NAF
NETWORK AUTOMATION FORUM

Proton

Practical Network Reliability Engineering

John Howard - Head of Network Infrastructure @ Proton AG

Applying NRE principles in a real network for fun and profit!

AUTOCON 5

THE NETWORK AUTOMATION CONFERENCE

whoami



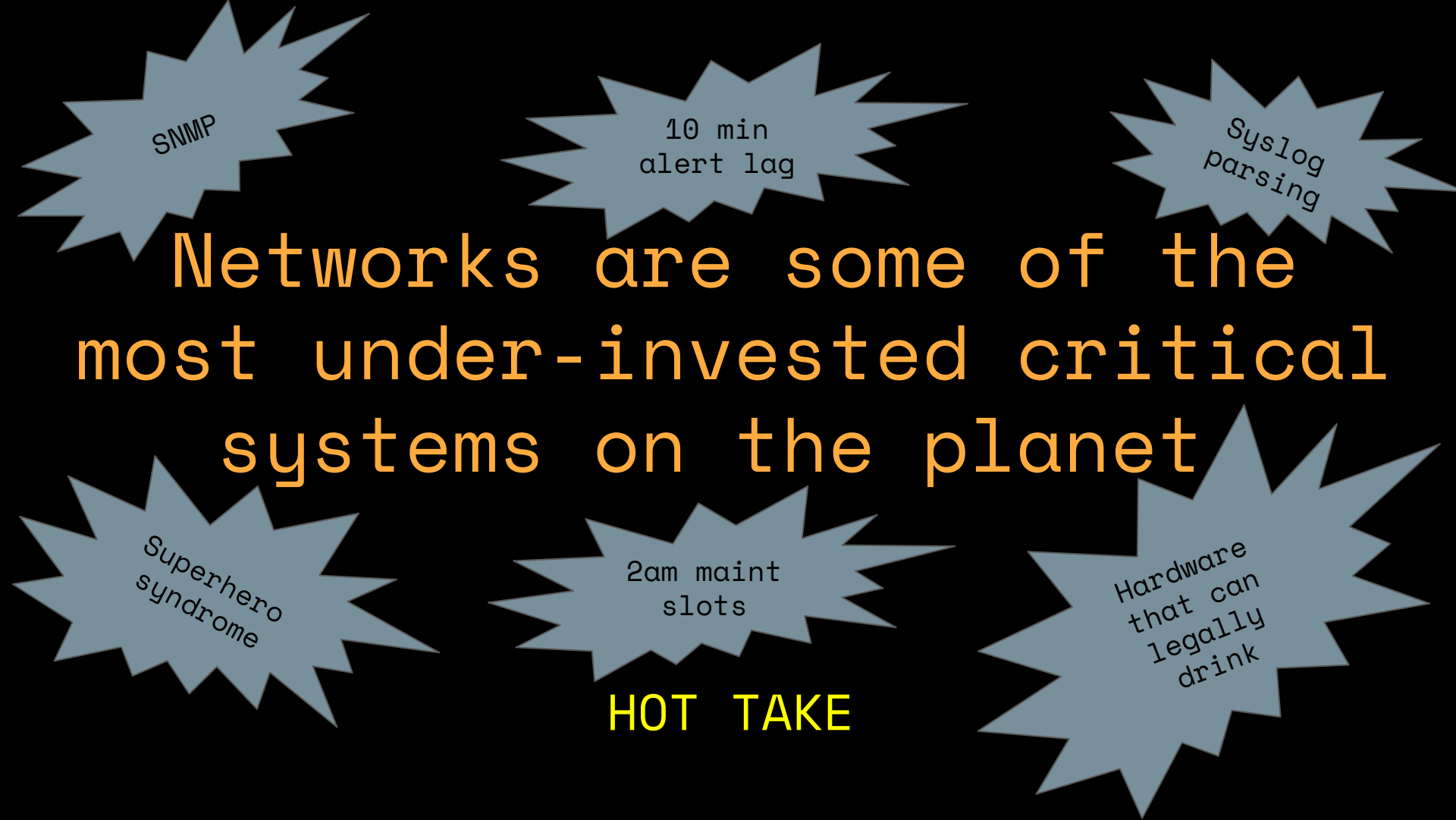
John Howard (@fatred)

