The Exim Mail Transfer Agent

(A brief introduction)

http://www.exim.org

Configuration file

- Exim uses a single runtime configuration file, divided into a number of sections

- The first section contains global option settings

- The other sections start with "begin *sectionname*"

- They are all optional, and may appear in any order

- Comments, macros, if-then-else, and inclusions are available
    Some Debian versions use inclusions to support a multi-file
    configuration arrangement

- Option settings can refer to auxiliary data files, for example, a file of aliases (traditionally **/etc/aliases**)

Changing the runtime configuration

- Edit **/usr/exim/configure** with your favourite text editor

- New Exim processes will pick up the new file right away

- You need to SIGHUP the daemon (as root) to restart it

```
kill -HUP \
    $(cat /var/spool/exim/exim-daemon.pid)
```

- Check the log to see if it restarted successfully

```
tail /var/spool/exim/log/mainlog
```

Configuration file sections

- Global options
  General and input-related options
- Address rewriting rules
  Specify rewriting of envelope and header addresses
- Retry rules
  Control retries after temporary failures
- Router configuration
  Specify recipient address processing
- Transport configuration
  Specify how actual deliveries are done
- Authenticator configuration
  Specify SMTP authentication methods
- Access Control Lists (ACLs)
  Define policy controls for incoming messages

July 2005

Default configuration file layout

Global option settings

`begin ACL`
Access control lists ⎤ required for SMTP input

`begin routers`
Router configuration ⎤
`begin transports` ⎥ required for message delivery
Transport configuration ⎦

`begin retry`
Retry rules

`begin rewrite`
Rewriting rules

`begin authenticators`
Authenticator configuration

Examples of common global options

- SMTP input limits

```
smtp_accept_max = 200
smtp_accept_queue = 150
smtp_accept_reserve = 10
smtp_accept_reserve_hosts = 192.168.0.0/16
smtp_connect_backlog = 100
```

- Overloading

```
queue_only_load = 5
deliver_queue_load_max = 7
```

- Message size limits

```
message_size_limit = 10M
return_size_limit = 65535
```

July 2005

Router overview

- Exim contains a number of different routers

  Example: the *dnslookup* router does DNS processing
  the *redirect* router does address redirection
  (aliasing and forwarding)

- Routers decide how to deliver to addresses

- The configuration defines which routers are used, in which order, and under what conditions

  Example: routers are often restricted to specific domains

- The order in which routers are defined is important

- The same router may appear more than once, usually with different configurations

Exim routing
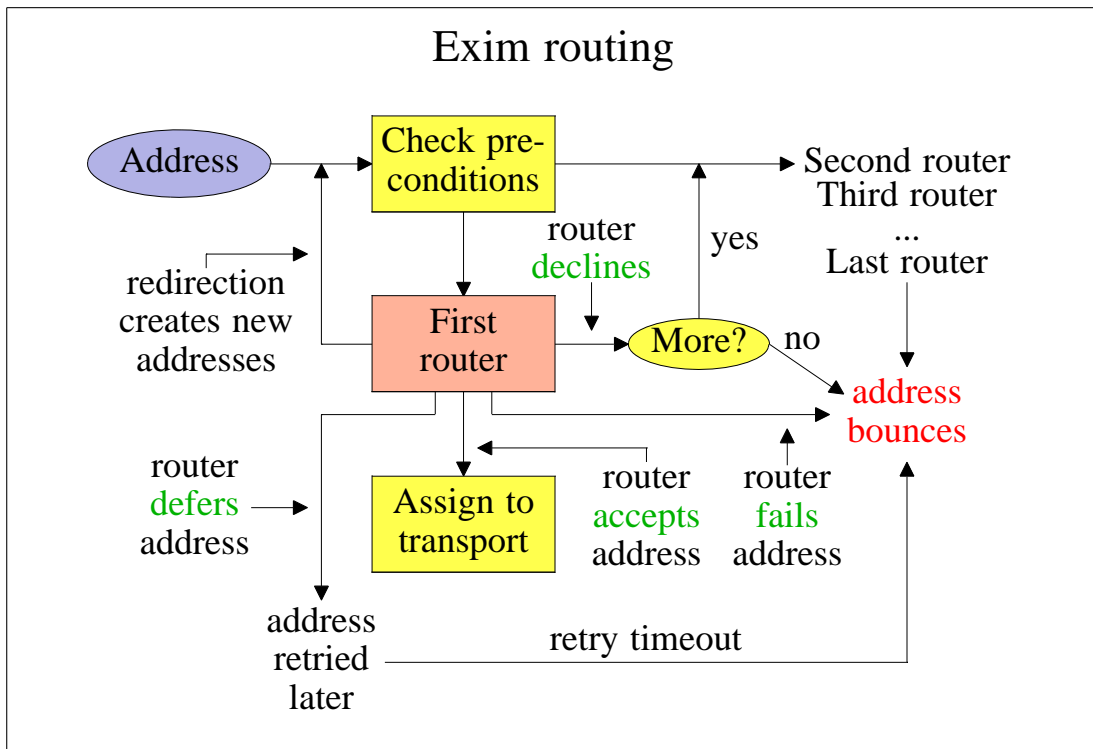
Simple routing configuration

