filtro composto gerado.

Custo de Erro de Classificação: Como já definido, não necessariamente a efetividade pode ser utilizada para medir o custo de erro de classificação. Este custo é dependente diretamente dos custos de classificação incorreta de mensagens maliciosas e de mensagens legítimas. Se um componente é capaz de classificar corretamente todos os *spams* e ao mesmo tempo classificar todas as mensagens legítimas também como *spams*, este pode trazer consequências negativas para o filtro composto.

Esses componentes podem ser filtros simples ou até mesmo os mais complexos comumente adotados em sistemas reais [51, 36, 8]. Quanto mais complexo, provavelmente maior será a carga computacional exigida para a classificação. Quando combinados, a carga agregada exigida pelos componentes deve ser avaliada.

Um maior número de componentes simplificados, com baixo consumo computacional, porém, com efetividade provavelmente baixa, pode ser uma boa opção à utilização de filtros poderosos juntos. Esses filtros mais robustos provavelmente irão, isoladamente, consumir maior quantidade de recursos, já os componentes simples irão consumir poucos recursos e permitirão que as mensagens sejam classificadas antes mesmo que todos sejam acionados.

4.3.2 Escolha das Mensagens de Treinamento

O levantamento de um conjunto de mensagens completas (corpo e cabeçalho) previamente classificadas como maliciosas ou legítimas deve ser realizado para geração de uma base de treinamento mais precisa. Essas mensagens devem estar dentro do contexto do ambiente no qual o filtro será aplicado. Assim como outros mecanismos de aprendizado de máquina, a efetividade do filtro composto depende do conhecimento de informações já geradas anteriormente [50]. Esse conhecimento permite que sejam absorvidas lições aprendidas [1].

[1]Entende-se por lições aprendidas as características das mensagens classificadas no passado como maliciosas, além das características das mensagens classificadas no passado como legítimas e os fatores

4.3.3 Preparação da Base de Treinamento

Uma vez obtidas todas as mensagens classificadas, o próximo passo então é submetê-las para a realização da classificação por todos os componentes considerados. O resultado desta classificação é o insumo a ser utilizado na geração da base de treinamento.

Para melhor esclarecer, será utilizado o exemplo apresentado na seção 4.2. Neste, temos os componentes A, B, C e D. Após submetermos as mensagens a todos os componentes, teremos uma base de dados resultante com suas instâncias no seguinte formato: $msg_i =< C_A, C_B, C_C, C_D, C_{Real} >$, onde msg_i é a i-ésima mensagem da base, C_A é a classificação fornecida pelo componente A, C_B é a classificação fornecida pelo componente B, C_C é a classificação fornecida pelo componente C, C_D é a classificação fornecida pelo componente D e C_{Real} é a classificação real da mensagem. Isto se deve ao fato de que, neste trabalho, os atributos não são as palavras dentro do conteúdo das mensagens, como ocorre nas demais propostas envolvendo classificadores, e sim os componentes que realizam a filtragem previamente. Estamos buscando criar um filtro composto que esteja de acordo com um modelo no qual é explorada a capacidade de decisão de cada componente e que um maior número de detecções ocorra de forma antecipada, sem que a maioria das mensagens tenha de ser repassada de um componente para outro.

4.3.4 Geração da Árvore de Detecção

O algoritmo de geração da árvore de detecção de mensagens maliciosas é bastante simples e pode ser visto através dos algoritmos 1, 2 e 3. A idéia básica é a geração da árvore recursivamente. No algoritmo 1 é realizada a inicialização, buscando a base de treinamento e a lista original dos componentes do sistema. O algoritmo que, efetivamente, gera a árvore é o 2. Esse recebe o componente raiz da sub-árvore a ser gerada. Na primeira execução do algoritmo, o valor de entrada é nulo uma vez que, neste ponto, ainda não está definido um componente de origem. A segunda entrada do algoritmo é a base de treinamento. Na primeira execução esta é a base completa. Além dessas, as entradas seguintes são: um dos valores que o componente pode assumir, ou seja, as possíveis classificações realizadas e a lista de componentes.

No procedimento *GeraArvore* busca-se, primeiramente, o melhor componente para aquela base de dados. Entende-se como melhor componente aquele que trará maiores ganhos com relação à capacidade de classificação correta e com relação aos custos a serem considerados. Uma vez selecionado, este componente deve ser removido da lista pois é incluído na árvore. Em se tratando do componente raiz, não vai existir um com-

que separam essas duas classes.

ponente origem e nem mesmo um valor que o liga a esse componente. Portanto, será gerado algo como $MATRIZ(nulo, comp) = \{\emptyset\}$. O passo seguinte é a criação de duas partições da base de dados: uma somente com as instâncias classificadas como maliciosas pelo componente selecionado e outra somente com as instâncias classificadas como legítimas pelo componente selecionado. Uma vez que o componente selecionado classifica as mensagens da base de dados, a seleção dos próximos deve ser realizada dentro de cada universo referente à cada uma das duas classificações realizadas. Esta execução é realizada recursivamente até não haver mais componentes para serem selecionados.

Uma importante otimização deve ser realizada: a poda da árvore. Se gerarmos uma árvore na qual as mensagens devam percorrer todos os componentes para serem classificadas, estamos na verdade gerando um filtro composto sequencial, onde a sequência é determinada pela decisão de cada componente do filtro. A estrutura resultante neste caso é bastante carregada e não traz muitos ganhos, uma vez que não existe a possibilidade da antecipação da detecção de mensagens maliciosas. Se pudermos interromper a criação de ramos na árvore em um determinado momento, este indicará que é a hora da realização da classificação efetiva, permitindo desta forma que as mensagens sejam entregues ou descartadas antecipadamente sem que tenham de percorrer todos os componentes.

A função *RealizaPoda* adotada é utilizada com o objetivo de verificar se o componente selecionado traz ganhos significativos ou não. O critério que define este ganho é definido caso a caso. O algoritmo C4.5 apresentado em [60] considera o erro de classificação como critério de poda.

A partir de um certo momento, por mais que sejam adicionados componentes no filtro, o erro de classificação não é reduzido de forma significativa, ou porque o novo componente a ser adicionado apresenta baixo poder de classificação ou porque as mensagens já receberam uma classificação precisa e poderiam ser entregues ou descartadas. Portanto, neste trabalho adotaremos como critério de poda o erro de classificação, uma vez que existe um limiar a partir do qual os componentes adicionais só criarão maiores custos.

A função *RealizaPoda* pode ser definida pela equação 4.1. Se o erro estiver abaixo do limiar ϵ, é o momento da realização da poda, caso contrário, continua gerando as sub-árvores com o objetivo de melhorar a efetividade do filtro. Neste trabalho, o limiar ϵ é ajustado de forma empírica.

$$RealizaPoda = \begin{cases} sim, \ se\ erro < \epsilon \\ nao, \ caso\ contrario \end{cases} \tag{4.1}$$

O algoritmo 3 é utilizado para a seleção do melhor componente. O mesmo realiza uma consulta à uma função de utilidade do componente, que retorna o valor do mesmo

para a classificação daquele conjunto específico de dados. Os algoritmos tradicionamente utilizam a função ganho de informação para medir essa utilidade [60, 46]. O ganho de informação fornece suporte para identificar o quanto um determinado atributo, ou componente no contexto deste trabalho, auxilia na determinação da classe das instâncias da base de dados. Quanto maior o ganho de informação de um atributo, maior é o potencial do mesmo para a identificação de classes das instâncias da base de dados. O ganho de informação é diretamente relacionado com o valor de entropia. Essa métrica determina o grau de confusão dos dados. Quanto maior este valor, maior é a organização e mais fácil é a identificação das classes baseada no atributo e quanto menor é este valor maior é o grau de confusão, o que impede uma definição precisa da classe de uma instância baseada no valor do atributo em questão. O ganho de informação pode ser determinado pela equação 4.2. Nesta, $H(classe)$ é a entropia do atributo classe da base de dados e é definida na equação 4.3. Já $H(classe|atributo)$ é a entropia condicional do atributo classe e é definida na equação 4.4. O valor de m neste caso é 2, uma vez que temos apenas duas classes a serem consideradas: mensagens maliciosas/mensagens legítimas. $Prob(atributo = valor_i)$ é a probabilidade do atributo ter o valor v_i. $H(classe|atributo = v_i)$ é a entropia da classe somente no conjunto de dados cujo atributo tenha valor igual a v_i.

Em nosso trabalho, conforme já relatado, o atributo é na verdade um componente do sistema de filtro, cada valor é a classificação que o mesmo realiza sobre a instância da base de treinamento e as classes são as classificações finais de cada instância. No entanto, não iremos optar por essa métrica uma vez que a mesma não leva em consideração que os valores associados a cada componente interferem diretamente na efetividade do sistema. Um exemplo claro é a situação onde o componente realiza uma classificação invertida na qual todos as mensagens maliciosas são classificadas como legítimas e todas as mensagens legítimas são classificadas como maliciosas. Pela métrica ganho de informação, esse componente é uma boa escolha uma vez que o mesmo consegue gerar uma classificação para a base de dados, porém apresenta uma péssima efetividade na detecção de mensagens maliciosas. Em situações onde um filtro realiza uma classificação invertida, uma solução alternativa seria considerar tal inversão. Isto significa classificar como *spams* as mensagens que esse filtro tenha marcado como legítimas e classificar como legítimas as mensagens que esse tenha marcado como *spams*. No entanto, determinar quando a inversão deve ser aplicada é um problema de difícil solução.

$$GanhoInfo = H(classe) - H(classe|atributo) \tag{4.2}$$

$$H(classe) = -\sum_{i=1}^{m} p_i log_2 p_i \tag{4.3}$$

$$H(classe|atributo) = \sum_{i=1}^{m} Prob(atributo = v_i) H(classe|atributo = v_i) \tag{4.4}$$

Algoritmo 1: Criação da Árvore de Detecção

```
Resultado: MATRIZ contendo o grafo gerado
1 begin
2   LST_COMPONENTES ← lista de componentes do sistema
3   BASE_TREINAMENTO ← base de treinamento previamente gerada
4   for i ∈ LST_COMPONENTES do
5     for j ∈ LST_COMPONENTES do
6       MATRIZ(i,j) ← {∅}
7   GeraArvore(∅,BASE_TREINAMENTO,∅,LST_COMPONENTES)
8 end
```

A função de utilidade a ser adotada neste contexto pode ser considerada como uma peça chave para alcançar nosso objetivo principal e, portanto, será dedicada uma seção própria para descrevê-la com maiores detalhes.

4.3.5 Função de Seleção de Componentes

O principal objetivo deste trabalho é a obtenção de um modelo composto por múltiplos filtros onde a efetividade seja maximizada, bem como os custos de processamento e de erro de classificação sejam minimizados. A seleção dos componentes durante a montagem da árvore de detecção interfere diretamante nos resultados e portanto deve ser feita de forma que esses critérios sejam considerados.

Desta forma, a função de utilidade apresentada no algoritmo 3, além do componente e da base de dados, deve considerar a função $\Gamma(efetiv, custo_{proc}, C_{erro})$ definida no capítulo 3. Lembrando que $efetiv$ é a efetividade do componente para classificação da base considerada; $custo_{proc}$ é o custo de processamento por mensagem deste componente e C_{erro} é o custo do erro de processamento.

No entanto, existe uma restrição não-trivial para a combinação desses três critérios: são valores definidos em unidades completamente diferentes. A combinação dos mesmos e até a adoção de um fator que garanta a importância de um critério sobre o outro fica comprometida. Neste caso, uma estratégia é realizar uma normalização de forma que todos sejam convertidos para uma mesma unidade e fiquem dentro de uma mesma

Algoritmo 2: GeraArvore

Entrada: O componente *origcomp* raiz calculado na iteração anterior. É *nulo* na primeira execução da *GeraArvore*

Entrada: Partição da base de dados *basedados* contendo o subconjunto da base de dados selecionado na iteração anterior. Na primeira execução da *GeraArvore* este recebe toda a base de dados de treinamento

Entrada: Valor *val* do atributo *origcomp* considerado nesta iteração. É *nulo* na primeira execução da *GeraArvore*

Entrada: Lista de componentes *listacomp* a ser considerado nesta iteração. Na primeira execução da *GeraArvore* recebe a lista completa dos componentes

Resultado: MATRIZ contendo o grafo gerado

```
1  begin
2      if listacomp ≠ ∅ E basedados ≠ ∅ then
3          comp ← SelecionaMelhorComponente(listacomp,basedados)
4          listacomp = listcomp − {comp}
5          if naoRealizaPoda(comp) then
6              MATRIZ(origcomp, val) ← comp
7              baseSpam ← {dados ∈ basedados | basedados(comp) = spam}
8              baseNSpam ← {dados ∈ basedados | basedados(comp) = naospam}
9              GeraArvore(comp,baseSpam,spam,listacomp)
10             GeraArvore(comp,baseNSpam,naospam,listacomp)
11 end
```

Algoritmo 3: SelecionaMelhorComponente

Entrada: Lista de componentes *listacomp* a ser considerado nesta iteração. Na primeira execução da *SelecionaMelhorComponente* recebe a lista completa dos componentes

Entrada: Partição da base de dados *basedados* contendo o subconjunto da base de dados selecionado na iteração anterior. Na primeira execução da *SelecionaMelhorComponente* este recebe toda a base de dados de treinamento

```
1  begin
2      maiorUtl ← 0
3      for c ∈ listcomp do
4          utl ← FuncaodeUtilidade(c,basedados)
5          if utl > maiorUtl then
6              selComp ← c
7              maiorUtl ← utl
8      retorna c
9  end
```

escala, porém sem comprometer seu comportamento original. No caso da efetividade, uma vez que ela é definida no capítulo 3 pela equação 3.1, pode-se facilmente perceber que a mesma já se encontra normalizada e seus valores variam entre 0 e 1. Sendo assim, somente devem ser normalizadas as variáveis $custo_{proc}$ e C_{erro}.

A função de normalização adotada pode ser expressa por meio da equação 4.5. É possível perceber que é necessário conhecer os limites de cada variável a ser normalizada. Com relação ao custo de processamento, esses limites são facilmente definidos. O limite inferior é exatamente o valor, dentre todos os componentes considerados, do menor custo de processamento. O mesmo deve ocorrer com o limite superior, neste caso o valor considerado é o maior custo de processamento dentre os custos de todos os componentes do sistema.

$$N = \frac{valor - limite_{inferior}}{limite_{superior} - limite_{inferior}} \tag{4.5}$$

Sendo o custo do erro de classificação também definido no capítulo 3 pela equação 3.2, é facilmente perceptível que seu valor pode variar de 0 (nenhuma mensagem é classificada incorretamente) até $M_L * \alpha + M_S * \beta$, onde M_L é o total de mensagens legítimas da base de dados considerada e M_S é o total de mensagens maliciosas desta mesma base. Esse valor superior reflete a situação na qual um filtro realiza a classificação invertida.