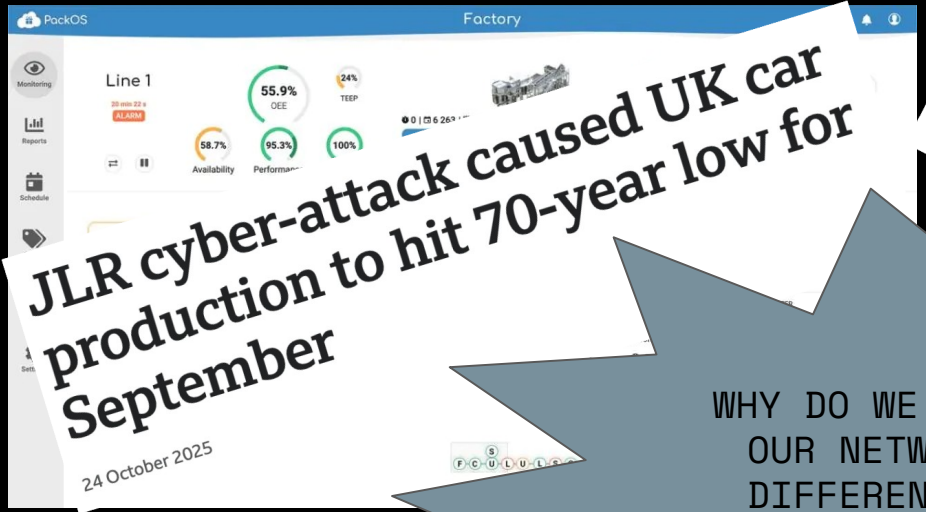
Head of Network Infra @ Proton (home to ProtonMail, VPN, Pass, Drive and Docs)

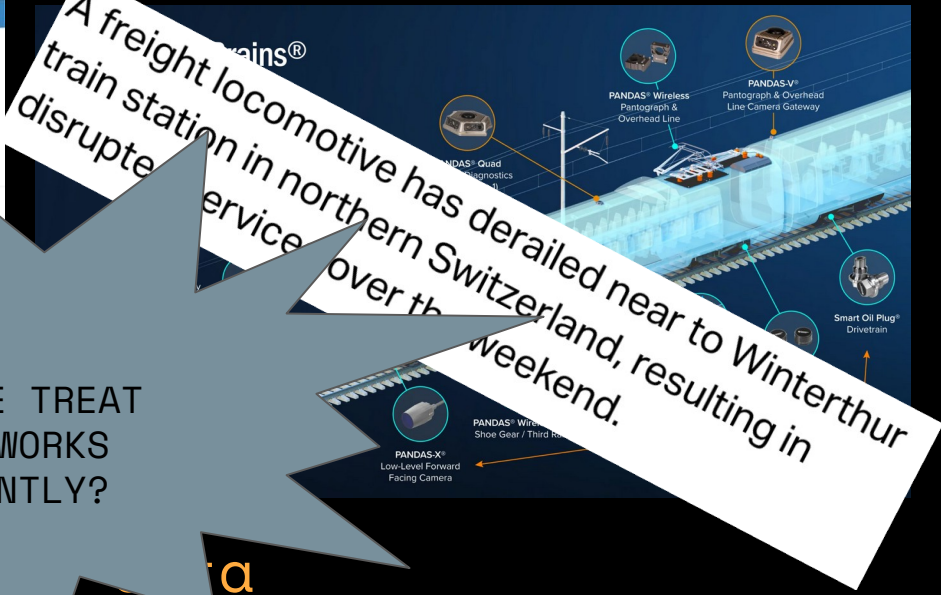
20y+ “experience” automating networks (with varying success)

Professional ranter

#OfficeHours-EMEA co-host (with Bart Dorlandt)


SNMP
10 min alert lag
Syslog parsing
Networks are some of the most under-invested critical systems on the planet
Superhero syndrome
2am maint slots
Hardware that can legally drink
HOT TAKE





WHY DO WE TREAT OUR NETWORKS DIFFERENTLY?

[illegible]a
[illegible]sed tooling
Persistent high utilisation
Reliable forecasting

HOT TAKE

What is this NRE buzzword?

- First(?) defined by Matt Oswalt and James Kelly (then of Juniper) at ONS 2018
- Merge of ideas from DevOps and Google's SRE
- Pivots the mindset from monitoring and operating on individual network components to services and availability



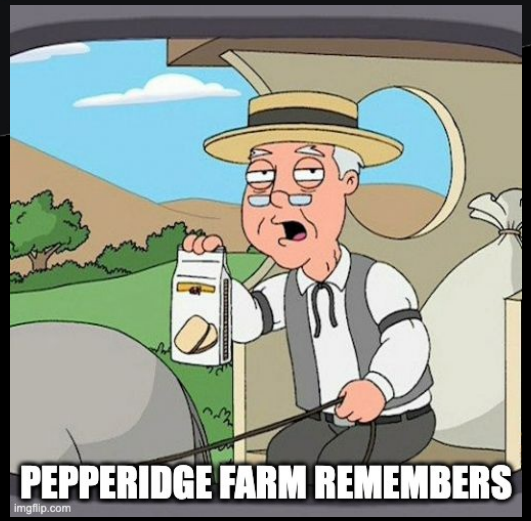
The internet speaks…

Cheeze_It • 7y ago

Those are just clickbaity buzzwords people like to throw around

An NRE is 1/2 network engineer, 7/8 scripter, 1/8 software engineer.

