Resisting Spam Delivery by TCP Damping[*]

Kang Li
Department of Computer Science
University of Georgia
kangli@cs.uga.edu

Calton Pu, Mustaque Ahamad
Collage of Computing
Georgia Institute of Technology
{calton,mustaque}@cc.gatech.edu

Abstract. Spam has become a major problem that is threatening the efficiency of the current email system. Spam is overwhelming the Internet because 1) emails are pushed from senders to receivers without much control from recipients, and 2) the cost for delivering emails is very low. In this paper, we present an anti-spam framework that slows down spammers: by adding delay to email delivery, and by consuming more sender resources. Both delay and resource consumption are controlled based on the likelihood of the source of email messages being a spammer, so that our technique only impact the spammers and has negligible impact on normal email senders. The mechanisms are implemented in the TCP level at the recipient side without requiring any modifications at the sender side. Our evaluations show that selectively delaying connections can effectively slow down a spammer thousands of times when they use a simple setup or use open relays. The mechanism of increasing sender's resource consumption can significantly slow down spammers even when they are spamming from their own optimized servers.

1. Introduction

Unsolicited email messages, also known as spam, have become a serious problem for users of the Internet. They are causing disruptions to many users everyday [1]. Recent studies show that between one-seventh and one-half of email messages going into an Internet user's mailbox are spam [2]. Handling spam consumes valuable user time. Worst yet, important emails might be overlooked in the sheer influx of spam.

In the war against spammers, the current anti-spam mechanisms fall short. Right now, the main tool for protecting victims is the spam filter, e.g., white-lists and content-based filters [3][4]. Although spam filters have been designed in analogy to virus detectors, spam filters are less than perfect protection for two reasons. First, spam filters that are less discriminating have the problem of *false positives* — they mistakenly block or misclassify legitimate messages as spam [5][6]. The damage of a single false positive can be very serious, depending on the email content [7]. Another problem with spam filters is that they only work after emails have been delivered to the mail servers of recipients, and give no incentives to spammers to remove unwanted users from their lists.

We argue that for an anti-spam system to be effective, we need to combine victim protection with active resistance to spam. Moreover, the resistance should be sufficiently effective as to encourage the spammer to *want* to remove the victims that fight back from the spam list. Existing examples of such anti-spam approaches include client puzzle [8] and pricing email [9][10]. They increase the cost of delivering a message by requiring real money or system resources from the sender, e.g., network bandwidth, CPU cycles, and memory. While they achieve the goal of slowing down the spammers, they require significant modifications to the system software.

In this paper, we present a framework that slows down the spammers by carefully tuning email delivery parameters, without changing the current email delivery software. The anti-spam mechanisms under this framework directly add delay to the email delivery process, and increase computation cost at the sender side for delivering emails. Our mechanisms differ from traditional cost-based ones by requiring no authentication infrastructure or replacements of sender software.

Within the anti-spam framework, we use existing spam filters to generate a spam likelihood value for each message at its delivery time. The likelihood information is used by the recipient to drive the spam resistance mechanisms that control the delay and cost of email delivery.

The mechanisms of adding delay and cost are implemented at the TCP [11] level. TCP is the transport protocol used by the Simple Mail Transfer Protocol (SMTP) [12][13], which is the protocol used by Internet email. Slowing a TCP connection from the receiver side can be achieved in many ways, such as postponing TCP acknowledgements

[*] This research is partly supported by UGARF grant and NSF grant ITR-0121643.

or injecting congestion notification bits to spoof congestion. To increase sender computation cost, the receiver could force the sender to generate more packets (e.g., of a smaller size) to deliver the same message. Since all emails are delivered over TCP, selectively delaying or adding cost to a suspicious TCP sender would effectively slow down spammers. This spam resistance effect is achieved based on the assumption that the TCP connections carrying spam are more likely to be identified as suspicious connections by many users, whereas the TCP connections for normal emails are unlikely to be misclassified by a large number of users simultaneously.

