Optimal Queueing Strategies for Email Virus Scanning

Richard J. Perry
ECE Department, Villanova
University, Villanova, PA 19085
perry@ece.villanova.edu

Mark G. Thomas
MGT Consulting, PO Box 666,
Blue Bell, PA, 19422
mark@misty.com

Abstract — **Optimal queueing strategies are derived for an email virus scanning system consisting of multiple queues of varying message size limits running in parallel. The general VirScan system is described, a queueing model is defined, and expressions are derived for the overall average time that messages spend waiting in the queues. The distributions used for message sizes are based on statistics from real email servers. Queue size limits are then determined numerically to minimize the wait time under two different queueing strategies for several email server examples.**

I. Introduction

Email predates the Internet, and was one of the first services to take advantage of the Internet. In the early 1990s, commercial vendors started to realize the potential of Internet email, and began to move from proprietary message formats and closed systems, to standardized message encoding schemes and servers with integrated Internet SMTP (Simple Mail Transfer Protocol) capabilities [1].

Today, graphical desktop email programs such as Netscape Communicator and Microsoft Outlook are staples of Internet users' desktops. These easy to use but powerful programs have the ability to send and receive plain text messages, as well as messages containing binary attachments, such as word processing documents, spreadsheets, executable scripts, hyperlink tags to web sites, and program binaries. Vendors have given insufficient attention to risks involved with the handling of potentially malicious or disruptive message content. Some of these risks have been documented in a series of Carnegie Mellon University CERT Coordination Center advisories [2, 3, 4, 5, 6, 7, 8, 9].

The ease of pointing and clicking has, for many end users, erased the distinction between launching applications and accessing data with applications. Users no longer need to distinguish - they just point at what they want to access and click with the mouse; the operating system determines which application to launch. To further complicate matters, common spreadsheets and word processors have embedded scripting capabilities in their document file formats, so even supposed documents can contain hidden executable program instructions [10]. Even technically advanced users are easily tricked into launching executable attachments received via email. Furthermore, bugs in email software can force execution of message attachments, even without any user intervention [6].

Internet email has become a significant transport mechanism for computer viruses, as well as junk messages that waste time and resources. Email is a primary means of business communication, but is simultaneously a huge liability. The Internet has become both a necessary business tool and a hostile working environment.

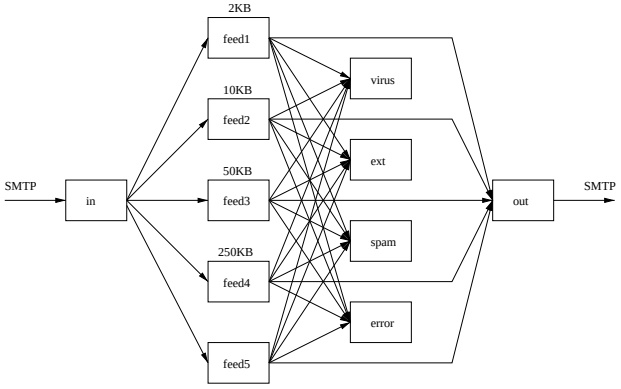

Figure 1: VirScan queue example with N=5 feeds and default message size limits.

System and network administrators must protect their users and networks from the results of email propagated virus attacks. They also need to prevent voluminous quantities of unsolicited junk email from consuming their system resources. Desktop protection is helpful, but is hard to update and support reliably for large numbers of users simultaneously. There is a clear demand for fast, reliable, centralized, server-side email filtering, such as the VirScan system described here.

Email Virus Scanning

In the VirScan Email virus scanning system [11], two sendmail [12] daemons are run. One daemon listens on the SMTP port and stores incoming messages in queue *in*; the other daemon scans queue *out* to deliver messages.

