IBM Research

# Postfix, past present and future

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

# Postfix expectations before the first release

> *[Postfix]: No experience yet, but I'd guess something like a wisened old man sitting on the porch outside the postoffice. Looks at everyone who passes by with deep suspicion, but turns out to be friendly and helpful once he realises you're not there to rob the place.*
>
> Article in alt.sysadmin.recovery, 1997

See http://home.xnet.com/~raven/Sysadmin/ASR.Quotes.html for contemporary comments on other mail systems.

# Overview

Buggy software works (why security is hard).

Strategies to make systems more secure.

Extensibility as a proverbial life saver.

Lies, d*mned lies, and market share.

Recent developments.

Work in progress.

Crystal ball.

# We create bugs faster than we can eliminate them

Source lines of code for contemporary software[1]:

Windows/XP: 40 million; Vista 50 million

Debian 2.2: 56 million; 3.1: 230 million

Conservative estimates:

- 1 bug per 1000 lines (Wietse’s pre-Postfix average)[2]
- 50 million new lines/year.
- 50 thousand new bugs/year. Ka-chink!

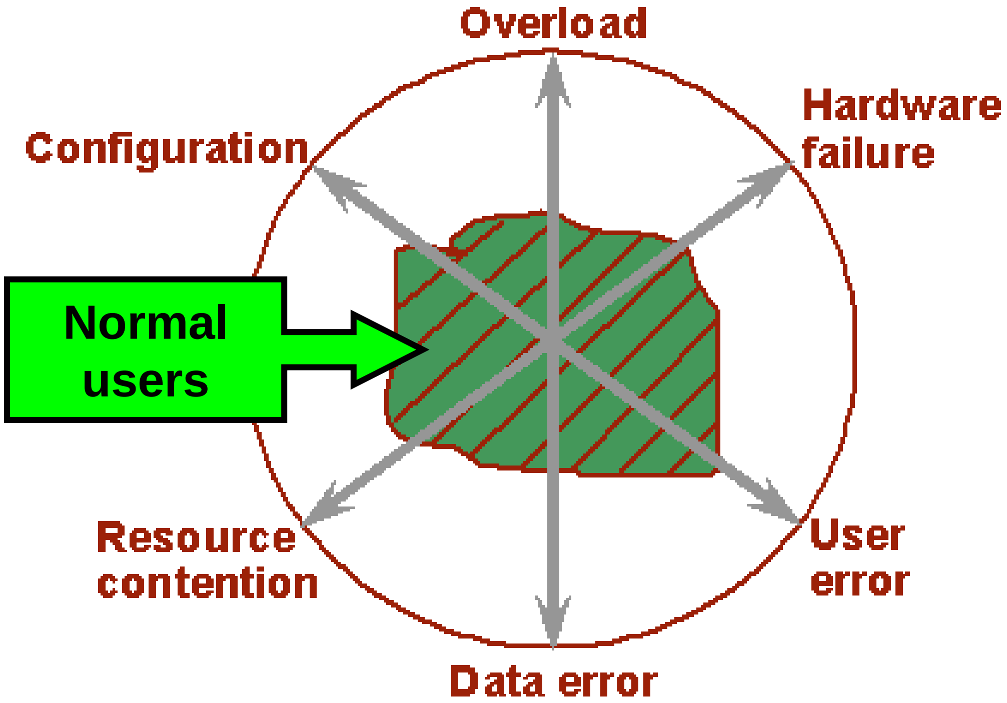
[1] http://www.dwheeler.com/sloc/

[2] Industry average: 10-20 bugs.

Security is hard

# Buggy software works

## And why attackers can break code easily



Secure region = tested (shaded)

Insecure region = untested (unshaded)

– “Normal” users experience “no problems”.

– Attackers explore the untested code paths.

Security is hard

# Bugfixes don’t make software more secure

## They just fix a small fraction of all the bugs

“*As long as there is support for ad hoc fixes and security packages for these inadequate designs and as long as the illusory results of penetration teams are accepted as demonstrations of computer system security, proper security will not be a reality*.”

– Roger Schell et al., “Preliminary notes on the Design of Secure Military Computer Systems”,1973.

Security is hard

# Strategies to improve software security

# Strategy 1: Eliminate programmers

## Less code, fewer bugs

Make programming a million times harder.

- Obviously, that is not happening. Low-barrier languages like PHP aim to make programming easier, not harder.
  - Non-expert programmers outnumber the experts.
  - First versions of code are being tested live on the web.
  - Remember, buggy software works, even when it is riddled with gaping security holes.
- We need a revolution that empowers users, like spreadsheets revolutionized computing in 1969.

Eliminate programmers

# Strategy 2: Plan for failure

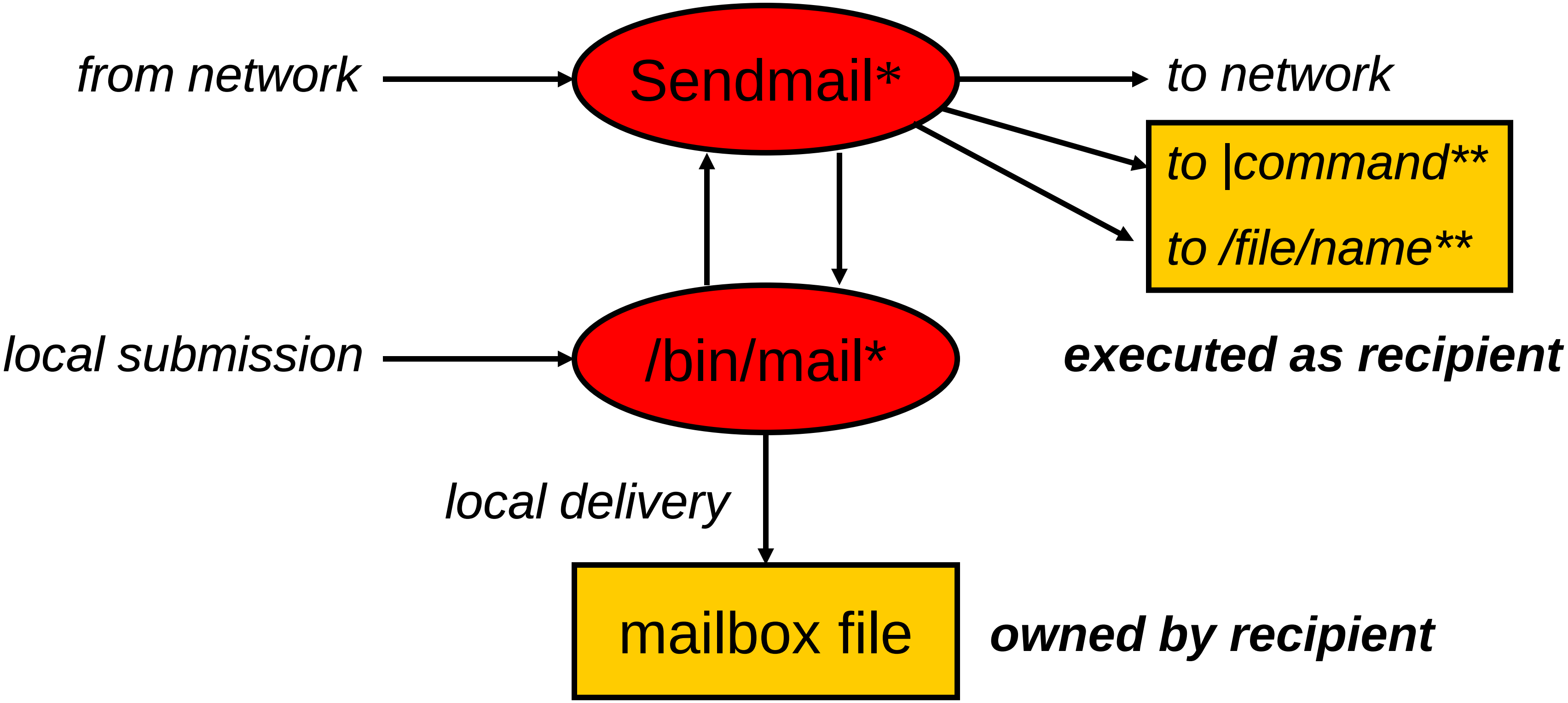
Limit the impact of program error.

- Example: Postfix architecture.
- Challenge: performance and security and flexibility.
- Fortunately, the security architecture has other benefits.

Plan for failure

# 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

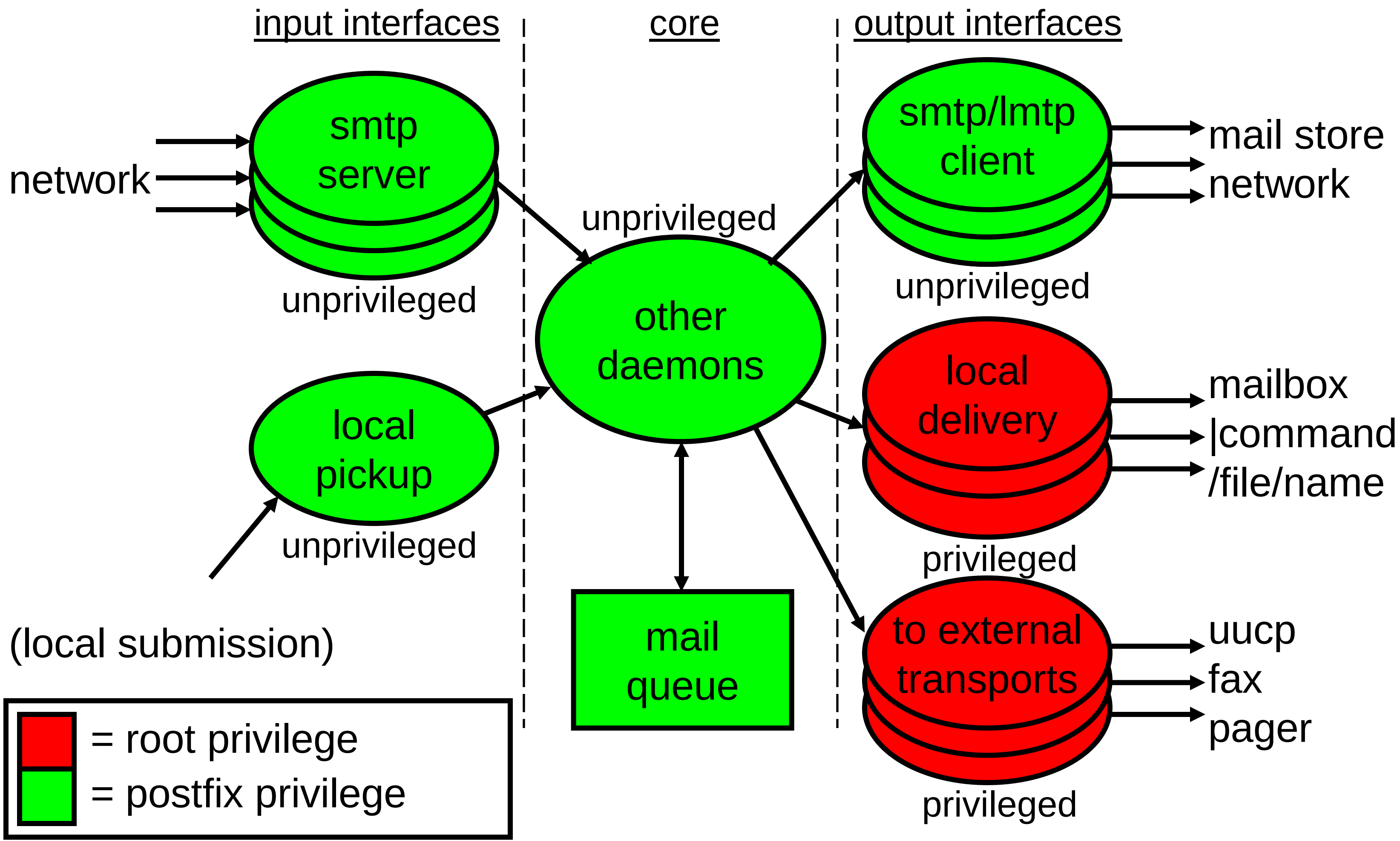
# CERT/CC advisories for Sendmail

| Advisory | Version | Impact |
|---|---|---|
| CA-1988-01 | 5.58 | Unprivileged access |
| CA-1993-16 | 8.6.3 | Unprivileged access |
| CA-1994-12 | 8.6.7 | Full system privilege |
| CA-1995-05 | 8.6.9 | Full system privilege |
| CA-1995-13 | 8.7.0 | Full system privilege |
| CA-1996-04 | 8.7.3 | Full system privilege |
| CA-1996-20 | 8.7.5 | Full system privilege |
| CA-1996-24 | 8.8.2 | Full system privilege |
| CA-1996-25 | 8.8.3 | Group privileges |
| CA-1997-05 | 8.8.4 | Full system privilege |
| CA-2003-07 | 8.12.7 | Full system privilege |
| CA-2003-12 | 8.12.8 | Full system privilege |
| CA-2003-25 | 8.12.9 | Full system privilege |

