netmap

or

the *mini-Ark* project

CAIDA/WIDE/CASFI, 4 April 09

WAND, 20 Mar 09

Nevil Brownlee

Background: Internet in Auckland

- U Auckland is a big content provider
- Internet in Auckland used to have a simple topology
- All the ISPs connected to APE, Auckland Peering Exchange

- Telecom NZ and TelstraClear left APE
- Now some larger ISPs connect to both TNZ and TCL
- It's hard to determine where congestion is occuring

- We'd like to have a more accurate topology
- and a near-realtime traffic weather map!

Backgound (2): U Auckland Campus Network

- The campus network is highly resilient
- Many services are concentrated in our central Data Centre
- Users often complain of ‘poor network performance’
- The topology is resilient (good) but poorly documented (bad)
- Again, would like an accurate topology and weather map

Solution: Distributed Monitoring

- Develop central server + ‘user-machine’ clients
- Clients to perform measurements between self and other clients
- Also (possibly) a set of fixed hosts (e.g. popular web sites)
- Use netmap measurement tools

Other Projects

- Scriptroute
  - Uses measurement servers at known sites
  - Long-term server support problems
  - Users can run Ruby measurement scripts on measurement servers
  - Co-ordinated via central web site
- Dimes
  - Uses many clients on ‘user’ machines
  - Widespread view of Internet from user point of view
  - Low maintenance (doesn’t matter if we loose clients)

Other Projects (2)

- Nettest
  - Clients on user machines
  - Passive measurement only
    - monitors flows and sends data to central server
  - Clients in C, specific to OS (XP, Vista, OS X, Linux)
  - *Allows for automatic upgrades of client software*
- Ark
  - CAIDA project, developed by Young Hyun
  - Co-ordination system for CAIDA's topology measurement infrastructure
  - Uses `scamper` to make IPv4 and IPv6 traceroutes
  - Written in Ruby, uses *tuples* for shared data
  - Uses dedicated measurement hosts (not 'user' clients)

Implementation Strategy

- Write everything in Ruby!
  - ruby + mysql for server
  - rails for database/web pages
  - fxruby for gui (if/when needed)
  - rubyscript2exe can create clients for all the OSes
    - Can determine OS in Ruby, should allow single script for all OSes
- What measurements can we make?
  - General (scriptroute-style) – too hard
  - Link capacity, e.g. using Pathrate – also too hard
  - Topology, i.e. links and (maybe) one-way delays – traceroute

Implementation Strategy (2)

- What about Firewalls?
  - measurement between clients needs e2e addressing
    - Firewalls block that
- Skype and friends have cunning schemes to get through firewalls
  - we don’t want to go there
- traceroute is single-ended
  - trace as near to target IP address as we can
- decided to just use traceroute initially

Implementation Strategy (3)

- Could we use `scamper`?
  - Good traceroute capability, good Ruby interface
  - Need to install and run `scamper` on client hosts
  - Decided to just use system traceroute (already installed)
- Make server do most of the work
  - keep clients minimal
  - simple TLV-over-TCP protocol
- How to Visualize Topologies?
  - use GraphViz
    - well documented
    - ruby module
- need to map IP addresses to ISP
  - Use *uspmon* IP address data (/24 prefixes)
  - Look up ASNs for prefixes using http://www.team-cymru.org/Services/ip-to-asn.html

