Usability and efficiency improvements of the (GNU Mailman) mailing lists

Rudy Borgstede (Rudy.Borgstede@gmail.com)

System and Network Engineering
University of Amsterdam

July 5, 2008

Versions

Version	Date	Changes
0.0.1	18 May 2008	First setup of the report
0.0.2	9 June 2008	Release Candidate 1 of the project proposal
0.1.1	17 June 2008	A rewrite of the document because of the change of project result. The project will deliver an advice rather then a product like a patch or add-on for GNU Mailman. This means that the report becomes an consultancy report instead of a project proposal.
1.0.0	30 June 2008	Final version 1 of the report.
1.0.1	1 July 2008	A spelling check of the report.
1.0.2	5 July 2008	Extending the conclusion en future work chapters.

Participants

Name	Contact Information
University of Amsterdam Rudy Borgstede (Student) Cees de Laat (Supervisor)	Rudy.Borgstede@gmail.com delaat@uva.nl
NLnet Michiel Leenaars (Supervisor)	Michiel@nlnet.nl

Abstract

This report is the result of a research project of four weeks at the NLnet Foundation[1] in Amsterdam. The NLnet Foundation is a foundation who financially supports the open-source community and their projects. The purpose of the project is to improve the usability and the administration of the mailing lists (of the foundation) and giving a more clear view on mailing list server software to anyone who is interested in using mailing list server software or developing new mail or mailing list server software. The report describes the research of the usability of several open-source mailing list server software for scalable environments with several well known mail servers. The report also describes which issues the NLnet Foundation encounters with using GNU Mailman in a scalable and open environment with more than one administrative domain.
The result of the report exists out of two parts:

1. How to improve the mailing list server software to survive the evolution of the internet and their flexible communities
2. How to improve the mailing lists at the NLnet Foundation

[1]The NLnet Foundation homepage: http://www.nlnet.nl

Contents

Chapter 1

Introduction

This report is the result of a research project of four weeks at the NLnet Foundation[12] in Amsterdam. The NLnet Foundation is a foundation who financially supports the open-source community and their projects. The research project is conducted by the student Rudy Borgstede of the University of Amsterdam[19] under supervision of Cees de Laat from the University of Amsterdam research group[19] and Michiel Leenaars, director of strategy of the NLnet Foundation. The purpose of the project is to improve the usability and the administration of the mailing lists (of the foundation) and giving a more clear view on mailing list server software to anyone who is interested in using mailing list server software or developing new mail or mailing list server software. The report describes the research of the usability of several open-source mailing list server software in scalable environments with well known mail servers. The report describes also the issues that the NLnet Foundation encounters with using GNU Mailman in their scalable and open environments. The report will generally not be focused on the past of these server software but rather on their future, what is their potential? This choice is made because there is enough documentation (books, websites, wikipedia's etc.) about different mail and mailing list servers but almost no strict reviews on what they are really worth.
The result of the report exist out of two parts:

- How to improve the mailing list server software to survive the evolution of the internet and their flexible communities
- How to improve the mailing lists at the NLnet Foundation

The report is setup in the following chapters:

- **Mail server software**. This chapter explain what the main functionalities of a mail server are and what extra functionalities each mail server has to offer. This chapter can be skipped by people who are well familiar with the security issues of the widely used mail servers.
- **Mailing list server software**. This chapter reviews the popular mailing list servers which are interesting for scalable environments with multiply administrative domains.
- **The NLnet Foundation Case**. This chapter explains the issues that the NLnet Foundation have with their current GNU Mailman mailing list server software.
- **Conclusion**. In this chapter a conclusion is made about the findings about the mailing list server software and the issues of the NLnet Foundation.
- **Future Work**. Within most reports this chapter is philosophical but in this report this chapter is very important because it describes the catching up that is needed to survive the evolution of the internet.
- **Appendices**. Some useful appendices that makes the project environment more clear like the planning and the official (Dutch) project description.

Chapter 2

Mail server software

Before the report starts about mailing lists, first the mail servers should be discussed because of their close relationship. In this chapter the basics of mail servers are explained and legacy issues are addressed. Also some popular mail servers will be discussed with their strength and weaknesses. The mail servers that are interesting for this report are mail servers that are open-source and have a free license e.g. the GNU General Public License version 2/3. Also the mail server must be widely used and accepted as a mature mail server. In the figure 2 the components

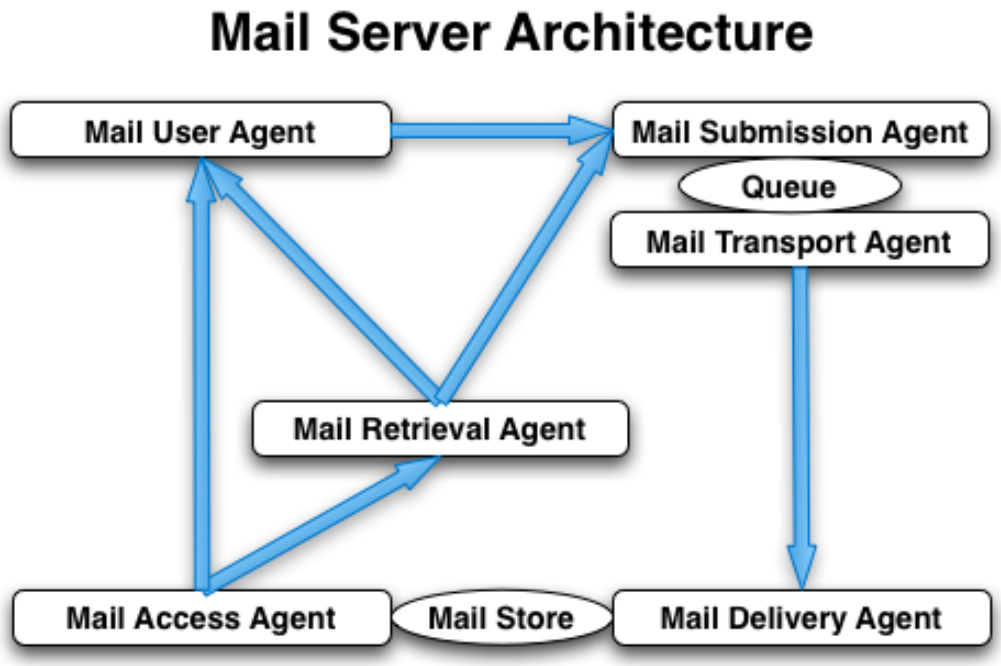


Figure 2.1: The mail server architecture

and the relationships between the components are shown of a mail server. Each component has it's own function in the mail process and is essential for a fully working mail process. It is important to remember that all these components are abstract and could be joined together or skipped if it delivers a better performance or architecture e.g. most graphical Microsoft Windows mail clients can be seen as the MUA, the MSA and the MTA.

2.1 Mail User Agent

The MUA is the mail client of the end user, which can be something graphical like Microsoft Outlook[1] but also a terminal client like mailx[2]. The main purpose of the MUA is to present mail to the user and to create new mail to send to another user.

2.2 Mail Submission Agent

After the mail is created by the MUA it is send to the MSA. The MSA validates the mail and can fix minor errors e.g. the Date header. If the MSA is finished it decides to either drop/delete the mail or forward it into the queue for transport to it's destination. The MSA can decide to drop the mail because of restrictions of the mail server e.g. I will only send mail to my own computer system or local area network (internal mail).

2.3 Mail Transport Agent

The MTA reads the mails out of the queue and sends it, by the use of the SMTP protocol[3], to the destination across the internet he thinks is right e.g. the destination can be an alias to another address which only the first destination knows.

