Computer Networks
COL 334/672

Software Defined Networking

Tarun Mangla

Slides adapted from KR

Sem 1, 2024-25

slido

Please download and install the Slido app on all computers you use

Audience Q&A

ⓘ Start presenting to display the audience questions on this slide.

Timescales

	Data	Control	Management
Time-scale	Packet (nsec)	Event (10 msec to sec)	Human (min to hours)
Tasks	Forwarding, buffering, filtering, scheduling	Routing, circuit set-up	Analysis, configuration
Location	Line-card hardware	Router software	Humans or scripts

Fundamentally different timescales!

Traditional Networks: Per-router control plane

- **Control plane:** computes the path that packets will follow
- Routers talk amongst themselves but create a forwarding table individually

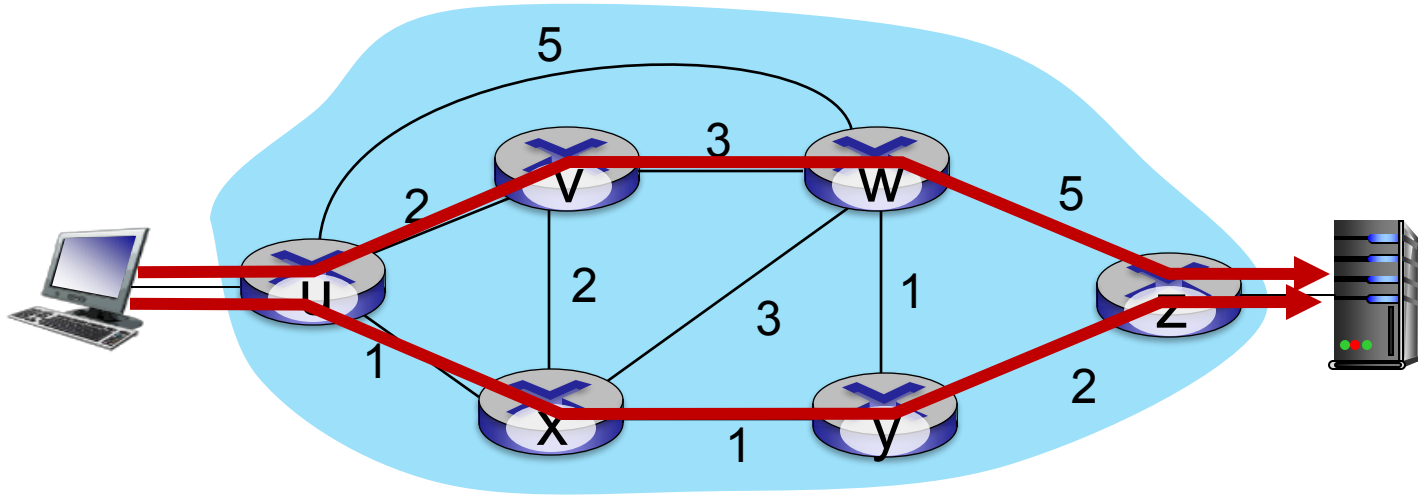
Traditional Networks: Per-router control plane

- **Control plane:** computes the path that packets will follow
- Routers talk amongst themselves but create a forwarding table individually

