CASSANDRA: A Collaborative Anti Spam System Allowing Node Decentralised Research Algorithms

by

Alan Gray

A dissertation submitted to the University of Dublin,
in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science

Department of Computer Science,
University of Dublin, Trinity College

September, 2003

Declaration

Signature of Author . . . .

Alan Gray

15 September, 2003

Acknowledgements

I would like to thank my supervisor, Mads Haahr, for his direction and advice during the project. Special thanks to my family for their love and support during this year. Thanks are due to Ioanna Stamouli, Greg Hayes, Andrew Jackson and Neil O'Connor for helping to test the system (and playing their roles!). Additional thanks are due to Stephen Gray, Ioanna Stamouli and Shane O'Conchuir for proof-reading this thesis. Finally, many thanks to my fellow NDS-ers who provided the healthy dose of insanity that kept me going through this year.

Abstract

Email has become a critical tool in many people's everyday lives, both professionally and personally. It is easily accessible, inexpensive, fast, versatile and is far-reaching both in terms of the number and the spectrum of people to whom it can deliver information. For exactly the same reasons, email is being exploited by mass-marketers, and the unsolicited bulk email (or spam) problem is reaching epidemic proportions, with half of the daily worldwide emails sent being spam. Some sources estimate that spam will cost companies a total of US$20.5 billion in 2003 alone.

The first generation of content-based spam filters has proven inadequate to the task of sorting legitimate email from spam email. This is because of the implicit assumptions in their design and the fact that the filters lack the sophistication to be able to classify all email correctly. Collaborative filtering has proven to be more successful than content-based filtering, but still suffers from lack of scalability and poor performance, due to centralisation of data and the assumption that all users are equally likely to receive the same spam.

This thesis reviews the state of the art in content-based and collaborative spam filters. The assumptions that cause poor accuracy, performance and scalability are identified. The concept of personalised, collaborative spam filtering is introduced. Personalised, collaborative filtering does not make the same inaccurate assumptions that other filters do, and are characterised by disseminating information about new spam to the users that are most likely to receive that spam. A versatile, extensible framework that details how personalised, collaborative spam filters can be built and interact is presented. This framework defines objects and messages that can be composed to form different network topologies and filtering systems.

A peer-to-peer (P2P), signature-based proof-of-concept instance of the framework is implemented on a popular Mail User Agent (MUA). This framework is tested on simulated spam and non-spam email by real users. The results of this experiment show that the implementation is accurate and efficient. The results also validate the framework and present directions for future work.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Email has become a critical tool in many people's everyday lives, both professionally and personally. It empowers people to disseminate and discuss ideas and opinions with many others through the use of mailing lists. It enables businesses to keep in close and regular contact with many clients, without the need for time restrictions and "opening hours". Email is easily accessible, inexpensive, fast, versatile, cheap and is far-reaching, both in terms of the number and the spectrum of people it can deliver information to. Unfortunately, it is exactly these qualities that are being abused and threaten to make email unusable.

Because email is so cheap, it means that a user can send tens of thousands of emails per day without incurring any significant cost to themselves. Mass-marketeers are abusing this fact by sending the same email to millions of people, advertising services and products often of dubious legality and morality. The recipients of the email are not targeted to receive the email, so the overwhelming majority of these mass-mailed advertisements are of no interest to the vast majority of the recipients. This phenomenon is known as "spam", and spam email has reached epidemic proportions. It is now roundly reported that spam constitutes at least 50% of all emails sent, as of September 2003 [5, 40, 54].

The problems with spam email are numerous. At the most primary level, spam is unwanted and often offensive email that annoys the vast majority of recipients. Legitimate email is displaced by spam and can often be lost in the deluge. At its most insidious, spam advertises scams, frauds and illegal activities. It steals resources from others, because the cost associated with spam lies on the underlying networks that relay email and the recipients' networks. Spam filters attempt to address the spam problem by identifying and eliminating spam emails.

1.2 Spam Filters

There have been many different approaches to implementing unilateral spam filters. These attempts have focused variously on identifying the source of spam, identifying the stereotypical content of spam (spams tend to have a certain “style”) or identifying bulk emails, each with only moderate success. There are three main reasons for this sub-optimal performance - insufficient accuracy, large effort associated with maintaining and updating the knowledge base and the scope of the filter.

Content and rule-based filters attempt to create a set of rules or features that can distinguish spam emails from normal emails. These systems are ill-suited to evolving to meet the demands imposed by mutating spam content and format. This is because the overhead of identifying new spams and creating rules that can classify them is burdened on the creator of the rule/feature base. At the same time these new rules must not conflict or contradict existing features/rules in the knowledge-base.

Mechanisms that filter email based purely on its origin are equally fraught with problems. They are either too selective, refusing delivery of many legitimate unsolicited emails, or they are too permissive and slow-reacting, permitting many spams through before the list is updated. Additionally, it is infeasible to try to maintain a listing of all possible origins of spam and/or all possible origins of non-spam.

False positives (non-spams that have been misclassified as spam) are the bane of these type of spam filters. Another shortcoming, though not as serious, are false negatives (spams that have been misclassified as non-spams). The presence of these is indicative that the precision of the spam filter is not good enough. Content-based and characteristic-based filters cannot distinguish spam from legitimate email with sufficient accuracy [32]. Collaborative filters do not suffer from these same shortcomings, because they approach the issue from a different perspective.

1.3 Collaborative Spam Filters

Collaborative filters do not attempt to model what characterises spam emails. Underlying this decision is the realisation that spam mutates and evolves, so any model will quickly become outdated and require updating. Collaborate filters therefore ignore the actual content of the email and instead concentrate on the fact that a given email will either be spam or non-spam. Bulk-mailing in general, and spamming in particular, is an impersonal process where the recipient is not distinguished. This means that there is a one-way communication channel and that many recipients will receive the same spam email.

In the main, recipients of spam can tell at a glance whether or not a given email is spam. This fact, and the fact that there are many recipients of the same spam are leveraged by collaborative spam filters. Classification of emails as spam is done by the users of the collaborative filters. This virtually eliminates false positives, because humans prove to be much more accurate at labelling spam than any automated process. By leaving it to the users of the system to label spam, collaborative systems are geared toward evolvability by their design. This is because new mutations and formats of spam will be labelled by the users as they emerge and added to the collection of known spam.

The two major detriments to the collaborative approach lie in the tacit assumptions made by both collaborative and rule-based filters. They are naïve assumptions, and they cause poor scalability and performance and also contribute to the false positive rate. These two assumptions are:

- **That all users are equally likely to receive the same spams.** A given email address will not appear on all spam-lists, so therefore will not receive all spams. By making the assumption that all spams are relevant to all users, the performance of the system is detrimentally affected. This is because each check made on behalf of the user must be made against all known spams and not just those that might be received by a given user.
- **That all users consider the same emails to be spam.** Spamming is profitable, otherwise it would not exist. For every spam that is sent, approximately 99.9% of the recipients do not want to receive it [3]. However, this means that 0.1% of recipients that

are interested in the contents of the email. All collaborative spam filters either punish users for making recommendations that go against the general consensus or at best, ignore them. Participants who are interested in the content of the spam they receive make revocations and recommendations based on what they perceive to be non-spam email. This contributes significantly to the false positive rate. A 0.1% false positive rate is unacceptable for a spam filter, yet the current generation of spam filters are designed towards this by assuming all users consider the same emails to be spam.

1.4 CASSANDRA - A Collaborative Anti Spam System Allowing Node Decentralised Research Algorithms

Users receive spam because they are on spam-lists that spam is sent to. Therefore, users need only be notified of spam that is sent to those lists. A personalised, collaborative spam filter is one that delivers the most relevant spam notices to each user from the collection of all spams that are reported by the group.

The first example of a personalised, collaborative spam filter was presented was Cottereau's MSc. thesis [34]. In [34], the first tacit assumption listed above is noted and his Anti-Spam Network (ASN) avoids making it. However, the second point above is not addressed in his solution. Additionally, the ASN's topology is fixed in a vanilla peer-to-peer topology as is the (relatively naïve) trust management system, which makes it very difficult to extend the solution.

CASSANDRA is a framework for personalised, collaborative spam filtering. It can be seen as an abstraction of Cottereau's work, in that the ASN is one possible implementation that is permitted within the framework. The network topology, trust management system (both unilateral and distributed), clustering mechanism and peer-ranking schemes are not mandated by the CASSANDRA protocols. Neither are the transport layer, filter placement or node-design. This framework is deliberately designed to be extensible and permit many different algorithms for each facet of the system with the explicit intention of enabling more sophisticated research into node-decentralised, personalised, collaborative spam filtering. In addition to the protocols, this work contributes an implementation allowed by CASSANDRA and presents real feedback from real user data.

1.5 Document Road-map

Chapter One outlines the motivation behind this work. It describes the spam epidemic and presents spam filtering and collaborative spam filtering. The implicit assumptions that undermine these processes are explained and the concept of a personalised, collaborative spam filter is introduced. Finally, the document road-map is given.

Chapter Two presents background information in order to put this work in context. Spam is defined and a broad overview of spam-combating techniques presented. The Electronic Mail protocols are outlined and the difference between centralised, distributed and decentralised topologies defined.

Chapter Three gives a detailed study on the various mechanisms employed in spam filters. A brief survey of how spammers attempt to circumvent these mechanisms is also given.

Chapter Four reviews the state of the art in collaborative spam filters.

Chapter Five details the design of the CASSANDRA framework. The possible choices for transport layer are presented and evaluated. A generalised structure for implementing a CASSANDRA-compliant component at participating nodes is presented. This structure is extensible and can be applied at either the MTA or MUA level. The objects and messages that are provided by the CASSANDRA framework are detailed. The protocols are used to show how different network topologies can be constructed from them.

Chapter Six presents a proof-of-concept of a personalised, collaborative spam filter that is permitted by the CASSANDRA framework. The specification of the filter is given and the various components of the design are described. Inter-peer communication is detailed, as is the filtering process and user interaction with the plug-in.

Chapter Seven details the experimental set-up and presents the results obtained. From these results, conclusions about the effectiveness of the prototype implementation are drawn. Finally, the implementation and the framework as a whole is evaluated using the criteria used in [34].

Chapter Eight summarises the conclusions drawn from the experiments in Chapter 7. Conclusions are drawn about the CASSANDRA framework as a whole. Some of the many future directions for this work are outlined. Finally, this chapter summarises and concludes the thesis.

Chapter 2

Background

In order to put this work in context, some background information on spam email is presented. We provide a robust and accurate definition of spam as it relates to email and what characterises it. A broad overview of the approaches for combating spam are presented, with a more in-depth discussion of the prevalent technological techniques to follow in Chapter 3. Salient information is given on Electronic Mail protocols and the technologies associated with them. Finally, a discussion is presented on network topologies and the differences between centralised, distributed and decentralised systems.

2.1 Spam

Spam is (reputedly) so named after a famous Monty Python sketch [15]. In a café, there is so much Spam[1]. on the menu that customers are unable to order a menu item that does not contain it. A few Vikings are present and shout "Spam" with increasing volume until it is nearly impossible to hear the person trying to speak. This is an accurate analogy of how most Internet users feel about communicating over the medium. The sheer volume of spam threatens to make Internet communication (by newsgroup posting or email) infeasible because of the ever-decreasing signal-to-noise ratio. Another possible etymology of the term, or at least

[1]"Spam" is a trademark of Hormel Ltd. Spam (short for Spiced Ham) is a canned meat product consisting of chopped pork pressed into a loaf.

its first appearance on a newsgroup comes from the MUD/MUSH community [13]. The story relates that a user, discouraged by the lack of intellectual discourse displayed by posters on the forum, wrote a keyboard macro to post “SPAM” every few seconds until he was booted from the forum.

Spam therefore, as it pertains to electronic communication, is not limited to ordinary email communication. On the Internet, spam is also rife on newsgroups. In this case, it has a fairly straightforward definition, as per spam.abuse.net [56]. This definition is simply that the post has appeared at least twenty times in various parts of the forums, regardless of the content of the post. Mobile phone spam is emergent also, with an estimated eighty percent of the 950 million daily messages on Tokyo's DoCoMo being unsolicited, according to CNN [10]. However, for the purposes of this work, where “spam” is used, it refers solely to spam email unless noted otherwise.

2.1.1 Definition Of Spam Email

There is no definitive definition for spam email that is universally accepted. A first approximation of a definition is simply that spam is unwanted email. This captures the essence of spam, but is too broad a definition. The two most generally accepted definitions are Unsolicited Commercial Email (UCE) [11] and Unsolicited Bulk Email (UBE) [47]. However, these definitions are both incomplete and inaccurate.

The term UCE does not cover cases where there are no explicit business opportunities, nor does it cover the case where the content of the email deals with illegal ventures. UCE as a definition also covers what would be viewed as legitimate commercial applications, such as an after-sales service for goods and services purchased over the Internet that are targeted to the recipient of the email.

The definition of spam as UBE has an inherent flaw, insofar as many countries have acts protecting freedom of speech, leaving it open to challenges on legal grounds. Prescinding from these legal arguments, UBE is not specific enough a definition. An example of this would be a family member sending an email to a number of relatives informing them of an event (good or bad). The recipients of the email would probably not have known beforehand about the email and so may not have the explicit opportunity to solicit the email prior to its being sent.

A recent publication by the Australian National Office for the Information Economy [43] states that "*For the purposes of this report, spam is defined as unsolicited electronic messaging, regardless of its content*". In isolation, this loose definition is very disturbing as it would mean that all email would have to be accepted by the recipient prior to the email being sent, with no requirement that it has been sent in bulk. In essence, this would mean that only whitelisted people could send you emails (see Section 3.3.1 for more on whitelists). This serves as an reminder of how careful legislators must be when drafting laws to deal with spam. The report acknowledges this by making reference to the fact that

> *Arriving at an agreed definition of spam is a potentially contentious issue, as the direct marketing industry, ISPs, spammers, blacklisters and privacy and consumer groups have their own interests and views.*

The most complete and accurate definition of spam that is in use today is that which is used by two of the best well-known spam blacklist sites [38, 67], and it is this definition of spam that is assumed for the purposes of this work:

> *An electronic message is "spam" IF: (1) the recipient's personal identity and context are irrelevant because the message is equally applicable to many potential recipients; AND (2) the recipient has not verifiably granted deliberate, explicit, and still-revocable permission for it to be sent; AND (3) the transmission and reception of the message appears to the recipient to give a disproportionate benefit to the sender.*

This definition is particularly interesting in that there is no requirement for the email to have any forged or fraudulent content or headers. The reason for this is that it is the act of indiscriminately mass-mailing an email that makes it spam, and not the content contained therein. It is also interesting to note that a number of enacted state laws in the US [58–60] make it a requirement that there be falsified information (headers or content) in the email.

2.1.2 What Characterises Spam?

Given that multifarious definitions exist for what exactly constitutes spam, it could be expected that humans have difficulties in identifying spam. This is not the case. In fact, most users can intuitively recognise a spam email at a single glance. This is because there are a number of defining characteristics of spam. A complete description is beyond the scope of this work, but a few of the major characteristics will be presented here. A thorough and current analysis of spam can be found in Lambert's M.Sc. thesis [2].

The stereotypical spam email will have some or all of the following traits:

- No personalisation in the content of the email or particular relevance to the recipient of the email. It is this characteristic that primarily identifies a spam as the email will not be targeted to the recipient of the email.
- No informational content of interest to the recipient. Again, this typifies spam in that the recipient was not explicitly chosen to receive the email on the grounds of any market research, prior business relationship, or any targeting save the fact that his/her email is known to the spammer.
- Forged header information including, but not limited to: relaying servers; originating IP address and/or email address; time zone; send time; list of recipients.
- The body of the email is primarily HTML marked-up content.
- The email contains lines comprised entirely of uppercase text.

The preceding list of tell-tale signs makes no reference to the actual intent behind the email, be it selling pornography, political propaganda, or advertisements for illegal products/services. Again, this is because it is not the content of the email that identifies it as spam, but merely the action of mass-mailing it. However, the actual body of the spam email will display some typical characteristics. A recent report [19] by the US Federal Trade Commission reports that there are eight general categories of spam, summarised in Table 2.1.

Type of Offer	Description	Fraction
Investment/Business Opportunity	work-at-home, franchise, chain letters, etc.	20%
Adult	pornography, dating services, etc.	18%
Finance	credit cards, refinancing, insurance, foreign money offers, etc.	17%
Products/Services	products and services, other than those coded with greater specificity.	16%
Health	dietary supplements, disease prevention, organ enlargement, etc.	10%
Computers/Internet	web hosting, domain name registration, email marketing, etc.	7%
Leisure/Travel	vacation properties, etc.	2%
Education	diplomas, job training, etc.	1%
Other	catch-all for types of offers not captured by specific categories listed above	9%

Table 2.1: *Breakdown of spam by category, as measured by the US FTC on April 30th, 2003.*

2.1.3 Counting The Cost Of Spam

It is virtually impossible to calculate the true cost of spam. This is due in no small part to the fact that it is nigh impossible to accurately gauge the actual number of spams sent every day. The Radicati Group in Palo Alto estimate that spam will account for 45% of the 10.9 trillion emails sent in 2003 [6]. This figure equates to approximately 4.9 trillion spam emails. By comparison, Brightmail reported on August 20th 2003 [5] that in July 2003, spam email comprised at least 50% of email traffic. Brightmail filtered 61 billion emails this July on behalf of its subscribers. Given that Brightmail filters nearly 10% of all worldwide email traffic and that half of these are spam, this represents a more conservative estimate of about 3.66 trillion spam emails in 2003, based on July's statistics. Although more conservative than the Radicati Group's estimate, this figure is still colossal and the costs associated with spam are equally non-trivial. There are a number of components that make up the true cost of a spam email, and these are discussed below.

Form	Cost to Sender	Cost to Recipient	% of Cost borne by Sender
Legal			
Telemarketing	$1.00	$0.10	91%
Postal Mail	$0.75	$0.10	88%
Illegal			
Fax	$0.3	$0.10	23%
Automated Phone	$0.7	$0.10	41%
Of Uncertain Legality			
Spam	$0.00001	$0.10	0.01%

**Cost figures per contact are estimated*

Table 2.2: *Cost Comparison Of Unsolicited Marketing Methods.*

2.1.3.1 Cost To The Spammer

Sending spam is a lucrative business. If we take as a premise that spammers spam because it is a profitable venture, then it is clear that spammers will continue to spam until it becomes prohibitively expensive to do so. Assuming that the spammer is not sued and has no legal costs (as is the case for the majority), the sole cost to the spammer is how much it costs to send an email. It is a fact that the cost to the spammer is orders of magnitude less than the cost to the recipient of the spam email. This is quite the opposite to the usual fraction of the cost borne by the advertiser for unsolicited mass-marketing methods, estimated in [64] and summarised in Table 2.2.

2.1.3.2 Lost Productivity

In a research report published recently [44], it is estimated that spam email costs US companies $874 per employee every year in lost productivity. This estimate is based on an average pay of $30 per hour and a 40 hour work week, compiled from interviews with employees and IT administrators from 76 US companies. Among the findings of the report are the following statistics:

- The average employee receives 13.3 spam emails per day.
- Spam is responsible for a 1.4% decrease in each employee's productivity every year.
- For every 690 employees, a full-time IT staff person will be required to manage spam alone.
- Companywide spam filters reduce the loss in productivity by only 26%.

From the billions of dollars being paid to employees in wages and salaries each year, it is clear that a 1.4% reduction in employee productivity represents a huge figure.

2.1.3.3 Server Costs

The server costs of spam email in enterprises is closely tied to the volume of email that must be processed by the enterprise on a daily basis in order to conduct its business. According to an article published very recently [54], the hard IT cost due to spam equates to a cost of $49 per user mailbox per year in 2003. The article also reports that:

> *If nothing is done to curtail the growth of spam, by 2007, we estimate that a 10,000-user company will have to maintain 25 additional servers to process spam, costing it a whopping $257 per mailbox per year. Calculated on a worldwide basis, the IT cost of dealing with spam will rise from $20.5 billion in 2003, to a staggering $198 billion by 2007.*

2.1.3.4 Anti-Spam Software Costs

Another associated cost of dealing with spam is the money outlaid in trying to combat spam. This means anti-spam software in all its forms. According to [6], the research firm International Data Corp reported that corporations paid out a total of $120 million in 2002 for anti-spam products. With the current rate of increase in the proliferation of spam, it is expected that this figure will be at least doubled for 2003, at an estimated quarter of a billion dollars.

2.1.3.5 Lost Opportunity Due To False Positives

False positives are the bane of anti-spam systems. A false positive occurs when a legitimate email is mistakenly classified as spam by an automatic filtering system. The actual cost of a misclassification is entirely related to the actual content of the email itself, and in that respect is unquantifiable in any accurate sense. However, the relative cost of a false positive can be determined by how aggressive the anti-spam filtering policy is. Spam filtering policies are discussed below.

2.1.3.6 Spam Filtering Policies

Given that an email has been classified as spam, there are a number of policies that can be implemented when deciding what to do with the offending email. The most aggressive policy is that when an email has been classified as spam, it is immediately deleted from the server. This is also the most transparent approach, as the intended recipient never knows of the existence of the email. This approach is also the most costly in terms of lost opportunity as the damage caused by the false positive is irrevocable. As a policy, it should only be used where the loss of an email is non-critical (in personal correspondence, for example).

