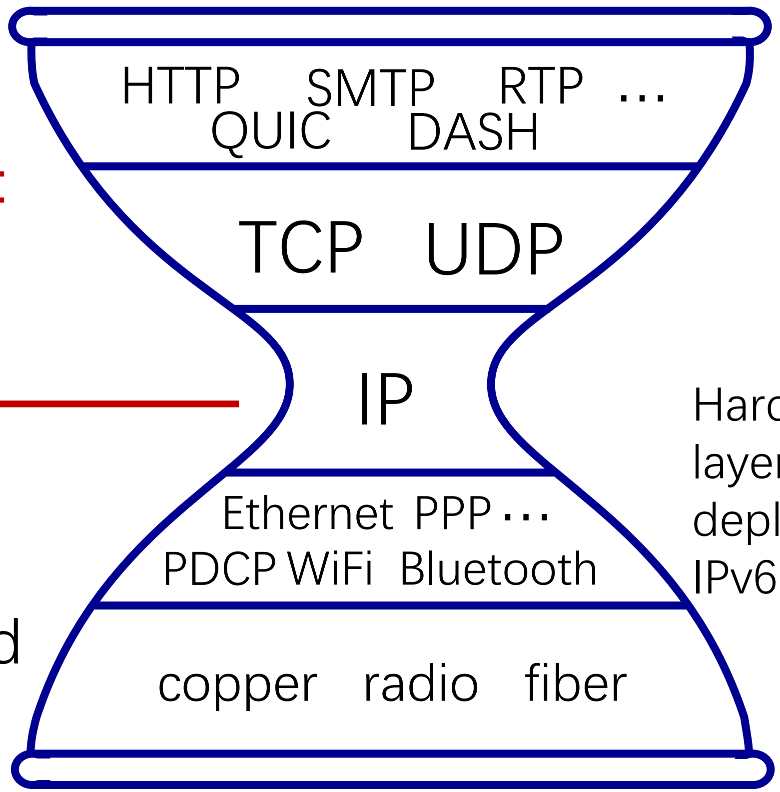
CS120: Computer Networks

Lecture 13. Other Topics in IP (SDN)

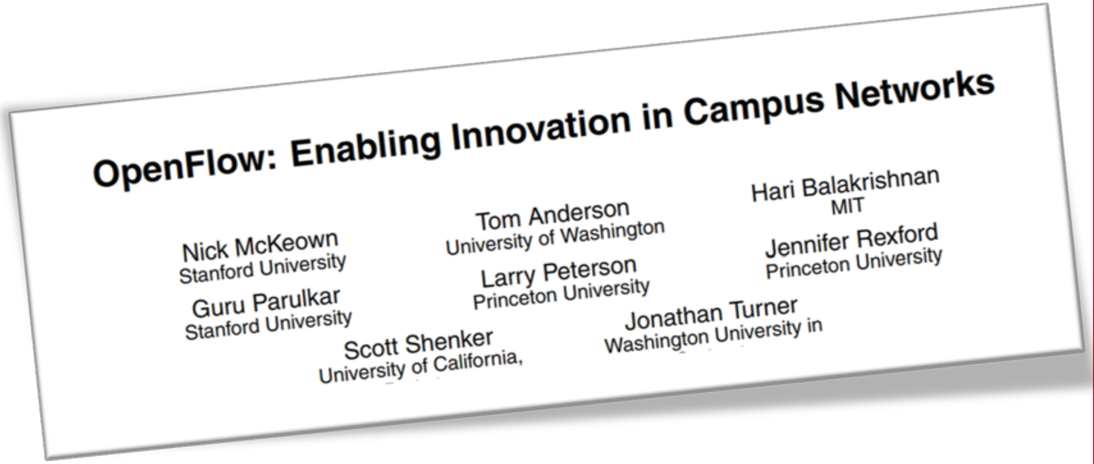
Zhice Yang

A Brief History

Internet's "thin waist":
- *one* network layer protocol: IP
- *must* be implemented by every (billions) of Internet-connected devices



Hard for new network layer protocols to deployment. Think about IPv6



OpenFlow: Enabling Innovation in Campus Networks

Nick McKeown, Stanford University
Guru Parulkar, Stanford University
Tom Anderson, University of Washington
Larry Peterson, Princeton University
Scott Shenker, University of California,
Hari Balakrishnan, MIT
Jennifer Rexford, Princeton University
Jonathan Turner, Washington University in

A Brief History

- Internet network layer: historically implemented via distributed, per-router control approach:
  - *monolithic* router contains switching hardware, runs proprietary implementation of Internet standard protocols (IP, RIP, OSPF, BGP) in proprietary router OS (e.g., Cisco IOS)
  - different "middleboxes" for different network layer functions: firewalls, NAT boxes, ..
- ~2005: renewed interest in rethinking the network layer

Generalized Forwarding

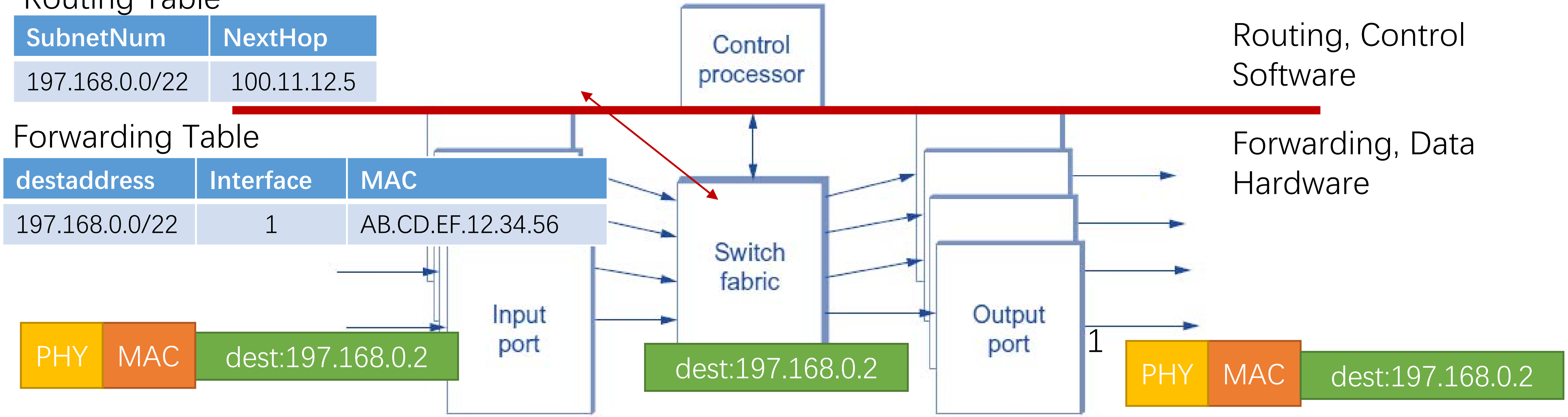
- Destination-based forwarding: forward based on dest. IP address

