Internet Service Provider Email Architectures

Martin Foster
martin.foster@pacnet.com

ABSTRACT

This report discusses Internet Service Provider (ISP) email system architectures. The goal is to frame these systems' history, markets, design constraints, and evolutionary challenges in order to understand the performance bottlenecks that are encountered and routed around as provider architectures scale.

The author has worked on large ISP email architectures with both Sprint Canada (now Rogers Communications) from 2000-2002 and Pacnet Internet in Australia (previously Pacific Internet) from 2003 to the present.

1. FRAMING

Internet Service Providers offer email as a product to a user base that has highly variable requirements. This paper is centric to the service provider market, herein always referred to as Internet Service Providers (ISPs). It is expected to be applicable to the plethora of service provider acronyms seen on the market today, such as Mail Service Providers (MSPs) and Application Service Providers (ASPs).

1.1 Non-ISP Email Providers

Here we wish to differentiate ISPs from other common providers of email services. These are not the focus of this paper as they usually have different business concerns, though they will use similar technological components to deliver email services.

Common non-ISP email providers are:

1. Institutional providers, usually subdivided into:
    a. Corporate or Enterprise providers. These are email services provided to employees by their employer.
    b. Academic providers. These are email services provided to academics: students, staff or faculty depending on the institution.

   Institutional providers differ by their funding model and use of policy. Providing email services to employees, students, or academic users is seen as a necessary cost center. Flexibility and features offered are chosen based on business need and limited by the cost to provide and maintain these features.

   The distinction of policy is important: institutional providers can enforce policies relating to usage and content in a uniform fashion. ISPs cannot generally enforce policy outside basic Terms of Service (TOS) clauses - the features described in the "Email as a Product" section must usually be made flexible and selectable to accommodate customer needs.
2. Webmail providers. These are email services provided by large internet-based companies at little or no cost to the user. Examples webmail providers are Google's gmail.com, and Microsoft's hotmail.com. They have a variety of features, but are usually differentiated from ISPs by their profit motive and support structure. Webmail providers usually rely on advertising revenue and not direct subscriber fees for income, hence have a tendency to encourage exposure to advertisement either on the web interface, or by adding a footer to sent or received messages. Webmail providers also tend not to providing telephone support, or service level guarantees as this requires a considerable investment in support infrastructure. That said, this component is in flux as webmail providers introduce enhanced products that make professional support and service guarantees available for a fee.

1.2 Email as a Product

Defined, email products usually involve a means of controlling an email address. This base unit is then enhanced with any or more of the options below that are directly tied to this address. These options recurring throughout this paper and are defined below.

1. Address ownership. Basic ISP mailboxes are usually bound to a domain owned by the provider, such as username@isp.com. These addresses are usually included with other services from the ISP, such as internet access. They are inexpensive to provide, and are a means of increasing customer retention by the fact that most people find it difficult to change email addresses. There is usually an option to host mail for a customer owned domains such as personname@company.com which tends to target companies or individuals with vanity domains.
2. A mailbox or mail store that is constrained to a certain storage quota. Customers usually have the option of purchasing more storage quota if required.
3. Access to the mailbox. This is a combination of a web interface (webmail), or email client that uses an open protocol such as POP3, IMAP4, or proprietary protocol such as MAPI. Email clients are also known as Mail User Agents or MUA. Some protocols and their underlying functionality are more expensive to provide or manage than other, and hence may carry additional cost.
4. Mail forwarding and rewriting. Not all addresses are directly tied to a mailbox. Often customers have many email addresses or domains accumulated over time that they wish to backhaul to a limited set of mailboxes. Email systems can be set to forward messages from one address to another, and optionally rewrite the message's sender or receiver address in the process to meet some endpoint need. Mail forwarding is a feature that many mechanisms that combat Unsolicited Commercial Email (UCE), herein spam, have a tendency to with suspicion unless some rewriting of the sender is done; the reasons for this will be elaborated in the spam segment of this paper.

Technical Report. 12 November 2008 Melbourne Victoria Australia

5. Content filtering. In the form of Anti-Spam and Anti-Virus filtering, which can also extend to some forms of more stringent policy that look at other attributes of the message, such as attachments or undesirable wording. A basic form of content filtering is usually included with email products. Chargeable additions are the use of more accurate filtering software, greater control of the filtering & message handling policy for one mailbox, and enterprise policy management for an entire customer-owned domain.
6. Archiving services. Almost always a chargeable feature, archiving services allow all of a customer's messages to or from the service provider to be archived in an immutable store separate to the user's regular mailbox. This service is targeted at customers with the need to retain messages for a given period of time due to a customer data retention policy or regulatory requirement.
7. Address book and calendaring services. These are not strictly an email product, though most address book and calendaring systems available today tie into email clients. These can range from simple systems provided with a web interface, to complex dedicated offerings such as Microsoft's Exchange product.

1.3 Email Product Manageability

With common email features itemized, the author wishes to remind the reader of a key tenet of operating a viable business: always remember that it is income that allows the company to operate. Do not give away services free of charge, and be particularly careful with how additional features are billed. Once customers begin using a service or feature, it is difficult to cease providing it without losing a portion of the customer base.

The author has worked on integrating and migrating many email platforms as a result of growth by acquisition, where the acquired company offered many of the features above without consideration to their non-zero cost to initially provide, and support in an ongoing fashion.

Beyond offering services to customers, ISP email systems should preferably be understood and manageable by a wide range of the provider's staff. This is not trivial. Beyond the concept of email addresses and that mail is stored somewhere, it is easy to misunderstand how email traverses the internet, how it's flow can be interfered with or delayed, and the many factors that determine a mail system's availability.

The recommendation is to provide ISP employees with simple service inspection, troubleshooting, and provisioning tools. A constant challenge that is easier said than done.

1.4 Service Provisioning Challenge

Service provisioning is controlling what email services are offered to customers, and how these services should behave.

Provisioning and de-provisioning services is often overlooked, yet the failure to design adequate provisioning tools to work with an email system can lead companies to any of the following business impediments:

1. Delays in providing a service. As it is generally the norm in the ISP market, customer come to expect a very short delay in providing a service once it is purchased. Automated provisioning systems should orchestrate the changes to the many parts of an email platform required to deliver the requested service.
2. Overreliance on manual provisioning, which is usually related to the inability to de-provision services once they are no longer required by customers. Provisioning systems help to ensure that the only revenue generating services are provided.

Elements affecting or related to service provisioning will be highlighted as specific technologies and architectural components are discussed.

2. A BRIEF HISTORY

Electronic messaging systems have been available almost as long as multi-user systems have existed [9]. Some knowledge of the origins of email systems is necessary to understand the resiliency, constraints and challenges to modern day ISP email architectures.

2.1 Early Systems

Multi-user systems are a prerequisite to needing a messaging application. These systems, such as MIT's Compatible Time-Sharing System provided inter-user messaging in 1965 [9].

The important leap from single system to messaging between interconnected systems arrived in 1971, using the ARPANET network with the first electronic messages transferred with a protocol that was not messaging specific: FTP.

Adoption of larger scale messaging applications, as with the networks on which these ran were largely restricted to proprietary vendor platforms in the 1970's. Digital Equipment's DECnet was made available in 1975 and included messaging functionality.

Messaging between heterogeneous systems was made possible with the Unix to Unix Copy Protocol (UUCP) in 1975. UUCP enabled store and forward messaging, the basis of modern email.

The author decommissioned the last UUCP delivered email service provided by his employer in 2005.

2.2 Core Email protocols: 1980's

The core protocols & technologies used by modern email systems were deployed 1980's, with design considerations discussed then [2]. Where applicable, short sample transactions are made with the given protocol to demonstrate the simplicity of the syntax.

2.2.1 Mail Transport: SMTP

Standards governing the exchange of email, RFC 821 [11] and the format of such messages, RFC 822 [12] were proposed in 1982. Made obsolete by RFC 2821 and 2822 respectively in 2001 the functionality of the original documents was not changed [14].

A simple SMTP transaction is shown below, note the difference between a message's envelope ("mail from", "rcpt to") and the stated sender, recipient and subject headers in the message body following the "data" instruction. The single period on an empty line indicates the end of a message (".").

```
$ telnet mailin1.pacific.net.au 25
Trying 61.8.0.80...
Connected to mailin1.pacific.net.au (61.8.0.80).
Escape character is '^]'.
220 mailin1.pacific.net.au ESMTP Gday mate
helo netlog.net
250 mailin1.pacific.net.au
mail from: <sender@netlog.net>
250 2.1.0 Ok
rcpt to: <martin_foster@pacific.net.au>
250 2.1.5 Ok
data
354 End data with <CR><LF>.<CR><LF>
Subject: This is my test message
From: "Some Sender" <fake@donottrust.net>
To: "Martin Foster" <martin_foster@pacific.net.au>

This is my message body.


.
250 2.0.0 Ok: queued as EDAAB5E4016
quit
221 2.0.0 Bye
```

Figure 1: Sending a message using SMTP

2.2.2 Network: TCP/IP

The largest network - ARPANET - switched from the Network Control Program (NCP) to TCP/IP on 1 January 1983 [10]. Email's addressing format of user@host or user@domain was first used on the ARPANET [9].

2.2.3 Mail Retrieval: POP, POP2, POP3

In 1984, the Post Office Protocol (POP) was proposed [16] to allow workstations to retrieve messages from mailboxes stored on remote systems. The important contribution of this protocol and others like it was standardizing the reading of remotely stored email, removing the need to make every system capable of reading mail a mail server. It was updated in 1985 [17] then 1996 [18] to version 3 that is prevalent today as POP3.

