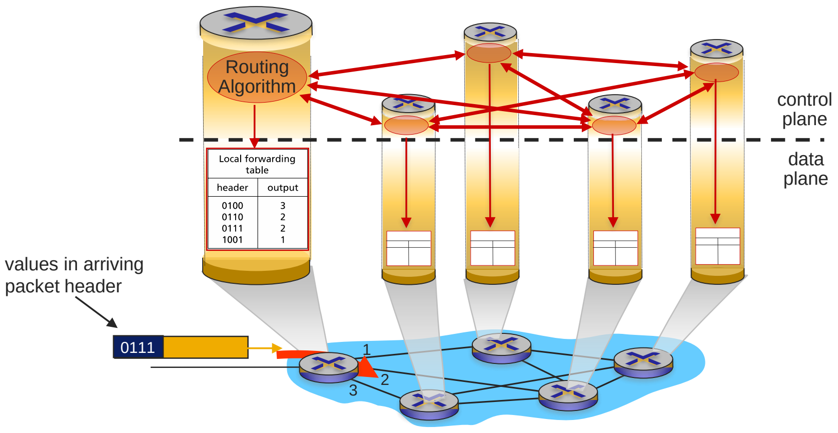
Software Defined Networks (SDNs)

Per-router control plane

Individual routing algorithm components *in each and every router* interact in the control plane

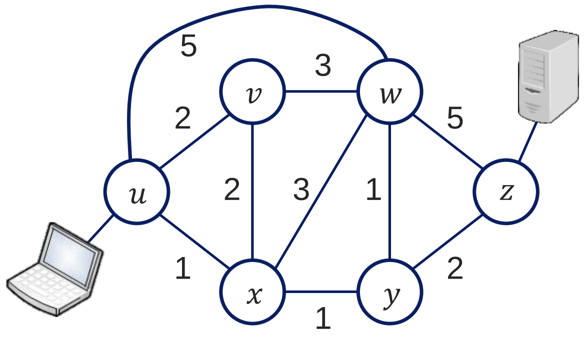


Traffic engineering

Difficulties in traditional routing

Q: what if network operator wants u-to-z traffic to flow along uvwz?

A: need to define link weights so traffic routing algorithm computes routes accordingly



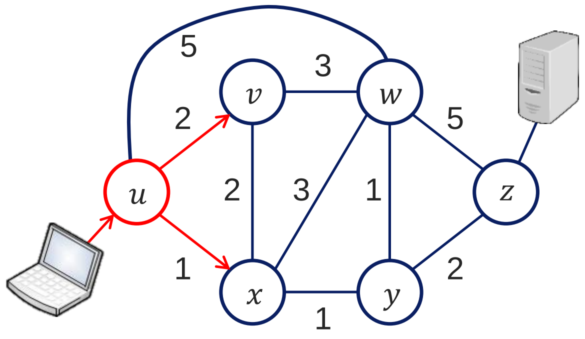
Link weights are only control "knobs" in distributed routing protocols: wrong!

Traffic engineering

Difficulties in traditional routing

Q: what if network operator wants to split u-to-z traffic along uvwz and uxyz (load balancing)?