Routing Table

SubnetNum	NextHop
197.168.0.0/22	100.11.12.5

Forwarding Table

destaddress	Interface	MAC
197.168.0.0/22	1	AB.CD.EF.12.34.56



Routing, Control Software

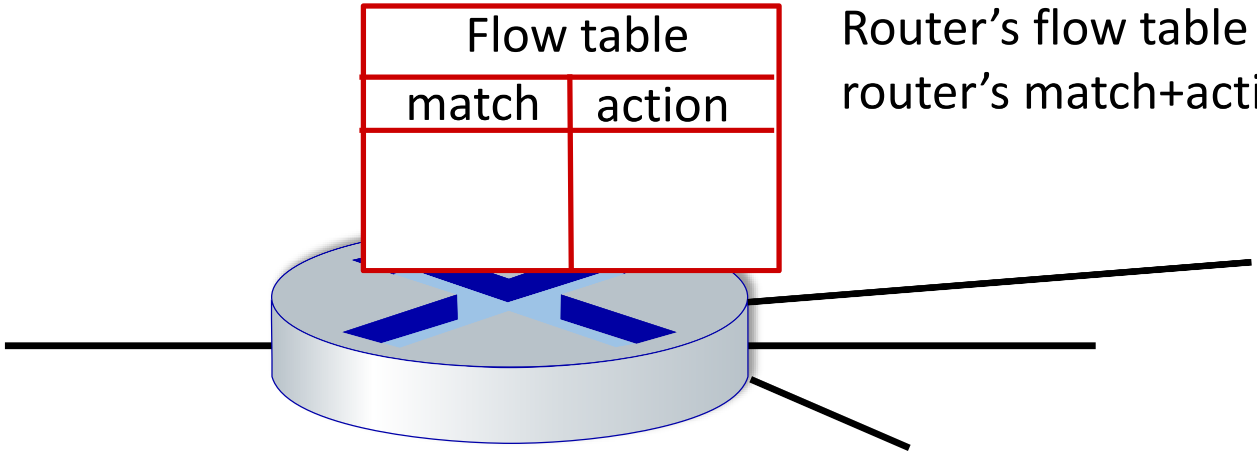
Forwarding, Data Hardware

Generalized Forwarding

- Destination-based forwarding: forward based on dest. IP address
- Background: modern switches/routers are programmable
- Generalized forwarding:
  - Many header fields can determine action
  - Many actions possible: forward/drop/copy/modify/log packet

Flow Table Abstraction

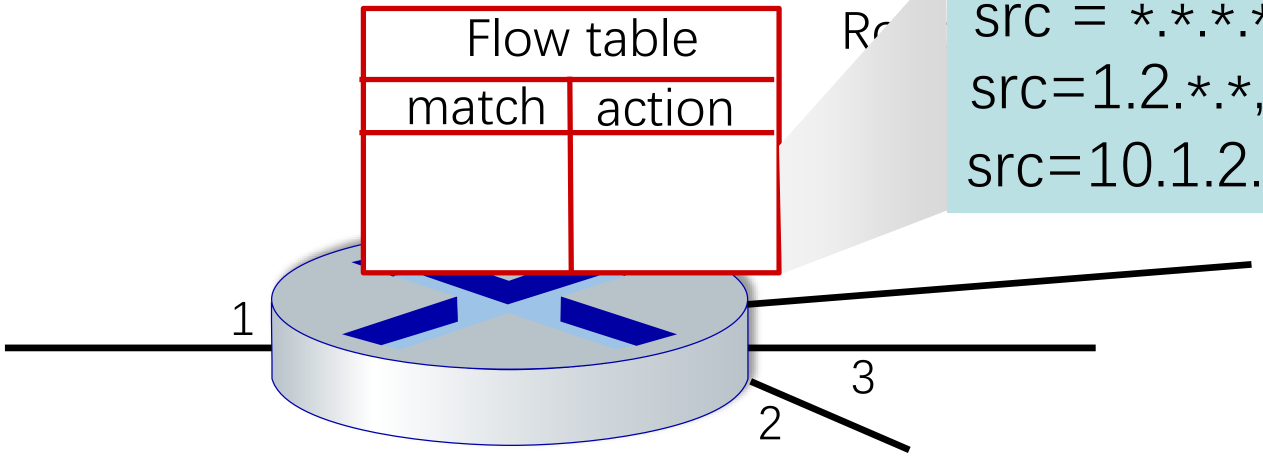
- Flow: defined by header field values (in link-, network-, transport-layer fields)
- Generalized forwarding: simple packet-handling rules
  - match: pattern values in packet header fields
  - actions: for matched packet: drop, forward, modify, matched packet or send matched packet to controller
  - priority: disambiguate overlapping patterns
  - counters: #bytes and #packets



Router's flow table define router's match+action rules

Flow Table Abstraction

- Flow: defined by header field values (in link-, network-, transport-layer fields)
- Generalized forwarding: simple packet-handling rules
  - match: pattern values in packet header fields
  - actions: for matched packet: drop, forward, modify, matched packet or send matched packet to controller
  - priority: disambiguate overlapping patterns
  - counters: #bytes and #packets



src = *.*.*.*, dest=3.4.*.*	forward to port2
src=1.2.*.*, dest=*.*.*.*	drop
src=10.1.2.3, dest=*.*.*.*	send to controller

* : wildcard

OpenFlow: Flow Table Entries

Match	Action	Stats

Packet + byte counters

1. Forward packet to port(s)
2. Drop packet
3. Modify fields in header(s)
4. Encapsulate and forward to controller

Header fields to match:

Ingress Port	Src MAC	Dst MAC	Eth Type	VLAN ID	VLAN Pri	IP Src	IP Dst	IP Proto	IP ToS	TCP/UDP Src Port	TCP/UDP Dst Port
	Link layer					Network layer				Transport layer	

OpenFlow: Examples

Destination-based forwarding:

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	VLAN Pri	IP Src	IP Dst	IP Prot	IP ToS	TCP s-port	TCP d-port	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	VLAN Pri	IP Src	IP Dst	IP Prot	IP ToS	TCP s-port	TCP d-port	Action
*	*	*	*	*	*	*	*	*	*	*	22	drop

Block (do not forward) all datagrams destined to TCP port 22 (ssh port #)

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	VLAN Pri	IP Src	IP Dst	IP Prot	IP ToS	TCP s-port	TCP d-port	Action
*	*	*	*	*	*	128.119.1.1	*	*	*	*	*	drop

Block (do not forward) all datagrams sent by host 128.119.1.1

OpenFlow: Examples

Layer 2 destination-based forwarding:

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	VLAN Pri	IP Src	IP Dst	IP Prot	IP ToS	TCP s-port	TCP d-port	Action
*	*	22:A7:23:11:E1:02	*	*	*	*	*	*	*	*	*	port3

layer 2 frames with destination MAC address 22:A7:23:11:E1:02 should be forwarded to output port 3

OpenFlow Abstraction

- match+action: abstraction 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