2.4 Mail Delivery Agent

The MDA can receive mails send by MTA of another mail server. When the MDA receives a mail then he decides if he wants to accept the mail e.g. I will only accept mail for mail accounts I locally posses or if the virus scanner recognizes the mail then it must be dropped. If the MDA accept the mail then it puts the mail into the mail store of the right mail account, which is described by most mail servers in an alias file. This file describes on which queue or program the mail should be forwarded to be accessed by the Mail Access Agent. If the mail-address doesn't exist in the local domain an error-mail can be returned by the MAA and MRA and then back into the mail system to the MSA.

2.5 Mail Access Agent

The MAA retrieves the right mails on request of an authenticated MUA out of the mail store and let the MUA access it by e.g. the POP3[4] or IMAP[5] protocol.

2.6 Mail Retrieval Agent

The MRA gets mail that can't be retrieved by the MUA direct and is used in two cases:

1. **The server must relay the mail to another mail server because he can't deliver it.** If the server doesn't know the mail address or can't deliver it (an error response) then the MRA will inject the mail into the MSA to send the mail to the next right destination.
2. **The mail is delivered to the MUA by a special program which plays the role of MRA.** The MRA will act as the MAA for the MUA. This is sometimes necessary to enforce policies onto received mails or for efficiency e.g. the real MAA isn't always accessible or available.

[1] Homepage Microsoft Outlook: http://www.microsoft.com/outlook/
[2] Wikipedia mailx: http://en.wikipedia.org/wiki/Mailx
[3] SMTP protocol: http://tools.ietf.org/html/rfc821
[4] POP3 protocol: http://tools.ietf.org/html/rfc1939
[5] IMAP protocol: http://tools.ietf.org/html/rfc3501

2.7 Sendmail

Sendmail is a very old mail server original developed by Eric Allman in the early 1980's as descendant of the ARPANET Delivermail mail server and comes with his own free license. Sendmail is one of the most used Unix mail servers today, which is probably because it is most times standard installed on Unix (Linux) based operating systems. The basic idea of Sendmail is to do all mail server tasks with one single heavy weight binary. With this vision Sendmail should be simple to configure, but it isn't, the configuration is based on complex rules which can be generated by so called m4 macro's. The system was effective in the age (before 2000) where effective GUI's were very costly or simply didn't exist but currently you probably need to read a book to edit the legacy based configuration. Besides the static mail server configuration there is also a so called alias file which contains the reference between the mail addresses of the mail server and the local users and mail queue's which is located as files in the spool directory on a Unix based operating system. [16]

2.8 QMail

QMail is a mail server developed by Daniel J. Bernstein which is used as the standard Freebsd 7.0 mail server. QMail need to be recompiled with any change to the mail server architecture e.g. by inserting plugin and patches in the official source-code. QMail has a history of being more secure then Sendmail and being distributed under the public domain license which means it has no legal restrictions to anyone. QMail works like Sendmail with an alias file, but have a much easier configuration file. Any complex features that QMail doesn't have can be integrated by patches.[15]

2.9 Exim 4

Exim 4 (once an abbreviation of Experimental Internet Mailer) is a mail server developed by Philip Hazel in 1995 at the University of Cambridge and distributed under de GNU General Public License version 2. Exim has compared to Sendmail and QMail an easy configuration and is further known because of his many plugin functionalities. The configuration of Exim 4 is divided in functional directories like acl (Access Control List), rewrite and auth (Authentication).[4]

2.10 Courier

The Courier mail server[2] is published by the Double Precision, Inc. under the GNU general Public License. Courier is the only mail server which has a web interface and a has built-in facility for modules which gives the external developers the possibility to create modules without have to worry about the Courier core. These modules can be seen as a loose binary with their own configuration script with their own syntax. While Courier is available on many Linux distributions it isn't well known e.g. the website Mailradar doesn't even know the existence of Courier[10], which can be because of a look-a-like signature, a mail signature which looks like e.g. Postfix.

2.11 Postfix

Postfix is a mail server published under the IBM Public License and can be seen as the alternative to Sendmail because it supports the same kind of configuration and alias files and fakes even the Sendmail binary for an easier transition to Postfix. Therefore Postfix has a lot of legacy material but it is necessary to update the old mail servers. But Postfix is different to the Sendmail philosophy because Postfix separate each mail sever component in loose binaries which optimizes the mail server performance. Postfix is slowly taking over the Sendmail servers because of it's superior performance it's becoming the most used open-source mail server.[20]

2.12 What mail server to choose?

There can't be one choice, if one mail server is significant better then the others I shouldn't have to describe all these servers. The choice of the right mail server is purely a matter of taste. What is clear within the open-source choices available today is that Postfix, QMail and Exim can be considered to have the biggest market share.[6] The result of this is that these mail servers have the biggest development groups and therefore should have the best prospect for the future.

2.13 Security

Most mail servers are developed many years ago and have developed security the hard way. Today the main stream mail servers could be considered safe because of different levels of security.

2.13.1 Configuration

The basic level of security is described by the configuration rules like not being enabled to relay mail for unknown domains or require a minimal set of information to detect e.g. mail loops. The configuration is different described by each mail server but have the same result e.g. QMail have a special patch and Postfix have filter rules. Besides these rules also external programs can ensure the safety of the final destination like black lists and virus scanners.

2.13.2 SSL

If the configuration is setup right the mail process could be considered bonded to the rules of the administrator. Only this doesn't include the transport of the mail between different servers. While the mail is transferred between two mail servers an attacker can modify the mail. This must be prevented therefore the mail protocols (IMAP, POP3 and SMTP) can be secured by SSL[7]. SSL delivers encryption and integrity validation for the mail transfers, which means that the sender of the mail can be sure that the mail is still the same and unread at the destinated mail server.

2.13.3 Sender and mail server authentication

Based on configuration and the SSL technology sending a mail could be considered secure but the complete mail process says nothing about the intensions of the administrator of the mail server and the sender of the mail. If the server is controlled by an attacker or by an unexperienced administrator the server may be configured wrongly so it can be used for anonymous spamming other mail servers because the server can send mail for users from another mail domain and can modify any mail relayed through that server. The mail server administrators of the world are trying to prevent this situation by blacklisting open-relay mail servers, mail servers that will forward any mail from the internet and not only for a specific mail domain. But even with this blacklist it can't be ensured that there aren't mail server that supports open-relay e.g. a botnet can act as a distributed anonymous mail server.

2.13.4 OpenPGP

OpenPGP [13] is a technology to sign and optional encrypt mail which results in a kind of end-to-end SSL connection to transfer a mail message. OpenPGP can be enabled in almost every mail client and is a possibility to validate the sender and makes the mail encrypted and therefore only readable to the right receiver. This makes it possible to drop every mail that hasn't been signed on the client side with the simple policy anyone who I don't know is a spammer. The problem with this policy is that people who don't use OpenPGP will be unable to communicate with you.

[6] MailRadar a mail statistics webpage: http://www.mailradar.com
[7] Secure Socket Layer wikipedia description: http://en.wikipedia.org/wiki/Transport_Layer_Security

So OpenPGP is only useful for applications that need validation of the sender, the integrity of the mail and optional the encryption of the mail. It is possible to presume that every mail needs OpenPGP signing and encryption but this will exclude anyone who doesn't support OpenPGP of reading your message, but still only signing a mail is a good solution until everyone uses the OpenPGP technology.

Chapter 3

Mailing list server software

Mailing list server software is software to manage lists of mail addresses which can be used to distribute in a highly efficient way a copy of a mail to a large number of people by the use of a special mail-adress (<listname>@mailserver) on the mail server, discussed in the previous chapter. There are two types of mailing lists:

- **The newsletter** is a mailing list where anyone or well defined group can subscribe or is subscribed but only can be send to by a single email-address or by a special email-address (member) group.
- **The (long-distance) discussion list** is a member based mailing list which can be divided into two lists[6]:
  - **The open discussion list** is a mailing list to which everyone can join, but it still allows for moderation for e.g. deleting misbehaving email-addresses. The main thing about this open discussion list is that it can be followed by everyone.
  - **The closed discussion list** is a mailing list to which nobody can subscribe or only mail-addresses can subscribe after review of a list moderator or list owner. Besides that not everyone simply can join, the mailing list history can only be seen by members which makes it possible to discuss sensitive and/or secret subjects.