Limitations

- Traffic management is challenging with per-router control plane

Traffic engineering: difficult with traditional routing

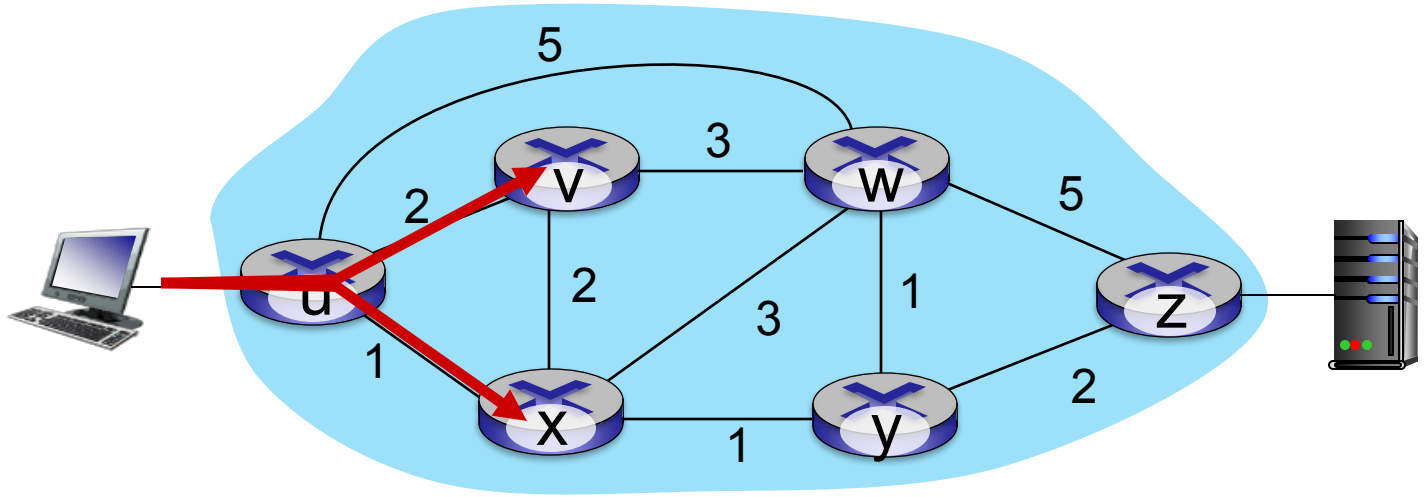


Q: what if network operator wants u-to-z traffic to flow along *uvwz*, rather than *uxyz*?

A: need to re-define link weights so traffic routing algorithm computes routes accordingly (or need a new routing algorithm)!

Indirect control: Changing weights instead of paths

Traffic engineering: difficult with 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 (or need a new routing algorithm)

Traditional Networks: Per-router control plane

- **Control plane:** computes the path that packets will follow
- Routers talk amongst themselves but create a forwarding table individually

Limitations

- Traffic management is challenging with per-router control plane
- Route convergence issues in case of link failure or weight changes

Difficult to manage networks!

Traditional Router Design

Monolithic, vertically integrated, sold by a single vendor

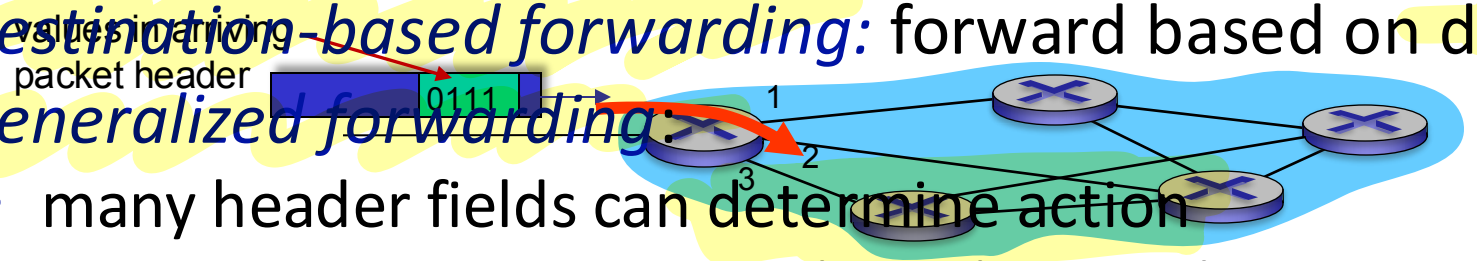
- Closed equipment
  - Software bundled with hardware
  - Vendor-specific interfaces
- Over specified
  - Slow protocol standardization
- Few people can innovate
  - Equipment vendors write the code

