



IBM Research

Postfix, past present and future

Wietse Venema
IBM T. J. Watson Research Center
Hawthorne, NY, USA

Postfix in a nutshell

- Who runs Postfix:
 - Providers with 10+M mailboxes (Outblaze, UOL).
 - Desktops & servers (MacOS X, Ubuntu, NetBSD).
 - Appliances (Brightmail, Frontbridge, etc.).
- Who provides Postfix:
 - Yours truly, helped by small number of volunteers with input from a small core of active users.
 - Code contributions are accepted. Sometimes as is, usually after some editing.

Overview

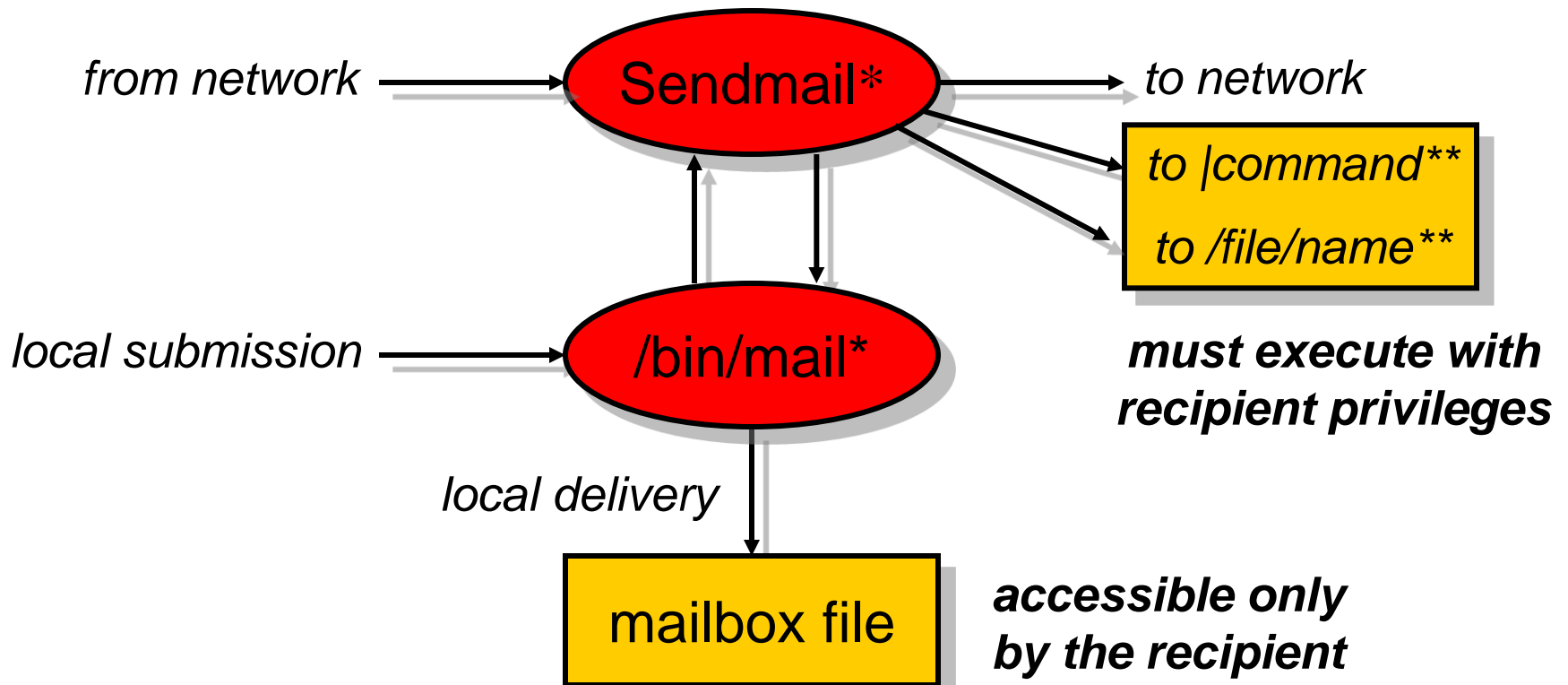
- Why (not) write another UNIX mail system.
- Postfix low-level implementation.
- Extensibility as a proverbial life saver.
- Catching up on Sendmail.
- Recent development: postscreen zombie blocker.
- Conclusion.

- Why (not) write yet another UNIX mail system

Idealism versus real-world compatibility.

Traditional (BSD) UNIX mail delivery architecture

(impersonation requires privileges; monolithic model hinders damage control)



* uses "root" privileges

** in per-user .forward files and in per-system aliases database

Plan for failure

CERT/CC UNIX email advisories (part 1 of 3)

Bulletin	Software	Vulnerability
CA-1988-01	Sendmail 5.58	run any command
CA-1990-01	SUN Sendmail	unknown
CA-1991-01	SUN /bin/mail	root shell
CA-1991-13	Ultrix /bin/mail	root shell
CA-1993-15	SUN Sendmail	write any file
CA-1993-16	Sendmail 8.6.3	run any command
CA-1994-12	Sendmail 8.6.7	root shell, r/w any file
CA-1995-02	/bin/mail	write any file

CERT/CC UNIX email advisories (part 2 of 3)

Bulletin	Software	Vulnerability
CA-1995-05	Sendmail 8.6.9	any command, any file
CA-1995-08	Sendmail V5	any command, any file
CA-1995-11	SUN Sendmail	root shell
CA-1996-04	Sendmail 8.7.3	root shell
CA-1996-20	Sendmail 8.7.5	root shell, default uid
CA-1996-24	Sendmail 8.8.2	root shell
CA-1996-25	Sendmail 8.8.3	group id
CA-1997-05	Sendmail 8.8.4	root shell

CERT/CC UNIX email advisories (part 3 of 3)

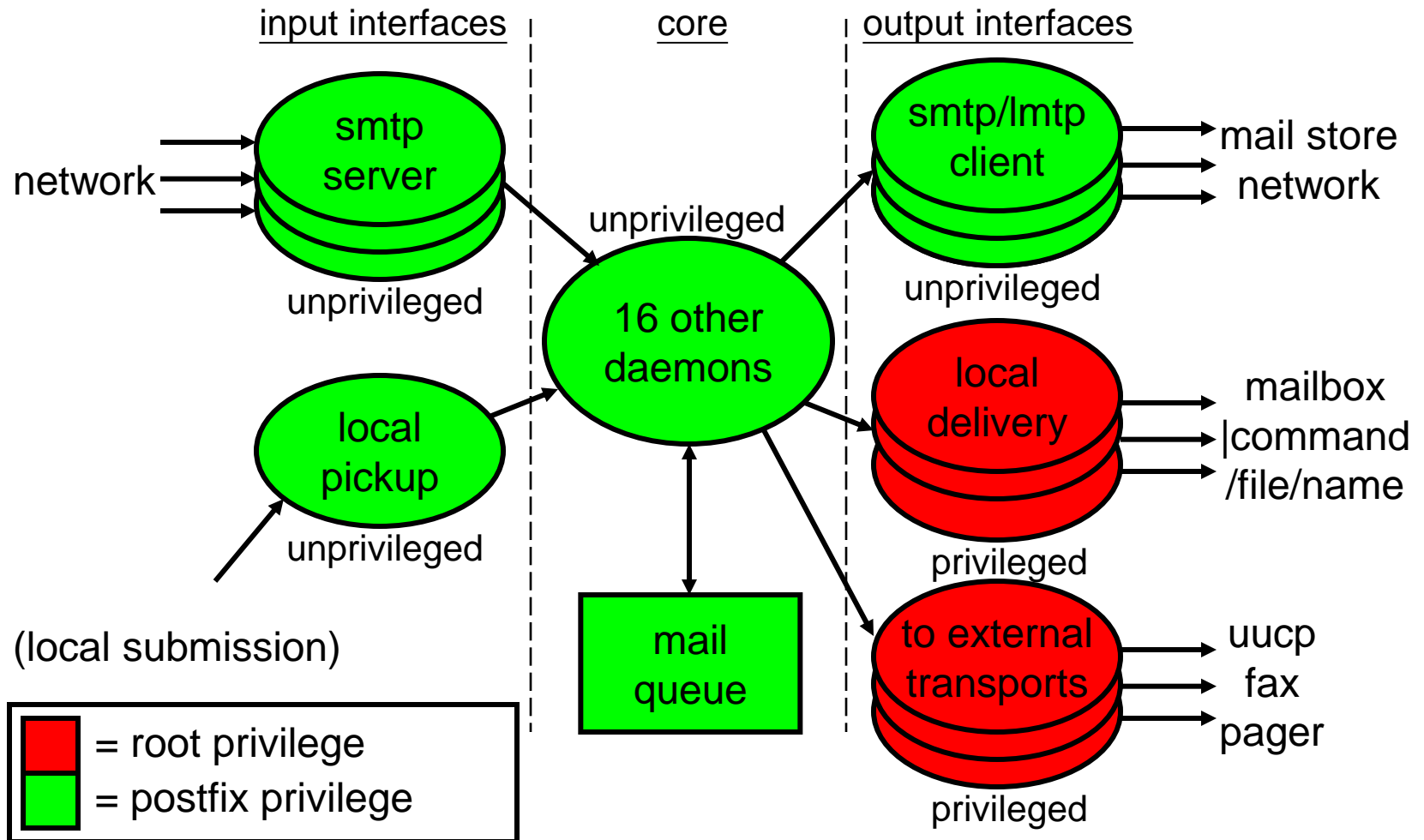
Bulletin	Software	Vulnerability
CA-2003-07	Sendmail 8.12.7	remote root privilege
CA-2003-12	Sendmail 8.12.8	remote root privilege
CA-2003-25	Sendmail 8.12.9	remote root privilege

Monolithic and privileged: no damage control

- One mistake can be fatal:
 - A remote client can execute any command with “root” privilege, or can read/write any file they choose.
- No internal barriers:
 - Very convenient to implement (not really, see later).
 - Very convenient to break into (yes, really).

Postfix distributed security architecture

(omitted: non-daemon programs for submission and system management)



Plan for failure

Major influences on Postfix architecture

- **TIS Firewall smap/smapd**
 - Low privilege, chroot jail, “air gap” between mail receiving and mail delivering processes.
- **qmail**: parallel deliveries, maildir file format.
- **Apache**: reuse processes multiple times.
- **Sendmail**
 - User interface; lookup table interface; some things to avoid.
- **Network routers**
 - Multiple interface types, but no queue-skipping fast path :-(

- Postfix low-level implementation

“I learned to program carefully for selfish reasons. I did not want to sleep on the floor next to my physics experiments”.

Wietse, date unknown

Optimization is the root of evil

- When a server is exposed to the Internet, the worst case will become the normal case, *and vice versa*.
 - Postfix is designed to deliver mail fast.
 - Not optimal when >90% of mail is spam ☹
 - Postfix assumes that SMTP clients move quickly.
 - Buggy Storm zombies clog up all SMTP server ports.
- Don't improve the common case at the cost of the worst case (example: Postfix content filter).

How to implement SMTP without screwing up

- Multi-protocol: SMTP/DNS/TLS/LDAP/SQL/Milter.
- Broken implementations: clients, servers, proxies.
- Concurrent mailbox “database” access.
- Complex email address syntax <@x,@y:a%b@c>.
- Queue management (thundering herd).
- SPAM and Virus control.
- Anti-spoofing: DKIM, SenderID, etc., etc.

