Available online at www.sciencedirect.com
ScienceDirect
Computer Networks 51 (2007) 2616–2630
Computer Networks
www.elsevier.com/locate/comnet

DMTP: Controlling spam through message delivery differentiation ☆

Zhenhai Duan [a,*], Yingfei Dong [b], Kartik Gopalan [c]

[a] Computer Science Department, Florida State University, Tallahassee, FL 32306, USA
[b] Department of Electrical Engineering, University of Hawaii, Honolulu, HI 96822, USA
[c] Department of Computer Science, Binghamton University, Binghamton, NY 13850, USA

Received 14 September 2006; accepted 20 November 2006
Available online 22 December 2006

Responsible Editor: Ian F. Akyildiz

Abstract

Unsolicited commercial email, commonly known as spam, has become a pressing problem in today's Internet. In this paper, we re-examine the architectural foundations of the current email delivery system that are responsible for the proliferation of email spam. We argue that the difficulties in controlling spam stem from the fact that the current email system is fundamentally sender-driven and distinctly lacks receiver control over email delivery. Based on these observations we propose a Differentiated Mail Transfer Protocol (DMTP), which grants receivers greater control over how messages from different senders should be delivered on the Internet. In addition, we also develop a simple mathematical model to study the effectiveness of DMTP in controlling spam. Through numerical experiments we demonstrate that DMTP can effectively reduce the maximum revenue that a spammer can gather. Moreover, compared to the current SMTP-based email system, the proposed email system can force spammers to stay online for longer periods of time, which may significantly improve the performance of various real-time blacklists of spammers. In addition, DMTP provides an incremental deployment path from the current SMTP-based system in today's Internet.

Keywords: Email spam; Unwanted internet traffic; SMTP; DMTP

1. Introduction

Unsolicited commercial email, commonly known as *spam*, is a pressing problem on the Internet. In addition to undermining the usability of the current email system, spam also costs industry billions of dollars each year in recent times [14,33]. In response, the networking research and industrial communities have proposed a large number of anti-spam countermeasures, including numerous email spam filters [5,9,18,17,29,30,32], sender authentication schemes [11,24,26], and sender-discouragement mechanisms (to increase the cost of sending email such as paid email) [16,22]. Some of

☆ A preliminary version of this paper appeared in the Proceedings of IFIP Networking 2006.

* Corresponding author. Tel.: +1 850 645 1561; fax: +1 850 644 0058.
E-mail addresses: duan@cs.fsu.edu (Z. Duan), yingfei@hawaii.edu (Y. Dong), kartik@cs.binghamton.edu (K. Gopalan).

1389-1286/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.comnet.2006.11.015

the schemes have been deployed on the Internet. On the other hand, despite these anti-spam efforts, in recent times the proportion of email spam seen on the Internet has been continuously on the rise [6,9]. In order to fundamentally address the email spam problem, it is our contention that we must understand the protocol features that have been exploited by spammers, and design and deploy new email delivery protocols and architectures that can inherently resist spam.

1.1. Why is it so hard to control spam?

The current email system uses the Simple Mail Transfer Protocol (SMTP) to deliver messages from sender to receiver [23]. While simple, such a system also provides an ideal platform for spammers to act as parasites. As a first step towards the goal of designing spam-resistant email delivery architectures, in this paper we examine the architectural aspects of the current email system that are responsible for the proliferation of spam and propose a novel Differentiated Mail Transfer Protocol (DMTP) that aims to overcome these limitations based on the following three key insights.

Moving to a receiver-driven model: First, the current email system is fundamentally sender-driven and distinctly lacks receiver control over the message delivery mechanism. For example, in the current SMTP-based email system, any user can send an email to another at will, regardless of whether or not the receiver is willing to accept the message. In the early days of the Internet development, this was not a big problem as people on the network largely trusted each other. However, since the commercialization of the Internet in the mid-1990s, the nature of the Internet community has changed. It has become less trustworthy, and the emergence of email spam is one of the most notable examples of this change [3]. In order to effectively address the issue of spam in the untrustworthy Internet, we argue that *receivers must gain greater control over if and when a message should be delivered to them* [27].

Eliminating economy of scale: Secondly, volume is the most crucial factor in making email spam a profitable business. In order to squeeze spammers out of business, we must eradicate the economy of scale they rely on. However, in the current email system, the sending rate of spam is, to a large extent, only constrained by the processing power and network connectivity of spammers' own mail servers, of which the spammers have complete control. Nowadays, with increasingly-powerful (and cheaper) PCs and ubiquitous high-speed Internet access, spammers can push out a deluge of spam within a very short period of time, making spamming profitable because of the economy of scale. We contend that *the sending rate of spam must be regulated, ideally under the control of email receivers*, in order to retain spam.

Increasing accountability: Lastly, the current email system makes it hard to hold spammers accountable for spamming. Spammers can vanish (go offline) immediately after pushing spam to receivers (recall that this can be done within a very short period of time). This makes it quick and easy for spammers to hide their identities and provides spammers with the flexibility to frequently change their locations and/or Internet service providers – complicating the effort to filter spam based on the IP addresses of sender mail servers, such as various real-time blacklists (RBLs) [29]. Indeed, recent studies [13,28] on the behaviors of spammers at both the mail server and network levels showed that the majority of spammers are only active for a short time period. For example, it was shown [13] that 81% of spam only mail servers and 27% of spam only networks sent spam only within one day out of a two-month email trace collection period. We argue that in order to hold spammers accountable and to make RBLs more effective, *we must force spammers to stay online for longer periods of time*.

1.2. Contributions of this paper

Based on these observations we propose a Differentiated Mail Transfer Protocol (DMTP) that is inherently spam-resistant. A key feature of DMTP is that it grants receivers greater control over the message delivery mechanism. In DMTP, a receiver can classify senders into different classes and treat the delivery of messages from each class differently. For example, although regular contacts of a receiver can directly send messages to the receiver, unknown senders need to store messages in the *senders' own mail servers*. Such messages are only retrieved by the receiver *if and when* the receiver wishes to do so.

DMTP provides us with several important advantages in controlling spam: (1) the delivery rate of spam is determined by the spam retrieval behavior of receivers instead of being controlled by spammers; (2) spammers are forced to stay online for longer periods of time (because the sending rate of

spam is regulated by the spam retrieval rate of receivers), which can significantly improve the performance of RBLs; (3) regular correspondents of a receiver do not need to make any extra effort to communicate with the receiver – correspondence from regular contacts is handled in the same manner as in the current SMTP-based email system; and (4) DMTP can be easily deployed on the Internet incrementally.