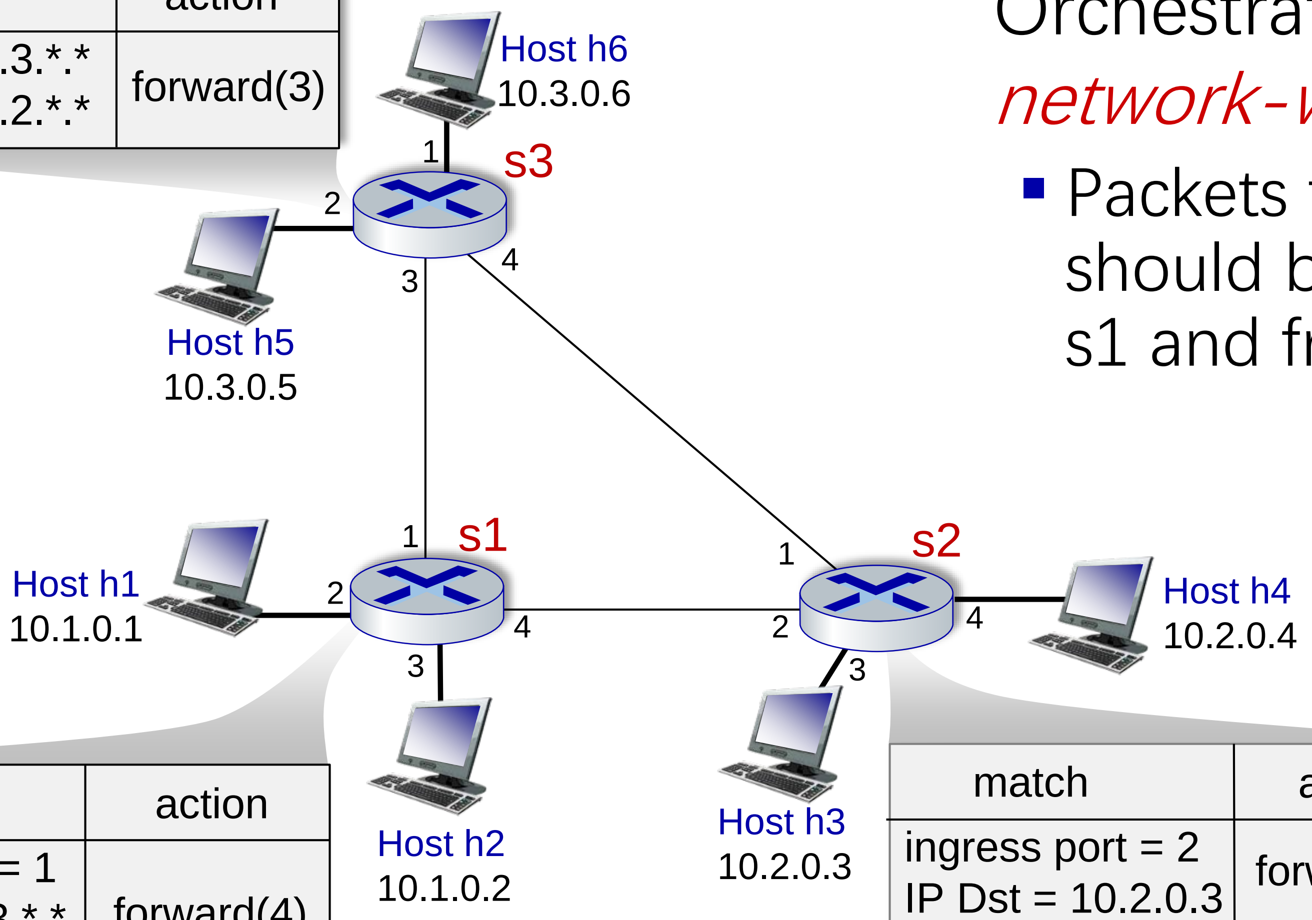
Firewall
- *match*: IP addresses and TCP/UDP port numbers
- *action:* permit or deny

NAT
- *match:* IP address and port
- *action:* rewrite address and port

OpenFlow: Fully Flexible Traffic Engineering

Orchestrated tables can create *network-wide* behavior, e.g.,:

- Packets from hosts h5 and h6 should be sent to h3 or h4, via s1 and from there to s2



s3:

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

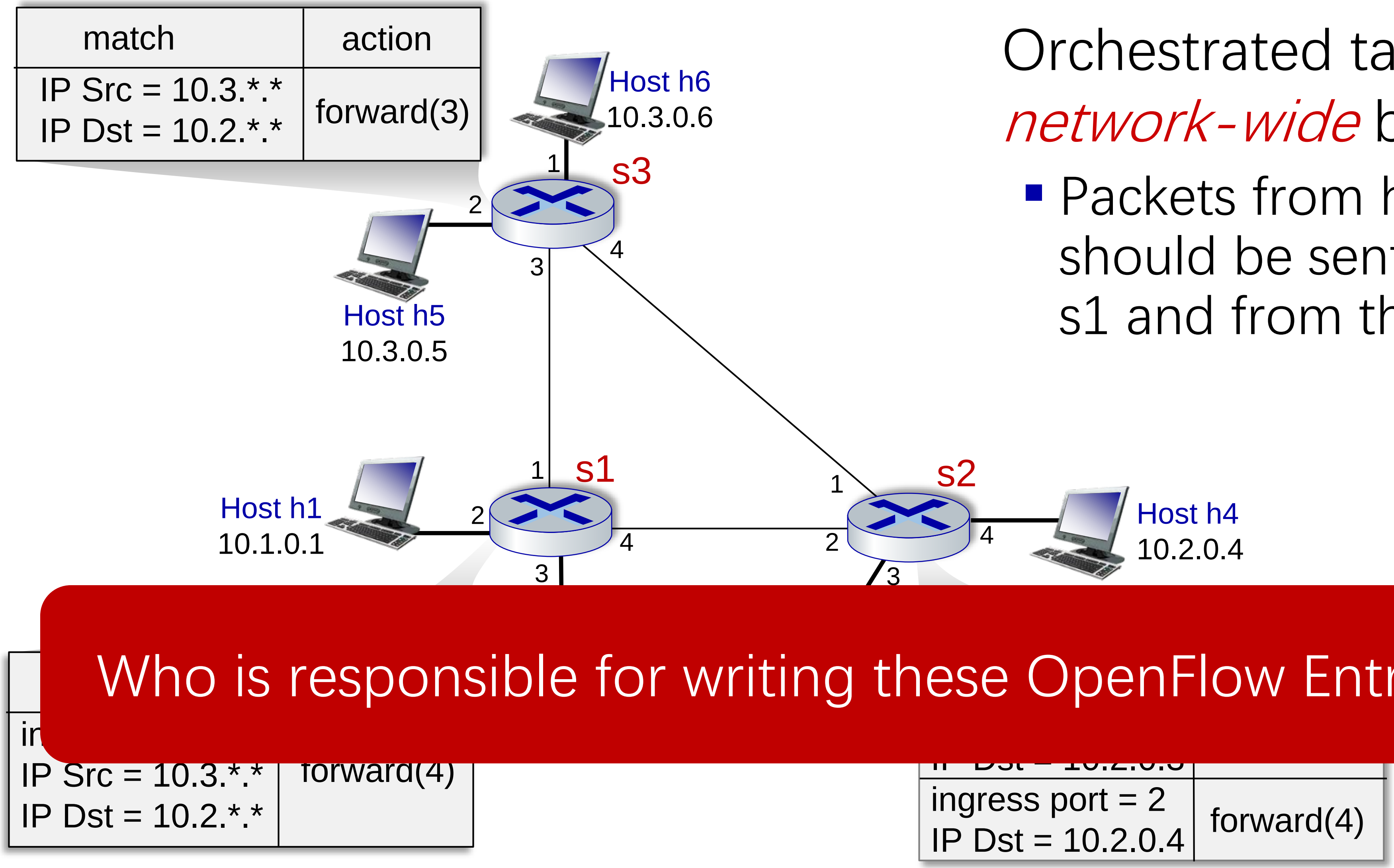
s1:

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

s2:

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: Fully Flexible Traffic Engineering

