FreeBSD server anti-spam software using automated TCP connection control

Minh Tran
Centre for Advanced Internet Architectures. Technical Report 040326A
Swinburne University of Technology
Melbourne, Australia
minhtran@swin.edu.au

***Abstract* – This paper describes a new approach to anti-spam techniques.**

Instead of having spam filtering software implemented at a mail client or server, we build a mail server agent called MT Proxy to deal with any spam related issue. This server acts a mediate firewall to securely protect our real SMTP mail server. All emails are allowed to come to their recipients, but at either slow or fast speed, depending on their spam characteristic.

A key advantage of this prototype is having a system, which is capable of controlling, but not blocking email traffic. As a result, MT Proxy software is guaranteed not to make any spam decision mistake. On the other hand, the software can effectively slow down email connection from spam sources; which in long term causes an economic loss for spammers.

Keywords- spam, MT Proxy, FreeBSD, SMTP, false positive, false negative, white list, black list, challenge-response, Bayesian filter, dummynet, DNSBL, inetd.conf.

I. INTRODUCTION

Spam is an Internet term for unsolicited junk emails, which are sent to multiple users with the same content to promote products and services [27]. Once you are targeted by spam, even though you are not interested in its dubious advertised products, you still have to pay your own "pocket money" to receive it. Spam definitely wastes your time and money. It can flood your inbox entirely and contain illegal corrupting materials.

In a continuous battle of fighting against spam, many techniques have been developed and improved. They are generally classified into 2 main categories: listing (white or black listing) and filtering (email analysis). No matter what anti-spam method is used, none of them guarantees 100% accuracy. There is always a trade-off between "false positive" (over blocking) and "false negative" (under blocking) in any filtering software. With false positive, legitimate emails are treated as spam, thrown in junk folders or even deleted by ISPs. False negative on the other hand allows spam to get into receivers' inbox.

In order to protect customers from an onslaught of spam, some ISPs have implemented more aggressive spam identification methods [2]. Consequently, many legitimate emails, including important business ones cannot reach their recipients. A research done by Ferris Inc. has shown a loss of exceeding $50 per person per year and $3.5 billions per a U.S. business in 2003 as a negative effect of false positive [3].

Another common shortcoming feature in most spam filtering software programs is that they normally allow spam to get into email spools before any classification process happens. This behavior leaves no painful impact on spammers. Marty Lamb says we want to cause spammers pain [1]. We want a software program, which is capable of slowing down real-time Internet traffic experienced only by spammers.

Our initial research is to develop a spam filtering software program for a Unix-based (FreeBSD) server to control email traffic based on senders' spam probability. The software called MT Proxy is written in C/C++ language. It scans email messages in real time, analyses messages' spam probability and uses FreeBSD dummynet to slow down TCP traffic from spammers.

II. BACKGROUND

A. Spam and False Positive:

Dictionary.com defines spam as "Unsolicited e-mail, often of a commercial nature, sent indiscriminately to multiple mailing lists, individuals, or newsgroups; junk e-mail" [5]. Thus spam has two distinct characters: no verified permission from recipients (unsolicited) and being delivered to many people with identical content. Spamhaus states "A message is Spam only if it is both Unsolicited and Bulk" [6].

Several filtering software programs identify only one spam character in a normal email but immediately classify that email as spam. We have experienced that some emails whose recipients are in mailing lists or not in "to:", "cc:", "bcc:" section of email envelops are directed by Hotmail into junk folders without any further question. Hotmail needs to reconsider their spam classification criteria. Obviously, their software mistreats a message as spam though it only sees the second spam character (bulk). Hotmail is one of many cases, in which anti-spam programs make wrong spam categorization decisions, namely "false positive".

As mentioned in the introduction section of this paper, there is a trade off between false negative and false positive. Having a more aggressive anti-spam method is equivalent to sacrificing more legitimate emails (false positive). However, reducing this aggressiveness level can allow more spam get into recipients (false negative). Thus, ISPs need to make a compromise between these two issues and improve their anti-spam software with more accurate spam identification techniques.

From our perspective, false positive is a more serious problem compared to its counter-part, false negative. Though false negative allows spam get into customers' mailbox, wastes their time and resources, it is still tolerated. On the contrary, false positive can result in losing important personal/business emails; which apparently would be harder to be accepted by customers.

Nowadays, as the Internet becomes cheaper and more popular, many people want to utilize its powerful communication network as a fast and easy way to make money by sending spam to random people. Most US email users say that up to 30% of messages they receive are spam whilst 39% say they receive more than that; including 18% who say that at least half their email message is spam. (A report by Gallup Poll-2001) [28]. The increasing popularization of spam forces some ISPs to implement harsher anti-spam techniques. As a result, legitimate-email-blocking-rates of these ISPs also increase considerably. A research done by www.returnpath.com in 2003 finds that there was a 17% percent chance of false positives caused by top ISPs anti-spam software, up compared with only 2% in fourth quarter of 2002 [2].

It is hard to deny that a real challenge of any anti-spam software is that not only how much spam it can catch but also how much spam it can catch without blocking a significant amount of legitimate emails. That is to reduce both false negative and false positive to a very small number [7].

B. Anti-spam Methods:

There are generally two main techniques to deal with spam: white/black-listing and rule-based filtering.

1. White/black-listing method:

Listing method is the most basic method for anti-spam software. In this method, the software will maintain databases of good and bad IP addresses (white and black list respectively).

A white list contains trustworthy IP addresses whilst a black list consists of addresses believed to be of spammers. Emails from senders in the white list are passed to mail inboxes while emails whose sender addresses are in the black list are deleted or directed to junk folders.

White-listing method has more access restriction than black-listing since it only allows email coming from specific reliable sources. Thus, white-listing is normally used at a mail client to let the client flexibly set his/her favorable addresses.

Black listing is more widely implemented; especially at ISP mail servers since this method only blocks email from a defined black list. It is preferable because it allows everyone on the Internet, who is not a spammer, to contact recipients. In this method, the software scans an email by comparing the email's IP address with that in a local black list file or by querying Internet black list databases (such as: SpamHaus [13], SORBS [14], ORDB [16]) in real-time.

Many software vendors now implement a combination of white and black listing method to have a more accurate and flexible spam identification product. They implement an additional mechanism called challenge-response, which allows an automatic white/black-listing categorization process. In this challenge-response scheme, when an email is scanned inside anti-spam software, there are three possibilities: its IP address is in a white list, in a black list, not in any of these lists. If its address is not in either white or black list, its sender is challenged with a reply message. Only if the sender replies to this challenge, he/she is put in the white list and the email is allowed to reach its recipient. Examples of some products that support both white and black listing are Mail Gate of Corvigo, ASG of Mail Frontier, Perimeter of Postini, Email Thread Management Service of MX Logic, Email Protection Service of Singlefin [29].

Spammers often use fake email addresses when sending spam. They also use automatic software for generating spam. This kind of software does not normally reply to its email responses. Thus, if senders reply to the challenge email, they are assumed by the anti-spam software not to be spammers. As a result, the challenge-response technique provides an automatic mechanism of spam categorization for a passing/blocking decision [4].

