Towards Self-Learning and Fully Transparent UCE Prevention

Wilfried N. Gansterer and Michael Ilger

wilfried.gansterer@univie.ac.at, michael.ilger@winf1.at

Abstract: A self-learning system for preventing unsolicited commercial or bulk e-mail (UCE or UBE) is presented. It acts at the source of each e-mail message and controls the traffic going *out* of a network, thus avoiding common drawbacks of standard spam filtering techniques.

The system is based on a token bucket mechanism proposed earlier. In this paper, it is shown how to develop this approach into a fully transparent and effective UCE prevention system by introducing adaptivity and learning capabilities. The first central component introduced in this paper is a framework for quantitatively analyzing the business model underlying UCE, allowing for insights into the effectiveness of filtering in the given situation. The second one is a strategy allowing the system to adaptively and intelligently (re-)configure itself in order to achieve almost arbitrarily high levels of *transparency*, i. e., to harm the business model underlying UCE *without* affecting regular e-mail users. Finally, a third component introduces adaptability for ensuring that only spam is blocked and that no regular mail traffic is affected.

1 Introduction

Numerous strategies for addressing the problem of unsolicited commercial or bulk e-mail (UCE or UBE, commonly denoted as "spam"), have been proposed. Many of them lack a lasting effect, because it is usually easy to circumvent them. Moreover, most of the approaches act *after* an e-mail is received, when most of the damage has happened.

The idea of *filtering* spam out of *incoming* e-mail traffic is relatively easy to implement and has also brought some success. It has a number of disadvantages, though: It generates relatively high costs and tends to achieve only limited improvements if it is based only on a certain number of static filtering rules.

Longer lasting, effective and comprehensive solutions for the spam problem need to focus on two aspects: (i) Detection and prevention of spam in an early stage, ideally already in *outgoing e-mail traffic* on the mail servers, and (ii) *intelligence* in the sense of being dynamic and capable of adapting to changes in the environment as well as to variations in the content and structure of UCE messages. Only if these two aspects are taken into account, the spam problem can be fought *at its source*. Many existing methods primarily "treat symptoms" *after* most of the damage caused by UCE has already been done (such as unexpected overload in bandwidth and storage capacity, network overhead as well as loss of end-user productivity).

1.1 Related Work

Methods for detecting and preventing spam can be categorized into three basic classes according to their point of action in the mail transfer process. *Pre-send methods* act *before* the message is transported over the network, whereas *post-send methods* act *after* the message has been transferred to the receiver. A third class comprises approaches based on modifying the transfer process itself.

Several suggestions for new protocols extending or replacing SMTP have been made in order to overcome its deficiencies in the context of the spam problem. The main problem with this approach is that a worldwide agreement on a drastic change in the protocol underlying e-mail transfer and a coordinated transition are very unrealistic in the foreseeable future.

Most of the research in the area of anti-spam methods has traditionally focused on post-send methods, more specifically, on filtering at the receiver side. Important examples are black- and whitelisting, Bayes filters [AKCS00], or rule sets (for example, SpamAssassin (`http://spamassassin.apache.org`). Some post-send methods are able to achieve reasonably satisfactory results *provided* they are properly trained, tuned, and maintained. This effort required for maintaining satisfactory performance of post-send approaches is one of their disadvantages. An even more substantial drawback is the fact that a big part of the damage caused by UCE is already done *before* post-send methods can become active. Thus, they are not able to reduce the overhead, the waste of bandwidth, processing power, memory and time caused by UCE. Moreover, privacy issues may arise with filtering methods since most of them have to access the message content. Other post-send approaches, such as greylisting [Har03], take effect a little earlier and currently yield very high true positive rates, but have the disadvantage of potentially introducing undesirable delays in mail delivery. Approaches for overcoming the drawbacks of conventional greylisting have been proposed in [GJL07, JGK08].

As mentioned above, it is essential to develop strategies which fight the spam problem at the *source* in order to avoid the overhead and waste of resources caused by spam. In this aspect, pre-send methods have an inherent advantage. Existing research in this area can be grouped into two categories: (i) Strategies for identifying sources of spam and, based on the results, for blocking those sources; and (ii) strategies for harming the business model of spammers by increasing the cost associated with sending out (spam) e-mail.

