Drinking from the Fire(walls) Hose: Another Approach to Very Large Mailing Lists

Strata Rose Chalup, Christine Hogan, Greg Kulosa, Bryan McDonald, and Bryan Stansell
Global Networking and Computing, Inc.

Drinking from the Fire(walls) Hose: Another Approach to Very Large Mailing Lists

Strata Rose Chalup, Christine Hogan, Greg Kulosa, Bryan McDonald, and Bryan Stansell – Global Networking and Computing, Inc.

ABSTRACT

This paper describes a set of tools and procedures which allow very large mailing lists to be managed with the freeware tool of the administrator's choice. With the right approach scaling technology can be applied to a list management tool transparently.

In recent years, many ingenious methods have been proposed for handling email deliveries to mailing lists of several thousand subscribers. Administration of a mailing list is not limited to message delivery, however. Tasks such as managing subscribers, dealing with mail bounces, and preventing list spamming also become more difficult when applied to very large lists.

As a case study, this paper describes the process of moving the well-known "Firewalls" mailing list from its original home at GreatCircle Associates to a new infrastructure at GNAC. The process was thought to be straightforward and obvious, and it soon became apparent that it was neither. We trust that our discoveries will benefit other systems administrators undertaking similar projects, either concerning large mailing lists or moving complex "legacy systems."

Introduction

"And you may ask yourself,
Well . . . How did I get here?"
– Talking Heads

The Firewalls list began in 1992, at GreatCircle Associates. It quickly evolved into an important forum for new ideas, in-depth technical discussions, and impassioned flame wars. Eventually it would encompass roughly 4500 real-time subscribers and 4900 digest subscribers. A large number of total subscribers were "exploder" or reflector lists passing Firewalls list traffic to unknown third parties at companies and universities around the world.

Daily message counts ranging from a norm of 20 to peaks of 75 or more yield message deliveries of 95,000 to 342,400. In addition, a growing problem with spammers began raising both list traffic and the collective blood pressure of subscribers and list administrators.

In the fall of 1997, list founder Brent Chapman joined a startup company as a key player and realized that he would have little attention left over for anything else. In his own words:

> "Firewalls and Firewalls-Digest are very popular, high-volume mailing lists, and they take a fair amount of time and effort to maintain. Life in a high-profile Silicon Valley startup doesn't leave much time for anything else, though, so Great Circle Associates is going into hibernation. Therefore, after five and a half years and 111+ Mbytes of discussions spanning 45,517 messages and 3018 digests, the Firewalls and Firewalls-Digest mailing lists are moving to a new home at GNAC, which is a consulting and managed services firm based here in the Silicon Valley that I think highly of." [0]

Given the explosive growth of the list over the years, and the demands on Brent's time, it is very much to his credit that the list was still functional up to that point. Now it was GNAC's turn to re-examine the list and find out how to bring it back up to speed.

The Existing System

"Like a crystal cathedral afloat on the tide
comes a mountain of ice
on the course to collide,
while passengers sleep thinking
God's on their side.."
– Peter Schilling,
"Terra Titanic"

The Firewalls list environment that GNAC inherited turned out to be, as we expected, a complex system of many moving interdependent parts.

We quickly discovered that the core of the Firewalls list was the expected Majordomo list manager [1] wrapped around a dual-sendmail queueing structure.

To optimize the handoff between majordomo and sendmail, Brent had set up a special outbound queue area for list traffic. The sendmail_command and sendmail_command_flags in majordomo.config were modified to implement a queue-only sendmail [2] in a custom queue area.

In addition to the basic Majordomo processing of the lists, part of the "Firewalls list" functionality was providing archives via FTP and HTTP. The Mhonarc [3] text to HTML converter and some scripting glue took care of the web-accessible archives, and scripts regularly copied standard Majordomo archives into an FTP hierarchy.

There were a number of watchdog scripts to warn about majordomo list processing malfunctions (e.g., list truncation), as well as some behind the scenes scripting that created a meticulous ‘‘clean archive’’ of the list postings for paranoia’s sake.

While there were certainly intricacies, we could see that the basic structures were sound, and working well enough on the Great Circle server.

Where Angels Fear to Tread

We had made some detailed queries about the Great Circle server, looking for load patterns and other duties performed by the machine. We’d chosen a similar (in fact, more heavy-duty) server for our purpose and felt confident that it could handle anything that the older Great Circle server had been handling.

For the configuration of the server, we decided to make minimal changes to the operating systems and messaging configurations while the list moved. We carefully prepared a tarball from Brent’s server, installed it on our host, and did the basic hostname customizations required to make it run. The preliminary tests looked good. Mail queued into the right places, appeared in archive directories, FTP storage, web pages. Digest files grew. We were ready for prime-time.

