High Capacity Email

Horms (Simon Horman)
horms@zipworld.com.au.
Senior Technician
Zip World

June 1999

http://linux.zipworld.com.au./

"But Mr. Anderson, what good is a phone call if you can't speak."
–*Agent Smith, The Matrix*

Acknowledgements

As always I would like to acknowledge the generous support of my employer Zip World without whose help this paper would not have been possible. In particular, my colleague John Ferlito has been tirelessly patient.

Version Information

Version	Date	Description
1.0.0	20th, June 1999	Initial Release
1.0.1	22nd, June 1999	Minor Corrections
1.0.2	25th, June 1999	Minor Corrections

Presented

Conference of Australian Linux Users (CALU)
9th-11th July 1999
Monash University
Melbourne
Victoria
Australia

Linux World Expo
9th-12th August 1999
San Jose Convention Center
San Jose
California
United States of America

Abstract

Email is one of the most well established and widely used services available on the Internet. With the popularity of Linux in the server market, especially amongst Internet Service Providers it should be no surprise that Linux is the platform of choice for many mail servers. The current growth of the Internet places increasing demands on both the hardware and software to deliver more mail to more users more quickly than ever before.

There comes a point where a single server can no longer cope. This is particularly a problem as the PC hardware market, which provides affordable hardware for applications such as this, is driven by increasing processor speeds and not the increases in I/O that are required to shuffle increasing volumes of mail.

One solution to this problem is to implement multiple email servers or a server farm to deal with the load. Messages are typically independent of each other and hence, parallelisation is a natural step to take. Implementing this is a manner that is transparent to users is non-trivial, but can can yield significant benefits, providing both increased capacity and high availability.

Contents

1 Introduction

Email continues to be one of the most popular services used on the Internet. UUNET reports that "Email comprises the bulk of network traffic and has increased an average of 20 percent a year" [Sen98]. Email using the Simple Mail Transfer Protocol, SMTP for incoming mail and Post-Office Protocol 3, POP3 or Internet Messaging Access Protocol, IMAP for mail retrieval is inherently I/O bound, as very little processing is done to messages. Typically the role of a mail server is to take a message off the network and write it to disk, later reversing the process. This places great contention on the file systems where messages are stored and relief from a PC hardware industry driven by CPU intensive applications is scarce.

There comes a point for service providers where, for price and performance reasons, upgrading the mail server where users' mail is stored is no longer practical and multiple mail servers need to be deployed. This poses several interesting problems in providing a homogeneous view of the mail service to end users across multiple chassis.

This paper will examine building a multi-host mail system to store email for users primarily in a single domain. This is opposed to a high capacity mail system designed to distribute messages to a large number of users in a mailing list environment, an entirely different problem. The solution developed will be designed to utilise existing technologies and protocols to provide a simple, scalable solution. In particular this solution will *not* examine clustering as this brings a degree of complexity beyond the scope of this paper. The fundamental philosophy behind this solution is to multiplex incoming email messages and requests from users to retrieve email across multiple servers.

2 Multiplexing SMTP

Simple Mail Transfer Protocol traffic can be split across multiple machines by assigning multiple IP addresses to the primary Mail Exchanger for a domain. To illustrate the way this will work an understanding of the semantics of Mail Exchanger DNS records is required.

2.1 Mail Exchanger Records

Mail Exchanger or MX records are part of the DNS information for a domain. An MX record has a preference. Preferences are an unsigned 16bit number and hence, range from 0 to 65535. MX records with a lower preference are preferred. MX records with equal preference are valid and are treated equally. As an example the MX records for the domain bigisp.com.au. are;

```
IN  MX  10  mail.bigisp.com.au.
IN  MX  20  mail2.bigisp.com.au.
```

When a foreign mail server wants to send a message to an address at bigisp.com.au. the foreign server will first attempt to deliver the message to mail.bigisp.com.au. If this fails then the foreign mail server will attempt to deliver the message to mail2.bigisp.com.au. If this fails then the foreign mail server should wait and then retry the process starting with mail.bigisp.com.au. If the mail is successfully delivered to mail2.bigisp.com.au. then it is up to that server to deliver the mail but typically it will deliver the message to mail.bigisp.com.au. In this way mail for a domain passes through the lowest preference Mail Exchanger.

2.2 Splitting Incoming Email

As will be discussed in later sections, once email arrives on one of the lowest preference Mail Exchangers for a domain it may need to be relayed to another server. Various multiplexing protocols are possible, but regardless, the mechanism for determining the end destination for an email is the same.