No processo de cálculo da função de utilidade de cada componente devem ser considerados como entrada os seguintes dados dos componentes: custo de processamento por mensagem, o número de mensagens legítimas classificadas como maliciosas, o número de mensagens maliciosas classificadas como legítimas e a efetividade geral. Devem ser considerados ainda os valores máximo e mínimo do custo de processamento por mensagem dos componentes, o total de mensagens legítimas da base de dados e o total de mensagens maliciosas da base de dados.

Feita a normalização, a função $\Gamma(efetiv, custo_{proc}, C_{erro})$ torna-se passível de ser calculada. Para efeito de exemplificação, a função adotada no estudo de caso apresentado no capítulo 5 é $\Gamma(efetiv, custo_{proc}, C_{erro}) = \varrho * efetiv + \kappa * custo_{proc_{norm}} + \rho * C_{erro_{norm}}$, onde ϱ, κ e ρ são fatores de ajustes que determinam a importância de cada um dos critérios no processo de geração da árvore de detecção. A princípio esses terão o mesmo peso no processo, portanto, $\varrho = \kappa = \rho = 1$.

4.4 Considerações

Neste capítulo apresentamos como a geração de um filtro composto que esteja de acordo com o modelo proposto neste trabalho pode ser realizada através de técnicas

inspiradas em mineração de dados. Focamos aqui mais especificamente em técnicas de classificação de dados. Basicamente, na mineração de dados, normalmente, são buscados modelos que permitem a classificação eficaz de grandes massas de dados. Neste trabalho buscamos gerar um modelo eficaz, mas que também atenda a critérios como custo de processamento e erro de classificação. O modelo resultante deve explorar a capacidade de decisão de cada componente do filtro composto de forma que, mensagens que sejam facilmente classificadas possam receber uma classificação definitiva nos primeiros componentes que as processarem.

Uma vez feito o mapeamento, nos direcionamos para a estratégia propriamente dita. Primeiramente abordamos a questão da importância dos critérios na seleção dos componentes. Tal seleção causa um impacto direto no modelo resultante. As duas etapas seguintes apresentadas atuam na preparação da base de treinamento.

Apresentamos ainda como a árvore de detecção deve ser gerada. Em trabalhos relacionados à mineração de dados, a métrica que atua influenciando na geração de uma árvore mais precisa é o ganho de informação. No entanto, mostramos que no nosso modelo esta não representa uma boa opção. Apresentamos que, para alcançarmos nosso objetivo final, devem ser considerados os critérios: efetividade, custo de processamento e custo de erro de classificação.

Foi demonstrado também que os critérios adotados, em seu estado natural, não podem ser combinados diretamente pois não se encontram em uma unidade e escala única. Sendo assim, a normalização mostrou-se uma alternativa satisfatória. Por meio de uma função genérica de combinação dos critérios, podemos controlar quais são os mais importantes para a geração da árvore de detecção. Com as diversas combinações de importância, poderemos avaliar o impacto nos resultados obtidos em cada modelo gerado.

Uma última observação é que a árvore de detecção é uma das possíveis estratégias de montagem de modelo de filtro composto. Como o mesmo pode ser representado como um grafo acíclico direcionado, existem inúmeras estratégias alternativas para montagem do mesmo. Porém, esse assunto é objeto de trabalhos futuros.

Capítulo 5

Estudo de Caso

5.1 Introdução

Neste capítulo são apresentados os componentes que compõem o filtro composto montado neste trabalho. A base desses componentes é a divisão de um filtro real em vários elementos que realizam algum tipo de avaliação nos *e-mails*. O objetivo é obter um filtro composto que possa ser comparado diretamente com uma proposta real e que seja efetivamente utilizada.

Dentre diversas propostas de filtros de mensagens maliciosas como [51, 36, 70, 8], será destacado neste trabalho o *SpamAssassin* [8]. Esse filtro tem como base principal a avaliação de mensagens através da utilização de expressões regulares. Essas regras podem ser divididas em grupos que correspodem aos objetivos da avaliação, como por exemplo, grupo de avaliação de cabeçalho das mensagens. Uma importante vantagem do *SpamAssassin* é a sua implementação como um sistema de código livre, o que permite um estudo mais detalhado de seu funcionamento e uma rápida adaptação do seu código fonte às necessidades deste trabalho.

Os componentes que utilizamos foram gerados a partir da divisão do *SpamAssassin*. Essa quebra foi baseada nos tipos das regras implementadas no filtro.

O restante deste capítulo encontra-se organizado da seguinte forma. A seção 5.2 apresenta um estudo sobre o *SpamAssassin*, dando ênfase aos tipos de testes realizados por essa ferramenta. Na seção 5.3 apresentamos os componentes selecionados para a montagem do filtro composto cujos resultados encontram-se no capítulo 6. Nessa seção também apresentamos uma justificativa para tal seleção. A seção 5.4 é dedicada ao estudo detalhado do funcionamento de cada um dos componentes, fazendo comparações entre os mesmos com relação ao tempo de processamento, número de regras implementadas e efetividade.

5.2 O *SpamAssassin*

O *SpamAssassin* é uma ferramenta desenvolvida sobre a linguagem de programação *Perl* e é baseada na análise de conteúdo de mensagens através de regras. Essas regras são, geralmente, implementadas por meio de expressões regulares que visam buscar padrões conhecidos nas mensagens avaliadas. Cada regra possui uma pontuação, podendo ser negativa ou positiva.

O funcionamento dessa ferramenta é bastante simples. Primeiramente deve ser escolhida uma nota de corte que serve para separar as mensagens classificadas como maliciosas das mensagens legítimas. Para cada mensagem que é testada, todas as regras são processadas. Essa mensagem, no início do processo, recebe o valor zero como nota inicial. Quando um determinado padrão é encontrado, a nota da mensagem é incrementada com os pontos referentes à respectiva regra. Desta forma, ao final do processo de classificação, o *e-mail* tem uma nota final que é comparado com o valor de corte. Se essa nota estiver acima desse valor, o *e-mail* é classificado como malicioso, caso contrário, é classificado como legítimo.

Para melhor exemplificar, vamos supor que as regras implementadas no *SpamAssassin* de um determinado sistema de correio eletrônico sejam R_A, R_B, R_C e R_D cujos pontos de avaliação sejam, respectivamente, 5 -3, 2 e 1. A nota de corte é definida como 4. Um exemplo de resultado obtido após a submissão das mensagens m_0, m_1, m_2 e m_3 à este sistema está disponibilizado na tabela 5.1. A única mensagem classificada como maliciosa é a m_1. Mesmo a m_2, recebendo a nota alta referente à regra R_A, é classificada como legítima pois encontra-se exatamente no limiar. Este exemplo serve para demonstrar que, basicamente, regras com pontuação positiva buscam identificar padrões encontrados em *spams*, já as regras com pontuação negativa buscam padrões comumente encontrados em mensagens legítimas.

	Regras				Nota Final
	R_A	R_B	R_C	R_D	
m_0	0	-3	0	1	2
m_1	5	0	0	1	6
m_2	5	-3	2	0	4
m_3	0	0	2	1	3

Tabela 5.1: Tabela de Exemplo de Avaliação de Mensagens no *SpamAssassin*

A escolha da nota de corte é um ponto chave no processo. Notas muito baixas tornam o sistema extremamente rigoroso, fazendo com que ocorram muitos falsos positivos. Em contrapartida, quando estas são muito altas, ocorre então um alto número de falsos negativos [8]. Em nenhuma dessas situações a efetividade do sistema é satisfató-

ria. O valor ideal para a nota de corte dependerá do padrão de utilização do sistema de correio eletrônico. Como exemplo, em [8] recomenda-se que em Provedores de Serviços de Internet (ISP) esse valor deve ser mais conservador, variando entre 8 e 10 e, em caso de sistemas onde as mensagens classificadas como maliciosas são sumariamente descartadas, este valor deve ser superior a 15.

A maioria das regras são implementadas pelos próprios administradores de sistemas de correio eletrônico e devem ser revisadas e ajustadas periodicamente. Isto deve ocorrer porque os transmissores de mensagens não desejadas aprimoram suas sofisticadas técnicas a cada nova proposta de combate [24].

Apesar dessas regras variáveis, existe um conjunto padrão que é utilizado para implementação das técnicas de combate presentes no *SpamAssassin*, as quais são detalhadas na próxima seção.

5.2.1 Testes Implementados no *SpamAssassin*

Todos os testes realizados pelo *SpamAssassin* são baseados em algum tipo de regra. Na definição desses, em [8] são apresentadas 5 classes de testes distintas sobre as quais são criados todos os testes. Essas classes podem ser separadas em duas categorias maiores: local e remota. A primeira refere-se aos testes de conteúdo baseados em regras disponíveis localmente no servidor. Já a segunda é baseada em serviços oferecidos por terceiros. Nessa, as mensagens são enviadas para servidores externos que são dedicados à identificação de assinaturas de mensagens maliciosas. Nas próximas seções é feita uma apresentação de cada uma dessas categorias.

5.2.1.1 Análise Local de Mensagens

Na análise local realizada pelo *SpamAssassin*, são encontradas 4 classes de testes distintos: *body*, *header*, *rawbody* e *uri*. Conforme já relatado, a base desses testes é a análise de conteúdo das mensagens, podendo ser feitas por meio de expressões regulares ou mesmo através de mecanismos de aprendizado, como redes bayesianas. Nessa seção apresentamos, de maneira sucinta, essas 4 classes.

Testes com Regras de Cabeçalhos (*header*): Regras criadas exclusivamente para avaliação do cabeçalho das mensagens.

Testes com Regras de Corpo Puro (*body*): Regras criadas exclusivamente para avaliação do corpo da mensagem. No entanto, neste contexto considera-se como

corpo somente a parte textual da mensagem, sendo descartados todos objetos definidos como *"non-Text MIME"*[1], as quebras de linhas e os objetos HTML.

Em algumas dessas regras também é adotada a rede bayesiana. Através dessas, os *spams* são identificados por meio de avaliação de combinação de conjuntos de palavras. No entanto, para que funcione corretamente é necessário que ocorra um treinamento com um número mínimo de mensagens. É recomendável que a base de teste inicial seja composta por 200 mensagens maliciosas distintas e por 200 mensagens legítimas distintas [4, 54]. Apesar do treinamento inicial, a ferramenta permite que o filtro seja constantemente treinado de forma a aumentar a sua precisão da classificação.

Testes de *URI*: São testes baseados em regras definidas para busca de padrões em todas as URI's presentes na parte HTML das mensagens. Normalmente são utilizadas para buscar endereços de sites de propagandas vinculadas por meio das mensagens maliciosas.

Regras de Corpo sem Pré-Processamento (*rawbody*): São regras similares às regras de corpo puro. A diferença aqui é que não são executados determinados pré-processamentos nas mensagens. Basicamente, são mantidas algumas marcações (*tags*) HTML e quebras de linha. Dessa forma, é possível que sejam definidas regras que busquem informações sobre mensagens maliciosas que estejam camufladas na porção HTML das mesmas.

5.2.1.2 Análise Remota de Mensagens

Nesta categoria encontra-se a classe definida como *full* em [8]. Nessa estão definidas as regras que fazem consultas a servidores externos ao sistema de correio eletrônico. Nesses servidores são normalmente analisados o histórico dos remetentes das mensagens, além de buscarem assinaturas de *spams*. Na atual implementação do *SpamAssassin* são considerados 3 tipos de servidores, os quais são descritos a seguir.

DCC (*Distributed Checksum Clearinghouse*): Este é um sistema colaborativo e distribuído de filtragem de mensagens maliciosas baseado na avaliação dos clientes agregados ao sistema. O servidor DCC trabalha com o conceito de assinatura, gerando um número denominado *checksum*. Os clientes associados diretamente ao sistema, geram um tipo de pontuação quando consideram as mensagens como *spam* e reportam este valor para o servidor. Nesse é realizada uma totalização e só

[1] Tipo não textual. Esse é definido pelo protocolo MIME(*Multipurpose Internet Mail Extensions*), que permite a troca de vários tipos de arquivos pelos usuários de correio da Internet

a partir de um determinado limite é que uma mensagem é considerada maliciosa ou não. O *SpamAssassin*, ao enviar uma mensagem para o servidor DCC, espera que este retorne sua classificação.

***razor*:** Este é um sistema colaborativo usado para bloqueio de *spams*. Aqui, os servidores mantém atualizados catálogos compartilhados das mensagens maliciosas que estão sendo propagadas e que são constantemente consultadas pelos clientes de correio. A detecção também é baseada em um mecanismo de assinaturas de mensagens que frequentemente identifica mudanças no conteúdo dos *spams*. Este sistema leva em consideração as notificações enviadas pelos usuários, sendo essas validadas por meio dos relatórios já gerados. Essas notificações são utilizadas para computar os valores de confiança associados às assinaturas geradas para as mensagens.

***pyzor*:** Assim como o *razor*, este é um sistema colaborativo usado para bloqueio de mensagens maliciosas também baseado em identificação por meio de assinatura de mensagem. Os clientes reportam as mensagens maliciosas para os servidores *pyzor* e esses geram a assinatura dessas mensagens. Este sistema é totalmente baseado no *razor* e foi implementado para ser um sistema de código livre, servindo como uma alternativa ao servidor *razor* descrito anteriormente.

5.3 Componentes do Filtro Composto

Uma vez conhecidas as características principais do *SpamAssassin*, torna-se possível partir para a busca de componentes derivados das técnicas implementadas. Uma sugestão inicial é o mapeamento direto das classes de testes definidas em [8]. Neste caso teremos 6 filtros mais específicos que poderiam compor o filtro composto.

No entanto, se considerarmos a implementação de um componente com o teste da segunda categoria aqui apresentada, o ambiente experimental estará comprometido, uma vez que os resultados serão dependentes de fatores externos. O objetivo deste trabalho é apenas levantar um estudo de caso para a realização de uma validação do modelo proposto. Dessa forma, é plenamente viável a utilização apenas dos componentes baseados na categoria de análise local. Sendo assim, serão considerados 4 componentes que estão nomeados aqui como: *header*, *body*, *uri*, *rawbody*.

5.4 Estudo dos Componentes

Nesta seção faremos um detalhamento do comportamento dos componentes selecionados, avaliando as configurações ideais para os mesmos, compreendendo os custos de processamento e a efetividade de cada um.

Conforme já relatado, uma configuração básica do filtro *SpamAssassin* é a nota de corte. Quando esta é muito baixa, o número de falsos positivos aumenta consideravelmente, enquanto que quando esta é alta, o aumento ocorre no número de falsos negativos. Para efeito de comparação, estamos adotando o valor padrão da ferramenta para este limiar. Entretanto, se tomarmos esta mesma decisão para os componentes gerados estaremos negligenciando o fato que em cada um teremos menos regras sendo avaliadas. As notas assinaladas às mensagens em cada componente são potencialmente diferentes, uma vez que regras que somariam valores positivos e negativos não estão presentes em um ou outro componente.