Even though challenge-response has advantages from the flexible implementation of white/black-listing and convenient provision of an automated classification process for unknown emails, there is no guarantee that this method will not make mistakes. If spammers know the challenge-response rule, they can generate an automatic email reply scheme to entirely get around the anti-spam system. This possible danger needs to be seriously taken into account by anti-spam vendors. As a consequence, besides white/black-listing functions, many anti-spam products also provide rule-based filtering functions to actually look inside email content before making any decision.

2. Rule-based filtering method:

In rule-based method, email header and its content are examined by a filter. This filter will base on some "rules" or keywords to determine whether an email comes from a spam source. It is necessary to keep the

rule up to date and to implement a suitable algorithm for spam identification [4].

An important algorithm used in this method is the Bayesian technique. Bayesian software builds a dictionary of spam/non-spam, decodes the email message and calculates statistic spam probability [8]. More information about mathematics of Bayesian statistics can be found in a Stanford University website [31].

The Bayesian technique works on a basic assumption that most events are dependent and a future occurrence probability of an event can be determined from its occurrence in the past. This assumption is applied for spam identification. If a word is contained in many spam emails; say about 500 emails out of 1001; that word is likely to be a spam word and is assigned a large number for its spam probability.

For example, if the word “bonus” appears 500 times out of 1001 spam emails and only 10 times out of 1002 non-spam emails, its spam probability would be:

```
(500/1001)/(500/1001+10/1002)  =  0.9804  =  98.04%
```

Thus having large databases of spam emails and non-spam emails, we can construct a table of spam probability (guesstimate probability) for each word. When an email arrives at the filter, it is tokenized into single words. Each word is compared with spam/non-spam databases. If the word is found in the databases, its spam probability is returned. Otherwise, it is assigned a certain chosen value. These values are used to calculate a final spam probability of an email. If the probability is greater than some threshold, the message is classified as spam and deleted or thrown in a junk folder.

Paul Graham’s approach is quite famous for spam filtering using the Bayesian technique [17]. In his method, he builds two different large databases: one corpus of spam (*bad*) emails and another of non-spam (*good*) emails. Two harsh tables are built for these two corpuses, in which he counts a number of occurrences for each word in each corpus. After this has been done, he constructs another harsh table for the combined spam probability of each word by referring from both *good* and *bad* harsh tables. If a word appears in one corpus, but not in another, it is assigned a value of 0.01 and 0.99 (decided by trial and error). When an email arrives, its most fifteen *interesting* tokens are used to calculate a final email spam probability. (*Interesting* is determined by how far the word probability is from a 0.5 neutral).

Gary Robinson argues that Paul Graham’s approach is extreme and asymmetric [18]. He suggests an improvement by calculating both compound spaminess (P) and non-spaminess (Q) value of an email. A final spam indicator (S) is computed based on P and Q. The calculation is as below:

If p(w) is the guesstimate probability of each word.

$$P = 1 - ((1 - p_1)(1 - p_2)\ldots(1 - p_n))^{1/n}$$

$$Q = 1 - (p_1 p_2 \ldots p_n)^{1/n}$$

$$S = ((P - Q) / (P + Q) + 1) / 2$$

S lies from 0 to 1 and; as Gary Robinson believes; is symmetric in its treatment of spam-indicating and non-spam-indicating words.

C. Simple Mail Tranfer Protocol (SMTP):

1. Introduction:

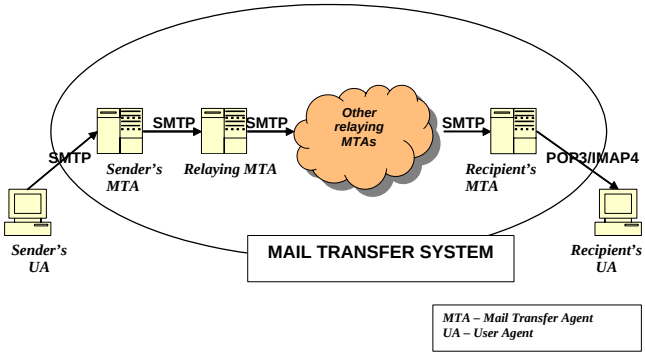

Figure 1: A basic model of an Email Transfering System

An email message relies on SMTP for transferring from a sender mail client to his/her mail server, then to other mediate mail servers until this email reaches a recipient mail server.

Communication from the mail client with client's SMTP server is established on a standard port 25 of a TCP connection. After the server accepts to transfer an email from client, the client notifies the server with necessary transferring information.

SMTP is an end-to-end delivery system where emails are kept in email spools at receivers' mail server. The servers will then try to deliver emails to their recipients at a specific time. Emails that cannot be delivered will be returned to their senders.

RFC 822 defines a standard for email format. It considers an email as having two parts: an envelope and content. An envelope contains information for posting an email to its recipient. This information includes sender' email address, receiver email address and delivery mode. Email content consists of a header that is automatically added by sender SMTP server and a body, which is an actual message to be delivered [9].

RFC 821 [10] defines an email transmission standard. A user agent (UA) sends user's email message to its Internet mail server (also called Mail Transfer Agent - MTA). The MTA in turn delivers the email message to other mediate MTAs until this message reaches recipient MTA. Each MTA will

need to contact its Domain Name Server (DNS) before packet delivery process. The DNS returns with either an IP address of the recipient's MTA or other relaying mediate MTAs. Once the IP address is known, connection from a MTA to a MTA is established.

2. SMTP transaction:

-A SMTP transaction goes through these following basic steps [10] [20]:

-A client (a sender UA or a mediate MTA) establishes a TCP connection with a server (another mediate MTA or a recipient MTA) on port 25.

-The server responds with a message code of 220 (ready).

-The client sends a Hello command

```
(HELO + client host name or domain).
```

-The server responds with a message 250 + receiver domain. This means connection is open and ready to go.

-The client sends a Mail from command

```
(MAIL <space> FROM:<reverse-path><CRLF>).
```

A reverse-path starts with the client's nearest MTA, then other mediate MTAs and finally original sender email address. This reverse-path is modified by each MTA as the email goes along the path.

-The server responds with a message 250 OK to acknowledge its client's email sources.

-The client sends a Recipient to command

```
(RCPT <space>TO:<forward-path><CLRF>.
```

A forward-path contains routing addresses of the next MTAs ending by the absolute email address of the recipient.

At the server (receiving MTA), as it processes transferring the email, the first entry in the forward-path is added into the first entry of the reverse-path. If it cannot identify the first entry in the forward-path, it uses this information to find the next MTA to relay the message. In this case, it keeps the first entry in the forward-path and adds its own entry to the reverse-path. In both case, the server responds with a message code of 250 OK to confirm the relaying process.

However, if it cannot find any path to the recipient, it will reply with a 550-message code. It then constructs an undeliverable message and sends this back to the message originator following the reverse-path.

-After the server confirms that it knows where to relay the message, the client issues a DATA command.

-The server responds with a message code of 354 to tell the client that it is ready to receive the message.

-The client sends email content by adding email header information and message body. A single dot "." is entered to tell the server the end of the message.

-The server responds with an acknowledgment 250 OK.

-The client can start sending a new email by resuming a Mail from process or end the connection by issuing a QUIT command.