Next the different open-source mailing list server software[3] which are interesting for the project are discussed. These are open-source or free mailing list servers which have at least a web interface for subscription to the lists and administration of the lists.

3.1 Main functionalities

The next sections describes the basic mailing list functionalities[18]. This does not mean that any mailing list have each of these functions but it should have. These sections are used to measure the completeness of each mailing list server software.

3.1.1 Subscribe

A user can subscribe to a mailing list by admitting at least their mail-address to a e.g. a web form. After the user asked for subscription it's optional that a list owner or list moderator must approve the subscription. If the subscription is approved the subscriber gets a mail with his password. It is optional for the list owner to enforce that the user must acknowledge the mail for finishing the subscription to prevent a subscription with a false email-address. If all these steps have been completed then the user email-address is subscribed. Sometimes subscribing is not allowed and members are subscribed by the list moderators and list owner.

3.1.2 Unsubscribe

A user can unsubscribe from a mailing list by e.g. a web form by admitting at least their mail-address. Sometimes it is impossible to leave the list because of the functional requirements e.g. the mailing list is the domain administrator mailing list to announce maintenance but for such a mailing list you probably don't need to subscribe to!

3.1.3 List the mailing list subscribers

This function shows every subscribed mail-address of a mailing list. This functionality should only be available to non-subscribers because this information is very attractive to spammers and therefore should never be available to regular subscribers of a list.

3.1.4 Show lists where you are a member of

This function shows all the mailing lists where a mail-address is subscribed to. As listing all mailing list subscriber this is information which should only by available to the owner of the mail-address and non-regular-subscribers.

3.1.5 Provide mailing list information

Providing information about the mailing list options should only be available to the list owner and system administrator because people should subscribe to a list because of it contents not because of it rules. Providing information about the mailing list statistics could be available to anyone even anonymous users because it gives the potential subscriber the possibility to judge the list first.

3.1.6 List the mailing lists of the server

Listing all the mailing lists of the server should be available to anyone even anonymous users except for the list which only can be subscribed to by the list owner or list moderator of the mailing list.

3.1.7 Change password

Every administrator and subscriber should be able to change their password since the given password is most times generated in the most strange sequences which are for humans hard to remember.

3.1.8 Receive help

The concept of mailing lists isn't always clear to everyone and the mailing list server software should be used by anyone. This means giving help to the people who need it is really important.

3.1.9 Digest mail

On busy mailing lists it is maybe better to prevent spamming behavior by sending digest mail messages. These are mail messages which contains a number of separate mail messages of the mailing list. This can reduce the number of mail messages send to the mailing list member mail addresses greatly.

3.1.10 Stop and start delivering mail

This function can stop and starts, on command of the user who owns the mail-address, the mail delivery to a specific mail-address. This is useful for busy mailing lists, think of a user who is away for over a month and returns to a mailbox with over a 1000 mail messages. Also some users have an auto-response function on their mailbox. Consider that the user is allowed to post messages to the list and that every mail he receives from the list has the sender address of the list (which is normal for a discussion list). If someone post a message the user that is away automatic generate an away response. If the user also gets it's own message then the user also gives an away response on it's own away response. This loop will quite fast fill up any mailing list server and therefore should be prevented by this function. Besides the stop and start delivering mail function an user most times is prohibited to spam (sending more then ... messages a day) a mailing list, which means that such a mail address with auto response will be blocked before doing to much harm.

3.1.11 Conceal your address

This function is for anonymous reactions or announcements without the return-address of the sender but rather a no-reply-address or mailing list email-address.

3.1.12 Do not receive own posts

This function can be set if posts send to the mailing list should not be forwarded back to the sender to confirm the post, which should be a natural thing for discussion lists.

3.1.13 Email plaintext or html

Some people are using terminal based mailboxes which do not support html, therefore it is sometimes useful to enforce plain text mail messages. HTML messages will then be converted into plaintext without the HTML style elements.

3.1.14 Attachments

Mailing lists with a great number of members will be happier not to send big mail messages with attachments because this will generate much data traffic on the mail server therefore it is smart to make it possible to drop all attachments or just attachments of a certain type and size.

3.1.15 Archive mails

Archiving mail isn't important for a newsletter but for a discussion list it is quite useful to read back a discussion before subscribing. Since the mail messages are archived it could be made available through multiply interfaces like HTTP, RSS, SOAP and REST which makes it easier to connect the mailing list mail messages to other services and administrative systems.

3.1.16 Search archive

When it is useful to archive the send messages it is also useful to add a search function e.g. for finding specific computer problem solutions within a long discussion list!

3.1.17 Personalization of mails

The personalization of mails can be useful for any kind of newsletters think about getting a personalized mail like Dear <user name>. The disadvantage of this function is that the mailing list does require additional user information from the describers.

3.1.18 Extra interfaces

Since mail and web interfaces aren't the only interfaces available it is sometimes smart to create multiply interface e.g. to approve subscriptions. A good example of such an interface is making the mailing list archive available by RSS[1] or making the administration interface available over a SOAP[2] webservice interface.

3.2 Administration of the mailing lists

In this section de administrative roles are described and where they are authorized to.[9]

3.2.1 Subscriber

A subscriber is someone who is subscribed to a mailing list, receives messages posted to the list, and may also post messages for distribution to the mailing list. Who may post, and how it is done, may be controlled by the list owners. A subscriber has no administrative power over list operation, but can change a few of his or her own subscription settings e.g. a subscriber could decide to receive message "digests" rather than individual postings.

3.2.2 List Moderator

List moderators are optional for a mailing lists. List moderators are assigned by the list-owner and can manage list subscriptions and postings for the list owner. They can be seen as a kind of elite subscribers.

3.2.3 List Owner

List owners can manage mailing lists, adding and deleting list members and making changes to how the list operates with an administrative webpage to setup their lists. While a list owner can manage a list without being a member of the list, each mailing list must have an owner who is responsible for configuration, maintenance and operation of the list. The list owner optionally can establish other owners and have others roles such as moderating the discussion or managing subscription requests. List owners may execute commands for their subscribers including adding and deleting subscribers. Each list must have at least one owner.

3.2.4 System Administrator

A system administrator is responsible for the operation of the mailing list server on which many lists reside. Administration tasks include installation and maintenance of the software and the computers where the mailing list server software runs on as well as creation and deletion of the individual lists. The system administrator is also entitled to act as the list owner of any mailing list on the server.

3.3 GNU Mailman

GNU Mailman[7] is a mailing list server software distributed under de GNU General Public License version 2. The project has been setup by Barry Warsaw but has been developed by the open-source community. GNU Mailman is besides Sympa the only mailing list server that is available on the popular Linux distributions (Gentoo, Debian 4.0 and Fedora 8 and 9). The current stable version 2.1 can be found on Launchpad[3] which can be retrieved by the Bazaar version control[4].

[1] RSS protocol: http://en.wikipedia.org/wiki/Really_Simple_Syndication
[2] SOAP protocol: http://en.wikipedia.org/wiki/SOAP
[3] Launchpad website: https://code.launchpad.net/mailman
[4] Bazaar website: http://bazaar-vcs.org/

Version 2.2 is the development version and version 3.0 can be seen as the experimental version.[8] GNU Mailman can been installed with different mail servers but need an extensive mail server configuration which most times can be found in the GNU Mailman documentation. GNU Mailman has a very simple architecture which is purely based on a queue where new mail messages can be dropped. Then GNU Mailman writes all the received messages away in files in the filesystem even the archive of mails exists out of simple html files. This file based architecture is great for (Unix) file based operating systems but isn't great for a scalable mailing list server software and easy to handle e.g. searching through the mail requires continuous opening of files which requires a lot of costly resources. A negative effect of this architecture reflects in the problem that GNU Mailman have in renaming it's mailing lists since renaming of content of files without a strict structure is very difficult.