Plan for failure

# Postfix distributed security architecture

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



input interfaces

core

output interfaces

network

smtp server

unprivileged

local pickup

unprivileged

(local submission)

unprivileged

other daemons

mail queue

smtp/lmtp client

unprivileged

mail store

network

local delivery

privileged

mailbox

|command

/file/name

to external transports

privileged

uucp

fax

pager

🟥 = root privilege

🟩 = postfix privilege

Plan for failure

# Major influences on Postfix architecture

TIS Firewall smap/smapd: least privilege, chroot jail, “air gap” between receiving and delivering processes.

qmail: parallel deliveries; the maildir format (the MH mail handling system introduced a “one file per message” mailbox store 20 years before qmail).

Apache: reuse processes multiple times.

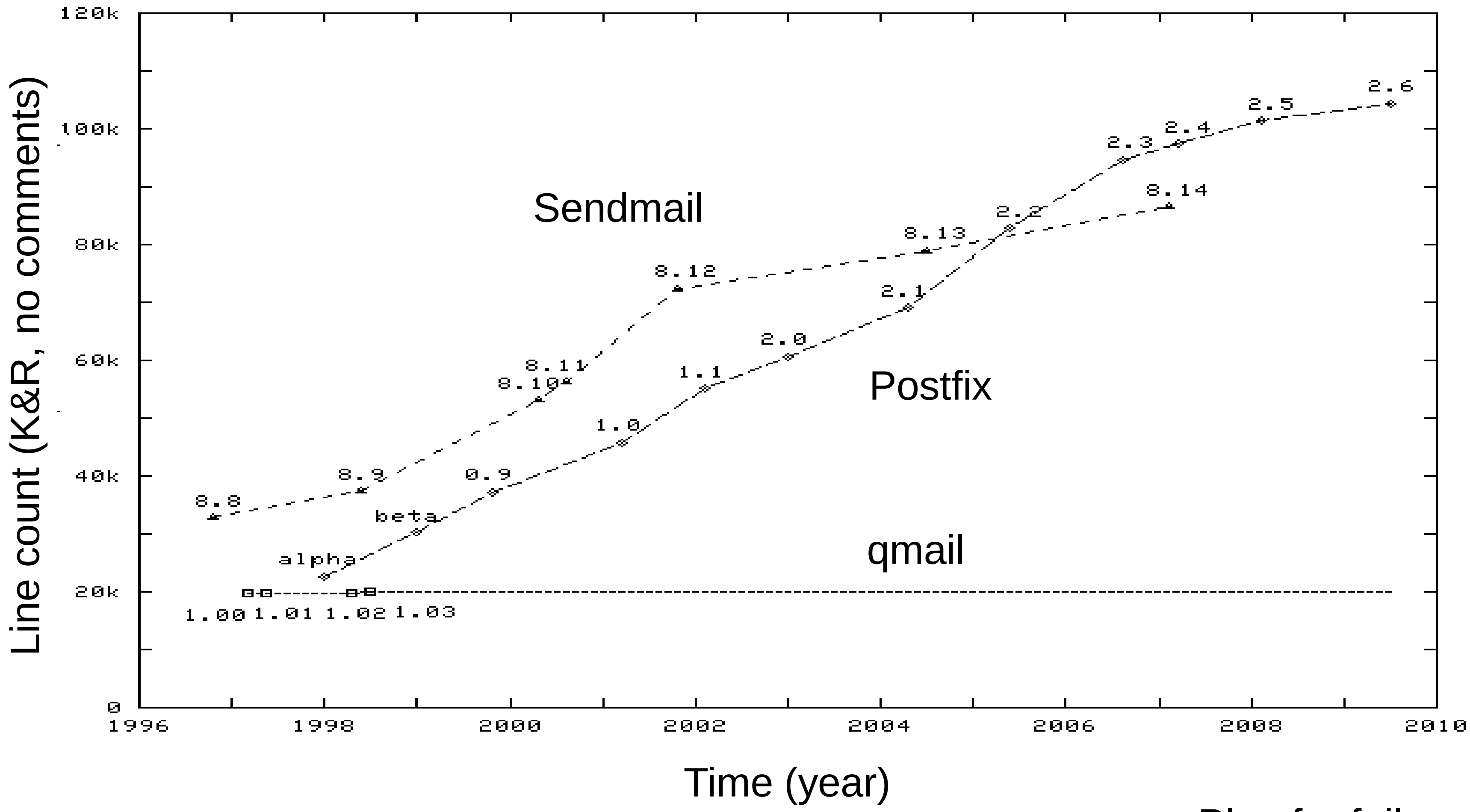
Sendmail: user interface; lookup table interface.

Classical routers: multiple interfaces/encapsulations, central core, but alas no queue-skipping fast path :-(

Plan for failure

# MTA Source lines versus time

## Putting more functionality into fewer lines of code



Plan for failure

# Security architecture has other benefits

Small programs are easier to understand and easier to maintain than large programs. K.I.S.S.

- Minor functions are implemented by changing one small program, without changing other programs.
- Major functions are implemented by adding small programs that are loosely coupled to the rest of Postfix.
- All this is good for system stability and integrity.
- Present breakdown: 23 daemons, 13 commands.
  - There will be more daemons by the end of this presentation.

Plan for failure

# 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 main vehicle for malware distribution.

– *Short term*: block “known to be bad” strings in message header text (body strings came later).

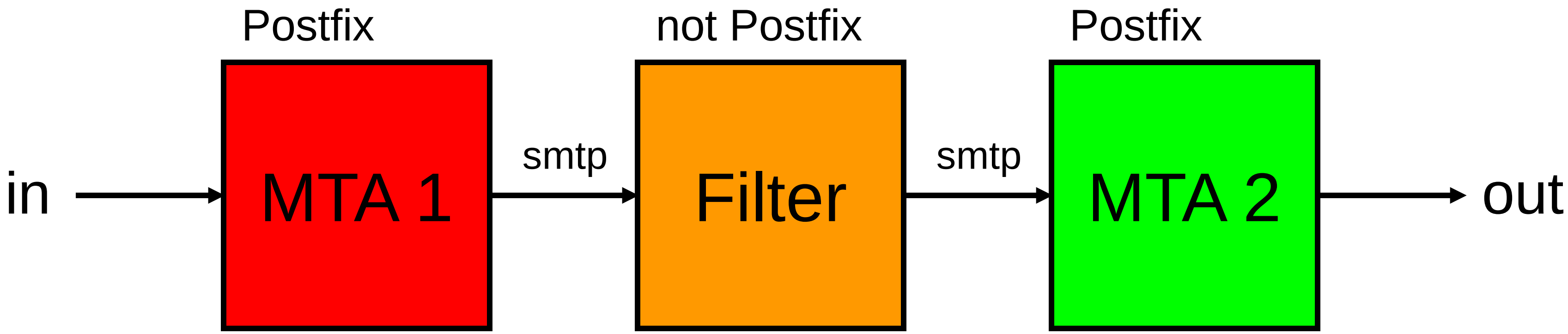
– *Long-term*: delegate deep inspection to third-party software.

Emergence of specialized protocols: CVP, Milter, etc.

– We already use SMTP for email distribution. Why can’t we also use SMTP to plug in anti-{spam,virus}?

Invent sparingly

# Postfix content inspection via SMTP (post queue)



Red = dirty, green = clean.

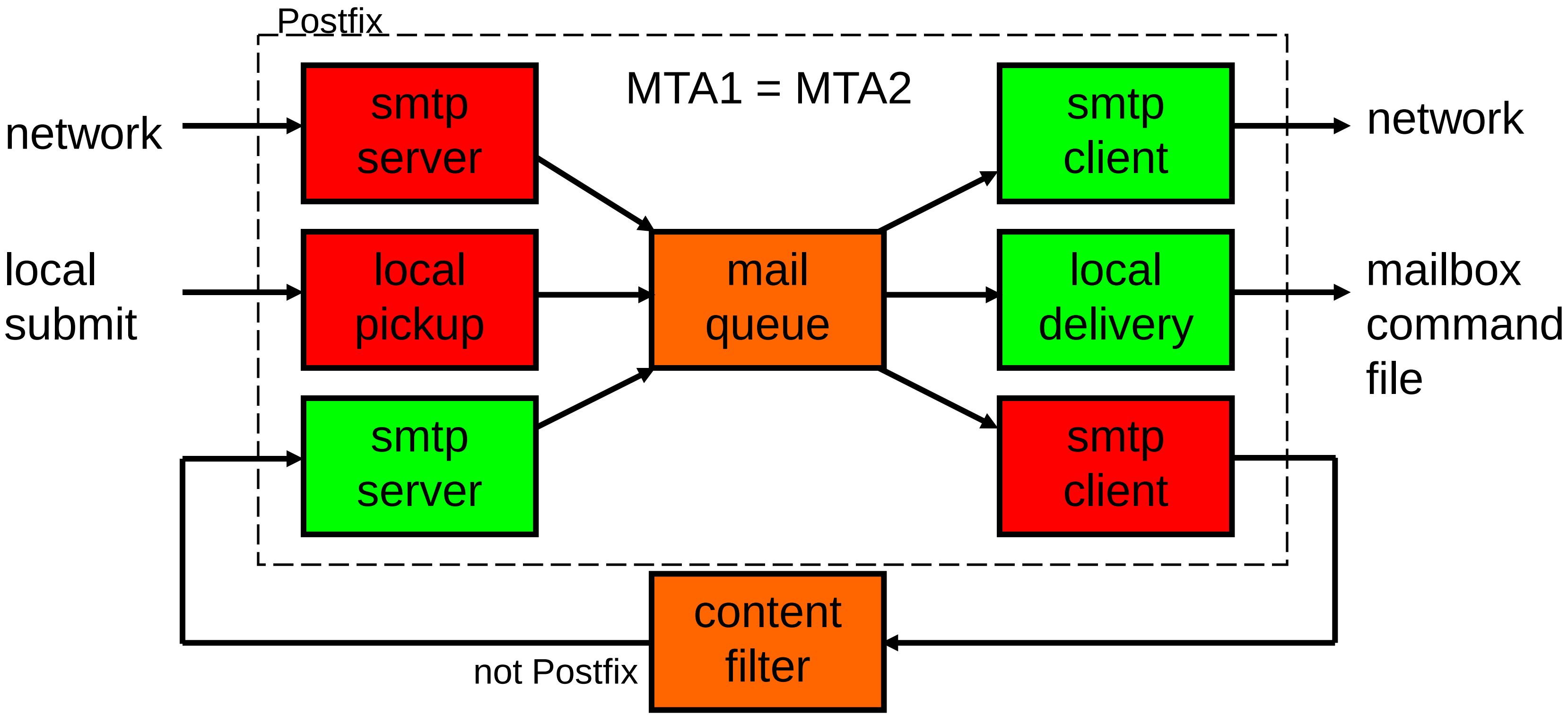
But it can’t be that simple, right?

