WAN Networking: TE & SDN

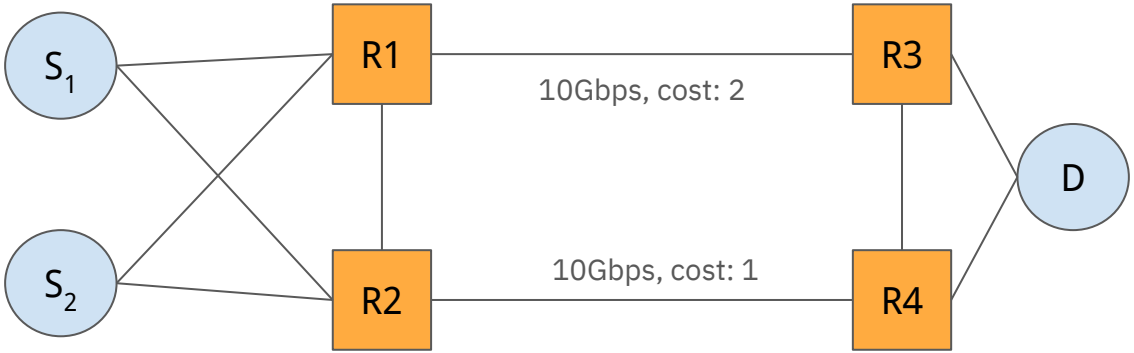
Autumn 2024
cs168.io

Rob Shakir

Today

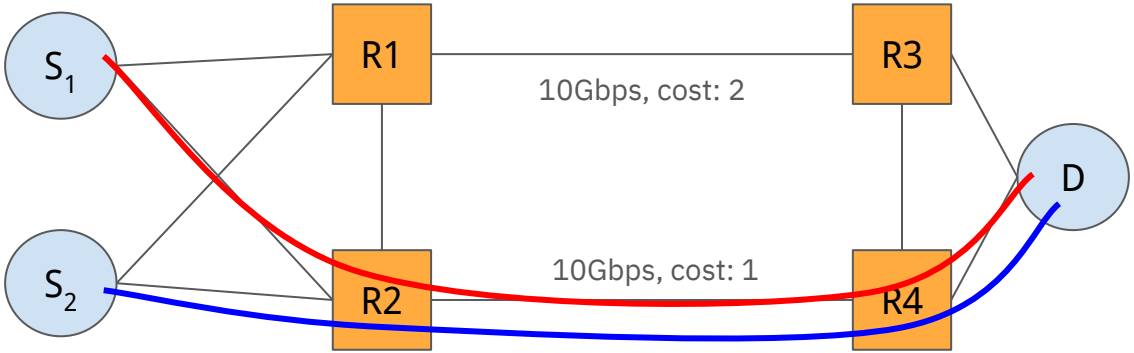
- We'll talk about some of the challenges that wide-area networks (WANs) experience.
  - Particularly, making efficient use of capacity.
- We'll consider some of the limitations of the routing protocols that we've discussed – and look at alternatives.
- We'll understand the move towards software-defined networking.
  - What is SDN?
  - How did it solve some of the WAN networking problems for ISPs?

Traffic Engineering



$S_1 \rightarrow D = 10Gbps$
$S_2 \rightarrow D = 8Gbps$

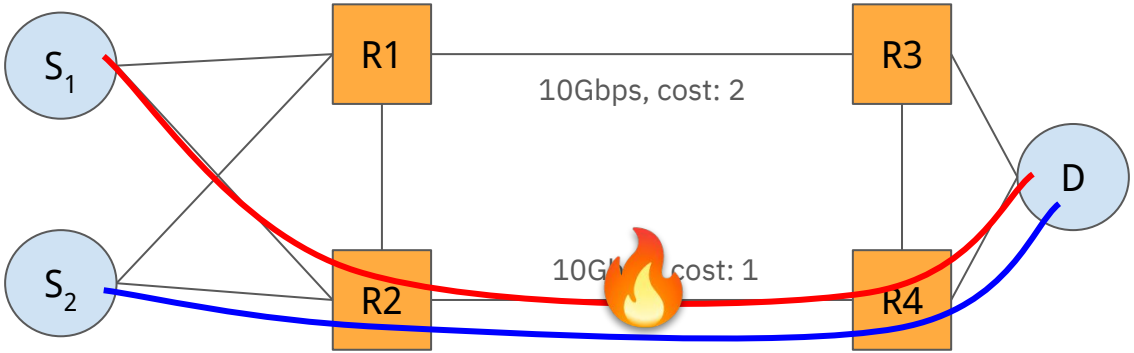
Traffic Engineering



$S_1 \rightarrow D = 10Gbps$

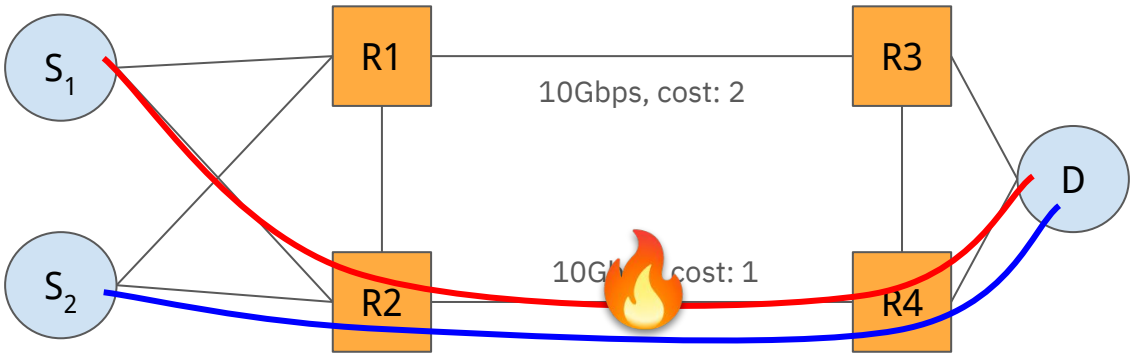
$S_2 \rightarrow D = 8Gbps$

Traffic Engineering



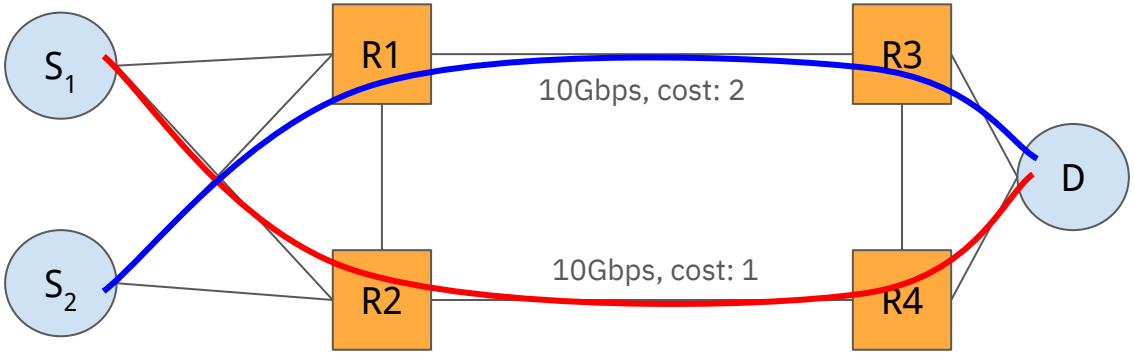
Route selection *without* traffic engineering results in congestion.

Why not just congestion control?



Network utilisation is a performance metric of concern:
Total $S_{[12]} \rightarrow D$ capacity: 20Gbps, used with shortest path: 10Gbps
Can we do better?

Traffic Engineering



Using traffic engineering – we can tell S_2 to send traffic over a higher cost path.

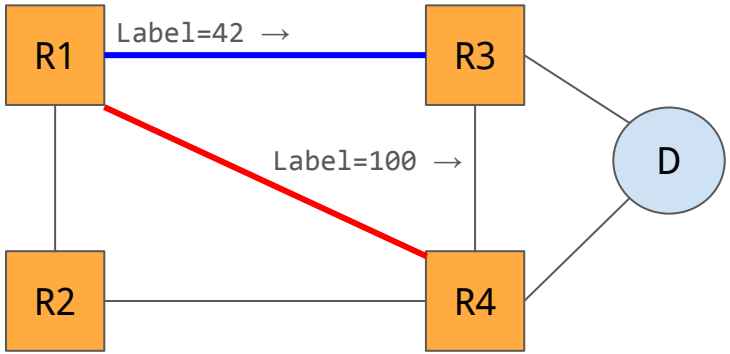
Traffic Engineering - cSPF

- Rather than saying “traffic should be on the shortest path”.
  - i.e., vanilla shortest-path calculation
- Say “traffic should be on the shortest path that has sufficient capacity”.
  - Or meets some other criteria (latency etc.)
- This introduces a constraint to our shortest path calculation.
  - Hence constrained Shortest Path First.

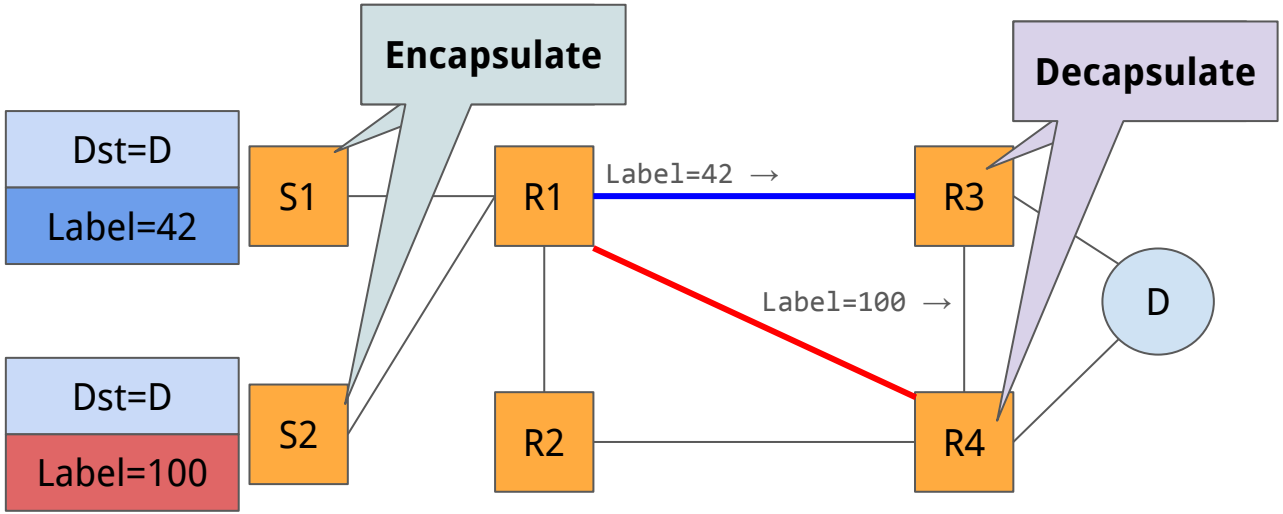
Traffic Engineering: Forwarding

- Uses **encapsulation** like was used in the datacenter.

- Encapsulating with a new header that indicates the path to be taken through the network allows traffic engineering – and hides the ultimate packet destination.



Using encapsulation to achieve TE



By **encapsulating** the packet, we can tell R1 to route based on our new header, even though both packets are to the same destination.

Questions?

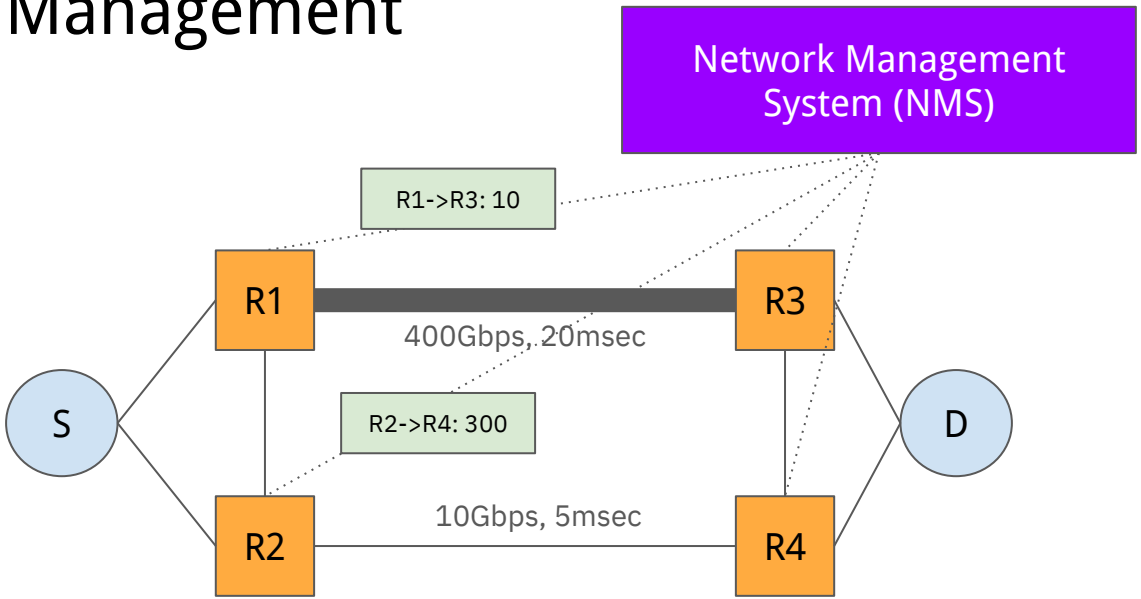
Challenge: How do we inform the network of our policy?

- We have defined a policy.
  - e.g., encapsulate with label 100 at S_1
- Typically, we have said that operators configure the network.
  - e.g., specify link costs, assign IP addresses to links, specify static routes.
- Can we simply configure the network to use specific paths?
  - Using the same approach to configuration via the management plane.

The Management Plane

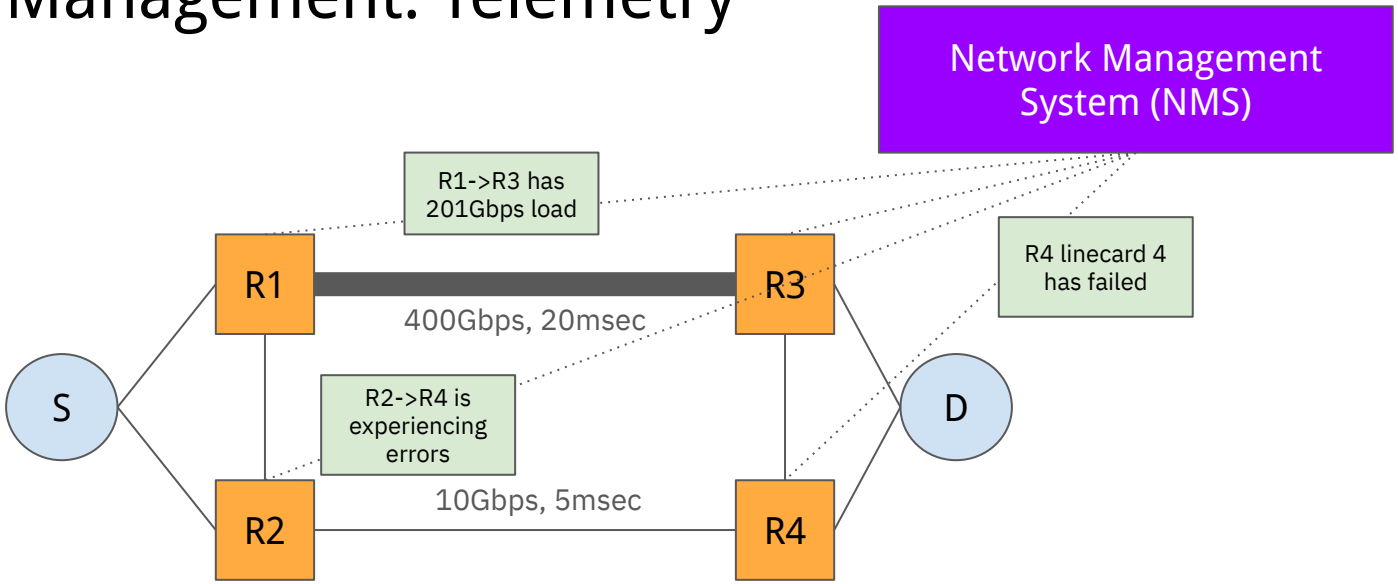
- Recall:
  - Data Plane - packet forwarding - O(nanoseconds)
  - Control Plane - real-time, populates routing state on network devices – O(1 second)
- Management plane.
  - Configuration of network device functions (including routing protocols).
  - Monitoring – statistics, alarms etc. required to run the network.
- The management plane is generally how operators tell routers what to do, and see what they are doing.
- O(10s to 100s of seconds)

Network Management



A network management system allows us to generate and configure parameters such as link costs.