A VirScan feed process scans the *in* queue and distributes messages among N feed queues, *feed1*, ..., *feedN*, depending on the message size and queue load. Each feed queue except for *feedN* has a limit on message size. In the example setup shown in Figure 1, messages over 50KB in size will only be placed in queues *feed4* and *feed5*. Each feed queue load is measured by the time at which the queue is expected to be empty, and the feed process places messages in the least-loaded queue which can accept the message size. This helps smaller messages to get processed faster.

N VirScan work processes are run, one for each feed queue. We assume that the system has at least N CPUs so that the work processes are run in parallel. Alternatively, each work process could run on a separate single-CPU system, with the feed process run on a separate load-balancing front-end system.

Each work process scans messages for viruses, bad file name extensions in attachments, and spam. Bad messages are moved to the *virus*, *ext*, or *spam* quarantine queues. Clean messages are moved to the *out* queue. If an error occurs while

processing a message, the message is moved to the *error* queue.

II. Queueing Parameters

We assume that messages arrive in accordance with a Poisson process having rate λ messages per second. The service time for a message of size s bytes in feed queue q_i, $i = 1, \ldots, N$, is

$$t_i = t_0 + t_s s \ , \tag{1}$$

where t_0 is a per-message overhead time in seconds (related to starting up the virus scanner process and moving files between queues) and t_s is the virus scan rate in seconds per byte. t_0 and t_s could be functions of s or q_i, but to simplify notation we will assume that they are constant. On a single CPU system, t_i should be scaled by a factor of N.

The first and second moments of t_i can be expressed in terms of the moments $\bar{s}_i$ and $\bar{s_i^2}$ of the message sizes in q_i

$$\bar{t}_i = t_0 + t_s \bar{s}_i \ , \tag{2}$$

$$\bar{t_i^2} = t_0^2 + 2 t_0 t_s \bar{s}_i + t_s^2 \bar{s_i^2} \ . \tag{3}$$

The distributions of service times and message sizes are generally not exponential (see Section V), so we must use an M/G/1 model for the feed queues (i.e. Memoryless interarrival times, General service time distribution, and 1 server process per feed queue). In this case, the average time spent by messages in q_i waiting for processing is given by [13]

$$W_i = \frac{\lambda_i \bar{t_i^2}}{2(1 - \lambda_i \bar{t}_i)} \ , \tag{4}$$

where $\lambda_i = \lambda F_i$ is the message arrival rate for q_i, and F_i is the fraction of messages placed in q_i. For the system to be stable we must have

$$\lambda_i \bar{t}_i < 1 \ , \tag{5}$$

which sets an upper bound λ_{max} on the message arrival rate

$$\lambda_{max} = \min_i \frac{1}{F_i \bar{t}_i} \ . \tag{6}$$

$\bar{t}_i$ and $\bar{t_i^2}$ can be computed from (2) and (3) given the distribution of message sizes s in q_i. They will be functions of s_i, the message size limit for q_i.

The overall average time messages spend waiting in queues $q_1, \ldots, q_{N-1}$ is

$$W = \sum_{i=1}^{N-1} F_i W_i \ . \tag{7}$$

This is the objective function to be minimized by choosing the queue size limits s_i, $i = 1, \ldots, N-2$ under the constraint $s_{i-1} \leq s_i$, $i = 1, \ldots, N-2$ with $s_0 \equiv 0$. We assume that s_N is infinite, and s_{N-1} is fixed. This corresponds to a system policy on processing very large messages (which occur infrequently) in a separate queue (q_N) so that their processing does not slow down the processing of smaller messages. If the system has additional CPUs, q_N could be divided into multiple queues run in parallel. Alternatively, s_{N-1} could be treated as a free parameter and the limit on the summation in (7) changed from $N-1$ to N to minimize the overall wait time including the largest messages.

Additional penalty factors can be incorporated into (7) if desired, for example to give more priority to minimizing the average delay experienced by small messages.