5



…ish

Why did it disappear?

- Accusations of Marketecture:
  - Juniper centric (cynical?)
  - Automation heavy (in 2018…)
  - NetEng are CLI addicts (see Andy @ ac4)
- “move fast and break things” is antithetical to network operations
- SLI/SLO concepts were poorly defined and hard to implement in SNMP era

Why did it disappear?



Fi… …ere you automate it

late night throwing … waste troubleshooting sessions.

…work-reliability-engineering-more-than-automation/



Think system no…

https://rule11.tech/is-it-planning-or-just-…

AUTOMATING CHAOS IS BUSY WORK

My definition of NRE

- Focus on capabilities & services, not devices & protocols
- Engineer resilience in, to make failure normal
- Leverage telemetry to *observe* the network, not *react*
- Iterate complexity & snowflakes OUT of the platform
- Accelerate data driven decision making
- Observability & Assurance don’t require SWEs

NETWORK AUTOMATION FORUM

I thought you said this was practical?



Service Composition

Service:

- I have a 100G wavelength from DC1 to DC2 spanning 1000km
- I have a 100G wavelength from DC1 to DC2 spanning 800km

Capability:

- I can deliver packets from DC1 to DC2
- I have 100G max redundant capacity from DC1 to DC2
- I have 8-10ms latency expectation from DC1 to DC2

Takeaway:

- Capabilities translate well to Service Level Indicators
- Services are “thing” that provide & constrain the capability
- I can build SLOs for capabilities easier than “the service”

```
name: DCI DC1 DC2 MyISP
  rules:
  - record: dci_dc1_dc2_myisp_port_state
    expr: min(interface_operstate{instance="dc1-router1", interface_name="et-0/0/0.0"}) or vector(0)

  - record: dci_dc1_dc2_myisp_crypto_state
    expr: min(macsec_connectivity_association{instance="dc1-router1", ca-name="dc1-dc2-wave"}) or
vector(0)

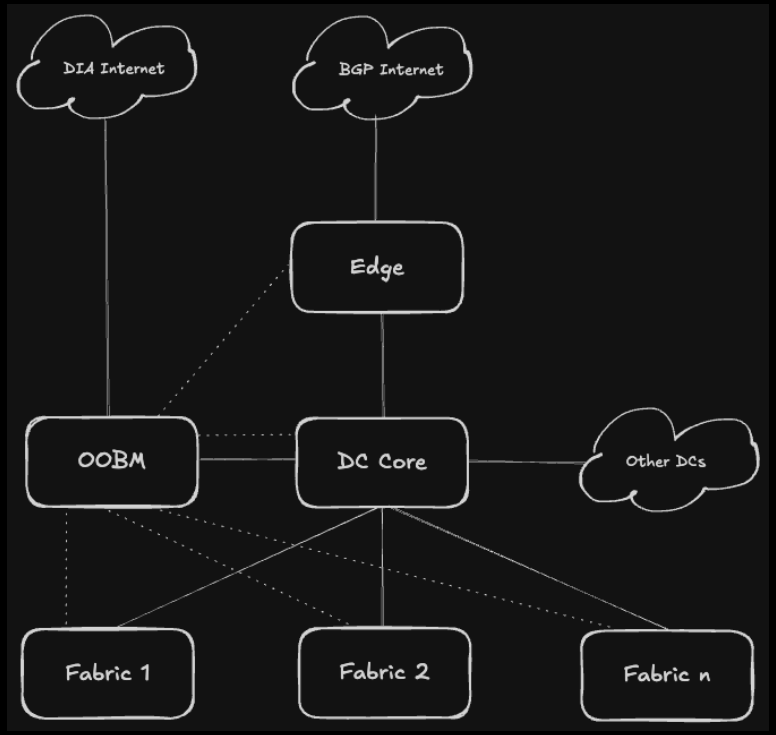
  - record: dci_dc1_dc2_myisp_underlay
    expr: min(router_isis_adjacencies{instance="dc1-router1", interface_name="et-0/0/0.0"} == bool 1)
or vector(0)

  - record: dci_dc1_dc2_myisp_overlay
    expr: min(router_bgp_peer_state{instance="dc1-router1", network_instance="dc1-dc2-wave",
peer_addres="169.254.0.1"} == bool 1) or vector(0)

  - record: dci_dc1_dc2_myisp
    expr: dci_dc1_dc2_myisp_port_state + dci_dc1_dc2_myisp_crypto_state + dci_dc1_dc2_myisp_underlay
+ dci_dc1_dc2_myisp_overlay == bool 3 or vector(0)
```

Design Composition

You can't automate complexity. Simplicity is the prerequisite.