These techniques to selectively slow down email delivery are implemented at the receiver, without any changes to the sender side or the underlying systems software. Because these resisting mechanisms make a TCP connection less efficient, "*damping*" TCP connections is their goal. Therefore, we call these mechanisms *TCP damping*.

The rest of the paper is organized as follows: Section 2 reviews the related work. In section 3, we first describe the anti-spam framework, and then we focus our discussion on the TCP damping mechanisms – how a TCP receiver increases delay or resource consumption for a sender. In section 4, we evaluate two TCP damping mechanisms by measuring their impacts on the sender. Because increasing the sender cost requires the receiver to consume more resources to receive messages, we also measure the receiver side cost of applying TCP damping mechanisms. In section 5, we address the possible reactions from spammers, the issues of potential DoS threat, and the impact on open relays and public mail servers. We conclude the paper in section 6.

2. Related Work

Various forms of filters, such as white-lists, black-lists and content-based lists, are widely used to defend against spam. A large amount of research has been done to increase the accuracy of spam filters [18], and sophisticated algorithms, such as Bayesian filtering [4], have been used in spam filter products. Although spam filters help users significantly, they have several problems. All types of spam filters have false positives. In addition, content-based filters only filter a message after it has already been delivered and stored in the receiver's mail server. By that time, large amounts of resources have already been consumed to deal with the spam. Ideally, spam should be filtered before it reaches users.

While filters are dominant anti-spam techniques, there are other approaches. One of the alternatives is the client puzzle approach [8]. The client puzzle approach can deter spammers because it requires considerable effort from the sender to deliver a message to one recipient, and a spammer needs to deal with a huge number of recipients. However, since the sender is required to solve puzzles and confirm messages, the level of human involvement needed to make this work prevents the mechanism from being widely accepted.

There are other approaches, such as pricing emails or using authentications [9][10]. Both of these approaches require considerable changes to the SMTP protocol and the email software. The approaches presented in this paper differ from these approaches by requiring changes to only the client side without any modifications to the protocol or the sender side software. Indeed, our solution can be implemented as a gateway at the receiver side, transparent to existing software. We consider this easy deployment as a great advantage of our approach.

While the possibility of adding delay to spammers has been considered before (e.g. teergrubing [14] and tarpit [15]), the existing work is purely at the application level. Delaying in the SMTP and application level does not work well. Spam messages are typically small. A spammer can push whole messages to the socket buffer at the recipient's mail server. As long as the recipient side TCP has acknowledged the arrival of those data, the spammer can immediately switch to spamming others and ignore the delay caused by the recipient. As a result, application level delay has almost no impact on the efficiency of spammers. In this paper, we take a cross-layer approach that combines application level information and TCP level mechanisms. It is based on damping mechanisms that we build in the TCP level, and it uses application information to drive the TCP level mechanisms. The recipient has control over the progress of the TCP connection and which pieces of data it acknowledges to the sender.

3. An Anti-Spam Framework

In this section, we briefly describe our anti-spam framework. We first describe how we combine filter information with TCP level mechanisms, and then we focus our discussion on the TCP level mechanisms.

3.1. Why Use a Cross-layer Approach

Although spam filters block a large amount of spam messages, solely relying on filters will not stop spam from spreading in the Internet. A purely network based approach is not likely to stop spam either. Whether a message is spam or not is based on its content, not the protocol behavior. It can only be determined by applications.

Our anti-spam framework is a cross-layer approach that combines filter output and TCP level mechanisms. We propose a new way to use spam filters – apply them at email delivery time, during a TCP connection, and within an SMTP session. We do not use filters to mark or drop messages but rather use them to provide “hints” (the spam likelihood information) to the mechanisms that selectively control the cost of the connection. Controlling cost rather than filtering messages is crucial to email systems because of the potential damage of false positives.