In this paper, we first study the implications of protocol design choices, in particular, the application-level communication models, for the control of unwanted Internet traffic such as email spam. Based on the insights obtained, we then present the design of DMTP and formally model its effectiveness in controlling spam. Through numerical analyses we show that DMTP can significantly reduce the maximum revenue that a spammer can obtain. In addition, a spammer has to stay online for a much longer period of time in order to obtain the maximum revenue. Moreover, DMTP also helps reduce the total amount of spam traffic traveling on the backbone networks of the Internet.

The remainder of the paper is organized as follows. In Section 2 we re-examine the common application-level communication models on the Internet and discuss their implications for controlling spam. In Section 3 we present the design of DMTP using the sender-intent-based receiver pull communication model. We formally model the effectiveness of DMTP in controlling spam and conduct numerical analyses to contrast the performance of DMTP and SMTP in Section 4. In Section 5 we discuss a few practical deployment issues of DMTP on the Internet. After describing related work in Section 6, we conclude the paper and outline our ongoing work in Section 7.

2. Push vs. pull: implications of protocol design choice

The choices made during the protocol design phase have fundamental implications for security, usability, and robustness of any distributed message delivery systems. One such important design decision is whether to adopt a sender-push or a receiver-pull model or a combination of the two models (see Fig. 1). In this section we discuss the implication of these design choices and make the case that the receiver-pull model can prove to be highly effective in discouraging unwanted traffic including email spam. Based on the insights obtained here, in the next section we present the design of the new Differentiated Mail Transfer Protocol (DMTP) that has many desirable properties in controlling spam.

2.1. The sender-push model

In the sender-push model, the sender knows the identity of a receiver in advance and pushes the message in an asynchronous manner to the receiver. The receiver accepts the entire message, may choose to optionally examine the message, and then accept or discard it. An important aspect of sender-push model is that the entire message is received before any receiver-side processing is performed. A number of communication services in the Internet rely on the sender-push model. A prime example is email in which the sender relies on the Simple Mail Transfer Protocol (SMTP) to push an entire email message to a passive receiver. Asynchronous voice messages over the telephone network (both traditional and IP based) represent another important application of the sender-push model.

A common variation of the sender-push concept is the *receiver-intent-based sender-push* (RISP) model (Fig. 1). The most common examples of the RISP model are the subscription-based services such as mailing lists, where users need to subscribe to the services to get content. Similarly, Instant Messaging is another application where the message itself is pushed by the sender, but the receiver can allow or disallow messages from specific users. Other popular applications of the RISP model include stock and news ticker applications and automatic software updates, where user subscribes to a service which subsequently pushes the data to the receiver.

A common feature among all the above examples is that the content itself is pushed to the receiver, whereas the receiver may optionally provide mini-

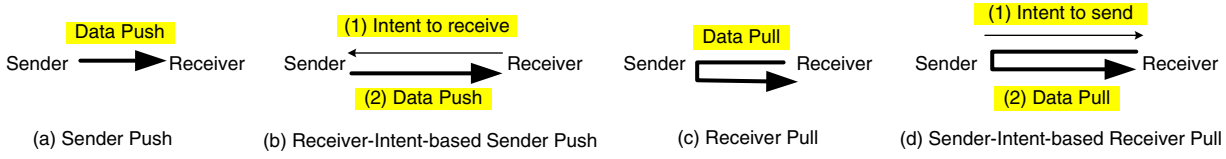

Fig. 1. Common application-level communication models.

mal control feedback to the sender. The primary advantage of the sender-push model is that its asynchronous message delivery framework is conceptually simple and fits naturally for many useful applications such as email and text messaging. Sender initiates message transfer when the message is ready, the receiver simply waits passively for any message to arrive and accepts one when it does arrive. Furthermore, there is no significant storage requirement on the sender side.

The biggest disadvantage of the sender-push model is that it is the sender who completely controls *what* message is delivered and *when* it is delivered. The receiver has neither the knowledge of what message he will receive, nor when he will receive the message. The receiver is ideally expected to receive the entire message before processing or discarding it. Apart from generating and transmitting the message, the sender does not commit any resources for the transmitted message. On the other hand, the receiver has to wait, receive, process and store (or discard) the message even if the message is not of interest to the receiver.

The RISP model alleviates this concern to some extent by allowing receivers to provide control feedback. However, it is not easy to implement in many popular applications. For example, adopting the RISP model for applications such as email, mobile text and voice messages requires the receiver to maintain an exhaustive white-list or black-list of email addresses and phone numbers of potential senders. Indeed, approaches such as RBL [29] adopt this philosophy in trying to blacklist email spammers. However, most potential correspondents, such as first time senders, fall in neither of the two categories. To handle such unclassified cases, receivers end up relying on content-based-filters, i.e. they receive the entire message, scan it to determine if it is wanted and then either accept or discard it. *The fundamental problem here lies in having to accept and examine the entire message before culling it.* Note that professional spammers are paid based on the number of messages delivered to the receivers (or rather their mail servers) instead of the number of receivers responding to the spam [16]. This is one of the key reasons that spammers have the motive for sending more messages.

An additional disadvantage of the sender-push model is that the sender can vanish (go offline) immediately after pushing unwanted content to the receiver. This makes it quick and easy for a malicious sender to hide its identity. Once the receiver accepts the content, it is difficult at best to trace back a malicious sender.

In summary, while the sender-push model is both simple and convenient, it comes with a serious baggage, namely, that senders control what to send and when to send, and cannot be easily held accountable for sending unwanted content to receivers.

2.2. The receiver-pull model

In the receiver-pull model (Fig. 1), it is the receiver who initiates the message transfer by explicitly contacting the sender. The sender passively waits for the receiver and delivers the entire content upon receiving a request. Since it is the receiver who initiates the message transfer, the receiver would have explicit greater control over the message transfer and implicit greater trust in the received content, than in the sender-push model.

A number of successful communication services rely on the receiver-pull model. The most important examples using the receiver-pull model are the FTP and HTTP protocols. In both cases, the receiver initiates the data transfer by opening an FTP connection and by typing/clicking on a URL, respectively. (Interestingly, HTTP supports both receiver-pull and as well as sender-push, though the former is more commonly used. Examples of sender-push techniques in HTTP include automatic page refreshes and the hugely unpopular popup windows.)

An interesting and useful variation of receiver-pull model, which is of special interest to us, is the *sender-intent-based receiver-pull* (SIRP). In this model, the sender first expresses an intent to send content to the receiver via a small intention message. If the receiver happens to be interested, it contacts the sender and retrieves the content. A common example of the SIRP model is the *pager* service. Here the caller expresses an intent to talk to a callee by paging the latter and leaving a callback number. If the callee is interested, he contacts the caller back on the callback number. The main feature of the SIRP model is that the content itself is pulled by the receiver whereas only a short intent is pushed by the sender.

