Evolve or Die

High-Availability Design Principles Drawn from Google's Network Infrastructure

Ramesh Govindan, Ina Minei, Mahesh Kallahalla, Bikash Koley and Amin Vahdat
… and a cast of hundreds at Google

Network availability is the biggest challenge facing large content and cloud providers today

The push towards higher 9s of availability

At four 9s availability

- Outage budget is **4 mins per month**

At five 9s availability

- Outage budget is **24 seconds per month**

How do providers achieve these levels?

By learning from failures

What has Google Learnt from Failures?

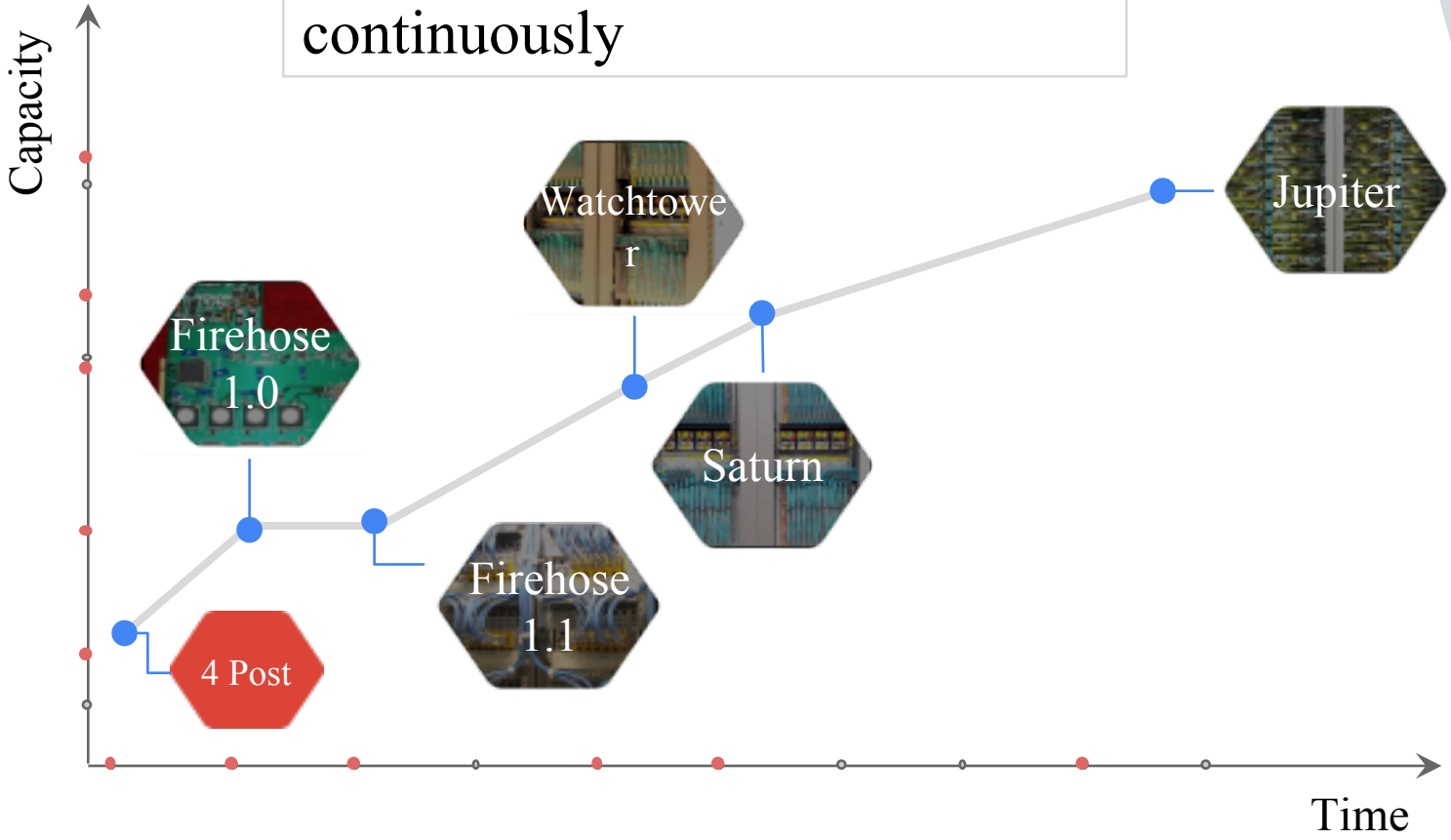
Why is high network availability a challenge?

What are the characteristics of network availability failures?

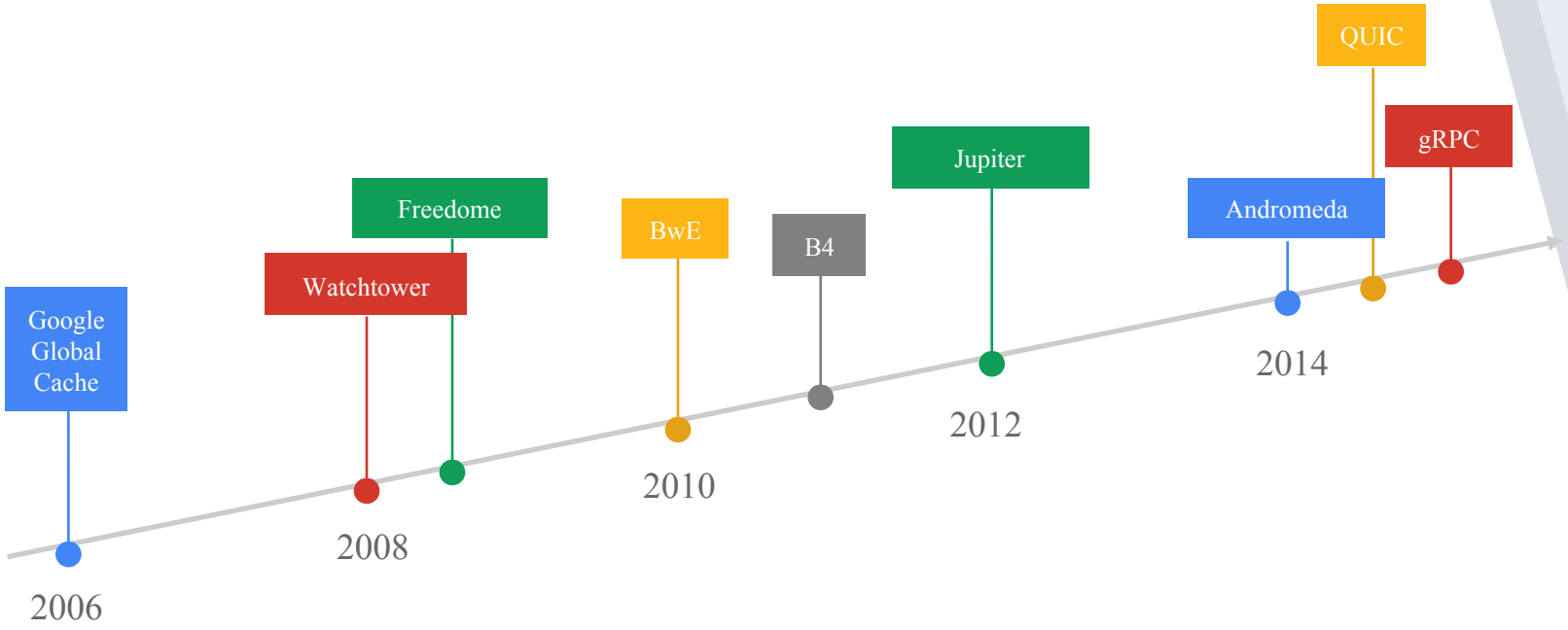
What design principles can achieve high availability?

Why is high network availability a challenge?

Network hardware evolves continuously



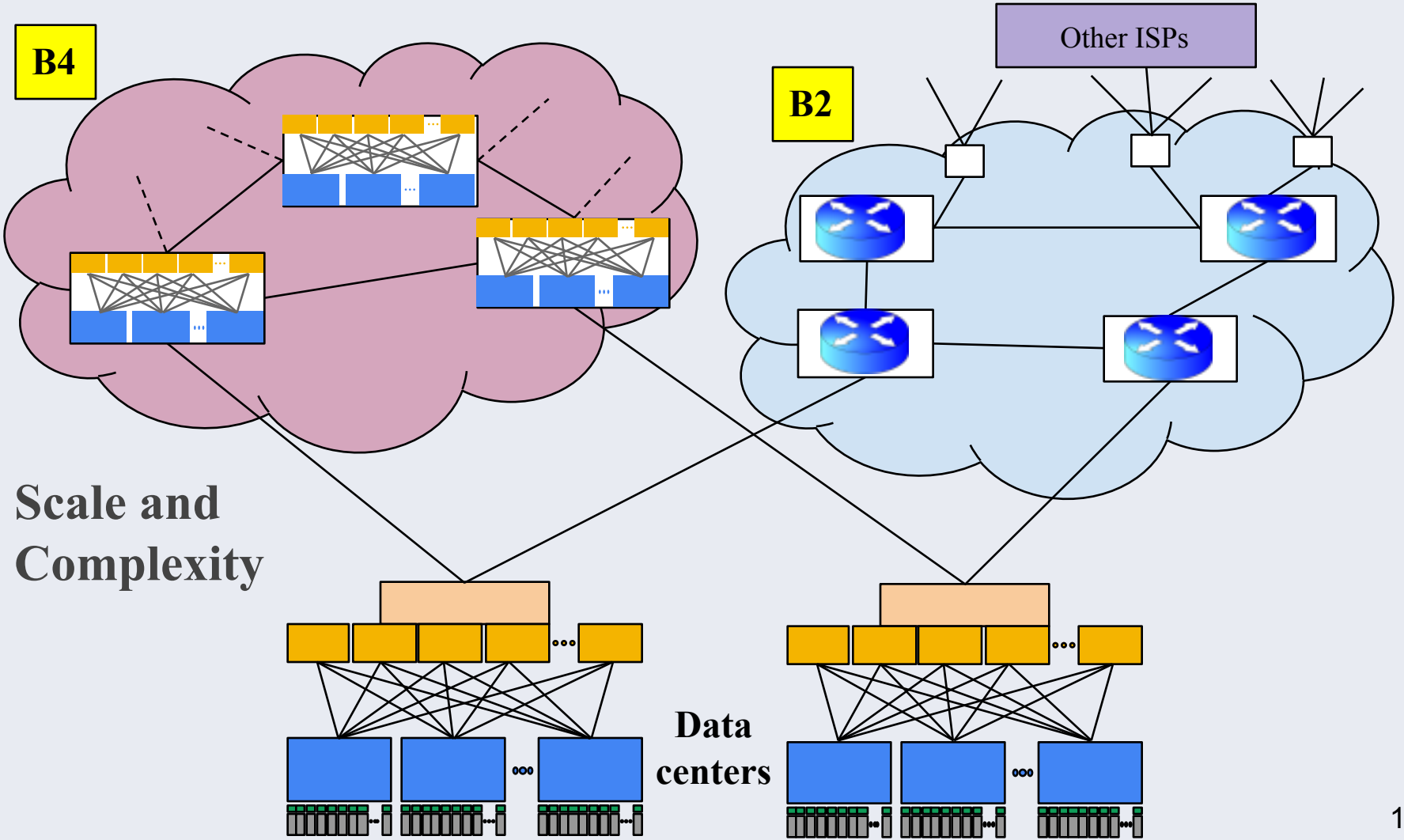
So does network software



New hardware and software can

- Introduce bugs
- Disrupt existing software

Result: Failures!


B4
B2
Other ISPs
Scale and Complexity
Data centers

Design Differences

B4 and Data Centers

- Use merchant silicon chips
- Centralized control planes

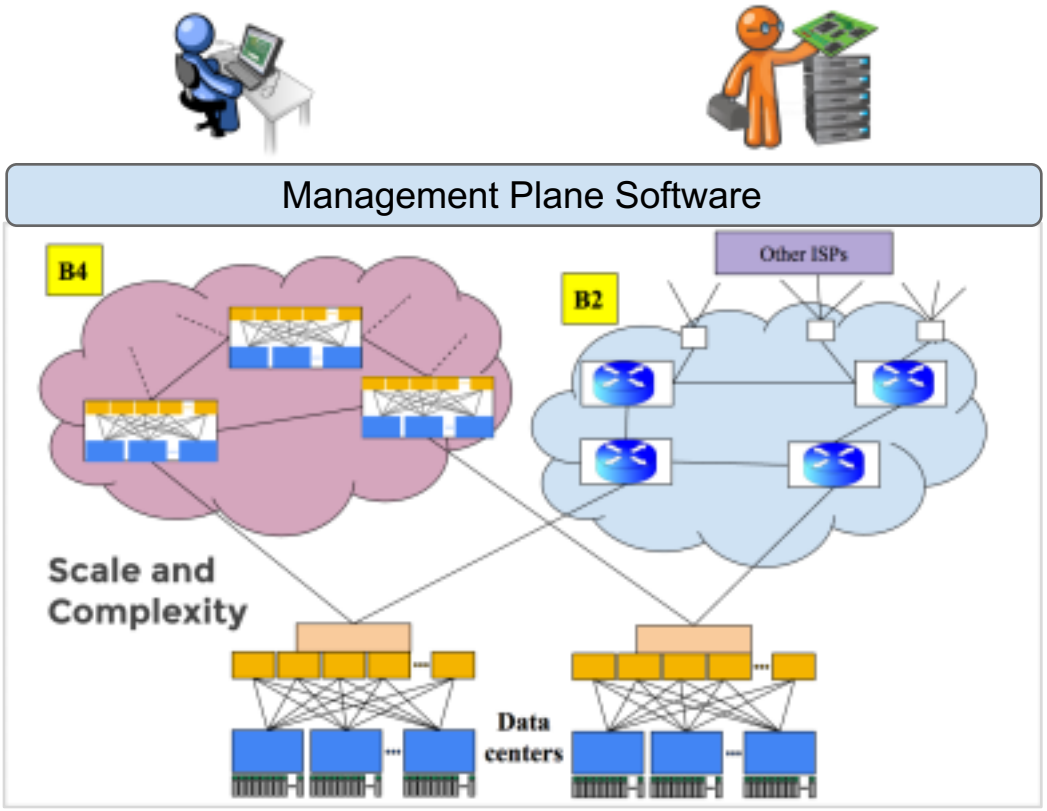
B2

- Vendor gear
- Decentralized control plane

Design Differences

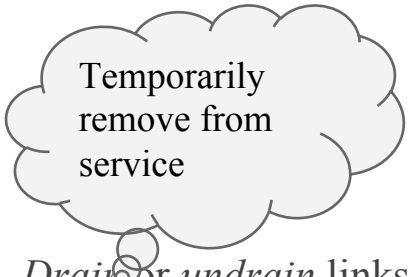
These differences increase management complexity and pose availability challenges

The Management Plane



Manages network evolution

Management Plane Operations



Connect a new data center to B2 and B4

Upgrade B4 or data center control plane software

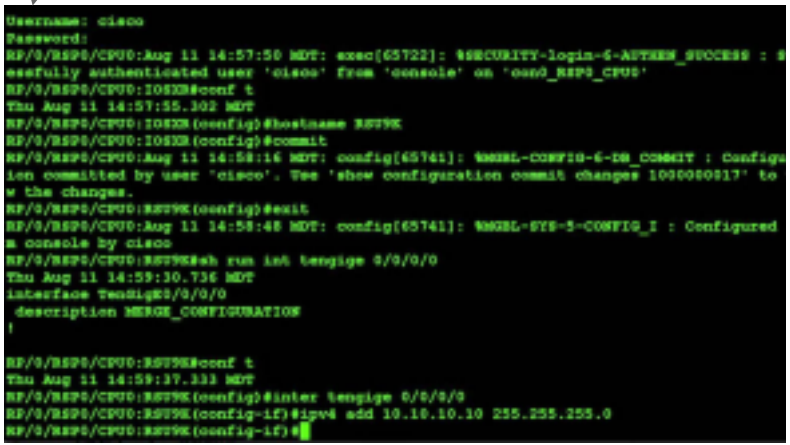
Drain or *undrain* links, switches, routers, services

Many operations require multiple steps and can take hours or days

Low-level abstractions for management operations

- Command-line interfaces to high capacity routers

A small mistake by operator can impact a large part of network



Why is high network availability a challenge?