A less aggressive approach is to deliver the email to a junk email folder. A user can periodically check his/her junk email folder for any false positives and can update rules for his/her spam filter to allow similar email through in future. The net effect of this approach can be the delay of the delivery of the email (in terms of when the user learns of its existence) in the case where it is found in the junk folder. In the case where it is not noticed in the junk folder, the email is effectively lost to the user.

Another less aggressive policy is to hold the email back in a queue until the sender of the email has been verified. This is typically done by sending an email to the return address asking the sender to confirm that it is an active email account. Unfortunately, this will prevent emails automatically sent by mailing lists or order confirmation emails from e-commerce sites from being delivered. This policy is mostly transparent to the end user, except for the time delay whilst waiting for verification.

The least aggressive and most invasive approach is simply to add a header to the email, or adding a notice to the subject line of an email (inserting the text "***SPAM***" at the beginning of the subject line, for instance). This approach can be adopted where the timely delivery of an email is very important. The modification to the header or subject line serves as a hint to the user that the email might be spam. A user can then create inbox rules to filter emails containing this header into a junk mailbox, or even directly to the trash, depending on how aggressive an anti-spam policy the user wishes to enforce.

2.1.3.7 Non-Financial Costs

These are the costs that do not have a monetary value easily estimable. These include

- Loss in innocence of a child who receives sexually explicit content from spam that advertises pornographic material/services. It is estimated that four of every five children with email accounts receive sexually explicit spam.
- Loss in reputation of a company whose identity has been spoofed on spam that has advertised materials of dubious legality/morality.
- Loss in reputation of a company whose server has been used to relay spam and has been mistakenly identified as the spammer.
- The annoyance factor associated with receiving a deluge of email with offensive material or as a result of misplacing legitimate email.
- Reduction in quality of service to subscribers as ISPs devote more bandwidth, processing power and storage space to dealing with spam.

2.2 Combating Spam - An Overview

Although spam is a technological phenomenon, there are a number of fronts on which spam is being countered. This section outlines some of the approaches that can be taken. It is clear that no single one of these approaches will prove a panacea to the problem of spam. In order to successfully combat spam, there needs to be clear, consistent and comprehensive legislation brought in that is backed up by rigorous and accurate spam filtering and reporting techniques.

2.2.1 Sociological

Spamming is perhaps one of the purest expressions of capitalism. Email provides a medium of communication that is extremely cheap and accessible and it can reach a huge section of Earth's inhabitants with a single email. It is therefore to be expected that people will try to exploit the medium as a means of mass-marketing. As noted earlier, spamming is a lucrative business and will only attract more people to it as long as it continues to be so profitable. "Legitimate" spammers (i.e. those that do not send fraudulent mails or spam advertising illegal products and services, but do send it without solicitation) maintain that what they are doing is legal within the jurisdiction of the area the spam is sent from, and will continue to do so unabashedly.

Some futile attempts have been made to try to change spammers' attitudes. The "Name and Shame" approach, whereby the names and addresses of spammers are published in an attempt to put public pressure from their peers on them has been ineffective. Spamhaus operate ROKSO (Register Of Known Spam Operations) which collates and publishes information on hard-line spam operations that have been kicked off three or more consecutive Internet Service Providers for serious spam offences. According to the Spamhaus Project [68], 90% of all spam received by Internet users in North America and Europe is sent by a hard-core group of under two hundred spam outfits, almost all of whom are listed in the ROKSO database. It is clear however, that spammers' attitudes will not be changed by consideration for the recipients of their spam, but by financial and legal considerations.

2.2.2 Legislative

There are already a myriad of laws in existence regarding spam. These laws have been characterised by contrasting and often contradictory requirements. This is in no small part due to the fact that legislation typically lags technology by two to three years, meaning that new legislation must be very forward-facing and preemptive. There is a danger that any hastily-approved legislation could be draconian and ill thought-out, as happened with the Digital Millennium Copyright Act (DMCA) [1] in response to the problems associated with illegal mp3 sharing over the Internet. It is beyond the scope of this work to go into a detailed discussion of these laws, but the major current debate warrants mention.

2.2.2.1 Opt-In Versus Opt-Out

The European Union has issued a directive regarding mass-marketing and solicited and unsolicited bulk email [17]. The scheme for mass-marketing here is described as "Opt-In", as the recipient of a mass-marketing email must explicitly and provably give still-revocable consent to be placed on the mailing list. Any mass-email that is sent without explicit solicitation from the recipients is deemed spam and is outlawed by this directive, regardless of its content.

By comparison, some of the bills before the US Senate at the moment advocate what are described as "Opt-Out" schemes. In this scenario, mass-marketers are required to place a link to an email address or URL at which the recipient must register their desire to opt out of receiving any further emails sent to the list. Having an individual opt-out mechanism for every email-list suffers from problems of scale (assuming that the senders of the email actually honour the opt-out decision made by the user). EuroCAUCE estimate that if only 1% of Europe's 18.4 million businesses decided to operate in this way, it would take a user an entire working year to opt-out of all the mailing lists at a rate of two per minute [16]. The Spamhaus Project summarises the opt-in versus opt-out debate with the information presented in Table 2.3 [66] .

Confirmed opt-in is the obvious choice for a practical and scalable legislative scheme for mass-mailing regulation. However, spammers and marketing associations fronting for spammers attempt to fudge the issue and confuse recipients by using their own meaning for the term "Opt-In":

- **Opt-In:** Wording used by spammers to mean that the recipient "has not opted-out, therefore they are opt-in". This usually means any address the spammer can get hold of.
- **Double-Opt-In:** Used usually by spammers to imply the recipient has "opted-in twice". The first time was when the address was obtained and "opted-in" by the spammer without consent, the second when the recipient failed to opt-out after receiving spam.
- **Triple-Opt-In:** Verbatim from a spam outfit (BricecoInc.com), who describe themselves as a "Leading provider of marketing Solutions": "A triple-opt-in email is a person who subscribes and fills out name, address, and interest". This equates to what is called "Confirmed Opt-In" by legitimate mass-marketers.

	Spam
Opt-Out	Spam. Any bulk email sent to recipients who have not expressly registered their permission to be placed on the mailing list, and which requires recipients to opt-out to stop further unsolicited bulk mailings, is by definition, spam.
Unconfirmed Opt-In	The recipient has (according to the sender) apparently, but not verifiably, somehow somewhere initiated a request for the address to be included on the sender's mailing list. The sender has subscribed the address to the mailing list without verifying if the address owner has in fact granted permission or not. Unconfirmed Opt-in presumes anyone can subscribe anyone, and the recipient will receive bulk mailings whether he likes it or not until he opts-out. As there is no verification, all spammers naturally claim to practice 'Opt-in' which is why the vast majority of spam on the Internet claims to be Opt-in. **In case of dispute** The burden of proof is on the RECIPIENT to prove that he/she did not opt-in. This is almost always impossible for the recipient to prove.
	Legitimate Bulk Email
Confirmed Opt-In	Also known as "Closed-loop" or "Verified Opt-in". The recipient has verifiably confirmed permission for the address to be included on the specific mailing list, by confirming (responding to) the list subscription request verification. This is the standard practice for all Internet mailing lists, it ensures users are properly subscribed from a working address and with the address owner's consent. **In case of dispute** The burden of proof is on the SENDER to prove the recipient did opt-in. The sender is fully and legally protected because the confirmation receipt proves that the recipient did in fact opt-in and grant consent.

Table 2.3: *Summary of Opt-In and Opt-Out schemes.*

2.2.3 Technological

As with any problem, possible solutions can fall into two general classes - those that combat the symptoms of the problem and those that combat the roots of the problem. Spam can be described as the result of people abusing and exploiting a system that is not authenticated and open to such abuse. Anti-spam filters can be thought of addressing the symptoms of this abuse, i.e. the deluge of spam. AMTP - a new, more secure derivative of SMTP can be described as addressing the root of the problem.

2.2.3.1 Anti-Spam Filters

Technology has lead the way thus far in the fight against spam. This mainly takes the form of anti-spam filters, email address obfuscation and sender blacklists/whitelists. These approaches are described in more detail in Chapter 3.

2.2.3.2 AMTP - Authenticated Mail Transfer Protocol

The Simple Mail Transfer Protocol (SMTP) has been an accepted Internet Engineering Task Force (IETF) since 1982 [49]. There are no in-built authentication techniques. Spam has proliferated because senders have been able to obfuscate the sources of the email, and there have been no universally accepted set of policies that classify abuse. AMTP is a new IETF draft standard, published as recently as August 19th, 2003 [72]. It has been designed to replace SMTP with a more secure derivative of itself, with security features that are specifically targeted to reduce the impact of spam email and reduce the cost of running mail-servers. From the IETF draft:

> *AMTP uses TLS and X.509 certificates to authenticate MTAs, and introduces a Mail Policy Code that allows MTAs to advertise policies. A server can decide whether to accept a particular message based on its own policies. Using the authentication capability, a server can manage traffic at any stage in the transaction by denying access to clients that have a history of abusive behaviour.*

At present, there are no reference implementations of AMTP available for evaluation. Reference libraries in Perl and C are being written, as well as several implementations. These will be made available by AMTP's author, Bill Weinman, via the AMTP home-page [71].

2.2.4 Hybrid Techniques

Hybrid techniques are those that do not easily fit into the other categories discussed above. These techniques are typified by having some financial component. This means that not only must they have the technological development to be secure, sophisticated and scalable, but they must be strictly regulated. Because no legislation exists that can perform this regulation (especially not in an international context), these techniques are strictly voluntary. In other words, these techniques must be opted into by mass-marketers. Therefore, they are a mechanism by which legitimate mass-emailers can distinguish themselves from spammers, but do not usually contribute to a reduction in the volume of spam being sent.

2.2.4.1 Certification

The most prevalent technique is certification, or accreditation. This technique is primarily done using bonds or licensed headers. In both cases, the reputation of the certificating company or authority is paramount. If the integrity of the certification body is compromised in any fashion, then the certificate or license loses its value. Additionally, it is difficult for newer certification bodies to create and maintain a good reputation for themselves. This means that there are relatively few players in the business.

With bonded sending, senders take out a bond with a trusted third party. If a recipient or ISP complains to the third party that holds the bond and the complaint is upheld, then a penalty is deducted from the bond. Ironport's Bonder Sender [29] is the best well-known example of this approach.

In the second kind of certification, a set of headers is licensed so that emails containing these headers can be verified as being legitimate by recipients. If the headers appear in emails that originate from a sender who has not explicitly licensed their use, then it equates to copyright violation and the sender of the email is liable to legal action. This approach has been described

```
X-Mailer: ListManager Web Interface
Date: Mon, 28 Oct 2002 13:42:31 -0600
Subject: [Lockergnome Bits & Bytes] Hussein Hacked
To: <emailaddress>
From: Lockergnome Bits & Bytes <subscriptions@lockergnome.com>
List-Unsubscribe: <emailaddress>
List-Subscribe: <subscribe-lgbnb@sprocket.lockergnome.com>
List-Owner: <emailaddress>
X-URL: <http://www.lockergnome.com/>
Reply-To: <emailaddress>
Sender: <emailaddress>
X-Habeas-SWE-1: winter into spring
X-Habeas-SWE-2: brightly anticipated
X-Habeas-SWE-3: like Habeas SWE (tm)
X-Habeas-SWE-4: Copyright 2002 Habeas (tm)
X-Habeas-SWE-5: Sender Warranted Email (SWE) (tm). The sender of this
X-Habeas-SWE-6: email in exchange for a license for this Habeas
X-Habeas-SWE-7: warrant mark warrants that this is a Habeas Compliant
X-Habeas-SWE-8: Message (HCM) and not spam. Please report use of this
X-Habeas-SWE-9: mark in spam to <http://www.habeas.com/report/>
```

Table 2.4: *Example of Habeas Sender Warranted Email licensed headers.*

as "ugly, but effective" [33]. Habeas' Sender Warranted Email [25] is the leader in this type of certification and an example of the headers used is given in Table 2.4.

2.2.4.2 Escrow-Based Systems

Escrow-based systems are those where money is set aside for each email sent, and is held subject to acceptance by the intended recipient. The idea is similar to the one presented in [52]. A nominal fee is set aside for each email that is sent (say 10c). This money is held in escrow by the system until the email is received by the recipient. Based on whether or not the recipient deems the email to be spam or not, he/she can refund some, part, or all of the escrow amount. The money that is not refunded goes into the system to fund infrastructural costs and disputes.

This type of system is an interesting idea, but does not scale well. For ordinary users, an escrow amount of $10 would mean they could have up to 100 emails pending delivery without having to put more money into escrow. For the average user this is acceptable enough, based on their usage of email. For mass-marketers or mailing-listers with thousands of subscribers, it would mean putting a large amount of money in to escrow(say $5,000 per edition for a readership of 50,000), which is economically infeasible for small open-source and non-profit organisations.

2.3 Technology

Before launching into a discussion of our system, it is necessary to first describe the electronic mail protocol that will be used, including how the mail-servers interact and how email is sent and routed towards its destination. Additionally, the network topologies that will be considered are described and characterised.

2.3.1 Electronic Mail Protocols

It is an irony of technological history that the *de facto* protocol for email messaging is not the International Standards Organisation (ISO) defined one. In fact, X.400 - the ISO prescribed standard - is very rarely seen in use today. The primary reason for this is the fact that the ISO took a very long time to actually finalise the specification. In the meantime, usage of the electronic messaging increased exponentially. This lead to the interim solution, SMTP, being adopted and accepted as the protocol used for electronic messaging between interconnected systems on the Internet. Defined in a series of Requests For Comments (RFC) by various vendors, SMTP was intended to be an interim solution that could be extended to comply with the ISO X.400 standard once it was finalised. It was originally introduced in RFC 281, which was superseded by RFC 2821 [30]. RFC 2821 was extended by the following RFCs:

- **RFC 2822.** RFC 2822 [48] describes the basic message format that should be used when creating Internet email messages.
- **RFC 2920.** RFC 2920 [41] defines a SMTP extension to improve the SMTP performance by bundling multiple SMTP commands within one TCP send operation.
- **RFC 3030.** RFC 3030 [23] defines two extensions to the SMTP protocol for the transfer of large and binary MIME messages.

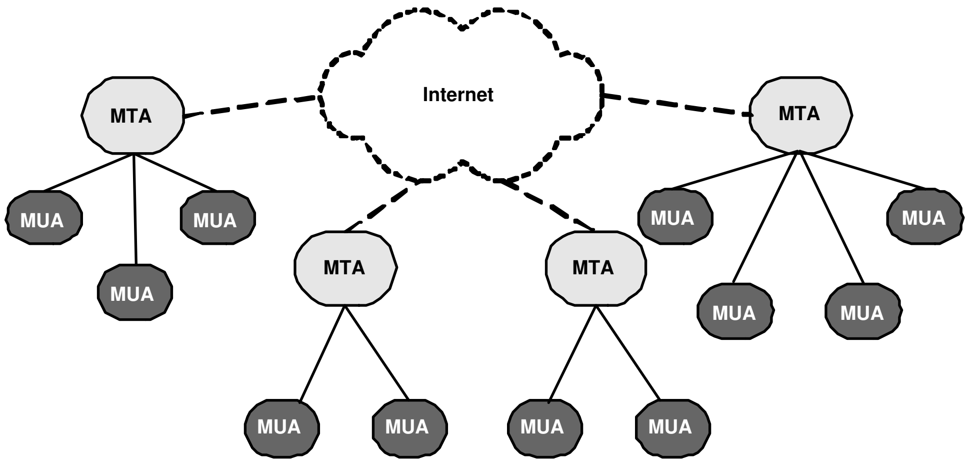


Figure 2.1: *Conceptualisation of Internet Email.*

- **RFC 2487.** RFC 2487 [46] defines a SMTP extension for transport-layer security during SMTP sessions.

Because SMTP is the prevalent protocol for email in use today, it will be implicitly considered the protocol for use in this work.

2.3.1.1 Terminology

The Internet email system can be visualised as the system presented in Figure 2.1.

MTAs - Mail Transfer Agents

MTAs are programs that reside on the mail-server of ISPs. The primary functionality of an MTA is to route email to its intended recipient. They are normally comprised of two components - a server for sending outgoing email (conforming to SMTP) and a server for receiving incoming email (conforming to either IMAP or POP3).

MUAs - Mail User Agents

MUAs are software that enable a user to create, send and receive email. Historically, an MUA (such as Mutt, MS Outlook, Eudora and Pine) would reside on a user's desktop. However, as

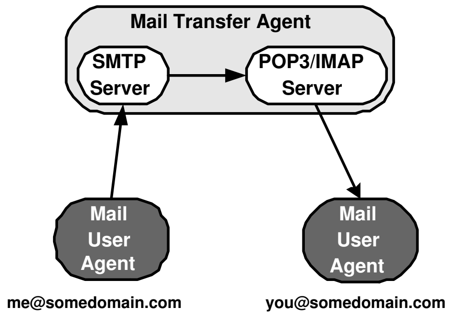


Figure 2.2: *Email Client/Server Interaction*

mobility is now a requirement for most users, there are MUAs for PDAs and mobile devices and many ISPs offer a web mail service through which users can access their email without having to install any software.

2.3.1.2 Email Client/Server Interaction

Whenever a user sends an email, it is first sent to the SMTP server on the MTA of the ISP that is providing the user with the email service. If the recipient of the email is in the same domain, for example if "me@somedomain.com" sends an email to "you@somedomain.com", then the SMTP server hands the email over to the POP3 or IMAP server for delivery, as per Figure 2.2. This is a classic example of client/server interaction.

POP3

The Post Office Protocol 3 is an offline mailer. It operates by storing all emails for a user in a hold-back queue until they connect to the server. Once the user connects, the emails are downloaded to their MUA and removed from the POP3 server. This approach is sufficient for users that access email exclusively from one machine. If a user accesses their email from another machine they will only see those messages that are new since the last time they accessed their inbox. This is because the messages were deleted from the mail-server as soon as they are downloaded. For users who do not use the same machine to access their email, this is quite inconvenient as their inbox becomes fragmented and spread across a number of machines.

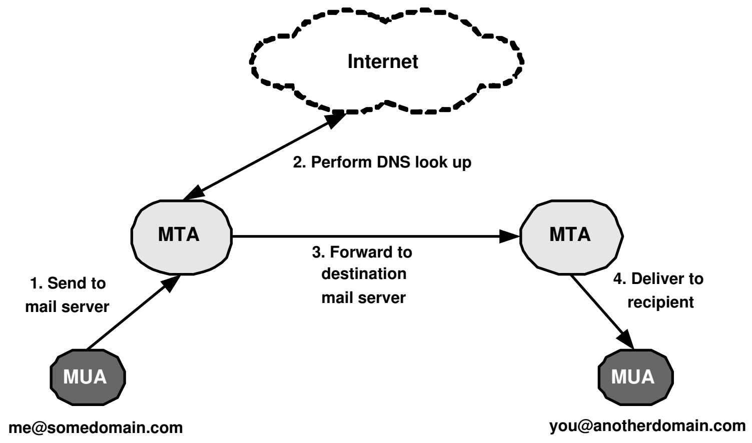


Figure 2.3: *Email Server/Server Interaction.*

IMAP

The Internet Message Access Protocol is an online mailer. Messages are not deleted from the server when they are accessed. This is much more useful to users that check email from different machines at different times as they have a complete and consistent view of their mailboxes available wherever they check their email. This is due to the fact that both old and new messages are stored on the server. This provides the ability to use “data-less” client machines and have platform-independent access to email.

2.3.1.3 Email Server/Server Interaction

If an email is sent from a user in one domain to a user in another domain, the sending SMTP server must locate the IP address of the server for the recipient email address. This is done by performing a Domain Name Server (DNS) look up. Consider Figure 2.3 . The user “me@somedomain.com” wishes to send an email to “you@anotherdomain.com”. The somedomain SMTP server performs a DNS lookup for the IP address of the SMTP server of anotherdomain. The somedomain SMTP server then connects to the anotherdomain SMTP server using the same protocol that a client of the anotherdomain SMTP server would. This is an example of centralised peer-to-peer interaction, as described in Section 2.3.2.2. An email can be forwarded through a number of servers before it reaches its destination server, as in Figure 2.4. These

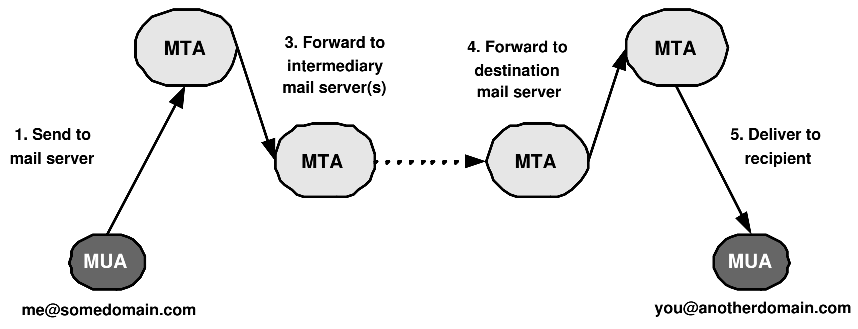


Figure 2.4: *Email forwarded through intermediary mail-servers.*

servers are listed in the "Received" header, which is useful in spam filtering, as discussed in Section 3.3.

2.3.2 Network Topologies

There are a number of different network topologies, but the two of interest to this work are the client/server and peer-to-peer topologies.