A big advantage of GNU Mailman is that it has an very fast and bug free web interface which could even be loadable in text browser since it has no javascript and dynamic html. It is written in very clean html 4 code and the GNU Mailman web interface only supports a single system administrator with only a password without an username to adapt the lists e.g. creating lists.

3.3.1 Consult Joost van Baal about the current status of the GNU Mailman project

At the 6th of june 2008 I spoke with Joost van Baal when the project goal was still about improving GNU Mailman. Joost van Baal is an administrator of the University of Tilburg and he is familiar with the GNU Mailman project because of his involvement of building a OpenPGP implementation into GNU Mailman to validate mail messages send to a mailing list. This prevents the problem described in the mail server chapter: a bad configured mail server relaying mail messages for domains that they don't own or aren't responsible for. This makes it possible to send mail message to a mailing list with a spoofed sender address to anonymous spam protected mailing lists. To use OpenPGP keys GNU Mailman should know the OpenPGP keys of their list members

which can be enforced but block out unexperienced OpenPGP users.

3.4 Majordomo

Majordomo is developed by Brent Chapman of the Great Circle Associates[11]. It is written in Perl and works with the Sendmail mail server or Sendmail compatible mail server like Postfix, but only Sendmail is described in the documentation. Majordomo is being distributed under an own Great Circle Associates license but is extendable by several 3rd parties patches and plugins. But Majordomo has a downside it isn't updated for a long time and doesn't have it's own web interface. The web interface is available by the majorcool 3rd party web cgi application. Like the QMail mail server, the Majordomo mailing list server has many functionalities but need a lot of work to get it work e.g. the simple README file has already almost 600 lines of installation exceptions and the INSTALL has laterally the sentence:

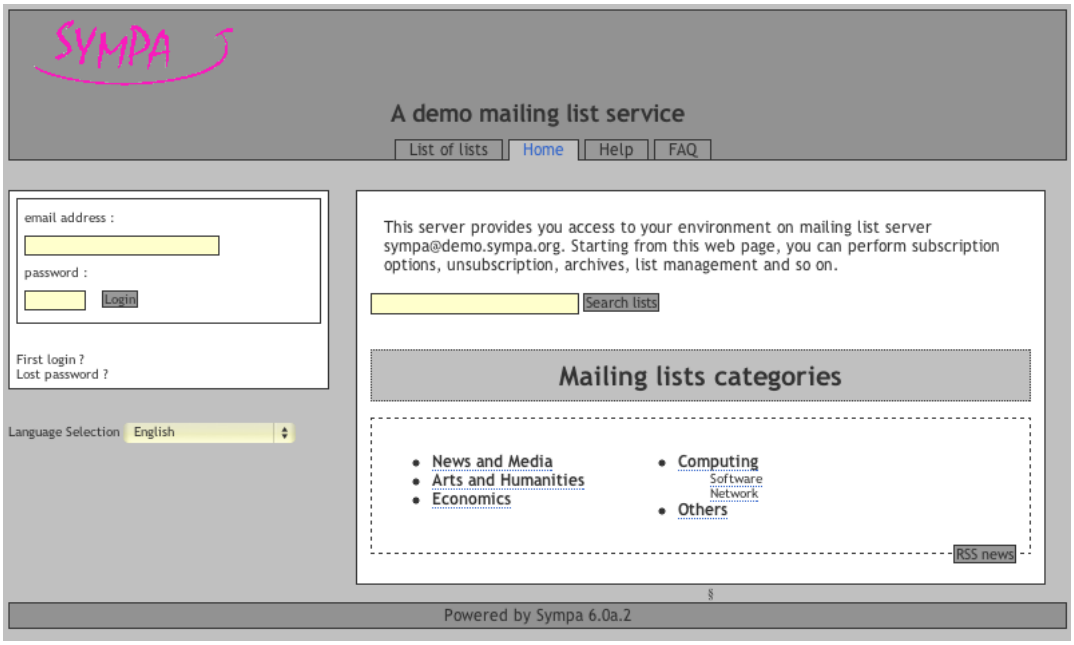
> The default installation of Majordomo, including the checks that config-test does, WILL NOT RESULT IN A SECURE INSTALLATION.

Therefore I consider Majordomo as a multifunctional[1] mailing list server software, which is still experimental.

3.5 ListServ

ListServ is the first mailing list server which supported subscription without human intervention, it's developed in 1986 by Eric Thomas. The mailing list server software was free and open-source until it became in 1993 commercially. ListServ now distributes still a free version but under a strict license of the commercial L-Soft company [5] and the non-opensource free version has a maximum of 10 lists with 500 subscriptions. The advantage of ListServ is that it works almost on every modern operating system and architecture and is superior to any other free mailing list server software because of it's years of development, but because it isn't really open-source and free (strict license) it doesn't apply to this report but still was important to mention because it shows that in the case of mailing list throwing a lot of money (at least 200 dollar per month) against the mailing list problems solves the problem. But since this gives a legacy problem e.g. has it's own built-in mail server and not everyone can afford it isn't the solution for this report.

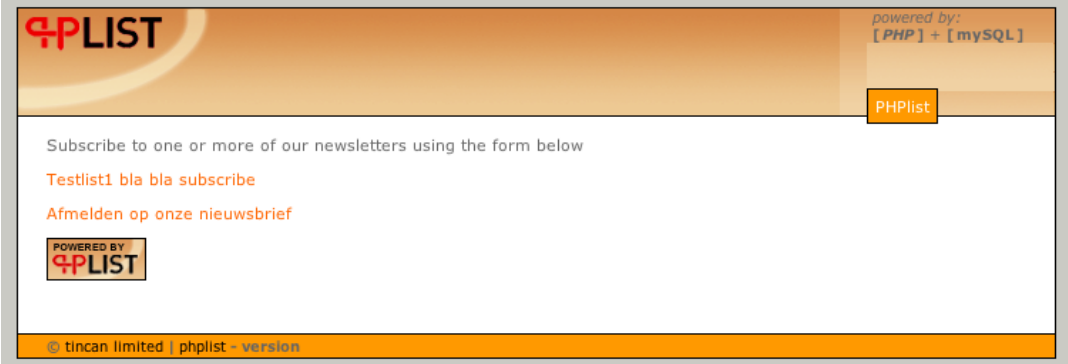
3.6 Sympa



Sympa[17] is an open-source mailing list server software distributed under the GNU General Public License version 2. Sympa supports only full featured categorized discussion lists without a system administrator. This means that every internet user can start a new list. Sympa has support for customized themes, Atom/RSS version 1 and 2 interface, SOAP webservice interface and personalized mails e.g. hello <user name >. But besides a wide range of functionalities Sympa does have a long list of perl dependencies where some only are retrievable from CPAN[5]. Sympa is the only mailing list server software that does sender authentication and encryption by the use of S/MIME[6]. Sympa has chosen for this technology because it can be used to authentication of the web interface to. The issue with S/MIME is that it isn't easy to create a certificate for S/MIME that is correctly signed, therefore it isn't really scalable to generate a certificate for each list member. Sympa is available on Redhat, Mandrake Debian, Freebsd and Solaris 7 but these are old and much adapted versions. The latest stable version of sympa can be tested by a demo on the sympa homepage[17].