Para comprovar esta alteração, na tabela 5.2 estão disponibilizadas as notas médias providas por cada componente gerado e pelo filtro original. Esses valores foram obtidos a partir da carga de mensagens públicas fornecidas pelo próprio *SpamAssassin* [9]. Em todos os casos, o coeficiente de variação CV[2] apresentou-se bastante alto. Esse resultado demonstra que as notas geradas para as mensagens não seguem um comportamento homogêneo em nenhum dos componentes. A causa desse comportamento é a dependência direta das notas com relação ao conteúdo das mensagens.

	Componentes				
Carga	***body***	***header***	***rawbody***	***uri***	***Spamassassin* Padrão**
Todas as Mensagens	1,60/1,67	-1,00/-1,69	0,04/6,69	0,22/4,07	0,86/4,62
Mensagens do tipo *Spam*	4,87/0,87	0,51/3,64	0,19/2,84	0,91/1.66	6,31/0,85
Mensagens Legítimas	0,84/1,52	-1,36/-1,07	0,01/26,64	0,05/9,75	-0,40/-5,34

Tabela 5.2: Notas médias/CV de cada filtro em cada carga

Comparando as notas médias e os totais de regras apresentados na tabela 5.3 podemos perceber que os dois primeiros componentes fazem uma avaliação mais rigorosa, onde um grande número de regras positivas são avaliadas. As regras que contribuem negativamente também estão distribuídas nos dois primeiros componentes. Isto faz com que o número de regras negativas, que contribuem para a redução das notas, seja menor em cada um desses componentes do que no filtro original. Já nos dois últimos, o número de regras é muito menor, logo a soma de suas notas acaba sendo inferior.

[2]Coeficiente de Variação é a razão entre o desvio padrão e a média

	Componentes				
Número de	***body***	***header***	***rawbody***	***uri***	***Spamassassin* Padrão**
Regras	1647	344	13	31	2035
Regras Positivas	1621	315	13	31	1980
Regras Negativas	26	29	0	0	55

Tabela 5.3: Totais de regras

Definir que as notas de corte nos componentes devem ser similares à nota de corte configurada no *SpamAssassin* padrão irá fazer com que a efetividade desses componentes fique comprometida. Uma forma de fazer uma escolha mais acertada é avaliar a efetividade de cada filtro na base de treinamento já citada variando-se a nota de corte. Os gráficos das figuras 5.1, 5.2, 5.3 e 5.4 demonstram claramente como tal nota impacta na efetividade dos componentes. Nesses gráficos variamos a nota de corte (eixo X) e verificamos os percentuais de mensagens maliciosas classificadas corretamente, mensagens legítimas classificadas corretamente e a efetividade geral: a razão entre o total de mensagens classificadas corretamente e o total de mensagens.

A seleção da nota de corte vai variar dependendo do contexto onde os componentes serão integrados. A razão dessa variação é que, em todos os casos, há valores para os quais a efetividade é máxima, porém, o número de mensagens maliciosas classificadas corretamente é baixo. Existe também uma nota onde os três percentuais chegam a um equilíbrio. No nosso caso, vamos selecionar as notas de corte para cada componente baseando-se na efetividade geral. Quando esses componentes forem integrados, espera-se gerar uma configuração na qual os erros sejam amenizados. As notas selecionadas para o componente *body*, *header*, *rawbody* e *uri* foram, respectivamente: $3,0$; $-0,5$; $0,1$ e $0,1$.

Uma vez ajustadas as notas de corte, é possível conhecer os resultados gerados por cada componente. Os mesmos estão dispostos na tabela 5.4. É possível perceber que nenhum dos componentes representa uma boa opção para detecção de mensagens maliciosas, uma vez que, dado o ajuste feito, todos estão priorizando a classificação correta das mensagens legítimas. O ajuste feito aqui não deve ser relacionado diretamente à natureza da carga. O que buscamos foi a identificação de uma nota de corte mais precisa do que simplesmente escolher um valor aleatório. Este ajuste poderia ser feito baseado em qualquer outra base previamente classificada.

O gráfico da figura 5.5 apresenta um comparativo dos componentes gerados com relação ao tempo de processamento. É facil perceber que o pior desempenho é apresentado pelo componente *body*, seguido do *header*. Os outros dois componentes não

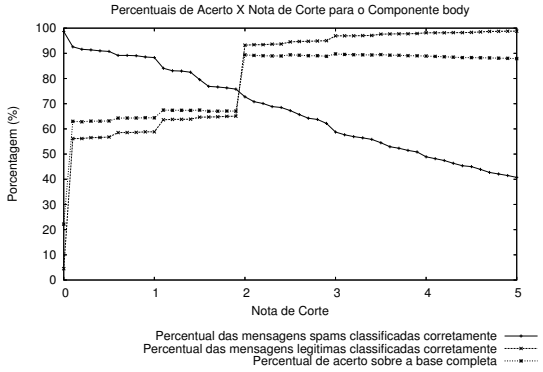

Figura 5.1: Variação da Nota de Corte para o Componente *body*

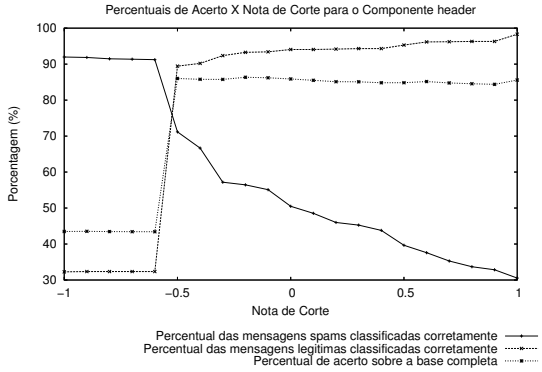

Figura 5.2: Variação da Nota de Corte para o Componente *header*

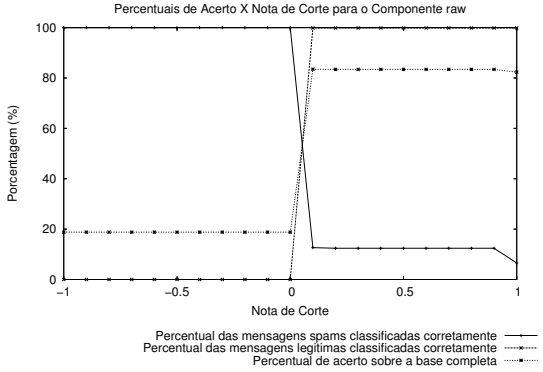

Figura 5.3: Variação da Nota de Corte para o Componente *rawbody*

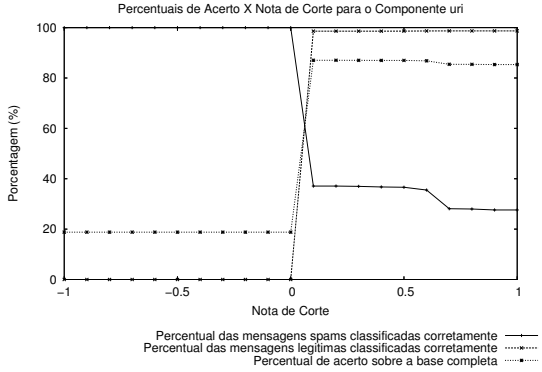

Figura 5.4: Variação da Nota de Corte para o Componente *uri*

	body	***header***	***rawbody***	***uri***
Efetividade	0,89	0,86	0,83	0,87
% de Mensagens *Spams* Detectadas	61,56	71,17	12,65	37,10
% de Mensagens Legítimas Detectadas	95,02	89,44	99,86	98,59

Tabela 5.4: Dados referentes aos componentes

apresentaram diferenças significativas. A justificativa para esse cenário é o número de regras associado a cada componente. O que realiza uma avaliação com um maior número de regras é exatamente o *body*, seguido do *header*. A diferença entre o total de regras do *rawbody* e do *uri* não é significativa, sendo assim, os tempos observados foram comparáveis. A tabela 5.5 contém os tempos médios gastos por cada componente para processar cada uma das mensagens e a soma de todos os tempos de processamento observados para cada um dos componentes. O CV demonstra que os tempos observados não são homogêneos.

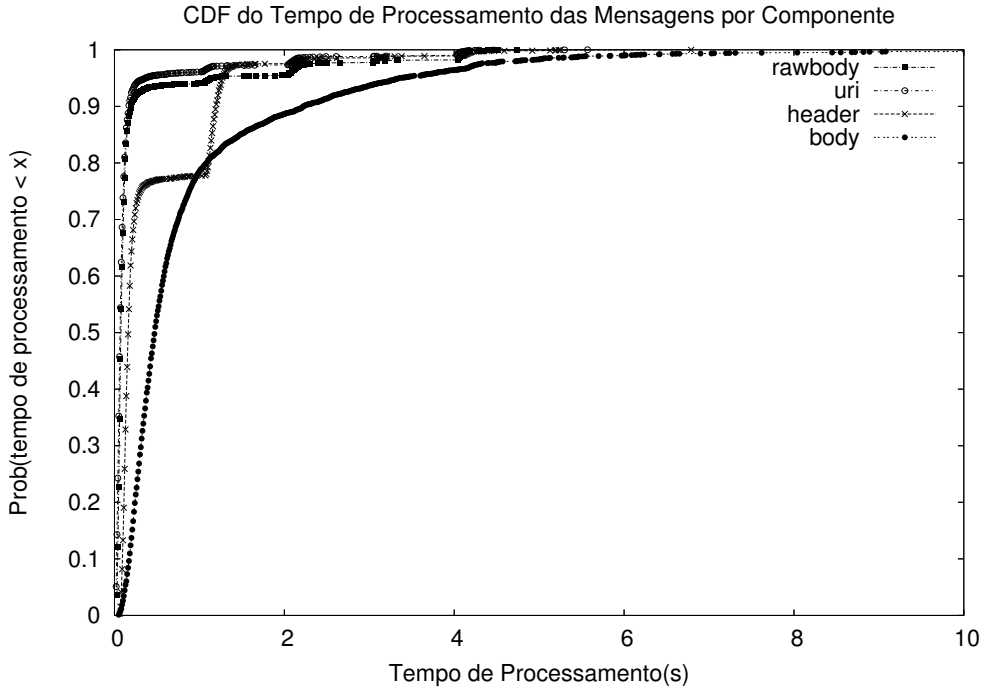

Figura 5.5: Tempo de processamento(acumulado) das mensagens por componente

	body	***header***	***rawbody***	***uri***
Tempo médio	0,92	0,44	0,24	0,18
Coeficiente de Variação	1,72	1,44	2,72	2,92
Tempo Total	4042.67	1941.69	1061.37	771.28

Tabela 5.5: Tempo médio de processamento das mensagens por componente

5.5 Considerações

Neste capítulo foi apresentado como estudo de caso o *SpamAssassin*, um filtro de grande utilização, não limitado a uma única técnica de filtragem e com ótimos resultados referentes à efetividade frente a outras propostas. Este servirá como base para a montagem do filtro composto seguindo o modelo proposto neste trabalho.

Apresentamos em detalhes o funcionamento do *SpamAssassin*, além dos pontos de quebra do mesmo. Partindo de tais pontos, geramos os componentes para o filtro composto e fizemos uma breve avaliação dos mesmos, levando em consideração o tempo de processamento, a efetividade e o número de regras utilizadas em cada um. Estes componentes serão utilizados no capítulo 6 para geração e avaliação de um filtro composto.

Capítulo 6

Avaliação Experimental

6.1 Introdução

Neste capítulo avaliamos experimentalmente o modelo proposto e a estratégia de árvore de detecção de mensagens maliciosas. A avaliação foi realizada em duas etapas distintas: comparativo do filtro composto com o *SpamAssassin*; avaliação das árvores de detecção geradas a partir da variação dos parâmetros de aeração da mesma.

Em cada uma das etapas foi utilizada uma carga de trabalho específica. Na primeira, foi utilizada uma carga, previamente classificada, disponilizada pelo projeto *SpamAssassin* [8]. Utilizamos esta carga para avaliar a instância do *SpamAssassin* utilizada e para compará-la com o filtro composto gerado. Já na segunda etapa foi utilizada uma carga proveniente de um servidor de correio eletrônico real. Devido a restrições existentes no servidor, a coleta das mensagens foi realizada antes do processamento no filtro de mensagens maliciosas pertencente ao sistema.

Em cada uma das etapas, utilizando os componentes gerados no capítulo 5, geramos diferentes árvores de detecção para parâmetros distintos. Na segunda etapa foi necessário realizar a classificação das mensagens coletadas. Para tal, utilizamos o próprio *SpamAssassin*, que permitiu que fizéssemos uma comparação da classificação realizada pelo mesmo e a classificação realizada pelas árvores de detecção geradas.

Este capítulo encontra-se organizado da seguinte forma: na seção 6.2 são apresentadas as métricas utilizas tanto para avaliação dos filtros quanto para geração da árvore de decisão. Na seção 6.4 descrevemos a metodologia de avaliação utilizada. Na seção 6.3 é apresentado um breve estudo referente às cargas de trabalho. Os resultados obtidos estão descritos na seção 6.5.

6.2 Métricas de Avaliação

O objetivo deste trabalho é combinar várias propostas de firma a gerar um filtro composto no qual sejam balanceados a efetividade e o custo de processamento, além de buscar uma redução do custo do erro de classificação.

A efetividade neste contexto será a medida da taxa de acerto dos filtros avaliados, sendo esta calculada pela razão entre as mensagens classificadas corretamente e o número total de mensagens da base de dados.

O custo de processamento adotado refere-se ao tempo médio de processamento de cada mensagem. No filtro composto, este custo é a soma dos tempos gastos em cada componente.

O custo do erro de classificação tem a função de fornecer uma medida quantitativa de um problema subjetivo. O custo de mensagens maliciosas que são, erroneamente, armazenadas na caixa de entrada dos usuários pode ou não ser impactante no trabalho dos usuários. No entanto, normalmente, o maior problema está no erro de classificação de mensagens legítimas como maliciosas. Tal erro torna-se crítico em sistemas onde as mensagens maliciosas detectadas são removidas do sistema. Este custo pode ser definido como $C_{erro} = M_{L \to S} * \alpha + M_{S \to L} * \beta$, onde α e β são os custos de se classificar de forma incorreta, respectivamente, cada mensagem legítima e cada mensagem maliciosa. A medida precisa desses fatores é dependente basicamente do ponto de vista de cada usuário. Uma forma comumente empregada é a determinação da relação $\lambda = \alpha/\beta$. Essa taxa indica o quanto é mais caro classificar incorretamente uma mensagem legítima do que classificar incorretamente uma mensagem maliciosa [30, 13, 36, 64].

Neste trabalho estamos interessados em avaliar a eficácia das árvores de detecção geradas. Como o número de componentes é bastante limitado, a variação da taxa λ em uma ordem de grandeza já é o suficiente para avaliar o impacto do custo do erro de classificação na eficácia do filtro resultante. Portanto, adotaremos neste trabalho λ igual a 1, 2 e 10.

