Advanced Topics in NNTP and INN

David Lawrence <tale@uu.net>
UUNET Technologies

Paul Vixie <paul@vix.com>
Internet Software Consortium

Tutorial Overview

- Keeping up with the volume of news.
- Managing the volume of news.
- Handling `nnrpd` in its many uses.
- Running a mail and news gateway.
- Supporting moderators at your site.
- Looking toward the future.

The News on the Volume of News.

- The bad news — Usenet is huge, ahead of its historic growth curve.
- The better news — Technology is always improving.
- The best news — the worst of the growth is very localized.

Falling Behind on your Incoming Feed

- Monitoring the age of incoming articles:

```
tail -1000 $N/history | \
  awk '{print $2}' | \
  awk -F~ '\
    {$delay += $1 - $3} \
    END {print $delay / NR / 3600} \
  '
```

- *There is no overall definition of behindness.*

Falling Behind (2)

- Best ways to tell if you are behind your upstream site:
  - Your feed tells you that you are.
  - Your incoming volume doesn't dip at all on weekends.
  - Time it yourself; feed an article to your upstream server that should come back to you.

- Problem could be on either end or in between.

How Fast is Your Server?

- Build a huge batchfile; 10000 articles is good.
- Ensure the articles will not be propagated.
- Feed the batch with `rnews`.
- Summarize:

```
tail -500 /var/log/news/news | \
  cut -c1-15 | \
  uniq -c | tee /dev/tty | \
  awk '{i+=$1} END {print i/NR}'
```

- `innd` does better with a few feeds.

How Fat is Your Pipe?

- Measuring your upstream links:

```
traceroute upstreamhost 1024
ping -q -c 100 -s 1024 upstreamhost
ping -s upstreamhost 1024 100
```

- Theoretical speed of your pipe is a clear cap.
- How accessible is your server?

Incoming Machine Slowness

- Article writing too slow.
- History access too slow.
- Not enough RAM.
- Not enough CPU.

`history.dir` Innards

- Check first line of `history.dir`:

```
dbz 3 3486767 9 C 8 7 28 4 3 2 1 0
```

- First number is how much of the history file is kept in memory.
- Sixth number (28 here) is key to how big history file can get before things start becoming less efficient.
- Sixth number will not decrease if your history file shrinks notably.

Incoming Network Delays

- Reduce latency issues.
- Get a bigger pipe.
- Get a cleaner pipe.
- Get closer to the source.
- Reduce competition on the link.

Independent Solutions

- `alt.*` is more than half of a “full feed.”
- `alt.binaries.*` is more than half of *that*.
- GET A SMALLER FEED!

Group Selection

- Take only what people at your site read.
- alt.binaries groups are extremely popular.
- Cut off groups at your feed, not in `active`.

Expiration

- Time-based expire not suitable for many sites.
- Space-based expiration works well.
- Good: Run expire when low on space.
- Better: keep lists of size and ages.
- `ftp://ftp.ee.lbl.gov/dexpire.tar.Z`
- Putting `alt.binaries.*` on its own partition caps it.

Reader Processes

- Not all readers are created equal.
- Decide what you want to support.
- Implement your decisions.

Non-interactive Readers

- `NEWNEWS` method.
- `LIST ACTIVE` method.
- Both are hard to provision for.
- Neither is likely to benefit from improvements in article distribution technology.

Discouraging Pull Feeds

- Disable `NEWNEWS` in `nnrpd/nnrpd.c`
- At least stop distribution matching in `nnrpd/newnews.c`
- Slow down `ARTICLE` in `nnrpd/commands.c:CMDfetch()`

Problematic Interactive Readers

- `trn` uses HEAD and BODY commands.
- Older NETSCAPE™ used one connection per article.
- Newer NETSCAPE™ doesn't close connection.
- Other readers can cause regular traffic with no user action.

nnrp.access

- `inn1.4unoff4` allows more control than `inn1.4sec`.
- In `1.4sec`, authentication only needed for `POST`.
- Also in `1.4sec`, groups field doesn't apply to `POST`.
- In `unoff4`, most reader commands require `AUTHINFO`.
- Groups in `POST` must match accessible groups in `unoff4`.

AUTHINFO

- Many readers don't support `AUTHINFO`.
- Many of those that do don't support `generic`.
- Best access control method is still by site.

Limiting Readers

- Doing it in `innd:RChandoff()` saves `fork()`s.
- UUNET does it by associating customers with host address ranges.
- Maximum reader count is kept in running `innd`.
- Rejected if count is zero or can't be identified.
- `innd` watches for `nnrpd` exit, increments count.
- Same technique can be used to limit total readers for all sites.

Running a Gateway with `newsgate`

- Get it from `http://www.isc.org/isc/`.
- Compiling and installing it is straightforward.
- Be polite: check with listowner before gatewaying.
- Configuration is simple, but experience is needed.

Basics of Moderation

- Archive at `ftp://ftp.uu.net/networking/news/moderating`.
- Moderated groups are marked `m` in `active` file.
- Server sends posting only to first moderated group it finds.
- Looks for address in `moderators`, then `inn.conf`.
- Moderator receives as mail, massages into news article, posts.

Supporting a Moderator

- You don't *have* to do anything.
- Enable a few mail aliases.
- Allow `rnews` access.
- Install useful software: `perl`, `procmail`.
- Offer your expertise.

Helping off-site moderators.

- Addressing propagation issues.
- Don’t maintain own `moderators` list for non-local groups.

The Future of INN

- Embedded TCL — due in 1.5, any day now.
- Different article storage methods.
- A higher level API.
- Multicast propagation.

TCL Hooks

- TCL support must be compiled in.
- TCL article functions can be dynamically reloaded.
- Two script files loaded at startup: `startup.tcl` and `filter.tcl`.
- Must define:

```
proc filter_news
  proc filter_before_reload
  proc filter_after_reload
```

- First function is used on every article, can accept or reject article.
- Others bound loading of `filter.pl` for initialization.
- Filtering can be turned on or off via `ctlinnd`.

TCL Article Processing

- TCL function is called after the article is transferred but before it is accepted.
- Headers available in array named Headers.
- Body available in variable named Body.
- Function can return string accept to take article.
- Function can return any other string to reject article.

Simple SPAM™ Catcher

```
proc filter_news {} {
  set j 0
  set sum [checksum_article]
  puts $o "$Headers(Message-ID) $sum"
  set newsgroups [split $Headers(Newsgroups) ,]
  set subj [split $Headers(Subject) ,]
  foreach i $newsgroups {  incr j  }
  if {[string match "*make*money*fast*" $subj]} {
     return "dont like make.money.fast"
  }
  if {[string match "*cash$*" $subj]} {
     return "dont like Need cash$$$"
  }
  if {$j > 20} {
     return "dont like massive crossposts"
  }
  return "accept"
}
```