What are the characteristics of network availability failures?

Key Takeaway

Content provider networks evolve rapidly

The way we manage evolution can impact availability

We must make it easy and safe to evolve the network *daily*

We analyzed over 100 **Post-mortem** *reports written over a 2 year period*

Blame-free process

What is a Post-mortem?

Carefully curated description of a ***previously unseen*** failure that had ***significant availability impact***

Helps learn from failures

What a Post-Mortem Contains

Description of failure, with detailed timeline

Root-cause(s) **confirmed by reproducing the failure**

Discussion of fixes, follow up action items

Examples

- Entire control plane fails
- Upgrade causes backbone traffic shift
- Multiple top-of-rack switches fail

Impact

- Data center goes offline
- WAN capacity falls below demand
- Several services fail concurrently

Key Quantitative Results

70% of failures occur when management plane operation is in progress

Evolution impacts availability

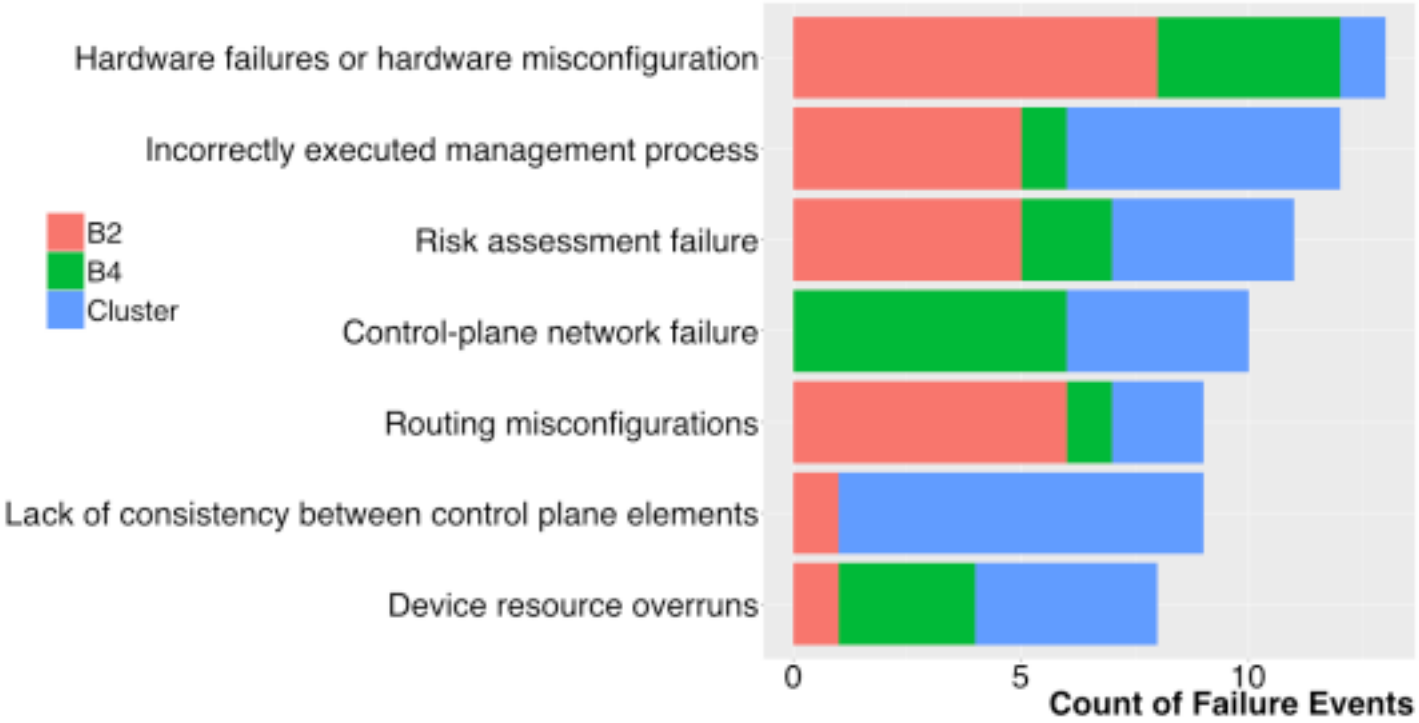
Failures are everywhere: all three networks and three planes see comparable failure rates

No silver bullet

80% of failure durations between 10 and 100 minutes

Need fast recovery

Lessons learned from root causes motivate **availability design principles**

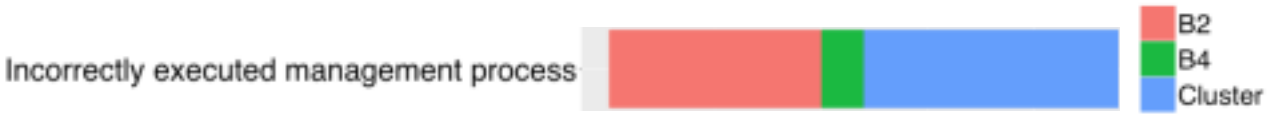


Why is high network availability a challenge?

What are the characteristics of network availability failures?

What design principles can achieve high availability?

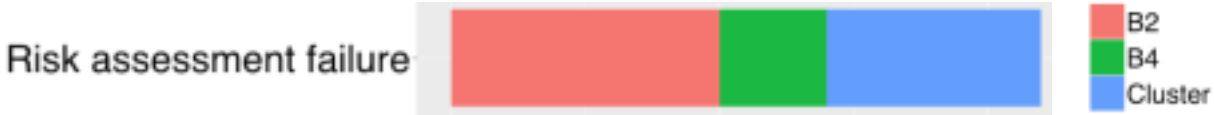
Re-think the Management Plane



Operator types wrong CLI command, runs wrong script

Backbone router fails

Minimize Operator Intervention



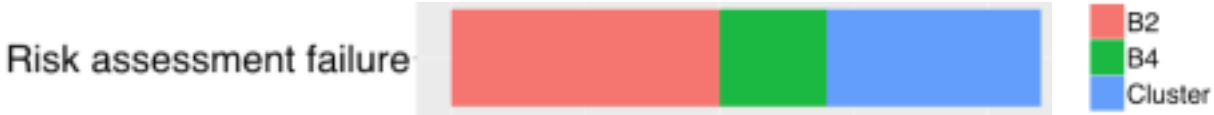
Necessary for upgrade-in-place

To upgrade part of a large device…

- Line card, block of Clos fabric

… proceed while rest of device carries traffic

- Enables higher availability



Risky!

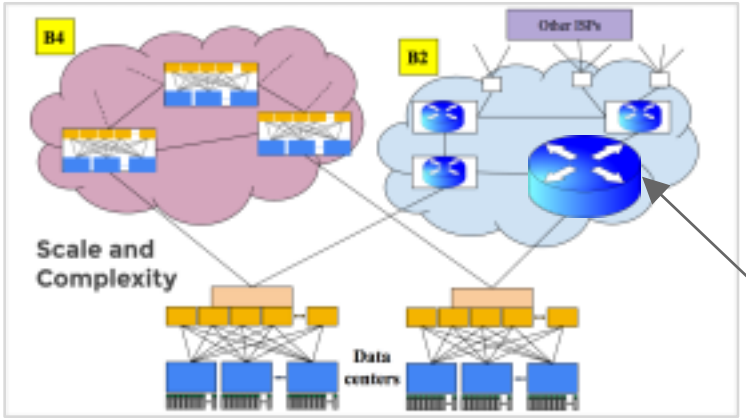
Ensure *residual capacity* > *demand*

Early risk assessments were **manual**

High packet loss

Assess risk continuously

Re-think the Management Plane



"Intent"

I want to upgrade this router

Management Plane Software

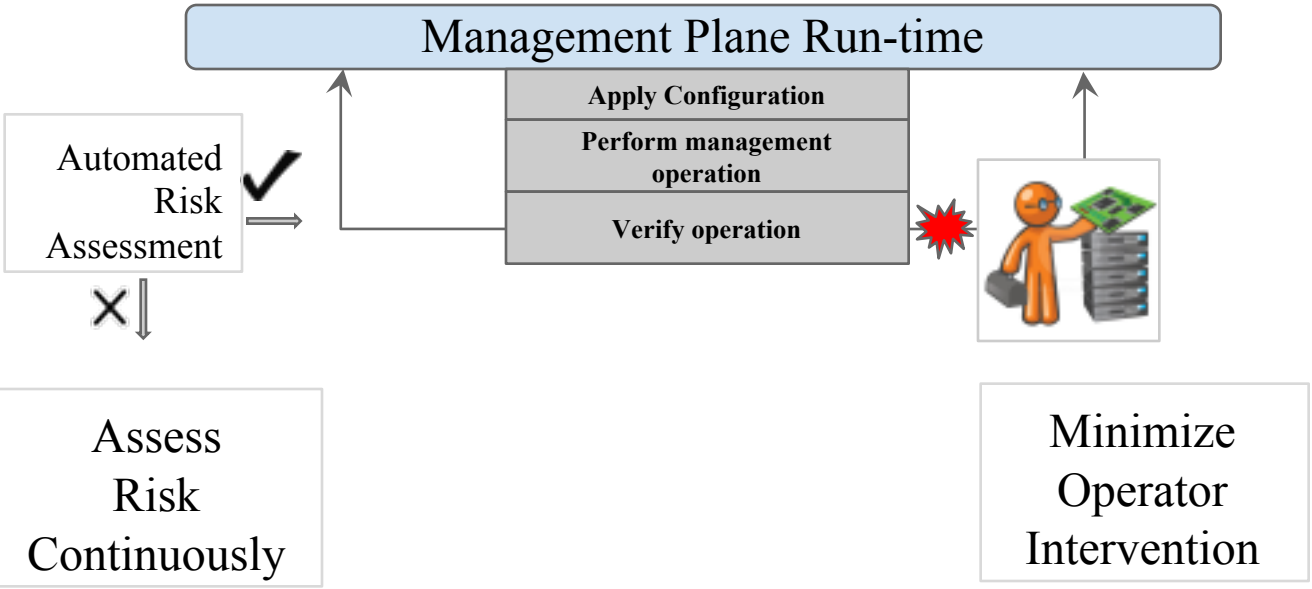
Management Operations

Device Configurations

Tests to Verify Operation

Re-think the Management Plane

Management Operations	Device Configurations	Tests to Verify Operation



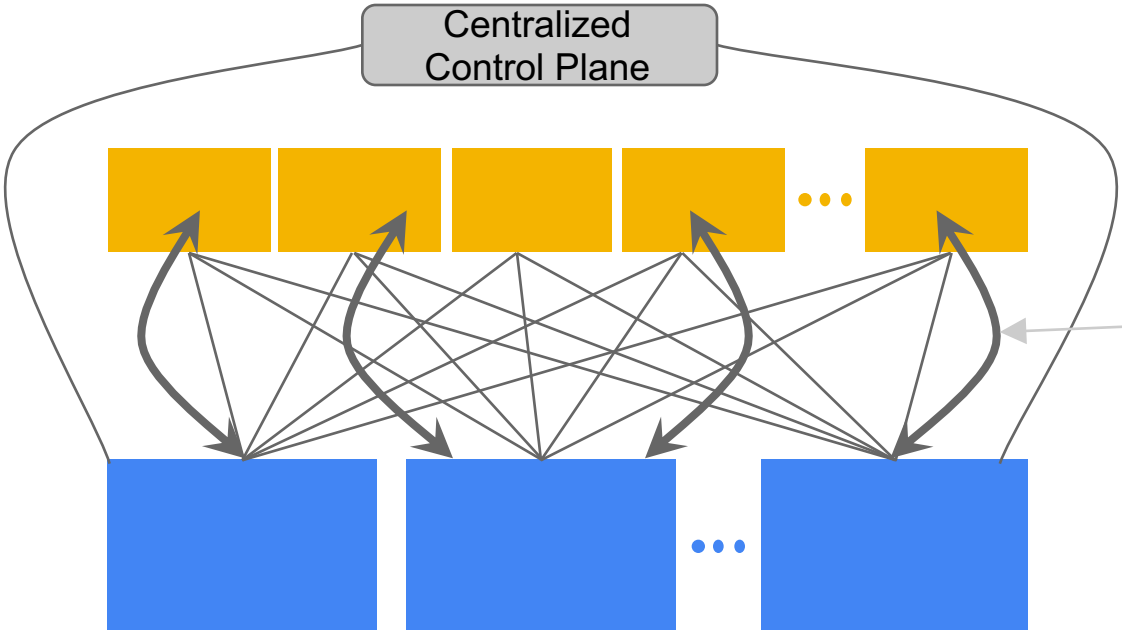
*Avoid **and** Mitigate Large Failures*

B4 and data-centers have dedicated *control-plane network*

- Failure of this can **bring down entire control plane**

Contain failure radius

Fail open



Exceedingly tricky!

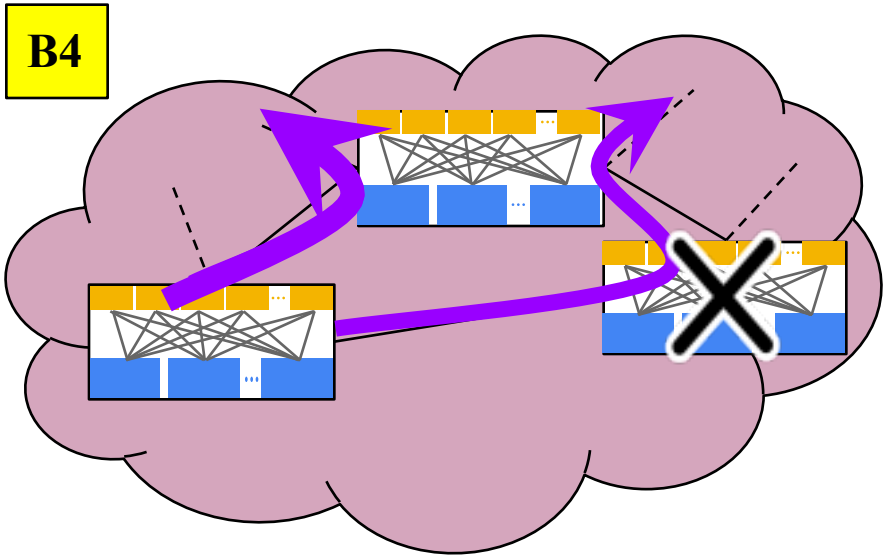
Preserve forwarding state of *all* switches