2.3 Rule Based Multiplexing

It was initially thought that the best way to split email would be to make an arbitrary division in the address space for users and route mail based on this. For instance, if a system is set up where mail is split over two servers it could be determined that addresses beginning with the letters a–k have mail delivered to one server and mail for all other addresses is delivered to the other server. A is be programmed into the MTA to efficiently arbitrate mail between the two servers. Under sendmail[1] a simple rule in the sendmail configuration file as follows would split mail as described.

```
Kalf -s1 regex -a<@alf.bigisp.com.au.> (^[a-k][^@]*)<@bigisp.com.au.>
Kbarney -s1 regex -a<@barney.bigisp.com.au.> (^[^a-k][^@]*)<@bigisp.com.au.>
```

These maps define how an address will be translated. The alf map looks for addresses that begin with a–k and are addressed to @bigisp.com.au. If this is found then then an address @alf.bigisp.com.au. is returned. The barney map looks for addressed that do not begin with a–k and are addressed to @bigisp.com.au. and returns and address @barney.bigisp.com.au. To use regex maps you need either sendmail 8.8 patched with map-regex or sendmail 8.9. In either case you need to compile with `DBMDEF= -DMAP_REGEX` set in the Makefile [Krü].

These maps should be placed under the definition of `Dn` in sendmail.cf or under `LOCAL_CONFIG` in sendmail.mc the m4 preprocessor is used to generate you sendmail.cf. Now place the following under ruleset 98 in sendmail.cf or `LOCAL_RULE_0` in sendmail.mc.

```
R$*                    $: $(alf $1 $)
R$*                    $: $(barney $1 $)
RERROR $*              $#error $: $1
```

The first line will apply the alf map which translates users beginning with the letters a–k. The second line translates all other users as defined by the barney map. The RERROR line causes the rules to abort it an error has been encountered. [All], [wEA97].

2.4 Per-User Map Multiplexing

The rule based solution is simple to implement, but lacks power. In particular, mailbox migration as discussed in section 9 is restricted, as the host that

[1]Sendmail is a trademark of Sendmail, Inc.

mail for a user resides is dictated by a simple rule. Making more complicated rules for this soon defeats the elegance of this solution as maintaining these rules would soon become a major chore.

A more flexible approach is to define a map that can redirect each individual's mail to the server that their mail is hosted on. If a user's mailbox needs to be moved to another server, then it is a simple matter of changing the entry for that user. The map should be in the form of a hash, to enable fast retrieval of data for an individual user. Conveniently, many MTAs including sendmail [All] and qmail using the fastforward add on [Nel] provide such a map in the form of an aliases file.

Aliases are typically used to assign additional addresses to a single mailbox. For instance, sales@bigisp.com.au. might be aliased to jill@bigisp.com.au. This allows flexibility in directing mail for sales, so if Jill decides that she is fed up with sales this alias could be changed. Aliases are also commonly used to distribute mail to multiple addresses. Majordomo, a list manager relies on aliases to distribute mail. For the application of multiplexing mail, aliases are used to alias an address to the same address on the mail server that the user's email is stored on.

For instance, if mail for John resides on the server ingrid.bigisp.com.au. and mail for Alison lived on fritz.bigisp.com.au. then the aliases file would be as follows;

```
john:    john@ingrid.bigisp.com.au.
alison:  alison@fritz.bigisp.com.au.
```

If sendmail is being used as the MTA then the aliases flat file is rebuilt into a hash using the command `sendmail -bi`.

2.5 Hybrid Multiplexing

The aliases mechanism provides great flexibility in assigning users to a mail server. However, it places a burden on administrators to ensure that *all* users have an alias. Users who receive mail and do not have a valid alias will —

depending on the mail server setup — have their mail delivered locally, on whichever mail server it arrives on or alternatively the mail will be rejected. This onus is particularly great when a new user is added to the system, in which case it is preferable for users' mail to work with minimal configuration.

A hybrid system that combines simple rules with aliases that can override the rules provides both simplicity and flexibility. New users to the system are covered by the general rules and users whose mailbox has been migrated for one reason or another are covered by their own alias. This can be achieved with the following;

```
Kalf -s1 regex -a<@alf.bigisp.com.au.> (^[a-k][^@]*)<@bigisp.com.au.>
Kbarney -s1 regex -a<@barney.bigisp.com.au.> (^[^a-k][^@]*)<@bigisp.com.au.>
Kuser_map hash /etc/mail/user_map

R$+ < @ $+ > $*      $: $(user_map $1 $: $1 < @ $2 > $3 $)
R$*                  $: $(alf $1 $)
R$*                  $: $(barney $1 $)
RERROR $*            $#error $: $1
```

The rules are as for rule based multiplexing with the addition of a a map to allow per-user server assignment. This is analogous to the aliases file but is a separaate map for greater flexibility. The flat file `user_map` is built into a hash using the command:

```
makemap -v hash per_user.db < user_map
```

The general rules will only take effect in the absence of a valid entry in user_map. Per-user server assignements are processed first and if the address to which the user_map entry points is not @bigisp.com.au. then the more general mapping will not take place.

