High-Capacity Email Systems

Steve VanDevender
University of Oregon

Goals of this tutorial

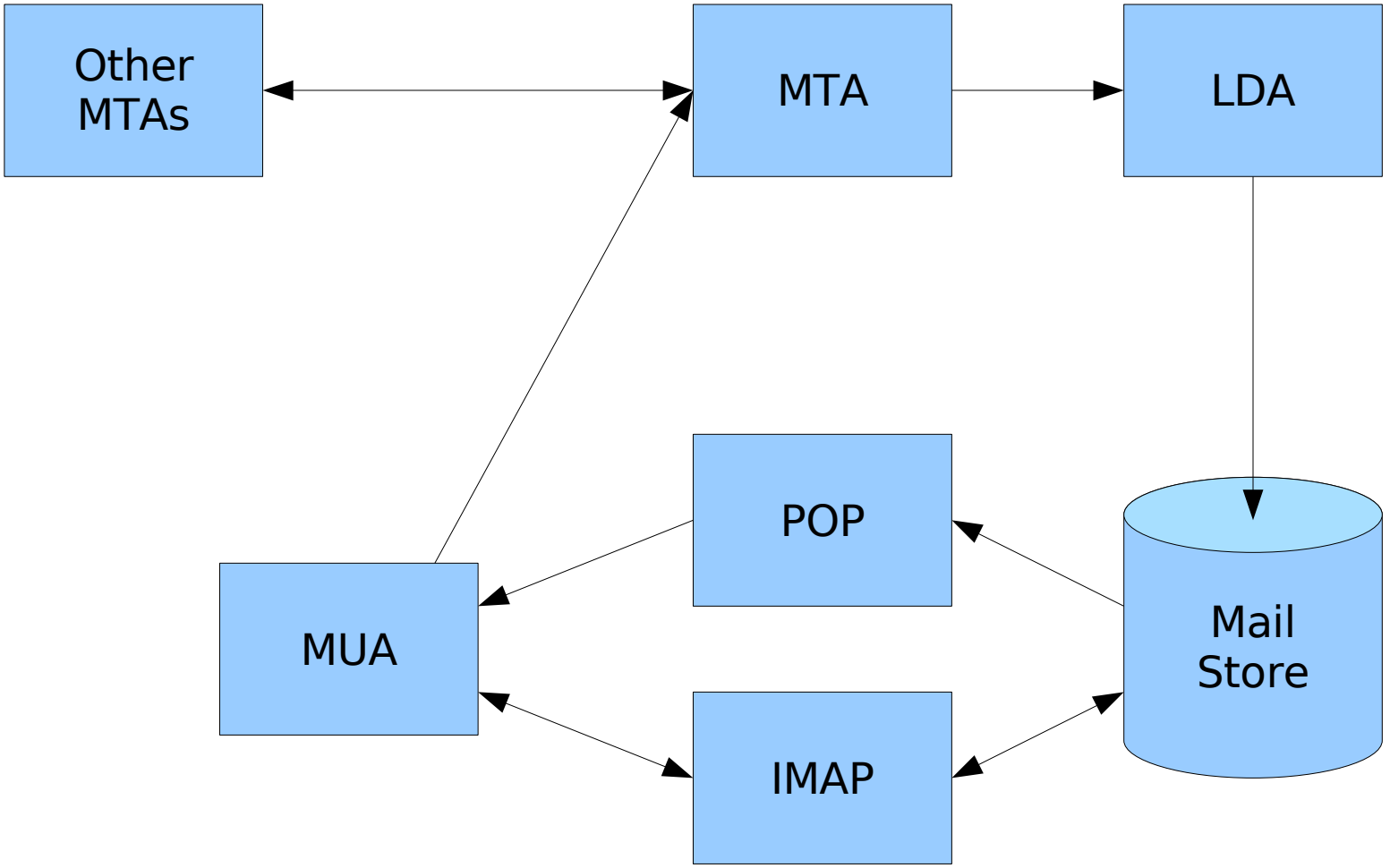
- Present a comprehensive overview of email systems (transmission, delivery, access)
- Discuss issues important to large email systems (efficiency, interoperability, abuse prevention)
- Discuss factors that affect email system performance and ability to handle growth

Overview

- Introduction: Email system architecture, components, and protocols
- Implementation considerations for components of large email systems
- Techniques for high availability and growth
- Case studies

Introduction: Mail System Architecture, Components, and Protocols

Email system block diagram



MTA (Mail Transfer Agent)

- Routes email based on recipient information
- Common MTA software: Exim, Postfix, Sendmail
- Mail destined for local users is passed to LDA (Local Delivery Agent)
- Mail to remote users is passed to another MTA using SMTP (“Simple” Mail Transfer Protocol)

SMTP (Simple Mail Transfer Protocol)

- Defined in RFC 2821 (update of original RFC 821)
- Simple, text-based protocol
  - Four-letter commands (usually) with arguments (`HELO/EHLO`, `MAIL From:`, `RCPT To:`, `DATA`, `QUIT`)
  - Three-digit status codes in response to commands, with optional text comments
- Essentially the only network protocol for email transmission on the Internet

SMTP example

```
220 smtp.uoregon.edu ESMTP Sendmail 8.13.8/8.13.8; Wed, 14 Feb 2007 12:56:06 -0800
>>> EHLO hexadecimal.uoregon.edu
250-smtp.uoregon.edu Hello hexadecimal [128.223.142.97], pleased to meet you
250-ENHANCEDSTATUSCODES
250-PIPELINING
250-EXPN
250-VERB
250-8BITMIME
250-SIZE 10485760
250-DSN
250-ETRN
250-AUTH PLAIN LOGIN
250-DELIVERBY
250 HELP
>>> MAIL From:<stevev@hexadecimal.uoregon.edu> SIZE=342 AUTH=<>
250 2.1.0 <stevev@hexadecimal.uoregon.edu>... Sender ok
>>> RCPT To:<stevev@uoregon.edu>
250 2.1.5 <stevev@uoregon.edu>... Recipient ok
>>> DATA
354 Enter mail, end with "." on a line by itself
>>> From: stevev@hexadecimal.uoregon.edu
>>> To: stevev@uoregon.edu
>>> Subject: example
>>>
>>> .
250 2.0.0 l1EKu60m023771 Message accepted for delivery
>>> QUIT
221 2.0.0 smtp.uoregon.edu closing connection
```

Common SMTP commands

- EHLO (Extended HeLO)
  - If SMTP greeting includes the substring ESMTP, EHLO asks for ESMTP capabilties
- MAIL From:<sender>
  - Specify “envelope sender” (not necessarily same as From: header) used to route bounces
- RCPT To:<recipient>
  - Specifies recipient (may be used more than once)

Common SMTP commands (cont)

- DATA
  - Begins transfer of actual message content
- QUIT
  - Completes SMTP transaction
- STARTTLS (after EHL0)
  - Negotiate SSL/TLS (session restarts)
- AUTH <method> (after EHL0)
  - Attempt user authentication with specified method

SMTP responses

- General form (always starts with 3 digits)
  - `999 comment text`
- First digit is general status
  - 2: OK, action completed
  - 3: OK, continue
  - 4: Temporary error, try again later
  - 5: Permanent error, don't try again
- Remaining digits provide more detail; comment may start with 9.9.9 extended status code for even more detail

MTA vs. MSA SMTP

- MTA SMTP operates on TCP port 25
- MSA SMTP operates on TCP port 587
- MSA mode is specifically intended for user agent message submission
  - Can be configured with different behavior like required TLS encryption or authentication, or different acceptance and relaying rules
- Ideally port 25 should just be for inter-MTA traffic and user agents should use only port 587, but some still use port 25

LDA (Local Delivery Agent)

- Delivers a message into the mail store for a specified user
- More sophisticated LDAs (such as procmail) can do user-configurable delivery to alternate locations (such as different folders, pipes, /dev/null, forwarding) based on inspection of message content

RFC 822 message format

- Internet email messages use a conventional format originally defined in RFC 822 (also used for other things like USENET, HTTP)
- Messages consist of headers and body
  - headers: `header-name: data`
  - multi-line headers indent lines with whitespace
  - body starts after blank line, free format (although lines should be <1000 chars)

Common headers

- `From:`
  - header sender, may not match real sender
- `To:`
  - header recipient, may not match real recipient
- `Subject:`
- `Date:`
- `Message-ID:`
  - (ideally) globally unique message identifier

Common headers (cont)

- `Received:`
  - each MTA which processes message inserts one with tracking information
- `Return-Path:`
  - original envelope sender
- `MIME-Version:, Content-type:`
  - MIME headers
- `X-*`
  - nonstandard application-specific headers

The Mail Store

- Message storage for users
- Each user at least has a primary inbox folder (default location for delivery and message retrieval) and may also have access to additional folders
- The mail store format is probably the biggest influence on performance of your mail system
- All components that interact with the mail store must use compatible locking

Mail access with POP

- POP (Post Office Protocol) is the original protocol for remote mail access (POP3 is latest protocol revision)
- Supports only a single inbox folder
- Each mail check requires login, scanning folder to index messages and identify new messages, logout
- Messages can be downloaded and (optionally) deleted

POP session example

```
+OK Dovecot ready.
>>> user stevev
+OK
>>> pass password
+OK Logged in.
>>> stat
+OK 1 403
>>> retr 1
+OK 403 octets
Content-Type: text/plain; charset="us-ascii"
Content-Disposition: inline
Content-Transfer-Encoding: 7bit
MIME-Version: 1.0
X-Mailer: AlphaMail 1.0.22
Message-ID: <1164891521.43097.alphamail@mailapps1.uoregon.edu>
X-Originating-Ip: 71.236.255.164
From: "" <stevev@uoregon.edu>
Reply-To: "" <stevev@uoregon.edu>
To: stevev@uoregon.edu
Subject: test
Date: Thu, 30 Nov 2006 04:58:41 -0800

test
.
>>> quit
+OK Logging out.
```

Mail access with IMAP

- IMAP (Internet Message Access Protocol) was originally developed for the PINE mail client, but is now a predominant client access protocol
- Supports multiple folders
- Supports persistent connections
- Oriented towards leaving mail in the server repository for access from multiple locations and clients

IMAP session example

```
* OK Dovecot ready.
>>> 1 login stevev password
1 OK Logged in.
>>> 2 select inbox
* FLAGS (\Answered \Flagged \Deleted \Seen \Draft)
* OK [PERMANENTFLAGS (\Answered \Flagged \Deleted
\Seen \Draft \*)] Flags permitted.
* 1 EXISTS
* 0 RECENT
* OK [UIDVALIDITY 1157275920] UIDs valid
* OK [UIDNEXT 2] Predicted next UID
2 OK [READ-WRITE] Select completed.
>>> 3 logout
* BYE Logging out
3 OK Logout completed.
```

MUAs (Mail User Agents)

- Provide a user interface for viewing and composing mail messages
- Standalone MUA applications include Outlook, Thunderbird, Eudora, PINE, mutt, and many, many more
- Web-based email access has become very popular and common
- MUA behavior is often what causes your mail performance problems