We arranged a special ‘‘test mode’’ that would simulate list traffic [4] and, with great anticipation, we turned the key. We knew that there was a heavy spam load, and a lot of traffic, but we were on a faster server with more disk spindles, greater memory, and a wider network pipe. Nothing could go wrong. Go wrong. Go wrong. [5]

In the Cold Light of Day

Test messages would never make it out of the queue. The server would chronically lock due to forking problems. There were crashes and lockouts due to memory problems. Well, this is why we test things in the first place.

The dismal failure of the first ‘‘production test’’ caused us to re-consider our stand on keeping the list management machine and software as close to the original as possible. We realized that we needed to go back to ‘‘square zero’’ and examine the fundamental structure of how the message flow worked. After a couple of false starts, detailed in subsequent sections, we soon had messages turning around in record time. At that point we turned our attention to the secondary challenges which we we had inherited with the management of the list, bounce mail and spam. These at least had been the focus of directed planning for future enhancements.

The amount of bounce mail that a list of the size and volume of Firewalls can produce is phenomenal. Interspersed with the bounce mail would be real requests from real people who needed something from the list managers. We had to find some way to ensure that we were responding in a timely manner to these requests without getting overwhelmed by the bounce traffic.

We also knew that the list had become a favorite venue for spammers sending their useless and annoying missives. Stopping the spam while still allowing legitimate posting would pose its own challenges, due not only to the sheer number of subscribers but to the percentage of ‘‘subscribers’’ which were actually mailing lists rather than individuals.

The volume of messages to the list over time is shown in the graph below. Shortly after GNAC took over the list the spam problem reached a critical point,

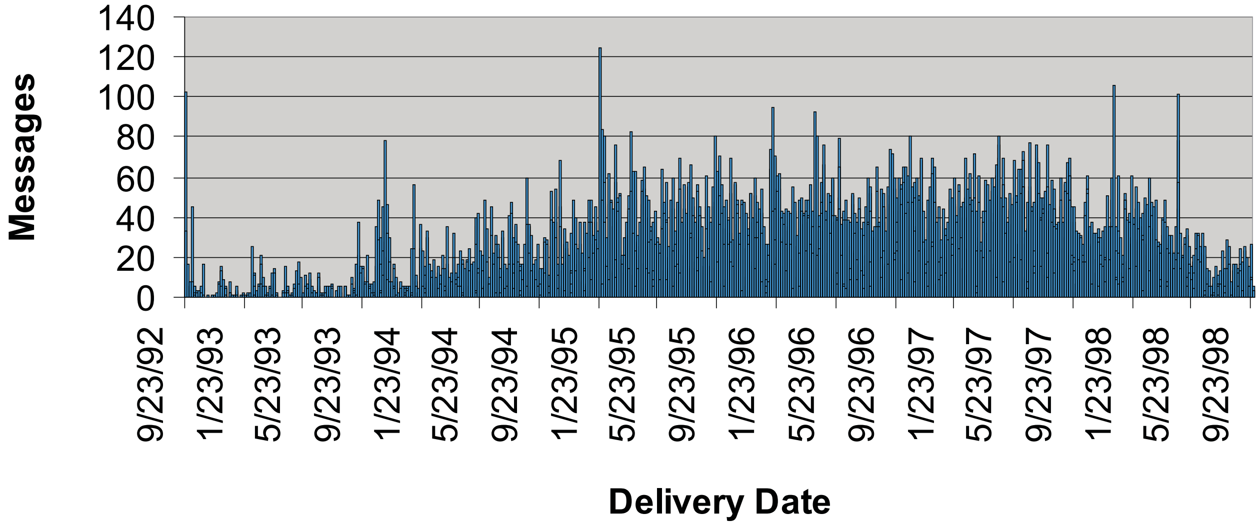

Figure 1: Daily message deliveries.

traffic to the list was high and consisted mostly of spam and complaints about the spam. Immediately after we implemented spam-control measures, list traffic dropped to an uncharacteristic low for a while before resuming more normal levels, but without the spam.

Host Tuning

"Close to the middle of the network,
It seems we're looking for a center.
What if it turns out to be hollow?
We could be fixing what is broken."
– S. Vega,
"The Big Space"

The Great Circle server had been largely untuned, which surprised us in light of the implied message delivery load of the Firewalls and Firewalls-Digest lists. However, in the interest of minimizing changes, we had gone with an "out of the box" BSD/OS config. We had also slavishly copied the configurations of the Firewalls-specific application services in our zeal for compatibility.

As we discovered, our server was doing much more than the original server. As you will see below, the Great Circle server was not actually delivering all the messages to the subscribers, but instead was using relaying services of a large ISP for the actual delivery. Our machine was consequently not tuned correctly to deal with the memory and process usage profile that this task required.