Postfix implementation

Strategies: divide and conquer

Juggle fewer balls, basically

- Partitioned “least privilege” architecture.
- More-or-less safe extension mechanisms:
 - Use SMTP or “pipe-to-command” for content inspection; let other people provide applications that do the work.
 - Simple SMTP access policy protocol; let other people provide SPF, greylist etc. applications.
 - Adopt Sendmail Milter protocol; let other people provide DKIM, SenderID etc. applications.
- More-or-less safe C programming API (example).

Postfix implementation

Example: buffer overflow defense

- 80-Column punch cards become obsolete years ago.
 - Fixed buffers are either too small or too large.
- Dynamic buffers are not the whole solution.
 - IBM httpd (and qmail 1.03 on contemporary platforms):

```
forever { send "XXXXXX...." ; }
```
- Postfix has bounds on memory object count and size.
 - Don't run out of memory under increasing load.

- Adding anti-spam/virus support, part 1:
Use standard protocols where you can.

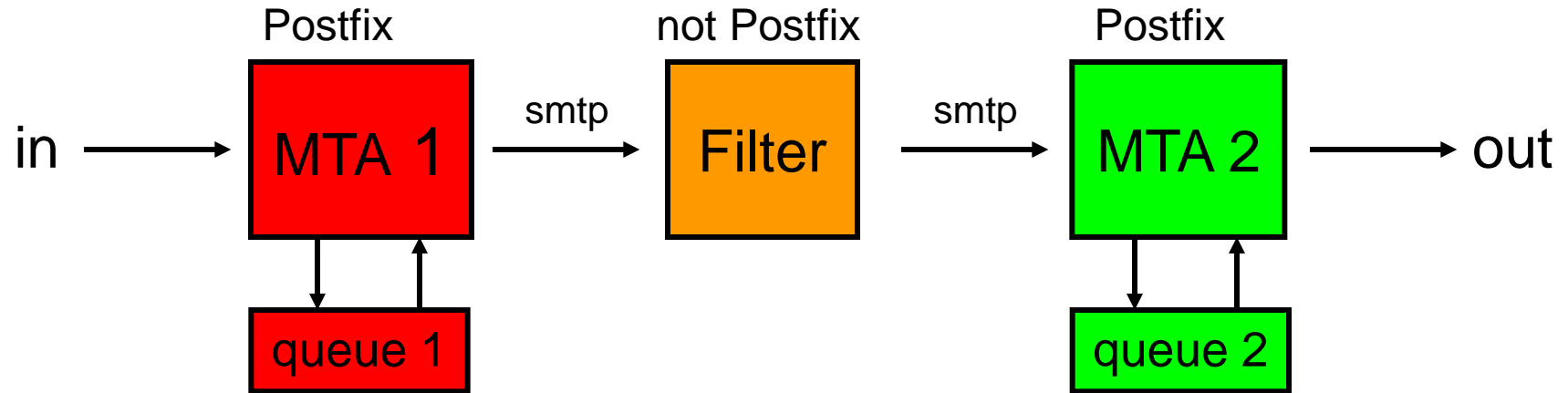
“Junk mail is war. RFCs do not apply.”

Wietse on Postfix mailing list, 2001

1999 - Melissa ravages the Internet

- You can run from Windows but you can't hide: Postfix becomes a vehicle for malware distribution.
 - *Short term*: block “known to be bad” strings in message.
`/^Subject:.*Important Message From/ REJECT`
 - *Long-term*: *delegate* deep inspection to third-party code.
- Emergence of specialized protocols: CVP, Milter, etc.
 - We already use SMTP for email distribution world-wide.
 - Why can't we also use SMTP to plug in anti-spam/virus?

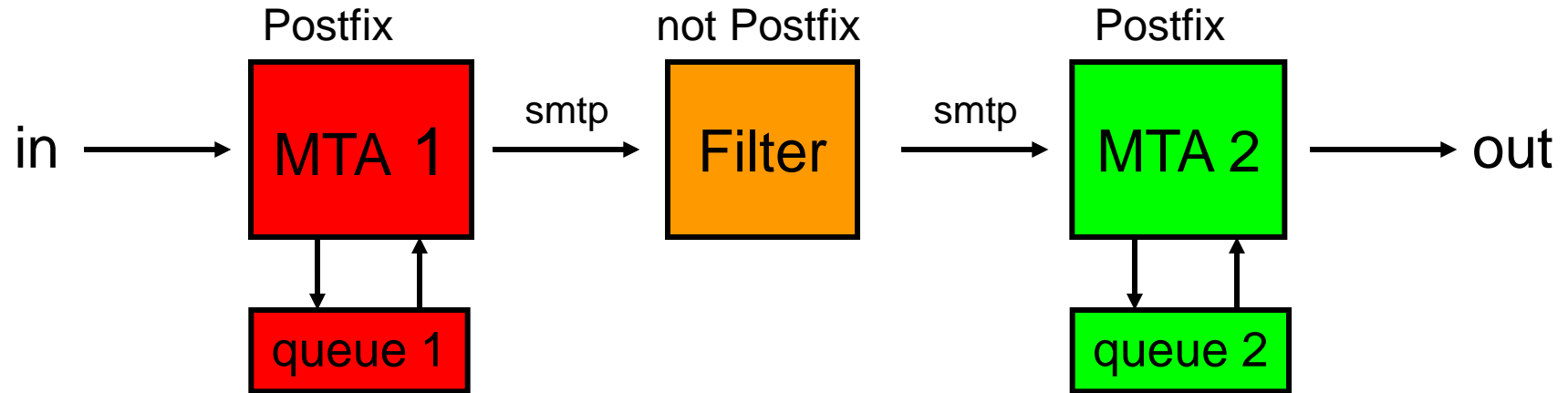
Postfix content filter via SMTP (after-queue)



- MTA = Mail Transport Agent.
- Red = dirty, green = clean.
- But it can't be that simple, right?
- Using two MTAs must be wasteful!

Invent sparingly

After-queue content filter support

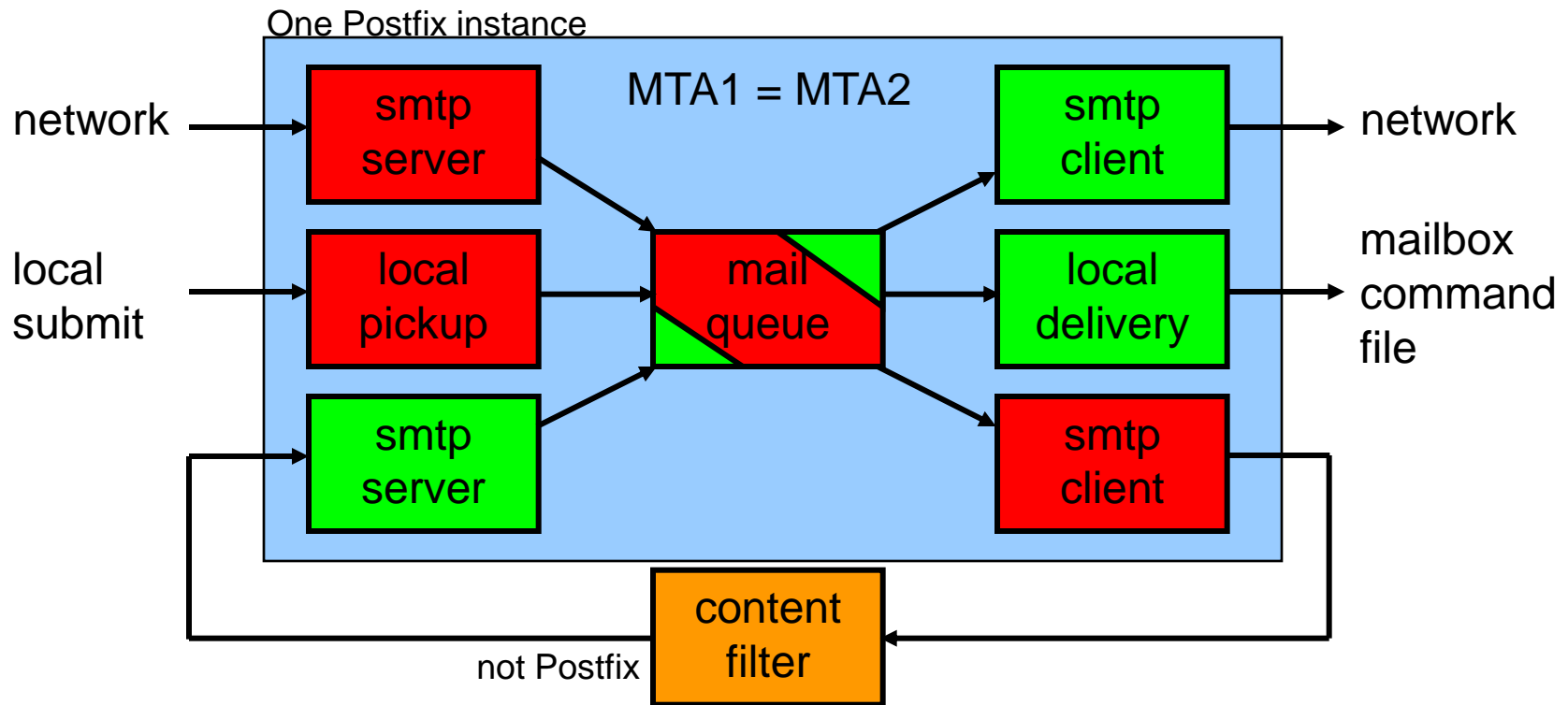


- Advantage of after-queue content filters:
 - Performance: 10 after-queue filter processes can handle the traffic from 100 before-queue SMTP sessions.
- Disadvantage: after-queue filter must *quarantine* or *discard* bad mail, instead of *reject* (don't become a backscatter source).
 - Problem: discarding mail is problematic e.g. in Europe.

Invent sparingly

Postfix content filter via SMTP (after-queue)

Two MTAs combined into one – optimization is the root of evil

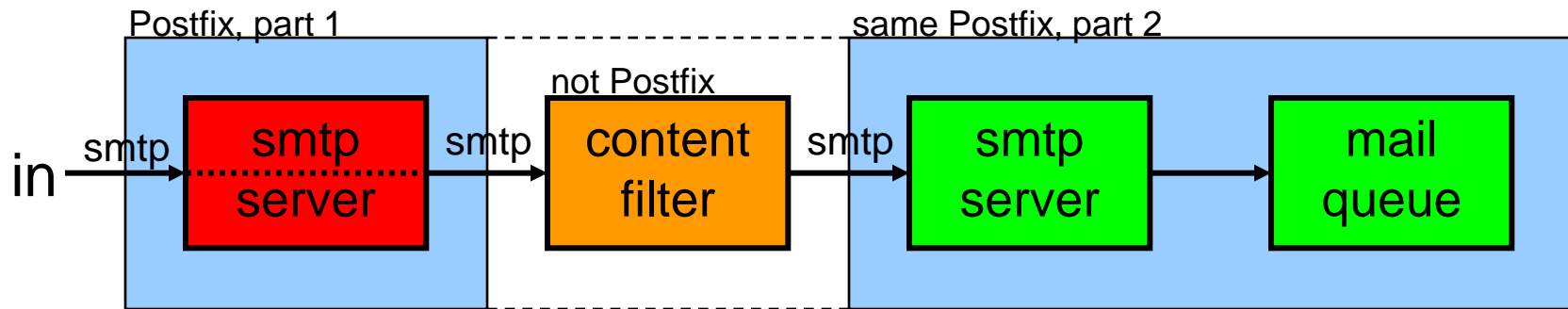


- Combining two MTAs into one increases complexity.
 - Two MTA behaviors, but only one set of configuration files.