3 Single Layer SMTP Multiplexing

Incoming email can be multiplexed, using multiple servers that will accept incoming mail from foreign hosts. Each server stores mail for a subset of users in the domain. To simplify configuration it is convenient to give each server a second IP address.

3.1 Topology

Each mail server that is to handle users' mail, should be configured with two IP addresses. This is most easily done using an IP alias which allows a second IP address to be assigned to an existing interface. One of the IP addresses assigned to a host should be allocated a name that is unique to that host. The other IP address assigned to the host should be assigned to mail.bigisp.com.au., DNS records for two servers would be;

```
barney  IN  A   61.8.0.100
mail    IN  A   61.8.0.101
alf     IN  A   61.8.0.102
mail    IN  A   61.8.0.103
```

Corresponding records for the reverse lookup of the IP addresses to these names are also required.

```
100     IN  PTR  barney.bigisp.com.au.
101     IN  PTR  mail.bigisp.com.au.
102     IN  PTR  alf.bigisp.com.au.
103     IN  PTR  mail.bigisp.com.au.
```

Recent DNS servers including BIND 8.x and BIND 4.9 will deterministically issue the different IP addresses assigned to mail.bigisp.com.au. in a round-robin fashion [LA98]. Hence, hosts that access mail.bigisp.com.au. have an equal chance of accessing the eth0:0 interface on barney or alf. If the connection is for a user whose mail is hosted on the machine that is accessed then the connection is dealt with locally, otherwise the connection is forwarded to the server where the users' mail is hosted.

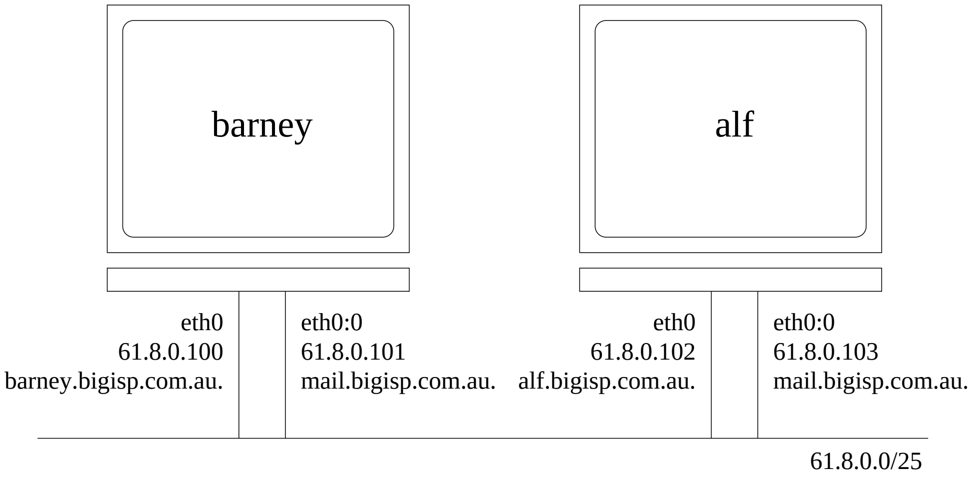

Figure 1: Servers with 2 IP addresses

3.2 Analysis

The deterministic round-robin behavior of the DNS resolution of mail.bigisp.com.au. has an equal chance of issuing each IP address. If there are n mail servers with A records for mail.bigisp.com.au the chance of one of the servers receiving a connection from a foreign host is $\frac{1}{n}$. Assuming that all users receive an equal number of messages of equal size the chance of a message from a foreign host being delivered locally is $\frac{1}{n}$. The number of messages that must be relayed to another one of the mail.bigisp.com.au machines is $1 - \frac{1}{n} = \frac{n-1}{n}$. So the probability that each server must relay is $\frac{1}{n} \times \frac{n-1}{n} = \frac{n-1}{n^2}$. As n becomes large this approximates to $\frac{n}{n} = 1$, as $\lim_{n\to\infty} n - 1 = n$. Effectively the probability of a message being relayed internally is 1 and as a result SMTP network traffic is doubled.

This analysis is still valid for a system where users have different usage patterns, provided that mailboxes are distributed across the servers in such a way that the servers are equally loaded. Distribution of mailboxes is discussed in Section 9.

4 Double Layer SMTP Multiplexing

Figure 2: Network Setup for Double Layer SMTP Multiplexing

Another approach to multiplexing incoming mail is to have two layers of servers, a front line of relays that accept connections from foreign mail servers and a back line of mail servers that actually house users' mail. The front line servers or *relay hosts* should be set up so that their IP addresses are A records for mail.bigisp.com.au. In addition an A record should assign a unique host name to each relay host so that specific servers can be accessed for administration purposes. The back end servers that hold users' mail should only be accessed by the relay hosts and can be given any host name as long as it is unique. In this example barney and alf are relay servers, while ingrid and fritz are the back end servers where mail is stored. The forward DNS records for this setup are as follows;

```
; Mail addresses to be accessed by foreign hosts
mail   IN  A  61.8.0.100
mail   IN  A  61.8.0.102
; Names for relay hosts for administrative access
barney IN  A 61.8.0.100
alf    IN  A 61.8.0.102
; Back end servers where mail is stored
ingrid IN  A 61.8.0.101
fritz  IN  A 61.8.0.103
```

