Mailman: The GNU Mailing List Manager (Extended Abstract)

John Viega
Reliable Software Technologies
viega@rstcorp.com

Barry Warsaw and Ken Manheimer
Corporation for National Research Initiatives
bwarsaw@cnri.reston.va.us
klm@cnri.reston.va.us

Abstract

Electronic mailing lists are ubiquitous community-forging tools that serve the important needs of Internet users, both experienced and novice. The most popular mailing list managers generally use textual mail-based interfaces for all list operations, from subscription management to list administration. Unfortunately, anecdotal evidence suggests that most mailing list users, and many list administrators and moderators are novice to intermediate computer users; textual interfaces are often difficult for such people to use effectively.

This paper describes Mailman, the GNU mailing list manager, which offers a dramatic step forward in usability and integration over other mailing list management systems. Mailman brings to list management an integrated Web interface for nearly all aspects of mailing list interaction, including subscription requests and option settings by members, list configuration and Web page editing by list administrators, and post approvals by list moderators. Mailman offers a mix of robustness, functionality and ease of installation and use that is unsurpassed by other freely available mailing list managers. Thus, it offers great benefits to site administrators, list administrators and end users alike. Mailman is primarily implemented in Python, a free, object-oriented scripting language; there are a few C wrapper programs for security.

Mailman's architecture is based on a centralized list-oriented database that contains configuration options for each list. This allows for several unique and flexible administrative mechanisms. In addition to Web access, traditional email-command based control and interactive manipulation via the Python interpreter are also supported. Mailman also contains extensive bounce and anti-spam devices.

While the features discussed in this paper are generally improvements over all mailing list packages, we will focus our comparisons on Majordomo, which is almost certainly the most widely used freely available mailing list manager (MLM) at present.

Extended Abstract

[Reviewers: Images showing the system's interface have been omitted from this extended abstract in the interest of space.]

Introduction

Electronic mailing lists often have humble beginnings: a user collects a list of email addresses of like-minded people, and these people begin sending email to each other using an explicit distribution list. This type of simple list is fairly easy for a novice user to start, and many end-user mail applications let the user set up such distribution lists.

Often however, lists will grow from this original distribution list, and as they do, explicit lists of addresses become extremely unwieldy. List administrators quickly tire of adding and removing subscribers manually, and answering email pertaining to the list. As a result, administrators generally turn to MLM software to automate the process.

The first generation of mailing list managers automated tedious administrative functions such as subscribing and unsubscribing from mailing lists, as well as many of the other common requests, such as getting background information on a list, and getting a list of subscribed members. They also

allowed for lists to be administrated via an email interface, so that list administrators would not need to have direct access to the machine on which the list software ran. However, this generation of MLMs has traditionally been quite complex; users are often unable to figure out how to get on or off a list, leading to many messages along the lines of "please unsubscribe me". List administrators often find it time consuming and difficult to perform administrative tasks by email, especially when editing special message headers is required, as is the case with approving held messages in Majordomo. In fact, many of the most popular mail user agents (MUAs) of today (including the Netscape mail reader) make it fairly difficult for the user to edit arbitrary headers. System administrators frequently have a difficult time setting up such software, especially when many commonly desired features such as list archiving are only available as third-party add-ons, if at all.

Mailman is helping to pioneer the second generation of free mailing list managers. While even three years ago email messages were the only reasonable user interface that would make mailing lists accessible to every Internet user, today the World Wide Web is generally considered ubiquitous. In fact, the Web offers a high level of familiarity and usability for mailing list users, who are typically at least as experienced, if not more so, at browsing the Web. Considering the frequency with which most users interact with the administrative interface of a mailing list, using a Web form that presents all the options is much less of a burden than having to learn or relearn an arcane syntax for mail commands. Ironically enough, instructions for interacting with mailing lists are commonly found on Web pages.

Functionality Overview

Mailman's primary distinction from other mailing list managers is its Web interface, which is discussed in the following section. However, in addition to having all of the features people expect from a MLM, such as digests and moderators, Mailman integrates a rich set of general-purpose features.

One such feature is automatic bounce handling. Much like the SmartList mailing list manager [Sma98], Mailman looks at all delivery errors, and uses pattern matching to figure out which email addresses are. By default, once the number of bounces from an address reaches a configurable threshold, the address becomes disabled, but not removed. The administrator is then sent a message and can decide whether the address should be re-enabled or removed. However the administrator could set the list to be more aggressive, automatically removing addresses after a certain number of bounces.