Invent sparingly

Before-queue content inspection via SMTP

Responding to popular demand, despite performance limitation



- Before-queue spam/virus filtering is needed in Europe.
 - Reject bad mail before it is accepted into the mail queue.
 - Once you accept mail, you can't discard it.
- One content filter per SMTP client is expensive.
 - Reduce filter count with ~40%, by receiving the complete message before delivering it to the filter.

Invent sparingly

- Adding anti-spam/virus support part 2:
Embrace de-facto standards.

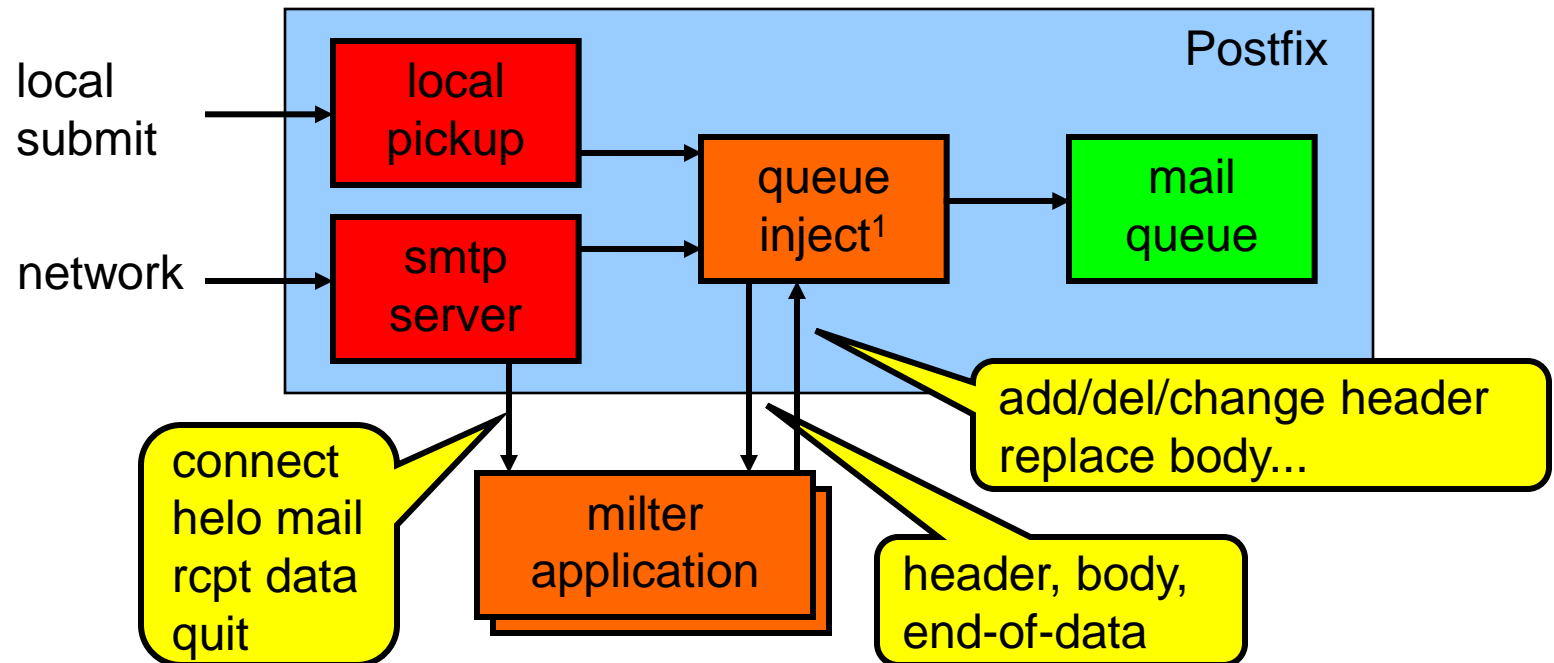
“It's not the spammers who destroy [email], it's well-meaning people who insist on broken anti-spam measures.”

Wietse on Postfix mailing list, 2003

2005 - Proliferation of authentication technologies

- *Alphabet soup*: SPF, SenderID, DomainKeys, DKIM, BATV, SRS, ADSP, and the end is not in sight.
 - Building everything into Postfix is not practical.
 - Some distributions are two or more years behind on Postfix.
 - Using SMTP-based filters to sign or verify is overkill.
- *Solution*: adopt Sendmail Milter protocol and open up access to a large collection of available applications.

Retrofitting Milter support into a distributed MTA



- Red = dirty, green = clean.
- The effort was heroic, but the reward was sweet.

¹With local submission, sends surrogate connect/helo/mail/etc events

Plan for change

Postfix author receives Sendmail innovation award

MOUNTAIN VIEW, Calif. October 25th, 2006 Today at its 25 Years of Internet Mail celebration event, taking place at the Computer History Museum in Mountain View, California, Sendmail, Inc., the leading global provider of trusted messaging, announced the recipients of its inaugural Innovation Awards.

...

Wietse Venema, author, for his contribution of extending Milter functionality to the Postfix MTA.

http://www.sendmail.com/pdfs/pressreleases/Sendmail%20Innovation%20Awards_10%2025%2006_FINAL.pdf

Plan for change

- Catching up on Sendmail

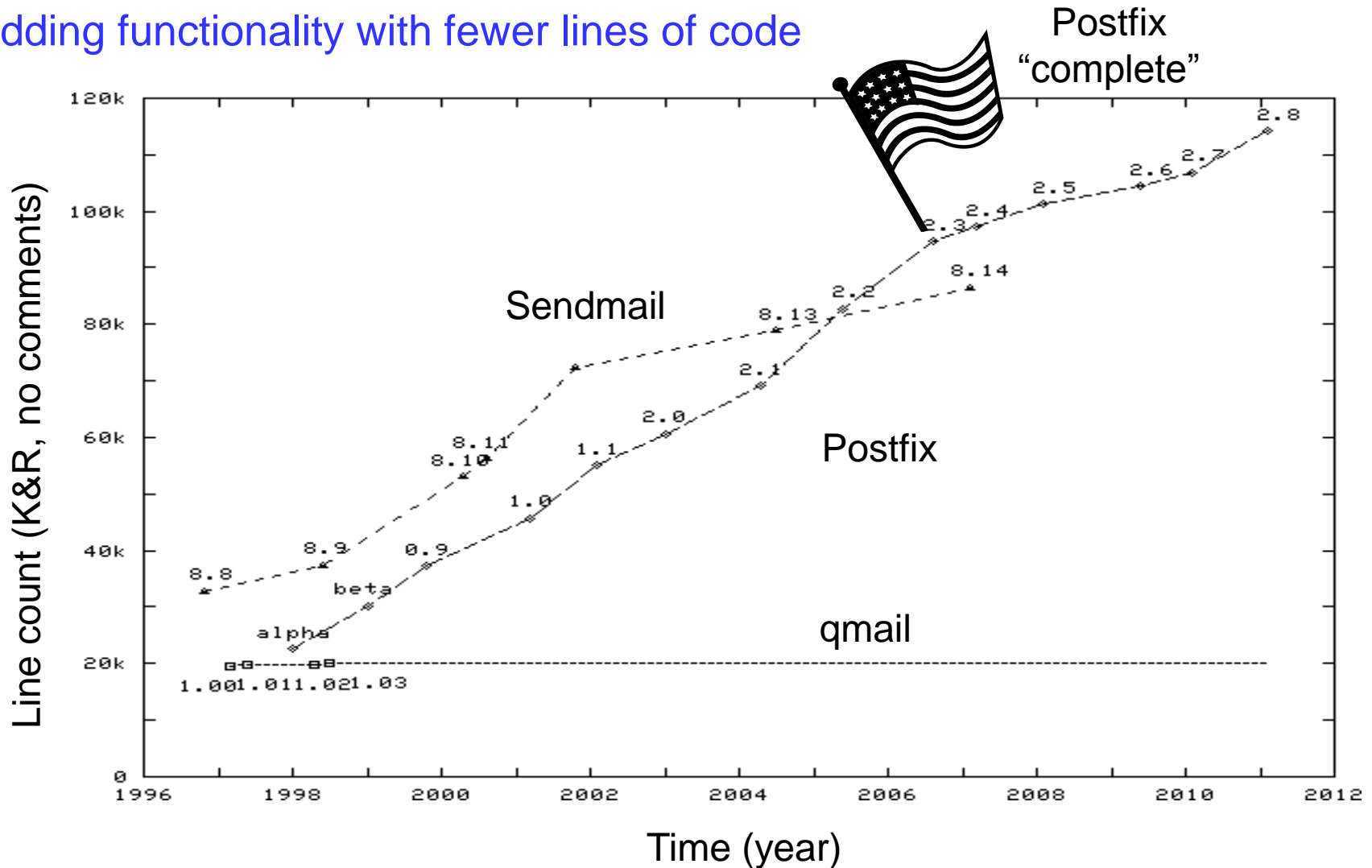
Why Postfix did not become a bloated mess.

How Postfix has grown in size

- Initial trigger: the Postfix source tar/zip file was *larger* than the Sendmail source tar/zip file.
 - Marcus Ranum asked if I had dragged in an XML parser.
- Analyze Sendmail, Postfix, and qmail source code:
 - Strip comments (*shrinking* Postfix by 45% :-).
 - Format according to “Kernighan and Ritchie” style (*expanding* qmail by 25% :-).
 - Delete repeating (mostly empty) lines.

MTA Source lines versus time

Adding functionality with fewer lines of code



Catching up on Sendmail

Why Postfix did not become a bloated mess

Benefits from a partitioned architecture

- Small programs are easier to maintain.
 - That is, after you build the communication infrastructure.
 - Minor features: easier to *modify* a small program.
 - Major features: easier to *add* a small program.
 - Present breakdown: 25 daemons, 13 commands.
 - Small is a relative term.
 - The SMTP server daemon now weighs in at almost 10k lines, half the size of the entire Postfix alpha release.

- Recent developments

Developments since the 2009 conference

See [RELEASE_NOTES-2.*](#) for details

- Postfix 2.7:
 - Speed-adjust (before-queue content filter)
 - Receive the entire message before connecting to content filter.
 - Per-sender IP reputation.
 - Use different source IP address for different sender class.
 - `milter_header_checks` (on request by AOL).
- Postfix 2.8:
 - DNSWL support, DNSBL/WL reply filter patterns.
 - `smtpd_reject_footer` (helpful information in “reject” messages).
 - `postscreen` zombie blocker.

- Current developments

“Zombies suck the life out of the mail server.”