The reverse DNS records should include mail.bigisp.com.au. and not the administrative names for the relay hosts. A reverse lookup on an address given for mail.bigisp.com.au should match the forward delegation. It is important to keep a consistent view of mail.bigisp.com.au. as some mail servers are very pedantic about matching DNS records for mail servers in an attempt to filter out unsolicited email. The resulting reverse DNS records for 0.8.61.in-addr.arpa. are;

```
100  IN  PTR mail.bigisp.com.au.
101  IN  PTR ingrid.bigisp.com.au.
102  IN  PTR mail.bigisp.com.au.
103  IN  PTR fritz.bigisp.com.au.
```

4.1 Analysis

Every message must be relayed as it must first be accepted from the sending host by one of the relay hosts and then relayed to one of the back-end servers.

This results in the same traffic as for the single layer model as discussed in section 3.2.

The advantage of this system is that only the relay hosts need to be externally visible. It is quite possible to place the back end servers on an internal network. The back-end servers could then placed behind packet filtering protection and even placed on one of private address space networks, 10.0.0.0/8, 172.16.0.0/12 or 192.168.0.0/16 as defined in RFC 1918 [RMK+96]. Only the relay hosts need to be exposed to traffic from foreign hosts and as a result the servers that are most vulnerable to attack now contain no user content, adding extra protection for end users.

There is no restriction on the number of relay hosts although it is envisaged that at least two would be sensible as a single relay server in front of several back end servers would create an unnecessary single point of failure. As the relay hosts do not hold any user data it would be possible to use techniques such as IP address takeover [Hor98] to switch in either a backup server or one of the other relay hosts if a relay host fails or is taken down for maintenance.

5 SMTP Multiplexing Performance

5.1 Test Environment

- **SMTP Server (kirsty):** Sun SPARCstation 5, 70MHz SPARC MB86904, 32MB RAM, 10Mb/s NIC running Linux kernel 2.2.9, glibc 2.0.7 and sendmail 8.9.3.
- **SMTP Server (voodoo):** Sun SPARCstation 5, 70MHz SPARC MB86904, 32MB RAM, 10Mb/s NIC running Linux kernel 2.2.9, glibc 2.0.7 and sendmail 8.9.3.
- **Client:** Dual 133MHz i586, 64MB RAM, 100Mb/s NIC running Linux kernel 2.2.5 with glibc 2.0.7.

  The Client ran a modified version of rlytest that allows a file to be sent as the contents of a mail to a list of up to two users. A parameter allows a time limit to be specified in which case the programme will loop sending messages over and over and report how many messages were sent once the time limit has expired.

Two test messages were used, of 843 bytes and 1047674 bytes (1MB) in delivered size on the SMTP server respectively. This were measured by sending the message to a user that required only one SMTP hop and recording the size of the message including headers. All machines were connected on a single 10Mb/s ethernet segment for the tests. The servers were arranged in the single layer topology as detailed in section 3. The number of connections was measured by running the test for one minute and recording how many messages were sent by the client in this time. There was at least a one minute pause between each test to allow for any backlogged processing to take place. This was repeated three times for each test and the mean is reported. For each connection messages were sent sequentially alternating between addresses where more than one recipient was specified by the test. The message data transfered is calculated by multiplying the message size by the number off connections in one minute. This does not take into account any protocol overheads. 1kB is measured as 1024 bytes and 1MB is measured as 1024 kB = 1048576 bytes.

The following tests were conducted for both message sizes. Where only one SMTP server is used both SMTP servers, kirsty and voodoo where tested and the average taken. Local addresses were aliased to `/dev/null` to remove file system speed as an issue. Each of the tests were run with one and three simultaneous connections.

- **One user, one SMTP server**: A message that will be delivered locally is sent to a single SMTP server.
- **Two users, one SMTP server**: A message addressed to two different users that will be locally is sent a single SMTP server.
- **Two users, two SMTP servers**: A message addressed to two users each of whom has mail delivered locally is sent to two different SMTP servers.