- Check for non-local domains: run *dnslookup* router
  - Accept: assign to *smtp* transport
  - Decline: "no_more" set, so address bounces

- Check for system aliases: *redirect* router
  - Accept: generates new address(es)
  - Decline: passed to next router

- Check for local user forwarding: another *redirect* router
  - Accept: generates new address(es)
  - Decline: passed to next router

- Check for local user: run *accept* router
  - Accept: assign to *appendfile* transport

- No more routers: address bounces

Exim transports

- Transports are the components of Exim that actually deliver copies of messages

  The *smtp* transport delivers over TCP/IP to a remote host
  The *appendfile* transport writes to a local file
  The *pipe* transport writes to another process via a pipe
  The *lmtp* transport does likewise, using the LMTP protocol
  The *autoreply* transport is anomalous, in that it creates an automatic response instead of doing a real delivery

- The order in which transports are defined is not important

- A transport is used only when referenced from a router

- Transports are run in subprocesses, under their own uid, after all routing has been done

- Multiple remote deliveries can happen simultaneously

July 2005

Named item lists

```
domainlist local_domains = @ : plc.com
hostlist   relay_hosts = 192.168.32.0/24
```

- Abstraction: list is specified in one place only
  References are shorter and easier to understand

- Optimization: matches are cached where possible
  Example: several routers testing the same domain list
  Cannot cache by default if list contains expansion items

- A named list is referenced by prefixing its name with +
  ```
  hosts = 127.0.0.1 : +relay_hosts
  ```

- A named list can be negated
  ```
  domains = !+local_domains
  ```
  This is not possible with macros

Named lists in the default configuration

- The default configuration uses three named lists

  ```
  domainlist local_domains = @
  domainlist relay_to_domains =
  hostlist   relay_from_hosts = 127.0.0.1
  ```

- Local domains are going to be delivered on this host
  @ means "the local name of the local host"

- No domains are defined for relaying by default

- The local host is permitted to relay through itself
  Some clients send mail this way

- These lists are used later to set up these controls
  The above settings just define the lists

July 2005

Default routers (1)

- The first router handles non-local domains

```
dnslookup:
  driver = dnslookup
  domains = ! +local_domains
  ignore_target_hosts = 0.0.0.0 : 127.0.0.0/8
  transport = remote_smtp
  no_more
```

- The **domains** precondition checks for a non-local domain

- Silly DNS entries are ignored

- If the domain is found in the DNS, assign the address to the **remote_smtp** transport for delivery

- Otherwise, **no_more** changes "decline" into "fail"

Default routers (2)

- The second router handles system aliases

```
system_aliases:
  driver = redirect
  data = ${lookup{$local_part}lsearch\
         {/etc/aliases}}
  allow_fail                        allows :fail:
  allow_defer                       allows :defer:
  pipe_transport = address_pipe
  file_transport = address_file
# user = exim
```

- Alias files look like this

```
postmaster:  pat, james@otherdom.example
majordomo:   |/usr/bin/majordom ...
alice:       :fail: No longer works here
```

July 2005

Default routers (3)

- The third router handles users' *.forward* files

```
userforward:
  driver = redirect
  check_local_user
  file = $home/.forward
  no_verify
  no_expn
  check_ancestor
  pipe_transport = address_pipe
  file_transport = address_file
  reply_transport = address_reply
  allow_filter                allows filter files
```