6.3 Carga de Trabalho

Conforme apresentado na seção 6.1, foram utilizadas duas cargas de trabalho distintas. A primeira, disponibilizada pelo projeto *SpamAssassin* [9], serve como base de avaliação da eficácia das regras implementadas no *SpamAssassin*. A segunda é uma carga proveniente do servidor central de correio eletrônico da Universidade Federal de Minas Gerais (UFMG). Em respeito à privacidade dos usuários do serviço de correio eletrônico dessa instituição, todas as mensagens foram processadas de forma anônima, onde nenhuma informação referente aos remetentes ou destinatários foram avaliadas ou

mesmo consultadas. Nesta seção fazemos uma descrição dessas duas cargas de trabalho, levando-se em conta o volume de dados e o número de mensagens.

A carga fornecida pelo *SpamAssassin* não é de um período bem definido, sendo mensagens dos anos de 2002, 2003 e 2005. São mensagens que passaram por um tratamento preliminar, no qual foram eliminadas partes nocivas, como códigos *JavaScript* maliciosos e também algumas informações de cabeçalhos. A carga da UFMG foi coletada no dia 27/02/2006 no turno da manhã, entre 08hs e 12hs. Um comparativo resumido dessas duas cargas pode ser observado na tabela 6.1.

Medidas	Carga do *SpamAssassin*	Carga da UFMG
Número de mensagens	7341	1644
Tamanho total das mensagens (bytes)	21305576	189228121
Média de bytes/mensagem (CV)	2902, 27(2, 88)	115102, 26(5, 28)
Tamanho da menor mensagem observada (bytes)	70	28
Tamanho da maior mensagem observada (bytes)	299584	12985360

Tabela 6.1: Comparativo resumido das cargas de trabalho utilizadas

Apesar da carga fornecida pelo projeto *SpamAssassin* ser em número de mensagens maior que a da UFMG em torno de 440%, essa segunda apresenta um volume de dados total muito superior. O gráfico da figura 6.1 apresenta a função de densidade de probabilidade do tamanho das mensagens nas duas cargas. Percebe-se que, na primeira carga, praticamente todas as mensagens possuem tamanho inferior a 100KBytes. Já para a carga da UFMG, em torno de 10% das mensagens possuem tamanho superior a 100KBytes.

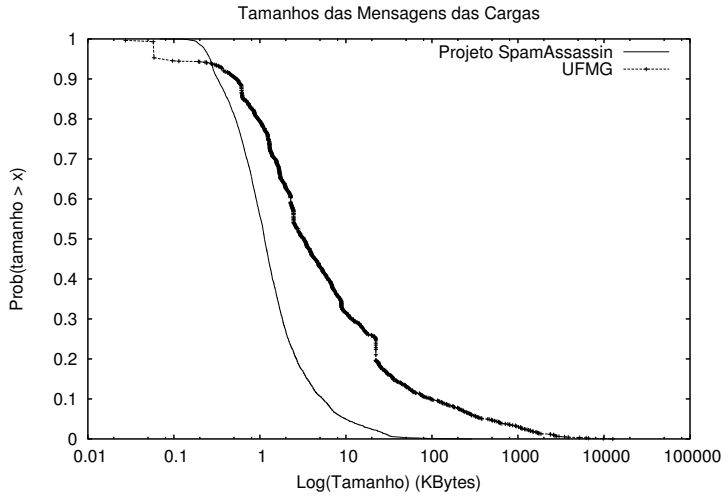

Figura 6.1: Função de densidade de probabilidade do volume de dados das cargas utilizadas

6.4 Metodologia

A geração da árvore de detecção, como já relatado, exige a realização de uma fase de treinamento. Para tal, dividimos a carga em 3 grupos com mesmo número de mensagens em cada um. A partir dessa divisão, utilizamos 2 dos grupos gerados como base de treinamento e 1 deles como base de testes. Realizamos esta avaliação para três cenários diferentes, de forma que em cada cenário fosse utilizado um dos grupos como base de testes.

Buscamos comparar, em cada cenário, como o filtro composto se comporta frente ao *SpamAssassin*, avaliando questões como a eficácia, o tempo de processamento das mensagens e o percentual de mensagens classificadas incorretamente. Levantamos também o número de mensagens classificadas antecipadamente no filtro composto e, dessas, qual foi a efetividade desse filtro.

Nessa primeira fase não estamos interessados em apresentar o impacto de cada critério na montagem da árvore de detecção. A função de utilidade $\Gamma(efetiv, custo_{proc}, C_{erro})$, definida no capítulo 4, faz o balanceamento entre a efetividade ($efetiv$), custo de processamento ($custo_{proc}$) e custo de erro de classificação (C_{error}) durante a montagem da árvore de detecção. Conforme já descrito naquele capítulo, para determinar o grau de importância de cada critério nesta função, foram criados, respectivamente, os fatores ϱ, κ e ρ. Os mesmos podem variar entre 0 e 1, onde 0 indica que o critério não deve gerar impacto algum na montagem da árvore e 1 indica que o critério deve ser considerado em sua totalidade durante a montagem da árvore. Como não queremos comparar as árvores com relação a cada critério neste primeiro momento, todos receberão o mesmo grau de importância e serão analisados em sua totalidade.

Nesse primeiro momento o custo de classificar incorretamente mensagens maliciosas será igual ao custo de classificar incorretamente mensagens legítimas.

O segundo momento já será utilizado para realização da avaliação do comportamento do filtro composto quando submetido a uma carga de mensagens reais. Conforme mencionado na seção 6.1, a carga fornecida pela UFMG não possui uma classificação prévia definida no servidor. Portanto, um primeiro passo é a classificação da mesma. Essa deverá ser realizada com a utilização do próprio *SpamAssassin* com as mesmas regras utilizadas na primeira avaliação. Essa classificação não permitirá obter a efetividade real do filtro composto para a carga avaliada, mas sim compreender o impacto dos critérios utilizados na montagem das árvores de detecção de mensagens maliciosas.

Buscamos explicar como devem ser os filtros resultantes, verificando suas diferenças, dada a variação dos fatores que determinam o grau de importância de cada critério utilizado na montagem da árvore.

Para validar os resultados, a segunda carga de trabalho também foi separada em

três partes de mesmo número de mensagens. A avaliação mais uma vez baseou-se em três cenários. Em cada cenário uma partição foi selecionada como base de testes e as outras duas foram utilizadas para o treinamento.

Nesse segundo momento fazemos o levantamento das árvores de detecção de cada cenário e avaliamos os resultados de cada árvore. Esses resultados referem-se aos totais de mensagens classificadas de acordo com a classificação realizada pelo *SpamAssassin*, os tempos totais de processamento, os erros de classificação e a efetividade na entrega antecipada de mensagens por cada um dos filtros compostos.

Nessa etapa, variamos os valores dos fatores de ajuste de importância da função de utilidade Γ com o objetivo de levantar exemplos de árvores de detecção geradas. Lembrando que a função de utilidade serve para selecionar, dado um conjunto de mensagens, o melhor componente para compor a árvore de detecção. A mesma faz um balanceamento dos valores da efetividade, custo de processamento e custo de erro de classificação de cada componente a ser selecionado. Na avaliação realizada nessa segunda etapa, também consideramos os valores da taxa λ referente ao custo do erro de classificação apresentada na seção 6.2.

O conjunto de valores distintos dos fatores de ajuste serão determinados para cada cenário, de forma a obter árvores de detecção que sejam verdadeiramente distintas. Para isto, iremos variar cada um desses fatores gerando um considerável número de combinações e determinar as árvores distintas geradas a partir dessas combinações.

6.5 Resultados

Os resultados foram obtidos a partir de experimentos realizados em uma máquina com CPU AMD Atlhon XP 64bits 3200+, 2GB de memória RAM e disco rígido de 160GB. As cargas foram disparadas por meio de uma rede ethernet de 100Mbits/s. O sistema operacional utilizado foi o Debian Linux 64 bits, Kernel versão 2.6.8-11. A versão do *SpamAssassin* utilizada foi a 3.0.4. O mesmo foi executado sobre o interpretador *Perl* versão 5.8.7. O gerador de carga foi implementado na linguagem *Perl* e foi utilizada a ferramenta *spamc* para envio das mensagens via TCP para o *SpamAssassin*. Esse último recebe as mensagens através de uma aplicação conhecida como *spamd*. A mesma é carregada em memória e aguarda as mensagens que serão processadas através de um porto TCP específico, normalmente o 783. Assim como o servidor *spamd*, a ferramenta *spamc* é disponibilizada pelo projeto *SpamAssassin*.

Conforme já relatado, a avaliação do modelo foi realizada em dois momentos distintos. Os resultados serão apresentados seguindo essa mesma divisão.

6.5.1 Filtro Composto X *SpamAssassin*

A primera fase do processo comparativo é a geração da árvore de detecção. Para isto precisamos gerar uma base de treinamento a partir da carga de trabalho utilizada nessa etapa. Conforme já descrito na seção 6.4, fizemos uma avaliação para três cenários distintos. Em todos esses cenários, foi gerada uma distribuição do número de mensagens como a apresentada na tabela 6.2. Nessa carga de trabalho, o número de mensagens legítimas representa mais de 80% do total. Essa característica irá favorecer a efetividade de detecção dos filtros menos rigorosos, que classificam a maioria das mensagens como legítimas. Seguindo a estratégia descrita no capítulo 4, esta é a fase da seleção das mensagens. A seleção dos componentes já foi apresentada no capítulo 5.

Medidas	**Base de Treinamento**	**Base de Testes**
Total de Mensagens	4894	2447
Total de Mensagens do tipo *Spam*	930	465
Total de Mensagens Legítimas	3964	1982

Tabela 6.2: Distribuição das mensagens para os cenários da primeira etapa

A base de treinamento ainda deve passar por um processo de transformação no qual devem ser identificadas as classificações dadas por cada componente a cada mensagem. A transformação fornece um conhecimento importante no que se refere à redundância na classificação das mensagens. Esta redundância é encontrada nas mensagens do tipo *spam* que são detectáveis por uma grande quantidade de filtros. São aquelas que podemos considerar como mensagens de fácil detecção. Este conhecimento vai favorecer a montagem de uma árvore de detecção onde esse tipo de mensagem possa ser detectado logo nos primeiros componentes do filtro composto montado a partir da árvore.

A sumarização das bases de treinamento geradas para cada cenário pode ser visualizada através da tabela 6.3. Nessa estão dispostas as médias dos totais observados nos três cenários de avaliação, assim como a média da efetividade de cada componente e dos tempos gastos por cada componente. Em sua maioria, as características apresentadas nessa tabela apresentam baixa variabilidade entre os cenários avaliados. No entanto, ocorre uma diferença considerável quando é considerado o componente *body*. A tabela 6.4 apresenta os valores totais, por cenário, referente ao componente *body*. A tabela demonstra que a carga de trabalho possui um conjunto de mensagens que o componente *body* não consegue classificar corretamente, o que causa o grande número

falsos positivos e negativos. No entanto, no 3ºcenário, esse conjunto é menor, o que favorece a efetividade nesse caso. Esta variabilidade não ocorre nos demais componentes porque os mesmos classificam a maioria das mensagens como legítimas.

Os estudos apresentados no capítulo 5 mostram que os componentes *header*, *rawbody* e *uri* possuem baixo número de regras positivas. Mesmo no caso dos dois últimos componentes que não possuem regras negativas, as suas regras positivas não chegam a gerar notas relativamente altas nas mensagens avaliadas, o que fez com que esses componentes classificassem grande parte das mensagens como legítimas. Essa classificação não afeta muito a efetividade final de cada componente uma vez que o número de falsos positivos apresentados por esses é muito baixo e as mensagens legítimas representam mais de 80% do total.

Médias Observadas nos Três Cenários	**Atributos**			
	body	***header***	***rawbody***	***uri***
Mensagens classificadas como legítimas (CV)	3999, 33(0, 00)	4889, 33(0, 00)	4887, 33(0, 00)	4488, 67(0, 00)
Mensagens classificadas como *spam* (CV)	894, 67(0, 01)	4, 67(0, 25)	6, 67(0, 31)	405, 33(0, 02)
Legítimas classificadas corretamente (CV)	3747, 33(0, 01)	3959, 33(0, 00)	3959, 33(0, 00)	3914.67(0, 00)
Spams classificados corretamente (CV)	678, 00(0, 05)	0, 00(0, 00)	2, 00(0, 50)	356, 00(0, 05)
Número de falsos positivos (CV)	216, 67(0, 18)	4, 67(0, 25)	4.67(0, 25)	49, 33(0, 18)
Número de falsos negativos (CV)	252, 00(0, 13)	930, 00(0, 00)	928, 00(0, 00)	574, 00(0, 03)
Efetividade (CV)	0, 90(0, 02)	0, 81(0, 00)	0, 81(0, 00)	0, 87(0, 01)
Tempo médio de processamento (s)(CV)	0, 46(0, 32)	0, 26(0, 22)	0, 17(0, 15)	0, 17(0, 12)

Tabela 6.3: Sumário da base de treinamento nos três cenários

Médias Observadas nos Três Cenários	**Componente *body***		
	1ºCenário	**2ºCenário**	**3ºCenário**
Mensagens classificadas como legítimas	3997	3995	4006
Mensagens classificadas como *spam*	897	899	888
Legítimas classificadas corretamente	3724	3725	3793
Spams classificados corretamente	657	660	717
Número de falsos positivos	240	239	171
Número de falsos negativos	273	270	213
Efetividade	0, 90	0, 90	0, 92
Tempo médio de processamento (s)	0, 63	0, 40	0, 35

Tabela 6.4: Componente *body*: resultados por cenário

Através da tabela 6.3 é possível conhecer, para esse conjunto de dados, as particularidades de cada componente utilizado na montagem do filtro composto. O componente que apresentou a melhor efetividade foi o que possui um maior número de regras implementadas. Porém, tal efetividade não está ligada diretamente ao número de regras.

Outros fatores estão associados, como a qualidade dessas regras e a característica da carga de trabalho. No caso desta avaliação, o que contribuiu para que o componente *uri* apresentasse uma boa efetividade foi a característica da carga de possuir em seu conjunto mais de 80% de mensagens legítimas. Este quadro seria completamente diferente se estivéssemos trabalhando com uma carga com alta taxa de mensagens maliciosas. Nessa, a efetividade do componente *uri* seria muito baixa uma vez que este classificaria uma grande quantidade das mensagens da carga de trabalho de forma incorreta.

Outro fato importante é com relação aos totais de falso positivo e negativo. O *body* é o único componente que apresentou um número significativo de falsos positivos. Isto ocorre porque sua base de regras de avaliação é mais rigorosa do que a dos demais componentes. Os valores referentes às regras estão sumarizados na tabela 5.3 do capítulo 5. Este componente implementa mais de 80% das regras totais do *SpamAssassin* que estamos utilizando. No entanto, quando separamos em regras positivas e negativas, isto não é refletido nesse segundo tipo de regra. O *body* possui menos de 50% do total das regras negativas implementadas no *SpamAssassin*. Essas regras são aquelas que buscam padrões de mensagens legítimas e, por isto, tendem a reduzir as notas finais de cada mensagem. Como consequência da baixa proporção entre regras negativas e positivas no componente *body*, esse classifica um menor número de mensagens como legítimas. Apesar das variabilidades encontradas para os resultados do componente *body* em cada cenário, podemos considerar que existe uma grande similaridade entre as bases de treinamento geradas.

