The following paper was originally published
in the Proceedings of the
Tenth USENIX System Administration Conference (LISA X)
Chicago, IL, USA, Sept. 29 - Oct. 4, 1996

Many Mail Domains, One Machine: The Forwarding Mailer

Hal Pomeranz
NetMarket/CUC International

Many Mail Domains, One Machine: The Forwarding Mailer

Hal Pomeranz – NetMarket/CUC International

ABSTRACT

With the explosion of domain registration in the last 18 months, many organizations find themselves having to maintain multiple separate email domains. Existing solutions for this problem are capital- or administrator-intensive. Here we explore a simple solution that allows an organization to support arbitrarily many separate domain spaces on a single host machine with only a single Sendmail server. This system has been used in production at the author's organization since 1994 with great success.

Background

Whether for fun or profit, many more organizations than ever before find themselves maintaining multiple domains. While BIND makes it very easy to configure an maintain multiple DNS domains, sendmail has historically assumed that every domain you have is simply an alias: if you have MX records for *foo.com* and *bar.com* pointed at the same machine, then *user@foo.com* is equivalent to *user@bar.com*. For many organizations (ISPs, Web development shops, and private citizens providing Internet access for friends, to name a few) this assumption is no longer sufficient.

So, your organization has registered *foo.com* and *bar.com* and you want it to be the case that *user@foo.com* to be somebody different from *user@bar.com* (at least from an email perspective). Historically there have been three different types of solution for this problem:

1. Run multiple physical servers. The easiest solution from a sendmail configuration perspective is to simply run a different physical piece of hardware for each distinct mail domain you require. From an administrative perspective, however, this model simply doesn't scale to dozens of domains. This approach also doesn't make sense from a capital cost and resource utilization perspective-- 90% of the time your dozen mail servers are mostly idle, while perhaps one or two of them are overloaded.
2. Run multiple copies of sendmail on one machine. Multiple copies of sendmail bound to different ports with a single master process to handle dispatching mail appropriately can simulate the effect of having multiple distinct hardware components. However, not only does this approach require guru-level knowledge of sendmail configuration, running dozens of sendmail processes simultaneously is incredibly memory inefficient.
3. Build a multi-zone class DB. Under sendmail v8, it is possible to build a class database which causes a single sendmail daemon to route mail for *user@foo.com* to be delivered to a different place than mail for *user@bar.com*. You must, however, exhaustively and explicitly list all addresses which require special handling: you may not apply rules which apply to all *user@host* addresses within a particular domain. Imagine the impossibility of maintaining this scheme if you are an ISP providing mail services for dozens of clients, or a mail administrator for a nation- or world-wide company.

Now assume that, like everybody else in the world, you are an organization with limited capital budget, limited computing resources, and without the administrative resources to maintain a very complex system on an ongoing basis. None of the choices above are very appealing.

Requirements

An ideal system to meet this set of requirements should:

- Never lose mail or generate bounce messages to sender
- Not require a different machine for each domain
- Not require operating a separate sendmail process for each domain
- Simultaneously allow both specific (*user@host.foo.com*) and global (*all user@host pairs in foo.com*) forwarding rules
- Allow addition of new forwarding rules without modifying `sendmail.cf` or restarting active sendmail processes
- Be as simple to administer as `/etc/aliases`
- Not require modification to the sendmail source distribution
- Not significantly increase delivery time of mail or put undue load on mail server

Our solution was to add an entirely new mailer to our `sendmail.cf` file (though this mailer uses the same sender/recipient rules as the *prog* and *local* mailers). "Delivery" via the forwarding mailer is triggered by the presence of the domain in a new class file (`/etc/mail/forward.for` on Solaris systems). The forwarding mailer is actually a Perl program which uses an external file of regular expression based rules to determine the actual delivery address for *user@host.foo.com*. Once the real delivery address is determined, the forwarding mailer simply hands the message back to sendmail for delivery.

Implementation

While one never approaches a project like this without some large amount of trepidation (or perhaps "fear and loathing" is a more apt description), the standard m4 tree provided with sendmail v8 makes the whole configuration process a breeze.