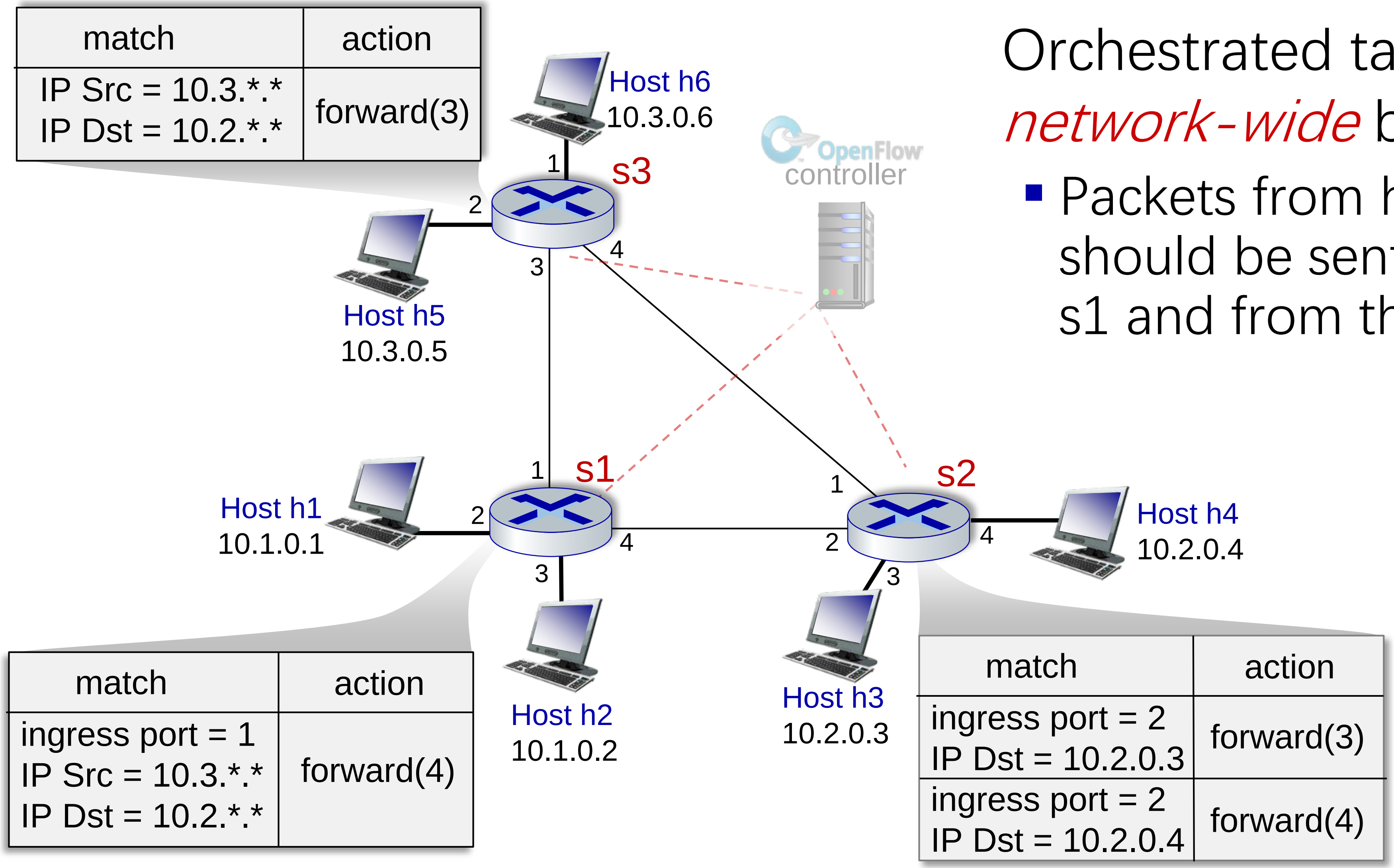


Orchestrated tables can create *network-wide* behavior, e.g.,:

- Packets from hosts h5 and h6 should be sent to h3 or h4, via s1 and from there to s2

Who is responsible for writing these OpenFlow Entries ?

OpenFlow Controller

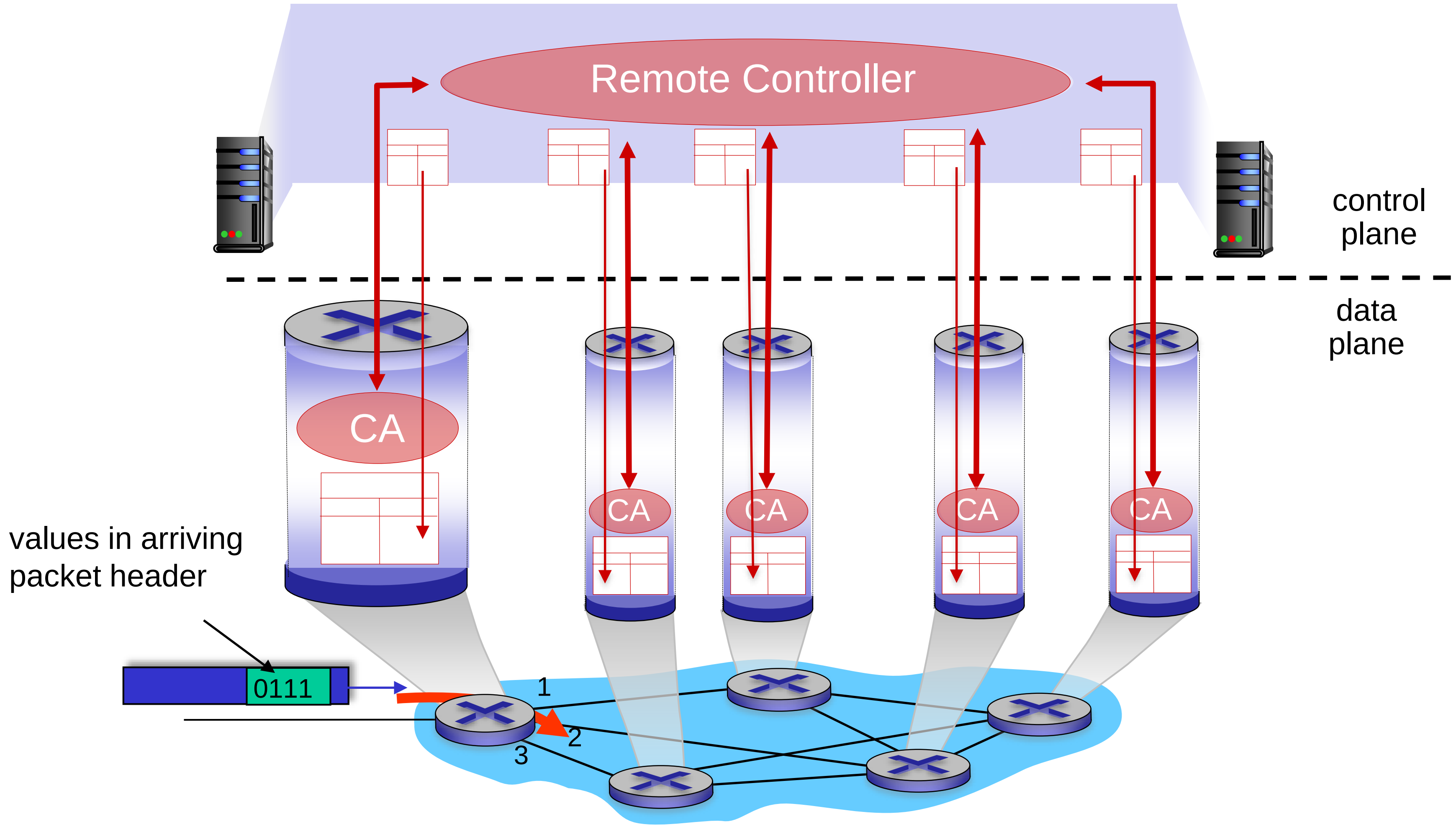


Orchestrated tables can create *network-wide* behavior, e.g.,:

- Packets from hosts h5 and h6 should be sent to h3 or h4, via s1 and from there to s2