A sample POP3 transaction is shown below:

```
$ telnet mail.pacific.net.au 110
Trying 61.8.0.22...
Connected to mail.pacific.net.au (61.8.0.22).
Escape character is '^]'.
+OK POP3 Ready mailproxy1 0001c04e
user martin_foster@pacific.net.au
+OK USER martin_foster@pacific.net.au set, mate
pass **************
+OK You are so in
list
+OK 12411 messages:
1 21825
2 4849
3 4729
(full list of messages, by number and size in bytes)
12410 5560
12411 5096
.
top 12411 5096
+OK
Delivered-To: martin_foster@pacific.net.au
From: "Some Sender" <fake@donottrust.net>
To: "Martin Foster" <martin_foster@pacific.net.au>

This is my message body.
```

Figure 2: Retrieving a message using POP3

2.2.4 Name Resolution: DNS MX Records

In 1986, the Domain Name System (DNS) was updated to carry Mail Exchange (MX) records that would indicate the systems designated for receiving a domain's email [13]. This uncoupled SMTP based email from having to derive where to deliver messages based on the *syntax* of an address to *attributes* of a domain. Prior to this RFC, email addresses implicitly defined the delivery point - the host listed in the address. For username@machine.domain.com, a connection would be opened to machine.domain.com. In order to have a "clean" or easy to remember email address of username@domain.com, the machine specified as "domain.com" had to also be a mail router, an availability and throughput bottleneck. With this DNS MX records, an address of username@domain.com would trigger a lookup for MX records related to "domain.com", which must return at least one, but usually returns at least two preferentially weighted names of mail routers for the domain.

A sample MX lookup is shown below, with one of these addresses being used in the SMTP sample transaction. The even weight (20) of both systems indicates that there is no preference for recipient mail host out of the two, they should be chosen in a round-robin manner by the client or DNS resolver depending on the implementation.

```
$ dig +short mx pacific.net.au
20 mailin1.pacific.net.au.
20 mailin2.pacific.net.au.
```

Figure 3: Lookup of a DNS MX record

With the technology basically set in the 1980's, one asks why challenges remain in the field of ISP Email systems.

2.3 Spam or Paradise Lost

Greed and lack of accountability complicates everything.

2.3.1 Why Spam Exists

In the Internet's earlier days, systems were primarily connected for research and collaborative purposes. The first persons to use these networks the messaging services they provided for marketing or advertising purposes were ostracised. Yet as the network and its access points grew, and more users joined for commercial purposes, etiquette alone no longer restrained marketing and advertising activities.

It would be simplistic to say that the advertisers are evil. Neither legitimate advertisers or unscrupulous spam distributors would use email as a medium if it did not generate sales. The reason there is so much spam attempting to be delivered is that it is profitable to do so because someone buys the advertised product.

The challenge with email is that extremely low sales ratios (or *conversions*) are required in order to make a profit, in the order of 1x10-3 to 1x10-7 percent [19]. In other words, a spam sending network that distributes 10,000,000 messages to make a single sale still generates profits.

2.3.2 Spam: Countermeasures but no Silver Bullet

The profit motive explains why approximately 92% of messages that are delivered or attempted to be delivered at the author's site are classified as spam.

There is no silver bullet to eliminating Spam. From a social standpoint, users can be educated to resist falling for the sales pitches or attempts to steal their identities (phishing) that hit their inboxes on a daily basis, but at some point in time a new tactic will be used - or someone will simply desire the advertised product - and the cycle will proceed.

From the technical standpoint, a plethora of technologies exist to attempt to identify Spam. These fall into two main categories, reputation system and content filters. At the author's site, approximately 85% of messages identified as spam are blocked by reputation systems, and a further 7% by content filters.

2.3.2.1 Reputation Systems

Reputation systems, and Reverse Block Lists (RBLs). These attempt to gauge the likely hood that a given host is a spam source based on its recorded communication history and reject communication with it entirely [7]. Commonly used systems are Spamhaus lists (http://www.spamhaus.org), and the Mail Abuse Prevention System - MAPS (http://www.mail-abuse.com, now owned by Trend Micro). Reputation systems tend to operate with the DNS protocol, crafting a query to the reputation system provider that includes the connecting hosts' IP address. An answer indicates a "hit" or listing as a very likely spam source. The primary concern with RBLs is their binary nature, a source is either tagged as likely to be a spam source, or not. Concerns arise with sites being listed improperly - particularly with ISPs who by design service thousands or more customers, a certain percentage of which will constantly be sending some spam. A good reputation system provider understands the existence of this ratio and be tolerant of it as long as the spam flows are continuously addressed by the ISPs support staff.

Figure 4 shows a DNS based query to the Spamhaus Xen RBL for host IP 190.17.39.126. Note the reverse octet notation, similar to a reverse DNS (rDNS) lookup. A response of any address indicates a hit.

```
$ dig +short 126.39.17.190.zen.spamhaus.org
127.0.0.4
127.0.0.11
```

Figure 4: Looking up a host IP with a Reputation Service

2.3.2.2 Content Filters

Commonly known as Anti-Virus and Anti-Spam systems, or AVAS, content filters examine the message body to determine whether a message is likely to be infected by a virus or Spam.

This process is far more computationally expensive than the DNS lookup used by reputation systems, hence usually used as a secondary measure.

There are both commercial and open source content filtering offerings. SpamAssassin (http://www.spamassassin.org) and ClamAV (http://www.clamav.org) are the two most popular open source packages. All anti-spam packages that the author has been exposed to used some form of statistical analysis on a corpus of known spam and known not-spam ("ham") messages to identify parts of a message or behaviors that are likely to be spam. Weights are generally assigned to these behaviors, with a certain numerical sum or percentage identified as the threshold for identification as spam. It is common to use two thresholds, one for quarantining for possible future analysis, and a higher number for marking as definitively spam.

In an ISP environment, these packages are usually invoked by another software package that governs the filtering workflow - hopefully based on the policy the customer has selected. As mentioned earlier in this paper, the ISP market cannot make assumptions for their customers - some will want as aggressive filtering as possible, whereas an increasingly rare minority will want no filtering. Open Source software packages such as Amavisd-new and MailScanner can be used to accomplish this.

Figure 5 shows two log messages from a SpamAssassin anti-spam process run from Amavisd-new policy engine:

- the first has been marked as spam, with a score of 8 for a threshold of 5. As this is lower than the upper discard threshold of 10 by which the system is configured, the message will be quarantined.
- the second is thought to be not-spam, with a score of 1.312 for a threshold of 5.

```
SPAM, <nobody@host.doctorquek.com> ->
<obfuscatedaddress1@isp.com >, Yes, score=8.014 tag=0
tag2=5 kill=5 tests=[HTML_IMAGE_ONLY_12=2.245,
HTML_MESSAGE=0.001, MIME_HTML_ONLY=1.672,
RDNS_NONE=0.1, URIBL_BLACK=1.961,
URIBL_PH_SURBL=2.035], quarantine Q68CZdZ53LCX (spam-
quarantine)


SPAM-TAG, <obfuscated2@outsideisp.com> -> <
obfuscated3@isp.com>, No, score=1.312 tagged_above=0
required=5 tests=[HTML_MESSAGE=0.001,
MSGID_MULTIPLE_AT=1.211, RDNS_NONE=0.1]
```

Figure 5: Tagging use to determine if a message is spam

3. EMAIL ARCHITECTURE

This section covers basic mail flows within an ISP email system, and the components traversed by these flows.

3.1 Basic Flows

Putting together the previously discussed protocols, there are three primary mail flows seen within ISP mail system to consider.

3.1.1 Outside In

In this case, the ISP mail system in Figure 6 receives mail from the outside. The remote user in (1) sends a message to their mail remote mail server (2) which runs a DNS MX lookup on (3) to find a server for the ISPs domain, here mailserv.destination.com for the destination.com domain. The remote server then uses SMTP (4) to send the message to the ISP mail server (5).

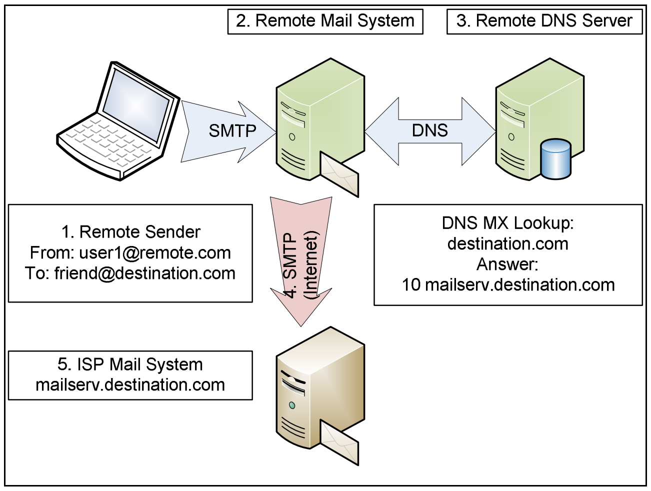


Figure 6: Mail from outside arriving at an ISP mail system

3.1.2 Inside Out

This scenario in Figure 7 is equivalent to the outside in example, except this time an ISP customer is mailing out to a remote domain.

