SIGCOMM 2024

— SYDNEY —

A Decentralized SDN Architecture for the WAN

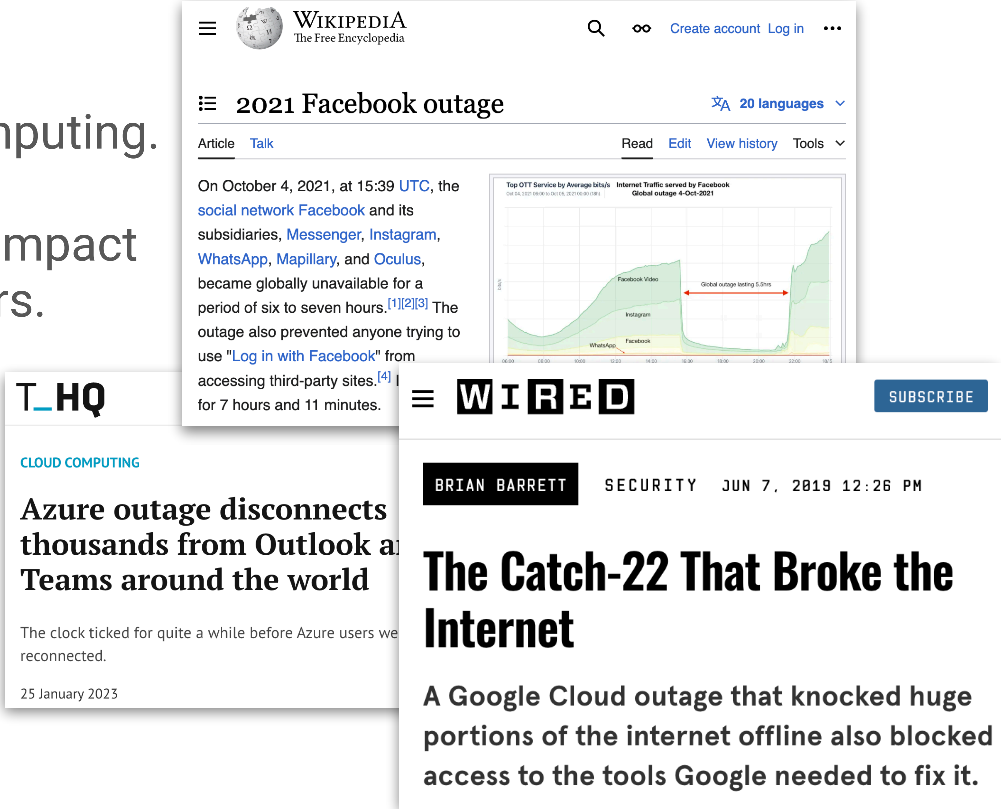
Alexander Krentsel[1,2], Nitika Saran[3*], Bikash Koley[1], Subhasree Mandal[1], Ashok Narayanan[1], Sylvia Ratnasamy[1,2], Ali Al-Shabibi[1], Anees Shaikh[1], Rob Shakir[1], Ankit Singla[1], Hakim Weatherspoon[3]

[1]Google, [2]UC Berkeley, [3]Cornell

*work done while at Google.

Problem statement

- Global WANs enable planet-scale computing.
- When outages occur, they have large impact on cloud providers and their customers.



2021 Facebook outage

On October 4, 2021, at 15:39 UTC, the social network Facebook and its subsidiaries, Messenger, Instagram, WhatsApp, Mapillary, and Oculus, became globally unavailable for a period of six to seven hours.[1][2][3] The outage also prevented anyone trying to use "Log in with Facebook" from accessing third-party sites.[4]

for 7 hours and 11 minutes.

Top OTT Service by Average bits/s
Oct 04, 2021 06:00 to Oct 05, 2021 00:00 (18h)
Internet Traffic served by Facebook
Global outage 4-Oct-2021
bits/s
Facebook Video
Instagram
Facebook
WhatsApp
Global outage lasting 5.5hrs
06:00
08:00
10:00
12:00
14:00
16:00
18:00
20:00
22:00
10/5

CLOUD COMPUTING

Azure outage disconnects thousands from Outlook a

Teams around the world

The clock ticked for quite a while before Azure users we reconnected.

25 January 2023

BRIAN BARRETT SECURITY JUN 7, 2019 12:26 PM

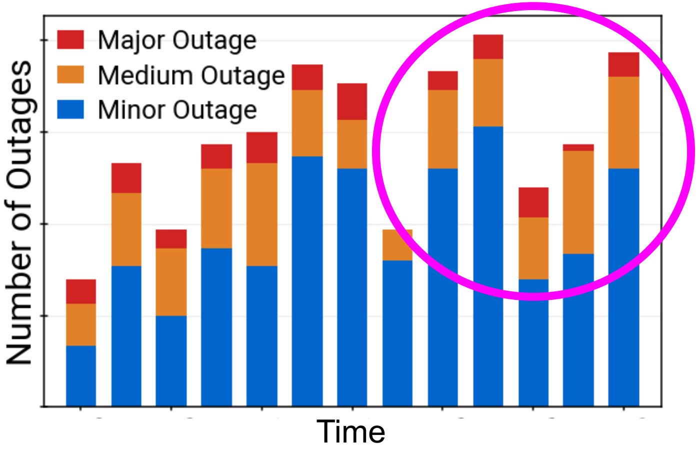
The Catch-22 That Broke the Internet

A Google Cloud outage that knocked huge portions of the internet offline also blocked access to the tools Google needed to fix it.

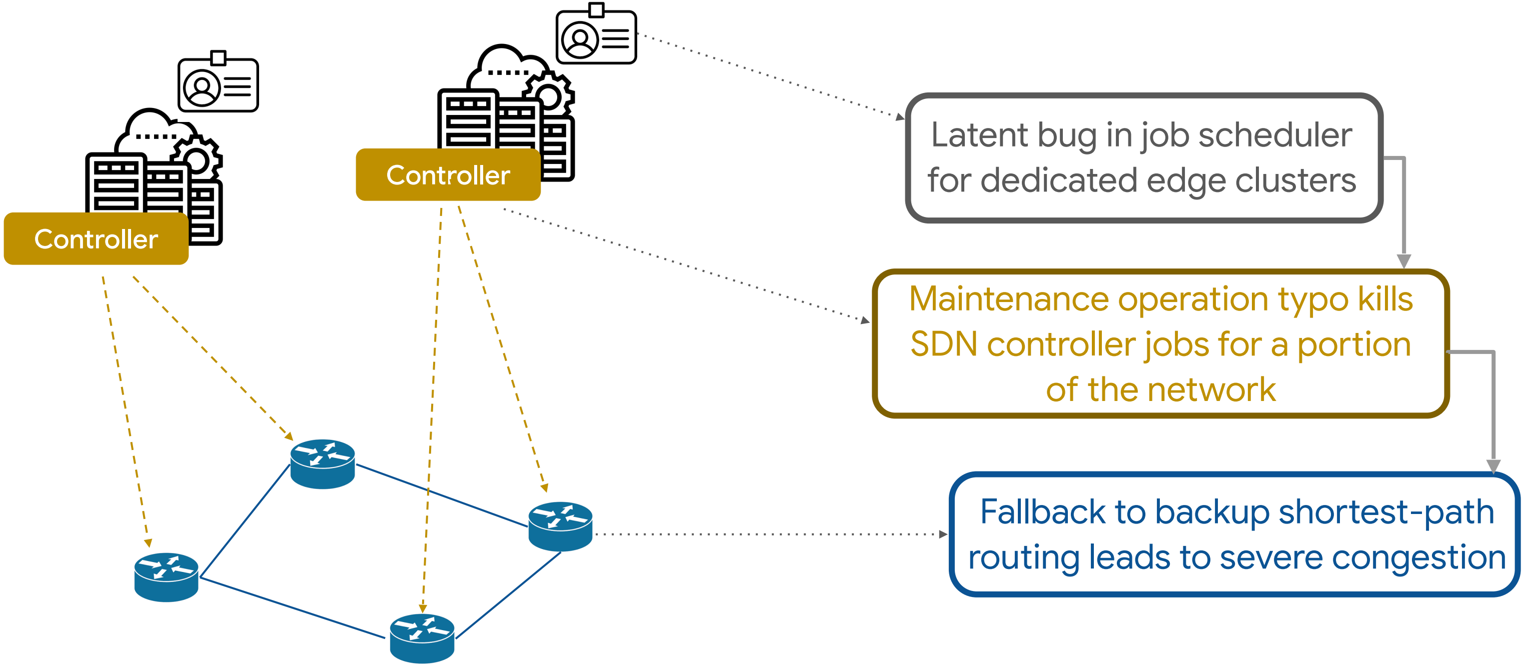
Problem statement

- Outages continue *despite* decades of experience and vast literature of best practices