- *Fail-open* the **entire data center**

A **bug** can cause state inconsistency between control plane components

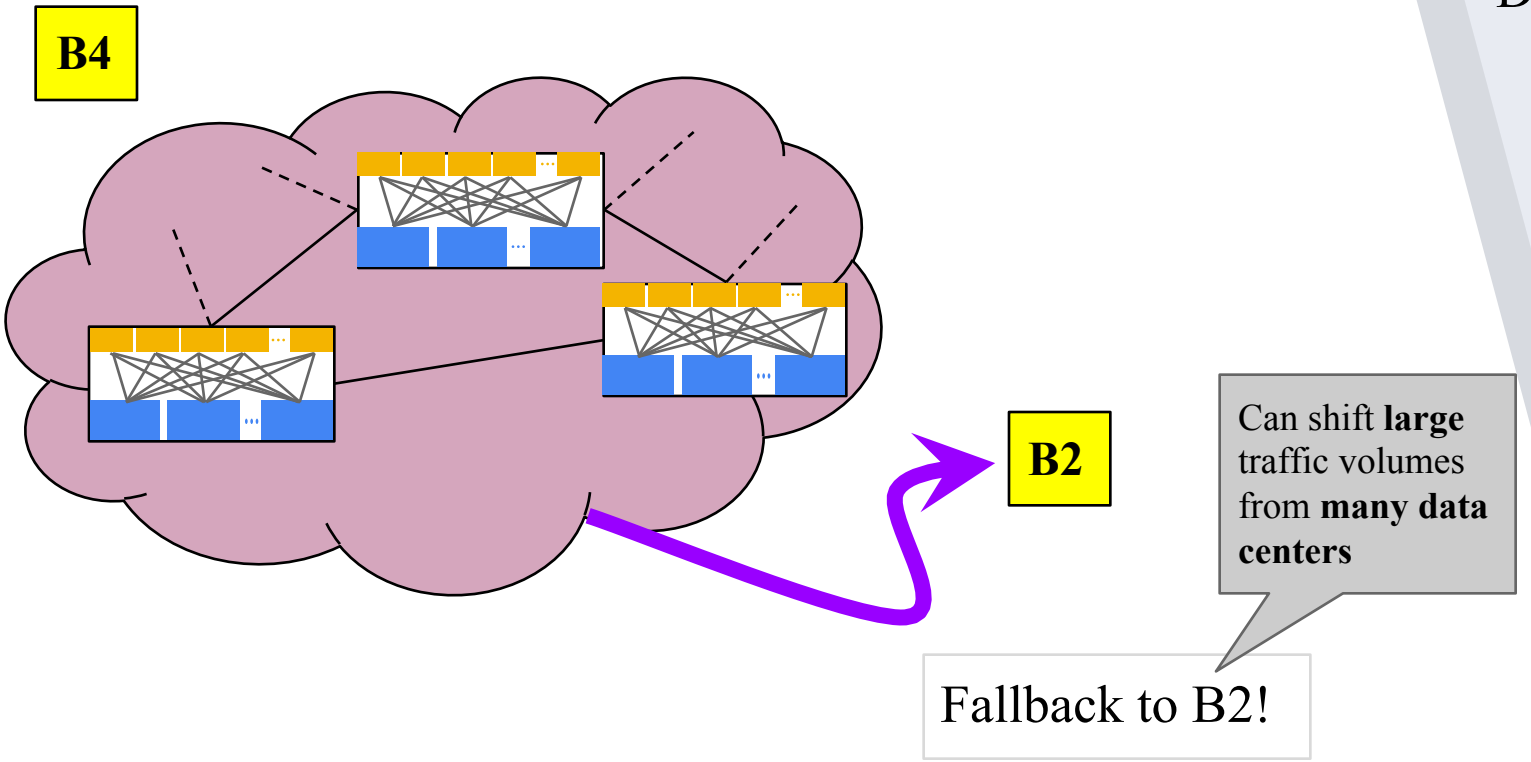
➔ **Capacity reduction** in WAN or data center

Design fallback strategies



A large section of the WAN fails, so **demand exceeds capacity**

Design Fallback Strategies



When centralized traffic engineering fails...

- … **fallback to IP routing**

Big Red Buttons

- For every new software upgrade, design controls so operator can initiate fallback to “safe” version

Evolve or Die!

We cannot treat a change to the network as an **exceptional** *event*

Make change the *common case*

Make it easy and safe to evolve the network *daily*

- Forces management automation
- Permits small, **verifiable** changes

Content provider networks evolve rapidly

The way we manage evolution can impact availability

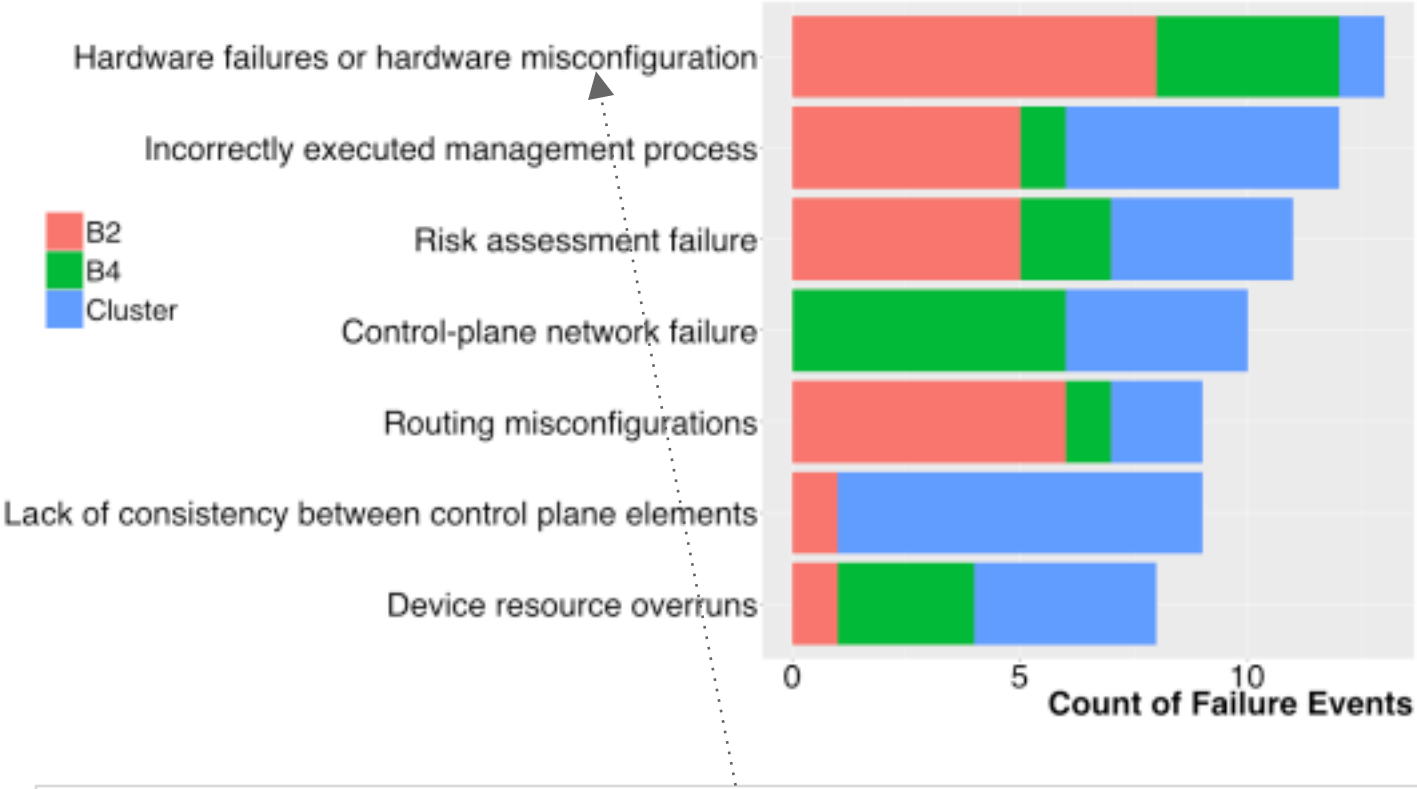
We must make it easy and safe to evolve the network *daily*

Evolve or Die

High-Availability Design Principles Drawn from Google's Network Infrastructure

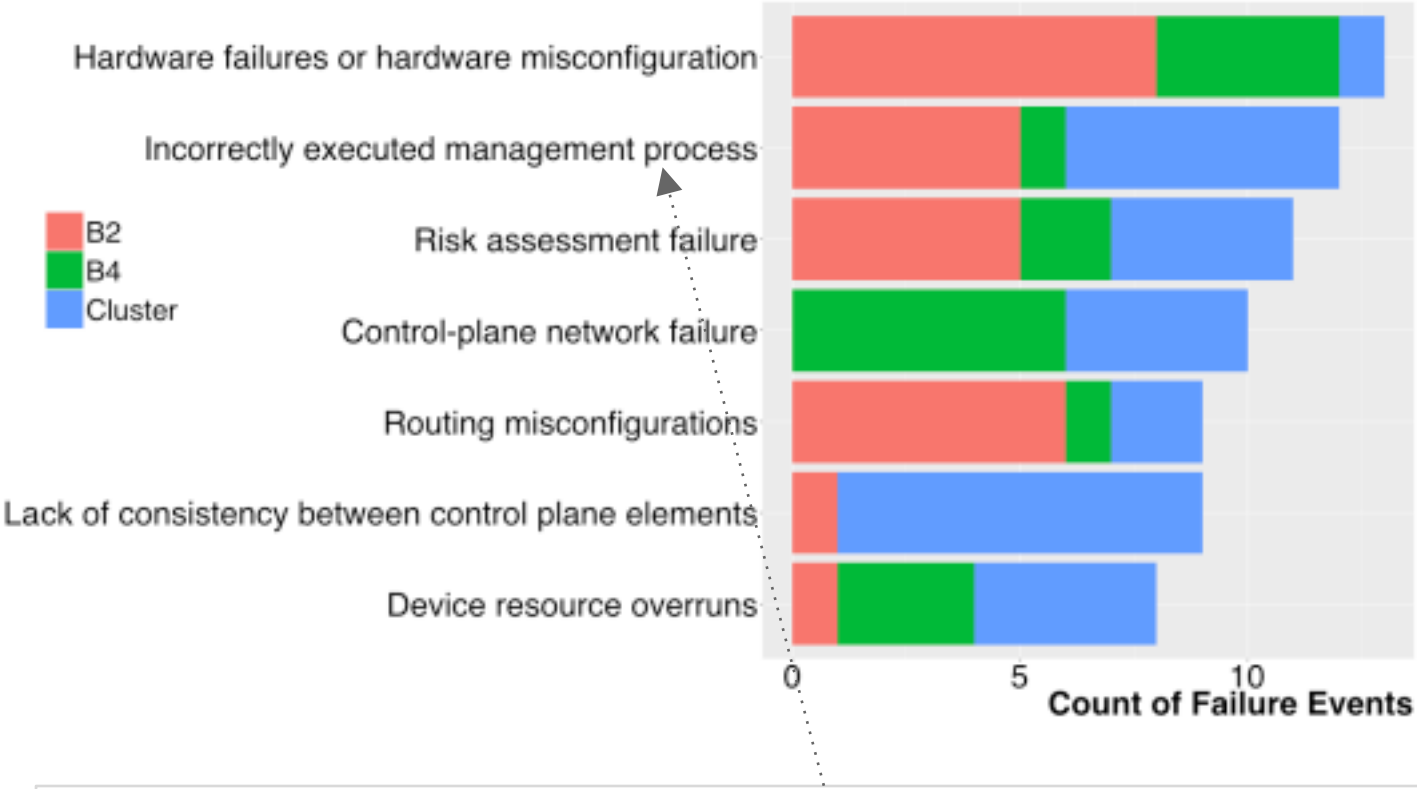
Older Slides

Popular root-cause categories



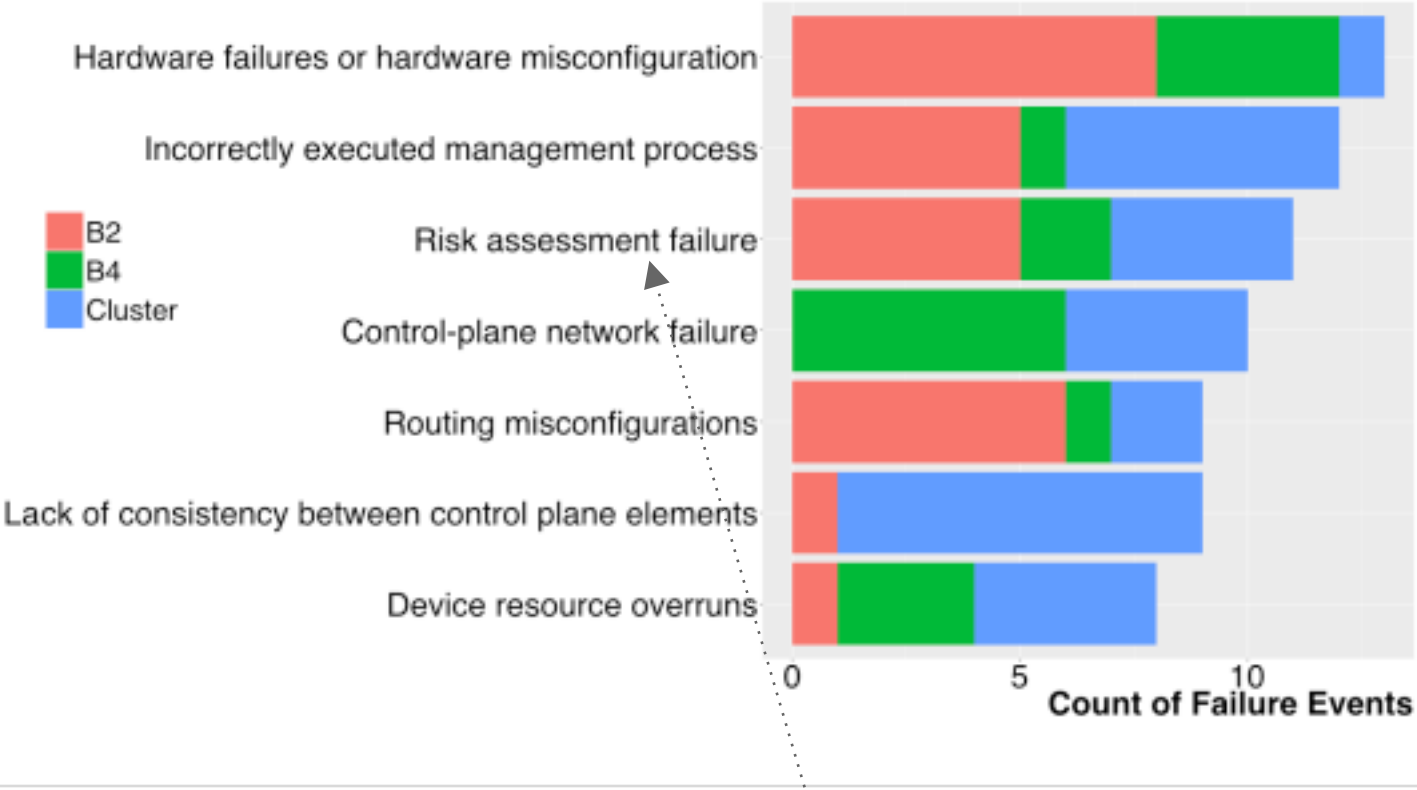
Cabling error, interface card failure, cable cut….

