Throttling Outgoing SPAM for Webmail Services

Zhenyu Zhong
Dept. of Computer Science
University of Georgia
Athens, GA, USA
zhenyu@cs.uga.edu

Kun Huang
Dept. of Computer Science
University of Georgia
Athens, GA, USA
huang@cs.uga.edu

Kang Li
Dept. of Computer Science
University of Georgia
Athens, GA, USA
kangli@cs.uga.edu

Abstract

Spam has become a serious problem of Internet, and the current defense is limited to the filters deployed at the recipient side. Little known research has been applied to reduce the volume of spam messages being generated. In this paper, we present a system that dynamically throttles emails based on the message content at the email server provider (ESP) side. The goal of this system is to reduce the spam generated by the ESP while not introducing long delay to legitimate messages. This goal is achieved by applying spam filters during the email delivery time and by using filter scores to control the throttling effect. The throttling effect is implemented through a computational puzzle system. We present experiments and results that show the effectiveness of this anti-spam system that under state of the art hardware, we can limit the ability of the spammer even though he possesses 1000 times as many CPU resources as the normal sender.

1 Introduction

A large amount of research and industrial efforts have gone into reducing spam messages. The current practice is heavily limited to spam filtering at the receiver side, but this practice neglects a serious fact that all spam traffic consumes unnecessary internet bandwidth. Also analyzing those huge amounts of spam overloads the receiving mail server. Since sending spam from ESP is a substantial source of spam emails, ESPs suffer when their IP addresses get added onto blacklists [1, 2]. Also complaints from the recipients not only hurts the ESP's reputation, but handling them consumes expensive human support efforts.

This paper addresses the issue of solving those problems by reducing spam messages generated from the ESPs, mostly in forms of webmail services (e.g. Hotmail, Yahoo, and 163.com). The contributions of this work are: 1) *we push the spam filter to the early stage of the email delivery time*, and 2) *we combine the spam filter with the computational cost approach and dynamically assign costs to the senders.*

By pushing the spam filter to the early stage, we can reduce outgoing spam from ESP side and decrease the bandwidth usage in the Internet. Consequently this alleviates the burden on the receiving mail server.

But naively pushing the spam filter earlier at the ESP side is not acceptable, because the ESP would not like to take the responsibility of blocking the suspicious messages. In addition, lack of recipient's preference makes the sender more vulnerable to the false positives.

To address this, we propose a novel mechanism that reduces an ESP's outgoing spam by adaptively assigning computational costs to the senders based on their behaviors and the *quality* of their message content analyzed by the spam filter at the email delivery time. We define the quality of an email as the likeliness of that message being accepted by the recipient as a useful message. Cost-based approaches, such as computational puzzles, have been proposed as a general anti-abuse approach. However, recent research [3] showed that a static cost approach that assigns a constant cost to every email failed to prevent spam. In a similar work by Goodman [4], he discussed solutions to outgoing spam from an economic perspective and designed systems by challenging in the initial period.

Our work is different from previous outgoing SPAM controls in that we challenge each message and overcome the problem of challenges with constant cost. Our approach does not rely on the precise knowledge of which email is spam. We use the spam filter to estimate the quality of a message. The email filter

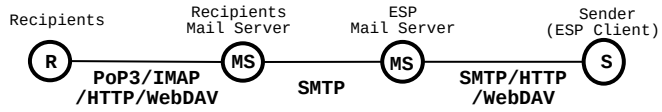

Figure 1: Protocols for Delivering one Email

has to be score-based (like most of the Bayesian filters), and it would produce a spam likelihood score, not a zero-one decision. We choose to delay emails with computational puzzles, and the puzzle difficulty level is based on the filter score. With this approach, users sending low quality messages would be assigned a high computational cost puzzle.

To demonstrate this approach, we integrate this adaptive control and a computational puzzle system into the ESP email delivery system. In today's Internet, delivering one email typically involves multiple protocols. Figure 1 shows the popular protocols used today. Because webmail is the most popular form of email interface for most ESPs, we apply this approach to the webmail interface, including mail systems that use HTTP Post and those with Web-based Distributed Authoring and Versioning (WebDAV) [5].