- **data** and **file** are mutually exclusive options for **redirect**
  data expands to a redirection list
  file expands to the name of a file containing a redirection list

Default routers (4)

- The final router handles local users' mailboxes

```
localuser:
  driver = accept
  check_local_user
  transport = local_delivery
```

- Recap: an address is routed like this:
Remote address	=> **remote_smtp** transport, fail
System alias	=> new address(es), fail, defer
User's *.forward*	=> new address(es)
Local user	=> **local_delivery** transport
Unrouteable address	=> bounce

- This is just one of many possible configurations
  There are other routers that we have not met yet...

Default transports (1)

- Main transports

```
remote_smtp:
  driver = smtp

local_delivery:
  driver = appendfile
  file = /var/mail/$local_part
  delivery_date_add
  envelope_to_add
  return_path_add
# group = mail
# mode = 0660
```

- Default local delivery assumes a "sticky bit" directory
  Setting **group** and **mode** is an alternative approach

Default transports (2)

- Auxiliary transports

```
address_pipe:
  driver = pipe
  return_output

address_file:
  driver = appendfile
  delivery_date_add
  envelope_to_add
  return_path_add

address_reply:
  driver = autoreply
```

July 2005

Routing to smarthosts

- Replace the first router with

```
send_to_smarthost:
  driver = manualroute
  domains = ! +local_domains
  route_list = * smarthost1.example:\
                  smarthost2.example
  transport = remote_smtp
```

- A **route_list** rule contains space-separated items
  The first is a single domain pattern: * matches any domain
  The second is a list of hosts for the matching domain

- Not shown in the above example
  The third is **bydns** or **byname** (default tries both)
  A transport name may also be given

Virtual domains

- Straightforward cases are just an aliasing application

```
virtual_domains:
  driver = redirect
  domains = lsearch;/etc/virtual-domains
  data = ${lookup{$local_part}lsearch\
          {/etc/valias/$domain}}
  no_more
```

- Or use a **dsearch** lookup to save having a separate list

```
domains = dsearch;/etc/valias
```

  Ensure Exim is built with **dsearch** support

- For large virtual domains, use something better than *lsearch*

Message filtering

- Exim supports three kinds of filtering
  User filter: run while routing ("**.forward** with conditions")
  System filter: run once per message per delivery attempt
  Transport filter: external program added to transport

- User and system filters are run for each delivery attempt
  Simple control language, designed for end users
  Exim also supports Sieve filtering (RFC 3028)
  If delivery is deferred, filters run more than once
  Filter can detect first time run

- User and system filters use the same syntax
  Documented separately for the benefit of end users
  The system filter has some additional commands (**fail**, **freeze**)
  These can be enabled for redirection filters

User filter example (1)

```
# Exim filter
# Don't touch bounces
if error_message then finish endif

# Throw away junk
if
  $sender_address matches \N^\d{8}@\N or
  $h_Subject: contains "Make money" or
  $h_X-Spam_bar: contains "+++++" or
  $message_body contains "this is not spam"
then seen finish endif

# Conditional forwarding
if $h_from: does not contain "personal" then
  unseen deliver my.secretary@example.com
endif
```

User filter example (2)

```
# Sort mailing list messages
if $h_List-Id:
  contains "<exim-users.exim.org>" then
    save $home/Mail/exim-list
    finish
elif $h_List-Id:
  contains "<exim-dev.exim.org>" then
    save $home/Mail/exim-dev
    finish
endif

# Auto-reply
if personal alias ph10@cam.ac.uk then
  mail subject "Re: $h_subject:"
  file $home/auto-reply/message
  log  $home/auto-reply/log
  once $home/auto-reply/once
endif
```

Filter commands

- **deliver** does "true" forwarding (sender does not change)
- **save** delivers to a named file or directory
- **pipe** delivers via a pipe to a given command
- **mail** generates a new mail message
- **logwrite** writes to a log file, defined by **logfile**

- **deliver**, **save**, and **pipe** are *significant* by default
  Normal deliveries are bypassed if anything significant is done
  Can be made not significant by **unseen**