Popular root-cause categories



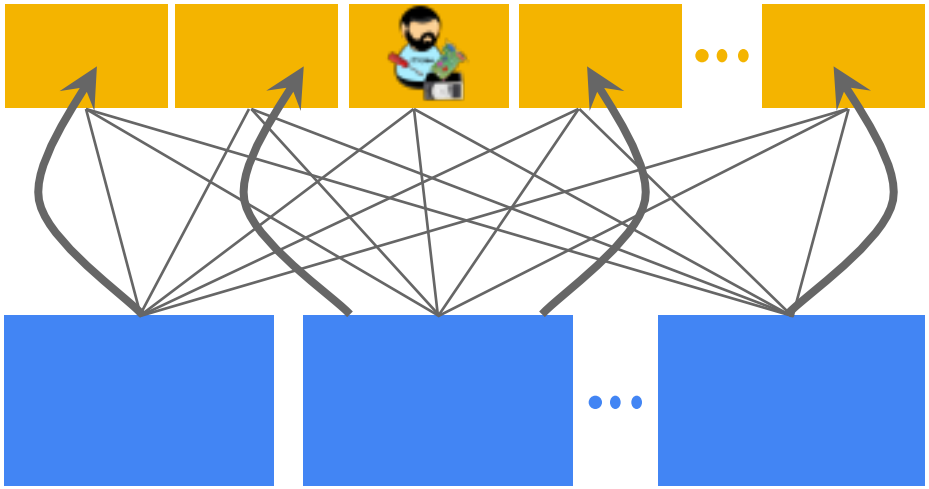
Operator types wrong CLI command, runs wrong script

Popular root-cause categories

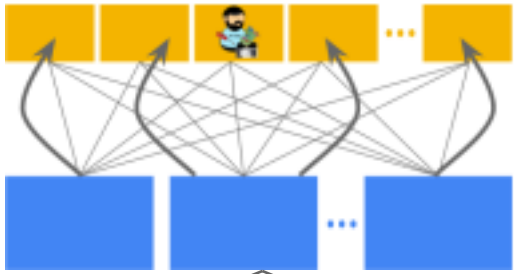


Incorrect demand or capacity estimation for upgrade-in-place

Upgrade in place

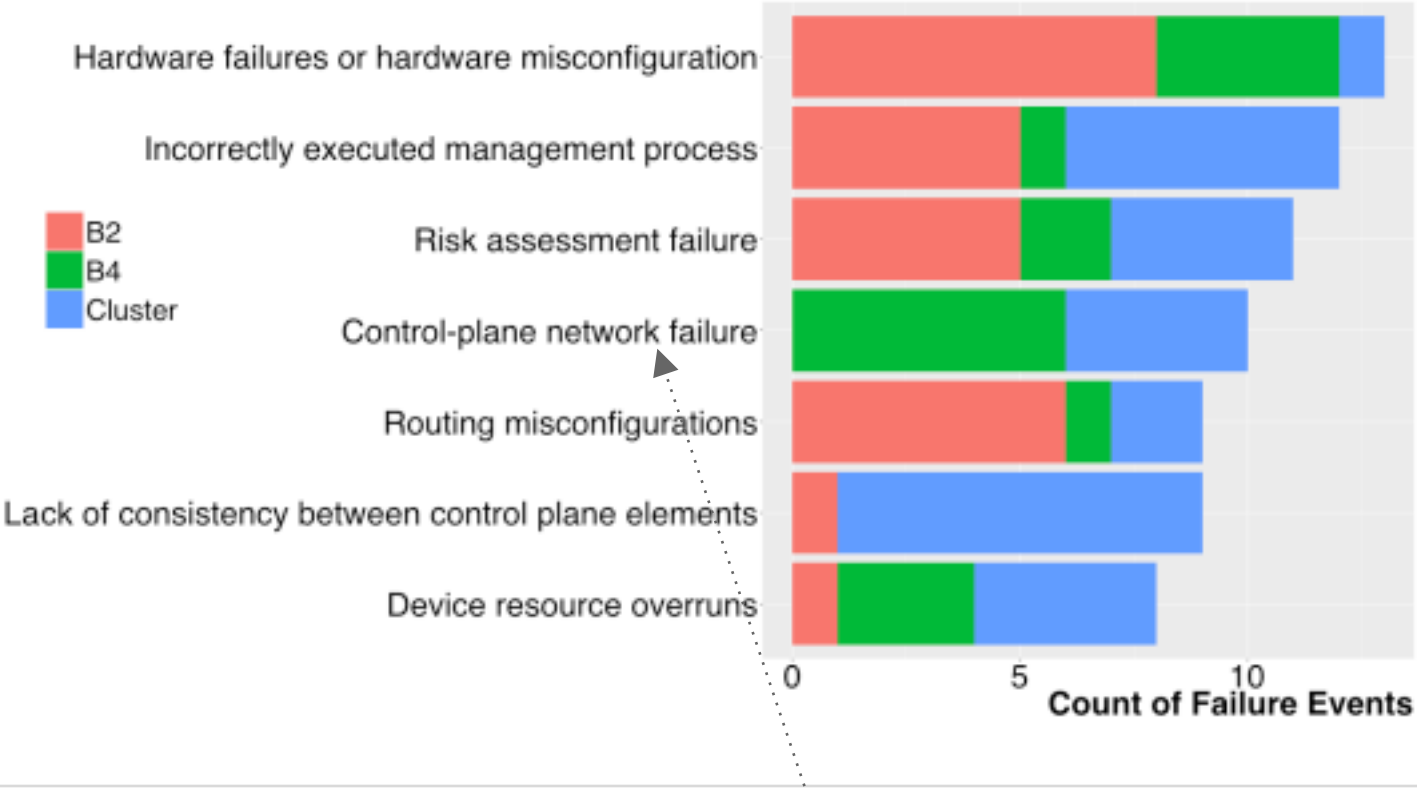


Assessing Risk Correctly



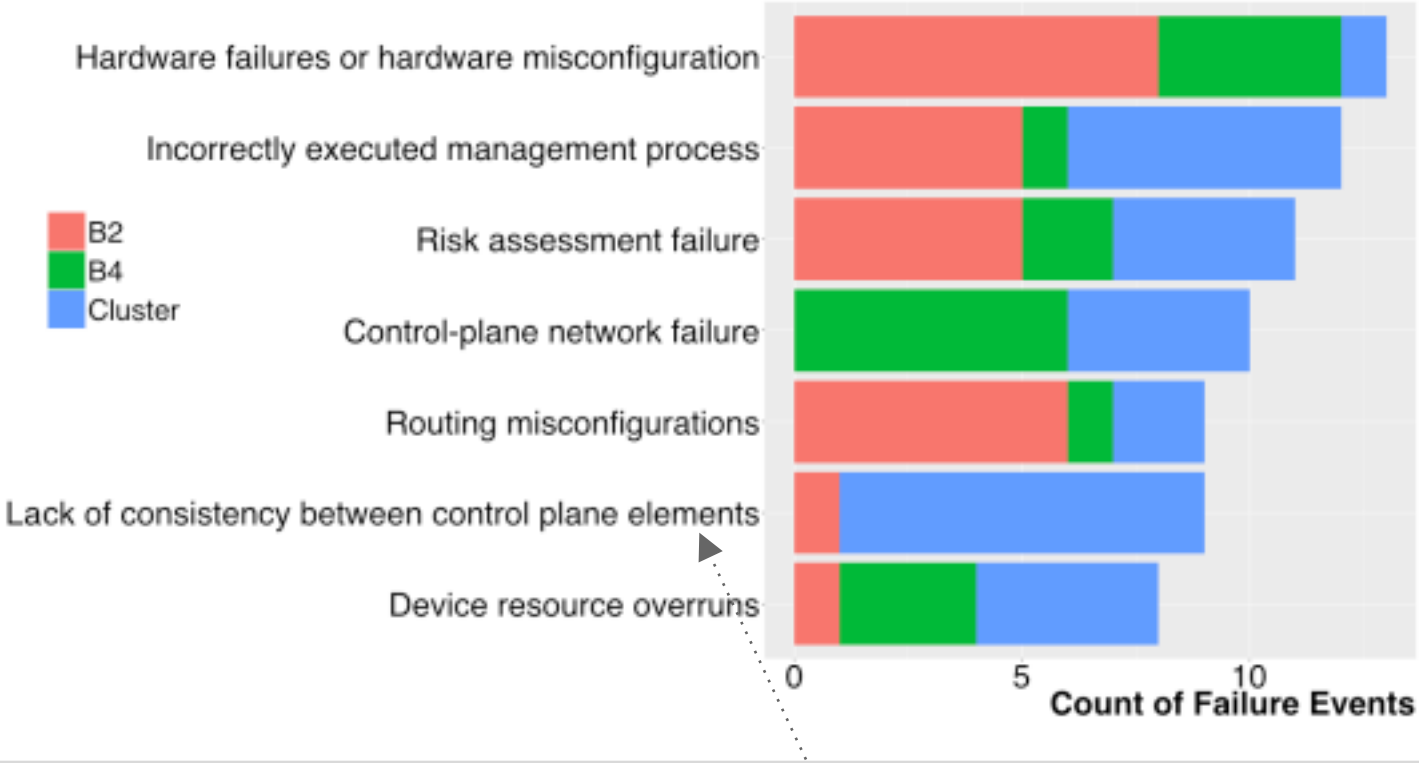
Residual Capacity?	Demand?
Varies by interconnect	Can change dynamically

Popular root-cause categories



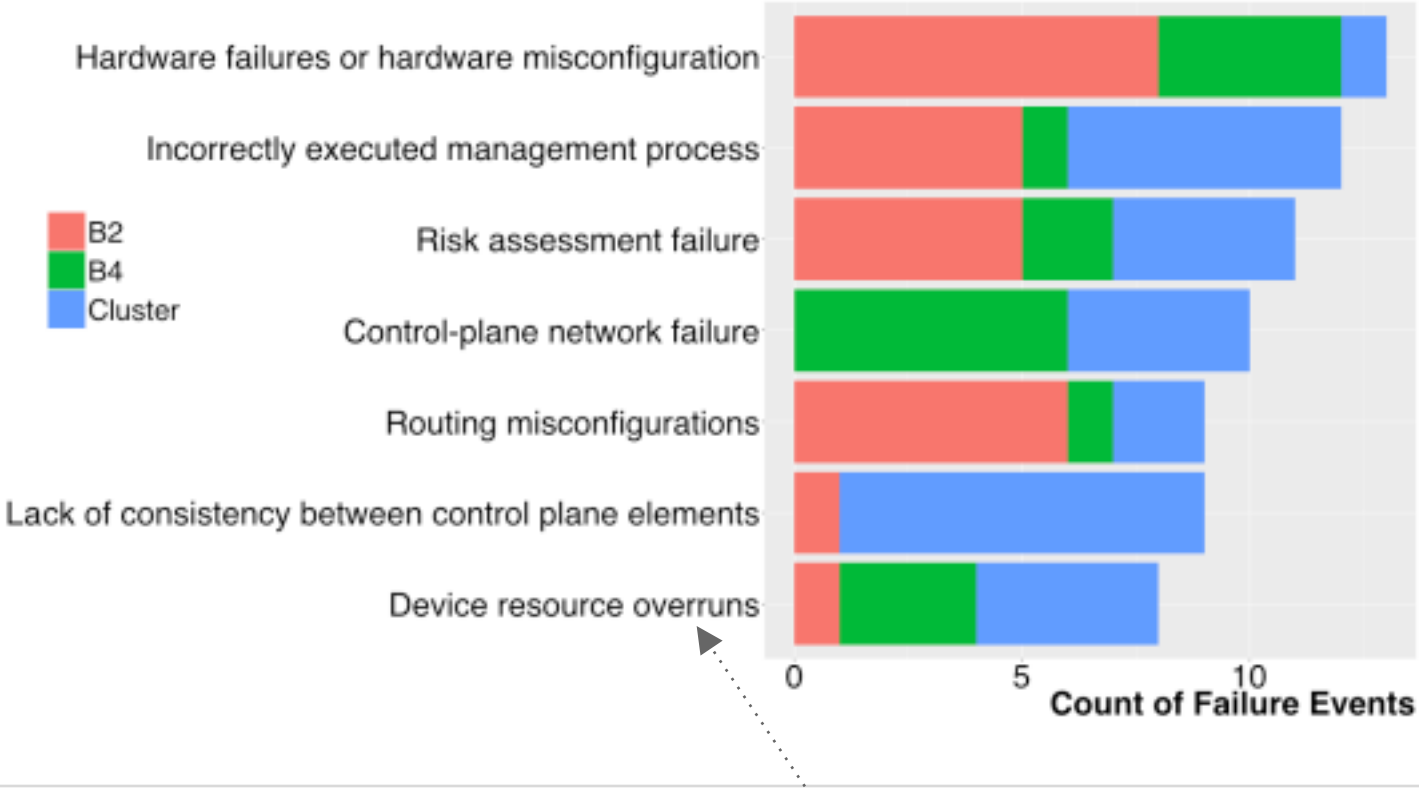
Hardware or link layer failures in control plane network

Popular root-cause categories



Two control plane components have inconsistent views of control plane state, caused by bug

Popular root-cause categories



Running out of memory, CPU, OS resources (threads)...

Lessons from Failures

The role of evolution in failures

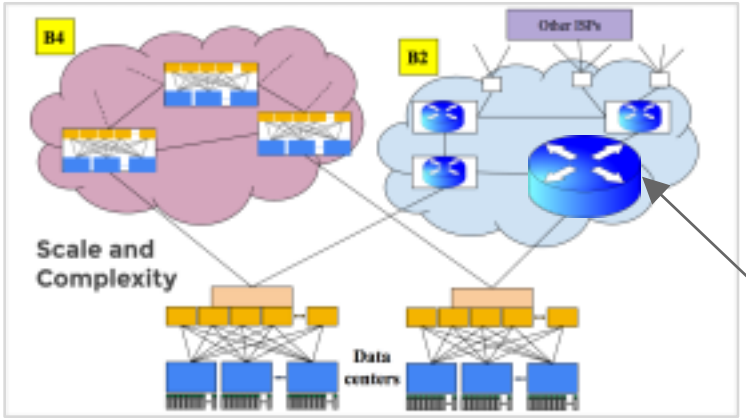
- Rethink the Management Plane

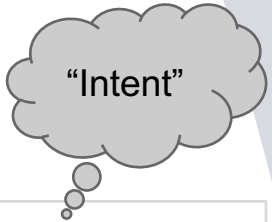
The prevalence of large, severe, failures

- Prevent and mitigate large failures

Long failure durations

- Recover fast



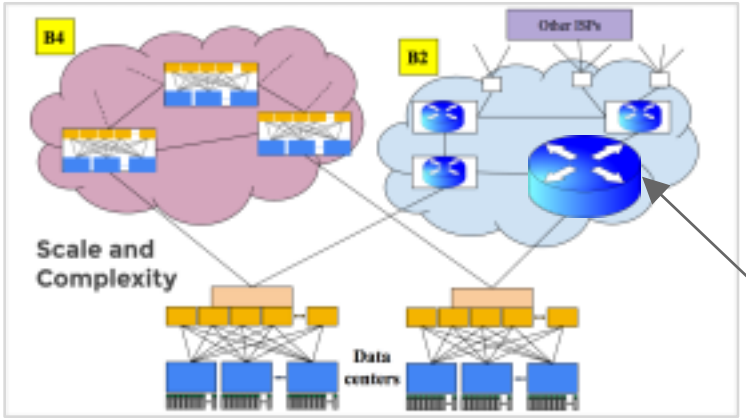


I want to upgrade this router

Why is this difficult? Modern high capacity routers:

- Carry Tb/s of traffic
- Have hundreds of interfaces
- Interface with associated optical equipment
- Run a variety of control plane protocols: MPLS, IS-IS, BGP all of which have network-wide impact
- Have high capacity fabrics with complicated dynamics
- Have configuration files which run into 100s of thousands of lines

High-level Management Plane Abstractions



"Intent"

