Introduction

Building a high-performance mail delivery system is one thing; building one that does not knock over other systems is a different story. Some mailers suffer from the thundering herd syndrome: they literally flood other systems with mail. Postfix tries to be a fast mailer and a good neighbor at the same time.

On the inbound side, the Postfix SMTP server has defenses in place against malicious or confused clients. They won't protect against an all-out denial of service attack on your infrastructure, but then nothing will except pulling the plug.

Unless indicated otherwise, all parameters described here are in the main.cf file. If you change parameters of a running Postfix system, don't forget to issue a postfix reload command.

Process limits

The default_process_limit parameter (default: 50) gives direct control over inbound and outbound delivery rates. This parameter controls the number of concurrent processes that implement a Postfix service (smtp client, smtp server, local delivery, etc.). On small systems, or on systems connected via dialup networks, a default_process_limit of 10 is probably more than adequate. Use a larger value if your machine is a major mail hub.

You can override this setting for specific Postfix daemons by editing the master.cf file. For example, if you do not wish to receive 50 SMTP messages at the same time, you could specify:

```
  #
==========================================================================
  # service type  private unpriv  chroot  wakeup  maxproc command + args
  #               (yes)   (yes)   (yes)   (never) (50)
  #
==========================================================================
  . . .
  smtp      inet  n       -       -       -       5       smtpd
  . . .
```

Destination concurrency

So, you have this huge mailhub with tons of disk and memory, and have configured Postfix to run up to 1000 SMTP client processes at the same time. Congratulations. But do you really want to make 1000 simultaneous connections to the same remote system? Probably not.

The Postfix queue manager comes to the rescue. This program implements the analog of the TCP slow start flow control strategy: when delivering to a site, send a small number of messages first, then increase the rate as long as all goes well; back off in the face of congestion.

The initial_destination_concurrency parameter (default: 2) controls how many messages are initially sent to the same destination before adapting delivery concurrency. Of course, this setting is effective only as long as it does not exceed the process limit and the destination concurrency limit for the specific mail transport channel.

The default_destination_concurrency_limit parameter (default: 10) controls how many messages may be sent to the same destination simultaneously. You can override this setting for specific delivery channels (local, smtp, uucp etc.). The main.cf file recommends the following:

```
local_destination_concurrency_limit = 2
default_destination_concurrency_limit = 10
```

The local_destination_concurrency_limit parameter controls how many messages are delivered simultaneously to the same local recipient. The recommended limit is low because delivery to the same mailbox must happen sequentially, so massive parallelism is not useful. Another good reason to limit delivery concurrency to the same recipient: if the recipient has an expensive shell command in her .forward file, or if the recipient is a mailing list manager, you don't want to run too many instances at the same time.

A destination concurrency limit of 10 for SMTP delivery seems enough to noticeably load a system without bringing it to its knees. Be careful when changing this to a much larger number.

Recipient limits

The default_destination_recipient_limit parameter (default: 50) controls how many recipients a Postfix delivery agent (smtp, uucp, etc.) will send with each copy of an email message. If an email message has more than $default_destination_recipient_limit recipients at the same destination, the list of recipients will be broken up into smaller lists, and multiple copies of the message will be sent.

You can override this setting for specific Postfix delivery agents (smtp, uucp, etc.). For example:

```
uucp_destination_recipient_limit = 100
```

would limit the number of recipients per UUCP delivery to 100.

You must be careful when increasing the recipient limit; some SMTP servers abort the connection when they run out of memory or when a hard recipient limit is reached, so the mail won't get through.

The smtpd_recipient_limit parameter (default: 1000) controls how many recipients the SMTP server will take per delivery. That's more than any reasonable SMTP client would send. The limit exists just to protect the local mail system against a malicious or confused client.

Always postponing delivery

The defer_transports parameter allows you to specify what mail should always be deferred until Postfix is explicitly asked to deliver.