The first category comprises not only simple methods such as shutting down spam sources based on user complaints, but also more sophisticated ones. For example, one could try to detect abnormal sending behavior or spam sources by examining log data produced by outgoing and incoming mail transfer agents (MTAs). Heuristics for the latter have been described, for example, in [Cla01] and [Cla05]. Besides log data, network traffic on the TCP layer can be used to gather information about spamming machines. A link analysis technique described in [DS04] identifies nodes with anomalous behavior. The underlying assumption is that e-mail servers form a community with strong internal interaction and that nodes which are not part of the community (for example, spam sources) differ in their behavior. However, some of these approaches assume a lot of detailed information about

the network structure. Another major disadvantage is that they are mostly *reactive* (a host is identified as spam source only *after* spam was sent out).

It seems more promising to investigate approaches which prevent spam messages from being sent out in the first place. Important ideas in this direction are based on economic analyses how to compromise the business model underlying UCE, for example, in [DHD+06]. Conceptually, such techniques can be divided into two groups: (i) Approaches which allow unlimited sending of e-mail, but increase the costs for the sender; and (ii) methods which (in some way) limit the number of messages a user is allowed to send out. A cost increase can, for example, be achieved by micro-payment models [Tur03] or by imposing computational costs for sending out e-mail messages. Representatives of the latter approach are CPU bound [DHD+06, Bac02] and memory bound [ABMW03, DN93] techniques. A combination of three possible techniques is proposed in [GR04], comprising HIP (Human Interaction Proof) challenges, computational challenges and paid subscription. In [GHI+05] an approach has been presented for dynamically limiting the number of outgoing messages which is based on the adaptation of a token bucket mechanism. This approach avoids several of the drawbacks of the other cost-based approaches. Nevertheless, several important questions with respect to fine tuning and deployment in practical situations have not been addressed adequately so far.

1.2 Focus of this Paper

In this paper, we address these questions related to the approach presented in [GHI+05] in order to make progress towards the goal of self-learning and fully transparent UCE prevention. The remainder of this paper is organized as follows. In Section 2 the basic structure of the token bucket-based approach introduced in [GHI+05] is reviewed, and in Section 2.5 the open questions related to this approach are summarized. In Section 3 solutions to these questions are presented and the concept for an intelligent system for UCE prevention is developed. Section 6 concludes our paper.

2 Token Buckets for UCE Prevention

As summarized above, existing approaches for preventing UCE have various drawbacks. The most effective ones impose restrictions which tend to also affect regular e-mail users. Ideally, a pre-send method (i) limits the sending of e-mails in such a way that the business model of spammers is harmed while (ii) its effects are *unnoticed* by regular e-mail users. Investigations of the business models underlying spamming indicate that this goal can be reached [ISGP06]. The concept outlined in [GHI+05] was developed accordingly. In the following, this concept is reviewed briefly.

If individual S intends to send UCE to a number of recipients, S needs to be connected to the Internet through some ISP X. Nevertheless, S does not necessarily have to send e-mail through the outgoing mail server of X. Theoretically, S could connect to an open proxy,

to an open relay or to a third party e-mail provider *inside* X's network and use that to send out e-mail. S could also run his own e-mail server (or a spamming tool) locally. In their Terms of Use, ISPs often explicitly prohibit the installation of an open proxy, open relay or of a private e-mail server inside their network. Consequently, the latter two situations involve questions concerning the enforcement of the internal structure and regulations of the ISP, which are beyond the scope of this paper.

The strategy originally proposed in [GHI+05] comprises two core components in order to handle both of the relevant scenarios.

- A *token bucket* component limits the flow of e-mail through the outgoing mail server of X. If parameterized properly (see Sections 2.5 and 3), this happens *unnoticed* by a regular e-mail user.
- A component consisting of whitelists restricts or inhibits the outflow of e-mail through channels other than the outgoing mail server of X (open proxies, open relays or third party e-mail providers).