Network Management: Telemetry

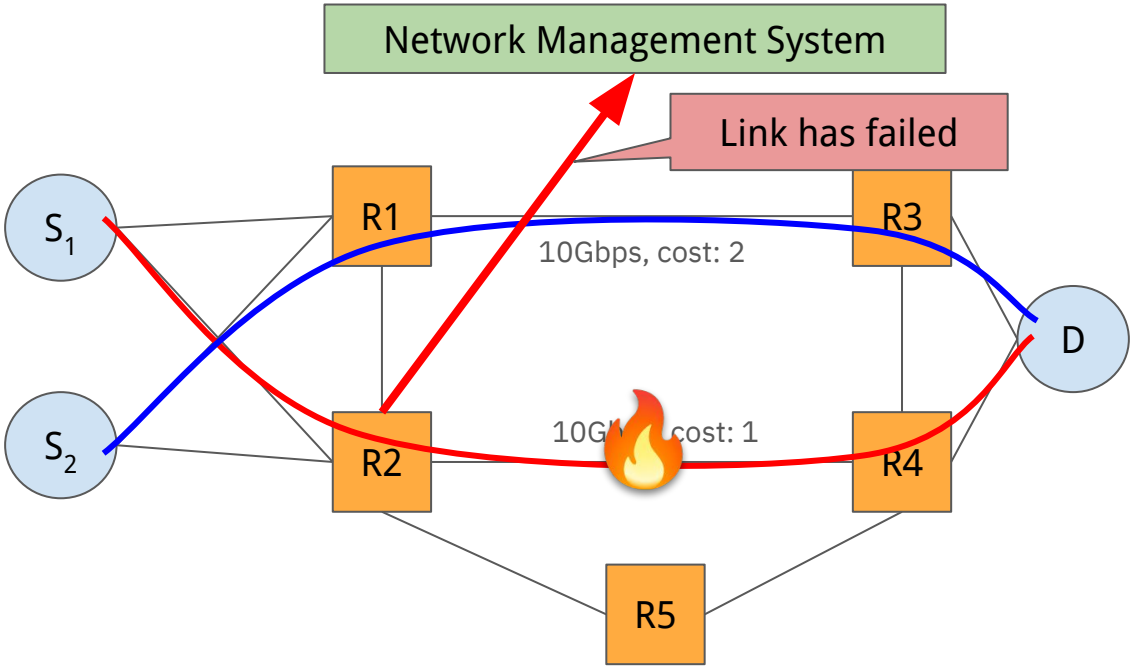


A network management system also allows statistics and events to be read from routers.

Was the management plane a feasible solution?

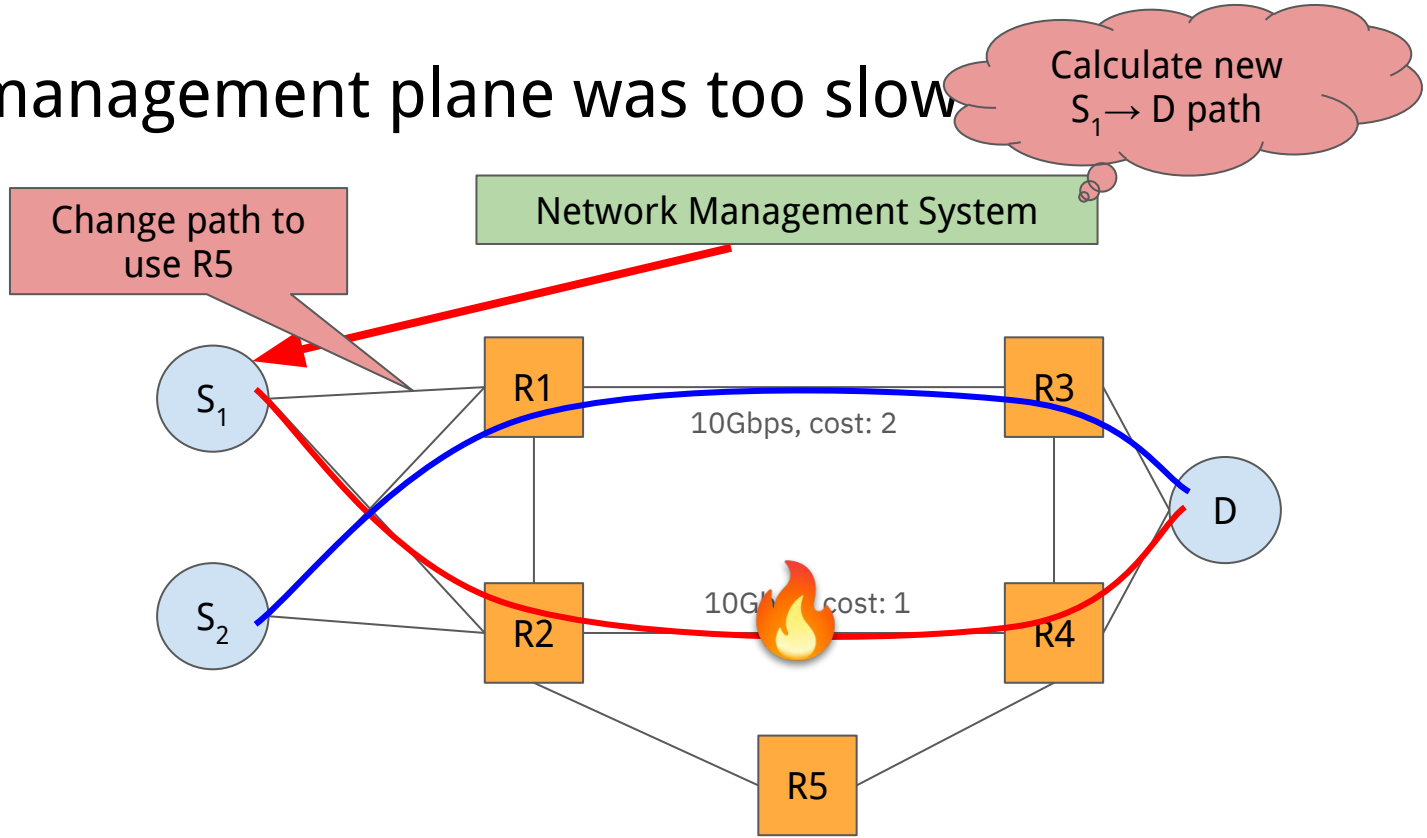
- There has been less focus on the management plane than other areas of networking.
  - Even though it is critical for network operation!
- Slow evolution towards using scripts to be able to programmatically control network.
  - Allowed automation of processes (e.g., adding routers and links)
  - Started to allow automation of repair of the network when things went wrong.
- But, it was the bottleneck for many network operations.

The management plane was too slow!



Telemetry updates often sent only every 5 minutes.

The management plane was too slow



NMS not designed for real-time communication.
Management (configuration) changes could take minutes.

Management Plane != Real Time

- The management plane of networks has generally not been considered to be real-time.

- Applications like traffic-engineering are real-time.
  - If we don't make a change when a failure happens, then routing is invalid – and we have dead-ends.
  - Doesn't meet the performance requirements of network applications.

- Applies to normal routing too.
  - If we had a management application that could change "static" routes in real-time, we might not need routing protocols.

Questions?

An alternative: control-plane protocols.

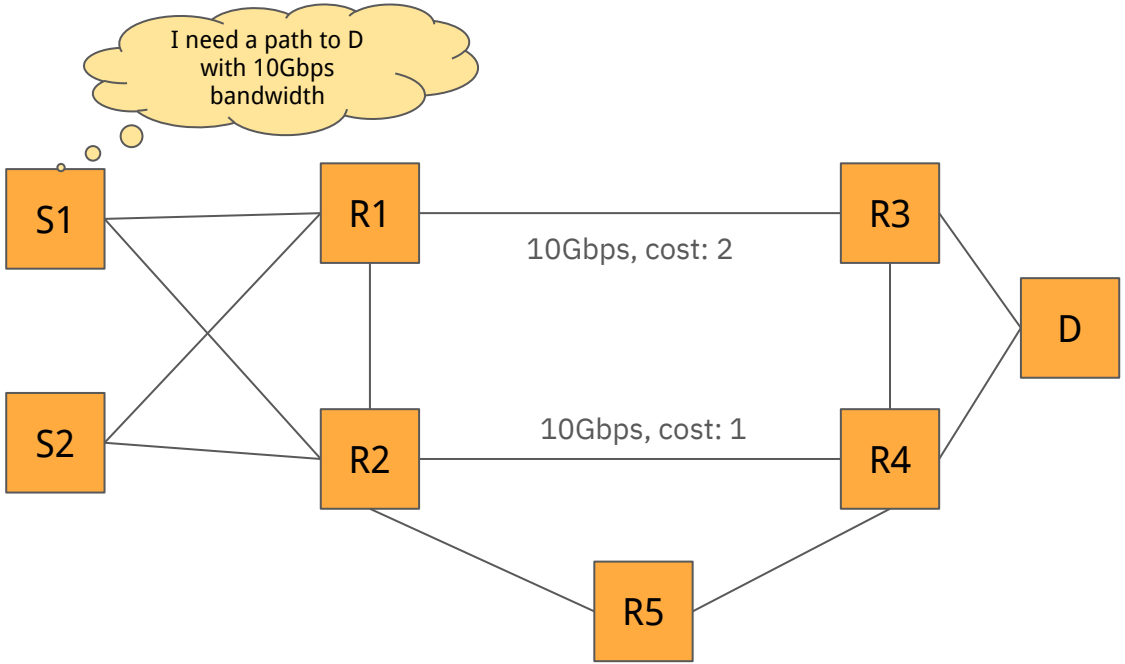
- Traffic engineering needs end-to-end (router-to-router) paths to be understood by the network.

- Allows:
  - Tracking how much bandwidth will be used by A→B, and C→B on each link independently.
  - Tracking any constraints that exist for A→B but not C→B (e.g., avoid geopolitical region A).

- So, a natural answer would be to invent a new type of routing protocol.
  - Note, we actually thought about one that would do this before – which was connection-oriented.

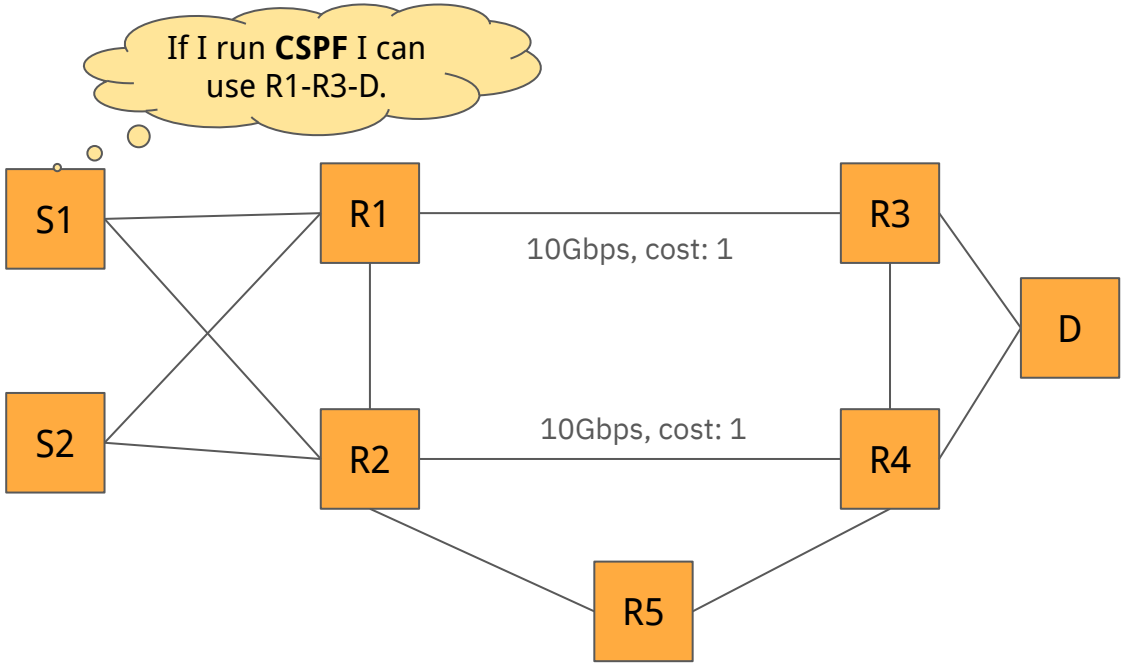
- Industry solution was **R**e**S**ource reser**V**ation **P**rotocol for **T**raffic **E**ngineering – RSVP-TE.

Basic RSVP-TE Operations



Source router is configured with the paths that it needs – i.e., $S_1 \rightarrow D$ – and their constraints (e.g., 10Gbps)

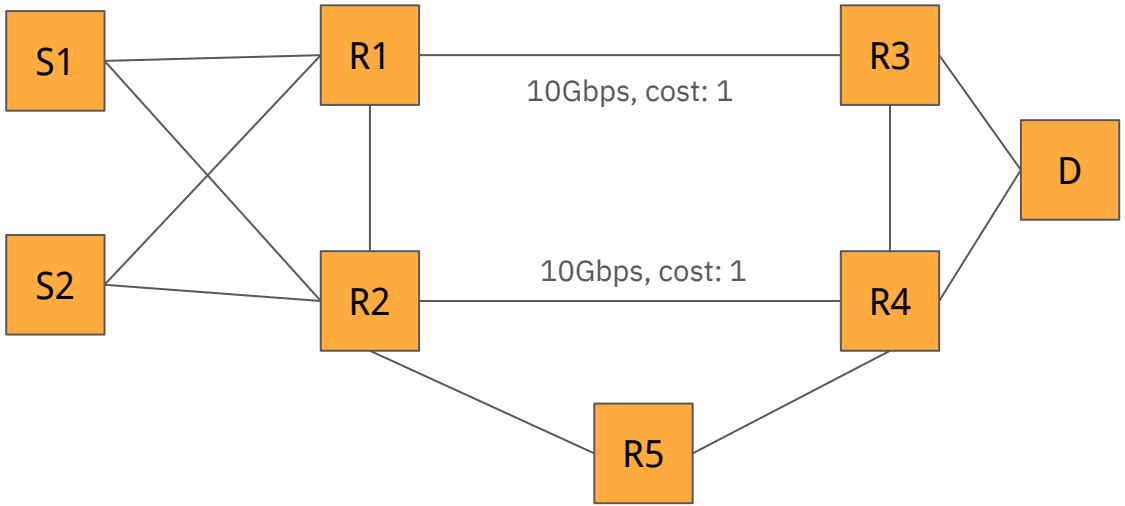
Basic RSVP-TE Operations



Source router knows the network topology (via a link state protocol) - and can run CSPF.

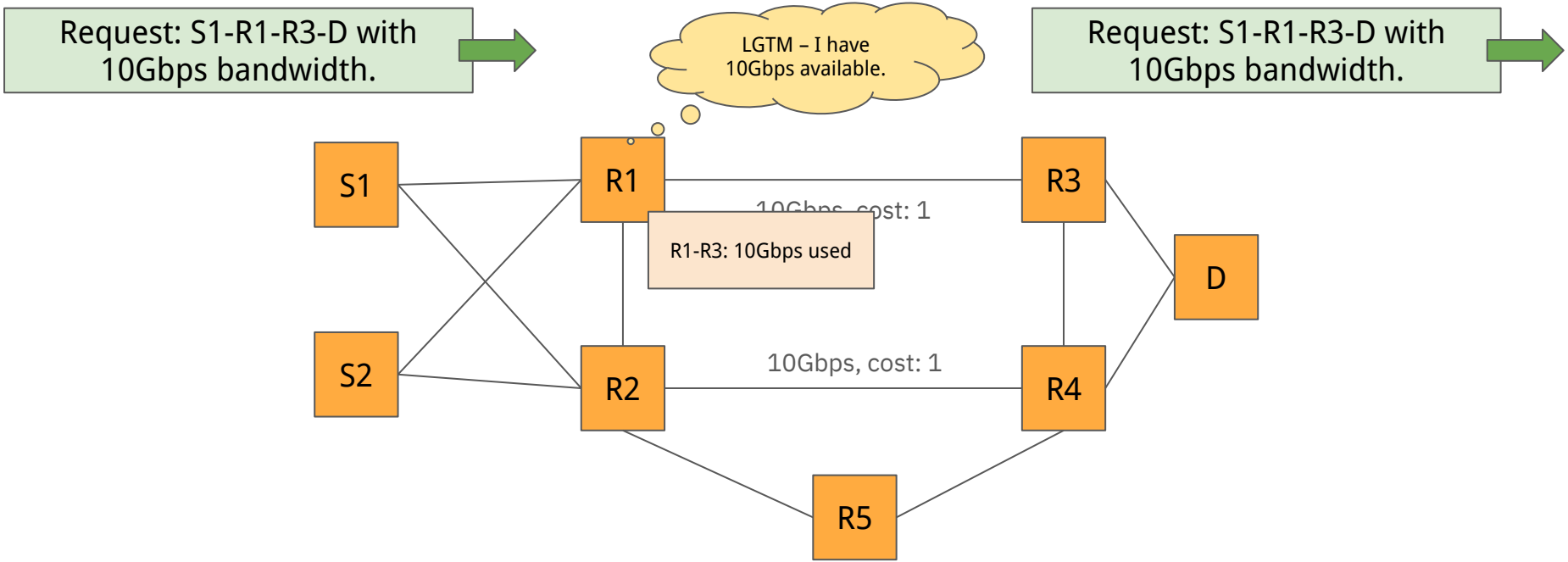
Basic RSVP-TE Operations

Request: S1-R1-R3-D with 10Gbps bandwidth.