-The server responds with a 221-message code to confirm that transaction is completed.

3. SMTP server's response code:

The server responds back to its client's command with a message code of 3 digits ABC [21]:

The first digit A indicates status of the email:

-1-3: success

-4: temporary negative

-5: failure

The second digit B indicates categories of the message's status:

-0: syntax

-2: connection

-5: mail

The third digit C is about a specific message in the above categories.

4. Email retrieving:

After an email has traveled a long way through different MTAs, it finally reaches its recipient's post office. This message store is a large directory that contains many sub-directories (or email spool) for each individual user. The sub-directory in turn divides into many sub-directories, which essentially are user mailbox (inbox), draft, sent message and trashcan. Some ISPs also provide an optional junk sub-directory to store spam messages.

Basically, a user's mailbox has two main sub-folders for read and unread messages. User can access and read their emails in different ways. There are three different access models: offline, online and disconnected [22].

Offline access model is the most basic model for email retrieving. Messages stored in an email spool are retrieved by mail clients such as Kmail, Mozilla, Netscape, Outlook Express... After these email messages are downloaded and stored in user hard drives, users can process them locally. This model has an advantage of being easy to read and process emails as easy as accessing local files on user computers. It also saves users' Internet bill since they are only charged for a period of emails downloading.

In online access model, emails messages are stored and processed on a mail server. Users connect to their mail server in real time, type in their username and password to be able to retrieve and read their messages. The advantage of this model is

flexibility of accessing emails no matter where you are. Because the emails are stored on server, they also do not occupy spaces on your hard drives. Typical examples of this model are web front-ends such as Yahoo and Hotmail, which support their clients to retrieve emails online.

The last access model is disconnected which is a hybrid of offline and online model. This model allows users to retrieve and store their emails locally as in offline model. The difference is that in disconnected model, any change made to a local version of an email is updated by the mail server, which also stores a copy of this email.

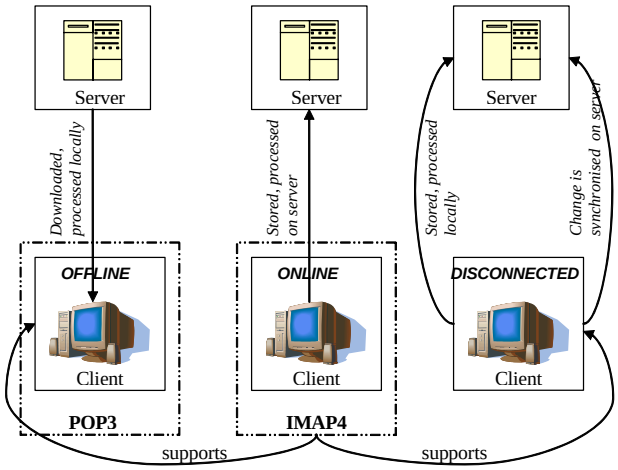

Figure 2: Three email acess models

5. Post Office Protocol Version 3 (POP-3) and Internet Message Access Protocol 4 (IMAP-4):

POP3 is a popular and basic Internet standard based protocol used for email retrieving in offline model. A POP3 server listens for TCP connection on port 110. Some basic commands to communicate with a POP3 server are:

```
user – name of user retrieving emails
pass – a string of password
list – lists individual message size
stat – shows total message size
quit – close the connection
```

For more information about POP3, please refer to its Request for Comments (RFC) 1725 [23].

IMAP4 is a more advanced standard based-protocol [22]. It has everything that POP3 does not have. IMAP4 can be used in all access models. It also provides authentication and multiple server communications. In IMAP4, a client can concurrently talk to multiple servers. IMAP4 also provides extra options for clients, such as the abilities of creating, handling different email messages and folders. With stronger password authentication, IMAP4 is more reliable and is raising its importance in mail client implementation market. More information on IMAP4 standard can be found at its RFC 2060 [24].

D. Dummynet and bandwidth control:

Dummynet is FreeBSD's kernel resident functionality for adding configurable latency or bandwidth limits to different IP packet flows. It can be used on a FreeBSD workstation or a FreeBSD computer acting as a router to control traffic going through these machines.

In dummynet, a pipe is created from a source to a specific destination. This pipe is configured with desired parameters, such as: bandwidth, delay, queue size, packet loss, mutipath for controlling IP/TCP traffic as indicated in a dummynet rule (ipfw) [12].

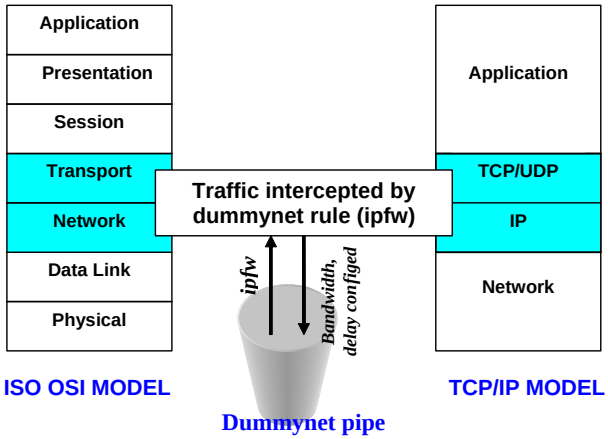

Figure 3: Dummynet in related to TCP stack

Packets are intercepted by a dummynet rule as they travel along the protocol path. These packets are passed through pipe objects, which can be individually configured to add different delays, bandwidth limits to different IP/TCP traffic [11].

To enable dummynet in a FreeBSD kernel, the following commands are used:

```
root#kldload ipfw.ko
root#kldload dummynet.ko
```

A pipe is a virtual fixed bandwidth channel. Traffic capped in this pipe will conform a dummynet rule set for this channel. A pipe is created with following commands:

- Pipe creation for ip connection:

```
root#ipfw add rule_number pipe pipe_number ip from host_ip_address/mask to host_ip_address/mask in/out
```

- Pipe creation for tcp connection:

```
root#ipfw add rule_number pipe pipe_number tcp from host_ip_address/mask port_number to host_ip_address/mask port_number in/out
```

The configuration of pipe parameters is as below:

```
root#pipe pipe_number config bw channel_bandwidth delay delay_time queue queue_size prl random_packet_loss
```

To delete a dummynet pipe, following command is used:

```
root#ipfw pipe pipe_number delete
```

To delete all current pipes, this command is issued:

```
root#ipfw pipe flush
```

To view all current ipfw rule and pipe configuration, following commands are used:

```
root#ipfw list

root#ipfw pipe show
```

Dummynet is a powerful traffic management tool. It is implemented in MT Proxy to efficiently reduce bandwidth and increase delay of spammers' traffic. Dummynet rules are executed as kernel's shell script commands embedded in C/C++ code for MT Proxy.

III. METHODOLOGY

A. Research Overview:

The first part of the project is to build a prototype of a SMTP proxy server which can filter and control email traffic from senders before it forwards email messages to a real SMTP server.

Any email sent from outsides will go through our MT Proxy without knowledge of a real SMTP server. This allows early detection of spam as well as reduces load and possible harms on our real email server.

