Lecture 21 (Datacenters 3)

Software-Defined Networking

CS 168, Fall 2026 @ UC Berkeley

Slides credit: Sylvia Ratnasamy, Rob Shakir, Peyrin Kao

Software-Defined Networking: A Brief History

Lecture 20, CS 168, Spring 2026

Software-Defined Networking

- **A Brief History**
- Implementation (OpenFlow API)
- Benefits (Traffic Engineering)
- SDN in Datacenters

What is Software-Defined Networking?

Software-defined networking (SDN) is a new way to think about network management.

- So far: Routers each run a distributed protocol to compute paths.
  - What do we do if these protocols don't meet our needs?

- Today: A centralized piece of code computes paths and gives them to routers.
- SDN applies to other aspects of network management as well, but we'll focus on routing.

Innovating Routers

Suppose we want a specialized routing protocol for our own network.

Why is implementing it hard?

- Routers are **standardized**.
  - We have to submit our change to a standards body.
  - We might have to wait years for them to approve it.
- Routers are **inflexible**.
  - Major vendors want to implement solutions that work well for everybody.
  - If nobody else wants our protocol, vendors won't build it.
- Routers are **vertically integrated**.
  - The data, control, and management planes are all wired on the chips.
  - No easy way to swap out specific parts of the router.

Similar problems if we want to experiment.

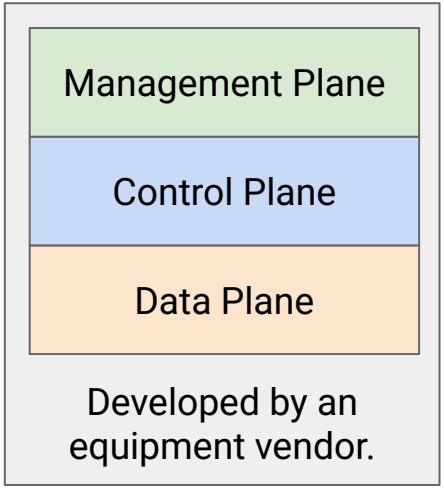
- Vendors don't want to build something that might not work.

Innovating Routers

Routers are standardized, inflexible, and vertically integrated.

The data plane is highly standardized.

- Recall: The data plane forwards packets.
- Routers from different vendors need to send bits on the wire in the same consistent format.
- Not much to innovate here (mostly faster speeds).

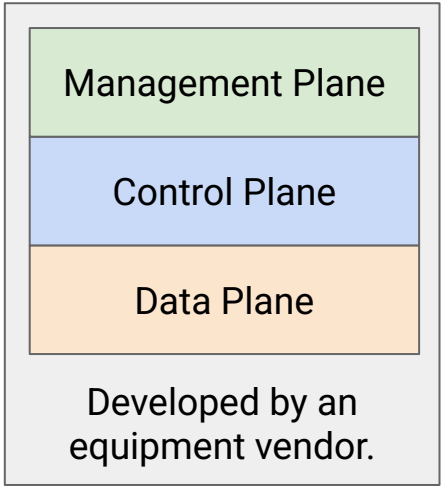


Innovating Routers

Routers are standardized, inflexible, and vertically integrated.

The control plane is mostly standardized.

- Recall: The control plane runs routing protocols.
- Some extensions exist, but routers from different vendors ultimately need to follow the same routing protocol.
- We want to innovate by creating our own routing protocols.



Researchers and network operators began to think about how to better control the network.

This led to some radical thinking about how to redesign routers!

A Clean Slate 4D Approach to Network Control and Management Link

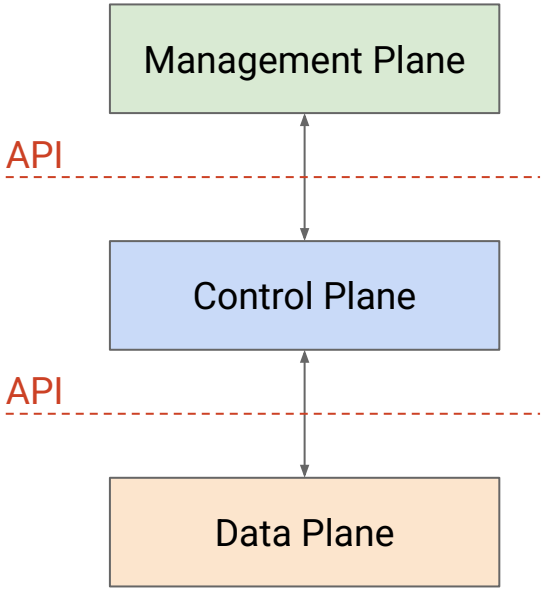
Albert Greenberg et. al. *2005*

Today's data networks are surprisingly fragile and difficult to manage. We argue that the root of these problems lies in the complexity of the control and management planes.