- Existence of small outages is expected, *large* outages is not
  - complex, cascading root causes; rarely due to loops/ deadends, simple link cuts, protocol bugs



A complex outage example



Latent bug in job scheduler for dedicated edge clusters

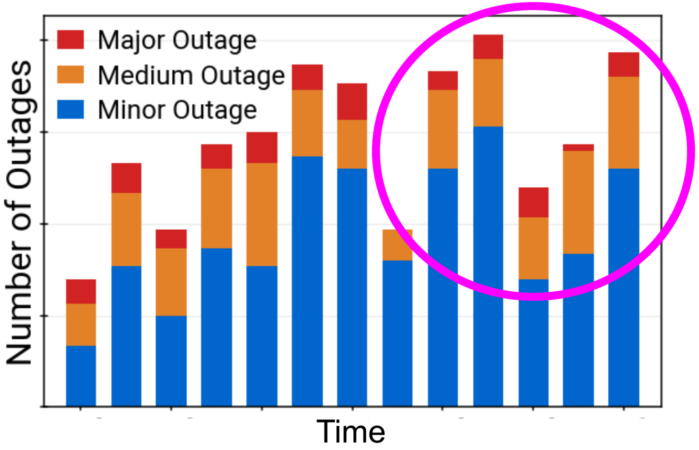
Maintenance operation typo kills SDN controller jobs for a portion of the network

Fallback to backup shortest-path routing leads to severe congestion

Problem statement

- Outages continue *despite* decades of experience and vast literature of best practices

- Existence of small outages is expected, *large* outages is not
    - complex, cascading root causes; rarely due to loops/ deadends, simple link cuts, protocol bugs



What can we do at *design* time to limit the occurrence of complex failures?

Designing networks to avoid complex failures

- Practitioners apply some techniques today:
    - isolation
    - diverse redundancy
    - formal verification

- Orthogonal approach in our work: ***simplification***

Concretely: what components can we can take out?

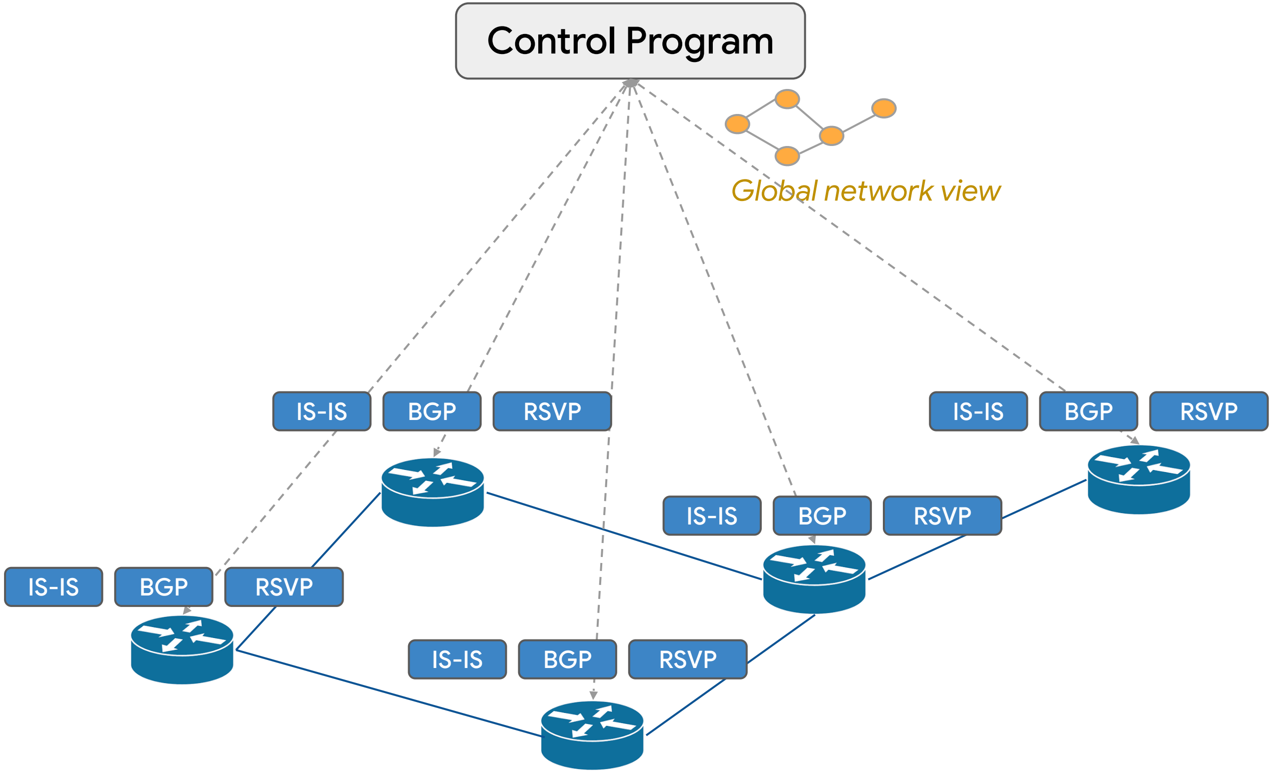
...while maintaining functionality, performance, and cost