As discussed in the introduction and background section of this paper, false negative and lack of capability of causing spammers pain is a common limitation of most anti-spam software programs. Some studies have shown considerable loss for personals and businesses due to false positive problem [2] [3]. However, to solve this problem we do not want to reduce an aggressiveness of spam identification techniques since this could inefficiently lead to false negative problem. Instead, we want software that never blocks any legitimate email and capable of slowing down the spammers' connection, consuming their time and resources.

MT Proxy software is written to never block any email. It only slows down traffic which it believes come from spammers. Only Spammers' TCP connection will experience remarkable delay and bandwidth limitation. This delay is proportional to a final compound analysis result of a listing and filtering technique.

B. Spam Analysis methods used in MT Proxy sotfware:

Our software uses both blacklisting and filtering method for spam identification.

1. Blacklisting method:

The software is written to be able to query both a local internal black list file as well as Internet databases in real-time.

In the local black list database, administrators can add, modify IP addresses and host names, which they believe, are sending spam.

Our MT Proxy's Domain Name Server (DNS) list is configured with addresses of some Internet Domain Name Server Black Lists (DNSBL), such as: SpamHaus, Spam and Open Relay Blocking System (SORBS), Open Relay Database (ORDB) [13][14][16].

Administrators can flexibly modify this list, add more DNSBLs. They can also assign an individual trustiness number for each DNSBL server and its local black list file. This trustiness number is used to prioritize DNSBL databases that administrators believe to be more accurate in identifying spam sources.

An IP address of client is inversed, attached in front of DNSBL host names before being sent to an Internet DNS server for spam source queries. If the DNSBL server does not find this address in its black list, it replies with the same address, otherwise it replies with an address of 127.0.0.2 indicating that this address comes from a spammer.

2. Filtering method:

MT Proxy uses a very simple algorithm for calculating spam probability of an email message.

Email content is scanned through the filter. Each word in the email is computed with their guesstimate probability. A number closed to 0 (non-spam) or a number closed to 1 (spam) is returned for each word.

According to a famous article "A Plan for Spam" by Paul Graham [17], each word in the email is associated with a guesstimate probability which is calculated from two different databases: one corpus of spam emails (*good*) and one corpus of non-spam emails (*bad*). He also creates three harsh tables: one for a number of occurrences of each word in the *good* corpus, one for a number of occurrences of each word in the *bad* corpus, and one for a final computed guesstimate probability of each word.

MT Proxy only uses one database of *bad* (spam) words. It tokenizes the email messages and checks if these tokens are in the spam database. The guesstimate probability of each email word is simply assigned a number closed to 1 (0.9999) if this word is found in the spam database. Otherwise, it is assigned a value closed to 0 (0.0001).

A combined spam probability of the entire email is calculated by applying Gary Robinson's method [18] in which a compound spaminess (P) and non-spaminess (Q) are both used to calculate a final spam indicator (S) value.

If p(w) is the guesstimate probability of each words.

$$P = 1 - ((1 - p_1)(1 - p_2)\ldots(1 - p_n))^{1/n}$$

$$Q = 1 - (p_1 p_2 \ldots p_n)^{1/n}$$

$$S = ((P - Q) / (P + Q) + 1) / 2$$

S lies from 0 to 1.

Spam percentage is calculated by: S * 100 * 0.5 (where 0.5 is a weight (priority value) for filtering method compared to black-listing method).

Spam word database used in MT Proxy is in Appendix A of this paper. For an example of spam calculation in MT Proxy, please refer to Appendix B.

C. Dummynet rule in MT Proxy:

The first time when a spam analysis process (either black-listing or filtering) returns a value greater than 0, a dummynet pipe is added by executing a kernel's shell commands. The following code is used in MT Proxy software for adding a new pipe:

```
sprintf(docommand_string,"ipfw add %d pipe %d tcp from %s %d to any 6543 in", pipe_number, pipe_number, client_ipaddress, ntohs(client_port));

docommand(docommand_string);
```

ipfw only intercepts email traffic coming into MT Proxy server (of default port number 6543). The pipe is created with an assigned global variable *pipe_number* for each client connection.

Every time when a filtering analysis result is greater than a current worst spam value, the above pipe configuration is updated with a new worst spam. The following code shows bandwidth limitation (in Kbits/s) and delay (in ms) as a function of worst spam:

```
sprintf(command,"ipfw pipe %d config bw %dKbit/s delay %d0ms ",pipe_number,(112-spam)/2,spam);

docommand(command);
```

D. MT Proxy's functionality:

The program is written by extending an open-source Unix-based SMTP proxy server [19]. We have modified the program by adding a black listing (with both local and Internet DNSs) and filtering method for spam analysis. We have also added bandwidth/delay control functionality using dummynet and made MT Proxy be an `inetd.conf`-independent server.

Client emails from UAs or MTAs are directed to our MT Proxy. All email traffic is controlled at MT Proxy. Before reading email messages, MT Proxy first checks to see whether client IP addresses are from either a local or Internet black list. If it finds the addresses in any black list, it triggers dummynet (ipfw) to decrease bandwidth, increase delay of only TCP connection from these clients. Ipfw is only used when there is any detection of spam. If clients are not in black list, there is no dummynet rule set on clients' connection at this stage.

After a client starts sending email content, MT Proxy performs a scanning process to look through the entire email message. It then computes a statistic spam number for each email. This spam number is used to further reduce bandwidth and add more delay of TCP connection from the client.

By default, for a final spam number, black-listing and email filtering method contributes the same weight of 0.5 (both methods add up to a unity weight). This sets a certain spam, delay and bandwidth reduction limit for each method.

A result of email filtering process is done after every 5 lines of client input. This result is accumulated until the entire email is read. If this number reaches a limit set for filtering method, there is no more update on spam, delay and bandwidth reduction.

The filtering result and dummynet rule are updated only if after each analysis, a current spam number is larger than previous ones. This “worst-spam memory” characteristic reduces the load on the server since it does not have to execute dummynet shell commands every single analysis.

The client’s TCP traffic control is done in real-time as the client is transferring its email content to the server. This makes sure that as soon as the software discovers any spam in client's input (default setting is 5 lines), the client will experience a significant disadvantage of bandwidth reduction and slower TCP connection. This immediate action efficiently consumes spammers' time and resources though the software still delivers their emails to recipients.

After the client finishes connection with the server, the server simply deletes the dummynet rule and clears the pipe configuration of that client.

E. MT Proxy with and without inetd.conf:

The first version of the program uses `inetd.conf` for server-client connection. `inetd.conf` sits insides MT Proxy server, watches for TCP traffic coming to this server. If the client establishes a TCP connection with the server on port 30000 (arbitrary number set for MT Proxy software in the first version), the server `inetd.conf` binds traffic from/to client to/from server terminal as if the server only interacts with its terminal input (stdin) and output (stdout) stream.

The configuration of `/etc/inetd`.conf is as below:

```
mtranproxy stream tcp nowait nobody /usr/local/libexec/smtp.proxy
```

where `/usr/local/libexec/smtp.proxy` is the location of the MT Proxy program.

Service `mtranproxy` is configured with port number of 30000 in `/etc/services`. Thus MT Proxy program will be executed when there is any TCP connection to host `mtran.caia.swin.edu.au` (FreeBSD server running MT Proxy) on port 30000.