Using two MTAs must be wasteful!

Invent sparingly

# Postfix content inspection via SMTP (post queue)

## Two MTAs combined into one



Combining two MTAs into one increases complexity - one set of configuration files for two MTAs.

Invent sparingly

# Post-queue anti-spam/virus support

The advantages of post-queue SMTP-based anti-spam/virus filters outweigh many disadvantages:

– *Compatibility*: many products are SMTP enabled. SMTP is well understood, as are the workarounds for common SMTP implementation errors.

– *Performance*: no need to run one filter process per remote SMTP client. This allows for  better resource utilization than possible with before-queue filters.

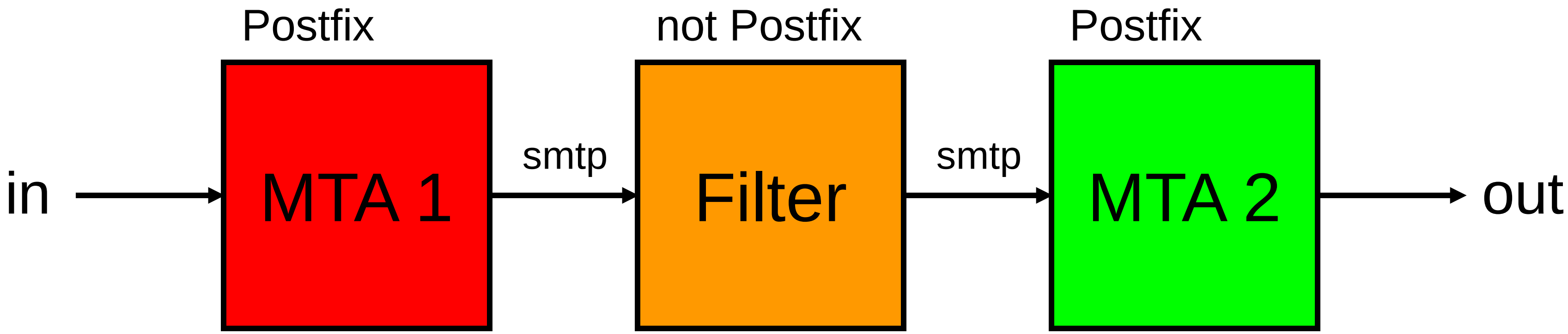
Workarounds for loss of original SMTP client context:

– Xforward, etc.

Invent sparingly

# Post-queue content inspection as of Postfix 2.6

## It is never too late do do something right



Red = dirty, green = clean.

Postfix 2.6 multi-instance support reduces complexity.

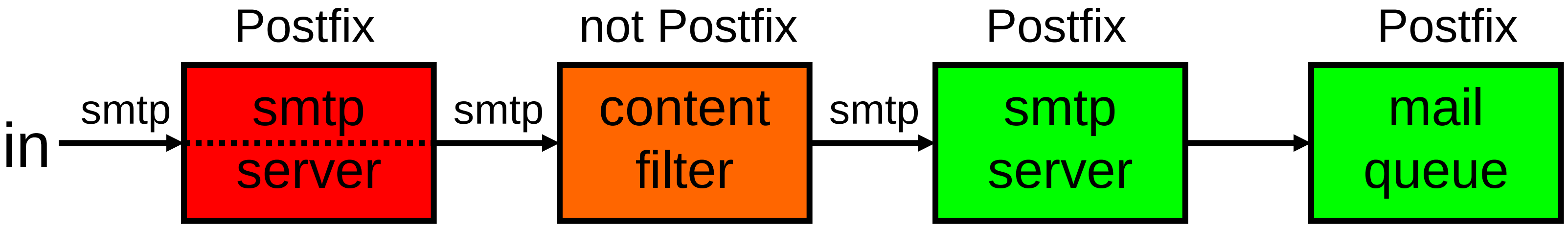
– Different sets of config files for different Postfix instances.

See MULTI_INSTANCE_README for suggestions.

Invent sparingly

# Pre-queue content inspection via SMTP

## Responding to popular demand, despite limited performance



SMTP “pass-through” feature built into SMTP server.

One filter per SMTP client: no decoupling of remote network latencies from local filter concurrencies.

Less scalable, due to poorer resource management.

But the user wanted pre-queue spam/virus filtering.

Invent sparingly

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

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

Wietse on Postfix mailing list, 2003

# 2005 - Proliferation of authentication technologies

SenderID, Domainkeys, DKIM, SPF, BATV, SRS, and the end is not in sight.

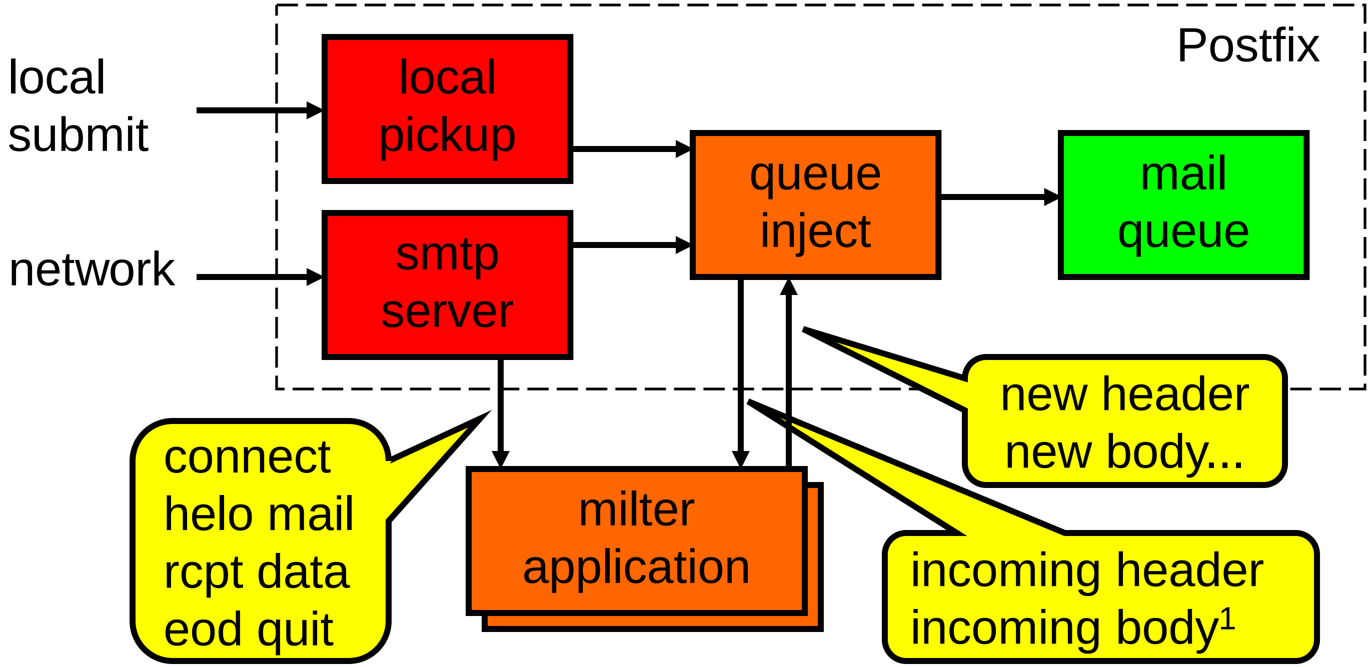
*Problem*: using SMTP-based filters just to “sign” or “verify” can be clumsy (e.g., missing original SMTP client context). Tighter coupling to MTA is desirable.

Building into the MTA is not practical; besides, many (Linux) distributions are two years behind on Postfix.

*Solution*: adopt Sendmail Milter protocol and open up access to a large collection of available applications.

Plan for change

# Retrofitting Milter support into a distributed MTA



Red = dirty, green = clean.

The effort was heroic, but the reward was sweet.

[1] With local submission, this also sends ersatz connect/helo/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**.

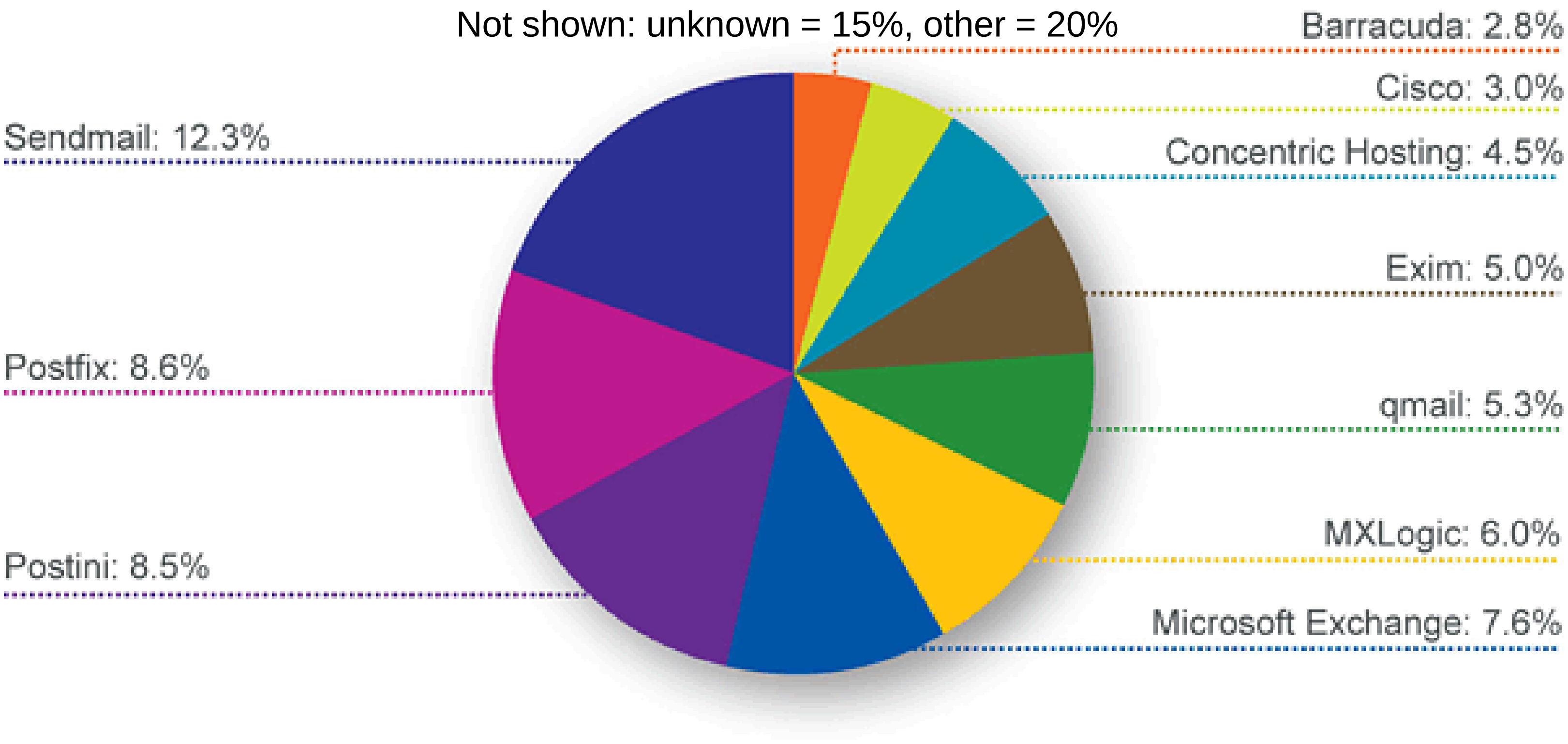
http://www.sendmail.com/pdfs/pressreleases/Sendmail%20Innovation%20Awards_10%2025%2006_FINAL.pdf

Plan for change

# Market share (lies, d*mned lies, and ...)

# Fingerprinting 400,000 company domains remotely

Not shown: unknown = 15%, other = 20%



After: Ken Simpson and Stas Bekman, O’Reilly SysAdmin, January 2007.