We first became aware that the original machine was relatively un-tuned when we discovered that the sticky bit was not set on the system copy of Perl. From there we did further digging, and decided we needed to go over the entire system and analyze it from scratch. [6,7]:

Here are the major changes we introduced. None of them is necessarily dramatic in impact, but together they represent considerable improvement.

- put operating system and application binaries on different disks
- doubled our swap space
- balanced swap between disks
- set sticky bits on Perl, sendmail, other key system apps
- set sticky bits on all Majordomo binaries & scripts
- rebuilt kernel and upped syslimits.h variables (MAX_CHILD, NPROCS)
- installed a cacheing named [8]

Later we would come to double the physical memory and increase KMEMSIZE to handle some unusual custom processes that we will be describing below.

Message Delivery

"I've been standing here waiting,
Mr. Postman,
so-o-o patiently –
for just a card or just a letter . . ."
– The Marvelettes,
"Please, Mr. Postman"

Once we had the machine tuned, we turned to the list processing itself. Message turnaround time had been in the order of days before we took over the list. It was still at that order of magnitude, and when the message volume was high, our outbound queue was growing faster than mail was getting delivered. There was potential for serious backlogs that would cripple the list.

Mail basically wasn't moving. We knew that historically the list was plagued by slow mail, but that the queues didn't back up too badly. Why was our mail backing up? We went back to the Great Circle server to find out. The answer turned out to be our choice of "smart host" in the sendmail-lists.cf file, which we had blithely customized to work with the usual GNAC environment.

As we mentioned earlier, Majordomo was configured to use a queue-only sendmail (designated "sendmail-lists") for message generation. A separate sendmail daemon would process that queue and keep it moving. We discovered that for the sake of expedient message processing by Majordomo, all recipients of the message were packed into a single RCPT line. Those of you who have dealt with sendmail extensively are wincing right now, aren't you?

We also discovered that, in a neat private arrangement dating back several years, Brent had arranged for his servers at greatcircle.com to have relaying capabilities through the UUNet mail servers. Thus the Great Circle sendmail-lists.cf file merely specified "mail.uu.net" as the smart host, causing everything to be forwarded to it for processing. Due to the way UUNet round-robins its mail services, this would effectively spread out the processing of these troublesome messages with monstrous RCPT headers.

Of course, GNAC did not have the option of passing those messages to UUNET. We would have to deliver these messages directly, 4K of RCPT addresses or not. While GNAC has an excellent mail infrastructure, their core business does not involve ISP-style mail for thousands of individual subscribers. Thus GNAC did not have the quantity of dedicated mail-delivery resources to simply toss the messages into the network and let them go.

Chunk-Style, Just Like Home-Made

A message with over 4000 recipients can take literally days to deliver. A sendmail instance processing the message will work its way laboriously down the recipient list, pausing at every time-out. It may run out of resources and die, causing a new sendmail to take up the torch. No problem, you say, since the new sendmail instance can use the xfNNNNN queue file to pick up where the old one left off. Yes, but first it has to retry all the "deferred" hosts that timed out. Even if the messages are being farmed out to multiple servers, each individual message is going to reach individual subscribers in a highly non-deterministic fashion.

In analyzing our logs and transfer status files, we found that message deferrals would typically be due to remote name servers or mail servers failing to respond before timeout. Due to the large and diverse population of the list, we would see rather shocking ratios of failures to successful deliveries. Many of those failures were multiple failures trying repeatedly to deliver the same message to the same site.

At an architectural level, we knew that we had to get away from the multi-thousand RCPT lines business. We also knew that we couldn't simply force one recipient per message without making the sendmail queue directory so large that directory search time would became a significant factor. Directories with over 10K nodes are generally undesirable [6] and at over 4K subscribers we would quickly flood the queue directory.

Initially we assumed that we would get our biggest "win" by employing a program such as mailcast [9] to batch and sort the recipients by domain and MX record. Mailcast would simply queue them up and send one nice copy off to the right host and we'd shake each others' hands and go off to hoist a cold one or two. Imagine our consternation when we discovered that in fact out of approximately 5,000 individual subscription addresses, some 4,000 were in fact unrelated by host, domain, or MX record. For the roughly 4,000 digest recipients, we found about 3,500 uniquely unrelated addresses. Ouch. For us, this approach was largely indistinguishable from "one recipient per message."