Wietse at mailserver conference, 2009

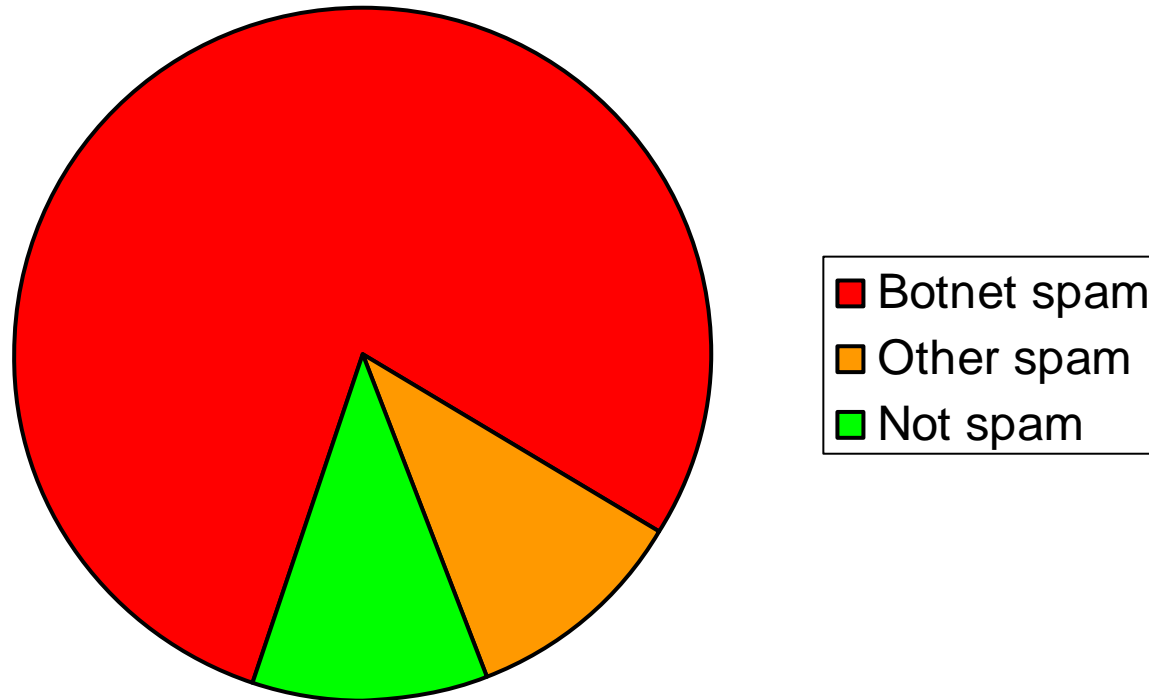
Changing threats

- 1999: You built a mail system that runs on UNIX, so you didn't have to worry about Windows viruses.
 - *Problem*: your UNIX-based mail system becomes a major distribution channel for Windows malware (Melissa).
 - *Solution*: outsource the job to external content filters.

Changing threats

- 2009: You built a mail system that has world-class email delivery performance.
 - *Problem*: your world-class performing mail system is now spending most of its resources *not delivering mail*.
 - *Solution*: work smarter. Stop the bots before the mail server.

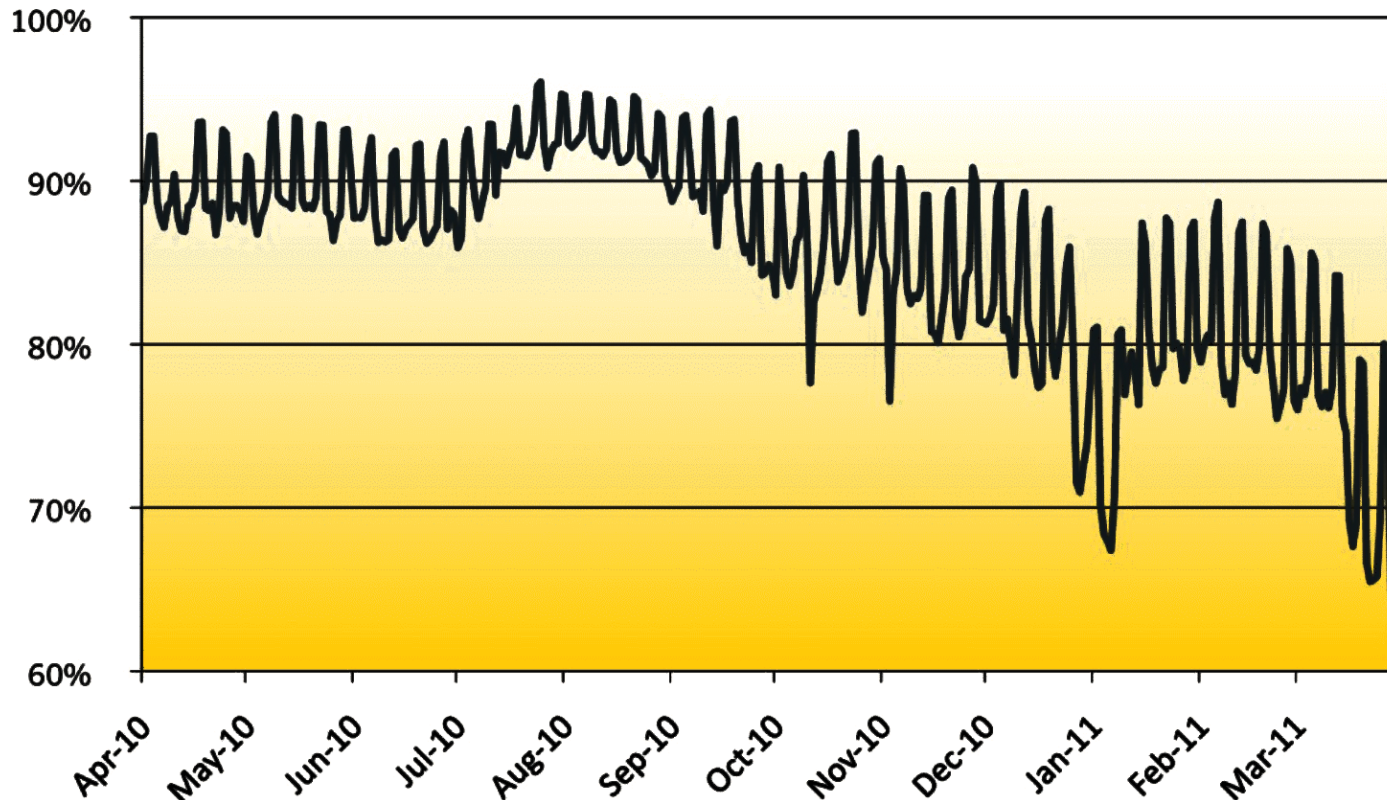
89% Of email is spam, 88% of spam from botnets



Source: MessageLabs Intelligence 2010 Annual Security Report

Global spam percentage drop from ~90% to ~70%

Major drops December + March when Rustock botnet stopped sending

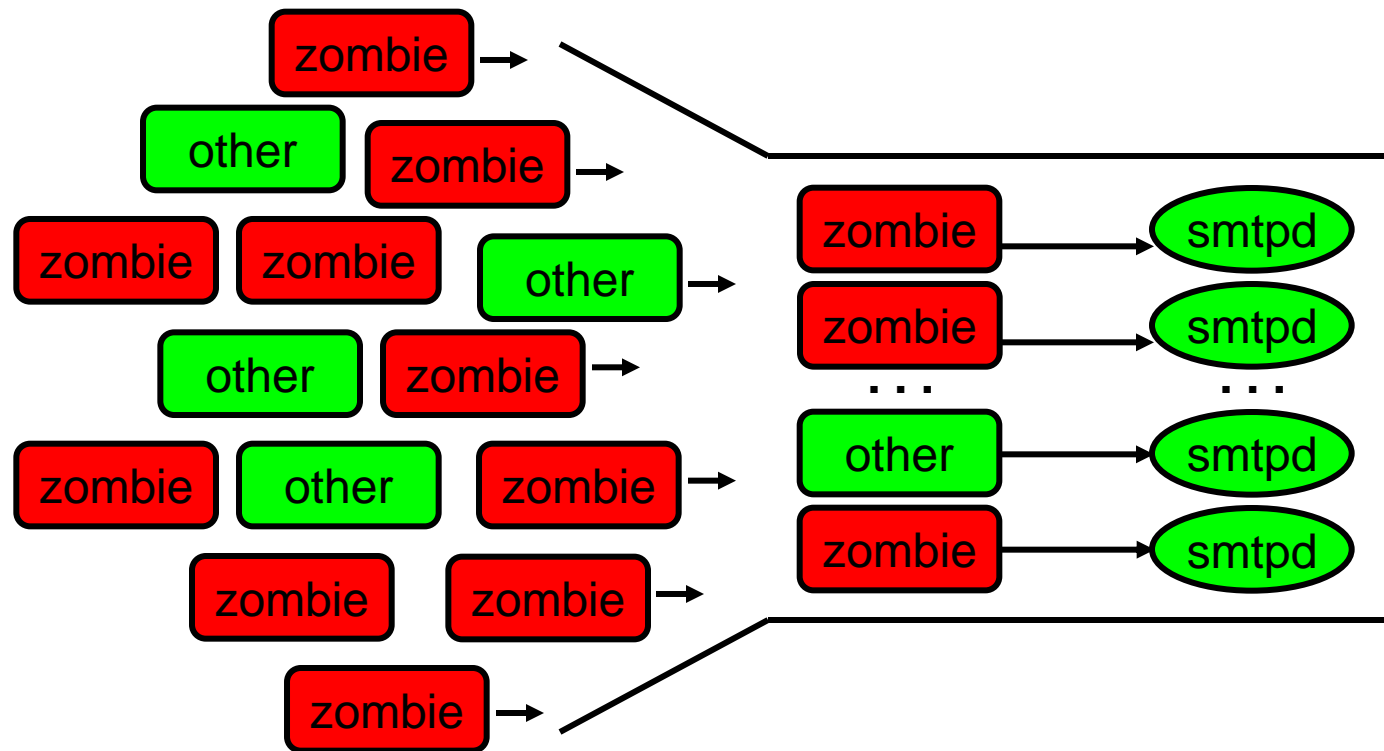


Source: Symantec state of spam and phishing report, April 2011

Zombies keep mail server ports busy

Connections waiting for service
(queued in the kernel)

Connections handled by server
(Postfix default: 100 sessions)



Changing threats

Zombies suck the life out of the mail server

- Worst-case example: Storm botnet.

```
13:01:36 postfix/smtpd: connect from [x.x.x.x]
```

```
13:01:37 postfix/smtpd: reject: RCPT from [x.x.x.x]:  
550 5.7.1 blah blah blah
```

```
13:06:37 postfix/smtpd: timeout after RCPT from [x.x.x.x]
```

- RFC 5321 recommends 5-minute server-side timeout.
 - Postfix implements SMTP according to the standard.
 - Result: all SMTP server ports kept busy by Storm zombies.

Mail server overload strategies

- Assumption: the zombie problem will get worse before things improve (if ever).
- Temporary overload:
 - Work faster: spend less time per SMTP client.
- Persistent overload:
 - Work harder: handle more SMTP clients (forklift solution).
 - Work smarter: stop spambots up-stream (postscreen).

Temporary overload strategy

- Work faster: spend less time per SMTP client.
 - Reduce time limits, number of rejected commands, etc.
 - Will delay *some* legitimate email, *temporarily*.
 - From sites with large network latency or packet loss.
 - From list managers with aggressive timeouts.
 - Better to receive *some* legitimate mail, than *no mail*.
 - OK as long as the overload condition is *temporary*.