The advantage of the receiver-pull model is that a receiver exercises control over when and what it receives. The receiver has the freedom to first determine its own level of interest in the content (as well as the reputation of the sender) *before* it actually

requests the content. Furthermore, it becomes the responsibility of the sender to store and manage the content till the receiver is ready to retrieve it. For instance, an FTP or web server needs to store and manage its own files whereas receivers access it only when they are interested. Additionally, there is a large window of time over which a malicious sender is forced to reveal its identity. For the pure receiver-pull model, this window is forever before the content is retrieved by the receiver. For the SIRP model, this window is from the moment sender expresses its intent to send till the time receiver retrieves the content. Thus, unlike the sender push model, there is a large window of time in which the receiver is free to verify a sender's identity. Moreover, the receiver-pull model also helps reduce the total amount of unwanted traffic *traveling throughout the Internet* [20]. For example, the total amount of spam traveling throughout the Internet in the receiver-pull model is determined by the number of spam messages retrieved by receivers, instead of the number of spam messages pushed out by spammers as in the sender-push model. For the sender-intent-based receiver-pull model, the total amount of spam-related traffic traveling throughout the Internet is the sum of the (short) intent messages pushed out by spammers, and the spam messages retrieved by receivers.

One obvious disadvantage of receiver-pull model is that the sender is burdened with greater content management complexity. The sender needs to store outgoing messages and keep them available at least till the intended receivers are willing to retrieve them, and needs to have a deletion policy if a message is never retrieved by the receiver. Another issue that the sender needs to grapple with is to ensure that the party retrieving a message is indeed the originally intended receiver. However, another angle to look at these disadvantages is that, in the sender-push model, it is the receiver who needs to deal with the very same issues.

2.3. Implications on unwanted traffic

Given that the receiver-pull model grants more control to receivers in terms of traffic delivery, and only receivers know what they want to receive, the receiver-pull model has clear advantages in restraining unwanted traffic compared to the sender-push model. Moreover, the above discussion also makes it clear that the sender is accountable to a greater degree in the receiver-pull model than in the sender-push model. This brings us to the following key idea which underlies the theme of this paper: *When designing any application-level communication protocol, it is advantageous to first consider using a receiver-pull model, which inherently provides greater protection against unwanted traffic.*

The receiver-pull based model is a relatively low-cost design choice that can be considered early during any communication system design. Even if the receiver-pull model results in slightly greater protocol complexity, it can greatly help to simplify accountability and authentication issues by placing the overheads where they truly belong – at the sender of the unwanted traffic.

A legitimate concern with a receiver-pull model is that it may end up increasing the cost of sending messages for malicious as well as honest senders. We will show in the next section through an example of a receiver-pull based email architecture that, using simple design optimizations, one can easily lower the sending cost for honest senders while still holding senders of unwanted content accountable.

We do not claim that a receiver-pull based model may be universally suitable for all forms of communications. For example, soldiers in the middle of a desert war may not want to rely on remote senders being reachable when trying to retrieve their messages. However, in many important applications, such as civilian use of email, mobile text messages, and asynchronous voice messages, the receiver-pull architecture appears to offer strong advantages in fight against unwanted traffic.

3. DMTP: a differentiated mail transfer protocol

DMTP is designed based on the sender-intent-based receiver-pull (SIRP) model, where senders are allowed to first express an intent to send message to a receiver via a small intention message. If the receiver happens to be interested, he contacts the sender and retrieves the content message. Fig. 2 illustrates the basic architecture of the new

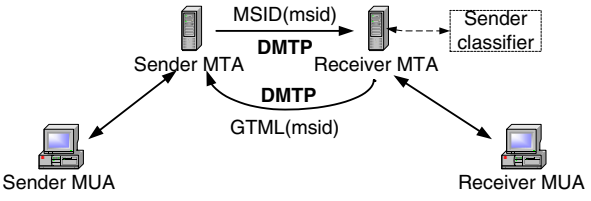

Fig. 2. Illustration of DTMP-based email system.

Table 1
News commands/reply code defined in DMTP

Commands/Replies	Explanation
MSID	For SMTA to inform RMTA the *msid* of a message
GTML	For RMTA to retrieve a message from SMTA
253	For RMTA to inform SMTA to send *msid* (MSID) instead of message body (DATA)

email delivery system. Before we delve into the details of DMTP, it is worth noting that the new system extends the current SMTP protocol [23] by adding two new commands – MSID and GTML, and one new reply code – 253 (see Table 1). All the commands and reply codes in SMTP are also supported in the new system. We explain the new commands and reply code when we use them. In Table 1, SMTA denotes a sender Mail Transfer Agent (MTA), and RMTA refers to a receiver MTA.

3.1. Differentiating message deliveries

As discussed in the last section the receiver-pull model increases the cost of sending messages for both malicious and legitimate senders. To address this issue DMTP is designed to support a hybrid email delivery system where both the sender-push and receiver-pull models can be employed. Specifically, each receiver can classify email senders into three disjoint classes and treat the delivery of messages from each of them differently: (1) *well-known spammers*, whose messages will be directly rejected; (2) *regular contacts*, whose messages can be directly pushed from the senders to the receiver using the current SMTP protocol; and (3) *unclassified senders* – senders that are neither well-known spammers nor regular contacts. Unlike regular contacts, unclassified senders cannot directly push a message in its entirety to the receiver. Such messages need to be stored and managed by the *senders' mail servers*, and only the envelope of the messages can be directly delivered to the receiver to notify the pending messages.

Senders can be defined at the granularity of email addresses as well as IP addresses (and domain names) of sender mail servers. Given that it is easy to fake email addresses in the current Internet, we envision that sender classification will be performed at the granularity of IP addresses when DMTP is first deployed. Note that, the IP address of a sender mail server cannot be easily forged, given that multiple command/reply transactions are required to deliver a message from the sender mail server to the receiver mail server [23]. We discuss the implications of misbehaving email users, who send spam through well-known mail servers, in Section 5. We also note that the proposed DMTP-based email delivery system has important implications for spammers who send spam through hacked zombie machines. Although spammers can easily turn an infected machine into a spam mail server, it is much harder for them to fake it as a *legitimate* mail server. Therefore, they cannot directly push spam messages in their entirety to receivers through zombie machines in the new system.