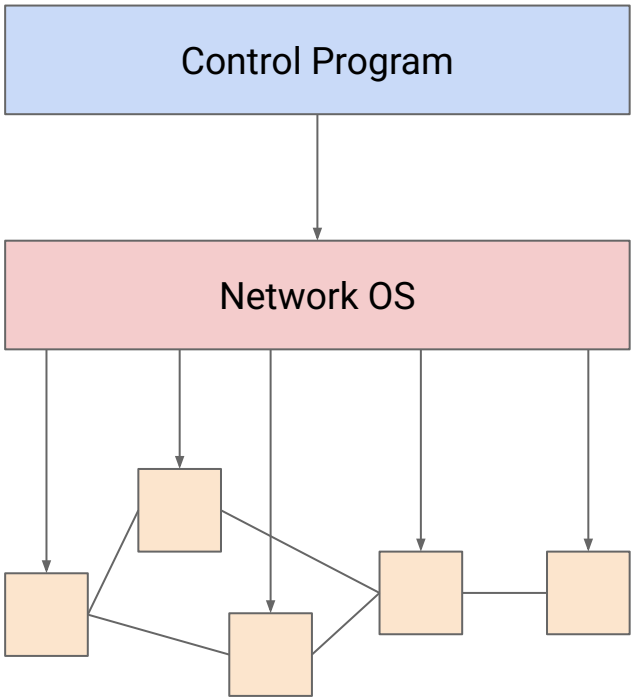
Radical Idea: Disaggregating Routers

Radical idea: Split the network planes apart.

- Each plane has an API that communicates to the plane below it.
- Allows for more customization and innovation at the control plane.



Radical idea: A totally new system architecture that splits up the three planes.



Operator receives a network abstraction (e.g. graph) from the OS, and computes routing decisions.

Allows the operator to use their own routing protocol.

Provides a network abstraction (e.g. graph) to the operator.

Takes the operator's decisions and uses them to program routers.

Commodity routers that forward based on operator decisions.

Can be cheap – they don't need to compute routing protocols.

A Brief History of SDN

Separating the control and data planes isn't a new idea.

- 2003: IETF created a working group to study this concept.
  - Forwarding and Control Element Separation (ForCES).
  - Didn't get significant momentum.
- 2004: New management paradigms being researched.
  - RCP, 4D at Princeton, CMU. SANE, Ethane at Stanford, UC Berkeley.
- 2008: More momentum. OpenFlow interface at Stanford, Nicira.
- 2011: Industry acceptance. Open Networking Foundation (ONF) created with tech companies (Google, Microsoft) and vendors (Cisco, Juniper, HP, Dell).

Nicira OpenFlow: Networking's Next Big Thing? Link

Charles Babcock *February 4, 2012*

$40-million startup emerges with OpenFlow platform for virtualizing the network as effectively as servers.

Implementation: OpenFlow API

Lecture 20, CS 168, Spring 2026

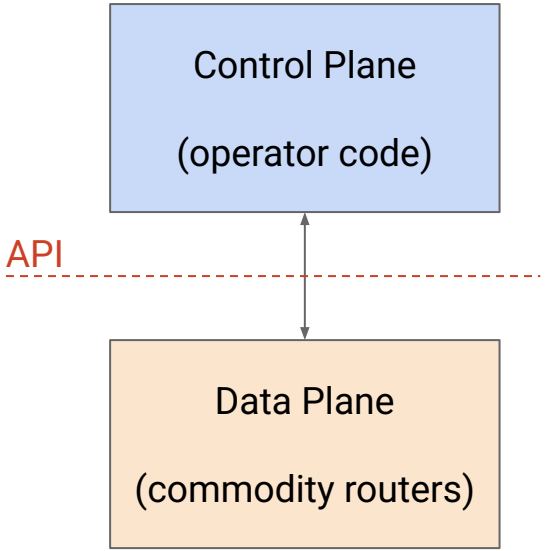
Software-Defined Networking

- A Brief History
- **Implementation (OpenFlow API)**
- Benefits (Traffic Engineering)
- SDN in Datacenters

OpenFlow API

To separate the control and data plane, we need an API.