Implementation concerns for MTAs and SMTP

Mail relaying

- Mail relaying is the acceptance of messages from one network location for transfer to another network location (as opposed to origination of email or final delivery to a local recipient)
- Originally mail relaying was not usually restricted, but spammers exploited unrestricted relays and restriction of relaying has since become a necessity

Relaying restrictions

- Domain of envelope sender
  - Risky since any spammer could choose a domain that let him relay
- Network address of client
  - Simple for clients that lived in the permitted ranges, but difficult for roaming clients
- SMTP Authentication
  - ESMTP AUTH could be used to authenticate to SMTP in much the same way clients authenticated for POP or IMAP

SMTP AUTH

- Clients can supply a username/password pair to the SMTP server, or enage in some challenge/response interaction, to obtain relaying privileges
- Methods commonly supported in clients (`AUTH PLAIN` and `AUTH LOGIN`) use base64-encoding for authentication data (not encrypted! not network-safe!)
- AUTH methods with encryption are not commonly supported

STARTTLS

- STARTTLS provides optional SMTP session encryption based on SSL/TLS
- Clients have to support ESMTP and can issue STARTTLS if the ESMTP server advertises STARTTLS as a capability in response to EHL0
- Restarts SMTP session with encrypted traffic once established, so it prevents sniffing of SMTP protocol exchange and message data

STARTTLS saves AUTH

- You'd like to use AUTH to allow users to authenticate for relaying, but then they'd leak their passwords
- Clients that support AUTH usually support STARTTLS too
- Set up your MSA (but not MTA) port to advertise and require AUTH after STARTTLS
- Roaming and remote users are happier, and have somewhat better privacy too

TLS && AUTH Mini-howto

- Sendmail
  - `define(`confAUTH_OPTIONS', `p')`
- Postfix
  - `smtpd_tls_auth_only = yes`
- Exim
  - `auth_advertise_hosts = ${if eq{$tls_cipher}{}{}{*}}`

Spam

- Unsolicited Bulk Email
  - not requested by recipients, sent to multiple recipients
- Many sites report >90% of their incoming email is spam, so it becomes a big scaling problem for large sites
- You're stuck with trying to use technical means to solve a social problem
- There are no perfect solutions, and the imperfect ones keep changing

Characteristics of spam

- Most is now coming from a huge pool of randomly-selected proxies, typically compromised Windows systems
- A lot of spam uses forged sender addresses
  - Any approach which might return spam to forged addresses should be avoided
- Some still comes from relatively fixed locations that are easier to block

Considerations for spam mitigation

- Should be driven by user needs and organizational considerations
- Blocking vs. filtering
  - Blocking: refuse spam at SMTP time
  - Filtering: accept spam, but tag or divert it based on content
- Should allow user customization
  - Opt-in or opt-out for system-wide blocking
  - Filtering mechanisms are often customizable, but not always easily

Preventing spam from your site

- Have abuse@, postmaster@ contact addresses **that are not blocked** to ensure you can be notified of issues
- Prevent open relays, open proxies
- Watch for vulnerable web applications
  - Lots of PHP crud seems to be easily exploited for spam injection
  - Avoid formmail scripts that take addresses from form input
- Monitor message volumes, mail queues

Blocking: DNSBLs

- DNS BlackLists
  - Lists of IP addresses accessible by DNS lookups in special zones
- Maintainers have many listing policies
  - Known spam sources
  - Spammer-controlled networks
  - Dynamically-assigned (non-server) IPs
  - Examine policies, choose carefully!
- DNSBLs are about the only approach that reduces spam without adding server load

Blocking: DNSBLs (cont)

- May need to use local access control features to override DNSBL listings (“whitelisting”)
- Some DNSBLs allow you to download copies of their zone data (perhaps for a fee) to reduce lookup latency
  - Zones can be very large (>10^6 entries) and take significant resources to serve
- Spammer use of zombies/proxies is largely a reaction to DNSBLs

Blocking: Local access control

- Supplement or override DNSBL coverage
- Can be labor-intensive to manage
- Local user opt-in or opt-out of blocking
  - Sendmail: `FEATURE(delay_checks)`, `Spam:user@` entries in access_db
  - Postfix: http://www.postfix.org/RESTRICTION_CLASS_README.html
  - Exim: `recipient_reject_except = <address list>` (may also refer to external file or database)

Blocking: “Greet-pause”

- Proper SMTP clients are supposed to wait for SMTP banner before proceeding
- Pause before displaying banner, reject clients that send data before banner
- 30-second limit imposed by RFCs
- Interferes with some real sites (like Gmail)
- Slows down processing of email
- Spammers are already adjusting for this

Blocking: Greylisting

- Spamware often doesn't implement full MTA behavior, like queuing
- Defer initial attempt to deliver with 4xx reply, accept on subsequent retry
- Requires keeping a database to remember connection attempts
- Successful clients can also be remembered and not delayed
- Slows down mail processing **a lot**, depends on queuing behavior of client

Content filtering

- Examine message contents for spam or malware indicators
  - Can be applied during DATA phase using Sendmail/Postfix milters
- Can be very resource-intensive
  - SpamAssassin uses more CPU than any other component in delivery process
- Adaptable and customizable
- Also easily worked around by spammers (image spam, random text)

Spam mitigation methods you should avoid

- Challenge-response
  - Unknown senders sent back a challenge, have to respond to get message to recipient
  - Spam sender forgery means lots of innocent third parties get useless challenges
  - Even valid senders find it annoying
- SMTP Callbacks
  - Attempt to connect back to purported sender's SMTP server to verify sender
  - Once again, problematic because of forgery

My personal favorite methods for spam mitigation

- Use an appropriate DNSBL to keep spam from getting in to your mail system
  - spamhaus.org is conservative and effective
- Provide content filtering for what gets in
  - ClamAV is very effective for virus filtering and some phishing spam, clamav-milter can reject at end of SMTP DATA phase to minimize backscatter
  - SpamAssassin seems to be leading spam filtering solution, with many content heuristics and trainable Bayesian filtering

What if your site is blocked?

- Many DNSBLs provide information about their listings with evidence
- Remote sites that are blocking you will hopefully tell you why
- Make sure you fix any spam problems that caused blocking
- Other things that might cause blocking:
  - nonexistent or inconsistent rDNS
  - RFC-compliance issues

Mail queuing

- Email is designed to be store-and-forward, to handle temporary delivery problems
- Mail that can't be delivered immediately is placed in a queue for later retries
- Large sites can carry large queues
- Large site or network outages can greatly increase your queue

Handling large mail queues

- Typical queue defaults hold messages for five days before giving up
  - You could reduce this a little
- Most MTAs are configurable for multiple queues with different policies
  - Simple policy retries at fixed intervals
  - Muliple queues can implement different intervals or an exponential-backoff policy
- Queue items are usually text files that can be managed by hand or with scripts

Mailing list management

- Use mailing list management software that provides for subscription confirmation and user subscription management
- Mailing lists generate almost entirely outbound traffic; you may want to use a separate MTA for the mailing list tuned to its traffic patterns

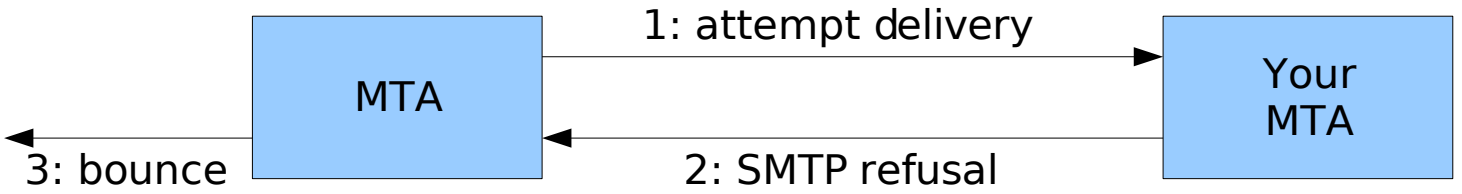
Refuse-or-deliver philosophy

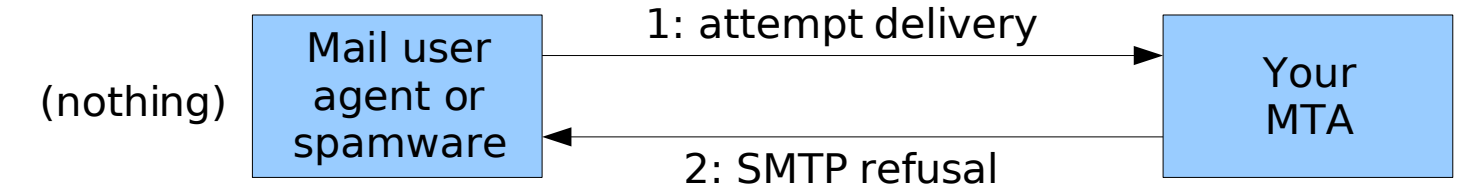
- Messages should be accepted for delivery, or refused at SMTP time
- Other dispositions cause problems
  - Discarding silently leaves a sender with the impression the mail was delivered when it wasn't
  - Returning a non-delivery notification after acceptance can send messages to forged addresses (“backscatter”) or cause loops
- When a message isn't accepted, a valid sender can find out

Bounce messages

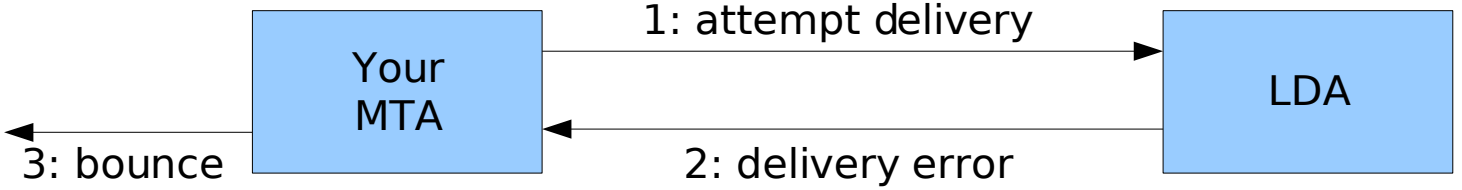
- Non-delivery notifications (“bounce messages”) are returned by MTAs to a sender when a message is undeliverable
- Real users who are using an MTA can get valid bounces
- Spamware/malware in communication with an MTA tend not to generate a bounce when they fail to deliver
  - Refusing at SMTP time avoids “backscatter” of delivery notifications to forged senders

Bounce scenarios

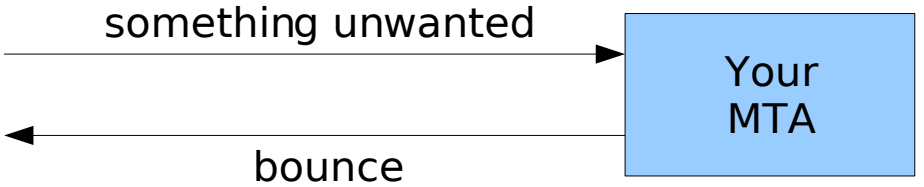




Avoid this



DON'T DO THIS:



Avoiding undesirable bounces

- Refuse unwanted messages during SMTP with 550 status
  - Blocked sender, unknown/invalid user at RCPT `To:` (so recipients can opt out of blocking)
  - Spam/virus content at end of `DATA`
- Try to avoid local delivery errors that cause your MTA to send bounces
- Spam filtering or virus scanning should not return messages or send notifications to (usually forged) senders