A: can't do it with – DV or LS mechanisms do not keep per flow state

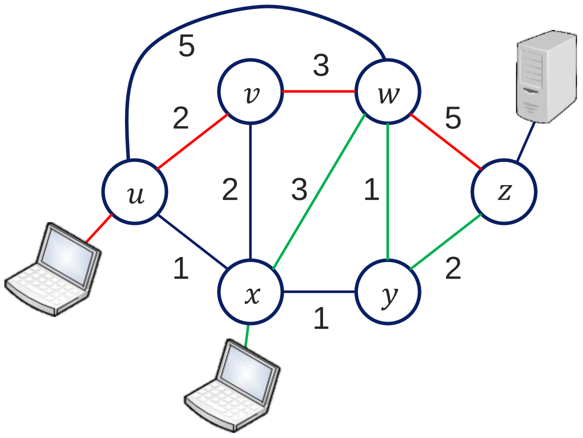


Traffic engineering

Difficulties in traditional routing

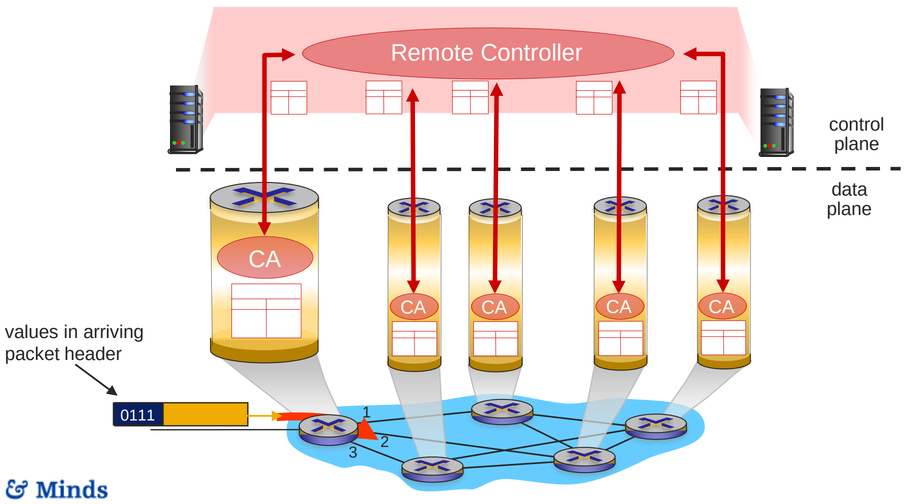
Q: what if w wants to route blue and red traffic differently?

A: can't do it (with destination based forwarding, and LS, DV routing)



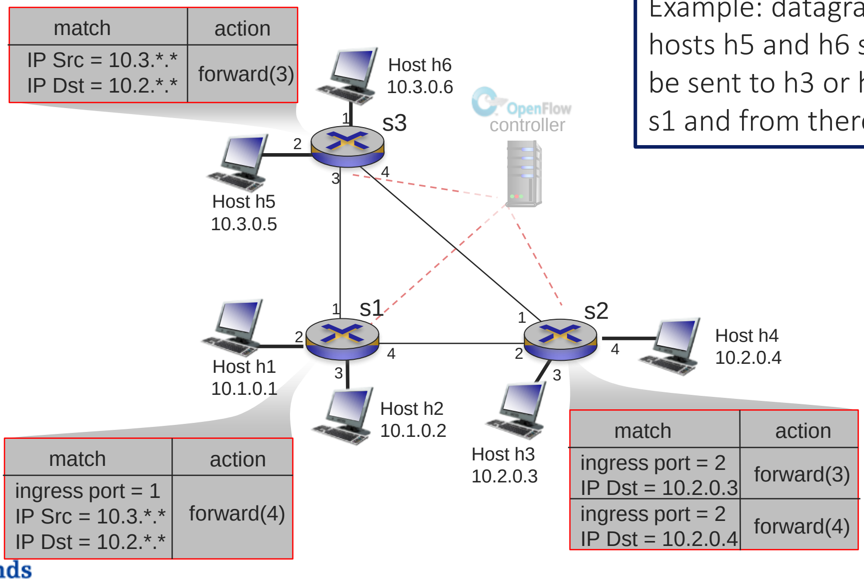
Logically centralized control plane

A logically centralized controller interacts with local control agents (CAs) and installs forwarding rules (forwarding tables)



OpenFlow example

Example: datagrams from hosts h5 and h6 should be sent to h3 or h4, via s1 and from there to s2



match	action
IP Src = 10.3.*.* IP Dst = 10.2.*.*	forward(3)