[5]Comprehensive Perl Archive Network: http://www.cpan.org/
[6]Wikipedia S/MIME description: http://en.wikipedia.org/wiki/S/MIME

3.7 PHPList



PHPList[14] is an open-source mailing list server software distributed under de GNU General Public License version 2. It is designed for only newsletters and doesn't have an archiving functionality. PHPList can extensively tune the rights of the list owner, list moderator and system administrators. Also there is already the start of a built-in logging function. The unique functionality of PHPList is the possibility of subscribing to bulk lists, which are mailing lists which refer to multiply other mailing lists. PHPList has webpages which are theme-able by the use of CSS. Installing PHPList was a relieve because it has a modern database architecture. PHPList has only a dependency of a RDBMS like a mysql database and apache web-server with PHP and can be setup within a hour. But before installing PHPList the development and stable version can be tested from the PHPList homepage[14]. The disadvantage PHPList is only mails can be send to the mailing list by the administration web interface.

3.8 Compare the mailing list server software

In the next figure the main functions of mailing lists are compared to the found mailing list server software. In the table ListServ is left out because it doesn't fit the case of the report because it has a strict license which limits it's functionalities. Also Majordomo is left out of the table because it doesn't deliver a stable mailing list server software and web interface and therefore should never be used as a production mailing list server software. Before I go on to the table I must note that dropping these mailing lists from the final comparison table doesn't make ListServ and Majordomo a bad product but they aren't useful to compare for the case of the NLnet foundation, described in the next chapter. They are left in the report to keep a wide vision on the commercial, new and recovering mailing list server software.

Functionalities	GNU Mailman	PHPList	Sympa
Ease of install	⋆⋆	⋆⋆⋆	⋆
Newsletter functionality	⋆	⋆⋆⋆	⋆⋆
Open discussion list functionality	⋆⋆	⋆	⋆⋆⋆
Closed discussion list functionality	⋆⋆⋆	⋆	⋆⋆
Subscribe	•	•	•
Unsubscribe	•	•	•
List the mailing list subscribers	•	•	•
Show lists where you are a member of	•		•
Provide mailing list information	•		
List the mailing lists of the server	•	•	•
Change password	•		•
Receive help	•		
Digest mail	•		•
Stop and start delivering mail	•		•
Conceal your address	•	•	•
Do not receive own posts	•		
Email plaintext or html	•		•
Attachments	•		
Archive mails	•		•
Search archive			•
Personalization of mails		•	
Extra interfaces		Experimental RSS	SOAP and RSS

The more ⋆'s a mailing list server software have, the more suitable it is for the task.

In the above table you can see how each mailing list server has his strength and weaknesses. But like most software it is better to try the mailing list server software then to read about it therefore I will describe in the next section my experience and opinion about each compared mailing list server software.

3.8.1 PHPList

PHPList is the only mailing list server software that uses a pure modern database architecture. The database is setup in logical tables that are easy to comprehend with names like phplist_user_blacklist or phplist_eventlog. PHPList is especially build for newsletters mailing lists, therefore there are no accounts for subscribers only for administrators.
The strong points of PHPList are:

- The subscribe page is editable (HTML and CSS) from an administrative interface.
- The users can be extended by self defined attributes which can be used in the (personalized) newsletters.
- There is an extensive administrator framework, which support administrators with different right attributes and super administrators for managing the administrator group.
- Because of the database architecture mailing list information can be easily inserted and backup-ed.
- PHPList is the only mailing list server that supports (event) logging. It is still primitive but it is a good start towards a mature logging system.

.
.
The weak points of PHPList are:

- There is no real archiving functionality. Therefore it is rather a mailing list manager then a mailing list server.
- PHPList isn't standard coupled with the mail server to receive mail for it's mailing lists. There are hacks to accomplish this but they aren't supported by the PHPList developers. Officially PHPList can only send mails from their own administrative webpage.

3.8.2 Sympa

Sympa is the open discussion mailing list divided in categories. Sympa is made for online discussions between social groups about any subject.
The strong points of Sympa are:

- On an University server it will probably make a great informal discussion (forum like) mailing list.
- Because of the well defined categories it is a mailing list server software made for a big number of mailing lists
- The main navigation is very useful because it shows everything you want to see e.g. the subscribed mailing lists.
- Creating a new mailing list is easy because of the standard templates:
  - discussion list
  - hotline
  - html news letter
  - intranet list
  - news letter
  - private working group
  - public web forum
- Sympa offers a wide range of several e.g. a good RSS interface to the posted messages and LDAP, CAS and Shibboleth authentication support.
- Sympa is the only mailing list who offers efficient search capabilities to it's archives

The weaknesses of Sympa are:

- The mailing list server software doesn't feel like it belongs to the domain it is installed for e.g. it hasn't space for logo's.
- The web interface of Sympa by long isn't as good as the GNU Mailman and PHPList interface (graphical and functional).
- It is easy to create new lists, but it is hard to configure them because of the messy menu setup.
- Sympa can be rather a forum were you share data and messages then a discussion list.
- Sympa misses the system administrator role and therefore can never be used in big environments because illegal or bad content can only by removed by people who can edit the database.
- There is no way to block out or delete accounts.

3.8.3 GNU Mailman

GNU Mailman is the choice for the highly stable controlled mailing list which offers performance. The strong points of GNU Mailman:

- GNU Mailman has a clean HTML 4 webpage, as the only interface, with most of the help information integrated in the webpage and it will probably runs in every browser (except from mosaic).
- It has an archive which is purely based on separate HTML files, which makes browsing through the archive very efficient.
- GNU Mailman works with almost all mail servers, because it is based on mail queue's.

The weaknesses of GNU Mailman are:

- GNU Mailman has no search function on the mailing list archive. Also this functionality is hard to integrate within GNU Mailman because the archive is directly saved into HTML files.
- Because of not using a database it is hard to develop new functionalities.
- The GNU Mailman has simple webpages without the use of Javascript, CSS or DHTML. The effect of this is that the interface is compatible with all the browsers but is also less efficient.

3.8.4 Mailing lists and mail server interfaces

The interface between mailing lists server software and mail server exist out of three parts:

- A program to inject mail into the mail server e.g. Sendmail or a smtp server.
- A program to configure the relay to the mailing list mailboxes. This is always implemented by the use of the alias file (/etc/aliases) in the current mailing list server software.
- A location to receive mails on e.g. a mail queue in the mail spool or a POP3 or IMAP address.

Chapter 4

The NLnet Foundation Case

In this chapter the NLnet Foundation case is described:

- Who is the NLnet Foundation?
- What are the issues of the NLnet Foundation with the current GNU Mailman mailing lists?

4.1 Who is the NLnet Foundation?

The non-profit NLnet Foundation was formally started in 1989, but incorporates networking activities which go back as early as 1982. The foundation (in Dutch called "Stichting NLnet" or shortly "NLnet") has played a major role in raising the so-called pan-European "UNIX" Network and the commercial and public internet network provision in the Netherlands.[12]

Some NLnet milestones are:

- The first internet backbone in the Netherlands.
- The first local dial-in and ISDN infrastructure with full country coverage and the definition and implementation of a low cost connectivity structure.

These activities caused Amsterdam to become the major exchange point for European internet traffic. People like Piet Beertema, Daniel Karrenberg, Ted Lindgreen, and the employees of NLnet have played a major role for NLnet and the internet in Europe.

In the summer of 1997, the Foundation sold its commercialized internet provision activities to UUNET (the internet subsidiary of WorldCom), which was later renamed into Verizon.

From this money the NLnet Foundation financially supports organizations and people that contribute to an open information society. It funds e.g. software, events and educational activities[1]. Also the NLnet Labs Foundation, which has close relations with the NLnet Foundation, financially supports and shepherd student who build open-source software[2].