Mail forwarding

- Users often want to forward mail
  - funnel mail from multiple accounts to one
  - get mail in old account to new account
- Most MTAs support user-controlled forwarding
  - .forward file checked for forward address
- Some LDAs also do forwarding
  - procmail: `!forward@address.com`

Mail forwarding issues

- .forward-style forwarding is usually safer
  - MTAs apply null-sender checking for bounces, Received: header thresholds to prevent indefinite looping
- LDA forwarding can be dangerous
  - LDA forwarding to a non-working destination may create a mail loop, especially if no attempt is made to match that condition
- SPF vs. traditional forwarding
  - same-sender forwarding blocked by SPF

Implementation concerns for LDAs

- Good to have one with configurable delivery behavior, especially for content filtering
  - Users can also do their own mail sorting
  - I like procmail, although it is ugly to configure
- Quotas
  - Good: prevents one mailbombed user from hosing everyone else
  - Bad: causes bounces and user confusion for over-quota users

Mail Store Implementations

“mbox” Mail Store

- Earliest and most common mail store format
- One file contains multiple messages separated by “From_” lines
  - `From stevev@uoregon.edu  Sun Feb 18 18:00:00 2007`
- Standard inbox location for users is /var/mail/$USER, but can be changed to spread those over more directories/disks
- Additional folders normally placed in user's home directory

mbox locking

- Changes to mbox files must be serialized and exclusive to avoid corruption
- Exclusion locking traditionally done with “dot-lock” files
  - Create temp file with random unique name
  - Attempt to link temp file to folder name with “.lock” appended (/var/mail/stevev.lock)
  - Remove temp file
- OS-level file locking (`fcntl()`, `flock()`) can also be used in addition to dot-locks

mbox problems

- Large folders become unwieldy to handle
  - Getting a folder index requires reading and parsing the entire folder
  - Updates to delete messages, modify headers, etc. usually involve complete rewriting of folder file
- Lock contention becomes problematic (clients fight with delivery, or high-rate delivery fights with itself) because some updates lock a folder for a long time

Maildir Mail Store

- Store individual messages in individual files in a subdirectory
- Standard Maildir format uses three subdirectories in `$HOME/Maildir`:
  - `cur/` holds current folder contents
  - `new/` holds newly-delivered mail
  - `tmp/` used for message delivery and deletion
- Maildir++ supports multiple folders
  - Subdirectories like `$HOME/Maildir/.folder`
  - Path flattening: `sub/folder => .sub.folder`

Maildir Mail Store (cont)

- Delivery is lock-free, based on unique filenames
  - `1171963019.6003_0.mserv5`
  - (UNIX time).(PID, serial).(hostname)
- A new message file is initially created in `tmp/`, then moved to `new/` when it is fully written out
- Access renames message files from `new/` to `cur/` appending status flags to name
  - `1171936019.6003_0.mserv5:2,S`

Maildir locking

- Almost unnecessary, although some servers implement folder-level locking for extra safety
- Done at folder level when updates are done, to ensure other clients get a consistent view of folder contents

Maildir and large folders

- Maildir can do much better than mbox in many cases, if underlying OS and filesystem is good at handling lots of small files and fast at directory traversal and open/read/close transactions
- Many benefits come from many kinds of updates being faster file link/unlink or renaming, rather than slow rewriting of large files
- Tends to interact better with NFS

Database Mail Store

- Few current UNIX systems implement these
- Advantages from databases:
  - Indexing and access can be very fast
  - Can save space by linking a single copy of a message to multiple recipients
  - Databases tend to have solid record-locking primitives

Database Mail Store (cont)

- Disadvantages from databases
  - Data structures are complex and fragile; damage can cause widespread data loss or corruption
  - Require more specialized tools for backup and message manipulation, compared to mbox or Maildir which can use more basic file manipulation tools
- Still more of a research topic than a common solution, although performance of trial implementations is encouraging

Putting a Mail Store on NFS

- Delivering mail into NFS was traditionally avoided
  - Tended to be slower and higher-latency than direct-attached storage
  - Less stable than direct-attached storage
  - Locking was unreliable
- NFS got better
  - Network speed increasing faster than disk transfer speed
  - Locking somewhat better, and avoidable

Putting a Mail Store on NFS (cont)

- NFS is about the only storage technology that allows multiple hosts to access the same storage concurrently, allowing parallelization of SMTP, POP, IMAP servers working on a single mail store
- NFS “toasters” have become fast and reliable storage devices
- Most really large mail systems have gone to NFS

POP/IMAP implementation

- Traditional POP and IMAP pass authentication data in the clear
- Optional TLS commands or secure authentication methods are not widely supported in clients
- Standard service ports with required TLS are well-standardized
  - pop3s = TCP port 993, imaps = TCP port 995
- Client support for required TLS is widely available now, so you should require it

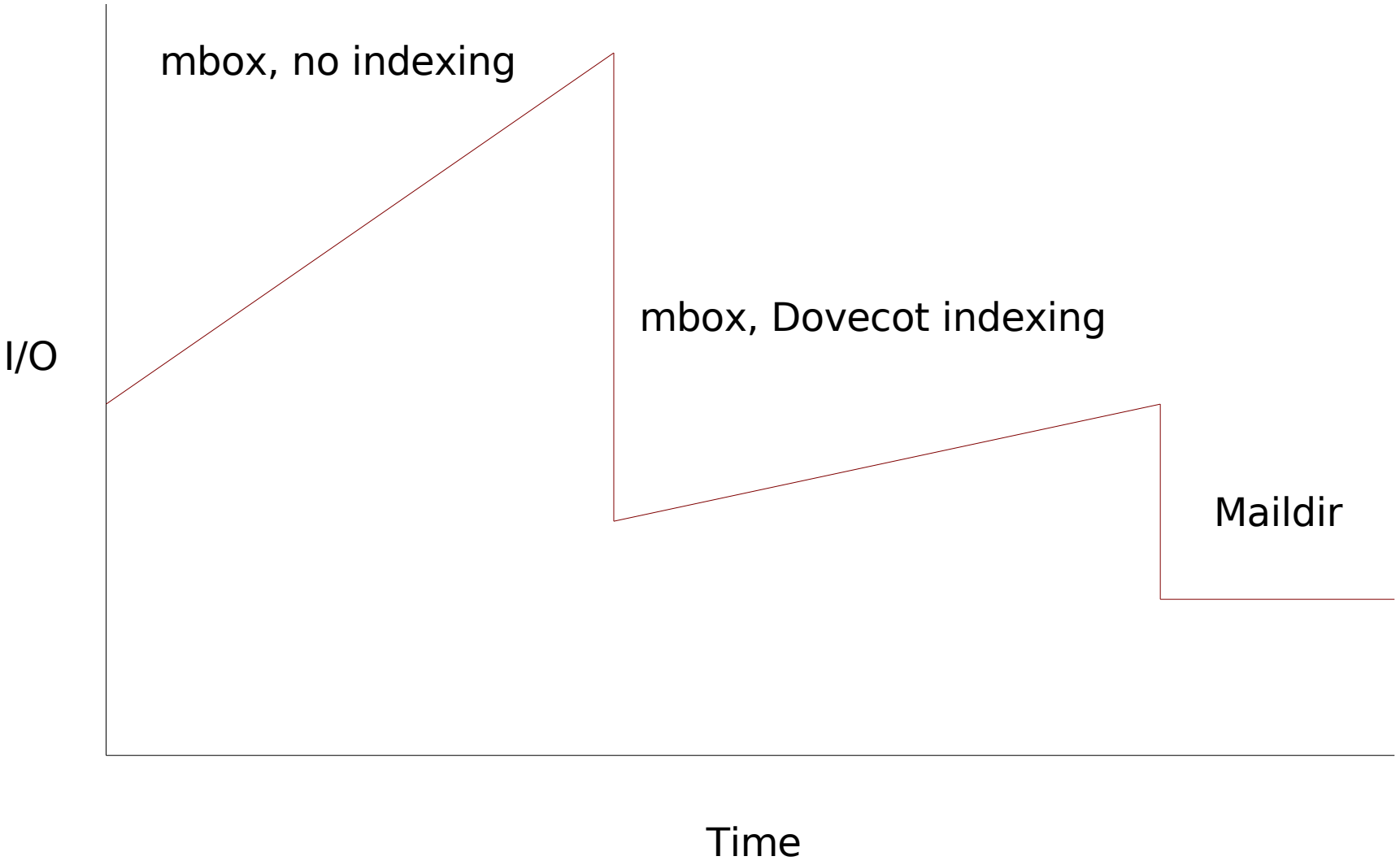
POP/IMAP performance

- POP is transactional (log in, read inbox to index and check for new messages, download/delete messages, log out)
- IMAP can be persistent (log in, do whatever, hang out, do some more, etc.) making it somewhat less I/O intensive
- Large folders still tend to be a problem
  - mbox: lots of file reading
  - Maildir: lots of readdir/open/read/close transactions

POP/IMAP performance (cont)

- Some POP/IMAP daemons support index caching to speed up indexing phase
  - UW IMAP: special “mbx” format that stores index data at beginning of mbox-like folder
  - Dovecot: supports auxiliary index cache files that store index data for both mbox and Maildir folders
- Index caching helps both mbox and Maildir perform better by eliminating unnecessary folder rescanning

POP/IMAP performance (cont)



IMAP shared access

- More and more, people want to access mail from multiple locations
- This often results in multiple clients making overlapping accesses to the same folders and “lockfighting”
- Some clients open multiple sessions on the *same folder*!
- Maildir is about the only thing that helps avoid problems that result from these behaviors, but not completely

Web email clients

- Popular for ease of use and flexibility of access
- Generally like other IMAP client, but often more resource-intensive
  - Lots of quick login/<single command>/logout transactions caused by each web page view
- Load on your IMAP servers can be reduced with an IMAP proxy in front of web email system, or web email system with integrated IMAP session caching

Techniques for high availability and growth

Backup MX hosts

- Traditional method for providing higher reliability for mail transfer
- DNS has a special MX resource record indicating cost and intended SMTP host
- SMTP protocol says to try multiple MXes for a domain in order from lowest to highest cost (or pick at random from those with same cost) until success
- MTAs (but not all MUAs) can be directed to alternate servers when one is down

Problems with backup MXes

- All MXes for a domain need to be configured with exactly the same SMTP-time behavior (blocking, known users) or they can be used to inject spam or generate backscatter
- MTAs have to time out (delays of minutes) trying to contact a nonworking MX before trying the next
- Having equal-priority MXes doesn't guarantee fair round-robin behavior

Load-balancing SMTP

- Load-balancing works well with SMTP due to relatively short transactional nature of SMTP sessions
- Load-balancer can detect and remove a nonfunctional SMTP server from use faster than MTAs time out or DNS updates propagate
- Reduces visible downtime to MUAs
- Load-balancing policy is completely under your control