Although this model has an advantage of clearer, easier coding and implementing, it puts a heavy load on `inetd.conf` when there are a large number of concurrent client connections on different port numbers. This can considerably impact our email network performance.

As a result, MT Proxy software is modified to set up different sockets for each client email connection. Communication between the server and a client does not have to go through an `inetd.conf` agent.

MT Proxy listens for incoming client connections on port 6543 (arbitrary number set for the modified `inetd.conf`-independent version), establishes sockets with these clients and nominates child processes to deal with client requests. Because client input content has slightly changed in this new MT Proxy version, client input is reformatted to be accurately processed by the existing spam filtering function.

IV. PROTOTYPE DEMONSTRATION:

A demonstration has been set up to test MT Proxy functionality. In the demonstration, MT Proxy software runs on a machine of address 136.186.229.90. This machine (mtran.caia.swin.edu.au) is configured to query Swinburne DNS server of address 136.186.229.111 for host name translation and to forward emails to Swinburne SMTP server of address 136.186.229.34.

Two other machines are set up with their outgoing mail server's name as mtran.caia.swin.edu.au (or 136.186.229.90).

One machine (136.186.229.95) runs FreeBSD operating system and Kmail mail client while the other (136.186.229.23) runs Windows XP operating system and Netscape mail client.

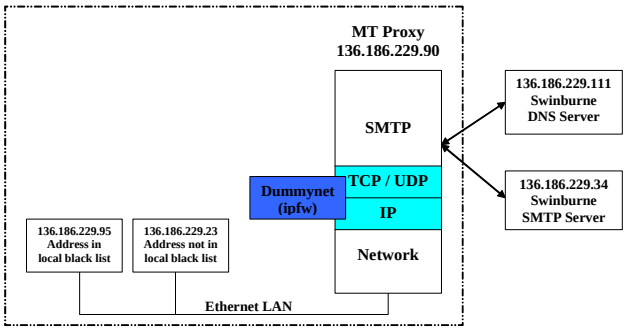

Figure 4: A Test Model of MT Proxy used in prototype demonstration

Both computers send a few spam and non-spam emails to MT Proxy.

IP address of the 136.186.229.95 machine is in a local black list of MT Proxy server. Connection from this computer experiences a certain delay and limited bandwidth even though its emails content does not contain any spam word. Transferring a large attachment file from this computer takes a significant amount of time at Kmail mail client as observed from Kmail window. This is the result of bandwidth reduction to 33Kbit/s for any sender in MT Proxy local black list database. Local black-listing method is set with 90% trustiness, hence spam percentage is calculated as:

0.5 (priority number for black-listing method) * 90% (trustiness number for local black list database) = 45%.

Thus, bandwidth is reduced to (112 – 45) /2 = 33 Kbits/s (Please refer to section III.C for the bandwidth calculation formula).

By inspecting the email arrival time at the recipient inbox, we can see that the email takes a "longer than normal" time to get its recipient.

This email (with non-spam content) sent from 136.186.229.95 black list machine and the server dummynet "ipfw list && ipfw pipe show" output during TCP connection with this client are shown as below:

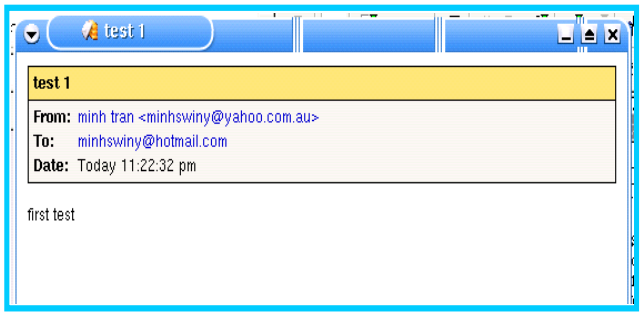

Figure 5: An example of a non-spam email sent by 136.186.229.95

ipfw output:

```
00001 pipe 1 tcp from 136.186.229.95 3704 to any 6543 in

65534 allow ip from any to any

65535 deny ip from any to any

00001:  33.000 Kbit/s  450 ms  50 sl. 0 queues (1 buckets) droptail

mask:  0x00  0x00000000/0x0000  -> 0x00000000/0x0000
```

As we can see on the above output, the email bandwidth is limited to 33Kbits/s and experiences a delay of 45ms.

The machine 136.186.229.95 is configured to send another email with spam content. In this case, the sender's spam percentage is the sum of that in both blacklisting and filtering method; which we would expect to be quite closed to 100%. It has been observed that sender bandwidth is eventually reduced to 8Kbit/s.

It also takes a significant time delay for the email to arrive at its recipient's inbox.

Following is an example of a spam email by this client and the server's "ipfw list && ipfw pipe show" output:

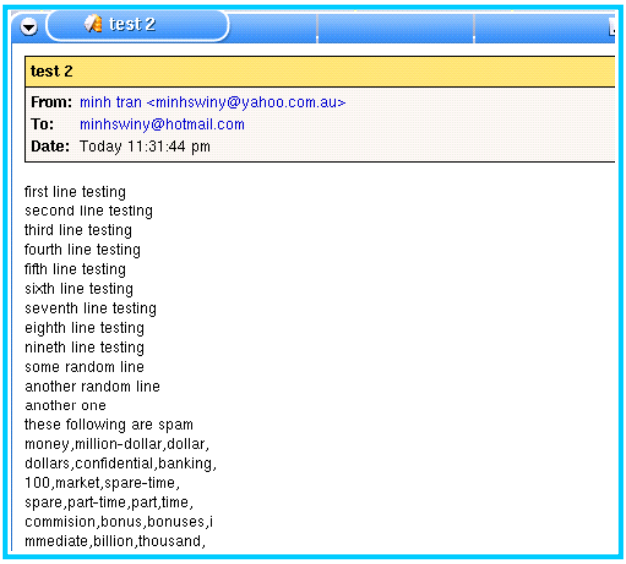

Figure 6: An example of a spam email sent by 136.186.229.95

ipfw output after black list analysis:

```
00003 pipe 3 tcp from 136.186.229.95 3716 to any 6543 in

65534 allow ip from any to any

65535 deny ip from any to any

00003:  33.000 Kbit/s  450 ms  50 sl. 0 queues (1 buckets) droptail

mask:  0x00  0x00000000/0x0000  -> 0x00000000/0x0000
```

Output during filtering analysis (of every 5 lines):

First analysis:

```
00003 pipe 3 tcp from 136.186.229.95 3716 to any 6543 in

65534 allow ip from any to any

65535 deny ip from any to any

00003:  28.000 Kbit/s  550 ms  50 sl. 1 queues (1 buckets) droptail

mask:  0x00  0x00000000/0x0000  -> 0x00000000/0x0000

BKT Prot ___Source IP/port____ ____Dest. IP/port____ Tot_pkt/bytes Pkt/Byte Drp

  0  tcp  136.186.229.95/3716 136.186.229.90/6543  6  1158 0  0  0
```