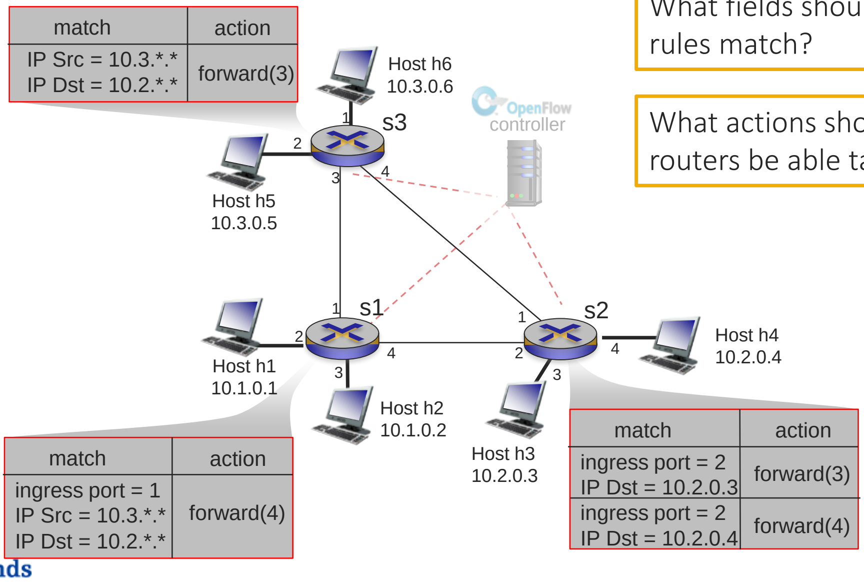
match	action
ingress port = 1 IP Src = 10.3.*.* IP Dst = 10.2.*.*	forward(4)

match	action
ingress port = 2 IP Dst = 10.2.0.3	forward(3)
ingress port = 2 IP Dst = 10.2.0.4	forward(4)

OpenFlow example

What fields should SDN rules match?

What actions should routers be able take?

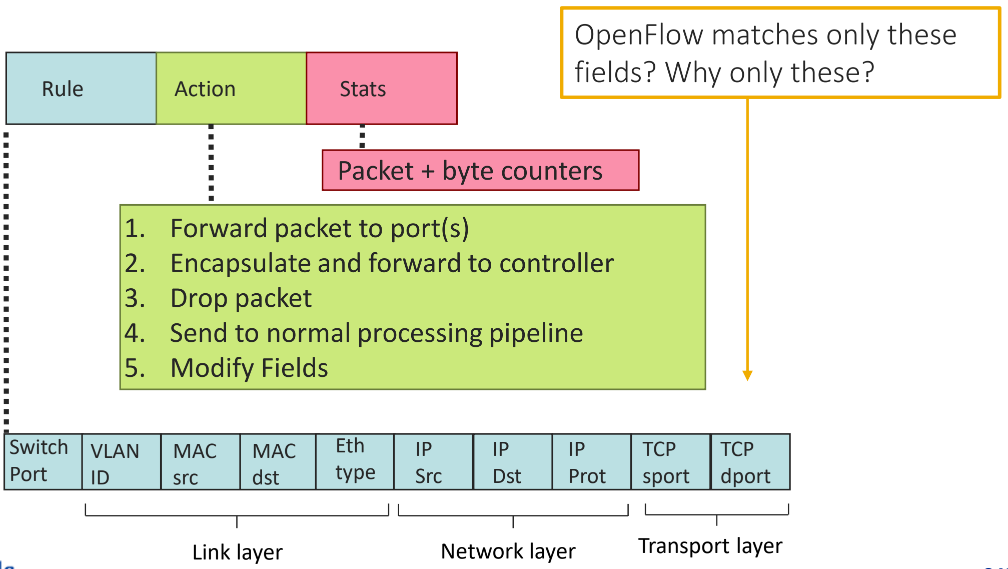


match	action
IP Src = 10.3.*.* IP Dst = 10.2.*.*	forward(3)

match	action
ingress port = 1 IP Src = 10.3.*.* IP Dst = 10.2.*.*	forward(4)

match	action
ingress port = 2 IP Dst = 10.2.0.3	forward(3)
ingress port = 2 IP Dst = 10.2.0.4	forward(4)

OpenFlow: Flow Table Entries



1. Forward packet to port(s)
2. Encapsulate and forward to controller
3. Drop packet
4. Send to normal processing pipeline
5. Modify Fields

Examples

Destination-based forwarding:

Pattern → ← Action

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	*	*	*	*	51.6.0.8	*	*	*	port6

IP datagrams destined to IP address 51.6.0.8 should be forwarded to router output port 6

Firewall:

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Forward
*	*	*	*	*	*	*	*	*	22	drop

do not forward (block) all datagrams destined to TCP port 22

Destination-based layer 2 (switch) forwarding:

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	22:A7:23: 11:E1:02	*	*	*	*	*	*	*	*	port3

layer 2 frames from MAC address 22:A7:23:11:E1:02 should be forwarded to output port 6

OpenFlow abstraction

match+action: unifies different kinds of devices

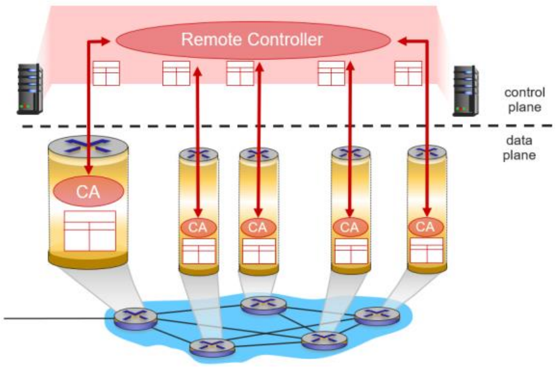
- Router
  - match: longest destination IP prefix
  - action: forward out a link
- Switch
  - match: destination MAC address
  - action: forward or flood
- Firewall
  - match: IP addresses and TCP/UDP port numbers
  - action: permit or deny
- NAT
  - match: IP address and port
  - action: rewrite address and port

SDN Implementation

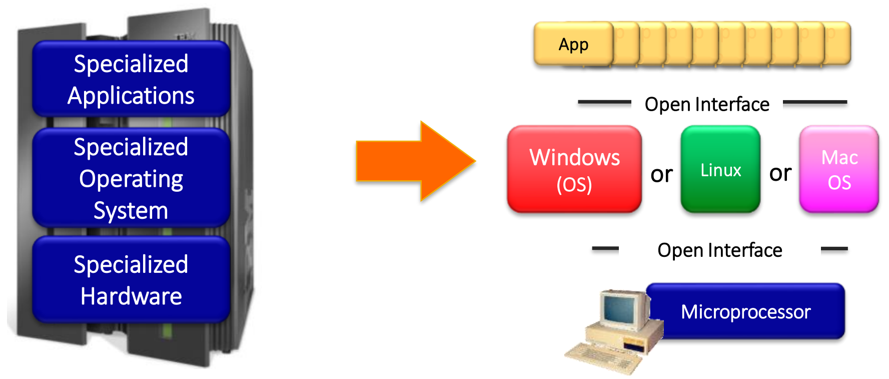
- So we have switches and a logically centralized controller.

1. What is the functionality of the controller?

2. How would you organize that functionality architecturally into layers and functional units?



Analogy: mainframe to PC evolution



- Vertically integrated
- Closed, proprietary
- Slow innovation
- Small industry

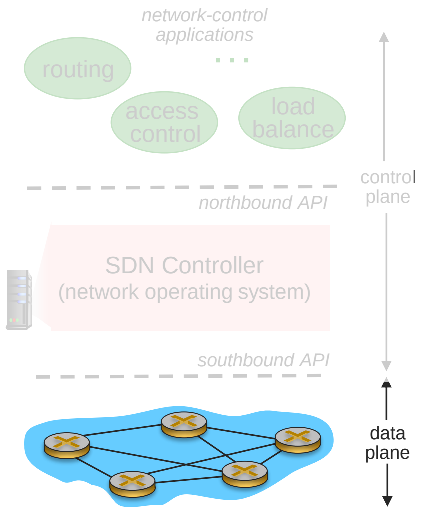
- Horizontal
- Open interfaces
- Rapid innovation
- Huge industry

249

SDN perspective: data plane switches

Data plane switches

- Fast, simple, commodity switches implementing generalized data-plane forwarding in hardware
- Switch flow table computed, installed by controller
- API for table-based switch control (e.g., OpenFlow)
  - defines what is controllable and what is not
- Protocol for communicating with controller (e.g., OpenFlow)