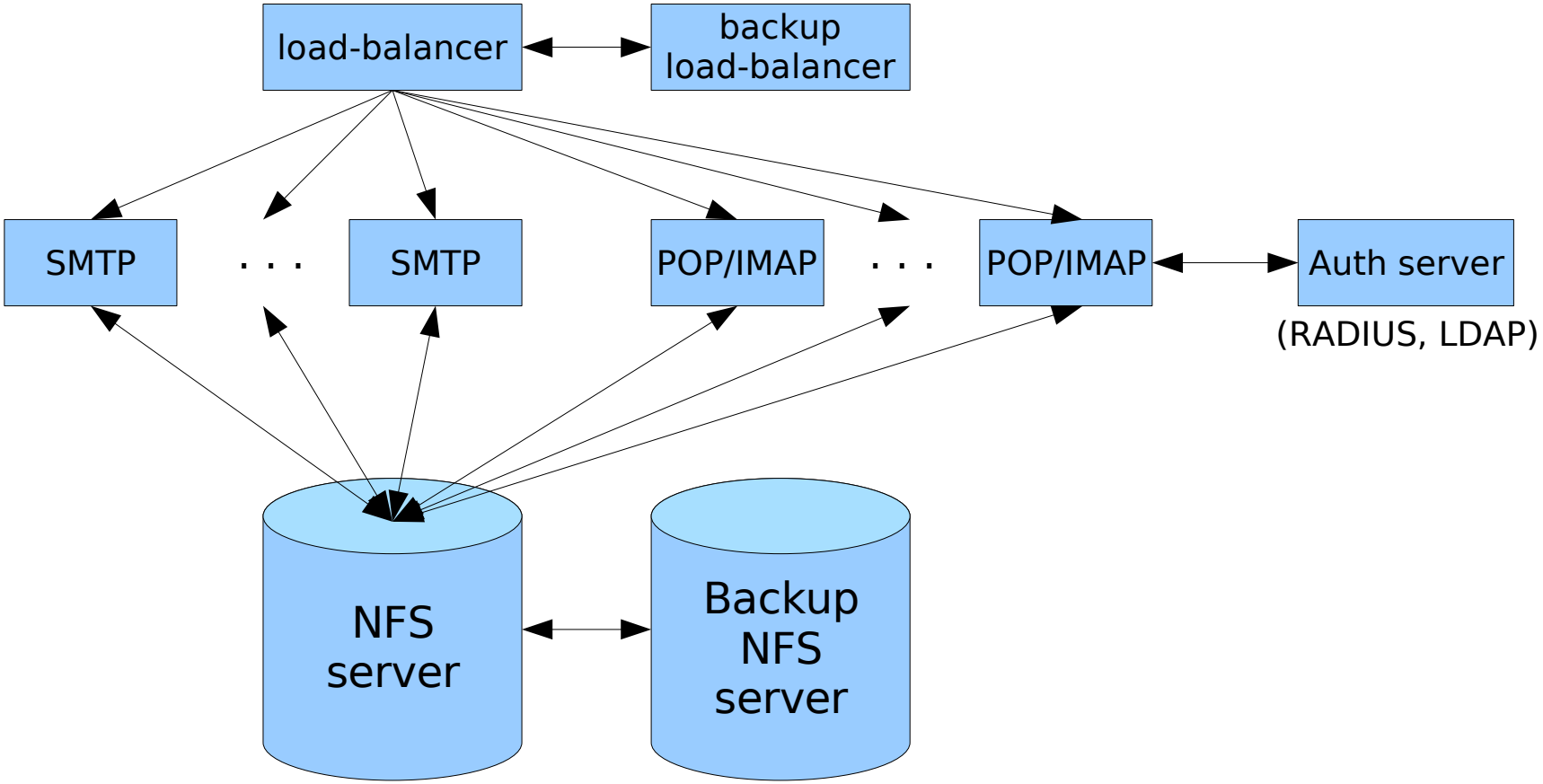
Issues with load-balancing SMTP

- You still have to ensure consistent SMTP-time behavior across all servers
  - Configuration management tools to automate and synchronize updates help here
- Mail queue items distributed across multiple hosts
  - If a host is lost, you might also lose its queue
  - Sharing queue across hosts is problematic
- Centralized log collection

Load-balancing POP and IMAP

- Main concern is reliable exclusion locking
  - User sessions often distributed across multiple hosts in server pool
  - Maildir helps by removing most need for locking and reducing lock durations
- If you're using NFS (you almost have to):
  - `noac` (no attribute caching) mount option ensures NFS clients see consistent file states
  - `fcntl()` locking tends to be more NFS-safe
  - Do same things on SMTP servers

Load-balancing high-availability architecture example



Techniques for growth

- Load-balancing of SMTP, POP, IMAP for parallelization (not just availability)
- Distribute mail store across multiple storage backends
  - /home1 on nfs1, /home2 on nfs2, etc.
- LDAP (also set up for high-availability) for centralized authentication, fields to handle user mail routing and service routing

Case studies

“Brownouts”

- Older, traditional mail system with mbox-format inboxes in `/var/mail/user`
- ~15,000 user accounts
- System “browns out” at midday on busy days; delivery and access become slow
- I/O on /var/mail disk actually *goes down* during brownouts
- System recovers when demand falls off

“Brownouts” (cont)

- Why? Dot-locking in single directory becomes bottlenecked on OS serialization of directory updates
- Solution: Relocated inboxes to home directories (`~user/.mail`)
  - Split I/O across multiple home directory disks, increasing all performance
  - Users no longer contend with everyone else for inbox locks
  - Did require coordinated reconfiguration of LDA, POP, IMAP, UNIX shell MUAs

Maildir conversion

- Biggest, scariest system administration project I've ever been involved with
  - 40,000 users, 50 million messages, 1.6 TB, lots of unhappy people if done badly
- Motivated by chronic problems with mbox performance and lock contention issues
  - Users with >100MB folders
  - MUAs opening multiple sessions on folders and fighting with themselves
  - Stale NFS lock issues

Maildir conversion goals

- Goal: All messages accessible by POP or IMAP should remain accessible after conversion
  - Subgoal: POP and IMAP become only supported access methods after conversion
- How do you find all that mail when it's scattered all over home directories?
  - We were lucky and clever: After previously converting to Dovecot with mbox indexing, index files could be used to find accessed folders

Maildir conversion: how to turn mbox into Maildir?

- mb2md
  - Perl script written by people who did similar conversion
    - http://batleth.sapienti-sat.org/projects/mb2md
  - Splits mbox folder out into Maildir, including parsing headers for message status flags
  - Can also process all mbox folders in a subdirectory
- Chose Maildir++ layout, installed test POP/IMAP daemons set up for Maildir, converted some willing ~~victims~~ users

Per-user Maildir conversion

- Always convert `~user/.mail` (standard home directory inbox)
- Always convert standard `~user/mail` folder directory
- Find Dovecot-created `.imap` index directories containing index files, convert corresponding folder for each index file
  - Many users had folders outside recommended `~user/mail` directory
- Clean up: remove converted mboxes

Maildir conversion: outage planning

- For maximum safety we wanted to avoid changes to stored messages during conversion, but this meant disabling mail services for however long it took
- Ran benchmarks by converting existing mail to scratch location (also validated automated conversion methodology)
- Benchmarks showed some benefits from parallelization, confident of <2 days conversion time (actually took ~26 hours)

Maildir conversion: the big day arrives

- Raised `maxfiles` setting in NetApp file server to accommodate Maildir
  - trial conversions showed average message size of 32 kB, used to set global space::files ratio
- Turn off POP and IMAP servers
- Hide procmail from sendmail
  - Sendmail leaves messages destined for local users in queue if it can't exec LDA
  - Also raised Timeout.queuewarn to suppress “not delivered in 4 hours” warnings

Maildir conversion: the big day arrives (cont)

- Also create snapshots of home directory volumes in case of backout
- Disable user quotas (conversion temporarily more than doubles space usage for a user)
- Run per-user conversion script on a few more test users and validate carefully
- Fire off batch conversions spread over 8 hosts
- Wait . . .

Maildir conversion: post-conversion

- Turn on quotas with somewhat modified quota limits
  - 25% space increase for fragmentation, greatly increased file quotas
- Re-enable procmail configured for Maildir delivery
  - `DEFAULT=$HOME/Maildir/`
- Flush bulging mail queues
- Re-enable POP/IMAP with Maildir configuration

Maildir conversion: interesting problems

- Manual conversion/cleanup for people with odd or nonstandard configurations
  - procmail sorting to Maildir++ folders instead of mbox folders
  - Mail that hadn't been converted because it hadn't been accessed/indexed by Dovecot
- Nasty e1000 driver bug tickled by new NFS traffic patterns with Maildir
  - Interfaces on POP/IMAP servers would shut down due to packet rate and memory stress
  - Ultimately had to install locally-built driver

Maildir conversion: the aftermath

- Goal reached: 99+% of users noticed no difference after conversion
- Really did eliminate issues with lock contention and NFS
- Performance is mainly better, but I/O load on NFS server turned into CPU load from higher rate of NFS requests

High-Capacity Email Systems

Steve VanDevender
University of Oregon

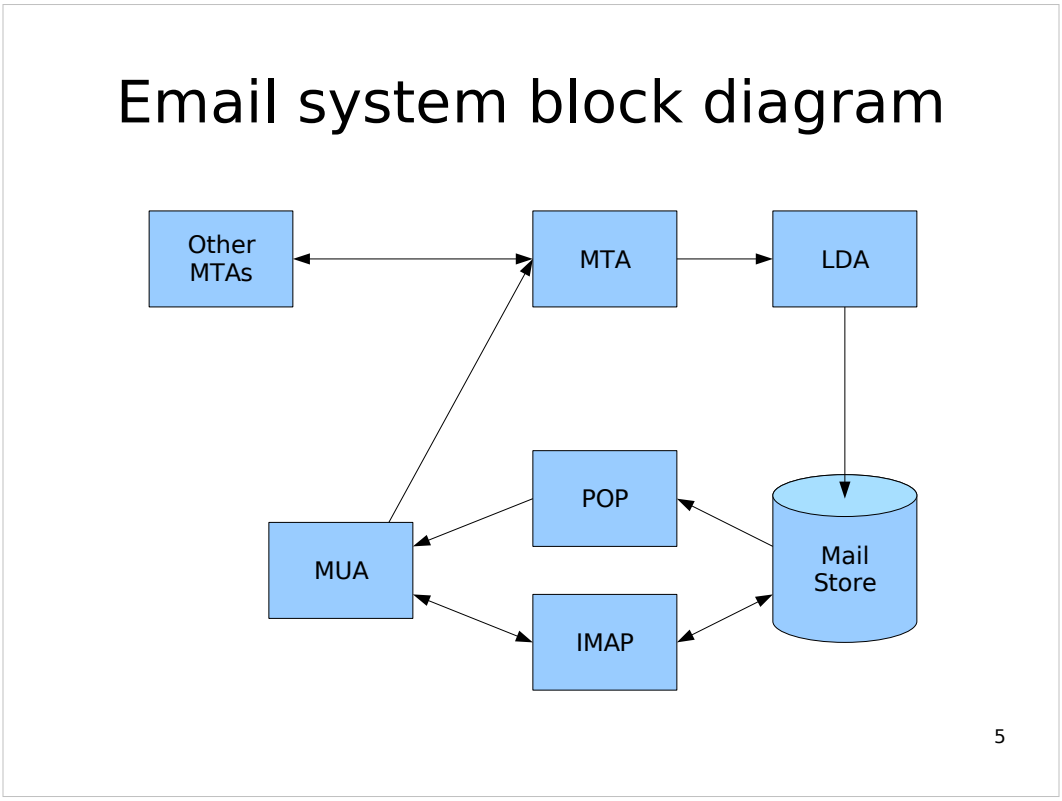
Goals of this tutorial

- Present a comprehensive overview of email systems (transmission, delivery, access)
- Discuss issues important to large email systems (efficiency, interoperability, abuse prevention)
- Discuss factors that affect email system performance and ability to handle growth

Overview

- Introduction: Email system architecture, components, and protocols
- Implementation considerations for components of large email systems
- Techniques for high availability and growth
- Case studies

Introduction: Mail System Architecture, Components, and Protocols



Key:

MTA: Mail Transfer Agent

LDA: Local Delivery Agent

POP: Post Office Protocol

IMAP: Internet Message Access Protocol

MUA: Mail User Agent

MTA (Mail Transfer Agent)

- Routes email based on recipient information
- Common MTA software: Exim, Postfix, Sendmail
- Mail destined for local users is passed to LDA (Local Delivery Agent)
- Mail to remote users is passed to another MTA using SMTP (“Simple” Mail Transfer Protocol)

6

Exim: http://www.exim.org
Postfix: http://www.postfix.org
Sendmail: http://www.sendmail.org

There are lots of other possibilities; I list these as the leading open-source options that are commonly used and still being developed.

