Some thoughts on MTA architecture

http://dotat.at/writing/mta-arch

Tony Finch
⟨fanf2@cam.ac.uk⟩ ⟨dot@dotat.at⟩

University of Cambridge
Computing Service
Mail Support

Monday 2 June 2008

UNIVERSITY OF CAMBRIDGE UCS

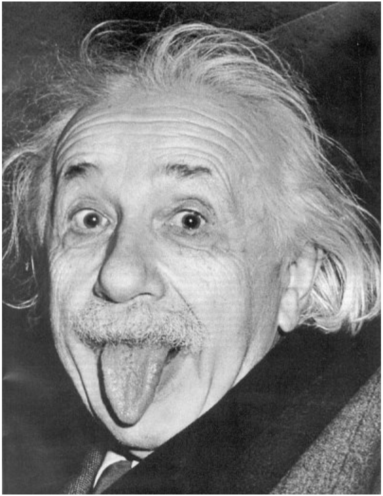
About me

1994 – 1997	⟨fanf2@cam.ac.uk⟩	computer science
1997 – 2000	⟨fanf@demon.net⟩	web server admin
2000 – 2001	⟨fanf@covalent.net⟩	Apache httpd coder
2002 – now	⟨fanf2@cam.ac.uk⟩	postmaster

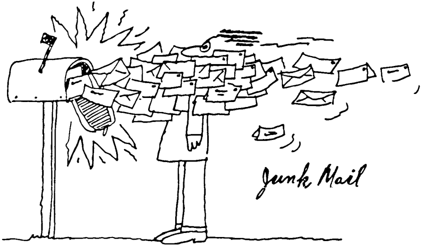
1997 . . .	⟨dot@dotat.at⟩	
1999 . . .	⟨fanf@apache.org⟩	httpd
2002 . . .	⟨fanf@FreeBSD.org⟩	unifdef
2004 . . .	⟨fanf@exim.org⟩	
2006 . . .	⟨fanf@apache.org⟩	SpamAssassin

“Wouldn’t it be nice if. . . ?”

- theoretical musings on MTA architecture
- originally a series of postings on my blog, Feb 2006 – March 2007
- there is no code and no likelihood of code



A snapshot of the problem



Average email traffic
(legitimate and spam):

Mar 2005	15
Mar 2006	20
Mar 2007	35
Mar 2008	80

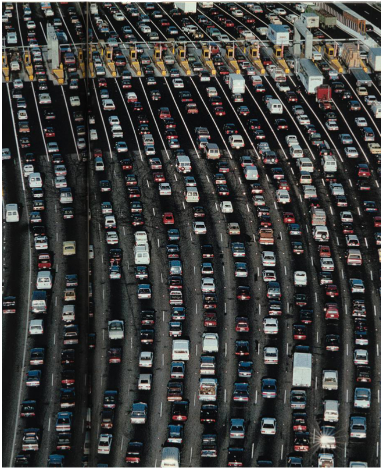
- all numbers in messages (or rejections) per second

Current traffic classification:

relay attempts	0.5 – 1.5
known malware	2 – 4
blacklisted	60 – 75
invalid recipient	1.5
invalid sender	1.2
SpamAssassin	2
legitimate email	3
internal email	2.5

Concurrency

- concurrency requirements grow with spam volumes
- most MTAs use an OS process per connection
- really inefficient!



Waste vs efficiency

- event-driven connection multiplexing
- high-level languages with lightweight threads

better software performance ⟹ better hardware efficiency

Waste vs efficiency

- best use of the available resources . . .

Some partial solutions

- SAUCE – software against UCE
  http://www.chiark.greenend.org.uk/~ian/sauce/
  (written in Tcl)
- qpsmtpd-async – anti-spam smtpd for qmail
  http://smtpd.develooper.com/
  (written in Perl)
- MailChannels Traffic Control™
  http://www.mailchannels.com/products/traffic-control.html

Address verification

- most verifications are for messages that will be rejected
- email address routeing can be arbitrarily complicated so verification can be too!
- concurrency useful for multi-recipient messages as well as multiple messages

Avoid bouncing

- reject unwanted email as early as possible
- try hard not to accept and bounce
- reduce spam backscatter & forwarded spam
- avoid wasting your MTA's resources