The rest of the paper is organized as follows: Section 2 reviews the existing work on anti-spam, including efforts at both the recipient and sender side. Section 3 presents our approach in detail within the context of a Web-based email service. We evaluate the approach with emulations. Section 4 presents the emulation results, and finally Section 5 concludes this paper.

2 Related Work

Anti-spam is a very active area of research, and recently many anti-spam techniques have been produced. We classify them into two categories: spam filters and cost based approaches. This section first reviews these two categories and then discusses some existing practices for controlling outbound spam at the ESP side.

2.1 Spam Filters

Most of the current anti-spam research focuses on spam filters. Various forms of filters, such as white-lists, black-lists [1, 2], and content-based filters [6] are widely used to defend against spam. White-list based filters only accept emails from known addresses. Black-list filters block emails from addresses known to send out spam. Content-based filters make estimations of spam likelihood based on the text of that email message and filter messages based on a pre-selected likelihood threshold. For example, the famous filter from Paul Graham [6] assigns a likelihood value to each word or phrase based on its history of use in spam and takes the average as the overall spam-likelihood for the message.

Unfortunately all types of spam filters have false positives, with which legitimate messages are misclassified and get lost. Another problem with spam filters is that it can only filter a message after it has already been delivered and stored in the receiver's mail server.

The approach presented in this paper also uses spam filters, but for a different purpose – not filtering messages, but estimating their "quality". The quality of the information is then used for selectively delaying messages. Thus a misclassification would only cause a small delay to a message, and the impact of a false positive would be much less severe than the method of dropping messages. This approach is applied at the sender side to reduce outgoing spam, thus it can be used as a complementary technique for the current filtering methods at the recipient side.

2.2 Cost-based Approach

A cost-based approach is the most promising general solution for resisting network abuse, such as spam [7, 8] and network DoS attacks [9, 10]. Cost takes many forms, such as monetary payments [11], "hashcash" [12], and computational puzzles [13]. By requiring the remote peer to consume some computational resources before granting the service, the protected side can reduce the risk of network abuse.

The most famous adoption of the cost approach is probably the challenge-response anti-spam scheme [14]. It has been used by Earthlink and a few other ESPs to filter incoming messages. With this scheme, a mail server automatically returns a challenge message which requires the client to perform a task, such as reading a picture, before it will deliver the message to the final recipient.

Dwork and Naor [13] proposed a general mechanism that requires a sender to compute a moderately hard pricing function or cryptographic puzzle for each message; the cost to compute the pricing function is negligible for normal users, but high for mass mailers. Recently, the use of cost-based approaches [7, 8, 15, 16] mostly address server resource exhaustion.

2.3 Previous Work on Outgoing Spam

Reverse turing test is one well-known cost approach that has been widely adopted by many ESPs to reduce spam. In this approach, users are required to pass a simple test (e.g. reading text strings from a picture) before getting an account. Some ESPs (e.g

mail.sina.com) even move a step further and require a reverse turing test before sending any email messages.

A recent work by Goodman et. al. [4] shows that the sign up cost of the reverse turing test is not large enough to deter spammers, and they propose an alternative that periodically imposes costs on senders only at the early signing up stage. Goodman et. al. shows that these costs at the initial stage is enough to deter spammers, thus reducing outbound spam messages for the ESP. Also Clayton[17] proposes to find spam senders by inspecting ISP logs. We can combine this log information with the throttling control proposed in this work by providing additional input information. Besides, rate limiting [18] is not a new idea, and it has been applied to emails. Our approach differs from previous works in that we provide an automatic control that drives the throttling effect rather than choosing a fixed throttling level.

Our approach in this paper requires no human interaction for either tracking the senders or assigning and solving reverse turing tests. Instead, it automatically assigns computational costs based on two factors, the individual email quality and the overall outgoing messages quality.