5.2 Results

5.2.1 843 byte message

One user, one SMTP server

Client Processes	SMTP messages in one minute	Message Data Transfered
1	239	192 kB
3	249	200 kB

Two users, one SMTP server

Client Processes	SMTP messages in 1 minute	Message Data Transfered
1	237	190 kB
3	238	191 kB

Two users, two SMTP servers

Client Processes	SMTP messages in 1 minute	Message Data Transfered
1	316	254 kB
3	528	424 kB

5.2.2 1MB message

One user, one SMTP server

Client Processes	SMTP messages in 1 minute	Message Data Transfered
1	7	7 MB
3	11	11 MB

Two users, one SMTP server

Client Processes	SMTP messages in 1 minute	Message Data Transfered
1	7	7 MB
3	11	11 MB

Two users, two SMTP servers

Client Processes	SMTP messages in 1 minute	Message Data Transfered
1	12	12 MB
3	18	18 MB

5.3 Discussion

The results indicate very little difference between performance in serving one or two users with one or three simultaneous connections on a single box. Given the low power of the SMTP servers used in testing, the server being tested quickly became overloaded by the client continuously sending email over the 10Mb/s network. Tests using both the smaller and larger message size, indicate significant benefits in multiplexing mail between two servers with a peak of 528 messages per minute for the smaller message size compared to around 240 messages per minute for a single server system.

It is of note that the maximum amount of data transfered over the network was 18MB of email. 50% of this traffic was relayed to a second SMTP server, as half the traffic sent to each server during this test was addressed to a user whose mail is hosted on the other SMTP server. This gives a total of $18MB \times 1.5 = 24MB$ of data. Even taking into consideration protocol overhead this seems rather poor on a 10Mb/s network.

6 Multiplexing POP3

Multiplexing of mail retrieval, accessed using POP3 is done using perdition, a POP3 proxy written with this purpose in mind. Perdition allows users to connect to a content-free POP3 server, that will proxy a connection to their real POP3 server. This enables POP3 for a domain to be split across multiple real POP3 servers on a per-user basis. This is intended to partner the splitting of incoming email, as discussed in section 2. Perdition is freely available, from http://linux.zipworld.com.au/linux/perdition/ and is distributed under the GNU General Public License.

Perdition should be run on each server that users access to read their email via POP3. Typically, this would be the same servers that foreign hosts connect to when sending mail via SMTP. When a connection is made to perdition, it reads the USER and PASS commands and then refers to its pop map to find where the user's connection should be forwarded to. Perdition then makes a connection to the POP3 server, where the user's mail is housed. Perdition enters the USER and PASS commands to the foreign POP3 server. If authentication is successful then perdition pipes data between the client and the foreign server. If authentication fails then the foreign server connection is closed and the client connection is reset to the state it was in on initial connection. That is, new USER and PASS commands are expected.

6.1 Pop Map

The pop map is analogous to the aliases file and user_map used to multiplex incoming mail on a per-user basis. The pop map determines the POP3 server to which each user will be directed once they have connected to perdition. The format is;

```
<username>:<hostname|ip address>[:<port number|service name>]
```

```
E.g.
```

```
john:fritz.bigisp.com.au
alison:fritz.bigisp.com.au
```

To build the pop map into a binary format makegdbm, which is provided as part of perdition is used. To rebuild the pop map run:

```
makegdbm popmap.db < popmap
```

At this stage there is no support in perdition for more general rules as are used for the multiplexing of incoming mail in sections 2.3 and 2.5. Support for this is planned. In the mean time it is suggested that a pop map can be generated using a script that examines the aliases file or user_map for the MTA in use and the system password file.

6.2 Single Layer POP3 Multiplexing

When using single layer SMTP multiplexing as discussed in section 3, it makes sense to use the same topology for POP3 multiplexing. In this case, the real POP3 daemons will need to be run on a port other than 110 and perdition should be run on port 110. If the real pop server is run on port 7777, then a sample pop map would be as follows;

```
john:barney.bigisp.com.au:7777
alison:alf.bigisp.com.au:7777
```

This has the unfortunate side affect, that a TCP/IP connection will have to be set up between perdition and the real pop server even if they are running on the same host. The traffic analysis, for this setup is the same as for single layer SMTP multiplexing as discussed in section 3.2.

6.3 Double Layer POP3 Multiplexing

If a double layer topology is employed for multiplexing incoming mail as per section 4, then the same topology can be used to multiplex POP3 connections. Using this topology, perdition should run on the relay hosts and a real POP3 daemon should run on the back end servers. As perdition and the real POP3 daemon do not run on the same box, there is no contention for port 110, as there was for single layer POP3 multiplexing. This being so, a sample pop map follows;

```
john:fritz.bigisp.com.au
alison:ingrid.bigisp.com.au
```

The traffic analysis for this setup is the same as the SMTP counterpart, as discussed in section 4.1.

7 POP3 Multiplexing Performance

7.1 Test Environment

- **Perdition Proxy Server:** Sun SPARCstation 5, 70MHz SPARC MB86904, 64MB RAM, 10Mb/s NIC running Linux kernel 2.2.9 with glibc 2.0.7.
- **Client**: 75MHz i486DX4, 20MB RAM, 100Mb/s PCMCIA NIC running Linux kernel 2.0.36 with glibc 2.1.