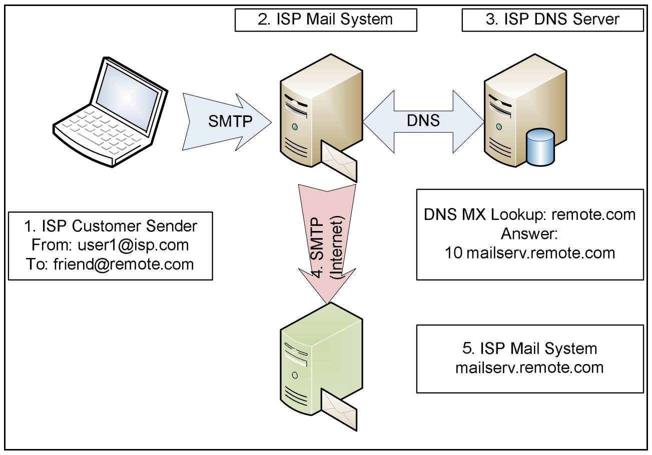


Figure 7: Mail originating from the ISP for a remote domain

3.1.3 Intra-ISP Delivery

This third case in figure 8 is rare in terms of mail volumes yet important in the design of mail routing databases. It occurs when one customer sends a message to another domain also hosted within the same ISP mail architecture. Prudence must be taken to ensure that the policies applied to the recipient aren't bypassed because the message originates from within the ISP.

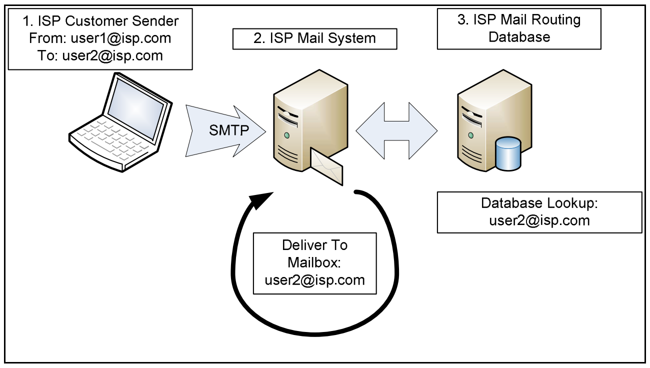


Figure 8: Intra-ISP Mail Routing

3.2 Components

Within the ISP mail system, many different daemons will be called within the mail delivery chain. These can be hosted on one or many systems depending on the architecture's expected load.

3.2.1 Mail Routing Databases

While figure 8 may appear to be a loop, it demonstrates that mail remains within the ISP's systems. This scenario introduces the concept of a mail routing database that is not DNS. DNS can only specify how to reach mail exchangers or specific hosts, not the hops to take within the mail system.

Mail routing databases can take many forms, but are predominantly flat files with soure:destination pairs, high performance key:value lookup databases such as Berkeley DB which have near O(1) access time, or external network databases such as LDAP directories or SQL servers.

3.2.2 Mail Transfer Agents

Daemons used to store and forward mail to an appropriate location. These make heavy use of the mail routing database in deciding whether to accept mail, and if accepted where to send it next. A key function of Mail Transfer Agents (MTAs) is to incorporate a queuing mechanism and store messages in a persistent manner until delivered.

Examples of open source MTAs are Sendmail, Postfix, Exim and Qmail. Commercial mail systems such as MDaemon, Microsoft Exchange and Lotus Notes all incorporate an MTA.

Specialist applications of MTAs in a mail flow:

1. Inbound message acceptance. To combat spam, it is usually the MTA on inbound mail exchange (MX) servers that consults previously discussed RBLs via a DNS lookup to determine the reputation of the sending mail system and hence whether to accept the connection.
2. Outbound source diversification. This technique implicitly acknowledges that ISP mail systems will send spam, and hence invariably have their own outbound IP addresses sporadically listed by RBLs. Here we use different batches of outbound addresses, served by different MTAs to cluster customers into categories that reduce the risk of having all outbound IP addresses listed in RBLs at once. A method that has worked well for the author is grouping customers both by spend and age with the ISP. On age, customers that have been

with the company longer are less likely to be deliberate spam sources. On value, customers that bring in more revenue are often offered higher grades of service - either explicitly stated in contracts or implicitly by their value to the ISP's bottom line.

3.2.3 Mail Delivery Agent

Mail Delivery Agents or MDAs are used to deliver messages to mailboxes. They are special purpose in the sense that they usually do not communicate via SMTP, but rather are invoked by a command or communicate via Local Mail Transfer Protocol (LMTP) - a subset of SMTP that implies that no queuing is done [20]. This provides the MTA with assurance that once a transaction is completed a message is place in the recipient mailbox.

MDAs tend to be incorporated with the mail store.

3.2.4 Mail Store: Mailbox Repository

Mail stores have many types of on disk format, from proprietary to open. There are generally two approaches to storing mail on disk:

1. One file per message, in one directory per mailbox. This places a lot of small files on the file system - a disadvantage if the file system places an upper limit on the number of files that can be stored. This has the advantage of quick retrieval with a good index and the ability to provide multi-system read if the file system is shared by multiple mail stores. Open formats such as Maildir++ write one file per message.
2. One big file. The first way of storing mail was in a big flat file, with messages concatenated by arrival time. Reads, writes and deletes were simple to implement but computationally expensive: files have to be split or recreated on any delete. The open "mbox" format does this. The alternative to writing a flat text file is to use a database binary, such as Berkeley DB to provide efficient read, write, update and delete performance. The caveat with using one big file is that multi-access is rarely supported - one must be careful with locking on shared file systems.

The mail store usually provides access mechanisms, typically the previously discussed POP3 or IMAP4 [21] protocols to be used for access to the mail store by email clients.

3.2.5 Mail Policy: Content Filtering

These systems were discussed in section 2.3.2.2 in the context of the discussion on spam and virus filtering.

3.2.6 Information Hiding: Proxies

Mail proxies can be deployed to provide a processing layer between customers accessing the ISP mail infrastructure, either for reading or sending message. This allows another layer of message rewriting to occur, often used when merging acquired mail systems in a means that is transparent to the users.

For mailstore access, proxies enable migrating the content of mailstores one mailbox at a time, then update the routing records used by the proxy to point at the new store. Network addresses remain the same, and the odds of a customer wanting to check a mailbox in the instant that it is being migrated tend to be low in systems service thousands of customers.

For mail delivery, proxies provide another queue that can be monitored for suspicious behaviour. Spam outbreaks originating from within an ISP can be observed by the volume of mail being submitted to the proxies by customers. Detection here allows ISPs to identify and act on the problem customer without interrupting other mail flows.

3.2.7 Provisioning

Returning to the service provisioning challenge mentioned at the beginning of the paper. In a well designed mail system, only three aspects of the system should need to be altered by a provisioning system to configure a customer's mail services:

1. The DNS MX records for the domain, to get mail to the ISP mail system from remote servers.
2. The Mail Routing Database, to define how mail is routed inside the ISP mail system
3. The Mail Store, to ensure that there is a physical location for mail delivery.

4. COPING WITH SCALE

ISP mail systems either grow organically as the number of customers or services provided increases, or by acquisition as one company buys out the customers of another and hopefully eventually seeks to merge the resultant infrastructure.

Growth is interesting to the practitioner; it exposes problems of scale, and often provides one with the opportunity to critique the architectural decisions that others made to the system beforehand.

We look at how the components specified in the last section are assembled to provide low and high volume mail architectures.

The reference figures is 9, which shows all previously discussed components and flows in an ISP mail architecture.

4.1 One System Monolith

Usually seen as the simple case, or the starting point. Single system mail architectures are usually deployed in support of some other flagship product. It is estimated that this sort of system would be processing less than 10,000 messages per day.

In this deployment, all necessary daemons run on one machine, with separate inbound and outbound MTA processes, and proxy processes the most likely to be omitted.

This can simplify provisioning by having a single point to alter. Single system deployments like this will often rely on flat files or simple databases such as Berkeley DB.