Temporary overload implementation

- Postfix master(8) daemon detects “all SMTP ports busy” and updates SMTP daemon command lines¹:

```
smtpd -o stress=yes
```

- Default parameter settings (Postfix 2.6 and later):

```
smtpd_timeout = stress?10stress:300s  
smtpd_hard_error_limit = stress?1stress:20
```

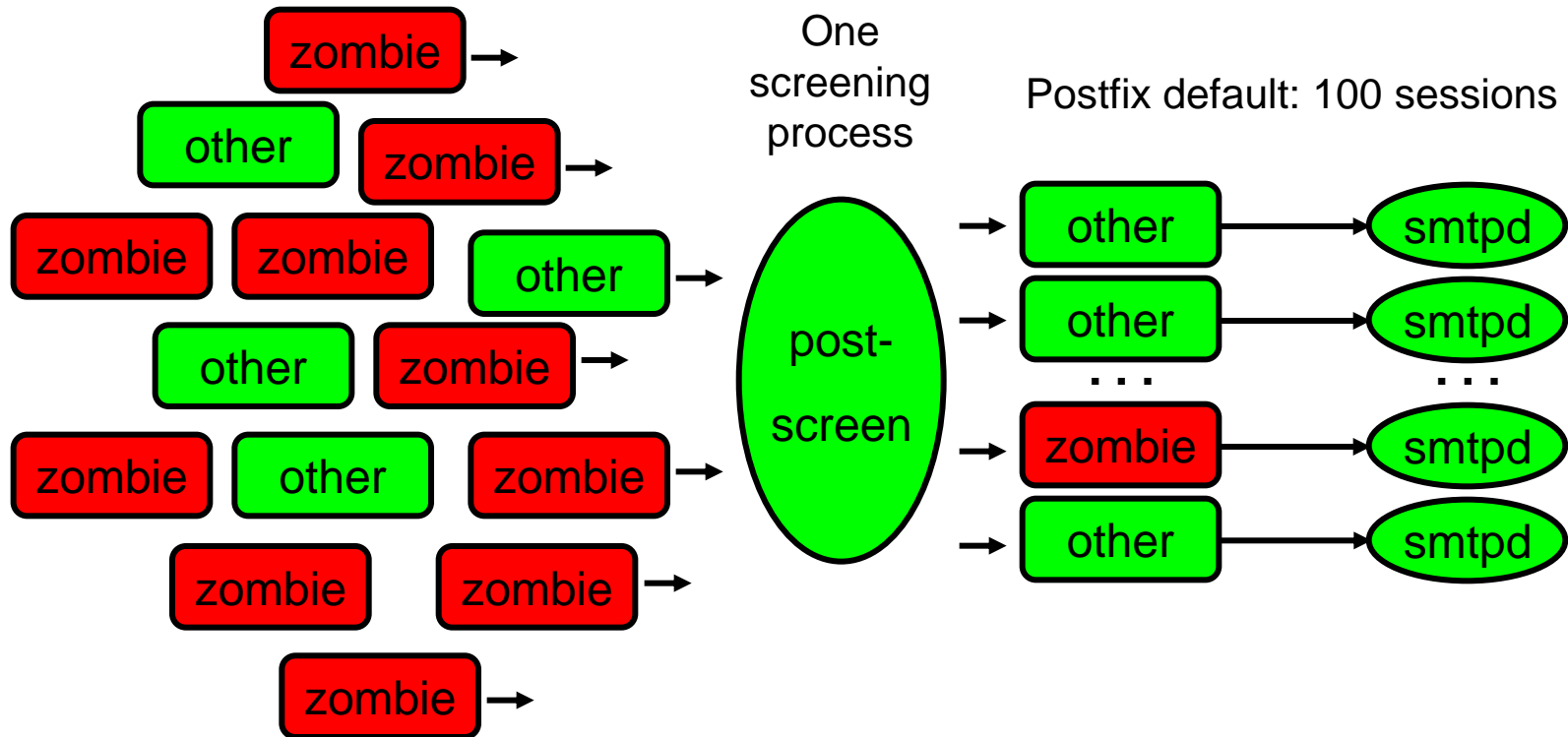
¹Feature is called “stress”, and implemented in 21 lines, because of author overload.

Persistent overload strategies

- Work harder: configure more mail server processes.
 - The brute-force, fork-lift approach for rich people.
 - OK if you can afford the network, memory, disk, and CPU.
- Work smarter: keep the zombies away from the server.
 - Before-server connection filter.
 - More SMTP processes stay available for legitimate email.

Persistent overload - before-smtpd connection filter

Prior work: OpenBSD spamd, MailChannels TrafficControl, M.Tokarev



Changing threats

Postscreen – reducing the cost of spam elimination

- First layer of a four-layer defense.
 - Postscreen connection filter (least expensive)
 - Block spambots that send ~90% of all SPAM.
 - Postfix SMTP server
 - Block SPAM with more complex SMTP-level access policies.
 - Header/body patterns
 - Block SPAM and backscatter with simple signatures.
 - Deep content inspection (most expensive)
 - Block SPAM that remains.
- This talk: survey of first-layer blocking techniques.

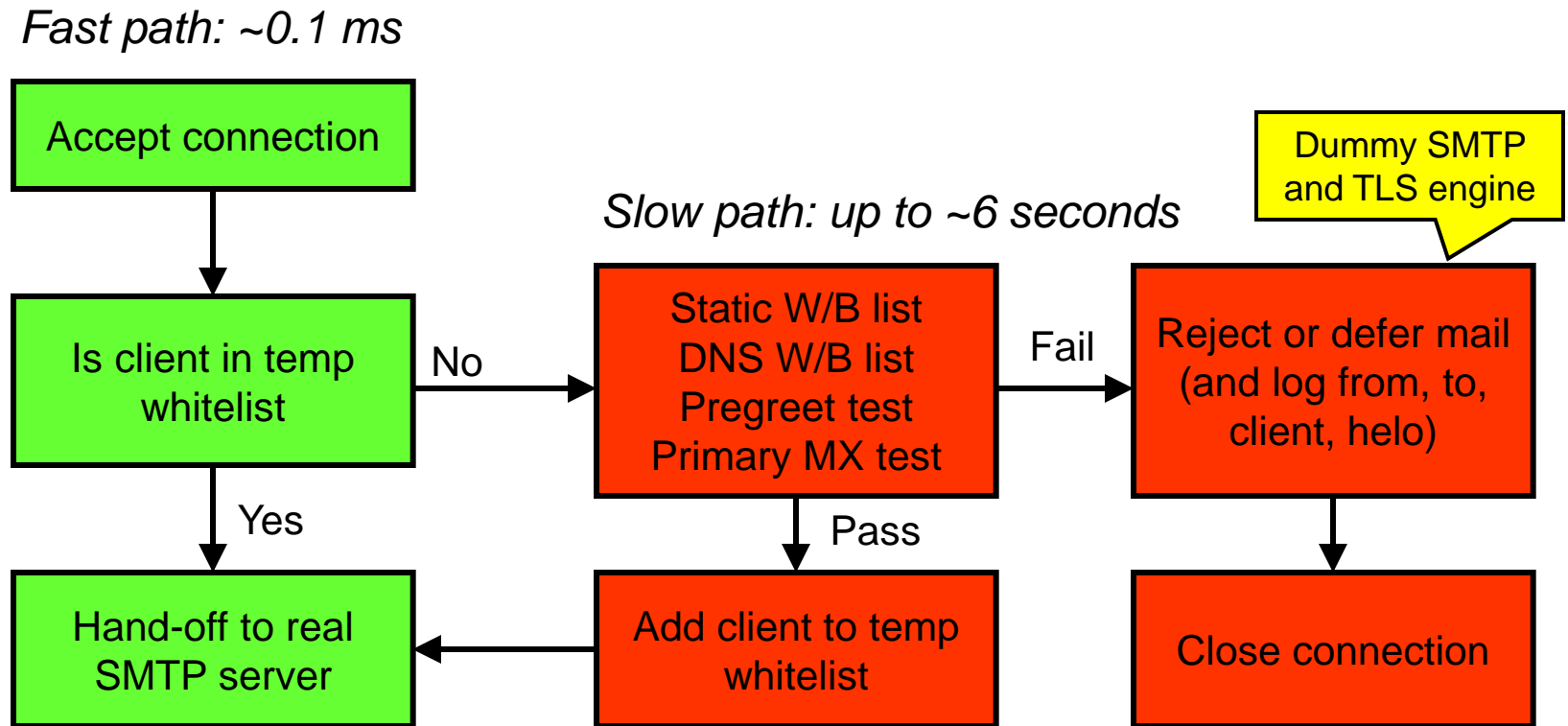
postscreen opportunities and challenges

- Zombies are blacklisted within hours¹.
 - Opportunity: reject clients that are in a hurry to send mail.
 - Pregreet, command pipelining, ignoring 5XX replies.
 - Connecting to backup MX hosts only.
 - Other protocol violations.
- Zombies avoid spamming the same site repeatedly.
 - Challenge: decide “it’s a zombie” for a single connection.
 - Use DNS white- and blacklists as shared intelligence source.

¹Chris Kanich et al., Spamalytics: An Empirical Analysis of Spam Marketing Conversion, CCS 2008.

Workflow - tests before the SMTP handshake

One daemon screens multiple connections simultaneously



Changing threats

Postscreen DNSBL/DNSWL support

- Parallel DNS lookups.
- Weight factors (to whitelist, use *negative* numbers).

```
postscreen_dnsbl_sites = zen.spamhaus.org*2, bl.spamcop.net*1,  
                        b.barracudacentral.org*1, dnsbl.njabl.org*1  
postscreen_dnsbl_threshold = 2
```

- Reply filters.

```
postscreen_dnsbl_sites = zen.spamhaus.org=127.0.0.4 ...  
postscreen_dnsbl_sites = zen.spamhaus.org=127.0.0.[1..11] ...
```


Detecting spambots that speak to early (pregreet)

- Good SMTP clients wait for the SMTP server greeting:

```
SMTP server: 220 server.example.com ESMTP Postfix<CR><LF>
```

```
SMTP client: EHLO client.example.org<CR><LF>
```

- Sendmail *greet_pause* approach: wait several seconds before sending the 220 greeting.
 - Very few clients greet spontaneously.
 - Some clients give up after waiting several seconds.
 - Poor performance under load (“all ports busy” condition).
 - Requires manual whitelisting.

Question for dog catchers

- Q: How do I quickly find out if a house has a dog?
- A: Ring the doorbell, and the dog barks immediately.



- postscreen uses a similar trick with botnet zombies.