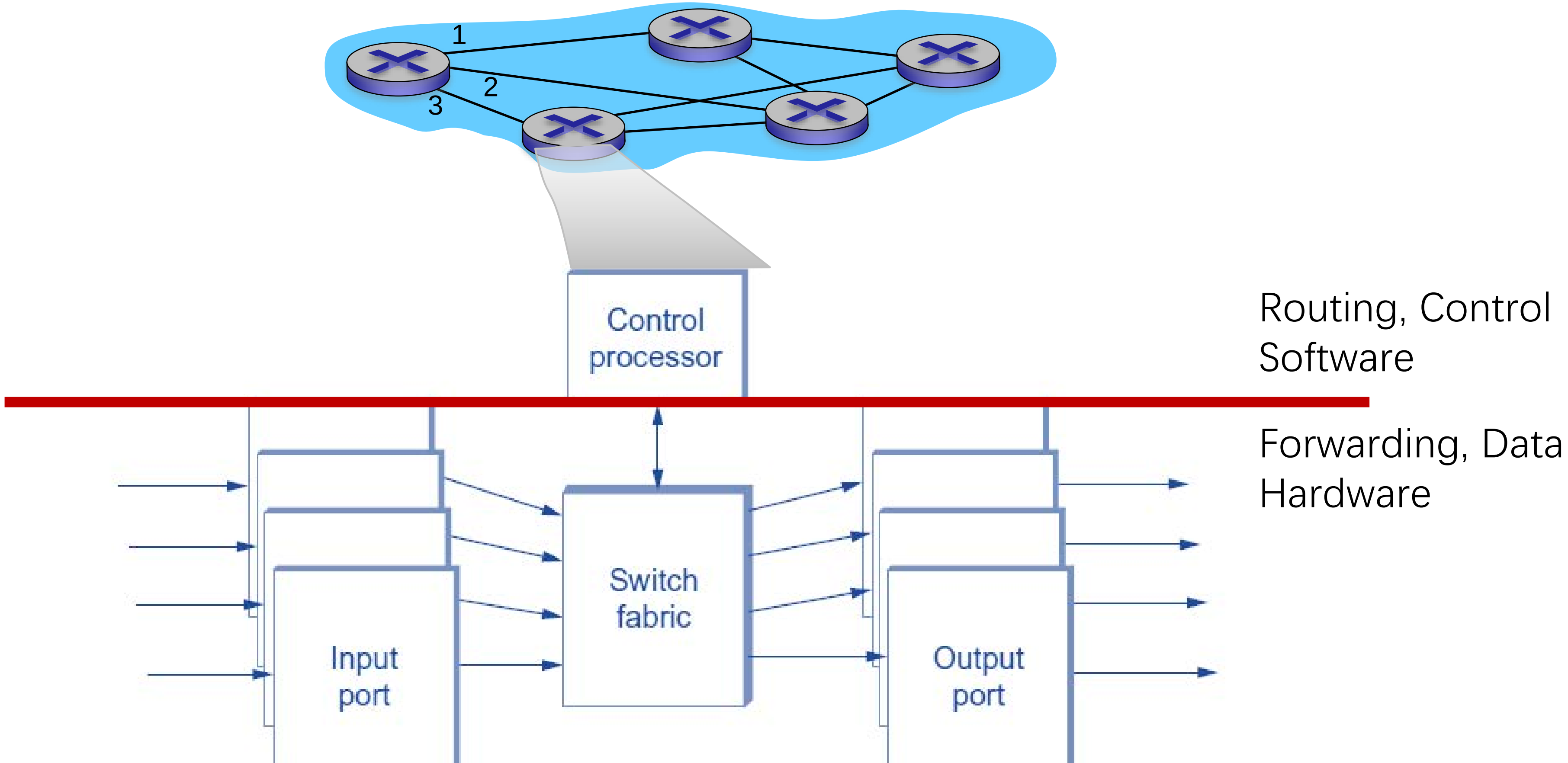
SDN Control Plane

Remote controller computes, installs forwarding tables in routers



Per-router Control Plane

- Individual routing algorithm components in each and every router interact in the control plane to computer forwarding tables



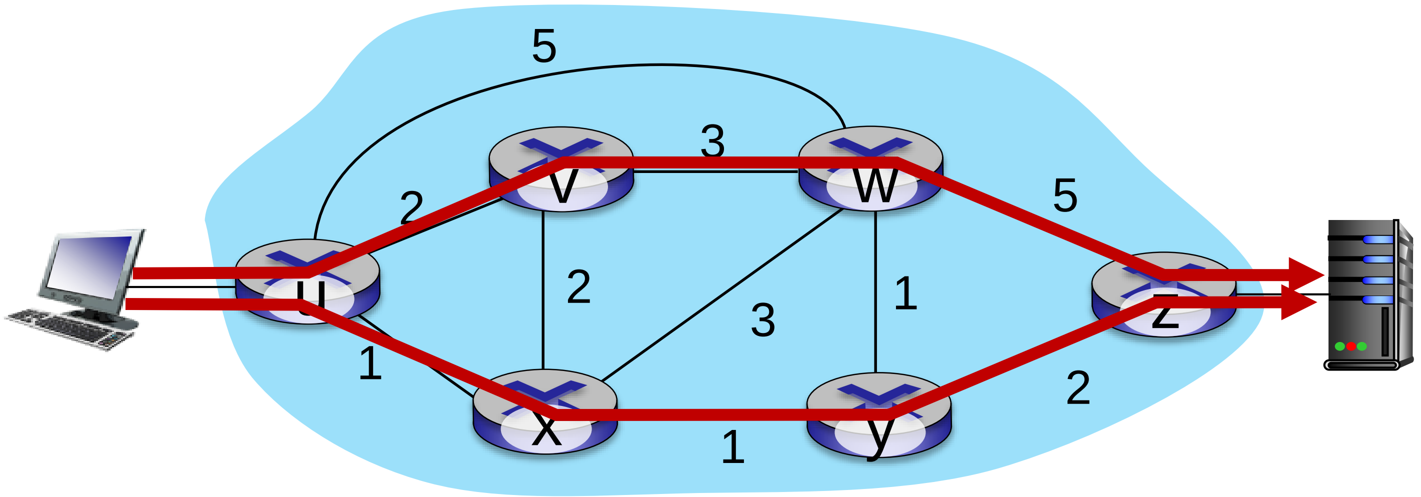
Software defined networking (SDN)

Why a *logically centralized* control plane?

- Easier network management: avoid router misconfigurations, greater flexibility of traffic flows
- Table-based forwarding (recall OpenFlow API) allows "programming" routers
  - Centralized "programming" easier: compute tables centrally and distribute
  - Distributed "programming" more difficult: compute tables as result of distributed algorithm (protocol) implemented in each-and-every router
- Open (non-proprietary) implementation of control plane
  - Foster innovation: let 1000 flowers bloom

Example: Traffic Engineering

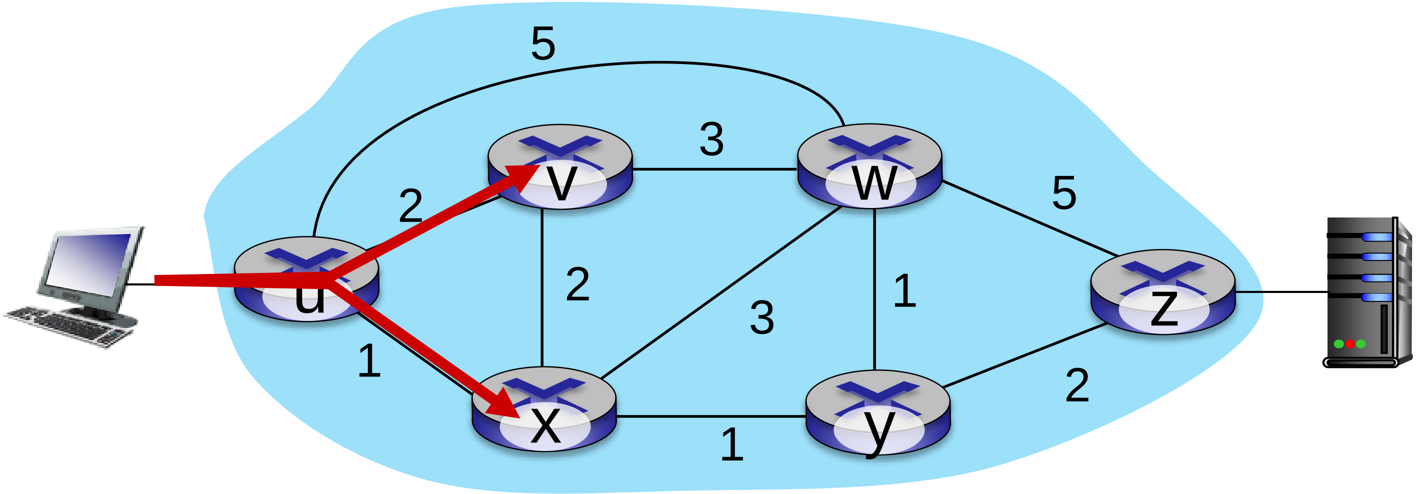
-Difficult with traditional routing



- Want u-to-z traffic to flow along *uvwz*, rather than *uxyz*?
- OSPF: need to re-define link weights so traffic routing algorithm computes routes accordingly (or need a new routing algorithm)!