Since there was not a strong natural grouping between addresses, it seemed that arbitrary recipient chunking would be the way to go. We immediately thought of bulkmail [10], a mail-sending utility that can perform chunking on huge recipient lists. As we looked into the specific configuration of bulkmail, we found that bulkmail and majordomo were not trivially compatible. Majordomo wants to invoke a mail command and send a message to it. Bulkmail wants to read in files with a message and a recipient list. We spent a couple of lunches wrangling over which of them to hack to accept the other's view of the world, and how exactly to structure the changes to minimize future-release porting issues. Any way we looked at it, it looked ugly.

The Portable Queue

At this point, having gone far enough down the rathole to smell cheese, we popped back up into the sunshine to re-examine the original goal, namely chunking the messages into deliverable size. We realized that we were already producing a clean, queued message with a highly well-defined structure [11] in a place that we could control. There was no reason that we had to perform the chunking at message generation. We could do it right in the queue itself.

Before beginning the move of the Firewalls list, we had done some preliminary mailflow architecture. Our original plan was to move the list without structural changes, then to apply our idealized architecture in careful stages. Based on the production testing, we clearly had to accelerate things quite a bit.

One of the original elements we'd planned to introduce was time-based queuing, where messages are recursively sifted among various queues based on how long they have been pending [12]. Reading up on this approach, we were reminded once again of what every postmaster knows: queue files are portable.

One of the standard postmaster rites of passage is dealing with a major multi-day mail backlog on your bastion host. You eventually realize that the most sane thing to do is to turn off incoming mail, move all the queue files into a holding directory, then turn on mail again. Meanwhile, you do a little quick scripting to sort things based on which internal mailhubs are in the envelope headers, make a few tarballs, and just FTP them down into the right mailhub's queue, unpack, voila! We decided to go one step further and "MIRV" [13] the queue files.

Split Personality

We turned off the "sendmail-lists" invocation of sendmail and replaced it with a cron job called "qsplit." The qsplit Perl script runs every 5 minutes and examines the sendmail-lists queue directory. Each queue file is parsed. The unique portion of the name (e.g., "ABC12345" in "qfABC12345") is stored as $ident and used to generate new qf files. If the number of RCPT lines in the qf file exceeds the qsplit variable $CHUNKSIZE, the message is processed into multiple messages of $CHUNKSIZE recipients and zero or one messages of less than $CHUNKSIZE.

Each new qf file has a sequential number appended to $ident. Thus the first split file from "qfABC12345" would be "qfABC123451," then "qfABC123452" and so on. Since sendmail will generate unique queue file identifiers within a given sendmail queue area, using this method guarantees unique identifiers for split queue files.

Qsplit is also configured to know about an arbitrary number of sendmail queue directories. If the number of recipients in a parsed qf file is less than $CHUNKSIZE, qsplit will move the message into one of the preconfigured queue directories. A round-robin effect is achieved by keeping track of the last queue directory into which a file has been placed and putting the next one in the subsequent directory, wrapping around as necessary.

For efficiency, each ''new'' df file is merely an appropriately numbered hard link. This is particularly important for the Firewalls-Digest postings, where the df file can be quite large. Note that since hard links will not work across partitions, the ''sendmail-lists'' directory and the processing queue directories must be on the same filesystem. Qsplit of course removes the original qf/df files after the splitting is accomplished. This is why we use hard links rather than symbolic links, since hard links have no concept of ''the original file.''

Splitting the qf files in this way had a dramatic effect on message turnaround time, as the following graph shows. For more than a week before we finally settled upon and implemented our splitting solution, we had been manually splitting qf files and distributing them as described. That period is seen in the graph as a low before a final spike.

Spawn 'til You Die

Each of the queue directories (10, at present) runs separate instances of sendmail. All of the directories are managed by a shared spawning daemon, called simply ''spawn.pl''. Some experimenting was necessary to find the right timing to use within the configurations, so various copies named things like ''slow-spawn'' were created for test runs.

The queue management daemon (spawn.pl) spawns as many copies of itself as there are queue areas. It then watches its children and re-starts any of the spawned processes that die. Each of these child spawners is chartered with keeping ten sendmail daemons running to process its queue area. The child spawners keep track of their sendmail children, restarting a new sendmail whenever one dies.

There is logic built into the spawners to check configurable variables for the load average on the system, and the amount of memory available. If the load is too high, or memory too scarce, the child waits until there are more resources available before starting a new process. The initial spawning of the children processes themselves is also subject to the same limitations. The load average limit in the spawner is set lower than the sendmail threshold, since starting a sendmail will cause a load average spike that might cause sendmail to not do anything once started. In addition, a variable controls the timing between each child spawner or new sendmail, so as to minimize load disruption.

To further speed things up, we also implemented sendmail's host-status feature. This creates a directory structure containing information about when a sendmail last tried to contact a given host, and whether it was up or down. If it is down, sendmail doesn't try that machine again unless the specified re-try timeout has expired. We used the sendmail default of one hour.

