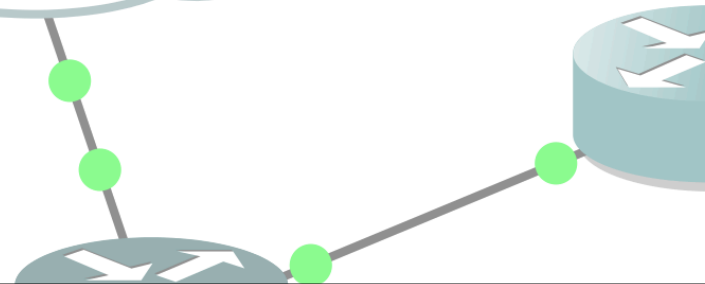


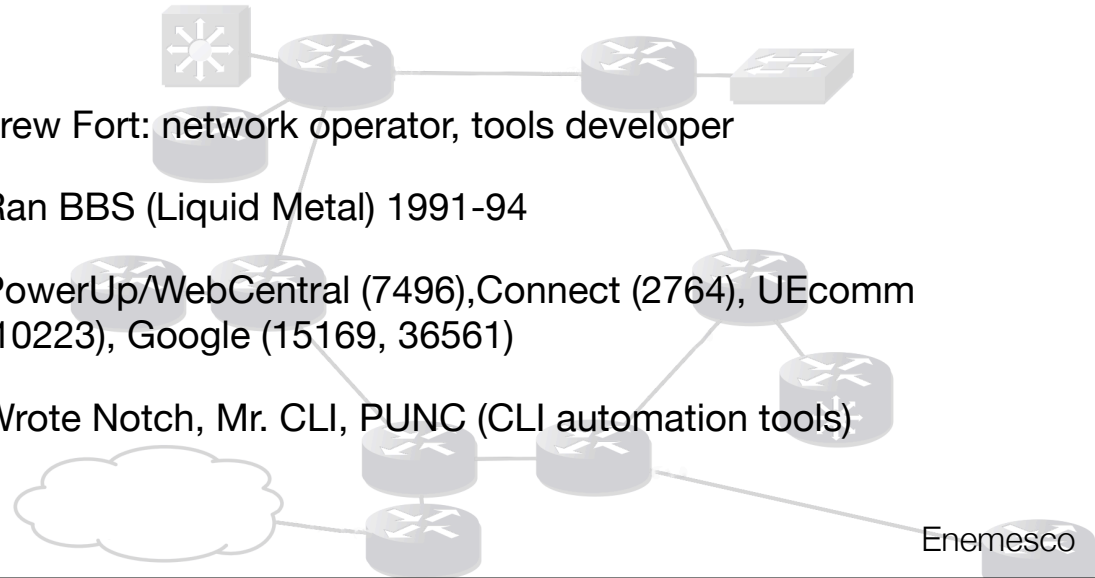
Pragmatic Network Management Systems

Andrew Fort, EnemESCO
andrew.fort@enemESCO.net



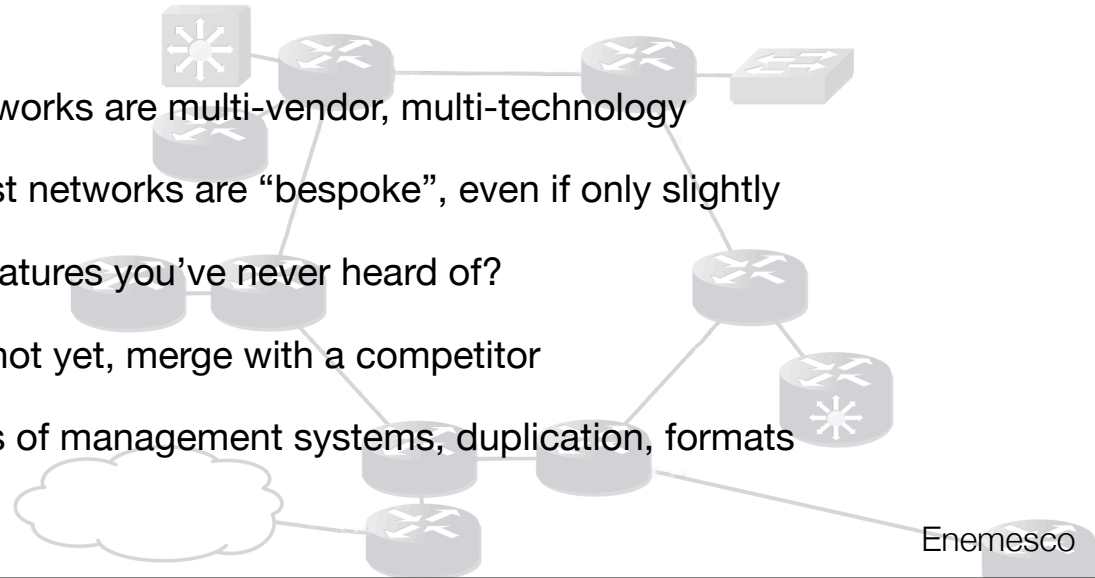
Introduction

- Andrew Fort: network operator, tools developer
 - Ran BBS (Liquid Metal) 1991-94
 - PowerUp/WebCentral (7496), Connect (2764), UEcomm (10223), Google (15169, 36561)
 - Wrote Notch, Mr. CLI, PUNC (CLI automation tools)