I want to upgrade this router

Management Plane Software

Management Operations

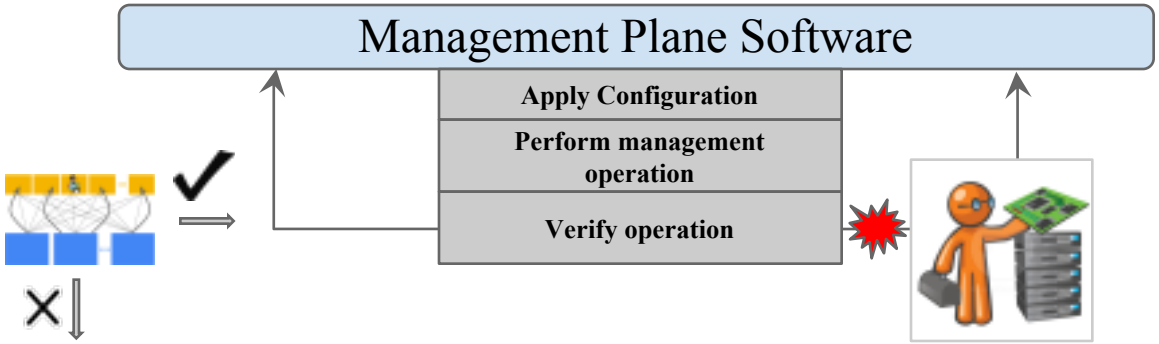
Device Configurations

Tests to Verify Operation

Management Operations

Device Configurations

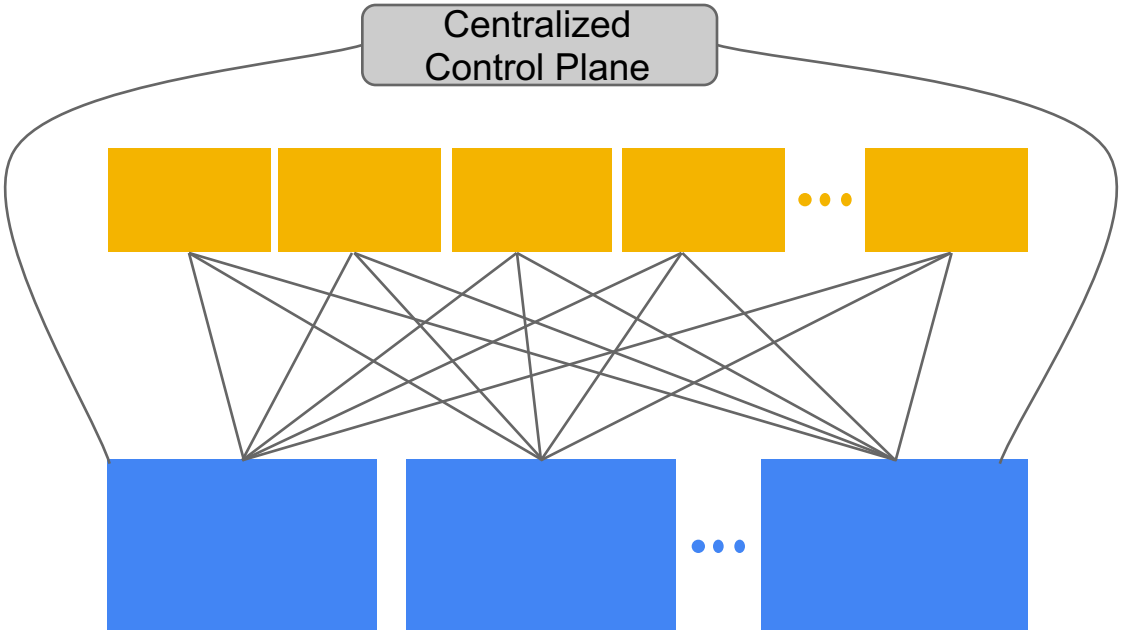
Tests to Verify Operation



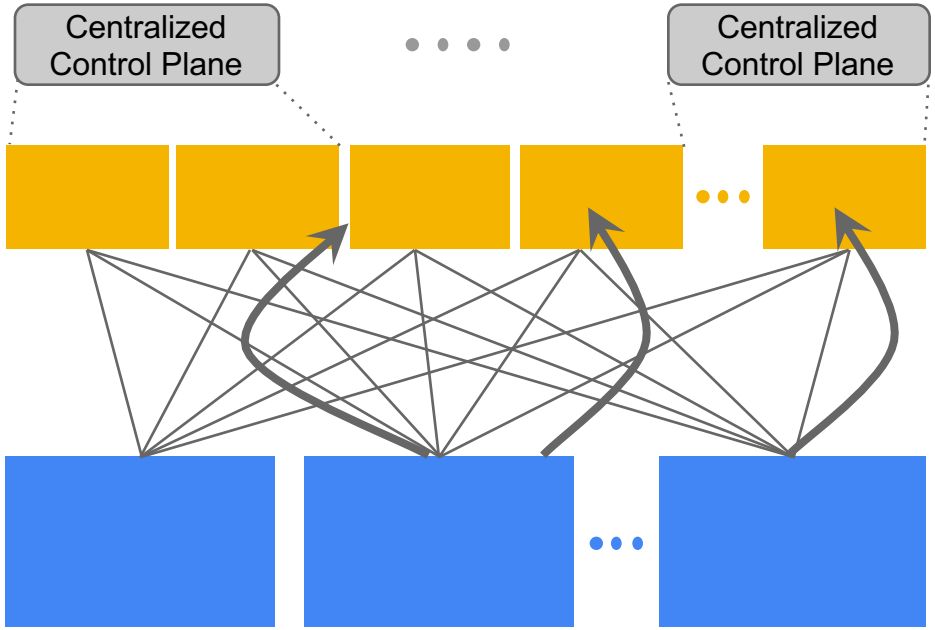
Assess Risk Continuously

Minimize Operator Intervention

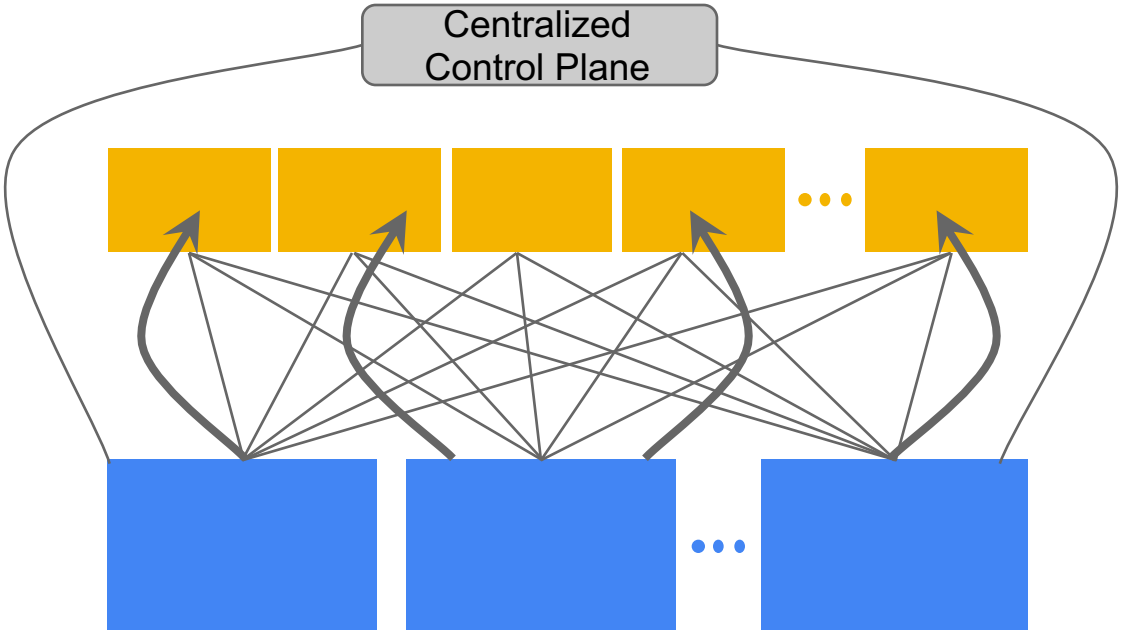
Large Control Plane Failures



Contain the blast radius



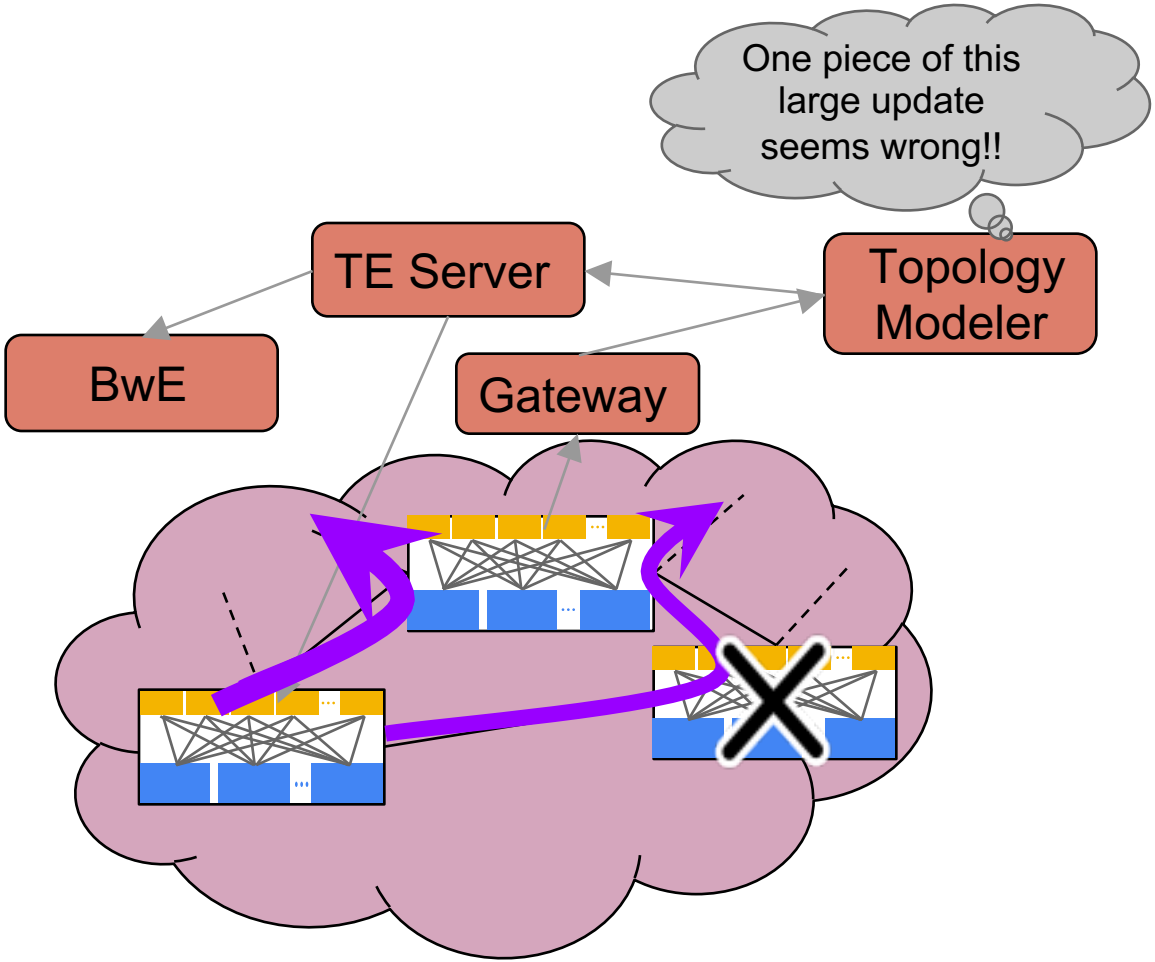
Smaller failure impact, but increased complexity



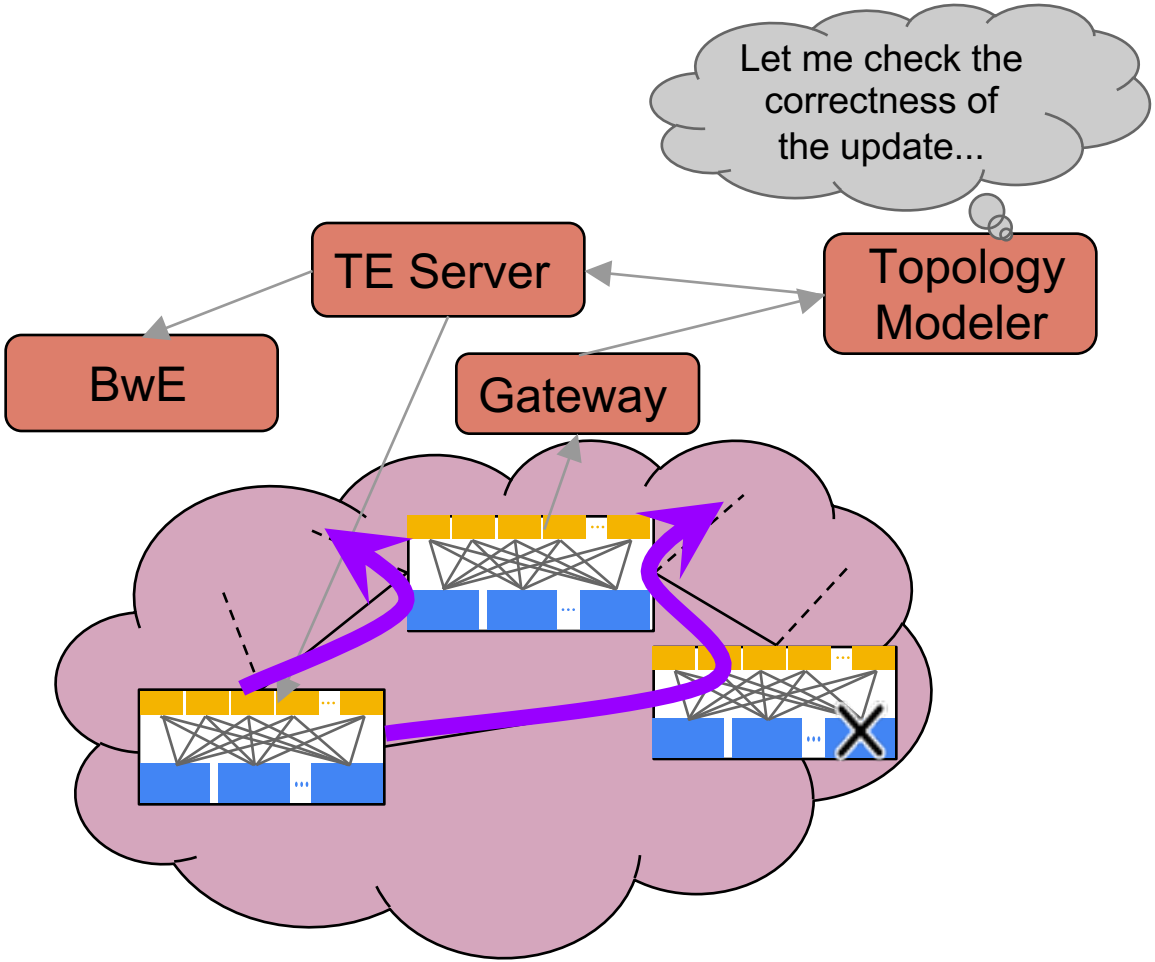
Preserve forwarding state of *all* switches