Fig. 3 summarizes the algorithm of handling message delivery requests at a DMTP receiver. In the figure we have assumed that the sender classification is only supported at the IP addresses (and domain names) level. Sender classification defined at email address level can be easily incorporated into the algorithm. In the rest of this section we

```
Require: denied: well-known spammer class;
Require: allowed: regular contact class;
Receiving TCP session open request on port 25;
ip = Get IP address of sender mail server;
if (ip ∈ denied) then
    /* well-known spammers */
    reply with 554 (to decline TCP session opening request);
else if (ip ∈ allowed) then
    /* regular contacts */
    reply with 220 (to accept TCP session opening request);
    proceed as if SMTP used;
else
    /* unclassified senders */
    reply with 253 (see Table 1);
    accept MSID command;
    reject DATA command;
end if
```

Fig. 3. Algorithm for receivers to handle message delivery requests in DMTP.

focus on the handling of messages from unclassified senders. The handling of messages from well-know spammers and regular contacts is the same as in the current practice [23,29], and we omit the description.

3.2. Unclassified sender: message composition and receiver notification

Like in the current email system, an (unclassified) sender uses a Mail User Agent (MUA) to compose outgoing messages [23]. After a message is composed by the sender, the sender delivers the message to the sender Mail Transfer Agent (SMTA).

All the outgoing messages of unclassified senders are stored at the SMTAs. For this purpose, an SMTA maintains an outgoing message folder for each *sender*. Instead of a complete message being directly pushed from the SMTA to the RMTA, only the envelope of the message is delivered. In particular, the SMTA notifies the RMTA about the pending message via the new *message identifier* command `MSID` (see Table 1), which contains the unique identifier *msid* of the message. The *msid* is used by the receiver to retrieve the corresponding message.[1] The identifier of a message is generated based on the sender, the receiver, and the message.

3.3. Receiver: pulling messages from unclassified senders

The new email delivery system grants greater control to receivers regarding if and when receivers want to read a message; senders cannot arbitrarily push a message to them. Receivers can be discriminate about which messages need to be retrieved, and which ones need not. If a receiver indeed wants to read a message, he will inform his own RMTA, and the RMTA will retrieve the message from the SMTA on behalf of the receiver. An RMTA retrieves an email message using the new *get mail* command `GTML` (see Table 1), which includes the identifier *msid* of the message to be retrieved. After the message has been pulled to the RMTA, conventional virus/worm scanning tools and content-based spam filters can be applied to further alert the receiver about potential virus or spam. Therefore, the new email system does not exclude the use of existing email protection schemes. For security reasons, when an SMTA receives the `GTML` command, it needs to verify that the corresponding message is for the corresponding email receiver, and the requesting MTA is the mail server responsible for the receiver.

[1] Note the fundamental difference between message pull in the new email system and URL embedded in many current spam messages. The address in the URL is normally not related to the sending machine of the message. In contrast, outgoing messages in the new email system have to be stored on the sender mail servers.

3.4. Minimizing the impact of intent messages

It is conceivable that before the majority of spammers are squeezed out of business, a large number of small intent messages may be delivered to Internet email users when DMTP is first deployed on the Internet. A legitimate concern is that email users may be overwhelmed by such small intent messages. This problem can be alleviated by, e.g., quarantining intent messages: RMTA will only deliver messages from regular contacts to receivers immediately; all the intent messages from unclassified senders will be first quarantined at the RMTA and only delivered to the receivers periodically in a single digest message. The interval over which the RMTA delivers the digest email of intent messages to a receiver can be configured by the receiver. A similar idea has been supported in commercial products and employed in real-world systems to handle spam messages [31,21]. Although spammers can push out intent messages, they are unlikely to get paid if they cannot deliver spam messages to receivers. We will develop a simple mathematical model to study the revenue of spammers in the DMTP-based system (and that in the SMTP-based system) in the next Section. As more spammers run out of business because of the increased adoption of DMTP, intent messages related to spamming will decrease and be less of a concern. (Rather, they are used for legitimate reasons for first-time correspondents to communicate.)

4. Performance evaluation

In this section we first develop a simple mathematical model to investigate the revenue that a spammer can gather by spamming a message to a set of Internet users. Based on this model, we then perform numerical experiments to study the effectiveness of DMTP in controlling spam, and how the behaviors of both spammers and receivers affect the spammers' revenue. At the end of this section, we illustrate the advantage of the receiver-pull

model in reducing the total amount of spam traffic *traveling throughout the Internet* using real-world spam traces.

4.1. A simple model of spammer revenue

Table 2 summarizes the notations used in this section. Consider a spammer s. We assume that s maintains a set of N email addresses to which he can send spam emails. In this model we establish the expected revenue the spammer can gather by sending a single message to the N email addresses. We assume the spammer owns or rents x machines to send spam (each with a unique IP address). On average, each machine is capable of sending k messages per unit time (which is only constrained by the processing power and Internet access speed of the machines). The spamming task is equally partitioned over the x machines, that is, each machine needs to send the message to N/x receivers. For each machine, the spammer needs to pay y units of cost for each unit of time (e.g., for Internet access or renting machines from hackers or time spent in recruiting zombies). In return, the spammer obtains g units of gain for each message delivered.

For simplicity, we assume there is a central real-time blacklist of well-known spammers, which is used by all receivers. Before the spammer starts spamming, we assume that none of the x machines managed by the spammer is listed by the central RBL. Instead, they are in the unclassified-sender class of all N receivers. (Sender classification is defined at the granularity of IP addresses.) When a receiver retrieves a message from the spammer, it will report the IP address of the corresponding SMTA to the central RBL with a probability of p. (We assume that intent messages are directly delivered to end users instead of first being quarantined at the RMTAs.) Furthermore, the central RBL requires at least q reports of a spamming machine before adding the corresponding IP address into its blacklist. After an IP address is added to the blacklist, the spammer can no longer send messages from the corresponding SMTA. To simplify, we assume that the spammer has the precise knowledge of the time when an SMTA is blacklisted and will disconnect the machine to minimize its own cost.

We assume the arrivals of spam retrievals from receivers follow a Poisson distribution, with a mean arrival (i.e., retrieval) rate r (retrievals per unit time). Given that the list of email addresses maintained by a spammer is in general large, we assume the spam retrieval rate r is a constant over time. Below we derive the expected revenue $U(t)$ of the spammer at time t, assuming the time for the spammer to start spamming the message to the N receivers is zero.