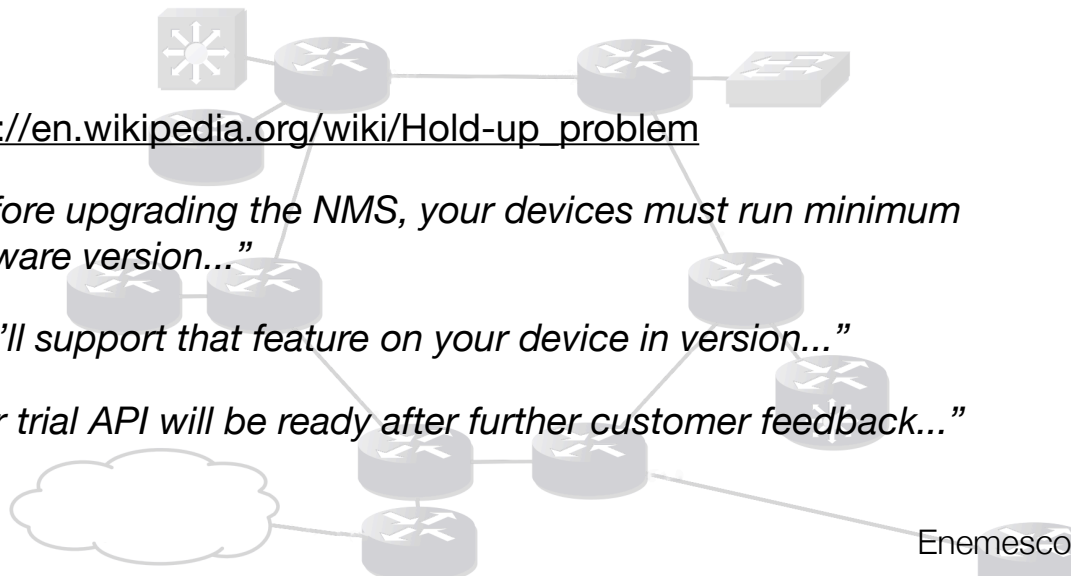
The problem...

- Networks are multi-vendor, multi-technology
- Most networks are “bespoke”, even if only slightly
 - Features you’ve never heard of?
 - If not yet, merge with a competitor
- Lots of management systems, duplication, formats



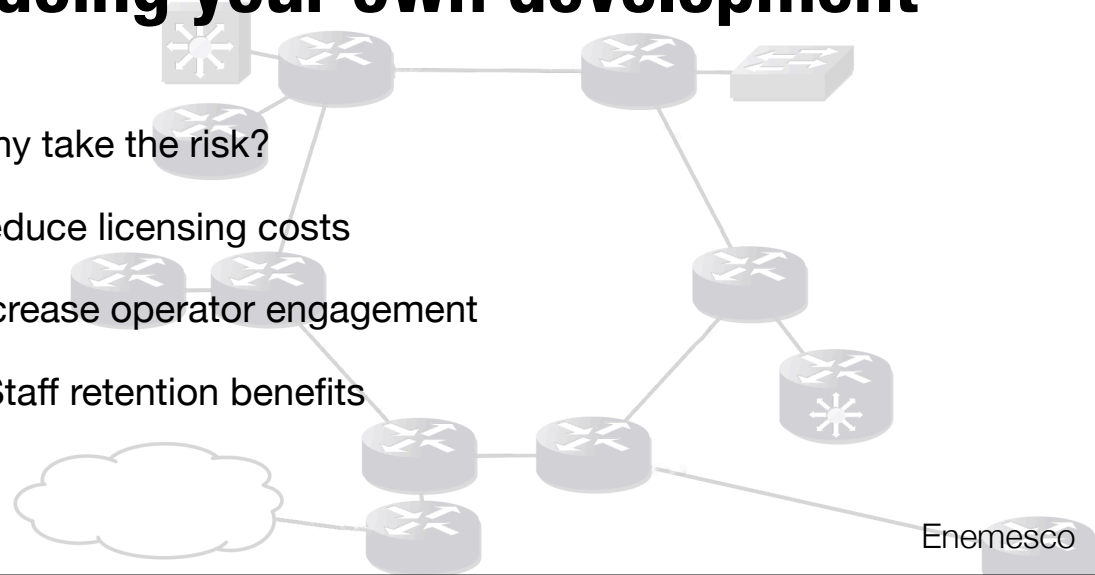
Hold-up problem

- http://en.wikipedia.org/wiki/Hold-up_problem
- *“Before upgrading the NMS, your devices must run minimum software version...”*
- *“We’ll support that feature on your device in version...”*
- *“Our trial API will be ready after further customer feedback...”*



