INTERNET SOFTWARE CONSORTIUM

Managing IP Networks with Free Software

Joe Abley jabley@isc.org

Stephen Stuart stuart@isc.org

Why use Free Software?

- Free software can give you room to experiment
  - Figure out what your requirements are without guessing
  - Answer questions quickly
  - Inefficiently-gathered answers to ad-hoc questions are better than no answers
- Insert other usual foam-at-mouth reasons here

What do you know today?

- Network topology
  - Half-erased diagrams on whiteboards
  - “Ask Leo, I think he knows”
  - traceroute
- Traffic
  - MRTG
  - “show interface”
- Configuration Changes
  - syslog (maybe)
  - tac-plus logs (maybe)

Who here can say with absolute confidence that there are no undefined route-maps on any router in their network?

Do you *know*, or are you guessing?

Grand Unified Network Management Tool

- Various vendors sell this tool already!
  - ho ho
- Many developers have already started trying to write this tool
  - some have even not yet stopped trying.
  - scheduled for release in `$(date -v+1y)`
- Fortunately, we don't actually need this tool
  - focus on making things incrementally better, and real progress can be made
  - small, well-defined problems are easy to solve

General Architecture

- Data in
  - Router configurations, interface counters, a list of customers, a table of things to ping
- Do Something
  - Store stuff, compare stuff, ping stuff, pipe stuff through awk
- Data out
  - Draw a map, generate a zone file, send mail

Ad-Hoc Modularity

- Ad-hack, maybe
- Scripts breed
  - Output of one ad-hoc script might provide an opportunity to write another ad-hoc script
  - Script food pyramid
- Look after your ecosystem
  - The top carnivore will get sad if all the cows stop mooing

Choice of Tool, Language

- Focus on results
  - Use whatever makes you the most productive
  - Don’t get distracted by linguistic esoterica or architectural grand visions
- Concentrate on getting something written within half an hour
  - You can always throw it away and do it properly later if it turns out to be useful
  - In the mean time you can answer some questions

Gathering and Storing Router Configuration

Shrubbery

- Easy access to the configuration of all your network elements in one place makes it easy to find things out
  - grep
  - More complicated stuff
- Revision history is also cool
- At least two freely-available packages exist
  - ciscoconf (slightly crusty, barely maintained)
    - http://dist.automagic.org/ciscoconf-1.1.tar.gz
  - rancid (much better, use this instead)
    - http://www.shrubbery.net/rancid/

rancid

- Really Awesome Network ConfIg Differ
- Data in:
  - various show commands on routers
- Data out:
  - Processed output of show commands, stored in CVS
  - diffs by mail
- Hey! Things got better already, and we haven’t even started yet

Router Configuration Audit

Config Audits

- This is an example config audit
  - checks for particular kinds of self-consistency within router configs
  - this is a simple example
  - there are lots of other things you can audit
- Some vendors allow you to refer to filters and lists which are not defined

```
no ip prefix-list USEFUL-FILTER-BBB
!
router bgp AAA
 neighbor 192.168.0.1 remote-AS BBB
 neighbor 192.168.0.1 prefix-list USEFUL-FILTER-BBB in
```

Undefined/Unused Filter Audit

- What information do we need to find problems?
  - router configs from rancid
- What should we do when we find problems?
  - Send mail
- Suddenly the problem looks easy to solve

3-Minute Design Exercise

- Every `$time`, run a script against each router config file stored by rancid
  - cron
  - By way of example, we’ll do cisco configs, but there are certainly other vendors for whom such audits are useful
- As the config file is being processed:
  - Make a note every time you see a filter definition
  - Make a note every time you see a filter reference
- At the end of the file, find out what doesn’t add up

Architecture Reminder

- Data in:
  - Configs retrieved and stored by rancid
- Data manipulation:
  - Identify inconsistent bits of config
- Data out:
  - Send mail
  - Could also open tickets, page people, etc.