The trade-off between the massive amount of disk access that this caused and the saved processing and wait time has proven to be worthwhile. Setting all of the sendmails across all of the spawn-managed queue areas to use the same host-status caching has given us even greater efficiency.

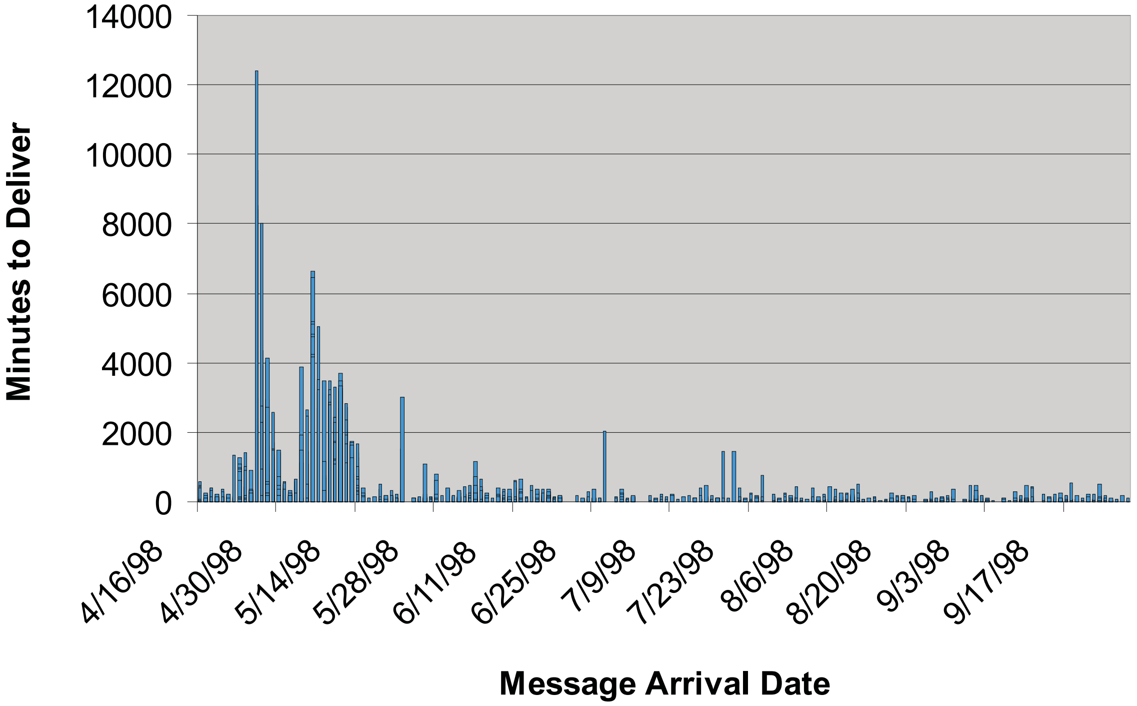

Figure 2: Message turnaround time.

Less than half a day after implementing this new queue processing method, only 7,000 recipients out of the initial 300,000 were still in the queue. All the recipients remaining in the queue were for "problem" addresses. At some point in the future, we may implement time-based queues as a subset of the spawn-managed queues.

Design Trade-offs

The astute reader will notice that it requires two passes of qsplit for a message to go from its initial bloated qf file in a holding directory to a chunked qf file in a live sendmail directory, awaiting delivery. This means that there is guaranteed to be at least five minutes, possibly as long as 10 minutes, between the generation of a list message by Majordomo and the earliest possible outbound opportunity for the message.

This is quite deliberate, for two reasons. The first, as you might guess, is simplicity of coding. Given that we were under a deadline to announce the list changeover, and that this level of rearchitecture had been slated for several weeks down the road, it was an expedient choice.

The second reason is directly functional for list purposes. Historically the Firewalls list has been plagued by flame wars of varying length and duration. We had been told that introducing a slight delay into message propagation, within reason, was the most expedient way to minimize the occurrence of flamefests. Exchanging one-liners over a few hours rather than a few minutes tends to spoil a bit of the fun and allow cooler heads to prevail. Given the requirement for slightly delayed propagation, we chose to retain our 5-10 minute granularity rather than try for a more immediate delivery.

List Management Issues

"So when I dropped it in the mailbox,
I sent it 'Special D'
Bright and early next morning
it came right back to me."
– E. Presley,
"Return to Sender"

The outbound mail was only the tip of the iceberg. We discovered that the Firewalls list generated an astounding 80M or more of bounce mail daily. Keeping bounce mail from overrunning list traffic had been the primary reason why Brent went to a dual-sendmail system for the list years ago.