- **logwrite** happens during filtering

- The others are set up during filtering, but happen later
  This means the result of **pipe** is not available during filtering

- The sysadmin can lock out certain facilities in user filters
  The **save**, **pipe**, **mail**, and **logwrite** commands
  File existence tests, lookups, calling Perl, **readfile**, **readsocket**,
  and **run** in expansions

The system filter (1)

- Runs once per message, at every delivery start
  Use **first_delivery** to detect the very first time
  Can see all recipients in **$recipients**

- Can add to recipients or completely replace recipients
  Non-significant delivery adds, significant delivery replaces

- Can add header lines that are visible to routers, transports, and user filters

- Can remove header lines

- Can freeze a message, or bounce a message

- The system filter is set up by options like these

```
system_filter = /etc/exim/sysfilter
system_filter_file_transport = address_file
system_filter_pipe_transport = address_pipe
system_filter_user = exim
```
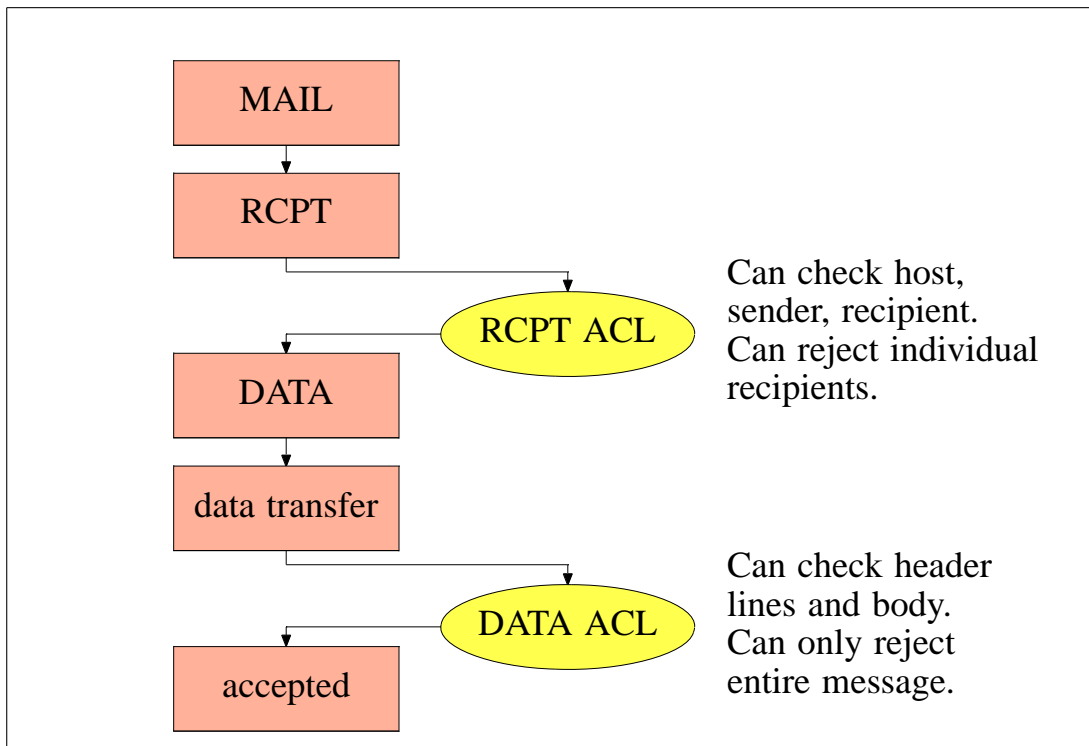
The system filter (2)

- Not powerful enough to do detailed spam checking

- Useful for per-message logging or archiving tasks

- Example

```
# Exim filter
if first_delivery and
  ${mask:$sender_host_address/24}
  is 192.168.34.0/24
then
  noerror unseen save
    /var/mail/archive/${substr_0_10:$tod_log}
endif
```

- Cannot use for per-recipient tasks, but can see all recipients

July 2005

Access control lists

- Most ACLs are relevant for SMTP input
  They do apply to local (*stdin/stdout*) SMTP (the **-bs** option)
  An ACL is available for non-SMTP input

- For incoming SMTP messages the main ACLs are these
  acl_smtp_rcpt defines the ACL to be run for each RCPT
    Default is "deny"
  acl_smtp_data defines the ACL to be run after DATA
    Default is "accept"

- Tests on message content can be done only after DATA or in a non-SMTP ACL

- Other ACLs can be used for AUTH, ETRN, EXPN, EHLO, MAIL, STARTTLS, VRFY, at the start of DATA, and at the start of an SMTP session

Can check host, sender, recipient. Can reject individual recipients.