Let $R(t)$ denote the expected number of receivers who have retrieved the message at time t. It is not too hard to see that $R(t) = \min\{rt, xq/p\}$. Let $f(t)$ denote the expected number of messages delivered by the spammer at time t (across all x machines), we have $f(t) = \min\{N, xkt, R(t)\}$. Consequently, the expected income of the spammer at time t is $gf(t)$. On average, it takes N/r units of time for the spammers to deliver the message to all receivers, and it takes $(q/p)/(r/x)$ units of time for the central RBL to blacklist an SMTA (assuming $r \ll k$ and all x machines are accessed with the same probability). Therefore, the total expected cost $c(t)$ paid by the spammer at time t is $c(t) = xy\min\{t, N/r, (q/p)/(r/x)\}$. Hence, in the DMTP-based email system the total expected revenue of the spammer at time t is

$$U_{\mathrm{DMTP}}(t) \approx gf(t) - c(t) = g\min\{N, xkt, R(t)\} - xy\min\{t, N/r, (q/p)/(r/x)\}. \quad (1)$$

Table 2
Notations used in the spammer revenue model

Notation	Explanation	Setting
N	Number of email addresses maintained by spammer	10 M
x	Number of machines used by spammer	62
k	Sending speed of a machine (messages/unit time)	100 K
y	Cost paid by spammer per machine per unit time	0.1
g	Gains of spammer for each message delivered	0.005
p	Probability that a receiver reports a spamming machine	0.001
q	Number of reports required for RBL to blacklist a machine	50
r	Mean spam retrieval rate of receivers (retrievals/unit time)	2500

We can similarly derive the total expected revenue of the spammer at time t in the current SMTP-based email system, which is given below:

$$U_{\mathrm{SMTP}}(t) \approx g \min\{N, xkt\} - xy \min\{t, (N/x)/k\}. \quad (2)$$

In the above equation, we have assumed that k is large enough that the spammer can finish sending the message to all receivers before the SMTAs are blacklisted.

Comparing Eqs. (1) and (2), we see that while the revenue of the spammer is largely determined by the *sending* speed of its SMTAs in the current SMTP-based email system, in the DMTP-based email system its ability to spam is greatly constrained by the message retrieval behavior of the receivers. The slower the receivers are in retrieving the message, the longer the spammer needs to stay online; the higher the probability is for receivers to report spamming SMTAs to the central RBL, the earlier the spamming SMTAs are blacklisted.

4.2. Numerical studies

In this section we perform numerical experiments to study the effectiveness of the proposed DMTP protocol in controlling spam using the model developed in the last subsection. We also investigate how the behaviors of both spammers and receivers affect the spammers' revenue. Table 2 (third column) presents the parameter values we used in the numerical studies, unless otherwise stated.

First, we study how the proposed DMTP protocol helps to reduce the maximum revenue of a spammer (by spamming a message to N receivers) and forces the spammer to stay online (to improve the performance of RBLs). Fig. 4 shows the revenues of the spammer as time evolves in both the current SMTP-based email system (curved marked as *Without DMTP*) and the proposed DMTP-based email system. From the figure we see that, in the current email system, the spammer can gather the maximum revenue (49,990) within 2 units of time. This means that the spammer can quickly push out the message to all the receivers and then vanish, long before any RBLs can identify it. In contrast, in the DMTP-based email system, the maximum revenue is 7812 units, only about 16% of the spammer maximum revenue in the current email system. Moreover, in order for the spammer to gather the maximum revenue, the spammer has to stay online

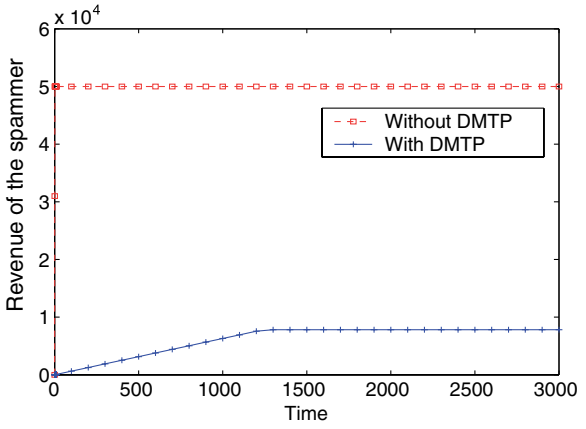

Fig. 4. Expected spammer revenue.

for a much longer time window (1240 units of time). This can significantly improve the performance of RBLs. Note also that the revenues will not decrease once they reach the maximum values. This is because a spammer disconnects an SMTA to minimize the cost once the SMTA has finished sending the message to all receivers (in SMTP) or it is blacklisted (in DMTP).

Next, we investigate the impact of the number of SMTAs employed by a spammer on the maximum spammer revenue in the DMTP-based email system. Fig. 5 shows the *maximum* spammer revenue as a function of the number of SMTAs employed by the spammer for $k = 50$ K and 100 K, respectively. Note first that increasing the sending speed of spam from $k = 50$ K to 100 K will not result in a higher maximum spammer revenue. Indeed, after the spam sending speed exceeds the spam retrieval rate of receivers, it will not affect the maximum spammer

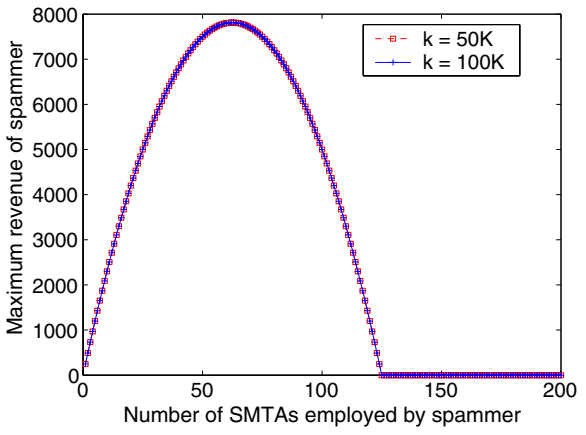

Fig. 5. Impact of number of SMTAs.

revenue. Now let us examine how the number of SMTAs employed by a spammer will affect the maximum spammer revenue. As we can see that the spammer has some initial gains by increasing the number of SMTAs (when the number is less than 62). This is because as the number of SMTAs increases, it takes a longer time for all the SMTAs to be blacklisted by the central RBL, and the message can be retrieved by more receivers. Fortunately, the spammer cannot indefinitely increase the number of SMTAs to evade RBLs. When the spammer employs more than 62 SMTAs, his maximum revenue actually starts to drop, as the income of delivering the message to new receivers can no longer recompense the cost to deploy the new SMTAs.