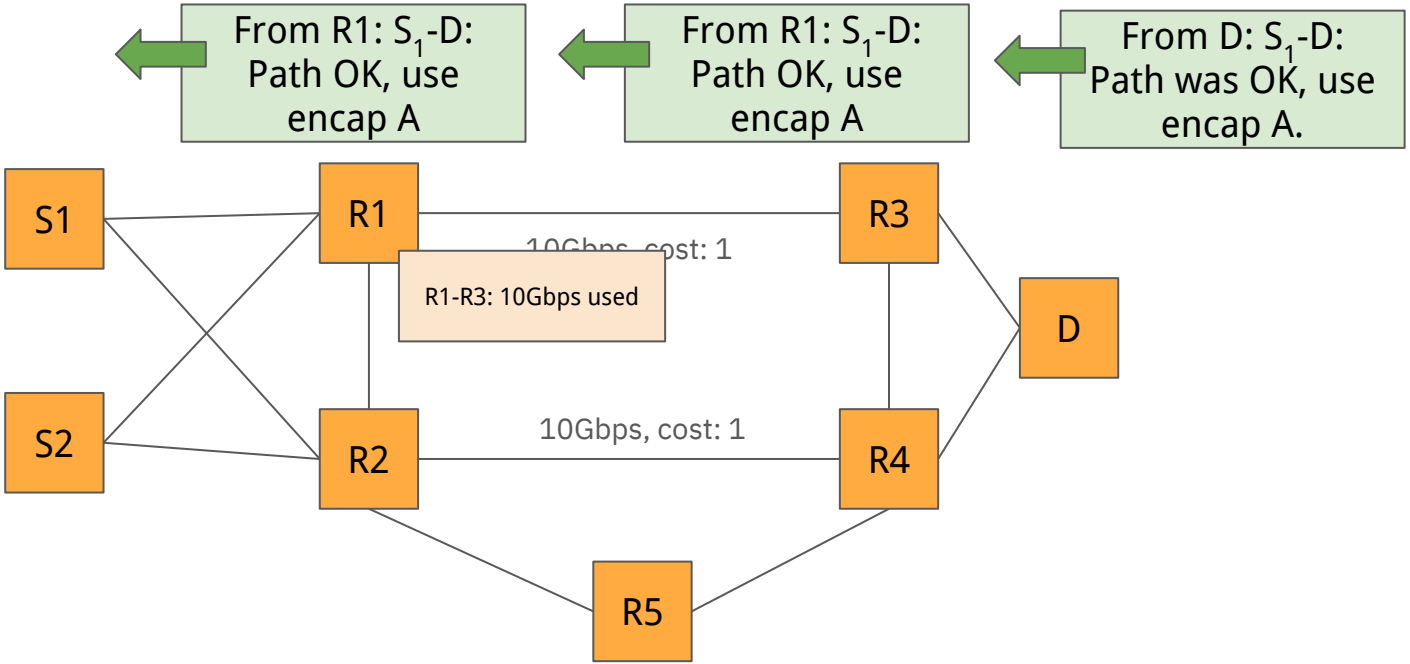
Source router knows the network topology (via a link state protocol) - and can run CSPF.

Basic RSVP-TE Operations



Each router checks whether it has the available capacity – and forwards the request if so.

Basic RSVP-TE Operations



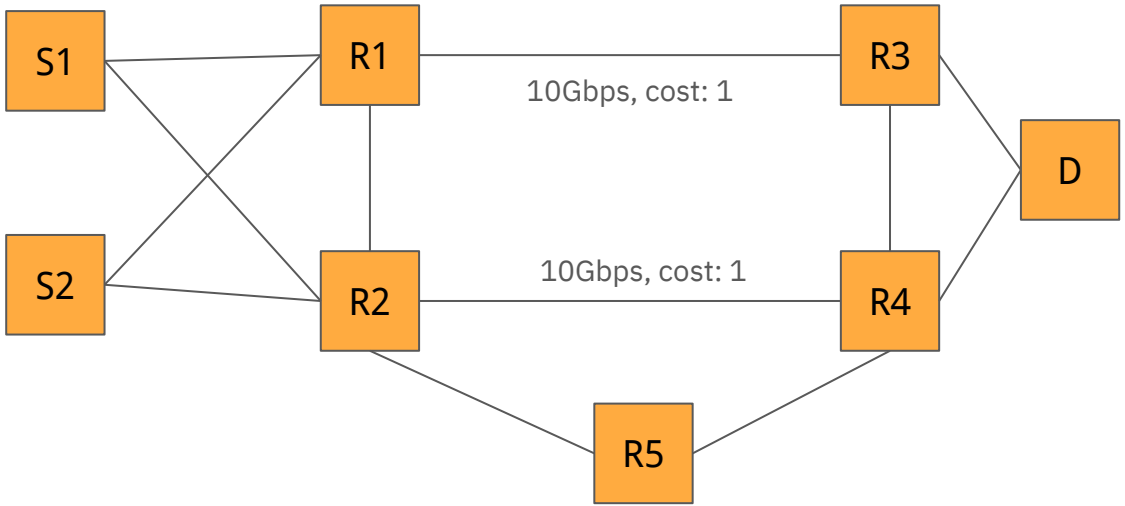
When paths are accepted the source router (“head end”) is informed.

Basic RSVP-TE Operations

Dst	NH
A	R3

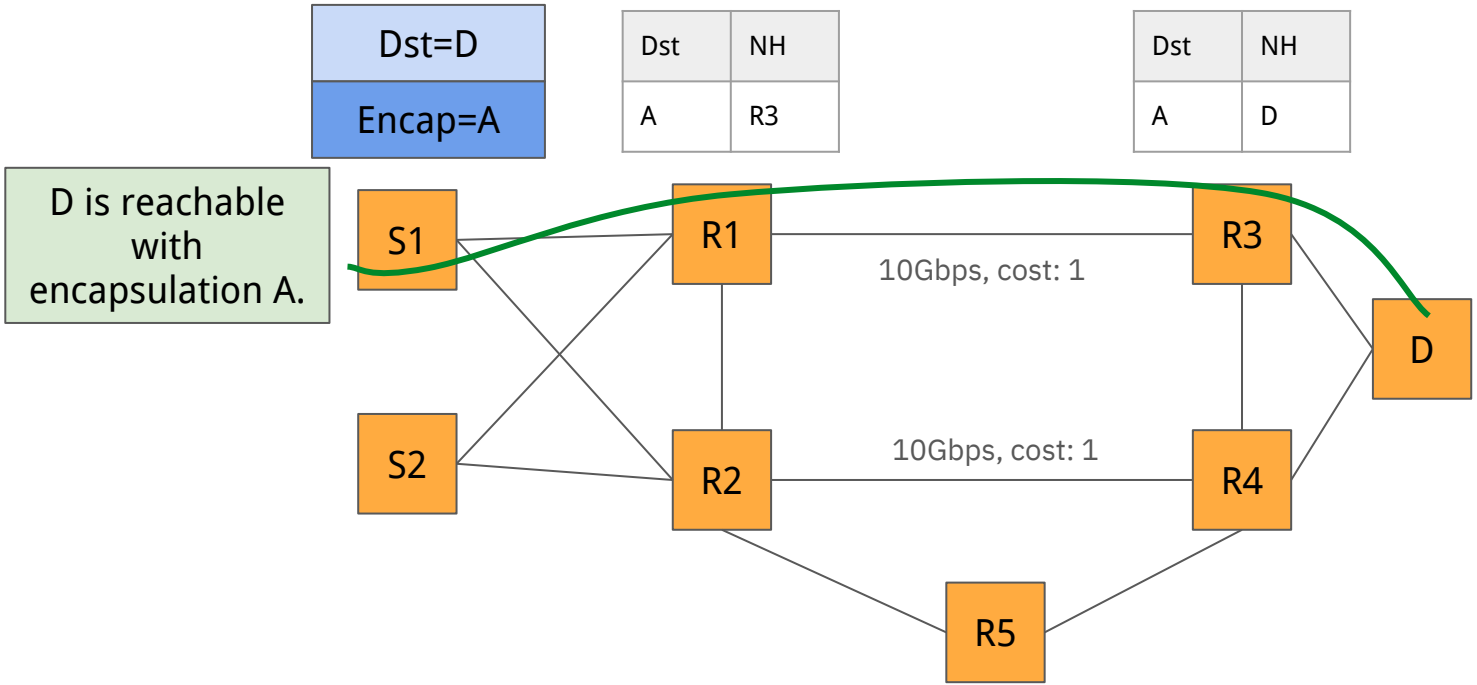
Dst	NH
A	D

D is reachable with encapsulation A.



The “head-end” then installs a route with encapsulation that indicates the specific source-destination path.

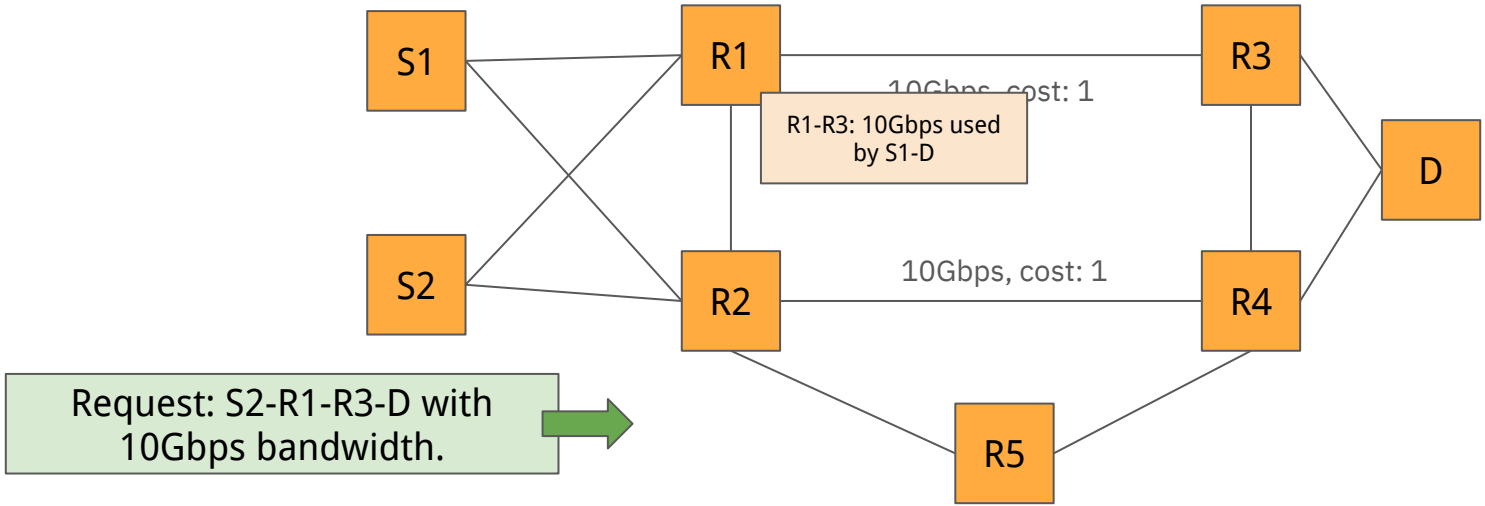
Basic RSVP-TE Operations



The “head-end” then installs a route with encapsulation that indicates the specific source-destination path.

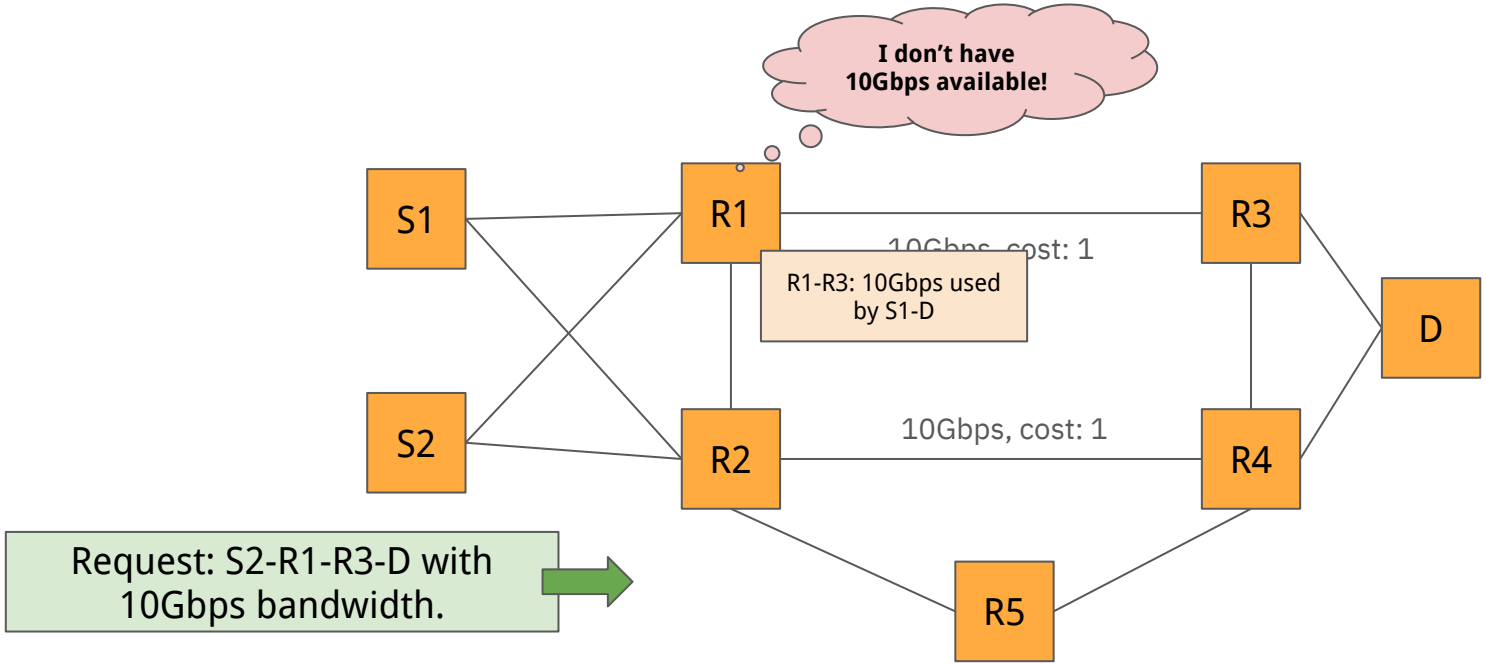
Questions?