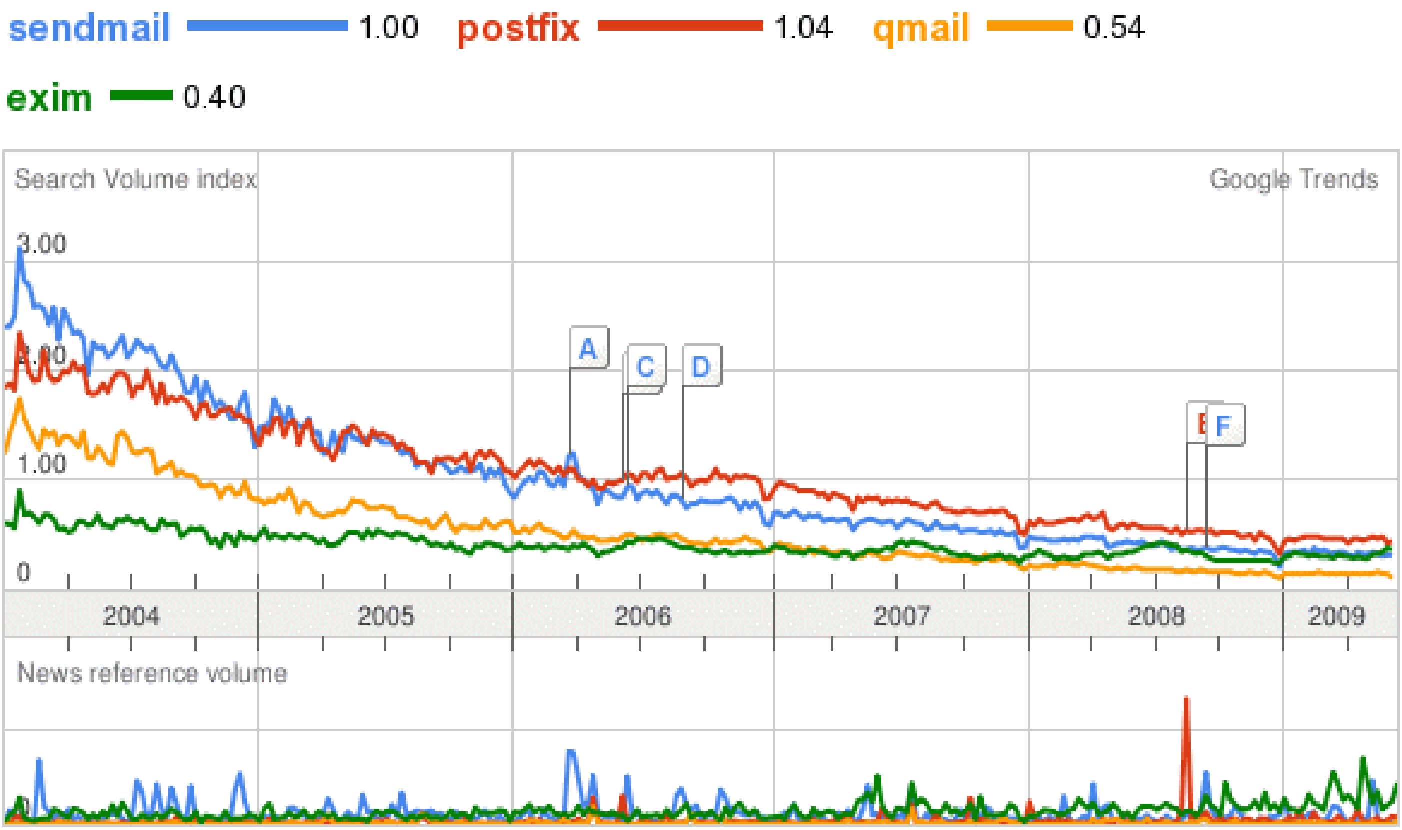
http://www.oreillynet.com/pub/a/sysadmin/2007/01/05/fingerprinting-mail-servers.html

Market share

# Interesting result, but what does it mean?

Query = sendmail, postfix, qmail, exim

Relative search volume



Market share

# Introducing Google trends

Website: trends.google.com (google.com/trends).

Search for RELATIVE popularity of search terms.

- Recursive Google?

Result is a time distribution.

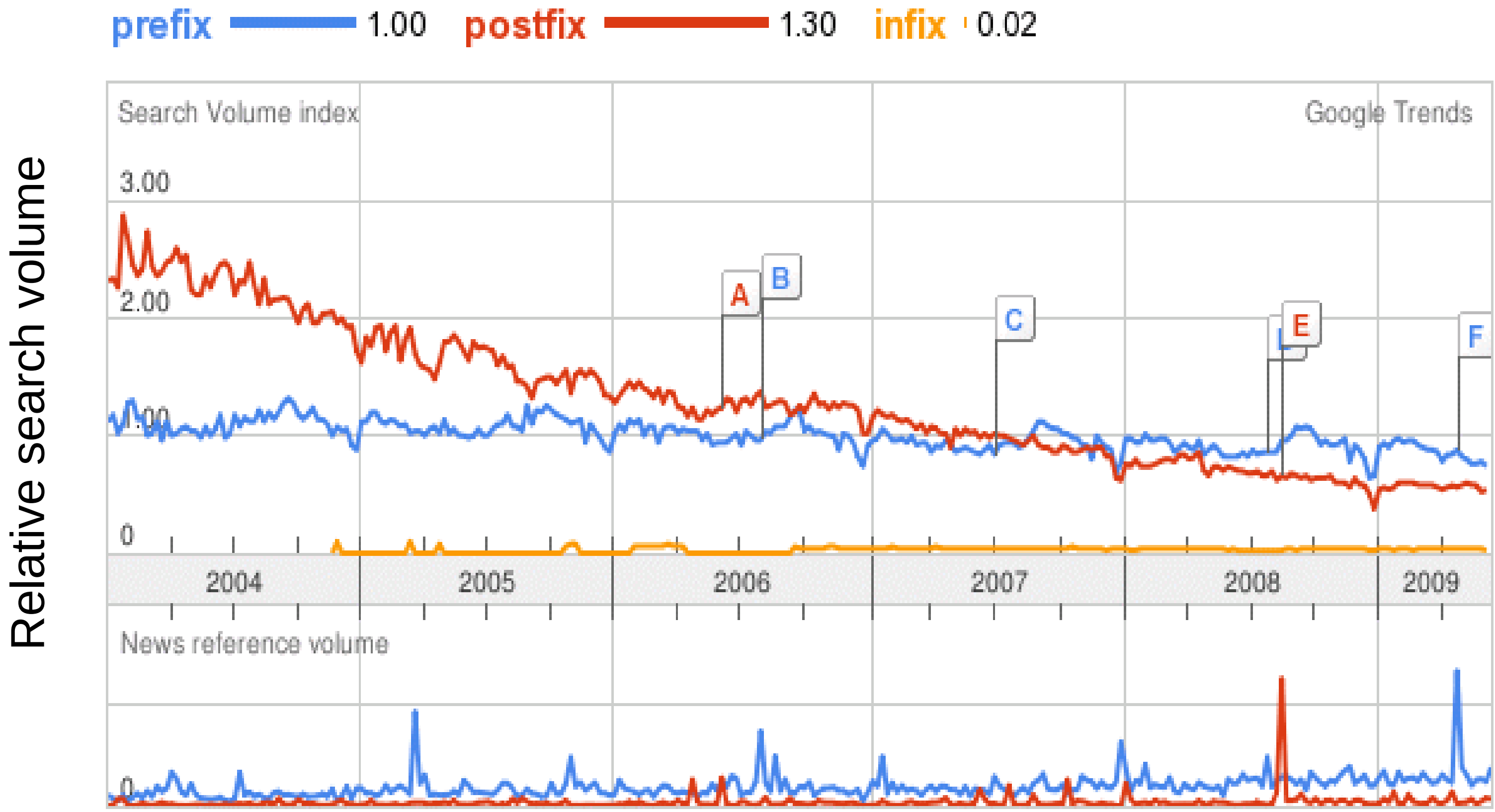
- Different colors for different search terms.

Peaks are correlated with on-line news articles.

Market share

# Pollution by common words and name collisions

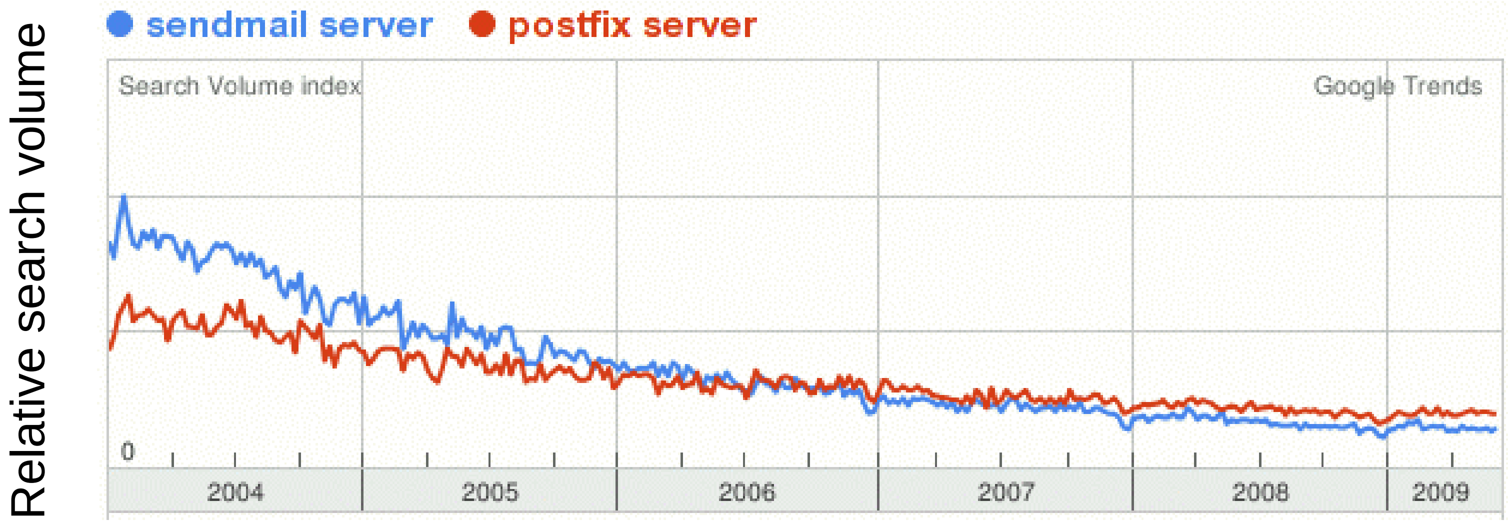
Query = prefix, postfix, infix



Market share

# Tweaking the query to avoid pollution

## Query = sendmail server, postfix server



Market share

# Google trends caveats

As always, the answer you get is only as good as the question you ask. Beware of name collisions, common words, and other forms of pollution.

Sobering lessons:

- Only a minority of users is interested in mail servers.
- Their proportion is steadily declining.

# Recent developments

# Recent developments part 1 of 2

DSN and enhanced status codes (Postfix 2.3)

– Standardized confirmation of (non-)delivery.

– Standardized x.y.z status codes. MUAs can translate these into the user’s own language.

Bounce message templates (Postfix 2.3)

– Typical use: native language + English version.

Sendmail Milter support (Postfix 2.3, 2.4, 2.5, 2.6)

– Authentication (DKIM etc.) and before-queue filtering.

# Recent developments part 2 of 2

Kernel-based event filters (kqueue/epoll/devpoll) (2.4)

– Makes Postfix scalable to thousands of connections.