Summer 2008-9 Project Goals

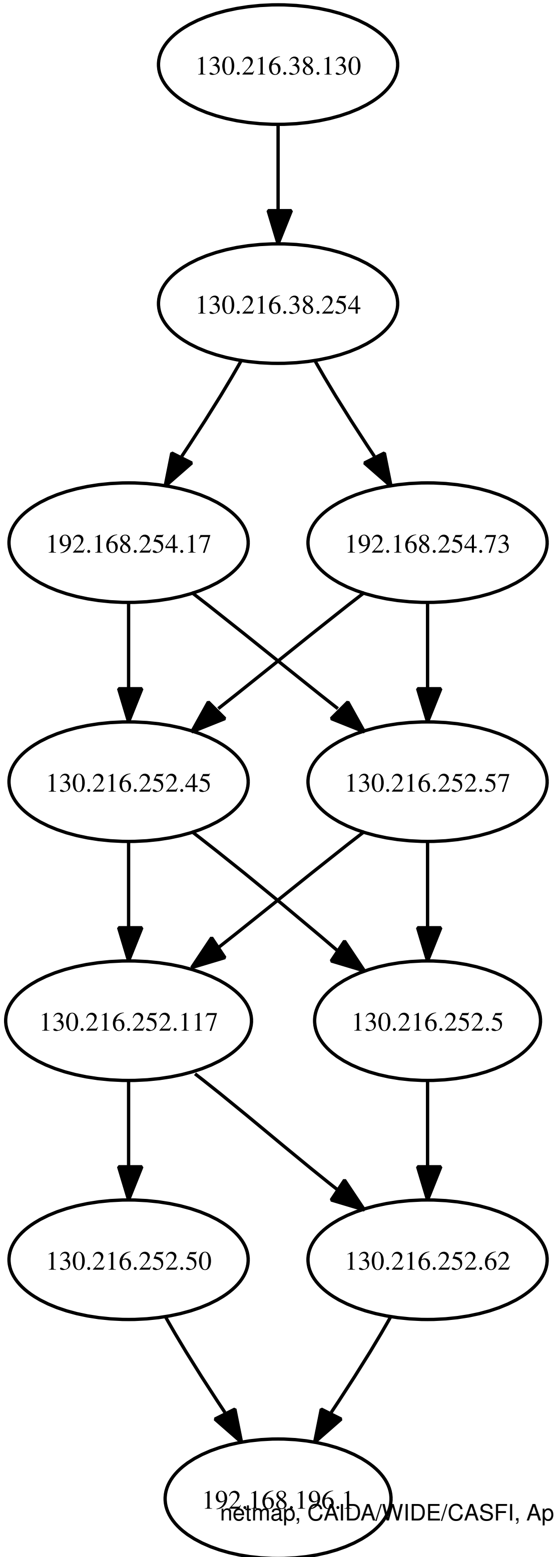
- Implement server and client in Ruby
- Use system traceroute as only measurement tool
  - make mysql database for Traceroutes and Hops only
- Deploy several clients around U Auckland campus network
- Collect (lots of) traceroute data
- Use the traceroute data to draw topology diagrams (if time)

System Structure

- OB starts thread for each client
  - (better to use Ruby EventMachine)
- mysql tables
  - ClientInfo, Hops, TraceRoutes
- Server threads
  - handle client login
  - tell client IP addresses to traceroute to
  - receive data and store it in database
- Client
  - login to server
  - ask server for target IP addresses
  - traceroute to them, send data to client
  - sleep for ‘measurement interval’
  - loop with next set of targets