A small site that is on-line only part of the time, and that wants to defer all deliveries until the command sendmail -q is executed (e.g., from a PPP dialout script) would use:

```
defer_transports = smtp
```

An ISP can use the defer_transports feature for customers that are off-line most of the time. The customer can trigger delivery by issuing an ETRN command at the SMTP port. The following examples show how to configure such a customer:

/etc/postfix/main.cf:

```
defer_transports = hold
```

```
  You can specify any number of transports here. The example gives just one.
```

/etc/postfix/transport:

```
  customer.com   hold:[gateway.customer.com]
  .customer.com   hold:[gateway.customer.com]
```

```
  The [] are necessary to avoid MX lookups, which might point to your local
machine. The second entry is necessary only if you want to relay mail for
customer subdomains.
```

/etc/postfix/master.cf:

```
  hold   unix   -   -   n   -   -   smtp
```

```
  This is just the master.cf entry for regular SMTP, with the first field
changed to hold.
```

Backoff from unreachable hosts

When a Postfix delivery agent (smtp, local, uucp, etc.) is unable to deliver a message it may blame the message itself or the receiving party.

```
  If the delivery agent blames the message, the queue manager gives the
queue file a time stamp into the future, so it won't be looked at for a
while. By default, the amount of time to cool down is the amount of time
that has passed since the message arrived. This results in so-called
exponential backoff behavior.
```

```
  If the delivery agent blames the receiving party (for example a local
recipient user, or a remote host), the queue manager not only advances the
queue file time stamp, but also puts the receiving party on a "dead" list so
that it will be skipped for some amount of time.
```

As you would expect, this whole process is governed by a bunch of little parameters.

queue_run_delay (default: 1000 seconds)

```
  How often the queue manager scans the queue for deferred mail.
```

maximal_queue_lifetime (default: 5 days)

```
  How long a message stays in the queue before it is sent back as
undeliverable.
```

minimal_backoff_time (default: 1000 seconds)

```
  The minimal amount of time a message won't be looked at, and the minimal
amount of time to stay away from a "dead" destination.
```

maximal_backoff_time (default: 4000 seconds)

```
  The maximal amount of time a message won't be looked at after a delivery
failure.
```

qmgr_message_recipient_limit (default: 1000)

```
  The size of many in-memory queue manager data structures. Among others,
this parameter limits the size of the short-term, in-memory "dead" list.
Destinations that don't fit the list are not added.
```

Slowing down bad clients

First of all, no defense will protect against an all-out denial of service attack. I just don't want to raise impossible expectations. But there are a few simple things one can do in order to deal with confused or malicious client programs.

Some defenses are part of a more general strategy: for example, how long a line of text may be before it is broken up into pieces, and how much text may be carried in a multi-line message header. See the resource controls documentation for details.

The Postfix SMTP server increments a per-session error counter whenever a client request is unrecognized or unimplemented, or whenever a client request violates UCE restrictions or other reasons. The error counter is reset when a message is transferred successfully.

As the per-session error count increases, the SMTP server changes behavior. The idea is to limit the damage by slowing down the client. The behavior is controlled by the following parameters:

smtpd_error_sleep_time (default: 5 seconds)

```
  When the per-session error count is small, the SMTP server pauses only
when reporting a problem to a client. The purpose is to prevent naive
clients from going into a fast connect-error-disconnect loop.
```

smtpd_soft_error_limit (default: 10)

```
  When the per-session error count exceeds this value, the SMTP server
sleeps error_count seconds before responding to a client request.
```

smtpd_hard_error_limit (default: 100)

```
  When the per-session error count exceeds this value, the SMTP server
disconnects.
```

Unfortunately, the Postfix SMTP server does not yet know how to limit the number of connections from the same client, other than by limiting the total number of SMTP server processes (see process limit). Things could be worse: some mailers don't even implement an SMTP server process limit. That's of course no excuse. I'm still looking for a good solution.