EDGE

Internet boundary - does all the work of getting traffic in and out of the site

DC CORE

L3 transit between fabrics - Defines your scale limits too

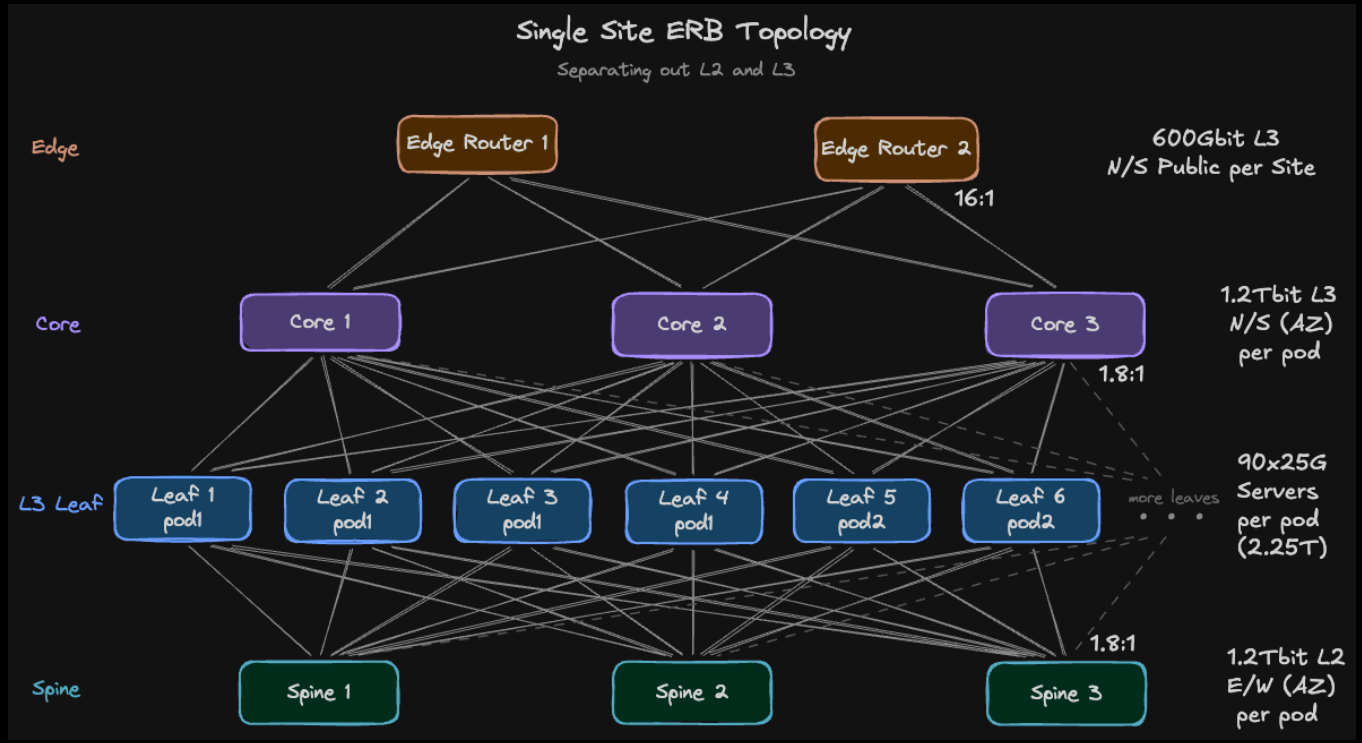
FABRIC

L3 segmentation with VRFs, L2 segmentation with EVPN. Defines the blast radius for a user facing change

OOBM

Control plane access. Diverse hardware and connectivity from that which it manages.