The first step is to add a new mailer definition (sendmail expects these definitions to reside in its `cf/mailer` directory). Figure 1 shows the complete `cf/mailer/forw.m4` definition file. Notice that the file contains more comments than actual configuration lines. Several parameters are defined at the top of this file, but they should not be customized here (or modified at all, in most cases). The `ifdef` syntax used in `forw.m4` means that site administrators can change these macros in their site configuration files and override the default values given here.

The second step to configuring your mail host to use the forwarding mailer is to modify your machine-specific m4 configuration file. Add the lines from Figure 2.

The first line demonstrates how the site administrator would change the values defined at the top of `cf/mailer/forw.m4` (in this case to locate the forwarder program to a hypothetical read-only device for security). The `MAILER(forw)` directive causes the configuration information that is stored in `cf/mailer/forw.m4` to be included in `sendmail.cf`. Next we define a sendmail class `F` from the file `/etc/mail/forward.for` (more on this file in a moment).

The last two lines add new rules in ruleset 0 which cause the forwarding mailer to be invoked if the recipient address matches one of the domains in `/etc/mail/forward.for` (don't forget that you

```
ifdef(‘FORW_MAILER_FLAGS’,, ‘define(‘FORW_MAILER_FLAGS’, ‘DFhMnsu’)’)
ifdef(‘FORW_MAILER_PATH’,, ‘define(‘FORW_MAILER_PATH’, /etc/mail/forwarder)’)
ifdef(‘FORW_MAILER_ARGS’,, ‘define(‘FORW_MAILER_ARGS’, ‘forwarder $u@$h’)’)

#############################################################################
###   NetMarket Forwarding Mailer Hack                                    ###
###   Copyright (C) NetMarket/CUC International, 1995-6                   ###
###   All rights reserved.                                                ###
###   Permission to freely distribute under the same terms as Sendmail.  ###
###   No warranty expressed or implied.                                   ###
#############################################################################

Mforw,          P=FORW_MAILER_PATH,
                F=FORW_MAILER_FLAGS, S=10, R=20/40, D=$z:/tmp,
                A=FORW_MAILER_ARGS

#####
# Uses Sender and Recipient rules from prog/local mailer definition
#####
```

Figure 1: Complete forw.m4 definition file

```
define(‘FORW_MAILER_PATH’, ‘/read-only/bin/forwarder’)
MAILER(forw)

LOCAL_CONFIG
# Used by NetMarket forwarding mailer hack
FF/etc/mail/forward.for

LOCAL_RULE_0
# pass things off to forwarding mailer
R$* < @ $=F > $*        $#forw $@ $2 $: $1
R$* < @ $=F . > $*      $#forw $@ $2 $: $1
```

Figure 2: Machine specific m4 configuration file

MUST HAVE TABS between the left and righthand columns in these rules). Note that we are requiring an explicit match: if you only have `foo.com` in the `forward.for` file, the forwarding mailer will NOT be invoked on *user@host.foo.com*. Because large numbers of deeply nested subdomains are unusual (due mostly to the inconvenience of having to type and remember *user@host.location.org-unit.domain.com*), we have found that this configuration decision has made administration simpler without significantly increasing the size of the `forward.for` file.

Any additional site-specific configuration parameters should be added to the above rules, and a `sendmail.cf` file generated using `m4`. Install this file in the appropriate location for your system (`/etc/sendmail.cf` and `/etc/mail/sendmail.cf` are common).

The third step in the installation process is to use `/etc/mail/forward.for` to list all of the domains you want the forwarding mailer to be invoked on. Here is a sample `forward.for` file

```
beeper.netmarket.com
infersys.com
ftd.com
world.ftd.com
```

One domain or host address per line, any number of lines. Comments and blank lines are not allowed. Note that you kill and restart sendmail every time you update this class file, which (along with being unable to enter comments in the file) we consider to be a defect in sendmail v8.