Outline

Rest of this talk…

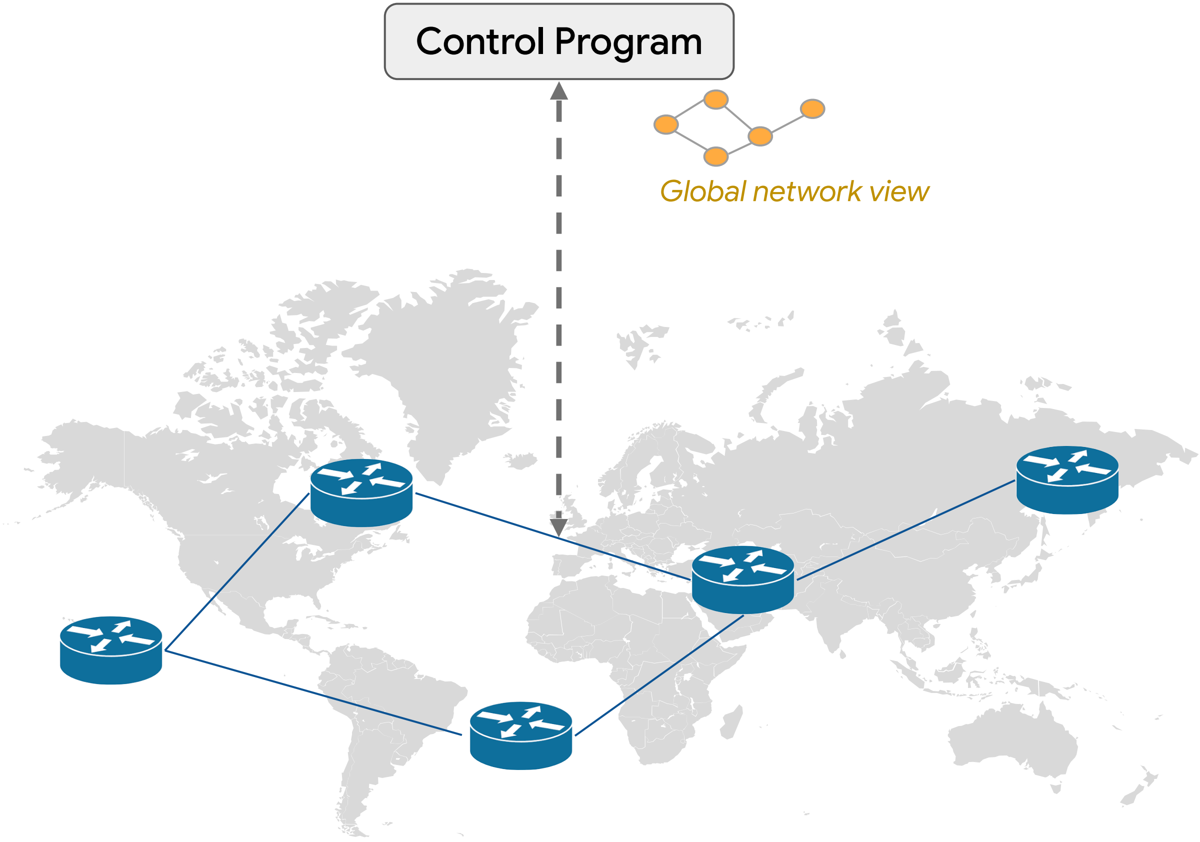
- SDN WAN architectures today
- Opportunities for simplification
- Our approach: **Decentralized SDN (dSDN)**
- dSDN evaluation

SDN WAN architectures today: from protocols to SDN



Protocols survivable, but inefficient and difficult to manage, evolve, and scale

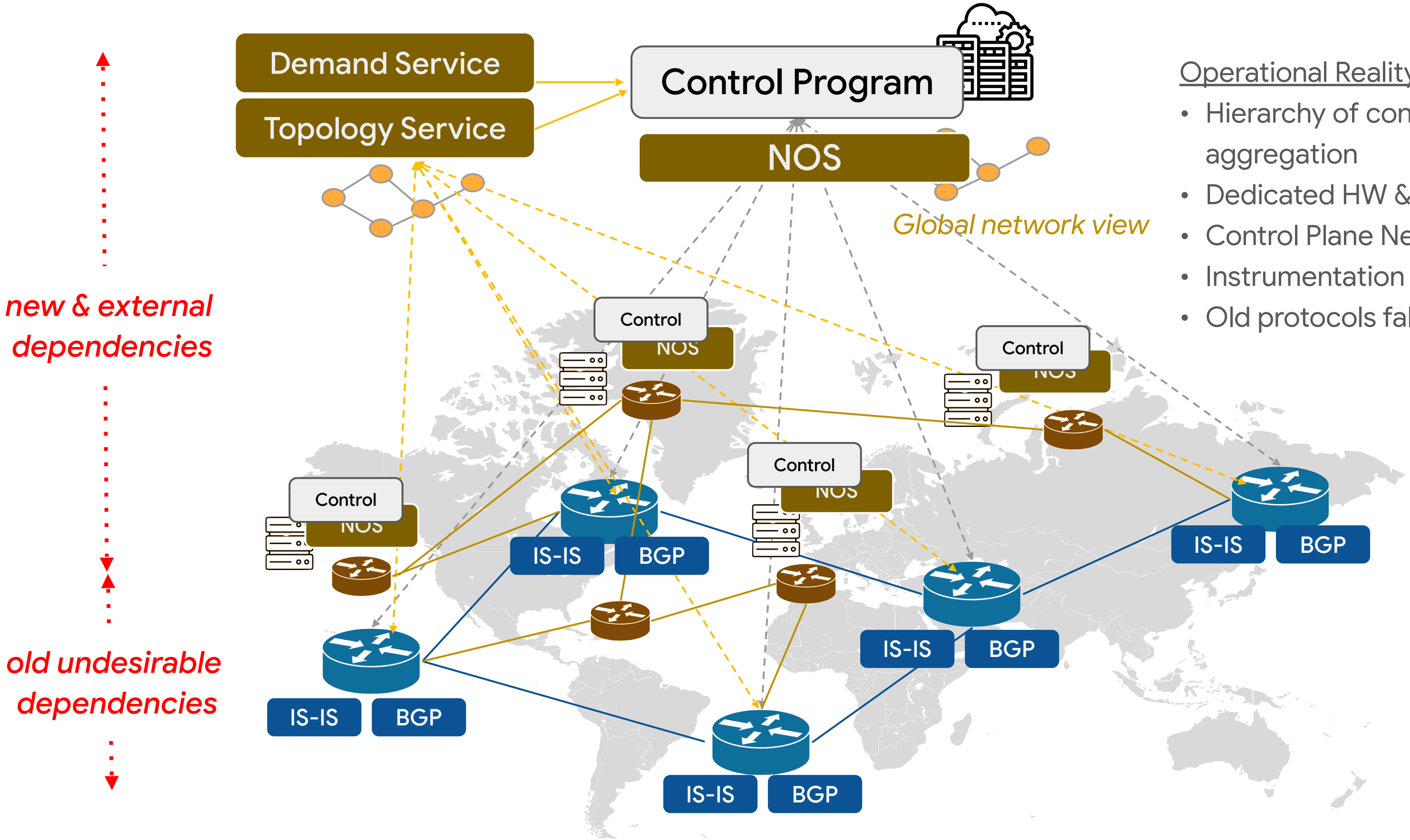
SDN WAN architectures today: from protocols to SDN