If you're wondering why qmail isn't on the list, it's because it has significant licensing concerns, requires a large number of third-party patches to provide the same functionality as other options, and has some default behaviors (like accept-then-bounce) that are undesirable.

SMTP (Simple Mail Transfer Protocol)

- Defined in RFC 2821 (update of original RFC 821)
- Simple, text-based protocol
  - Four-letter commands (usually) with arguments (`HELO/EHLO`, `MAIL From:`, `RCPT To:`, `DATA`, `QUIT`)
  - Three-digit status codes in response to commands, with optional text comments
- Essentially the only network protocol for email transmission on the Internet

SMTP example

```
220 smtp.uoregon.edu ESMTP Sendmail 8.13.8/8.13.8; Wed, 14 Feb 2007 12:56:06 -0800
>>> EHLO hexadecimal.uoregon.edu
250-smtp.uoregon.edu Hello hexadecimal [128.223.142.97], pleased to meet you
250-ENHANCEDSTATUSCODES
250-PIPELINING
250-EXPN
250-VERB
250-8BITMIME
250-SIZE 10485760
250-DSN
250-ETRN
250-AUTH PLAIN LOGIN
250-DELIVERBY
250 HELP
>>> MAIL From:<stevev@hexadecimal.uoregon.edu> SIZE=342 AUTH=<>
250 2.1.0 <stevev@hexadecimal.uoregon.edu>... Sender ok
>>> RCPT To:<stevev@uoregon.edu>
250 2.1.5 <stevev@uoregon.edu>... Recipient ok
>>> DATA
354 Enter mail, end with "." on a line by itself
>>> From: stevev@hexadecimal.uoregon.edu
>>> To: stevev@uoregon.edu
>>> Subject: example
>>>
>>> .
250 2.0.0 l1EKu60m023771 Message accepted for delivery
>>> QUIT
221 2.0.0 smtp.uoregon.edu closing connection
```


Input to the session is prefixed with >>> and in bold.

EHLO is used so that the ESMTP capabilities will be listed.

Otherwise this is just a minimal example of how a message might be passed from one MTA to another.

Common SMTP commands

- EHL0 (Extended HeLO)
  - If SMTP greeting includes the substring ESMTP, EHL0 asks for ESMTP capabilties
- `MAIL From:<sender>`
  - Specify “envelope sender” (not necessarily same as From: header) used to route bounces
- `RCPT To:<recipient>`
  - Specifies recipient (may be used more than once)

In ESMTP MAIL From: and RCPT To: may also accept additional option specifications after the addresses, Many of these are used to request non-default DSN (Delivery Status Notification) behavior.

Common SMTP commands (cont)

- DATA
  - Begins transfer of actual message content
- QUIT
  - Completes SMTP transaction
- STARTTLS (after EHL0)
  - Negotiate SSL/TLS (session restarts)
- AUTH <method> (after EHL0)
  - Attempt user authentication with specified method

RSET will also reset the session to the state just after HELO/EHLO. Sendmail, at least, can also list its commands when given HELP and even let you view descriptions of its commands with HELP <command>.

SMTP responses

- General form (always starts with 3 digits)
  - `999 comment text`
- First digit is general status
  - 2: OK, action completed
  - 3: OK, continue
  - 4: Temporary error, try again later
  - 5: Permanent error, don't try again
- Remaining digits provide more detail; comment may start with 9.9.9 extended status code for even more detail

RFC 2821 lists the basic response codes for SMTP. RFC 1893 lists the extended response codes.

MTA vs. MSA SMTP

- MTA SMTP operates on TCP port 25
- MSA SMTP operates on TCP port 587
- MSA mode is specifically intended for user agent message submission
  - Can be configured with different behavior like required TLS encryption or authentication, or different acceptance and relaying rules
- Ideally port 25 should just be for inter-MTA traffic and user agents should use only port 587, but some still use port 25

Microsoft provides an SMTP-over-SSL service in Exchange on TCP port 465. (This is different from STARTTLS in that SSL negotiation is mandatory at session-open time, as in HTTPS.) Unfortunately they never actually registered TCP port 465 with the IANA and it is allocated to another somewhat obscure routing protocol URD, which also has the property that any router that uses URD intercepts any TCP port 465 traffic that passes through it.

Most MTAs will let you configure SMTP-over-SSL on port 465 but because of the behavior of URD some users may find it doesn't work if they're behind a URD-enabled router.

LDA (Local Delivery Agent)

- Delivers a message into the mail store for a specified user
- More sophisticated LDAs (such as procmail) can do user-configurable delivery to alternate locations (such as different folders, pipes, /dev/null, forwarding) based on inspection of message content

Traditional UNIX systems had a /bin/mail that was frequently used as the Sendmail LDA.

Sendmail also bundles a basic mail.local LDA (which also supports LMTP, an SMTP-like protocol for interacting with the LDA).

Postfix also bundles an LDA “local”.

Procmail has a really ugly configuration syntax but is very powerful and flexible. It also supports mbox, MH, and Maildir delivery so it is useful in variety of installations.

RFC 822 message format

- Internet email messages use a conventional format originally defined in RFC 822 (also used for other things like USENET, HTTP)
- Messages consist of headers and body
  - headers: `header-name: data`
  - multi-line headers indent lines with whitespace
  - body starts after blank line, free format (although lines should be <1000 chars)

Common headers

- `From:`
  - header sender, may not match real sender
- `To:`
  - header recipient, may not match real recipient
- `Subject:`
- `Date:`
- `Message-ID:`
  - (ideally) globally unique message identifier

Common headers (cont)

- `Received:`
  - each MTA which processes message inserts one with tracking information
- `Return-Path:`
  - original envelope sender
- `MIME-Version:, Content-type:`
  - MIME headers
- `X-*`
  - nonstandard application-specific headers

The Mail Store

- Message storage for users
- Each user at least has a primary inbox folder (default location for delivery and message retrieval) and may also have access to additional folders
- The mail store format is probably the biggest influence on performance of your mail system
- All components that interact with the mail store must use compatible locking

Mail access with POP

- POP (Post Office Protocol) is the original protocol for remote mail access (POP3 is latest protocol revision)
- Supports only a single inbox folder
- Each mail check requires login, scanning folder to index messages and identify new messages, logout
- Messages can be downloaded and (optionally) deleted

It's great (from the server administrator's point of view) if you can get your users to do the traditional download-and-delete behavior of POP but it's uncommon for people to do that any more, especially if they also want to use a webmail system (which likely uses IMAP) for alternate access to their mail.

POP session example

```
+OK Dovecot ready.
>>> user stevev
+OK
>>> pass password
+OK Logged in.
>>> stat
+OK 1 403
>>> retr 1
+OK 403 octets
Content-Type: text/plain; charset="us-ascii"
Content-Disposition: inline
Content-Transfer-Encoding: 7bit
MIME-Version: 1.0
X-Mailer: AlphaMail 1.0.22
Message-ID: <1164891521.43097.alphamail@mailapps1.uoregon.edu>
X-Originating-Ip: 71.236.255.164
From: "" <stevev@uoregon.edu>
Reply-To: "" <stevev@uoregon.edu>
To: stevev@uoregon.edu
Subject: test
Date: Thu, 30 Nov 2006 04:58:41 -0800

test
.
>>> quit
+OK Logging out.
```


This simple session illustrates retrieving (but not deleting!) a single new message.

DELE is the POP command for deleting messages.

Mail access with IMAP

- IMAP (Internet Message Access Protocol) was originally developed for the PINE mail client, but is now a predominant client access protocol
- Supports multiple folders
- Supports persistent connections
- Oriented towards leaving mail in the server repository for access from multiple locations and clients

IMAP session example

```
* OK Dovecot ready.
>>> 1 login stevev password
1 OK Logged in.
>>> 2 select inbox
* FLAGS (\Answered \Flagged \Deleted \Seen \Draft)
* OK [PERMANENTFLAGS (\Answered \Flagged \Deleted
\Seen \Draft \*)] Flags permitted.
* 1 EXISTS
* 0 RECENT
* OK [UIDVALIDITY 1157275920] UIDs valid
* OK [UIDNEXT 2] Predicted next UID
2 OK [READ-WRITE] Select completed.
>>> 3 logout
* BYE Logging out
3 OK Logout completed.
```


This is about as much IMAP as I know. The protocol is really quite complicated. Every command is prefixed with a tag that is included in server responses so a response can be associated with its command. The server is allowed to respond to commands asynchronously and out of order. There are also mechanisms for multi-line commands and responses and quoting, such as {n} to indicate n literal characters follow.

MUAs (Mail User Agents)

- Provide a user interface for viewing and composing mail messages
- Standalone MUA applications include Outlook, Thunderbird, Eudora, PINE, mutt, and many, many more
- Web-based email access has become very popular and common
- MUA behavior is often what causes your mail performance problems

As will be discussed later, in particular several common MUAs (Mac OS X Mail.app, Outlook) attempt multiple simultaneous sessions on the same folder.

Also, users who have large folders and who make frequent mail checks (once a minute or more frequently) can really hammer on your POP or IMAP server's I/O.

Implementation concerns for MTAs and SMTP

Mail relaying

- Mail relaying is the acceptance of messages from one network location for transfer to another network location (as opposed to origination of email or final delivery to a local recipient)
- Originally mail relaying was not usually restricted, but spammers exploited unrestricted relays and restriction of relaying has since become a necessity

Sendmail 8.9.0 (released in May 1998) was the first version that restricted mail relaying by default. Previous versions defaulted to unrestricted relaying and at least for Sendmail 8.8 there were various add-on rulesets to restrict relaying to prevent exploitation by spammers.