SOHO (Small Office/Home Office; Postfix 2.3, 2.5)

– Per-sender ISP accounts, output rate control.

Multiple instance support (Postfix 2.6)

– Simplifies content filters.

– Separates local null client from MTA service instances.

Stress-adaptive behavior (Postfix 2.5, 2.6)

– Workaround for temporary overload.

# 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 MTA becomes a major distribution channel for Windows malware (Melissa).

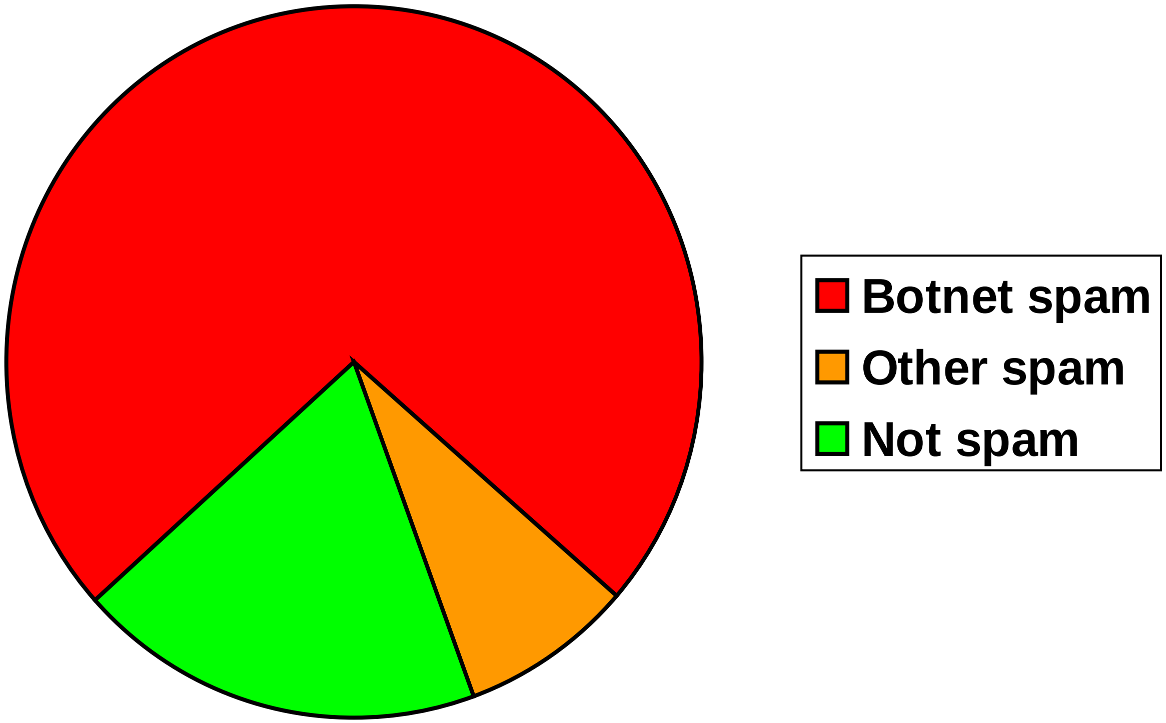
– *Solution*: outsourcing to external content filters.

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 email.

Changing threats

# 81% of email is spam, 90% is from botnets[1]



[1] MessageLabs 2008 annual report

Changing threats

# Zombies suck the life out of the mail server

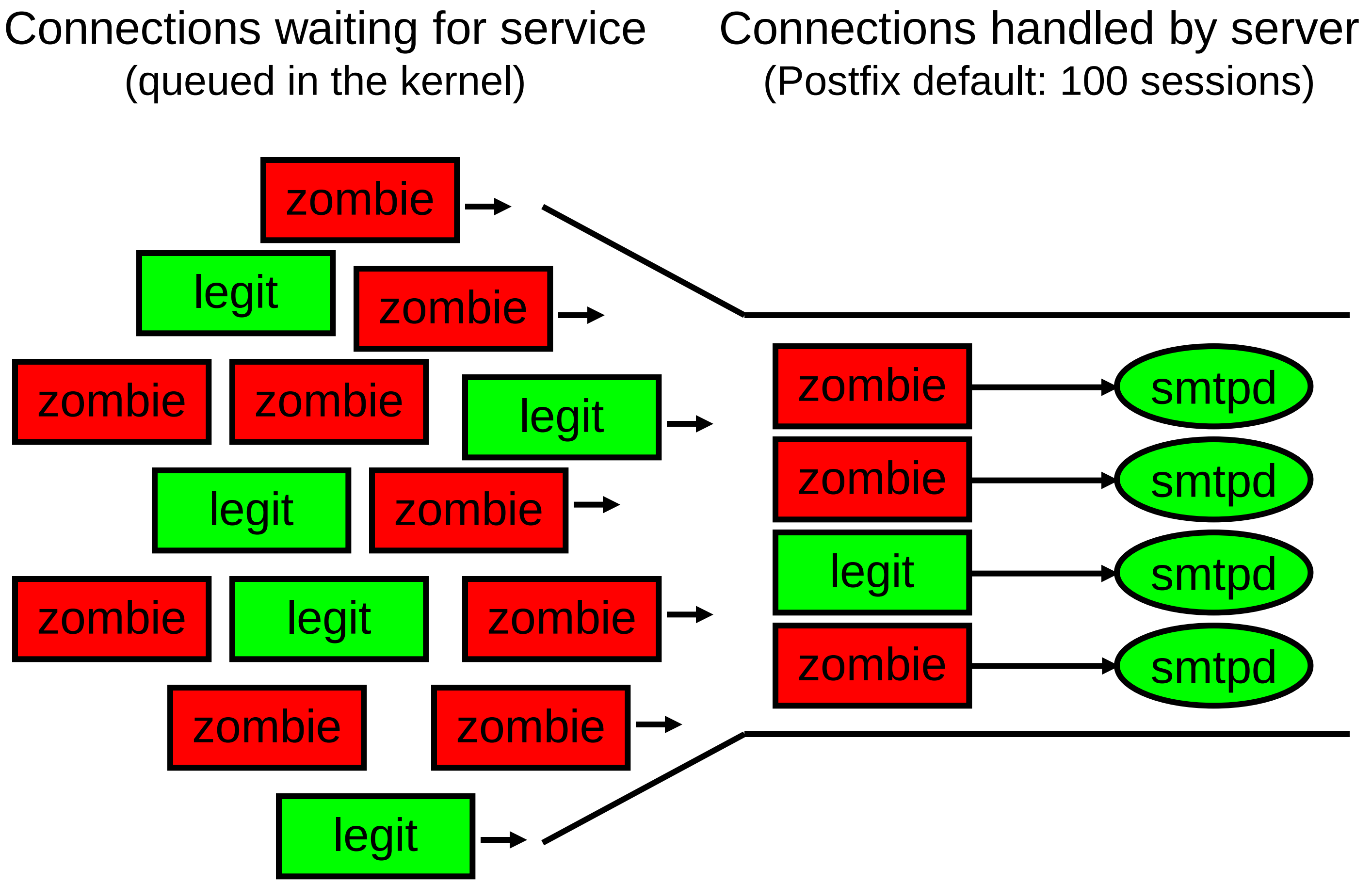
Worst-case example: Storm botnet, August 2007.

```
15:16:55 postfix/smtpd: connect from [x.x.x.x]

15:16:56 postfix/smtpd: reject: RCPT from [x.x.x.x]:
          550 5.7.1 blah blah blah

15:21:56 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 are kept busy by zombies.

# Zombies keep mail server ports busy



Connections waiting for service
(queued in the kernel)

Connections handled by server
(Postfix default: 100 sessions)

Changing threats

# Symptoms of mail server overload

Clients experience delays before the server responds.

- Not to be confused with delays due to broken DNS configurations.

Servers log large numbers of “lost connection” events.

- Clients hang up before the server responds.
- Not to be confused with “lost connection” due to portscanning activity.

Postfix ≥ 2.3 logs “all server ports busy” warnings.

# Overload handling strategies, part 1 of 2

Strategies for *temporary* overload:

- Work faster: spend less time per (zombie) client: reduce time limits, number of failed commands per session, etc.
  - May delay *some* legitimate email messages.
    - Better to receive most legitimate mail than almost no email.
  - OK if the overload condition is temporary.

# Temporary overload - default “on” in Postfix 2.6

## Off by default in Postfix 2.5, patches for Postfix 2.4 and 2.3.

Postfix master(8) daemon sets “stress” configuration parameter on network daemon command line[1]:

```
smtpd -o stress=yes      (overload)

smtpd -o stress=         (normal)
```

Postfix main.cf settings:

```
smtpd_timeout = ${stress?10}${stress:300}s

smtpd_hard_error_limit = ${stress?1}${stress:20}

smtpd_junk_command_limit = ${stress?1}${stress:100}
```

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

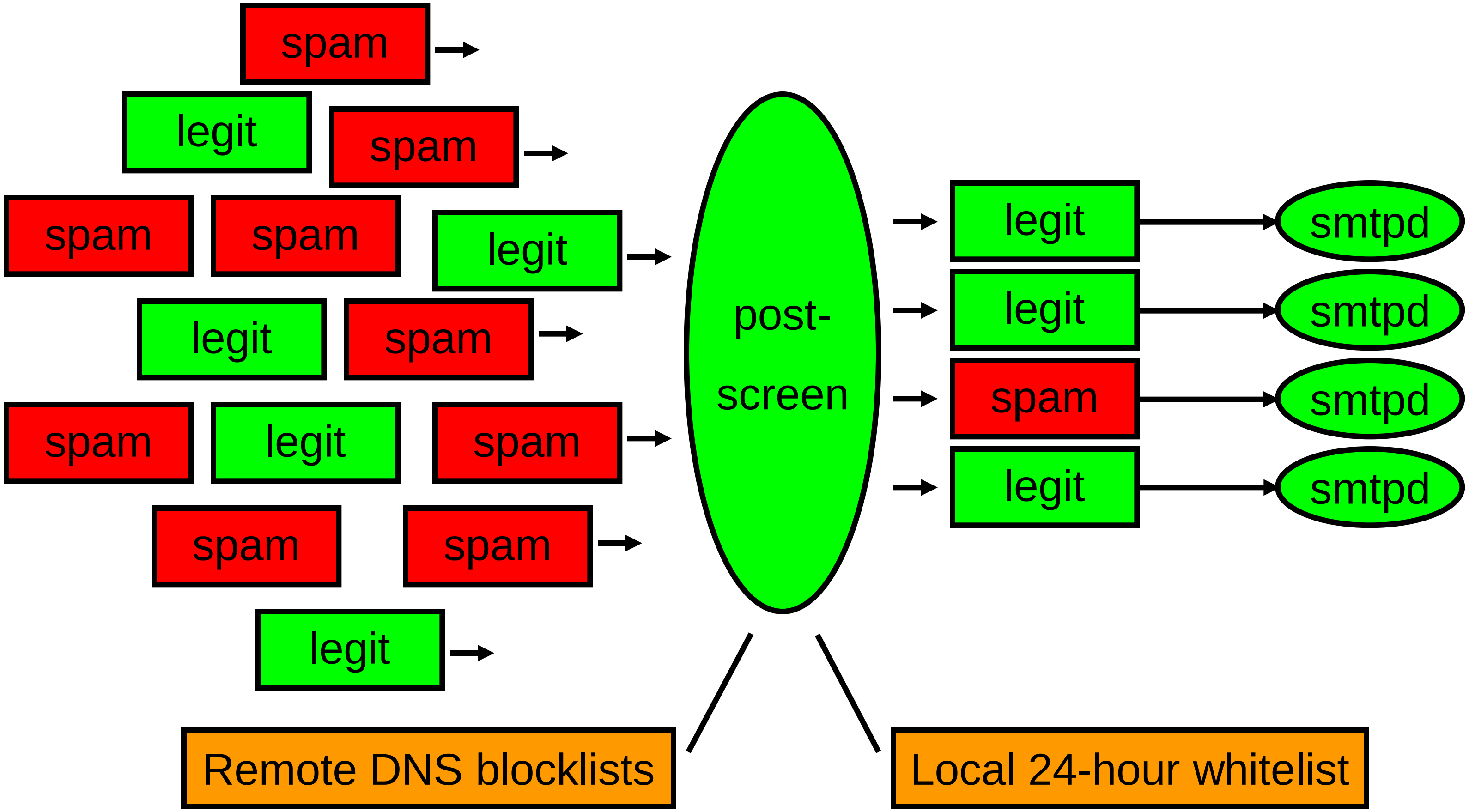
# Overload handling strategies, part 2 of 2

Strategies for *persistent* overload:

- Work harder: configure more SMTP server slots.
  - OK if you can afford the memory, disk, and cpu resources.
- Work smarter: keep mailbots away from SMTP server.
  - More SMTP server slots remain available for handling email.

# Persistent overload - before-smtpd connection filter

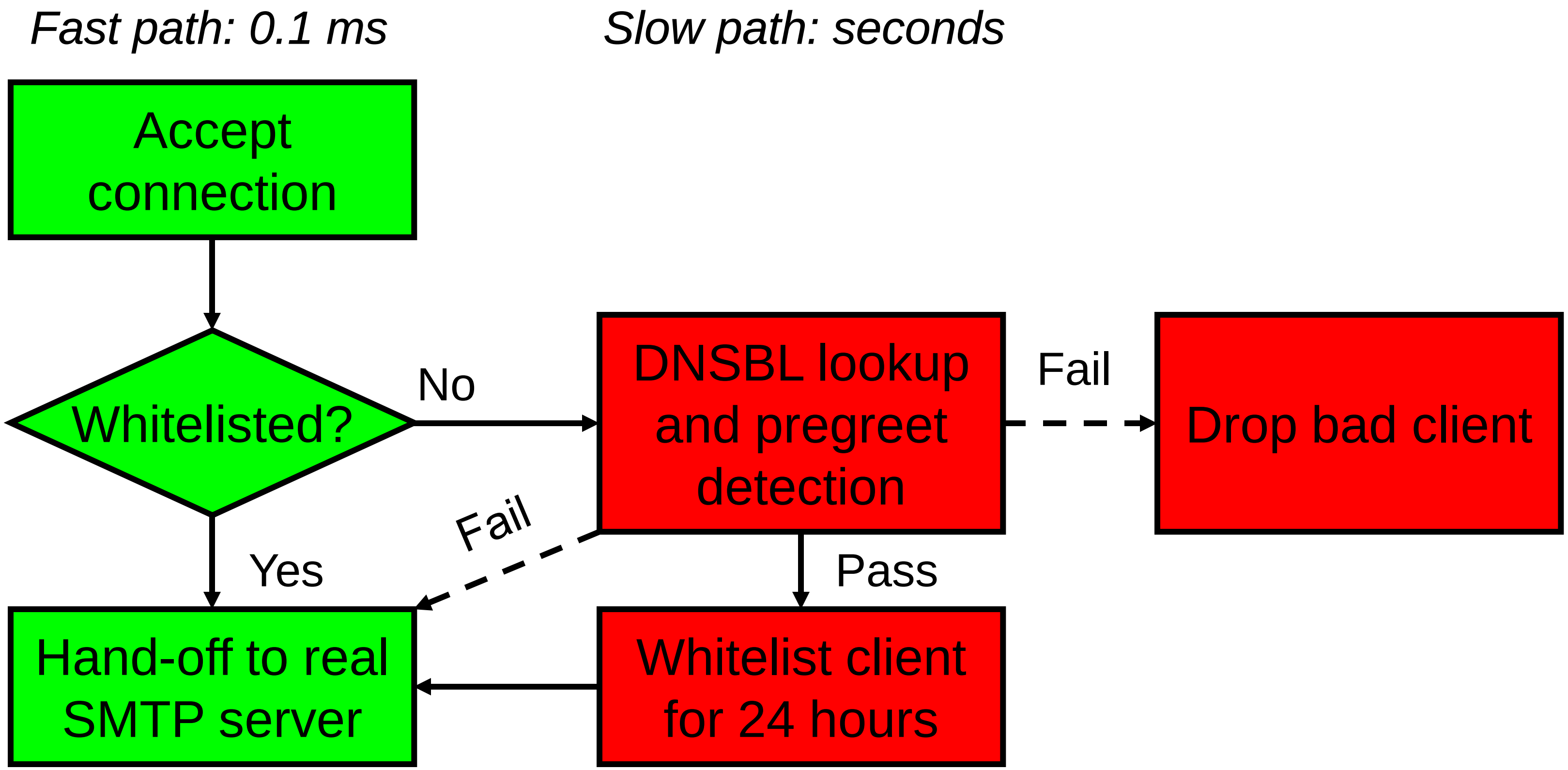
## Prior work: OpenBSD spamd, MailChannels TrafficControl, M.Tokarev



Changing threats

# Prototype postscreen architecture

## All lookups and connections are handled in parallel



Changing threats

# Non-production prototype (source code is on-line)

Code name *postscreen*, but name will likely change.

Single daemon checks all connections *first,* so that Postfix SMTP server processes waste less time.

- PREGREET detection (bots that start talking too soon).
- Parallel lookups for multiple DNS blocklists.
- Fast-path cache (24 hours) for clients that pass the tests.

Not a proxy. No need to handle STARTTLS, etc. Just send the network socket to the real SMTP server.

- Add mini-SMTP engine to log rejected sender/recipient.

Changing threats

# Pregreet detection

## Botnet/proxy SMTP clients that speak before their turn

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>
```

Poor results with the Sendmail *greetpause* approach: wait several seconds before sending the 220 greeting.

- Some clients spontaneously send QUIT after 5 seconds.
- Some clients spontaneously hang up after few seconds.

Bad idea to do such delays in the SMTP server itself!

# Quick question

Q1: How do you find out if a house has a dog?

A1: You listen and wait until a dog barks.

Q2: What if I don’t want to wait?

A2: You ring the doorbell, and it will bark immediately.

Changing threats

# Pregreet detection improved - multi-line reply trap

## Catching more SMTP clients that speak before their turn

Good clients wait for the end of multi-line server reply:

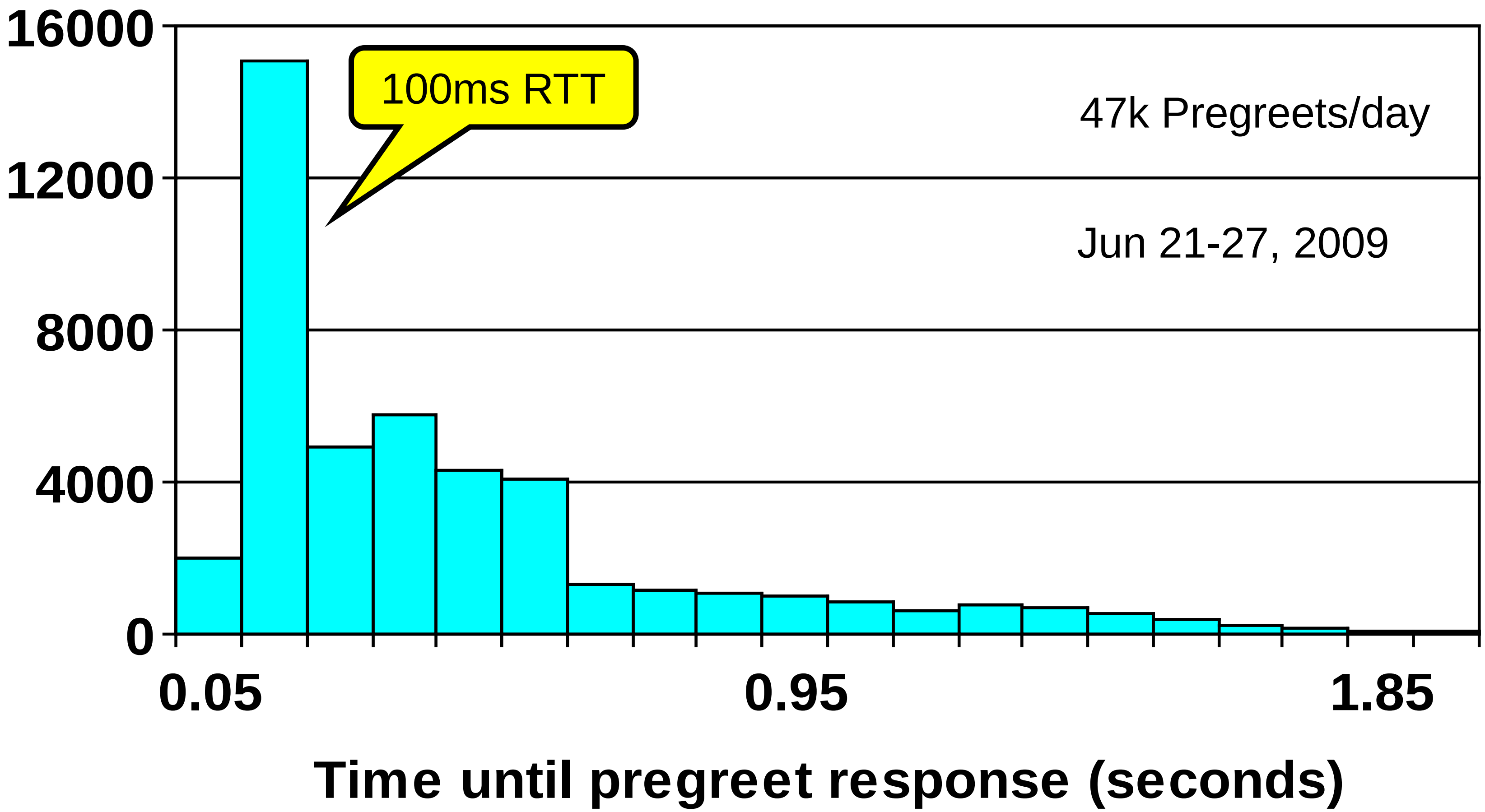
```
postscreen:  220–server.example.com ESMTP Postfix<CR><LF>
postscreen: [pause a few seconds here]
real smtpd:   220  server.example.com ESMTP Postfix<CR><LF>
good client:   HELO client.example.org<CR><LF>
```

Some 50% of the bots starts talking immediately:

```
postscreen:   220–server.example.com ESMTP Postfix<CR><LF>
spambot:      HELO i-am-a-bot<CR><LF>
```