We have examined several thousands of bounce messages received while administrating Majordomo-based lists, from which we determined the current set of patterns. Applying these patterns to bounces has a two-fold benefit: we do not need to answer "`-request`" mail, and we rarely need to handle bounce disposition manually. On large lists, this automation can be important, since bounced email can easily produce 10 to 100 times as much email as actual list submissions [Lev97].

Mailman also contains several anti-spam devices that significantly reduce the amount of spam that reaches end users. First, member addresses are never presented in a form that traditional spammer-launched webcrawlers will recognize. Second, Mailman's delivery scripts apply a number of configurable and extensible filters to the incoming message, such as requiring the list address to be named in the `To:` or `Cc:` fields, or rejecting messages from known spam sites. These, as well as other measures, have proven to be very effective in preventing most spam from reaching the list, while still allowing valid messages to propagate.

Mailman also offers integrated support for many things that have traditionally been provided in add-on packages, or have required hacking with other MLMs. Mailman is distributed with such features as archiving messages sent to a list, fast bulk mailing by multiplexing SMTP connections, multi-homing for virtual domains and gating mail to and from NNTP news groups. Mailman also uses the GNU *autoconf* tool to make the setup process easy; in contrast, the Majordomo maintainers admit that Majordomo is extremely difficult to install [Bar98].

Thus, Mailman is able to provide a system administrator with a mailing list manager that is not only easy to install, but also is easy to use at every

level, and includes the major pieces of functionality a list administrator might want without requiring additional searches and downloads.

Web Interfaces to Mailing Lists

While Mailman does provide Majordomo-like mail-based commands for compatibility, we downplay it, as we feel that a good Web-based user interface is much more desirable to the majority of users. Our Web-based interface allows for full access to all of Mailman's features, including subscription and option requests, browsing lists on the same (potentially virtual) host, viewing Web-based Hypermail-like archives, etc.

Mailman's administrative interface requires authentication, but optional short-lived cookies can be used to lower the barrier imposed by continually entering the administrative password. Once authenticated, administrators have control over all the usual options, including the administrator's contact address, identifying headers and footers placed on every message, welcome messages, and the content of the general list information Web pages. Administrators can control general list privacy options, maximum message sizes, digest and archival options, bounce policies and anti-spam devices. They may also review and act upon posts and requests that are being held for approval.

There are many third-party Web front-ends to Majordomo [Bar98]. However, most of them are little more than simplistic interfaces to subscribing and unsubscribing. The most notable exception is MajorCool [Hou96], which additionally provides end users with a way of browsing all mailing lists on a machine, as well as a full-featured interface to the list configuration. However, MajorCool suffers from several usability problems, all of which are addressed by Mailman.

First, MajorCool has the problem that malicious users can subscribe and unsubscribe other people from mailing lists over the Web. Mailman, on the other hand, requires confirmation emails for subscriptions. For unsubscribing, users must enter a password into a CGI field, which can be generated by Mailman, and delivered to the subscribed email address on request.

Second, MajorCool requires that it and an HTTP server must be co-located on the machine running Majordomo and Sendmail [Hou98]. In contrast, Mailman has been tested with a mail transport and Web server running on separate machines in an NFS environment, and has been tested with the transport, Web server, and Mailman all running on separate machines, where Mailman scripts are run via `rsh` or `ssh`.

Third, MajorCool's interaction with end users is limited. Its goal with respect to end users is to give them a way to browse all the lists on a machine, not to provide a nice Web-based mechanism for interacting with the mailing list. Mailman provides full support for editing options such as digest mode, visibility on the subscriber list, whether posts to a list should be sent back to the user, and so on.

Finally, MajorCool's administrative interface is mainly geared towards interfacing to the traditional Majordomo configuration. In contrast, many of Mailman's administrative options allow for customization of the list's Web interface. In fact, Mailman also allows the list administrator to provide a "real" Web page for his list. He may edit HTML via a password protected Web-based interface. MajorCool essentially lacks the notion of each list having its own home page.

Architecture

Mailman is written almost completely in Python [Pyt98], a freely available object-oriented scripting language. There are a few C wrapper programs for security purposes. Mailman currently requires at least Python 1.5, which is freely available from `http://www.python.org/`.