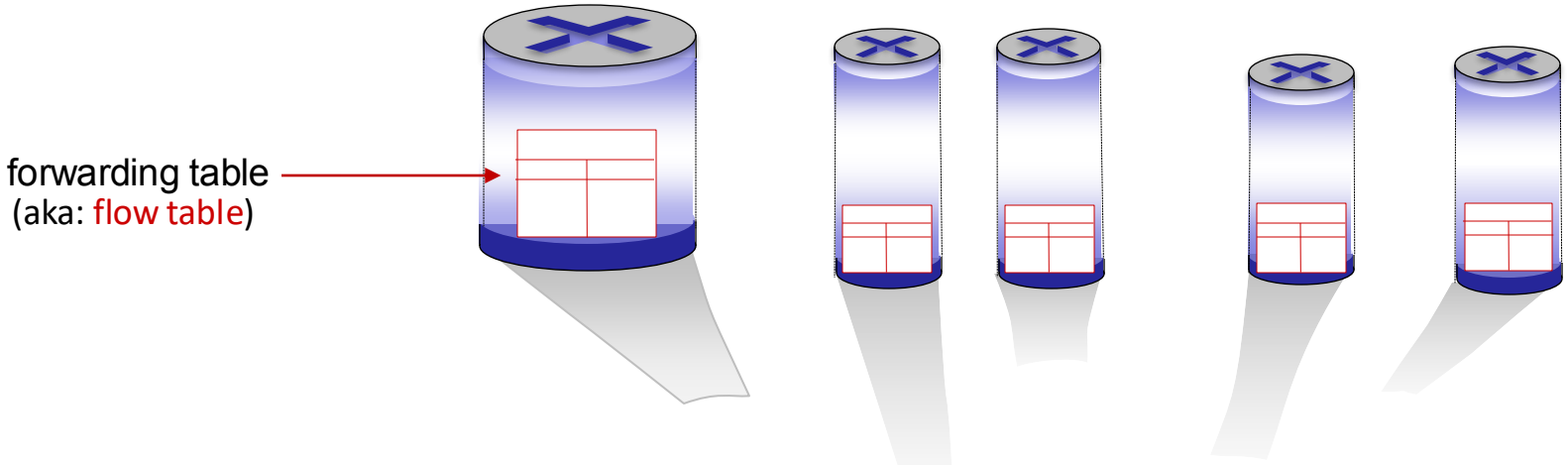
Slows down network innovation!

Towards a Generalized Forwarding Abstraction

Review: each router contains a forwarding table (aka: flow table)

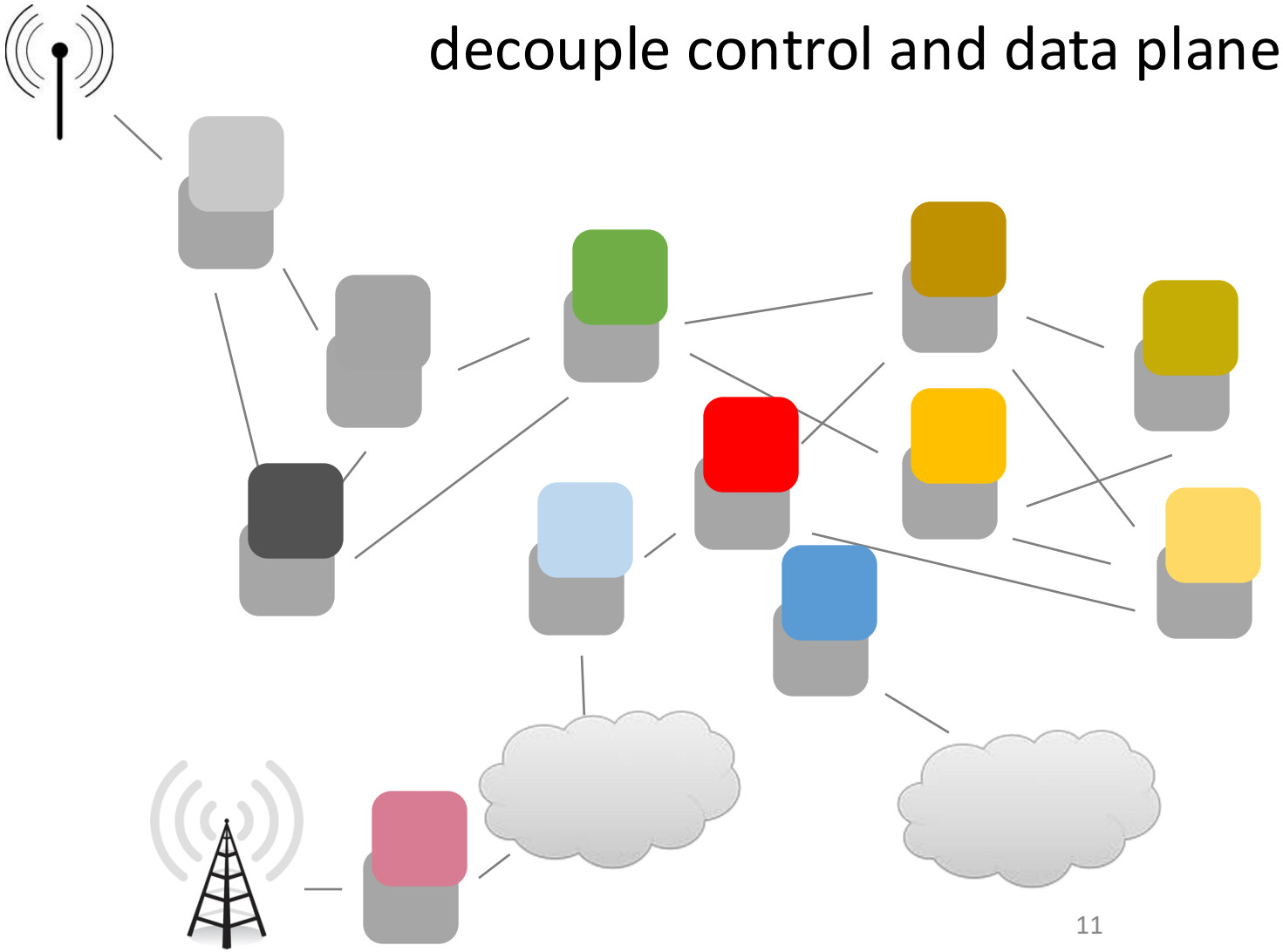
- "match plus action" abstraction: match bits in arriving packet, take action
  - *destination-based forwarding:* forward based on dest. IP address
  - *generalized forwarding:*
    - many header fields can determine action
    - many action possible: drop/copy/modify/log packet