Design Implementation



L2/L3 Split

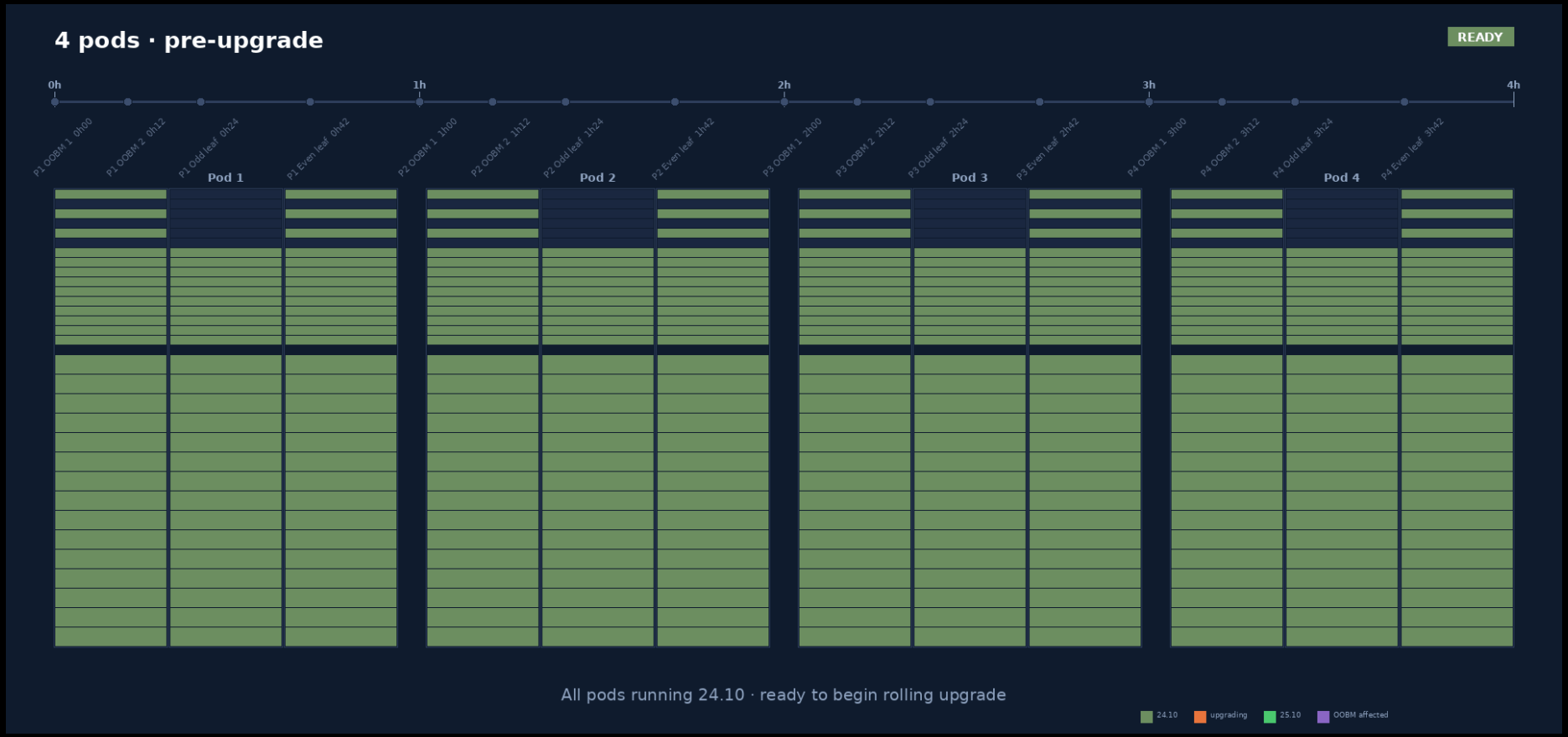
Data workloads are heavy L2 users - sharding/ replication

Tenants (VRFs) have man cross tenant flows

N/S and E/W have same capacity, no sharing

Predictable latency between AZs for tenants

Border leaves suck


4 pods · pre-upgrade
READY
0h
1h
2h
3h
4h
P1 OOBM 1 0h00
P1 OOBM 2 0h12
P1 Odd leaf 0h24
P1 Even leaf 0h42
P2 OOBM 1 1h00
P2 OOBM 2 1h12
P2 Odd leaf 1h24
P2 Even leaf 1h42
P3 OOBM 1 2h00
P3 OOBM 2 2h12
P3 Odd leaf 2h24
P3 Even leaf 2h42
P4 OOBM 1 3h00
P4 OOBM 2 3h12
P4 Odd leaf 3h24
P4 Even leaf 3h42
Pod 1
Pod 2
Pod 3
Pod 4
All pods running 24.10 · ready to begin rolling upgrade
24.10
upgrading
25.10
OOBM affected