2.3.2.1 Client/Server

Client/server is the oldest and best understood architecture. There is a server to which multiple clients connect. All communication - including that between clients - is channelled through the server. The server is assumed to be always available, with the clients connecting transiently. This architecture is best exemplified by the classic web architecture. There are a number of problems associated with the client/server architecture, including scalability, single point of failure, single point of attack and availability.

2.3.2.2 Peer-To-Peer

Peer-to-peer first came to public knowledge via the mp3 music sharing application Napster [42]. It is characterised by direct communication between peers in the network. The advantages of peer-to-peer are that it scales particularly well, there is no centralised point of failure and anonymity is possible. Another very salient feature of peer-to-peer networks is the transience of connections. There are also a few inherent problems with peer-to-peer applications, including maintaining privacy (which is different to anonymity), network partitioning, free-riding, peer-overloading and trust management. One of the driving ideas behind the peer-to-peer architecture was the desire to "harness the dark matter of the Internet". This dark matter is the unused cycles and storage space available on the machines that are used to access the Internet. A few different definitions of peer-to-peer as it pertains to the Internet exist, yielding subtly different forms of peer-to-peer topologies.

Not P2P - Ultra Client/Server

C. Shirky [7] defines peer-to-peer as:

> *...a class of applications that takes advantage of resources - storage, cycles, content, human presence - available at the edges of the Internet.*

This definition catches the essence of peer-to-peer without defining it correctly. This is because it allows "ultra" client/server topologies, such as Compute Against Cancer [12] and SETI@home [55]. These methods use client machines to perform computation on easily parallelisable problems, but have no direct communication between clients.

Centralised P2P

Peer-to-peer is characterised by sharing of computer resources and services by direct exchange. However, there can still be some centralisation of the system, just not for information exchange. This is exemplified by Instant Messaging techniques and the Napster paradigm, whereby a peer looks up other peers in a centralised repository, but connects directly to the peers for communication and information exchange. This has the requirement that the centralised component must always be available to perform look ups on behalf of their peers, though the peers can be transiently connected. Figure 2.5 illustrates this, where peers p1 to p3 look up information in s1 (dotted arrows), but actually perform data transfer directly (solid arrows).

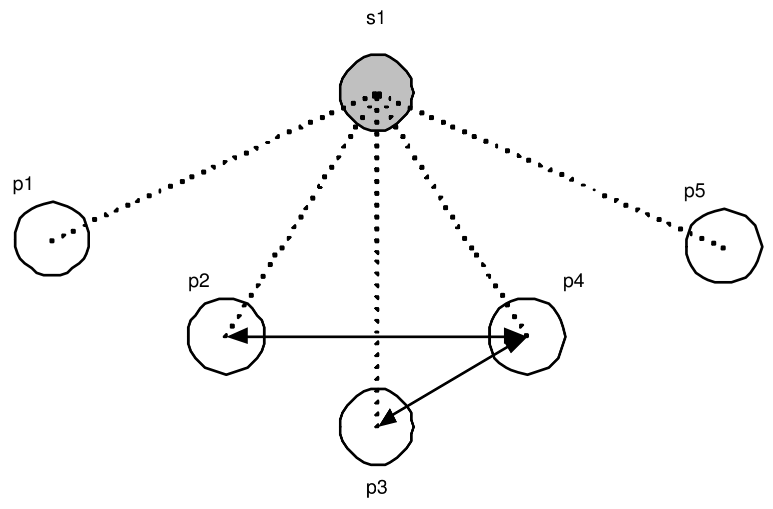


Figure 2.5: *Sample Centralised Peer-To-Peer Topology.*

Decentralised P2P

By extending the definition of peer-to-peer as the sharing of computer resources and service by direct exchange to include the stipulation that there be no centralisation, we get what is called decentralised peer-to-peer. These networks are the most transient, with no requirement for any specific peer to be connected at any given time. Examples of this are Freenet [28] and Bluetooth [4].

Super-Peer P2P

In some decentralised peer-to-peer networks, powerful peers can be designated as super-peers in the network. A super-peer will typically facilitate the ordinary peers that neighbour it in the network by performing services on their behalf. An example of this is in the current generation of Internet file-sharing applications where super-peers in the network store meta-data on behalf of their neighbours in order to speed up searching. Figure 2.6 shows a super-peer based topology, where meta-data lookups are depicted by dotted lines and information transfer by solid lines. Information transfers go directly from one peer (normal or super-peer) to another. Table 2.5 summarises the transfer types possible in the network.

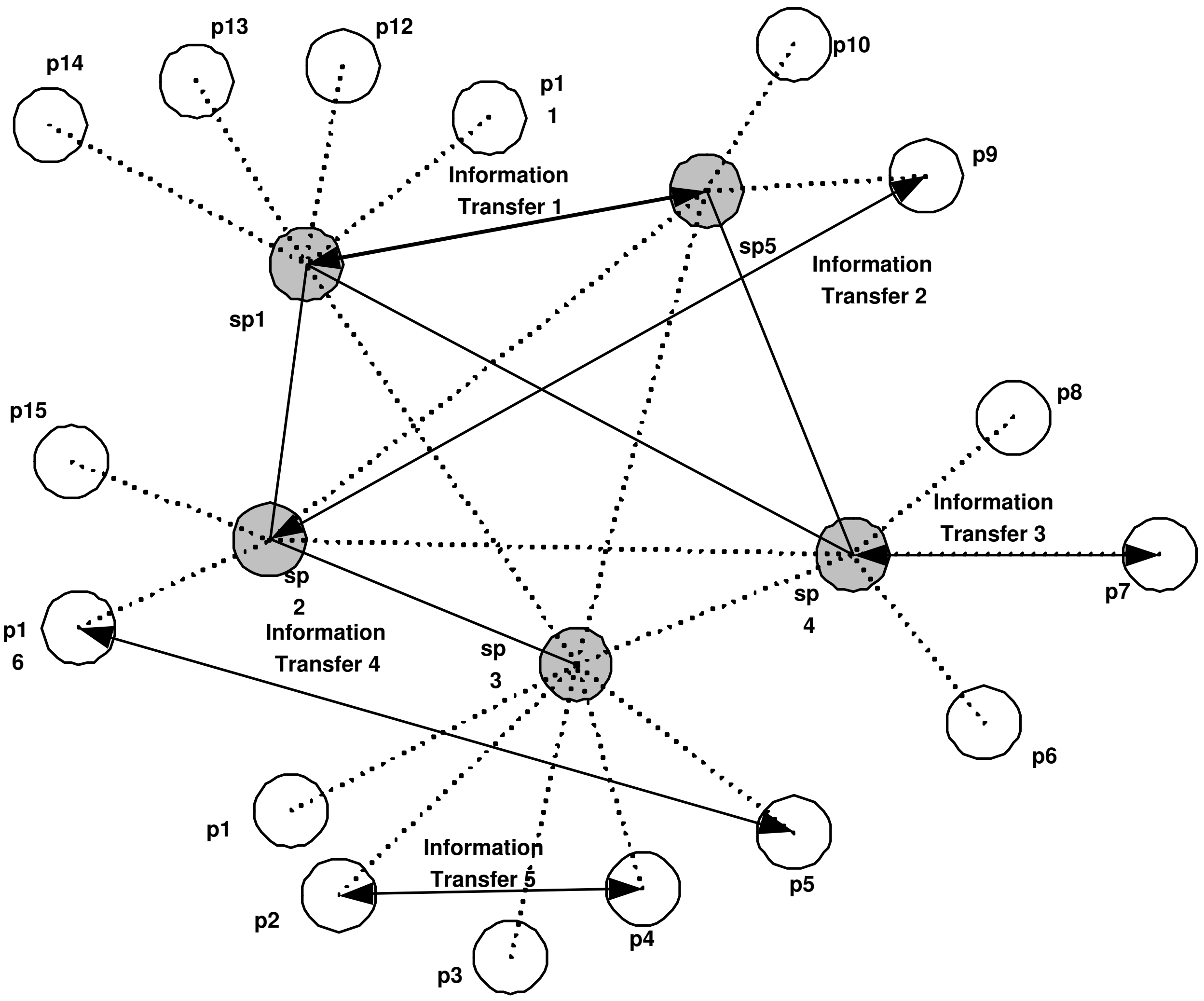


Figure 2.6: *Sample Super-Peer Based Topology.*

Transfer #	Description
1	Super-peer to super-peer
2	Super-peer to peer in different super-peer section of network.
3	Super-peer to peer in same super-peer subsection of network.
4	Peer-to-peer in different super-peer subsection of network.
5	Peer-to-peer in same super-peer subsection of network.

Table 2.5: *Summary of Transfer Types in Super-Peer Based Network.*

2.3.2.3 The Difference Between Centralised, Distributed And Decentralised Systems

For the purposes of this work, there is a very subtle, but important distinction to make between centralised, distributed and decentralised systems as they pertain to spam filtering. Centralised systems are those where there is one copy of the information which can reside either locally or remotely. Distributed systems are those where there are many copies of the same body of information is replicated through the system. Again, the copies of this information may be remote or local. A decentralised system differs to a distributed system because although there are many collections of data in the system, these stores of data are not the same. The stores of information are (potentially overlapping) disjoint subsets of the total data in the system. These systems are characterised by the fact that no one store has a complete view of the total data or even amount of data in the system (although this is not a requirement of the system).

Chapter 3

Combating Spam - The Technological Approaches

There are a panoply of approaches to combating spam. This chapter describes the main techniques, as well as giving a brief overview of how spammers attempt to circumvent them.

3.1 Beat The Filters - A Bluffer's Guide To Spamming

In order to better understand the various technological mechanisms for spam filtering, it is useful to know how spammers operate. It is beyond the scope of this work to completely analyse the state of the art in spamming, but a cross-section of the main spamming techniques and attitudes is presented in this section.

3.1.1 Get As Many Email Addresses As Possible

Even though most spam has a very low "hit-rate" of approximately 0.1%, it is still profitable [3]. This is because spam reaches such a high number of recipients. The cost to the sender associated with sending an email is not proportional to the number of recipients. Therefore, the more people who receive the email, the greater the return to the spammer. Spammers obtain email addresses by:

- Harvesting email addresses from web pages using web crawlers/spiders. There are a great many email addresses posted on web sites, intended to enable people to contact others. By harvesting these email addresses a spammer can get access to many live email accounts.

- Trading email addresses with other spammers.

- Using dictionary attacks on well known ISPs. A dictionary attack is where all possible permutations of words and numerals are tried. The spammer is unconcerned about emails that bounce, because the return address is usually to an inbox that does not exist.

- Including a fake "unsubscribe here" link. Users who follow these links have obviously opened the email and their email addresses are "live". By keeping a list of these, a spammer can get a list of people who open spam and will spam these addresses even more.

3.1.2 Trick People Into Opening The Spam

Obviously, if people do not read the spam, there is no chance that the spammer will accrue any profit. Therefore, spammers must lure recipients into opening the spam. This can be done by:

- Creating a subject line that has nothing to do with the spam. If a recipient receives an email with "how to enlarge your penis" in the subject line, they are unlikely to open it. Additionally, rule based filters will easily intercept the email and recognise it as spam. By including the text "Re:" or "Fwd:", spammers can trick recipients into opening the email by pretending the email is a response to one the recipient has sent.

- Spoofing the sender's address in the spam. A recipient is very likely to open an email if it appears to come from someone the recipient knows. Therefore, if the sender's address is spoofed to appear like it comes from someone at the same domain, or someone who appears on a mailing list the recipient has subscribed to, the chances of the email being opened are greatly increased. Email addresses that appear on the same page as the recipient's are a good source of possible spoofing addresses because the page could be a directory listing of the person's organisation, or a group they are affiliated to.

3.1.3 Hide The Origin Of The Email

Blacklists are a very effective way of combating spam. A blacklist is a list of IP addresses or email addresses that are known for sending spam. Any email originating from these addresses is known to be spam and will not be delivered. So if you wish to be a successful spammer you must stay off blacklists by hiding the origin of your spam. Ways of doing this are:

- Faking headers in the email. The return-path header of an email contains a list of servers that have routed the email. By including extra mail-servers at the beginning of this list, the origin of the spam can be obfuscated.
- Relaying spam through open relay servers. Open relay servers are mail-servers that will send an email on behalf of someone not in the domain of the mail-server. These can be exploited by spammers because anyone can connect on port 25 and send email. Open relay servers often get blacklisted because spam is sent from them, leaving the spammer and the true origin of the spam unidentified.
- Signing up to an ISP and just subscribing to a new one every time the spammer is banned for abusing it. This is often done as each time spammers join a new ISP they get a window in which to send spam before they are identified as spammers.

3.1.4 Obfuscate The Content Of The Email

Many spam-filters operate by analysing the content of an email. These filters examine the lexical content of an email to determine if the email is spam. These techniques can be circumvented by:

- Including words and phrases that have no bearing to the spam. These can be hidden in white text on a white background or in Java-script and HTML comments, for example.
- Misspelling common spam words. Examples of this are substituting a 1 for the "l" in Adult (this appears as Adu1t, which is carries the same semantic content to the human eye), or spelling "porn" as "p0rn" or even "pr0n".

- Using tables and images to break up the delivery of the content of the spam, making it harder for filters to catch while remaining readable to the human eye.
- Including randomisations in text or comments in order to beat signature-based filters.

3.2 Email Address Obfuscation

It is an accepted rule of thumb that you cannot keep your email address unknown to spammers. However, it is possible to obfuscate your email address so that it is not easily harvestable to spammers.

3.2.1 Spam Bait

Spam bait aims to make harvesters useless by hiding real email addresses behind a deluge of fake addresses. There are quite a number of sites that host spam bait pages. The idea is quite simple. Dynamic web pages are generated and populated by random strings of the form <randomname>@<randomdomain>.<randomtopleveldomaintype>. The plan is to increase the noise-to-signal ratio of useless addresses to real addresses. Unfortunately, many of these pages negate themselves by including an explanation of what the page intends to do. Spammers' harvesters simply have to be tuned to ignore email addresses stripped from pages that contain phrases like "spam bait" or "spam bot".

3.2.2 Mail-to Randomisation

Spammers attempt to beat filters by changing spam phrases so that they are difficult for the filter to spot, but still sensible to a human reader. This technique can be applied in reverse to help protect email addresses from harvesters. One technique is to spell out punctuation when posting email addresses, such as <username> at <domainname> dot <topleveldomaintype>. Another technique is to post the HTML entity reference value of the characters that form the email address. This would manifest itself as the name "John" being posted as "&74;&111;&104;&110;". This can foil the vast majority of harvesters at present, but it is only a matter of time before such a simplistic approach will be beaten by simple regular expression matching.

Another form of mail-to randomisation is to include some simple riddle to decipher the mail-to address. Two examples of how "username@domain.tld" could be rewritten are "reversethis: dlt dot niamod at emanresu" or "reversethis: {dlt}.{niamod}@{emanresu}". These email addresses are decipherable to humans with a little bit of effort. Parsing these addresses is a trivial task for programs, but only once the decryption has be identified and coded by the spammer who writes the harvester.

3.2.3 Mail-to Riddles

Mail-to riddles are another form of email address obfuscation. It is an idea that has been successfully applied by large ISPs to stop bots from automatically creating email accounts. A simple riddle is included in the subject line of any email that is created by a mail-to link, such as in the following sample link:

```
<a href=''mailto:username@domainname.tld?Subject=Unit of currency in US?''>
```

The answer to the riddle must be substituted by a person wishing to send an email to the email address. It is possible to program a harvester to parse the riddle and substitute the word "dollar" for "Unit of currency in US?", but because there are an almost infinite number of simple riddles it would take a long time to write a harvester savvy enough to recognise and solve all of them.

A stronger form of this is to include an image with a word printed in it on the web page where the address appears. The word must appear in the subject line of any email sent to the address, or it will be rejected. It is a trivial task for a user to enter, for example, the second word from the left of the image with the green background on a web page. It is virtually impossible to write a bot intelligent enough to be able to firstly follow the instructions and secondly pick out text from an image. More importantly, it is a computationally intensive process to extract and parse text from images and therefore spammers would have to perform a lot of work in order to get a single email address.

3.2.4 Strong Usernames

Spammers can attack major ISPs' email addresses by using what is known as a dictionary attack. Permutations of words and numbers are used to create email addresses that may or may not exist on ISPs. As an example, it is a reasonably safe bet that the usernames "john", "john01" and "john02" are taken at Yahoo, AOL and Hotmail. By choosing strongly alpha-numeric addresses, most dictionary attacks can be circumvented. By comparison to the previous example, it is unlikely that the username "f6th35da$dy" is taken at any ISP. Thus choosing long, strongly alpha-numeric usernames means that the spammers will have to try many more permutations and combinations to get a valid email address.

3.3 Characteristics-Based Filtering

We define characteristics-based filtering to be filtering that is concerned with the content of the email that is not human-readable. This means that characteristics-based filtering is primarily concerned with the headers of the email. Very often, characteristics-based filtering is not explicitly mentioned in the literature and is assumed to be a subset of content-based filtering. However, it is sufficiently large an area that it warrants discussion on its own. Although characteristics-based filtering can be a very good first approximation to a complete spam filter, it is just that - a first approximation. A characteristics-based filter is insufficient for the task of spam filtering and must be used as part of a more complete spam filtering system.

3.3.1 Whitelisting

Whitelisting is the process whereby emails are allowed through the filter on the basis of the originating email address or IP address. The filter (which can reside on the MTA or the MUA) maintains a list of verified and accepted email addresses/IP addresses. Addresses that appear on this list are automatically permitted through to the user's inbox without filtering. The policy can be exclusive or permissive.

In the exclusive policy, only emails from addresses that appear on the list will be allowed through the filter - all other emails are blocked. For example, only people to whom we have sent an email in the past will be allowed send an email to us. Exclusive whitelisting would be suitable for a child's email account so that parents can have direct control over what the child receives, so that no inappropriate unsolicited email reaches them.

In the permissive policy, emails from whitelisted addresses will be permitted through without filtering, whereas emails from addresses will be subject to filtering in the normal fashion, using one or more of the other spam filtering techniques described. This is suitable for an enterprise-sized LAN, where all email originating from inside the LAN is whitelisted and not filtered, for example. Any other email is filtered. This ensures that legitimate unsolicited email (such as client queries or job applications) can still be received.

3.3.2 Blacklisting

Blacklisting can be seen as the opposite of permissive whitelisting. A list of IP/email addresses is maintained (either on the MUA, MTA, or a remote server). Any email that originates from these addresses is identified as spam by default and not delivered to the recipient. Any email from a sender who does not appear on the blacklist is filtered in another fashion. Blacklists are effective for blocking spammers who operate in a jurisdiction with no anti-spam laws. This is because the spammers do not tend to change ISPs at such an extraordinary rate (the spam gangs in Boca, Florida are an example of this). However, blacklisting on its own is insufficient as an anti-spam mechanism, because of the volume of spam that is relayed by open-relay servers or has forged return addresses.

3.3.3 Header Analysis

Spammers forge other headers in order to circumvent spam filters. These headers can be the sender IP/email addresses, which can circumvent blacklists. Very rarely can forged sender addresses circumvent exclusive whitelists, unless the sender is lucky enough to choose an address that appears on the recipient's whitelist. At any rate, it is very unlikely that a a given address will appear on a significant number of whitelists.

Another header that is often forged is the time-stamp of when the email was sent. These can be forged to be ahead of the current time so that if it gets through a filter, then the spam will appear at the top of the recipient's unread email list. Conversely, the time-stamp can be forged to appear a long time in the past in the hope that a filter will ignore old emails, but this has been recognised and will very rarely succeed nowadays. The originating time-zone can also be faked. This is done in the hope that a filter will apply a different set of filtering rules/techniques based on where the email was sent from. By including a non-existent time-zone, spammers hope that the email will be not be filtered by default.

The "To" header is also often forged. This is done to hide the fact that the email has been sent to a large number of people. Most often this is set to read "Undisclosed Recipients", in an attempt to show that it is to a legitimate mailing-list. Forged "Received" headers are included to give a false return path for the email. This is done in an attempt to hide the true origin of the spam.

3.4 Content-Based Filtering

Content-based filtering is filtering an email based on the human-readable content of an email. There are a great many variations on content-based filters. It is also the area of spam filtering that has seen the most research with machine learning techniques. Content-based spam filters are the most high maintenance spam filters for two reasons. The first is that the knowledge base of what features of the content indicate spam must be built-up before the system can be used. The second is that this information base must be kept up to date as the concept of spam drifts and new types of spam as well as spam content emerges.

3.4.1 Rule-Based Filtering

As remarked earlier, it is comparatively trivial for a human to glance at an email to determine if an email is spam or not. Rule-based filters attempt to ascribe "spamminess" metrics to various aspects that can appear in an email. This is done by creating a set of rules for typical spam words and also words that are typically non-spam. Other features are also used, such as the HTML content of the email, the fraction of the email in uppercase and number of exclamation marks, amongst others.

Each rule has a score associated with it, with positive scores for rules that indicate spam and negative scores for those that are indicative of non-spam emails. A sum of the scores of all the rules whose criteria are fulfilled is performed, with the result compared to a fixed threshold. If the score is above the threshold it is identified as spam and dealt with according to the site's spam policy, otherwise it is it is allowed through.