Mailman's architecture provides great flexibility for accessing list data structures. Each list has an associated database containing configuration options and member addresses. This database can be accessed via the Web interface (the most common mode of access), via a Majordomo-like email command language, and most uniquely, via interactive sessions with the Python interpreter.

[Reviewers: we will also discuss other aspects of the architecture in a full-blown paper, such as our system for site configurations, the OO design of Mailman, and its positive impact on extensibility.]

While Mailman is too new to have much hard data in the way of performance metrics, we do

know that, given a well designed MLM, the performance of the mail transport agent (MTA) will have a much more important impact than the MLM implementation. Also, we have found that even a low-end configuration can handle large amounts of traffic. For example, one mailing list managed by Mailman has had up to 3000 subscribers, and often receives 100 messages in a day (i.e., hundreds of thousands of daily deliveries). The list runs on a low-end Pentium with 48MB of RAM. The machine runs Sendmail on GNU/Linux. The machine also hosts an NNTP news feed for a small ISP, and is able to handle the load, although Sendmail sometimes needs to queue messages.

[Reviewers: we intend to gather more detailed performance data by the date of publication.]

Future Directions

Mailman development is ongoing and highly active. Major projects to be undertaken in the near future include:

- Integrating searching with list archives.
- Manually configurable and automatically used relays for distributing server and network load (along the lines of RFC 1429 [Tho93]).
- An optional threaded persistent server, as opposed to the current "start-by-request" model shared with Majordomo.
- A separation of the roles of list administrator and list moderator.
- PGP integration.

Mailman for System Administrators

Mailman 1.0 is currently in beta release, but is already being used at a number of sites. More information on Mailman can be had at `http://www.list.org`.

[Reviewers: this URL will almost certainly change to a gnu.org address by the time of final submission; the beta status of the software is also subject to change by that time.]

Mailman should work out of the box on any Unix-based platform on which Python runs. It is known to work on SunOS, Solaris, all major distributions of Linux, FreeBSD, Irix and NextStep. Mailman will work with any MTA, since it communicates via the SMTP port instead of through a command. However, Mailman currently generates sendmail-style aliases only. Therefore, aliases for MTAs such as qmail must be modified and installed by hand.

Also, Mailman should work with any HTTP daemon that allows for CGI directories. It is known to work with Apache, NCSA, and Java Web Server.

For current Majordomo users, the transition to Mailman is straightforward; there is a command-line script in the distribution that imports a Majordomo list into Mailman.

Acknowledgements

Mailman was originally written by John Viega. It has since been extended, and is currently being developed and maintained by John Viega, Ken Manheimer, and Barry Warsaw. The Mailman-developers mailing list and the Python community have provided invaluable feedback on this software, including Guido van Rossum, Scott Cotton, Janne Sinkkonen, Michael McLay and Hal Schechner.

Mailman uses free software by Timothy O'Malley for dealing with HTTP cookies. It also integrates Pipermail, free software by Andrew Kuchling that handles message archiving. The archiving code also uses free software by Aaron Watters.

We would like to give special thanks to the Python Software Activity (PSA), and the Corporation for National Research Initiatives (CNRI) for hosting the PSA.

We would also like to give special thanks to Richard Stallman and the Free Software Foundation for their support and guidance.

References

[Bar98] D. Barr. The Majordomo FAQ. `http://www.greatcircle.com/majordomo/majordomo-faq.html`

[Cha92] D. Chapman. Majordomo: "How I Manage 17 Mailing Lists Without Answering "-request" Mail". Proc. *Usenix LISA VI*, Oct. 1992.

[Hou96] B. Houle. "MajorCool: A Web Interface To

Majordomo". Proc. *Usenix LISA X*, Oct. 1996.

[Hou98] B. Houle. `http://ncrinfo.ncr.com/pub/contrib/unix/MajorCool/Docs`

[Lev97] J. Levitt. Ten Questions For Majordomo (An Interview With D. Brent Chapman). `http://techweb.cmp.com/iw/author/Internet8.htm`

[Pyt98] The Python Language Website. `http://www.python.org/`

[Sma98] The SmartList FAQ. `http://www.mindwell.com/`. April 1998 revision.

[Tho93] E. Thomas. *RFC1429: Listserv Distribute Protocol*. Feb. 1993. Available from: `http://www.faqs.org/rfcs/rfc1429.html`