Control Program

Global network view

SDN WAN architectures today: reality is complex



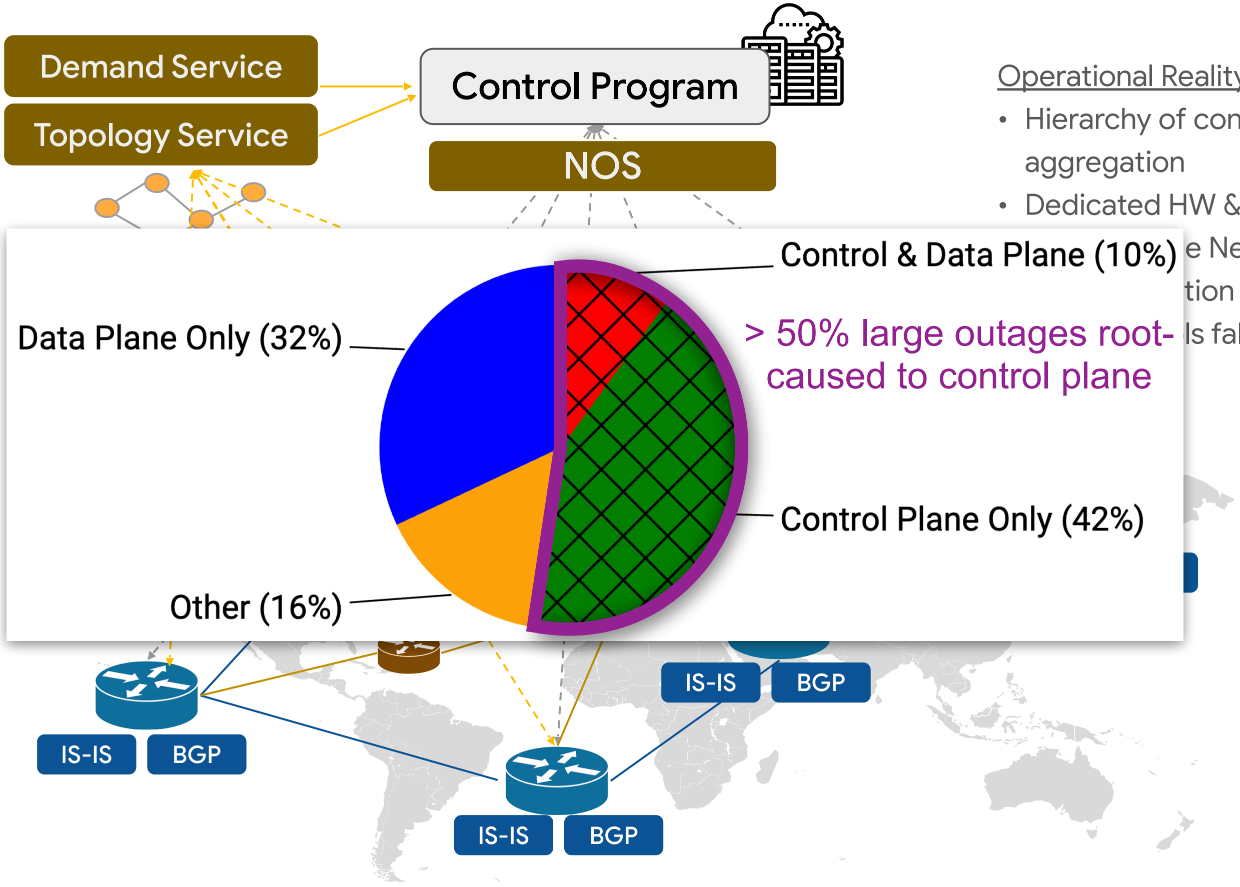
Operational Reality

- Hierarchy of control programs to provide aggregation
- Dedicated HW & SW to run them
- Control Plane Network to reach routers
- Instrumentation services to collect inputs
- Old protocols fallback in case of split-brain

SDN WAN architectures today: reality is complex

new & external dependencies

old undesirable dependencies



> 50% large outages root-caused to control plane

Operational Reality

- Hierarchy of control programs to provide aggregation
- Dedicated HW & SW to run them
- e Network to reach routers
- tion services to collect inputs
- ls fallback in case of split-brain

Achieving simplification

Look to tackle *complex* outages head-on by aiming for *simplification*.

Why not go back to distributed protocols? Because shift to SDN gave us benefits that we’ve come to rely on…

1. **Operator-defined** code, which enabled innovation
2. **Optimized** computations (TE) on global view of network
3. **Simplicity** of “consensus free” path selection

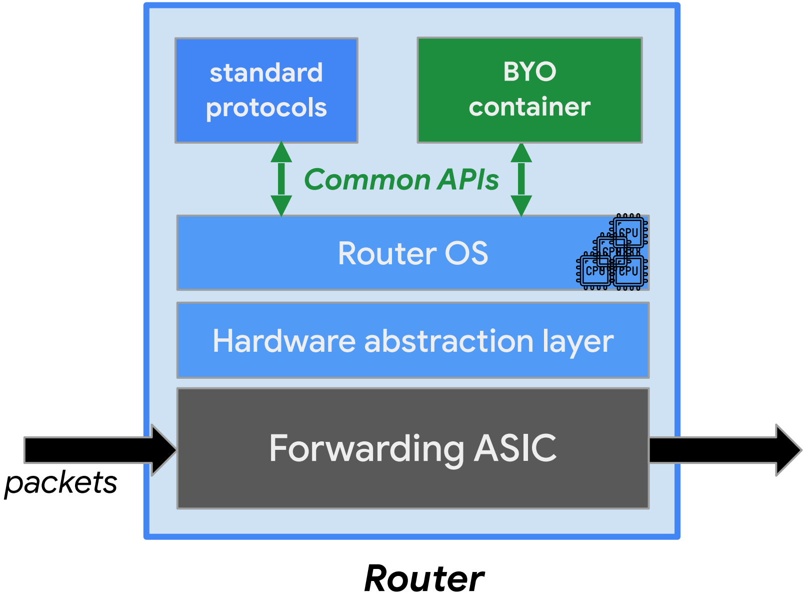
Goal: Simplify while *retaining SDN’s benefits*.

Opportunity: operator applications on-router

1. Router vendors now support running 3rd-party containerized code directly “on the box”
   → operators can run custom control code on router CPU

2. New generation of standardized control/config APIs (*e.g.*, gRIBI, gNMI, OF/P4)
   → control code is uniform across vendors

3. Expanded on-router CPU resources
   → from single-core to multi-core multi-GHz CPUs

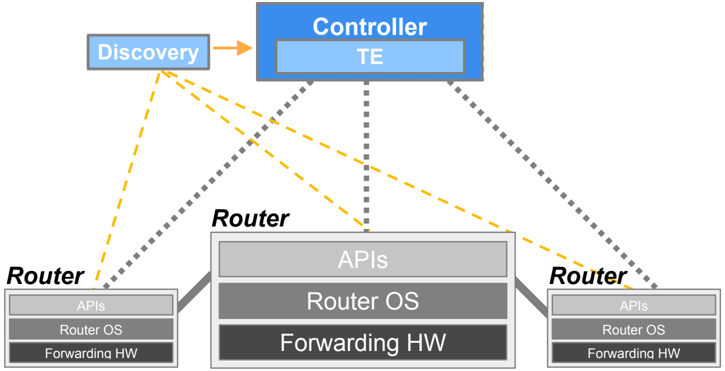


Operator-defined control applications can run on-box.

Essential idea: decentralizing SDN

Decentralize SDN by moving *operator-written controller code onto the router*

➡ ...replicate controller on each router



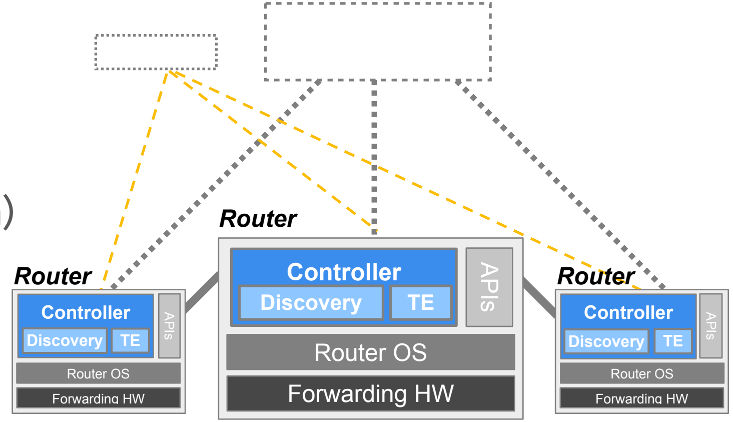
Essential idea: decentralizing SDN

Decentralize SDN by moving *operator-written controller code onto the router*

➡ ...replicate controller on each router

Concretely, every router runs a ***dSDN controller*** that

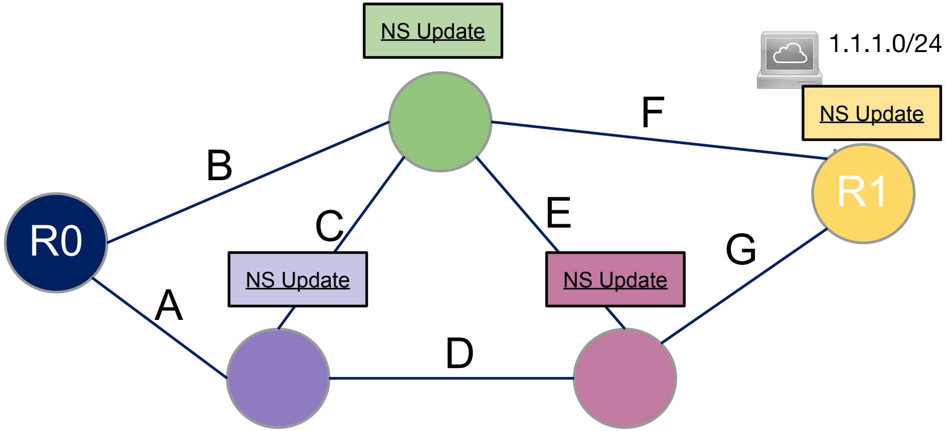
1. **floods** its local node state; learns global network view
2. locally **computes *all* paths** (using a traffic eng. algorithm)
3. “programs” locally originating paths as ***source-routes***