Example: Traffic Engineering

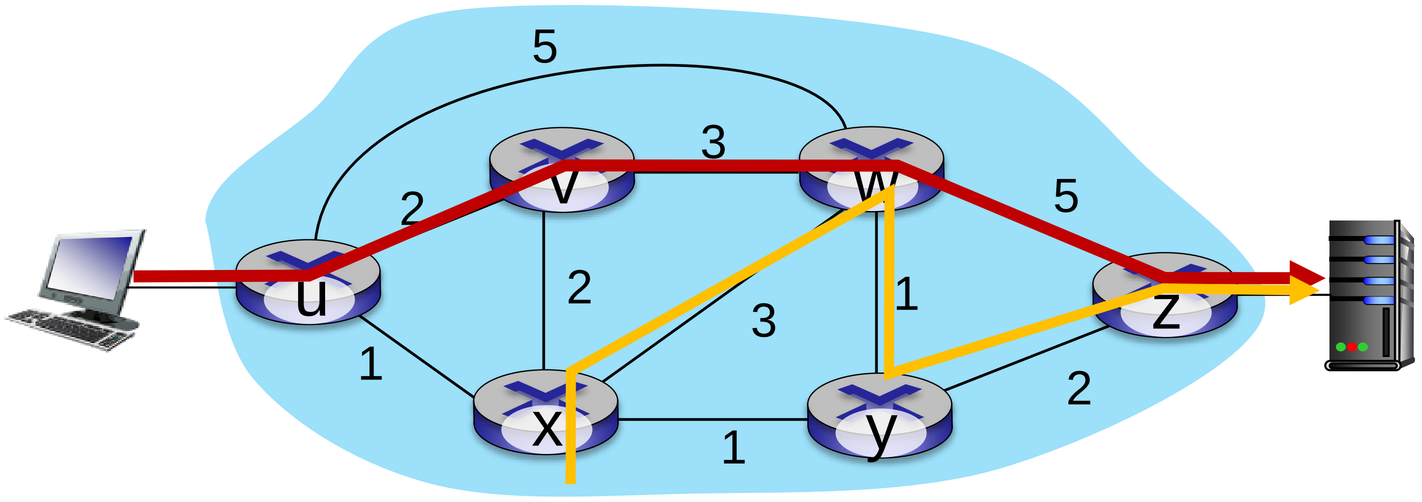
-Difficult with traditional routing



- Split u-to-z traffic along uvwz *and* uxyz (load balancing)?
- Can't do it (or need a new routing algorithm)

