AfNOG 2003

The Exim Mail Transfer Agent

(A brief introduction)

http://www.exim.org

Configuration file

- Exim uses a single runtime configuration file, which is divided into a number of sections
- The first section contains global option settings
- The other sections start with “begin *sectionname*”
- They are optional, and may appear in any order
- Comments, macros, and inclusions are available
- Option settings can refer to auxiliary data files, for example, a file of aliases (usually **/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 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 for incoming SMTP

Default configuration file layout

Global option settings

`begin ACL`

Access control lists

`begin routers`

Router configuration

`begin transports`

Transport configuration

`begin retry`

Retry rules

`begin rewrite`

Rewriting rules

`begin authenticators`

Authenticator configuration

required for SMTP input

required for message delivery

Examples of common global options

- SMTP input limits

```
smtp_accept_max = 200
smtp_accept_queue = 150
smtp_accept_reserve = 10
smtp_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
```

Exim 4 routers

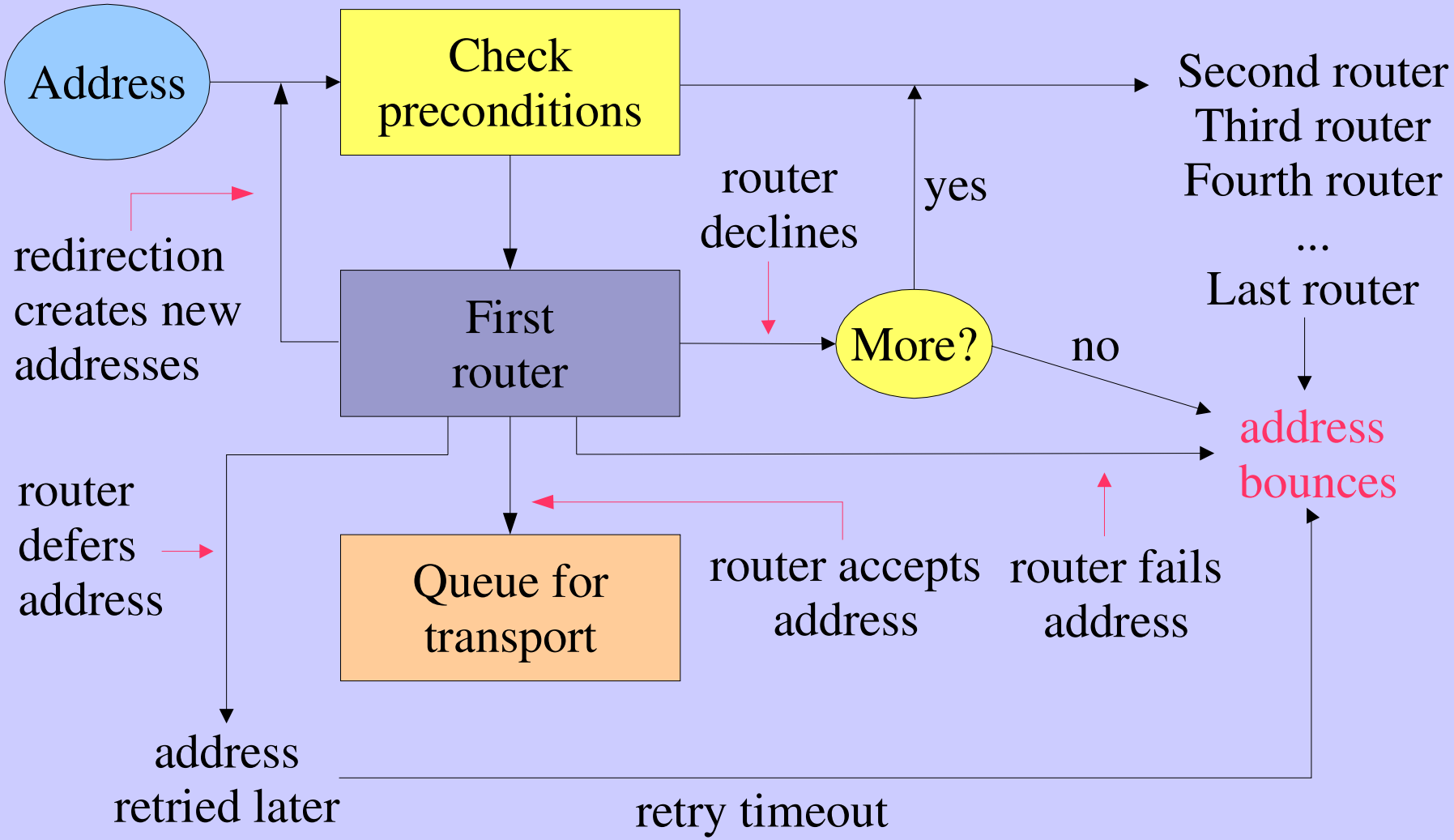
- Exim contains a number of different routers
  Example: the *dnslookup* router does DNS processing
  the *redirect* router does address redirection
  (aliasing and forwarding)