Finally, we denote by p_i the probability that a message size s is between two adjacent queue size limits

$$p_i = P(s_{i-1} < s \leq s_i) \ , \quad i = 1, \ldots, N \ . \tag{8}$$

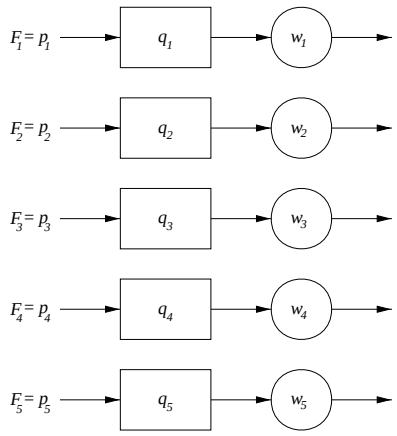

Figure 2: Size-based queueing strategy example with N=5 feeds.

III. Size-based Queueing Strategy

A simple queueing strategy based only on the incoming message size s sets $F_i = p_i$, $i = 1, \ldots, N$ and assigns the message to q_i such that $s_{i-1} < s \leq s_i$. Numerical minimization of the overall average wait time W (7) in this case is easily accomplished, for example using a Nelder-Mead simplex method such as *fminsearch* in Matlab [14].

Figure 2 illustrates the size-based queueing strategy for N=5 feed queues showing that a fixed fraction F_i of message arrivals, based on message size, are placed in q_i for processing by VirScan work process w_i. In this case the message arrivals to q_i are a Poisson process with rate $\lambda_i = \lambda F_i$.

IV. Size-and-load-based Queueing Strategy

The size-based strategy underutilizes the queues. For example, a lot of small messages may build-up in q_1 while the other queues remain empty. A better strategy is to keep track of the queue loads, and place messages in the least-loaded queue which can accept the message size. This will help smaller messages to get processed faster. In this case the message sizes in q_i will range from 0 to s_i.

When a message of size s arrives, with $s_{k-1} < s \leq s_k$ for some $k \in \{1, \ldots, N-1\}$, let W_i^* represent the time at which q_i will be empty, $i = k, \ldots, N-1$. The message will be placed in the queue with minimum W_i^*.

Letting $f_{i,k}$ represent the probability that W_i^* is the smallest of $\{W_k^*, \ldots, W_{N-1}^*\}$, we may then express F_i as

$$F_i = \sum_{k=1}^{i} p_k f_{i,k} \ , \tag{9}$$

where p_k is defined in (8). $f_{i,k}$ may also be interpreted as the fraction of messages in size range $s_{k-1} < s \leq s_k$ which are placed in queue q_i. This is illustrated in Figure 3 where $\hat{q}_1, \ldots, \hat{q}_4$ represent virtual queues for each message size range. For example, when work process w_3 is free, the next message to be processed will be selected from $\hat{q}_1$, $\hat{q}_2$, or $\hat{q}_3$.

Figure 4 illustrates the actual queues. In this case the message arrivals to q_i are not a Poisson process since they depend on the load in q_i, so (4) can not be used to compute W_i. Since there does not seem to be any way to compute $f_{i,k}$ or the distribution of W_i^* analytically, we will use simulation results to evaluate W_i while numerically minimizing W.

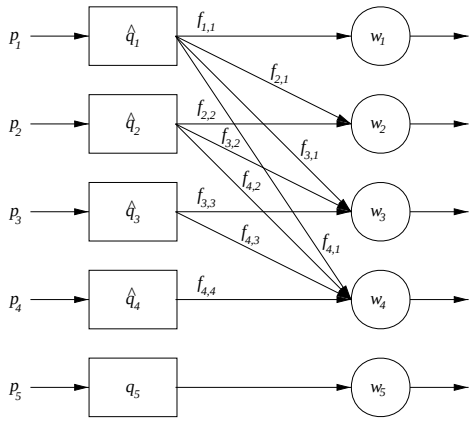

Figure 3: Size-and-load-based virtual queues example with N=5 feeds.

