Getmail: A New Email Delivery Architecture to Control Spam

Zhenhai Duan, Kartik Gopalan
Computer Science Department
Florida State University
Tallahassee, FL 32306
Email: {duan,kartik}@cs.fsu.edu

Electronic mail (Email) is one of the most popular applications on the Internet. As the ability to send and receive Emails is being integrated into hand-held devices such as cell phones, we have increasingly come to rely on Email as an indispensable communication tool. However, the current Email delivery infrastructure also provides an essentially effort-free platform for spammers to send a deluge of unsolicited commercial messages. It is estimated that nowadays spam messages constitute 79% of all business Emails, up from 68% since the US federal Can-Spam Act of 2003 took effect in January 2004 [1]. In monetary terms, dealing with spam costed businesses more than $20 billion in 2003 [3].

Recent research and development efforts to control spam can be classified into three categories: 1) spam filters, which try to identify spam so that users do not need to spend time processing them; 2) sender verification tools such as caller-ID or SPF (Sender Policy Framework) [4], which prevent spammers from forging sender addresses; and 3) mechanisms to discourage spammers from sending a large number of messages, such as paid Emails or challenge-response based mechanisms [5]). It is our belief that, to completely control Email spam, we need an integrated system that incorporates the mechanisms in all the three functionalities. In this paper, we focus on the third category, namely the discouragement mechanisms.

The fundamental reason why spam is so attractive as an advertisement medium is that it costs virtually nothing more than a single Internet connection for the spammer to send a deluge of Emails. In other words, the spammer does not does not nned to pay any price for the resources consumed from the digital society in terms of time, bandwidth, computation, storage, management effort, or lost productivity. As mentioned above, two discouragement mechanisms have been floated around recently to act as disincentive for spammers - (1) paid Email and (2) challenge-response based systems. Although the basic idea behind paid Email is quite simple, many practical deployment issues remain to be addressed, such as how much how much postage a user needs to pay for each Email and how to collect and distribute postage on the global Internet. Additionally, spammers usually have deep pockets and monetary considerations might even encourage ISPs to attract large spammers as their customer-base. The second scheme aims to make it expensive for a spammer to to send bulk mail via a challenge-response interaction for each Email sent out. However such a scheme would also penalize well-behaved customers due to larger latency in sending each Email, and consequently turning Email into a less attractive communication tool in general.

In this paper, we propose a fundamentally different discouragement mechanism. We identify spammer's identity and Email storage space as two parameters that can be exploited to encourage responsible behaviour among e-marketers. We propose a new message delivery architecture, called *Getmail*. The principal idea behind Getmail is to force the spammer to dedicate storage for the Emails being sent out. Instead of the entire message being directly delivered to the receiver (or more precisely the receiver side Mail Transfer Agent, MTA) from the sender, only the message header (including the *Subject* line with a limited length) is sent to the receiver. The message header is augmented with a field informing where the receiver can retrieve the complete Email. In this way, Email senders are forced to store the messages on their own MTA servers, from where receivers can retrieve them, if they happen to be interested.

Before we describe the architecture in greater detail, we first outline some of the prime advantages of Getmail. First, Getmail forces the senders to maintain their messages and keep their MTA servers up, before receivers retrieve the messages. Thus Getmail makes it relatively easy to identify spammers and facilitates the design of schemes

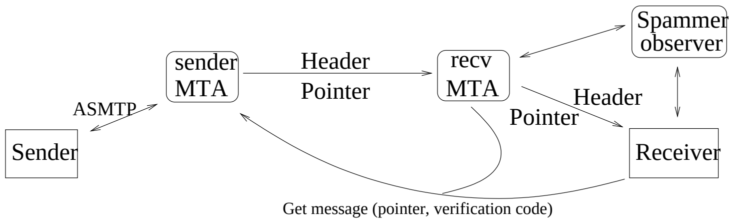

Fig. 1. Illustration of Getmail architecture.

that rely on IP addresses to block spam messages. Of course, spammers can still crack into other users' machines to send spam, but clearly it is orthogonal to the problem we are dealing with here and beyond the scope of this paper. Second, in the current system it costs money and time for receivers to receive and handle spams, this is especially true for users who dial up to their ISPs. By only delivering message headers from senders to receivers, less bandwidth, storage, and time will be occupied at the receiver side, and senders will have a greater share of responsibility in managing the messages they send. For example, senders hold their messages until they are retrieved by receivers and clean up the messages that are not read by receivers for a certain amount of time. Third, by forcing sender side MTA servers to hold messages, spammers will notice that a large portion of the spam messages are not read at all. As a consequence, the incentive to send spams is reduced. Moreover, in the long run, spammers will learn who are interested in a certain specific advertisement (the ones who read the spam message), and commercial advertisement can be targeted at a certain group of interested receivers, instead of being sent blindly to everyone. In this way, both advertisers and Internet users will benefit from this learning process in a long run.

Figure 1 illustrates the basic architecture of Getmail. After a message is composed by a sender, it is delivered to the local outgoing MTA through ASMTP, an augmented SMTP protocol. Besides the current functions of SMTP, ASMTP also allows the sender to maintain his messages on the outgoing MTA, for example cleaning up old messages that are not read by receivers after a certain amount of time. Messages are stored at the outgoing MTA, a message header is sent to the receiver side MTA, along with a *pointer* indicating where the message can be retrieved from. For security purposes, some verification code is also included in the header. Later this verification code is used by the sender MTA to verify if a message retriever is indeed authorized to retrieve the message. Before a user retrieves a message, he will contact the (local) *spammer observer* to check if the sender was listed as a spammer in the past. If so, the message header is deleted silently without any further action. Otherwise, the user may retrieve the message from the sender MTA (by following the pointer with the given verification code). A message is only removed after it has been retrieved by the receiver (the sender can also explicitly delete it or edit it before it is retrieved). A certain spam filter can be installed on the receiver's local machine, if a retrieved message is identified as a spam (by the spam filter before the receiver reads it or by the receiver himself), the receiver may report this to the (local) spammer observer. The corresponding IP addresses and Email accounts will be added into the spam observer. To simplify the architecture and also to improve its incremental deployment, the receiver side MTA can also be configured in such a way that, for MTAs with good reputation, it will accept the whole message instead of only the message header. For these messages, the receivers do not need to retrieve them from the sender MTAs.

Currently we are working on the details of the architecture and protocols. We will also develop a prototype of Getmail and study its performance and effectiveness on the Internet in the next stage.

References

[1] Information Week, "Big guns aim at spam," Mar. 2004.
[2] Information Week, "Product of the year: Spam?," Jan. 2004.
[3] M. Lentczner and M. W. Wong, "Sender policy framework (spf): A convention to describe hosts authorized to send SMTP traffic," Internet Draft, Feb. 2003, Work in Progress.
[4] The Economic Times, "Free lunch ends: e-mail to go paid," Feb. 2004.