filter_audit

- Written in awk and sh, as nature intended
- Sends mail to a nominated recipient
  - e.g. to internal NOC mailing list
  - Daily mail lists all undefined and unused filters
  - Immediate mail when the list changes
- ftp://ftp.isc.org/isc/toolmakers/filter_audit.tar.gz
  - Gratuitous bullet points added
  - To make URL above fit on one line

Max Prefix

20 prefixes into the future

Peer Prefixes

- Record the number of prefixes that BGP peers send you
  - Can be a useful troubleshooting tool
  - Might help choosing max-prefix limits
- Who *is* Max Prefix?
  - Lame joke, designed to make Stephen suffer
  - Everybody groan at Stephen

Handwaving

- Not obvious how to poll for this data using SNMP
  - A standard-across-vendors way of getting data would be nice, too bad there isn't one now
  - We could use MRTG if the data were in a MIB, available within our lifetimes?
- Various vendor-specific show commands give us the information we need
  - "show ip bgp summary"
  - "show bgp summary"
- Let's go with what we've got

2-Minute Design Exercise

- Every `$time` we’ll connect to all our routers and issue “show” commands
- We’ll parse the output, and record data
  - (time, neighbor address, neighbor AS, prefixes)
- We’ll store the data in flat-files
  - One file per (neighbor address, neighbor AS)
  - Store a timestamp and the number of prefixes

Architecture Reminder

- Data in:
  - List of routers, usernames and passwords
  - rancid’s router.db and .cloginrc
  - Output from show commands
- Data manipulation:
  - Strip out the columns we’re interested in
- Data output:
  - Flat files

peer_prefixes

- awk and sh
  - You weren’t expecting perl, were you? Hello?
- Could piggyback on rancid to do the show commands
  - Would require minor rancid tweaking (don’t want a rancid diff every time a peer drops a prefix)
  - Would result in fewer logins to routers, which is probably good
- ftp://ftp.isc.org/isc/toolmakers/peer_prefixes.tar.gz

Sample Use

- Implement sensible max-prefix limits on peers
- max(announced prefixes, IRR registered prefixes) + N%
  - Scale N% according to your comfort level
  - ~1,000 routes, N% can be “large” (say, 20)
  - ~30,000, maybe N% ought to be “small” (say, 5)
- Evaluate and update daily

Topology Visualisation

kc? You here?

IP Network Topology

- Representation of the network as an undirected graph
  - Edges are “circuits” which connect router interfaces int-a and int-b
    - (int-a-name, int-a-addr, int-b-name, int-b-addr)
  - Nodes are routers
    - router name
- We can build this easily from router configs
  - Match up interfaces on different routers that are in the same subnet

Applications

- If we have a representation of the network as an undirected graph, we can use diagramming tools to build maps for us
  - Since we have edges identified according to interfaces, we could even colour circuits in maps according to error conditions or traffic load
- With a bit more effort we can answer questions
  - “What are all the ways of getting from Vancouver to New York in under four hops?”
  - Assuming that’s a useful question to ask :)
  - It’s a fun question, anyway *(NB, Joe is from Canada)*

Applications

- The topology information contains other things which could be useful
  - (address, interface name, router name)
  - We can generate DNS zone files from that, with great ease
  - Hooray for software reuse

1-Minute Design Exercise

- These design exercises are getting quicker
  - Note design reuse
- Every `$time`, run a script against each router config file stored by rancid
  - Where have we seen this before?
- Match up connected interfaces
- Output the result as some kind of flat file

Architecture Reminder

- Data in:
  - Router configs, retrieved and stored by rancid
- Data manipulation:
  - Match up the interfaces
- Data out:
  - Topology file

mktop

- Written in perl (only kidding, you know it's awk)
- Pipe a concatenated set of cisco or Juniper configs together, and feed them to mktop

```
cat configs/* | mktop >net.top
```

- Output file format is too ugly to put on a slide
  - Colon-delimited flat-file of doom
  - There's a man page, though
- ftp://ftp.isc.org/isc/toolmakers/mktop.tar.gz