V. Numerical Results

We first examine the message size distribution from a real email server, and describe how the measured sizes are used to construct interpolation tables for the size-based queueing strategy. Then we show results for minimization of the average wait time, and calculation of the maximum message arrival rates. Finally, an example using an arbitrary dual-Gaussian message size distribution is presented.

Message Size Distribution

Figure 5 shows the size probability distribution and density measured from a sample of approximately 100,000 email messages, together with an exponential model with mean chosen to minimize the mean-square-error between the two distribution curves. The measured data does not fit an exponential model very well. There are fewer small messages and more very large messages observed in the data than would be predicted by an exponential model.

For the following results, the measured data from Figure 5 was used to construct tables at $M = 101$ points of message sizes in a geometric scale from which values for p, $\bar{s}$, and $\bar{s^2}$ are interpolated. For D measured sizes $S_1, \ldots, S_D$, with minimum and maximum sizes S_{min} and S_{max}, we set the geometric ratio $r = \left(\frac{S_{max}}{S_{min}}\right)^{\frac{1}{M-2}}$ and size values $z_1 = 0$, $z_2 = S_{min}$, $z_i = rz_{i-1}$, $i = 3, \ldots, M$. Then we set J_i to be the set of indices j for which $z_{i-1} < S_j \leq z_i$, $i = 2, \ldots, M$ and let K_i represent

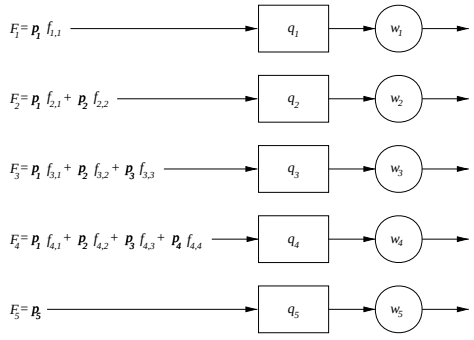

Figure 4: Size-and-load-based queues example with N=5 feeds.

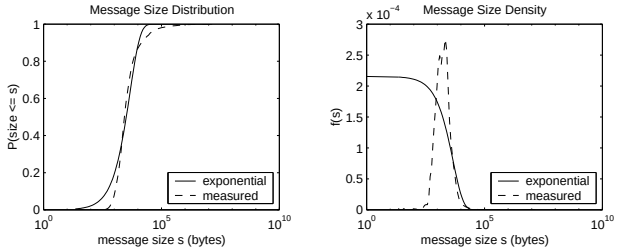

Figure 5: Message Size Probability Distribution and Density

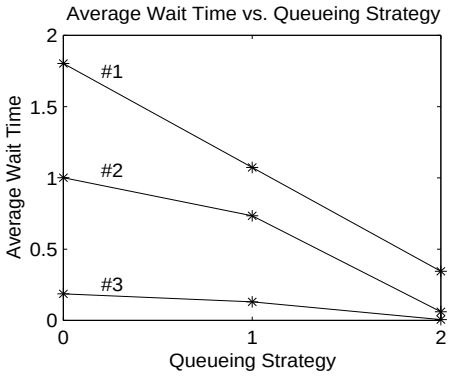

Figure 6: Average wait-time for three different mail servers: queueing strategy 0 uses arbitrarily defined default queue size limits with the size-based strategy; strategy 1 uses optimum queue size limits with the size-based strategy; strategy 2 uses optimum queue size limits with the size-and-load-based strategy.

the number of elements in J_i. The table for the cumulative distribution of sizes s is then $C_1 = 0$, $C_i = C_{i-1} + K_i$, $i = 2, \ldots, M$. The table for $\bar{s}$ is $S_1 = 0$, $S_i = S_{i-1} + \sum_{j \in J_i} S_j$, $i = 2, \ldots, M$ and the table for $\bar{s^2}$ is similar but uses S_j^2 in the summation.

p_i is then obtained directly from two interpolated values of C as $p_i = C_{2,i}/D - C_{1,i}/D$, where $C_{2,i}/D = P(s \leq s_i)$ and $C_{1,i}/D = P(s \leq s_{i-1})$. $\bar{s_i}$ is obtained from $C_{1,i}$ and $C_{2,i}$ and two interpolated values of S as $\bar{s_i} = (S_{2,i} - S_{1,i})/(C_{2,i} - C_{1,i})$, and $\bar{s_i^2}$ is obtained similarly.