Business benefits to doing your own development

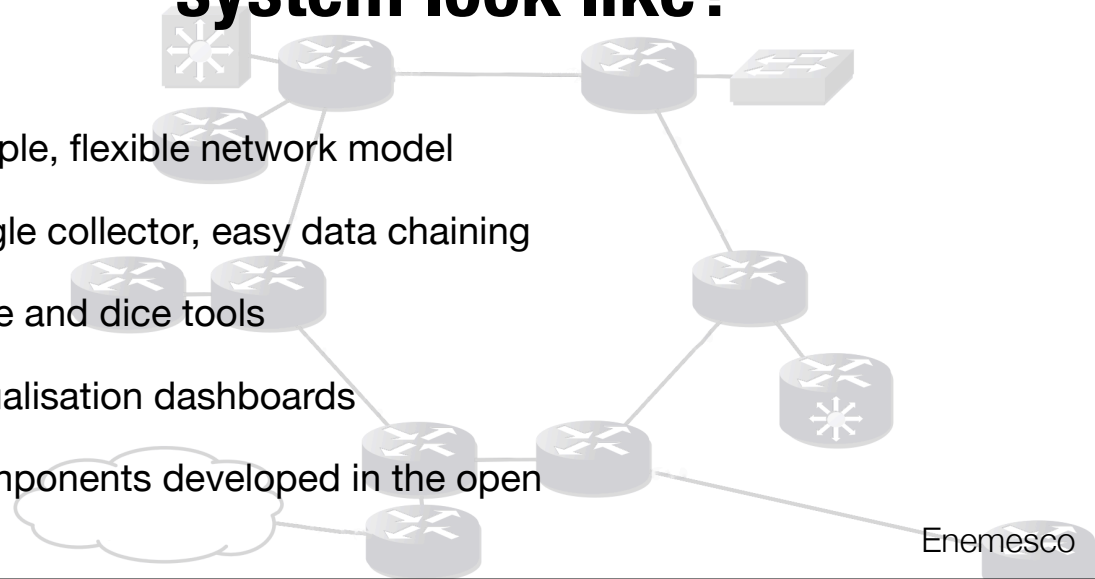
- Why take the risk?
- Reduce licensing costs
- Increase operator engagement
- Staff retention benefits



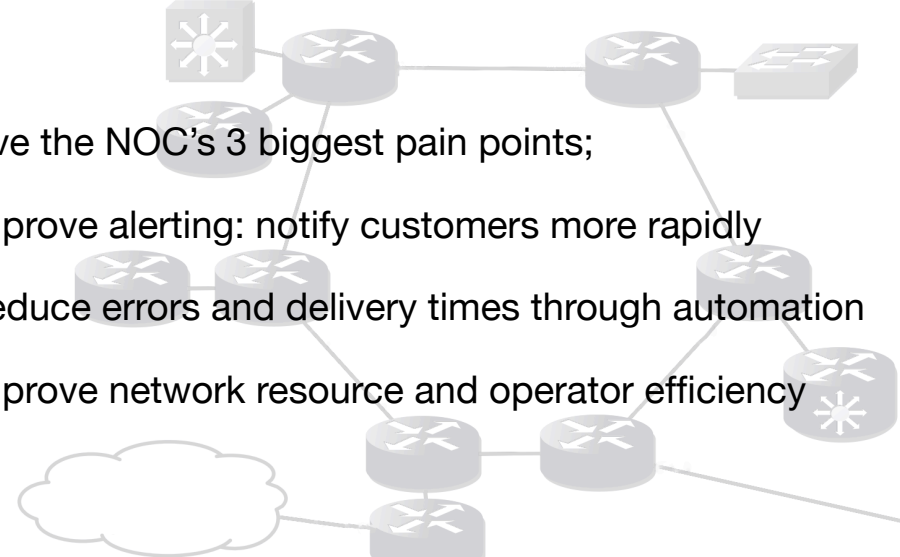
Operators tend to feel like packages get dumped on them

What does a pragmatic system look like?

- Simple, flexible network model
- Single collector, easy data chaining
- Slice and dice tools
- Visualisation dashboards
- Components developed in the open



Project goals

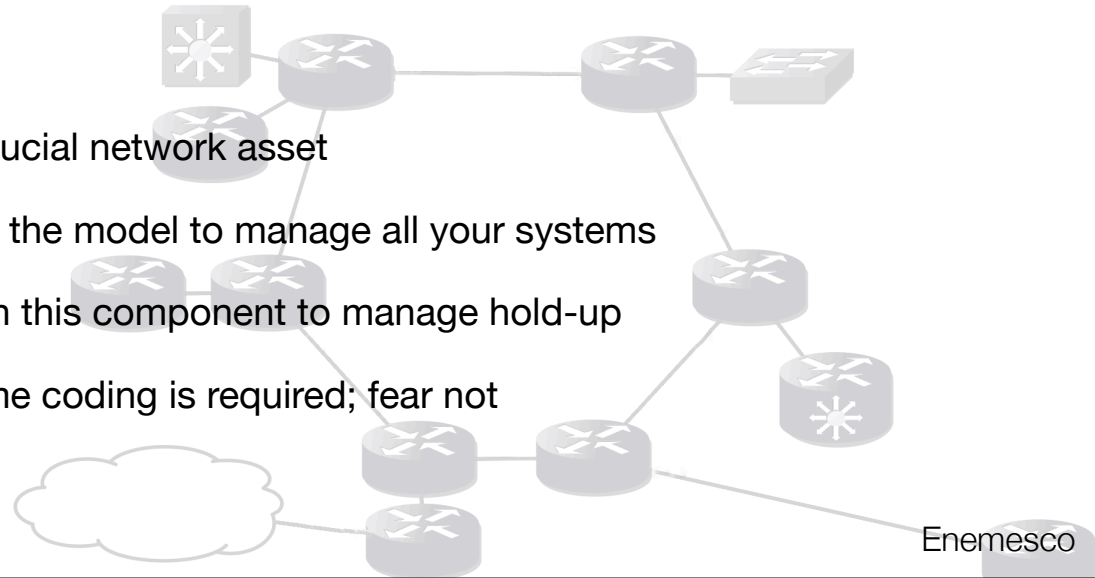
- 
- Solve the NOC's 3 biggest pain points;
 - Improve alerting: notify customers more rapidly
 - Reduce errors and delivery times through automation
 - Improve network resource and operator efficiency