Making zombies bark - multi-line greeting trap

- Good clients wait for the full multi-line server greeting:

```
postscreen: 220-server.example.com ESMTP Postfix<CR><LF>
```

```
smtp server: 220 server.example.com ESMTP Postfix<CR><LF>
```

```
good client: HELO client.example.org<CR><LF>
```

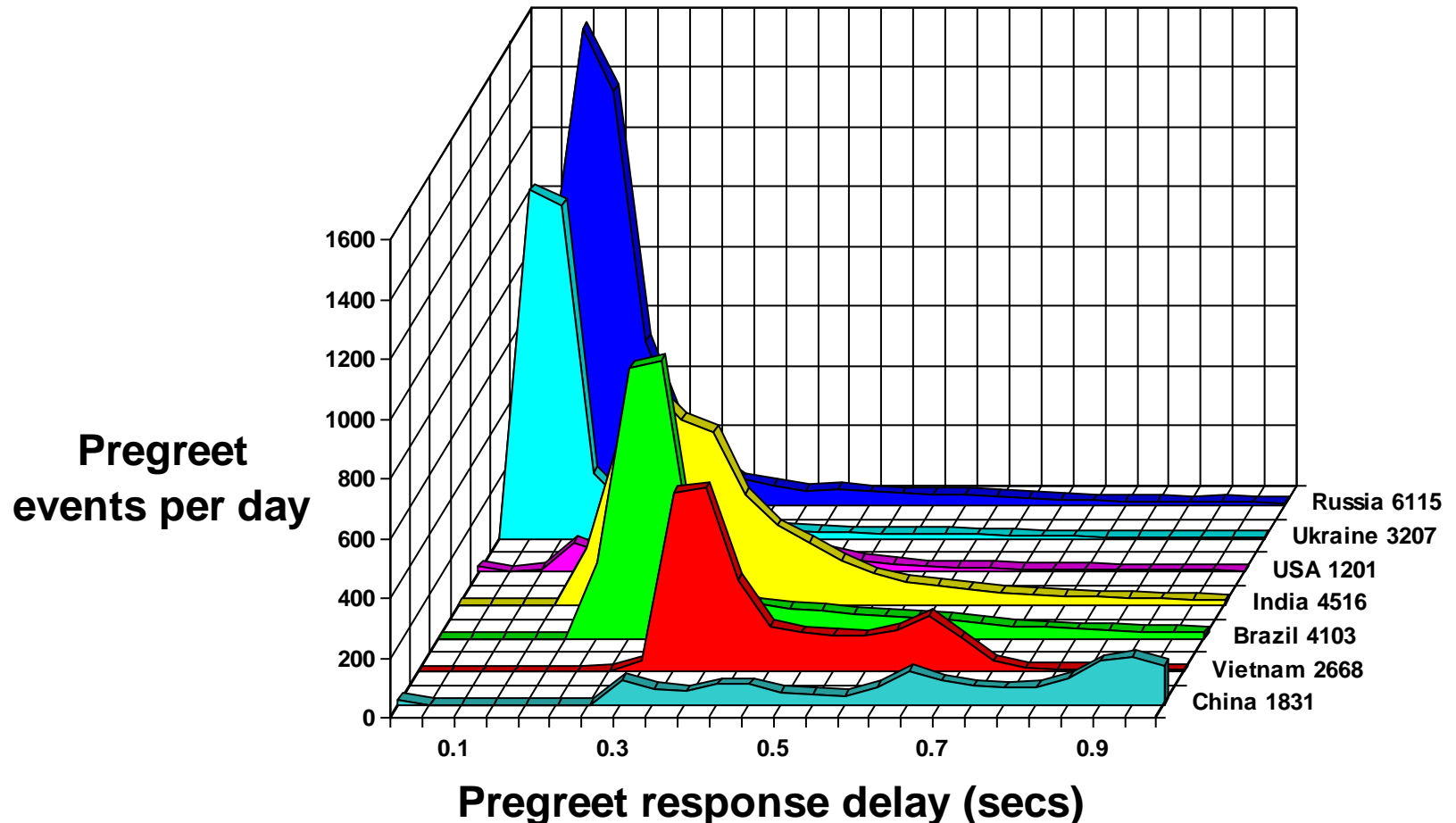
- Many spambots talk immediately after the first line of the multi-line server greeting:

```
postscreen: 220-server.example.com ESMTP Postfix<CR><LF>
```

```
spambot: HELO i-am-a-bot<CR><LF>
```

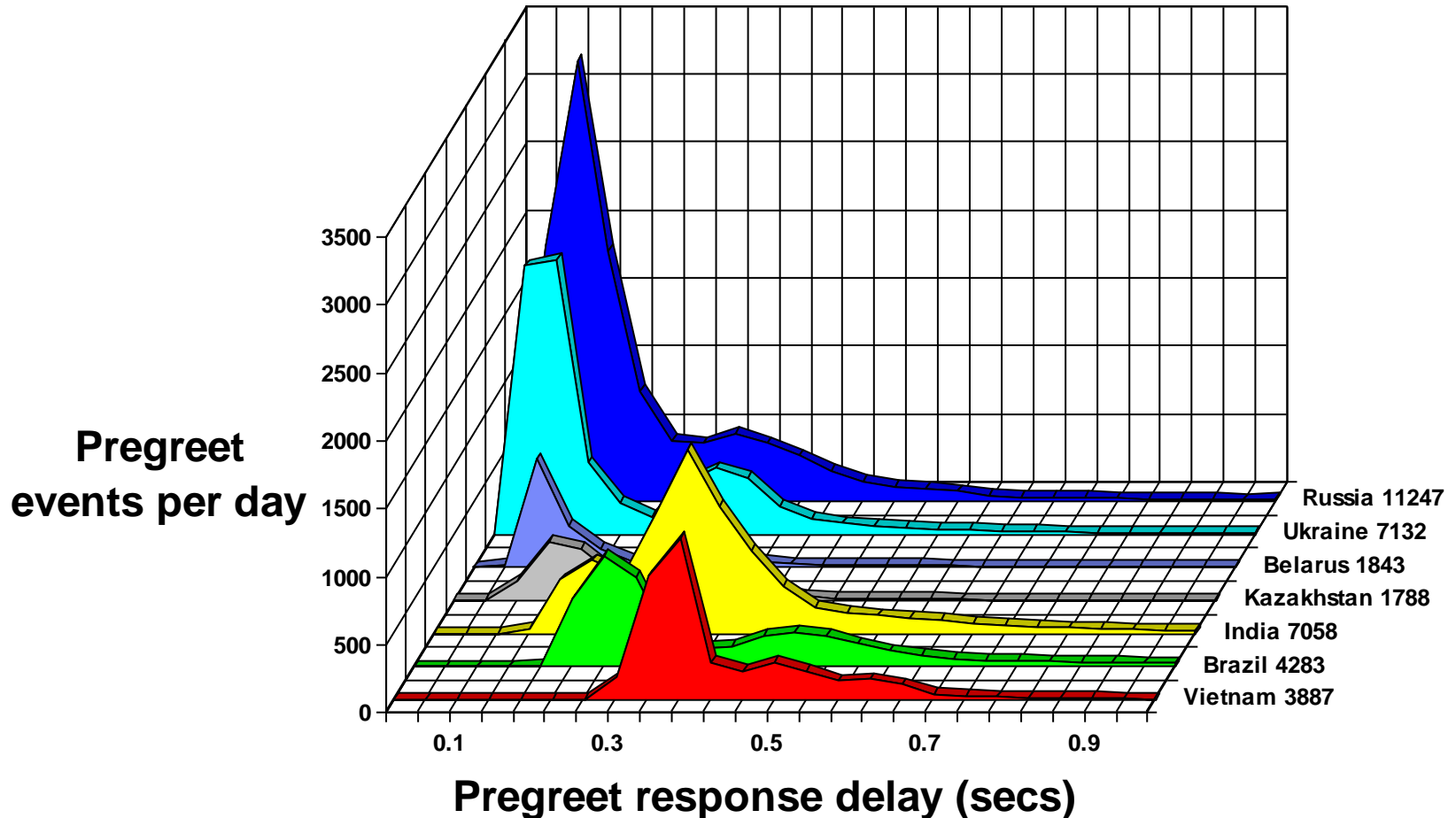
Over 60% of bots pregreet at mail.charite.de

8% Not on DNS blacklists. Berlin, Aug 26 – Sep 29, 2010



Over 70% of bots pregreet at mail.python.org

1% Not on DNS blacklists. Amsterdam, Sep 16 – 29, 2010

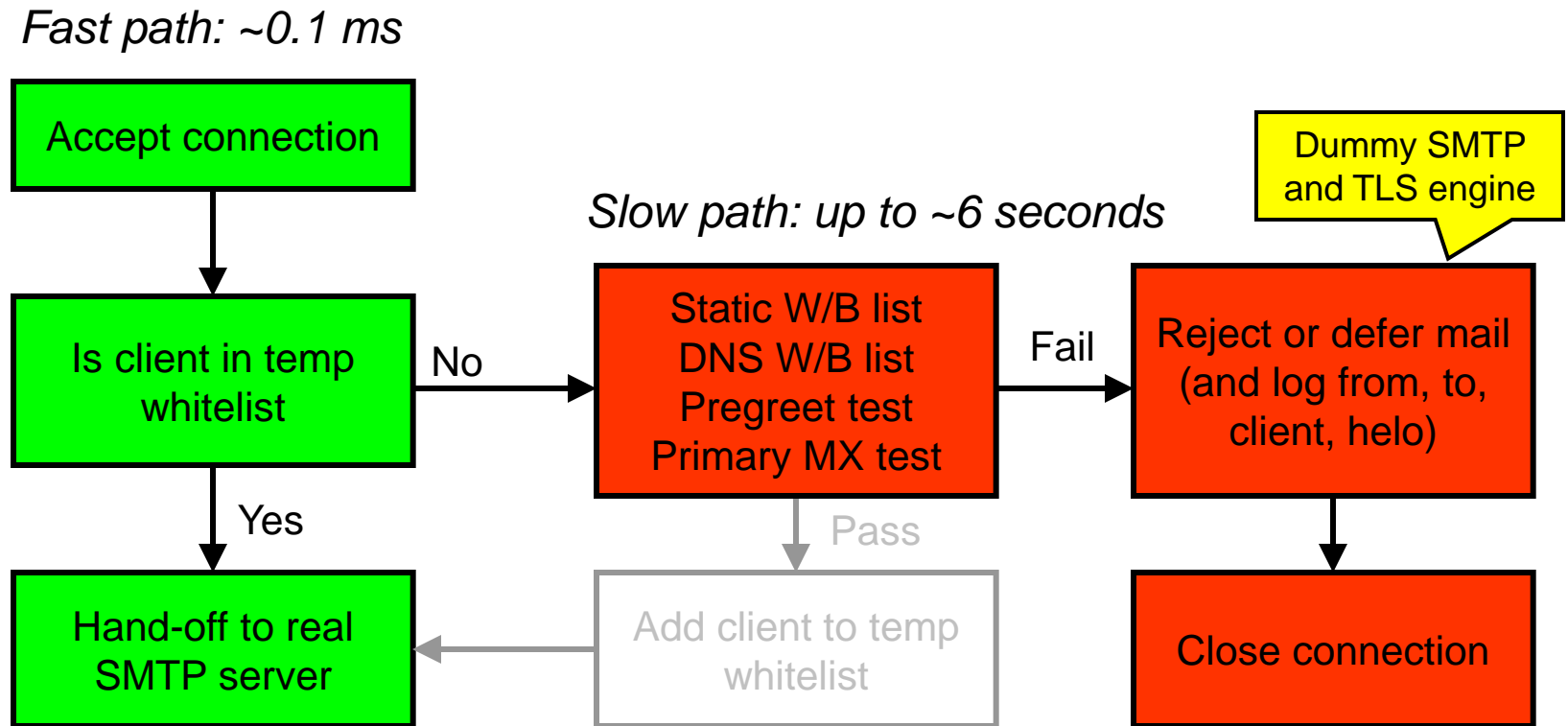


Primary/backup MX policy test

- **Good SMTP clients:**
 - First, try to deliver to primary MX host(s).
 - Only if no primary MX is available, deliver to backup MX.
- **Bad SMTP clients:**
 - Connect only to backup MX hosts (exploit weaker defense).
- **postscreen solution:**
 - Listen on primary and backup MX address on the same host.
 - Don't temp-whitelist clients on backup MX address (reply 4XX).
 - About 14% of spambots fall into this trap.