Rule-based filters can be applied at nearly all the stages in sending email. It can reside on the MUA of the user where it is analogous to simple email sorting rules. Alternatively, it can be centralised on the MTA of a particular domain. Additionally, the rules for these filters can be distributed and replicated across MTAs, as per SpamAssassin (See Section 4.3).

3.4.2 Machine Learning Techniques

Humans can classify an email as spam almost instantly and, if given a large corpus of spam, a number of humans will consistently label the same emails as spam. Given that this is the case and that it is the lexical content of the spam that identifies it as such to a human, it is natural to assume that textual analysis of these emails can yield a set of features that can be used to determine the "spamminess" of an email. This is the aim of machine learning techniques as applied to spam filtering. By performing some kind of analysis of spams that have been identified by a human, a set of rules or a mechanism that can distinguish spams from legitimate email can be obtained, and these are used to classify emails in future. There has been a moderate amount of interest in applying machine learning techniques to the realm of spam filtering.

3.4.2.1 Bayesian Learning

Naïve Bayesian approaches have proven very capable in text classification problems [14]. For this reason, it has been the most popular machine learning technique used in spam filtering to date [26,27,36]. Bayes' rule of conditionality is used to determine the probability of a hypothesis, given that some information that bears on the probability is true. More formally, Bayes' rule of conditionality states that if there exists a hypothesis h, and observed data D that bears on h, then

$$P(h|D) = \frac{P(D|h) \times P(h)}{P(D)}$$

A Naïve Bayesian (also known as first-order Bayesian) classifier assumes that the features used in the analysis are independent (i.e. the presence of one feature does not affect the probability of another feature appearing). Despite this and the fact that many spams have common attributes, the Naïve Bayesian classifier has proven effective.

Naïve Bayesian classification can be tuned to have a cost associated with the decisions that have been made. This is of particular interest to spam filtering when it comes to false positives. Androutsopoulos et al [27] showed that a Naïve Bayesian classifier is not suitable where the cost of a false positive is high.

3.4.2.2 Case-Based Reasoning

There are a number of case-based approaches to spam filtering that have been applied, including k-Nearest Neighbours [22, 45] and Decision Trees [73]. In these cases, a set of features that predicate whether an email is or is not spam is extracted from a corpus of training data. These features are organised into a framework.

In decision trees, features are organised into a tree structure, such that by taking a branch at any decision point (determined by a feature at each point), the tree can be traversed from the root to a leaf at which the email is classified as spam or non-spam.

In k-NN [63], the feature vector for each email is classified as the average classification of the nearest k neighbours in the feature space. The dimensionality of the feature space depends on the number of features that are deemed predictive of spam or non-spam. In each case, these have been shown to outperform Naïve Bayesian spam filters. As noted in [45],

> *This is because spam is a disjoint concept; spam about software for pirating DVDs has little in common with spam selling pornography. Case-Based classification works well for disjoint concepts whereas Naïve Bayes tries to learn a unified concept description.*

It is also worth noting that this is analogous to storing centralised data remotely as opposed to keeping it decentralised to local machines in collaborative filters, as discussed in Section 3.5.3

3.5 Collaborative Filtering

Collaborative filters are concerned purely with the fact that an email is spam, regardless of what the actual content of the spam is. This is potentially the strongest form of spam filtering, as the content is not relevant. However, it will be shown that to be truly effective, the collaborative filters need to be use techniques that are content-aware.

Collaborative filtering is based on a very simple concept. There are a group of entities (individuals or organisations) that do not wish to receive spam email. If one of these entities receives an email that they determine to be spam, then they inform the others in the group that the email is spam. Typically this is done by computing a signature on the content of the email and sharing it with the other members of the group. In the future, if any member of the group receives an email that has a signature that matches the known spam, then the new email can be identified as spam. This is very useful because an early receiver of spam can notify the others in the group when new spam arrives and the other members of the group need never receive it.

3.5.1 Fuzzy Hashes

The advantage of collaborative filters over content-based filters is entirely dependent on the method used to create the signature of a spam. Historically, hashing algorithms come from the cryptographic community and are used to verify message integrity. This means that if the content of the body is changed slightly then the resulting signature will change dramatically (c.f. SHA1, MD5 hashing algorithms). If this type of hash were used by collaborative filters, then slight randomisations in the content of the email will render the collaborative solution useless.

Fuzzy hashes are hashing algorithms that are robust to slight randomisations in the message content. More generally, fuzzy hashes are hashes that yield little or no change in the signature if there are small randomisations in the input text. When applied to spam filtering, it is desirable that little customisations such as “Dear John” or “Dear Mary” will not change the output hash if the remainder of the input text is the same. Fuzzy hashes must be content-aware in order to be truly robust. This is clear because fuzzy hashes must be able to ignore randomisations in the irrelevant content of the email whilst not ignoring changes in pertinent information. There is still a lot of work to be done in this area.

3.5.2 Clearing Houses

Clearing houses are servers that store signatures of emails that have been classified as spam by members of the group. Every time a new email is received by a group member, a signature is computed on it and sent off to the clearing house. If the signature matches one currently stored in the clearing house, then the server informs the group member that it is a known spam email. If a group member receives a new spam that is not known by the server, they compute a signature on the spam and send it to the server to add to the corpus of known spam signatures.

3.5.3 Centralised Versus Distributed Versus Decentralised

Most of the collaborative spam filters are centralised or only moderately distributed. This is because either they are proprietary and they are centralised in order to reduce server costs and to protect intellectual property, or because they lack the sophistication to be truly decentralised. Centralisation of the signature corpus leads to performance degradation of the system, because of the implicit assumption that all spam signatures are of interest to all users of the system. This is clearly not the case. By trying to hold a complete encyclopedia of spam, the system is more prone to both false positives and false negatives.

In the current state-of-the-art systems, distribution of spam signatures in collaborative anti-spam systems is done merely from an efficiency point of view. This means that although the system and the signatures are distributed, there is no personalisation or optimisation based on which signatures are relevant to which users. A truly decentralised collaborative system (as per Section 2.3.2.3) is one where the signatures themselves are propagated through the system to the users which they will most benefit.

Users receive spam because they are on spammers' lists. Therefore, the only signatures that are interesting to a given user are signatures that match spam that has been sent to a spam-list that the user is on. A much more useful collaborative system would be one that distributes the signatures to the users who are most likely to receive spam that matches them. In other words, a truly decentralised system that clusters users that receive similar spam would be much more personalised to the user. It would also have the benefit of reducing the storage costs of irrelevant signatures and the increased accuracy due to the increased relevance of the stored signatures.

Chapter 4

State Of The Art

There are many conceptual delineations between the various spam-filtering techniques, but in practice most of the solutions available at present draw from a number of these techniques. Indeed, the most effective spam filters are those that combine a range of these approaches described in Chapter 3. This chapter details the state of the art in distributed, collaborative spam-filtering.

4.1 Vipul's Razor

In the context of the categorisations made in the previous chapter, Vipul's Razor is a moderately distributed collaborative anti-spam system. This is because although the signatures are distributed, they are merely copies of the same information, replicated in order to enable scalability. This is also a part of the reason for Razor's relatively poor performance (approximately 60 % to 90% of spams caught). As stated in Section 3.5.3, centralised and moderately distributed collaborative systems perform sub-optimally due to the vast number of signatures that are irrelevant to any given user.

Vipul's Razor [69] describes itself as:

> *a distributed, collaborative, spam detection and filtering network. Through user contribution, Razor establishes a distributed and constantly updating catalogue of spam*

in propagation that is consulted by email clients to filter out known spam. Detection is done with statistical and randomized signatures that efficiently spot mutating spam content. User input is validated though reputation assignments based on consensus on report and revoke assertions which in turn is used for computing confidence values associated with individual signatures.

Currently, Vipul's Razor is in its second version, and Razor v2 is almost a complete rewrite of Razor v1. From the v2 documentation [70], the following list summarises the most significant new features:

- **New Protocol.** The Razor v2 protocol has been completely redesigned. The new protocol is based on exchange of *Structured Information Strings*, which are similar to URIs and can be parsed with URI decoding libraries. The Razor v2 protocol supports *Pipelining*, which means that Razor Agents can keep a connection open to a server to eliminate the latency introduced by TCP 3-way handshake and 4-way breakdown for every connection. The new protocol semantics allow seamless introduction of new signature schemes.

- **Nilsimsa Signatures.** Nilsimsa is a *fuzzy signature* algorithm based on statistical models of n-gram occurrence in a piece of text, which disregards small changes (mutations) in text that are statistically irrelevant. These signatures can be compared to determine the similarity (between 0 - 100%) of source texts. Razor v2 includes support for Nilsimsa signatures.

- **Ephemeral Signatures.** Ephemeral Signatures are short-lived signatures based on collaboratively computed random numbers. Ephemeral Signatures select a section of text from the spam message based on a random number that changes every so often. This makes the hashing scheme a moving target, and spammers cannot exploit it because they do not know which part of the message will be hashed after the random number rollover.

- **Preprocessors.** Razor v2 supports several preprocessors. Preprocessors alter the text of a spam before a hash is computed. This version includes preprocessors to decode Base64 encoded messages, decode QP encoded messages and convert HTML to plaintext. Spammers employ several techniques that hide mutations in various encoding. Preprocessors defeat such techniques by hashing the content that a recipient actually sees in his/her Mail User Agent.

- **Multiple Filtration Engines.** Razor v2 supports multiple engines. An engine is a logical unit that encapsulates a particular type of filtration service. Razor v2 currently supports four engines - VR1 which is equivalent to Razor v1, VR2 that is based on SHA1 signatures of bodytext, VR3 that is based on Nilsimsa signatures, and VR4 based on Ephemeral hashes. New engines can be seamlessly plugged into the service as and when required.

- **Complete Backward Compatibility with Razor v1.** The VR1 engine is functionally equivalent to the Razor v1 service and uses the same database. This means users who transition from v1 to v2 will still get the benefit of several million signatures known to the v1 service.

- **Base64 signature encoding.** Signatures are now encoded as base 64 numbers instead of base 16 (hex), reducing traffic that goes over the wire by 33%.

- **Truth Evaluation System (TeS).** Razor v2 has a transparent, back-end component known as TeS. TeS is a combination of a reputation system and pattern recognition heuristics that assigns trust to reporters and confidence values (between 0-100) to every signature. Users can set an acceptable confidence level in their Razor configuration. The server also publishes a recommended confidence level. TeS has been designed to eliminate false positives of legitimate bulk email that were occasionally generated by bad reports in Razor v1.

- **Submission of entire spam messages.** Razor v2 accepts the entire body text of spam messages not previously known to the system. This lets Razor v2 compute new Ephemeral Signatures every n hours as well as seed the database whenever a new signature scheme and/or preprocessor is introduced. It should be noted that Razor v2 *does not* accept contents of legitimate email during a check dialogue. Only signatures are sent when checking email. This ensures that users' privacy is respected.

- **Revocation.** Razor v2 allows users to revoke messages that they do not consider to be spam. Revocation input is fed into TeS, that adjusts the confidence value of a signature or remove it from the database as necessary. Revocation is done through a tool called razor-revoke, which is a part of the new Razor distribution.

- **Reporter Registration.** Razor v2 requires reporters to be registered. This lets reporters build a reputation over time, so their reports and revocations are weighed according to their reputation value. Report requires users to authenticate which is done using a CRAM-SHA1 authentication scheme.

- **Content classes.** Razor v2 introduces the concept of content classes. A content class is a set of messages that represents variations on the same content. As new reports come in, Nomination servers associate them to an existing content class, if a (close) match is found. Additionally, Razor v2 treats each MIME attachment as a separate content class, so spammers' MIME attachments can be individually tracked (which is very useful in case of viruses).

4.2 MAPS Realtime Blackhole List

The Mail Abuse Prevention Realtime Blackhole List [39] maintains an up-to-the-minute *"list of networks which are known to be friendly, or at least neutral , to spammers who use these networks either to originate or propagate spam"*. In the context of the definitions in Chapter 3, the MAPS RBL is a moderately distributed blacklist. It is amongst the most well-known blacklist site on the Internet and has been incorporated into a number of spam filtering solutions. MAPS describes the rationale behind the RBL as:

> *The MAPS RBL is a system for creating intentional network outages ("blackholes") for the purpose of limiting the transport of known-to-be-unwanted mass e-mail. The MAPS RBL is a subscription system, such that no one is ever denied connectivity to a non-RBL-subscriber. If your network seems to have been blackholed by us, be aware that the places you cannot reach have* ***deliberately chosen*** *not to exchange traffic with you. We are not the network's police force, but rather, a method to identify likely spam origin.*

There are two ways to use the MAPS RBL - transfer mode and inquiry mode.

In inquiry mode, the IPv4 dotted quad notation (e.g. 127.0.0.1) of some host or mail-relay is known, and MAPS is asked whether or not this host is listed in the RBL. MAPS will reply whether or not the IP address is known to be a spammer/spam haven and the inquirer then can take whatever action is consistent with the site's spam policy (See Section 2.1.3.6 for more on spam policies).

In transfer mode, the entire MAPS RBL is replicated at a host owned by the user using a network protocol such as DNS or BGP that allows the copy to be updated instantly whenever changes occur. This is very important so that sites that are blacklisted in error are quickly removed and do not suffer long-lasting effects of being temporarily blacklisted. Because of the legality involved with denying access to listed parties, an indemnification agreement against damages claimed by these parties must be signed before a participant can work in transfer mode.

4.3 SpamAssassin

Currently, SpamAssassin [31] is probably the leading spam filter in the public sphere. This is due to three primary reasons: SpamAssassin uses a number of different techniques to combat spam; SpamAssassin has a well-defined, extensible Application Programming Interface (API); and SpamAssassin is open-source and transparent. SpamAssassin can distinguish accurately between spam and non-spam with an accuracy of between 95% and 99, according to [57].

Because of the wide range of tactics used to identify spam, SpamAssassin is not neatly categorisable. This echoes the assertion in Section 2.2 that no single approach will be sufficiently accurate and rigorous to solve the problem of spam. SpamAssassin uses characteristics-based, content-based and collaborative- based spam filtering techniques as part of its arsenal. As a result of this spectrum of tests for spam, spammers find it very difficult to subvert or circumvent all of the tests. This is because there is no one aspect that spammers can attempt to work around. The four main techniques that SpamAssassin uses are:

- **Rule-Based Filtering.** SpamAssassin has over 850 rules that are used to differentiate spam emails from legitimate emails. These rules analyse both the headers (characteristics) and the text (content) of the email. These rules operate exactly as described in Section 3.4.1. The scores for these rules are created using a genetic algorithm. This means that the scores associated with rules may not be very intuitive to humans. It also means that the scoring system has to be retrained every time the rule-base is altered.

- **Blacklists.** Many different blacklists are supported by SpamAssassin, including the mail-abuse.org RBL (Section 4.2).
- **Razor.** Vipul's Razor (Section 4.1) is also incorporated into the SpamAssassin email filter as the collaborative component of the filter.
- **Bayesian Filtering.** SpamAssassin 2.50 onwards includes a Bayesian learning filter.

SpamAssassin can be classified as a moderately distributed anti-spam filter. There are a number of replicated copies of the rule-base. These will typically reside on MTAs of participating mail-servers, although there are a number of implementations that will work at the MUA level. The rules and their scores can be customised locally by users. However, this process is not transparent to the end users and requires a degree of computer-savvy in order to customise it. It is for this reason that SpamAssassin cannot be considered truly decentralised. The personalisation of the filter is not an aspect of the solution and must be done by explicit intervention outside the normal operation of the filter.

4.4 Cloudmark's SpamNet & Authority

Formed by Vipul Ved Prakash (author of Vipul's Razor [69]) and Jordan Ritter (chief architect for Napster [42]), Cloudmark has emerged as the one of the leading proprietary anti-spam solutions with two products. SpamNet [9] is aimed towards individuals and small businesses. Authority [8] is geared towards large companies and businesses.

4.4.1 SpamNet

SpamNet works in the classical model of a centralised collaborative spam filter. It is implemented as a plug-in to an MUA (currently only MS Outlook 2000, MS Outlook 2002 and MS Outlook

Number of SpamFighters	607,639
Time Saved (in hours)	76,913
Number of Emails Processed	90,641,529
Number of Spams Caught	27,688,591

Table 4.1: *Statistics reported by Cloudmark's SpamNet, as of 9th September 2003.*

XP), with a centralised server being maintained by Cloudmark. Users compute a signature on an email as it arrives and send the signature to the SpamNet servers to see if it matches any known spam signature. If there is a match, then the email is filtered into a junk email folder on the client's MUA. When new spam is reported, the entire spam is sent to the SpamNet servers for analysis. The SpamNet knowledge base is then updated with the signature for the new spam.

SpamNet reports impressive statistics, with 90% of spam filtered out. The Trust Evaluation System (TES) used by the system ranks users based on the number and accuracy of spams they report. Because the signature algorithm and the TES are proprietary and hence not available for evaluation, it is difficult to determine how scalable and efficient they are. However, with the statistics presented in Table 4.1, it is safe to say that they are robust and adequate for the current load experienced by SpamNet. Potentially, this approach suffers from the same two problems as the other centralised collaborative spam filters - poor scalability due to centralisation and sub-optimal performance and accuracy due to the lack of personalisation in the system.

4.4.2 Authority

Authority works as a moderately distributed collaborative system. It is implemented on the MTA of the organisation who has licensed the Authority software. SpamNet is used to collect spam emails for the service. These spams are analysed to extract what Cloudmark call "spamDNA" (features that are predictive of spam emails). A Bayesian classifier is used at the MTA to classify emails as they come into the organisation. By using the extensive SpamNet network to get accurate and immediate classifications of new spams, Authority can remain up-to-the-minute. The Cloudmark process is shown in Figure 4.1 (taken from [8]).

Another salient feature of Authority is that the spam email is stopped at the MTA level, so storage and processing costs that are normally associated with spam are saved. Additionally, the

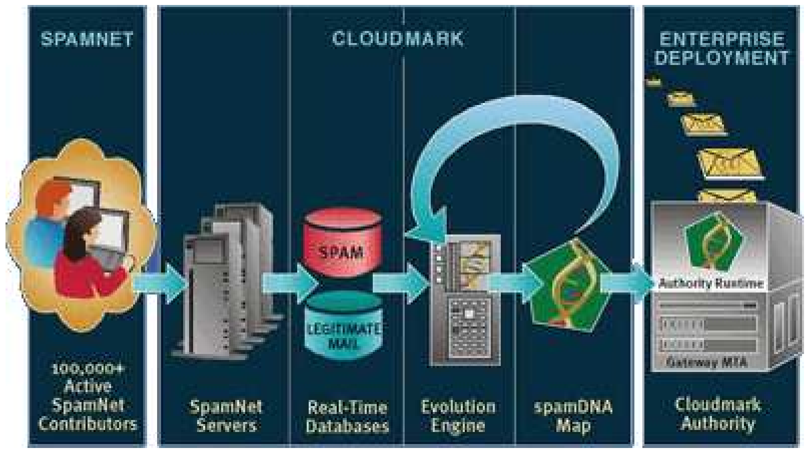


Figure 4.1: *The Cloudmark Process.*

loss in productivity that is associated with employees deleting spam is curtailed. Cloudmark claim that *"a company with 10,000 employees will save over $2 million a year in lost wages with Authority deployed"*. Authority is a nice example of a collaborative, content-based filtering system. However, it is still only moderately distributed, as the same knowledge base (gleaned from SpamNet) is replicated at each of the MTAs participating in the network. Additionally, there is no tailoring of the solution to the characteristic email that is seen at any MTA - all are required to maintain a copy of all of the data, so there is a lot of redundant and unused data stored.

4.5 Distributed Checksum Clearinghouse

The Distributed Checksum Clearinghouse [50] is a mechanism for identifying bulk emails. Once an email has been identified as a bulk email, it is compared against local whitelists to see if it has been solicited or not. The rationale behind this is that legitimate bulk emails will have been solicited, and hence can be put on a whitelist. The DCC is an example of a collaborative system comprised of distributed remote servers where bulk emails are identified and a local whitelist where the legitimate bulk emails are sorted from the unsolicited bulk emails.

Approximately 35 million emails are filtered by the DCC each day. Figure 4.2 shows the trend

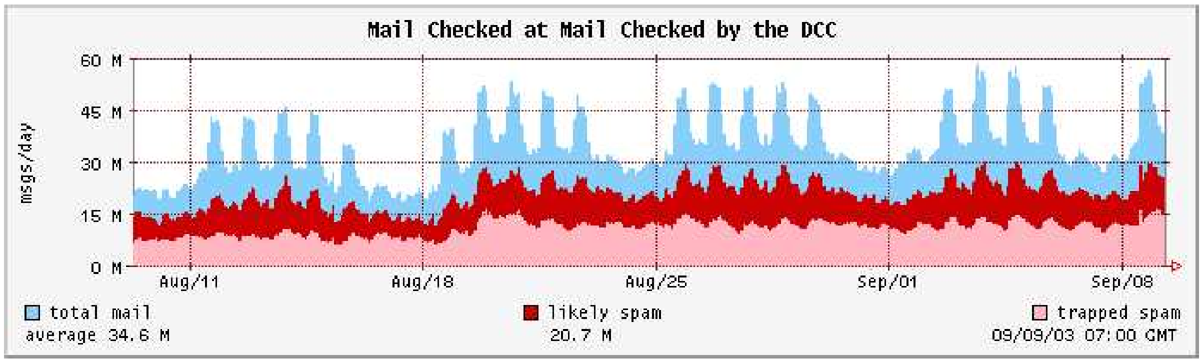


Figure 4.2: *Emails filtered by the DCC for the month ending 9th September 2003.*

of non-spam to spam received for the month up to September 9th, 2003. It is interesting to note that the pattern of legitimate email has very large peaks corresponding to the five working days whereas spam traffic remains fairly constant.