walkthrough ➡

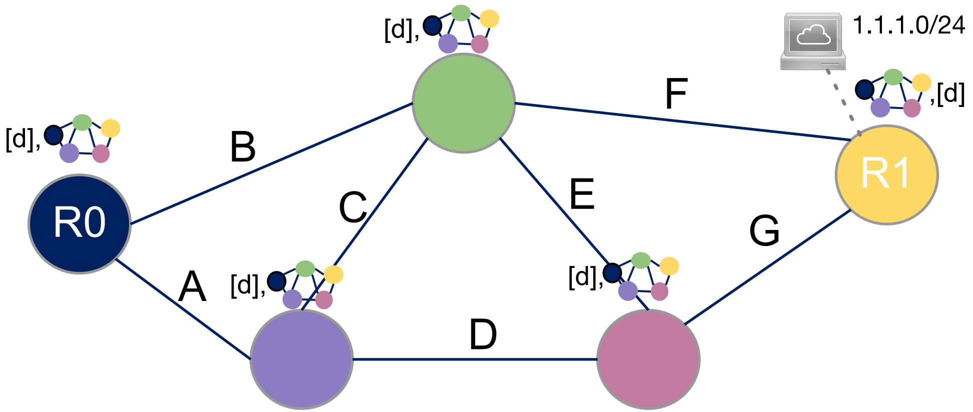
(1) Building a global view

Node State Update
NodeID: R0
Prefixes: 1.1.2.0/24
Neighbors: …
Link Capacity: (A:10G), (B:20G)
Demand Vector: <…>



- Each node floods “Node State” (NS) updates to all other nodes
  - Contains *local* topology and demand info
  - ⇒ each node discovers a global network view

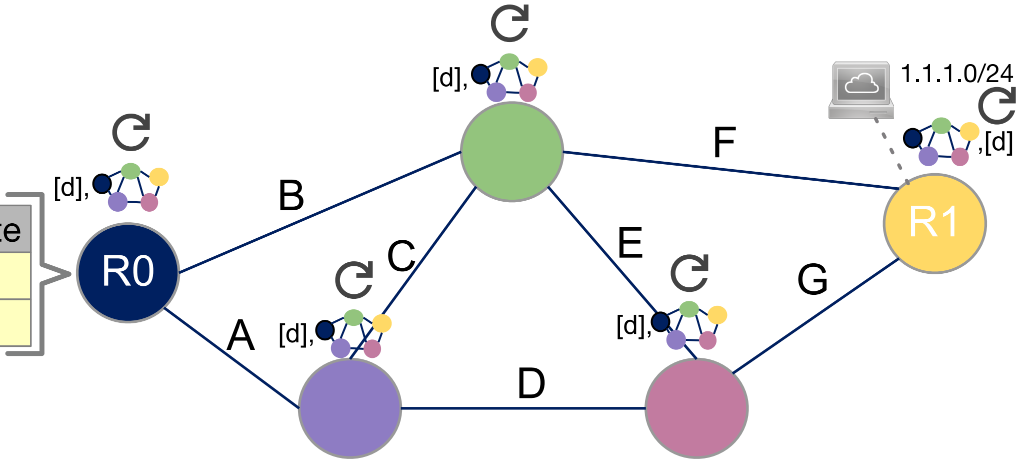
(1) Building a global view



- Each node floods “Node State” (NS) updates to all other nodes
  - Contains *local* topology and demand info
  - ⇒ each node discovers a global network view

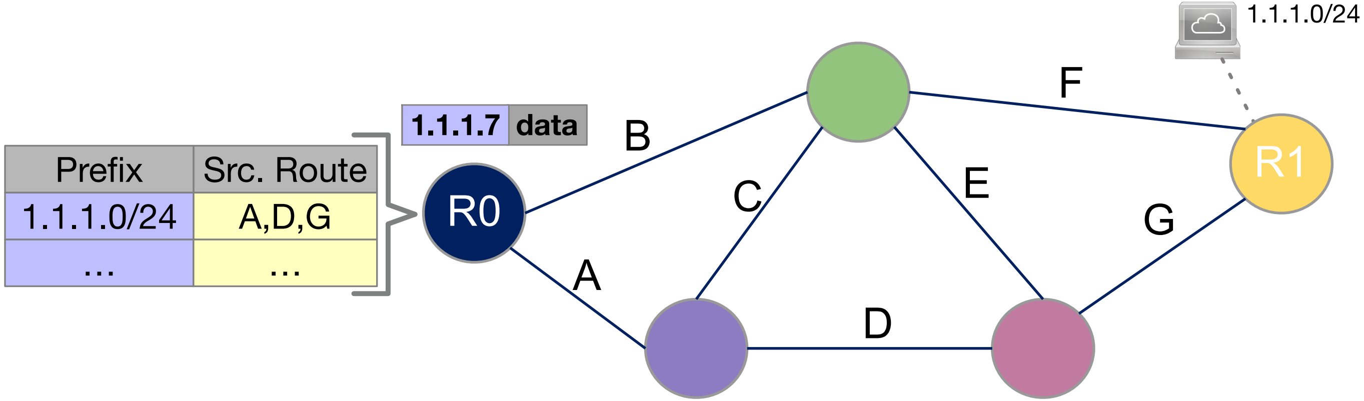
(2) Each node computes *all* paths

Prefix	Src. Route
1.1.1.0/24	A,D,G
…	…



- Each node…
  - …locally runs operator’s *global* TE algorithm
  - …encodes the set of paths it originates as strict source routes
  - …programs them into its forwarding table

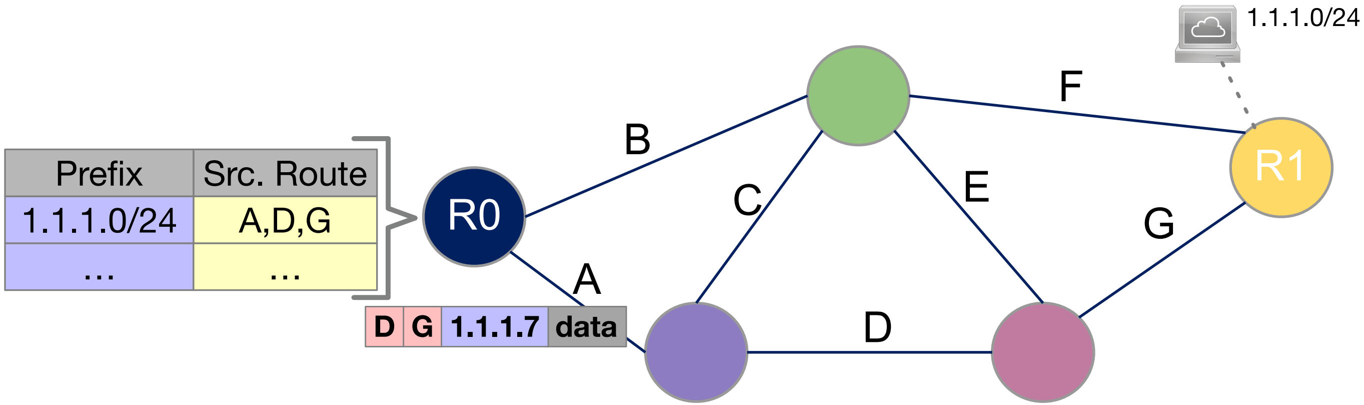
(3) Forwarding follows source route



Prefix	Src. Route
1.1.1.0/24	A,D,G
…	…

- Full source route inserted into packet at ingress node
  - path followed by all transit nodes
  - ⇒ **“consensus-free”** pathing; no coordination across routers to program