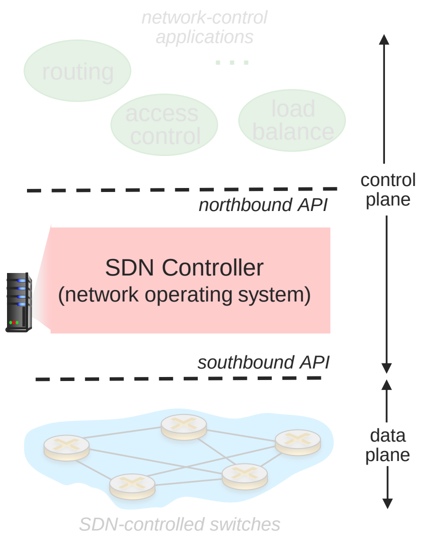
SDN-controlled switches

250

SDN perspective: data plane switches

SDN controller (network OS):

- Maintain network state information
- Interacts with network control applications "above" via northbound API
- Interacts with network switches "below" via southbound API
- Implemented as distributed system for performance, scalability, fault-tolerance, robustness

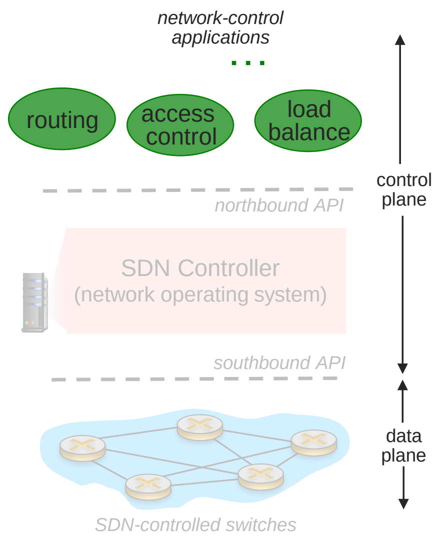


SDN perspective: data plane switches

Network-control apps:

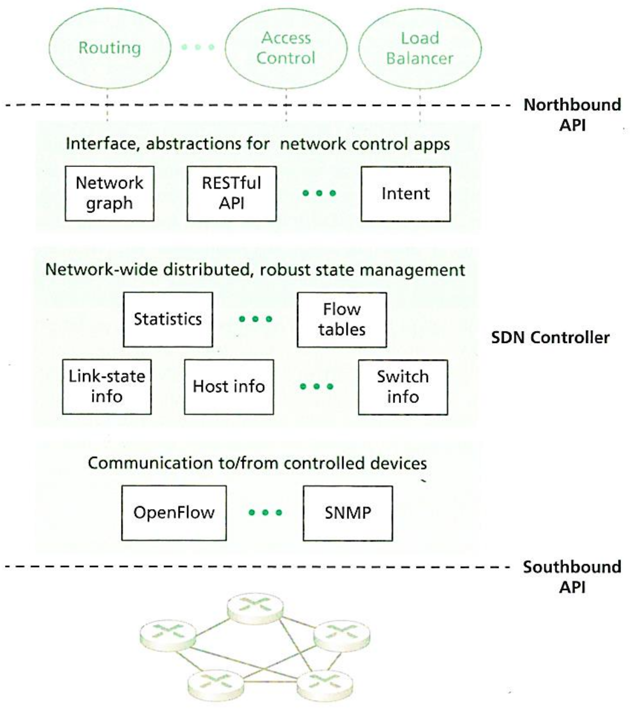
- "Brains" of control: implement control functions using lower-level services, API provided by SDN controller
- Unbundled: can be provided by 3rd party: distinct from routing vendor, or SDN controller

Can we do anything? Transport (flow and congestion control, reliable delivery)? Application (caching)?



SDN Organization

- Interface layer to network control apps:
  - Abstractions API
- Network-wide state management layer:
  - State of networks links, switches, services: a distributed database
- Communication layer:
  - Communicate between SDN controller and controlled switches