Tendo as bases de treinamento montadas, o próximo passo é a geração da árvore de detecção. Para isto, aplicamos o algoritmo definido no capítulo 4 com os fatores de ajuste ϱ, κ e ρ e a taxa de erro λ com o valor 1.

Dada a similaridade das bases de treinamento de cada cenário e o limitado número de componentes utilizados, ao criar as árvores de detecção para os três cenários, foi observado que eram árvores idênticas. Esta equivalência entre as árvores de cada cenário ocorre porque, neste contexto, os componentes apresentaram um comportamento similar nas três situações. A árvore de detecção resultante para os três cenários pode ser visualizada na figura 6.2. Essa é uma árvore de detecção disjuntiva, na qual basta apenas um dos componentes classificar a mensagem como *spam* para ela ser colocada na caixa de mensagens maliciosas.

Uma vez obtida a árvore de detecção, cabe agora realizar a comparação entre o filtro composto e o *SpamAssassin*. O resultado pode ser observado através da tabela 6.5. De forma a avaliar a efetividade dos dois filtros nos três cenários, na tabela estão apresentadas as médias dos resultados obtidos em cada cenário.

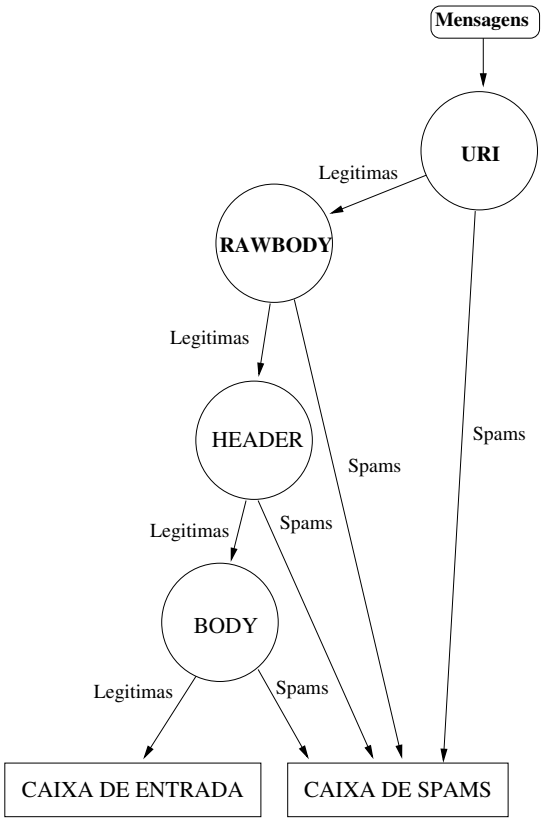

Figura 6.2: Árvore de detecção gerada para a carga de trabalho do projeto *SpamAssassin*

Médias Observadas Observadas nos Três Cenários	***Filtro Gerado***	***SpamAssassin***
Mensagens classificadas como *spams* (CV)	646, 67(0, 06)	298.33(0.08)
Mensagens classificadas como legítimas (CV)	1800, 33(0, 02)	2148.67(0.01)
Legítimas classificadas corretamente (CV)	1755, 67(0, 03)	1965.67(0.01)
Spams classificados corretamente (CV)	420, 33(0, 04)	282(0.11)
falsos positivos (CV)	226.33(0.23)	16.33(0.67)
falsos negativos (CV)	44.67(0.39)	183(0.17)
efetividade (CV)	0.89(0.03)	0.92(0.02)
Tempo médio processamento(s) (CV)	0.82(0.06)	0.92(0.06)
Tempo total processamento(s) (CV)	1994.11(0.06)	2260.83(0.06)

Tabela 6.5: Comparativo: Filtro Gerado X *SpamAssassin* (Média entre os três cenários)

Percebe-se nesses resultados, uma ligeira vantagem do filtro *SpamAssassin* sobre o filtro composto gerado no que se refere à efetividade. Esta diferença encontra-se por volta de 3%. Pela variabilidade existente entre os três experimentos, percebe-se que esta diferença pode ser considerada pequena uma vez que o desvio padrão no filtro composto para a efetividade é de $0,03$ e no *SpamAssassin* é de $0,02$. Sendo assim, podemos considerar que os filtros apresentam efetividades próximas.

A efetividade do filtro composto foi prejudicada pela ocorrência de um grande número de falsos positivos. Percebe-se aqui a forte influência do componente *body* no resultado. A explicação é o fato de estarmos trabalhando com uma árvore de detecção

disjuntiva. Os demais componentes (*header*,*rawbody*,*uri*) classificam, em média, mais de 90% de todas as mensagens como sendo legítimas. Nessa árvore, uma mensagem só é classificada como legítima se for processada por todos os componentes. Essa característica fez com que um percentual considerável das mensagens fossem processadas pelo componente *body*, que apresenta um alto índice de ocorrência de falsos positivos.

O gráfico apresentado na figura 6.3 apresenta o percentual de mensagens entregues por cada um dos componentes no filtro composto. Através deste é possível perceber que apenas 20% das mensagens são detectadas antes de serem processadas pelo último filtro, o *body*.

Um segundo fator comparativo dos componentes é a precisão dos mesmos para a detecção de mensagens *spams*. Esta precisão é dada pela equação 6.1, onde $M_{S \to S}$ é o total de mensagens *spams* classificadas corretamente e $M_{L \to S}$ é o total de mensagens legítimas classificadas como *spams*. O gráfico da figura 6.4 apresenta a precisão obtida em cada um dos componentes. Através desse resultado é possível notar que existe um alto índice de acerto logo nos dois primeiros componentes do filtro composto. De todas as mensagens detectadas como *spam*, aproximadamente 90% delas são classificadas corretamente. Esta detecção antecipada evita que mensagens maliciosas sejam avaliadas de forma desnecessária em todas as etapas de um filtro.

É possível perceber através desses dois gráficos apresentados que o componente *rawbody* não contribui, em questão de volume, de forma significativa para a detecção de mensagens maliciosas, no entanto apresenta a mais alta precisão na detecção de *spams*, indicando que, quando este componente classifica uma mensagem como maliciosa, a probabilidade de acerto é alta.

$$prec = M_{S \to S} / M_{S \to S} + M_{L \to S} \tag{6.1}$$

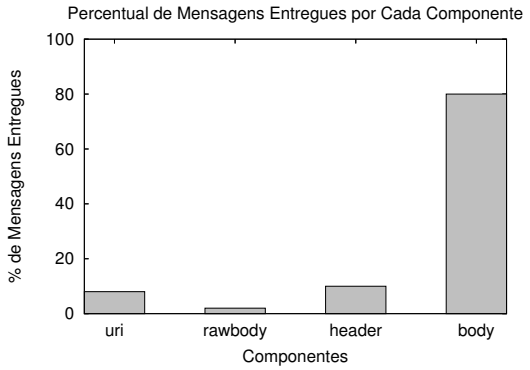

Figura 6.3: Percentual de mensagens processadas por componente

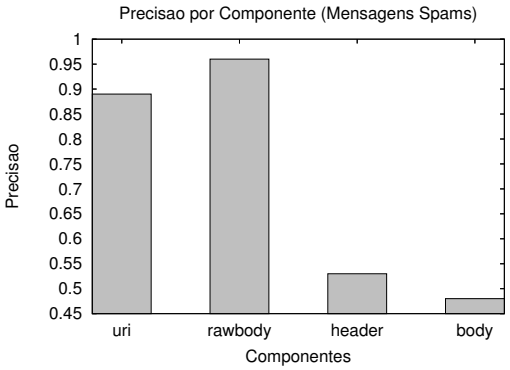

Figura 6.4: Precisão na detecção de *spams* em cada componente

Comparado com a efetividade, o tempo é o critério de menor relevância na classificação de mensagens. No entanto, fazendo uma rápida avaliação do mesmo é possível perceber que o filtro composto apresentou melhores resultados. As médias dos tempos observados encontram-se apresentadas na tabela 6.5. O melhor desempenho do filtro composto ocorre porque existe um considerável número de mensagens que não são processadas por todos os componentes, sendo que essas são processadas exclusivamente por aqueles que, conforme apresentado na tabela 6.3, apresentam baixo custo de processamento. Os gráficos dispostos nas figuras 6.5 e 6.6 permitem comprovar que o desempenho do filtro composto é realmente melhor. Através desses gráficos é possível perceber que a diferença é obtida com as mensagens processadas entre 0 e 0.5 segundos. Enquanto que no filtro composto temos aproximadamente 40% das mensagens sendo processadas nesse intervalo, no *SpamAssassin* este percentual é pouco maior do que 25%.

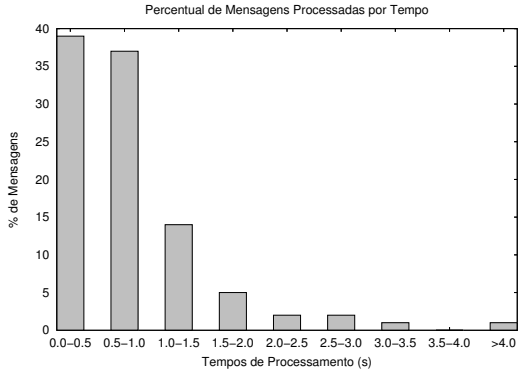

Figura 6.5: Filtro Composto

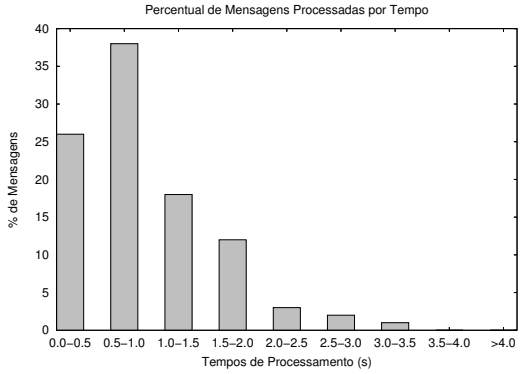

Figura 6.6: *SpamAssassin*

Figura 6.7: Percentual de Mensagens Processadas dentro do Intervalo Específico

O desempenho superior do filtro composto sobre o *SpamAssassin* encontra-se na antecipação da detecção de mensagens maliciosas. Para comprovar isto, avaliamos, para cada cenário, o tempo médio de processamento das mensagens entregues no primeiro componente (*uri*) do filtro composto e comparamos com o tempo médio de processamento dessas mesmas mensagens no filtro *SpamAssassin*. Repetimos essa comparação para o segundo (*rawbody*), terceiro (*header*) e quarto (*body*) componente. Os resultados com as médias dos valores observados em cada cenário estão disponibilizados na tabela 6.6. As mensagens entregues no primeiro componente são processadas em um tempo aproximadamente 5 vezes menor do que se as mesmas estivessem sendo processadas no *SpamAssassin*. O mesmo ocorre para as mensagens entregues pelo componente *rawbody*. Este ganho de desempenho torna-se menor quando as mensagens são entregues pelo *header* e chega a se inverter quando elas são processadas por todos os

filtros. Tal inversão ocorre pela existência de custos adicionais de envio e recepção de mensagens entre os componentes.

Componentes	Filtro Composto	SpamAssassin
uri (CV)	$0,32(0,00)$	$1,62(0,00)$
rawbody (CV)	$0,29(0,59)$	$1,26(0,29)$
header (CV)	$0,59(0,09)$	$0,96(0,07)$
body (CV)	$0,90(0,06)$	$0,84(0,09)$

Tabela 6.6: Tempo médio de processamento das mensagens entregues em cada componente X Tempo médio de processamento dessas no *SpamAssassin* (segundos)

Quando comparados por meio do custo do erro de classificação, percebe-se que, independente dos fatores α e β, o *SpamAssassin* apresenta um diferencial que é refletido na efetividade. Isto é um ponto negativo desse filtro composto, uma vez que o impacto para o usuário quando é perdida alguma mensagem legítima por erro de classificação pode causar grandes transtornos.

6.5.2 Variações do Filtro Composto

Nessa segunda fase avaliamos o impacto no filtro composto quando são variados os critérios utilizados na montagem da árvore de detecção. Para realizar esta tarefa, seguiremos a metodologia apresentada na seção 6.4 e, para a montagem da árvore, adotaremos a estratégia definida no capítulo 4, assim como foi realizado na etapa anterior.

A classificação realizada pelo *SpamAssassin* resultou em uma carga de trabalho com um total de 1401 mensagens consideradas legítimas e um total de 243 mensagens consideradas maliciosas. Nessa carga de trabalho classificada também ocorreu um predomínio do número de mensagens legítimas, representando mais de 85% do total.

Nesta etapa também trabalhamos com a visão de três cenários distintos. Para a geração desses cenários dividimos a carga de trabalho em três grupos com mesmo número de mensagens. Cada cenário é composto por uma base de treinamento formada por dois desses grupos e uma base de testes com o grupo restante. Em cada um dos cenários um dos grupos foi utilizado como base de testes. Uma apresentação da distribuição dos totais de mensagens nos cenários pode ser visualizada na tabela 6.7.

Medidas	Base de Treinamento	Base de Testes
Total de Mensagens	1096	548
Total de Mensagens do tipo *Spam*	162	81
Total de Mensagens Legítimas	934	467

Tabela 6.7: Distribuição das mensagens nos cenários gerados para a carga de trabalho da UFMG

Assim como foi apresentado na primeira etapa de avaliação, a base de treinamento também sofreu um processo de transformação para a geração de um conhecimento referente ao comportamento de cada um dos componentes sobre essa carga de trabalho. Os resultados encontram-se sumarizados na tabela 6.8. Os valores apresentados representam as médias observadas em cada um dos cenários de avaliação. Uma diferença clara dessa tabela para a apresentada na seção anterior é que não referenciamos aqui a questão de classificação correta ou não, nem mesmo aos falsos negativos e positivos. Essas medidas não fazem sentido uma vez que não estamos trabalhando com uma classificação efetiva e sim relativa à classificação realizada pelo *SpamAssassin*, que não apresenta 100% de efetividade.