Second analysis:

```
00003 pipe 3 tcp from 136.186.229.95 3716 to any 6543 in

65534 allow ip from any to any

65535 deny ip from any to any

00003:  18.000 Kbit/s  760 ms  50 sl. 1 queues (1 buckets) droptail

mask:  0x00  0x00000000/0x0000  -> 0x00000000/0x0000

BKT Prot ___Source IP/port____ ____Dest. IP/port____ Tot_pkt/bytes Pkt/Byte Drp

  0  tcp  136.186.229.95/3716 136.186.229.90/6543  6  1158 0  0  0
```

Third (last) analysis:

```
00003 pipe 3 tcp from 136.186.229.95 3716 to any 6543 in

65534 allow ip from any to any

65535 deny ip from any to any

00003:  8.000 Kbit/s  950 ms  50 sl. 1 queues (1 buckets) droptail

mask:  0x00  0x00000000/0x0000  -> 0x00000000/0x0000

BKT Prot ___Source IP/port____ ____Dest. IP/port____ Tot_pkt/bytes Pkt/Byte Drp

  0  tcp  136.186.229.95/3716 136.186.229.90/6543  6  1158 0  0  0

FINAL STATISTIC SPAM AT END OF RECEIVE DATA 95
```

The demonstration test is done similarly for the other machine (136.186.229.23) that is not in the server local black list. When this machine does not send any email containing spam words in a subject and content, its email transferring process performs normally since there is no dummynet rule set on its email connection.

This machine is then configured to send a few spam words. As the spam analysis process goes through these words, its bandwidth decreases. Sending a big attachment file takes a considerable time at its Netscape mail client; which caused by dummynet bandwidth limitation on its email connection. Its spam email also has a certain delay before reaching the receiver's inbox.

V. Research Limitations and Further Work:

A. *MT Proxy software:*

The project prototype is built with an attempt to eliminate false positive and cause spammers pain. As a result, our main focus is to build anti-spam software with automatic TCP bandwidth control. Within time

constraint, we have not yet provided a challenge-response and complete statistic email analysis methods.

Further work can be done in developing a responsive challenge-response as well as a complex and accurate statistic-filtering algorithm.

At this stage, after a client ends email process with MT Proxy server, dummynet rule set for this client is simply deleted. This is not a very efficient solution. If the client resumes the email process just a few seconds after ending a previous connection, the server has to add a new dummynet rule and resume the entire spam analysis process. This behavior results in an unnecessary traffic load on the server. This could be totally a waste of time and even worst, a wrong decision if the next email appears to be much less spam than the previous one.

Thus future work can be done on having a suitable scheme on keeping a dummynet rule for a specific time after the client has finished connection with the server. During this time, unless the client sends more spam, the dummynet rule is not updated (to less spam value). The count-down timer keeps track of the spam history time. During its count-down time, if the client sends any spam (as indicated in the analysis result), it resets to a maximum value (say 5 minutes). Otherwise, it keeps counting down. When the time expires (after 5 minutes) and there is no spam sent from the client, the dummynet rule set for this client can be deleted.

The prototype demonstration (as in section IV) has been performed to test functionality of MT Proxy software. One shortcoming of this demonstration is that it is not practical to test black list queries from Internet DNSBLs since our network IP address is certainly not in these DNS black lists. Thus the DNSBL test can be done in the future by building our own DNSBL server and let the program send queries to this server.

The demonstration is a simple functionality test model for MT Proxy. Utility/performance test and evaluation will be carried out in the next part of our project. A simple version of Testbed model is described in the next section.

B. A simple Testbed model:

Four virtual hosts are configured to automatically send mail using shell scripts.

They are configured as below:

-One machine whose IP address is not in local black list and email content is not spam.

-One machine whose IP address is not in local black list and email content is spam.

-One machine whose IP address is in local black list and email content is not spam.

-One machine whose IP address is in local black list and email content is spam.

Content of these emails are automatically fed with random file names from either spam or non-spam directory (depending on the above host allocation). Each file name is numbered with a corresponding certain level of spam for easy spam classification of emails sent from these machines.

These hosts are configured with their mail server address as that of MT Proxy server, which later forwards emails to a real SMTP mail server after scanning through all emails for spam.

MT Proxy server will run TCPdump to dump statistic information of TCP traffic from the senders.

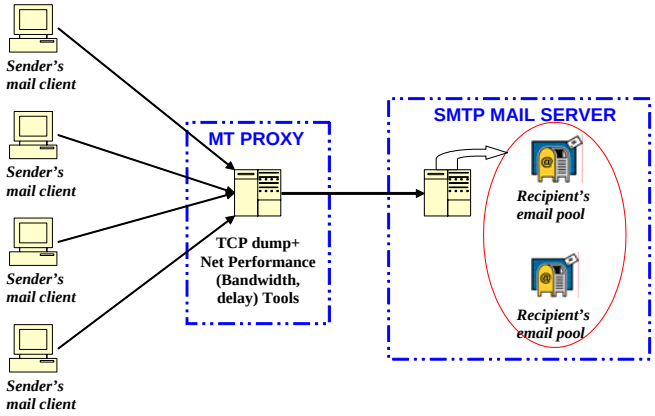

Figure 7: A simple MT Proxy's Testbed Model

TCP dump is a network tool used by `root` to sniff the traffic traveling through a specific network interface (typically Ethernet) in promiscuous mode [26]. TCPdump also provides options of writing the traffic summary into a file for later reference (with switch *–w*) and capability of capturing only interesting traffic (with a boolean filter *expression*). A synopsis of TCPdump command is as below [25]:

```
tcpdump [-adeflnNOpqRStuvxX ] [-c count ]
[-C file_size ] [-F file ] [-i interface ] [-m
module ] [-r file ] [-s snaplen ] [-T type ]
[-w file ] [-E algo:secret ] [expression ]
```

A network analysis utility (such as: libpcap, nttcp, netpearl, pathchar, ethernetreal) is used to monitor and analyse TCPdump files. Bandwidth and delay summary is pulled out from these network analysis tools. Statistic plots of bandwidth v/s packets received, delay v/s packets captured of different spam/non-spam traffic types will be carried out for performance and utility evaluation.

An alternative method is to write shell scripts to allow mail clients to ping MT Proxy. Round-trip time from clients' connection can be extracted and plotted for delay statistic measurement and comparison.

Basically we would expect to see characterized difference between plots of bandwidth, delay v/s packets captured of these clients; as such that MT Proxy could cause a significant bandwidth reduction and delay for clients sending spam in contrast to non-spam clients.