The sendmail-lists was set up for outbound mail only, with a conventional sendmail receiving incoming list traffic and the plethora of bounces. Brent and his compatriot, Michael C. Berch, had put together a series of scripts for winnowing the postmaster wheat from the bouncing chaff. While the scripts identified many user queries and passed them on for human action, the bulk of the mail was discarded. This meant that secondary bounces would go uncorrected and trigger new recursive bounces.

Our plan from the beginning was to isolate bounce traffic even further, putting in a third separate sendmail structure solely for bounces. We would accomplish this by defining a virtual interface on the Internet-facing ethernet port and assigning it to "bounces.gnac.net". By tagging outgoing mail with From and Reply-To addresses at this host, we could control bounce traffic.

Automation scripts have been implemented in Perl, and have proven able to handle the formidable task of crunching through the huge volume of mail. We originally explored queuing the bounces via procmail as each message arrived, but quickly found that the overhead of calling procmail for each inbound message made batch processing of the bounce mail a better solution. The scripts, run out of cron, are explored below.

Automation of Bounce Handling

Since we had made the decision to automate wherever possible, we designed the script to identify and sort each message according to its potential for automation. Thus we arrived at three "bins" into which to toss processed bounce mail: "HUMAN," "AUTOMATABLE," and "JUNK."

The first category, JUNK, is for things which need throwing away. In particular, bounce messages to a bad address frequently generate second-generation bounce messages. We had planned on a scripting solution for these as well, but were pleased to note the "confDOUBLE_BOUNCE" option in the new sendmail 8.9 configuration. [15] Designed to catch just such occurrences, this option will let you specify an address, such as "| /dev/null," for these. To save on general I/O and processing wear and tear, however, it would be desirable to add a double-bounce ruleset and reject these messages right at the check_compat stage. [2]

The second category, AUTOMATABLE, is for messages which will eventually be handled by a programmatic response. A good example of this is the all-too-frequent "I unsubscribed but am still getting mail, help, get me off this list" query. A script will eventually be written which will pull out the sender's name and search for it in the database, then send off a canned reply describing message propagation and the results of the search.

Of particular interest in this category are routine bounce messages. We are in the process of adding a bounce manager which will extract addresses from a standard bounce message and process them. By hashing on the address and updating a counter, the script can quickly determine whether or not this is a repeat bounce offender. If so, the address can be automatically removed after a certain number of bounces. This represents a great improvement over the old ''bouncer'' list functionality which required hand editing of the list files to accomplish.

The last category, HUMAN, is for what is left. These are items that usually require human intervention, either to answer the question posed, or to figure how this particular item can be automated successfully as above.

Spam

''There is one thing you must be sure of,
I can't take any more!''
– Peter Gabriel,
''Shock the Monkey''

Spam was a serious problem on the Firewalls list. The machine that the list runs from does not relay spam, but spam is sent directly to the list. When we took on the list, this was one of the issues we intended to tackle. Based on anecdotal information, we did not expect traditional solutions to scale to cope with the Firewalls list. As we looked into our options, an increasing quantity of Firewalls list traffic became discussions on how to make the list usable again from the perspective of the subscriber community.

One conventional approach to cutting down on spam is to block certain domains or IP address ranges from successfully sending mail through your sendmail daemon. This can either be done through sendmail configuration, or at the network layer using Vixie's black-hole BGP feed, or simple filters. These approaches would not work in our case because much of the Firewalls mail was still being forwarded from greatcircle.com, and that mail was a mixture of real messages and spam.

The simplest-sounding solution was to make the list a closed list, where only members can post. However, there were two problems with this. Firstly, a large number of the lists subscribers were local exploders at remote sites, whose membership we had no way of knowing. We did not want to prevent these people from posting, or to force them to all subscribe directly to the list. Secondly, Brent had concerns about how well the majordomo feature to do this would scale to a list this size, which is the reason that he had never activated it on his version of the list. He felt that the interlocking programs that make up Majordomo's interpreted Perl core could not feasibly keep up with the traffic.

Sendmail Database Approach

We also considered taking this same idea, restricting posting to list members, to a lower level, and having sendmail do the work via a database lookup mechanism. The members of both the Firewalls and the Firewalls-digest list would be automatically added to the database. In addition, a list called Firewalls-post could be created for offsite list exploder members who wish to post. The Firewalls-post list is maintained by majordomo so that subscribe and unsubscribe requests can be handled automatically. All three lists would be made into a generic key/value sendmail database at regular intervals by a cron script.