Can check header lines and body. Can only reject entire message.

A simple ACL

- In the main section of the configuration

  `acl_smtp_rcpt = acl_check_rcpt`

- In the ACL section of the configuration

  ```
  acl_check_rcpt:
    accept    local_parts = postmaster
              domains     = +my_domains

    require   verify      = sender

    accept    domains     = +my_domains
              verify      = recipient
  ```

- Conditions are "anded" together
  Conditions may be repeated
  Evaluation is in order
  Evaluation stops as soon as the outcome is known

- Implicit "deny" at the end

ACL statements

- Each statement contains a verb and a list of conditions
  verb   *condition 1* (one per line)
              condition 2
              ...

- If all the conditions are satisfied
  accept Accepts SMTP command or non-SMTP message (else
              may pass or reject – see later)
  defer   Gives a temporary rejection (= **deny** for non-SMTP)
  deny   Rejects (else passes)
  discard Like **accept** but discards recipients
  drop   Like **deny** but drops an SMTP connection
  require Passes (else rejects)
  warn   Takes some warning action (writes log or adds header)
              Always passes

July 2005

ACL modifiers

- **message** defines a custom message for a denial or warning

```
deny     message  = You are black listed at \
                    $dnslist_domain
         dnslists = rbl.mail-abuse.org : ...
```

- **log_message** defines a custom log message

```
require log_message = Recipient verify failed
        verify      = recipient
```

- **endpass** is used with **accept** for a 3-way outcome

```
accept  domains = +local_domains
        endpass
        verify  = recipient
```

Above **endpass**, failure causes the next statement to be run
Below **endpass**, failure causes rejection

The default ACL (1)

```
acl_check_rcpt:

  accept  hosts        = :

  deny    domains      = +local_domains
          local_parts  = ^[.] : ^.*[@%!/|]

  deny    domains      = !+local_domains
          local_parts  = ^[./|] : \
                         ^.*[@%!] : \
                         ^.*/\\.\\./

  accept  local_parts  = postmaster
          domains      = +local_domains

  require verify       = sender
```

(continued)

The default ACL (2)

```
accept   domains        = +local_domains
         endpass
         message        = unknown user
         verify         = recipient

accept   domains        = +relay_to_domains
         endpass
         message        = unrouteable address
         verify         = recipient

accept   hosts          = +relay_from_hosts

accept   authenticated = *

deny     message        = relay not permitted
```
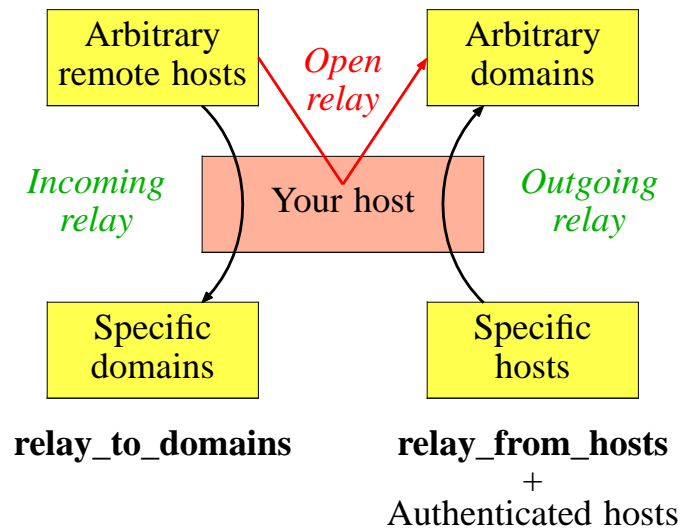
Good and bad relaying

relay_to_domains   **relay_from_hosts**
                                    +
                           Authenticated hosts

Content scanning

- These features were created by Tom Kistner
  Originally a separate patch called "Exiscan"