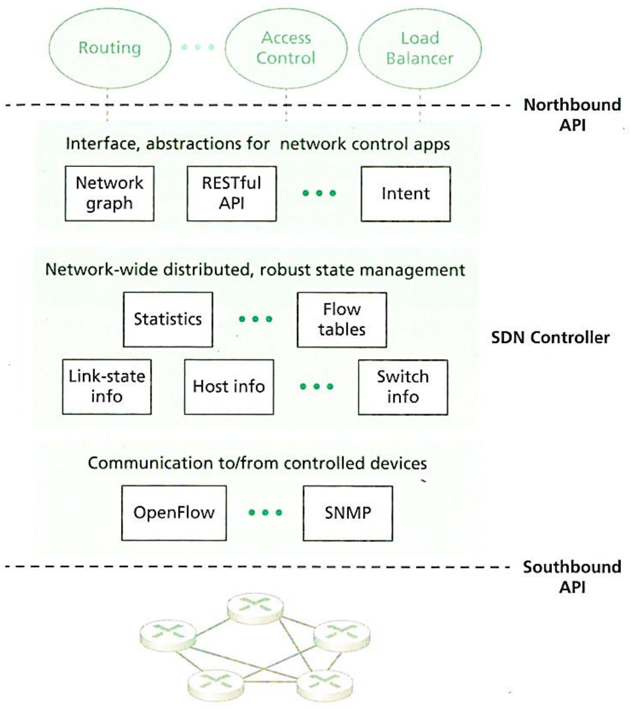
Mountains & Minds

253

SDN Organization

What information needs is exchanged across the Southbound API?

- Southbound API
  - Switch to controller
    - Port status – includes hosts
    - Packet in – includes packets
    - Flow removed – timeout
  - Controller to switch
    - Read state – counters
    - Modify state – set forwarding table
    - Send packet – includes packet



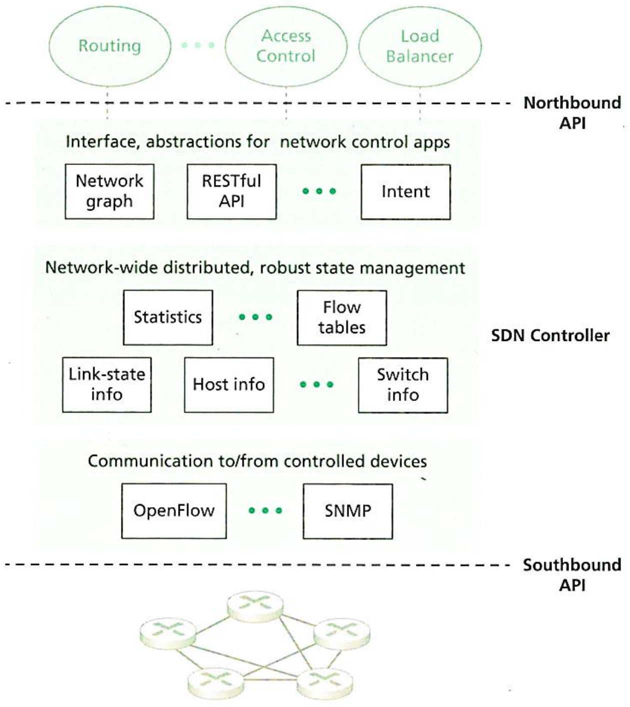
Mountains & Minds

254

SDN Organization

What information needs is exchanged across the Northbound API?

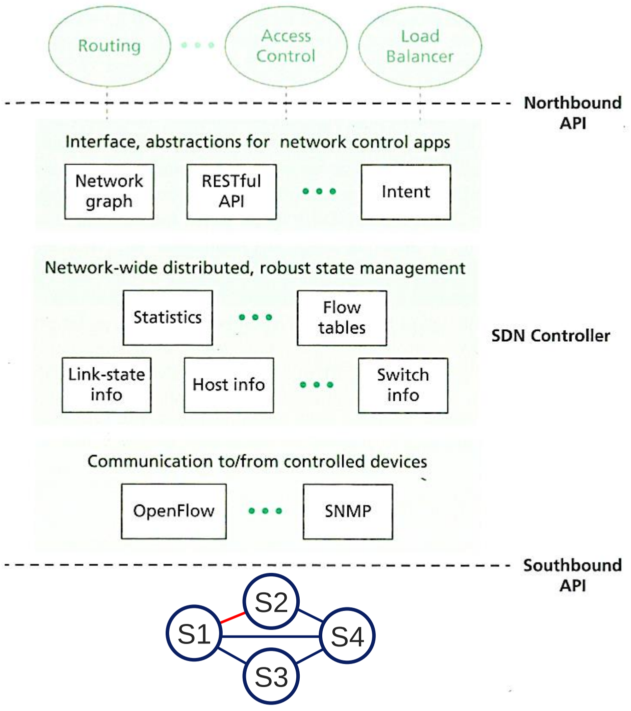
- Northbound API
  - Accept intents
  - Set flow tables explicitly
  - Query network graph
  - Subscribe to notifications