Another scenario which has become extremely important in recent years is that spam is sent out via *bot nets*. In fact, nowadays this is most likely the case for the biggest portion of spam. In this scenario, computers of regular users are hacked, "infested" with programs and abused for sending out UCE. Also for this scenario the strategy discussed here provides important improvements, since a well designed token bucket system helps to detect whether e-mail traffic originating from certain computers starts deviating from its "normal" behavior, which may be an indication of this kind of abuse.

2.1 Token Bucket Fundamentals

A *token bucket* is a way for implementing a shaping algorithm for generic network traffic with inherent bit rate saving capabilities (see Fig. 1).

In Fig. 1, *two* "buckets" are present, one for storing incoming traffic in a queue with a capacity of K bits, another one for storing a maximum of β so called *tokens*. Each incoming bit needs to remove a token from the token bucket to be forwarded, and the token bucket is filled with a rate of ρ tokens per time unit [Sta02]. If a sender does not send data for some time, the token bucket will fill up. If the token bucket is full, no more tokens can be added to the bucket. If the traffic queue is full, incoming packets are dropped. If the token bucket contains $0 \leq n \leq \beta$ tokens, then up to n incoming bits may be forwarded in a single burst (maximum burst size). However, in the long run, the committed sustainable bit rate is limited by ρ.

The token bucket concept has been adapted to e-mail traffic (which can be considered a special form of network traffic) in [GHI+05].

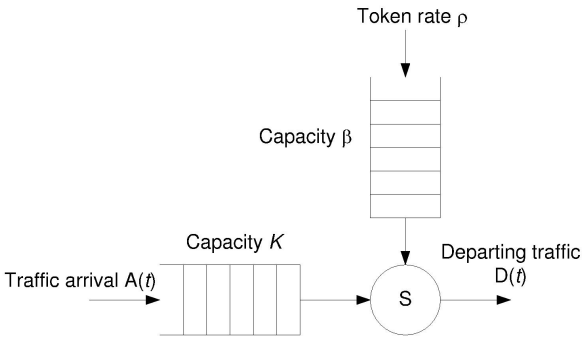

Figure 1: A token bucket.

2.2 Implementation

Our token bucket approach for spam prevention was implemented as a policy delegation server for the open source mail server Postfix (`http://www.postfix.org`). For each user, a triplet consisting of (i) the mailbox name, (ii) the time t_0 of the last successful submission of the "RCPT TO:" command (SMTP status code "250"), and (iii) the number T of tokens currently available in his bucket is stored. Every time a client transfers a message to the outgoing mail server, a plug-in is triggered after the execution of the "RCPT TO:" command within the SMTP dialogue. If the e-mail has r recipients, this plug-in is executed r times.

Each execution of the plug-in computes the number of available tokens based on the following parameters: capacity β (= maximum number of tokens available per user), token consumption T_c per recipient, and token growth ρ per time unit.

Based on the current system time t (when the sending takes place) and on t_0 when the previous sending happened, the new amount T of tokens currently available in the bucket can be computed using the following equation:

$$T(t) = \min\left(T(t_0) + (t - t_0) \cdot \rho, \beta\right)$$

This number is the basis for the decision whether the e-mail can be sent to a certain recipient or not. If $T(t) - T_c \geq 0$, the recipient is accepted, the number of available tokens is updated in the user's triplet as $T := T(t) - T_c$, and the plug-in returns status code "250 OK" to the mail server. If $T(t) - T_c < 0$, the e-mail cannot be sent, the recipient is refused, and the plug-in returns "554 Not enough tokens available" back to Postfix. Alternatively, it is possible to *delay* sending of this e-mail until enough tokens are available.

2.3 Parametrization

The central question is how to choose the parameters β and ρ. A simple and straightforward solution is to put a *static limit* on the number of e-mails which can be sent out per time unit (for example, allowing each user to send out at most 100 e-mail messages per day), or on the number of recipients for a single e-mail (cf., for example, the terms of service of Yahoo). Obviously, such rigid limits easily cause notable restrictions for regular e-mail users, because the e-mail to be sent out is usually not distributed uniformly over a day. Generally speaking, for the success and widespread acceptance of any measure against spam it is crucial to design it as transparent and imperceptible as possible for a regular e-mail user.