4.2 What are the issues of the NLnet Foundation with the current GNU Mailman mailing lists?

The NLnet Foundation has project all over the Netherlands and it uses mailing lists to communicate with the communities they financially supports and sending news letters to the press. The NLnet Foundation uses for this an Unix(-based) environment with a Fedora 8 server with the GNU Mailman mailing list. After speaking with Michiel Leenaars of NLnet the issues of GNU Mailman became clear:

[1] Vrije Universiteit van Amsterdam - IIDS: http://www.iids.org/
[2] NLnet Labs homepage: http://www.nlnetlabs.nl

- The administration of GNU Mailman isn't optimal because GNU Mailman only supports a single system administrator. It has no exact framework to manage the rights of the system administrators, list owners, list moderators and subscribers. It misses a attribute or role based authorization system.
- The current mailing list server software does not have a logging system for changes to the mailing lists. This functionality is needed to trace faults or abuse of the mailing lists.
- It costs to much effort to administrate the multiply mailing lists with GNU Mailman, because the web interface of GNU Mailman isn't setup to administrate more then one list at the same time.
- A better strategy is needed for sub-lists and super-lists, including dealing with the resulting password reminders and authorization to modify the sub & superlists of the mailing lists e.g. if a message has no known OpenPGP key then it is automatically spam and should be dropped.
- GNU Mailman can only manage the mailing lists by the web interface. It would be an improvement to manage the mailing list by e.g. a SOAP webservice interface or a REST interface.
- GNU Mailman is a piece of software with it's own technical components. It should fit the NLnet Foundation to combine a mailing list server software with user management system like a LDAP software. This means that subscribers not only subscribe their mail-address but also other personal information like their telephone number or personal interests. So these information can be used for interesting offerings like events.
- The developed OpenPGP functionalities of Joost van Baal should be integrated within any solution because privacy is important for the community members that the NLnet Foundation support. This new functionality of Joost van Baal also gives posibilities to improved administration
- For any new solution or improvements at least the current functionality of GNU Mailman should remain. Also the solution that is found should be as secure as GNU Mailman and should have atleast the performance of GNU Mailman because the whole community (of NLnet) is dependent on GNU Mailman.

Chapter 5

Conclusion

In the report there are different mailing list servers software handled which each their own advantages and disadvantages. It's clear that no open-source and free mailing list server software is really optimized for the discussion lists **and** news letters. They are all built with a special purpose:

- Open discussion lists where everyone can join and start a discussion (Sympa).
- Closed discussion lists where groups can discussion their project (GNU Mailman).
- A news letter mailing list server to send personalized news letters to specified mail-address lists (PHPList).

5.1 The NLnet Foundation

The NLnet Foundation is searching for a complete multi functional mailing list server which have all the functionalities of the researched mailing list server software: the stability and performance of GNU Mailman, the database architecture, rights administration and personalization of PHPList and the categories, RSS, SOAP and authentication interface of Sympa. Unfortunately such a mailing list server software like ListServ is only available for a lot of money and has different legacy problems and therefore the best solution for the NLnet Foundation should be the development of a new mailing list software from either extending PHPList, because this mailing list server software is easy to extend because of it's basic (database) architecture or developing a complete new mailing list server software with the support of well known open-source libraries like the libesmtp library for implementing the client side SMTP protocol for the MTA component. Personally I think that building a new mailing list server software is the best solution because then the philosophy behind such a new server software could be setup right from the start, focused on having a good administrative interface with the discussion list and news letter functionalities. This new mailing list server software can be build by the NLnet Labs Foundation who have active open-source project like DNSSEC[1] and are closely related to the NLnet Foundation.

[1] DNSSEC Homepage: http://www.nlnetlabs.nl/projects/dnssec/

5.2 Creating a better implementation of the mailing lists within the mail server architecture

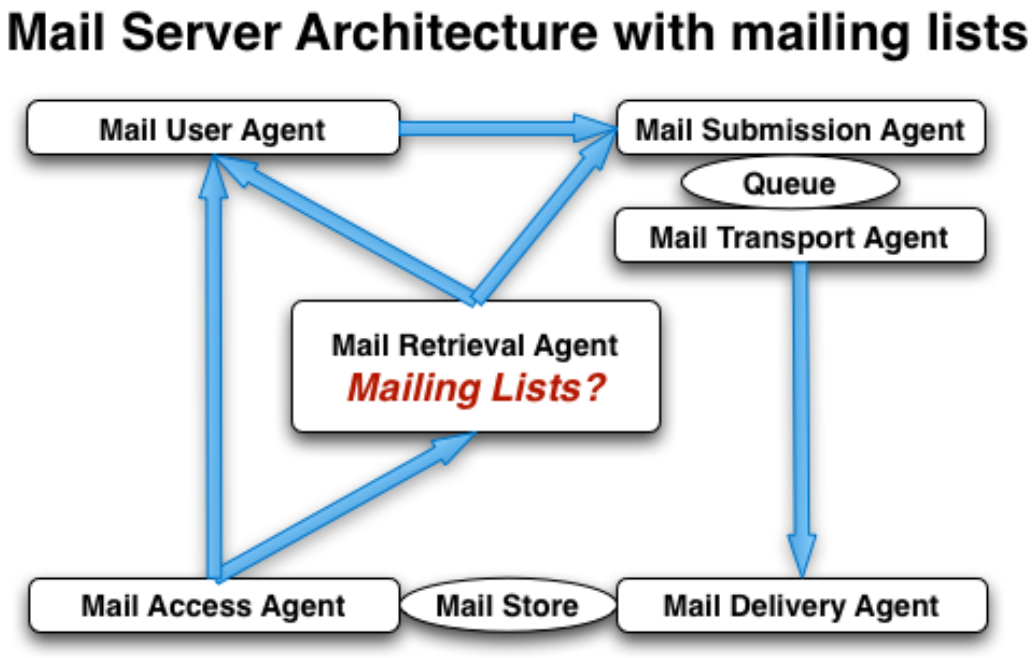
The main problem in the current mailing list server software implementations is the interface with the mail server. The mailing list server software is so far integrated with the mail server software that the question arise why the mailing list server software isn't part of the mail server? Most of the functionality already exists in the mail server architecture except of the mailing list configuration interface. The biggest problem of mailing lists server software is making clear to the mail server which mails should be relayed to the mailing list back-end components. On many mailing list server software this still must be done by an administrator who has configuration access to the mail server, because the current used mail servers aren't build to have a variable alias configurations. This problem can be resolved by integrating the mailing list server software in the Mail Retrieval Agent of the mail server architecture so the mailing list server software can use the accounting facilities of the mail server. But to integrate the mailing list server software, a mail server should be adapted to separate the incoming and outgoing mail-addresses. This is required because the current mail server implementations are developed to give a mailbox to every locally known mail address which isn't relayed. While rebuilding the mail server also the data architecture could be upgraded to a purely database architecture, which results in a more scalable mailing list/mail server software which can grow with the evolution of the internet e.g. this gives the mail server administrator the possibility to locate the data on a different (database) server also the current RDBMS based database software offers much better facilities for searching the archives, retrieving statistics, concurrency and backing up the mailing list members and mail messages then the current hard-disk tools.

Chapter 6

Future Work

In this chapter my vision on the new mail server is explained, a mail server which delivers more mail based services then the traditional mail servers like QMail and Postfix who are more focussed on the performance of the mail server, which in my opinion shouldn't be an issue with the currently available hardware and redundancy. The second part of this chapter is about integrating the mailing list (administration) functionalities into the SMTP protocol[1] or a web interface.

6.1 The new mail server



Talking about the mail server architecture is difficult and hard because it is a proven architecture, it is over 30 years old. In my opinion the current mail server architecture should **not** change because the current architecture have a description which support the mailing list server functionalities except from a configuration interface for the mailing list and the mail server itself.

[1]SMTP protocol: http://tools.ietf.org/html/rfc821

But before talking about the configuration in the SMTP protocol or the web interface, first the new mail/mailing list server software should be defined.

6.1.1 Mail User Agent

The MUA is a component on the system of the client, which is based on the standardized interfaces of the mail server like POP3, IMAP and the local mail sender like the Sendmail binary or internal MS Outlook (GUI) component. This means that the new mail server implementation shouldn't have to posses a design for the MUA, because this component is installed on the user side (users choice and taste) and their are enough good open-source and free MUA's available.