In the last set of numerical experiments, we study the effects of the spam retrieval rate of receivers on the maximum spammer revenue. Fig. 6 depicts the *maximum* spammer revenue as a function of the spam retrieval rate of receivers for number of SMTAs $x = 100$, 200, 400, respectively. As we can see from the figure, the maximum spammer revenue decreases as the receivers reduce their retrieval rate of messages from the unclassified SMTAs for all three cases. Moreover, when the retrieval rate is sufficiently low (for example, less than 2000 retrievals per unit time when $x = 100$), the spammer cannot gather any revenue from spamming. More importantly, when a spammer recruits more SMTAs to send spam, it requires a larger threshold of spam retrieval rates for the spammer to gather any revenue (for example, 4000 when $x = 200$, compared to 2000 when $x = 100$). This again demonstrates that spammers cannot gather more revenue by indefinitely recruiting more SMTAs. As more spammers run out of business because of the increased adoption of DMTP, the email spam problem will be effectively controlled on the Internet.

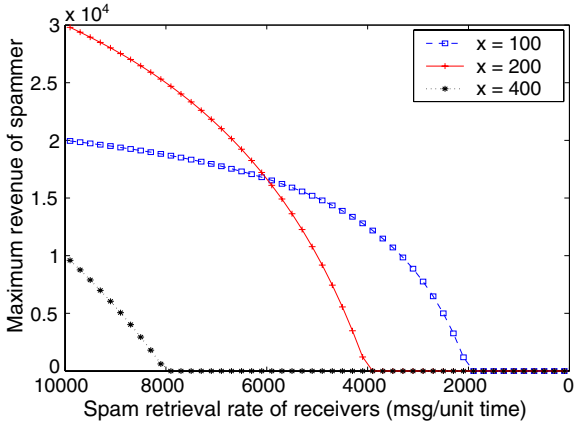

Fig. 6. Impact of spam retrieval rate.

4.3. Effect on spam traffic traveling on the internet

In the DMTP-based email system, receivers have greater control over if and when a message from a sender can be delivered. Senders cannot push spam to receivers at will; instead, they can only send short intent messages (i.e., the envelope) to receivers to express the intention to send. By only delivering the envelope of a message from sender to receiver, less bandwidth, storage, and time will be occupied at the receiver side. On the other hand, if a receiver indeed wants to read the message from an unclassified sender, extra bandwidth and time will be used. However, receivers will most unlikely be interested in messages from unclassified sender; therefore, the majority of such messages will not be retrieved. More importantly, if the majority of messages from unclassified senders are not delivered, much less bandwidth on the Internet as a whole will be consumed by spam. To have a better understanding on this, we conduct a simple empirical study on the lengths of spam messages. We use the data from the Spam Archive site [1]. This site maintains archives of email spam contributed by Internet email users. We (randomly) select spam archived by the site on 5/15/2004, 6/15/2004, 7/15/2004, 8/15/2004, 9/13/2004 (no archives on 15th and 14th of the month), and 10/15/2004. Due to email forwarding problems (from spam receivers to the Spam Archive site), some messages in the archives are damaged or incomplete, we remove such messages from the data sets before we analyze the data. Table 3 shows the total number of messages and the number of complete messages. We refer to the data set *after excluding the damaged and incomplete messages* as a "Spam Archive."

Fig. 7 depicts the fraction of spam message body lengths. We treat message headers as the envelopes of the messages and therefore exclude message

Table 3
Number of messages in each Spam Archives

Messages	5/15	6/15	7/15	8/15	9/13	10/15
Complete	478	2742	415	346	462	393
Damaged	4	2	0	2	1	5
Total	482	2744	415	348	463	398

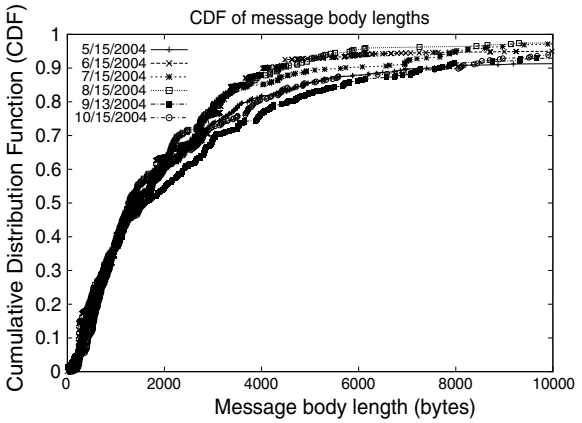

Fig. 7. Fraction of spam message body lengths.

headers when we compute this length. Note that, in general, the envelope of a message is much shorter than the message header. We can see from the figure that more than half of spam messages have a body longer than 1.5 kB, and 60% of spam longer than 2 kB across all the spam archives we examine. Although the bandwidth saving from not retrieving a single message may not be significant for individual receivers, the overall potential saving on the Internet backbone network bandwidth can be promising if such messages are not delivered over the Internet, given the massive volume of spam on the Internet. These observations may be specific to the spam messages archived at [1], it is worthwhile to independently investigate the properties of spam messages from other sources to have a more comprehensive understanding of the impacts of DMTP on minimizing the Internet backbone bandwidth consumed by email spam traffic.

5. Incremental deployment and discussions

5.1. Incremental deployment

DMTP can be easily deployed on the Internet incrementally. The basic idea is to combine DMTP with a sender-discouragement scheme (such as asking senders to solve a puzzle [22] or Greylisting [19]). However, unlike existing sender-discouragement schemes, we only require senders in the unclassified-sender class to make the extra effort in sending a message. In this section we outline one such approach, where unclassified senders need to solve a puzzle before their messages can be delivered to end users. It is worth noting, however, that DMTP can be incrementally deployed on the Internet in other fashions (see [12]). In the following, we assume that the RMTA in consideration supports the DMTP protocol, and show how it interacts with the rest of the world. For simplicity we assume that the sender classification is performed at the granularity of IP addresses (or domain names) of SMTAs.

In order to support incremental deployment, RMTA supporting DMTP needs to know if the SMTA also supports DMTP. For this purpose, an SMTA supporting DMTP will inform the RMTA this fact by including keyword "DMTP" in the `MAIL` command. Fig. 8 presents the algorithm used by receivers to handle message delivery requests in supporting incremental deployment of DMTP.

5.2. Discussions

5.2.1. Security of message retrieval

A potential concern with the receiver-pull model is security. However, as we discuss below, the potential security issue arising from this model is no worse than the current SMTP model. First, important messages are normally communicated amongst regular contacts, which are handled in DMTP in the same way as in the current email system. Secondly, individual users cannot retrieve messages from a remote SMTA directly, they rely on their corresponding RMTAs to retrieve messages (from unclassified senders). Lastly, *msid*s are generated randomly based on the messages (and senders and receivers); they cannot be easily guessed.