- The configuration defines which routers are used, in which order, and under what conditions
  Example: routers are often restricted to specific domains

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

- The order in which routers are defined matters

Exim 4 routing



Simple routing configuration

- Check for non-local domain: run *dnslookup* router
  - Accept: queue for smtp transport
  - Decline: “no_more” set => 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: queue for 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 LMTP
  - 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 unimportant
- 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

Default routers (1)

- The first router handles non-local domains

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

- The precondition checks for a nonlocal domain
- Silly DNS entries are ignored
- If the domain is found in the DNS, queue for **remote_smtp**
- 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 file lines look like this

```
postmaster:  pat, james@otherdom.example

retired:     :fail: No longer works here

majordomo:   |/usr/bin/majordom ...
```

Default routers (3)

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

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

- **data** and **file** are mutually exclusive options for **redirect**

  data expands to a redirection list

  file expands to the name of a file containing such a list

Default routers (4)

- The final router handles local user's mailboxes

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

- Recap - an address is routed like this:

Remote address	=> **remote_smtp** transport
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 out of many posssible configurations

Default transports (1)

- Main transports

```
remote_smtp:
  driver = smtp

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

- Default assumes a “sticky bit” directory

  Setting **group** and **mode** is an alternate approach

Default transports (2)

- Auxiliary transports

```
address_pipe:
  driver = pipe
  return_output

address_file:
  driver = appendfile
  delivery_data_add
  return_path_add
  envelope_to_add

address_reply:
  driver = autoreply
```

Routing to smarthosts

- Replace the first router with this

```
send_to_smarthost:
  driver = manualroute
  domains = ! +local_domains
  route_list = * smart-host1.example:\
                 smart-host2.example
  transports = route_smtp
```

- A **route_list** rule contains three space-separated items
  - The first is a domain pattern: * matches any domain
  - The second is a list of hosts for the matching domains
  - The third is **byname** (default) or **bydns**

- Set **hosts_randomize** to sort the hosts randomly each time

Virtual domains

- Straightforward cases are just aliasing

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

- An alias with no domain assumes the local qualify domain

```
philip:  ph10
jc:      julius@other.domain.com
```

Access control lists

- ACLs are relevant only for SMTP input
  - But they do apply to local SMTP (**-bs** and **-bS**)
- For incoming SMTP messages
  - **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 only be done after DATA
- Other ACLs can be used for AUTH, ETRN, EXPN, VRFY

A simple ACL

```
acl_smtp_rcpt = acl_check_rcpt

begin acl

acl_check_rcpt:
  accept   local_parts = postmaster
           domains     = +my_domains