A métrica de avaliação nesse contexto é a efetividade relativa, que mede o quanto o filtro composto conseguiu se aproximar do *SpamAssassin*. Seu cálculo pode ser realizado através da equação 6.2, onde: $M_{L_{SA} \rightarrow L_{FC}}$ é o número de mensagens classificadas como legítimas pelo *SpamAssassin* e pelo filtro composto; $M_{S_{SA} \rightarrow S_{FC}}$ é o número de mensagens classificadas como *spams* pelo *SpamAssassin* e pelo filtro composto e M é o número total de mensagens da base.

$$efetiv_{relativa} = \frac{M_{L_{SA} \rightarrow L_{FC}} + M_{S_{SA} \rightarrow S_{FC}}}{M} \tag{6.2}$$

Médias Observadas nos Três Cenários	Atributos			
	body	***header***	***rawbody***	***uri***
Mensagens classificadas como legítimas(CV)	593,33(0,04)	1094,00(0,00)	1096,00(0,00)	1024,67(0,02)
Mensagens classificadas como *spam*(CV)	502,67(0,04)	2,00(0,50)	0,00(0,00)	71,33(0,25)
Mensagens classificadas como legítimas pelo *SpamAssassin* e pelo componente (CV)	591,33(0,04)	932,00(0,00)	934,00(0,00)	888,67(0,02)
Mensagens classificadas como *spams* pelo *SpamAssassin* e pelo componente (CV)	160,00(0,01)	0,00(0,00)	0,00(0,00)	26,00(0,42)
Mensagens classificadas como legítimas pelo *SpamAssassin* e como *spams* pelo componente (CV)	342,67(0,07)	2,00(0,50)	0,00(0,00)	45,33(0,35)
Mensagens classificadas como *spams* pelo *SpamAssassin* e como legítimas pelo componente (CV)	2,00(0,50)	162,00(0,00)	162,00(0,00)	136,00(0,08)
Efetividade Relativa (CV)	0,69(0,03)	0,85(0,00)	0,85(0,00)	0,83(0,02)
Tempo médio de processamento (s) (CV)	1,07(0,19)	0,27(0,06)	0,20(0,06)	0,18(0,10)

Tabela 6.8: Sumário da base de treinamento nos três cenários (carga de trabalho da UFMG)

Os resultados gerados pelo processamento das bases de treinamento nos três cenários pelos componentes encontram-se apresentados na tabela 6.8.

Observando os dados, nota-se que o componente *body* foi o que apresentou pior efetividade relativa. Assim como ocorreu na primeira etapa, a causa direta está associada às regras que compõem esse componente. A proporção do número de regras negativas para o de regras positivas implementadas no *body* é baixa. Sendo assim, existe uma forte tendência à geração de notas maiores, quando comparado ao *SpamAssassin* para cada mensagem avaliada. Essas notas resultantes tornam o componente mais rigoroso e, consequentemente, é gerado um maior número de falsos positivos.

Conforme já relatado na seção 6.5.1, os demais componentes possuem poucas regras positivas. O *header* é composto por aproximadamente 16% das regras implementadas no *SpamAssassin* e os demais componentes juntos totalizam pouco mais de 2% das mesmas. Seus resultados estão associados ao baixo número de regras positivas que os compõem. Dessa forma, as notas finais assinaladas a cada mensagem acabam sendo baixas, o que faz com que esses componentes classificassem quase todas as mensagens como legítimas. A efetividade relativa é alta porque as mensagens legítimas representam mais de 85% da base.

Após a montagem da base de treinamento, o próximo passo é a geração das árvores de detecção. Para isto variamos os fatores de ajuste ϱ, κ e ρ referentes, respectivamente, aos critérios efetividade, custo de processamento e custo de erro de classificação.

Além desses fatores, também foi variado o valor da taxa λ, que mede a relação entre o custo de classificar incorretamente uma mensagem legítima e o custo de classificar incorretamente uma mensagem maliciosa. Os valores utilizados para essa taxa foram 1, 2 e 10. A escolha baseou-se no objetivo de comparar as seguintes situações: custo de classificação incorreta de mensagens legítimas igual ao custo de classificação incorreta de mensagens maliciosas; custo do primeiro tipo de erro duas vezes maior do que o segundo; e custo do primeiro tipo de erro 10 vezes maior do que o segundo.

Cabe ressaltar que quando os fatores ϱ, κ e ρ forem assinalados com o valor 0 é para indicar que o critério associado não tem influência alguma na geração da árvore. Em contrapartida, quando forem assinalados com o valor 1 é para indicar que o critério influencia em sua totalidade na geração da árvore.

Cada fator foi variado de 0 a 1, com intervalos de $0,1$. Desta forma geramos 1331 combinações diferentes. Associados aos valores assinalados para λ, obtivemos uma combinação total de 3993 valores distintos.

O algoritmo apresentado no capítulo 4 foi aplicado a cada uma das bases de treinamento de cada cenário. Como entrada desse foram utilizadas todas as combinações de valores de ϱ, κ e ρ. Ao final, foi gerado um total de 6 árvores de detecção distintas.

As árvores podem ser visualizadas através das figuras 6.8, 6.9, 6.10, 6.11, 6.12 e

6.13. É fácil observar que nesse conjunto as árvores de detecção Nº2, Nº3 e Nº5 são do tipo disjuntiva, ou seja, são árvores nas quais as mensagens são detectadas como maliciosas sempre que qualquer um dos componentes as classificarem dessa forma.

Nas demais árvores ocorre uma inversão no componente *body*. A causa da inversão encontra-se no mecanismo de poda da árvore e de seleção de componentes implementados e descritos no capítulo 4.

Sobre o algoritmo de poda, definido pela função $RealizaPoda$, somente é montada uma sub-árvore à partir da classificação realizada por um determinado componente se a taxa de erro desse estiver acima de um determinado limiar ϵ. Neste trabalho definiu-se empiricamente um ϵ ideal igual a 30%. Para valores acima, eram realizadas podas e, consequentemente, geradas árvores pouco efetivas, com um ou dois componentes apenas. Para valores abaixo a poda praticamente não era realizada, gerando árvores maiores. Nessas árvores a mensagem, independente do caminho que percorresse, seria processada por praticamente todos os componentes. Dessa forma o principal ganho da árvore de detecção, que é a antecipação da detecção da mensagem, seria perdido.

Com relação à seleção de componentes, para determinadas combinações dos fatores ϱ, κ e ρ o componente *body* apresentou uma utilidade $\Gamma(efetiv, custo_{proc}, C_{erro})$ maior. No conjunto das mensagens classificadas como legítimas por esse componente, a taxa de erro é inferior ao limiar definido. Sendo assim, o algoritmo realiza a poda sempre que o componente *body* é selecionado, fazendo com que nas árvores geradas as mensagens classificadas como legítimas por esse componente sejam entregues, ao contrário do que ocorre com os demais componentes.

O próximo passo é a montagem dos filtros compostos a partir das árvores geradas. Os resultados comparativos dos filtros podem ser encontrados na tabela 6.9. Os valores apresentados representam as médias observadas em cada cenário. Nossa primeira avaliação refere-se à efetividade relativa.

Os filtros compostos com classificação disjuntiva foram os que apresentaram pior desempenho. Já o filtro Nº1, onde o componente *body* é o primeiro a processar as mensagens, foi o que apresentou melhores resultados com relação à efetividade relativa.

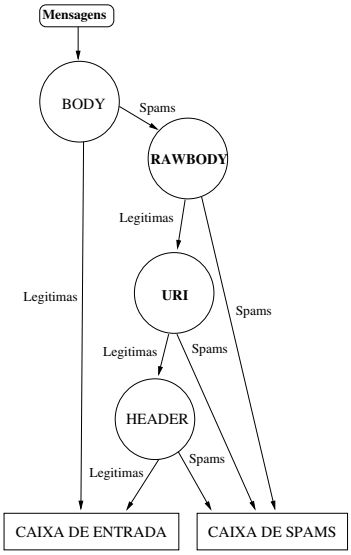

Figura 6.8: Árvore de detecção Nº1

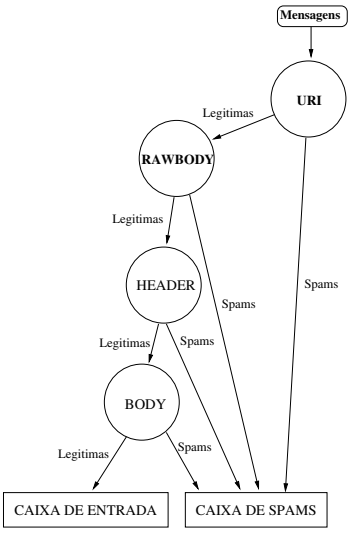

Figura 6.9: Árvore de detecção Nº2

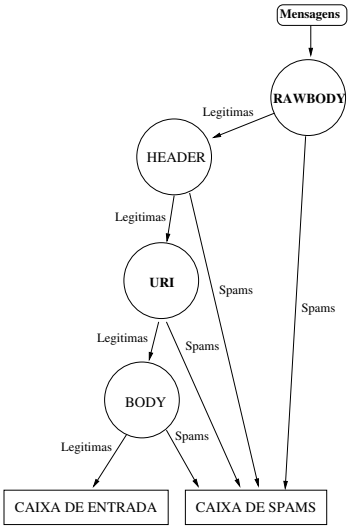

Figura 6.10: Árvore de detecção Nº3

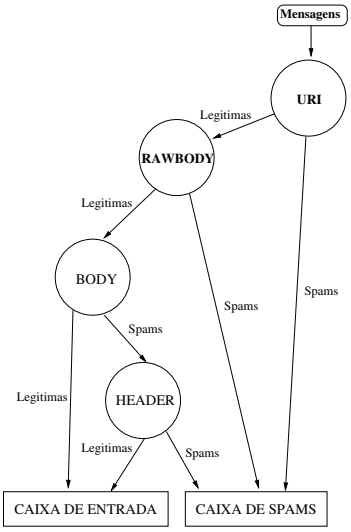

Figura 6.11: Árvore de detecção Nº4

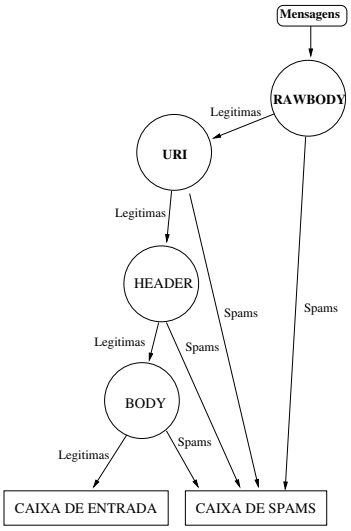

Figura 6.12: Árvore de detecção Nº5

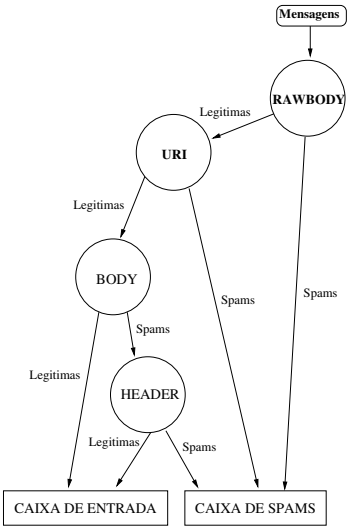

Figura 6.13: Árvore de detecção Nº6

Médias Observadas nos Três Cenários	**Filtros Compostos**					
	#1	**#2**	**#3**	**#4**	**#5**	**#6**
Mensagens classificadas como legítimas(CV)	527, 67(0, 02)	238, 33(0, 06)	238, 33(0, 06)	508, 33(0, 02)	238, 33(0, 06)	508, 33(0, 02)
Mensagens classificadas como *spam*(CV)	20, 33(0, 59)	309, 67(0, 05)	309, 67(0, 05)	39, 67(0, 30)	309, 67(0, 05)	39, 67(0, 30)
Mensagens classificadas como legítimas pelo *SpamAssassin* e pelo componente (CV)	459, 00(0, 00)	238, 00(0, 06)	238, 00(0, 06)	440, 33(0, 03)	238, 00(0, 06)	440, 33(0, 03)
Mensagens classificadas como *spams* pelo *SpamAssassin* e pelo filtro composto (CV)	12, 33(0, 95)	80, 67(0, 01)	80, 67(0, 01)	13, 00(0, 83)	80, 67(0, 01)	13, 00(0, 83)
Mensagens classificadas como legítimas pelo *SpamAssassin* e como *spams* pelo filtro composto (CV)	8, 00(0, 13)	229, 00(0, 06)	229, 00(0, 06)	26, 67(0, 48)	229, 00(0, 06)	26, 67(0, 48)
Mensagens classificadas como *spams* pelo *SpamAssassin* e como legítimas pelo filtro composto (CV)	68, 67(0, 17)	0, 33(1, 73)	0, 33(1, 73)	68, 00(0, 16)	0, 33(1, 73)	68, 00(0, 16)
Custo de erro de classificação $\lambda = 1$ (CV)	76, 67(0, 15)	229, 33(0, 06)	229, 33(0, 06)	94, 67(0, 22)	229, 33(0, 06)	94, 67(0, 22)
Custo de erro de classificação $\lambda = 2$ (CV)	84, 67(0, 13)	458, 33(0, 06)	458, 33(0, 06)	121, 33(0, 27)	458, 33(0, 06)	121, 33(0, 27)
Custo de erro de classificação $\lambda = 10$ (CV)	148, 67(0, 08)	2290, 33(0, 06)	2290, 33(0, 06)	334, 67(0, 40)	2290, 33(0, 06)	334, 67(0, 40)
Efetividade Relativa (CV)	0, 86(0, 02)	0, 58(0, 04)	0, 58(0, 04)	0, 83(0, 05)	0, 58(0, 04)	0, 83(0, 05)
Tempo médio de processamento (s) (CV)	1, 29(0, 02)	1, 66(0, 02)	2, 06(0, 01)	1, 90(0, 03)	1, 63(0, 21)	1, 92(0, 09)
Tempo total de processamento (s) (CV)	705, 86(0, 02)	910, 36(0, 03)	1130, 72(0, 01)	1044, 12(0, 03)	893, 68(0, 21)	1050, 80(0, 09)

Tabela 6.9: Comparativo dos Filtros Compostos gerados para a carga da UFMG

Componentes	**Filtros Compostos**					
	#1	**#2**	**#3**	**#4**	**#5**	**#6**
body	$53,95\%(0,08)$	$85,83\%(0,03)$	$84,00\%(0,02)$	$50,43\%(0,06)$	$85,83\%(0,03)$	$50,43\%(0,06)$
header	$42,46\%(0,13)$	$7,06\%(0,07)$	$7,36\%(0,03)$	$42,46\%(0,13)$	$7,06\%(0,07)$	$42,46\%(0,13)$
rawbody	$0,00\%(0,00)$	$0,00\%(0,00)$	$0,00\%(0,00)$	$0,00\%(0,00)$	$0,00\%(0,00)$	$0,00\%(0,00)$
uri	$3,59\%(0,61)$	$7,12\%(0,32)$	$8,64\%(0,13)$	$7,12\%(0,32)$	$7,12\%(0,32)$	$7,12\%(0,32)$

Tabela 6.10: Percentual de mensagens entregues por componente nos filtros compostos

Componentes	Filtros Compostos					
	#1	#2	#3	#4	#5	#6
body	1,00(0,00)	0,65(0,06)	0,65(0,06)	1,00(0,00)	0,65(0,06)	1,00(0,00)
header	0,71(0,06)	0,00(0,00)	0,00(0,00)	0,71(0,06)	0,00(0,00)	0,71(0,06)
rawbody	0,00(0,00)	0,00(0,00)	0,00(0,00)	0,00(0,00)	0,00(0,00)	0,00(0,00)
uri	0,52(0,51)	0,32(0,80)	0,32(0,80)	0,32(0,80)	0,32(0,80)	0,32(0,80)

Tabela 6.11: Efetividade de cada componente na entrega das mensagens

A tabela 6.10 apresenta o percentual de mensagens entregues por cada componente nos filtros compostos criados. Uma primeira constatação é que, para as bases de testes utilizadas, o componente *rawbody* não contribuiu em nada para a efetividade, uma vez que o mesmo não realiza nenhuma detecção de mensagens maliciosas. Ele ainda acrescenta um custo de processamento desnecessário ao filtro composto gerado uma vez que, mesmo atuando simplesmente como uma ponte entre dois componentes, ele realiza processamento sobre as mensagens trafegadas.