Basic RSVP-TE Operations



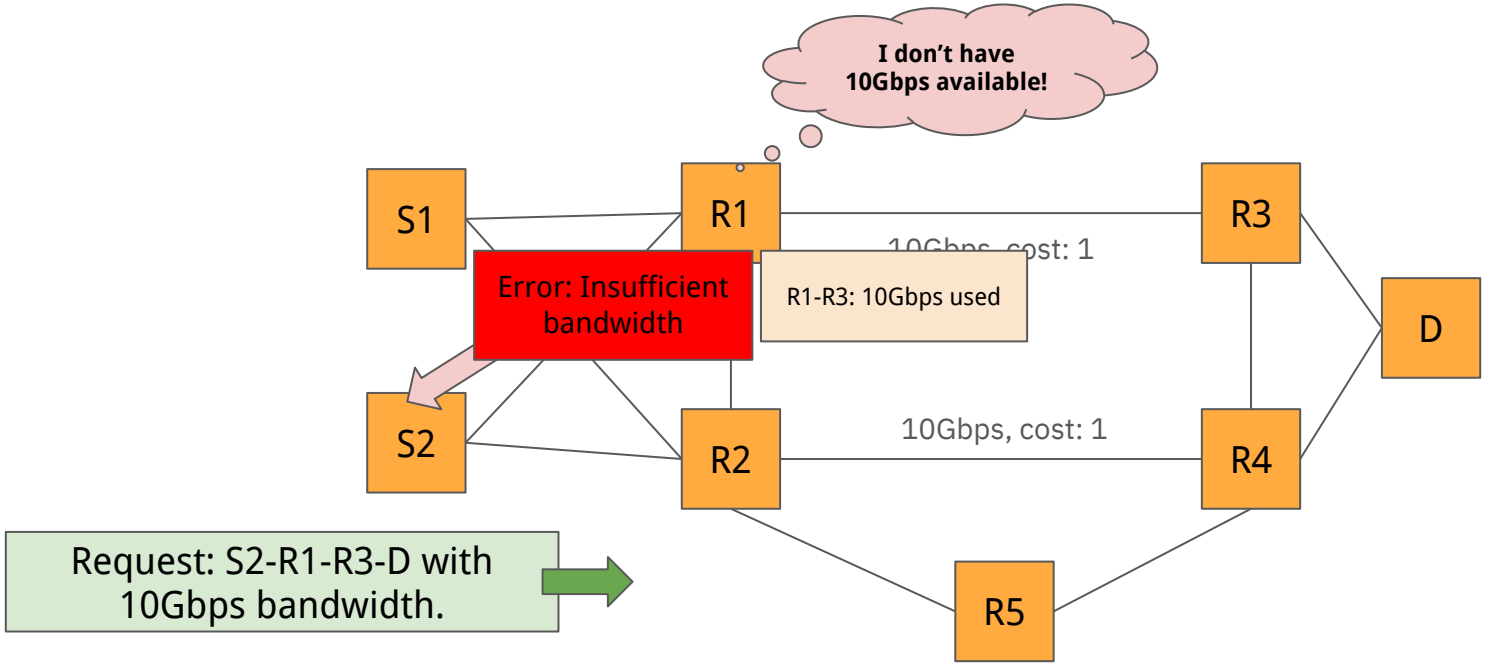
Routers keep state as to the paths that they have accepted on the network.

Basic RSVP-TE Operations



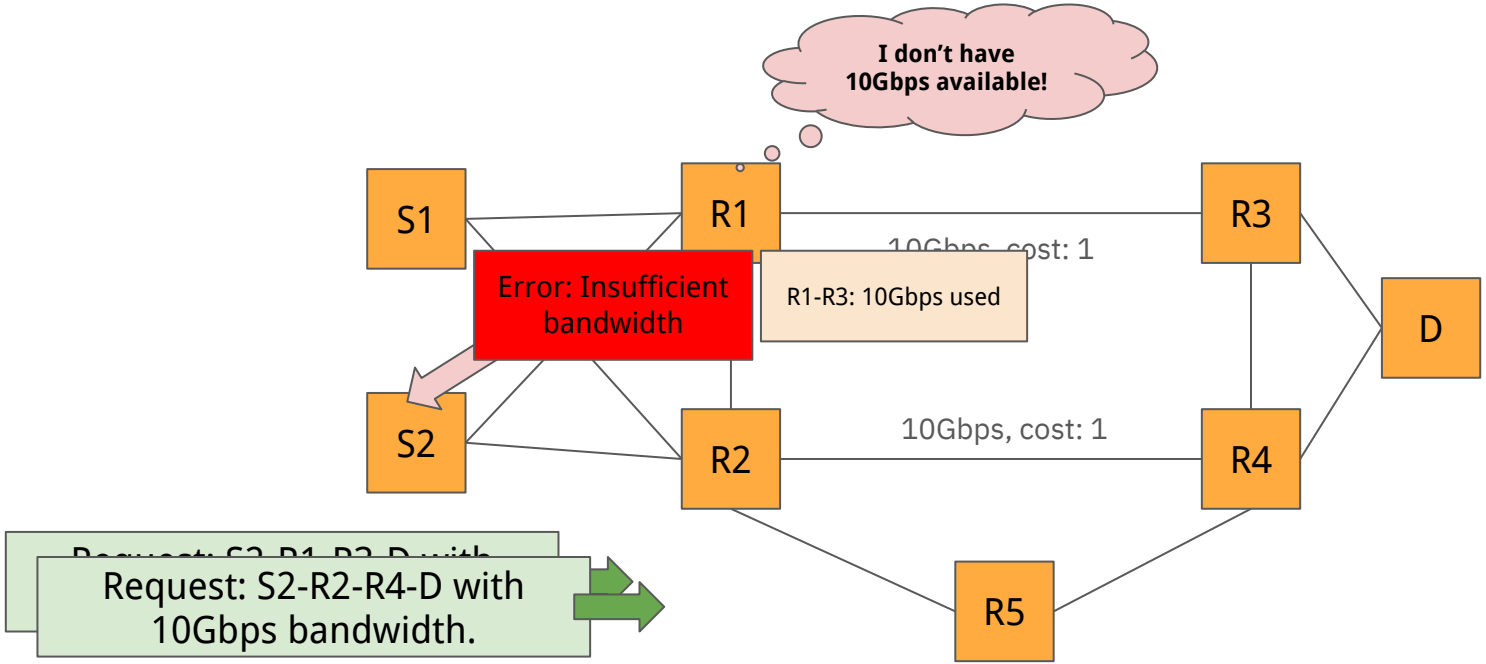
Routers keep state as to the paths that they have accepted on the network.

Basic RSVP-TE Operations



…and return errors if there is insufficient capacity, allowing S2 to recalculate an alternate path with capacity.

Basic RSVP-TE Operations



…and return errors if there is insufficient capacity, allowing S2 to recalculate an alternate path with capacity.

Downsides of a distributed control-plane protocol.

- Lots of new computation needed.
  - Run CSPF.
  - Send messages for each reservation which every router needs to respond to.
  - Deal with sending errors when a failure happens.

- All on a per-source-destination path basis.
  - Routing protocols did not have to care about sources before.
  - They just provided updates about when destinations change.

- Scaling issues.
  - Routing convergence is slowed down, and sometimes networks do not converge.
  - Lots of operational overhead! [Rob first gave a lecture on this 10 years ago!]

Questions?

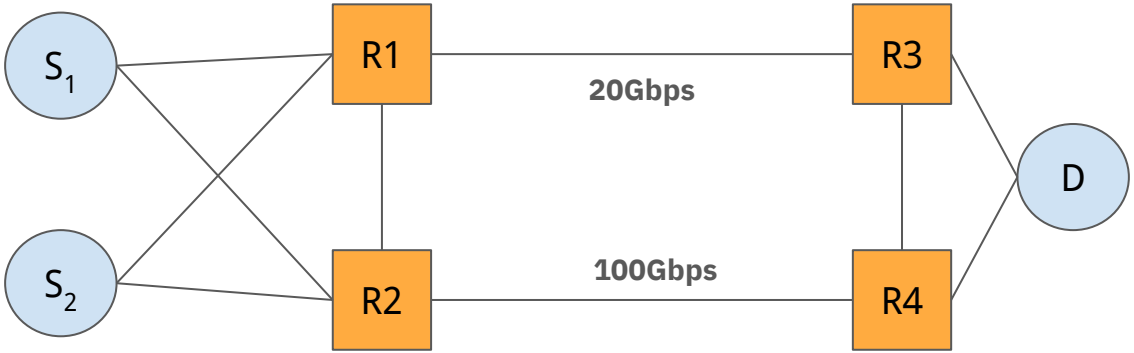
Another alternative: centralise.

- Try and make a control-plane (i.e., realtime) approach that allows:
  - Learning the state of the network.
  - Calculations of the best paths to use.
  - Programming those paths into the network.

- This would be a centralised control plane.
  - Historically, non-IP networks have used these.
  - But they have tended to be slower to converge.

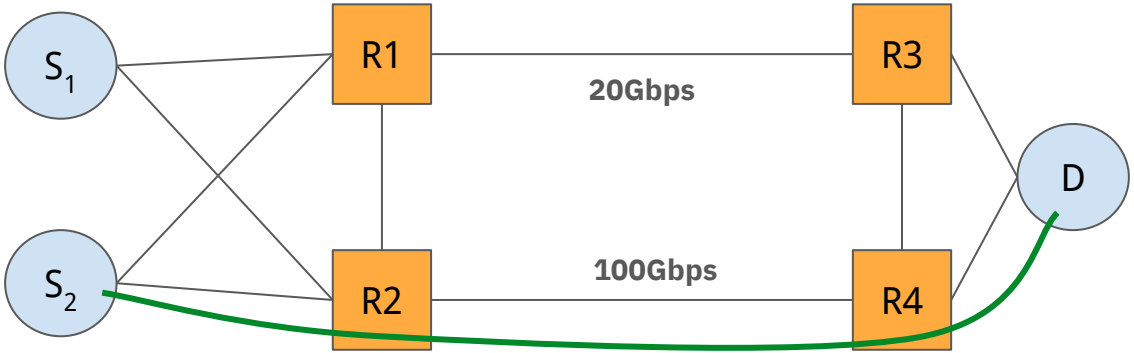
- There are other advantages of centralisation for TE.

Traffic Engineering - why centralise?



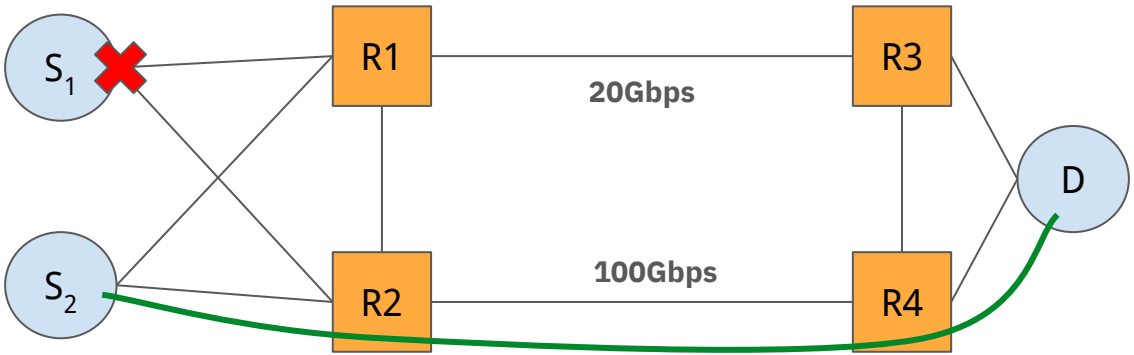
$S_1 \rightarrow D = 100Gbps$
$S_2 \rightarrow D = 20Gbps$

Traffic Engineering - why centralise?



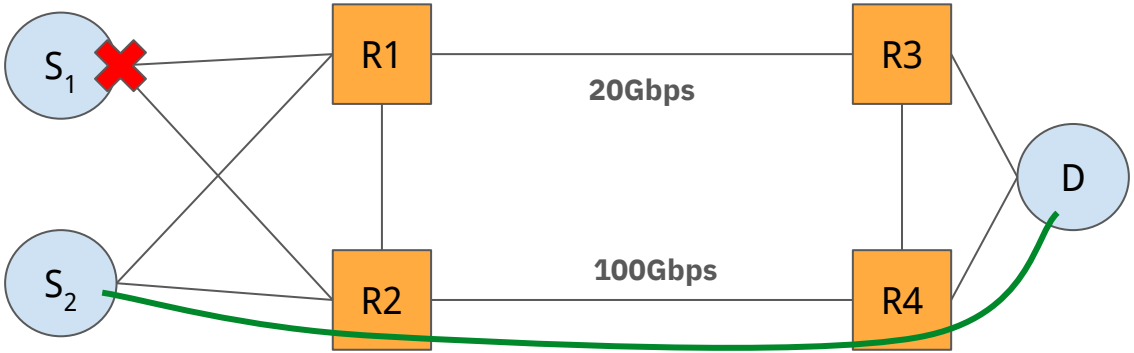
S_2 chooses to place its traffic via the highest bandwidth link – a **locally optimal** decision**.**

Traffic Engineering - why centralise?



If each node acts independently, may not be able to place demands on the network.

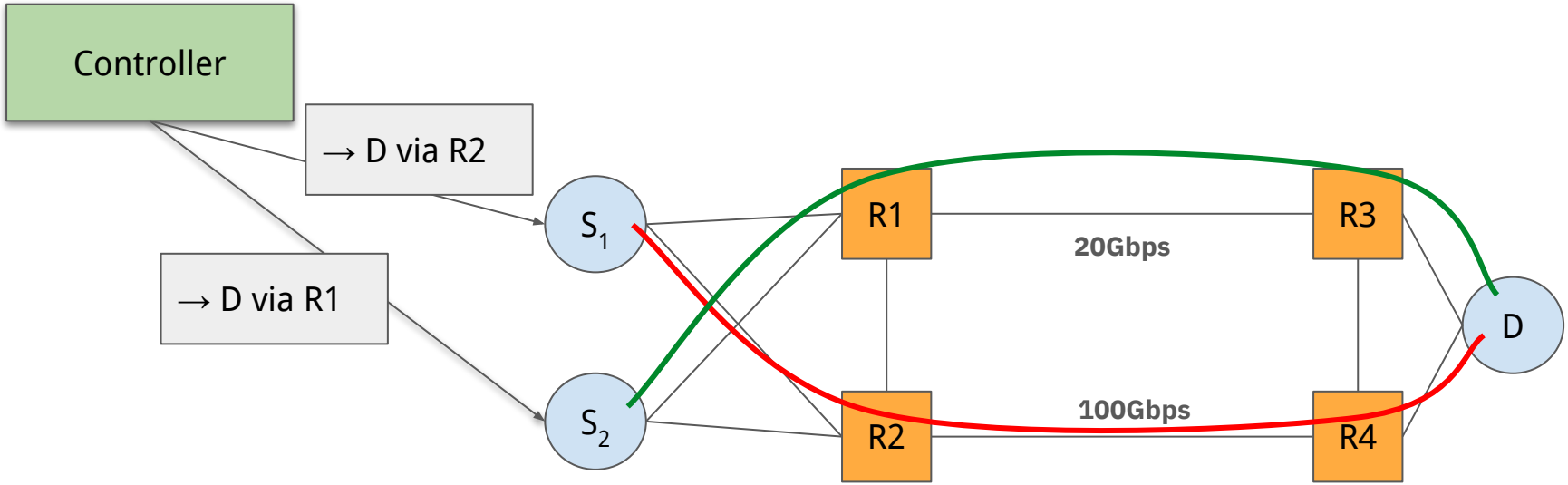
Traffic Engineering - why centralise?



$S_1 \rightarrow D = 100Gbps$

$S_2 \rightarrow D = 20Gbps$

Traffic Engineering - why centralise?



Introducing a controller allows **globally** optimal decisions to be made — increasing network efficiency.

Beyond Capacity Utilisation

- Off-router traffic engineering controllers can consider additional attributes and routing policies.

- Geofencing.
  - “Do not send traffic via links that are in Canada”.

- Path diversity.
  - Make $S_1 \rightarrow D$ and $S_2 \rightarrow D$ go via paths that never intersect.
  - Useful in applications where two paths are backups for each other – e.g., broadcast TV.