2.4 Dynamic Cost Control

While the possibility of adding delay and cost to abusers has been considered previously in works such as teergrubing [19] and tarpit [20], the works are limited to the recipient side. In our previous paper [21], we studied the effects of introducing cost at the network transport layer on the email recipient side. The work presented in this paper is similar to our previous work in the sense of selectively applying cost. The difference is that our previous work is purely recipient centered and requires a considerably large deployment to be effective. The work in this paper is completely on the sender side, and because all the users have to be authenticated by ESPs and therefore use their proprietary process to forward messages, it is easier to deploy.

3 Our Approach

This section presents our approach of adaptively reducing outgoing spam on the ESP side. We first review the current email relaying practice of ESPs, and then we explain how to build the cost mechanism into the ESP message relaying process. Finally we present our adaptive cost assigning system that selectively adds cost to users.

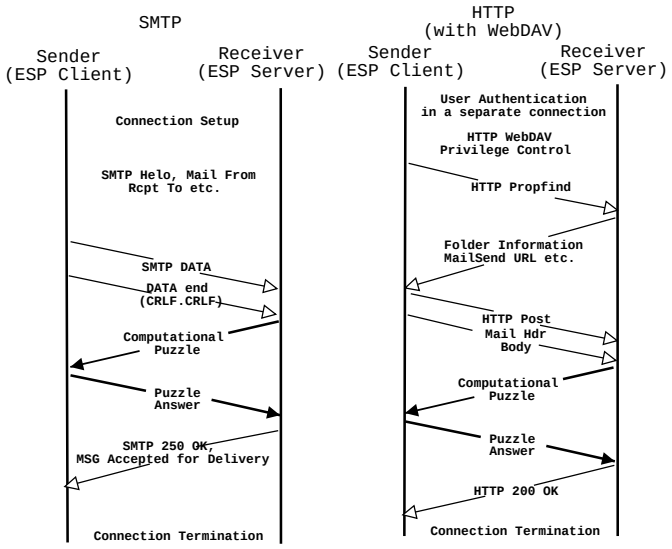

Figure 2: ESP Mail Protocols with Cost Mechanisms

3.1 ESP Email Delivery Protocol

To our knowledge, the current practice is limited to the following protocols: (1)SMTP, (2)HTTP, and (3)HTTP with WebDAV. The latter two are more popular for web based ESP service (used by Hotmail, yahoo, mail.sina, etc) because they provide identifications to the ESP. When HTTP is used, messages are delivered to the server with the HTTP Post command. WebDAV is an extension of HTTP that is designed to enable multiple users to manage and modify the files in a remote system. With WebDAV enabled clients, users can view, open, edit, and save files directly into the filesystem of the website as if it was a local system. Since email data are still delivered through HTTP Post command, we present the mail client with WebDAV in the same way as the client purely using HTTP.

3.2 Cost Mechanism

The goal of this work is to integrate the cost approaches into these systems and show that putting spam filters at an early stage of email delivery can help reduce the spam from the sender. With the cost mechanism, the server would be able to assign a computational task to the client with a controllable difficulty. The server would then verify the computational results before accepting the messages for forwarding. The cost mechanism has to be robust, tamper resistant, and efficient. Many existing studies [7, 8] have addressed these issues and designed algorithms for this task. We are not going to repeat this task. Instead, we focus on how to combine them with spam filters.

We picked a simple computational puzzle algorithm for our system. In this algorithm, when a sender makes a

connection and delivers a message to the ESP server, the server randomly generates a string for this connection, calculates and saves the MD5 hash output, and sends the hash output back to the sender (the ESP client). The email sender is asked to search for a string that has the same hash output and send back the string as the answer. The server controls the puzzle complexity by controlling the string length and the search space size.

Since we want to determine the computational complexity based on the message content, the server can only generate the computational puzzle after the message arrives and passes through our spam filters (to get a quality estimation).

Figure 2 illustrates where the computational mechanism is inserted into the original email delivery process for the two protocols, SMTP and HTTP, respectively. When the client uses SMTP to forward messages to the ESP server, the client's SMTP agent has to be modified. When SMTP is used, and ESP has no control over which SMTP client a user adopts, then adding this mechanism would be considerably harder than the HTTP or HTTP with WebDAV cases. For the later case, the client side software is embedded in the web interface, which can be easily modified by the ESP server to add this cost mechanism, by using a client side script.