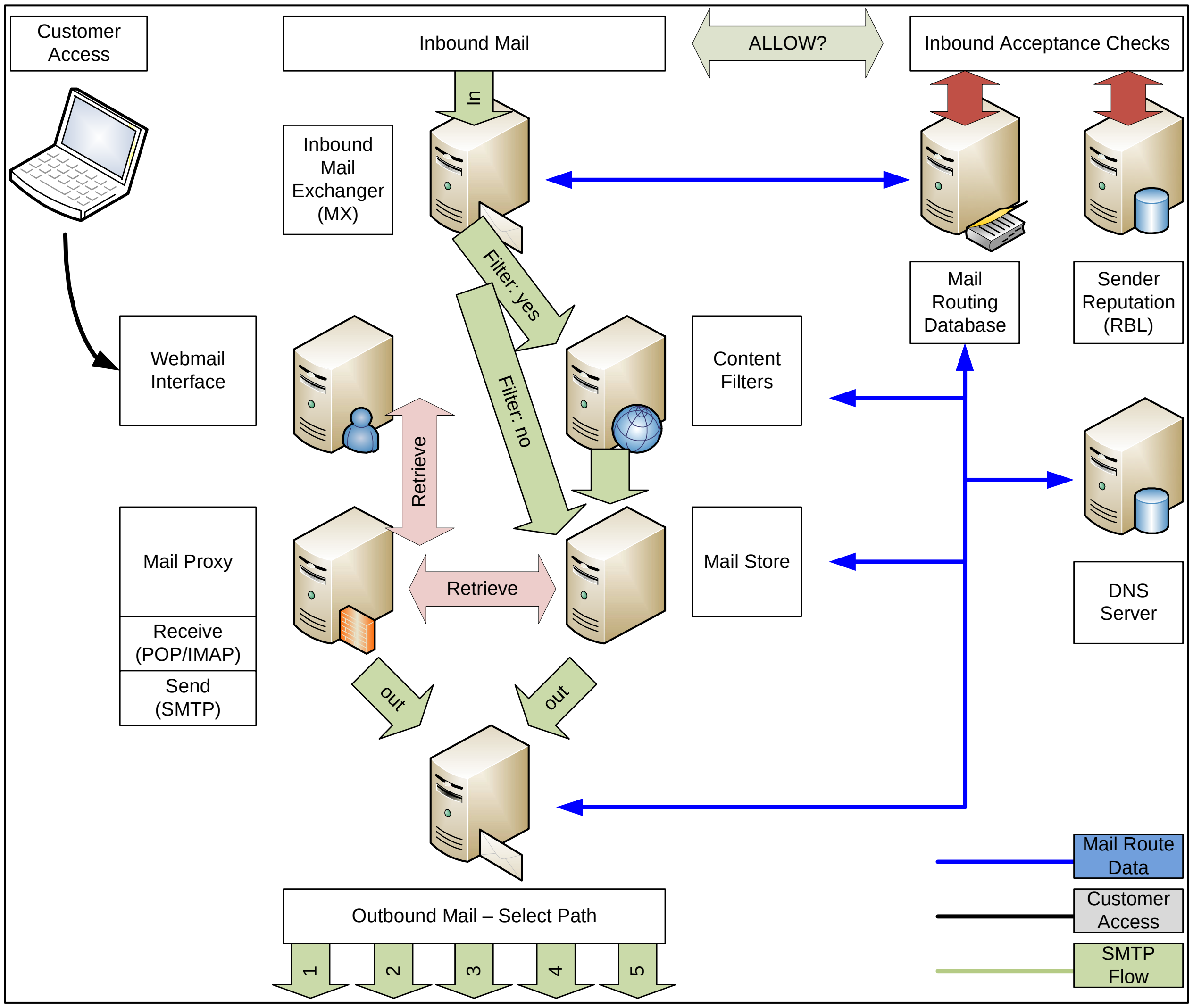


Figure 9: All ISP Mail Components and Flows

4.2 One to a Few

Availability will be the first casualty of a single system deployment. This configuration would be adequate for processing 10,000 to 1,000,000 messages per day.

Availability to the outside world will be the first concern, making it sensible to ensure that an inbound mail exchanger is always available. The mail store will most probably still be a single system.

The simplest growth path is another MTA configured for secondary MX service. These are characterized by having a higher (less preferable) DNS MX record value than the primary system. Such systems will accept any message for specified domains and attempt to forward them to the primary machine, a near-instant transaction if it is available else messages will simply queue.

It used to be standard practice to provide this service to ISP customers that hosted their own mail. With the advent of spam, many "blind" secondary MX services - those that accept any mail for a given domain - are being withdrawn in favour of systems that are aware of the entire mail routing table and only accept mail for valid recipients.

The next systems to double up or separate will be the outbound mail system, and DNS servers.

It is usually at this point that growth most negatively impacts provisioning systems. The more systems are added, the more difficult it is to keep consistent mail routing tables with flat files. It is likely that directory services in the form of LDAP, or perhaps even storing routing information in a read-optimized SQL database will be experimented with here. The goal is to write once and read many, not forgetting that the mail routing database must also be kept available.

4.3 Few to Many

The ISP now has tens of thousands of customers, and is processing millions of messages per day. Component outages must be reasonably guarded against with a design intention of offering N+1 redundancy, that is that the failure of any one instance of a component can be absorbed without loss of service.

Refer to Figure 10 for a full system diagram.

Provisioning is hopefully understood to be centralized to a update once and read by many style mail routing database, with configuration files rarely altered.

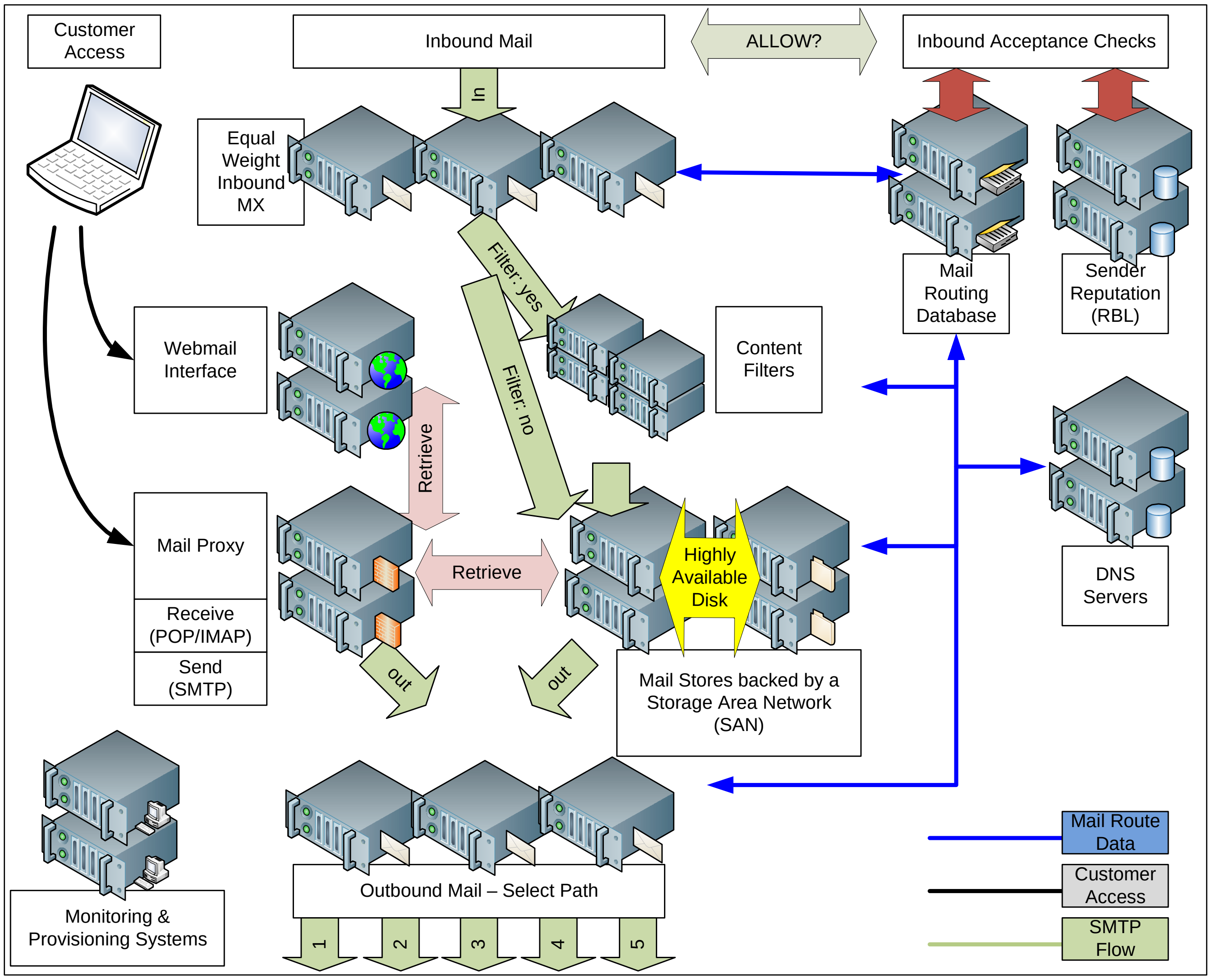


Figure 10: Large Scaling with Redundant Everything

Having two or more of everything is relatively straightforward. In the first case load balancing can be accomplished between systems by using DNS round robin A records, such as a "mailfilter" address that resolves to the IPs of all content filters. SMTP can cope with the failure of a system in such pools by re-queuing. For more robustness and predictability Layer 4 load balancers with downtime detection can be used, such as the Linux keepalived and IP Virtual Server project, or Cisco System's Stateful Load Balancing (SLB) product.

The most complicated component to multiply will be the mail stores. To tolerate the loss of a store, a highly available shared filesystem will have to be made readable and writable by multiple store systems. This implies that some form of distributed lock management must be implemented by the filesystem, or mail store/delivery agent. The author has found it preferable to use a MDA that implements its own locking protocol with simple breakable dot locks in order to provide graceful failure or timeouts when concurrent access is detected.

High availability and high performance network or block storage is available from many Storage Area Network (SAN) vendors, at a price. Alternatively, reasonable performance solutions are possible with commodity hardware and judicious use of open source software.

5. PERFORMANCE BOTTLENECKS

The commercial world has often and rightly been accused of outright ignoring the science behind complicated system design, preferring instead to cope with growth related challenges reactively [3].

Hence there is little published research or even reports on mail system performance that do not isolate a component of the mail architecture for analysis. Prior work has focused on the exhaustion of one specific resource and not their interrelationship, for example:

- [5] looked at computational / processor limits, generally ignoring the impact of disk input/output rates.
- [6] focuses only on disk input/output rates, disk reliability, and the impact on performance related to storage systems rebuilding RAID arrays following the failure of a component disk.
- [4] and [8] mention the components of email systems as this paper has done, but then inspect them individually.

This section will introduce observed load patterns against all system resources in the delivery chain.

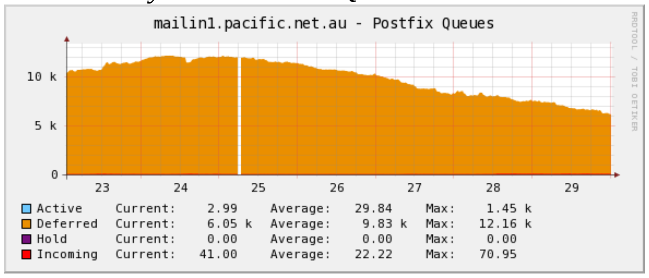
Note that the source mail system demonstrates peak usage times that are usually dependent on time of day (business hours 09:00-17:00) and day of week (Monday to Friday peaks, with Monday being the biggest). This is typical of ISPs with a customer base composed primarily of small & medium sized business.

