Google

How to repair Networks at scale? SRE approach

Ruairí Carroll, SRE

Number of repairs

Growth over time:

- Smaller failure domains == smaller boxes, but more of them
- Organic growth
- Routers getting cheaper
- Hardware getting more complex

Non answers

- Hire more Network engineers
- Buy routers that don’t break
- Make your Network engineers work harder year over year
- Don’t repair at all

Repair:Human ratio

1:1

- Pick a phone and call TAC
- Use your CCIE/JNCIE skills
- Art not craft
- Network-engineer centric

10:1

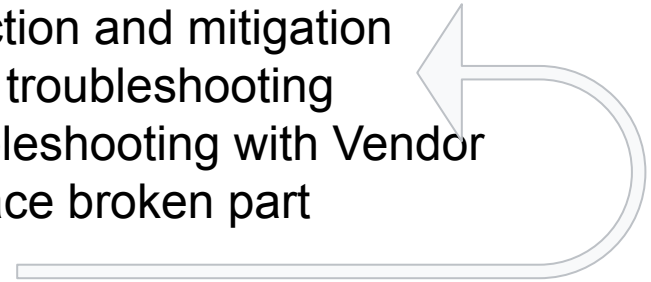
- Tooling-assisted repair
- Common operations done by tools
- Abstraction layers start to emerge
- Engineers orchestrate tools in order and deal with escalations

100:1

- Fully automated repair (humans involved just for physical work)
- API towards vendors
- API towards partner teams (logistics, remote hands)
- Engineers handle code changes, not actual repair

Life of a repair

1. Detection and mitigation
2. Local troubleshooting
3. Troubleshooting with Vendor
4. Replace broken part
5. Verify
6. Close



API towards Router

- Collect diagnostic information and download it
- Inventory data
- Offline/online components
  - Fetch component status

API towards vendor

- Open/Close vendor case
- Upload diagnostic information
- Follow up requests in a structured way (commands/files/onsite work)
- RMA - approval/status/return

Vendor TAC API

Historic:

- Human centric
- Phone call
- Email

Now:

- Fetch full text updates of the case via REST
- Upload files to TAC via API

Future:

- API towards all requests from TAC
  - Request command from router
  - Download file from router
  - Request onsite work (LC/SFP swap)
  - RMA pipeline

Create TAC API for Vendor Requests

Existing Vendor TAC API is one-sided. There is not much communication back from Vendor in structured way.

Create our own and request Vendor use it:

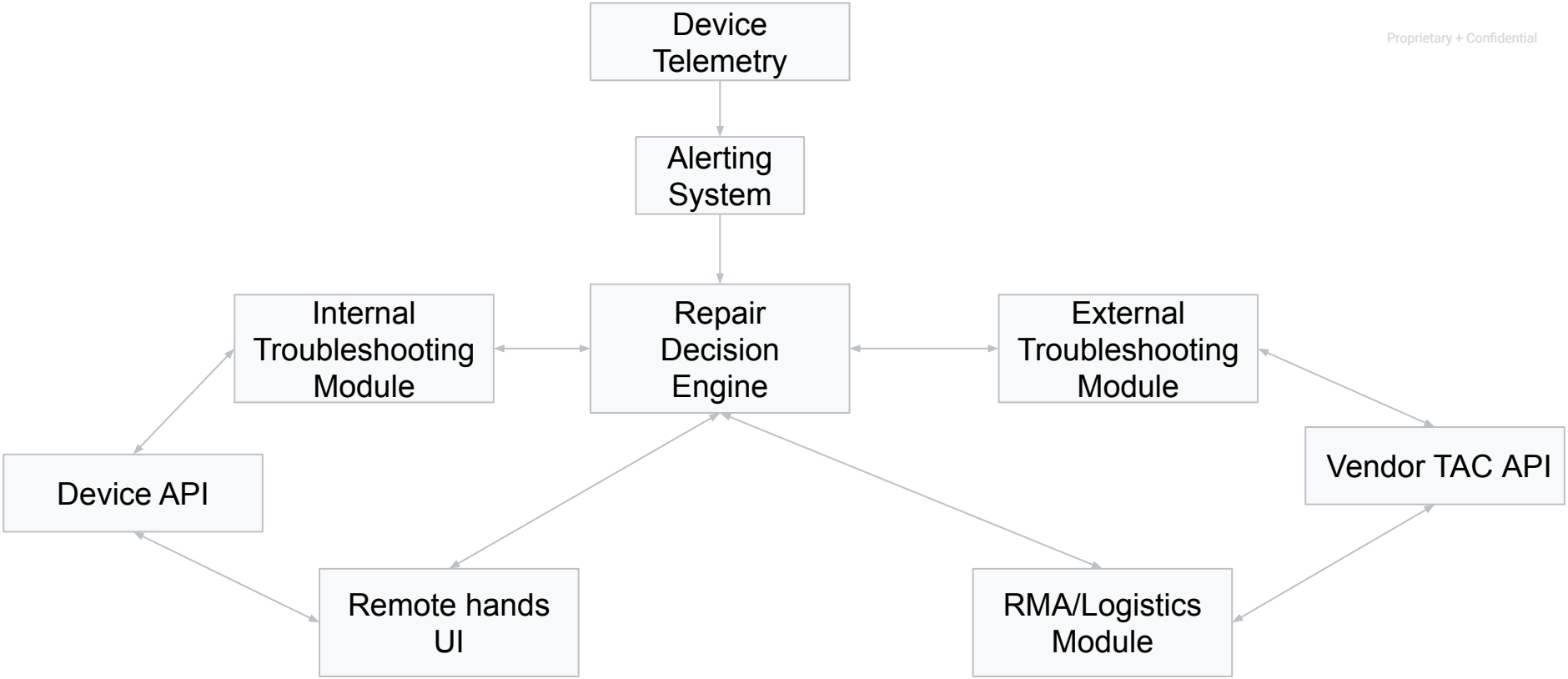
- Request command from router in structured way
- Select component to RMA in structured way and start logistics

Benefits:

- Requests are processed automatically without involving engineer
- Faster turnaround time
- Analyze historic requests
- Can validate requests before execution

Life of a repair

1. Alert for a component failure.
2. Diagnostic information is collected automatically.
3. Component is restarted and we wait for it to come up
4. If it doesn't, vendor case is opened, diaginfo provided.
5. Additional requests of information from Vendor are processed automatically.
6. Once RMA approved, and part is shipped, remote hands uses self-service UI that guides them through replacement.
7. Verify component is replaced, and passses audit.
8. Close case on success.



Conclusion

- Cattle not pets
- Start with API
- Aim for fully automatic repair
- Write software not Playbooks
  - Existing Playbooks are easy to translate into code

Questions?

Appendix

Comments about API towards Router

- FoSS options exist
  - ansible, nornir, paramiko etc
- Fetching structured data mostly depends on vendor
  - SNMP, RestCONF, NetCONF widely supported
  - Text Parsing :(
- Device abstractions are part of chosen language
  - Python: abc, Go: interfaces