- *Fail-open* the entire fabric

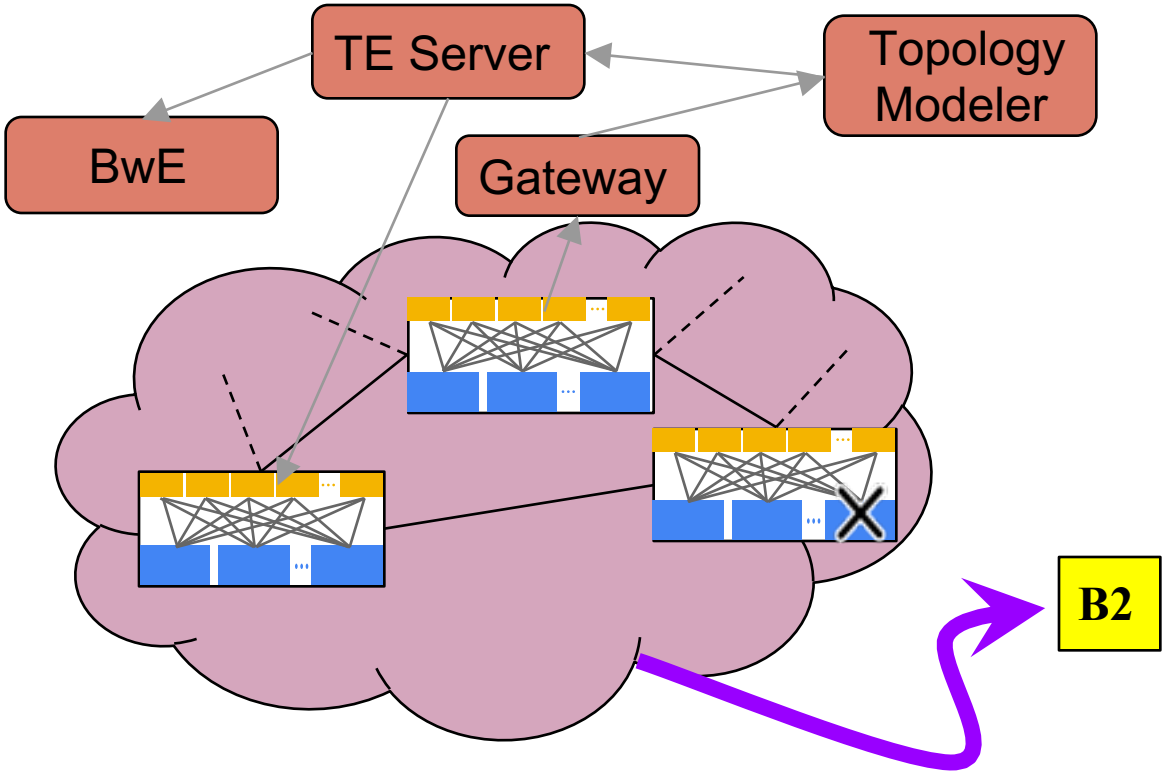
Defensive Control-Plane Design



Trust but Verify



Fallback to B2

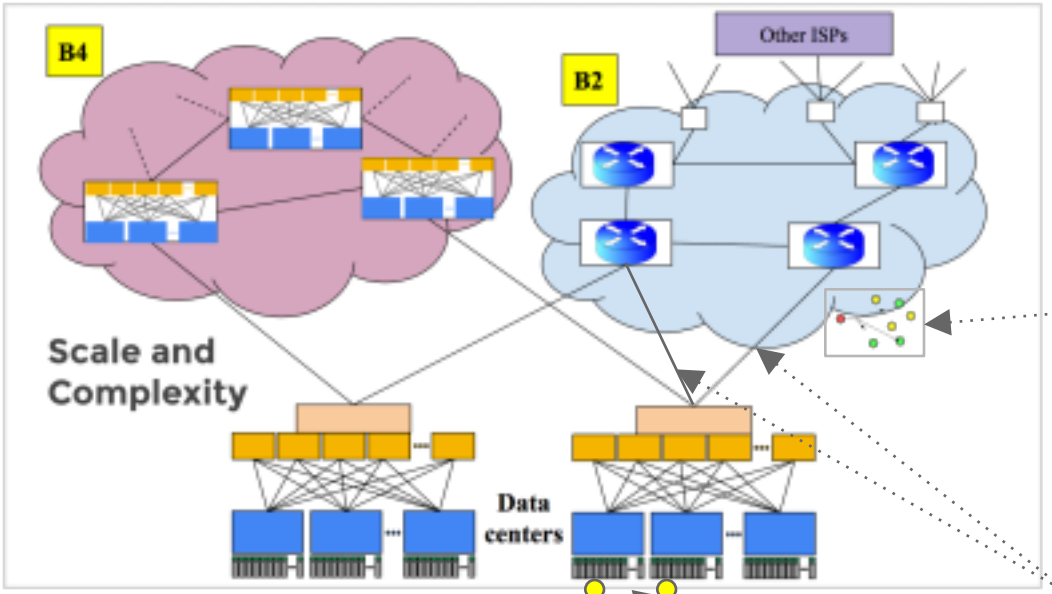


Mitigating Large Failures

Design Fallback Strategies

- B4 → B2
- Tunneling → IP routing
- Big Red Buttons

Continuously Monitor Invariants



Anycast route must have AS path length of **3**

Data center must peer with **two** B2 routers

Must have **one** functional backup SDN controller

This Alone isn’t Enough...

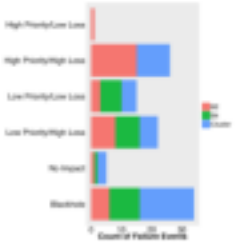
Lessons from Failures

The role of evolution in failures

- Rethink the Management Plane

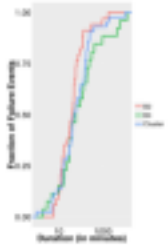
The prevalence of large, severe, failures

- Prevent and mitigate large failures



Long failure durations

- Recover fast



53

We cannot treat a change to the network as an **exceptional** *event*

Make change the *common case*

Make it easy and safe to evolve the network *daily*

- Forces management automation
- Permits small, verifiable changes

Key Takeaway

Content provider networks evolve rapidly

The way we manage evolution can impact availability

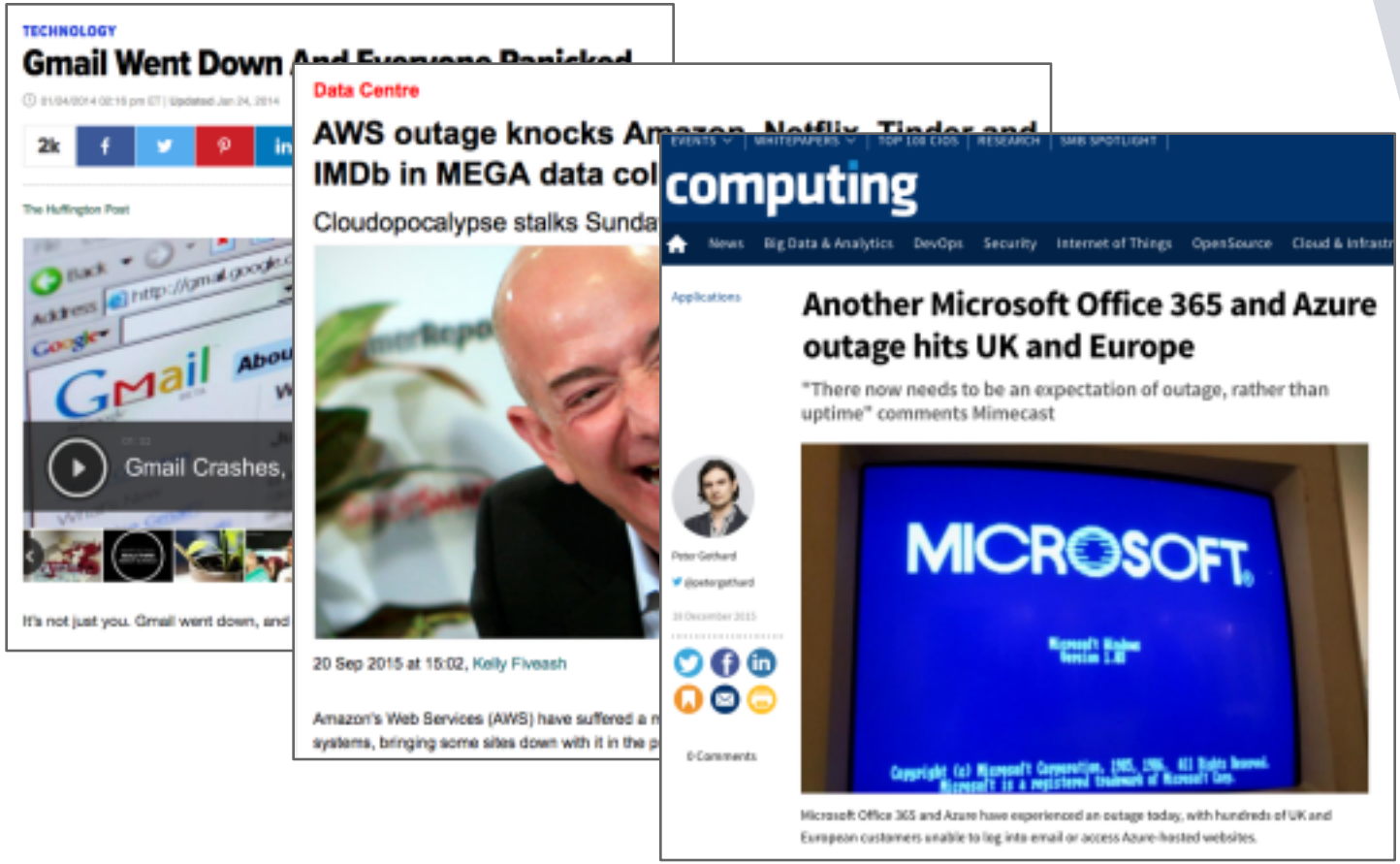
We must make it easy and safe to evolve the network *daily*

Ramesh Govindan, Ina Minei, Mahesh Kallahalla, Bikash Koley and Amin Vahdat
… and a cast of hundreds at Google

Evolve or Die

High-Availability Design Principles Drawn from Google's Network Infrastructure

Impact of Availability Failures



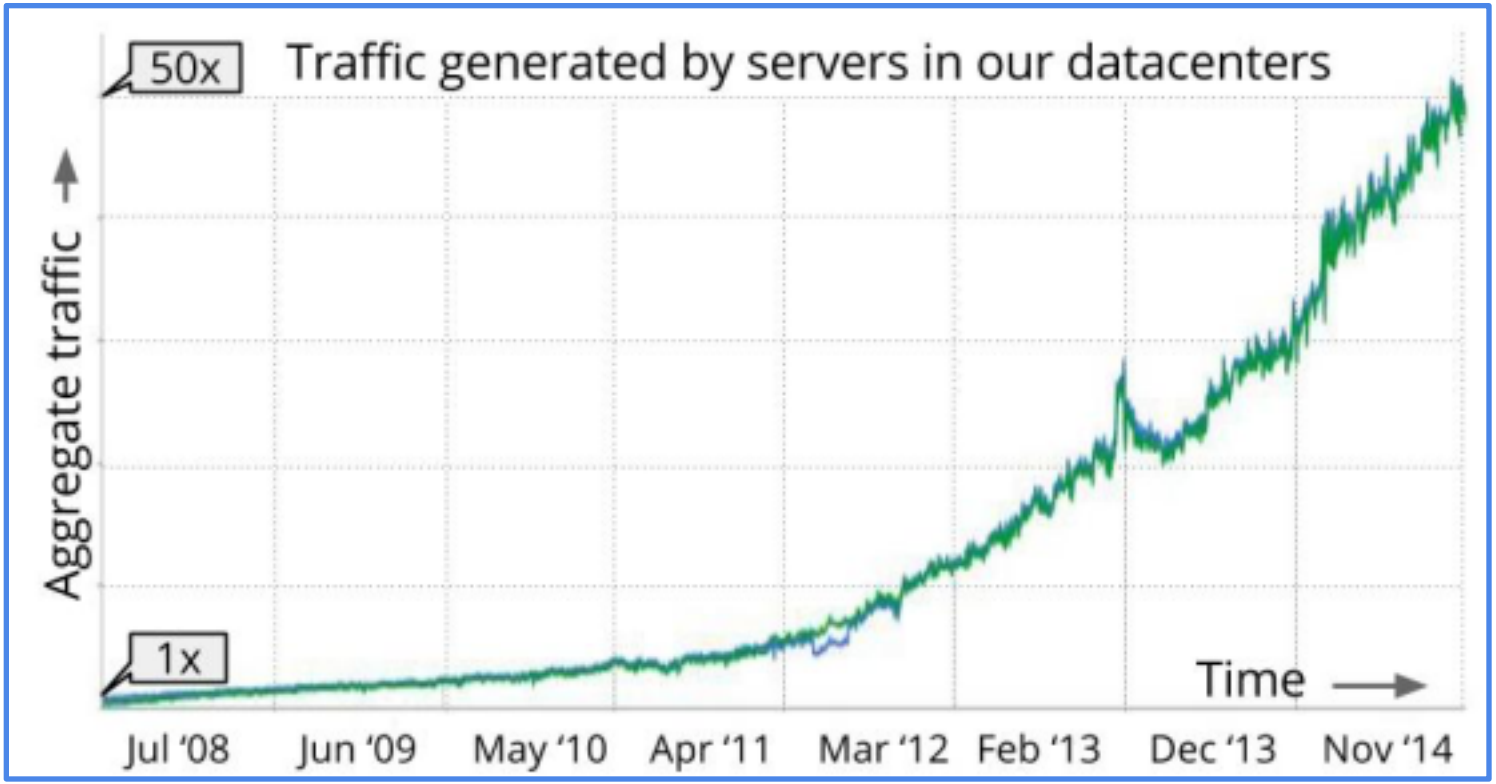
A Case Study: Google

Why is high network availability a challenge?

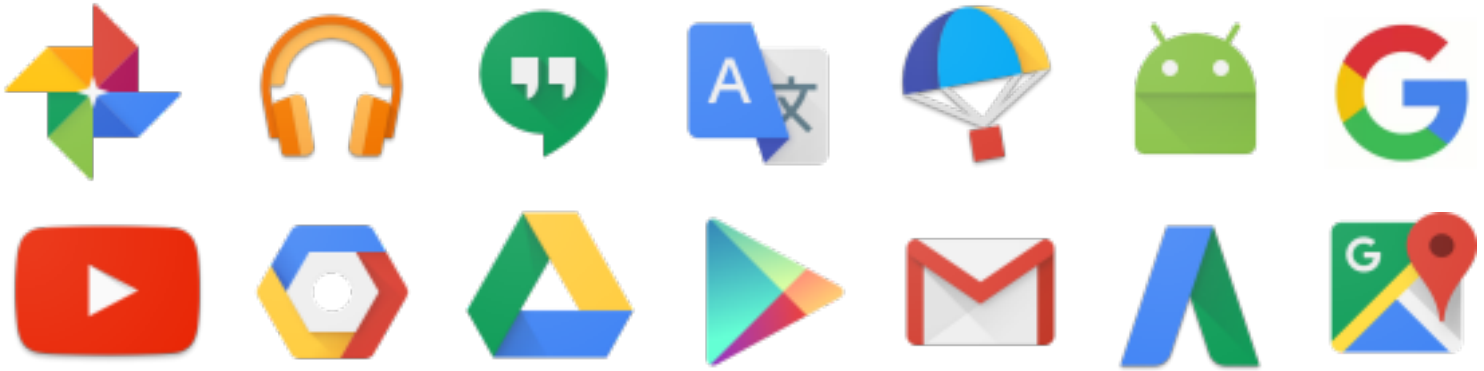
What are the characteristics of network availability failures?

What design principles can achieve high availability?

The velocity of evolution is fueled by traffic growth...