The effectiveness of this anti-spam framework is based on the aggregated behavior of many users, and possibly with many different filters, rather than the behavior of a single user with one particular filter. We assume messages from a spammer statistically get a greater chance of being delayed and higher cost for delivery. The aggregated delay and cost from multiple users would reduce the efficiency of the spammer. Compared to spammers, normal users that address messages to a small group of people would have no problems because the overall cost is low even when some recipients misclassify and apply damping to them. For the senders of bulk messages, as long as the recipients are soliciting the messages and thus do not add delay or cost to the message delivery, these senders would be immune to delay and cost impact.

3.2. SpamAssassin at Email Delivery Time

In this section, we use *SpamAssassin* [19] as an example to show that we can apply a spam filter to generate spam likelihood estimation at the email delivery time. SpamAssassin uses a list of heuristic tests to determine a message’s spam likelihood, a probability value for the email to be a spam message. Normally SpamAssassin is used to classify email messages. Messages with a spam likelihood larger than a pre-defined threshold would be marked (or dropped) by the filter, and the rest are passed to users. The majority of the SpamAssassin tests are heuristics based on SMTP header information, such as the test of reverse DNS matching – if the domain name following the SMTP hello command does not match to the reverse DNS of the sender IP then the message’s spam likelihood increases.

Most of these tests can be applied at delivery time. To illustrate this, we present an example of an SMTP session in Figure 1. In the mail delivery stage, every message is delivered over an SMTP session similar to this.

At the beginning of the session, the email sender requests a TCP connection on port 25 to the recipient's mail server. Some SpamAssassin tests can be applied as early as this stage to produce a spam likelihood value. For example, one SpamAssassin test checks the sender’s IP against a black-list. If the message is from an address that is on the list that is known to send many spam messages, then the spam likelihood value increases.

The spam likelihood estimation can also be updated at the SMTP handshake stage (step 1~3) after using the reverse DNS test. It can also be updated based on the email address information (step 4~6). For example, a long recipient address list could lead to a high spam likelihood. The spam likelihood can continue to be updated based on the body of the email message.

```
After a TCP connection setup between the sender (Snd) and the receiver (Rcv):
1.      Rcv: 220 mailserver.MrX.com
2.      Snd: HELO springfield.net
3.      Rcv: 250  Hello springfield.net, pleased to meet you
4.      Snd: MAIL FROM: <bart@springfield.net>
5.      Rcv: 250  bart@springfield.net... Sender ok
6.      Snd: RCPT TO: homer@MrX.com
7.      Rcv: 250 homer@MrX.com ... Recipient ok
8.      Snd: DATA
9.      Rcv: 354 Enter mail, end with "." on a line by itself
10.     Snd: From: bart@springfield.net
11.     Snd: To: homer@MrX.com
12.     Snd: Are we there yet?
13.     Snd: .
14.     Rcv: 250 Message accepted for delivery
15.     Snd: QUIT
16.     Rcv: 221 MrX.com closing TCP connection
```

Figure 1: An SMTP session example (Due to space limit, we omit the blank lines in the body part).

3.3. TCP Damping

Once we can estimate the spam likelihood of a message at delivery time, the next question is how to control the delay and cost for a TCP connection based on this spam likelihood information. We present the approach of slowing down spammers by damping TCP connections in this section. We first briefly review TCP to allow those unfamiliar with its operational details to understand the damping mechanisms.

TCP is a connection-oriented, reliable, ordered, byte-stream protocol. A TCP sender divides the data stream into individual segments, each of which is no longer than the Maximum Segment Size (MSS) negotiated between the sender and receiver. Each segment is labeled with explicit sequence numbers to guarantee ordering and reliability. When a host receives an in-sequence segment it sends a cumulative acknowledgement (ACK) in return, notifying the sender that all the data preceding that segment's sequence number has been received and can be retired from the sender's retransmission buffer. If an out-of-sequence segment is received, then the receiver acknowledges the next contiguous sequence number that was expected. If outstanding data is not acknowledged for a period of time, the sender will timeout and retransmit the unacknowledged segments. The timeout value is controlled based on the monitored round-trip-time (RTT) and its variation.