Enemesco

- Faults, operations
 - Which customers are affected?
 - What equipment is involved?
 - Recent network changes, events
 - What are the devices reporting right now?
- Planning
 - What equipment do we have? In which lifecycle state?
 - Growth and deal planning
 - Historical network views, reporting

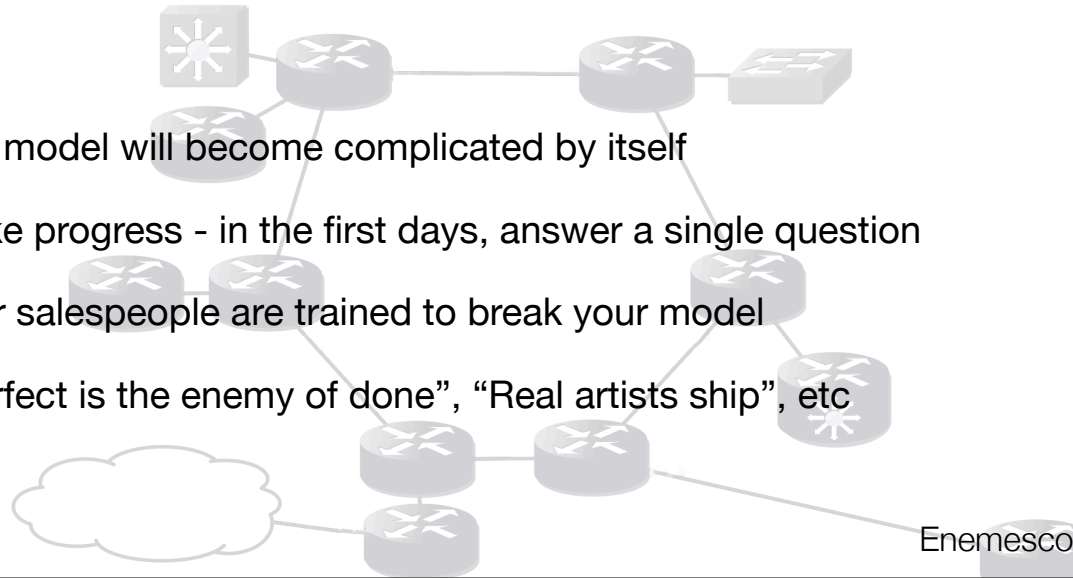
Build your own network model

- A crucial network asset
- Use the model to manage all your systems
- Own this component to manage hold-up
- Some coding is required; fear not



Don't get (too) clever

- The model will become complicated by itself
- Make progress - in the first days, answer a single question
- Your salespeople are trained to break your model
- “Perfect is the enemy of done”, “Real artists ship”, etc



Derive everything you can

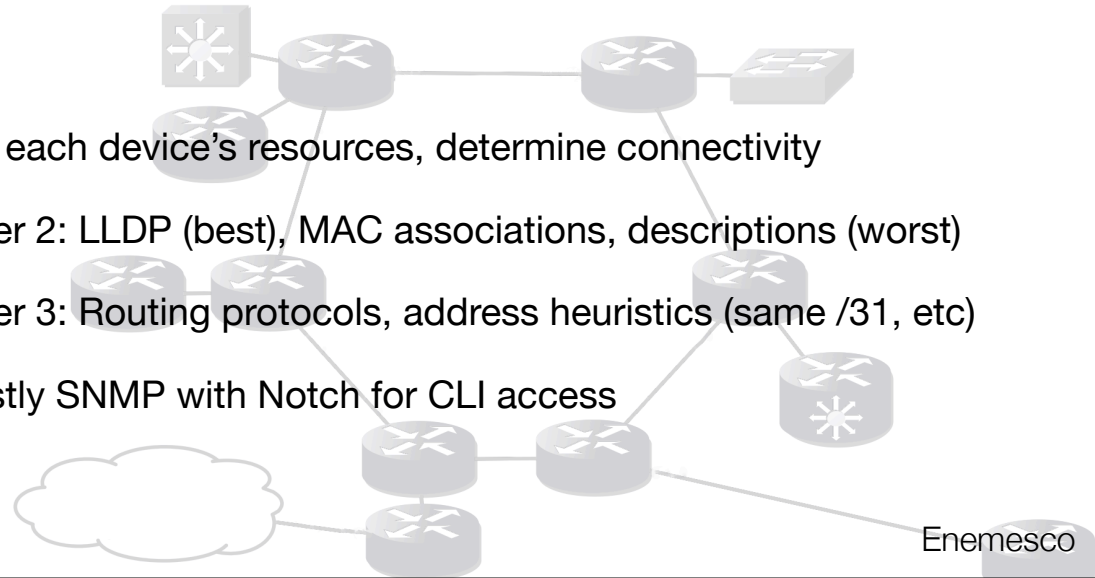
You can keep historical as well as future state, and the system knows what to overwrite with learnt data and what not to.

Some limited “live data” in your authoritative model is acceptable. For example, Interfaces modelled during discovery will not update the flags like ‘monitored’ on existing interfaces, but other changes (like an IP address change), you would like to automatically update.

Alot of the fears of mixing live data in the authoritative model go away as history can be used to synchronise and revert changes.