There are thousands of clients and over 150 servers in the DCC that collect and count checksums[1] related to several million messages per day. A DCC server totals reports of checksums of messages from clients and answers queries about the total counts for checksums of emails. For each email message received, a client computes a checksum and sends it to the server. The server responds with the total number of recipients of mails that match this checksum. If the total is greater than some threshold, then the email is identified as bulk. Comparison against local whitelists will identify it as solicited or unsolicited. The checksums in the DCC use fuzzy algorithms and since the DCC first came online in late 2000 and there have been several new checksum algorithms developed.

Figure 4.3 shows the breakdown of spam versus legitimate email for different types of ISP and ISPs in different regions. This shows that there is a distinctly different pattern of spam received by different types of ISP. Spammers will tend to avoid spamming government ISPs to try not to incur their wrath. European ISPs receive a lower proportion of spam than their US and Canadian counterparts, most probably due to the incoming Opt-In legislation.

[1]Checksums are generally referred to as signatures or hashes in this document

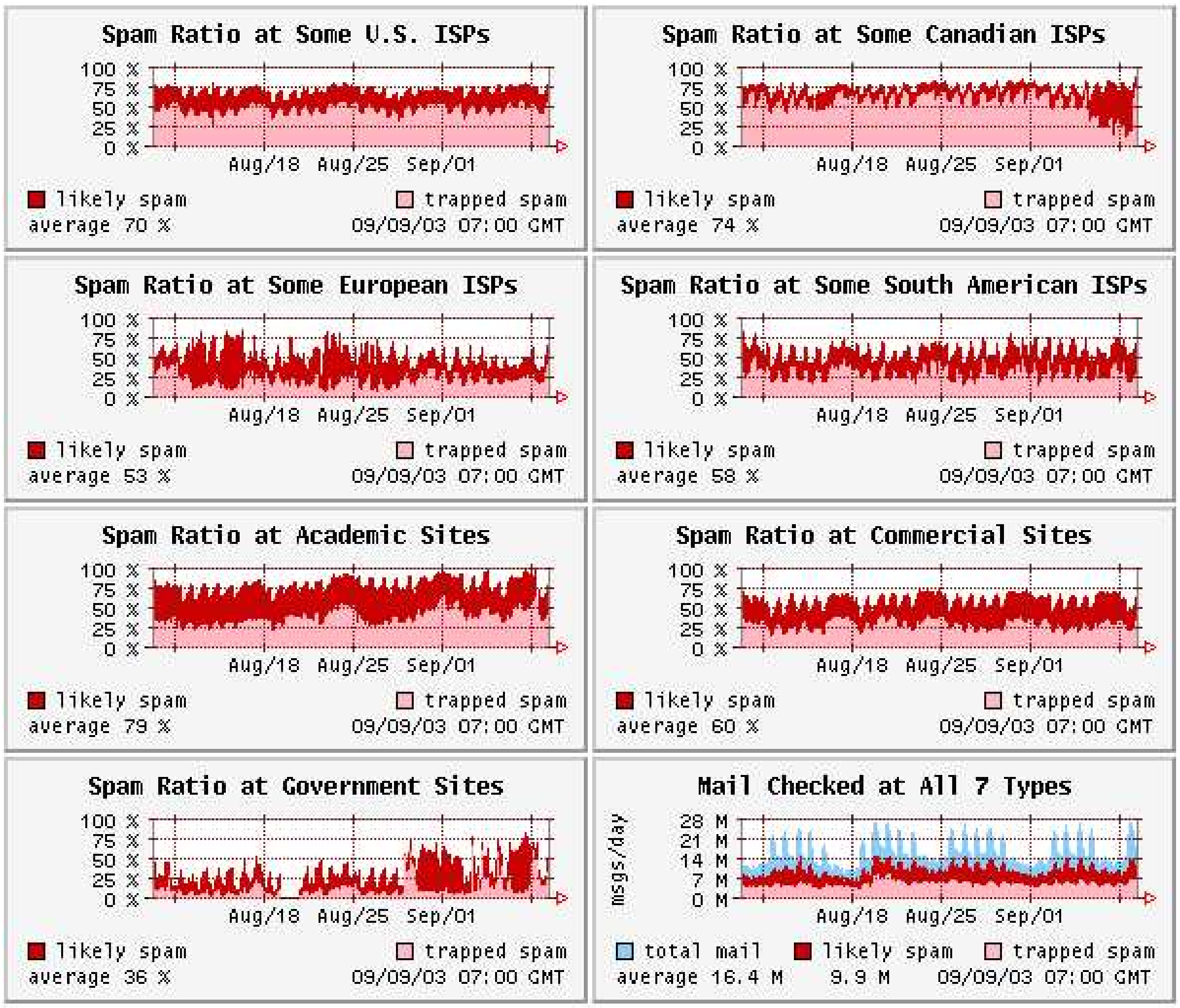


Figure 4.3: *Spam filtered by DCC, broken down by ISP type for two week period 18th August to 1st September, 2003.*

Chapter 5

Design

CASSANDRA is a collaborative anti-spam system, allowing node-decentralised research algorithms. It is a framework for personalised, collaborative spam filtering. There are many possible topologies, spam filtering techniques and levels at which this filtering can be achieved. These facts mean that CASSANDRA must be extensible, so much so that it will be adaptable to filtering systems beyond straightforward signature-based collaborative filtering.

This chapter will detail the design of the CASSANDRA framework. The possible choices for transport layer are presented and evaluated. A generalised structure for implementing a CASSANDRA-compliant component at participating nodes is presented. This structure is extensible and can be applied at either the MTA or MUA level. The objects and messages that are provided by the CASSANDRA framework are detailed. The protocols are used to show how different network topologies can be constructed from them.

5.1 Communication

In order for CASSANDRA to be open and interoperable, it is vital that all communication is done in a format that is consistent, well-defined and extensible. Web services are a prime example of a well-designed, interoperable, heterogeneous network. Communication is done using Extensible Markup Language (XML) [62]. XML is a versatile, extensible markup language with many parsers available for many languages on many platforms. Although verbose, it is the obvious choice for use as the communications language in the CASSANDRA framework.

5.2 Transport Layer

The choice of what protocol to use for the transport layer is a tricky one. By specifying one particular protocol for the CASSANDRA framework, the range of possible implementations is restricted. For this reason, no specific transport layer is dictated by CASSANDRA. The potential benefits and detriments to three of the possible transport layer options are discussed in this section. It must be noted that these are not the only possible protocols to use for the transport layer and that the use of any of these is not mandated by CASSANDRA.

5.2.1 JXTA

JXTA is the Java protocol suite for peer-to-peer networking [61]. From the description given on the project home-page:

> *JXTA technology is a set of open protocols that allow any connected device on the network ranging from cell phones and wireless PDAs to PCs and servers to communicate and collaborate in a P2P manner.*
>
> *JXTA peers create a virtual network where any peer can interact with other peers and resources directly even when some of the peers and resources are behind firewalls and NATs or are on different network transports.*

If a language-specific engine such as JXTA was used as the transport layer, there are a number of instant advantages and disadvantages. The advantages include the fact that group management, peer discovery and multi-hop routing can be obtained "for free" (i.e. the application level layer will not have to manage them). The disadvantages include the fact that nodes must be able to interpret Java and the fact that MUA implementations may not have sufficient resources over dial-up or wireless to participate fully. Additionally, users in large organisations may not have the permission to open ports in a corporate firewall. It is possible to conceive a Java-based solution that is situated at the MTA-level with Java mail-servers communicating over JXTA. However, it is not really feasible for an MUA-level solution.

5.2.2 HTTP

The Hyper Text Transfer Protocol (HTTP) is used for the transport layer in the Web Services model. HTTP is a long-established standard and practically every programming language that is in use today can interpret HTTP. This has the obvious advantage of interoperability and heterogeneity of nodes in a network. Additionally, the overwhelming majority of firewalls will have a port open for HTTP, so there is no extra configuration necessary by system administrators if users in their network wish to participate. HTTP enables very fast, synchronous communication between peers with low latency.

The downside to using HTTP for the transport layer is that any participant in the network must maintain a HTTP server in order to accept incoming connections. This is infeasible for many MUA level users who may not have the resources available (PDA users) or the permission (users inside a corporate firewall) to do so.

5.2.3 SMTP or AMTP

Using SMTP or AMTP for the transport layer is a very interesting solution. They have the same advantages as using HTTP, except that SMTP/AMTP is a high-latency form of communication. This means that communication should be asynchronous as far as possible if SMTP/AMTP is chosen for the transport layer. However, because SMTP/AMTP messages are queued by the MTA for delivery when the user is not online, SMTP/AMTP has an advantage over both HTTP and a language-specific solution. SMTP/AMTP enables a user to participate even though he/she is not online. Protocol messages will be queued at the MTA level and delivered the next time a user accesses his/her mailbox. Consequently, a user will never miss a protocol message (although there is no upper bound on how long it takes to reach them).

The disadvantages of using SMTP/AMTP are also interesting. Messages must be as asynchronous as possible. This is because of the inherent latency in SMTP/AMTP communication and also the possibility of long periods between a user accessing his/her email. At present, AMTP is still in its infancy and has no implementations, but when it becomes widespread there is another huge advantage to using AMTP. Senders are authenticated and message integrity assured, which greatly increases the system's security and resilience to attack.

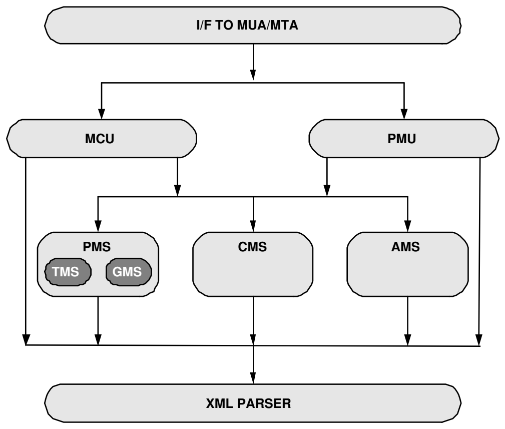


Figure 5.1: *Reference Diagram For A Node In CASSANDRA.*

5.3 Framework For Implementation Of A Participant

Figure 5.1 shows a reference diagram of how a CASSANDRA-compliant node can be constructed. The following list describes each component of the reference diagram.

- **I/F To MTA/MUA.** This is the interface to either an MTA or a MUA. Its main function is to integrate into the MTA or MUA so that the component can be used. This includes enabling users to report and revoke spam as well as configuring the node.
- **MCU.** Mail Classification Unit. The primary function of this unit is to perform the classification of an incoming email as either spam or non-spam.
- **PMU.** Protocol Message Unit. This unit is responsible for sending and receiving all protocol messages sent to and received from other participants in the network.
- **PMS.** Peer Management Service. This provides services to both the MCU and the PMU. The MCU relies on the PMS for classification purposes (if the system is being implemented as a collaborative whitelist). The PMU uses the PMS to format peer information and to rank peers when it sends messages to other participants. The PMU also sends new information about peers to the PMS upon receipt of such information from other participants. The PMS has two sub-components - the TMS and the GMS which are used to manage trust and cluster similar peers respectively.

- **TMS.** Trust Management Service. This service is responsible for computing and controlling how much this peer trusts other peers in the network to classify emails the same way as this peer would.

- **GMS.** Group Management Service. This maintains the view of the network from this peer's perspective. It identifies different groups of peers that share commonalities with this peer.

- **CMS.** Classifier Management Service. Responsible for maintaining a knowledge base of good classifiers. These classifiers can be many different things - peers, signatures, rules and so on. The MCU and PMU both use this service - the MCU to classify incoming emails and the PMU to format the top classifiers to share with other peers.

- **AMS.** Algorithm Management Service. This service is responsible for obtaining and maintaining the algorithms used to create particular classifiers or for using classifiers. The AMS is also responsible for obtaining and interpreting code modules so that new classifiers can be created/used.

- **XML Parser.** The XML Parser is used by all the other components. All communication between peers is formatted to adhere to the CASSANDRA XML schema. The parser interprets these and creates objects in the language the implementation is written in.

5.4 Objects

CASSANDRA defines a number of objects and messages that can be used in a personalised, collaborative spam filter. These are all marked-up in XML and as a result, are extensible. These objects and messages can be composed to build a personalised, collaborative spam filter based on signatures, rules or even a personalised, collaborative whitelist. They can be used to construct a personalised, collaborative case-based filter using machine learning techniques. Because these objects are defined in XML, they can be added to or extended in order to form composition blocks for other kinds of filter. There are three basic objects defined in CASSANDRA: Peers, Classifiers and Algorithms.

Component	Description
Peer ID	A string representing a unique identification for the Peer. This is probably a user's email address in most implementations because email addresses are ready-made unique identifiers.
Peer Type	A string representing the type of Peer. It can be "superPeer", "normalPeer" or "lightPeer". These enable different topologies to be built.
Trust	A float value in the range [-1.0, 1.0] representing how much the peer is trusted/distrusted to make the same classifications as another Peer. This encompasses most numerical computational models for trust and also enables trust models based on fuzzy logic.
Similarity	A float value in the range [0.0, 1.0] representing how similar the spam seen by a Peer is to the spam received by another Peer.
Rank	A positive integer representing the Peer's rank.
List of Classifiers	A list of Classifier objects that the Peer asserts are accurate and current.

Table 5.1: *Components of a Peer Object.*

5.4.1 Peers

Peers are the participants in a CASSANDRA network. Table 5.1 shows the components of a Peer object.

5.4.2 Classifiers

Classifiers are objects that are used to classify emails as spam or non-spam. These are the rules in a rule-based filter, the features in Bayesian filters, the signatures in collaborative filters or the cases in a case-based filter. The classifier is the unit that is typically recommended to and shared between peers in the network. Table 5.2 details the components of a classifier object.

5.4.3 Algorithms

Algorithms are a representation of the mechanism used to create classifiers. In a collaborative filter, this could be the hashing function used (SHA1 or Nilsimsa, for instance). In a case-based filter, this could be the machine learning technique used. The constituent components of an algorithm are described in Table 5.3. It is necessary to include the algorithm name and type for a number of reasons. First of all and most obviously, it is so that an accurate comparison

Component	**Description**
Value	A String representing the value of the Classifier. This value will be interpreted according to the Algorithm that is used to create the Classifier. It could be a hex value representing the value of a signature, or an XML marked-up string representing a rule, for example.
Algorithm	The Algorithm object that was used to create the value of this Classifier.
Date	A timestamp of when the last email was classified using this Classifier. Useful for working out the current "working group" of spam types and for LRU replacement of old Classifiers.
Creating Peer	A reference to the Peer that created this Classifier.
List Of Peers	A list of Peers who claim that this Classifier is accurate and current.

Table 5.2: *Components of a Classifier Object.*

Component	**Description**
Algorithm ID	A string representing the name of the Algorithm.
MUA/MTA ID	The MUA/MTA implementation that the Algorithm runs on.
Version	A string representing the version of the Algorithm.

Table 5.3: *Components of an Algorithm Object.*

can be made. It makes no sense to try and match a SHA1 signature against a set of Nilsimsa signatures, for example.

The second reason is to enable dynamic loading of algorithms. Imagine that a peer receives a classifier that is created with an algorithm the peer does not know of. A very powerful capability would be if the peer could construct a Uniform Resource Identifier (URI) from the description of an unknown algorithm and dynamically download and install the new algorithm in a manner transparent to the user. This provides for a very evolvable design as new and better algorithms can be added to the system without any intervention by the end-users (which is very good if the end-users are not particularly computer-savvy).

5.5 Messages

There are four basic operations in the CASSANDRA framework that require inter-peer communication. The messages used to perform this communication are all marked-up in XML, as

Component	Description
Sending Node	A Peer object representing the Peer that is sending the Discovery Ping. It can either be the joining Peer or an existing Peer who is forwarding it on behalf of the joining Peer. This is included to distinguish between who is joining and who is forwarding this message. Also enables sender authentication against the email address the email originated from.
Joining Node	A Peer object representing the Peer that is attempting to join the CASSANDRA network.
TTL	An integer representing the time to live (TTL) on the Discovery Ping. This is interpreted and modified by recipients in a manner consistent with the topology.

Table 5.4: *Components of a Discovery Ping Message.*

per the communication requirements in Section 5.1. Depending on the interpretation of these messages, different topologies of network can be created. This is demonstrated in Section 5.6. Implementations are under no requirement to use any or all of the messages for the four basic operations, nor are they restricted to just these messages. Implementations with sophisticated distributed trust management algorithms and group clustering algorithms will almost certainly have to augment the basic messages provided.

5.5.1 Joining The Network

There are two basic messages required to successfully join a CASSANDRA network - the Discovery Ping and the Discovery Pong. The joining node sends a Discovery Ping to an existing peer in the network. The peer sends back a Discovery Pong to the joining node, with sufficient information so that it can join and participate in the network. The Discovery Ping is then forwarded on to other existing peers in the network in a manner that is consistent with the topology of the network. This might be a simple flooding algorithm in a vanilla peer-to-peer topology, as per the Gnutella 1 protocol [24]. A more structured peer-to-peer network would have a more sophisticated mechanism, such as in HyperCup [37]. Tables 5.4 and 5.5 outline the components in the Discovery Ping and the Discovery Pong, respectively.

Component	**Description**
Sending Peer	A Peer object representing the Peer that is sending the Discovery Pong. This is included to enable sender authentication against the email address the email originated from.
List of Peers	A list of Peers who the sender asserts are trustworthy and similar to it.
List of Classifiers	A list of Classifiers that the sender asserts are current and accurate.

Table 5.5: *Components of a Discovery Pong Message.*

Component	**Description**
Sending Peer	A Peer object representing the Peer that is sending the Share. This is included to enable sender authentication against the email address the email originated from.
List of Peers	A list of Peers who the sender asserts are trustworthy and similar to it.
List of Classifiers	A list of Classifiers that the sender asserts are current and accurate.

Table 5.6: *Components of Share Ping and Share Pong Messages.*

5.5.2 Sharing Information

There are two messages for sharing information - the Share Ping and the Share Pong. A Share Ping is sent by a peer when it determines it necessary to share information with other peers. This can be after any one of a number of different events: after a fixed interval; after a new spam that has not been seen before is classified; at application start-up or shut-down; or any combination of these. The recipients of a Share Ping are those peers who the sending peer deems most likely to benefit from receiving it. The exact semantics of this are determined by whatever group clustering algorithm the CASSANDRA implementation is using.

A Share Pong is sent in response to a Share Ping. It can be sent back to the sender of the Pong, to a group of peers or not sent at all, again at the discretion of the group clustering algorithm. The content of the Share Ping and the Share Pong are exactly the same, and is described in Table 5.6

Component	Description
Sending Peer	A Peer object representing the Peer that is sending the Revoke. This is included to enable sender authentication against the email address the email originated from.
Classifier	The Classifier that the sending Peer believes should be revoked.

Table 5.7: *Components of a Revoke Message.*

5.5.3 Revoking Information

"To err is human, to forgive, divine". People make mistakes. Therefore it is prudent to enable some revocation mechanism for classifiers. A classifier may need to be revoked if it is a signature mistakenly computed on a non-spam email or a rule that creates false positives/negatives, for example. The recipient of a Revoke is under no obligation to act upon it. It may be that the recipient believes that the classifier is good (i.e. that the particular type of email it filters is/is not spam), or that the recipient has more recommendations from other peers about the classifier and takes the majority vote. Table 5.7 shows the components of a Revoke recommendation message.

5.5.4 Leaving The Network

In order for a system to be complete, there must be provision for participants to opt out of it. It should be straightforward for a peer to successfully leave the network. For this to be true, two things must happen - all peers that know of the leaving peer should be notified of the decision to do so and the leaving peer should receive no more protocol messages after everyone has been notified. The semantics for routing this message are very similar to those for the Discovery Ping, except that an extra round of messages is needed to confirm the decision to opt out (this means that malicious peers cannot opt others out without their knowledge or consent). Therefore, there are three messages required - Leave, Leave Query and Leave Confirm.

A Leave message is sent to all known peers. This message is forwarded to every peer that knows of the leaving peer (as far as possible). Every peer must send a Leave Query back to the leaving peer, which must be answered with a Leave Confirm. The Leave message contains the same components as a Discovery Ping (see Table 5.1). Leave Queries and Leave Confirms contain a reference to the sending peer (for authentication) and a simple Boolean ("True" to confirm that the peer really wants to opt out and "False" to deny it).

5.6 Network Topology

No particular network topology is assumed in the CASSANDRA framework. Three different types of peer are defined - superPeer, normalPeer and lightPeer. These can be used to compose a sophisticated network topology, depending on how messages are interpreted by the different type of peer. LightPeers are included so that the framework can be extended to include wireless or PDA environments where resources (bandwidth, CPU cycles) are scarce.

5.6.1 Client/Server

A Client/Server topology can be constructed by having a network where there is only one superPeer and many normalPeers. To join, a node only has to send a Discovery Ping to the server, which will return a Discovery Pong that lists all the other clients in the network. To share information or revoke a classifier, a Share Ping can be sent to the server, which forwards it on to all the clients - no Share Pongs are necessary.