  The client accessed the server by running the following;

  ```
  while [ 1 ]; do
    nc <hostname> 110 < input > /dev/null
  done
  ```

  Input was a file, of the following format;

  ```
  user <username>
  pass <password>
  list
  retr 1
  quit
  ```

  Where multiple connections were run simultaneously, different usernames were used to eliminate mailbox locking, as a potential point of contention on the end POP3 server
- **POP3 Server:** Dual 133MHz i586, 64MB RAM, 100Mb/s NIC running Linux kernel 2.2.5 with glibc 2.0.7.

  The POP3 server was running a hacked version of cucipop, that did not delete messages on retrieval. Essentially the LAST command was removed from this daemon. The POP3 server, is several times faster than both the Client and Perdition Proxy Server, so as the end POP3 server would not be a bottle-neck during these tests.

Two test messages were used, of 1227 bytes and 1047674 bytes (1MB) in size respectively. All machines were connected on a single 10Mb/s ethernet segment for the tests. The number of connections were measured by running the test for one minute and then counting the number of connections registered for the server by syslog on the localhost. The message data transfered, is calculated by multiplying the message size, by the number of connections in one minute. This does not take into account any protocol overheads. 1kB is measured as 1024 bytes and 1MB is measured as 1024 kB = 1048576 bytes.

7.2 Results

7.2.1 1227 byte message

Perdition

Client Processes	Connections in 1 minute	Message Data Transfered
1	201	240 kB
2	253	303 kB
3	294	353 kB

Direct Connection to POP3 server

Client Processes	Connections in 1 minute	Message Data Transfered
3	910	1090 kB

7.2.2 1MB message

Perdition

Client Processes	Connections in 1 minute	Message Data Transfered
1	21	21Mb
2	24	24Mb
3	25	25Mb

Direct Connection to POP3 server

Client Processes	Connections in 1 minute	Message Data Transfered
3	43	43Mb

7.3 Discussion

For the larger message size, the bandwidth on the ethernet connection becomes the limiting factor. All data has must hit the wire twice, as it is downloaded to to the proxy server and then sent to the client. For the test with three simultaneous connections this results in $2 \times 25MB = 50MB$ of data on a 10Mb/s segment in one minute. Given protocol overhead this is a good result.

During the tests involving smaller messages, the limiting factor was the connection establishment time. For each connection, the client connects to the proxy server and the proxy server connects to the POP3 server. The dominance of this in timing, is shown by the 46% increase in the number of connections in one minute, when three connections are made simultaneously as opposed to one connection at a time.

These results show that perdition does suffer from latency problems, inherent in any proxying system where an extra layer is added between clients and servers. However, for larger messages this becomes insignificant and the limitation is that all data has to be both received and resent by the server, requiring twice as much bandwidth.

8 Multiplexing Other Protocols

Multiplexing of incoming mail has is covered by multiplexing SMTP, as this is the only protocol commonly used to distribute email on the internet. If other protocols were to be used for mail delivery, then these could be passed through an SMTP gateway in any case.

Multiplexing of mail retrieval is more complex. In my experience, POP3 is overwhelmingly the most popular method of mail retrieval. Other methods that are commonly used are IMAP and direct access to a mailbox using shell access. In the case of IMAP, as this is a TCP/IP connection oriented protocol it should be possible to make an IMAP aware version of perdition. Alternatively, depending on the user base it is may be satisfactory to have all IMAP customers connect to a single server. This can be done by assigning a CNAME to this server, as imap.bigisp.com.au. Users who wish to collect their mail using IMAP should set their IMAP server to this host. If the single layer topology discussed in sections 3 is used and 6.2 and alf is our designated IMAP server then the corresponding DNS record is;

```
imap IN CNAME alf
```

All users that wish to use IMAP, will need to have an alias or user_map entry directing their mail to alf.bigisp.com.au. In addition, these users will need to be exempt from any migration process. This does mean that all IMAP users need to be registered as such and does restrict mailbox migration, but if only a small proportion of users use IMAP, then this is satisfactory.

Similarly, users who wish to access their mail using shell access should be assigned to a single server and given shell access to the spool directory, possibly via a network file system accordingly. Another approach to offering shell access is to develop a method of transparently transferring mail mail for shell users into a local mailbox.

9 Mailbox Migration

It is important that all mail servers are under a load proportional to their relative power. If one server is under a great deal of load then some of this load should be migrated to other servers. To do this, a measure of the usage of each mail server and the usage of each user of the mail server needs to be made. A simple way to archive this is to measure usage from logs recorded by the MTA, POP3 and where applicable IMAP daemons. In this example sendmail and cucipop logs are examined to generate a metric of mail usage.

9.1 Measuring Mail Server Utilisation

For the sake of simplicity, it is assumed that all servers in the system are equally powerful. If this is not the case then weightings can be used, such that servers are allocated a load proportional to their power. Periodically a script can be run over the sendmail and cucipop logs to generate statistics. The following factors may be taken into account and weighted according to their resource consumption.