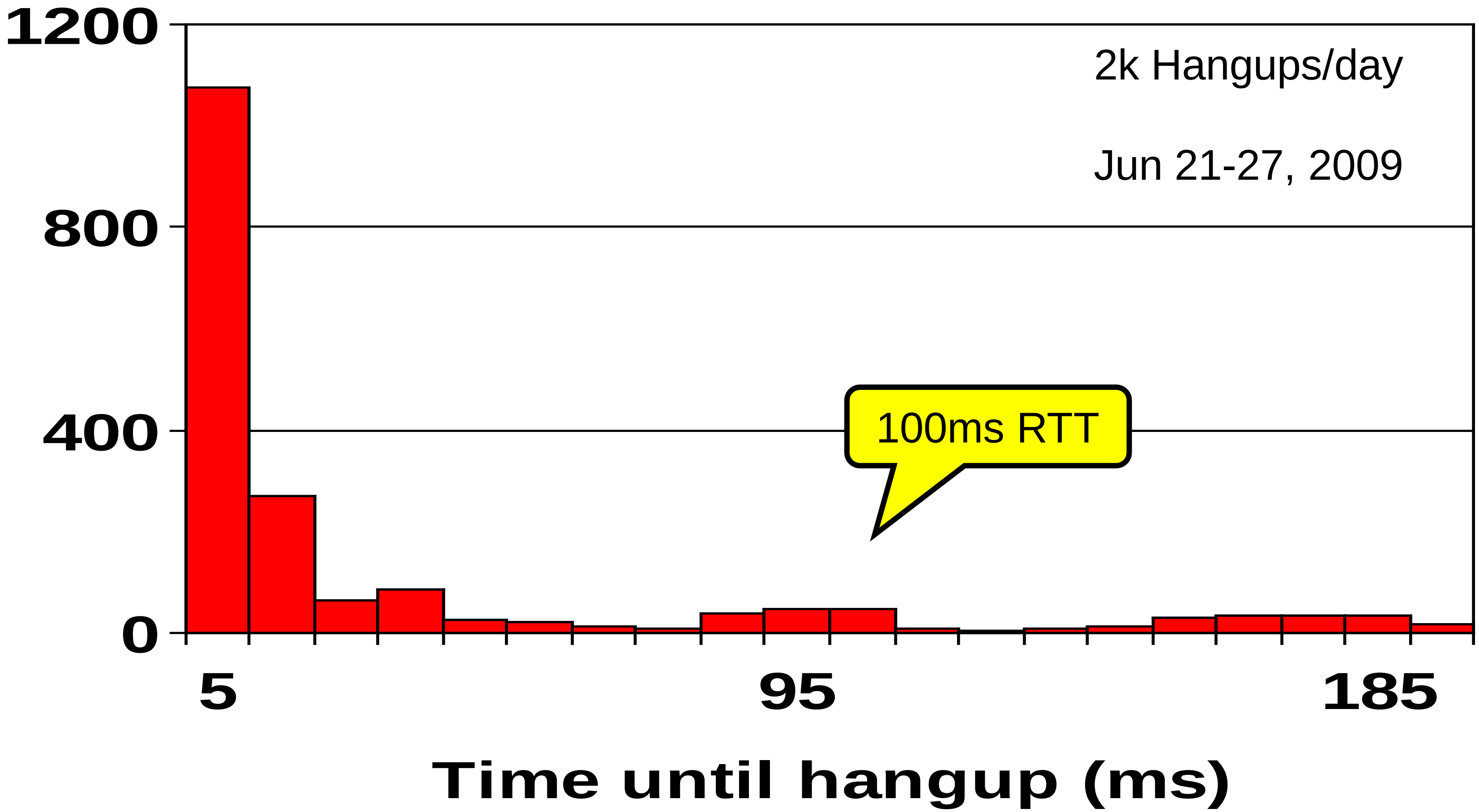
# Results: >50% of bots pregreet (charite.de)

## 99% are blocklisted, but that may change



Changing threats

# Result: 2% of bots hang up quickly (charite.de)

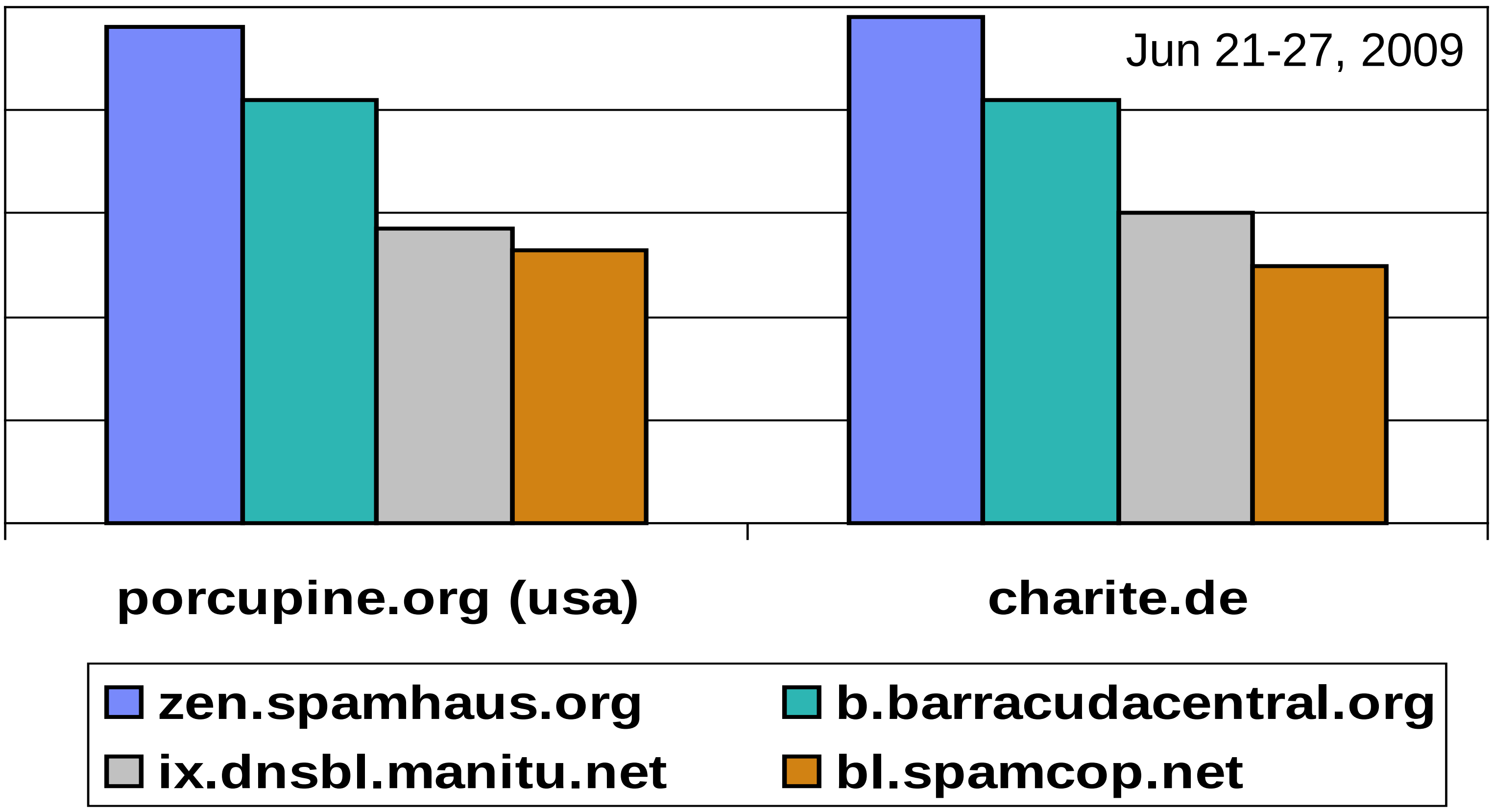


2k Hangups/day

Jun 21-27, 2009

Changing threats

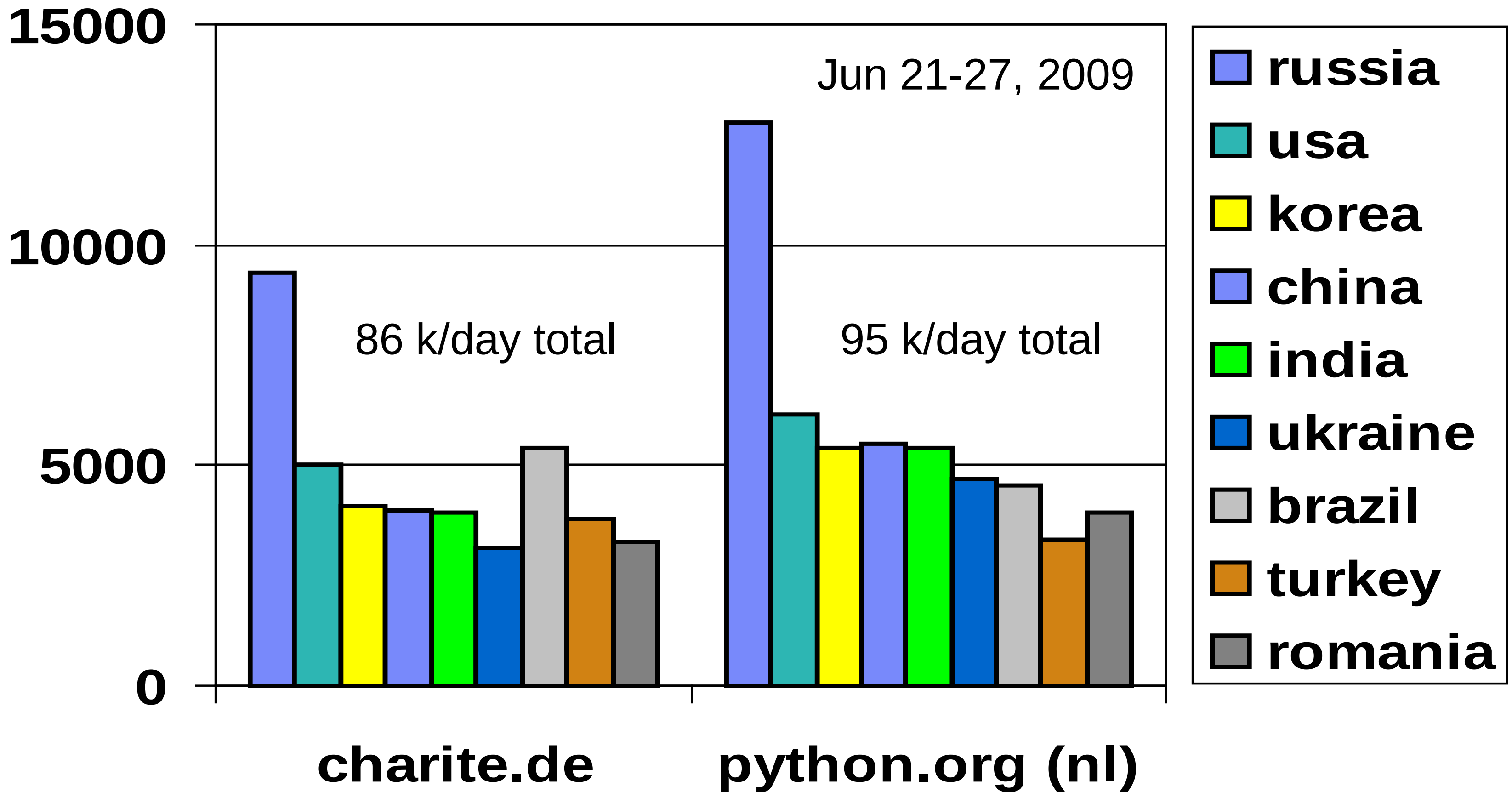
# Relative DNS blocklist client IP address coverage