Notice that the ESP server has to enforce this mechanism in the sense that if a sender failed to supply the result of the computational puzzle associated with a message, the ESP would refuse to forward the message.

Because the ESP can also associate each email delivery attempt with a user account, the ESP can also apply more advanced cost control based on the account's overall behavior. For example, cost could be doubled if many email bounces happen, which is a good indication of sending unsolicited messages.

Penalties could also be used when a client refuses to send back answers but keeps making delivery attempts.

3.3 Selective Cost Assignment

With the knowledge of where in the delivery process we assign the cost, this subsection describes the algorithm of assigning puzzle difficulty.

We chose two guidelines for the difficulty assignment. First, we would like to assign no computational cost to every connection if the spam messages are very rare overall. Second, we would like to assign no or negligible computational cost to good email messages even when the overall spam volume is high.

To achieve this goal, we design a two level adaptation system in which an email connection's cost is assigned based on a product $C(m) = Q \times q(m)$, in which $C(m)$ is the cost level for a message m, and Q is the overall average message quality level measured over a recent history, and its value is between 0 (low spam ratio) and 1 (high spam ratio), and $q(m)$ is the quality measurement for this individual message with a value also ranging from 0 to 1.

Both the overall and the individual message quality measurements are made by a spam filter. Although spam filters can't judge the spamminess of a message with a 100 percent accuracy, the average score over many messages gives a good indication of the spam-and-non-spam ratio.

We choose a Bayesian based spam filter called QSF (quick spam filter) [22] for the message quality estimation at the ESP mail server side. QSF is a lightweight statistical spam filter written in C. In QSF, an overall score is calculated to find out whether the email should be considered spam or not. An evaluation of QSF by a third party shows its filtering precision is 99.1% with a 0.27% false negative and a 0.02% false positive rate. [23]

With this approach, there is still a large design space for choosing an adaptation algorithm. A few issues need to be addressed including over how long a period should the average quality measurements be made, how responsive should the system be towards message quality changes, and how much we adjust the cost each time we sense the quality changes.

We ended up choosing one proportional control algorithm, in which the cost level is assigned with the following equation:

- If $\overline{S} - S_m > 0, Q = P \times (\overline{S} - S_m)^i$
- If $\overline{S} - S_m <= 0, Q = 0$

Q is the cost we want to calculate, $\overline{S}$ is the average email score over recent period of time, we update the $\overline{S}$ periodically, S_m is the mean score value of the good emails from QSF training set. We calculate the distance between the $\overline{S}$ and S_m, and we raise this distance to the power of i, so that when the quality of emails are low, the score will be most likely high, and sender will get more punishment for those low quality emails. P is a multiplier used to map the value into the puzzle generator's input range.

Even with this algorithm, there are several challenges towards achieving this goal. We need to make the false positive impact as tiny as possible when the spam filter makes a low quality estimation for a good mail. We

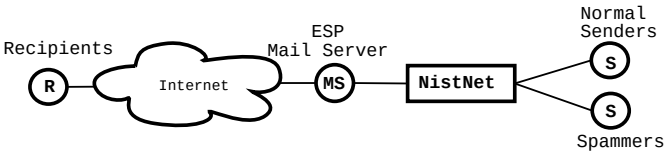

Figure 3: Emulation Topology

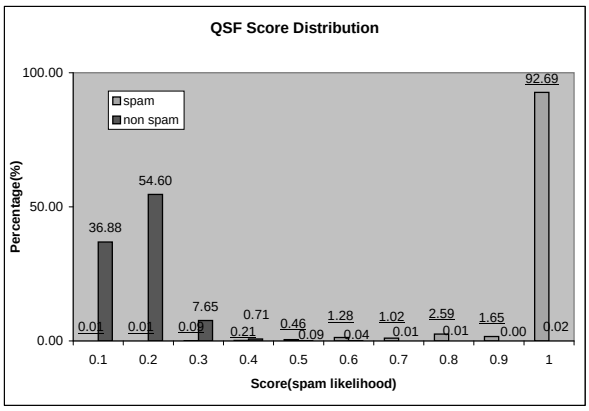

Figure 4: Email Score Distribution (The number with underline is the spam's percentage number.)

also need to avoid high processing overhead, so that the ESP server can still support a large number of accounts.