3.3.1. Increasing Delay

A TCP receiver can delay the process of a TCP connection by several methods: delaying the return of acknowledgements, closing the advertisement window, or faking congestion. We choose delaying the return traffic as one damping mechanism for its simplicity.

TCP Damping 1:

Once the damping mechanism starts, every packet from the receiver to the sender is delayed for a time period T before being delivered to the network. Here T is called the *damping factor* of this TCP damping mechanism.

This TCP damping mechanism works well with any TCP sender. Actually delayed acknowledgements (*delayed ACKs*) have been used in many TCP receiver implementations, but for a different purpose: to increase the chance to piggyback ACKs with data packets. The TCP damping differs from delayed acknowledgements by a user specified time period rather than a fixed period, e.g 200ms for the delayed ACK used in some BSD systems.

Delaying traffic from receiver to sender could cause TCP timeouts, especially when delay is inserted in the middle of a TCP connection. TCP timeout is not a problem for this damping mechanism, as long as it does not cause a normal sender to terminate the connection. If it causes a TCP connection to be terminated, a spammer can immediately start sending email to the next recipient, which defeats the goal of slowing down the spammer. A TCP connection is terminated only after at least 15 retransmission efforts. The time period of the 15 retransmissions depends on the RTO estimation. The delay has to be limited to avoid an SMTP level timeout which could terminate an email delivery even when the TCP connection is still alive. The SMTP timeouts are unlikely to happen when we use the above delay limit, because the SMTP specifications [12][13] require very coarse grain timeouts, for example, 10 Minutes for the delivery of the email body.

Pausing at the application level can also slow down a connection, but it is not as effective as inserting delay at the TCP level. Especially for spam messages, whose sizes are typically small compared to the default TCP receiver buffer size[1], pausing the application does not have an impact on the sender if all sender data have been acknowledged at the TCP level. In that scenario, the sender could close the socket regardless of the receiver's actions, because the sender knows the message has been safely delivered.

3.3.2. Increasing Resource Consumption

TCP Damping 2:

The receiver uses an advertised window of W bytes that forces the sender to send packets with a payload less than or equal to W.

Delaying TCP connections slows down an email delivery process and reduces the spam throughput by forcing the sending machine to idle. This is the case when a single delivery queue is used, and messages are delivered sequentially by a sender program. However, spammers can change their strategy by using multiple simultaneous processes to improve the overall email throughput. In this case, stalling a single sending queue becomes less

[1] Today’s spam is typically less than 1KB, whereas in most operating systems, the receiver buffer size is at least 64KB.

effective. At this time, the only limit to a spammer’s throughput is his resource bottleneck, such as network, CPU, or memory. When a spammer opens multiple simultaneous connections to achieve high throughput, his outgoing link is very likely to become a bottleneck. To reduce spam, we propose some TCP damping mechanisms to consume the sender's critical resources to reduce the spammer's throughput. Due to the space limit, we present only one mechanism in this paper. Details of this and additional damping mechanisms are available in a longer version [16].

This TCP damping mechanism gives the receiver control of the sender’s network resource consumption by controlling the amount of packets per email. The advertised window size W is the damping factor. It controls the magnitude of resource consumption that the receiver would like to introduce to the sender. For example, with a normal advertised window size, packets use the payload size 1460 Bytes. If a receiver keeps setting the advertised window to 1 Byte, a 1KB message body would be delivered by 1000 packets instead of a single packet in the ordinary case. Network devices are often limited by a packet forwarding rate. By setting a small advertised window size, the sender is forced to generate more packets to deliver a single email. In addition, with a small payload size, the header overhead becomes significantly higher, from 27% to 4000%. Since the network is the bottleneck, the sender’s efficiency is decreased when the per email cost increases.