- From release 4.50, Exiscan is part of the main Exim code
  Build-time options control its inclusion in the Exim binary
  Tom Kistner is still the maintainer

- Additional conditions for the DATA ACL
  malware detects viruses and other malware using 3rd party
       scanners such as ClamAV and Sophos
  spam calls and uses results from SpamAssassin
  regex does regex matches on a message

- Each condition passes back expansion variables that contain useful information

- There is also an additional ACL called **acl_smtp_mime**
  If defined, this is called for each separate MIME part
  Many variables are set to contain data about the MIME part

Content scanning examples

- In the DATA ACL:

```
deny   message = Found $malware_name
       malware = *

warn   message = X-Spam_score: $spam_score\n\
         X-Spam_score_int: $spam_score_int\n\
         X-Spam_bar: $spam_bar\n\
         X-Spam_report: $spam_report
```
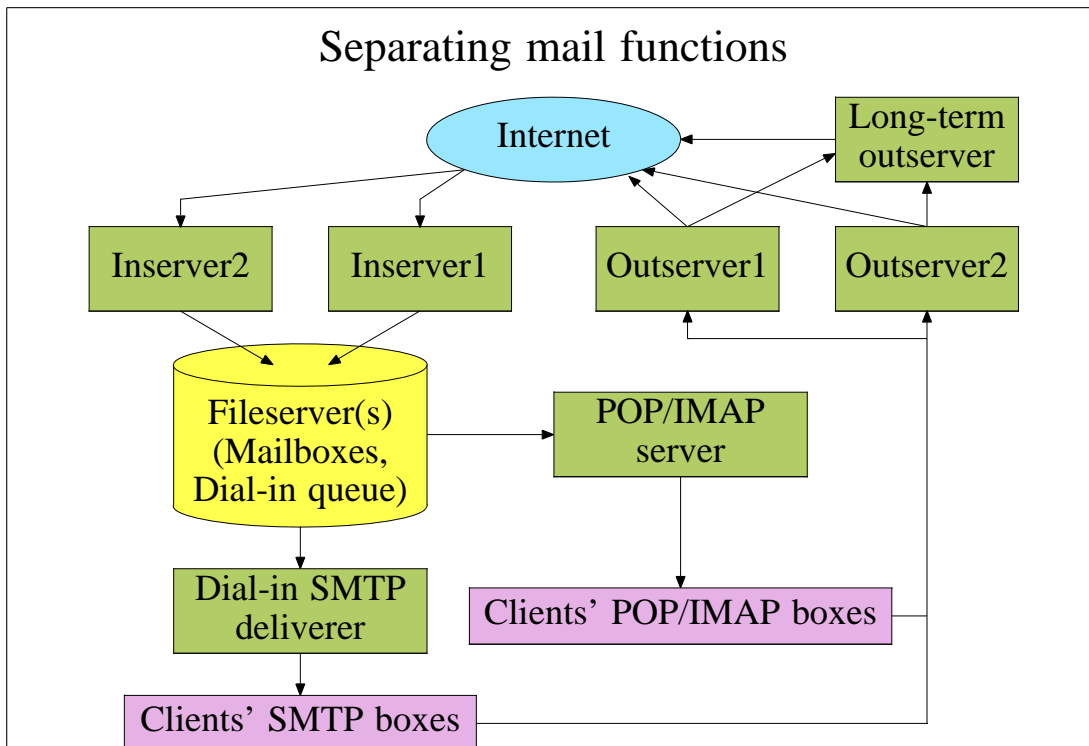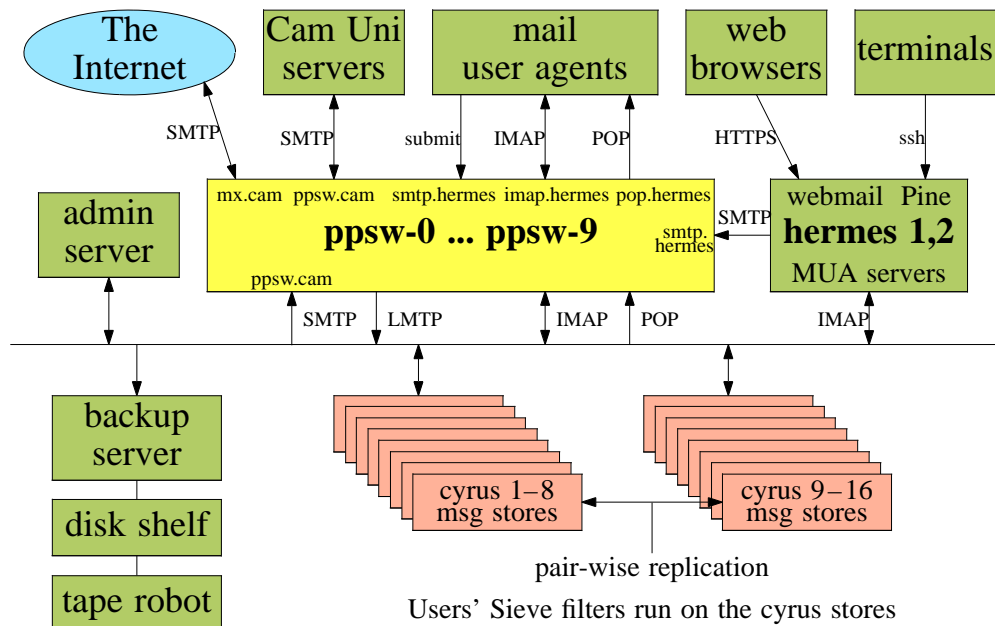
- In the MIME ACL:

```
deny   message = Executable attachments \
                   not permitted
       condition = ${if match{$mime_filename}\
                   {\N\.exe$\N}}
```