We would expect results, which look similar to these following graphs:

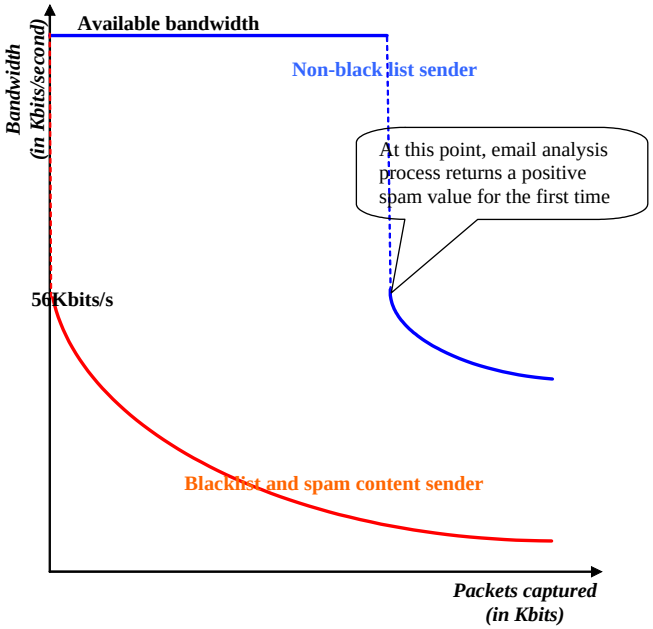

Figure 8: Bandwidth v/s packtes plot

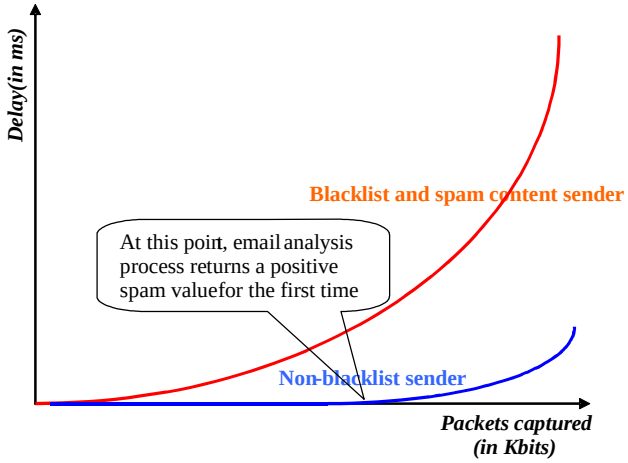

Figure 9: Delay vs packets plot

Analysis will also be carried out by inspecting email spools of different accounts at a SMTP mail server.

A few email accounts, each of them will receive different spam and non-spam emails, are set up inside this server. Statistic plots will be produced for growing rates as a function of time for different account spool sizes. Accounts, which do not receive many spam emails, will have their size increase normally whilst the ones with more spam will have a slower mailbox-growing rate.

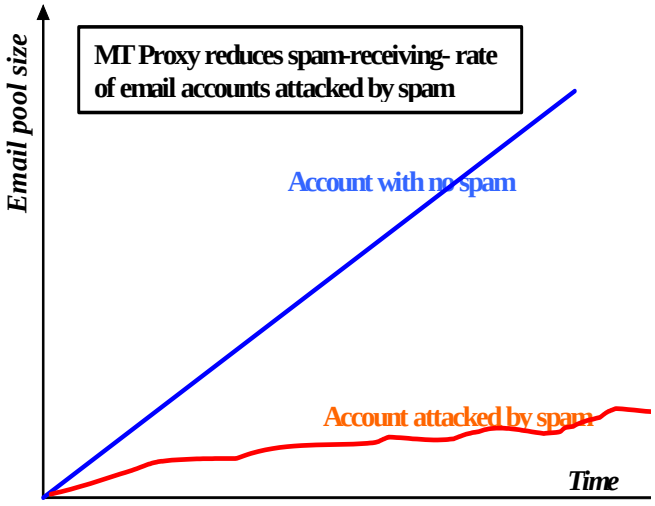

Figure 10: Recipients' email spool plot

We are also interested in looking at plots of email spool size v/s number of spam messages received at the recipient inbox. Both short and long emails will be examined to characterize MT proxy's behaviors.

Generally, efficiency and effectiveness of MT Proxy is evaluated on these following criteria:

- How much bandwidth is reduced due to black-listing and filtering analysis?
- How this bandwidth limitation affects spammers' spam generating software/ buffer/ mail client?
- How much delay is experienced by spammers' email packets?
- How this delay impacts other TCP traffic from spammers/ traffic to other spammers' recipients who are not in our network?
- Are bandwidth reduction and delay implemented in this MT Proxy scheme too much/ too little? What is the optimal function for bandwidth and delay in related to spam percentage?
- How much slower will spam emails reach the recipient's account? How will these accounts be affected by spam over a long? In comparison to accounts which are not attacked by spam?
- How is network performance with MT Proxy implementation compared to without this software?
- Is it worth sacrificing resources for this scheme?

VI. CONCLUSION:

Fighting against spam is not an easy job. As spammers become smarter, statistic spam analysis processes become more complicated. The needs of keeping white/black lists, filtering rules up to date also become crucial. These costly actions do not mean they can totally eliminate false positive, they could even lead to more serious false positive problems.

Our software guarantees to never block any email traffic, and as a result, never make any false positive mistake. Even though we might not obtain accurate analysis results of senders' spam probability, their emails still get to recipients eventually.

People do not normally send many personal and business emails within a short time, unless they are spammers. A slow connection at a mail client or a few minute delay on email transferring would not be a big worry for one individual email message. However, for spammers, who always send barrages of emails, this scheme would make a big difference.

A certain reduction on bandwidth can result in more emails being queued in a spammer's mail client buffer, which can go over limit if the spam generation rate is greater than a mail client transferring rate. Having a full buffer would possibly leads to situations where the mail client reluctantly drops its incoming emails and decreases its spam generating software speed. Significant bandwidth reduction and delay also consume spammers' time and resources; which, we hope, results in a loss in operating efficiency and profitability of their businesss.

Our software has been implemented to achieve these goals: to make no false positive mistake and to cause spammers "real pain".

VII. REFERENCE:

[1] M. Lamb, "Using Statistic to cause Spammers Pain", February 2003, http://www.martiansoftware.com/articles/spammerpain.html#note1

[2] G. J. Koprowski., "Spam filtering and the flague of False Positive", TechNewsWorld, September 2003, http://www.technewsworld.com/perl/story/31703.html

[3] C. Williams, D. Ferris, "The cost of spam false poristive", Ferris Analyzer Information Service. Report #385. August 2003, http://www.brightmail.com/pdfs/Cost_of_Spam_False_Positives___Ferris_Research_8_2003.pdf

[4] Anti-SPAM tools, http://www.spamresearchcenter.com/ (as of March 2004)

[5] Dictionary.com, http://dictionary.reference.com/ (as of March 2004)

[6] Spamhaus, http://spamhause.org/definition.html (as of March 2004)

[7] P. Graham, "Filter vesus Blacklist", September 2002, http://www.paulgraham.com/falsepositives.html

[8] C. Corrado, "Bayesian Spam filtering for the masses", NewsForcges, Oct 2003, http://www.newsforge.com/software/03/10/24/2031234.shtml?tid=74

[9] L. Chae, "Email and SMTP",Network Magazine, January 1997, http://www.networkmagazine.com/article/NMG20000726S0002

[10] J. B. Postel, "RFC 821 Simple Mail Transfer Protocol", August 1982, http://www.cis.ohio-state.edu/cgi-bin/rfc/rfc0821.html

[11] L. Rizzo, "Dummynet", http://cs.ecs.baylor.edu/~donahoo/tools/dummy/tutorial.htm (as of March 2004)

[12] Dummynet, http://cs.ecs.baylor.edu/~donahoo/tools/dummy/ (as of March 2004)

[13] SpamHaus, http://www.spamhaus.org/ (as of March 2004)