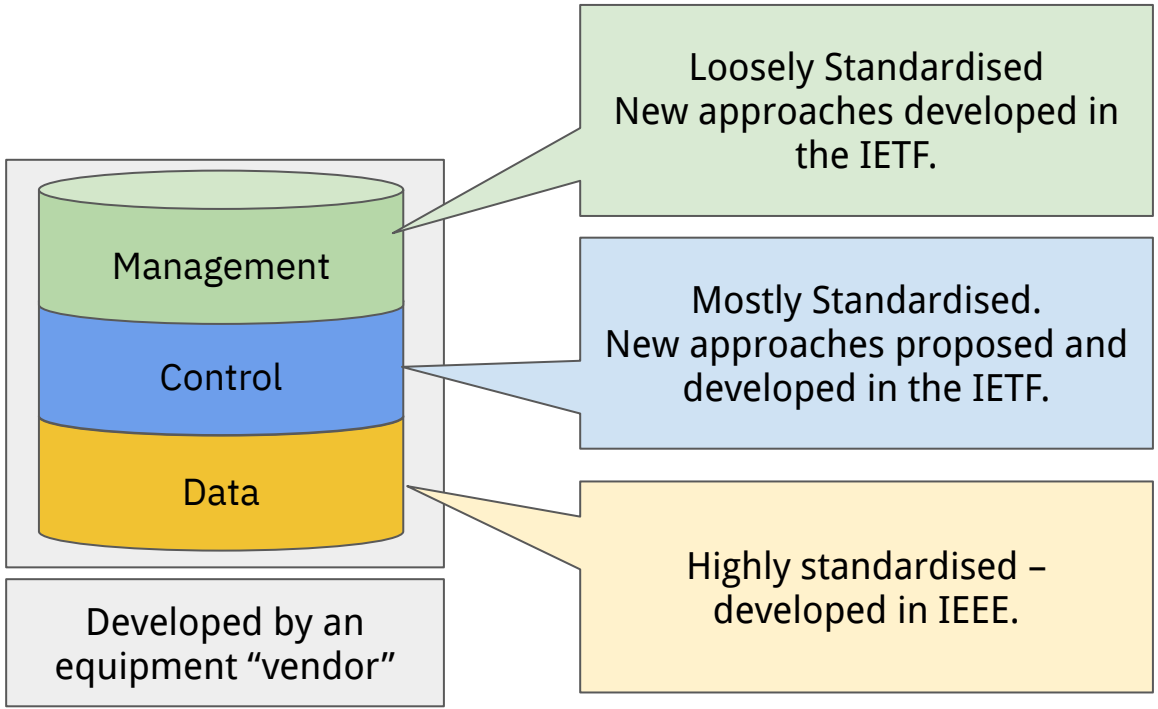
Questions?

Getting to a centralised TE control plane.

- Building a centralised traffic engineering solution needs:
  - A controller – able to calculate paths and program them.
  - Support on network devices to program these entries in real-time.

- Implementing this needs us to think about how network device development works.

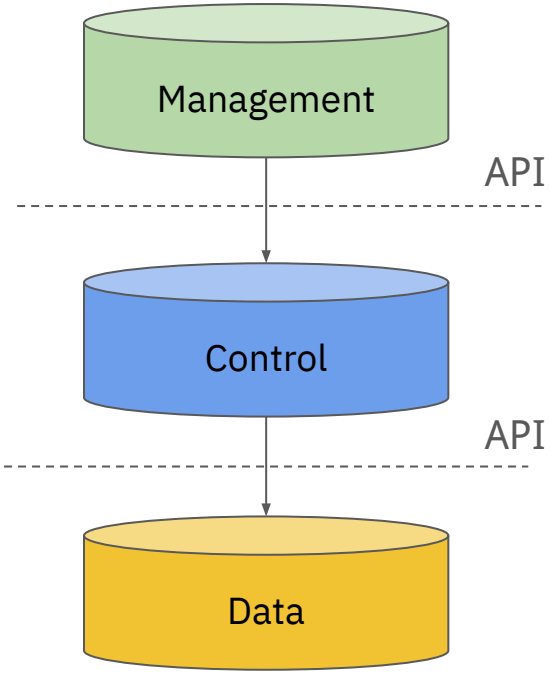
“Traditional” Network Development Process



Challenges of Vertical Integration

- Slow – must go through the standards process to add to the network.
- Inflexible – vendors aiming to implement solutions that work for multiple network operators.
  - What happens if my problem is different to other operators?
- Lack of ability to experiment.
  - How do I try new things in the network that I’m not sure will work out?

Disaggregating Routers.



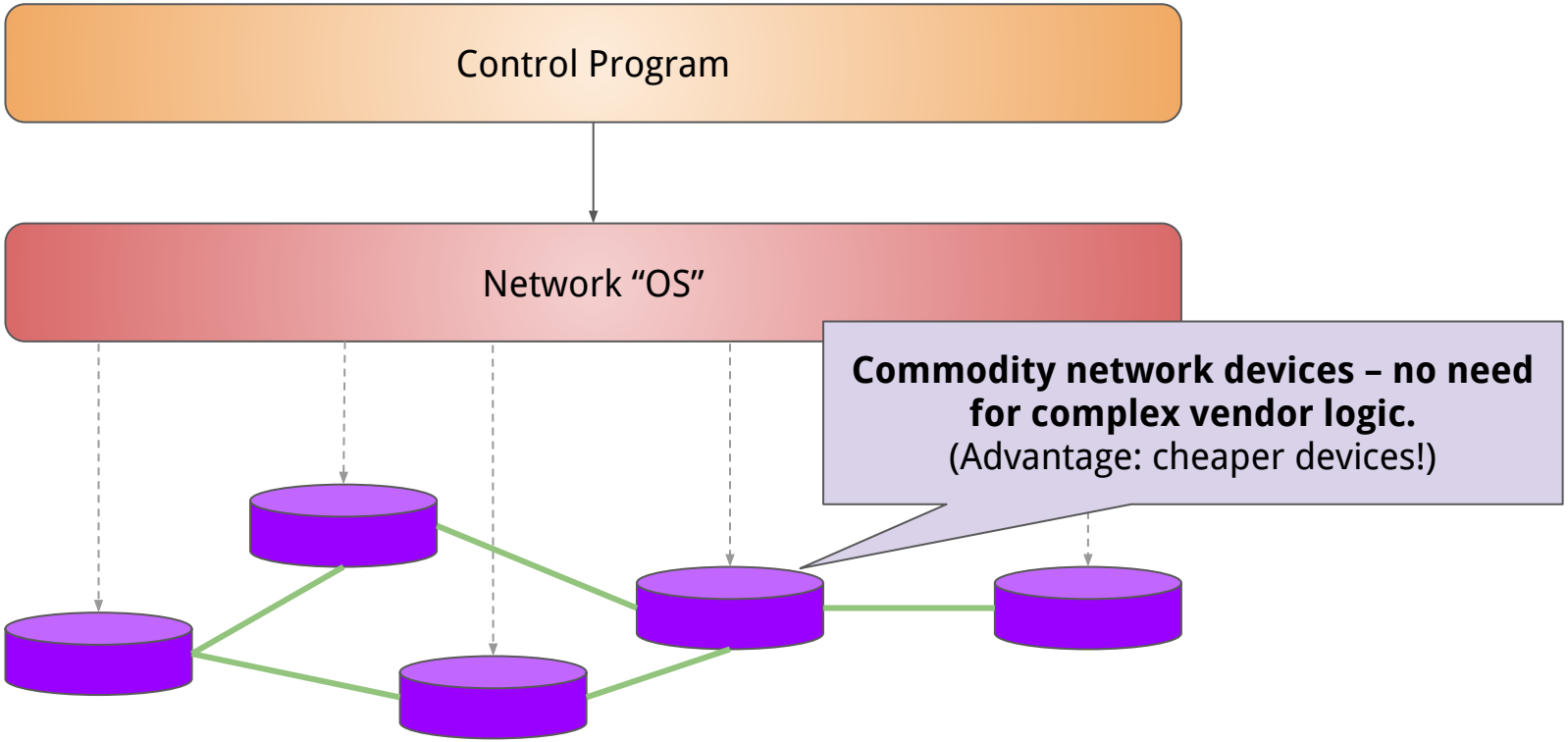
- Rather than procure everything from one place – split apart the network planes.
- Each plane has an API that communicates to the one below it.
  - e.g., the control plane computes forwarding tables and installs them in the data plane.
- Idea: can we allow more customisation in the control-plane by splitting the router apart?

Questions?

Separating Control and Data Planes

- Not a new idea!
- IETF standardisation process examining this concept since 2003!
  - Forwarding and Control Element Separation – forces.
  - Didn't get significant momentum.
- ~2004: New management paradigms being researched.
  - RCP, 4D [Princeton, CMU]
  - **SANE, Ethane [Stanford/Berkeley]** (Scott Shenker)
- 2008: More momentum!
  - NOX Network Operating System [Nicira]
  - **OpenFlow switch interface [Stanford/Nicira]**
- 2011: Open Networking Foundation (ONF)
  - Google, Yahoo, Verizon, Deutche Telekom, Microsoft, Facebook, NTT…
  - Cisco, Juniper, HP, Dell, Broadcom, IBM…
  - Significantly more momentum!

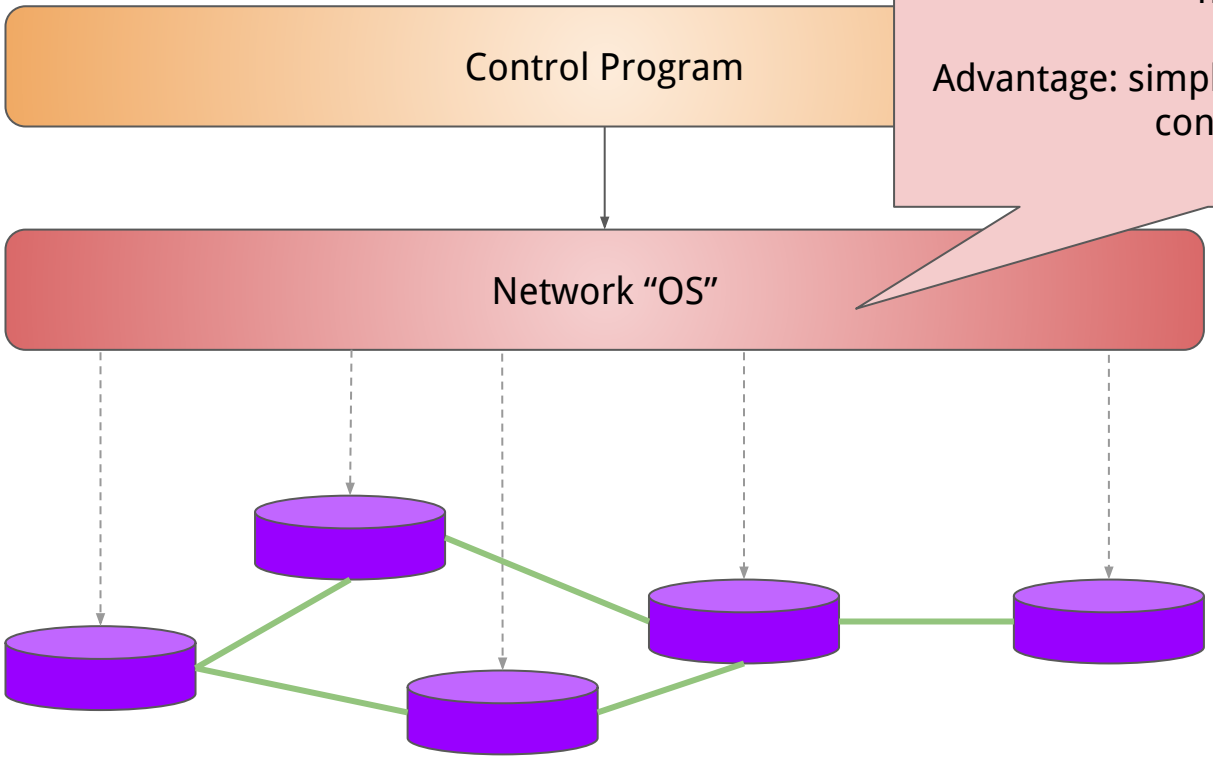
The “classic” SDN view



The “classic” SDN view

Logically centralised controller. Able to provide an abstraction of the network graph and programming of the network.

Advantage: simplification of the network control plane.

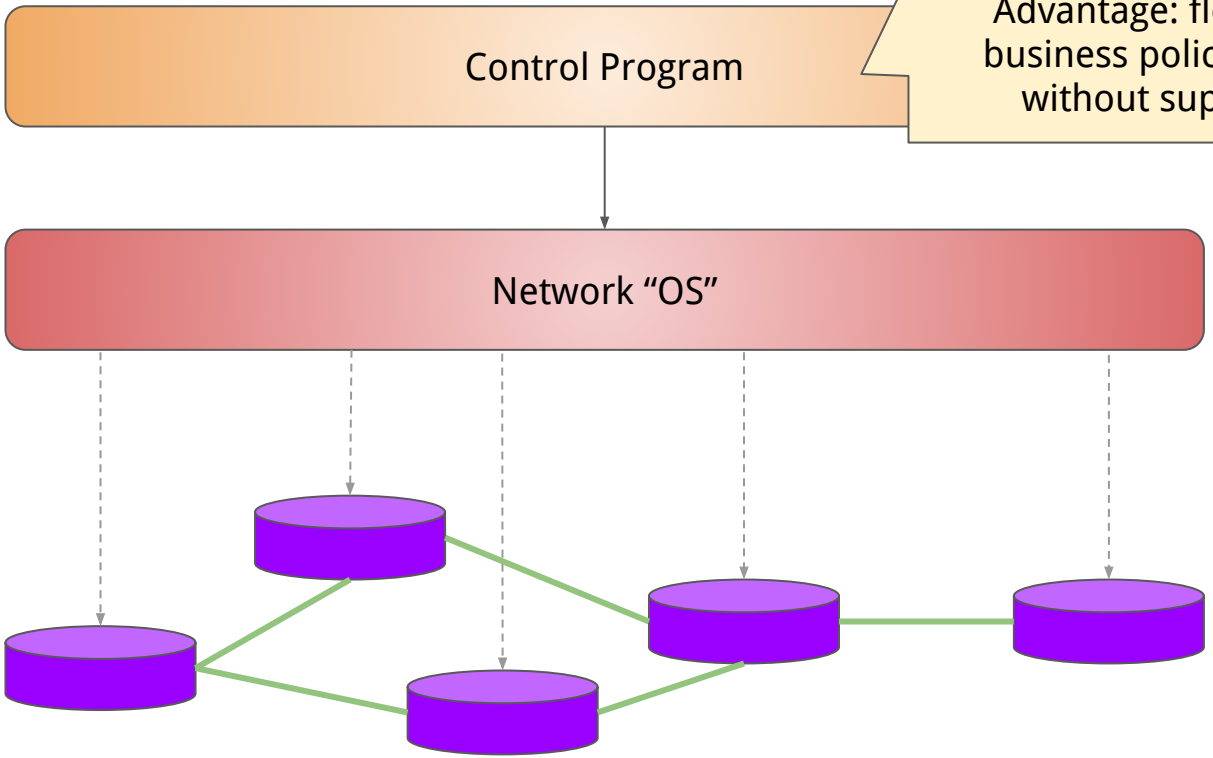


The “classic” SDN view

Operator-specific control programs.

Able to consider the abstract graph of the network and make routing decisions.

Advantage: flexible means to apply business policies to network routing without supplier dependencies.



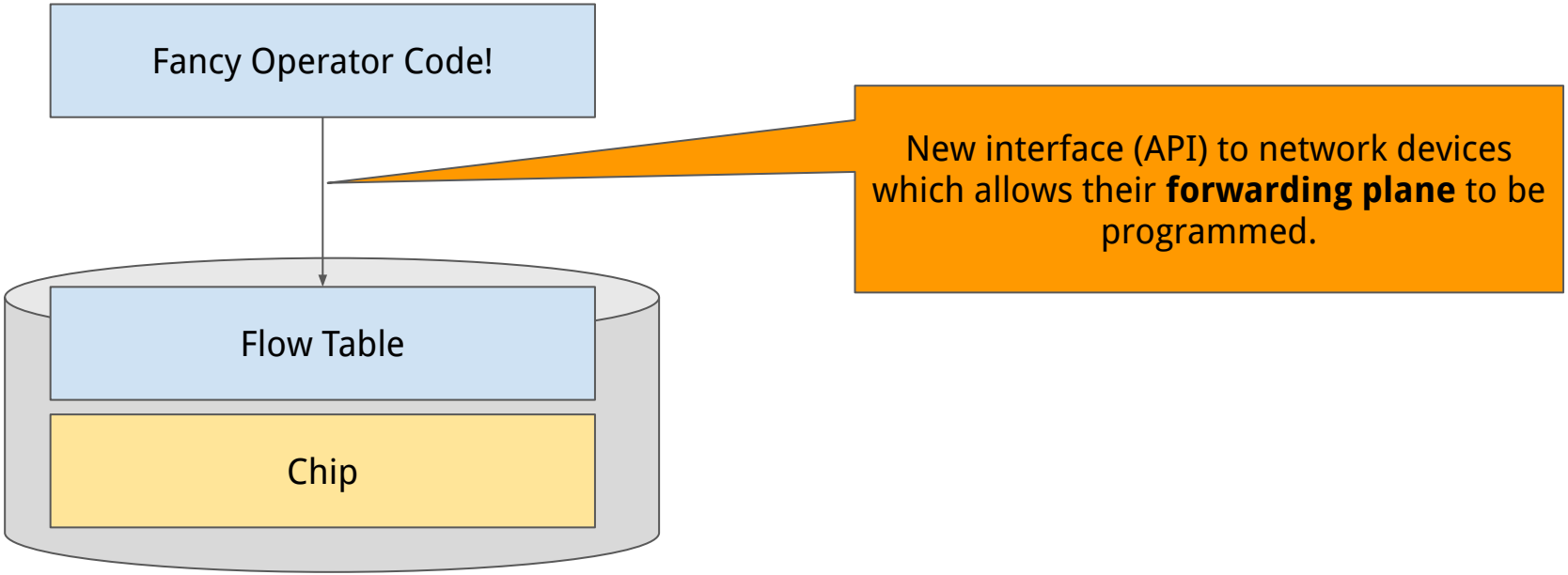
Questions?