Large installations

- Use a local name server with plenty of memory

- Exim is limited by disk I/O
  - Use fast disk hardware; evaluate hardware/OS/filesystem
  - With Reiserfs, disable disk block sharing
  - Put hints on RAM disk; spool and log files on different disks
  - Disable **msglog** files, **rejectlog**; set **split_spool_directory**
  - Use multiple directories for user mailboxes

- Avoid linear password files

- Use maildir format to allow parallel deliveries

- Plan to expand "sideways" with parallel servers
  - This also helps add more disk access capacity

- Keep output queue as short as possible
  - Use fallback hosts and/or **$message_age** for several levels

---

Separating mail functions

July 2005

Not separating mail functions


```
The Internet        Cam Uni servers    mail user agents    web browsers    terminals

    SMTP      SMTP      submit   IMAP   POP      HTTPS        ssh

    mx.cam  ppsw.cam  smtp.hermes  imap.hermes  pop.hermes          webmail  Pine
admin       ppsw-0 ... ppsw-9                         SMTP    hermes 1,2
server                                          smtp.hermes              MUA servers
            ppsw.cam

            SMTP  LMTP        IMAP  POP              IMAP

backup
server
                    cyrus 1–8              cyrus 9–16
disk shelf          msg stores            msg stores

tape robot
                        pair-wise replication
            Users' Sieve filters run on the cyrus stores
```

Using a uniform MTA cluster

- The Cambridge arrangement uses thorough address verification
  This keeps the queues small, which is vital

- My colleague Tony Finch has written some papers about it

- A full description of this configuration is at
  http://www.cus.cam.ac.uk/~fanf2/hermes/doc/talks/
  2005-02-eximconf/

- See also
  http://www.cus.cam.ac.uk/~fanf2/hermes/doc/talks/
  2004-02-ukuug/

  http://www.cus.cam.ac.uk/~fanf2/hermes/doc/talks/
  2005-02-ukuug/

July 2005

Exim resources

- ASCII documentation is included in the tarball
- Downloadable PostScript, PDF, Texinfo, and HTML versions
- The HTML documentation is online
- FAQ in ASCII and HTML with keyword-in-context index

- Website: **http://www.exim.org/**
- Discussion list: **exim-users@exim.org**
- Development list: **exim-dev@exim.org**
- Announce list: **exim-announce@exim.org**
- Indexed archive: **http://www.exim-users.org/**
- Wiki: **http://www.exim.org/eximwiki/**
- Book: **http://www.uit.co.uk/exim-book/**

Exim is available from

ftp://ftp.csx.cam.ac.uk/pub/software/email/exim/...

.../exim4/exim-4.*xx*.tar.gz (or **.bz2**)

under the GNU General Public Licence (GPL)