To address this limit, we set an upper bound to the cost level, so that even the maximum cost level would not cause a connection to delay more than 5 minutes. This number is estimated assuming the same level of computational power as our experiment machine at the ESP client side.

People have concerns that applying cost proportionally to the amount of messages might affect legitimate bulk email senders, such as Amazon and eBay. However, legitimate bulk email senders have motivations to identify themselves with the ESP so that they can be put on the white-list to avoid these computational cost. Furthermore, the cost is not only related to the message volume, but also the message quality. The aggregate cost for a large volume of good quality messages is still low.

4 Evaluation

This section presents an evaluation of our adaptive throttling system at the ESP. We first present the experiment methodology, including the experiment setup and the metrics we used to evaluate the system. Then we present the empirical results for both with and without the adaptive throttling system.

4.1 Evaluation Methodology

We evaluate the adaptive throttling approach through emulation. In the emulation, we setup a modified sendmail server as the ESP mail server, which accepts email from users through a webmail interface.

Figure 3 illustrates the topology used in our experiments. All the machines in the systems are 2.6GHz Dell PCs, running Linux 2.4.23, and they are connected through a 100Mbps switches.

The mail server is supposed to forward messages to the Internet. Since we only care about the quality of the outgoing email messages, we forward all the messages to /dev/null.

We use two machines to emulate normal senders and spammers. Both machines connect to the email server through a NIST Net router that emulates the network between the clients and the ESP mail server.

Because we are emulating many senders using a single machine, in our emulation, the cost of a computational puzzle given to the sender is reduced proportionally. A similar arrangement happens at the spammer side. The resource ratio is controlled with a parameter, and results of different ratios are presented in the next subsections. The cost mechanism is based on a MD-5 based client puzzle system described in the previous section.

We use real email messages (38591 non spam messages, and 18800 spam messages) obtained from the Internet for the experiments. The source of the non-spam messages are obtained from several mailing-lists, including the well-known end-to-end [24] and perl monger [25]. The spam messages are obtained from the spamarchive [26] for the archived messages from Mar 19, 2004 to Sep 14, 2004. The distribution of the email scores used in our experiments is illustrated in Figure 4.

To play like a spammer, we mapped the mail queue to the ramdisk, and also turned off the mail log. To remove the overhead of the disk i/o, we also redirect unnecessary output to /dev/null.

In our evaluation, we consider two possible spammer strategies. One is with a best effort approach, in which spammers keep sending as long as they have resources regardless of the cost the ESP assigned to them. The other strategy assume the spammers adapt their behavior based on the cost and only try to send messages when the cost is low. If an attempt to send leads to a high cost, the spammer abandons it immediately without devoting any resources.

To evaluate the effectiveness of the throttling system, we chose to measure three metrics: the *spam ratio,* the

goodput and the *normal email delay*.

The spam ratio is the percent of outgoing messages that are spam. We expect an ESP's outgoing messages have a low spam ratio, so that the ESP will not get onto black-lists or receive complaints. However, the spam ratio is not the only metric that concerns an ESP. The ESP should not drop all the messages, It has to keep a high throughput for legitimate messages. Therefore, we look at a second metric: the goodput. The goodput is the non-spam email throughput. We expect an effective control system can keep high goodput even with many spam attempts. We also measure the normal email delay in relaying the messages from the webmail interface to the mail server. Here the delay is measured by logging the connection initiation time and the time when the server accepts the message for delivery. The delay is the time difference between the two, which includes the time spent in generating and solving the computational puzzle. We measure both average delay and the worst delay for non-spam messages. Ideally, an effective anti-spam system should introduce very low delay to non-spam messages.