A tabela 6.10 apresenta o percentual de mensagens detectadas em cada componente do filtro composto. Na tabela 6.11 estão disponíveis os valores da efetividade relativa dos componentes sobre as mensagens entregues. Esses valores medem a razão entre as mensagens detectadas corretamente sobre o total de mensagens detectadas em cada um dos componentes.

Comparando as tabelas 6.10 e 6.11 é possível perceber que o ganho do filtro composto baseado na árvore Nº1 vem exatamente do fato do componente *body* ser a raiz da árvore. Esse realiza a entrega de mais de 50% das mensagens da base de testes. Isto significa que foram classificadas como mensagens legítimas mais de 50% do total. Nessas, o componente *body* apresentou uma precisão próxima dos 100%. Isto implica que, logo no primeiro componente, mais de 50% das mensagens são entregues e classificadas de acordo com a classificação gerada pelo *SpamAssassin*.

Um segundo fator que contribui para o aumento da efetividade relativa no primeiro filtro composto é que mais de 40% das mensagens foram entregues pelo segundo componente, o *header*. Dessas, aproximadamente 71% receberam a mesma classificação gerada pelo *SpamAssassin*. Os filtros compostos gerados à partir das árvores Nº4 e Nº6 tem uma efetividade relativa mais baixa do que o primeiro filtro composto porque nesses o componente *uri* realiza um processamento anterior aos componentes *body* e *header*. Nesses dois filtros, o *uri* realiza uma entrega antecipada de aproximadamente 7% das mensagens. No entanto, dessas, somente 32% são classficadas de acordo com a classificação do *SpamAssassin*.

Um fato e que reflete a eficácia do primeiro filtro composto é que o mesmo apresenta o melhor desempenho com relação ao tempo de processamento das mensagens. Este

filtro chega a ser aproximadamente 38% mais eficiente em termos de custo de processamento das mensagens do que os outros filtros compostos. Esse ganho encontra-se no fato das mensagens não serem processadas por todos os componentes. Nos filtros compostos 2, 3 e 5 o último componente, o *body*, processa aproximadamente 85% das mensagens. Isto implica que esses mesmos 85% foram processadas por todos os outros 3 componentes do filtro, o que faz com que ocorra uma soma dos tempos de processamento de cada um dos componentes.

Com relação ao custo do erro de classificação, mais uma vez temos um ganho do primeiro filtro composto. Na tabela 6.9 estão dispostos os custos de erro de classificação associados a cada filtro composto. Nesta avaliação, observamos os custos para λ igual a 1, 2 e 10. Lembrando que essa taxa refere-se à razão entre o custo do erro de classificação de mensagens legítimas e o custo do erro de classificação de mensagens maliciosas, e que α e β refletem esses custos por mensagem. Sendo $\lambda = \alpha/\beta$, podemos adotar β sempre igual a 1 e variamos o valor de α.

6.6 Limitações da Abordagem

Mesmo apresentando ganhos com relação ao tempo de processamento, o filtro composto não apresentou-se satisfatório com relação à efetividade quando comparado ao *SpamAssassin*.

A principal causa do mau desempenho refere-se diretamente à quebra do *SpamAssassin*. No filtro completo, as regras são definidas para coexistirem, trabalhando de forma conjunta e gerando uma nota final que reflita a classificação esperada para a mensagem avaliada.

Cada regra busca uma característica específica na mensagem. Somente quando um conjunto de características são encontradas, uma mensagem pode ser considerada legítima ou maliciosa. Quando as regras são separadas pelo tipo, o conjunto também é quebrado fazendo com que a informação agregada seja perdida. Essa perda ocorre porque a avaliação realizada em cada nodo da árvore desconsidera a avaliação anterior. Nossa abordagem busca corrigir os erros gerados por um determinado componente apenas realizando uma nova avaliação por um ou mais componentes.

Uma possível solução seria repassar a nota gerada em cada nodo da árvore binária de forma a manter o histórico das classificações e a partir desse realizar a classificação final.

Além dos problemas apontados, a estratégia de divisão por tipo de regra não foi a mais acertada uma vez que existem regras que são utilizadas por outras regras que não necessariamente são do mesmo tipo. Essa dependência direta também não foi

contemplada.

6.7 Considerações

Os resultados apresentados neste capítulo demonstram que o filtro composto gerado a partir da quebra do *SpamAssassin* não apresentou ganhos satisfatórios, principalmente com relação à efetividade. No entanto, se considerarmos as limitações e falhas da abordagem, podemos notar que os resultados encontrados podem ser considerados bons.

Para a primeira carga foi montado um filtro composto a partir de uma árvore de detecção disjuntiva. Para geração desta árvore, buscou-se utilizar os três critérios de montagem com o mesmo peso. Essa decisão interfere diretamente no desempenho do filtro composto resultante tanto com relação à efetividade, quanto com relação aos custos. Com a mesma, estamos buscando, através de uma estratégia gulosa, um filtro que atenda isoladamente a qualquer desses três critérios. A título de exemplo, dentre os componentes utilizados na montagem da árvore, o que apresentou melhor efetividade foi o que apresentou pior custo de processamento e o pior custo com relação ao erro de classificação. Isto fez com que ele fosse o último componente da árvore de detecção a processar as mensagens.

Na primeira avaliação, o ganho encontra-se relacionado ao tempo de processamento das mensagens. Esse ganho está associado diretamente ao fato do mesmo realizar uma detecção antecipada de aproximadamente 20% das mensagens maliciosas. O filtro composto consome um tempo menor do sistema para realização da detecção dessas mensagens. Outro ganho é relacionado à precisão sobre a classificação antecipada. Das mensagens entregues já no primeiro componente, mais de 90% são classificadas corretamente. Esta classificação antecipada faz com que mensagens que receberam uma mesma classificação em todas as etapas no *SpamAssassin* possam ser efetivamente entregues logo nos primeiros componentes.

Nesse contexto trabalhamos com um universo reduzido de componentes do filtro composto, o que interferiu diretamente na avaliação da influência gerada por cada critério de montagem da árvore de detecção, bem como a influência da taxa de erros de classificação. Através de uma considerável combinação de valores dos fatores que determinam a importância de cada critério utilizado na montagem da árvore, foi possível observar a existência de um conjunto limitado de árvores. A baixa variabilidade das árvores é causada pelo pequeno número de componentes envolvidos.

No segundo momento avaliamos uma carga classificada pelo próprio *SpamAssassin*, portanto, não trabalhamos com os conceitos de efetividade ou de falsos positivos e

falsos negativos. Utilizamos o conceito de efetividade relativa, que mede a taxa de mensagens classificadas pelo *SpamAssassin* que receberam a mesma classificação no filtro composto gerado.

O filtro composto no qual o componente *body* avaliava primeiramente as mensagens foi o que aproximou-se mais da classificação realizada pelo *SpamAssassin*. Nessa segunda avaliação, os filtros compostos baseados em árvores disjuntivas foram os que apresentaram pior desempenho. Esse resultado reflete a avaliação realizada por cada componente sobre a base de treinamento. Os mesmos apresentaram baixa taxa de detecção de mensagens maliciosas. Portanto, praticamente todas as mensagens que os mesmos processaram foram repassadas para o filtro seguinte até serem processadas pelo último filtro. Em todos os casos de árvores de detecção disjuntiva, o componente *body* estava na última posição. Nas árvores onde este componente estava entre os intermediários, o desempenho também foi melhor do que nas árvores disjuntivas.

O componente *body* apresentou uma boa precisão para classificação de mensagens legítimas, ou seja, um baixo percentual (aproximadamente $0,33\%$) de mensagens classificadas como *spams* pelo *SpamAssassin* foram classificadas como legítimas pelo *body*. Os ganhos trazidos por esse componente encontram-se também associados à antecipação da entrega de mensagens. Mais de 50% das mensagens são entregues logo na primeira avaliação realizada no primeiro filtro composto avaliado. Dessas, 100% foram identificadas de acordo com a classificação do *SpamAssassin*. Esse resultado demonstra a existência de uma redundância na avaliação realizada sobre as mensagens. Redundâncias podem ser amenizadas através de filtros compostos que antecipem a classificação.

A classificação antecipada também garantiu um melhor desempenho com relação ao tempo de processamento das mensagens no filtro composto não disjuntivo. O ganho ocorreu porque um grande percentual da carga de testes foi avaliada exclusivamente pelo primeiro componente.

Capítulo 7

Conclusões e Trabalhos Futuros

Neste capítulo apresentaremos as conclusões gerais deste trabalho. Também são apresentadas as oportunidades para o desenvolvimento de trabalhos futuros.

7.1 Conclusões

Neste trabalho apresentamos uma proposta do modelo de filtro composto (MFC) no qual diversos componentes trabalham de forma colaborativa para a detecção de mensagens maliciosas. A criação desse modelo leva em consideração a eficácia dos componentes associados, bem como os custos de processamento e erro de classificação.

Propomos também uma estratégia para a montagem de um filtro composto baseado no MFC. A estratégia foi inspirada em técnicas de mineração de dados voltadas para a geração de classificadores. Utilizamos um algoritmo similar a algoritmos de criação de árvores de decisão para gerarmos nossa árvore de detecção de mensagens maliciosas. Esta associação foi possível uma vez que nosso modelo foi definido como um grafo direcionado, onde os vértices são os componentes associados e as arestas são as mensagens trafegadas entre eles.

A estratégia foi dividida em quatro etapas distintas: levantamento de componentes; levantamento de mensagens previamente classificadas; preparação de uma base de treinamento; e geração da árvore de detecção. A última etapa difere das técnicas tradicionais de geração de árvores de classificação porque não são levadas em consideração propriedades como ganho de informação. Na estratégia apresentada utilizamos uma função de comparação, a qual tratamos como função de utilidade. Essa função mede a utilidade de cada componente que será utilizado para gerar a árvore de detecção e faz um balanceamento entre os critérios efetividade, custo de processamento e custo de erro de classificação de cada componente.

Apresentamos um estudo de caso baseado na quebra de um filtro tradicional em

vários componentes. Utilizamos para isto o *SpamAssassin* e fizemos uma comparação entre esse e o filtro composto criado a partir de uma árvore de detecção.

Mesmo apresentando ganhos com relação ao tempo de processamento, o filtro composto teve uma efetividade um pouco menor que o *SpamAssassin*. Esta redução é conseqüência da divisão do *SpamAssassin* em componentes. No filtro completo, as regras são definidas para coexistirem, trabalhando de forma conjunta e gerando uma nota final que reflita a classificação esperada para a mensagem avaliada. Cada regra busca uma característica específica na mensagem. Somente quando um conjunto de características são encontradas, uma mensagem pode ser considerada legítima ou maliciosa. Quando as regras são separadas pelo tipo, o conjunto também é quebrado fazendo com que a informação agregada seja perdida. Essa perda ocorre porque a avaliação realizada em cada nodo da árvore desconsidera a nota da avaliação anterior.

Os resultados encontrados quando comparados vários filtros compostos distintos demonstram a importância da adoção de uma estratégia para a montagem do mesmo. Os maiores ganhos foram associados ao filtro composto, que possuía como primeiro componente o de maior custo de processamento e que apresentou pior efetividade na base de treinamento. No entanto, o algoritmo de geração da árvore foi favorecido pelo fato desse mesmo componente classificar de forma mais precisa as mensagens legítimas. Isto garantiu uma entrega efetiva de mais da metade das mensagens avaliadas logo no primeiro passo da árvore reduzindo, inclusive, o custo de processamento total do filtro composto.

Apesar das limitações descritas, neste trabalho apresentamos uma abordagem nova, que explora a cooperação entre diversas técnicas existentes fazendo um balanceamento entre a efetividade e os custos das mesmas. Os resultados apresentados demonstram que, apesar de não serem considerados componentes com alta efetividade, o filtro composto gerado apresentou-se bastante próximo dos filtros tradicionais.

7.2 Trabalhos Futuros

Um importante trabalho que deve ser realizado é a geração de filtros compostos baseados em um maior número de componentes. Uma das limitações apresentadas neste trabalho foi o pequeno número de componentes adotados. Ainda com relação ao número de componentes, seria interessante o estudo com componentes sendo representados por filtros tradicionais, como o próprio *SpamAssassin* e o *BogoFilter* [7] que não apresentam dependência mútua.

Um estudo com uma carga de trabalho de um período superior também ajudará a conhecer os resultados apresentados pelo filtro composto dada a variação das taxas de

mensagens maliciosas durante um determinado período, podendo este ser horas, dias, semanas ou meses. Esse estudo permitirá compreender se o filtro seria efetivo durante todo o período ou não.

Em se tratando da variação das taxas de mensagens maliciosas dentro de períodos determinados, como é apresentado em [34], existe uma ótima oportunidade de um trabalho voltado para uma restruturação dinâmica do filtro composto. Essa restruturação poderia ser realizada através de mecanismos de aprendizagem que, avaliando os históricos das mensagens classificadas, poderia determinar qual seria a melhor configuração para um determinado momento do dia.

Outra limitação apresentada neste trabalho é a estratégia gulosa adotada. Não há garantias de que essa estratégia gerará as melhores árvores de detecção. Portanto, estratégias alternativas poderiam ser propostas de forma a explorar melhor a relação existente entre os critérios de avaliação.

As relações existentes entre os componentes poderiam ser utilizadas para compor uma estratégia alternativa. Essa poderia fazer uma avaliação das mensagens que recebem mesma classificação entre eles, além das contradições existentes. Dessa forma, poderia gerar uma base de conhecimento mais efetiva com relação aos erros e acertos de cada componente, além das reais oportunidades de correções existentes.

Uma vez que estamos falando de detecção de mensagens em diversos componentes, um estudo importante a ser realizado é uma avaliação do impacto no desempenho de uma arquitetura de servidor de correio eletrônico na qual um filtro composto fosse implementado.

Uma última oportunidade de trabalho é a criação de uma arquitetura totalmente distribuída de componentes do filtro composto. Cada componente poderia estar em um servidor diferente. Dessa forma poderiam ser criados servidores dedicados a avaliações mais rigorosas, que consomem mais recursos computacionais e servidores que trabalhariam com uma carga mais baixa, podendo abrigar mais componentes.