Changing threats

# Results: spam connections/day

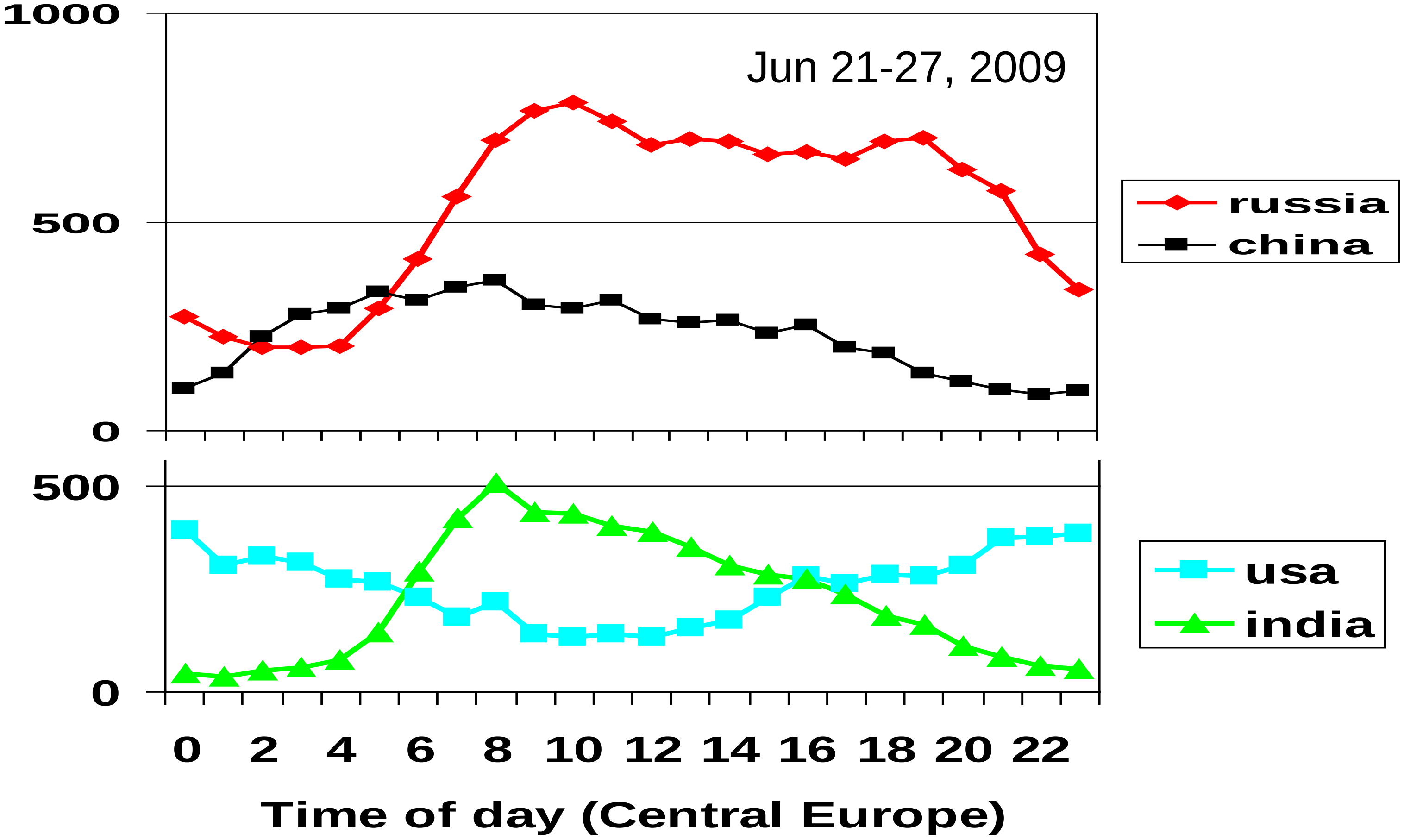
## Spam according to zen.spamhaus.org DNS blocklist



Changing threats

# Results: spam connections/hour (python.org)

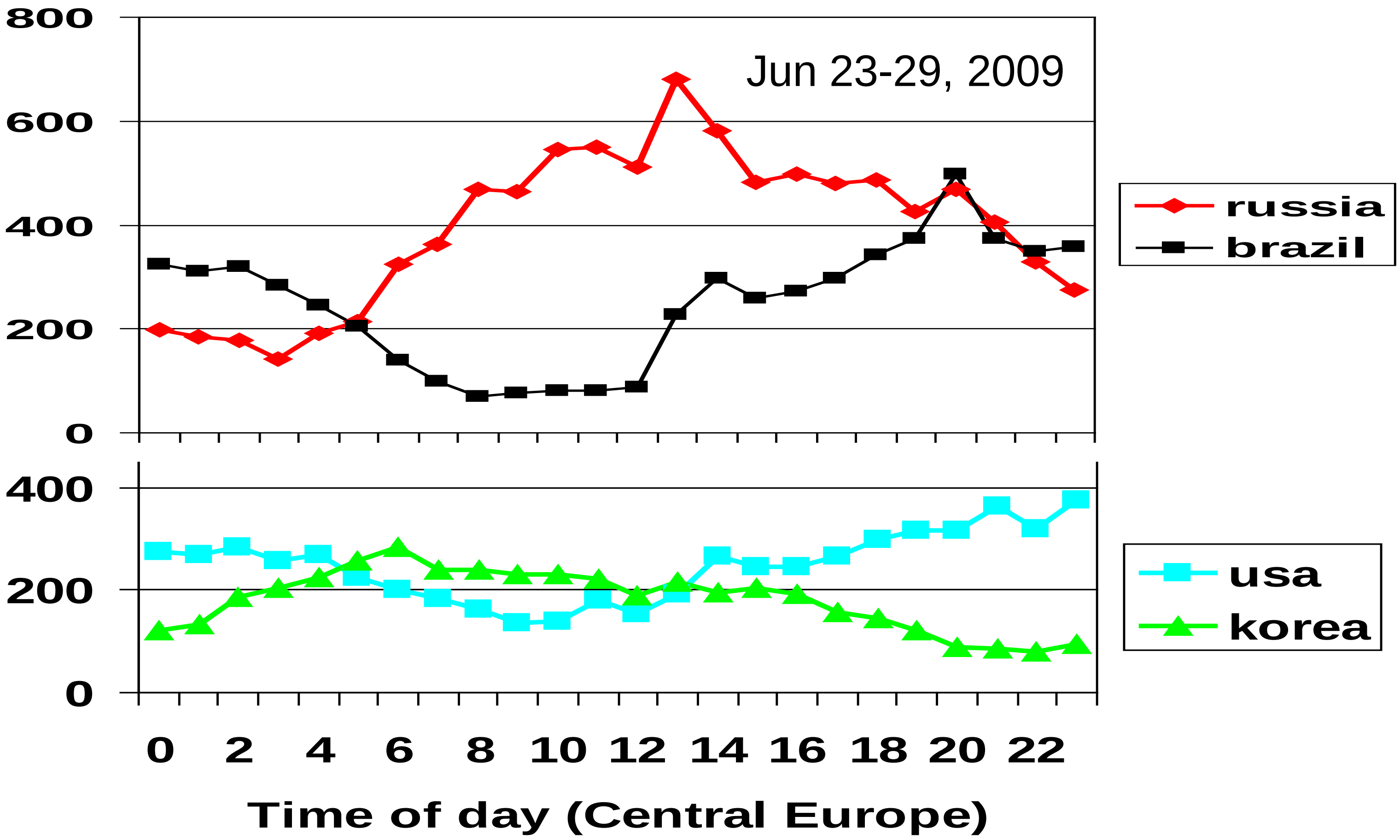
## Spam according to zen.spamhaus.org DNS blocklist



Changing threats

# Results: spam connections/hour (charite.de)

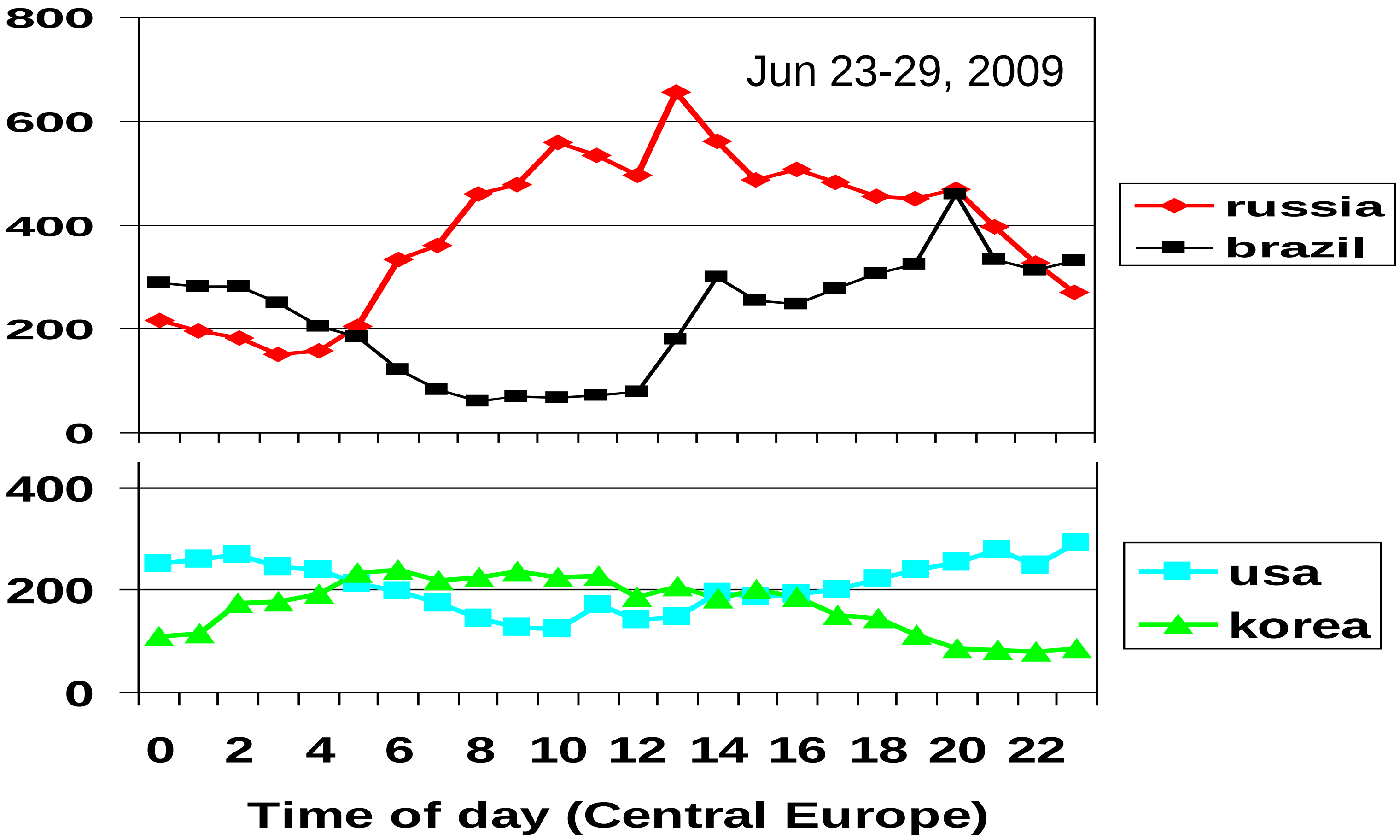
## Spam according to zen.spamhaus.org DNS blocklist



Changing threats

# Results: spam connections/hour (charite.de)

## Spam according to zen.spamhaus.org DNS blocklist



Changing threats

# Future developments

# Postfix-lite and the burden of compatibility

Compatibility makes adding / testing new code harder.

- Internal compatibility: people expect they can upgrade Postfix while it is running.
- External compatibility: workarounds in the SMTP protocol engines; configuration parameters have backwards compatible default settings with surprises.

Postfix-lite:

- Make it simpler - don’t try to support old ideas forever.
- Make it more pluggable - don’t try to solve all problems.

# Concluding remarks

# Postfix lessons learned

Don’t re-invent mechanisms that already work (e.g., SMTP, Milter, maildir, lookup tables). Invent sparingly.

Build the basic stable protols into the MTA: SMTP, LMTP, TLS, SASL, IPv6, DSN, MIME, LDAP, SQL.

Use plug-ins for future proofing: Anti-Spam, Anti-Virus, DKIM, SenderID, SPF, greylist, etc.

Know when to stop, at least for a while.

# Conclusion

Postfix has matured well. With a system implemented as many small programs, features can be added by changing small programs or adding small programs.

Extensibility is a life saver[1]. It eliminates the pressure to implement everything and the kitchen sink within the mail system itself.

The battle continues. For the near future, connection filtering can help to keep servers operable under increasing zombie loads.

[1] For both author and software.

# Postfix Pointers

The Postfix website at http://www.postfix.org/

Books by other people:

- Ralf Hildebrandt, Patrick Koetter, *The Book of Postfix* (2005).
- Peer Heinlein, Das Postfix-Buch - Sichere Mailserver mit Postfix, 3rd ed. (2008).
- Kyle Dent, *Postfix The Definitive Guide* (2003).
- Richard Blum, *Postfix* (2001).
- Original books and translations in German, Japanese, Chinese, Czech, and many other languages.