3.4. Receiver's Cost

As the sender’s computation cost increases, the required cost at the receiver side also increases. This is not a problem because the resource requirement for the receiver to apply the TCP damping is low. The effect of slowing down the spammer is based on an aggregated effect of many receivers. With the damping mechanisms, the per connection resource consumption for sender and receiver is about the same, at least in terms of traffic going each way. However, the spammer needs to connect to many receivers. Therefore, the load on the spammer’s side is considerably higher than a single receiver. Certainly the effect on the spammer depends on how many receivers identify the message as spam and adopt the damping mechanisms. A small number of receivers would need to damp each email connection hard in order to defeat the spammer, whereas with a large number of receivers damping the email connections, each one would only be required to devote a small amount of resources. Another advantage of the damping approach is that the receiver has control of the progress of the connection and the cost. A receiver can decide to employ damping for only a few email connections that he suspects as spam. The resources used for the counter attack are under receiver's control.

4. Evaluations

To evaluate the effectiveness of TCP damping as an anti-spam technique, we implemented the damping mechanism by modifying TCP in the Linux 2.4.18 kernel, and we made a receiver email proxy that uses the TCP damping mechanism. The resulting software, which we refer to as "Sticky Proxy", slows down email connections that are suspected of being spam and adds no effect to other connections. We demonstrate the ability of slowing down spam by measuring a bulk mail sender's email throughput under various recipient setups, such as the percentage of recipients viewing a message as spam and the damping factor used by recipients.

We used *sendmail* as the bulk email sender software. We configured sendmail to optimize it for high sending throughput. All the messages, header files, and the receiver addresses are put in a RAM disk to avoid disk accesses. The sendmail log is disabled for the same reason. Sendmail is set to interactive mode so that it will deliver emails immediately without any sleeping periods in between. We have two mail servers running as receivers; both also use sendmail as the mail server program, but one of them uses the sticky proxy in front of the receiver. The sendmail receivers are optimized for high receiving throughput. For example, all received messages are redirected to /dev/null rather than writing to disk to avoid disk I/O delay.

The sender sends the same message to different accounts in the two receivers. SMTP pipeline is disabled so that every message requires a TCP connection. For simplicity, the receiver that runs the sticky proxy treats all incoming messages as spam, whereas the other receiver treats all messages as legitimate email. Needless to say, in a real scenario, it is not easy to draw the line between spam and legitimate emails. Otherwise, spam could be simply filtered out without the need of developing any anti-spam techniques. Here our goal is to evaluate the impact of the percentage of receivers using the TCP damping mechanism. By controlling the number of messages sent to each receiver, we can control the number of TCP damping connections versus normal ones.

We conduct our experiments over the Internet2 connection. The two receivers are in the same subnet. The sender is separated from them by a 63 msec round-trip-time. The sender email throughput is measured as an average over 1000 messages. We choose 1K Bytes for the size of the spam message body. This size is chosen based on the average spam message size measured over a recent spam collection [17].

4.1. Damping TCP by Delaying

In this subsection, we evaluate the effect of TCP damping by delaying. We first let the sender use a single sendmail process. The emails to the two receivers are evenly spaced in the queue. The sendmail process delivers a queue of email messages one by one. In the experiments, we vary two parameters. One is the percentage of emails being sent to each receiver. The other is the per-connection damping factor: the delay introduced by the sticky proxy. By adjusting these two parameters, we expect to achieve two goals. One goal is to see the slowing down effect when different number of the recipients suspect the message as spam and apply TCP damping. The other goal is to see the effect of choosing different damping factors (delay T, window size W, etc.).

Figure 2 presents the sender's average email throughput versus the damping factor. As the delay factor increases, the sender throughput, in terms of number of emails delivered over time, decreases. The decrease is significant, more than 1000 times, when the percentage of messages sent to the sticky proxy is high. When the percentage of messages to the sticky proxy is low, the decrease is small.

This result illustrates that TCP damping has a big impact on spammers but a tiny impact on legitimate bulk email senders. For spammers, we assume a large group of receivers would like to damp them and thus delay the spammers hundreds or thousands times; for a legitimate sender, zero or a small group people might damp them because of misclassifications, which only causes a small delay. Certainly this claim is based on the assumption that TCP damping is widely deployed. If this is not the case, say only a small group of users are deployed with the damping mechanism, then each one would need to use a larger damping factor to achieve the same effect. Right now we are working on mechanisms for increasing the reaction rate by polluting the spammer’s recipient lists with addresses of anti-spam honeypots.