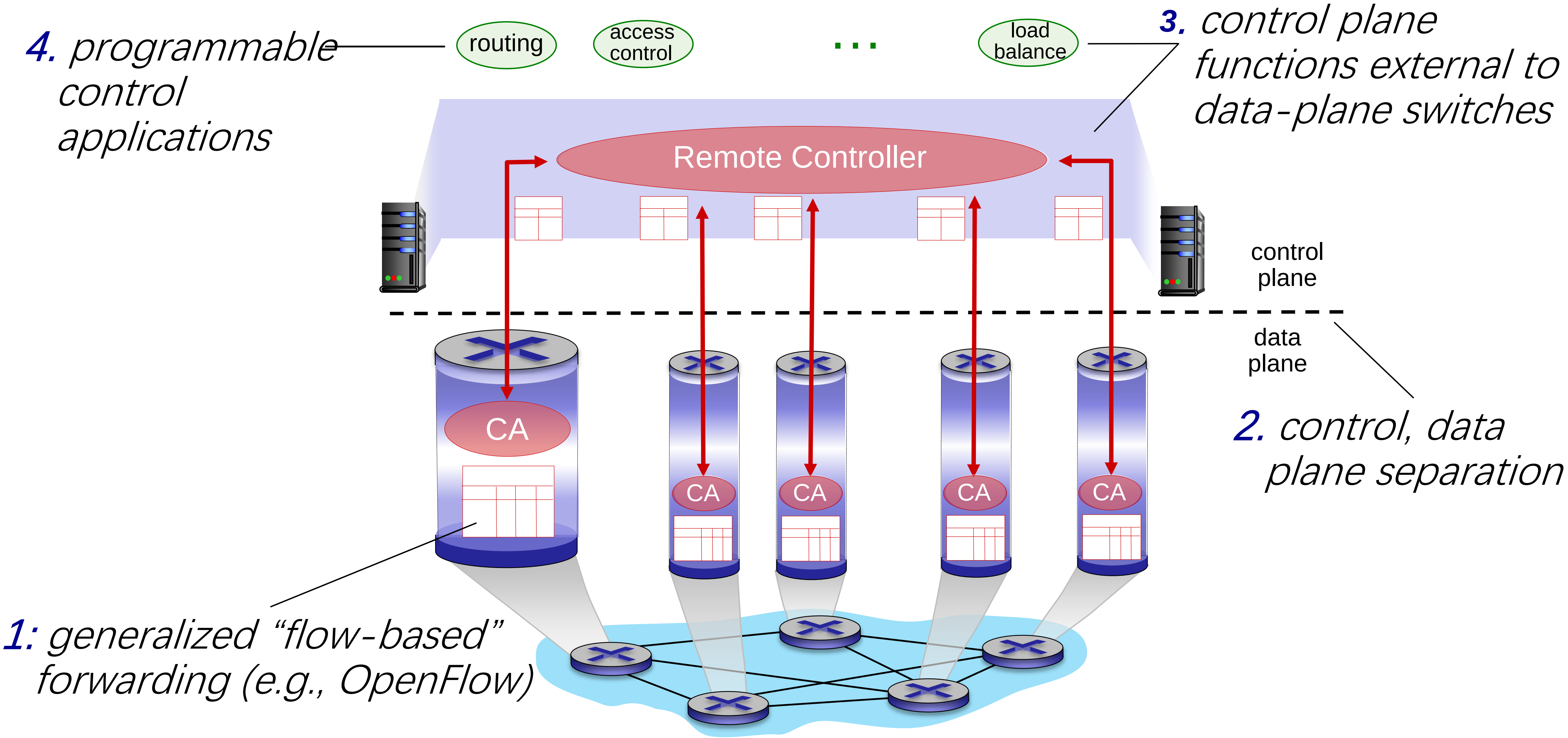
Example: Traffic Engineering

-Difficult with traditional routing



- w wants to route yellow and red traffic differently from w to z?
- Can’t do it

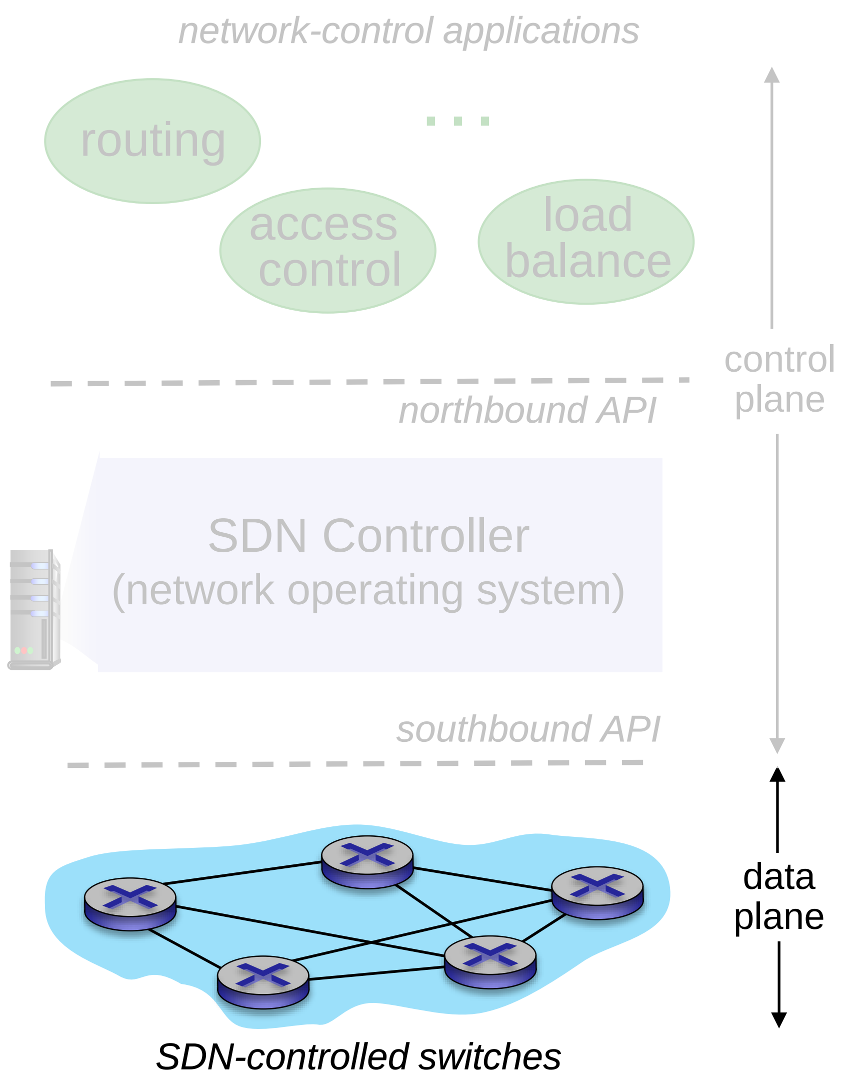
Software Defined Networking (SDN)



SDN Components

Data-plane switches:

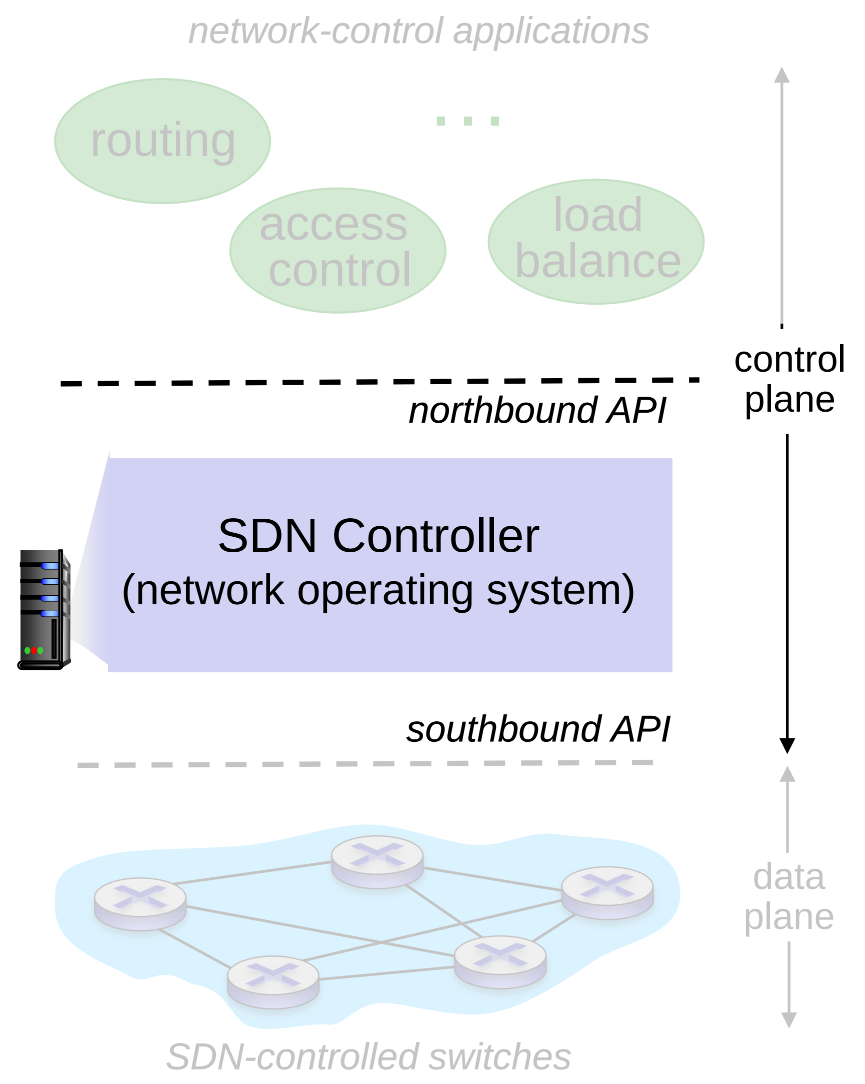
- Fast, simple, commodity switches implementing generalized data-plane forwarding in hardware
- Flow (forwarding) table computed, installed under controller supervision
- API for table-based switch control (e.g., OpenFlow)
  - defines what is controllable, what is not
- Protocol for communicating with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



SDN Components

SDN controller (network OS):

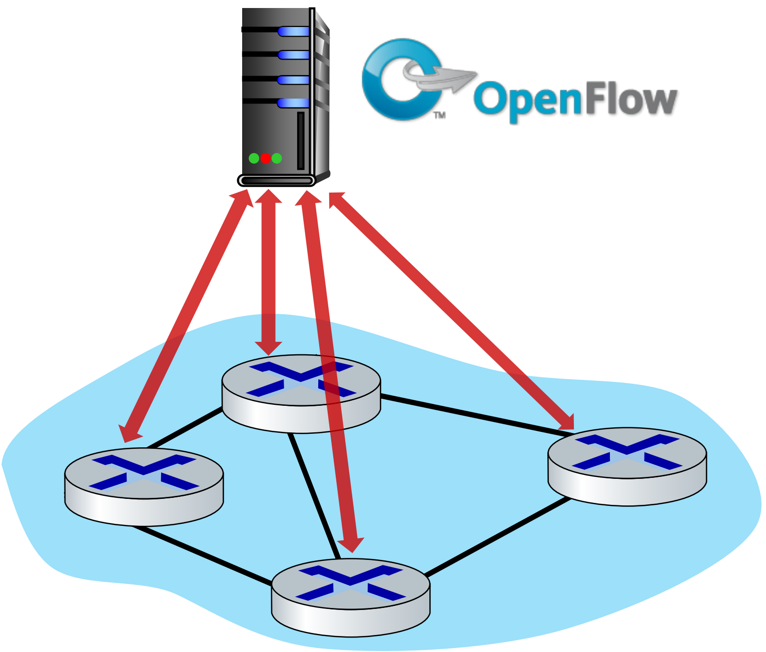
- Maintain network state information
- Interacts with network control applications "above" via northbound API
- Interacts with network switches "below" via southbound API
- Implemented as distributed system for performance, scalability, fault-tolerance, robustness