Step number four is to install the forwarding mailer program as `/etc/mail/forwarder` (or whatever the definition of `FORW_MAILER_PATH`). The complete code for the script is given as Appendix A. At the top of this script are three variables

```
$patfile = '/etc/mail/patterns';
$error_recip = 'postmaster';
$mail = '/usr/lib/sendmail';
```

The first line tells the forwarding mailer where to look for its forwarding rules. The second line gives the recipient for error messages so that the sender does not receive bounces from the forwarding mailer (errors include being unable to open the patterns file or to find a match for the recipient address, as well as the recipient address containing shell or Perl meta-characters). The last line gives the explicit path to sendmail. No additional configuration of the `forwarder` executable is required.

Forwarding Rules

The fifth and final step of the installation is to create your own `/etc/mail/patterns` file. The file is in a two-column format, with a Perl regular expression on the lefthand side and one or more recipient addresses on the right hand side. The forwarding mailer scans this file in order (ignoring blank lines and lines starting with #) until it finds a regular expression which matches the recipient address of the original message, and the mail is then forwarded to the recipient list specified on that line of the `patterns` file.

Note that this "first match and exit" behavior is completely unlike the way sendmail processes `sendmai.cf`. Also, the columns in the `patterns` file are whitespace separated, as opposed to being rigidly tab separated as in `sendmail.cf`. The downside of this flexibility is that the regular expression on the lefthand side of the `patterns` must not contain any whitespace (use the Perl \s operator if you really need to match whitespace).

One use of the forwarding mailer is to define pseudo-subdomains within your own organization. For

```
###
### beeper.netmarket.com -- people to annoy
###

(?i)^hal@beeper.netmarket.com$       beep-hal@netmarket.com
(?i)^jxh@beeper.netmarket.com$       beep-jxh@netmarket.com
(?i)^jim@beeper.netmarket.com$       beep-jxh@netmarket.com
(?i)^josh@beeper.netmarket.com$      beep-josh@netmarket.com
(?i)^kevin@beeper.netmarket.com$     beep-kevin@netmarket.com
(?i)^sysadmin@beeper.netmarket.com$  beep-sysadmin@netmarket.com
```

Figure 3: *beeper.netmarket.com* pseudo-domain

```
###
### beeper.netmarket.com -- people to annoy
###

(?i)^jim@beeper.netmarket.com$  beep-jxh@netmarket.com
(?i)@beeper.netmarket.com$      beep-$u@netmarket.com
```

Figure 4: Alternate configuration

example, we have a *beeper.netmarket.com* domain which enables us to send messages to a user's alphanumeric pager by sending mail to *user@beeper.netmarket.com*. Figure 3 shows the appropriate lines from the `patterns` file. Of course, there's no real value to this pseudo-domain other than we found it easier for people to remember *user@beeper* than *beep-user*.

The forwarding mailer allows the recipient list on the righthand side of the patterns file to contain `$u` tags which get expanded to be the username portion of the original message recipient. Using `$u` we can rewrite the six lines above as Figure 4. Note that we are not only utilizing the `$u` tag but also the "first match" property of the `patterns` file.

Handling forwarding for full toplevel domains with many users, mailing lists, and auto-responders can get tricky. Due to security concerns, the forwarding mailer does not handle piping mail to Majordomo or other autoresponder-type programs. We've adopted the convention of Figure 5 for handling these addresses. The `/etc/mail/aliases` entries for *<thing>-at-infersys@netmarket.com* not only feed the mail to Majordomo or our locally developed autoresponder as appropriate, but also archive the original message in a file.

Note that we don't want to use `$u` to simply forward every *infersys.com* address to *$u-at-infersys@netmarket.com* because we also have entries like those in Figure 6. The first line forwards mail for a specific user (in this case, the owner of the domain) to a user in the *netmarket.com* domain. The second line (the default) catches all other mail to users in *infersys.com* and sends it to the domain owner (triggering a second trip through the forwarding mailer when sendmail attempts to resolve *irilyth@infersys.com*).

Some of our forwarding domains are merely aliases on top of some other ISP's domain space as seen in Figure 7. These lines allow us to intercept mail for *webmaster* and *postmaster* for the given domain but then pass through all other mail to user accounts at some other ISP. We added the last line because a lot of people seemed to get confused between *world.ftd.com* (a non-existent host) and *world.std.com* (a large public-access Unix site in the Boston area): we created an auto-responder to send them a polite message.