  require  verify      = sender

  accept   domains     = +my_domains
           verify      = recipient
```

- Implicit “deny” at the end

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 in named lists are cached
  Example: several routers testing the same domain list
- 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

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 Allows the SMTP command to proceed (else may pass or reject - see next slide)
  deny Rejects (else passes)
  require Passes (else rejects)
  warn Takes some warning action (e.g. logs or adds header) Always passes

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 the **accept** verb for a 3-way outcome

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

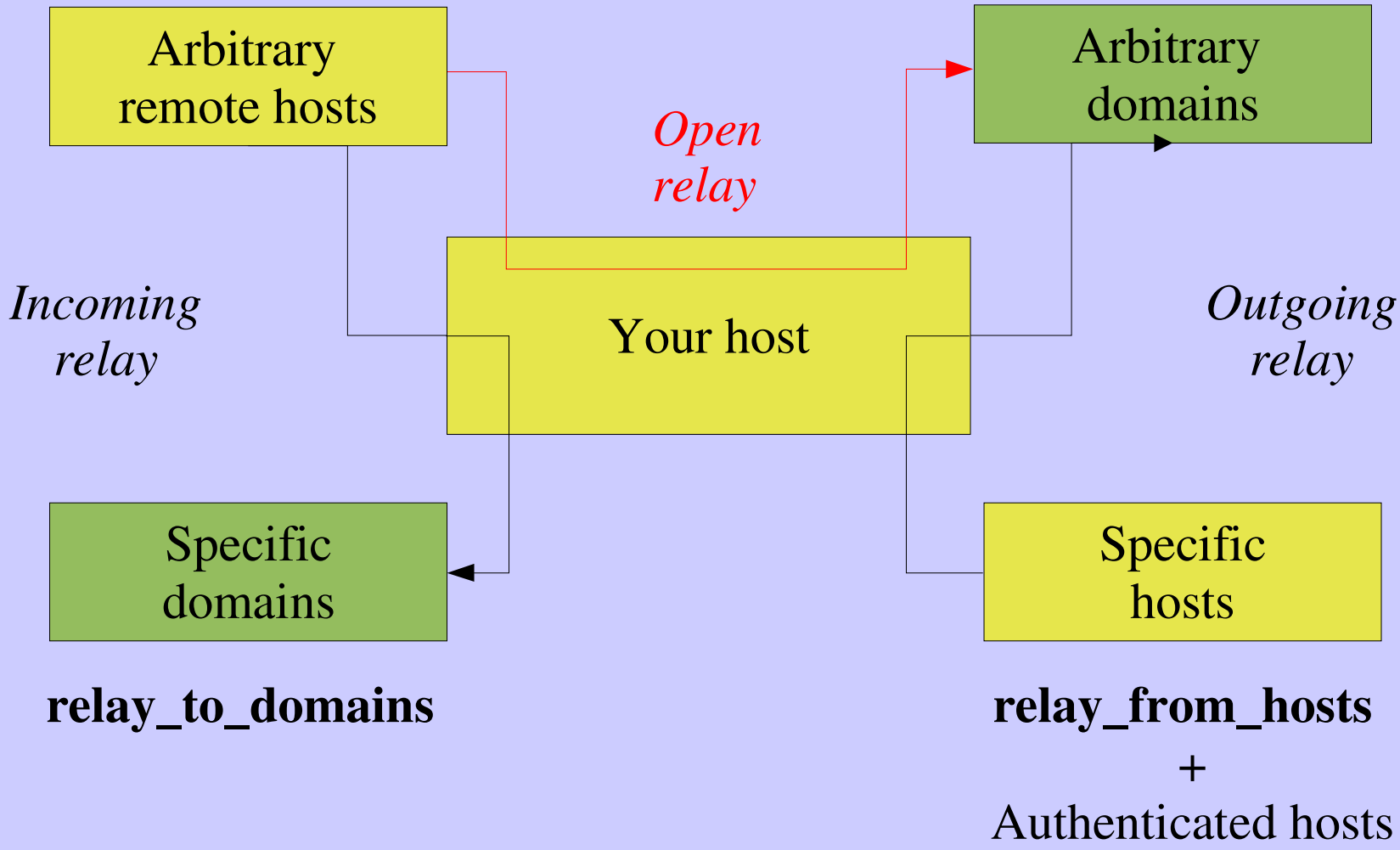
  Above **endpass**, failure causes the next statement to be run

  Below **endpass**, failure causes rejection

The default ACL

```
acl_check_rcpt:
  accept  hosts         = :
  deny    local_parts = ^.*[@%!|/] : ^\\.
  accept  local_parts = postmaster
          domains       = +local_domains
  require verify        = sender
  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



Message filtering

- Exim supports three kinds of filtering
  - User filter: run while routing ("***.forward** with conditions*")
  - System filter: run once per message
  - Transport filter: external program added to transport
- User and system filters are run for each delivery attempt
  - If delivery is deferred, filters run more than once
- User and system filters use the same syntax
  - System filter has some additional commands (**fail**, **freeze**)
  - They can be enabled for redirection filters
- Exim also supports a ***local_scan()*** function
  - Local C code can inspect a message at the point of arrival