Relaying restrictions

- Domain of envelope sender
  - Risky since any spammer could choose a domain that let him relay
- Network address of client
  - Simple for clients that lived in the permitted ranges, but difficult for roaming clients
- SMTP Authentication
  - ESMTP AUTH could be used to authenticate to SMTP in much the same way clients authenticated for POP or IMAP

SMTP AUTH

- Clients can supply a username/password pair to the SMTP server, or enage in some challenge/response interaction, to obtain relaying privileges
- Methods commonly supported in clients (`AUTH PLAIN` and `AUTH LOGIN`) use base64-encoding for authentication data (not encrypted! not network-safe!)
- AUTH methods with encryption are not commonly supported

Most other AUTH methods besides PLAIN and LOGIN require maintaining a separate authentication database or hooking into infrastructure like Kerberos; PLAIN and LOGIN are typically implemented to work against standard UNIX passwords (or RADIUS or LDAP authentication).

STARTTLS

- STARTTLS provides optional SMTP session encryption based on SSL/TLS
- Clients have to support ESMTP and can issue STARTTLS if the ESMTP server advertises STARTTLS as a capability in response to EHL0
- Restarts SMTP session with encrypted traffic once established, so it prevents sniffing of SMTP protocol exchange and message data

Why optional? For interoperability; normal SSL/TLS starts with encryption negotiation before any application data is passed, so an MTA that didn't understand SSL/TLS would be unable to communicate with one that required it. STARTTLS as an ESMTP capability allows clients that want the functionality to request it while remaining interoperable with servers that don't support STARTTLS.

STARTTLS saves AUTH

- You'd like to use AUTH to allow users to authenticate for relaying, but then they'd leak their passwords
- Clients that support AUTH usually support STARTTLS too
- Set up your MSA (but not MTA) port to advertise and require AUTH after STARTTLS
- Roaming and remote users are happier, and have somewhat better privacy too

TLS && AUTH Mini-howto

- Sendmail
  - `define(`confAUTH_OPTIONS', `p')`
- Postfix
  - `smtpd_tls_auth_only = yes`
- Exim
  - `auth_advertise_hosts = ${if eq{$tls_cipher}{}{}{*}}`

These are just the critical options needed to ensure that insecure AUTH methods are offered only after TLS is in effect.

Spam

- Unsolicited Bulk Email
  - not requested by recipients, sent to multiple recipients
- Many sites report >90% of their incoming email is spam, so it becomes a big scaling problem for large sites
- You're stuck with trying to use technical means to solve a social problem
- There are no perfect solutions, and the imperfect ones keep changing

The definition “unsolicited bulk email” is intended to be content-neutral (as “commercial” would not be). “Bulk” means multiple recipients, although people are reluctant to define a specific numeric threshold (partly because if bulk means N, then you can be sure spammers would promptly send all messages in batches of N-1).

Because spam is a social problem, it's important to remember that there really is no “final solution”. You can mitigate spam, but you can't eliminate it.

Characteristics of spam

- Most is now coming from a huge pool of randomly-selected proxies, typically compromised Windows systems
- A lot of spam uses forged sender addresses
  - Any approach which might return spam to forged addresses should be avoided
- Some still comes from relatively fixed locations that are easier to block

One tactic of spammers who are still using fixed address ranges (temporarily) is to sign up with a hosting provider, then use a large set of randomly-generated domain names for their hosts and spread spamming activity out over the address space to reduce the apparent level of activity from any single address.

Considerations for spam mitigation

- Should be driven by user needs and organizational considerations
- Blocking vs. filtering
  - Blocking: refuse spam at SMTP time
  - Filtering: accept spam, but tag or divert it based on content
- Should allow user customization
  - Opt-in or opt-out for system-wide blocking
  - Filtering mechanisms are often customizable, but not always easily

Preventing spam from your site

- Have abuse@, postmaster@ contact addresses **that are not blocked** to ensure you can be notified of issues
- Prevent open relays, open proxies
- Watch for vulnerable web applications
  - Lots of PHP crud seems to be easily exploited for spam injection
  - Avoid formmail scripts that take addresses from form input
- Monitor message volumes, mail queues

Blocking: DNSBLs

- DNS BlackLists
  - Lists of IP addresses accessible by DNS lookups in special zones
- Maintainers have many listing policies
  - Known spam sources
  - Spammer-controlled networks
  - Dynamically-assigned (non-server) IPs
  - Examine policies, choose carefully!
- DNSBLs are about the only approach that reduces spam without adding server load

Example: When your mail server receives a client connection from the IP address 10.1.2.3 and is using the (hypothetical) diespammers.org DNS blacklist, it attempts to look up the DNS A record for the domain name 3.2.1.10.diespammers.org. This works in a way analogous to the .in-addr.arpa domain used for reverse DNS (IP-to-name) mapping.

Most DNS blacklists return A records for addresses in the range 127.0.0.0/8, sometimes coded to provide more information, such as the last octet being a code for the kind of listing.

Blocking: DNSBLs (cont)

- May need to use local access control features to override DNSBL listings (“whitelisting”)
- Some DNSBLs allow you to download copies of their zone data (perhaps for a fee) to reduce lookup latency
  - Zones can be very large (>10^6 entries) and take significant resources to serve
- Spammer use of zombies/proxies is largely a reaction to DNSBLs

If you lease dialup or DSL space from a provider who might be listed on a DNSBL, you would certainly want to whitelist the IPs or subdomain that belongs to you to prevent your users from being denied access to your own mail server.

Blocking: Local access control

- Supplement or override DNSBL coverage
- Can be labor-intensive to manage
- Local user opt-in or opt-out of blocking
  - Sendmail: `FEATURE(delay_checks)`, `Spam:user@` entries in access_db
  - Postfix: http://www.postfix.org/RESTRICTION_CLASS_README.html
  - Exim: `recipient_reject_except = <address list>` (may also refer to external file or database)

Blocking: “Greet-pause”

- Proper SMTP clients are supposed to wait for SMTP banner before proceeding
- Pause before displaying banner, reject clients that send data before banner
- 30-second limit imposed by RFCs
- Interferes with some real sites (like Gmail)
- Slows down processing of email
- Spammers are already adjusting for this

Blocking: Greylisting

- Spamware often doesn't implement full MTA behavior, like queuing
- Defer initial attempt to deliver with 4xx reply, accept on subsequent retry
- Requires keeping a database to remember connection attempts
- Successful clients can also be remembered and not delayed
- Slows down mail processing **a lot**, depends on queuing behavior of client

Content filtering

- Examine message contents for spam or malware indicators
  - Can be applied during DATA phase using Sendmail/Postfix milters
- Can be very resource-intensive
  - SpamAssassin uses more CPU than any other component in delivery process
- Adaptable and customizable
- Also easily worked around by spammers (image spam, random text)

Bayesian statistical classification had a surge of popularity as a way of doing “trainable” spam filtering. The basic idea is that statistical analysis of a body of messages that have been human-sorted into spam and non-spam categories can produce criteria that can be applied to automatic classification. If users are willing to invest the effort into training (and occasional retraining) they can often get good results. Bayesian filtering is usually ineffective to apply on a system-wide basis because a large population may have a large variation in their criteria for spam.

Spam mitigation methods you should avoid

- Challenge-response
  - Unknown senders sent back a challenge, have to respond to get message to recipient
  - Spam sender forgery means lots of innocent third parties get useless challenges
  - Even valid senders find it annoying
- SMTP Callbacks
  - Attempt to connect back to purported sender's SMTP server to verify sender
  - Once again, problematic because of forgery

A fundamental problem with applying either of these on a large scale is that they magnify the network traffic caused by spam, which is already the majority of email traffic, making a bad problem worse. Then directing that extra traffic to essentially uninvolved third parties just expands the spam problem even more.

I'm actually pretty lukewarm about things like greet-pause and greylisting but they don't magnify the problem of spam the way these two techniques do.

My personal favorite methods for spam mitigation

- Use an appropriate DNSBL to keep spam from getting in to your mail system
  - spamhaus.org is conservative and effective
- Provide content filtering for what gets in
  - ClamAV is very effective for virus filtering and some phishing spam, clamav-milter can reject at end of SMTP DATA phase to minimize backscatter
  - SpamAssassin seems to be leading spam filtering solution, with many content heuristics and trainable Bayesian filtering

At my site we use spamhaus.org and njabl.org. Spamhaus also provides different subzones:
sbl.spamhaus.org: spammers
xbl.spamhaus.org: zombies/proxies
pbl.spamhaus.org: dynamic IPs that shouldn't be originating mail (as indicated by provider policies)
zen.spamhaus.org: all of the above
njabl.org has similar subzones but we use combined.njabl.org

ClamAV: http://www.clamav.net

SpamAssassin: http://spamassassin.apache.org

What if your site is blocked?

- Many DNSBLs provide information about their listings with evidence
- Remote sites that are blocking you will hopefully tell you why
- Make sure you fix any spam problems that caused blocking
- Other things that might cause blocking:
  - nonexistent or inconsistent rDNS
  - RFC-compliance issues

Example: Spamhaus provides listing data you can look up by IP address:
http://www.spamhaus.org/query/bl?ip=A.B.C.D

This assumes you're dealing with a DNSBL maintainer or remote site that wants to be reasonable, but unfortunately that's not always the case. Being polite and dealing with any real spam problems that you might be having are sometimes all you can do.

Nonexistent reverse DNS (rDNS) means not having PTR records that can be looked up from IP addresses.

Inconsistent rDNS means getting a PTR that can't be looked up to get the original A.

Mail queuing

- Email is designed to be store-and-forward, to handle temporary delivery problems
- Mail that can't be delivered immediately is placed in a queue for later retries
- Large sites can carry large queues
- Large site or network outages can greatly increase your queue

I'm currently only familiar with Sendmail's queue system, which uses files in /var/spool/mqueue:

qfXXXXXX: queue control file (has headers, other data used by Sendmail for queue management)
dfXXXXXX: data file containing message body
tfXXXXXX, xfXXXXXX: temporary files used in queue processing to lock entries, hold working data

Handling large mail queues

- Typical queue defaults hold messages for five days before giving up
  - You could reduce this a little
- Most MTAs are configurable for multiple queues with different policies
  - Simple policy retries at fixed intervals
  - Muliple queues can implement different intervals or an exponential-backoff policy
- Queue items are usually text files that can be managed by hand or with scripts

Something I find myself doing in Sendmail once in a while:

```
grep -l something qf* |
sed 's/qf\(.*\)/qf\1 df\1/' |
while read qf df
        do mv $qf $df /var/tmp/crap
done
```

This selects Sendmail queue files that contain a substring “something”, then moves all the associated queue files to a temporary directory. This can get rid of spam/mailbomb items piled up in your queue, but give you a chance to look over the items before discarding them.

Mailing list management

- Use mailing list management software that provides for subscription confirmation and user subscription management
- Mailing lists generate almost entirely outbound traffic; you may want to use a separate MTA for the mailing list tuned to its traffic patterns

A couple of common mailing list managers: Majordomo (http://www.greatcircle.com/majordmo/), Mailman (http://www.gnu.org/software/mailman)

Subscription confirmation (also called “confirmed opt-in”) typically means returning a mail message in reply to a subscription request containing a confirmation token which is returned by email or accessing a web page.