- The operator runs its own fancy code and outputs routing decisions.
- The operator installs these routes onto the forwarding chips.
- We need a specific format for these decisions that both operators and chips understand.



OpenFlow API Format – Flow Tables

The basic building block is a **flow table**, a more generalized forwarding table.

- Maps matches to actions.
- Matches can include more than just the destination IP.
  - Example: 5-tuple to match a specific flow.
- Actions can be more than just sending to a specific next-hop.
  - Example: Encapsulation and decapsulation.

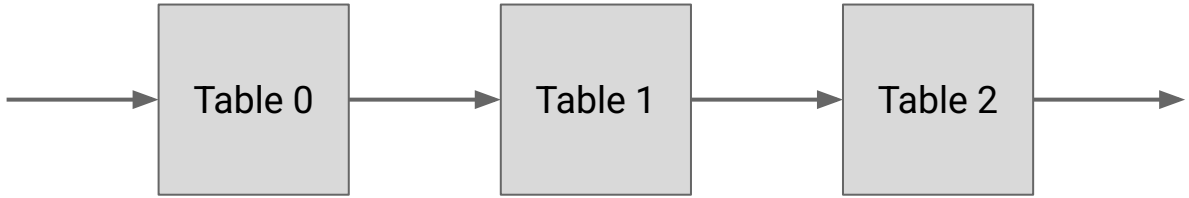
Example Flow Table	
Match:	Action:
Destination IP is in 192.168.0.0/16	Forward to next-hop R2
5-tuple is exactly (1.1.1.1, 2.2.2.2, TCP, 255, 53)	Add encapsulation and forward to R3
Source IP is exactly 10.2.15.3	Decapsulate and forward to R3

OpenFlow API Format – Flow Tables

The operator writes routing decisions as a sequence of numbered flow tables.

- Each table maps matches to actions, just like we saw.
- Packet is matched against each table in order, and actions are added.
  - Could add special actions like "skip to Table 2," or "clear all actions so far."
  - When we reach the last table, all actions are applied (e.g. packet is sent out).

You don't need to know the exact syntax, just the general idea.

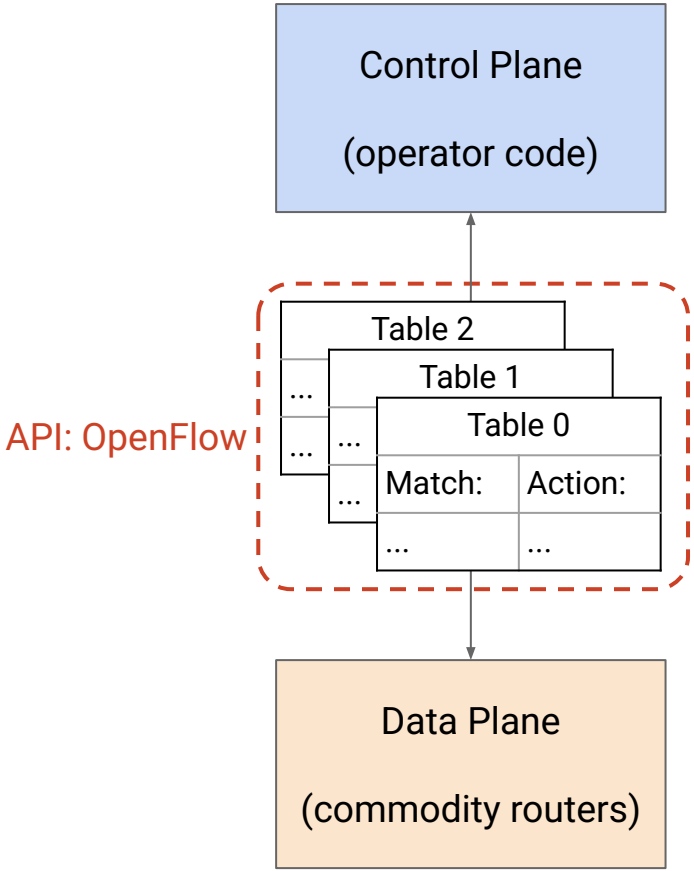


OpenFlow API Format – Flow Tables

Flow tables are still constrained by what forwarding chips can do in hardware.

- In practice, matches tend to be similar to "classic" forwarding tables.
  - Longest-prefix matching on destination IP addresses.
  - Policy-based routing on 5-tuples.
  - Exact match on virtual network ID.

Remember: The advantage is letting the operator decide the policies.