… and by an increase in product and service offerings



Networks have *very* different designs

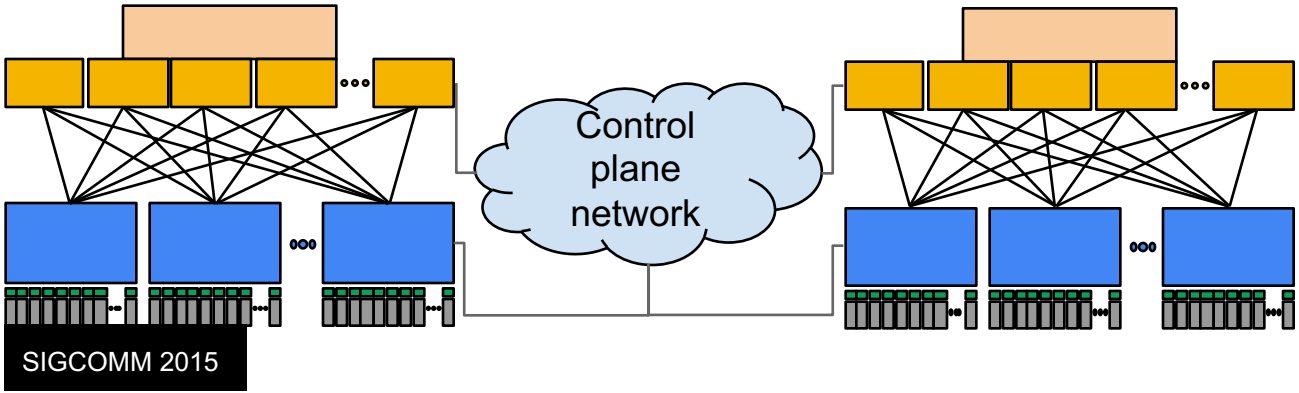
Different hardware

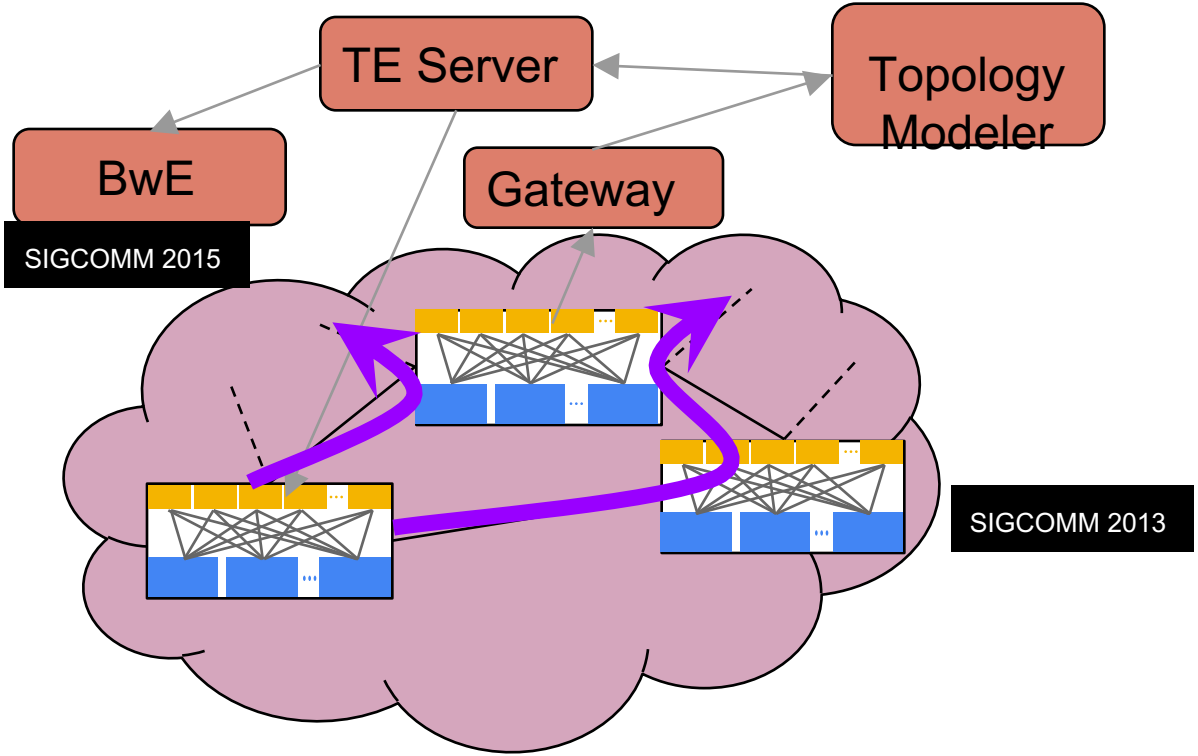
Different control planes

Different forwarding paradigms

These differences increase management and evolution complexity

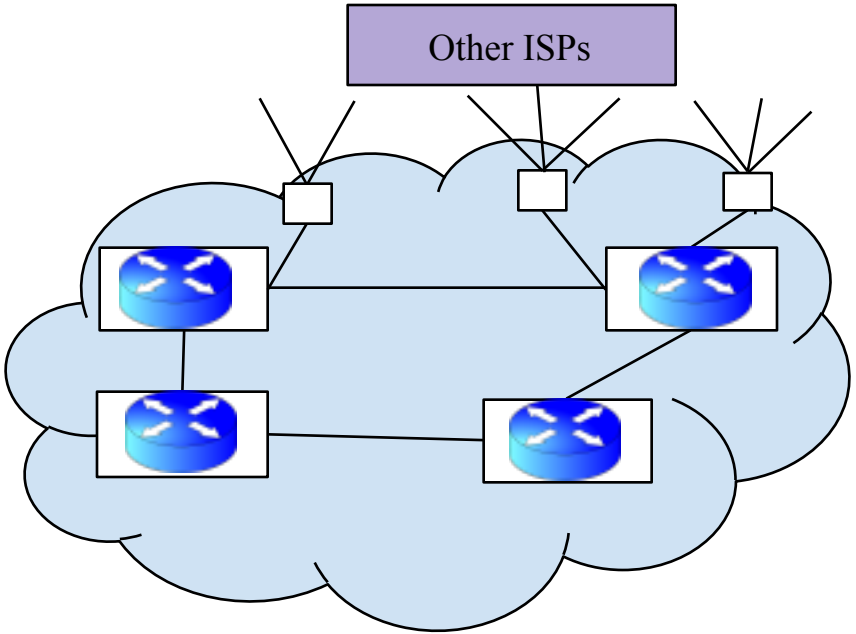
- Fabrics with merchant silicon chips
- Centralized control plane
- Out of band control plane network



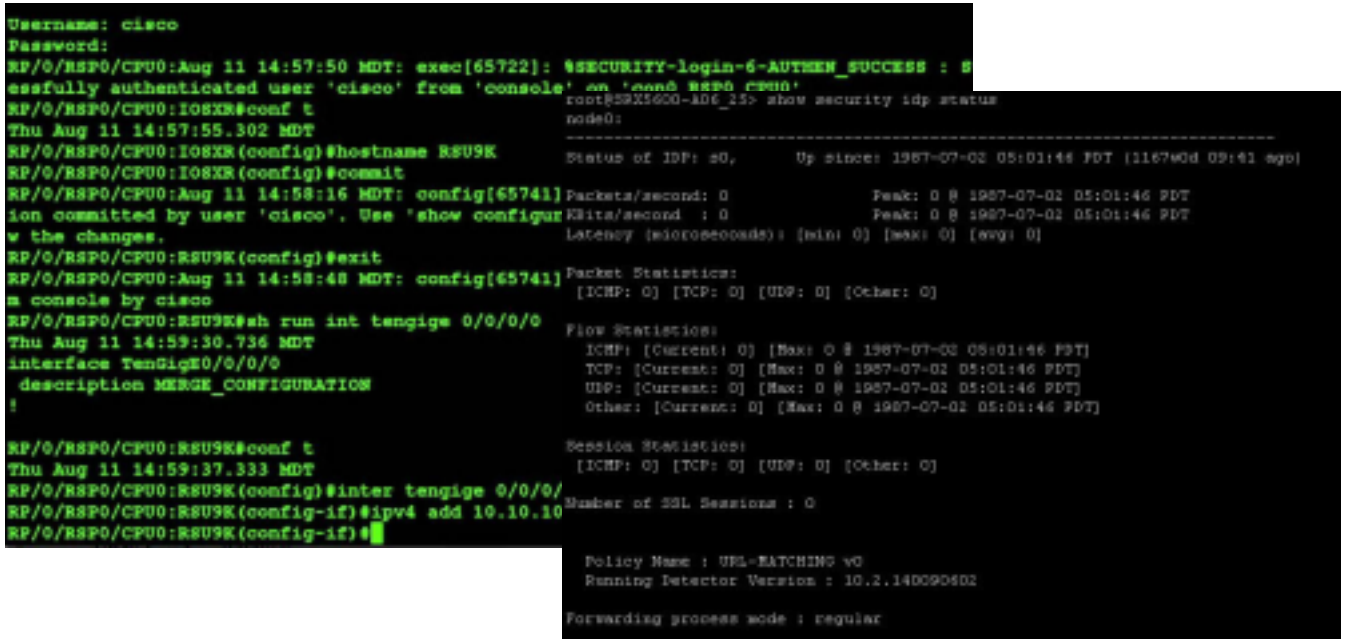


- B4 routers built using merchant silicon chips
- Centralized control plane within each B4 site
- Centralized traffic engineering
- Bandwidth enforcement for traffic metering

- B2 routers based on vendor gear
- Decentralized routing and MPLS TE
- Class of service (high/low) using MPLS priorities

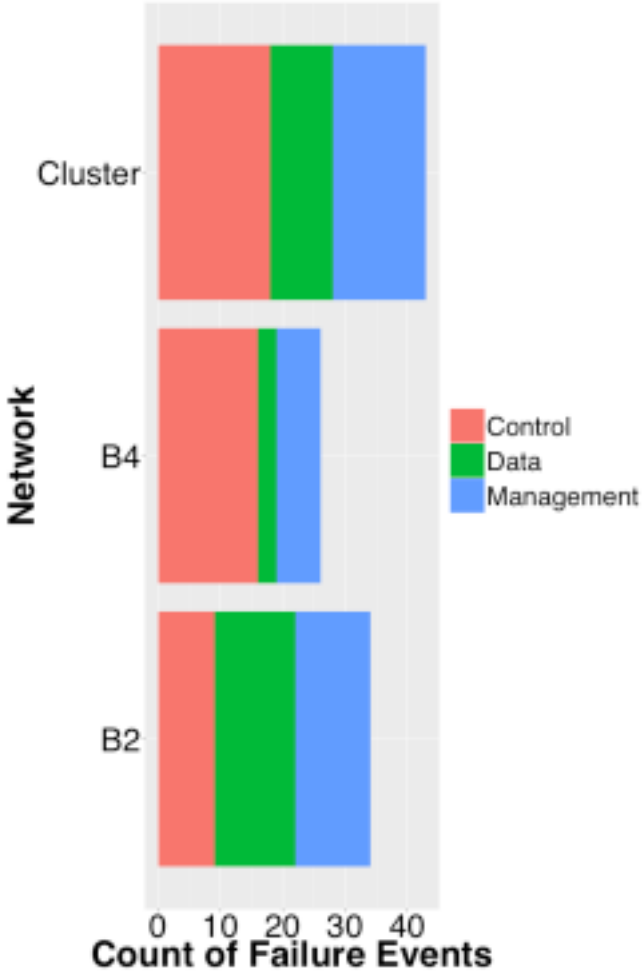


The Management Plane



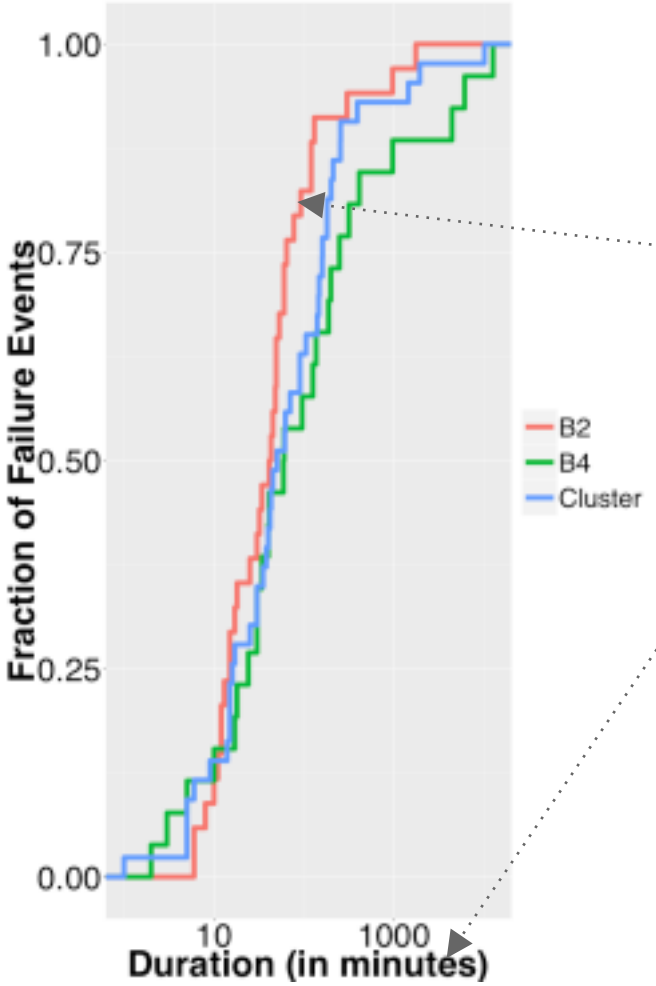
Low-level, per device, abstractions for management operations

Where do failures happen?



No network or plane that dominates

How long do the failures last?



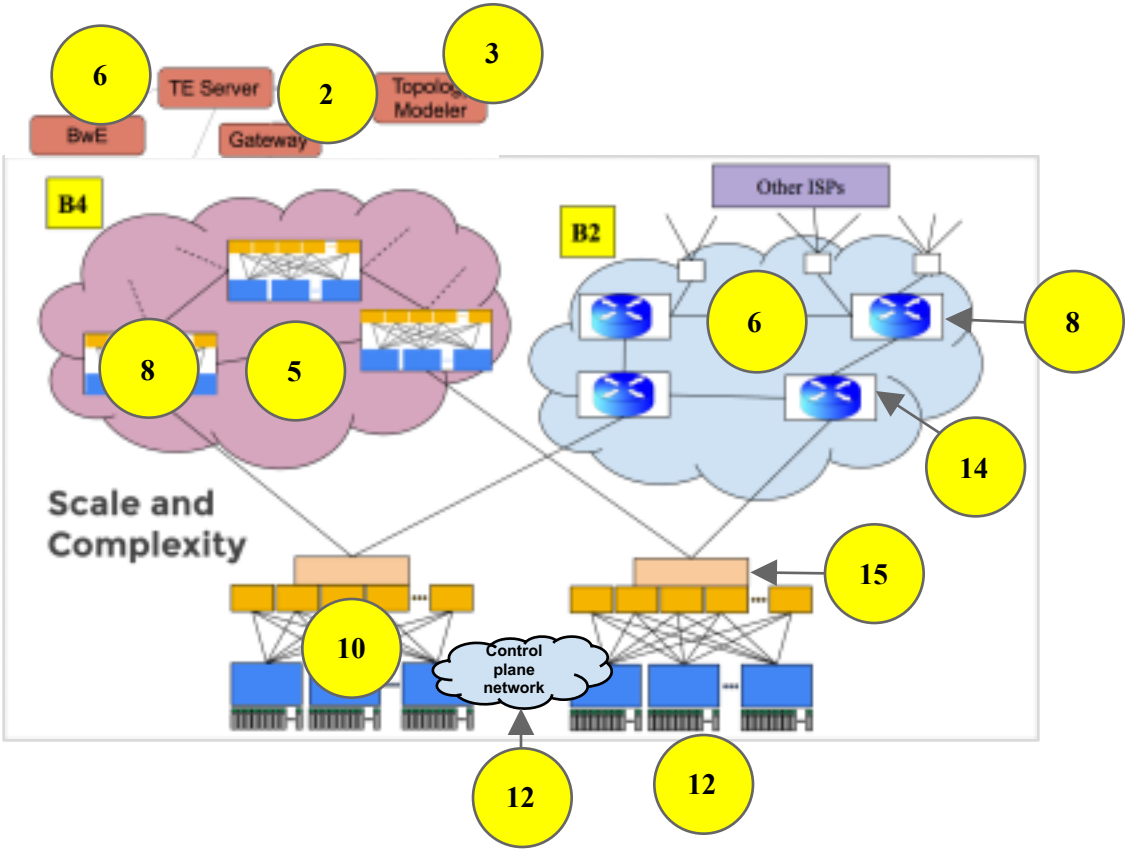
Shorter failures on B2

Durations much longer than outage budgets

What role does evolution play?