4.2 Effectiveness

As a reference to measure the effectiveness of this outgoing spam control, we first measure the overall spam ratio, the goodput, and the delay behaviors of normal emails.

We run a server with both emulated normal users and spammers. We further assume that the server supports 100,000 users, and each user in average sends five emails a day. This number was obtained from a recent study on a British ESP [3]. Our own measurement over a nation wide ESP shows a similar rate. We control the normal email rate according to this average, and we emulate a spammer that sends emails in a *best-effort* way. In our measurements, we found the CPU is the bottleneck for email senders, rather than memory or bandwidth. With the best-effort strategy, the spammer automatically accept the whatever cost assigned from the ESP. We vary the spammer's CPU resources to show its impact to the spam ratio and the goodput and delay for normal messages. This result is presented in Figure 5. In this result, the spammer's CPU resources are represented by its ratio to the normal users' average computational power. Typically the ratio is around 1, meaning that spammer uses a similar powerful machine as a normal user does. We consider the typical ratio range is between 0.1 and 10. The ratio is increased when spammer has a top-of-the-line system, or compromises a good number of zombie machines for sending emails. So we also consider some larger ratio to represent this scenario. The

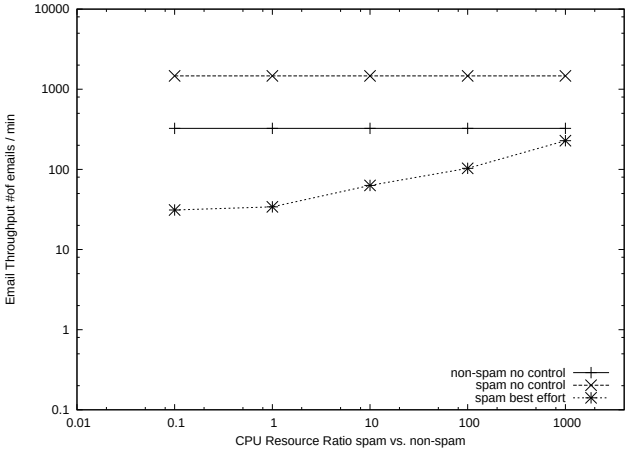

Figure 5: Throughput with best-effort spammers

Table 1: Delay for the normal messages with best-effort spammers (SD: Standard Deviation)

Resource Ratio	Average Delay(sec)	SD
0.1	0.56	1.46
1	0.85	3.51
10	2.43	6.66
100	6.9	10.63
1000	28.33	17.22

result indicates that when the spam volume increases proportionally with the spammer's resources (without requiring very powerful systems) the spam volume can bypass that of normal email messages (which is the goodput). The spam volume stops increasing once the email rate is high enough to hit the server's maximum throughput. Under this situation, the majority of messages are spam. Arguably, the Internet email system is getting close to this situation, given reports that more than 50 percent of Internet messages are spam.

Now we look at email throughput as well as delay when the ESP uses our adaptive cost control algorithms. In this section, we assumed all the senders automatically accept the cost assigned from the ESP. We expected the spammers would suffer with the high cost assigned to them because most of their messages would have a high spam score. Certainly spammers could choose to abandon some connections to avoid high computational costs and optimize their throughput. We discuss this situation in the next subsection.

Figure 5 shows the resulting throughput for various spam resource amounts. Overall, the spam ratio is decreased after we apply our control system. For example, when the spammer has the same CPU resource as normal user, the spam ratio before control is 0.81, it drops down to 0.09 after we use the control sys-

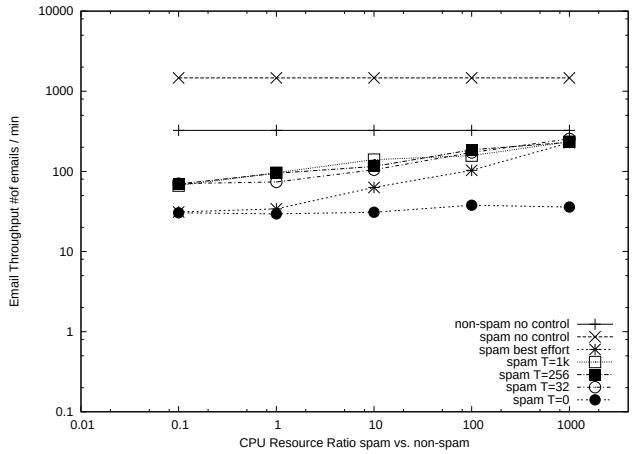

Figure 6: Throughput with "smart" spammers

Table 2: Delay for the normal emails with "smart" spammers (SD: Standard Deviation)