The effectiveness of damping is also related to the damping factor and the network. In Figure 2, the TCP damping has almost no impact on the throughput when the delay is small. It only starts to noticeably slow down the throughput when the delay is close to 60ms. This is because the RTT between the sender and receivers is about 63ms, and delay within an RTT does not have a significant impact on the connection. This effect varies depending on the network. In general, 100ms to 1000ms damping factor could show a big impact on most of the connections.

The measurements presented in Figure 2 were taken where a single sendmail process was used. This is set as default in many mail servers, in particular when the spammer does not have control over the server sending email, such as in the case of using an open relay.

We also evaluated TCP damping with delay for multiple sendmail processes. These results are presented in Figure 3. In this experiment, a fixed damping factor of 100 seconds is used, and we vary two other parameters in the experiments, the number of processes and the percentage of recipients using TCP damping.

Figure 3 shows that a spammer can significantly increase the email throughput by using multiple sendmail processes. The throughput increases along with the number of processes used until it hits a maximum rate. In general, the throughput can be increased 10 times by using 40 sendmail processes. More optimized sender software might be able to push this rate higher.

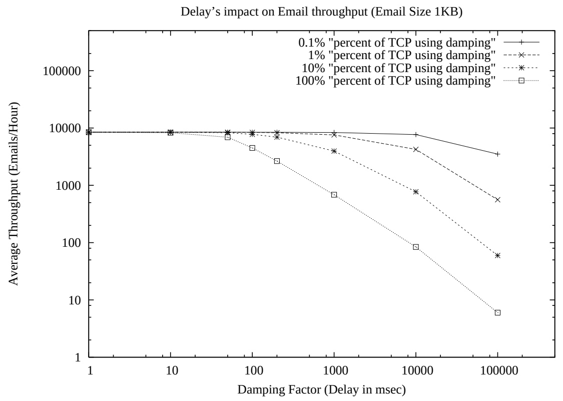

Figure 2: TCP Damping-1’s Impact on Throughput (single sending queue)

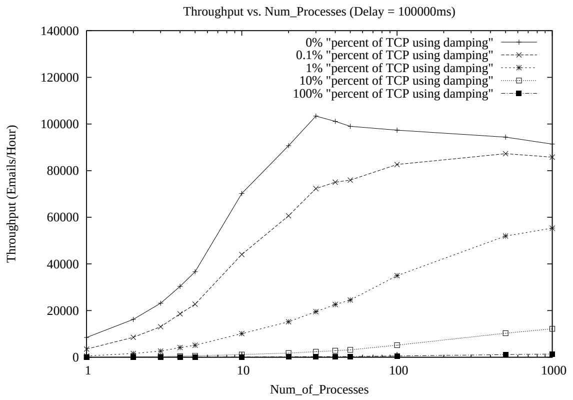

Figure 3: TCP Damping-1’s Impact on Throughput (multiple sending queues)

TCP damping by delaying becomes less effective especially when the percentage of users applying it is small. When a large number of recipients are using TCP damping against the spammer, the sender throughput can still be slowed down, but not as significantly as in the single queue case.

The sender could further increase the concurrency by using multiple hosts. By using multiple sending hosts, spammers can increase the number of possible concurrent connections. However, using more hosts would increase a spammer's costs, which would reduce the benefit to the spammers. Finally, when a spammer uses multiple machines, the network would more likely become the bottleneck.