Benefits (Traffic Engineering)

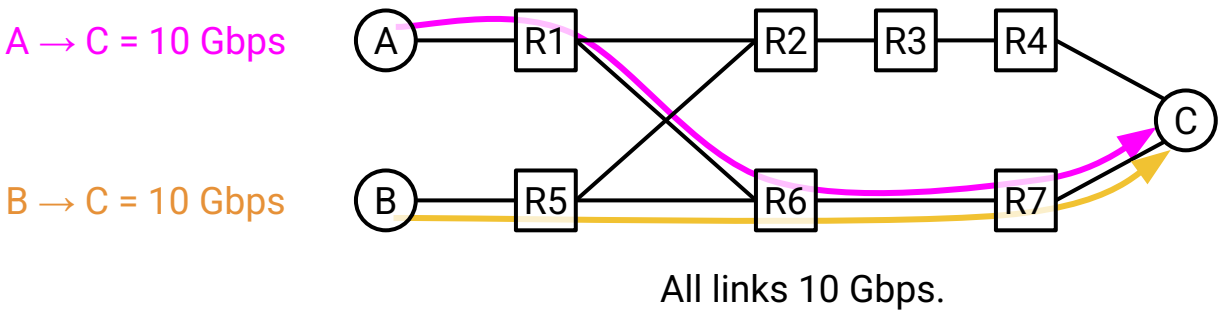
Lecture 20, CS 168, Spring 2026

Software-Defined Networking

- A Brief History
- Implementation (OpenFlow API)
- **Benefits (Traffic Engineering)**
- SDN in Datacenters

A flexible control plane lets the operator compute more intelligent routes.

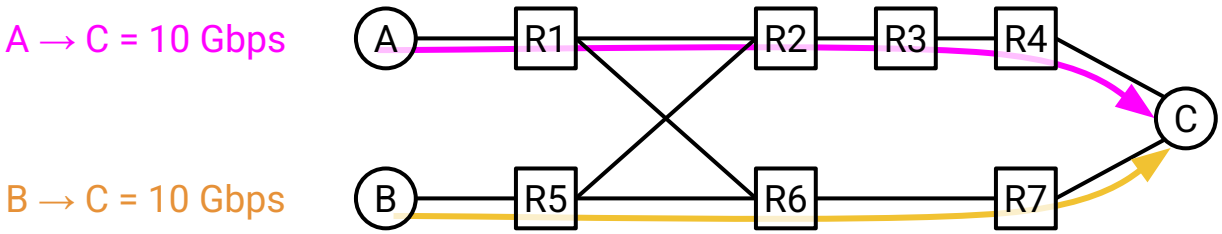
- With standard shortest-path routing, both flows use the bottom path.
- The bottom path is congested: 20 Gbps over a 10 Gbps link.



All links 10 Gbps.

A flexible control plane lets the operator compute more intelligent routes.

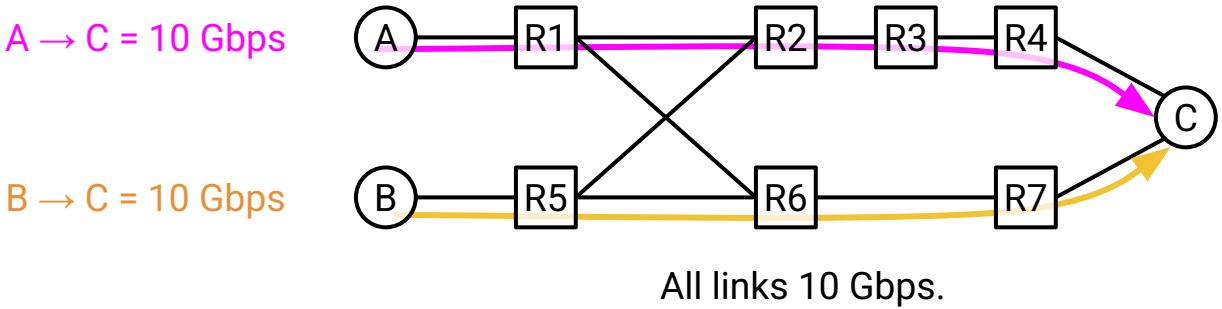
- A centralized operator could tell the flows to use different paths.
- Notice: A → C is not using the shortest path.
- Now, none of the links are congested anymore.



All links 10 Gbps.

Constrained Shortest Path First (cSPF) finds the shortest path that has sufficient capacity.