5.1 Inbound mail flows

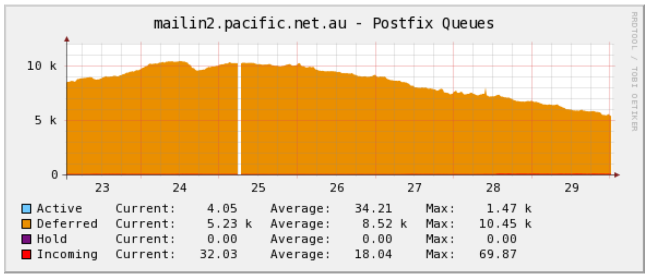
First a mention of mail queuing. Using a well designed queuing algorithm is key to MTA throughput. MTAs queue mail because there is no guarantee that the recipient host will be able to receive messages immediately. However, there is a well established pattern that most mail that will end up being delivered will do so within a reasonably short delay. This is in contrast to the SMTP RFC which specifies that MTAs *should* queue mail for redelivery up to 7 days [11]. This figure shows the age of the protocol: in the 1980's network connectivity was generally assumed to be sporadic. An exponential back-off queuing mechanism increases throughput by ensuring that relatively young messages get priority access to redelivery resources, and provide progressively less tries in time to aging messages.

These findings are interesting because the system under study allows legacy secondary MX support for other domains, meaning that the inbound queues will always remain rather large - bloated by messages that will probably never be delivered due to the spam-affected nature of such services.

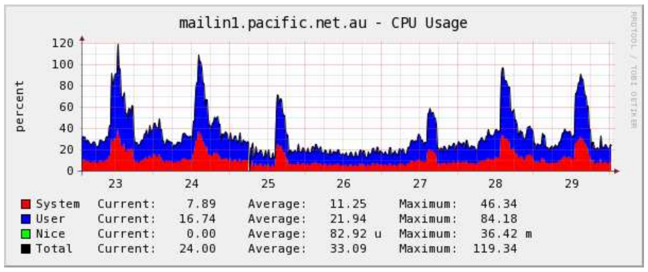
5.1.1 *Weekly Inbound Mail Queues to CPU*



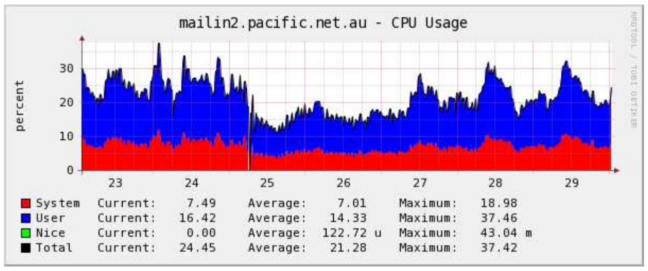
```
Active     Current:    2.99    Average:  29.84     Max:    1.45 k
Deferred   Current:    6.05 k  Average:   9.83 k   Max:   12.16 k
Hold       Current:    0.00    Average:   0.00     Max:    0.00
Incoming   Current:   41.00    Average:  22.22     Max:   70.95
```



```
Active     Current:    4.05    Average:  34.21     Max:    1.47 k
Deferred   Current:    5.23 k  Average:   8.52 k   Max:   10.45 k
Hold       Current:    0.00    Average:   0.00     Max:    0.00
Incoming   Current:   32.03    Average:  18.04     Max:   69.87
```

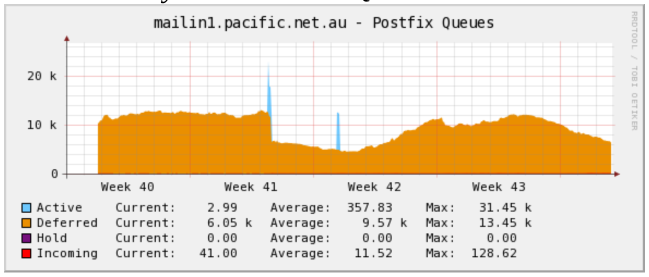


```
System   Current:    7.89    Average:   11.25     Maximum:   46.34
User     Current:   16.74    Average:   21.94     Maximum:   84.18
Nice     Current:    0.00    Average:   82.92 u   Maximum:   36.42 m
Total    Current:   24.00    Average:   33.09     Maximum:  119.34
```

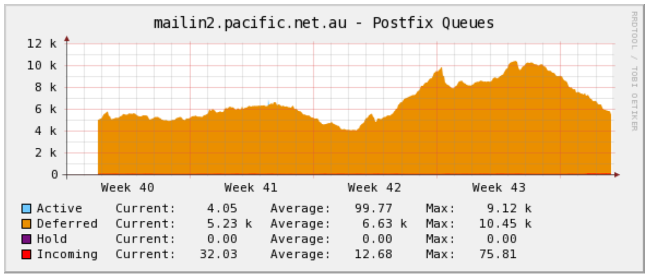


```
System   Current:    7.49    Average:    7.01     Maximum:   18.98
User     Current:   16.42    Average:   14.33     Maximum:   37.46
Nice     Current:    0.00    Average:  122.72 u   Maximum:   43.04 m
Total    Current:   24.45    Average:   21.28     Maximum:   37.42
```

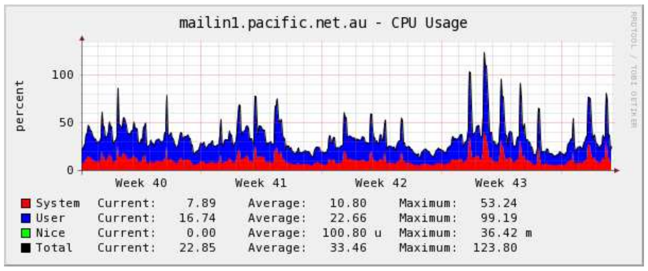
5.1.2 *Monthly Inbound Mail Queues to CPU*



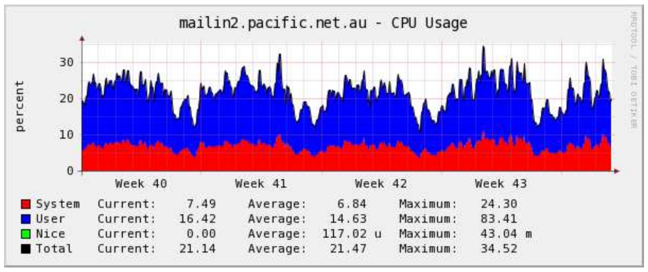
```
Active     Current:    2.99    Average:  357.83     Max:   31.45 k
Deferred   Current:    6.05 k  Average:    9.57 k   Max:   13.45 k
Hold       Current:    0.00    Average:    0.00     Max:    0.00
Incoming   Current:   41.00    Average:   11.52     Max:  128.62
```



```
Active     Current:    4.05    Average:   99.77     Max:    9.12 k
Deferred   Current:    5.23 k  Average:    6.63 k   Max:   10.45 k
Hold       Current:    0.00    Average:    0.00     Max:    0.00
Incoming   Current:   32.03    Average:   12.68     Max:   75.81
```



```
System   Current:    7.89    Average:   10.80     Maximum:   53.24
User     Current:   16.74    Average:   22.66     Maximum:   99.19
Nice     Current:    0.00    Average:  100.80 u   Maximum:   36.42 m
Total    Current:   22.85    Average:   33.46     Maximum:  123.80
```



```
System   Current:    7.49    Average:    6.84     Maximum:   24.30
User     Current:   16.42    Average:   14.63     Maximum:   83.41
Nice     Current:    0.00    Average:  117.02 u   Maximum:   43.04 m
Total    Current:   21.14    Average:   21.47     Maximum:   34.52
```

The first interesting result from these graphs is that load balancing by DNS MX "almost" works, but there is a bias for the first response provided - being mailin1 in this case.

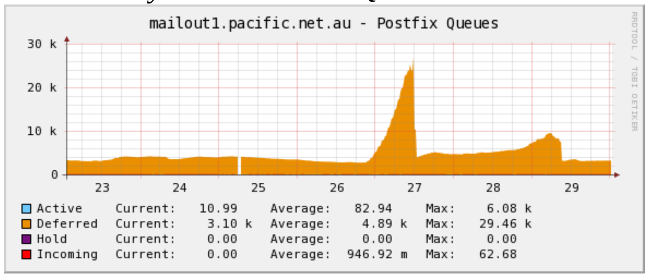
It is speculated that this occurs because customer mail domains are not always given both server names for load balancing, and because there are still many "broken" STMP and DNS clients - particularly spam sending botnets that do not fully or completely implement load balancing aspects of these protocols.

The result is bigger daytime load spikes on the mailin1 versus mailin2 machine, though interestingly for similar long run queue lengths.

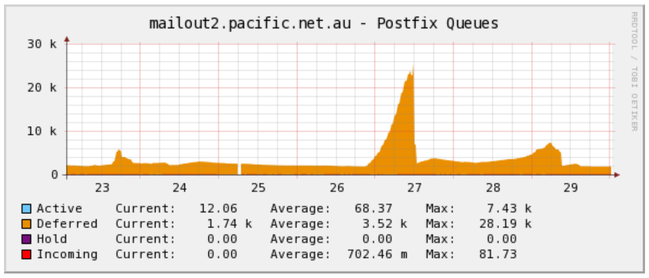
5.2 Outbound Mail Flows

Similar graphing on the outbound systems reveals a different story.