6.1.2 Mail Submission Agent

The MSA is implemented in three ways:

1. The GUI mail client on the user's system has almost always integrated this component and therefore isn't part of the new mail server and therefore shouldn't be part of the new design.
2. The terminal mail client is most times located on the same system as the mail server and uses most times the Sendmail program to send mail. This means that for backwards compatibility this interface should be faked as done by Postfix. Also other mail servers their local program interfaces should be faked if required.
3. A programming interface to e.g. relay the mail from the Mail Retrieval Agent e.g. error-mails and mailing list mails. But also this interface can be used for third party application to inject mail into the mailing process.

The mail submission agent can receive mail from any local program, mailing list or library therefore the mail should be authenticated before sending the mail message. Also the mailing lists can relay's mail messages to mail-addresses of domains it isn't responsible for these mails should be authenticated for authorization before sending them to a separated outgoing mail-addresses listed in a RDBMS based database (accounts) with the defined authorizations and rules. Implementing the MSA can be done on thread or process based interfaces/programs which should write, when accepting, the mail messages to the queue. The result of this design is that the queue should remain stable on many concurrent processes.

6.1.3 Queue

In the current mail servers the queue is a file, but with a new mail server this should be a RDBMS based database to decouple the data from the mailing process. This gives the system/network administrator the possibility to centralize the data into e.g. a scalable database system which is better in concurrency (ACID[2]) and most database software deliver utilities which a normal filesystem can never deliver. The second reason to decouple the data from the mailing process is for measuring the mail server statistics and optimizing the mail process, because databases giving extended features for counting records and filtering data on an efficient way.

6.1.4 Mail Transport Agent

The MTA should consist out of a single process or a worker thread group which sends the mail messages from the queue by the SMTP protocol, because a separate thread/connection for every queued mail message will result in a spamming like mailing pattern, when sending 100 mail messages the MTA will connect 100 times to maybe the same mail server. The problem with this design is that the MTA doesn't know when a MSA posts a mail message on the queue. Therefore it is useful to notify the MTA when posting a message which is useful to optimize the MTA behavior

[2]ACID database transactions: http://nl.wikipedia.org/wiki/ACID

but break with the mail server architecture like most optimizations do e.g. the MTA only starts sending if the queue is bigger then 50 mail messages or if a message is older then 10 minutes. The rest of the time the MTA (workers) can sleep to preserve system resources.

6.1.5 Mail Delivery Agent

The MDA receives messages by the SMTP protocol or by inserting a mail message by the use of the MDA library interface when the mail can be delivered locally. After it is received the mail message is validated e.g. the incoming mail-address is valid to relay or is of a incoming mail-address described in the RDBMS based database. If the mail message is accepted and no direct failure is returned the mail is stored into the mail store.

6.1.6 Mail Store

The mail store originally existed into /var/mail directory on Unix-based operating systems. In the new mail server this should exist in the RDBMS based database but still could be implemented e.g. updating only these mailbox files when they are being accessed which can be monitored by file systems hooks[3] which exists in most modern operating systems integrated for security reasons like virus scanners and auditing validation modules.

6.1.7 Mail Access Agent

Since the backwards compatible /var/mail directory is updated when accessed the MAA can be accessed by the terminal based MUA's. The rest of the non-local MUA's can access their mail by the MAA protocols (POP3 and IMAP protocol) and the MRA can access the mail messages by the internal MAA library.

6.1.8 Mail Retrieval Agent

The MRA has 3 functions:

- The MRA relays messages for other domains through the MSA that are described by the RDBMS based database incoming mail-addresses.
- Error and bounce mail messages that are send back through the MSA to the sender of a mail message e.g. if a mail loop is detected.
- Mailing list mail messages that are posted and then are forwarded by the MRA after authentication to multiply mail-addresses by the use of the MSA, which can be seen as just a complex alias function that relay's mail messages. The mail message it self can remain in the mailbox and the mailbox can then function as the archive for the mailing lists.

Since like the MTA the MRA, wants to know when there are mail message in the mail store which are interesting for him the MDA should notify the MRA on mail message where the MRA is interested in or subscribed to.

6.1.9 MRA Mailing List authentication

Posting on mailing lists on current mailing list server software can't ensure the authenticity of the sender, secrecy of the mail messages or the integrity of mail messages. Sympa supports S/MIME but this requires a lot of extensive configuration. To ensure on an easy way that the posted message comes from the right sender OpenPGP can be used. OpenPGP gives Pretty Good Privacy in the form of integrity validation and optional encryption. In the new mail server the outgoing mail-address list should contain keying information e.g. OpenPGP to ensure members of a list can only

[3] Examples of the filesystems hooks: http://www.usenix.org/events/sec02/full_papers/wright/wright_html/node14.html or http://technet.microsoft.com/en-us/sysinternals/bb545046.aspx

send a mail message if they are authorized and send secure messages to the list mail-address. By enforcing OpenPGP the privacy of a discussion can be guaranteed, but this forces the list members to use OpenPGP which is not always a possibility.

6.2 Why not to extend the SMTP protocol

Because so many people are using these mailing lists and they exist for over 20 years it is maybe interesting to discuss the implementation of the mailing list functionality in the SMTP protocol because the MDA is the most logical place to integrate the mailing list functionality. But is shouldn't be integrated into the SMTP protocol because it is an extension on the basic functionality of a mail server: sending and receiving mail. Also if every function of the current mailing list server software is integrated into the SMTP protocol then the protocol gets very messy and heavy weight and therefore shouldn't be integrated within the SMTP protocol.

6.3 The web (administration and configuration) interface

Since the administration and configuration of the mailing list isn't integrated within the mail server architecture it should be made available by an alternative interface, a web interface because this is supported by every user with a modern browser or operating system. But like Courier mail this interface should be a plugin or module which only have dependencies on the mail process like the model-view-controller architecture[4]. This means that the mail server should have an interface to which the web interface can connect to, to configure the mailing lists. I think the best way to implement this interface is by integrating it into the MAA by the use of the SOAP protocol which will be some work but because SOAP is a self describing protocol it will save the effort of creating an interface for every separate programming language e.g. ASP, JSP and PHP then a website can be build based on the SOAP interface. A website can then be an interface to the mailing list for the user to subscribe and unsubscribe, but can also be an interface to the mailing list archive. Also the SOAP interface makes it possible to reformat the mail message format because the information isn't already transformed in HTML and therefore the website can also offer the archive e.g. by the RSS protocol.

6.4 AAA

Now the new mail server has a separated MVC architecture (database, mail server process, user interface e.g. graphical webpage) the AAA[5] formula should be reviewed.

6.4.1 Authentication

Within the mail server architecture the components are defined but not the authentication to access the components. There are some possibilities like the authentication of the IMAP, POP3 and SMTP protocol, OpenPGP and S/MIME but I think it is necessary that these authentication methods should be extended to the SOAP web service interface and the webpage, because this prevents that people write insecure user authentication database systems and give inexperienced developers access to advanced authentication methods like a Shibboleth single sign-on[6]. This keeps the e.g. web interface (necessary for the mailing list functionality) light and easy to create or even integrate.

[4]MVC: http://msdn.microsoft.com/en-us/library/ms978748.aspx
[5]Authentication, Authorization and Accounting: http://en.wikipedia.org/wiki/AAA_protocol
[6]Shibboleth: http://shibboleth.internet2.edu/

6.4.2 Authorization

In this report the mail server rules are skipped but are necessary to harden the security of the mail server. These e.g. Postfix rules can block out mail-addresses or rewrite mail-addresses, they are written to fine tune the mail server and are located in the a local directory of the mail server which of course isn't practical. These rules should be available in the new mail server for backward compatibility by the use of a module or plugin. It is a nicer solution to separate the rules from the mail-addresses and create a variable, rules reference and function (drop, delete, forward, accept, OpenPGP_sign, etc.) based syntax to describe universal rules which then dynamically can be fitted to a mail-address or mail-address group which makes it possible to create a much more precise authorization scheme e.g. one of the necessary rules should be the incoming mail-address and outgoing mail-address rules. This means that a new user can do nothing until it's rule based allowed to receive and/or send mail by the outgoing and/or incoming mail-adress rule. This deny before allow scheme has proven to be very useful and secure for e.g. firewalls and web servers like Apache.