In summary, the objective has to be the following: Limit the number of e-mail messages sent out in order to compromise the spammers' business model *without* negatively affecting a regular e-mail user (ideally, a regular e-mail user should not notice anything). A token bucket mechanism is a suitable strategy for achieving this goal, because it provides a very flexible way of limiting the number of e-mail messages which can be sent out. It can also accommodate for traffic bursts, and, if parametrized and adjusted properly, can achieve the objectives formulated above. In the next sections, we will discuss proper parametrization and adjustment of a token bucket mechanism for controlling outgoing e-mail traffic.

2.4 Non-Standard Channels for Sending out E-Mail

Obviously, the token bucket component described here can only control the e-mail traffic passing through the ISP's outgoing e-mail server. As mentioned before, a spammer has various options for circumventing this mechanism and for sending out spam through other channels. Thus, we extend our concept by another component which allows one to control these "loopholes".

A first idea is based on a "filter-in" approach. This means, that the ISP allows outgoing connections *only* to certain registered IP addresses on a "whitelist" (for each customer individually). While this approach has some initial overhead in terms of implementation and user interaction, it offers completeness, because certain inspections of outgoing connections can be integrated into the registration process. In practice, the ISP has to check connection attempts going out of its network for IP addresses on the whitelist and decide accordingly whether to allow the connection or not.

A possible alternative is a "filter-out" approach which requires an efficient procedure for maintaining a reliable, up-to-date and comprehensive list of open proxies and open relays. Although several such lists are publicly available (for example, [rea]), it is virtually impossible to have a complete list of all currently open proxies and open relays which could potentially be used for spamming. In order to complement and extend publicly available lists, a feedback mechanism from a spam filter for incoming e-mail traffic could be used to identify those hosts in a completely automated fashion which does not require user interaction.

Between those two alternatives, the "filter-in" approach is more appealing in terms of performance. Each user's whitelist can be established as part of the registration procedure. During this registration process as well as in regular intervals afterwards the ISP can check whether these whitelisted IP addresses continue to be "trusted" hosts.

The most comprehensive, but technically most involved alternative would be to monitor the entire TCP stream directly and to integrate it into a token bucket mechanism. Whenever the observation of the TCP stream reveals that token bucket limits are approached or reached, the respective connection is slowed down or terminated.

2.5 Open Questions

As discussed in [GHI+05], the token bucket concept has several properties which are attractive in the context of spam defense: It is very simple, both to implement and to handle, and due to its simplicity it is very efficient and does not cause any severe performance overhead. One of its main attractions is the fact that a *flexible* and *adaptive* limitation of the outflow of e-mail messages becomes possible. Limiting this outflow has the advantage of removing one source for spam, and is in some sense also a "protection" for an ISP for not getting blacklisted.

The main challenge discussed in this paper is *how to fine-tune* a token bucket system for spam prevention so that the spammers' business model is harmed while the regular e-mail user is virtually not affected at all (see Section 3).

How Many Tokens per Message ? When applying a token bucket strategy to spam prevention, the system can either use one token per message or it can determine the number of tokens needed based on the size of the e-mail (which corresponds to using one token per fixed amount of data to be sent). In this paper we focus on the former variant since the essential information to be transferred in spam tends to be relatively compact, containing only a short text, a URL, or sometimes small images, and thus the size of spam messages tends not to vary a lot. In order to address this and related questions, we first discuss the economic background of the spam phenomenon.

3 Economic Background

The commercial success of spamming is related directly to the number of spam messages sent out per time unit. This is explained by the underlying business model: Spammers try to send out big numbers (millions) of small-sized messages. Although the relative response rates tend to be rather low – between 0.00001 % and 0.35 % (see [ISGP06] and references therein), this is sufficient in terms of absolute numbers as long as the number of messages sent out is large enough (assuming a fixed income associated with each response). Due to this dependency on high sending rates, a limitation of the number of outgoing messages is an effective measure. As motivated before, the concept we use to impose such a limit is