5.2.2. Mail forwarding and user-perceived system performance

DMTP does not support open relay mail servers, most of which are blacklisted even today [29]. An organization may deploy multiple mail servers for relaying inbound and outbound mails. Such mail relay servers can be adequately supported by DMTP. We refer the interested readers to [12]. In principle, a message from an unclassified sender is fetched directly from the sender's mail server by the receiver's mail server (or the border mail relay server) in DMTP. Given the ever-increasing network speeds, we do not expect any degradation of user-perceived email reading experience, although some messages – the ones from unclassified senders – need to be retrieved from a remote mail server. We plan to formally study this issue in our future work (but note the largely satisfactory web-surfing

```
Require: denied: well-known spammer class;
Require: allowed: regular contact class;
Receiving TCP session open request on port 25;
ip = Get IP address of sender mail server;
if (ip ∈ denied) then
    /* well-known spammers */
    reply with 554 (to decline TCP session opening request);
else if (ip ∈ allowed) then
    /* regular contacts */
    reply with 220 (to accept TCP session opening request);
    proceed as if SMTP used;
else
    /* unclassified senders */
    reply with 220 (to accept TCP session opening request);
    proceed to the MAIL command;
    if (found keyword "DMTP" in the MAIL command) then
        /* sender supports DMTP */
        proceed according to DMTP;
    else
        /* sender does not support DMTP */
        respond to DATA command with 354;
        receive message;
        respond with 550 (permanent error);
        store message, send puzzle;
        message invisible to user;
        /* message becomes visible to user only after puzzle solved */
    end if
end if
```

Fig. 8. Handling message delivery requests at RMTAs for incremental deployment of DMTP.

experience, where, in a similar manner, a web page needs to be remotely fetched).

5.2.3. Impact of misbehaved senders on sender mail servers

When sender classification is only supported at the granularity of IP addresses, a single misbehaved sender may destroy the reputation of the sender's corresponding mail server by sending out a large number of spam messages. Fortunately, public mail servers normally establish certain mechanisms to prevent their users from spamming, for example, by imposing a quota on the number of messages a user can send everyday. In general, it is hard for spammers to send spam messages through public mail servers. We will further investigate this problem in our future work.

5.2.4. Mailing list

We believe that in the future all mailing lists will be mediated and content-based spam filters will be universally deployed by all mailing lists. In DMTP, we suggest all users to add their mailing lists into their *regular contacts*. In this way, the RMTA of a

user can directly accept the message from the mailing list, without putting any extra burden on the MTA of a mailing list and mediator. Similarly, the MTA of a mailing list should also add all members into its *regular contacts*, such that it can directly receive messages from its members.

5.2.5. Electronic greeting card delivery services

This type of services puts great challenges on sender authentication. Sender Rewriting Scheme (SRS) [35] was proposed to mitigate this issue. A main challenge for DMTP is how to handle the delivery of messages whose sender addresses have been rewritten by SRS. One possible approach is to let MTAs maintain the reputations of the E-Card sites, and only allow sites with good reputation to directly deliver a message to the RMTA. For other sites, only the headers are delivered. End users need to contact the original senders before the complete message is retrieved from the E-Card sites.

5.2.6. Exporting user regular contacts to service providers

Users may not be willing to export their own regular contact lists (especially the ones at the email address level) to the service providers. Some secure mechanisms to conceal the exact identifications of users' regular contacts can be used, such as Bloom filters [4]. Users hash their regular contacts to a bloom filter and export the bloom filter to the corresponding RMTA instead of the exact regular contacts. The RMTA relies on the bloom filter to detect if a sender is in the user's regular contact list (note that bloom filters may incur some false positives).

6. Related work

The most widely deployed anti-spam solutions today are reactive content filters that scan the contents of the message at the receiver's MTA after the message has been delivered. However, none of them can achieve 100% accuracy, and spammers quickly adapt to counter the strategies used by these filters. In addition, content filtering will no longer serve as long-term viable solution once email messages begin to be encrypted using receivers' public keys [27]. Instead, we have advocated fundamental changes in protocol-level design to a pull-based model.

Like DMTP, FairUCE [8] also advocates the usage of sender classifiers. However, it is still a push-based model in which network reputation, along with receiver defined whitelist and blacklist, is used to determine whether to accept a message. IM2000 [2] also advocates a pull-based model like DMTP. However, unlike DMTP, all outgoing messages need to be stored at sender MTAs and receivers need to retrieve all the messages remotely, regardless of where the messages come from. In addition, IM2000 is not incrementally deployable and requires massive infrastructure changes. Li et al. proposed a method to slow down spam delivery by damping the corresponding TCP sessions [25]. However, the long-term impact of modifying the behavior of TCP for a specific application is not clear, and spammers may respond by changing sender MTA's TCP behavior. In the Greylisting [19] approach, a message from a new sender is temporarily rejected upon the first delivery attempt, the underlying assumption being that spammers will not re-send a message whereas regular MTAs will. However, it is only a matter of time before spammers adapt to this technique by re-sending their message. Sender authentication schemes such as [11,24] can help improve the accountability of email senders. However, they cannot control the delivery of spam by themselves.

The Internet Message Access Protocol (IMAP) allows a user to retrieve part of a message, such as the message header without fetching the complete message, from *his mail server* [10]. However, it works only between the user's MUA and his local mail server. The complete message is first delivered from the sender MTA to the receiver MTA. Email Prioritization was proposed in [34] as a way to control the impact of spam on legitimate messages. However, the performance of the system depends on how well it can predict that an incoming message is spam. Moreover, spammers still have the incentive to send a large number of messages given that the entire messages including both headers and bodies are still delivered from the sender to the receiver (even though they may do so at the cost of purchasing more machines). Gburzynski and Maitan proposed to use email aliases to fight email spam [15], where different email aliases can be created for different purposes and used over a specific duration. However, its effectiveness relies on hiding email addresses and their aliases. Moreover, users have more burdens to manage their accounts. For example, they need to create email aliases and disseminate them to intended correspondents.

7. Conclusion and ongoing work

In this paper we examined the architectural aspects of the current email system that are responsible for the proliferation of spam, and proposed a Differentiated Mail Transfer Protocol to control spam. In addition, we also developed a formal model to study the performance of DMTP. Through numerical experiments we demonstrated that DMTP can significantly reduce the maximum spammer revenue. Moreover, it also forces spammers to stay online for longer periods of time, which helps improve the performance of real-time blacklists of spammers. Currently, we are developing a prototype of DMTP by extending Sendmail [7]. We plan to further investigate the performance of DMTP based on the prototype and simulations. We will also study in detail the issues briefly discussed in Section 5.