Minimization of Average Wait Time

For numerical minimization of the overall average wait time for the size-and-load-based queueing strategy, 10,000 arrivals were simulated, with message sizes based on inverse interpolation of the distribution table C. Then the resulting queue size limits were used in 100 separate trials of 10,000 simulated arrivals, and the wait times for each queue were calculated by averaging over the trials.

The overall average message wait times using the default queue sizes from Figure 1 with N=5 were compared with using the optimum queue sizes for size-based and size-and-load-based strategies using queueing parameters from three different mail servers. Results are shown in Figure 6. Mail server #1 has parameters $\lambda = 1.5$, $t_0 = 0.1$, and $t_s = 0.0002$. Server #2 handles fewer messages and has parameters $\lambda = 0.6$, $t_0 = 1.0$, and $t_s = 0.0002$. Server #3 handles much fewer

	default	size	size+load
q_1	2000	4341	14505
q_2	10000	18862	23781
q_3	50000	67607	248780

Table 1: Queue size limits for mail server #1 vs. queueing strategy

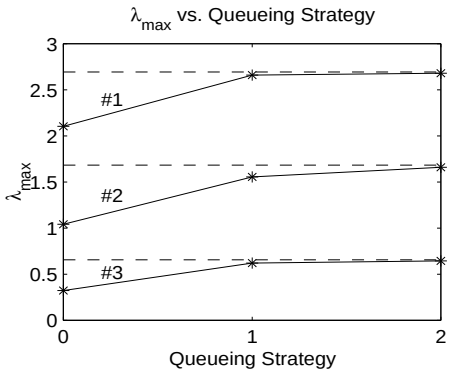

Figure 7: Maximum message arrival rate and bounds for three different mail servers

messages and has parameters $\lambda = 0.0255$, $t_0 = 5.23$, and $t_s = 0.0001314$. The parameters for server #3 are actual measured values from the same system used to produce the message size distribution shown in Figure 5. The parameters for servers #1 and #2 are an extrapolation of server #3 to higher load environments. The optimum queueing strategies demonstrate an improvement in performance in all cases.

Table 1 shows the queue size limits vs. queueing strategy for server #1. Compared to the default, the optimal strategies require significantly larger queue size limits, with the size-and-load-based strategy size limit for q_3 approaching that of q_4 (250000).

Maximum Message Arrival Rates

Figure 7 shows results for the maximum message arrival rates λ_{max} which can be supported by the different queueing strategies, using the same email servers and strategies as in Figure 6. The dashed lines in the figure represent upper bounds on λ_{max} based on feeding the incoming messages of size $s \leq s_{N-1}$ to the $N-1$ queues based on the queue loads but with no message size limits.

Note that λ_{max} for the optimal queueing strategies approaches the upper bounds, while providing lower average delay for smaller messages. This is illustrated in Table 2 which shows the average queue wait times for mail server #1 vs. queueing strategy.

	bound	default	size	size+load
Avg	0.448	1.803	1.073	0.3128
q_1	0.448	0.043	0.378	0.2015
q_2	0.448	1.284	1.221	0.1521
q_3	0.448	5.137	4.978	0.6081
q_4	0.448	27.29	18.20	0.6405