5.6.2 Vanilla Peer-To-Peer

This type of network can be composed solely of normalPeers. The reference implementation supplied is a vanilla peer-to-peer topology and is presented in Chapter 6.

5.6.3 Super-Peer

Obviously, a super-peer based network will have superPeers and normalPeers. One possible implementation could have normalPeers operating locally in a client/server fashion with a superPeer. The superPeers could then communicate between themselves in an ordinary peer-to-peer fashion at the super-peer level.

5.7 Security

There is no security beyond a simple authentication mechanism provided in the CASSANDRA framework. The transport layer that is chosen must provide message authentication, integrity and encryption (if required). There is no concept of malicious peers in the CASSANDRA framework, merely peers that are very unlike others, both in terms of what email (spam and non-spam) they receive and what they classify as spam. This means that peers that would be considered subversive in other protocols are ranked very poorly by their contemporaries in a CASSANDRA system. As a result, they will be ignored at the application level and so will be unlikely to achieve subversion of the system. It is incumbent upon the designer of any CASSANDRA implementation to pay particular attention to this in order to ensure that this is the case.

Chapter 6

Implementation

This chapter presents a proof-of-concept of a personalised, collaborative spam filter that is permitted by the CASSANDRA framework. The specification of the filter is given and the various components of the design are described. Inter-peer communication is detailed, as is the filtering process and user interaction with the plug-in.

6.1 Prototype Specification

This section details the specifications of the prototype implementation. The topology, transport layer, trust model and clustering algorithm are described. The events that occur in the system are defined. This is followed by a brief discussion on how security is addressed in the implementation. Finally, the MUA to be extended and how a user will use it is described.

6.1.1 Topology

The prototype implementation has a fully decentralised, peer-to-peer topology. Therefore, all peers in the network are of "normalPeer" type. The choice of peer-to-peer for the topology has ramifications for the clustering algorithm, the trust model and the mechanism for peer discovery. Each of these ramifications will be discussed in the following sections.

6.1.2 Type Of Filter

A signature-sharing collaborative system was chosen as the type of filter to implement. Therefore, the network will operate by users classifying spam emails as they are received, computing a signature on the spam and propagating the signature to the peers it suspects will most likely receive the same spam. In this implementation, the classifiers are signatures computed on spam emails. Algorithms are the hashing functions used to create these signatures.

6.1.3 Transport Layer

SMTP was chosen for the transport layer in the prototype implementation. There were three reasons behind this decision. The first was that SMTP as a transport layer is a relatively novel concept (only Cottereau [34] has done it before). The second major reason was that the testing was to be carried out across a firewall and system administrators were unlikely to open a port in the firewall to enable testing.

The third reason was that using SMTP for the transport layer would greatly decrease development time. This was because SMTP already exists and very little work needs to be done to interface to it. Additionally, sender authentication and message integrity can be ignored during testing, because the system can be migrated from SMTP to AMTP with only minor changes.

6.1.4 Data Storage

Given that all inter-peer communication is marked-up in XML, XML is the obvious choice for persistent data storage between application shut-down and the next time it is started up. There are five pieces of data that must be stored - the configuration data, the peers, the classifiers, the revoked classifiers and the algorithms. The CASSANDRA objects are each stored in separate XML files as a list of those objects. This means that they are easily exportable and importable between nodes in an offline manner.

6.1.5 Events In The Prototype Implementation

This section describes the events that are used in the prototype implementation. These events trigger actions in the prototype and drive the clustering, trust and peer interaction processes.

6.1.5.1 Hits

A hit is defined to be the event when a new, non-protocol email arrives and a signature computed on it matches one of the existing signatures. This identifies the email as a spam.

6.1.5.2 Misses

A miss is defined to be the event when a signature is dropped from the list of active signatures without having had a hit.

6.1.5.3 Revokes

A revoke is when an email that has been classified as spam is marked as non-spam by the user. The signature that classified the spam may have been computed in error by the peer, or have been recommended to the peer by another peer who has a different opinion of what constitutes spam.

6.1.5.4 New Classifications

When an email arrives at a peer that is not classified as spam, it is delivered to the user's inbox. If the user decides that the email is spam, it is marked as spam by the user. This is the "new classification" event.

6.1.5.5 Start-Up And Shut-Down

These are the events triggered when the prototype is started and stopped within the MUA. Ordinarily, these will be tied in with the MUA start-up and shut-down, but during testing they were provided as fireable events so that statistics could be written out to file.

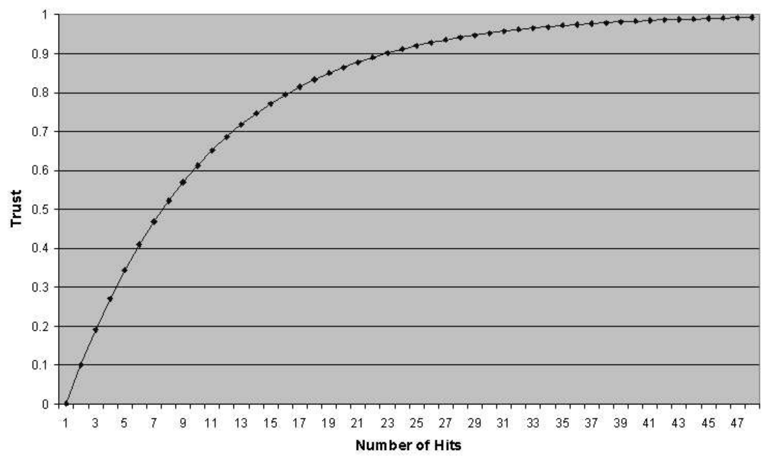


Figure 6.1: *Trust Scale.*

6.1.6 Trust

Managing trust in distributed systems is a non-trivial task. The degree of difficulty of trust management in a decentralised system is even higher again. Because of time constraints and the scope of the project, a rudimentary trust management system is used in the prototype implementation. There is no distributed trust, that is, a peer will only make modifications to its amount of trust in another peer based on direct interaction with that peer. This means that there is no reputation management protocol to construct. A peer's trust in another peer lies on the scale of [0.0, 1.0]. There are two events that can affect the trust a peer has in another peer - Hits and Revokes (see Section 6.1.5 for more on events).

Scale

The trust scale is in the range [0.0, 1.0]. However, the model is not linear in the prototype. An asymptotic scale is used, such that trust for an unknown Peer rises from 0.0 to an asymptote at 1.0 (i.e. it approaches 1.0, but never gets there). Figure 6.1 illustrates this. An asymptotic scale was chosen because it enables peers to accumulate trust very rapidly, so new peers can quickly build up good reputations for themselves.

Hits

Whenever a hit occurs, the existing signature has a list of peers that have recommended the signature to us. Each of these peers has therefore made the same classification as we have and so we can trust that they are likely to make the same classification as we will in other circumstances. Each peer in the list gains one tenth of the current value it is currently below 1.0. Programatically, this is expressed as: $trust = trust + (0.1 * (1.0 - trust))$.

Revokes

Whenever a revoke occurs, a signature has been recommended to the peer that the user disagrees is spam. All the peers in the signature's peer list have therefore classified the signature differently to the current peer. Therefore, these peers are more likely to recommend signatures that lead to false positives. False positives in spam filtering systems are potentially very costly and so the modification to the trust value is severe. Every peer who is in the hit list loses one quarter of its current value for trust.

6.1.7 Clustering

Time constraints meant that a sophisticated model could not be developed for the clustering algorithm. The clustering model in the he implementation does not try to mine what different groups a peer suspects other peers belong to. There is only a scale that gauges how similar the spam received by other peers is to that received by the peer. This is measured on the same scale as the trust algorithm, i.e. asymptotically approaching 1.0 in the range [0.0, 1.0]. The "hit" event is used to increase the similarity value in exactly the same fashion as the trust value.

In contrast to the trust model, the event that causes a decrease in similarity is a miss, because it means that a spam was received by the peers in the signature's peer list, but was not received by the current peer. Whenever a miss occurs, all the peers that have recommended the signature that is dropped lose one tenth of their current similarity value.

6.1.8 Security

There is no security beyond the basic sender authentication included in the CASSANDRA framework-defined messages. Message integrity and sender authenticity can be supplied by migrating to AMTP from SMTP at the transport layer. Because AMTP is not actually in use yet, these features could be supplied by existing products from vendors. Sender authenticity and message integrity could be assured by using free digital signatures on the emails, such Thawte's Personal Email Certificate [65], or a legally-valid digital signature, such as the In-Person Certificate offered by Ezitrust [18].

6.1.9 Mail User Agent

The MUA that was extended to use in the implementation was MS Outlook 2002. Outlook provides a good interface for plug-in extensions. It exposes application level events through its API, so it is easy to integrate plug-ins to events associated with emails. [21, 53] proved vital resources for learning how to program to the Outlook interface.

Choosing MS Outlook 2002 as the MUA to extend had the additional benefit of being a very popular MUA. This meant that there were quite a few users who agreed to help test the system. This is important because it means that the results presented in Chapter 7 are drawn from a "live" environment.

6.2 Plug-In Design

For completeness, this section describes what each of the components in the framework reference diagram does (see Table 5.1 for the reference diagram).

6.2.1 GMS

The Group Management Service maintains an ordered list of peers and the corresponding similarity value. It modifies these values and removes peers from its list when instructed to do so by the PMS.

6.2.2 TMS

The Trust Management Service maintains an ordered list of peers and the corresponding trust value. It modifies these values and removes peers from its list when instructed to do so by the PMS.

6.2.3 PMS

The Peer Management Service is responsible for maintaining a ranked list of other peers in the network. There is a simple ranking mechanism used - the peers are ordered by the product of their trust and similarity. This means that peers who do not receive the same email, or cannot be trusted to classify emails the same way as the user does will tend to be ranked down the list. The PMS is also responsible for making calls to the GMS and TMS as appropriate.

6.2.4 CMS

The Classifier Management Service maintains a list of the most recently hit signatures. It is responsible for maintaining and updating this list whenever new protocol messages arrive or new spam arrives. The CMS also maintains a list of revoked signatures, so that new signatures that are not wanted will not be re-added to the list once they have been removed.

6.2.5 AMS

The Algorithm Management Service is responsible for creating signatures using the most recently learned-of algorithm. The algorithms are also downloaded and installed dynamically and transparently by the AMS. This process is abstracted behind an interface, but was not implemented due to time constraints. The algorithm used to create signatures is the SHA1 function. This is not a fuzzy hash, but implementations of the SHA1 hash are readily available and it so was chosen, again due to time constraints.

6.2.6 MCU

The Mail Classifier Unit is responsible for filtering spam emails, non-spam emails and protocol emails. Protocol emails are delivered to the PMU, spam emails moved to the user's junk folder and normal email to the inbox. This process is described in more detail in Section 6.5.

6.2.7 PMU

The Protocol Message Unit controls the flow of events when protocol messages are received and sends protocol messages using Outlook MailItem objects.

6.3 Inter-Peer Communication

This section describes how peers interact with each other in the network.

6.3.1 Discovery

Discovery is done using a simple flooding algorithm, as per the original Gnutella 1 protocol [24]. A node who wishes to join the system must send a Discovery Ping to a peer in the network. This peer sends back a Discovery Pong, and forwards on the Discovery Ping to every peer it knows of, decrementing the time-to-live (TTL) value. The peers who receive this Ping each send a Pong to the joining node, decrement the TTL and forward it on. This continues until the TTL is zero. This is not the most efficient discovery protocol, as peers may receive the same Ping more than once. Opinions are divided over exactly how inefficient this mechanism is - some estimate it is of order n^2 [51] while others [35] disagree. It is not an optimal solution, but is sufficient for the purposes of the proof-of-concept implementation.

6.3.2 Sharing

Shares are triggered by the "New Classification" event. Whenever a new spam is classified by a user, the peer computes a signature on it and sends a Share Ping to the top ten ranked peers that it knows of. These peers each respond with a Share Pong.

6.3.3 Revocation

Because the trust model used in the proof-of-concept implementation does not permit sharing trust values (see Section 6.1.6), sending Revoke messages to other users is meaningless in this implementation.

6.3.4 Leaving

Due to time constraints, this feature was not implemented in the proof-of-concept system.

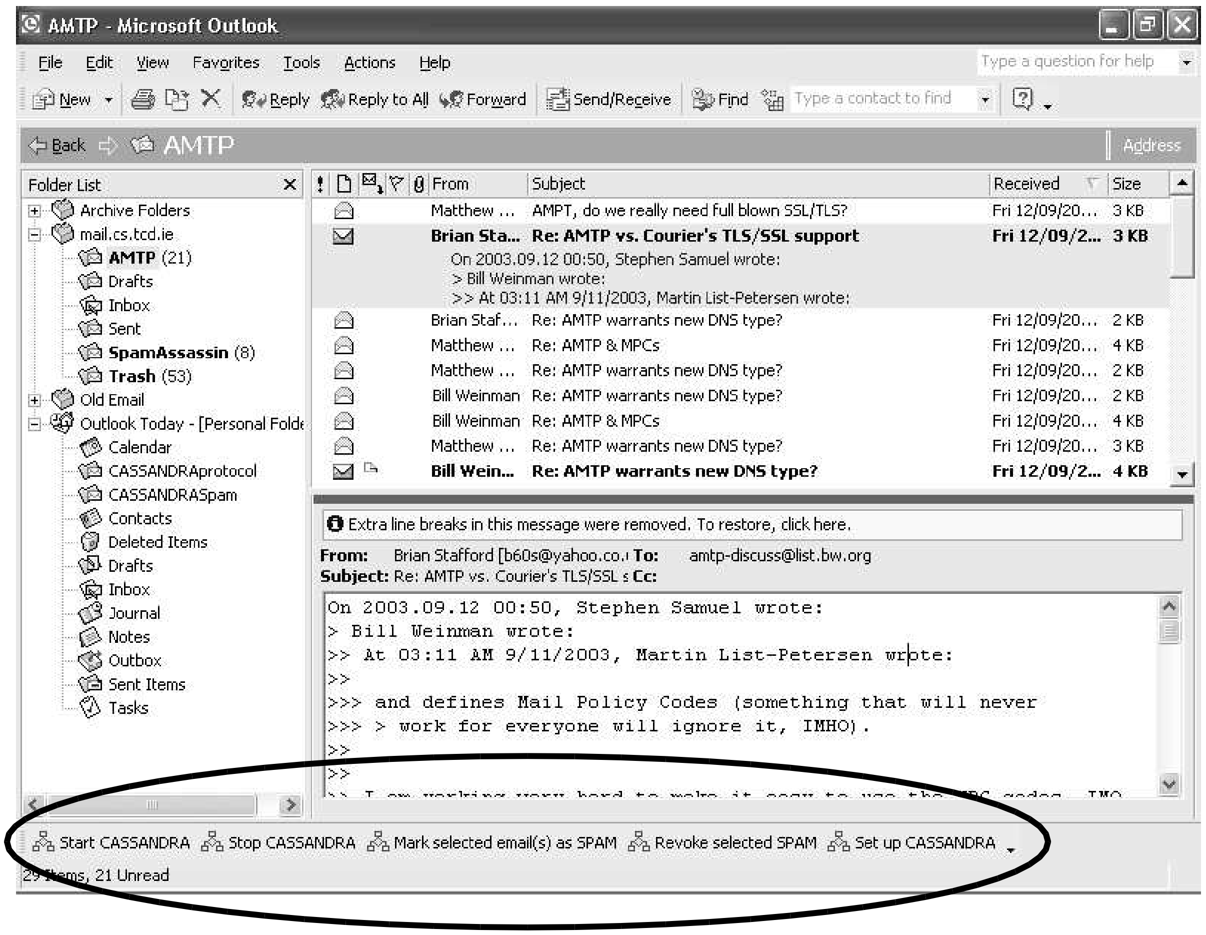
6.4 User Interface

The User Interface (UI) was implemented with the simple addition of an unobtrusive toolbar with five buttons at the bottom of the Outlook Window. Figure 6.2 shows a screenshot of the UI with the CASSANDRA toolbar highlighted. All five buttons are not strictly necessary. The “Start CASSANDRA” and “Stop CASSANDRA” buttons were put in to aid in statistics collection for the test. As remarked in Section 6.1.5.5, the start-up and shut-down events can be tied to the Outlook application’s start-up and shut-down events. Similarly, the button to set up CASSANDRA can be put into a drop down menu in the menu bar, but was placed on the CASSANDRA toolbar to speed up the testing and evaluation process.

The actual operation of the plug-in is entirely transparent to the user. The only intervention required from the user is to mark new spam email as such by simply highlighting the email(s) and clicking on the “Mark selected email(s) as SPAM” button. Similarly, a false positive can be revoked using the “Revoke selected SPAM” button. These buttons are very intuitive and simple to use. In fact, they are just as straightforward to use as the ordinary delete button, so users are more likely to use them than in other collaborative filters where reporting spam is a cumbersome process.

6.4.1 Marking New Spam

When an email is marked as spam, the spams are moved from the user’s inbox to a folder that has been specified by the user at install-time. Typically, this will be a junk email folder, but it



From	Subject	Received	Size
Matthew ...	AMPT, do we really need full blown SSL/TLS?	Fri 12/09/20...	3 KB
Brian Sta...	**Re: AMTP vs. Courier's TLS/SSL support**	**Fri 12/09/2...**	**3 KB**
Brian Staf...	Re: AMTP warrants new DNS type?	Fri 12/09/20...	2 KB
Matthew ...	Re: AMTP & MPCs	Fri 12/09/20...	4 KB
Matthew ...	Re: AMTP warrants new DNS type?	Fri 12/09/20...	2 KB
Bill Weinman	Re: AMTP warrants new DNS type?	Fri 12/09/20...	2 KB
Bill Weinman	Re: AMTP & MPCs	Fri 12/09/20...	4 KB
Matthew ...	Re: AMTP warrants new DNS type?	Fri 12/09/20...	3 KB
Bill Wein...	**Re: AMTP warrants new DNS type?**	**Fri 12/09/2...**	**4 KB**

On 2003.09.12 00:50, Stephen Samuel wrote:
> Bill Weinman wrote:
>> At 03:11 AM 9/11/2003, Martin List-Petersen wrote:

Extra line breaks in this message were removed. To restore, click here.

From: Brian Stafford [b60s@yahoo.co.u **To:** amtp-discuss@list.bw.org
Subject: Re: AMTP vs. Courier's TLS/SSL s **Cc:**

```
On 2003.09.12 00:50, Stephen Samuel wrote:
> Bill Weinman wrote:
>> At 03:11 AM 9/11/2003, Martin List-Petersen wrote:
>>
>>> and defines Mail Policy Codes (something that will never
>>> > work for everyone will ignore it, IMHO).
>>
>>
```

Figure 6.2: *Screenshot of MUA Interface.*

can be the trash folder if the user wishes to pursue a very aggressive spam policy. In Figure 6.2, spams are moved to the "CASSANDRAspam" folder. The plug-in will then send Share Pings to its top ten peers on the user's behalf.

6.4.2 Revoking Signatures

If there is a false positive, then the user can restore the email to his/her inbox by highlighting the email in the junk email or trash folder by clicking the revoke button. The email is then moved back to the inbox. The trust values for the peers who recommended the signature are then adjusted.

6.5 Filtering Emails

This section will briefly outline how emails received by the MUA are filtered.

Protocol Emails

Protocol emails are those that are used for inter-peer communication. These messages are delivered to the PMS, CMS and AMS as appropriate. Ordinarily, they will then be moved to the trash so that the user's inbox is not clogged up by these messages. However, in order to collect statistics, these messages are moved to a folder called "CASSANDRAprotocol" in the implementation presented.

- **Discovery Ping.** These are delivered to the PMS, so that the joining peer can be added to the list of known peers. The PMU then makes calls to the PMS and the CMS to get the list of top ranked peers and signatures to send to the joining node. The PMU formats these lists and sends a Discovery Pong to the joining node.
- **Discovery Pong.** When a Discovery Pong is received, the list of peers is sent to the PMS and the list of signatures to the CMS to be added to the knowledge-base.
- **Share Ping/Pong.** When either of these messages are received, the list of peers and the list of signatures are sent to the PMS and CMS, respectively. In the PMS, for every

peer that has a signature in its hit-list that we know of, the trust and similarity values are updated in the TMS and GMS. If a peer has a signature which we have previously revoked, then the TMS decreases its trust in that peer. In the CMS, each signature is compared to the list of signatures known by the peer. If it is not found, the signature is added to the list of known signatures in the CMS. If it is found, then the most recent of the two "last hit" dates is taken (for LRU replacement).

Non-Protocol Emails

Each non-protocol email that is received by the MUA has a signature created on it. The signature is compared to all the signatures in the CMS. If no match is found, then the email is delivered to the inbox as normal. Otherwise, it is marked as spam and moved to the spam folder. The CMS updates the "last hit" date for the signature and adds itself to the list of peers that have had hits with this signature. The GMS and TMS then update the similarity and trust values for all the other peers who are on the signature's hit list.

Chapter 7

Evaluation

The implementation was tested on simulated spam by users in a real environment. This chapter details the experimental set-up and presents the results obtained. From these results, conclusions about the effectiveness of the prototype implementation are drawn. Finally, the implementation and the framework as a whole is evaluated using the criteria used in [34].