(3) Forwarding follows source route

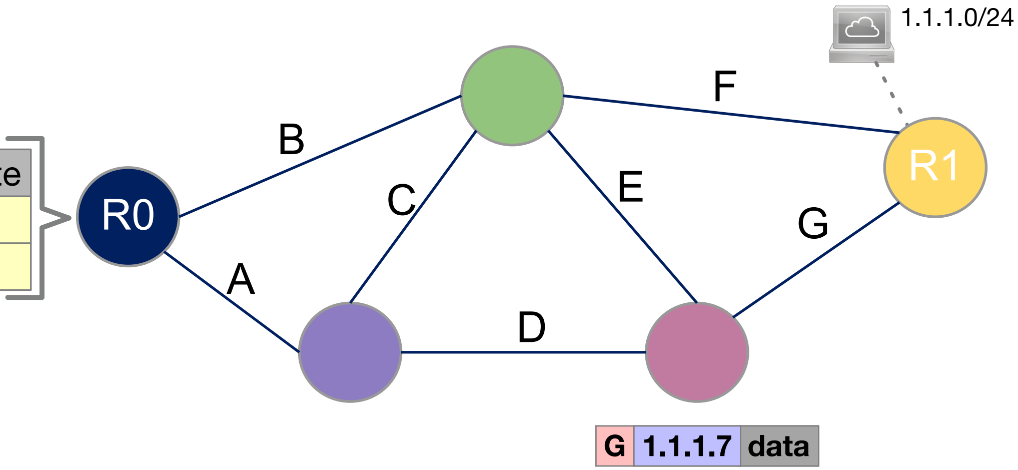


Prefix	Src. Route
1.1.1.0/24	A,D,G
…	…

- Full source route inserted into packet at ingress node
  - path followed by all transit nodes
  - ⇒ **“consensus-free”** pathing; no coordination across routers to program

(3) Forwarding follows source route

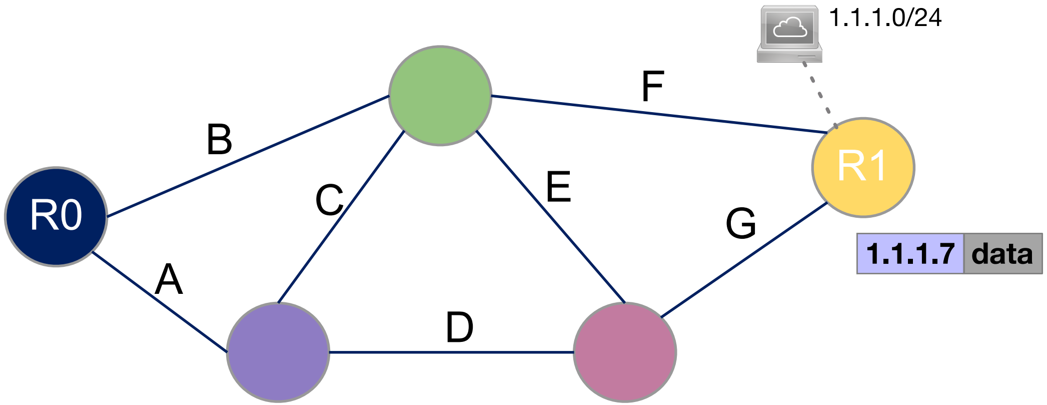
Prefix	Src. Route
1.1.1.0/24	A,D,G
...	...



- Full source route inserted into packet at ingress node
  - path followed by all transit nodes
  - ⇒ **“consensus-free”** pathing; no coordination across routers to program

(3) Forwarding follows source route

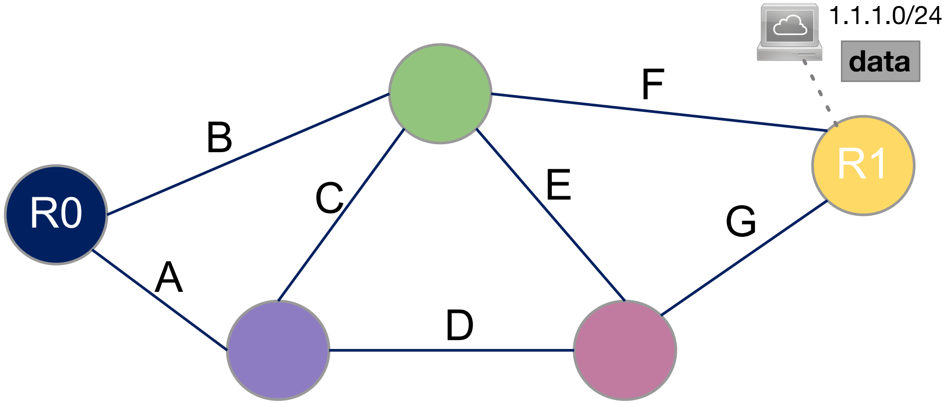
Prefix	Src. Route
1.1.1.0/24	A,D,G
…	…



- Full source route inserted into packet at ingress node
  - path followed by all transit nodes
  - ⇒ **“consensus-free”** pathing; no coordination across routers to program

(3) Forwarding follows source route

Prefix	Src. Route
1.1.1.0/24	A,D,G
…	…

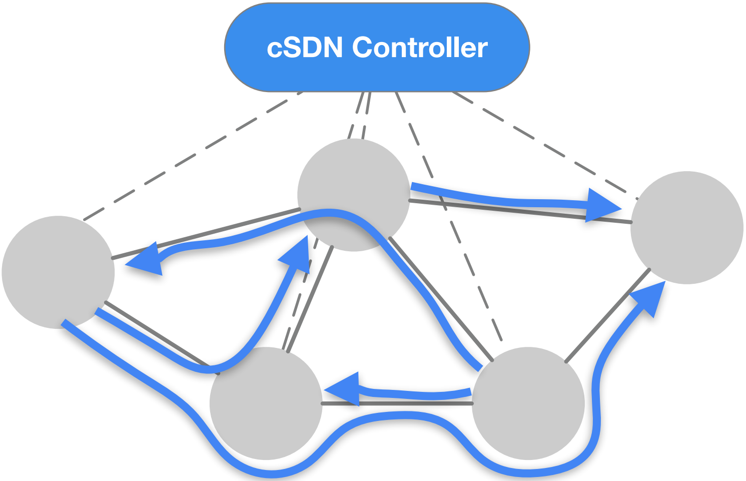


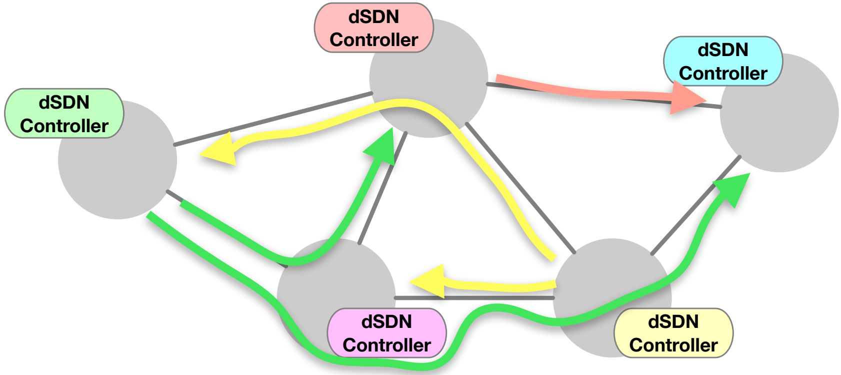
- Full source route inserted into packet at ingress node
    - path followed by all transit nodes
    - ⇒ **“consensus-free”** pathing; no coordination across routers to program