Contracts enable controlled disruption

Provider and Consumer model is well established design

Edge < * > Core

Agreements

- Edge provides Core defaults
- Core provides Edge /32 anycast, /21 tenant aggregates, /19 Site summaries
- Both sides always provide N+1 paths
- Edge is sole arbiter of internet path selection.
- Edge provides no L2 bridge domain for internet routed prefixes

Implementation

- Full mesh, eBGP + BFD
- v4 unnumbered over v6lla
- Drain via graceful restart (or just deactivate)
- Anycast or full routed subnet redistributed from fabric to edge

Core < * > Fabric

Agreements

- Core provides Fabric Defaults + summaries for other AZs + DCs
- Fabric provides Core /32 anycast, /21 tenant summaries
- Both sides always provide N+1 paths
- VRF egress traffic is deny by default
- 1ms Cross AZ latency, 30ms Cross site latency

Implementation

- eBGP + BFD + dot1q (per-VRF subifs)
- Loopback via v4 unnumbered over v6lla
- Drain via graceful restart (or just deactivate)
- ACLs function like route leaking

External SLA 99.9% · **Internal target 99.98%**

The Network/App contract

The contract that really matters.

App < * > Network

Agreements

- Never utilise more than 50% of assigned network capacity (p99) for more than 5 mins
- Anycast for HA addressing (no VRRP)
- Services placed diversely (HV/Rack/AZ/Site)
- All services load balanced with healthchecks

Implementation

- Servers L2 Bonded and/or L3 BGP
- Alerting for over-utilisation
- VM/K8s placement policies
- OKRs for non-diverse deployments
- Design reviews prior to new deploys

Network < * > App

Agreements

- Never reduce network capacity by more than 50% at any given time
- Reliably provide internal services
- Predictable port to port latency
- Reliable VRF to VRF connectivity
- Resilient connectivity to other sites
- All connectivity is default deny

Implementation

- ESI for multi chassis bonding/L3 ports
- Anycast gateways on the leaf
- Provide critical services such as DNS/NTP
- Provide latency expectations for DCI and Transit
- ACL Automation to self service policy

External SLA 99.9% · **Internal target 99.98%**

Make failure predictable

Redundancy can't solve everything

Risks left after redundancy

- VLAN / EVPN misconfigs
- Human error in changes
- Unclean hardware failure

Fabric wide blast radius!

Mitigations

- Multiple fabrics per location
- 4eyes review and runbooks for everything
- Canary rollouts with rapid rollback
- Auto isolation on failure

External SLA 99.9% · **Internal target 99.98%**

Tools: GNP-Stack

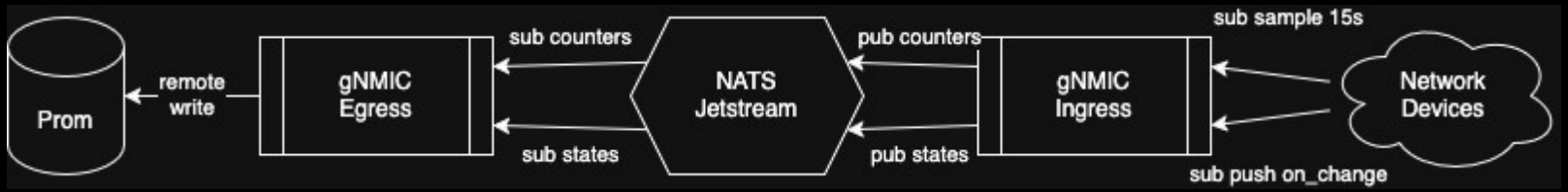
Building blocks:

gNMIc - capture
NATS - buffer
Prometheus - store
Grafana - visualise
Alertmanager - act

Fancy bits:

BMP - routing
IPFIX - traffic
Clickhouse - analysis

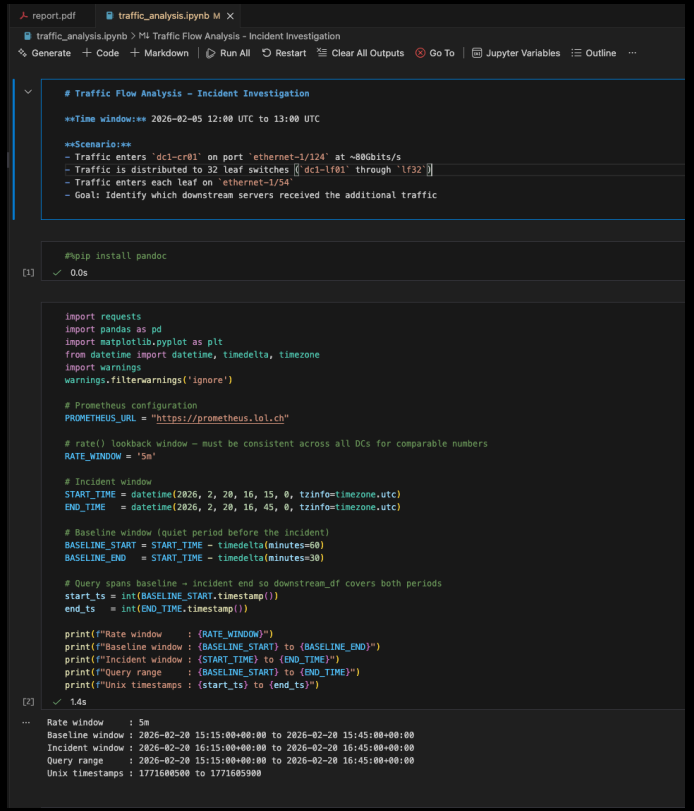
View the OST talk here:



Tools: Jupyter

Just another UI for Python!

Playground that brings data together with code to produce insights



Tools: Sloth

Turns PromQL into an SLI

Produces a burn down chart of SLO decay

Provides at-a-glance view of how much failure you can absorb “free”

Generate some rules at live.sloth.dev:



Not all networking SLIs are “networking”

- Pod - pod (local, cross-AZ, cross-site)
- Number of HTTPS 5xx vs total req
- Auth DNS correct response
- Leaf overlay loopback reachability
- BGP session uptime (per session)
- DNS p99 < 10 ms
- Link packet loss < 0.01% / 24h
- Cross-pod loss < 0.1% / 7d
- Zero uncorrectable FEC errors / day
- ≤ 1 link flap / week
- 99% of 5-min intervals < 45% utilisation

We missed targets early on. That was a feature not a bug - it surfaced which layer needed attention.

NETWORK AUTOMATION FORUM

Where did that get us?



The proof

10x
network capacity since 2021

-40%
per port cost

0
extra headcount

0
out of hours change windows

Call to action

Identify your building blocks. Draw them up. Be intentional.

Capture block requirements. Map protocols to contracts. Write them down.

Get telemetry into Prometheus. Streaming where you can, SNMP where you must.

Track SLIs. Set SLOs with reasonable targets. Maintenance moves to working hours.

Build runbooks for every maintenance task. Bias to telemetry over CLI.

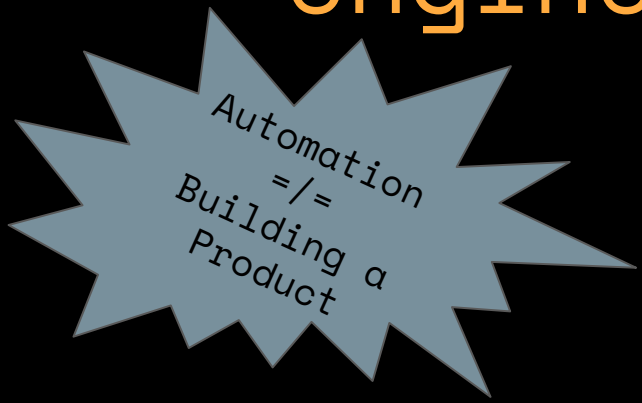
Add alerts for the conditions that caused your last outage.

Build and then Expose your SLI/SLO tooling to consumers.

Let them check network state without opening a ticket.

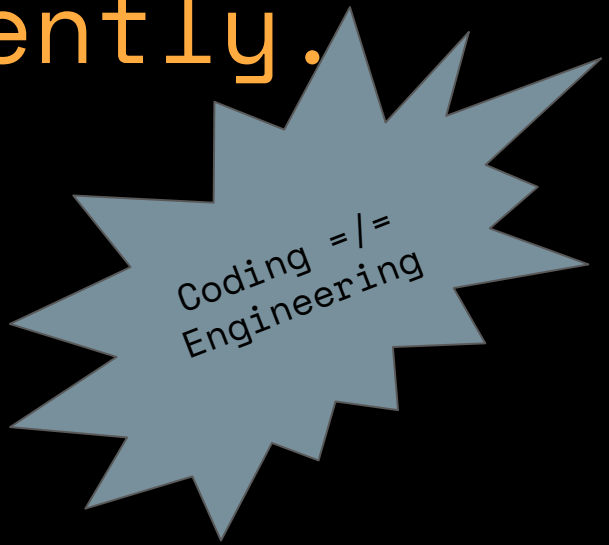
Failure and maintenance are inevitable. Apps must retry with exponential backoff, and provision to allow services to run elsewhere.