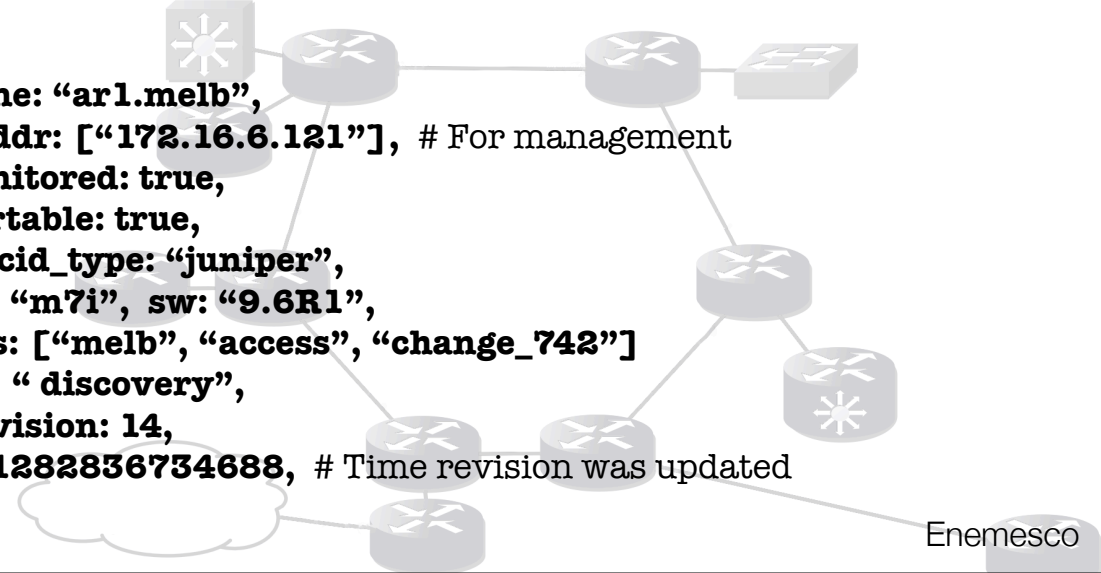
Discovery engine

- Poll each device's resources, determine connectivity
- Layer 2: LLDP (best), MAC associations, descriptions (worst)
- Layer 3: Routing protocols, address heuristics (same /31, etc)
- Mostly SNMP with Notch for CLI access



Example device model

```
{  
  name: "ar1.melb",  
  ipaddr: ["172.16.6.121"], # For management  
  monitored: true,  
  alertable: true,  
  rancid_type: "juniper",  
  hw: "m7i", sw: "9.6R1",  
  tags: ["melb", "access", "change_742"]  
  via: "discovery",  
  _revision: 14,  
  ts: 1282836734688, # Time revision was updated  
}
```



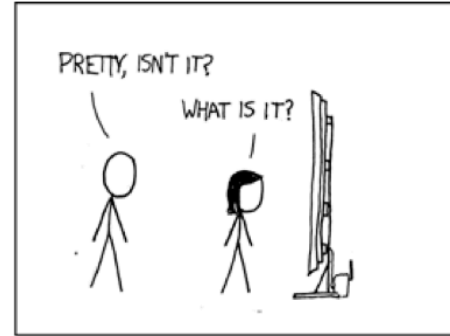
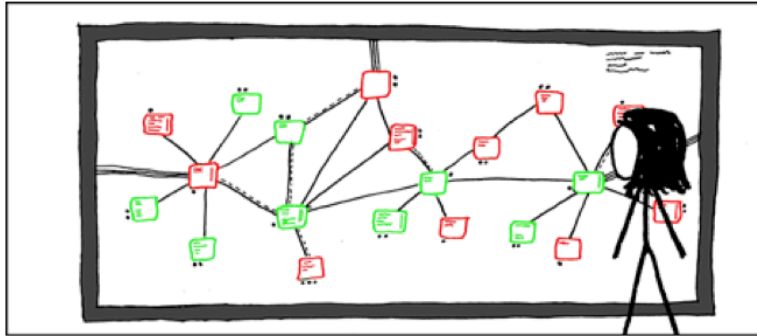
Enemisco

Each model can be a value in a key/value or document database.

Indexes and range queries allow us to slice and dice the data model effectively, and answer questions like

What was the metadata on this device three weeks ago last tuesday?