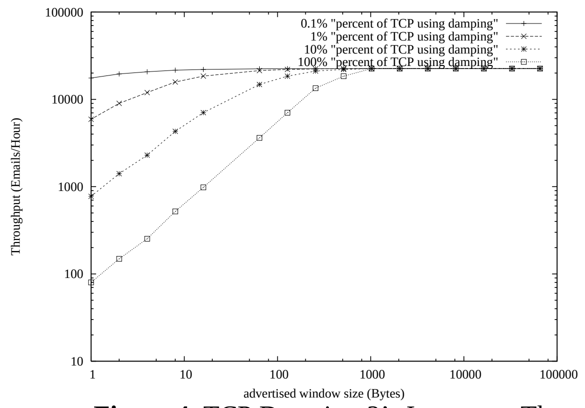

Figure 4: TCP Damping-2’s Impact on Throughput

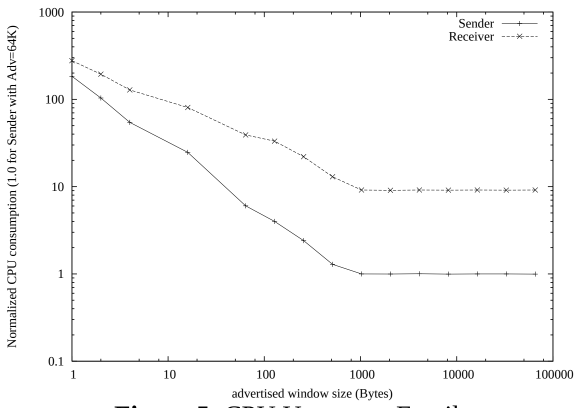

Figure 5: CPU Usage per Email

4.2. Damping TCP by Resource Consumption

The resource bottleneck for spammers could be their network bandwidth, CPU cycles, memory, or port space. Here we evaluate only one of the damping mechanisms: TCP damping-2, assuming the network is the bottleneck. We further assume that the network bottleneck is in the spammer's connection to his ISP. Whether this is the dominant case is still under investigation.

We create a network bottleneck by using the Linux kernel traffic shaper. We set a bottleneck rate to 128 Kbps to simulate the network connection of an ordinary Internet home user. Similar to the experiments in the previous section, we control the percentage of emails sent to the normal receiver and the sticky proxy.

We first evaluate the effectiveness of TCP damping by consuming sender resources. In this experiment, we choose to use 40 processes for the sender that is optimized for high throughput according to the early experiments. The first experiment we conduct is to show the effectiveness of TCP damping with different damping factors. In this experiment, we force the sender to generate more traffic by sending a small advertised window. The result is shown in Figure 4, which shows that by using a small advertised window, the sender’s throughput is reduced by more than 100 times with a high percent of recipients using the TCP damping mechanism. The slow down ratio drops below 10 times when the percentage of recipients that use TCP damping is below 1%.

We also measured the cost of applying TCP damping-2. Figure 5 presents the CPU cost resulting from TCP damping-2. The CPU usage is normalized by the usage of the sender at the time when no damping is applied. As shown in 5, when the advertised window size is normal (larger than 1024 Bytes), a recipient actually needs to consume more CPU and network resources for receiving an email than the amount of resources for sending the same message at the sender. When the advertised window size decreases, both the network and CPU cost increase. The resource consumption ratio gets closer to 1 when the advertised window is small, which means per connection cost for both sides is about equal. The spammer would be defeated by such resource consumption because, compared to the collective resources of all recipients, the spammer’s resources are very limited.

From the above experimental evaluation, we conclude that TCP damping can slow down the spamming rate. Depending on the damping factor and the percentage of recipients using TCP damping, spammer's throughput can be reduced hundreds of times, whereas legitimate bulk email senders would be negligibly impacted. The slowing down of spammers comes with a cost to the receiver, but the cost can be controlled by the recipients.

5. Discussions

TCP damping as an anti-spam tool is not fighting spam by assuming spammers are unaware of the technique. Actually, it is not very hard for a sender to identify a few connections that either last significantly longer than others or consume a larger amount of resources than normal connections. Thus, TCP damping would likely be recognized by the spammer. We argue that spammers would likely choose to terminate a connection because of its long delay or high computation utilization. This gives a hint to the receiver that the sender does not care too much about delivering this message and thus can be identified as a spammer. This would become a great motivation for more people to use TCP damping to defeat spam.