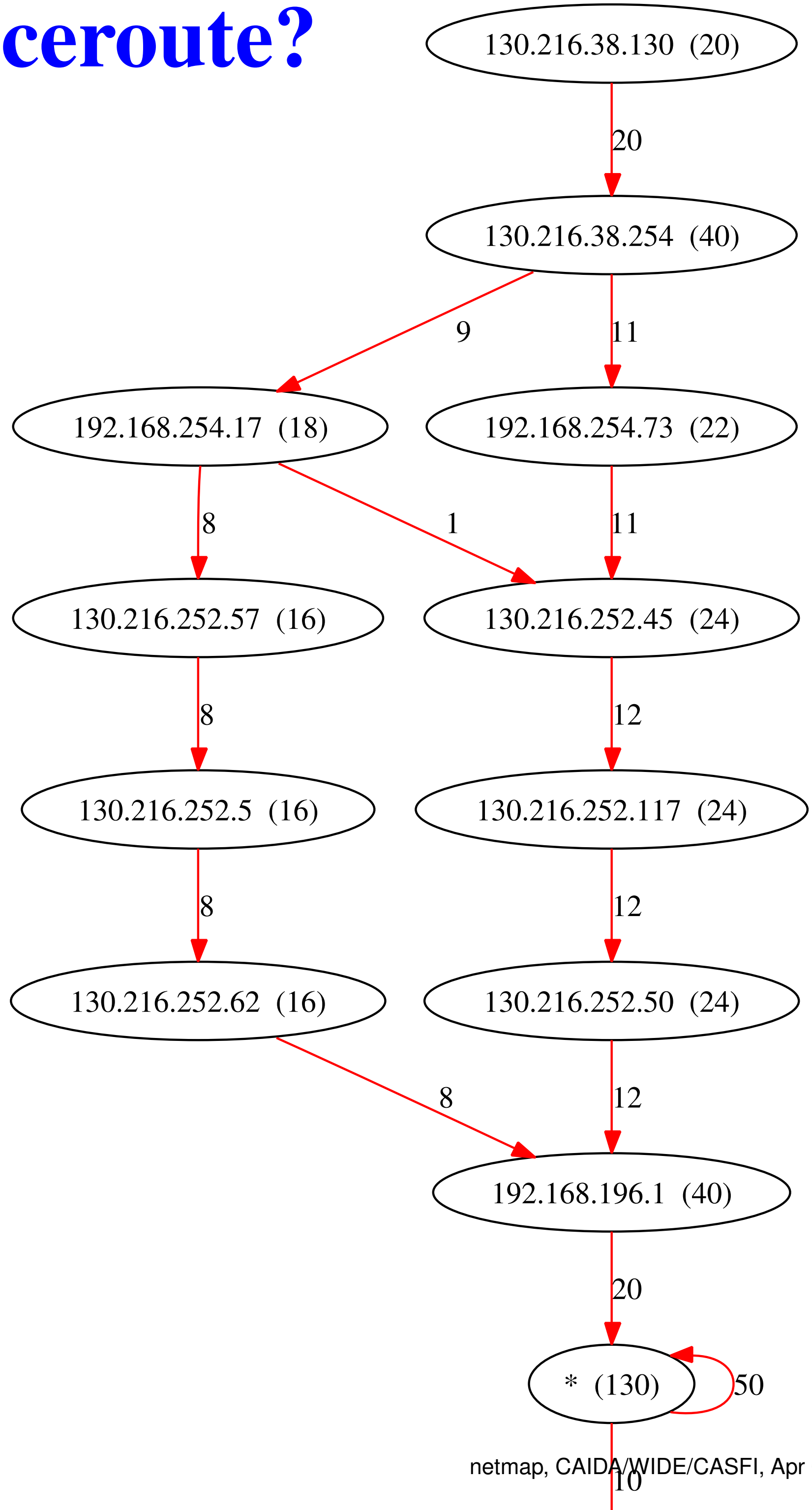
Topology from traceroute data?

- traceroute measures rtt for ttl = 1, 2, ...
- Default is three tries for each ttl
- Assume that each column of output is a route
- That's fine if there's only one path
- But U Auckland network has (lots of) resilience . . .
- Paths from Nevil's desktop Mac to *dnsparse* VM (in Data Centre) ⇒



‘Best guess’ routes from traceroute?

- Would n times 1 try per hop be better?
- Seems to be!
- Anyone got any better ideas?



Summer Project Summary

- Proof of concept achieved by two (end of 2nd-year) students in 10 weeks
- Next steps:
  - improve server/client code
  - get windows and OS X clients working
  - collect lost more data at U Auckland
  - explore ways to visualise the data well
  - make pretty web pages
  - try running clients in Auckland Internet
  - . . .

WAND feedback

- Questions about '$n \times 1$' traceroute strategy
- `scamper` tries really hard to map links
  - that includes keeping porobe packet fields same for all TTLs
  - routers/switches along path should use same hash from packets
- Trying `scamper` at Auckland on same path as before produced single paths
  - so did traceroute, same path, even with '$5 \times n$' strategy
  - ditto `scamper` using Paris traceroute,
    `scamper -c 'trace -P icmp-paris' -i 130.216.190.25`
- *But,* it was a zig-zag path on previous diagrams!
- Clearly, paths can and do change over time in the U Auckland network
- Matthew has a Windows `scamper` in development

And now . . .

- David McDonald, Postgrad Dissertation student, is working on netmap
- New server and client
  - uses SOAP to exchange data
  - will look at paths between clients (SOAP uses http transport)
  - will use `scamper` once Windows version is available
  - about to start collecting traceroute data in U Auckland network
- Concentrate on visualising topology
  - David has a strong background in viualisation
  - he's doing a lot of background reading
- It's now a work in progress !