Visualisation

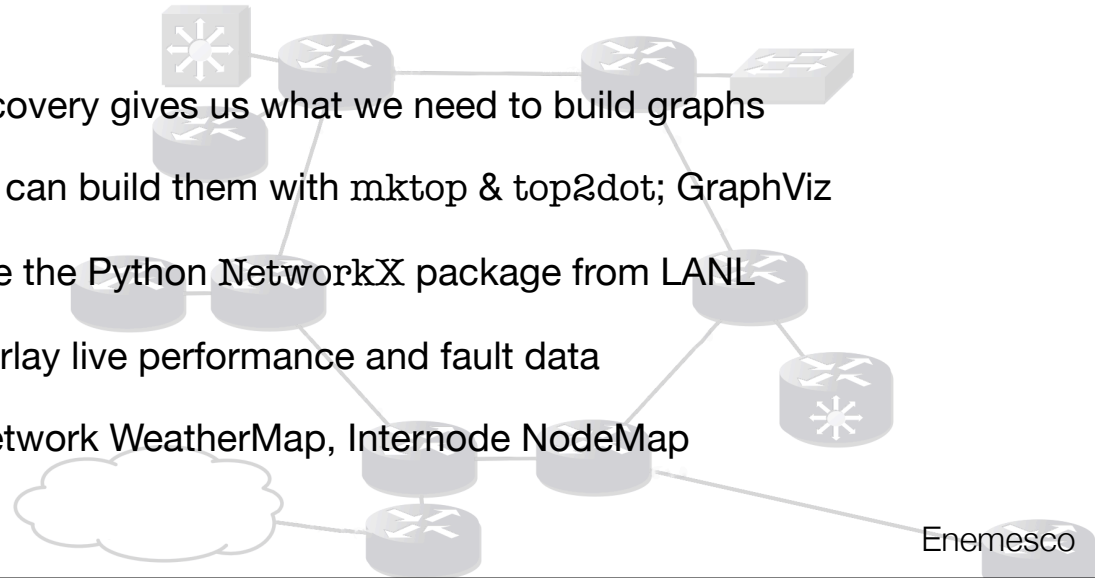


<http://xkcd.com/350/>

Enemesco

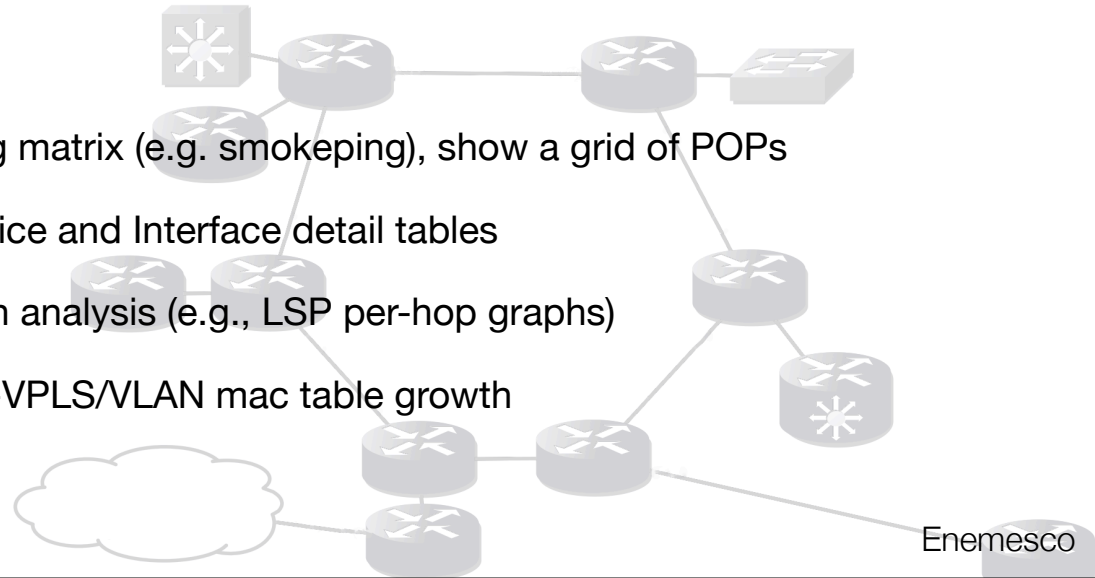
Visualisation / Dashboards

- Discovery gives us what we need to build graphs
- You can build them with mktop & top2dot; GraphViz
- I use the Python NetworkX package from LANL
- Overlay live performance and fault data
- Network WeatherMap, Internode NodeMap

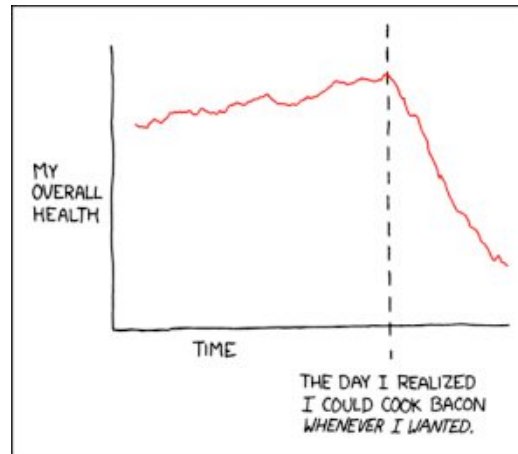


Dashboard types

- Ping matrix (e.g. smokeping), show a grid of POPs
- Device and Interface detail tables
- Path analysis (e.g., LSP per-hop graphs)
- Per-VPLS/VLAN mac table growth