The last lines of our `patterns` file are used to catch addresses that haven't matched any explicit patterns; see Figure 8. The first three lines grab all mail for *webmaster*, *postmaster*, *hostmaster*, *root*, or *MAILER-DAEMON* and deliver them locally. The last line forwards all other mail to an autoresponder which sends out a polite "nobody home" response.

Appendix B gives the complete text of a sample `patterns` file using the above examples.

Our Experiences Using the Forwarding Mailer

We have been operating the forwarding mailer for over a year at this point, supporting dozens of domains and subdomains with thousands of users at our high-water mark. The system has been incredibly robust and exceeded all of our requirements. The

```
(?i)^majordomo@infersys.com$        majordomo-at-infersys@netmarket.com
(?i)^majordomo-owner@infersys.com$  majordomo-owner-at-infersys@netmarket.com

(?i)^boston-b5@infersys.com$        boston-b5-at-infersys@netmarket.com
(?i)^owner-boston-b5@infersys.com$  owner-boston-b5-at-infersys@netmarket.com

(?i)^info@infersys.com$             info-at-infersys@netmarket.com
```

Figure 5: Convention for handling piping mail

```
(?i)^irilyth@infersys.com$          irilyth@netmarket.com
(?i)^.*@infersys.com$               irilyth@infersys.com
```

Figure 6: Trap other addresses

```
(?i)^webmaster@ftd.com$             webmaster@netmarket.com
(?i)^postmaster@ftd.com$            postmaster@netmarket.com
(?i)@ftd.com$                       $u@some-isp.net
(?i)@world.ftd.com$                 ftd-bounce-bot@netmarket.com
```

Figure 7: Aliases for forwarding domains

```
(?i)^(web|host|post)master@.*$      $u
(?i)^root@.*$                       root
(?i)^mailer-daemon@.*$              mailer-daemon
(?i)^.*$                            bounce-bot@netmarket.com
```

Figure 8: Catch-all patterns

forwarding mailer was so successful, in fact, that it was regarded as a core technology that gave us significant "competitive advantage" over other Internet development firms, and the author was enjoined from releasing it at the last LISA Conference.

Currently, our mail server (a Sparc 20 with 256MB of RAM) handles 200-250 distinct email messages per hour during working hours, approximately 10% of which move through the forwarding mailer. Overall, we process 4000-5000 email messages per day. This (relatively low) mail volume has no significant performance impact on the mail server, which is also used as a general purpose development machine.

A trip through the forwarding mailer adds less than one second to the overall delivery time of mail that ends up being delivered locally. Obviously, the cost of passing through the forwarding mailer is higher if the mail comes from an external address and must be sent back out through our firewall for delivery at an external site.

Adding new rules to the `patterns` file is straightforward: even administrators without a background in Perl regular expressions can easily cut-n-paste and adapt rules from elsewhere in the file. Initial setup of the system could be cleaner in some respects, but can be forgotten after the initial configuration is done. Having to kill and restart sendmail when updating the `forward.for` class file is particularly annoying.

Availability and Future Plans

A sharfile containing the software (with limited documentation) is available at ftp://ftp.netmarket.com/forw-mailer/forw-mailer.shar. This software is provided "as-is" and is freely distributable under the same terms as sendmail, provided all copyright statements are retained.

In the future, there are several avenues for improvement of the software:

- Too much configuration is required in the site m4 file. The `LOCAL_CONFIG` and `LOCAL_RULE_0` lines should merge into the mailer definition.
- The `forwarder` program could be rewritten in a compiled language to improve execution speed and reduce performance impact.
- The documentation can be improved significantly.

The author is particularly interested in any sites that employ the forwarding mailer on high-volume mail servers (more than 100,000 messages processed per day).