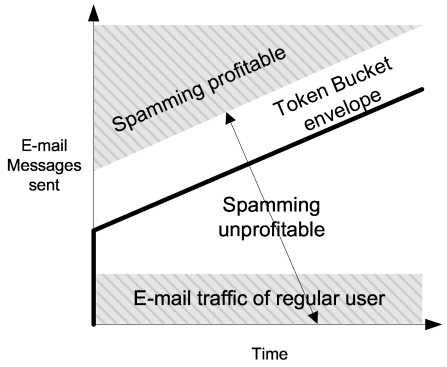

Figure 2: Token bucket envelope spearating the region of regular e-mail traffic from the region of profitable spamming.

a token bucket. In order to address the central question in Section 4 – how to choose the parameters so that the impact on regular users is negligible – we first need to take a closer look at some statistical data.

In case studies based on data from real cases available in the literature [ISGP06], one can see that a single spammer can create large daily revenues (basically only limited by the bandwidth of the internet connection) if the number of messages sent out is not restricted. A limit of a few hundred messages per day is normally sufficient to reduce the spammer's potential revenue below the marginal return. Roughly speaking, a spammer's operation does not start being profitable before he sends out at least in the order of several thousand messages per day [GIL+05, ISGP06].

3.1 Business Model of Spammers

The profit generated by spamming increases with the number of e-mail messages sent out. In other words, if one wants to render spamming unprofitable, outgoing e-mail traffic has to be shaped in a certain way such that profitable regions are not reached (see Fig. 2). In Sections 4 and 5, we describe how to design a token bucket shaper for outgoing e-mail traffic such that regular e-mail users are within the envelope, whereas (profitable) spamming is outside the envelope.

First, we need to take a closer look at the costs a spammer is facing when sending out a certain amount of messages and trying to create revenue. We will also try to relate this to the costs for a regular user when sending out e-mail in order to find economical barriers which impede spammers while being unnoticed by a regular user. With the knowledge gained here it is possible to create new tools for outgoing spam prevention which can help internet service providers (ISPs) reducing the load on their servers as well as securing their reputation by preventing their customers from spamming.

Cost and Revenue Factors for Spammers. Before different cost and profit models can be analyzed, the most important cost and revenue factors must be identified. The cost factors for a spammer can be classified in four categories—hardware cost H, software cost S, labour cost L and operating cost O. Some cost, such as hardware cost, is easy to evaluate, but others can only be estimated (software installation duration, time to compose a spam e-mail). So we define a simple cost model as

$$\text{total cost} \quad c = H + S + L + O.$$

Assuming that a single spammer uses his own home computer, hardware cost H comprises cost of a computer C, a monitor M and peripheral devices P:

$$H = C + M + P.$$

Software cost can be divided into basic software requirements, such as an operating system OS, and special software for spamming activities, such as remailers R, mail address harvesters MAH or web hosting WH:

$$S = \text{OS} + R + \text{MAH} + \text{WH}.$$

Operating cost are a sum of internet service cost I, electricity cost E for running the system, address collection cost A (e-mail addresses can be bought or self-collected) and open proxy cost OP:

$$O = I + E + A + \text{OP}.$$

Labor cost can be divided into cost for installation IN, maintenance MT, mail production MP and customer acquisition AC:

$$L = \text{IN} + \text{MT} + \text{MP} + \text{AC}.$$

All these cost factors must be calculated for a certain period of time (for example, per day). This daily cost must afterwards be divided by the number of messages which can be sent out in this period of time to get the per message cost. If the cost calculated here is higher than the revenue, then the attempt of destroying a spammers' business model can be considered successful.

While the cost factors can be estimated for these more "classical" forms of spamming businesses, it is much more difficult to estimate cost factors for spamming operations which are based on bot-nets. Revenue factors are also very hard to estimate, because spamming business usually operate secretly. There are two main payment schemes - a brokerage system, where a marketer gets a fee per sold item and a pay-per-mail campaign system (for example, an e-mail campaign to one million customers costs a certain amount of money). As it is not possible to get reliable data about the former we analyze the mail campaign system here. As discussed in [GIL+05], reasonable estimates for the average revenue per message are around 0.004 Euro.