- Contrast with standard shortest-path, which doesn't consider capacity.
- Could also enforce other constraints (e.g. latency, not capacity).
- **Requires a central operator to coordinate all the routers.**

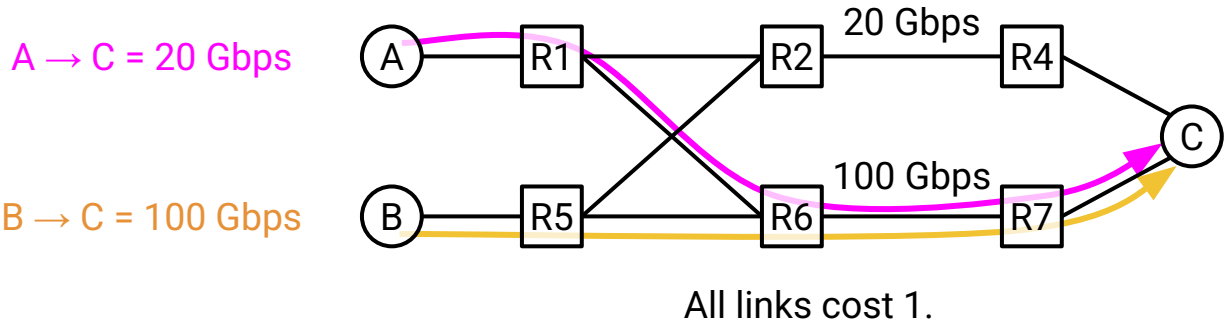


Traffic Engineering – Globally Optimal Decisions

In standard least-cost routing, each router might make a *locally optimal* decision.

But, since the nodes don't coordinate, the result might not be *globally optimal*.

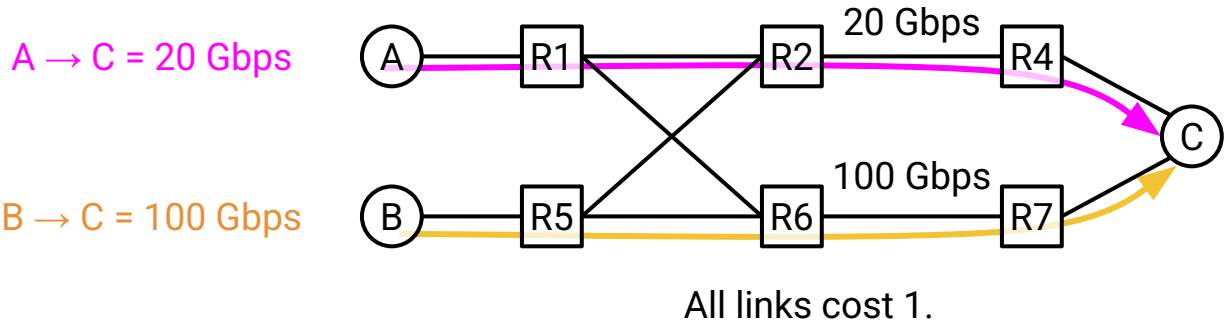
- In standard least-cost routing, routers could send A → C along the bottom path. (Valid shortest path.)
- Now, B → C will cause congestion on either the top or bottom path.



Traffic Engineering – Globally Optimal Decisions

Centralized routing lets us make *globally optimal* decisions.

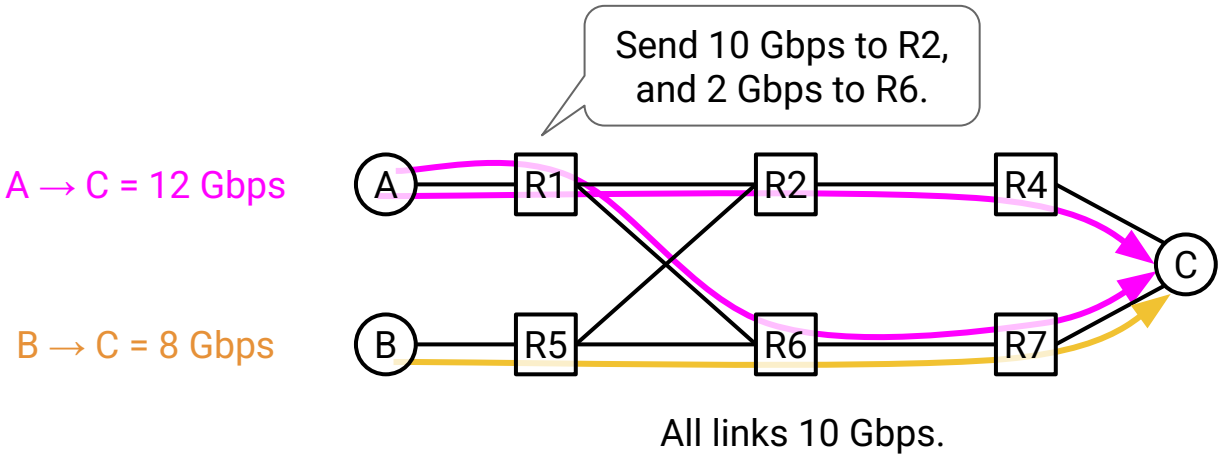
- A central operator could send A → C along the top path.
- Now, B → C can use the bottom path, and everybody is happy.
- Requires a central operator to coordinate all the routers.