Source routing enables *simple* decentralization

Strict source routing ⇒ ingress router dictates path *authoritatively*

- requires no consensus across transit routers to program path

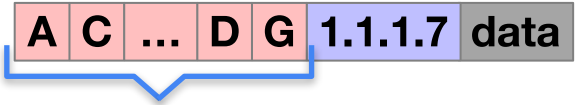




Source routing enables *simple* decentralization

Strict source routing ⇒ must enumerate path in header

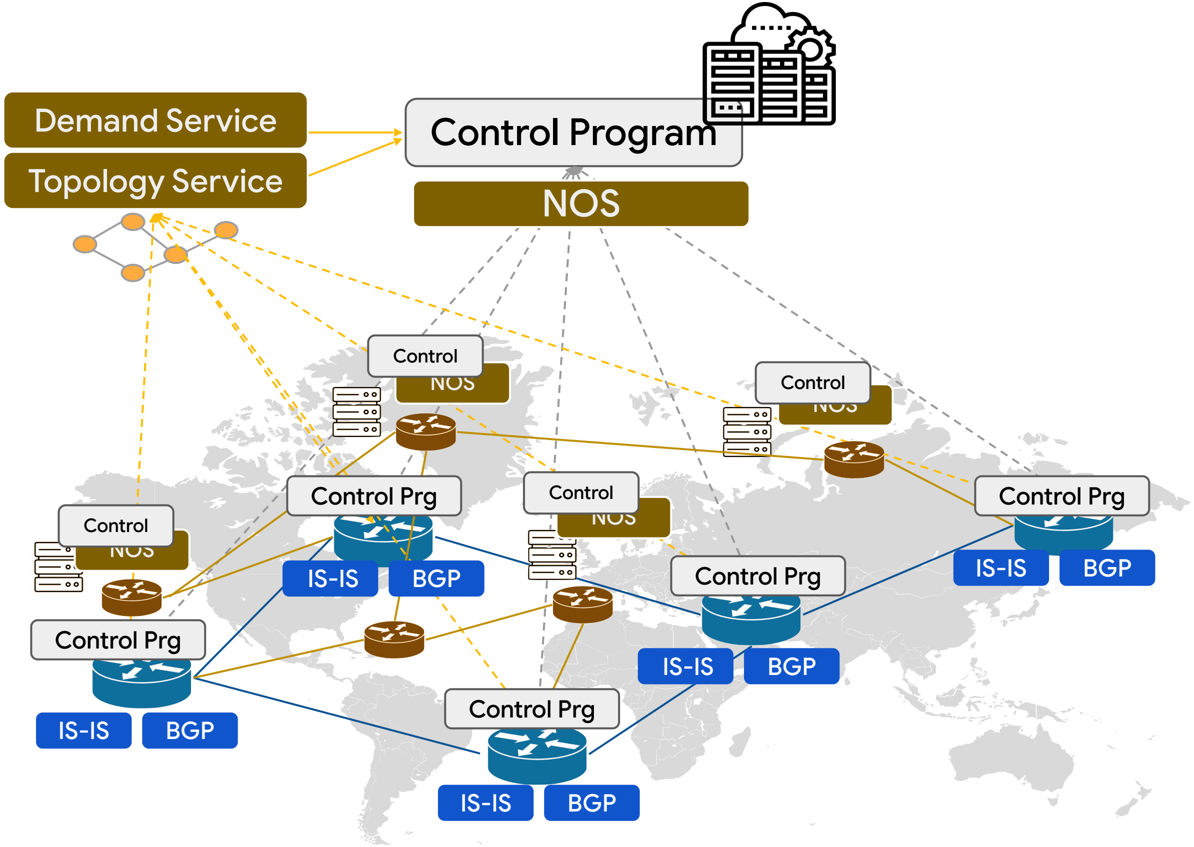
- Historically infeasible due to length of WAN paths



Enabled by hardware advancements and **novel encoding technique**

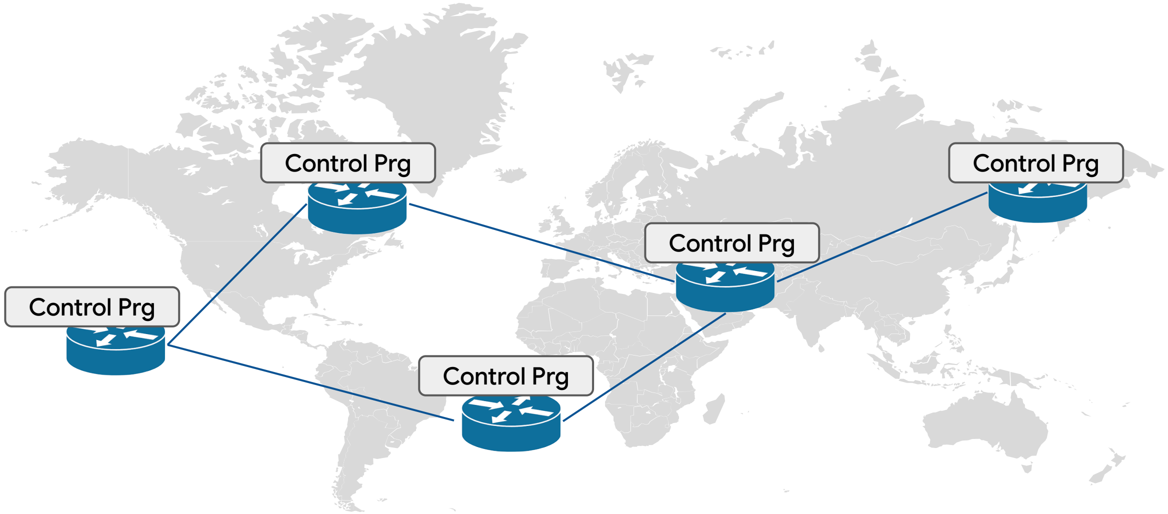
- See paper for details

dSDN cuts out a large fraction of infrastructure



dSDN cuts out a large fraction of infrastructure

- Smaller surface area for bugs
- Greatly decreased number of components on critical path for routing



dSDN achieves the benefits of SDN *and* decentralization

Original SDN Benefits

- **Operator-defined code**, which enabled innovation
  - ✓ *running our own containers on the router*
- **Optimized** computations (TE) on global view of network
  - ✓ *new APIs + simple dissemination → global view*
- **Simplicity** of "consensus free" path selection
  - ✓ *source-routing; "ingress" router authoritatively decides path*

Decentralization Benefits

- Drastically **fewer** external **dependencies**
  - ✓ *control plane running in-band*
- Distributed **survivability**
  - ✓ *no central point of failure*

Simplifying control infrastructure *improves* performance

Primary Goal: cutting complexity from the WAN architecture

Removed	Added
(1) Central controller jobs	On-router containers
(2) Regional controller jobs	
(3) Dedicated server hardware	
(4) Control plane network	
(5) Instrumentation services	
(6) Traditional protocols	

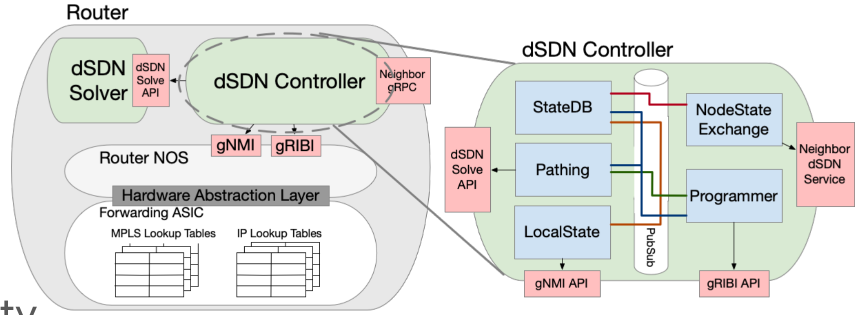
Have we lost performance in the process?