APIs become critical!

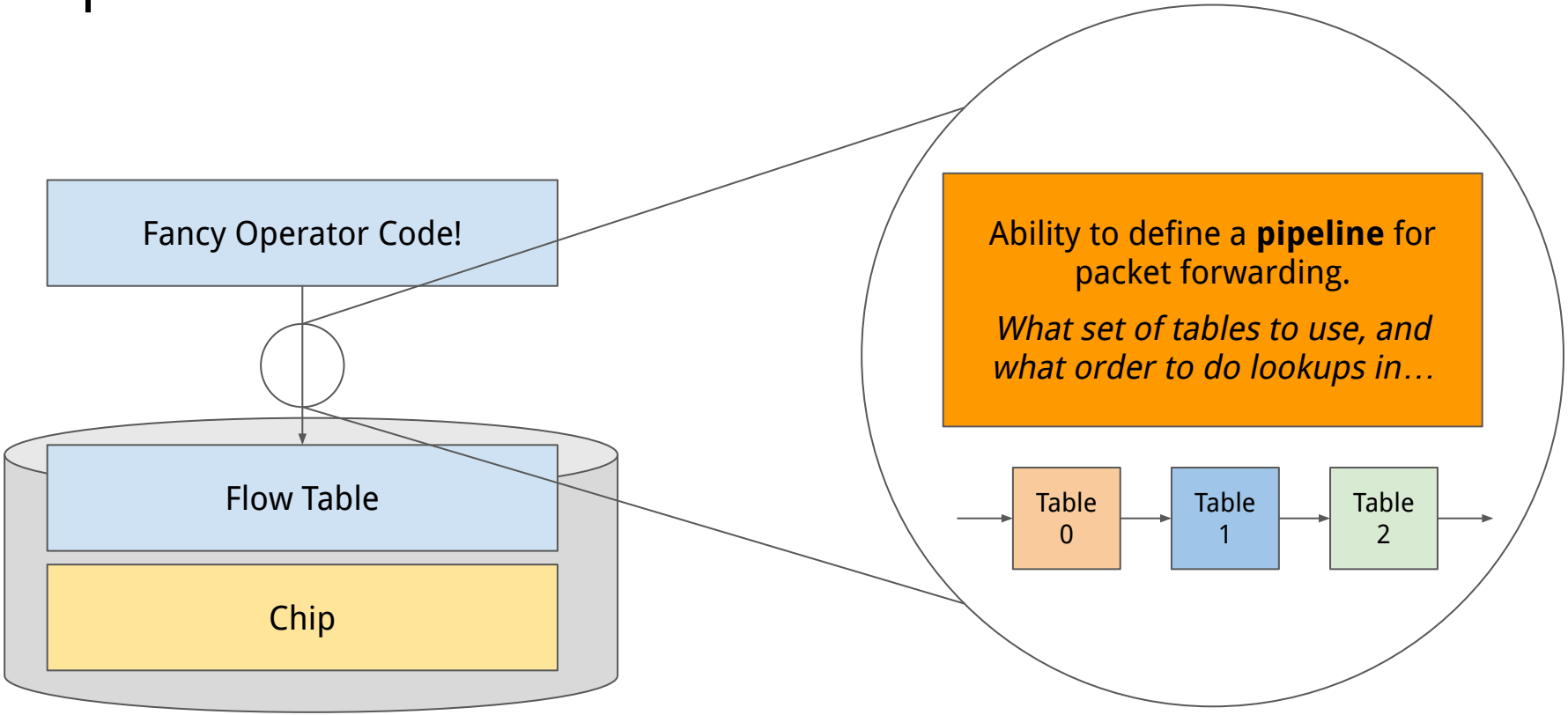
- If we separate the layers of the network – we need well-defined APIs between them.

- One of the most critical is the one that determines what forwarding entries should be installed.
  - Previously was entirely hidden within one vendor’s router implementation.
  - No standardisation.

OpenFlow


Fancy Operator Code!
New interface (API) to network devices which allows their **forwarding plane** to be programmed.
Flow Table
Chip

OpenFlow


Fancy Operator Code!
Flow Table
Chip
Ability to define a **pipeline** for packet forwarding.
What set of tables to use, and what order to do lookups in…
Table 0
Table 1
Table 2

OpenFlow

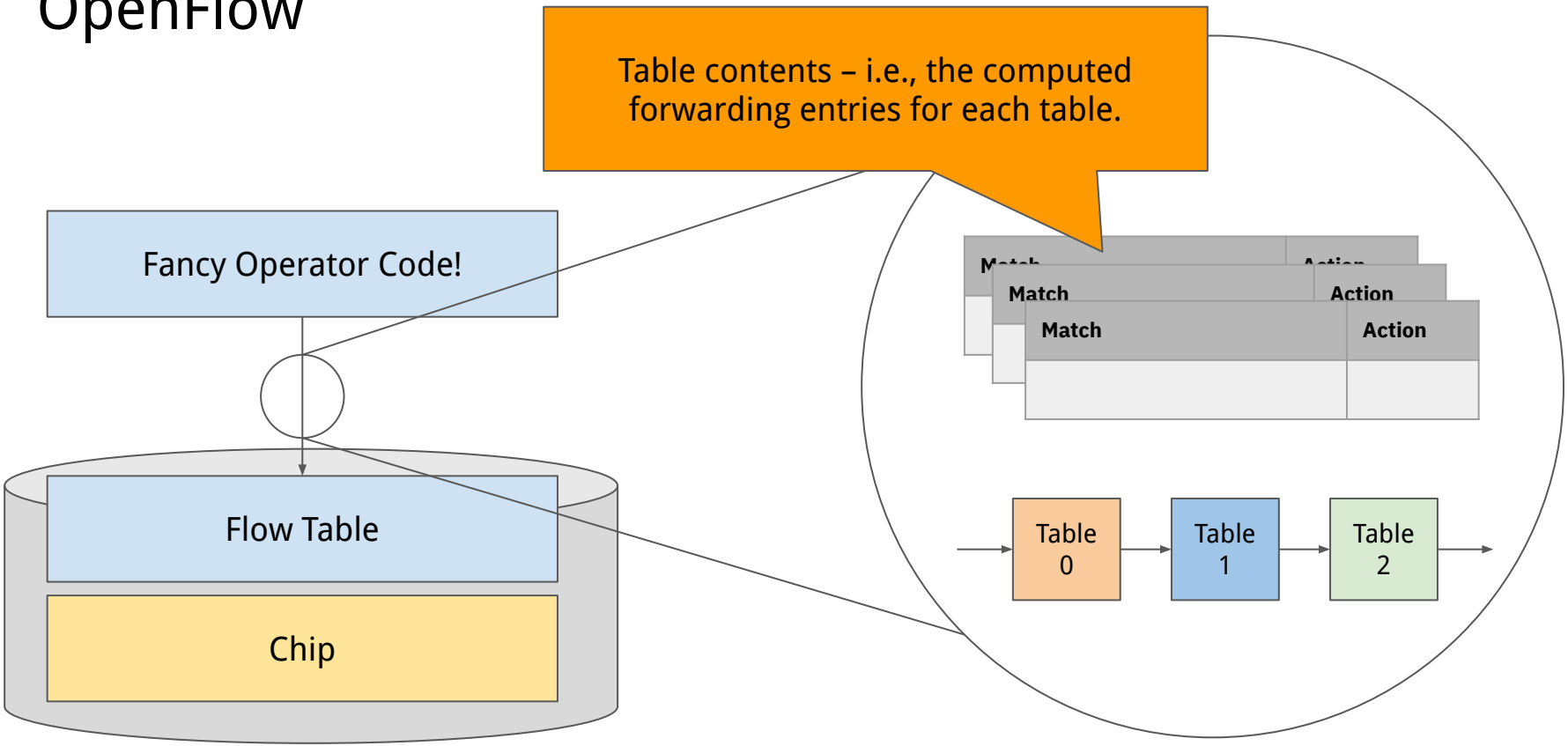


Table contents – i.e., the computed forwarding entries for each table.
Fancy Operator Code!
Flow Table
Chip
Match
Action
Match
Action
Match
Action
Table 0
Table 1
Table 2

OpenFlow

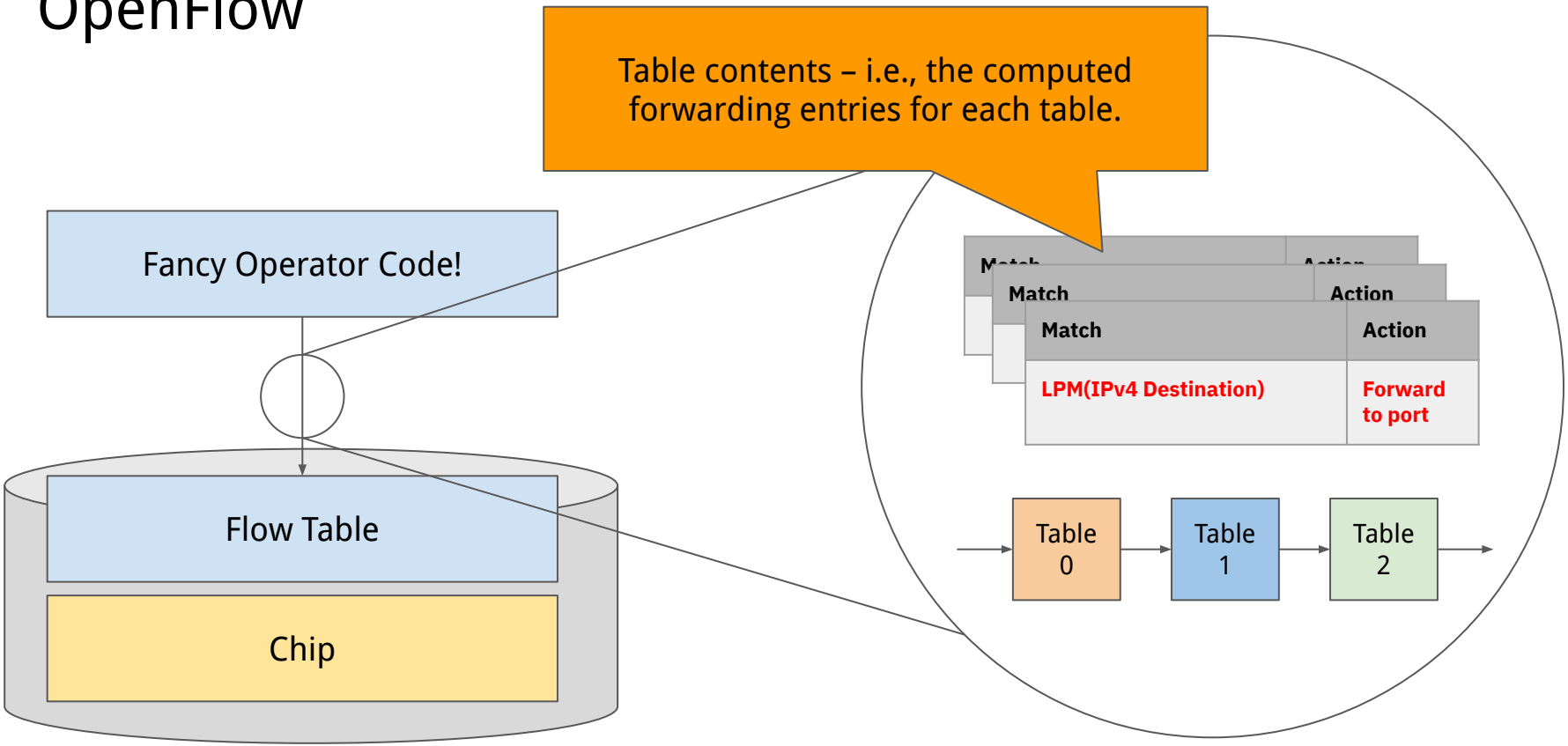


Table contents – i.e., the computed forwarding entries for each table.
Fancy Operator Code!
Flow Table
Chip
Match
Action
LPM(IPv4 Destination)
Forward to port
Table 0
Table 1
Table 2

OpenFlow

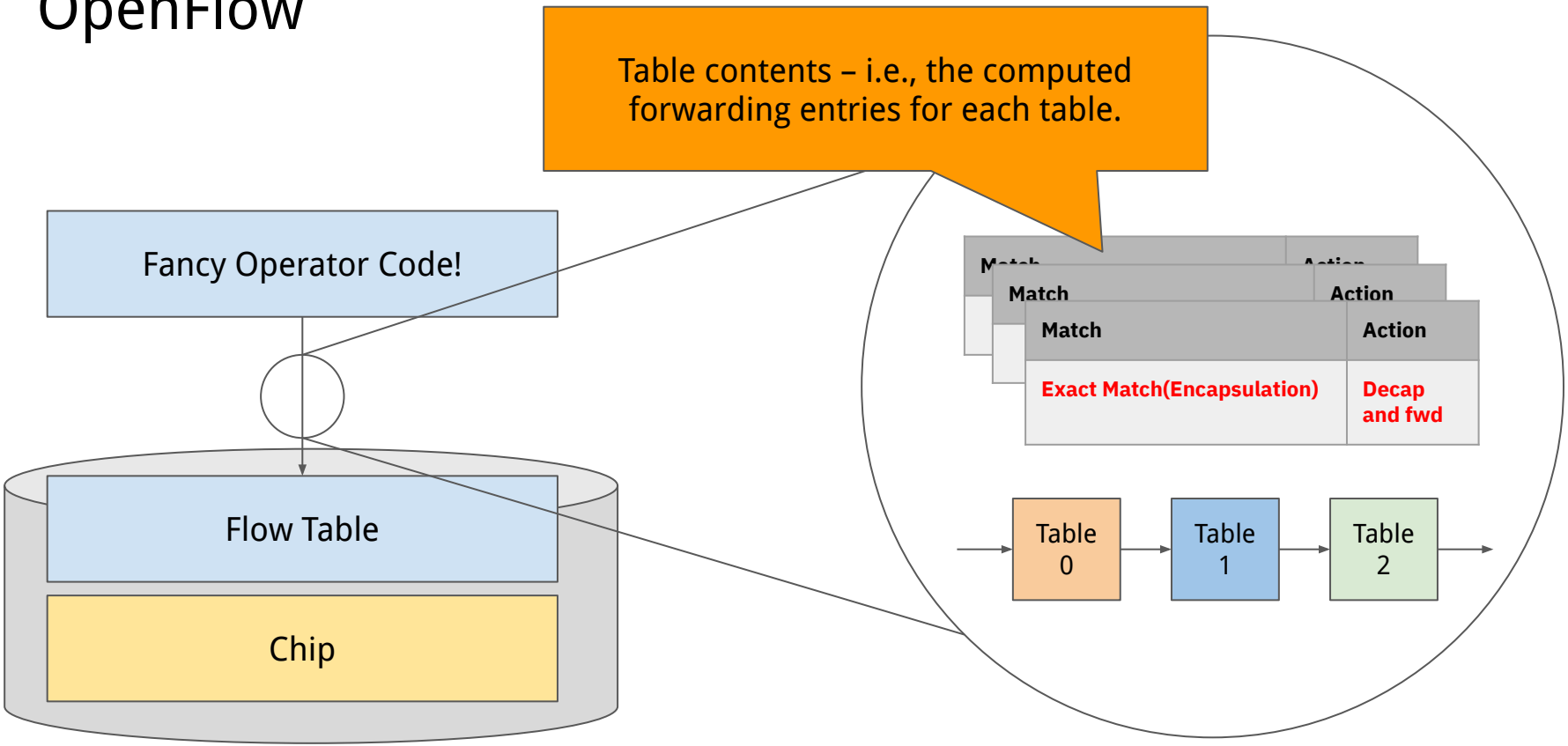


Table contents – i.e., the computed forwarding entries for each table.
Fancy Operator Code!
Flow Table
Chip
Match
Action
Exact Match(Encapsulation)
Decap and fwd
Table 0
Table 1
Table 2