Resource Ratio	Average Delay(sec)	SD
0.1	2.97	6.01
1	4.35	8.05
10	9.15	12.89
100	13.69	14.03
1000	29.49	18.07

tem. This benefit of a lower spam ratio and a higher goodput comes with a small impact to normal emails: the per email delay could increase. To quantify this impact, we present the measured normal email delay result in Table 1, including both average delay and standard deviation of the average delay. The result is that most of the messages have very low delay when the spammer's resources are low or comparable to a normal user's resources. The delay increases when the spammer's resources get higher. However, the average delay is still within tens of seconds. The worst delay to legitimate emails are controlled below the maximum cost level which is 5 minutes. This time interval has been commonly used in the email delivery for timeout value, such as the SMTP commands.

4.3 Smart Spammer

In this section, we consider a "smart" spammer, who selectively chooses to accept the ESP's cost assignment in order to send messages or chooses to abandon the messages when the cost is too high.

In this evaluation, we did not define the smartest strategy for spammers. Instead, we tried a range of spammer strategies, and studied the outcome of the spammer's throughput. A common aspect of these strategies was that the spammer chose to send only when the cost assigned from the ESP was below a threshold. In our evaluation, we tried static thresholds and also thresholds adjusted based on the sender side resources. The thresholds tried in our experiments include (1)$threshold = 0$, which means the spammer only sends message when the cost is zero. (2) $threshold = infinity$, which means the spammers always accepts and solves the computational puzzle regardless of the complexity level. Essentially it is the same as the best-effort. (3) $threshold = T$, the spammer only sends messages when the cost is below T, and abandons the connection for an assigned cost higher than T. We exponentially tried several T, as shown in Figure 6, the throughputs of different T didn't make much difference, but the per email average delay will be higher when the spammer choose a higher throughput.

The result of the email throughput is presented in Figure 6, and the normal email delay whose T equals to *1k* is presented in Table 2. The resulting spammer throughput for all the thresholds we chose are all below the normal email throughput for various spammer resource setups. And to achieve a high spam throughput, no matter which threshold a spammer chooses, he inevitably has to increase his computation capacity. This will definitely limit the ability of the spammer to abuse the email system with limited computation resources. And in this measurement, the spam ratio also decreased after we apply our control system in overall. For example, when the spammer has the same CPU resource as normal user, the spam ratio decreases from 0.81 to below 0.22 under different thresholds we attempt.

In regard to the delay result, with smart spammer strategies the delay result for normal emails is only a few seconds longer than the case with best-effort spammers. Although this result does not necessarily show the highest possible throughput for a spammer, it does show that the spammer cannot achieve a very high throughput using only a simple strategy.

The above results only cover a special group of spammer's strategies. Another interesting strategy for spammers is to send non-spam along with spam messages. With the hope of bringing up the overall quality and keeping our control system assigning low computational cost, the spammer might get a good amount of spams out. One way to address this problem is to keep an absolute counter for emails that have a filter score high enough, and take this into account when calculating the overall quality of emails, rather than only using average filter score. Certainly, investigating the best available strategy for spammers deserves more study, and we plan to pursue this in our future work.

5 Conclusion

A web-based email service is the most popular interface used by the existing email service providers. In this paper, we presented an anti-spam system for ESPs in order to reduce spam messages originating from them. The system dynamically assigns costs based on the estimated quality of the messages, and the quality is derived from scores produced by a spam filter. Experiments show that by dynamically assigning the cost based on the message quality, the system slows down spammers but assigns zero (or little cost) to the normal messages because they tend to have high quality. The majority of normal messages belong to this category. Misclassified messages (false positives) would not be dropped but only incur slight delay.