- On the contrary, we find simplification results in *better* performance

Evaluation Performance: dSDN running on real hardware

Methodology*

- ✦ Built production-grade dSDN implementation
  - Profiled on production-grade testlab routers
  - Fed live B4 topology and demand data
- ✦ Profiled cSDN performance in production
- ✦ Replayed production failure events in high-fidelity simulator of cSDN and dSDN



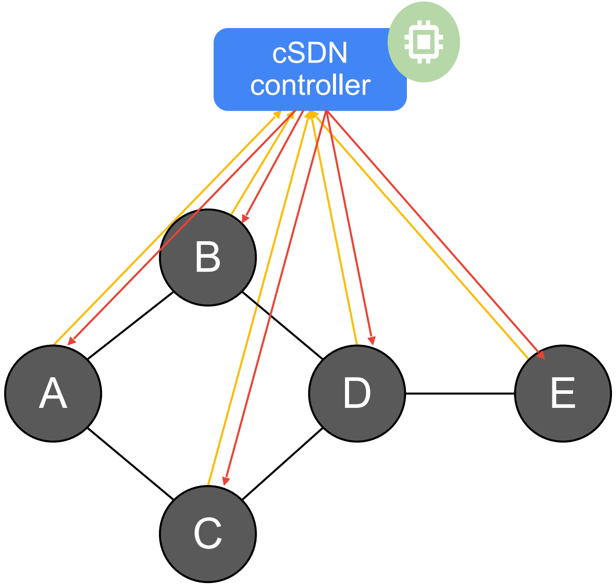
dSDN Implementation Architecture

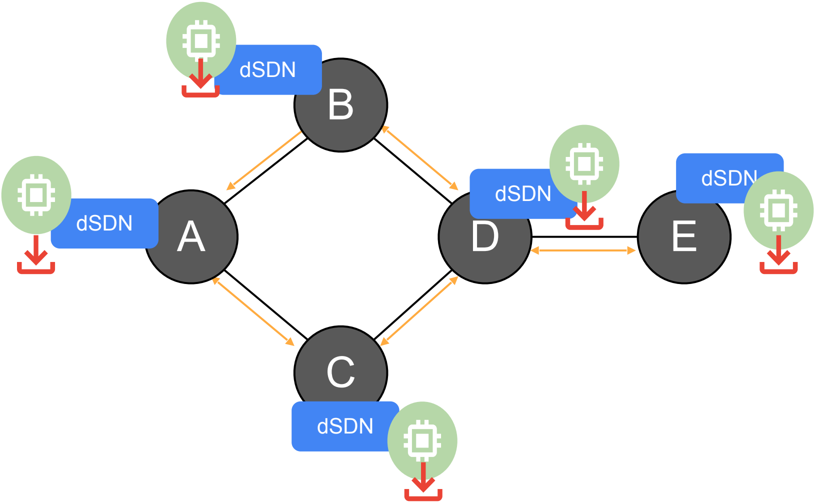
Evaluated (1) *convergence time* and (2) *convergence impact*

*See paper for more details

A closer look: convergence

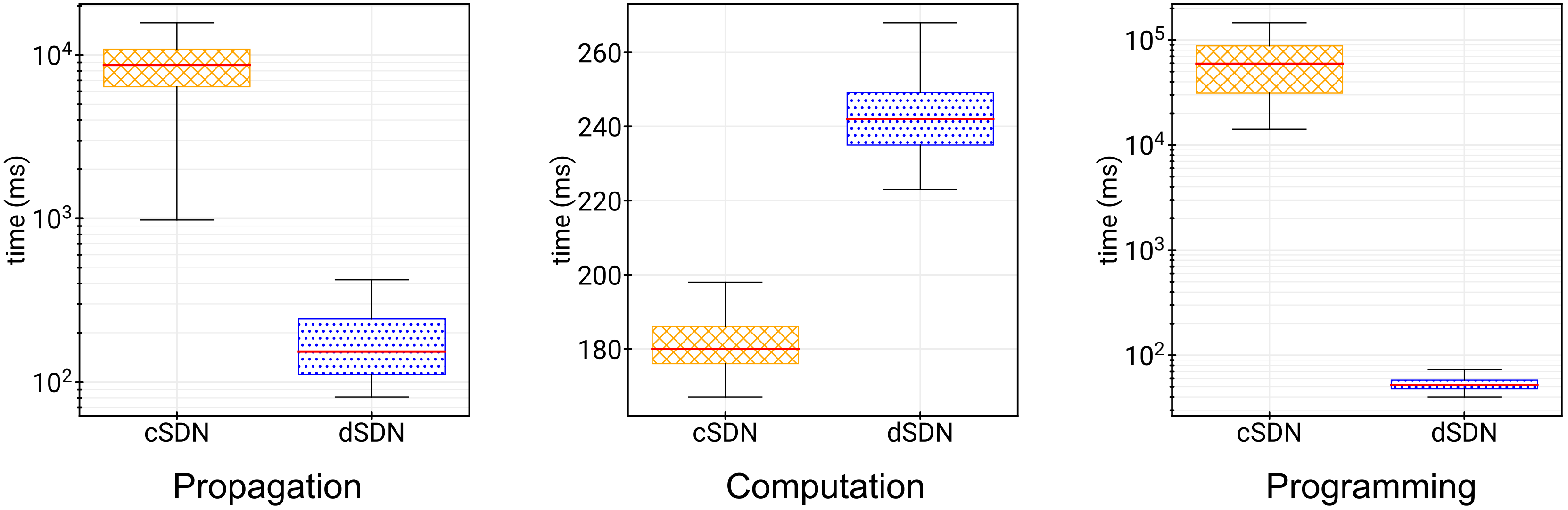
Three components: (1) propagation, (2) computation, (3) programming





Convergence time: 120x-150x faster

Note differing Y axes



dSDN has much faster propagation and programming, but *slower* computation

Convergence impact: significantly lower for all priorities

Convergence impact comprises both amount of time and number of flows affected

- **Bad seconds** metric defined in the paper, intuitively covers both

	cSDN	dSDN
Highest Priority	146.9	2.23
Lowest Priority	1122.9	57.3

Bad Seconds at the 98th Percentile

See paper for more details

dSDN sees big win in impact due to shorter convergence time

Further details

See paper for discussion on additional questions…

- ✦ Does dSDN scale?

  Yes, paper discusses forward-looking projections, TE optimizations
- ✦ Does dSDN handle high churn in network state?

  Yes, paper evaluates up to 20x production churn rate
- ✦ Are there other benefits to having control loops on the router?

  Yes, paper demonstrates additional applications of on-router control
- ✦ Can source routing work in very large networks?

  Yes, paper presents supporting techniques

Decentralized SDN: a new point in the design space

- We've spent...
  - 30 years trying to fix and evolve distributed protocols
  - 15 years trying to make SDN-based networks more reliable
- We present **dSDN**, a new point in the design space
  - achieves the best of both by decentralizing the SDN controller in a way that...
    - maintains benefits of SDN
    - significantly ***simplifies*** the control plane infrastructure
    - improves convergence performance

Paper Link

Alexander Krentsel – akrentsel@google.com