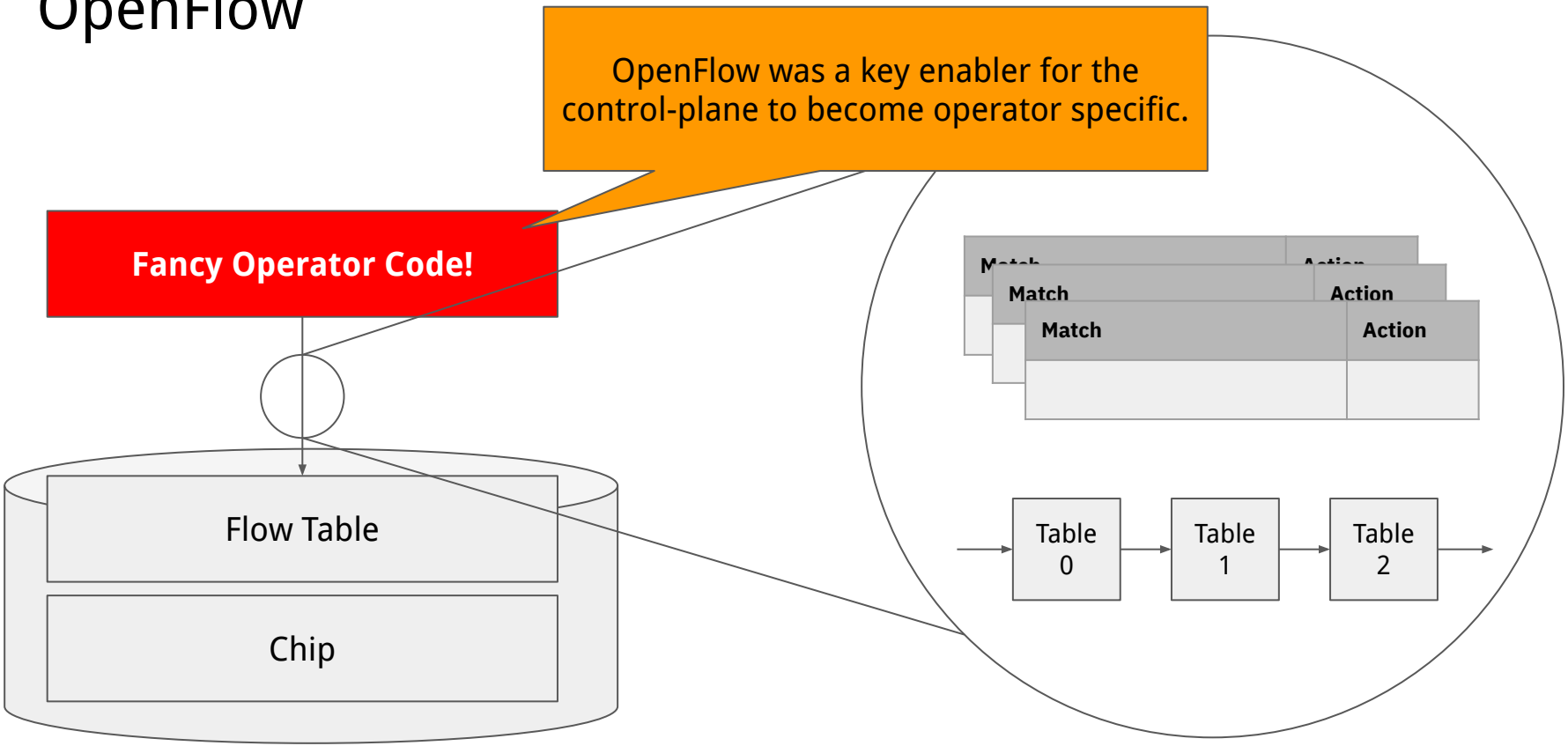
OpenFlow Forwarding Tables

- Still constrained by what forwarding chips can do.

- Tend not to be that different to our “classic” tables.
  - IPv4 & IPv6 destination-based (LPM).
  - Policy-based routing with 5-tuples.
  - Exact match (e.g., MPLS).

- The advantage is in the flexibility of **control-plane** that this allowed.

OpenFlow


OpenFlow was a key enabler for the control-plane to become operator specific.
Fancy Operator Code!
Flow Table
Chip
Match
Action
Match
Action
Match
Action
Table 0
Table 1
Table 2

OpenFlow and its evolution

- OpenFlow was the first adopted approach to programming network devices.
  - Still by a minority of network operators.
- Had both technical and commercial challenges.
  - Particularly, no clear coexistence with routers that have “traditional” protocols.
- Over the last 10 years, alternative approaches have built on the original OpenFlow idea.
  - P4Runtime – allows programmable dataplanes.
  - gRIBI - allows routes to be injected in a way that coexists with other routing protocols.

Questions?

SDN Control Planes

- An opportunity to simplify.
- The network control-plane is complex.
  - And was (and still is!) getting more and more complex based on different applications.
- Thinking about the network from a centralised viewpoint has advantages.
  - Can avoid the challenges of convergence.
  - Provides a way to make globally optimal decisions.
- So, where did it get us for traffic engineering?

SDN Adoption

ACM SIGCOMM 2013

B4: Experience with a Globally-Deployed Software Defined WAN

Sushant Jain, Alok Kumar, Subhasree Mandal, Joon Ong, Leon Poutievski, Arjun Singh,
Subbaiah Venkata, Jim Wanderer, Junlan Zhou, Min Zhu, Jonathan Zolla,
Urs Hölzle, Stephen Stuart and Amin Vahdat
Google, Inc.
b4-sigcomm@google.com

These characteristics led to a Software Defined Networking architecture using OpenFlow to control relatively simple switches built from merchant silicon. B4's centralized traffic engineering service drives links to near 100% utilization, while splitting application flows among multiple paths to balance capacity against application priority/demands. We describe experience with three years of B4 production deployment, lessons learned, and areas for future work.

SDN Adoption

ACM SIGCOMM 2013

Achieving High Utilization with Software-Driven WAN

Chi-Yao Hong (UIUC) Srikanth Kandula Ratul Mahajan Ming Zhang
Vijay Gill Mohan Nanduri Roger Wattenhofer (ETH)
Microsoft

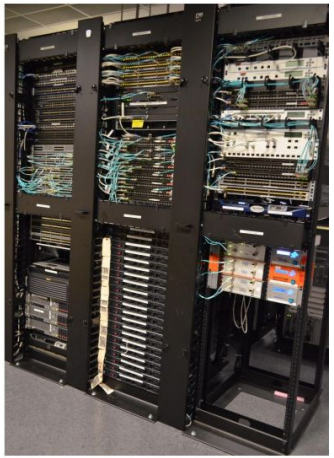
Prototype

16 OpenFlow switches
- Mix of Blades and Aristas

BigSwitch OpenFlow controller

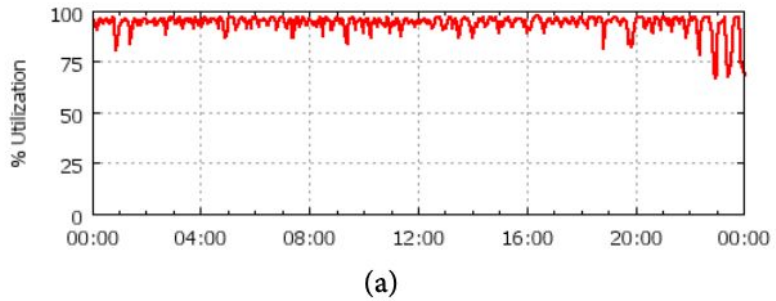
32 servers as traffic sources
- 25 virtual hosts per server

8 routers (L3)
- Mix of Cisco and Juniper

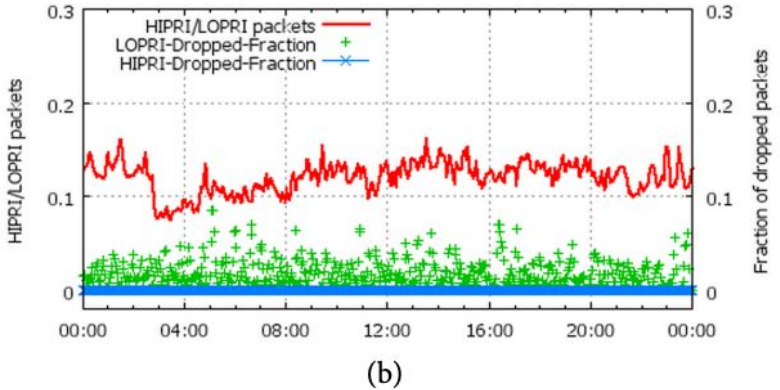


https://www.microsoft.com/en-us/research/wp-content/uploads/2017/01/swan-msrc-jul2013.pdf

Utilisation Benefits



(a)



(b)

Google B4

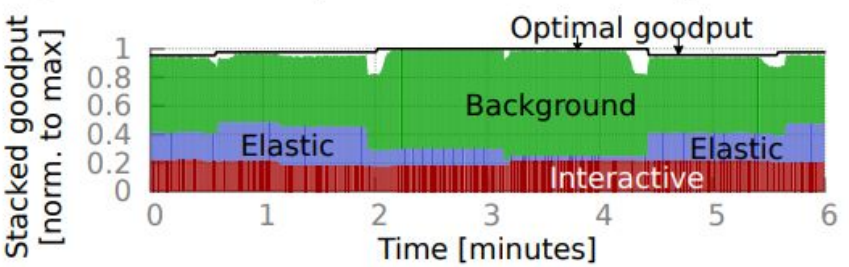


Figure 10: SWAN achieves near-optimal throughput.

Microsoft SWAN

- Significant increase in utilisation of expensive network assets (e.g., subsea fibre!).
- Ability to consider traffic that can be dropped to avoid additional build.

Why might we not want to centralise?

- As with all designs – nothing comes for free.
- Reliability.
  - Traditional IP networks – one router fails, routing protocols converge around the failure.
  - SDN networks - central controller fails, network “fails static” and does not converge.
- Scalability.
  - Central controller must deal with decisions on everyone’s behalf – not just at one router.
- Complexity.
  - Infrastructure for off-device control planes etc.
- Sylvia, Alex and I are researching this right now.
  - A Decentralized SDN Architecture for the WAN [SIGCOMM’24].

Questions?

SDN in Datacenters

Recall: Datacenter networks

- Single owner but multiple applications hosted in them.
- Multiple tenants, with different requirements of the network.

Private IP Addressing

- Wait? How can two hosts have the same IP address?

Private IP Addressing

- Wait? How can two hosts have the same IP address?
- Remember, we only had 2^{32} IPv4 addresses.
- Some hosts never want to be contacted from the Internet - so don’t need a unique address.
- So, we can save addresses by having private addresses that can be used in multiple networks.

Private IP Addressing

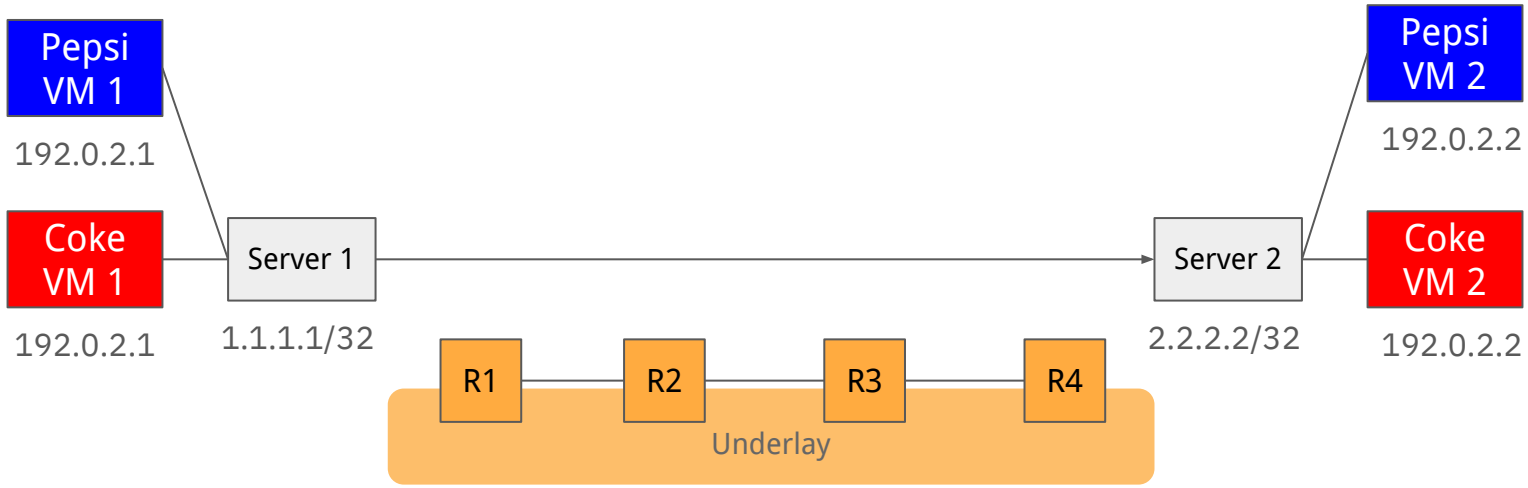
- Specific ranges defined for “private” use.
  - In RFC1918, so generally referred to as RFC1918 addresses.
- Most common:
  - 192.168.0.0/16
  - 10.0.0.0/8
- You’ll see these in your home network!

Private IP Addressing

- Specific ranges defined for “private” use.
  - In RFC1918, so generally referred to as RFC1918 addresses.

- Most common:
  - 192.168.0.0/16
  - 10.0.0.0/8

Recall: Multi-tenancy in a Datacenter



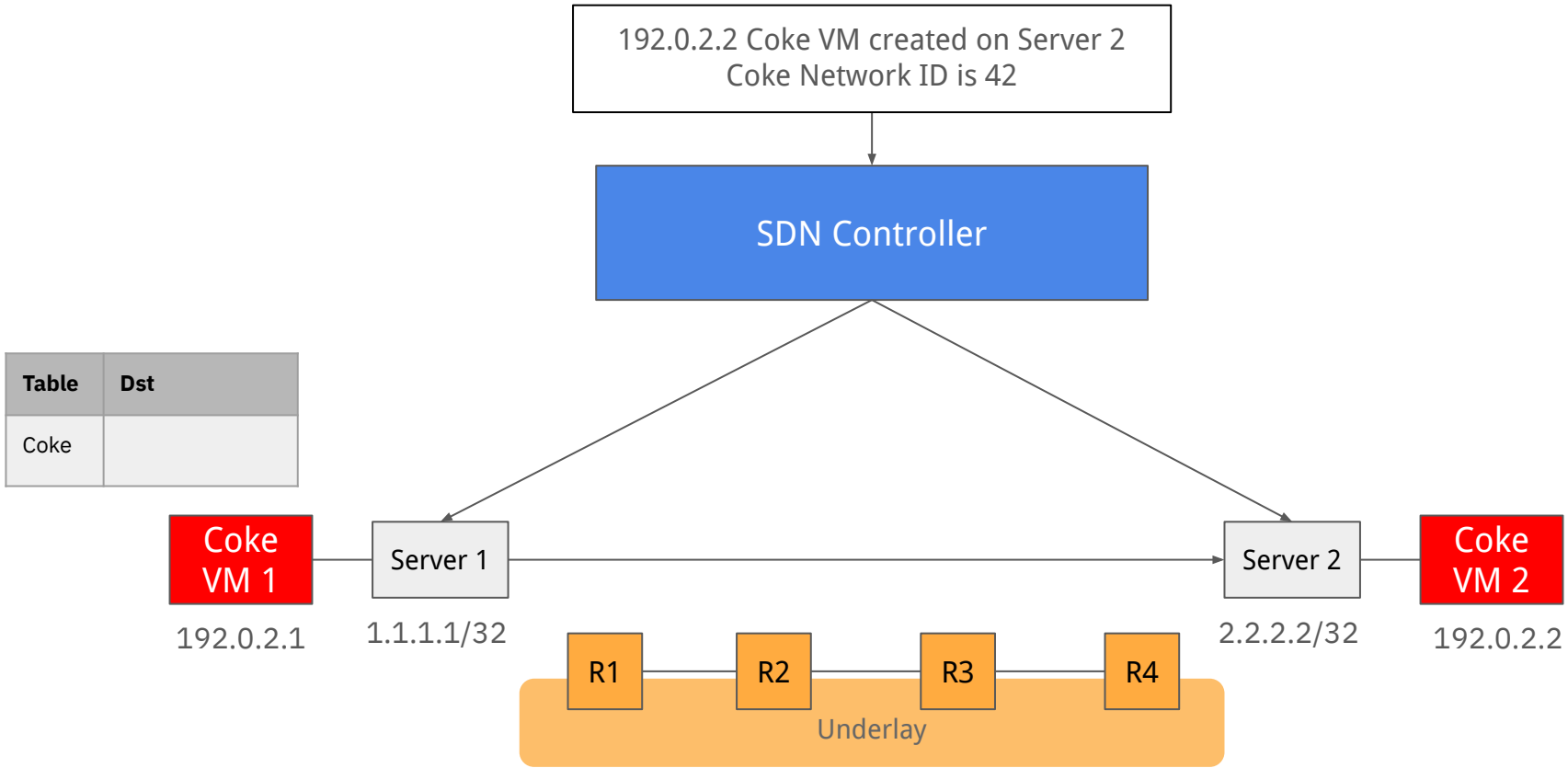
Our datacenter networks need to support multiple tenant networks - who don't coordinate with each other.