7.1 Experimental Set-Up

This section describes the experimental methodology used to evaluate the prototype implementation.

7.1.1 Simulated Spam

It is infeasible to expect that any email addresses chosen at random will receive the same spam. For this reason, simulated spam had to be sent to the peers in the network. There were eight peers in the network and five spam lists were created. Figure 7.1 shows how the peers were grouped into the spam lists. Different spam was sent to each of these lists. Additionally, non-spam email was sent to these lists. Table 7.1 shows the breakdown of what spam and non-spam was sent to each list.

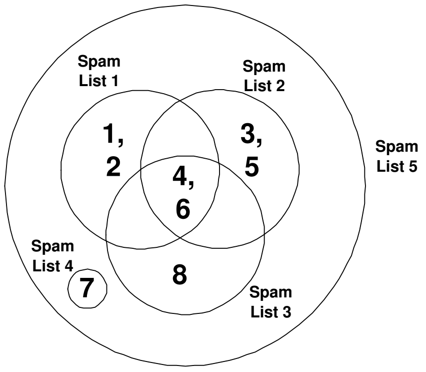


Figure 7.1: *Spam Lists used for Experiment.*

Spam List	Peers on List	Spam Emails	Non-Spam Emails
1	1, 2, 4, 6	3	3
2	3, 4, 5, 6	3	2
3	4, 6, 7, 8	3	2
4	7	7	0
5	1, 2, 3, 4, 5, 6, 7, 8	6	5
	Total Emails	**22**	**12**

Table 7.1: *Spam and Non-Spam sent to each Spam List.*

Peer	Role In Network
1	Active, early reporter of spam.
2	Lazy peer. Contributes no reports to network.
3	Active peer. Not early reporter of spam. Produced one false positive.
4	Active peer. Not early reporter. Central in network topology.
5	Lazy peer. Contributes no reports to network.
6	Active, early reporter of spam. Central in network topology.
7	Late Joiner. Active peer. Early reporter of spam from list no-one else is on.
8	Active peer. Not early reporter of spam. Produced one false positive.

Table 7.2: *Peer Roles In Experiment.*

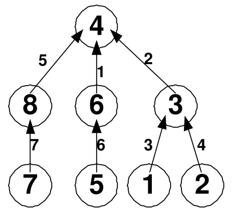


Figure 7.2: *Network Construction.*

7.1.2 User Roles

The eight users were given different roles to play in the experiment. These roles varied greatly in order to try to simulate a real environment. Table 7.2 shows the roles that each peer played. The false positives created by peers 3 and 8 were computed in error by the users.

7.1.3 Network Construction

The network in the experiment was created using the discovery protocols (i.e. it was not assumed to be in existence for the purposes of the experiment). Figure 7.2 shows the order in which each peer joined the network and which peer it joined to. The arrows are numbered in the order that the nodes joined the network. Peer 4 was the original node in the network.

7.2 Experimental Results

The system is a non-deterministic one, because of network latencies and the fact that users will not always check their email at the same times. This non-determinism is evident in a few places in the results where some inconsistencies arise. These will be highlighted as they are come across.

7.2.1 Discovery Protocol

Table 7.3 shows the number and type of protocol messages each peer received during the joining process. Not all peers joined at the same time. Additionally, spam and non-spam was sent to

Peer	Discovery Pings	Discovery Pongs	Total
1	21	3	24
2	19	4	23
3	24	2	26
4	24	0	24
5	12	6	18
6	22	1	23
7	7	7	14
8	17	5	22

Table 7.3: *Discovery Messages Received By Peers.*

the various lists while peers were joining to simulate a real environment. It can be seen that each node receives exactly one discovery pong from each peer that is in the network when it joins. Given that the network is created in a fragmented manner (c.f. Figure 7.2), it is clear that the network becomes more fully connected during the discovery process. This means that the discovery protocol is a success within the TTL horizon. It can also be seen that each peer receives the same number of superfluous discovery pings (i.e. its own ping forwarded back to it). In order to eliminate unnecessary traffic, the process can be optimised so that a ping is not sent back to the peer from which it originated.

7.2.2 Spam Signatures Known

Table 7.4 shows the break down of signatures known by each peer. From Table 7.1, the total number of spams in the system is 22. Even though no peer in the system received all the spam, every peer knows of every signature that was created (even if they disagree with the verdict that it is spam). This fact shows that the system can function as a non-personalised, collaborative filter, because the knowledge-base of signatures is essentially replicated at every node. The network used for testing was small (due to lack of resources available). As a result this does not give an accurate view of whether or not the clustering algorithm worked. Section 7.2.5 evaluates the clustering protocol.

Peer	Spams Received	Spam Signatures	Signatures Revoked	Signatures Known
1	9	22	0	22
2	9	22	0	22
3	9	20	2	22
4	15	21	1	22
5	9	22	0	22
6	15	20	2	22
7	13	22	0	22
8	9	21	1	22

Table 7.4: *Spams Received And Signatures Known.*

Peer	Spams	Non-Spam	**Total**	Pings	Pongs	**Total**	***Protocol Overhead***	***Overhead (minus Pongs)***
1	9	8	**17**	11	7	**18**	**51.4%**	**39.3%**
2	9	8	**17**	12	0	**12**	**41.4%**	**41.4%**
3	9	7	**16**	11	7	**18**	**52.9%**	**40.7%**
4	15	12	**27**	11	6	**17**	**38.6%**	**28.9%**
5	9	7	**16**	12	0	**12**	**42.3%**	**42.3%**
6	15	12	**27**	10	14	**24**	**47.1%**	**27.0%**
7	13	7	**20**	6	41	**47**	**70.1%**	**23.1%**
8	9	7	**16**	11	7	**18**	**52.9%**	**40.8%**

Table 7.5: *Sharing Protocol Overhead.*

7.2.3 Sharing Protocol

Table 7.5 shows the sharing protocol overhead as a percentage of total email traffic seen at each of the peers. The values for the overhead are in the range [38.6%, 70.1%]. This overhead is simply too much. Given that spam emails constitute 50% of all email traffic, the extra protocol overheads would mean that legitimate email would constitute only 20%-25% of email traffic. However, there are optimisations that can reduce the overhead significantly in the system.

In Section 7.2.1 it was seen that the network is fully connected. This means that all share pings will reach all peers in this system. Therefore, the share pongs are superfluous in a fully

Reporters	**Peer 1**	**Peer 2**	**Peer3**	**Peer 4**	**Peer 5**	**Peer 6**	**Peer 7**	**Peer 8**
Peer 1	6	6	6	6	6	6	5	6
Peer 2	0	0	0	0	0	0	0	0
Peer 3	1	1	0 (1)	1	1	0 (1)	1	1
Peer 4	1	1	1	1	1	1	1	1
Peer 5	0	0	0	0	1	0	0	0
Peer 6	6	6	5 (1)	6	6	7	6	7
Peer 7	7	7	7	7	7	7	7	7
Peer 8	1	1	1	0 (1)	1	0 (1)	1	0 (1)

**Revokes shown in brackets.*

Table 7.6: *Peers' Opinion Of First Reporters.*

connected system. By removing share pongs, the overhead is reduced to the range [23.1%, 42.3%] of total traffic, which is comparable to the amount of spam received. This overhead can be further reduced by using a fuzzy hash instead of SHA1. By using a fuzzy hash, the overhead would be reduced to being comparable to the number of *distinct types* of spam received. This will be considerably less and is entirely dependent on how good the fuzzy hash is.

7.2.4 First Reporters

This section is included to demonstrate the non-deterministic nature of the system. Table 7.6 shows which peers first reported each spam to each peer. This table is read across to get a peer and down to see how many signatures it thinks the other peers were first to report. Given that the network is fully connected, it is expected that these views would all be the same. However, this is not the case. These inconsistencies are caused by network latencies and multiple peers concurrently marking new email as spam.

7.2.5 Clustering And Ranking

Table 7.7 summarises Tables 7.8 and 7.9 below. It shows the score (the product of the trust and similarity values) that each peer has for the others, with the rank in brackets. The table is read across to get the peer and down to see how that peer ranked the other peers. The ranks were obtained by sorting the peers based on the product of their trust and similarity ratings, as per

	Peer1	**Peer2**	**Peer3**	**Peer4**	**Peer5**	**Peer6**	**Peer7**	**Peer8**	**Rank**
Peer1	1.00 (1)	0.32 (2)	0.22 (4)	0.38 (3)	0.17 (6)	0.51 (3)	0.01 (3)	0.17 (5)	**3**
Peer2	0.00 (7)	1.00 (1)	0.12 (6)	0.27 (4)	0.00 (8)	0.51 (3)	0.01 (3)	0.00 (6)	**5**
Peer3	0.16 (4)	0.16 (5)	1.00 (1)	0.27 (4)	0.38 (3)	0.44 (5)	0.01 (3)	0.27 (4)	**4**
Peer4	0.32 (2)	0.32 (2)	0.00 (8)	1.00 (1)	0.51 (2)	0.56 (2)	0.07 (2)	0.56 (2)	**1**
Peer5	0.00 (7)	0.00 (8)	0.27 (3)	0.00 (8)	1.00 (1)	0.27 (7)	0.00 (6)	0.00 (6)	**7**
Peer6	0.32 (2)	0.27 (4)	0.38 (2)	0.51 (2)	0.22 (4)	1.00 (1)	0.00 (6)	0.32 (3)	**2**
Peer7	0.10 (6)	0.10 (7)	0.10 (7)	0.10 (7)	0.10 (7)	0.00 (8)	1.00 (1)	0.00 (6)	**8**
Peer8	0.12 (5)	0.12 (6)	0.21 (5)	0.22 (6)	0.27 (5)	0.34 (6)	0.00 (6)	1.00 (1)	**6**

Table 7.7: *Peers' Overall Ranks.*

Similarity	**Peer 1**	**Peer 2**	**Peer 3**	**Peer 4**	**Peer 5**	**Peer 6**	**Peer 7**	**Peer 8**
Peer 1	1.0	0.56953	0.46856	0.61258	0.40951	0.71757	0.1	0.40951
Peer 2	0.0	1.0	0.3439	0.5217	0.0	0.71757	0.1	0.0
Peer 3	0.40951	0.40951	1.0	0.5217	0.61258	0.74581	0.1	0.5217
Peer 4	0.56953	0.56953	0.0	1.0	0.71757	0.74581	0.27	0.7458
Peer 5	0.0	0.0	0.5217	0.0	1.0	0.271	0.0	0.0
Peer 6	0.56953	0.5217	0.61258	0.71757	0.46856	1.0	0.0	0.56953
Peer 7	0.1	0.1	0.1	0.1	0.1	0.0	1.0	0.0
Peer 8	0.3439	0.3439	0.5217	0.5217	0.5217	0.65132	0	1.0

Table 7.8: *Peers' Similarity Values.*

the protocol. Obviously, each peer ranks itself highest. This is trivially intuitive because each peer will receive the same email as itself and classify these emails exactly as it would. The final column aggregates the other columns and produces a ranking for the entire system.

Similarity

Share Pings were fired from the new classification event. This has ramifications for the similarity metrics. Active, early reporters will send more share pings and receive more share pongs than those peers who do not contribute new signatures. Sending more share pings means that other peers learn more information about a peer. Receiving more share pongs means that a peer learns more about the general state of the network. As a consequence of these two facts, it is to be expected that active, early reporters will have good views of the network and be ranked

Trust	**Peer 1**	**Peer 2**	**Peer 3**	**Peer 4**	**Peer 5**	**Peer 6**	**Peer 7**	**Peer 8**
Peer 1	1.0	0.56953	0.46856	0.61258	0.40951	0.71757	0.1	0.40951
Peer 2	0.0	1.0	0.3439	0.5217	0.0	0.71757	0.1	0.0
Peer 3	0.40951	0.40951	1.0	0.5217	0.61258	***0.59665***	0.1	0.5217
Peer 4	0.56953	0.56953	0.0	1.0	0.71757	0.74581	0.27	0.7458
Peer 5	0.0	0.0	0.5217	0.0	1.0	0.271	0.0	0.0
Peer 6	0.56953	0.5217	0.61258	0.71757	0.46856	1.0	0.0	0.56953
Peer 7	0.1	0.1	0.1	0.1	0.1	0.0	1.0	0.0
Peer 8	0.3439	0.3439	***0.39128***	***0.41736***	0.5217	***0.52106***	0.0	1.0

Table 7.9: *Peers' Trust Values.*

very highly by their peers. Table 7.8 bears this out. It is read by going across to find a peer and reading down to see its similarity values for the other peers.

Trust

The values for trust are the same for similarity, except where a peer revoked a signature that another peer recommended to it. In these cases, the value for trust is at most a quarter less than the similarity value. One quarter of the trust a peer has for another is lost upon a revocation, but this trust builds back up again as common signatures are shared. Table 7.9 shows the trust values at each peer and is read the same way as the preceding tables. The values that are different to the corresponding entries in Table 7.8 are highlighted in bold italics. There were six revocations in the system, but only four values for trust were affected. This is because two revocations were by the peer who created the signatures. These classifications were done in error and the revocation was simply restoring the emails to the inbox. Additionally, it makes no sense to decrease a peer's trust for itself.

7.3 Non-Empirical Evaluation

Cottereau [34] uses a number of non-empirical metrics to evaluate the Anti-Spam Network. This section uses these metrics to evaluate the proof-of-concept implementation of the CASSANDRA framework.

7.3.1 Scalability

Coulouris et al [20] give the following description of a scalable system:

> *A system is described as* scalable *if it will remain effective when there is a significant increases in the number of resources and the number of users.*

Cottereau defines the following list of efficiency metrics:

- **Administration overhead.** This is the amount of protocol messages exchanged between peers. Should this grow to be too many, the system cannot be considered functional anymore.
- **The mail-server load.** The message load on the server. If the network generates a stream of messages that overloads the server, then the system cannot be considered functional.
- **The processing power overhead.** The total power required from the machines on the network. If this power grows to unmanageable proportions, then the network cannot be considered functional.
- **The success rate.** The number of spam detected and filtered by the system. If this number grows too small, the system is not functional. This metric is somewhat subjective, given that it depends on user expectations. This applies equally to the number of false positives in the system.

7.3.1.1 Administration Overhead

In Section 7.2.3 , the protocol overhead for the implementation was shown to be in the range [23.1%, 42.3%] if the share pongs were removed from the protocol. This is in a scenario where each peer received approximately 50% spam in its non-protocol email, beginning from an empty knowledge base of classifiers. This means that the figures for the protocol overhead are inflated above what they would be under normal conditions. Even still, the protocol overhead is approximately equal to that of the spam load experienced by each peer.

It must be noted that a fuzzy hash was not used. This means that with a more robust hashing algorithm, the protocol overhead can be reduced by a factor of (number of spams/number of types of spam). This is potentially a very large figure. As remarked in Section 7.3, there are no protocol messages sent when no new types of spam are sent. This means that the administration overhead is zero while there are no mutations in the type of spam received. In perpetuity, this means that there will be long periods of zero protocol overhead with brief periods of activity every time spam mutates.

During the discovery process, there is a flurry of messages sent. There is an approximate upper bound on number of messages sent dictated by the TTL value set in the discovery ping. In a very large network, this will be localised to the peers that are clustered around the first node the discovery ping is sent to. The protocol overhead associated with the discovery process grows almost exponentially from zero to the upper bound set by the TTL as the network grows. Once this upper bound is reached, the number of users can scale without the protocol overhead increasing.

7.3.1.2 Mail-Server Load

In this implementation, the mail-server load is synonymous with the administration overhead because SMTP is the transport layer.

7.3.1.3 Processing Power Required

The user specifies the maximum number of peers and signatures to maintain at install-time. Each peer and signature that was maintained by peers in the experiment took up less than one kilobyte of disk space. If a user was to maintain a list of one thousand peers and ten thousand signatures, the disk space requirements would come to a measly eleven megabytes. This is an insignificant amount for desktops and is even acceptable for the current generation of PDAs.

Each incoming email has a signature computed on it and this signature is compared against the knowledge base of existing signatures. For a SHA1 or other cryptographic hash, this can amount to a discernible delay (2-7 seconds) for each incoming email. However, for a fuzzy hash, this will be reduced by several orders of magnitude (due to fewer distinct signatures) and should cause a delay that is imperceptible to the user.

7.3.1.4 Success Rate

The success rate can be defined as the percentage of incoming emails that are filtered exactly as the user wants them to be. This does not distinguish between false positives and false negatives. The exact value of this percentage that is acceptable to each user is entirely subjective, but should be as close to 100% as possible. It is expected that this value will increase over time as similar peers cluster together.

All users will never reach a 100% success rate. This is implicit because of the design of collaborative filters. Early receivers of the spam will mark it as such and report it to other users. Again, this value can be increased by using a robust fuzzy hash. As the number of users in the network scales, the labelling task on any one peer will be reduced, consequently increasing the success rate for each user.

Collaborative filters suffer from what is known as the "pump-priming problem". In other words, the success rate is very low for a small knowledge-base and small number of users, but increases dramatically as these values increase. This is a very powerful concept for distributed systems, because the effectiveness of this type of system actually increases with scale, rather than decreases.

7.3.2 Portability

Two metrics are used to address the evaluation of the solution's portability. These are openness and heterogeneity.

7.3.2.1 Openness

Coulouris et al [20] define openness in a computer system as:

> *. . . the characteristic that determines whether the system can be extended and reimplemented in various ways.*

The CASSANDRA framework is specifically designed to be extensible. This is achieved through the definition of XML marked-up objects and messages. These can be used to compose other objects and can themselves be extended to add functionality. The trust, clustering, discovery and ranking protocols and the transport layer are not mandated by the framework, which leaves numerous different implementations possible. Indeed, the framework allows many different types of topology and type of spam filter to be created and can be extended for use in other realms, such as entertainment recommendation.

7.3.2.2 Heterogeneity

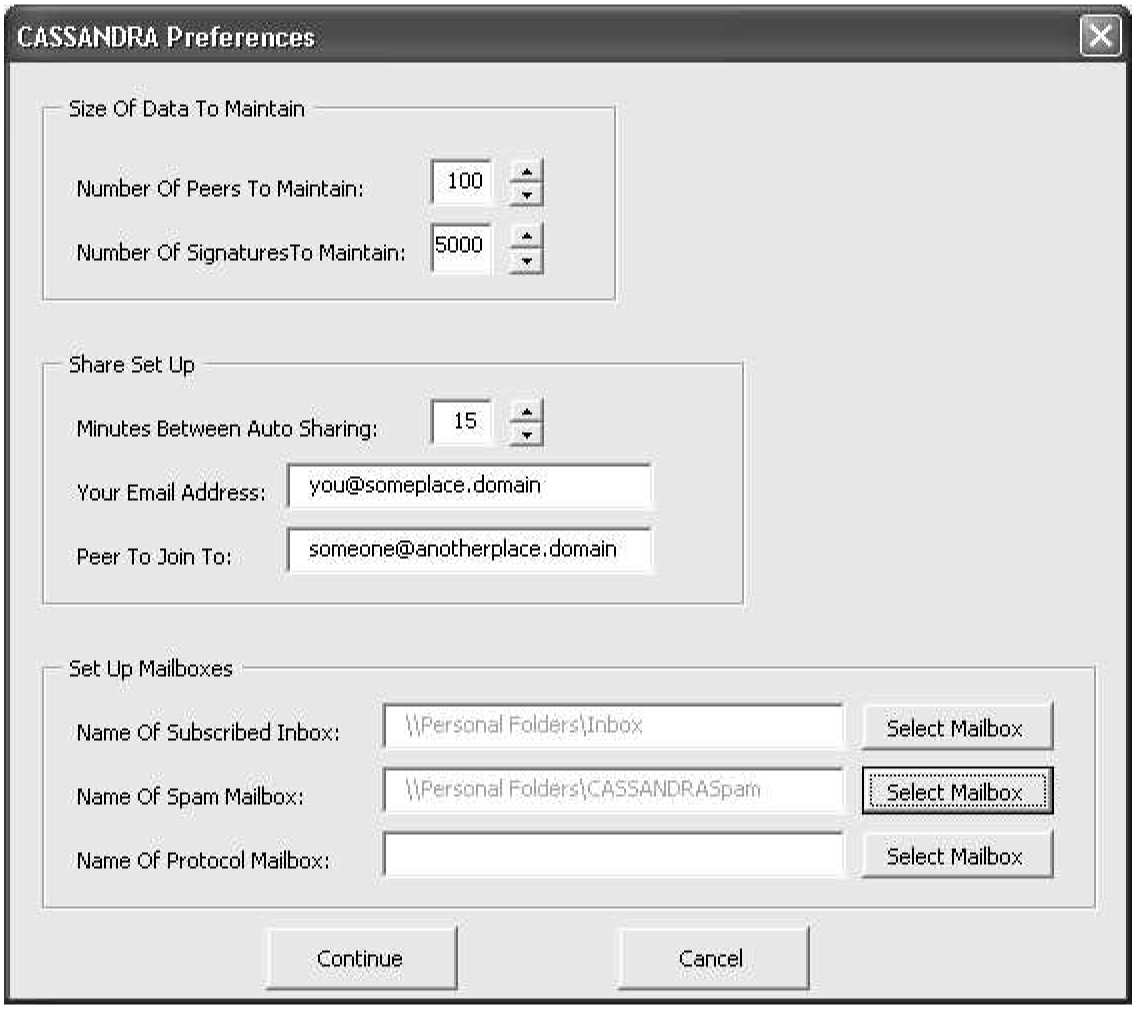
The CASSANDRA framework itself makes no determination on the technology used in possible implementations, except that inter-peer communication be conducted in XML marked-up messages. By design, this permits heterogeneity. The proof-of-concept implementation also enables heterogeneity. All inter-peer communication is done using XML marked-up messages over SMTP. As remarked previously in Section 5.2.3, virtually all programming languages that are used to send electronic mail can interpret SMTP. This means that anyone can participate in this proof-of-concept system as long as they can interpret XML and SMTP, and the back-end language is sophisticated enough to be able to perform the necessary computation.