Traffic Engineering – Splitting Flows

A centralized router can also let us split flows along different paths.

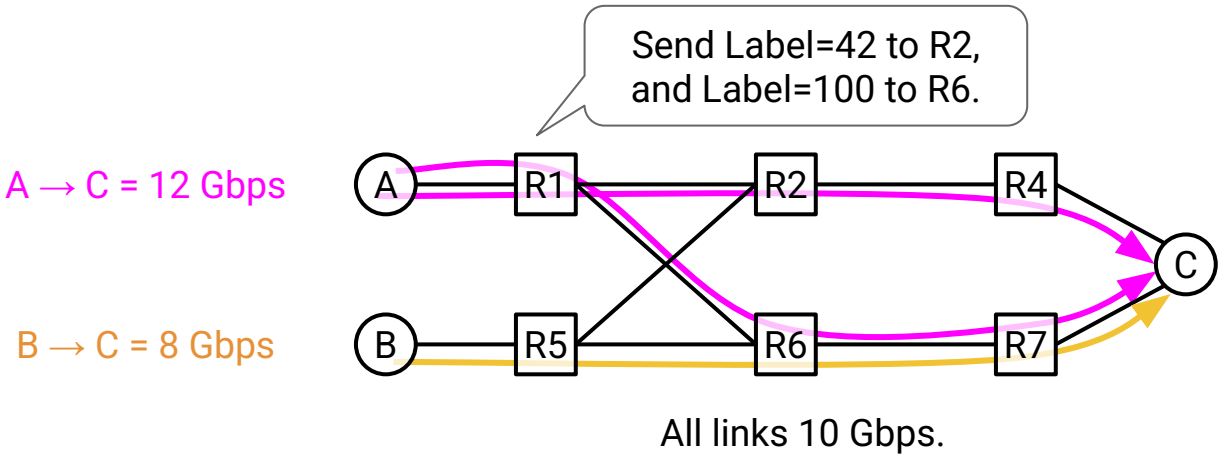
This results in better capacity use (efficiency) of the network.



Traffic Engineering – Splitting Flows

How do we actually split traffic across different paths?

- We can use **encapsulation**.
- Add a new header that indicates which path to take through the network.
- Now, routers forward based on the new header.
  - Simple enough task for commodity routers.



Benefits of SDN Control Planes

SDN control planes are **flexible**.

- No longer constrained by standards bodies and vendors.
- Operators can implement custom policies.
  - Geofencing: "Don't send traffic via links in Canada."
  - Diversity: "Use separate paths. If one link breaks, the other path will be ok."

SDN control planes are **simple**.

- Routing protocols can get really complex.
  - Example: Vendors need to support the needs of all its customers.
- With SDN control, the operator can implement only whatever they need.

SDN control planes are **centralized**.

- More efficient use of network resources (by making globally optimal decisions).
- Easier convergence: Operator uses global view of network to compute new paths.

Benefits of SDN Control Planes

B4: Experience with a Globally-Deployed Software Defined WAN Link

Sushant Jain et. al. (Google) *ACM SIGCOMM 2013*

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.