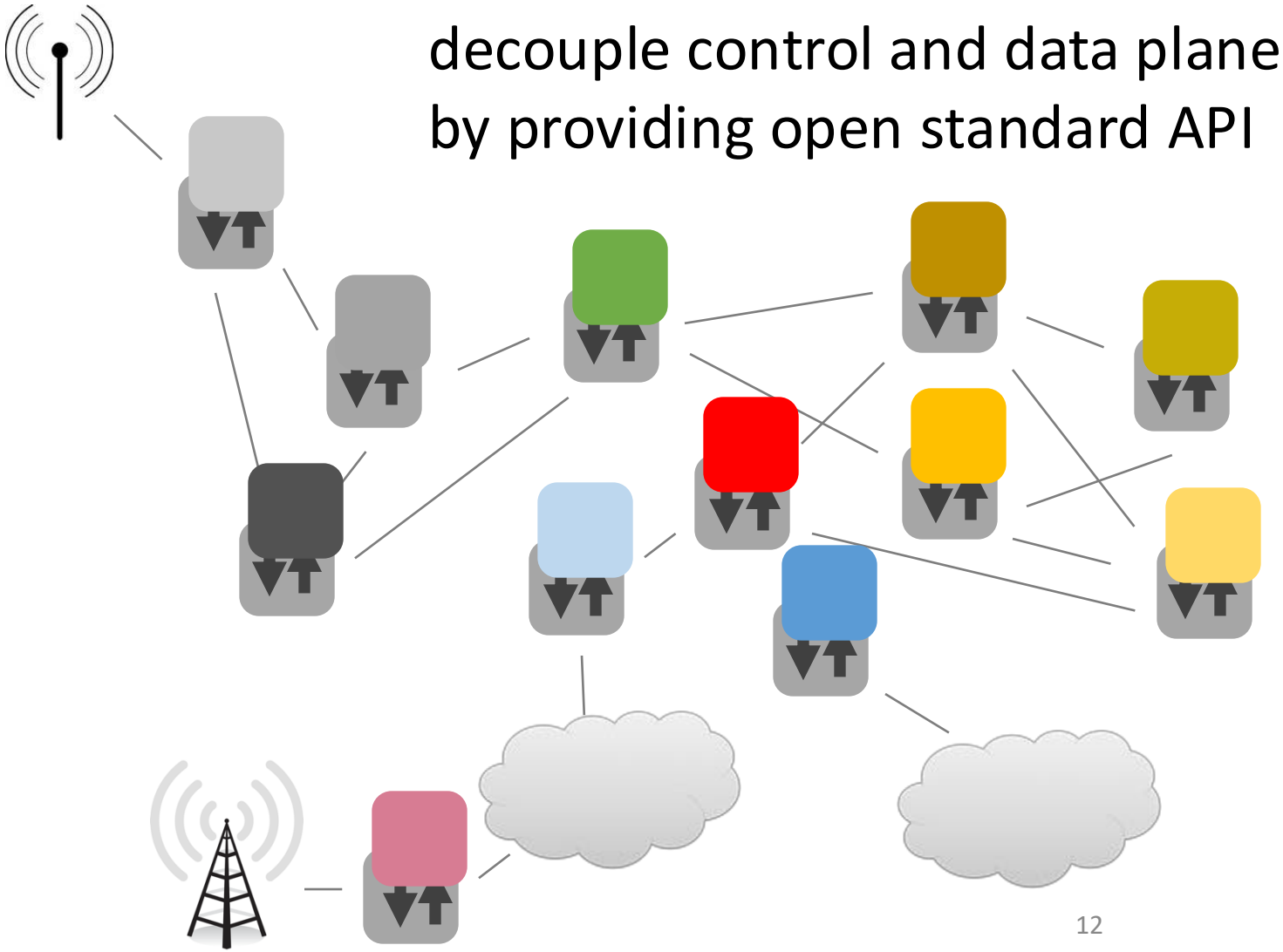


Chipset vendors started providing open APIs