We could trigger a database lookup only if the recipient was ''firewalls'' or ''firewalls-digest.'' Otherwise we would end up screening out routine majordomo requests or postmaster mail. By positioning the lookup in the check_compat phase of mail processing, we would be able to reject unauthorized postings directly at the SMTP connect. Note that system addresses such as ''firewalls-owner'' and ''majordomo'' need to be included to allow normal majordomo operation. These must be qualified with the full host and domain name in order to prevent spoofing, e.g., ''majordomo@lists.gnac.net'' rather than just ''majordomo.''

Using Majordomo

Clearly the Firewalls-post list that was suggested for the sendmail solution to the ''invisible subscribers'' problem could also be applied to majordomo. Just to make sure that we weren't re-implementing the wheel for no reason, we also ran some tests to evaluate the overhead of using the majordomo ''closed list'' feature.

After running a set of tests using the majordomo restricted-posting lookups, we found that on our machine it took about three seconds to perform the lookup. We decided to accept this as part of the system overhead, and implemented this feature over the sendmail based one. We have considered implementing the sendmail based variant of this on general principle, however, and to evaluate its use as a solution for other large lists not using majordomo.

The final component is the communication piece. We forewarned the list membership that we were going to implement this feature, and gave the message a couple of days to reach everyone.

In addition, when majordomo rejects a message due to this feature, directions on subscribing to the Firewalls-post list are returned to the sender.

Potential Future Problems

Strictly speaking, a clever spammer could handset the sender to be a legitimate sender such as ''majordomo@lists.gnac.net''. It would be wise for us to include a ''remote is identifying as me'' ruleset as part of this, so that this kind of spoofing would be caught and detected with prejudice. [14]

If spammers monitor the list and start spamming under spoofed names of legitimate posters, we would have to up the ante and turn on the sendmail features which do host authentication via DNS. [1, 15] While this would impose more of a load on the server, our split sendmail configurations would allow us to implement this on the main inbound sendmail only, so that the performance hit would not be too severe. We hope to avoid this, as many legitimate sites have business reasons to aggregate traffic or architect their mail infrastructure in such a way that they do not comply with strict sendmail checking.

Futures

"All the way to Malibu
from the Land of the Talking Drum:
Just look how far –
look how far we've come!"
– Don Henley,
"Building the Perfect Beast"

The cutover day for the Firewalls list move was April 15th, a red-letter day in its own right. Other than Brent's dual-sendmail structure, none of the facilities mentioned in this document existed on that date, nor had they been planned.

As of the writing of this paper, we are processing an average of 8184 messages per day. Turn around time for an individual message has dropped from pre-queue-split highs of 5-8 days to less than one day, and in many cases less than half an hour:

```
Statistics from Wed Apr 15 17:48:17 1998
 M msgsfr byt_from msgsto bytes_to Mailer
 0      0       0K 504929 2241822K  prog
 1      0       0K   3499   25520K  *file*
 3 402707 1831829K    270     914K  local
 4  51340  268777K   4584   59086K  smtp
 5  74148  421120K   2122    5626K  esmtp
 9     64     246K      0       0K  uucp-old
========================================
 T 528259 2521972K 515404 2332968K

date: Wed Jun 17 18:35:30 PDT 1998
```

In order to have better tracking of email flow through the list, we are intending to implement a script to take hourly snapshots of sendmail.st, process them, and feed the data to MRTG [16]. We have to do it that way since start/stop is impossible with so many sendmails all the time.

The script will need to aggregate the sendmail.st files of the variously spawned sendmails. They are separate files because sharing the same .st file could slow sendmail down unnecessarily as it waits on locking on the sendmail.st file.

When this is implemented, we intend to make the MRTG graphs available on the list website.

Further Processing

For lists with more "real-time" needs and less concern about flame wars, qsplit could be rewritten to deposit split files directly into processing queues at the time of splitting.

To improve message flow further, qsplit and spawner could be applied recursively to create time-based queues working with the existing spawn-managed queue directories. Messages over a certain age would be moved to a time-based queue and then split to a smaller $CHUNKSIZE. By employing progressively smaller chunks, one could force the qf files down to one RCPT per message by the time they reached a particular age.

At this point, problem addresses would be identifiable automatically by their queue position. This could enable management of bad addresses completely outside of the traditional bounce/postmaster processing used by most list admins.

List Exploders

We'd like to eventually add some special-case handling for exploder lists. When we receive a generic individual user bounce via a remote exploder, it is very difficult to find the origin exploder and pass on the error to that list administrator. In fact, there is no distinction made in the Firewalls or Firewalls-Digest lists between individuals and exploders, so there are undoubtedly many exploders which are completely opaque to us.