Postscreen workflow for backup MX connections

Don't temp-whitelist clients on the backup MX address



Changing threats

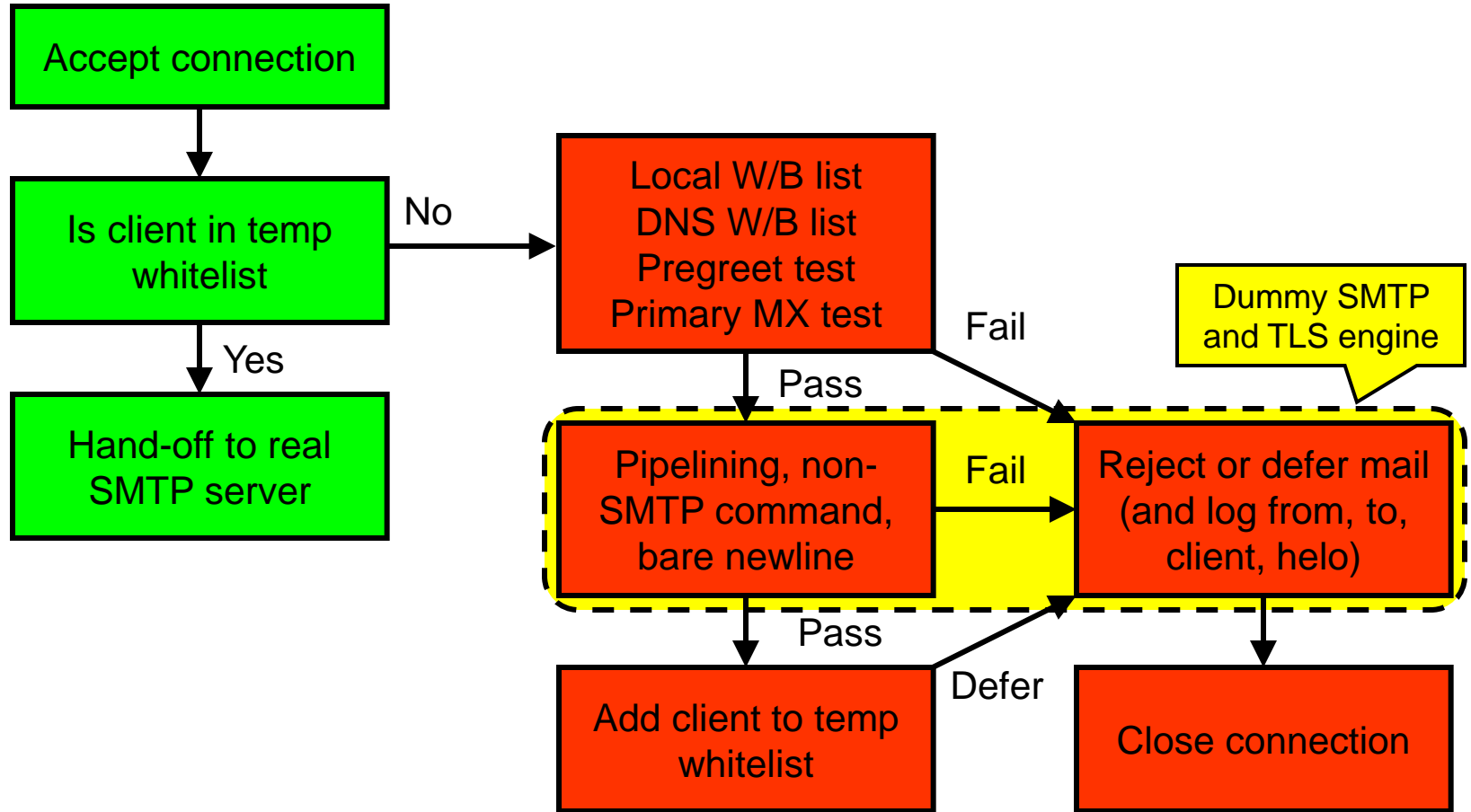
“Deep protocol” tests after the SMTP handshake

Currently, too little gain for the pain

- Announce STARTTLS just like real SMTP server.
- Don't announce PIPELINING support in EHLO.
 - 1% of spambots¹ pipelines anyway.
- Don't allow non-SMTP commands.
 - 3% of spambots¹ ignores 5XX and sends header/body anyway.
- Pain: no “good client” hand-off to SMTP server.
 - Instead, reply with 4XX and wait until the client hangs up.
 - Use second MX address on the same host to avoid “retry” delays.

¹Most of these spambots are already listed on DNSBLs.

Workflow for tests before+after SMTP handshake



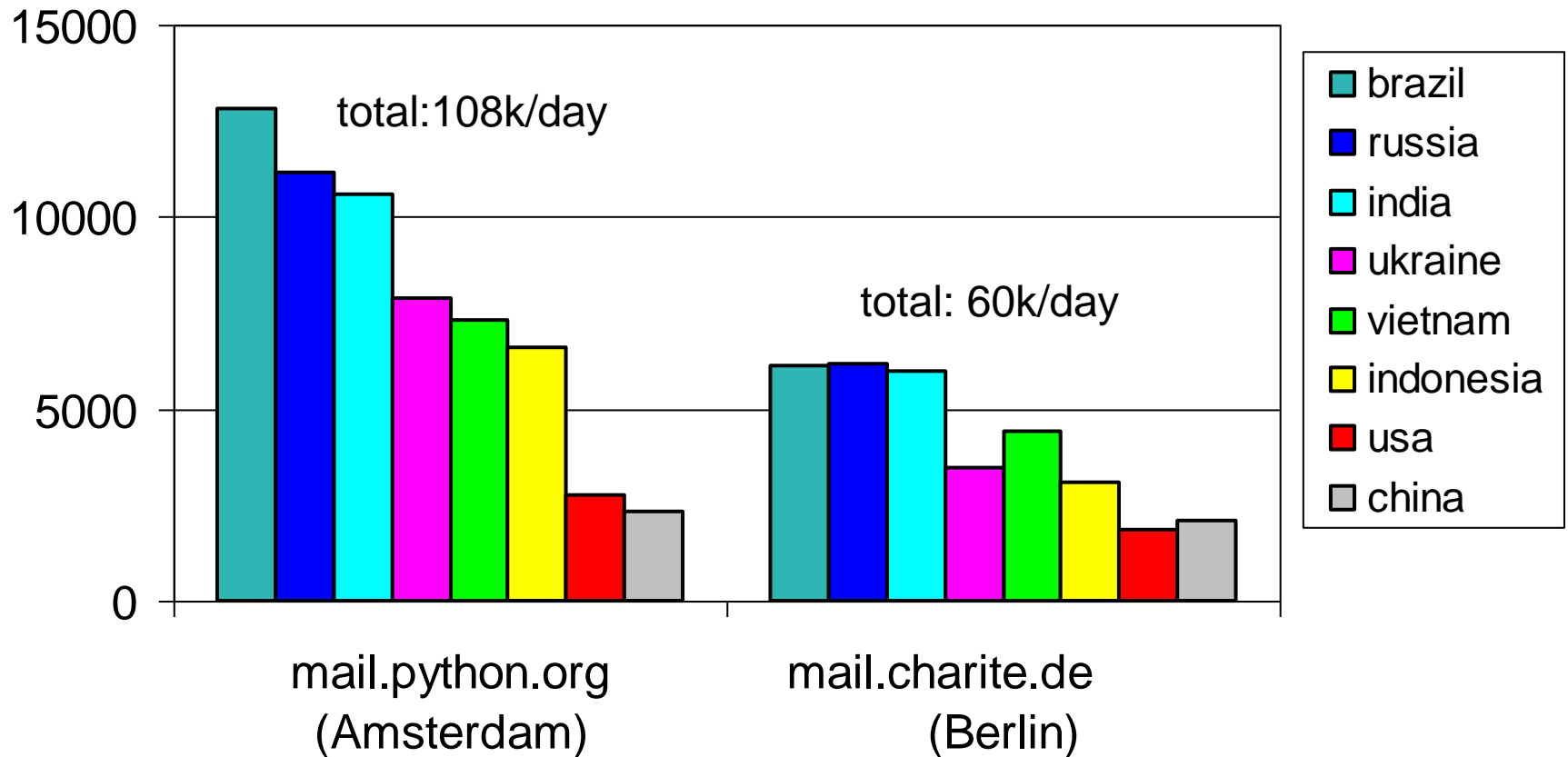
Changing threats

SPAM load varies by receiver, time of day, year

- SPAM load at different sites:
 - Charite.de (Berlin) and Python.org (Amsterdam). Thanks, Ralf.
 - A handful countries sends most of today's spam, but different receivers see different sender volumes.
- SPAM load at different times of day:
 - SPAM is a 24-hour operation, but most spambots are not.
- SPAM load longer-term variations.
 - At these sites, SPAM load dropped more than world average.
 - At my personal domain, SPAM load dropped less, if at all.