Further visualisation reading



<http://xkcd.com/418/>

- Read: “The Visual Display of Quantitative Information” (Tufte)
Enemesco

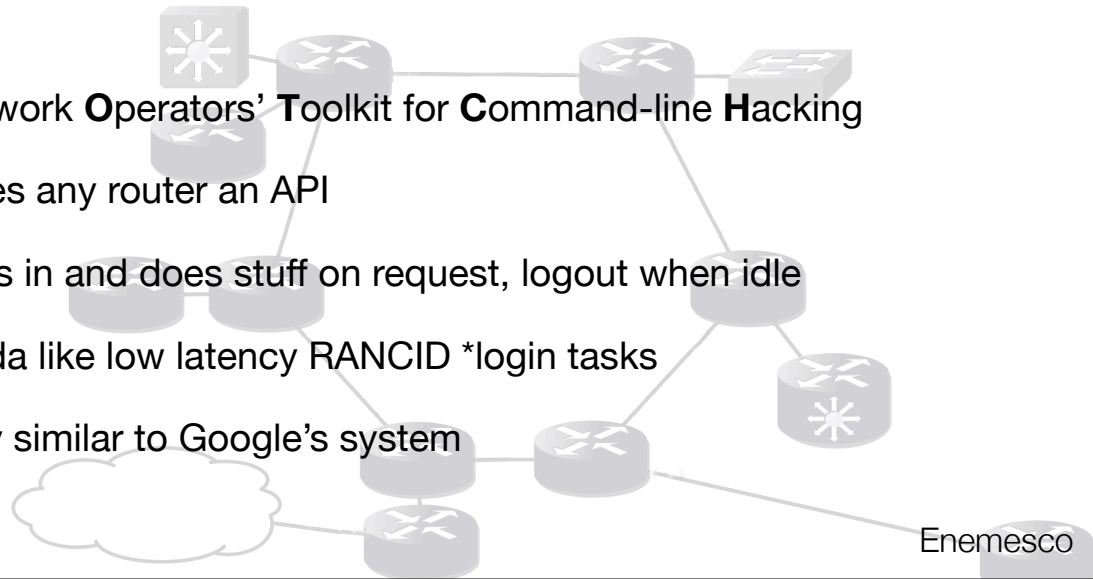
Notch: automating the CLI



Enemesco

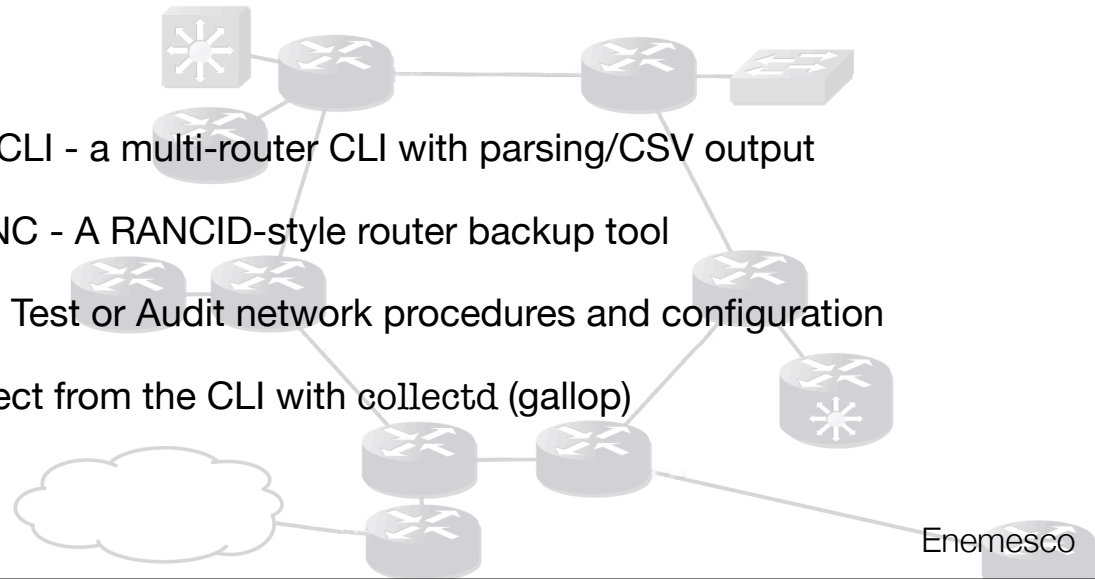
What's Notch?

- **Network Operators' Toolkit for Command-line Hacking**
- Gives any router an API
- Logs in and does stuff on request, logout when idle
- Kinda like low latency RANCID *login tasks
- Very similar to Google's system

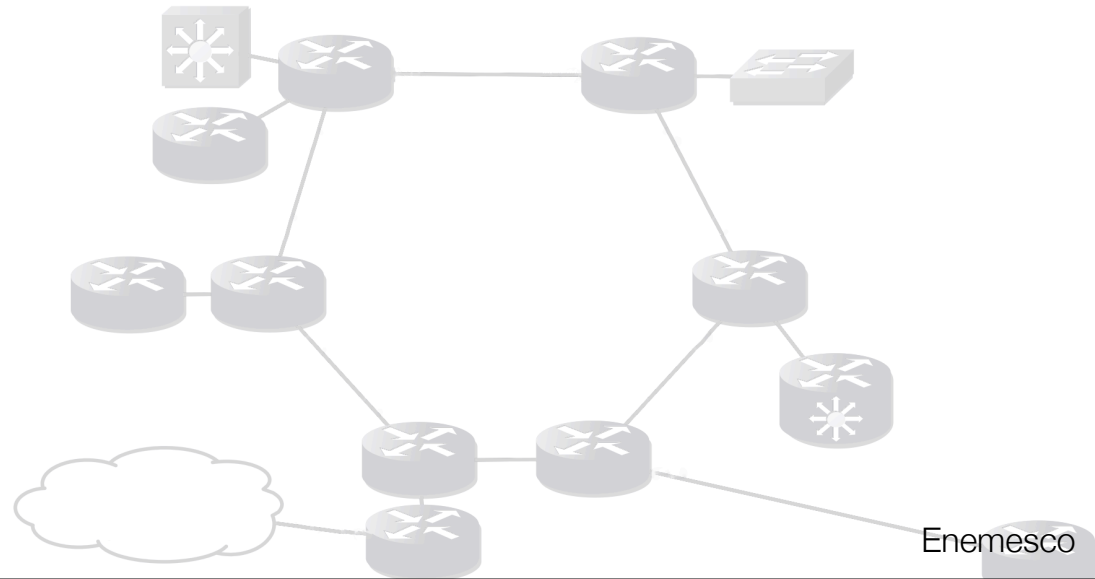


What's Notch used for?