Mountains & Minds

SDN: control/data plane interaction

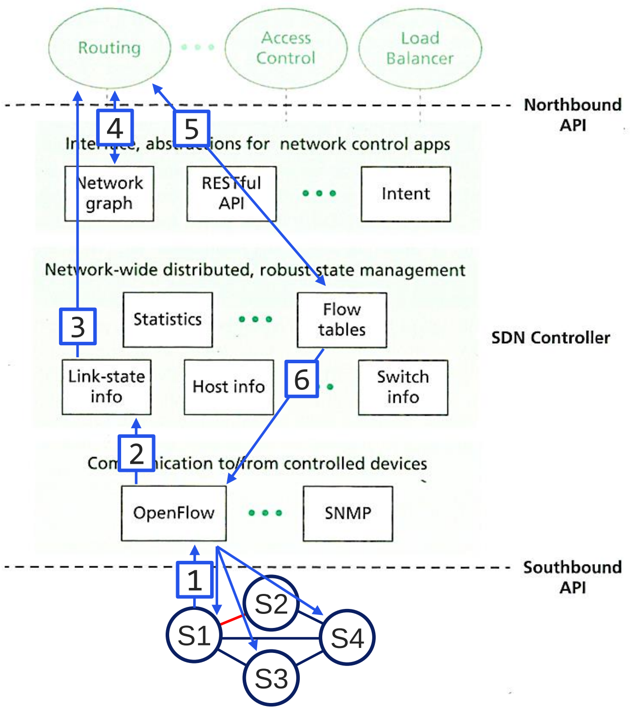
What would be the set of steps within the SDN network when the link S1-S2 fails?



Mountains & Minds

SDN: control/data plane interaction

1. S1, experiencing link failure using OpenFlow port status message to notify controller
2. SDN controller receives OpenFlow message, updates link status info
3. Dijkstra's routing algorithm application has previously registered to be called when ever link status changes. It is called.
4. Dijkstra's routing algorithm access network graph info, link state info in controller, computes new routes
5. Link state routing app interacts with flow-table-computation component in SDN controller, which computes new flow tables needed
6. Controller uses OpenFlow to install new tables in switches that need updating

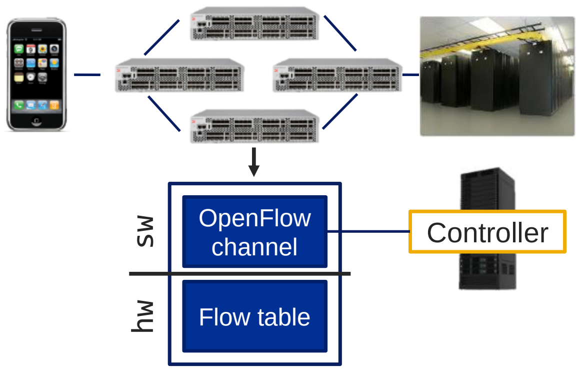


Mountains & Minds

Software Defined Networks (SDNs)

MONTANA STATE UNIVERSITY

- SDN is a strategy for system, software, functionality development
  - Holistic L2-7 management platforms like OpenDaylight
  - Frenetic (Ox, Merlin, Nettle) declarative languages for network configuration
- Avoid dogma!
  "If you listen to the hype, software-defined networking and other software-defined technology will solve all data center problems and allow your computing gear to be an undifferentiated mass of identical nodes."
- Limitations:
  - Inter-domain traffic?
  - Network virtualization and latency?
  - Computation speed vs. forwarding speed?
  - Table size explosion!



- Different controllers
  - OpenFlow – largest support for v.1
  - Floodlight – open source
  - OpenDaylight, OpenContrail – L2-7
  - Cisco APIC – proprietary extensions

Mountains & Minds