filetsf: A File Transfer System Based on lpr/lpd

John Sellens

Data Processing, University of Waterloo

jmsellens@uwaterloo.ca

USENIX LISA IX

September 22, 1995

Quick Introduction – What?

- Filetsf is a file transfer system
  - transfer, not replication
- Transfers files between cooperating machines
- Allows ad-hoc transfers between arbitrary users, without passwords or `.rhosts`
- Spooling, queueing, ordering of transfers

Quick Introduction – Why?

- We are a Data Processing department, with multiple, independent machines of various flavours
  - AIX, Solaris, SunOS, Ultrix, OSF/1, VM/CMS, ...
- We need to be able to transfer data between machines
  - e.g. course registrations, student records, financial data, ...
- Want to have a “reasonably” secure method of file transfer
- Want to avoid requiring operator intervention

Quick Introduction – How?

- Based on the LPD protocol and lpr/lpd
- Pairs of machines are configured to agree to allow transfers
  - One print queue for each destination, one to receive files
- *sendfile* command is a wrapper for *lpr*
- Files are spooled on destination with a print filter
- *acceptfile* retrieves files from the spool directory

Previous Approaches and History

- Magnetic tapes of various flavours
- FTP to a common account
  - Everyone knew the password, which led to inappropriate interactive use
  - Old trash accumulated since it was hard to control
  - Hard to identify the sender or recipient of a file
  - Potential for name clashes
- Some use of *rdist* and *rcp* – some of the same problems

Approaches Considered

- Software replication is not the same thing as file transfer
- FTP, *rdist*, *rcp*, NFS were all unsuitable
- SMTP mail and MIME weren't appropriate
- We had planned to construct a full-featured system, with queueing, spooling, etc. – we gave up
- Finally thought of using lpr/lpd
  - Queueing, ordering, sender identification, access control
  - Available everywhere, easy to implement

Implementation Overview

- Filetsf has 3 main commands (6 in total)
- *sendfile* – submits files one at a time to *lpr*
  – recipient name and options are encoded in -C class option
  – print queues are named for the destination machine
- *tsfif* – lpr/lpd print filter that runs on destination machine
  – deposits files in filetsf spool directory using naming convention to allow multiple versions of files, etc.
- *acceptfile* – retrieves or lists files from spool directory
- More details are available in the paper

Examples of Use

- Send a file to the same userid on a different host
  `% sendfile -h otherhost file`
- Send the current version of the `hosts` file
  `% sendfile -h otherhost -r \`
  `-u root /etc/hosts`
- Retrieve `file` sent by user foo@bar into `/tmp/blort`
  `% acceptfile -s foo@bar \`
  `-r /tmp/blort file`

Security and Reliability

- Relies on "acceptable" level of security provided by lpr/lpd
- Systems have mail and FTP, so we ignore the risk of sending private data
- Our systems are on private subnets, and our lpr/lpd has host-based access control
- *acceptfile* is the only privileged program
- Not intended to provide a high level of security
- Could build higher security or reliability on top of the basic tool

Good Things About Filetsf

- Simple to implement, which implies it's a good approach
- Appears to be a good framework that could be extended if needed or appropriate
- i.e. Conforms to the "UNIX Philosophy"
- Works on AIX and VM/CMS (sending only so far)
- Has queueing, spooling and ordering; handles network and/or system interruptions
- No esoteric tools needed to build, just *cc* and *sh*

Conclusions and Availability

- Seems to be a good, basic tool, that does the job
- It's better than what we had been doing
- Available via anonymous FTP from `math.uwaterloo.ca` as `/pub/filetsf/filetsf.tar.Z`
- Unsupported software, but fixes and enhancements gratefully accepted