[14] Spam and Open Relay Blocking System (SORBS), http://www.us.sorbs.net/ (as of March 2004)

[15] DNS BlackList Information, http://www.dnsbl.info/ (as of March 2004)

[16] Open Relay Database (ORDB), http://www.ordb.org/ (as of March 2004)

[17] P. Graham, " A Plan for Spam", August 2002, http://www.paulgraham.com/spam.html

[18] Robinson G. ,"Spam Detection", October 2003, http://radio.weblogs.com/0101454/stories/2002/09/16/spamDetection.html

[19] W. Zekoll, smtp.proxy software, http://www.quietsche-entchen.de/software/smtp.proxy.html (as of March 2004)

[20] E. Lewis, "SMTP 101", http://perl.about.com/cs/email/a/112703.htm (as of March 2004)

[21] "A Typical Mail Environment", http://www.students.cs.byu.edu/~cs460ta/cs460/slides/lecture39.ppt (as of March 2004)

[22] L. Chae, "The Basic of Email access", Network Magazine, January 1997, http://www.networkmagazine.com/article/NMG20000726S0002

[23] RFC 1725, "Post Office Protocol – Version 3", November 1994, http://www.apps.ietf.org/rfc/rfc1725.html

[24] RFC 2060, "Internet Message Acess Protocol – Version 4", December 1996, http://www.apps.ietf.org/rfc/rfc2060.html

[25] TCPdump, http://www.tcpdump.org/tcpdump_man.html (as of March 2004)

[26] TCPdump, http://www.freesoft.org/CIE/Topics/55.htm (as of March 2004)

[27] "Guide to Internet Terms: A Glossary", http://www.getnetwise.org/glossary.php (as of March 2004)

[28] J. M. Jones, "Almost All E-Mail Users Say Internet, E-Mail Have Made Lives Better", The Gallup Organization, July 2001, http://www.gallup.com/content/?ci=4711

[29] J. Snyder, "Test: Spam in the wild", Network World, September 2003, http://www.nwfusion.com/reviews/2003/0915spam.html

[30] GFI, "Why Bayesian filtering is the most effective anti-spam technology", http://www.gfi.com/mes/wpbayesian.htm#howitworks (as of March 2004)

[31] H. Thornburng, "Introduction to Bayesian Statistics", Stanford University, April 2001, http://www-ccrma.stanford.edu/~jos/bayes/bayes.html

VIII. APPENDIX:

A. Spam word database:

Flollowing are some words in MT Proxy 's local black list spam database:

```
money, million-dollar, dollar, dollars, confidential, banking, 100,m arket,s pare-time, spare,p art-time, commision, bonus, bonuses, immediate, billion, bllions, thousand, $, $$, $$$, marketplace, high-paid, work-from-home, huge-benefit, your-own, full-company-support, step-by-step, proven, working-system, product, sell, no-sales-experience, profit, free, amazing, star-making-money, from-home, mailing-list, low-price, credit-card, credit, visa-card, visa, card, master-card, master, save, turn, campaigns, supply, supplies, promote, promotional, stock, friends, beg, murder, rude, for-you, 4U.
```

B. An Example of filtering methods used in MT Proxy:

For an email message with header and content like this:

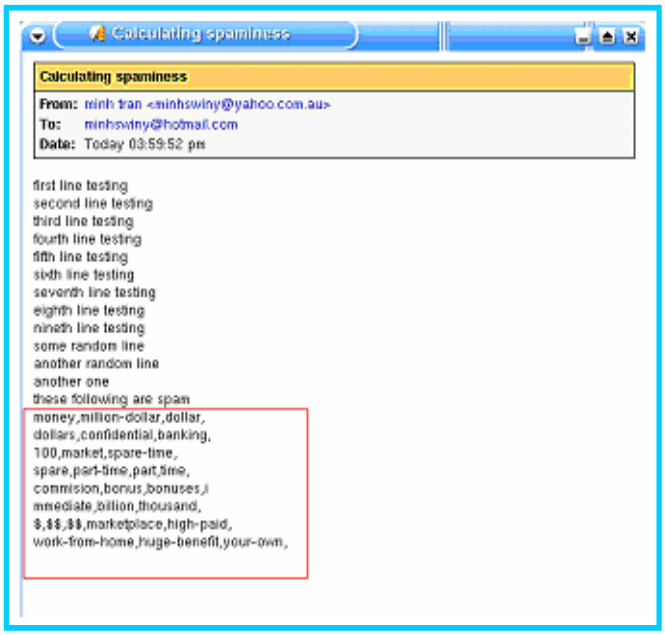

Figure 11: An example of a spam email

Every line of client input is analysed for spam. An email spam value is accumulated and compared with a worst spam value after every 5 lines (default setting). Dummnet is only updated if the new spam value (after 5 lines analysis) is greater than the worst spam.

In the above email, corresponding client input line and its spam percentage calculated using Gary Robinson method are in a table below:

Client input	P	Q	S	Worst spam percentage
From: minh tran <minhswiny@yahoo.com.au>	0.000100	0.999900	0.000100	-
To: minhswiny@hotmail.com	0.000100	0.999900	0.000100	-
Subject: Calculating spaminess	0.000100	0.999900	0.000100	-
Date: Thu, 4 Mar 2004 15:59:52 +1100	0.000100	0.999900	0.000100	-
User-Agent: KMail/1.5.4	0.000100	0.999900	0.000100	0
MIME-Version: 1.0	0.000100	0.999900	0.000100	-
Content-Type: text/plain;	0.000100	0.999900	0.000100	-
charset="us-ascii"	0.000100	0.999900	0.000100	-
Content-Transfer-Encoding: 7bit	0.000100	0.999900	0.000100	-
Content-Disposition: inline	0.000100	0.999900	0.000100	0
Message-Id: <200403041559.52040.minhswiny@yahoo.com.au>	0.000100	0.999900	0.000100	-
				-
first line testing	0.000100	0.999900	0.000100	-
second line testing	0.000100	0.999900	0.000100	-
third line testing	0.000100	0.999900	0.000100	0
fourth line testing	0.000100	0.999900	0.000100	-
fifth line testing	0.000100	0.999900	0.000100	-
sixth line testing	0.000100	0.999900	0.000100	-
seventh line testing	0.000100	0.999900	0.000100	-
eighth line testing	0.000100	0.999900	0.000100	0
nineth line testing	0.000100	0.999900	0.000100	-
some random line	0.000100	0.999900	0.000100	-
another random line	0.000100	0.999900	0.000100	-
another one	0.000100	0.999900	0.000100	-
these following are spam	0.000100	0.999900	0.000100	0
money,million-dollar,dollar,	0.999000	0.001000	1	-
dollars,confidential,banking,	0.999000	0.001000	1	-
100,market,spare-time,	0.999000	0.001000	1	-
spare,part-time,part,time,	0.999000	0.001000	1	-
commision,bonus,bonuses,i	0.999000	0.001000	1	50
mmediate,billion,thousand,	0.999000	0.001000	1	-
$,$$,$$,marketplace,high-paid,	0.999000	0.001000	1	-
work-from-home,huge-benefit,your-own,	0.999000	0.001000	1	-

Table 1: Gary Robinson calculation and final filtering spam percentage