You can improve mailing list delivery performance by reducing the number of recipients per queue item to increase delivery parallelization (but not too much).

Refuse-or-deliver philosophy

- Messages should be accepted for delivery, or refused at SMTP time
- Other dispositions cause problems
  - Discarding silently leaves a sender with the impression the mail was delivered when it wasn't
  - Returning a non-delivery notification after acceptance can send messages to forged addresses (“backscatter”) or cause loops
- When a message isn't accepted, a valid sender can find out

Specifically this is about system-wide policy. Users may have valid reasons for discarding their own mail (although they should understand the risks) but if so the choice and responsibility should be theirs.

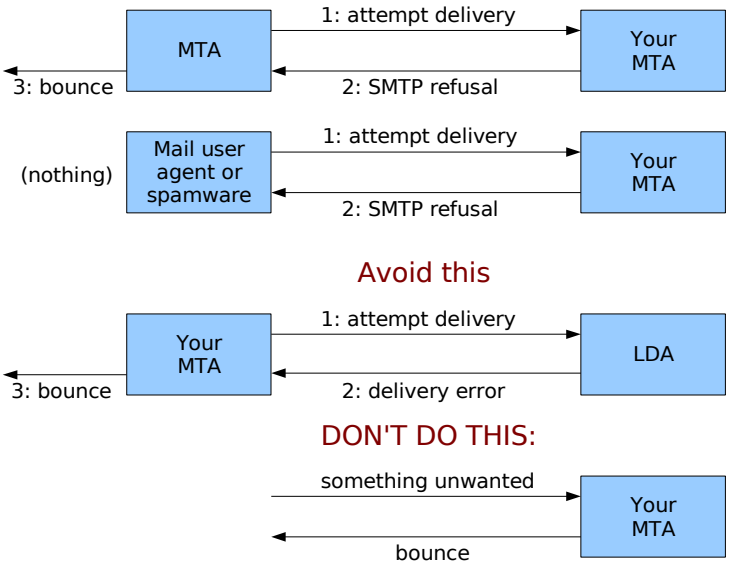
Bounce messages

- Non-delivery notifications (“bounce messages”) are returned by MTAs to a sender when a message is undeliverable
- Real users who are using an MTA can get valid bounces
- Spamware/malware in communication with an MTA tend not to generate a bounce when they fail to deliver
  - Refusing at SMTP time avoids “backscatter” of delivery notifications to forged senders

The important distinction here is that bounces are generated by MTAs in response to a delivery error; your MTA's refusal of a message won't cause a bounce unless a remote MTA is in communication with yours.

Bounces are supposed to be sent from the null sender (MAIL From:<>) to the original envelope sender (the original MAIL From: address is used for the RCPT To:) which indicates that no further bounce should be generated if this bounce is undeliverable. Often these double-bounces are delivered to the postmaster@ alias.

Bounce scenarios



Another way to look at this is that for the most part you should return bounces only to your own users, and if you refuse mail from another MTA then you should leave the generation of any bounce message to it.

Avoiding undesirable bounces

- Refuse unwanted messages during SMTP with 550 status
  - Blocked sender, unknown/invalid user at RCPT To: (so recipients can opt out of blocking)
  - Spam/virus content at end of DATA
- Try to avoid local delivery errors that cause your MTA to send bounces
- Spam filtering or virus scanning should not return messages or send notifications to (usually forged) senders

It can be somewhat difficult to prevent bounces due to local delivery issues like quotas or permission problems on mail files. Ideally you'd like the MTA to return an appropriate 4xx or 5xx reply to a remote client, but the traditional separation between MTA and LDA functions and the limited interface between them means that usually the LDA is the only component that can see those issues, but not be able to communicate much more than a delivery success/failure status to the MTA, which will typically generate a bounce on LDA failure.

Mail forwarding

- Users often want to forward mail
  - funnel mail from multiple accounts to one
  - get mail in old account to new account
- Most MTAs support user-controlled forwarding
  - .forward file checked for forward address
- Some LDAs also do forwarding
  - procmail: `!forward@address.com`

Mail forwarding issues

- .forward-style forwarding is usually safer
  - MTAs apply null-sender checking for bounces, Received: header thresholds to prevent indefinite looping
- LDA forwarding can be dangerous
  - LDA forwarding to a non-working destination may create a mail loop, especially if no attempt is made to match that condition
- SPF vs. traditional forwarding
  - same-sender forwarding blocked by SPF

SPF is an attempt to provide relatively simple verification that a mail server is authorized to send mail for a domain via auxiliary DNS records that indicate which subdomains/IPs should be sending mail for that domain.

If someone forwards mail through your site that originated from a domain with SPF records and is destined for a site that checks SPF, the forwarded message is rejected because mail with a sender in that domain is coming through your mail server which is not on their SPF list.

This is one of SPF's biggest problems.

Implementation concerns for LDAs

- Good to have one with configurable delivery behavior, especially for content filtering
  - Users can also do their own mail sorting
  - I like procmail, although it is ugly to configure
- Quotas
  - Good: prevents one mailbombed user from hosing everyone else
  - Bad: causes bounces and user confusion for over-quota users

Mostly procmail is terse.

```
:0:
^From:.*bob@mainframe.net
bobsmail
```

which means if the From: header contains "bob@mainframe.net" then put it in a folder named "bobsmail". Still, you can see why non-technical people find this a bit daunting.

It's also nice to have a local delivery agent that can be configured for whatever message store format you want to use.

Mail Store Implementations

“mbox” Mail Store

- Earliest and most common mail store format
- One file contains multiple messages separated by “From_” lines
  - `From stevev@uoregon.edu  Sun Feb 18 18:00:00 2007`
- Standard inbox location for users is /var/mail/$USER, but can be changed to spread those over more directories/disks
- Additional folders normally placed in user's home directory

mbox locking

- Changes to mbox files must be serialized and exclusive to avoid corruption
- Exclusion locking traditionally done with “dot-lock” files
  - Create temp file with random unique name
  - Attempt to link temp file to folder name with “.lock” appended (/var/mail/stevev.lock)
  - Remove temp file
- OS-level file locking (`fcntl()`, `flock()`) can also be used in addition to dot-locks

Almost everything does dot-locking, which can also be NFS-safe. The system-call-based methods might be used to supplement that. fcntl() is probably the other method you might want to use since it can also work via NFS and is more standardized than flock().

mbox problems

- Large folders become unwieldy to handle
  - Getting a folder index requires reading and parsing the entire folder
  - Updates to delete messages, modify headers, etc. usually involve complete rewriting of folder file
- Lock contention becomes problematic (clients fight with delivery, or high-rate delivery fights with itself) because some updates lock a folder for a long time

MUAs can also interact badly with locking, especially if they try to open the same folder multiple times; they either stall on their own attempts to issue multiple operations, or even worse fail when the POP/IMAP server returns an error because of locking.

Maildir Mail Store

- Store individual messages in individual files in a subdirectory
- Standard Maildir format uses three subdirectories in `$HOME/Maildir`:
  - `cur/` holds current folder contents
  - `new/` holds newly-delivered mail
  - `tmp/` used for message delivery and deletion
- Maildir++ supports multiple folders
  - Subdirectories like `$HOME/Maildir/.folder`
  - Path flattening: `sub/folder => .sub.folder`

Observed path-flattening behavior of a Maildir++ POP/IMAP server:

```
tr '.' '_'
tr '/' '.'
prepend '.'
```

So a folder name “foo/bar.baz/quux” turns into a subdirectory named “.foo.bar_baz.quux”

Maildir Mail Store (cont)

- Delivery is lock-free, based on unique filenames
  - `1171963019.6003_0.mserv5`
  - (UNIX time).(PID, serial).(hostname)
- A new message file is initially created in `tmp/`, then moved to `new/` when it is fully written out
- Access renames message files from `new/` to `cur/` appending status flags to name
  - `1171936019.6003_0.mserv5:2,S`

Maildir locking

- Almost unnecessary, although some servers implement folder-level locking for extra safety
- Done at folder level when updates are done, to ensure other clients get a consistent view of folder contents

Maildir and large folders

- Maildir can do much better than mbox in many cases, if underlying OS and filesystem is good at handling lots of small files and fast at directory traversal and open/read/close transactions
- Many benefits come from many kinds of updates being faster file link/unlink or renaming, rather than slow rewriting of large files
- Tends to interact better with NFS

Personally I have found that NetApp NFS does well with Maildir.

Linux ext3fs created with dir_index might be OK for less intensive setups, but I haven't benchmarked it. Whatever FS you use it should be efficient at handling very large directories and lots of relatively small files.

You will at least want to create a filesystem with a much higher number of inodes than the default. In our system we found that the global average message size was about 32K, so with 4K blocks we created filesystems with an 8::1 block::inode ratio.

Database Mail Store

- Few current UNIX systems implement these
- Advantages from databases:
  - Indexing and access can be very fast
  - Can save space by linking a single copy of a message to multiple recipients
  - Databases tend to have solid record-locking primitives

A USENIX paper benchmarked IMAP servers against different message store formats:

http://www.usenix.org/events/lisa03/tech/full_papers/elprin/elprin_html/index.html

They used UW-IMAP (mbox), Courier IMAP (Maildir), Cyrus (BerkeleyDB) and a model MySQL-based mail server.

Database Mail Store (cont)

- Disadvantages from databases
  - Data structures are complex and fragile; damage can cause widespread data loss or corruption
  - Require more specialized tools for backup and message manipulation, compared to mbox or Maildir which can use more basic file manipulation tools
- Still more of a research topic than a common solution, although performance of trial implementations is encouraging

Putting a Mail Store on NFS

- Delivering mail into NFS was traditionally avoided
  - Tended to be slower and higher-latency than direct-attached storage
  - Less stable than direct-attached storage
  - Locking was unreliable
- NFS got better
  - Network speed increasing faster than disk transfer speed
  - Locking somewhat better, and avoidable

Putting a Mail Store on NFS (cont)

- NFS is about the only storage technology that allows multiple hosts to access the same storage concurrently, allowing parallelization of SMTP, POP, IMAP servers working on a single mail store
- NFS “toasters” have become fast and reliable storage devices
- Most really large mail systems have gone to NFS

POP/IMAP implementation

- Traditional POP and IMAP pass authentication data in the clear
- Optional TLS commands or secure authentication methods are not widely supported in clients
- Standard service ports with required TLS are well-standardized
  - pop3s = TCP port 993, imaps = TCP port 995
- Client support for required TLS is widely available now, so you should require it