You cannot be a Network Engineer and a Software engineer concurrently.

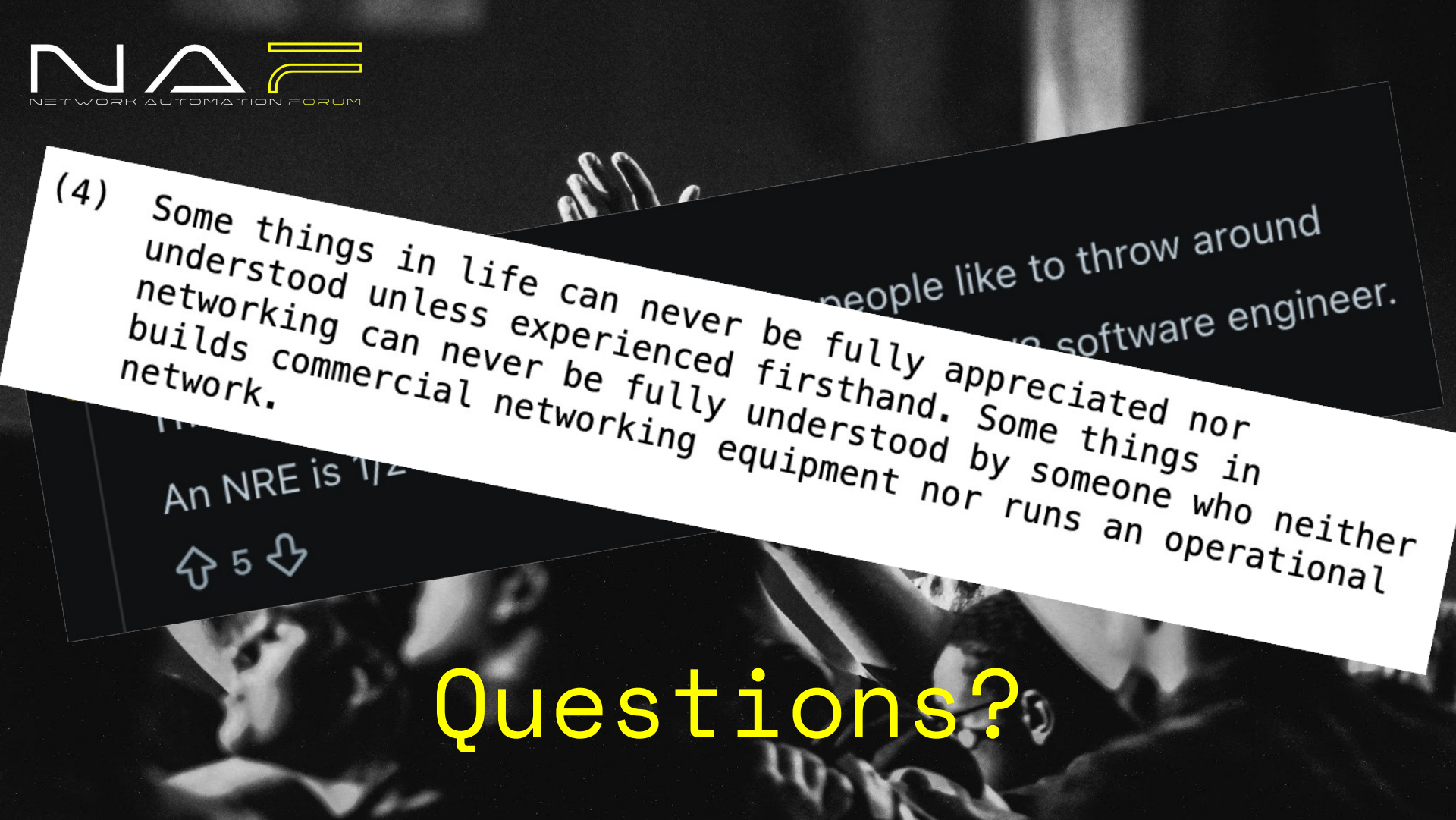


Pick one.

Or. Hire One.



HOT TAKE 2


NAF
NETWORK AUTOMATION FORUM
(4) Some things in life can never be fully appreciated nor understood unless experienced firsthand. Some things in networking can never be fully understood by someone who neither builds commercial networking equipment nor runs an operational network.
people like to throw around
software engineer.
An NRE is
5
Questions?