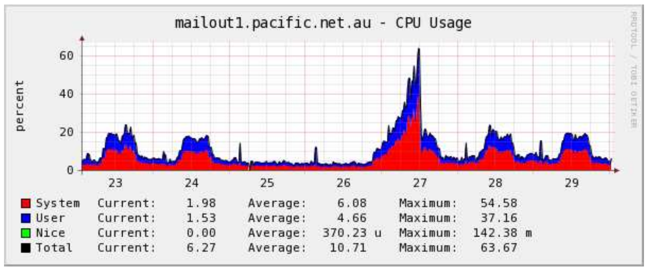
5.2.1 Weekly Outbound Mail Queues to CPU



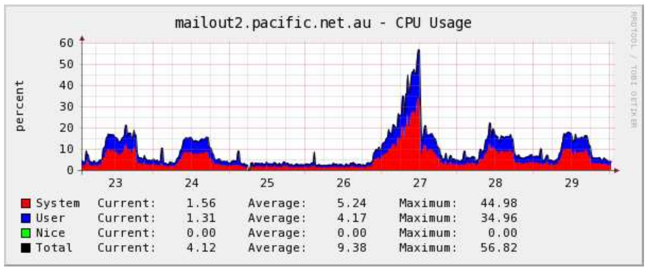
```
Active     Current:  10.99    Average:  82.94     Max:   6.08 k
Deferred   Current:   3.10 k  Average:   4.89 k   Max:  29.46 k
Hold       Current:   0.00    Average:   0.00     Max:   0.00
Incoming   Current:   0.00    Average: 946.92 m   Max:  62.68
```



```
Active     Current:  12.06    Average:  68.37     Max:   7.43 k
Deferred   Current:   1.74 k  Average:   3.52 k   Max:  28.19 k
Hold       Current:   0.00    Average:   0.00     Max:   0.00
Incoming   Current:   0.00    Average: 702.46 m   Max:  81.73
```

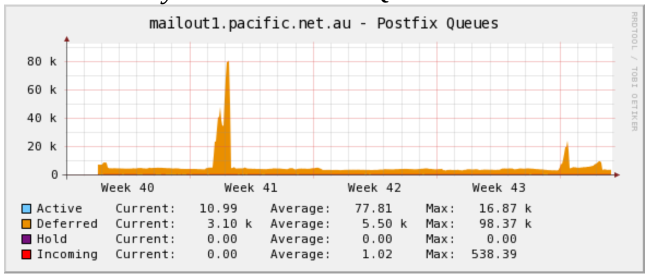


```
System  Current:  1.98   Average:   6.08     Maximum:  54.58
User    Current:  1.53   Average:   4.66     Maximum:  37.16
Nice    Current:  0.00   Average: 370.23 u   Maximum: 142.38 m
Total   Current:  6.27   Average:  10.71     Maximum:  63.67
```

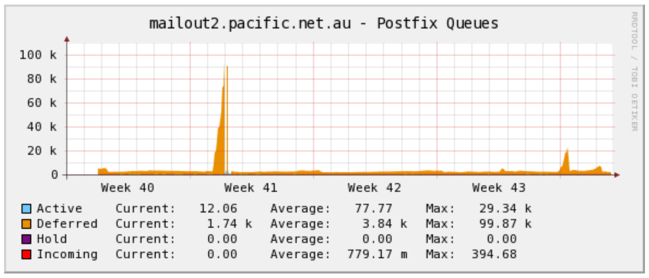


```
System  Current:  1.56   Average:  5.24   Maximum:  44.98
User    Current:  1.31   Average:  4.17   Maximum:  34.96
Nice    Current:  0.00   Average:  0.00   Maximum:   0.00
Total   Current:  4.12   Average:  9.38   Maximum:  56.82
```

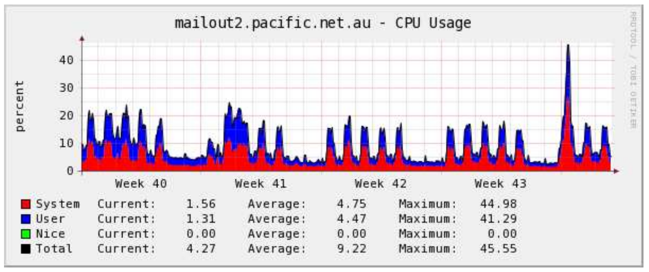
5.2.2 Monthly Outbound Mail Queues to CPU



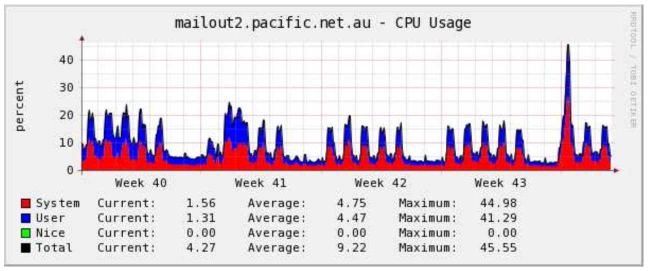
```
Active     Current:  10.99    Average:  77.81     Max:  16.87 k
Deferred   Current:   3.10 k  Average:   5.50 k   Max:  98.37 k
Hold       Current:   0.00    Average:   0.00     Max:   0.00
Incoming   Current:   0.00    Average:   1.02     Max: 538.39
```



```
Active     Current:  12.06    Average:  77.77     Max:  29.34 k
Deferred   Current:   1.74 k  Average:   3.84 k   Max:  99.87 k
Hold       Current:   0.00    Average:   0.00     Max:   0.00
Incoming   Current:   0.00    Average: 779.17 m   Max: 394.68
```



```
System  Current:  1.56   Average:  4.75   Maximum:  44.98
User    Current:  1.31   Average:  4.47   Maximum:  41.29
Nice    Current:  0.00   Average:  0.00   Maximum:   0.00
Total   Current:  4.27   Average:  9.22   Maximum:  45.55
```



```
System  Current:  1.56   Average:  4.75   Maximum:  44.98
User    Current:  1.31   Average:  4.47   Maximum:  41.29
Nice    Current:  0.00   Average:  0.00   Maximum:   0.00
Total   Current:  4.27   Average:  9.22   Maximum:  45.55
```

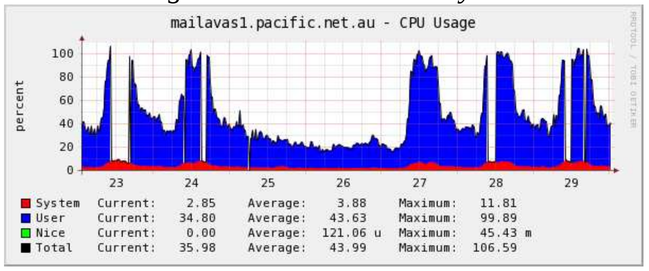
The first thing to note here is that the outbound load balancing is much more even, as it is entirely within the mail system's control - connection decisions are not left up to outside sources.

The second interesting bit is the behaviour during a load spike due to a set of exploited accounts attempting to send millions of spam messages on the 27th - clearly visible on the weekly graphs. This ties large queues and the redelivery attempts that are associated with them to increased CPU load, mostly due to system (file I/O) activity seen in red.

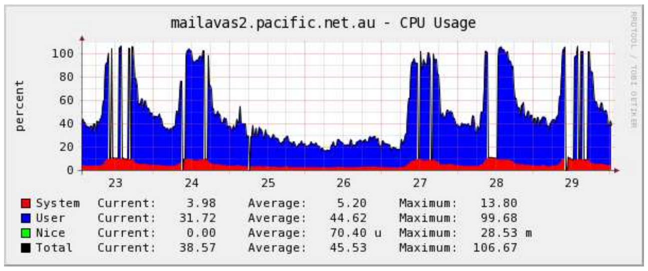
5.3 Filtering: CPU

The next element is the content filtering chain, which consumes relatively low I/O (low system busy CPU), but completely maximizes CPU resources to the content filters (high user busy CPU). Note that the measurement system did not return a result at very high load, creating the gaps seen in the graphs. These graphs demonstrate that at peak times there is no available spare computational power - the need to upscale the number of systems is evident.

5.3.1 Filtering CPU Utilization Weekly

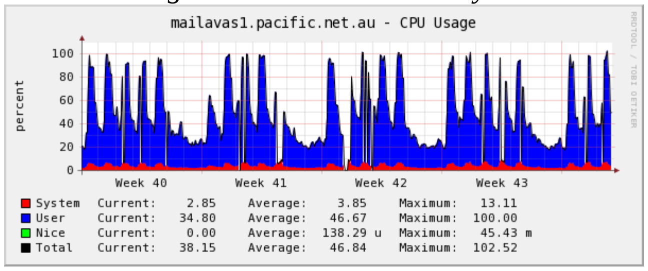


```
System  Current:   2.85   Average:   3.88     Maximum:  11.81
User    Current:  34.80   Average:  43.63     Maximum:  99.89
Nice    Current:   0.00   Average: 121.06 u   Maximum:  45.43 m
Total   Current:  35.98   Average:  43.99     Maximum: 106.59
```

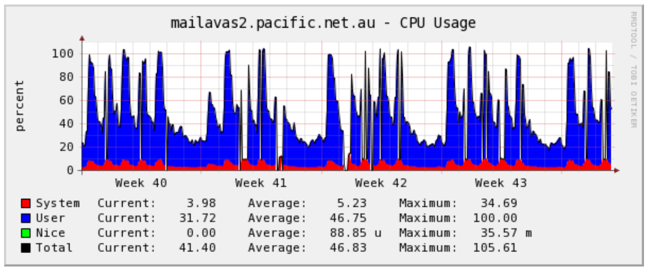


```
🟥 System  Current:    3.98   Average:    5.20    Maximum:   13.80
🟦 User    Current:   31.72   Average:   44.62    Maximum:   99.68
🟩 Nice    Current:    0.00   Average:   70.40 u  Maximum:   28.53 m
⬛ Total   Current:   38.57   Average:   45.53    Maximum:  106.67
```

5.3.2 *Filtering CPU Utilization Monthly*



```
🟥 System  Current:    2.85   Average:    3.85    Maximum:   13.11
🟦 User    Current:   34.80   Average:   46.67    Maximum:  100.00
🟩 Nice    Current:    0.00   Average:  138.29 u  Maximum:   45.43 m
⬛ Total   Current:   38.15   Average:   46.84    Maximum:  102.52
```



```
🟥 System  Current:    3.98   Average:    5.23    Maximum:   34.69
🟦 User    Current:   31.72   Average:   46.75    Maximum:  100.00
🟩 Nice    Current:    0.00   Average:   88.85 u  Maximum:   35.57 m
⬛ Total   Current:   41.40   Average:   46.83    Maximum:  105.61
```

The recommendation here is obvious, don't run at 100%.