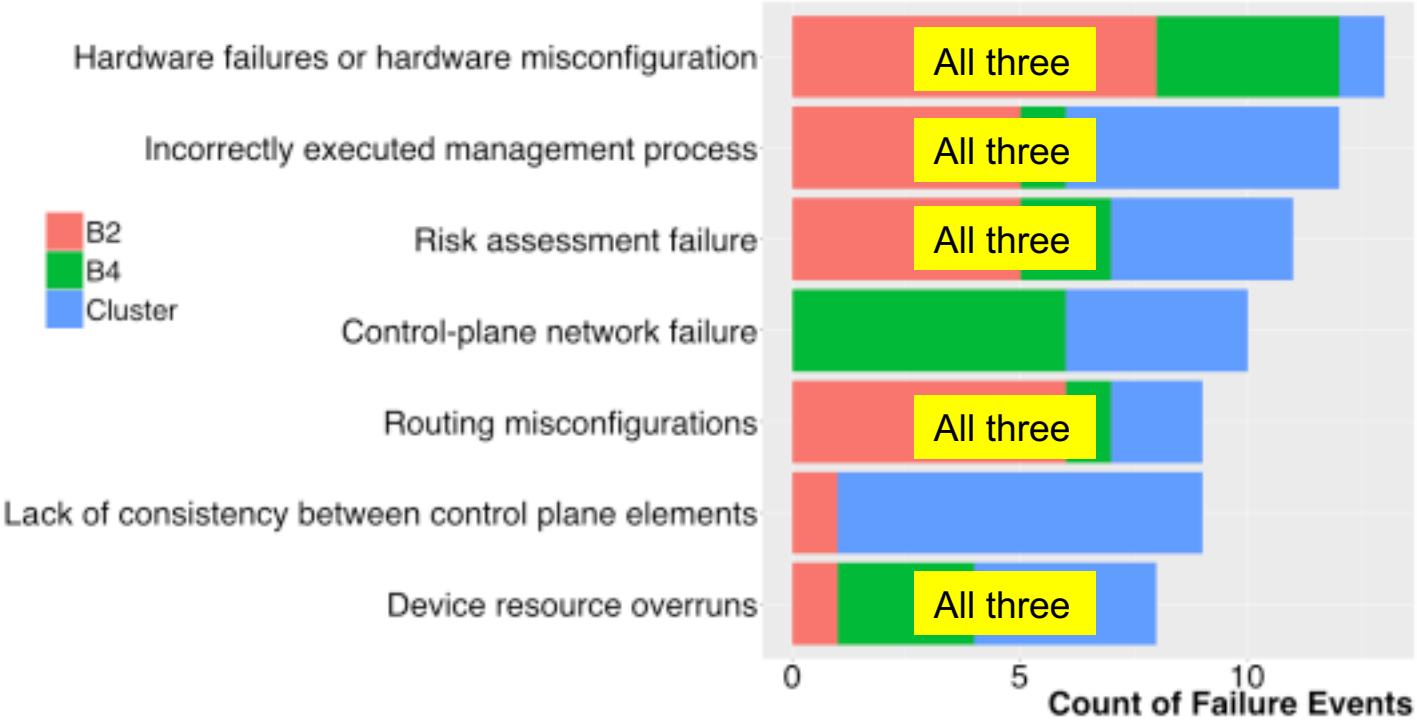
70% of failures happen when a management operation is in progress

Where do failures happen?

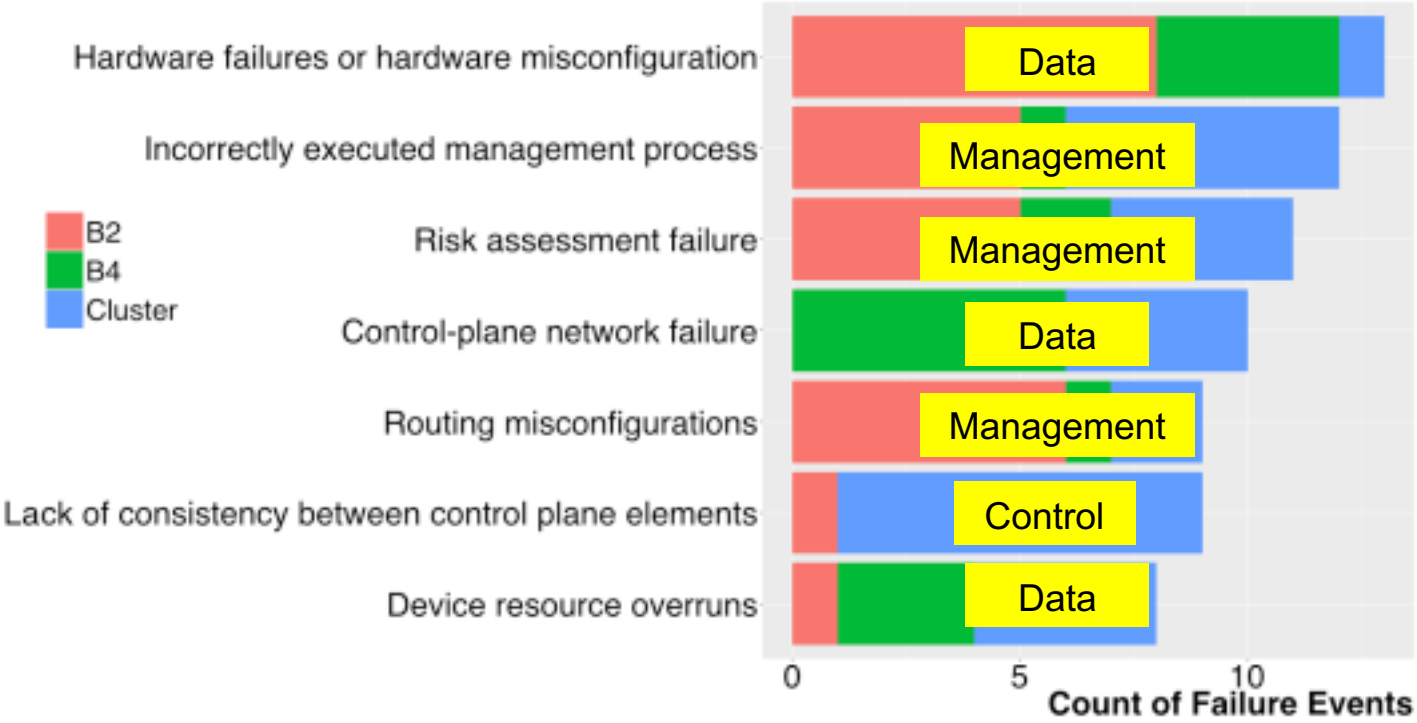


Failures are everywhere

Across networks



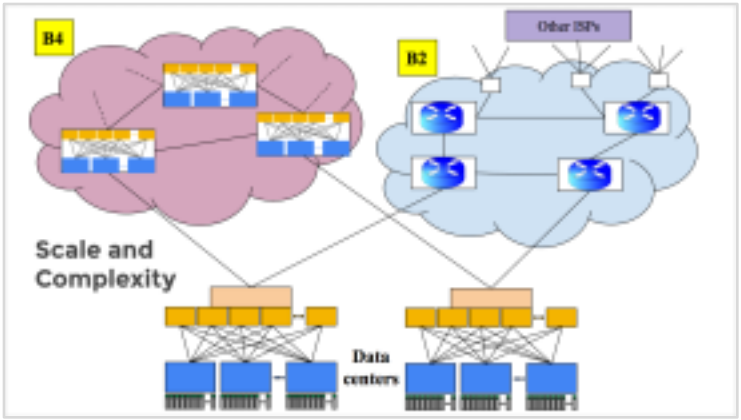
Across planes

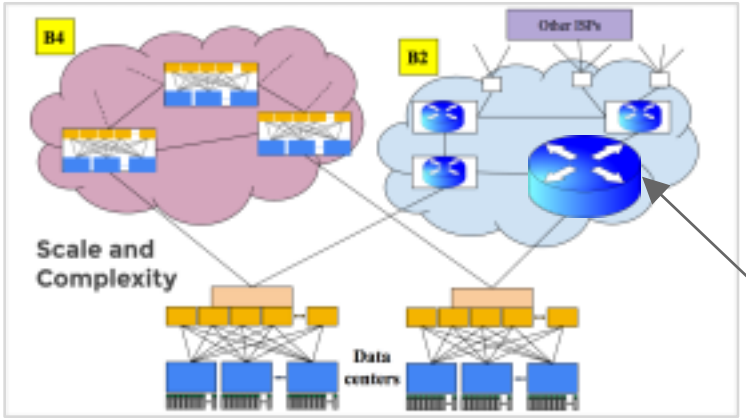


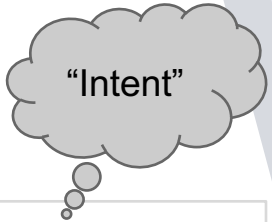
Root-Cause Categorization

What are the root causes for these failures?

Low-level network management cannot ensure high availability





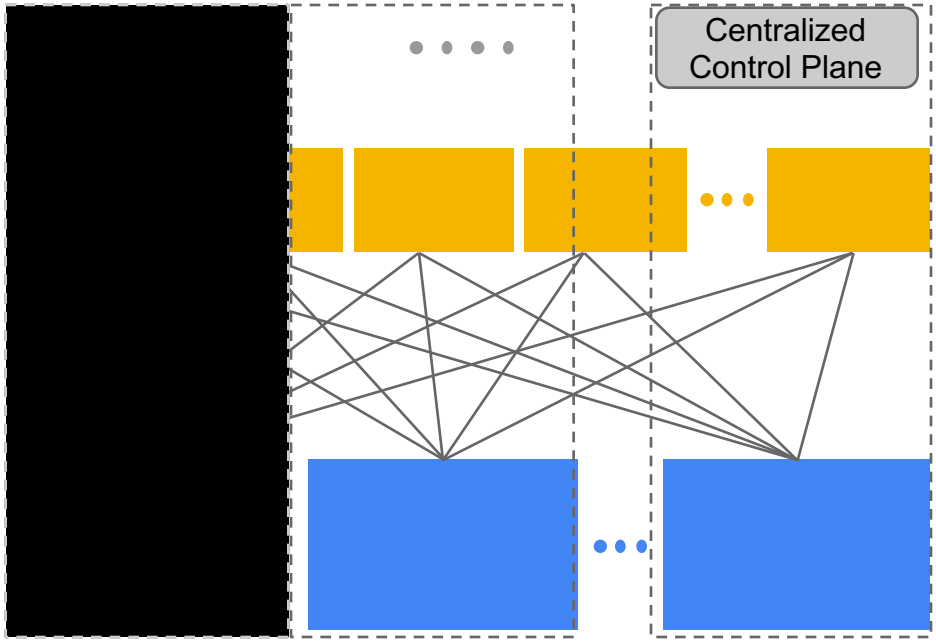


I want to upgrade this router

Lots of complexity hidden below this statement

- Carry Tb/s of traffic
- Have hundreds of interfaces
- Interface with associated optical equipment
- Run a variety of control plane protocols: MPLS, IS-IS, BGP all of which have network-wide impact
- Have high capacity fabrics with complicated dynamics
- Have configuration files which run into 1000s of thousands of lines

Contain failure radius



Each partition managed by different control plane

Adds design complexity

Even if one partition fails, others can carry traffic

Key Takeaway

Content provider networks evolve rapidly

The way we manage evolution can impact availability

We must make it easy and safe to evolve the network *daily*

By learning from failures

What has Google Learnt from Failures?

Why is high network availability a challenge?

- Factors that impact availability

What are the characteristics of network failures?

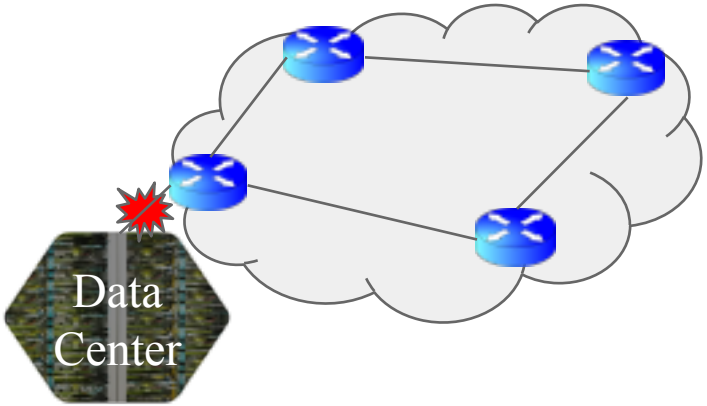
- Severity, duration, prevalence
- Root-cause categorization

What design principles can achieve high availability?

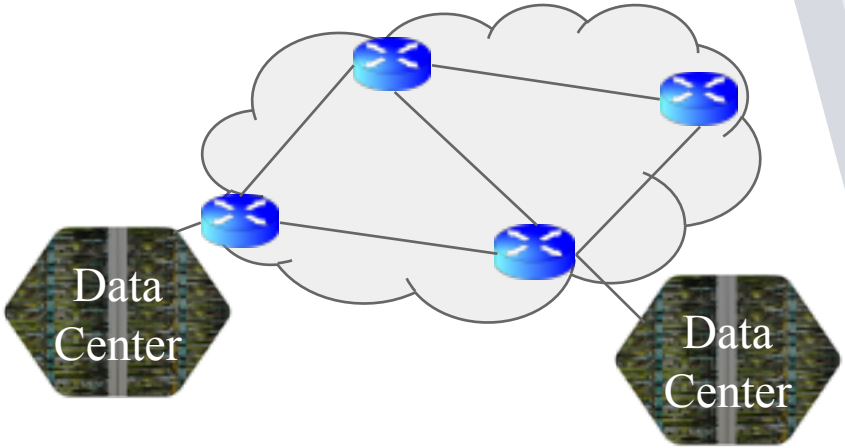
- Lessons learned from root-causes

In a global network

Failures are common



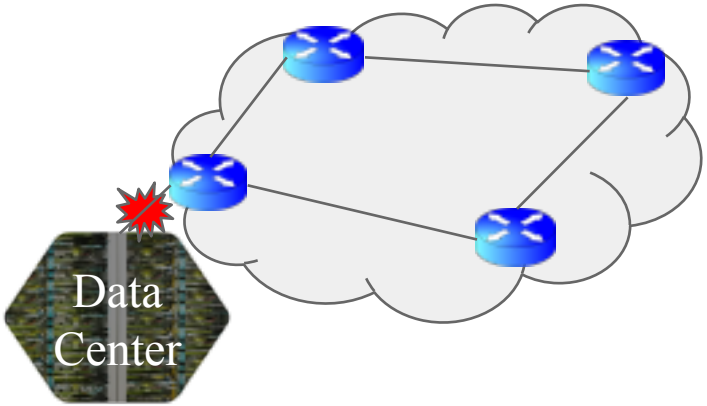
Configuration can change



These can impact network availability

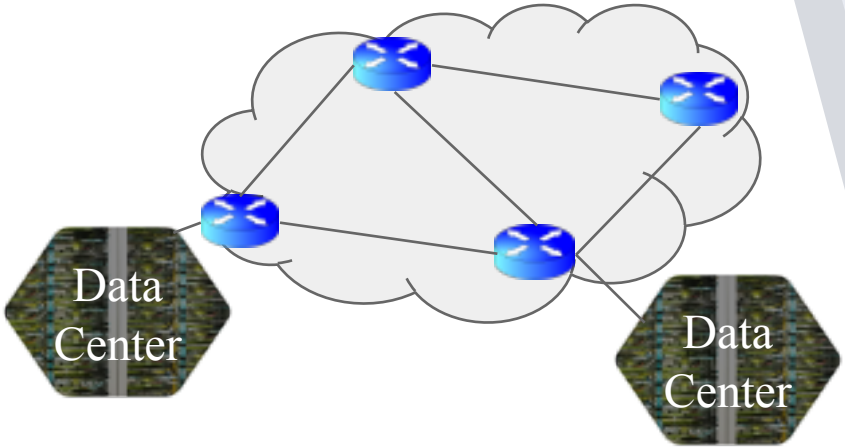
How long does it take...

… to **root-cause** a failure



10s of minutes to hours

… to **upgrade** part of the network



Hours to days

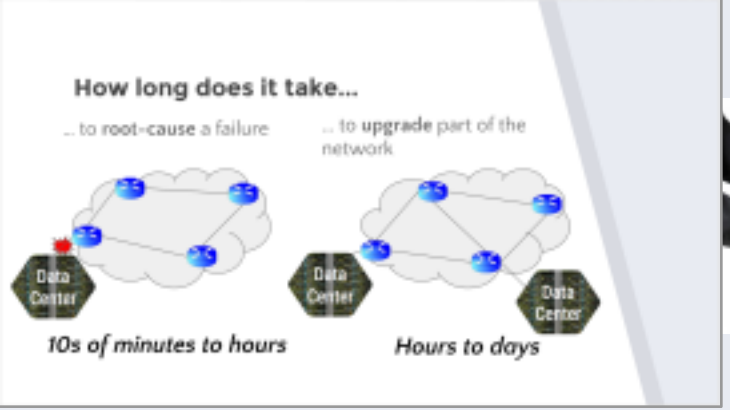
Outage budgets...

… for **four 9s** availability?

4 minutes per month

… for **five 9s** availability?

24 seconds per month



To move towards higher availability targets, it is important to ***learn from failures***