Spam connections/day Sep 3 – 23, 2010

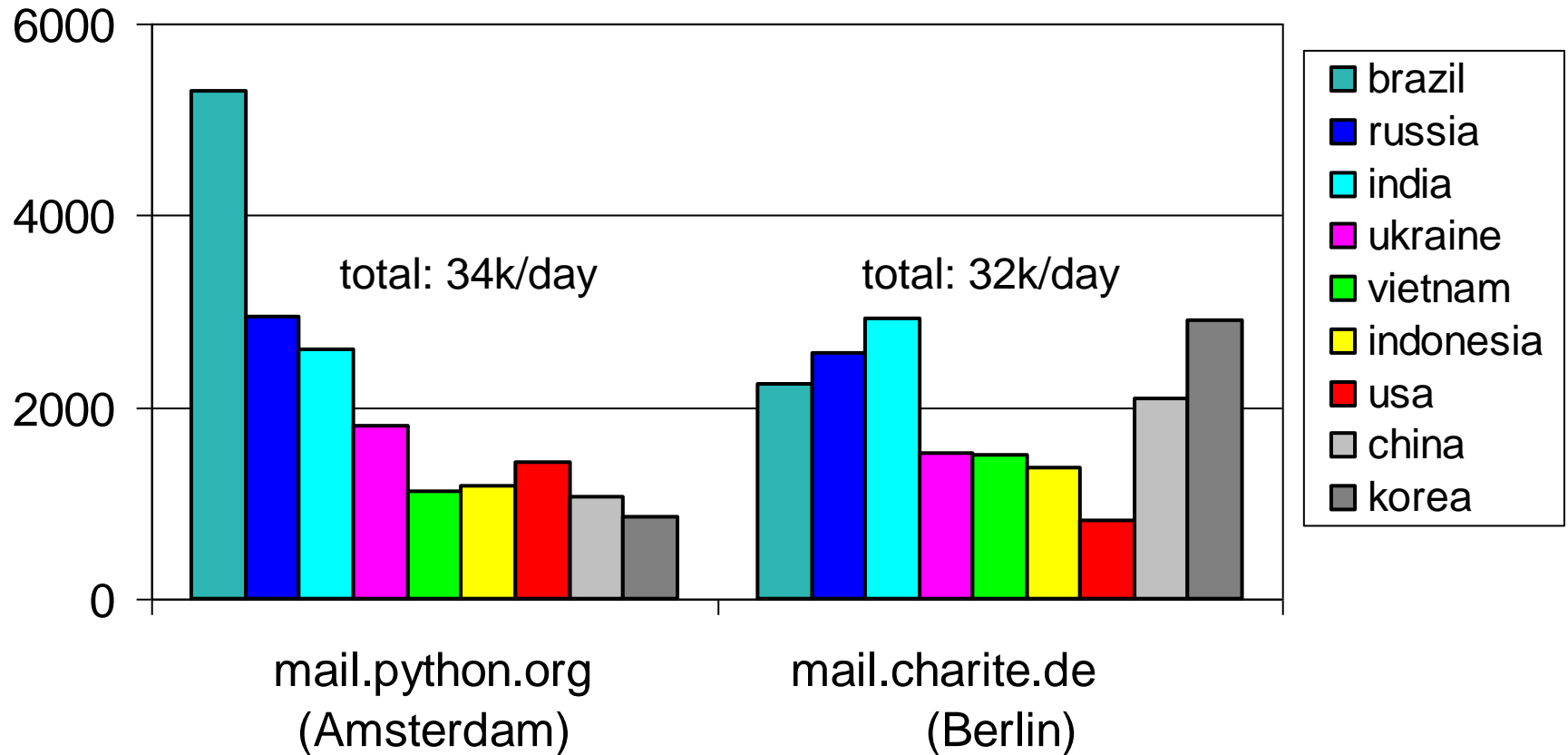
Spam according to zen.spamhaus.org



Changing threats

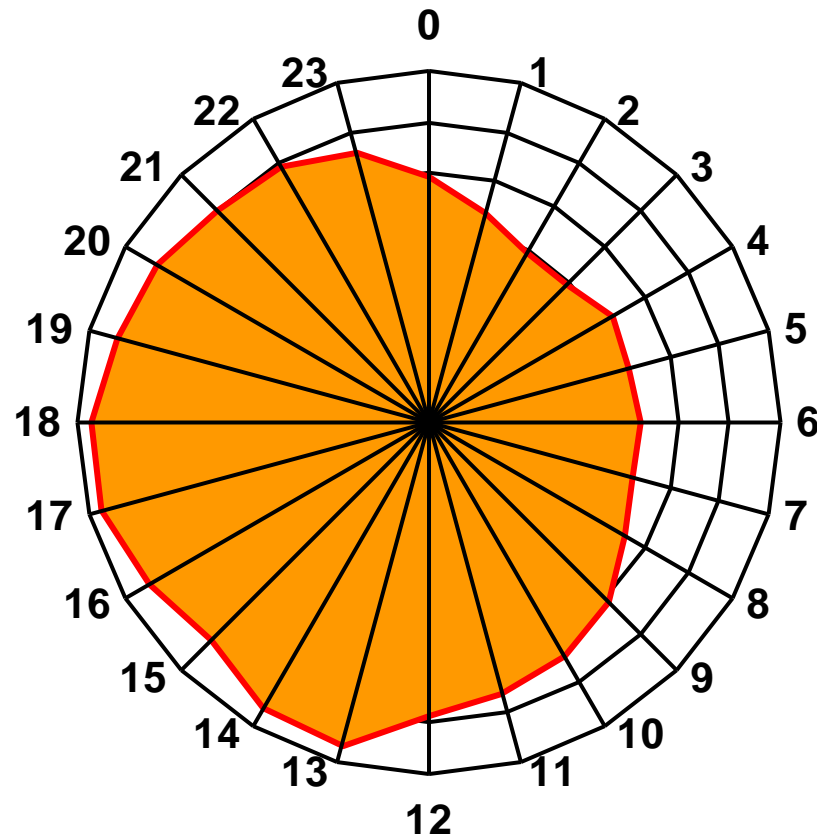
Spam connections/day Apr 13 – May 15, 2011

Spam according to zen.spamhaus.org



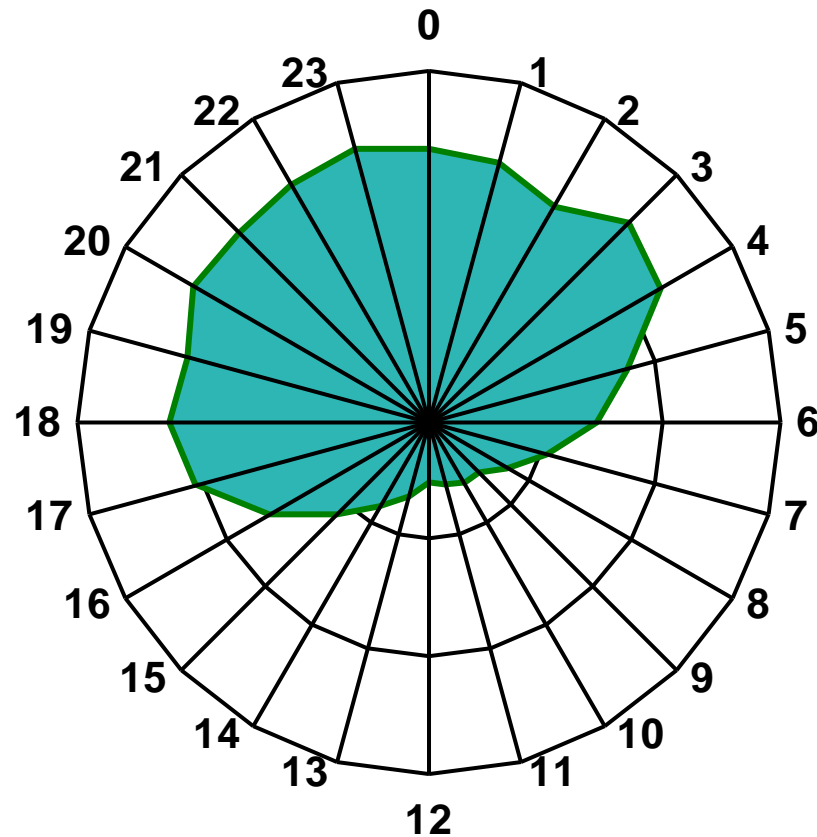
Changing threats

SPAM is a 24-hour operation...



- Spam connections to Charite.de (Berlin) Aug 26 – Sep 29, 2010. Time in UTC.

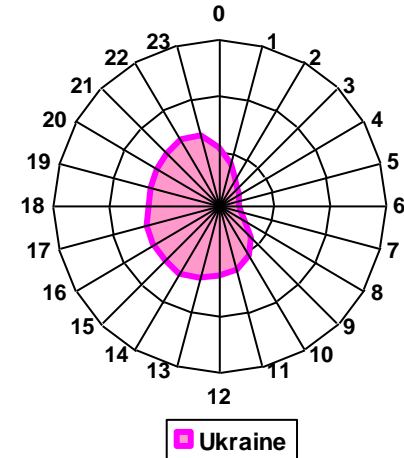
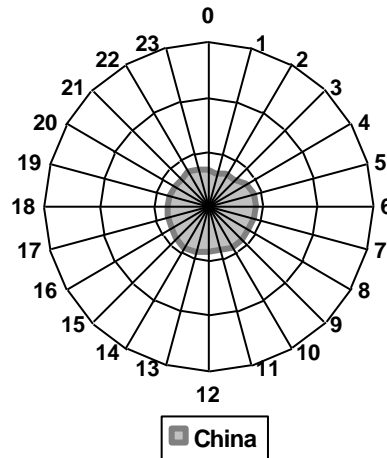
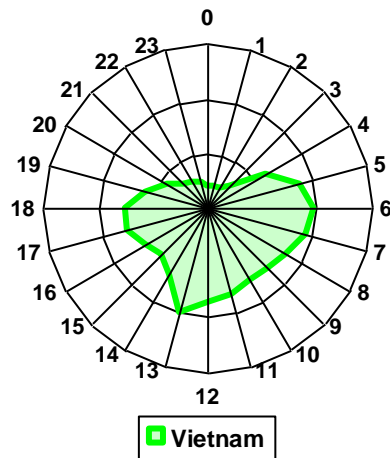
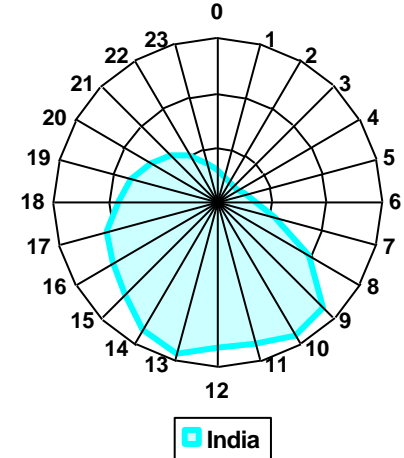
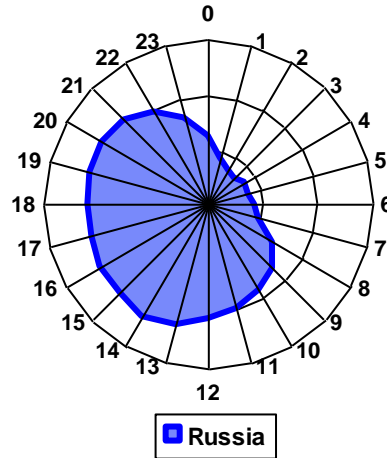
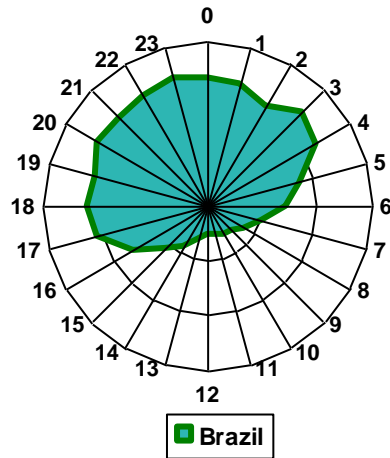
... but most spambots are not



- Spam connections from Brazil to Charite.de (Berlin) Aug 26 – Sep 29, 2010. Time in UTC.

Spam connections/hour at mail.charite.de (UTC)

Spam according to zen.spamhaus.org, Aug 26 – Sep 29, 2010



Changing threats

Postscreen - results and status

- Focus on performance and scalability:
 - Dynamic “fast path” cache minimizes delays for “good” clients.
 - Parallel, weighted, DNS white/blacklist lookup; static white/blacklist.
 - STARTTLS support, stress-dependent limits, no SASL AUTH support.
 - Use postscreen on smtp port (MTA-to-MTA), use AUTH on submission port.
- Pilot results (small sites, up to 200k connections/day):
 - Pregreet (talk early): 60+% of spambots (~10% not in DNSBL).
 - Connect to backup MX only: ~14% of spambots.
 - “Deep protocol” tests catch only a few percent of spambots.
- Included with the Postfix 2.8 stable release (Jan. 2011).

- Concluding remarks

Postfix lessons learned

- Don't re-invent mechanisms that already work. E.g., SMTP, Milter, maildir, lookup tables. Invent sparingly.
- Build the stable protocols into Postfix: SMTP, LMTP, TLS, SASL, IPv6, DSN, MIME, LDAP, SQL.
- Use plug-ins for future proofing: Anti-Spam, Anti-Virus, DKIM, SenderID, SPF, greylist, etc. Plan for change.
- Optimize both the worst case and the common case. Worst cases become normal cases.
- Don't let a C prototype become the final implementation.

Conclusion

- Postfix has matured well. With a system implemented by small programs, many features can be added by *changing* a small program or *adding* a small program.
- Extensibility is a life saver¹. It eliminates the pressure to implement everything within Postfix itself, and it gives the user more choice.
- The battle continues. For the near future, postscreen before-server connection filter keeps mail servers operable under increasing zombie loads.

¹For both author and software.