5.4 Mail Stores: Disk throughput and latency linked to CPU

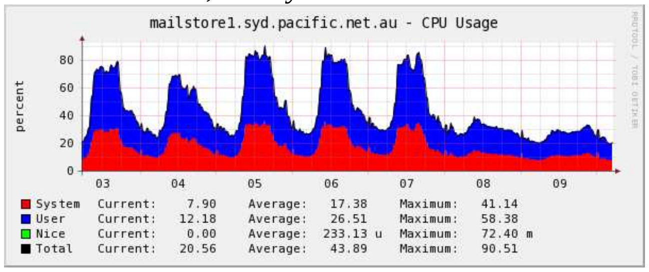
As demonstrated in the large ISP mail system graph, the mailstores are comprised of two parts. First the mail store servers, that operate the Mail Delivery Agents to write to the mailboxes, and the POP3 & IMAP4 daemons to read from the mailboxes. No encryption or other ancillary services are provided at the stores - if encryption is required by the customer it is provided by the proxies.

The stores are backed by NetAPP filers providing high availability NFS services that are shared by the front ends. We can see that as load increases, the service latency increases correspondingly.

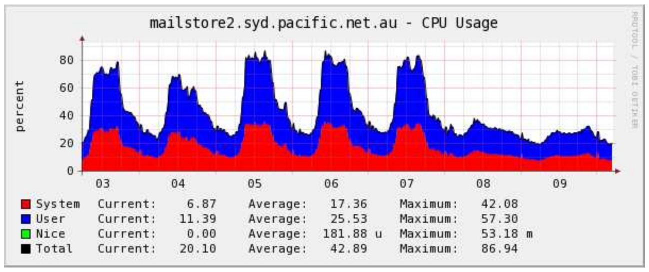
As the repeating cycle of weekly-monthly graphs has already been established, we will focus only on weekly data.

In the following graphs the 8th and 9th are a weekend.

5.4.1 *Store CPU, weekly*



```
🟥 System  Current:    7.90   Average:   17.38    Maximum:   41.14
🟦 User    Current:   12.18   Average:   26.51    Maximum:   58.38
🟩 Nice    Current:    0.00   Average:  233.13 u  Maximum:   72.40 m
⬛ Total   Current:   20.56   Average:   43.89    Maximum:   90.51
```

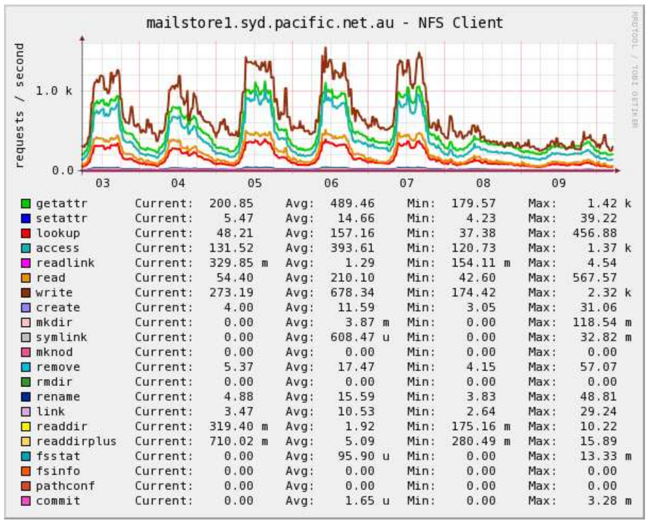


```
🟥 System  Current:    6.87   Average:   17.36    Maximum:   42.08
🟦 User    Current:   11.39   Average:   25.53    Maximum:   57.30
🟩 Nice    Current:    0.00   Average:  181.88 u  Maximum:   53.18 m
⬛ Total   Current:   20.10   Average:   42.89    Maximum:   86.94
```

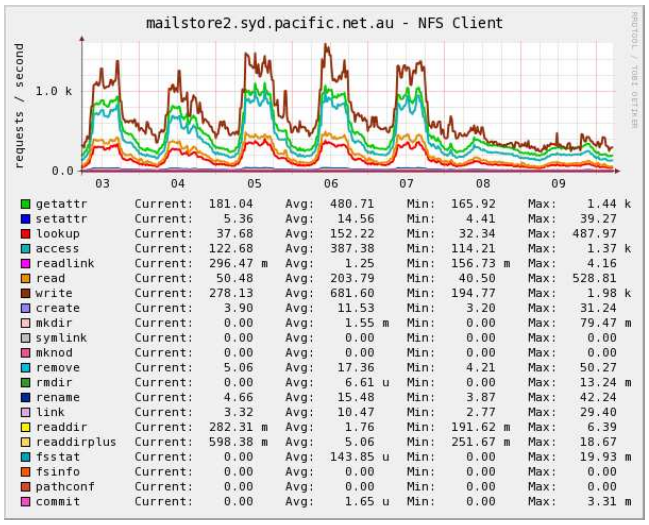
The division between System and User consumption is important here, as the user percentage is almost entirely due to the POP3 and IMAP4 daemons running for mail retrieval. The System component is almost entirely due to the underlying linux kernel's NFS client.

5.4.2 *Store NFS Client*

The NFS client's operation was graphed to investigate the types of operations being executed by the client.



```
🟩 getattr      Current:  200.85    Avg:  489.46    Min:  179.57    Max:    1.42 k
🟦 setattr      Current:    5.47    Avg:   14.66    Min:    4.23    Max:   39.22
🟥 lookup       Current:   48.21    Avg:  157.16    Min:   37.38    Max:  456.88
🟦 access       Current:  131.52    Avg:  393.61    Min:  120.73    Max:    1.37 k
🟪 readlink     Current:  329.85 m  Avg:    1.29    Min:  154.11 m  Max:    4.54
🟧 read         Current:   54.40    Avg:  210.10    Min:   42.60    Max:  567.57
🟫 write        Current:  273.19    Avg:  678.34    Min:  174.42    Max:    2.32 k
🟪 create       Current:    4.00    Avg:   11.59    Min:    3.05    Max:   31.06
⬜ mkdir        Current:    0.00    Avg:    3.87 m  Min:    0.00    Max:  118.54 m
⬜ symlink      Current:    0.00    Avg:  608.47 u  Min:    0.00    Max:   32.82 m
🟥 mknod        Current:    0.00    Avg:    0.00    Min:    0.00    Max:    0.00
🟦 remove       Current:    5.37    Avg:   17.47    Min:    4.15    Max:   57.07
🟩 rmdir        Current:    0.00    Avg:    0.00    Min:    0.00    Max:    0.00
🟦 rename       Current:    4.88    Avg:   15.59    Min:    3.83    Max:   48.81
🟪 link         Current:    3.47    Avg:   10.53    Min:    2.64    Max:   29.24
🟨 readdir      Current:  319.40 m  Avg:    1.92    Min:  175.16 m  Max:   10.22
🟨 readdirplus  Current:  710.02 m  Avg:    5.09    Min:  280.49 m  Max:   15.89
🟦 fsstat       Current:    0.00    Avg:   95.90 u  Min:    0.00    Max:   13.33 m
🟧 fsinfo       Current:    0.00    Avg:    0.00    Min:    0.00    Max:    0.00
🟥 pathconf     Current:    0.00    Avg:    0.00    Min:    0.00    Max:    0.00
🟪 commit       Current:    0.00    Avg:    1.65 u  Min:    0.00    Max:    3.28 m
```