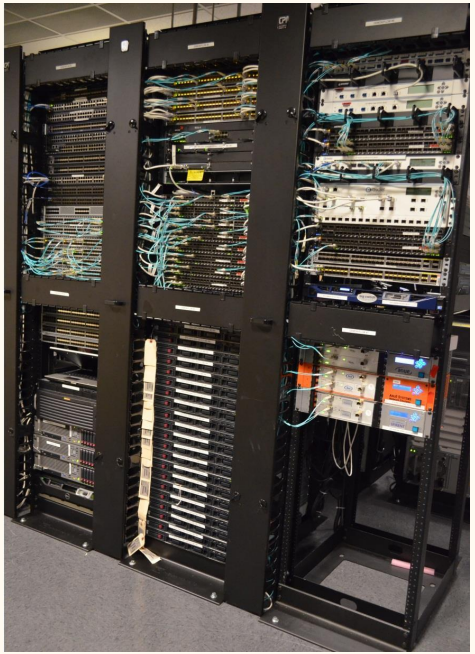
Benefits of SDN Control Planes

Achieving High Utilization with Software-Driven WAN Link

Ratul Mahajan et. al. (Microsoft) *ACM SIGCOMM 2013*

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



Drawbacks of SDN Control Planes

Reliability.

- Traditional networks: If one link fails, routers adapt and re-converge.
- SDN networks: If the controller fails, the network stops updating. Routers can't adapt to changes.

Scalability.

- Central controller must deal with decisions for everybody.

Complexity.

- Infrastructure challenge: Where do we put the controller?
- How do we reliability connect the controller to the routers?

Active area of research!

SDN in Datacenters

Lecture 20, CS 168, Spring 2026

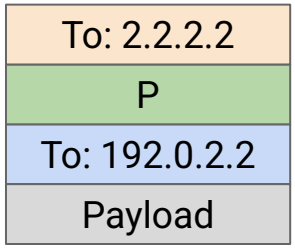
Software-Defined Networking

- A Brief History
- Implementation (OpenFlow API)
- Benefits (Traffic Engineering)
- **SDN in Datacenters**

SDN in Datacenters – Overlay

Recall: Encapsulation in datacenters.

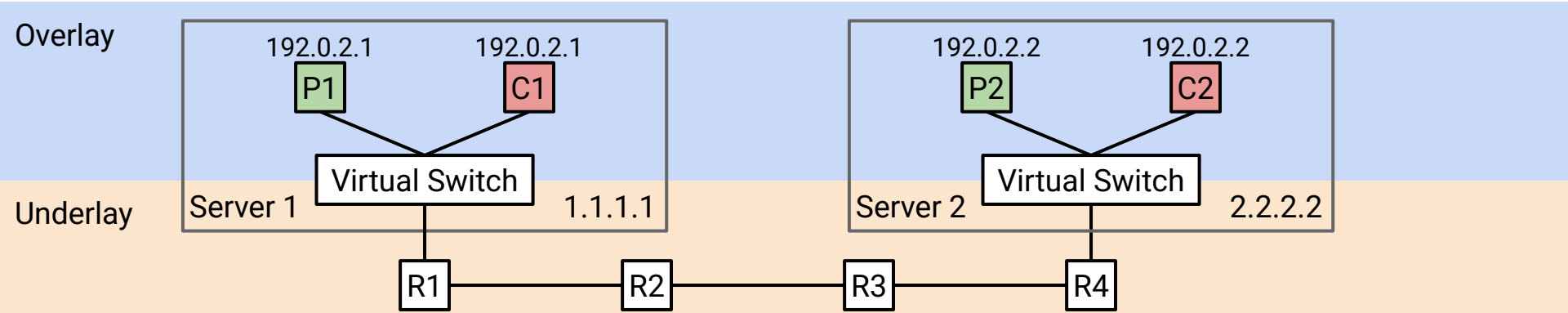
- We add a virtual network ID header to distinguish between tenants.
- We add an underlay header with the destination's physical address.



SDN lets us program servers to implement these headers.

The central controller can tell the servers:

- "192.0.2.2 Pepsi VM created on Server 2.2.2.2."
- "Coke network ID is 42."



SDN in Datacenters – Overlay

SDN handles distributing custom information in datacenters.

- Example: Virtual network IDs.
- This would otherwise have to be distributed/learnt by the router's control plane.

SDN allows the underlay of the network to remain simple.

- SDN maps overlay virtual addresses to underlay physical addresses.
- No need for the underlay to understand virtual addresses.