Table 2: Average queue wait times for mail server #1 vs. queueing strategy

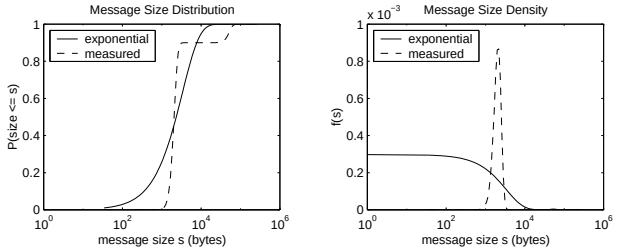

Figure 8: Dual-Gaussian Message Size Probability Distribution and Density

	default	size	size+load
q_1	2000	2062	2891
q_2	10000	9622	6431
q_3	50000	51904	97328

Table 3: Queue size limits for the dual Gaussian message size distribution vs. queueing strategy

An Arbitrary Message Size Distribution

To illustrate how the optimal queueing strategies can be applied to arbitrary message size distributions, an artificial model was created with 90% of the message sizes being Gaussian with mean 2000 and variance 400^2, and the other 10% Gaussian with mean 50000 and variance 10000^2. Figure 8 shows this dual-Gaussian model together with an exponential model with mean chosen to minimize the mean-square-error between the two distribution curves. In this case, the measured data does not fit an exponential model very well at all.

Using this dual Gaussian distribution, and the parameters for mail server #1 described previously, the average wait time using the default queue sizes from Figure 1 is $W = 2.4919$, mainly due to q_4 being overloaded. Using the optimum queue sizes for the size-based strategy yields an average wait time of $W = 1.71$, and using the size-and-load-based strategy yields $W = 0.76$.

Table 3 shows the queue size limits for this case. For the size-based strategy, the size limits are about the same as the default. For the size-and-load-based strategy, the size limit for q_3 is significantly larger than the default, which relieves the load on q_4.

VI. Conclusion

Optimal queueing strategies were derived for an email virus scanning system with message size distributions based on measured statistics. The strategies were shown to work well for several different email server examples, closely approaching the upper bound for maximum incoming message rate, while providing lower average delay for smaller messages.

References

[1] J. Klensin, *Simple Mail Transfer Protocol.* ftp://ftp.isi.edu/in-notes/rfc2821.txt, April 2001.

[2] CERT/CC, *Melissa Macro Virus.* http://www.cert.org/advisories/CA-1999-04.html, March 1999.

[3] CERT/CC, *ExploreZip Trojan Horse Program.* http://www.cert.org/advisories/CA-1999-06.html, June 1999.

[4] CERT/CC, *Love Letter Worm.* http://www.cert.org/advisories/CA-2000-04.html, May 2000.

[5] CERT/CC, *Microsoft Outlook and Outlook Express Cache Bypass Vulnerability.* http://www.cert.org/advisories/CA-2000-14.html, July 2000.

[6] CERT/CC, *Automatic Execution of Embedded MIME Types.* http://www.cert.org/advisories/CA-2001-06.html, April 2001.

[7] CERT/CC, *Nimda Worm.* http://www.cert.org/advisories/CA-2001-26.html, September 2001.

[8] CERT/CC, *Automatic Execution of Macros.* http://www.cert.org/advisories/CA-2001-28.html, October 2001.

[9] CERT/CC, *Microsoft Internet Explorer Does Not Respect Content-Disposition and Content-Type MIME Headers.* http://www.cert.org/advisories/CA-2001-36.html, December 2001.

[10] R. J. Martin, *MS Word 6.x Macro Viruses Frequently Asked Questions for the ALT.COMP.VIRUS Newsgroup.* http://www.bocklabs.wisc.edu/~janda/macro_faq.html, March 1996.

[11] CyberSoft, Inc., *VirScan (patent pending).* http://www.cyber.com/.

[12] The Sendmail Consortium, *sendmail.* http://www.sendmail.org/.

[13] S. M. Ross, *Introduction to Probability Models.* Academic Press, Inc., 1993.

[14] The MathWorks, *Matlab.* http://www.mathworks.com/.