- Number of message delivered via SMTP, as per sendmail logs. This is intended to take into account the system resource required to accept an SMTP message, regardless of its size. This resource includes accepting a connection from a remote host and anti-spam and anti-relaying checks that the MTA runs through for each message.
- Size of incoming SMTP mail, as per sendmail logs. This takes into account the additional system resources required to handle larger messages.
- Number of POP3 requests, as per cucipop logs. This takes into account the resource required to handle a POP3 request, regardless of the amount of data transfered and the size of the mailbox accessed. In particular, this takes into account overhead in authenticating a user.
- Mailbox size, as per cucipop logs. This takes into account the resource required to scan through a user's mailbox. This is particularly costly under some POP3 daemons, including cucipop.

This does not take into account the data transfered to the client via POP3, as this should correlate quite closely to the size of incoming SMTP messages and hence, has already been accounted for in the metric.

9.2 Selecting Mailboxes for Migration

A metric is calculated for each user and a total is generated for each server. It is advisable, to measure this metric over a reasonable period of time, such as a week as individual's mail usage varies from day to day and in particular, usage on weekends is quite different to that on weekdays. It is quite undesirable, to be in a situation where mail servers do mass migrations, when the weekend begins and ends, as the overhead involved in migration is non-trivial. Once the metric has been measured, mailboxes need to be migrated to or from the server, if the total for the server is below or above the mean server total by more than a tolerance. The tolerance is designed to avoid migration as the result of fluctuations in usage by users. A tolerance of 5% is a good starting point, though may prove a little generous on a well tuned system.

When performing migrations, it is preferable to perform as few migrations as possible, to minimise changes that need to be propagated throughout the system. For this reason, it makes sense to move users who have a large metric, as this has the same effect as moving a larger number of users with a smaller metric.

Servers that are overloaded and hence, require users to be migrated to another server, arrange all users in order of their metric into a list. Users whose removal from the server would cause the server's total metric to fall below the mean are omitted from the list. For each user in the list, a decayed metric is calculated as follows;

Let m be the user's metric.
Let N be the number of days that a migration is remembered for.
Let n be the number of days since the user's mailbox was migrated.
Let m_d be the user's decayed metric.
Let d be the rate of decay.

$$m_d = \begin{cases} m & \text{if } \exists n \text{ and } n < N \\ m \times n \times d & \text{otherwise} \end{cases}$$

If a $N = 10$, that is migrations are remembered for 10 days and the user's mailbox was migrated yesterday so $n = 1$. If $d = 0.1$ then $m_d = m \times 1 \times 0.1 = 0.1n$. The decayed metric is used to determine the probability of a mailbox being migrated. The reason for this, is to avoid a situation where the same mailbox is migrated over and over again, as may occur is a mailbox has a particularly large metric.

The ordered lists from each overloaded server are then merged and the most underloaded server then truncates the list, such that only users whose metric would not cause the server to become overloaded are included in the list.

This leaves a list of users who can be migrated to the underloaded server in question without overloading the server and can be removed from the server that currently handles their mail without causing that server to become underloaded. If this list is empty then no migration takes place, otherwise a user is chosen using the decayed metric as follows;

Imagine a dart board divided up into areas for each user. This is now modified, such that the area for each user corresponds to its decayed metric, plus a small amount to ensure that all users have some area on the dart board. A dart is thrown in such a way that it has equal probability of hitting any part of the dart board, which though, practically impossible, will do for this analogy. The user that the dart lands on is our selected user.

9.3 Relocating a Mailbox

Once the user has been selected, the mailbox is migrated. Migration of a user's mailbox can be achieved by the following steps;

1. Lock the user's mailbox, on the server to which the mailbox will be migrated to. This is to ensure that there is no contention on write access during migration. If the user's mailbox is MBOX and sendmail is used as the MTA, this can be achieved by creating a file MBOX.lock with the same user, group and mode as MBOX.

2. If the user has an existing alias or user_map entry then remove it. Enter an alias or update the user_map for the user in question and propagate this throughout the mail system. All servers should now send mail for the user to the server to which the mailbox is being migrated to. Mail will be queued up by this server, while the user's mailbox remains locked.

3. Update the pop map to reflect the new server for the user and propagate this throughout the system. The user will now be collecting mail from the new server, though there is no mail there yet. Note that users who indulge in the irritating habit of leaving mail on the server will more than likely download fresh copies of all messages, once their mailbox

has been transfered. In my opinion these users deserve whatever they get.

4. Lock the user's mailbox, on the mail server that it is being migrated from. If a message is currently being delivered to the user, then this will have to complete, before the lock is obtained. In the case of MBOX.lock semantics, this involves waiting for MBOX.lock to be removed. This shouldn't be too much of an issue as no new messages will be delivered to this mailbox, as a result of the alias or user_map changes that are now in effect.