One approach would be to increase dramatically the number of spawn-managed processing queues and set qsplit to always chunk RCPTs to one per message. We would further modify qsplit to add an RFC-822 compliant custom header [11] containing the envelope recipient to each qf file. This header line would be preserved in any remote mailer bounces, enabling us to see to which address the original message was sent. A bounce message whose "user not found" error did not match the address in the X-Custom-Recipient header could trigger a custom message to the address in the header, or be referred to a human administrator for hand-processing.

Availability

The scripts described in this paper will be made available at http://www.lists.gnac.net after the publication of this paper in December, 1998.

Conclusions

"Don't know much about history . . ."
– Sam Cooke

Look before you leap!

Taking over the management of the Firewalls list seemed like an attractive proposition. It should be easy – just copy over Brent's setup on to faster equipment with better Internet connectivity and you're done. As we soon found out, it was not that simple.

Follow First Principles

There was no magic involved in turning the list into something that now runs smoothly. We stepped through a number of system administration basics. When the machine was in trouble, we looked at the hardware to see if more memory would help, and we looked at the kernel parameters and tuned them appropriately. After that, understanding the problems in detail and how the various solutions would affect the system allowed us to choose the correct course of action. Questioning our assumptions and gathering real data on which to base our decisions also proved worthwhile. Experimenting with different options off-line to get real data without affecting the list is always the right approach for a production system.

Work smarter not harder

The machine was not CPU-bound, so throwing higher-end equipment at it would not help. It was also not even coming close to saturating our connectivity. What we needed to do was find a way to make the system work smarter. To that end, having understood the problems we looked at ways to split the recipient lists into smaller chunks, and at how to get multiple sendmail processes to constantly churn through the queues.

Communicate, communicate, communicate.

An important part of our work during the ''hard times'' when we had just taken over the list was to communicate with the readership and let everyone know what was going on, and that we were working on fixing each of the problems that arose. There were many people interested in helping out, and we got many interesting pointers from folks on the list (thanks folks!). Letting people know the list of problems that you are working on, and when you realistically expect to have them fixed is something we all need to remember to do. The implementor feels less pressured and the ''customer'' feels plugged in and listened to.

References

[0] D. Brent Chapman, http://www.greatcircle.com/lists/firewalls (and posted to the Firewalls list).

[1] ''Majordomo: How I Manage 17 Mailing Lists Without Answering '-request' Mail,'' D. Brent Chapman, USENIX, *LISA VI Proceedings*, October 1992. ISBN 1-880446-47-2.

[2] *Sendmail*, 2nd Ed., Brian Costales with Eric Allman, O'Reilly and Associates, 1997. ISBN 1-56592-222-0

[3] *Mhonarc, a Perl successor to mail2html*, http://www.oac.uci.edu/indiv/ehood/mhonarc.doc.html .

[4] Our ''test mode'' consisted of two parts. First, a parallel feed of Firewalls traffic provided by Brent. Second was merely the sending of a real message to the list recipients as part of a dry run. The message would be a precursor to the official announcements already drafted by Brent Chapman (Great Circle) and Christine Hogan (GNAC).

[5] *Westworld*, Metro-Goldwyn-Mayer, http://us.imdb.com/Title?Westworld+(1973).

[6] *System Performance Tuning*, Mike Loukides, O'Reilly and Associates, 1992. ISBN 0-937175-60-9

[7] *Sun Performance and Tuning*, Adrian Cockcroft, Sun Microsystems Inc., 1995. ISBN 0-13-149642-5

[8] *Managing Internet Information Services*, Cricket Liu, Jerry Peek, Russ Jones, Bryan Buus and Adrian Nye, O'Reilly and Associates, 1994. ISBN 1-56592-062-7

[9] Strata Rose, VirtualNet Consulting, Dave Ilstrup, WebAware; unpublished work 1995.

[10] *Debian bulkmail*, http://molec2.dfis.ull.es/debian/Packages/stable/mail/bulkmail.html

[11] *RFC-822: Standard for the Format of ARPA Internet Text Messages*, D. Crocker, August 13 1982.

[12] ''Tuning Sendmail for Large Mailing Lists,'' Rob Kolstad, USENIX, *LISA XI Proceedings*, October 1997. ISBN 1-880446-90-1.

[13] *Multiple Independently Targetable Re-entry Vehicle*, http://www.janes.com/defence/resources/glossary/defres_glosmi-ml.html.

[14] *Sendmail: Theory and Practice*, Frederick M. Avolio and Paul A. Vixie, Digital Press / Butterworth-Heinemann, 1995. ISBN 1-55558-127-7

[15] Eric Allman and Sendmail Inc staff, http://www.sendmail.org/ web site.

[16] *MRTG, (Multi Router Traffic Grapher)*, Tobias Oetiker and David Rand, http://ee-staff.ethz.ch/˜oetiker/webtools/mrtg/mrtg.html.