Referências Bibliográficas

[1] Getting off the maps rbl. http://mail-abuse.org/rbl/getoff.html.

[2] Postfix home page. http://www.postfix.org.

[3] Spam blockers home page. http://www.spam-blockers.com.

[4] Spamassassin bayesian filters. http://www.rose-hulman.edu/TSC/publications/the_kernel/.

[5] Tantalus - anti-spam milter. http://www.linuxmailmanager.com/tantalus.html.

[6] Value sensitive design project. http://www.ischool.washington.edu/vsd/.

[7] Bogofilter home page. http://bogofilter.sourceforge.net/, August 1997.

[8] Spamassassin home page. http://www.spamassassin.org, August 1997.

[9] Spamassassin public corpus (spam e legitimate email). http://spamassassin.apache.org/publiccorpus/, 2002.

[10] Rfc 3501: Internet message access protocol version 4rev1, March 2003.

[11] Shabbir Ahmed and Farzana Mithun. Word stemming to enhance spam filtering. In *Proceedings of the First Conference on Email and Anti-Spam (CEAS)*, 2004. Disponível em: `http://www.ceas.cc/papers-2004/167.pdf`.

[12] Keno Albrecht, Nicolas Burri, and Roger Wattenhofer. Spamato: An extendable spam filter system.

[13] I. Androutsopoulos, J. Koutsias, K.V. Chandrinos, G. Paliouras, and C.D. Spyropoulos. An evaluation of naive bayesian anti-spam filtering. In G. Potamias, V. Moustakis, and M.n van Someren, editors, *Proceedings of the Workshop on Machine Learning in the New Information Age, 11th European Conference on Machine Learning (ECML 2000)*, pages 9–17, Barcelona, Spain, 2000. Disponível em: `http://arXiv.org/abs/cs.CL/0006013`.

[14] I. Androutsopoulos, G. Paliouras, V. Karkaletsis, G. Sakkis, C.D. Spyropoulos, and P. Stamatopoulos. Learning to filter spam e-mail: A comparison of a naive bayesian and a memory-based approach. In H. Zaragoza, P. Gallinari, , and M. Rajman, editors, *Proceedings of the Workshop on Machine Learning and Textual Information Access, 4th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD 2000)*, pages 1–13, Lyon, France, 2000. Disponível em: `http://arXiv.org/abs/cs/0009009`.

[15] H. P. Baker. Authentication approaches. In *Proceedings of the 56th IETF meeting*, San Francisco, CA, March 2003.

[16] L. Bertolotti and M. C. Calzarossa. Workload characterization of mail servers. In *Proc. SPECTS 2000*, Vancouver, Canada, July 2000.

[17] L. Bertolotti and M. C. Calzarossa. Models of mail server workloads. *Performance Evaluation*, 46(2-3):65–76, October 2001.

[18] Hans Peter Brandmo. Solving spam by establishing a platform for sender accountability. In *Proceedings of the 56th IETF meeting*, San Francisco, CA, March 2003.

[19] Richard Clayton. Stopping spam by extrusion detection. In *Proceedings of the First Conference on Email and Anti-Spam (CEAS)*, 2004. Disponível em: `http://www.ceas.cc/papers-2004/172.pdf`.

[20] Richard Clayton. Stopping outgoing spam by examining incoming server logs. In *Proceedings of the Second Conference on Email and Anti-Spam (CEAS)*, July 2005. Disponível em: `http://www.ceas.cc/papers-2005/140.pdf`.

[21] Gordon Cormack and Thomas Lynam. Spam corpus creation for trec. In *Proceedings of the Second Conference on Email and Anti-Spam (CEAS)*, July 2005. Disponível em: `http://www.ceas.cc/papers-2005/162.pdf`.

[22] T. Cover and P. Hart. Nearest neighbor pattern classification.

[23] L. F. Cranor and B. A. LaMacchia. Spam! *Comm. of the ACM*, 41(8):74–83, August 1998.

[24] Dave Crocker. Challenges in anti-spam efforts. *Internet Protocol Journal*, 8(5), December 2005. Disponível em: `http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_8-4`.

[25] Rui Dai and Kang Li. Shall we stop all unsolicited email messages? In *Proceedings of the First Conference on Email and Anti-Spam (CEAS)*, 2004. Disponível em: `http://www.ceas.cc/papers-2004/189.pdf`.

[26] Harris Drucker, Donghui Wu, and Vladimir N. Vapnik. Support vector machines for spam categorization.

[27] Daniel Etzold. Improving spam filtering by combining naive bayes with simple k-nearest neighbor searches.

[28] Daniel Etzold. Parameter optimization for machine learning techniques for automated text classification.

[29] Deborah Fallows. Spam: How it is hurting email and degrading life on the internet. Technical Report 202-296-019, The Pew Internet & American Life Project, october 2003. http://www.pewinternet.org/reports/pdfs/PIP_Spam_Report.pdf.

[30] Tom Fawcett. 'in vivo' spam filtering: A challenge problem for data mining. *KDD Explorations*, 5(2), December 2003. Disponível em: `http://www.hpl.hp.com/personal/Tom_Fawcett/papers/spam-KDDexp.pdf`.

[31] Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurusamy, editors. *Advances in knowledge discovery and data mining*. American Association for Artificial Intelligence, Menlo Park, CA, USA, 1996.

[32] Jennifer Golbeck and James Hendler. Reputation network analysis for email filtering. In *Proceedings of the First Conference on Email and Anti-Spam (CEAS)*, 2004. Disponível em: `http://www.ceas.cc/papers-2004/177.pdf`.

[33] Luiz H. Gomes, Rodrigo B. Almeida, Luis M. A. Bettencourt, Virgílio Almeida, and Jussara M. Almeida. Comparative graph theoretical characterization of networks of spam and legitimate email.

[34] Luiz Henrique Gomes, Cristiano Cazita, Jussara M. Almeida, Virgílio Almeida, and Wagner Meira Jr. Characterizing a spam traffic. In *Proc. 2004 Internet Measurement Conference, Sponsored by ACM SIGCOMM and in cooperation with USENIX*, Taormina, Sicily, Italy, october 2004.

[35] Paul Graham. A plan for spam. In *Reprinted in Paul Graham, Hackers and Painters, Big Ideas from the Computer Age, O'Really, 2004*, 2002. Disponível em: `http://www.paulgraham.com/spam.html`.

[36] Paul Graham. Better bayesian filtering. In *Spam Conference*, January 2003.

[37] Alan Gray and Mads Haahr. Personalised, collaborative spam filtering. In *Proceedings of the First Conference on Email and Anti-Spam (CEAS)*, 2004. Disponível em: `http://www.ceas.cc/papers-2004/132.pdf`.

[38] Jiawei Han and Micheline Kamber, editors. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, August 2000.

[39] S. Hird. Technical solutions for controlling spam. In *Proceedings of the Annual Meeting of the Australian UNIX and Open Systems User Group (AUUG)*, 2002. Disponível em: `http://security.dstc.edu.au/papers/technical_spam.pdf`.

[40] Geoff Hulten, Anthony Penta, Gopalakrishnan Seshadrinathan, and Manav Mishra. Trends in spam products and methods. In *Proceedings of the First Conference on Email and Anti-Spam (CEAS)*, 2004. Disponível em: `http://www.ceas.cc/papers-2004/165.pdf`.

[41] J. S. Kong, P. O. Boykiny, B. A. Rezaei, N. Sarshar, and V. P. Roychowdhury. Scalable and reliable collaborative spam filters: Harnessing the global social email networks.

[42] Robert Kraut, Shyam Sunder, James Morris, Rahul Telang, Darrin Filer, and Matt Cronin. Markets for attention: Will postage for email help? In *Proceedings of the 2002 ACM Conference on Computer Supported Cooperative Work, New Orleans, Louisiana, USA*, pages 206–215, New York, NY, 2002. ACM Press. Disponível em: `http://www.som.yale.edu/Faculty/sunder/Email/Emarket.pdf`.

[43] Barry Leiba and Nathaniel Borenstein. A multifaceted approach to spam reduction. In *Proceedings of the First Conference on Email and Anti-Spam (CEAS)*, 2004. Disponível em: `http://www.ceas.cc/papers-2004/127.pdf`.

[44] John Levine. Experiences with greylisting. In *Proceedings of the Second Conference on Email and Anti-Spam (CEAS)*, July 2005. Disponível em: `http://www.ceas.cc/papers-2005/120.pdf`.

[45] Kang Li, Calton Pu, and Mustaque Ahamad. Resisting spam delivery by tcp damping. In *Proceedings of the First Conference on Email and Anti-Spam (CEAS)*, 2004. Disponível em: `http://www.ceas.cc/papers-2004/191.pdf`.

[46] Charles X. Ling, Qiang Yang, Jianning Wang, and Shichao Zhang. Decision trees with minimal costs. In *Proceedings of 2004 International Conference on Machine Learning (ICML'2004)*, 2004.

[47] Thede Loder, Marshall Van Alstyne, and Rick Wash. An economic solution to the spam problem. In *Proceedings of the ACM Conference on Electronic Commerce*, 2004. Disponível em: `http://www.citi.umich.edu/u/rwash/pubs/spam-acm.pdf`.

[48] Daniel Lowd and Christopher Meek. Goodword attacks on statistical spam filters.

[49] Nicola Lugaresi. European union vs. spam: A legal response.

[50] Bart Massey, Mick Thomure, Raya Budrevich, and Scott Long. Learning spam: Simple techniques for freely-available software. In *Proceeding of the 2003 Usenix Annual Technical Conference, Freenix Track*, 2003. Disponível em: `http://nexp.cs.pdx.edu/twiki-psam/pub/PSAM/PsamDocumentation/spam.pdf`.

[51] Eirinaios Michelakis, Ion Androutsopoulos, Georgios Paliouras, George Sakkis, and Panagiotis Stamatopoulos. Filtron: A learning-based anti-spam filter. In *Proceedings of the First Conference on Email and Anti-Spam (CEAS)*, 2004. Disponível em: `http://www.ceas.cc/papers-2004/142.pdf`.

[52] Evangelos Moustakas, Prof C. Ranganathan, and Dr. Penny Duquenoy. Combating spam through legislation: A comparative analysis of us and european approaches.

[53] Myers, J. G., and M. T. Rose. Rfc 1939: Post office protocol version 3, May 1996.

[54] Patrick Pantel and Dekang Lin. SpamCop: A spam classification and organization program. In *Learning for Text Categorization: Papers from the 1998 Workshop*, Madison, Wisconsin, 1998. AAAI Technical Report WS-98-05. Disponível em: `http://www.cs.ualberta.ca/~ppantel/Download/Papers/aaai98.pdf`.

[55] Jon Postel. Rfc 706: On the junk mail problem. http://www.ietf.org/rfc/rfc706.txt, November 1975.

[56] Jonathan B. Postel. Rfc 821: Simple mail transfer protocol. http://www.ietf.org/rfc/rfc821.txt, August 1982.

[57] Vipul Ved Prakash. Vilpul's razor. http://razor.sourceforge.net/.

[58] Matthew Prince, Arthur M. Keller, and Ben Dahl. No-email-collection flag.

[59] Matthew B. Prince, Lee Holloway, Eric Langheinrich, Benjamin M. Dahl, and Arthur M. Keller. Understanding how spammers steal your e-mail address: An analysis of the first six months of data from project honey pot.

[60] J. Ross Quinlan, editor. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[61] Lee Rainie and Deborah Fallows. The impact of can-spam legislation. Technical Report 202-296-019, The Pew Internet & American Life Project, March 2004. http://www.pewinternet.org/pdfs/PIP_Data_Memo_on_Spam.pdf.

[62] Isidore Rigoutsos and Tien Huynh. Chung-kwei: a pattern-discovery-based system for the automatic identification of unsolicited e-mail messages (spam). In *Proceedings of the First Conference on Email and Anti-Spam (CEAS)*, 2004. Disponível em: `http://www.ceas.cc/papers-2004/153.pdf`.

[63] Gordon Rios and Hongyuan Zha. Exploring support vector machines and random forests for spam detection. In *Proceedings of the First Conference on Email and Anti-Spam (CEAS)*, 2004. Disponível em: `http://www.ceas.cc/papers-2004/174.pdf`.

[64] Mehran Sahami, Susan Dumais, David Heckerman, and Eric Horvitz. A bayesian approach to filtering junk e-mail. In *Learning for Text Categorization: Papers from the 1998 Workshop*, Madison, Wisconsin, 1998. AAAI Technical Report WS-98-05. Disponível em: `http://roboticsCover.stanford.edu/users/sahami/papers-dir/spam.ps`.

[65] G. Sakkis, I. Androutsopoulos, G. Paliouras, V. Karkaletsis, C. D. Spyropoulos, and P. Stamatopoulos. A memory-based approach to anti-spam filtering for mailing lists. *Information Retrieval Journal*, 6(1), 2003. Disponível em: `http://www.eden.rutgers.edu/~gsakkis/docs/IR2003.pdf`.

[66] Georgios Sakkis, Ion Androutsopoulos, Georgios Paliouras, Vangelis Karkaletsis, Constantine D. Spyropoulos, and Panagiotis Stamatopoulos. Stacking classifiers for anti-spam filtering of e-mail. In *Proceedings of EMNLP-01, 6th Conference on Empirical Methods in Natural Language Processing*, Pittsburgh, US, 2001. Association for Computational Linguistics, Morristown, US. Disponível em: `http://www.arxiv.org/abs/cs.CL/0106040`.

[67] Richard Segal, Jason Crawford, Jeffrey Kephart, and Barry Leiba. Spamguru: An enterprise anti-spam filtering system. In *Proceedings of the First Conference on Email and Anti-Spam (CEAS)*, 2004. Disponível em: `http://www.ceas.cc/papers-2004/126.pdf`.

[68] Rhyolite Software. Distributed checksum clearinghouse. http://www.rhyolite.com/anti-spam/dcc/.

[69] Konstantin Tretyakov. Machine learning techniques in spam filtering.

[70] R. D. Twining, M. M. Williamson, M. Mowbray, and M. Rahmouni. Email prioritization: Reducing delays on legitimate mail caused by junk mail. In *Proc. Usenix Annual Technical Conference*, Boston, MA, June 2004.

[71] Bill Yerazunis. The plateau at 99.9% accuracy, and how to get past it. In *Proceedings of the Spam Conference*, 2004. Disponível em: `http://crm114.sourceforge.net/Plateau_Paper.pdf`.

[72] William S. Yerazunis, Shalendra Chhabra, Christian Siefkes, Fidelis Assis, and Dimitrios Gunopulos. A unified model of spam filtration. 2005.

[73] Zhenyu Zhong, Kun Huang, and Kang Li. Throttling outgoing spam for webmail services. In *Proceedings of the Second Conference on Email and Anti-Spam (CEAS)*, July 2005. Disponível em: `http://www.ceas.cc/papers-2005/164.pdf`.