Acknowledgement

We thank the anonymous reviewers of the IFIP Networking 2006 conference. Their insightful and constructive comments helped improve both the technical details and presentation of the paper.

References

[1] Spam Archive. Donate your spam to science. Available from: http://www.spamarchive.org/.
[2] D. Bernstein. Internet Mail 2000 (IM2000). Available from: http://cr.yp.to/im2000.html.
[3] M. Blumenthal, D. Clark, Rethinking the design of the internet: the end-to-end arguments vs. the brave new world, ACM Transactions on Internet Technology 1 (1) (2001).
[4] A. Broder, M. Mitzenmacher, Network applications of bloom filters: a survey, in: Proceedings of the 40th Annual Allerton Conference on Communication, Control, and Computing, 2002.
[5] X. Carreras, L. Márquez, Boosting trees for anti-spam email filtering, in: Proceedings of RANLP-01, 4th International Conference on Recent Advances in Natural Language Processing, Tzigov Chark, BG, 2001.
[6] T. Claburn, Big guns aim at spam, Information Week (2004).
[7] Sendmail Consortium. Welcome to sendmail.org. Available from: http://www.sendmail.org/.
[8] IBM Corporation. Fair use of unsolicited commercial email FairUCE, 2004. Available from: http://www.alphaworks.ibm.com/tech/fairuce.
[9] L. Cranor, B. Lamacchia, Spam! Communications of the ACM 41 (1998) 74–83.
[10] M. Crispin, Internet message access protocol – version 4rev1, IETF RFC3501 (2003).
[11] M. Delany, Domain-based email authentication using public-keys avertised in the DNS (domainkeys). Internet Draft, August 2004. Work in Progress.
[12] Z. Duan, K. Gopalan, Y. Dong, Receiver-driven extensions to SMTP. Internet Draft, July 2006. Work in Progress.
[13] Z. Duan, K. Gopalan, X. Yuan, Behavioral characteristics of spammers and their network reachability properties. Technical Report TR-060602, Department of Computer Science, Florida State University, June 2006.
[14] Ferris Research. Spam control research reports. Available from: http://www.ferris.com/.
[15] P. Gburzynski, J. Maitan, Fighting the spam wars: a remailer approach with restrictive aliasing, ACM Transactions on Internet Technology 4 (1) (2004) 1–30.
[16] J. Goodman, R. Rounthwaite, Stopping outgoing spam, in: Proceedings of EC'04, 2004.
[17] P. Graham, A plan for spam, January 2003. Available from: http://www.paulgraham.com/spam.html.
[18] P. Graham, Better bayesian filtering. Available from: http://www.paulgraham.com/better.html, January 2003.
[19] E. Harris, The next step in the spam control war: Greylisting, White Paper, 2003.
[20] C. Heun, The state of spam, Information Week (2006).
[21] ITS. Spam at the university of hawaii. Available from: http://www.hawaii.edu/infotech/spam/spam.html. Last checked: 11/19/2005.
[22] A. Juels, J. Brainard, Client puzzles: A cryptographic defense against connection depletion attacks, in: Proceedings of NDSS-1999, February 1999.
[23] J. Klensin, Simple mail transfer protocol, RFC 2821 (2001).
[24] M. Lentczner, M.W. Wong, Sender Policy Framework (spf): Authorizing Use of Domains in MAIL FROM. Internet Draft, October 2004. Work in Progress.
[25] K. Li, C. Pu, M. Ahamad, Resisting spam delivery by TCP damping, in: Proceedings of First Conference on Email and Anti-Spam (CEAS), July 2004.
[26] J. Lyon, M. Wong, Sender ID: Authenticating e-mail. Internet Draft, August 2004. Work in Progress.
[27] P. Mannion, Interview: Ethernet's inventor sounds off, Information Week (2005).
[28] A. Ramachandran, N. Feamster, Understanding the network-level behavior of spammers, in: Proceedings of ACM SIGCOMM, September 2006.
[29] RBL. Real-time spam black lists (RBL). Available from: http://www.email-policy.com/Spam-black-lists.htm.
[30] M. Sahami, S. Dumais, D. Heckerman, E. Horvitz, A bayesian approach to filtering junk E-mail. In Learning for Text Categorization: Papers from the 1998 Workshop, Madison, Wisconsin, 1998. AAAI Technical Report WS-98-05.
[31] Sophos Plc. Sophos plc. Available from: http://www.sophos.com/.
[32] SpamAssassin. The apache spamassassin project. Available from: http://spamassassin.apache.org/.
[33] The Editors. Product of the year: Spam? Information Week, 2004.
[34] R. Twining, M. Williamson, M. Mowbray, M. Rahmouni. Email prioritization: reducing delays on legitimate mail caused junk mail, in: USENIX Conference, June 27 to July 2 2004.
[35] M. Wong, What email forwarding services need to know about SPF. Available from: http://spf.pobox.com/emailforwarders.pdf.

Zhenhai Duan (S '97–M '03/ACM M '03) received the B.S. degree from Shandong university, China, in 1994, the M.S. degree from Beijing University, China, in 1997, and the Ph.D. degree from the University of Minnesota, in 2003, all in Computer Science. He is currently an Assistant Professor in the Computer Science Department at the Florida State University. His research interests include computer networks and multimedia communications, especially Internet routing protocols and service architectures, scalable network resource control and management, and network security. He is a co-recipient of the 2002 IEEE International Conference on Network Protocols (ICNP) Best Paper Award, and the 2006 IEEE International Conference on Computer Communications and Networks (ICCCN) Best Paper Award. He is a member of IEEE and ACM.

Yingfei Dong joined the faculty of University of Hawaii after received his Ph.D. degree in Computer and Information Science from the University of Minnesota in 2003. His main research interests are in the areas of computer networks, security, multimedia content delivery, Internet services, distributed systems, and advanced computer architecture. Dong holds two US patents and has published over 40 papers in journals and refereed conferences.

Kartik Gopalan is an Assistant Professor in Computer Science at the State University of New York at Binghamton. He received his Ph.D. in Computer Science in 2003 from Stony Brook University, M.S. in Computer Science in 1996 from Indian Institute of Technology, Chennai, and B.E. in Computer Engineering in 1994 from Delhi Institute of Technology. His research interests lie in operating systems, distributed systems, and wide-area/wireless networks with focus on performance guarantees and resource virtualization.