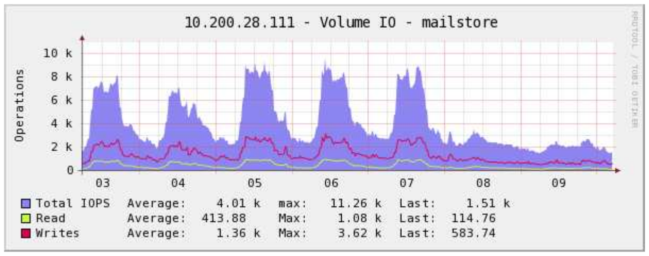


```
🟩 getattr      Current:  181.04    Avg:  480.71    Min:  165.92    Max:    1.44 k
🟦 setattr      Current:    5.36    Avg:   14.56    Min:    4.41    Max:   39.27
🟥 lookup       Current:   37.68    Avg:  152.22    Min:   32.34    Max:  487.97
🟦 access       Current:  122.68    Avg:  387.38    Min:  114.21    Max:    1.37 k
🟪 readlink     Current:  296.47 m  Avg:    1.25    Min:  156.73 m  Max:    4.16
🟧 read         Current:   50.48    Avg:  203.79    Min:   40.50    Max:  528.81
🟫 write        Current:  278.13    Avg:  681.60    Min:  194.77    Max:    1.98 k
🟪 create       Current:    3.90    Avg:   11.53    Min:    3.20    Max:   31.24
⬜ mkdir        Current:    0.00    Avg:    1.55 m  Min:    0.00    Max:   79.47 m
⬜ symlink      Current:    0.00    Avg:    0.00    Min:    0.00    Max:    0.00
🟥 mknod        Current:    0.00    Avg:    0.00    Min:    0.00    Max:    0.00
🟦 remove       Current:    5.06    Avg:   17.36    Min:    4.21    Max:   50.27
🟩 rmdir        Current:    0.00    Avg:    6.61 u  Min:    0.00    Max:   13.24 m
🟦 rename       Current:    4.66    Avg:   15.48    Min:    3.87    Max:   42.24
🟪 link         Current:    3.32    Avg:   10.47    Min:    2.77    Max:   29.40
🟨 readdir      Current:  282.31 m  Avg:    1.76    Min:  191.62 m  Max:    6.39
🟨 readdirplus  Current:  598.38 m  Avg:    5.06    Min:  251.67 m  Max:   18.67
🟦 fsstat       Current:    0.00    Avg:  143.85 u  Min:    0.00    Max:   19.93 m
🟧 fsinfo       Current:    0.00    Avg:    0.00    Min:    0.00    Max:    0.00
🟥 pathconf     Current:    0.00    Avg:    0.00    Min:    0.00    Max:    0.00
🟪 commit       Current:    0.00    Avg:    1.65 u  Min:    0.00    Max:    3.31 m
```

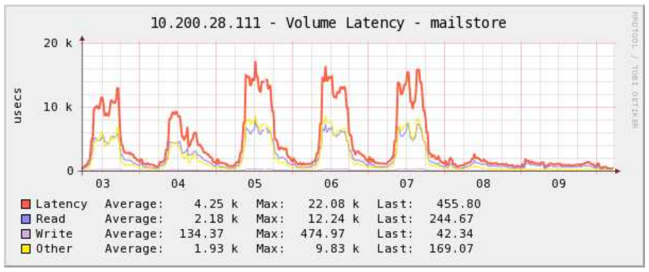
The surprising revelation here being that the stores are write intensive (dark brown). The read process causes moderately less direct read load, but is considerable given the fsstat and getattr instructions are associated with a read call.

5.4.3 Shared Mailstore SAN Volume

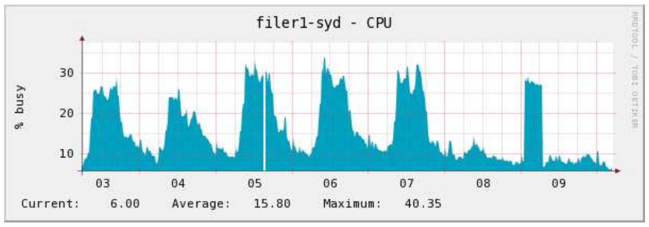
The SAN records an aggregate count of all Input/Output Operations Per Second (IOPS). The NFS part of which are read and write - and shown to be consistent with the conclusions drawn above. The difference between total and the sum of read and write is the other non-reported operations that are seen by the client.



We can see that the SAN's latency is related to utilization, service time increases linearly with load:



This is unsurprisingly linked the SAN's CPU, which provides the NFS service:



5.5 Problems: Attacks and Bugs

An outcome from observing performance metrics in a fully redundant system is analyzing the impacts of bugs or Spam delivery attacks.

It would be interesting to have heuristics that detect attacks as they are developing - that is notice a departure from the usual queue length, disk I/O and CPU load patterns.

Similarly, malformed messages still occasionally cause the processes within the content filters to fail or loop, leading to one of the content filtering servers to "spin" repeatedly over one message - reading from its work queue, into a faulty daemon that crashes, then back into the work queue of the next spawned process that will it too crash & restart. The effect is similar to a spam attack, a slowly growing mail queue with associated longer mail delivery times, and reduced overall throughput.

Attacks and bugs are not all bad, they do provide the ISPs staff with the opportunity to observe the mail system operating at peak capacity to clear an accumulated backlog once the fault is rectified or spam source squelched. It is useful to observe the mail system's component performance at full load to find other bottlenecks or impedance mismatches.

One discovered impedance mismatch was an MTA daemon capable of operating 128 simultaneous processes feeding into a policy daemon that is limited to 64, that may in turn be attempting to read from a LDAP-based mail routing database that is capped to 32. Discovering these allows system tuning to ensure that chained daemons respond to one another's request, instead of taking the far more damaging assumption that a failure has occurred.

6. CONCLUSIONS

It can generally be said that the protocols and systems that underpin modern ISP email architectures have aged well. There is demonstrable ability to scale mail systems and cope with service impeding threats such as spam.

Some would argue that the entire SMTP protocol should be shelved in favour of newer more robust designs, but the author believes that getting worldwide adoption for a completely new protocol to be a near impossible task to synchronize. Besides, SMTP has proven to cope with technical challenges, while allowing other adjunct mechanisms to cope with society's insatiable desire to offer and purchase services - and the resultant economics that make sending 11.8 Million messages to make 1 sale profitable.

6.1 Future Work

As mentioned, properly integrating provisioning systems with mail architectures is an often overlooked step. Yet as this report's raw data was being collected, it appeared that other related information was just as overlooked.

There is a strong association between the systems providing a service, the systems that provision a service, the systems that monitor the availability and efficacy of the delivered service, and the systems that allow observers to troubleshoot the behavior of a service and understand its many component parts. The author is unaware of any research or implemented products that deliberately relate more than 2 or 3 of these elements, yet the author does spend considerable time and resources manually or quasi-manually having to build these relationships whenever a system must be scaled, troubleshot, commissioned or decommissioned. More automation of system configuration tracking and resultant collection of historical performance statistics would be welcome in this field.

REFERENCES

[1] Hilal, W.B-E.D., Yuen, H.-T. (1988) Designing Large Electronic Mail Systems. *8th International Conference on Distributed Computing Systems* (June 1988). DOI: 10.1109/DCS.1988.12542

[2] McQuillan, J. and Walden, D. (1980) Designing electronic mail systems that people will use. *ACM SIGOA Newsletter* 1, 2 (May. 1980), 5-6. DOI= http://doi.acm.org/10.1145/1022232.1022235

[3] Ghandi, N. TIlbury, D.M. Parekh, S. Hellerstein, J. (2001) Feedback control of a Lotus Notes server: Modelling and Control Design. *Proceedings of the 2001 American Control Conference.*(June 2001). DOI: 10.1109/ACC.2001.946372

[4] Darst, C. Ramanathan, S. (1999) Measurement and Management of Internet Services. *Proceedings of the Sixth IFIP/IEEE International Symposium on Integrated Network Management.*(May 1999)

[5] Wang, J. and Hu, Y. 2004. A performance study on Internet Server Provider mail servers. In *Proceedings of the Ninth international Symposium on Computers and Communications 2004 Volume 2 (Iscc"04) - Volume 02* (June 28 - July 01, 2004). ISCC. IEEE Computer Society, Washington, DC, 56-61.

[6] Navarro, G. Manic, M. (2007) Predictive e-mail server performability Analysis based on Fuzzy Arithmetic. *International Joint Conference on Neural Networks* (12 - 17 August 2007). DOI: 10.1109/IJCNN.2007.4371337

[7] Alperovitch, D. Judget, P. Krasser, S. (2007) Taxonomy of Email Reputation Systems. *27th International Conference on Distributed Computing Systems* (22-29 June, 2007). DOI: 10.1109/ICDCSW.2007.78

[8] Caswell, D. Ramanathan, S. (2000) Using Service Models for Management of Internet Services. *IEEE Journal on Selected Areas in Communications, Vol 18, No5 (May 2000)* DOI: 10.1109/49.842985

[9] Koymans, C.PJ. Scheerder, J. (2007) Email. In *Handbook of Network and System Administration*, Bergstra, J. and Burgess, M. editors. Elsevier. 147-172.

[10] Network Control Program. In *Wikipedia*. http://en.wikipedia.org/wiki/Network_Control_Program. Retrieved 2008-11-09

[11] Postel, J.B. Simple Mail Transfer Protocol. IETF RFC 821, August 1982.

[12] Crocker, D.H. Standard for the Format of ARPA Internet Text Messages. IETF RFC 822, August 1982.

[13] Partridge, C. Mail Routing and the Domain System. IETF RFC 974, January 1986.

[14] Klensin, J. Editor. Simple Mail Transfer Protocol. IETF RFC 2821, April 2001.

[15] Resnick, P. Editor. Internet Message Format. IETF RFC 2822, April 2001.

[16] Reynolds, J.K. Post Office Protocol. IETF RFC 918, October 1984.

[17] Butler, M. Postel, J. Chase, D. Goldberger, J. Reynolds, J.K. Post Office Protocol: Version 2. IETF RFC 937, February 1985.

[18] Myers, J. Rose, M. Post Office Protocol: Version 3. IETF RFC 1996, May 1996.

[19] Kanich, C., Kreibich, C., Levchenko, K., Enright, B., Voelker, G. M., Paxson, V., and Savage, S. 2008. Spamalytics: an empirical analysis of spam marketing conversion. In *Proceedings of the 15th ACM Conference on Computer and Communications Security* (Alexandria, Virginia, USA, October 27 - 31, 2008). CCS '08. ACM, New York, NY, 3-14. DOI= http://doi.acm.org/10.1145/1455770.1455774

[20] Myers, J. Local Mail Transfer Protocol. IETF RFC 2033, October 1996.

[21] Crispin, M. Internet Message Access Protocol: Version 4rev1. IETF RFC 3501, March 2003.