OpenFlow: Controller-to-switch Messages

Key controller-to-switch messages

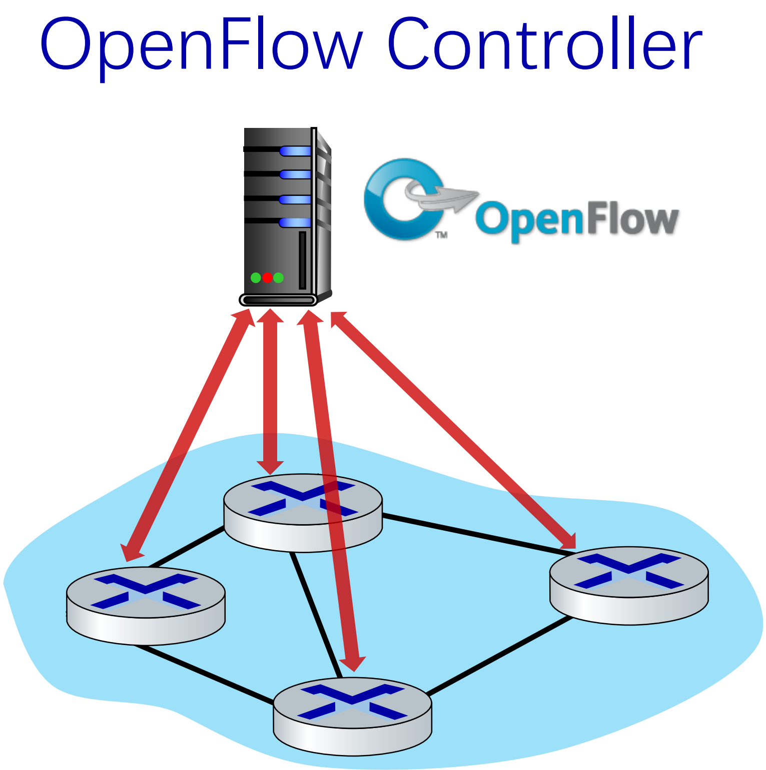
- *features:* controller queries switch features, switch replies
- *configure:* controller queries/sets switch configuration parameters
- *modify-state:* add, delete, modify flow entries in the OpenFlow tables
- *packet-out:* controller can send this packet out of specific switch port



OpenFlow: Switch-to-controller Messages

Key switch-to-controller messages

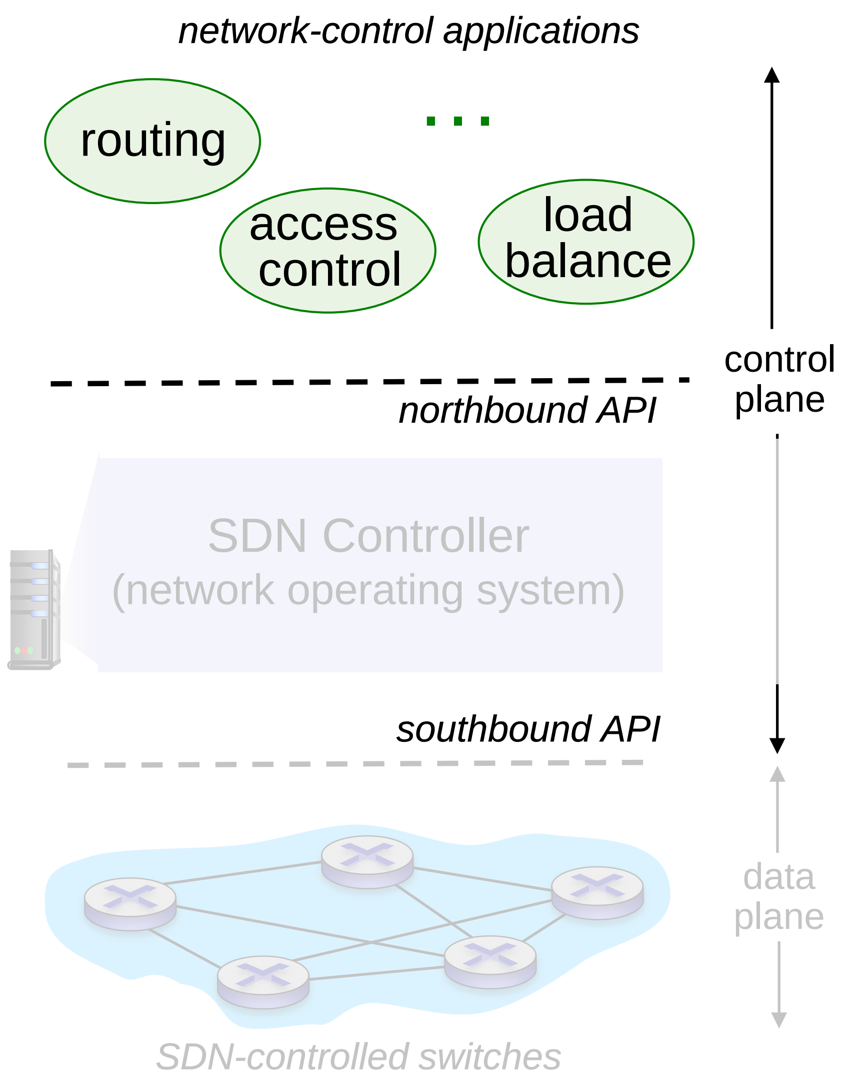
- *packet-in:* transfer packet (and its control) to controller. See packet-out message from controller
- *flow-removed:* flow table entry deleted at switch
- *port status:* inform controller of a change on a port.



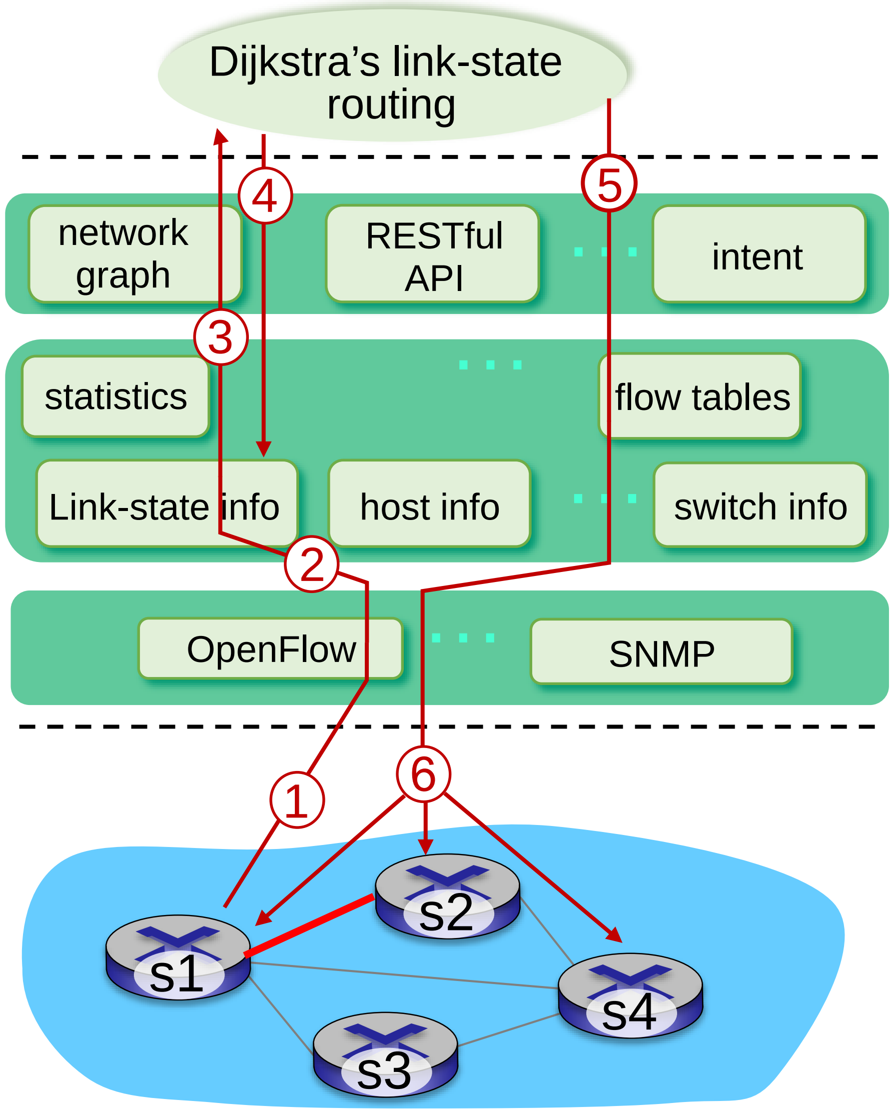
SDN Components

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



Control/Data Plane Interaction Example



1. S1, experiencing link failure uses 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

Reference

- Reference Textbook (Top-down Approach 7th) section 4.4 and section 5.5.
- Most slides are adapted from http://www-net.cs.umass.edu/kurose_ross/ppt.htm by Kurose Ross