Author Information

Hal Pomeranz (hal@netmarket.com) is Director of Infrastructure Development/Support and Information Security Officer at NetMarket, the Internet division of CUC International. He has been active in the system and network management/security field for over ten years, and is a regular speaker and organizer for technical conferences and professional gatherings. Hal is a columnist for USENIX and other technical publications and has served on the Board of Directors for both the BayLISA and BBLISA System Administration professional societies.

Appendix A: `forwarder` Source Code

```
#!/usr/local/bin/perl

$patfile = '/etc/mail/patterns';
$error_recip = 'postmaster';
$mail = '/usr/lib/sendmail';

   ###########################################################################
   ###   NetMarket Forwarding Mailer Hack                                  ###
   ###   Copyright (C) NetMarket/CUC International, 1995-6                 ###
   ###   All rights reserved.                                              ###
   ###   Permission to freely distribute under the same terms as Sendmail. ###
   ###   No warranty expressed or implied.                                 ###
   ###########################################################################

use Syslog;

$error_str = '';

$toaddr = $ARGV[0];
$error_str = "$0 called with null argument" unless ($toaddr);
$error_str="Hostile address $toaddr" if ($toaddr =~ m,[\{\}\(\)\/\;\&\|\$\=],);
($touser, $tohost) = split(/@/, $toaddr);

unless ($error_str) {
    if (open(PAT, $patfile)) {
        $match = 0;
        while (<PAT>) {
            next if (/^#/ || /^\s*$/);
            ($pattern, $recip) = /^(\S+)\s+(.*\S)\s*$/;
            next unless ($pattern && $recip);
            if ($toaddr =~ /$pattern/) {
                $match = 1;
                last;
            }
        }
        close(PAT);
        if ($match) {
            $recip =~ s/\$u/$touser/g;
        }
        else { $error_str = "No match found for $toaddr"; }
    }
    else {
        $error_str = "Can't open $patfile";
    }
}

$recip = $error_recip if ($error_str);
unless (open(MAIL, "| $mail $recip")) {
    syslog(LOG_MAIL|LOG_ALERT, "forward mailer can't invoke sendmail!\n");
    die "forward mailer can't invoke sendmail!\n";
}
select(MAIL);
if ($error_str) {
    print <<"EOErrHead";
To: $recip
From: postmaster\@netmarket.com (Forwarding Mailer)
Reply-to: postmaster\@netmarket.com
Subject: ERROR: Forwarding mailer

An error has occurred in the Forwarding mailer process. The error message
generated is:

        $error_str
```

```
The original message follows:
==============================================================================
EOErrHead
}
while (<STDIN>) { print; }
close(MAIL);
exit(0);
```

Appendix B: Sample `patterns` **File**

```
###
### beeper.netmarket.com -- people to annoy
###

(?i)^jim@beeper.netmarket.com$      beep-jxh@netmarket.com
(?i)@beeper.netmarket.com$          beep-$u@netmarket.com

###
### infersys.com (personal domain for irilyth@netmarket.com)
###

(?i)^majordomo@infersys.com$        majordomo-at-infersys@netmarket.com
(?i)^majordomo-owner@infersys.com$  majordomo-owner-at-infersys@netmarket.com

(?i)^boston-b5@infersys.com$        boston-b5-at-infersys@netmarket.com
(?i)^owner-boston-b5@infersys.com$  owner-boston-b5-at-infersys@netmarket.com

(?i)^info@infersys.com$             info-at-infersys@netmarket.com

(?i)^irilyth@infersys.com$          irilyth@netmarket.com
(?i)^.*@infersys.com$               irilyth@infersys.com

###
### ftd.com (Flowers and confused people at world.std.com)
###

(?i)^webmaster@ftd.com$             webmaster@netmarket.com
(?i)^postmaster@ftd.com$            postmaster@netmarket.com
(?i)@ftd.com$                       $u@some-isp.net
(?i)@world.ftd.com$                 ftd-bounce-bot@netmarket.com

###
### Some useful defaults to catch everything else
###

(?i)^(web|host|post)master@.*$      $u
(?i)^root@.*$                       root
(?i)^mailer-daemon@.*$              mailer-daemon
(?i)^.*$                            bounce-bot@netmarket.com
```