User filter example

```
# Exim filter
# Don't touch bounces
if error_message then finish endif
# Throw away junk
if
  $h_subject: contains "Make money" or
  $sender_address matches \N^\d{8}@\N or
  $message_body contains "this is spam"
then seen 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
- **pipe** delivers via a pipe to a given command
- **mail** generates a new mail message
- **logwrite** writes to a log file

- **deliver**, **save**, and **pipe** are significant by default
  Can be made not significant by **unseen**
- **logwrite** happens during filtering
- The others are just set up during filtering and happen later
  The result of **pipe** is not available during filtering

- Sysadmin can lock out a number of filter facilities
  save, **pipe**, **mail**, and **logwrite** commands
  existence tests, lookups, Perl, readfile, run in expansions

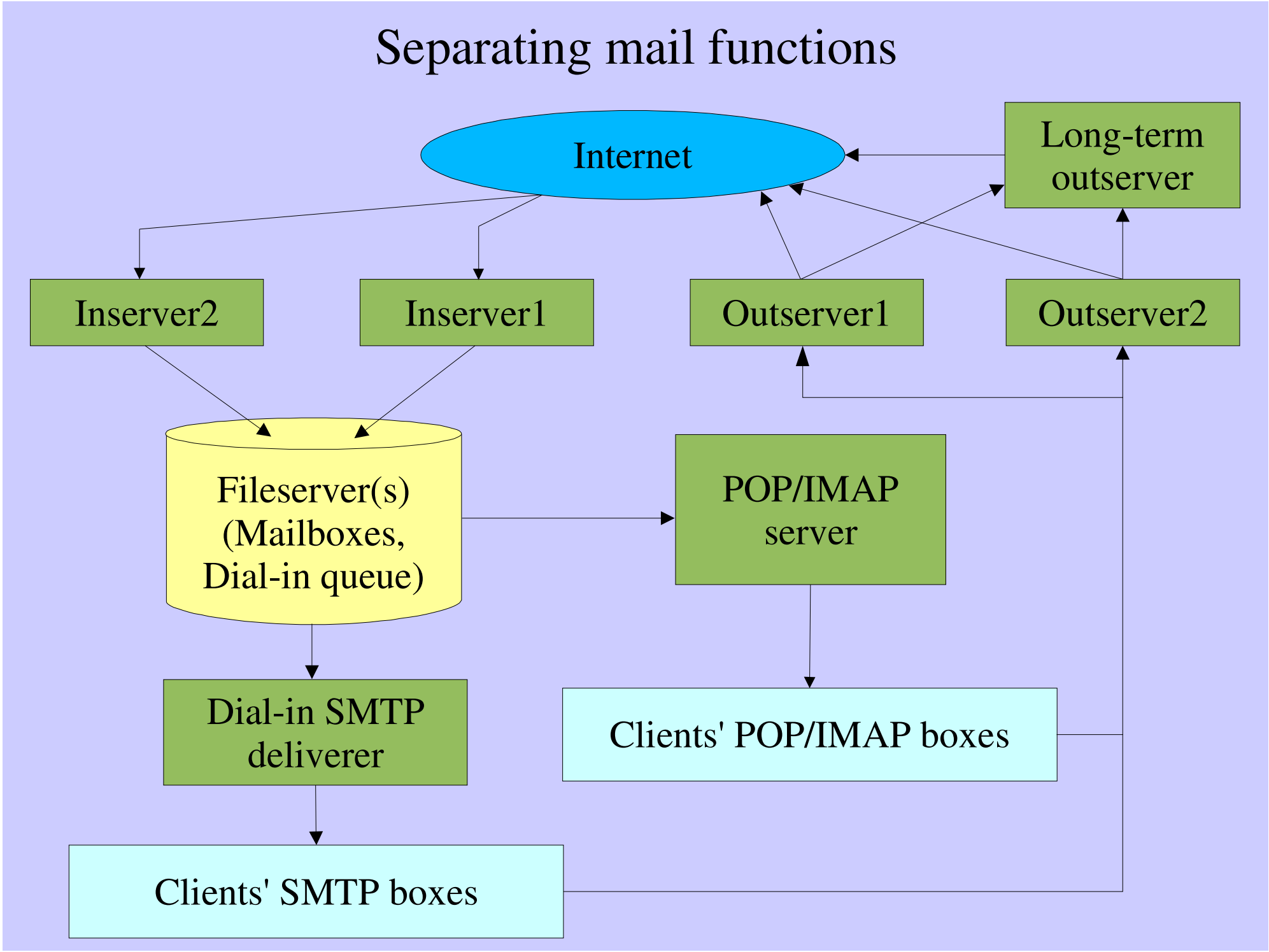
The system filter

- Runs once per message, at every delivery start
  - Use **first_delivery** to detect 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 the routers, transports, and user filters
- Can remove header lines
- Can freeze message or bounce it

- Set up by

```
system_filter = /etc/exim/sysfilter
system_filter_file_transport = address_file
system_filter_pipe_transport = address_pipe
system_filter_user = exim
```

Large installations

- Use a local name server with plenty of memory
- Exim is limited by disc I/O
  - Use fast disc hardware
  - Put hints on RAM disc
  - 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 disc access capacity
- Separate incoming and outgoing mail
- Keep output queue as short as possible
  - Use fallback hosts and/or **$message_age** for several levels

Separating mail functions