How email addresses are routed

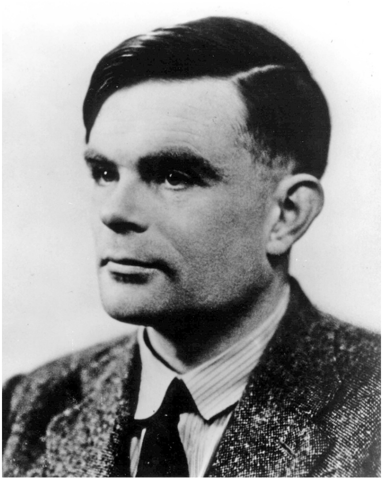
- DNS — MX/A/AAAA
- flat files — text or cdb
  - `aliases`
  - `mailertable`
  - `virtusertable`
- LDAP — “laser” schema
- SQL databases

User-defined filtering

- Sieve — RFC 5228
- address validity can be conditional on the sender's address
- selective sub-address validity, e.g. `fanf9+subaddress@hermes.cam.ac.uk`

Routeing with regular expressions

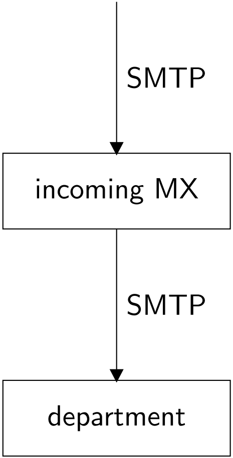
- try to match address against a series of regular expressions
- when one matches, replace address with corresponding result
- interpolate captured subexpressions
- route resulting address, repeating `regsub` if necessary



Verification: you're doing it wrong!

Postfix `local_recipients_map`

Verifying relayed addresses



SMTP

incoming MX

SMTP

department

Verification: you're doing it wrong!

- copy table of valid recipients from department to MX
- configure MX to query department's LDAP directory

Call-forward recipient verification

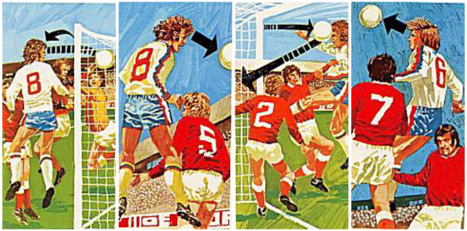
```
220 mx.cam.ac.uk
HELO dotat.at
250 Hello
MAIL FROM:<dot@dotat.at>
250 OK
RCPT TO:<?@cl.cam.ac.uk>  220 mta.cl.cam.ac.uk
                          HELO mx.cam.ac.uk
                          250 Hello
                          MAIL FROM:<dot@dotat.at>
                          250 OK
                          RCPT TO:<?@cl.cam.ac.uk>
                          550 Unknown user
550 Unknown user          QUIT
RSET                      221 Goodbye
...
```

Content scanning

- anti-spam
- anti-phishing
- anti-virus
- lots of CPU
- lots of memory

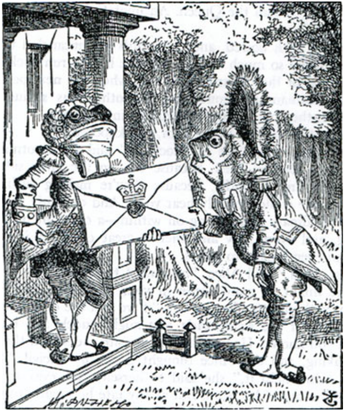
Content scanning goals

- decouple scanner from client concurrency & speed
- do not require entire message to be buffered in RAM
- avoid temporary on-disk buffers
- security boundary between content scanner(s) and MTA



Data callout

- use the normal
  local delivery mechanism
- efficiently transfer a file
  from the queue to a program
- cross security boundaries
- control concurrency
  and smooth load spikes



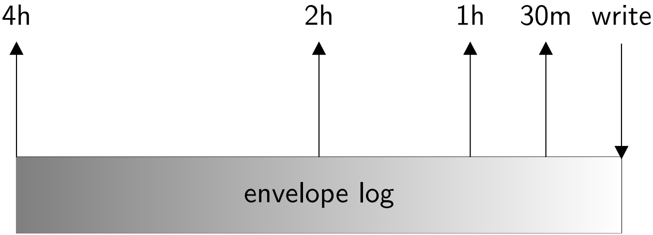
Queue layout

- MTAs typicall scatter messages all over the disk
- often separate files for envelopes and contents
- this makes queue runs particularly expensive

Log-structured queue

- write all metadata sequentially to one file
- queue runners read file sequentially
- updated envelopes also appended to the file
- queue runners act as garbage collectors
- size of log bounded by retry interval

Log-structured queue



Architectural principles

- lightweight concurrency througout the system
- load smoothing / scheduling of scarce resources
  - database connections, content scanners
- address routeing is verification
- content scanning is a data call-forward
- a log-structured queue minimizes disk seeks

That's all, folks!

- slides and notes available online:
  `http://dotat.at/writing/mta-arch`
- any questions?