Per-tenant Overlay Networks

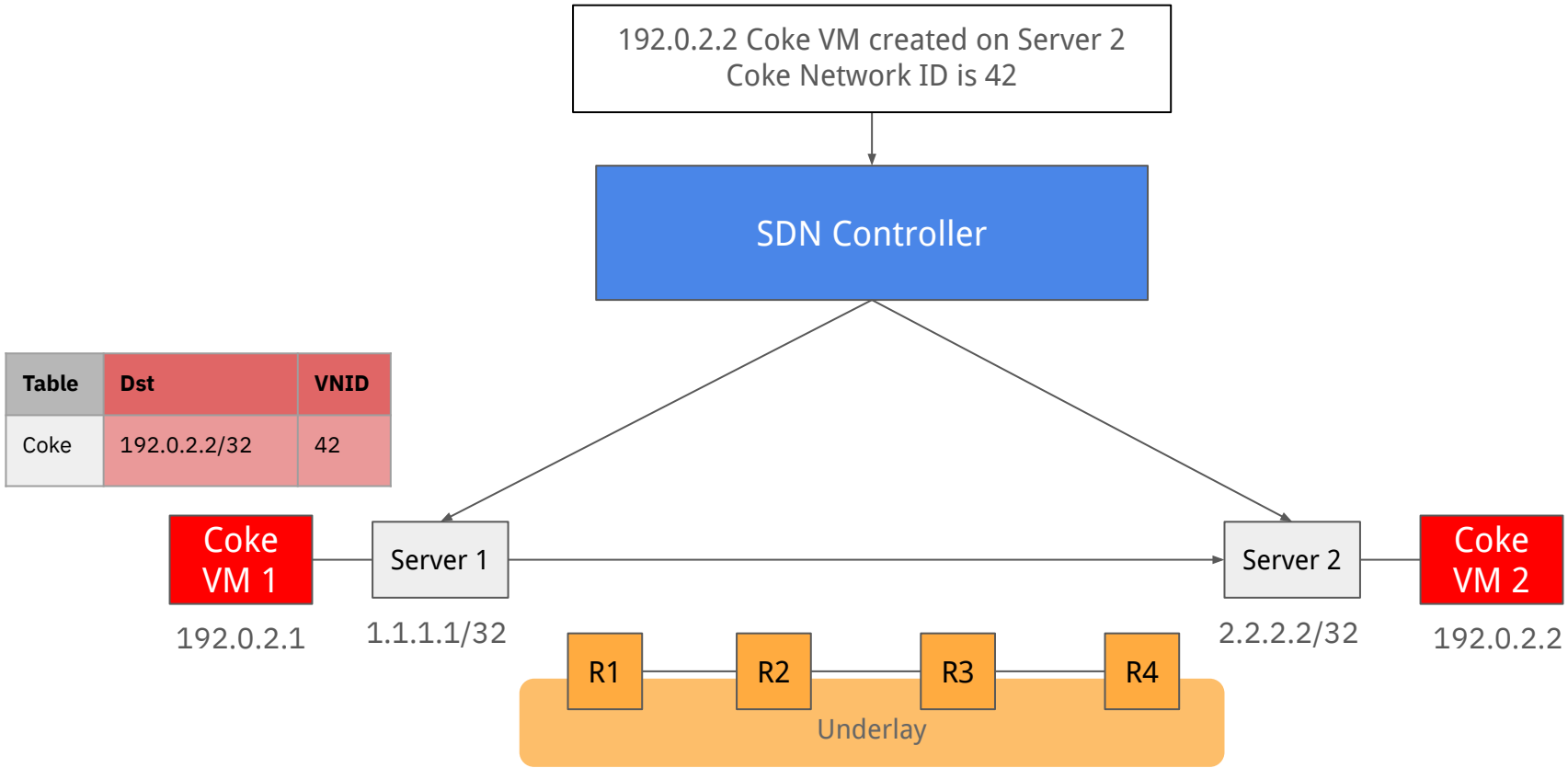
- Multiple tenants with their own *overlay networks.*

- How do we get different parts of the network to learn about the encapsulation that is needed, and where remote hosts are?

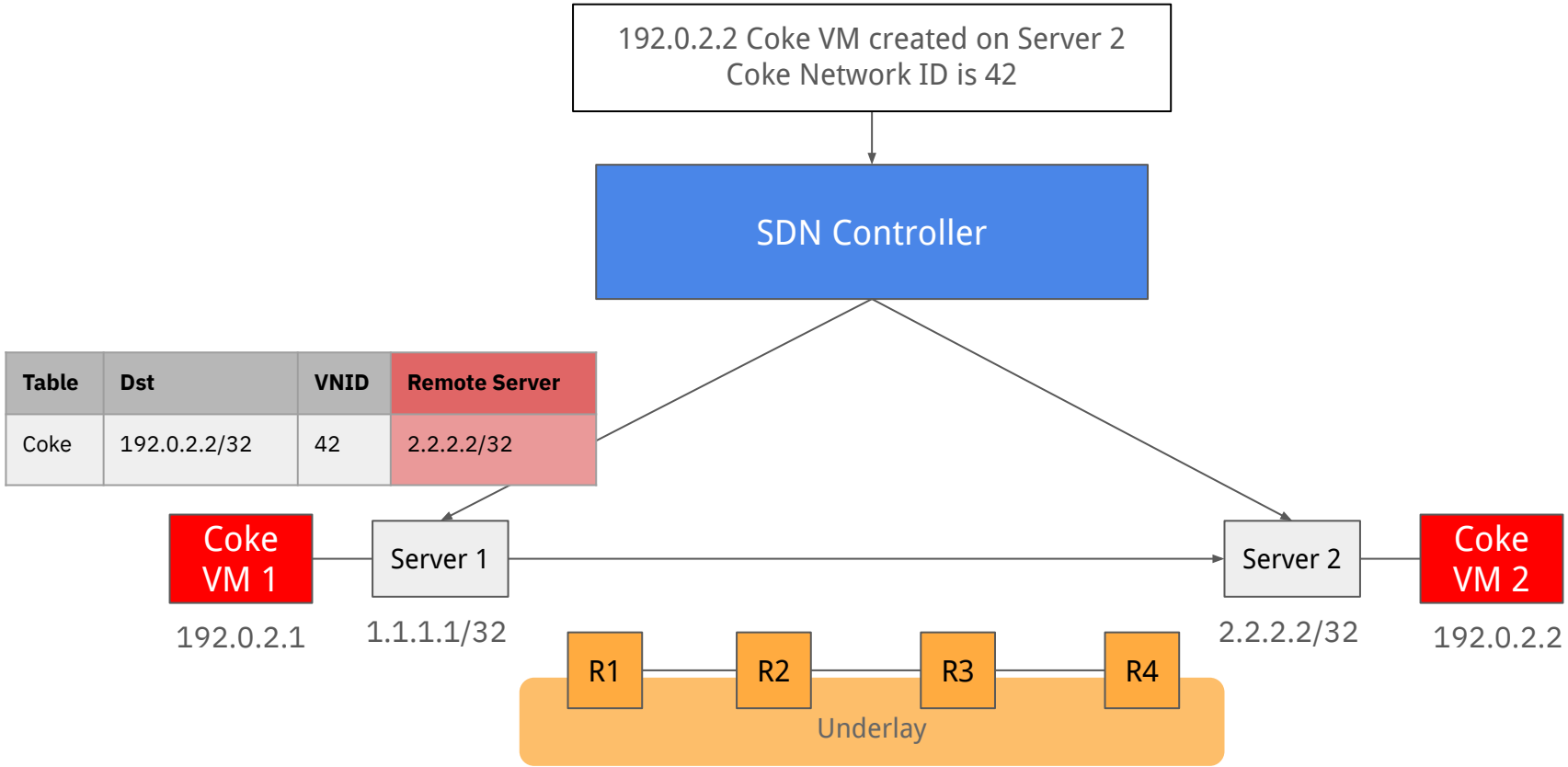
SDN in Datacenters



SDN in Datacenters



SDN in Datacenters



SDN for the Overlay

- Handles distributing custom information - that would otherwise be in the control-plane of the routers.
  - e.g., virtual network IDs.
- Allows the underlay of the network to remain as simple as possible.
  - No need for the underlay to understand anything about endpoints in virtual networks.
- Allows the control plane to be extended to servers without needing routing protocols.
  - Simpler mechanism to program endpoints.

Recap

- SDN evolved from inflexibility of the control-plane to meet different operational requirements.
- It splits the control- and data-plane – to make the control-plane more flexible.
- There are applications across both WAN and datacenter networks.

Thanks!

- This is my last lecture of the semester!
- Thanks very much for your attention and questions.
- I will be at (most) office hours until December 5th.
- You can contact me at rjs@rob.sh.

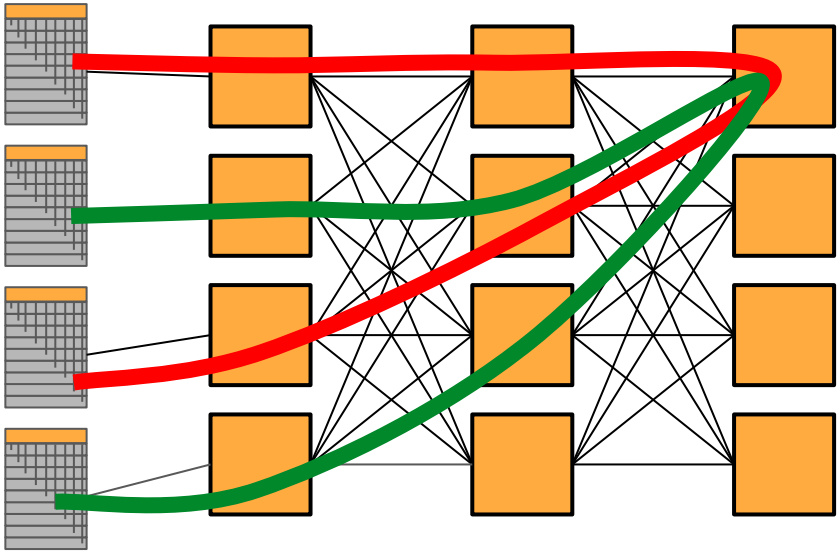
Bonus Material

SDN in the Datacenter Underlay

SDN for the Datacenter underlay

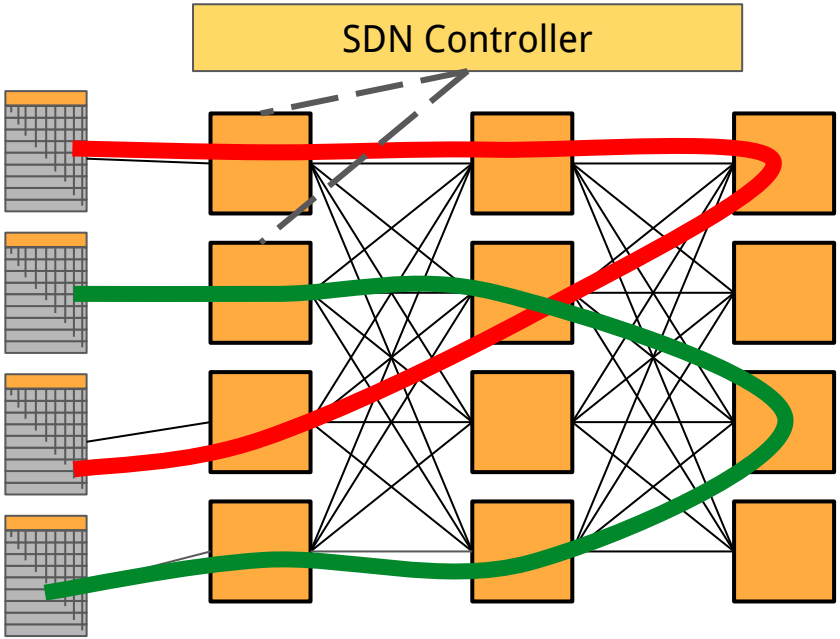
- The underlay network is just a physical network - with the same tradeoffs as designing WANs.
- Making good use of capacity, adapting to different elephant and mice flows.
- Hyperscale datacenters may use SDN in the overlay and underlay.
  - But these are decoupled systems.

Limits of Hashing in DC Topologies



Load balancing is per-flow using the 5-tuple - elephants can still choose the same path!

Limits of Hashing in DC Topologies



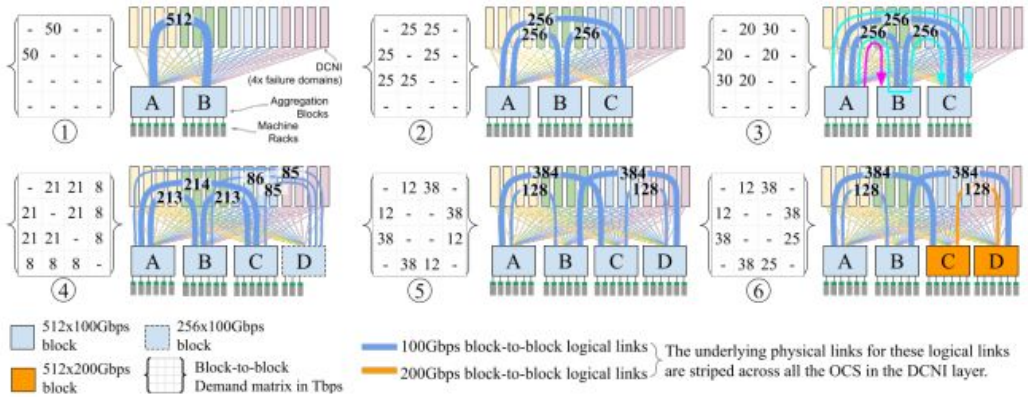
A global controller aware of different flow demands can place traffic more efficiently onto a Clos topology – improving performance.

Improving Efficiency of Datacenters using SDN

Jupiter Evolving: Transforming Google's Datacenter Network via Optical Circuit Switches and Software-Defined Networking

Leon Poutievski, Omid Mashayekhi, Joon Ong, Arjun Singh, Mukarram Tariq, Rui Wang, Jianan Zhang, Virginia Beauregard, Patrick Conner, Steve Gribble, Rishi Kapoor, Stephen Kratzer, Nanfang Li, Hong Liu, Karthik Nagaraj, Jason Ornstein, Samir Sawhney, Ryohei Urata, Lorenzo Vicisano, Kevin Yasumura, Shidong Zhang, Junlan Zhou, Amin Vahdat
Google
sigcomm-jupiter-evolving@google.com

- Elimination of stages in a Clos topology by adding dynamic links.
- Enabled by SDN.



Jupiter Evolving, ACM SIGCOMM '22

SDN beyond the control plane

Did SDN remove the need for the management plane?

- SDN is primarily focused on the control plane.
- But we still need some way to be able to tell routers in the network what to do, and see what they are doing.
- One of SDN's lessons is well-defined, programmatic APIs.
- This lesson also applies to the management plane of the network.

“SDN” in the Management Plane

- Manifested itself as more software being used in networking.

- Key ideas:
  - Data modelling – YANG (IETF data modelling language)
  - OpenConfig – use of standardised APIs to interact with the management plane (like OpenFlow did for the control plane).

- Not part of the original SDN vision – but something that has evolved during its implementation.

- Well-adopted across the industry by multiple operators.
  - Rob has spent 10 years on this problem space too!

SDN in the Data Plane?

- Again, SDN has focused on the control-plane.
- Relatively standard set of lookup tables and behaviours available.
  - Other than in software-based dataplanes (low scale!)
- What happens if the dataplane is the inflexible part?
  - Programming Protocol-Independent Packet Processors (P4)
  - Open source, domain-specific language for telling a device how to process packets.
  - Programmability at the data plane.
- Some challenges!
  - Fixed functionality forwarding chips (needed for speed) are not arbitrarily flexible.
  - Programmable chips are lower performance, and higher cost.