6.4.3 Accounting

Since now the authentication and authorization is separated the only task at hand is to create an user database or interface to an user database like LDAP. But this is the new problem of the new mail server: the news letter mailing list requires to personalize mail messages but the attributes in user databases like LDAP are already defined and doesn't always deliver configurable attributes. I think the best solution for this problem is not to copy all records into the new mail server RDBMS based database but rather to make a reference to the account and save the reference, authorization rules and variable attributes for the mailing list in the new mail server RDBMS based database. This makes it possible to support multiply user database resources but again raises the problem of when to delete a reference because the mail server can't expect of an user database source to share this information with him.

6.5 Future work conclusion

The main point I as author wanted to prove is that their is a great potential to improve the current mail and mailing list servers. I think the old mail server architecture still fits the needs of the current mail user but the reason of writing this report shows that the current software implementations of the mail servers are out dated and not prepared to be managed without local access and therefore will be forced to upgrade soon. The main goal of this new mail server should be getting out of the messy mail rules syntaxes and local queue's so extending the basic mail server architecture with e.g. mailing list functionalities should be easy and not a work around like the current mailing list server implementations. On the new mail server giving the mailing list new functionalities is nothing else then tuning the SOAP interface, because all the data that is needed can be easily retrieved from a RDBMS based database. Now the only questions remains who will create the new mail server and making the current loosely coupled mail (based) software obsolete?

6.6 Getting even more scalable

For people who are searching for even a more scalable design could couple each new mail server process component by the use of Message Queue Middleware. This makes it possible to make every component redundant which gives a better reliability and performance which is now possible because of the MVC architecture which separates the data. This solution also solves the performance excuse of administrators for still using mail servers like e.g. QMail or Postfix.

Bibliography

[1] D. B. Chapman. Great circle associates, report of the lisa, october 1992, long beach ca, majordomo: How i manage 17 mailing lists without answering "-request" mail.

[2] Double precision inc., courier mail server. Available from: http://www.courier-mta.org [cited 17 June 2008].

[3] emailman: Unix mailing list servers. Available from: http://www.emailman.com/unix/mailinglist.html [cited 17 June 2008].

[4] University of cambridge, exim. Available from: http://www.exim.org [cited 17 June 2008].

[5] L-soft listserv. Available from: http://www.lsoft.com/products/listserv.asp [cited 17 June 2008].

[6] Wikipedia, electronic mailing list. Available from: http://en.wikipedia.org/wiki/Electronic_mailing_list [cited 17 June 2008].

[7] Gnu mailman. Available from: http://www.gnu.org/software/mailman/ [cited 17 June 2008].

[8] Gnu mailman, developer wikipedia. Available from: http://wiki.list.org/display/DEV/Home [cited 17 June 2008].

[9] University of washington, gnu mailman guide. Available from: http://www.emailman.com/unix/mailinglist.html [cited 17 June 2008].

[10] Mailradar. Available from: http://www.mailradar.com [cited 17 June 2008].

[11] Great circle, majordomo. Available from: http://www.greatcircle.com/majordomo/ [cited 17 June 2008].

[12] Nlnet foundation. Available from: http://www.nlnet.nl [cited 17 June 2008].

[13] Openpgp message format rfc 4880. Available from: http://www.ietf.org/rfc/rfc4880.txt [cited 17 June 2008].

[14] Phplist. Available from: http://www.phplist.com [cited 17 June 2008].

[15] Qmail. Available from: http://www.qmail.org [cited 17 June 2008].

[16] Sendmail consortium, sendmail. Available from: http://www.sendmail.org [cited 17 June 2008].

[17] Sympa. Available from: http://www.sympa.org [cited 17 June 2008].

[18] P. Tsier. Report of the university of waterloo, a comparison of three mailing list managers: Lyris vs. sympa vs. mailman.

[19] University of amsterdam. Available from: http://www.uva.nl [cited 17 June 2008].

[20] W. Venema. Postfix. Available from: http://www.postfix.org [cited 17 June 2008].

Appendix A

Planning

A.1 Preparation

- Create the planning
- First setup of the report

A.2 Week 1 - Exploratory research

- Research GNU Mailman. What is GNU Mailman and how does GNU Mailman exactly work?
  - Read the documentation
  - Read forums and reviews about people who uses and implemented GNU Mailman
  - Consult Joost van Baal at the University of Tilburg (Netherlands)
    - Status of the GNU Mailman project
    - Software Architecture
- Find out the issues that the NLnet Foundation have with their current mailing lists (Michiel Leenaars)
- Compare the administrative functionalities of GNU Mailman with other popular mailing list server software
- Identify the (GNU Mailman) mailing list security issues

A.3 Week 2 - Main research

- Research the popular mail server software useful functionalities
- Research the found popular mailing list server software

A.4 Week 3 - Reflection

- Analyse the results of the researched mailing list servers
- Define the improvements to make to the mailing list server software (GNU Mailman)

A.5 Week 4 - Closing the project

- Make the report
- Present the project results at the University of Amsterdam

Appendix B

Original Research Project Description

B.1 Official Dutch Research Project Description

Usability- en efficientieverbeteringen Mailman
GNU Mailman is achter de schermen een van de belangrijkste communicatiemiddelen van onze tijd, en wordt gebruikt voor het beheren van ettelijke honderdduizenden mailinglijsten waar dagelijks vele tientallen miljoenen mensen gebruik van maken. Het beheer van Mailman is niet optimaal ingesteld op meerdere lijsten tegelijk en op beheer door moderators buiten het web om, waardoor beheerders van lijsten veel tijd kwijt zijn met het onderhoud van de abonnees en andere taken. Ook zijn er een aantal securitytekortkomingen te identificeren. De opdracht is drieledig:

- Vergelijk de beheersmogelijkheden van Mailman en een aantal andere populaire mailinglijst-servers, en doe suggesties voor verbeteringen van de Mailman lijst.
- Bouw een remote interface voor het beheer van Mailman (bijvoorbeeld op basis van SOAP)
- Demonstreer de succesvolle werking van de API aan de hand van een commandline client (vergelijkbaar met de tool listadmin) en een GUI-versie.

B.2 The translated in English Research Project Description

Usability and efficiency improvements Mailman
GNU Mailman is behind the scenes one of the most important means of communication of our time. It is used for managing several hundred thousand mailing lists which are daily used by millions of people. The administration of Mailman isn't setup optimal to use several lists at the same time and for managing these lists by moderators outside the web. Therefore it costs the administrators a lot of time and money to manage list members and other tasks. Besides this there are also some security issues which should be identified. The assignment exists out of three parts:

- Compare the administrative functionality of Mailman and a couple of other popular mailing list server software. Do a suggestion based on this research to improve GNU Mailman list.
- Build a remote interface for managing Mailman e.g. based on SOAP.
- Demonstrate the API of the remote interface by the use of a GUI and commandline client (based on the tool listadmin)

B.3 Changing the project result

Half way the research it was clear that the world of mailing list server software was to complex to easily find a solution which can be implemented in GNU Mailman within four weeks. Also there seems to be more then one mailing list server as scalable and suitable as GNU Mailman which fits within the case of the NLnet Foundation. Based on these facts it is decided to research each relevant mailing list server software to provide an overview to decide how much every mailing list server software is worth. From this result the best main functionalities can be concatenated and can be used to build a new or more optimized mailing list/mail server software.