Acknowledgment

We would like to thank Aura Morris, Bradley J. Wimpey, Barry Rountree, Rebekah Black for their reviews and suggestions for the paper, and Shifeng Chen for providing ESP measurement data.

References

[1] Philip Jacob. The Spam Problem: Moving Beyond RBSs, 2003. `http://theory.whirlycott.com/rbl`.

[2] Michelle Delio. Not All Asian E-Mail is Spam. In *Wired News*, Feb 19 2002.

[3] Ben Laurie and Richard Clayton. "proof-of-work" proves not to work. In *Proceedings of the Third Annual Workshop on Economics and Information Security (WEIS04)*, May 2004.

[4] Joshua T. Goodman and Robert Rounthwaite. Stopping outgoing spam. In *Proceedings of the 5th ACM conference on Electronic commerce*, pages 30–39. ACM Press, 2004.

[5] Paul Festa. Microsoft anti-spam campaign hypocritical. available at http://news.zdnet.co.uk/business/.

[6] Paul Graham. A Plan for Spam, 2003. `http://spamconference.org`.

[7] A. Juels and J. Brainard. Client Puzzles: A Cryptographic Defense Against Connection Depletion. In *NDSS*, pages 151–165, 1999.

[8] D. Dean and A. Stubblefield. Using Client Puzzles to Protect TLS. In *10th Annual USENIX Security Symposium*, 2001.

[9] F. Kargl, J. Maier, and M. Weber. Protecting Web Servers from Distributed Denial of Service Attacks. In *World Wide Web*, pages 514–524, 2001.

[10] J. Jung, B. Krishnamurthy, and M. Rabinovich. Flash Crowds and Denial of Service Attacks: Characterization and Implications for CDNs and Web Sites. In *International World Wide Web Conference*, pages 252–262, May 2002.

[11] D. Mankins, R. Krishnan, C. Boyd, J. Zaho, and M. Frentz. Mitigating Distributed Denial of Service Attacks with Dynamic Resource Pricing. In *Proceedings of Annual Computer Security Applications Conference (ACSAC 2001)*, 2001.

[12] A. Back. Hashcash: A Denial of Service Counter-Measure. Technical report, Cypherspace, August 2002. `http://cypherspace.org/hashcash/hashcash.pdf`.

[13] C. Dwork and M. Naor. Pricing via Processing or Combatting Junk Mail. In *Crypto*, August 1992.

[14] Jonathan Krim. 'challenge-response' technology rejects messages unless senders are cleared by recipients. Washington Post News article, May 7, 2003.

[15] T. Aura, P. Nikander, and J. Leiwo. DoS-Resistant Authentication with Client Puzzles. *Lecture Notes in Computer Science*, 2133, 2001.

[16] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. Wallach. Security for Peer-to-Peer Routing Overlays. In *Proceedings of OSDI*, December 2002.

[17] Richard Clayton. Stopping Spam by Extrusion Detection. In *Proceedings of the First Email and SPAM conference*, July 2004.

[18] Hotmail sets email limits. `http://news.zdnet.co.uk/business/0,39020645,2132358,00.htm`.

[19] Axel Zinser. Teergrubing. `http://iksjena.de/mitarb/lutz/usenet/teergrube`.

[20] Marty Lamb. Using statistics to cause spammers pain. `http://www.martiansoftware.com`.

[21] Kang Li, Calton Pu, and Mustaque Ahamad. Resisting SPAM Delivery by TCP Damping. In *Proceedings of the First Email and SPAM conference*, July 2004.

[22] Andrew Wood. Quick spam filter. available at http://freshmeat.net/projects/qsf/.

[23] Sam Holden. Spam filter evaluations. available at http://sam.holden.id.au/writings/spam2/.

[24] The end2end-interest archives. available at http://www.postel.org/pipermail/end2end-interest/.

[25] Perl monger mailinglist. available at http://mail.pm.org/archives/classiccity-pm/.

[26] Paul Judge. The spam archive. available at http://www.spamarchive.org.