Readers might worry about the impact of damping on open relays (or open proxies). The TCP damping effect is on the immediate email sender, which could be an open relay that is employed by the spammer as an intermediate place to deliver spam. In that case, the open relay's email throughput would be slowed down. The potential threat of this effect is to delay legitimate emails sent through the same open relay. We argue that TCP damping helps the administrator to find the open relay and reduces the potential damages. Most open relays forward email for spammers because the popular mail servers enable open relays by default in early versions. Compared to being black-listed, TCP damping just slows down the legitimate emails. In addition, the high resource load could be a good indicator to the system administrator to find the open relay and close it.

6. Conclusion

In this paper, we propose TCP damping as an anti-spam technique. Compared to the typical pushing mode that a spammer uses to send emails without control from receivers, TCP damping reduces spam by giving receivers control of the progress of each email connection. Our evaluation shows that damping TCP can significantly slow down spammers by simply delaying connections. The delay to a sender depends on the percentage of recipients that insert delays, and it only impacts spammers, not legitimate mailing lists. By consuming sender resources, TCP damping reduces the spammer's sending rate significantly, by requiring each recipient devote a little resources.

Spam and anti-spam are in an ongoing arms race. Similar to many other anti-spam approaches, TCP damping is just one new anti-spam weapon. It can be combined with other anti-spam techniques, such as spam filters and anti-spam honeypots. More important, TCP damping gives incentive to spammers to develop ways to clean up their lists rather than blindly increasing their size.

Reference

[1] Nicholas Graham, “AOL blocks a billion spam emails in 24 hours”, Mercury News Article, Mar 06, 2003.

[2] Sam Vaknin, "The Economics of Spam", published by *United Press International*. July 23, 2002.

[3] Xavier Carreras, and Llues Marquez, "Boosting Trees for Anti-Spam Email Filtering", in *Proceedings of 4th International Conference on Recent Advances in Natural Language Processing*, 2001.

[4] Paul Graham, "A Plan For Spam", in the first SPAM conference, http://www.paulgraham.com/spam.html.

[5] Philip Jacob, "The Spam Problem: Moving Beyond RBLs",http://theory.whirlycott.com/~phil/antispam

[6] Michelle Delio, "Not All Asian E-Mail Is Spam", Wired News article, Feb 19, 2002.

[7] Abby Williams, "Harvard acceptance letters discovered unkosher", *Daily Princetonian*, Feb 15, 2002.

[8] Ari Juels and John Brainard, "Client Puzzles: A Cryptographic Defense against Connection Depletion", in *Proceedings of NDSS-1999* (Networks and Distributed Security Systems), pages 151-165, 1999.

[9] The Penny Black Project, http://research.microsoft.com/research/sv/PennyBlack, Microsoft Research.

[10] C. Dwork and M. Naor, "Pricing via Processing or Combatting Junk Mail", in *Lecture Notes in Computer Science* 740 (Proceedings of *CRYPTO'92*), 1993, pp. 137—147.

[11] J. Postel, "Transmission Control Protocol", RFC 783, IETF, September 1981.

[12] Jonathan B. Postel, "Simple Mail Transfer Protocol", RFC 821, IETF, August 1982

[13] J. Klensin, "Simple Mail Transfer Protocol", RFC 2821, IETF, April 2001.

[14] Axel Zinser, et al, "Teergrubing" http://www.iksjena.de/mitarb/lutz/usenet/teergrube

[15] Marty Lamb, "Using Statistics to Cause Spammers Pain", http://www.martiansoftware.com/

[16] K. Li, C. Pu, M. Ahamad, “TCP Damping”, Tech Report, Georgia Institute of Technology, 2003.

[17] Paul Q. Judge, “SPAM Archive”. http://www.spamarchive.org/

[18] Thorsten Joachims, "Transductive Inference for Text Classification using Support Vector Machines", in *Proceedings of 16th International Conference on Machine Learning*, 1999.

[19] SpamAssassin, http://au.spamassassin.org