5. Transfer the user's mailbox, to the server that the user's mail is being migrated too. This can conveniently be done by using scp, the secure copy programme that is part of the Secure Shell or ssh package. Alternatively, the mailbox could be transfered using SMTP, however, this will add additional headers to the mail and compensation for such transfers will need to be made when calculating server usage.

6. Remove the mailbox on the server, that the mailbox is being migrated from and relinquish the lock on this mailbox. No more mail should be delivered on this server, for this user.

7. Relinquish the lock on the user's mailbox, on the server that mail is being migrated to. Mail for the user should now be delivered, on this server.

Once the mailbox has been relocated, the user should be added to a file listing users who were migrated, for use in calculating the decayed metric on subsequent days. The user should then be removed from the list for the server that the mailbox was migrated from and the total metric for this server should be updated accordingly. The list for this server should now been truncated, so that only mailboxes whose removal would not cause the server's metric to fall below the mean metric for all servers are on the list. If this list is empty, then the server is no longer overloaded. The total metric for the server to which the mailbox was migrated to, is also recalculated. If this server is no longer underloaded, then it is not considered for any further migration. Each underloaded server is considered for one migration at a time, in a round-robin fashion, until there are either no more underloaded servers or no more mailboxes suitable for migration.

10 Conclusion

The methods of distributing mail between multiple servers presented here, represents a scalable solution for applications where hosting mail for a domain on a single server becomes impractical. The cost of this method is, typically mail delivery and retrieval must travel though one extra hop adding to latency and effectively doubling network utilisation for mail. The former is a problem inherent in any multi-step mechanism and the latter can be alleviated by running multiple physical interfaces on a machine and spreading traffic between the interfaces.

At this stage, the solution is somewhat incohesive and some work needs to be done, drawing the different aspects of this solution together. In particular, pulling together the pop map, aliases or the user_map and general rules, into a single resource would simplify management greatly. Migration is another area that requires significant work and a scalable way for the mail servers to communicate with each other needs to be developed. It would be advantageous, if the mail servers could constantly be updated each other on their current load and users, so that migrations could occur as bottle-necks manifested.

The architecture developed has placed emphasis on a single user's mail only ever being located on a single server. This alleviates contention between servers for locks and avoids having to query each server for knowledge of a mailbox, as all servers have access to information on where an mailbox is housed. It is of particular note that intelligent construction of pop maps and aliases or user_map entries could be designed to migrate users' mailboxes to one of the servers closest to the user. When used in conjunction with routing techniques, to force traffic for a particular port onto a specific host, the user could be forced to always access one of the closest servers. Hence, a mail system distributed amongst separate physical locations can easily be created, enhancing the quality of service to end users though faster, more reliable access.

A Terms and Software

- **cucipop**: POP3 daemon written by Stephen R. van den Berg..
- **DNS**: Domain Name System, the distributed database used to map host names to IP addresses on the Internet.
- **I/O**: Input/Output: The transfer of data.
- **majordomo**: Mailing list manager that allows for access controls, digests, archives and management via email.
- **MTA**: Mail Transfer Agent: Software responsible for transferring email to or from another server.
- **nc**: NetCat: A network that enables TCP and UDP connections to and from arbitary ports.
- **rlytest**: Programme designed for testing SMTP servers for relaying ability. It is written by Chip Rosenthal of Unicom Systems Development.
- **Secure Shell**: See ssh.
- **sendmail**: The most popular MTA in use on the Internet today. Sendmail is a trademark of Sendmail Inc.
- **ssh**: Secure Shell: a secure replacement for rsh that enables users to securely log into remote systems. Can also be used to securely transfer files and provide arbitrary encrypted sessions between hosts.
- **qmail**: MTA written with security and performance in mind. qmail is copyright Dan Bernstein.

References

[All] Eric Allman. *Sendmail(tm): Installation and Operation Guide.* Sendmail Inc., 8.127 edition. For Sendmail Version 8.9.

[Hor98] Simon Horman. Creating redundant linux servers. In *Proceedings of the 4th Annual Linux Expo*, 1998.

[Krü] Jan Krüger. Using regular expresions in sendmail.cf: map-reges-2.8. http://www.stud.uni-hannover.de/ jk/rekursiv/map-regex/.

[LA98] Cricket Liu and Paul Albitz. *DNS and BIND.* O'Reilly and Associates, 3 edition, September 1998.

[Nel] Russell Nelson. Qmail web site. http://www.qmail.org/.

[RMK+96] Y. Rekhter, B. Moskowitz, D. Karrenberg, G. J. de Groot, and E. Lear. Request for comments 1918: Address allocation for private internets, February 1996.

[Sen98] Inc. Sendmail. Uunet relies on sendmail(tm) for explosive growth. http://www.sendmail.com/products/uunet.html, 1998.

[wEA97] Brian Costales with Eric Allman. *Sendmail.* O'Reilly and Associates, 2 edition, January 1997.