- Mr. CLI - a multi-router CLI with parsing/CSV output
- PUNC - A RANCID-style router backup tool
- Unit Test or Audit network procedures and configuration
- collect from the CLI with collectd (gallop)

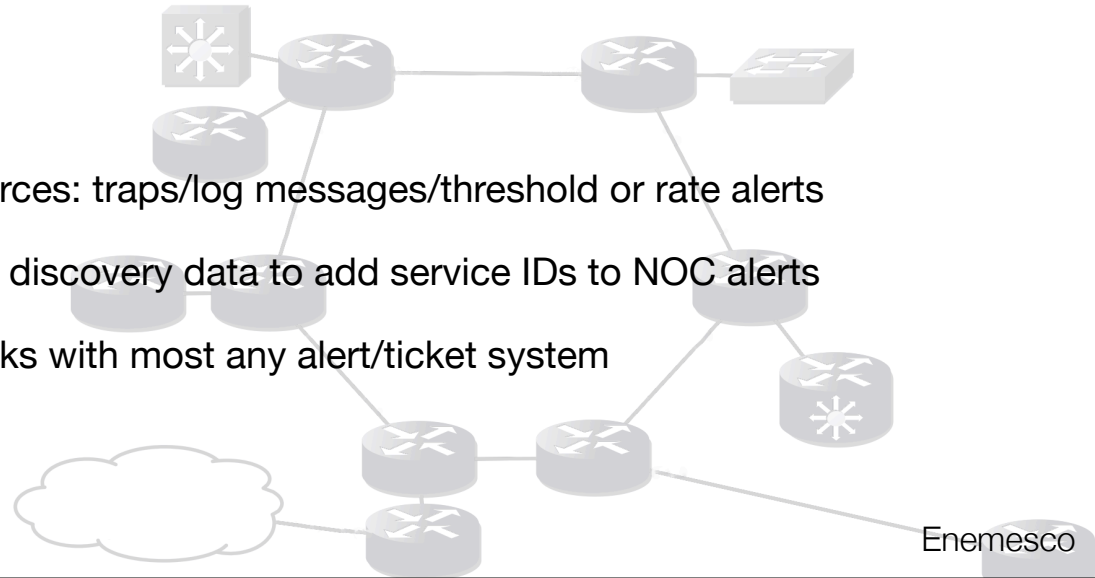


Notch/Mr. CLI demo



Alerting

- Sources: traps/log messages/threshold or rate alerts
- Use discovery data to add service IDs to NOC alerts
- Works with most any alert/ticket system



Improving alert value; how to do it

One piece of information that tells you what's broken, who's affected and where it is.

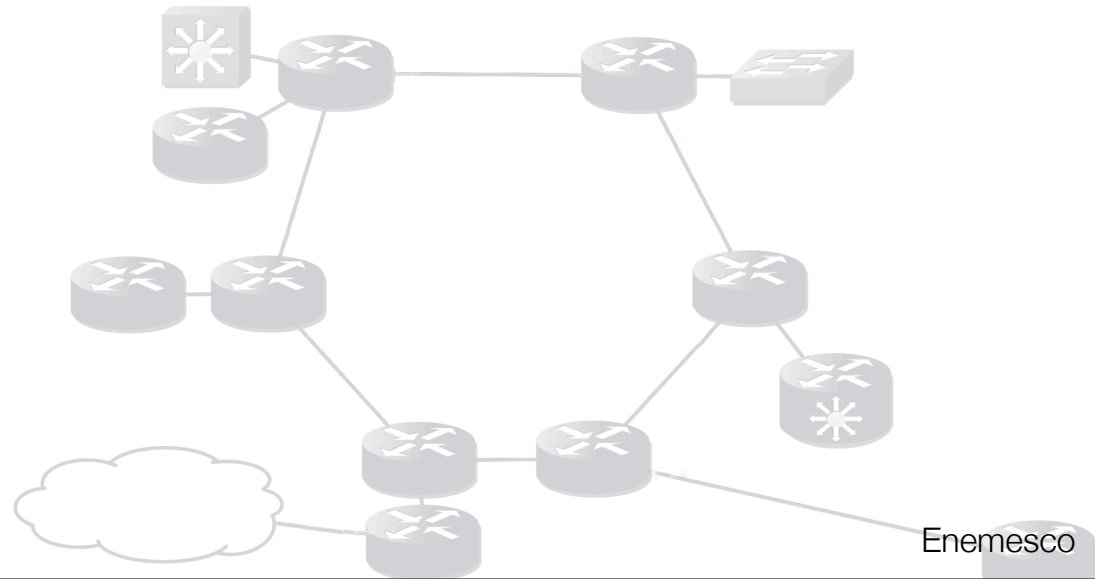
In summary

- Commercial tools are often under utilised, have high prices and are expensive to customise
- Free software is now available for every component
 - Like a modern web application
 - Integration costs are similar; no ticket cost
- Start with concrete goals

Enemesco



Questions?



Thank you!

- Read this blog to follow our network model development
- <http://blog.enemesco.net/>