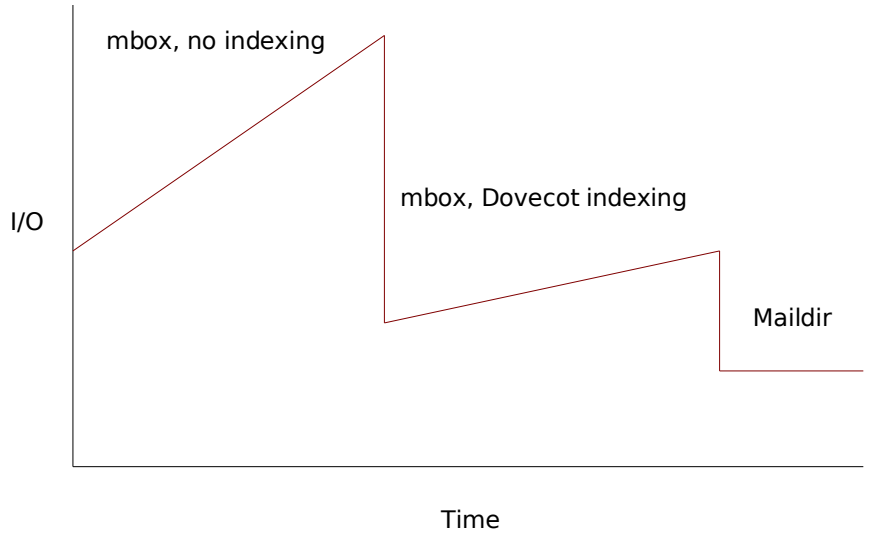
POP/IMAP performance

- POP is transactional (log in, read inbox to index and check for new messages, download/delete messages, log out)
- IMAP can be persistent (log in, do whatever, hang out, do some more, etc.) making it somewhat less I/O intensive
- Large folders still tend to be a problem
  - mbox: lots of file reading
  - Maildir: lots of readdir/open/read/close transactions

POP/IMAP performance (cont)

- Some POP/IMAP daemons support index caching to speed up indexing phase
  - UW IMAP: special “mbx” format that stores index data at beginning of mbox-like folder
  - Dovecot: supports auxiliary index cache files that store index data for both mbox and Maildir folders
- Index caching helps both mbox and Maildir perform better by eliminating unnecessary folder rescanning

POP/IMAP performance (cont)



68

This graph is mostly schematic, but is based on our experience with the different configurations shown.

Dovecot's indexing made mbox far more tolerable but still showed noticeable linear growth.

Maildir actually has expected linear growth behavior but it was hard to show that in the graph. Growth in client access rates and mail volume would also affect overall I/O.

IMAP shared access

- More and more, people want to access mail from multiple locations
- This often results in multiple clients making overlapping accesses to the same folders and “lockfighting”
- Some clients open multiple sessions on the *same folder*!
- Maildir is about the only thing that helps avoid problems that result from these behaviors, but not completely

Web email clients

- Popular for ease of use and flexibility of access
- Generally like other IMAP client, but often more resource-intensive
  - Lots of quick login/<single command>/logout transactions caused by each web page view
- Load on your IMAP servers can be reduced with an IMAP proxy in front of web email system, or web email system with integrated IMAP session caching

Techniques for high availability and growth

Backup MX hosts

- Traditional method for providing higher reliability for mail transfer
- DNS has a special MX resource record indicating cost and intended SMTP host
- SMTP protocol says to try multiple MXes for a domain in order from lowest to highest cost (or pick at random from those with same cost) until success
- MTAs (but not all MUAs) can be directed to alternate servers when one is down

Problems with backup MXes

- All MXes for a domain need to be configured with exactly the same SMTP-time behavior (blocking, known users) or they can be used to inject spam or generate backscatter
- MTAs have to time out (delays of minutes) trying to contact a nonworking MX before trying the next
- Having equal-priority MXes doesn't guarantee fair round-robin behavior

Spammers have actually been observed to try MXes in the opposite order, because many backup MXes don't have the same spam filtering as the primaries. Since the primary will accept mail from the backup MX, this bypasses many SMTP-time blocking methods.

A backup MX that queues mail for the primary, but can't reject unknown users at SMTP time, will generate bounces when the primary rejects unknown users.

Load-balancing SMTP

- Load-balancing works well with SMTP due to relatively short transactional nature of SMTP sessions
- Load-balancer can detect and remove a nonfunctional SMTP server from use faster than MTAs time out or DNS updates propagate
- Reduces visible downtime to MUAs
- Load-balancing policy is completely under your control

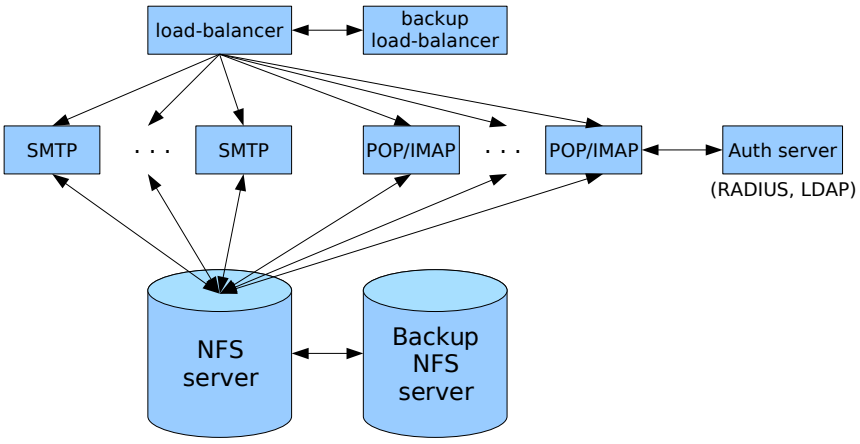
Issues with load-balancing SMTP

- You still have to ensure consistent SMTP-time behavior across all servers
  - Configuration management tools to automate and synchronize updates help here
- Mail queue items distributed across multiple hosts
  - If a host is lost, you might also lose its queue
  - Sharing queue across hosts is problematic
- Centralized log collection

Load-balancing POP and IMAP

- Main concern is reliable exclusion locking
  - User sessions often distributed across multiple hosts in server pool
  - Maildir helps by removing most need for locking and reducing lock durations
- If you're using NFS (you almost have to):
  - `noac` (no attribute caching) mount option ensures NFS clients see consistent file states
  - `fcntl()` locking tends to be more NFS-safe
  - Do same things on SMTP servers

Load-balancing high-availability architecture example



77

The load balancer here is directing TCP sessions for SMTP (25), MSA (587), POP-over-SSL (995), and IMAP-over-SSL (993). Having a backup in a failover configuration avoids making load-balancing a single point of failure.

The auth server actually communicates with all of the SMTP and POP/IMAP servers but to avoid clutter I didn't draw in all those arrows.

We are able to afford a backup NFS server for failover which is actively synchronized from the primary. Although we don't have active failover yet; if the primary fails we'd have to scramble to re-point everything to the backup.

Techniques for growth

- Load-balancing of SMTP, POP, IMAP for parallelization (not just availability)
- Distribute mail store across multiple storage backends
  - /home1 on nfs1, /home2 on nfs2, etc.
- LDAP (also set up for high-availability) for centralized authentication, fields to handle user mail routing and service routing

Case studies

“Brownouts”

- Older, traditional mail system with mbox-format inboxes in `/var/mail/user`
- ~15,000 user accounts
- System “browns out” at midday on busy days; delivery and access become slow
- I/O on /var/mail disk actually *goes down* during brownouts
- System recovers when demand falls off

“Brownouts” (cont)

- Why? Dot-locking in single directory becomes bottlenecked on OS serialization of directory updates
- Solution: Relocated inboxes to home directories (`~user/.mail`)
  - Split I/O across multiple home directory disks, increasing all performance
  - Users no longer contend with everyone else for inbox locks
  - Did require coordinated reconfiguration of LDA, POP, IMAP, UNIX shell MUAs

Maildir conversion

- Biggest, scariest system administration project I've ever been involved with
  - 40,000 users, 50 million messages, 1.6 TB, lots of unhappy people if done badly
- Motivated by chronic problems with mbox performance and lock contention issues
  - Users with >100MB folders
  - MUAs opening multiple sessions on folders and fighting with themselves
  - Stale NFS lock issues

Maildir conversion goals

- Goal: All messages accessible by POP or IMAP should remain accessible after conversion
  - Subgoal: POP and IMAP become only supported access methods after conversion
- How do you find all that mail when it's scattered all over home directories?
  - We were lucky and clever: After previously converting to Dovecot with mbox indexing, index files could be used to find accessed folders

The ulterior motive behind making POP and IMAP be the only supported access methods is that they put an abstraction layer in front of the mail store, so a later change in store format can be accomplished with less disruption.

I'm not sure what we would have done without the Dovecot index hack. We found that although most people had segregated mail folders into a ~/mail subdirectory, some had not, and some had done weird things (like using “-/mail”).

Maildir conversion: how to turn mbox into Maildir?

- mb2md
  - Perl script written by people who did similar conversion
    - http://batleth.sapienti-sat.org/projects/mb2md
  - Splits mbox folder out into Maildir, including parsing headers for message status flags
  - Can also process all mbox folders in a subdirectory
- Chose Maildir++ layout, installed test POP/IMAP daemons set up for Maildir, converted some willing ~~victims~~ users

Per-user Maildir conversion

- Always convert `~user/.mail` (standard home directory inbox)
- Always convert standard `~user/mail` folder directory
- Find Dovecot-created `.imap` index directories containing index files, convert corresponding folder for each index file
  - Many users had folders outside recommended `~user/mail` directory
- Clean up: remove converted mboxes

Maildir conversion: outage planning

- For maximum safety we wanted to avoid changes to stored messages during conversion, but this meant disabling mail services for however long it took
- Ran benchmarks by converting existing mail to scratch location (also validated automated conversion methodology)
- Benchmarks showed some benefits from parallelization, confident of <2 days conversion time (actually took ~26 hours)

Maildir conversion: the big day arrives

- Raised `maxfiles` setting in NetApp file server to accommodate Maildir
  - trial conversions showed average message size of 32 kB, used to set global space::files ratio
- Turn off POP and IMAP servers
- Hide procmail from sendmail
  - Sendmail leaves messages destined for local users in queue if it can't exec LDA
  - Also raised Timeout.queuewarn to suppress “not delivered in 4 hours” warnings

Maildir conversion: the big day arrives (cont)

- Also create snapshots of home directory volumes in case of backout
- Disable user quotas (conversion temporarily more than doubles space usage for a user)
- Run per-user conversion script on a few more test users and validate carefully
- Fire off batch conversions spread over 8 hosts
- Wait . . .

Maildir conversion: post-conversion

- Turn on quotas with somewhat modified quota limits
  - 25% space increase for fragmentation, greatly increased file quotas
- Re-enable procmail configured for Maildir delivery
  - `DEFAULT=$HOME/Maildir/`
- Flush bulging mail queues
- Re-enable POP/IMAP with Maildir configuration

Maildir conversion: interesting problems

- Manual conversion/cleanup for people with odd or nonstandard configurations
  - procmail sorting to Maildir++ folders instead of mbox folders
  - Mail that hadn't been converted because it hadn't been accessed/indexed by Dovecot
- Nasty e1000 driver bug tickled by new NFS traffic patterns with Maildir
  - Interfaces on POP/IMAP servers would shut down due to packet rate and memory stress
  - Ultimately had to install locally-built driver

Maildir conversion: the aftermath

- Goal reached: 99+% of users noticed no difference after conversion
- Really did eliminate issues with lock contention and NFS
- Performance is mainly better, but I/O load on NFS server turned into CPU load from higher rate of NFS requests