7.3.3 Ease Of Use

The implementation fulfils all of the criteria set out for ease of use.

7.3.3.1 Installation And Configuration

Installation is comparatively straightforward. A user must import eleven classes into their VBA environment in MS Outlook. However, this can be simplified by changing the implementation so that it is an ActiveX plug-in, rather than a series of VB macros. This would mean the plug-in could be installed with a simple double-click on a self-extracting installer. Configuration is trivial. A user merely has to click on the "Set up CASSANDRA" button on the toolbar (see Figure 6.2). The dialogue shown in Figure 7.3 is presented to the user to configure the plug-in. The options are straightforward and intuitive. No additional configuration is necessary on the


CASSANDRA Preferences

Size Of Data To Maintain

Number Of Peers To Maintain: 100

Number Of SignaturesTo Maintain: 5000

Share Set Up

Minutes Between Auto Sharing: 15

Your Email Address: you@someplace.domain

Peer To Join To: someone@anotherplace.domain

Set Up Mailboxes

Name Of Subscribed Inbox: \\Personal Folders\Inbox

Name Of Spam Mailbox: \\Personal Folders\CASSANDRASpam

Name Of Protocol Mailbox:

Figure 7.3: *Screenshot Of Configuration Dialogue.*

MTA level, nor are any changes required to a firewall if the user is inside an enterprise-level firewall.

7.3.3.2 Marking And Revoking Spam

As described earlier in Section 6.4, marking and revoking spam is trivial and intuitive. They can both be achieved by highlighting an email and clicking the appropriate button.

7.3.3.3 Transparency To End-User

The process is almost entirely transparent to the end-user of the plug-in. They must be aware of how to install the macros. Once this is done, the rest of the process is transparent to the user. Protocol messages are automatically moved from the inbox to the trash folder once they have been processed. Spam emails are moved directly from the inbox to the spam folder without any intervention from the user.

7.3.4 Reliability

Reliability of the system is evaluated against false positive rate and the resilience of the system to subversion by attackers.

7.3.4.1 False Positives

As remarked in a number of places in this work, false positives are not considered to be indicative that someone is making false reports, merely the fact that they have a difference of opinion as to whether or not the email is something they wish to receive. In a small network where there is no view horizoning, the clustering, trust and ranking algorithms do not perform well. However, as the number of participants increases, it is possible to cluster the network to a finer degree of accuracy. Therefore, it is expected (but untested) that as the system scales, the false positive rate for any one peer will decrease.

7.3.4.2 Resistance To Attacks

The majority of foreseeable attacks have been catered for in the reference implementation. These will be reiterated here and how they are countered will be briefly explained. It is naïve to claim that all possible attacks have been anticipated, but the most likely attacks will not succeed.

- **Overload Of A Peer/Denial Of Service.** If a particular peer is especially active in the network, malicious attackers may wish to remove that peer by forcing them to opt-out by overloading them with many messages. This cannot be directly stopped. However, a peer can ignore protocol messages from a given user if they send more than a fixed number in a given time-frame. Additionally, a peer propagates new signatures to other peers as they are created. This means that no one peer can be singled-out, because the information is replicated across many peers as soon as it is created. This makes it virtually impossible to remove a valid signature from the network, unless everybody revokes it on their own machine or it gets dropped from the list of active signatures after a long period without hits.

- **Flooding False Positives.** Malicious peers might try to flood false positives in order to subvert the usefulness of the system. This is taken care of by design, because peers who produce false positives will get ranked more and more lowly by their peers until they are just ignored. Additionally, peers who produce the same false positives will tend to cluster together and so will remove themselves further from others in the network the more false positives they create.

- **Sender Spoofing.** Another attack could be for malicious peers to spoof other peers' addresses and make false reports on their behalf. This would result in inaccurate ranking and clustering and so subvert the system. In the current implementation, this is possible. Sender authentication and message integrity assurance can be added by migrating from SMTP to AMTP at the transport layer, or else using a Public Key Infrastructure (PKI) system for authentication. This can be achieved by using existing vendor products, as previously detailed. By doing so, any protocol email from an unauthenticated sender can be ignored, and the sender spoofing attack eliminated.

Chapter 8

Conclusion

This chapter summarises the conclusions drawn from the experiments presented in Chapter 7. Conclusions are drawn about the CASSANDRA framework as a whole. Some of the many future directions for this work are outlined. Finally, this chapter summarises and concludes the thesis.

8.1 Conclusions From Experimental Results

The following list summarises the conclusions drawn from the experimental results:

- The discovery protocol works. The joining node learns of all peers in the TTL horizon and every peer in the TTL horizon learns of the joining node.
- A joining node learns of all the signatures currently active in the TTL horizon during the discovery process. This is important, as it means that late joiners gain knowledge from the experience of the group.
- The discovery protocol is not optimal. There are a number of redundant messages sent during discovery. However, this is acceptable for the proof-of-concept implementation.
- Partially fragmented networks become more connected over time. This is because of the share pong that is sent in response to a share ping. It also leads to faster propagation of new signatures through the network.

- Share pongs are superfluous in a fully connected network. Therefore, a decision needs to be reached about which is more important - a fully connected network (keeping the share pongs), or low overhead (removing share pongs from the protocol).
- Shares are instigated when a new spam is classified. Therefore, when there are no new spams (or if new spam email hashes to the same value as a previously known spam), there is no message overhead due to the protocol.
- The amount of message overhead is dependent on how fuzzy the hashing mechanism is.
- Every peer knows of all active signatures within its view horizon. This indicates that peers recommend signatures to other peers even if they have not had a "hit" with the signature (i.e. peers recommend signatures based on the advice of their trusted peers).
- Even though the trust and similarity protocols were simplistic, they provide a good overall ranking of the peers in the network. These protocols can be replaced by more sophisticated ones, but these suffice to demonstrate their effectiveness in the proof-of-concept implementation.
- Active, similar, trustworthy and early reporters of spam get ranked highly by their peers.
- Lazy, untrustworthy, dissimilar peers tend to get ranked lowly by their peers.

En masse, these results show that the proof-of-concept works, i.e. that it has the capability to identify those peers most likely to receive similar spam and also those peers who are most likely to classify emails in a manner consistent to how the peer itself would. These are central to personalised, collaborative spam filters and demonstrate that the proof-of-concept implementation validates CASSANDRA as a framework for building personalised, collaborative spam filters.

8.2 Future Work

There are many aspects of the CASSANDRA framework that could not be evaluated within the scope of this project. These aspects present avenues for this research to be continued and expanded. The following represent a section through the various possibilities for future work, but is in no way exhaustive.

8.2.1 Development Of Network Topologies

Only a single topology was evaluated in the reference implementation of the CASSANDRA framework. The vanilla peer-to-peer topology proved effective, but is likely to have issues with network partitioning and view horizoning as the network scales. This leaves scope for implementations using more advanced topologies. An interesting direction would be to evaluate the performance of a super-peer-based P2P network where each of the super-peers are exemplars taken from different groups (i.e. users on the same spam lists) that have been identified in the network. Another interesting realm to create a network topology would be in a wireless environment where an extra dimension to "closeness" in the network is introduced - physical distance to hubs with peers on them. This could prove interesting, especially with mobility added to the considerations.

8.2.2 Distributed Trust And Reputation Management

There has been a lot of interest in developing distributed trust and reputation models for distributed systems and fair interest in developing same for peer-to-peer networks. However, all of these models assume both a highly connected network, with ample bandwidth and low network latencies. These will not apply well to a CASSANDRA implementation with SMTP/AMTP at the transport layer. Therefore, there is scope for development for a lightweight distributed reputation management system that can work effectively in a high-latency, disconnected network (potentially with few resources).

8.2.3 Group Clustering Using Data Mining Techniques

The clustering algorithm used in the proof-of concept implementation was rudimentary at best. It only defined one global group and ranked peers on how they best fit into this group. There is scope for further work in trying to identify disjoint subsets of peers who receive common spam, i.e. those on the same spam lists. Data mining techniques can be applied to the knowledge of which peers recommend which classifiers to a given peer. Once this is done, the best three or four exemplars of each group can be maintained in the list of top peers. This could be better than maintaining a list of very similar peers, because diversity in recommendation systems can prove very powerful (c.f. Condorcet's Jury Theorem).

8.2.4 Experimentation With Type Of Filter

A personalised, collaborative signature-based filter was implemented in this thesis as a reference implementation of the CASSANDRA framework. Other types of filter that are permitted by the framework can be developed and evaluated.

8.2.5 Development Of Fuzzy Hashes

Although not a direct continuation of this work, there is plenty scope for research into the development of fuzzy hashes for spam filtering. This should take the form of creating a general framework for text categorisation and feature selection. This framework should be able to create algorithms that can dynamically take a set of features and ignore them or concentrate on them (as appropriate) in a piece of text. This can then be applied to the particular sphere of spam filtering.

8.2.6 Improving Security In The Framework

Security has not been fully addressed in the reference implementation. A set of protocols for message integrity, sender authentication and encryption could be developed that can be used to augment SMTP and have a clear migration path to AMTP.

8.2.7 Exploration Of Application In Other Spheres

A personalised, collaborative spam filter where classifiers are shared between peers of similar spam and similar opinions on what is interesting is essentially a collaborative recommendation system. This means that the framework can be extended to take account of best practices in this field and the lessons learned in this thesis could be applied. Possible applications include personalised e-newspaper delivery, entertained recommendation or personal digital assistants in the form of intelligent agents.

8.3 Summary

This thesis presented a thorough study of spam filtering technologies and a comprehensive state of the art review of collaborative efforts at spam filtering. A framework that permits personalised, collaborative spam filters was presented and the various objects, messages, inter-peer communication and a reference structure for implementation was defined. One of the many possible implementations of a personalised, collaborative spam filter was described. The design and implementation of this proof-of-concept prototype was detailed. The prototype was a fully-decentralised, peer-to-peer, signature-based, personalised, collaborative spam filter. A rigorous evaluation of the proof-of-concept implementation of the CASSANDRA framework was presented.

Bibliography

[1] 105th Congress of the U.S.A. Digital Millennium Copyright Act. Pub. L. No. 105-304, 112 Stat. 2860 (Oct. 28, 1998), October 1998.

[2] A. Lambert. Analysis Of Spam. M.Sc. thesis submitted to the University of Dublin, Trinity College Dublin, September 2003.

[3] B. Sullivan. Who profits from spam? MSNBC News. http://www.msnbc.com/news/940490.asp?0sl=-43&cp1=1, August 2003.

[4] Bluetooth. The Official Bluetooth Wireless Info Site. http://www.bluetooth.com/, 2003.

[5] Brightmail Incorporated. 50 percent of Internet E-Mail is Now Spam According to Anti-Spam Leader Brightmail®. Brightmail Press Release, August 20th 2003.

[6] Business Week Online. Before Spam Brings the Web to Its Knees. Special Report: The Social Web, June 10th 2003.

[7] C. Shirky. What is P2P...And What Isn't. Published on the O'Reilly Network, November 2000.

[8] Cloudmark. Authority. http://www.cloudmark.com/products/authority/, 2003.

[9] Cloudmark. SpamNet. http://www.cloudmark.com/products/spamnet/, 2003.

[10] CNN International Business. DoCoMo wins spam-mail fight. http://edition.cnn.com/2003/BUSINESS/asia/03/25/docomo.reut/, March 2003.

[11] Coalition Against Unsolicited Commercial Email. http://www.cauce.org/about/problem.shtml, 2003.

[12] Compute Against Cancer. The Compute Against Cancer Homepage. http://www.computeagainstcancer.org/, 2003.

[13] Cybernothing. Where did the term Spam come from? The Net Abuse FAQ, Section 2.4. http://www.cybernothing.org/faqs/net-abuse-faq.html, 2003.

[14] D. D. Lewis and M. Ringuette. Comparison of two learning algorithms for text categorization. In Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval, pages 81-93, 1994.

[15] Detritus. The Infamous Monty Python Spam Skit. http://www.detritus.org/spam/skit.html, 1994.

[16] EuroCAUCE. Opt-in vs. Opt Out. http://www.euro.cauce.org/en/optinvsoptout.html, 2003.

[17] European Union. Concerning the processing of personal data and the protection of privacy in the electronic communications sector (Directive on privacy and electronic communications). Directive 2002/58/EC of the European Parliament and of the Council of 12 July 2002.

[18] Ezitrust. In-Person Certificate. http://www.ezitrust.com/products.shtml, 2003.

[19] Federal Trade Commission. False Claims In Spam - A report by the FTC's Division of Marketing Practices. http://www.ftc.gov/opa/2003/04/spamrpt.htm, April 2003.

[20] G. Coulouris, J. Dollimore and T. Kindberg. Distributed Systems - Concepts and Design. Addison-Wesley, Third Edition, 2001. ISBN: 0201-619-180.

[21] G. Padwick and K. Slovak. Programming Microsoft Outlook 2000. Sams Publishing, 2000. ISBN: 0-672-31549-1.

[22] G. Sakkis, I. Androutsopoulos, G. Paliouras, V. Karkaletsis, C. Spyropoulos and P. Stamatopoulos. A Memory-Based Approach to Anti-Spam Filtering for Mailing Lists. Information Retrieval, 6, pages 49-73, 2003.

[23] G. Vaudreuil. RFC 3030: SMTP Service Extensions for Transmission of Large and Binary MIME Messages, December 2000. Obsoletes: 1830. Status: STANDARD.

[24] Gnutella. The Gnutella Homepage. http://www.gnutella.com/, 2003.

[25] Habeas. Warranted Sender Email. http://www.habeas.com/, 2003.

[26] I. Androutsopoulos, G. Paliouras, V. Karkaletsis, G. Sakkis, C. Spyropoulos and P. Stamatopoulos. Learning to filter spam e-mail: A comparison of a naive bayesian and a memory-based approach. In Workshop on Machine Learning and Textual Information Access, 4th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD), 2000.

[27] I. Androutsopoulos, J. Koutsias, K. Chandrinos, G. Paliouras and C. Spyropoulos. An Evaluation of Naive Bayesian Anti-Spam Filtering. In Proceedings of the workshop on Machine Learning in the New Information Age, 2000.

[28] I. Clarke. The Free Network Project Homepage. http://freenet.sourceforge.net/, 2003.

[29] IronPort. Bonded Sender Homepage. http://www.bondedsender.com/, 2003.

[30] J. Klensin (Editor). RFC 2821: Simple Mail Transfer Protocol, April 2001. Obsoletes: 821, 974, 1869. Status: STANDARD.

[31] J. Mason. The SpamAssassin Homepage. http://spamassassin.org/index.html, 2003.

[32] J. R. Mastaler. Tagged Message Delivery Agent (TMDA). http://tmda.net/, 2001.

[33] J. Roberts. Habeas in Practice: Ugly But Effective. Ezine-tips.com List Management tips http://ezine-tips.com/articles/management/20021029.shtml, October 2002.

[34] L. Cottereau. A Peer-to-Peer Architecture for Collaborative Spam Filtering. M.Sc. thesis submitted to the University of Dublin, Trinity College Dublin, September 2002.

[35] L.Gonze. Gnutella is Not O(N^2). http://www.thefengs.com/wuchang/work/cse581_winter2002/papers/gnutellan2.html, March 2001.

[36] M. Sahami, S. Dumais, D. Heckerman and E. Horvitz. A Bayesian Approach to Filtering Junk E-Mail. AAAI Technical Report WS-98-05. Workshop on Learning for Text Categorization, July 1998.

[37] M. Schlosser, M. Sintek, S. Decker and W. Nejdl. HyperCuP Hypercubes, Ontologies and Efficient Search on Peer-to-peer Networks. International Workshop on Agent and Peer-to-Peer Computing, Bologna, Italy, 2002, 2002.

[38] Mail Abuse Prevention System. Definition of "spam". http://www.mail-abuse.org/standard.html, 2003.

[39] Mail Abuse Prevention System, LLC. Realtime Blackhole List. http://www.mail-abuse.org/rbl/, 2003.

[40] Message Labs. Intelligence Monthly Report, July 2003.

[41] N. Freed. RFC 2920: SMTP Service Extension for Command Pipelining, September 2000. Obsoletes: 2197. Status: STANDARD.

[42] Napster. The Napster Homepage. http://www.napster.com, 2003.

[43] National Office for the Information Economy. SPAM - Final Report of the NOIE Review of the Spam Problem and how it can be countered, April 2003.

[44] Nucleus Research. Spam: The Silent ROI Killer. http://www.nucleusresearch.com/research/d59.html, July 2003.

[45] P. Cunningham, N. Nowlan, S.J. Delany and M. Haahr. A Case-Based Approach to Spam Filtering that Can Track Concept Drift. TCD Computer Science Technical Report TCD-CS-2003-16, May 2003.

[46] P. Hoffman. RFC 2487: SMTP Service Extension for Secure SMTP over TLS, December 2000. Status: STANDARD.

[47] P. Hoffman, Internet Mail Consortium. Unsolicited Bulk Email: Definitions and Problems. http://www.imc.org/ube-def.html, October 1997.

[48] P. Resnick (Editor). RFC 2822: Internet Message Format, April 2001. Obsoletes: 822. Status: STANDARD.

[49] J. Postel. RFC 821: Simple Mail Transfer Protocol, August 1982. Obsoletes RFC0788. Status: STANDARD.

[50] Rhyolite Software. Distributed Checksum Clearinghouse. http://www.rhyolite.com/anti-spam/dcc/, 2003.

[51] J. Ritter. Why Gnutella Can't Scale. No, Really. http://www.darkridge.com/ jpr5/doc/gnutella.html, February 2001.

[52] S. E. Fahlman. Selling Interrupt rights: A way to control unwanted e-mail and telephone calls. IBM Systems Journal, Volume 41, No. 4, pages 759 - 766, 2002.

[53] S. Mosher. Microsoft Outlook 2000 Programming in 24 Hours. Sams Publishing, 1999. ISBN: 0-672-31651-X.

[54] S. Radicati and M. Khmartseva. The IT Cost Of Spam. The Messaging Technology Report, Volume 12, Number 8, August 2003.

[55] SETI@home. Search for Extraterrestrial Intelligence At Home. http://setiathome.ssl.berkeley.edu/, 2003.

[56] Spam Abuse Net. What is spam? http://spam.abuse.net/overview/whatisspam.shtml.

[57] SpamAssassin. Release Documentation. http://spamassassin.rediris.es/full/2.5x/ dist/README.

[58] State Of Connecticut. Section 53-451 Computer Crimes (b) (7). General Statutes Of Cnnecticut, Title 53. Crimes, Chapter 949g. Computer Crimes (Added by P.A. 99-160, approved June 23, 1999, effective October 1, 1999), June 1999.

[59] State Of Delaware. Section 937. Un-requested or Unauthorized Electronic Mail or use of network or software to cause same. Delaware Code, Title 11, Section 937 and 938 (1999), July 1999.

[60] State Of Virginia. Transmission of unsolicited bulk electronic mail; penalty. Virginia Code, Title 18.2 Chapter 5, Article 7.1, Section 18.2-152.3:1,(2003), April 2003.

[61] Sun Microsystems. The Project JXTA Homepage. http://www.jxta.org/, 2003.

[62] J. Paoli T. Bray and C. M. Sperberg-McQueen (Eds). "Extensible Markup Language (XML) 1.0 (2nd Edition)". W3C Recommendation, 2000.

[63] T. Cover and P. Hart. Nearest neighbour pattern classification. IEEE Transactions on Information Theory 13, pages 21-27, 1967.

[64] T. Geller. Cost of spam vs. direct marketing. SpamCon Foundation newsletters, Issue 8, August 2001.

[65] Thawte. Personal Email Certificate. http://www.thawte.com/html/COMMUNITY/ personal/index.html, 2003.

[66] The SpamHaus Project. Mailing Lists -vs- Spam Lists. http://www.spamhaus.org/mailinglists.html, 2003.

[67] The SpamHaus Project. The Definition Of Spam. http://www.spamhaus.org/definition.html, 2003.

[68] The SpamHaus Project. The SpamHaus Homepage. http://www.spamhaus.org, 2003.

[69] V. V. Prakash. Razor. http://razor.sourceforge.net/.

[70] V. V. Prakash. Razor v2 Documentation. http://razor.sourceforge.net/docs/whatsnew.php.

[71] W. Weinman. AMTP - a replacement for SMTP. http://amtp.bw.org/, 2003.

[72] W. Weinman. AMTP - Authenticated Mail Transfer Protocol. IETF Network Working Group Internet-Draft, August 2003. Status: DRAFT.

[73] X. Carreras and L. Marquez. Boosting Trees for Anti-Spam Email Filtering. Proceedings of RANLP-01, 4th International Conference on Recent Advances in Natural Language Processing, 2001.