3.2 The SpamSim Tool

The different business models motivating the spam phenomenon have been summarized in [ISGP06]. In order to design and to calibrate an anti-spam method which is capable of

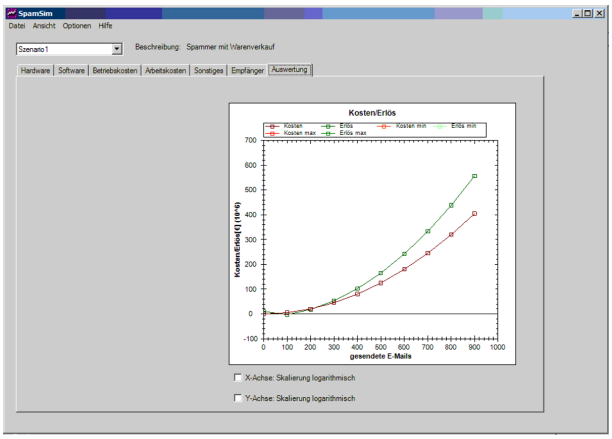

Figure 3: Screenshot of SpamSim application.

interfering with these business models, we created a simulation tool. This tool allows us to simulate the effect of various parameters on the profit achieved by spammers.

Our tool implements the most important cost and profit factors and models their dependencies, however, for the "classical" forms of spamming businesses. We try to answer some of the underlying questions concerning spam, like how many messages must be filtered out, or how many messages a spammer has to send out to be profitable. By examining the break-even point for spammers in terms of cost and profit, we can evaluate the effectivity of anti-spam methods. The tool has been implemented as a Windows GUI application, programmed in Visual C# operating on the Microsoft .NET framework.

Using this application it is easy to generate graphs representing the the cost and revenues of spammers (as shown in the screenshot in Fig. 3). Using this tool, we have analyzed the influence of a token bucket algorithm as outlined in [GHI+05] on the business model underlying UCE. Further details and the investigation of other scenarios are provided in [ISGP06].

Based various assumptions, including that a home PC and a leased line are used for sending out UCE, that hardware, software, operating cost and working cost are monthly fixed costs and that only open proxy costs are paid per e-mail sent and thus can be denoted as running costs, one can estimate that a sending limit of roughly 8 400 messages per day will make sending out UCE unprofitable.

4 Parametrization of Token Buckets

As summarized in Section 2, a token bucket is specified by two parameters: capacity β and token rate ρ. For determining good choices for these two parameters, a rough estimate of the number of outgoing e-mail messages from a *regular* user per day is needed.

Data from ISPs indicates that this number tends to be below 20 messages per day (averaged over all customers, cf. [Cla05, SGIS07]). We determine *global* estimate on the basis of (conservative) estimates for the total daily volume of e-mail messages, the total number of internet users, and the overall percentage of spam. Radicati (`www.radicati.com`) estimates that worldwide around 130 billion e-mail messages are sent out per day. According to `www.internetworldstats.com`, the number of internet users worldwide is roughly 950 millions, and Postini (`www.postini.com`) estimates that more than 80 % of e-mail messages are spam. Based on these numbers, we can estimate that around 26 billion non-spam ("ham") messages are sent out worldwide per day. Consequently, on average every internet user sends fewer than 30 ham messages per day.

Consequently, we (conservatively) estimate the average sending rate $\bar{s}$ per user as $\bar{s} \approx 50$ messages per day. Due to legitimate (ham) mass mailers, the sending rate of most private users can be expected to be significantly lower. Therefore, with a choice of $\beta = 2 \cdot \bar{s}$ and $\rho = 2 \cdot \bar{s}/86\,400$ (per second), we can limit the outgoing traffic to at most 100 messages per day, which is clearly below the volume needed for commercially successful spamming according to the previous considerations. At the same time, the big majority of users would hardly notice any restrictions, because they are allowed to send out single bursts of up to twice their average message volume. Large mailing lists or newsletters which are sent out infrequently are exceptions in this aspect, which have to be treated separately.

5 Self-Learning and Fully Transparent Outgoing UCE Prevention

In Section 4, we outlined an external static model for parameterizing the token bucket mechanism first introduced in [GHI+05]. Such a static model which is based on the economic aspects of the UCE phenomenon is very well suited for determining good initial values for β and ρ. However, for a system which is used over a longer period of time, intelligent adaptation to the potentially changing behavior of individual regular users is needed. Thus, an adaptive control strategy for achieving this is described in this section.