Automatic Map Construction

Drawing Maps

- Various tools exist which can generate graphical representations of undirected graphs
- One such tool is GraphViz
  - /usr/ports/graphics/graphviz (don't you just love ports?)
  - http://www.research.att.com/sw/tools/graphviz/ for the ports-less
- Input file format is "dot"
- We just need to convert "top" to "dot"

30-Second Design Exercise

- This is really easy
- Read a .top file, shuffle a couple of things around and make a .dot file
- By playing with the dot files, we can make the maps more readable
  - Choose a subset of routers to include
  - Make edges between cities longer
  - Ugly maps for now, since beauty is in the eye of the operator

Architecture Reminder

- Data in:
  - A “.top” file generated using mktop
- Data manipulation:
  - Trivial awkery
- Data out:
  - A “.dot” file which is edible by graphviz (dotty)
  - graphviz will give us PNGs, imagemaps, etc.

top2dot

- This is a very basic tool; it can be extended:
  - Include tags for imagemaps in DOT file
  - Colour links between routers in a useful way
  - Make links between cities stretchier than links within cities
- Run it periodically to generate maps, or run it on-demand from a CGI script
- ftp://ftp.isc.org/isc/toolmakers/top2dot.tar.gz

Router Interfaces in the DNS

PTRs and As

- Having accurate RRs in the DNS for router interface addresses and names makes traceroute output more comprehensible
  - Can help customers help themselves
  - Can improve quality of questions
  - Useful for quick troubleshooting
  - Makes you look as though you know what you’re doing

15-Second Design Exercise

- Take (interface address, router name, interface name) from a .top file
- Apply the naming policy that:
  - The architects just spent two years arguing over
  - Has only stabilised because it’s the only thing that everyone dislikes and can therefore agree upon
  - Fortunately by this stage nobody cares any more
- Spit out a hosts(5) file
  - Generate PTR and A records from the hosts(5) file
  - Throw the resulting zone files at BIND

Did Somebody Say hosts(5)?!

- Hey, give me a break, I only had 15 seconds to think about it
- There are existing tools to generate zone files from hosts(5) files, so by creating a hosts(5) file we avoid reinventing the wheel
- Also, not everybody uses BIND, and not all nameservers are driven using zone files

top2hosts

- Written in fortran^W awk
  - Those awk gags just don't get old
- The naming scheme represented in top2hosts is from a promising local ISP:
  - `er1a.iad2.us:so-3/2 □ so-3-2.er1a.iad2.us`
  - `pr2.pao1.us:Gig4/1.357 □ 357.ge-4-1.pr2.pao1.us`
- Substituting alternate naming schemes is not hard
  - Topology-based naming schemes can also be accommodated
- ftp://ftp.isc.org/isc/toolmakers/top2hosts.tar.gz

Parting Thoughts

Cause and Effect

- Think about synchrony, load on routers
  - Locking problems?
  - vty exhaustion? Other resources?
  - Set limits on tools running simultaneously
- Think about failure modes
  - Check that valid input was received
  - Consider failure modes, and contain damage rather than spreading it

Multiple Vendor Pain

- There are lots of reasons to run a multi-vendor network
- The greater the similarities between different vendors' instrumentation, the easier it is to write tools that talk to different kinds of routers

Pictures of a Better Place

- Vendor-specific features are at least presented in some vendor-neutral way
  - XML with a DTD per vendor (Juniper does this)
  - SNMP (ha!)
  - Some help would be an improvement over no help
- If vendors won’t do it, middleware would be useful
  - Something that can wrangle “show” commands into some consistent (e.g. XML) output format

Script Crappiness

- Some of the scripts presented in this tutorial are crappy
  - Deliberately crappy! Honest!
  - Stephen made me do it! *(I did)*
- Crappy != bad
  - Focus on the task at hand, not on code purity
  - Perfect is the enemy of done
  - Small scripts with well-defined purposes can always be replaced later, if they turn out to be useful

Toolmaker-thon

- This tutorial has hopefully legitimised some degree of empowered scripting
  - There is also a BOF
- There is a new mailing list for general discussion of network scriptery:

  toolmakers-request@isc.org

  “subscribe” in subject or message body
- Go Forth and Make Tools