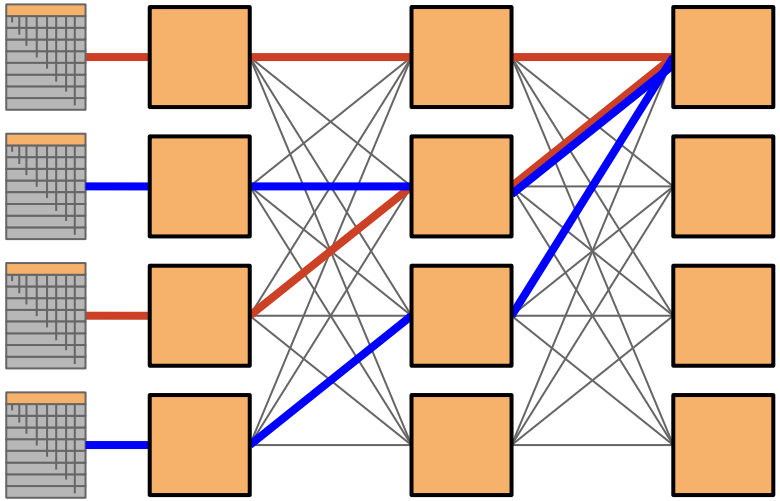
SDN allows the control plane to be extended to servers (e.g. virtual switches).

- Simpler mechanism for programming servers.
- No need to implement routing protocols for encapsulation.

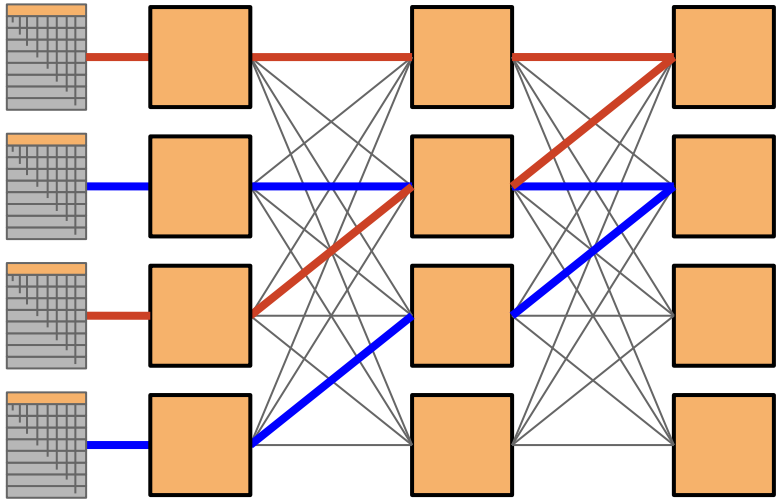
SDN in Datacenters – Underlay

The SDN underlay network is a physical network, just like any other.

- The benefits from earlier (simplicity, efficiency) apply to the underlay network too.
- Hyperscale datacenters might use SDN in both the overlay and underlay.
  - But these are decoupled systems.



Per-flow load balancing might cause 2 elephant flows to share a link.



A global controller can route traffic differently, improving performance.

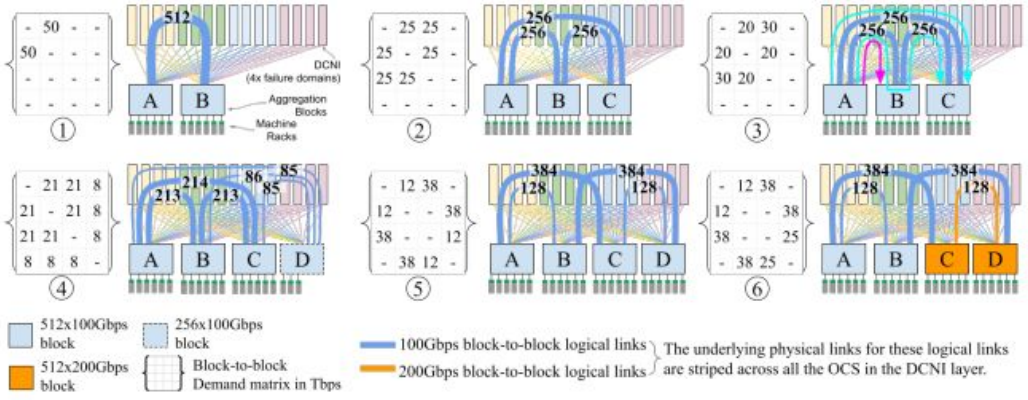
Improving Efficiency of Datacenters Using SDN

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

Enabled by SDN.

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

Leon Poutievski et. al. (Google) *ACM SIGCOMM 2022*



Summary: SDN

- SDN evolved from inflexibility of the control plane to meet different operational requirements.
- It splits the control and data planes to make the control plane more flexible.
- There are applications across both WAN and datacenter networks.
- Approaches exist to extend SDN to management and data planes as well, although it is not as widely-used as SDN for the control plane.
    - Key lesson: Well-defined, programmatic APIs that enable flexibility.