The central idea is based on the *Chebyshev inequality*. If a random variable X has a finite mean μ and finite variance σ^2, this inequality states that for all $k > 0$,

$$P\left(|X - \mu| \geq k\sigma\right) \leq \frac{1}{k^2}. \tag{1}$$

In the context discussed here, this can be applied as follows: Let the random variable X denote the number of e-mail messages sent out by a certain user within a certain period of time. Then Equ. (1) allows us to make statements about the probability that this number of outgoing e-mails exceeds a certain distance from the mean number of e-mails sent out so far. This distance is expressed in terms of multiples of the standard deviation of X. In practice, the adaptation of the token bucket mechanism to the observed user behavior proceeds as follows:

1. Choose a time interval t in seconds, which determines the granularity of the prevention mechanism, and another one $T > t$ which determines the time interval after

which the parameterization of the token bucket mechanism is updated. For example, $t = 60$ implies that all parameters relate to a basic time unit of one minute, and $T = 300$ implies that the system's parameterization is adapted every five minutes.

2. Choose a *level of transparency* $k > 1$. The larger k is chosen, the less likely it happens that a regular user wants to send out more e-mail messages than the limit set by the token bucket mechanism and thus the less likely it becomes that a regular user notices anything about this restriction mechanism.

3. Regularly update estimators $\bar{\mu}$ and $\bar{\sigma}$ for the mean and the standard deviation of the messages sent out per time interval t.

4. At the end of every update interval T, set the token rate $\rho = \bar{\mu} + k\bar{\sigma}$ to reflect changes in the observed volume of outgoing e-mail messages.

Such mechanisms potentially prevent spam by regulating the number of outgoing messages. However, spammers may be forced into finding new ways for sending out their messages. In particular, they may attempt to *train* the system if the configuration paramters are learned at an individual basis. Therefore, the estimators $\bar{\mu}$ and $\bar{\sigma}$ of individual users should be regularly compared to the averages over all users at the level of the ISP.

6 Conclusion

We have presented a concept for dynamically limiting the number of outgoing e-mail messages, to be implemented at outgoing e-mail servers. It is based on the adaptation of a token bucket mechanism which we introduced earlier. This concept provides a very flexible and adaptive way of limiting outgoing e-mail traffic. In contrast to static approaches, traffic bursts can be taken into account. We have discussed how to parameterize and tune our concept such that the imposed restrictions are virtually unnoticed by regular e-mail users while the business model of spammers is compromised.

In contrast to existing approaches, our strategy does not affect all e-mail users, but selectively targets spammers. We introduced the concept of applying the Chebyshev inequality for adapting to dynamically changing requirements. Our combination of a token bucket mechanism and the adaptation component makes it possible for an ISP to prevent the sending of spam out of its network. If widely used by ISPs, this strategy has the potential of significantly reducing the amount of spam on the Internet.

However, some open questions still remain. More efficient approaches for handling large mailing lists or infrequently sent newsletters need to be developed. Moreover, the effect of the concept presented here on the size and the dynamics of bot nets (fewer messages per bot should lead to larger bot nets) has to be investigated.

Acknowledgments. We would like to thank Internet Privatstiftung Austria, mobilkom austria, UPC Telekabel, and Internet Service Providers Austria for supporting this research and the anonymous referees for their valuable comments on earlier versions of this paper.

References

[ABMW03] M. Abadi, M. Burrows, M. Manasse, and T. Wobber. Moderately hard, memory-bound functions. In *In Proc. of the 10th Annual Network and Distributes Systems Security Symposium*, 2003.

[AKCS00] Ion Androutsopoulos, John Koutsias, Konstantinos Chandrinos, and Constantine D. Spyropoulos. An experimental comparison of naive bayesian and keyword-based anti-spam filtering with personal e-mail messages. In *SIGIR '00: Proceedings of the 23rd annual international ACM SIGIR conference on research and development in information retrieval*, pages 160–167, 2000.

[Bac02] Adam Back. HashCash—A Denial of Service Counter-Measure, 2002. `http://www.hashcash.org/papers/hashcash.pdf`.

[Cla01] R. Clayton. Stopping Spam by Extrusion Detection. *SIGMOD Rec.*, 30(1):13–18, 2001.

[Cla05] R. Clayton. Stopping Outgoing Spam by Examining Incoming Server Logs. In *Second Conferernce on Email and Anti-Spam (CEAS)*, 2005.

[DHD+06] Nathan Denny, Theodore El Hourani, Jaime Denny, Scott Bissmeyer, and David Irby. SpamCooker: A Method for Deterring Unsolicited Electronic Communications. In *ITNG '06: Proceedings of the Third International Conference on Information Technology: New Generations (ITNG'06)*, pages 590–591, Washington, DC, USA, 2006. IEEE Computer Society.

[DN93] C. Dwork and M. Naor. Pricing via Processing or Combatting Junk Mail. In *In Lecture Notes in Computer Science 740 (Proceedings of Crypto Â2)*, pages 137–147, 1993.

[DS04] P. Desikan and J. Srivastava. Analyzing Network Traffic to Detect E-Mail Spamming Machines. In *Workshop on Privacy and Security Aspects of Data Mining (in Conjunction with the 4th IEEE ICDM)*, 2004.

[GHI+05] W. N. Gansterer, Helmut Hlavacs, Michael Ilger, Peter Lechner, and Jürgen Strauß. Token Buckets for Outgoing Spam Prevention. In M.H. Hamza, editor, *Proceedings of the IASTED International Conference on Communication, Network, and Information Security (CNIS 2005)*. ACTA Press, November 2005.

[GIL+05] Wilfried Gansterer, Michael Ilger, Peter Lechner, Robert Neumayer, and Jürgen Strauß. Phases 2 and 3 of Project 'Spamabwehr': SMTP Based Concepts and Cost-Profit Models. Technical Report FA384018-2, Institute of Distributed and Multimedia Systems, Faculty of Computer Science, University of Vienna, May 2005.

[GJL07] W. N. Gansterer, A. G. K. Janecek, and P. Lechner. A Reliable Component-Based Architecture for E-Mail Filtering. In *ARES '07: Proceedings of the The Second International Conference on Availability, Reliability and Security*, pages 43–52, Washington, DC, USA, 2007. IEEE Computer Society.

[GR04] Joshua T. Goodman and Robert Rounthwaite. Stopping outgoing spam. In *EC '04: Proceedings of the 5th ACM conference on Electronic commerce*, pages 30–39, New York, NY, USA, 2004. ACM.

[Har03] E. Harris. The Next Step in the Spam Control War: Greylisting. Technical report, PureMagic Software, 2003. `http://projects.puremagic.com/greylisting/whitepaper.html`.

[ISGP06] Michael Ilger, Jürgen Strauß, Wilfried Gansterer, and Christian Proschinger. The Economy of Spam. Technical Report FA384018-6, Institute of Distributed and Multimedia Systems, Faculty of Computer Science, University of Vienna, September 2006.

[JGK08] Andreas G. K. Janecek, Wilfried N. Gansterer, and K. Ashwin Kumar. Multi-Level Reputation-Based Greylisting. In *Proceedings of ARES 2008 – International Conference on Availability, Reliability and Security*. IEEE Computer Society, 2008.

[rea] List of Real-Time Spam Black Lists (RBL). `http://www.email-policy.com/Spam-black-lists.htm`.

[SGIS07] Gerald Stampfel, Wilfried N. Gansterer, Michael Ilger, and Konrad Stark. The EU Data Retention Directive 2006/24/EC from a Technical Perspective. Technical Report FA396005-1, Department of Distributed and Multimedia Systems, Faculty of Computer Science, University of Vienna, October 2007.

[Sta02] W. Stallings. *High-Speed Networks and Internets, 2nd ed.* Prentice Hall, 2002.

[Tur03] D. Turner. The Lightweight Currency Protocol, Internet Draft. Technical report, Internet Engineering Task Force, 2003. `http://cis.poly.edu/~ross/papers/draft-turner-lcp-00.txt`.