Software Defined Networks

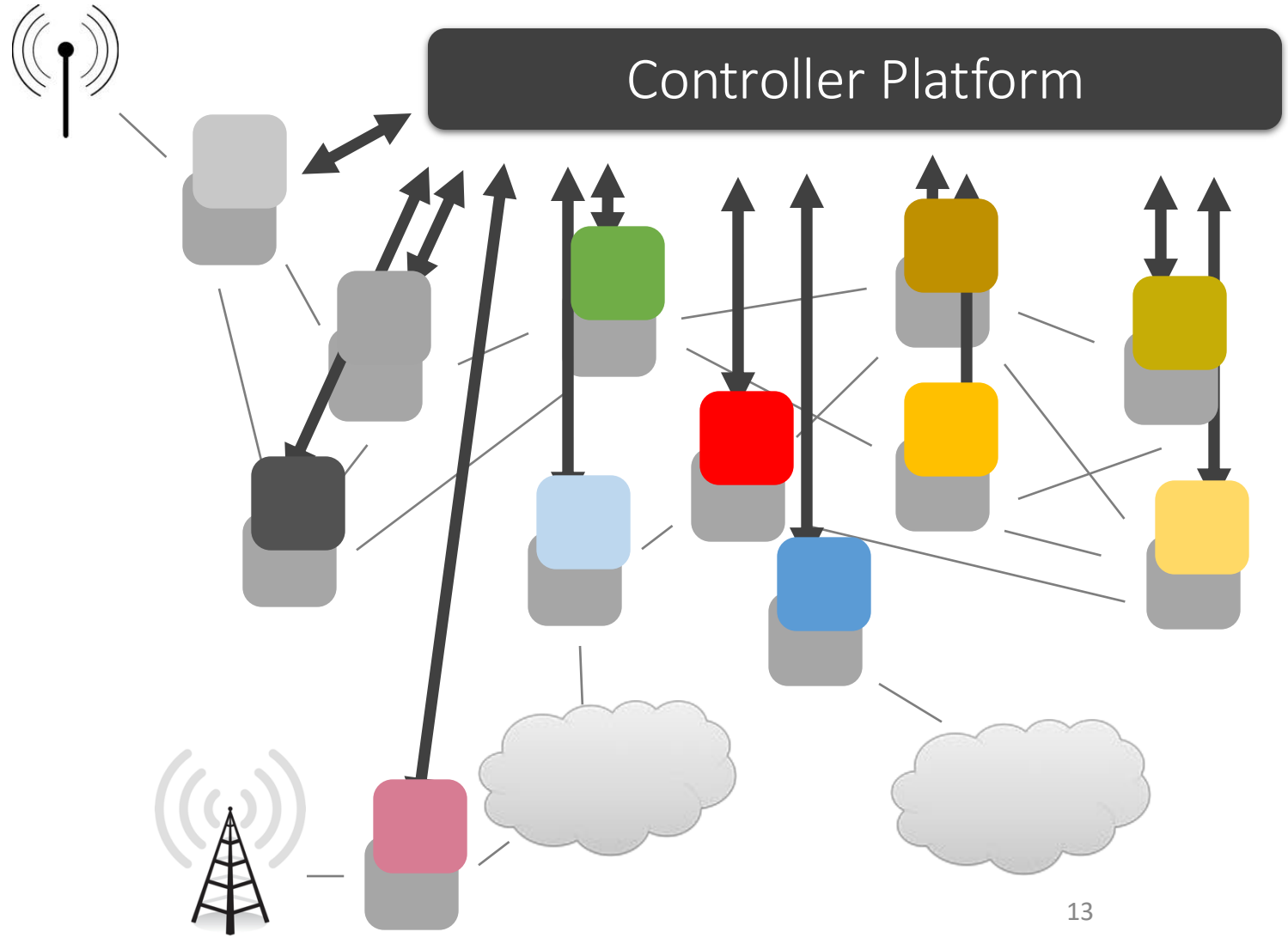


Software Defined Networks

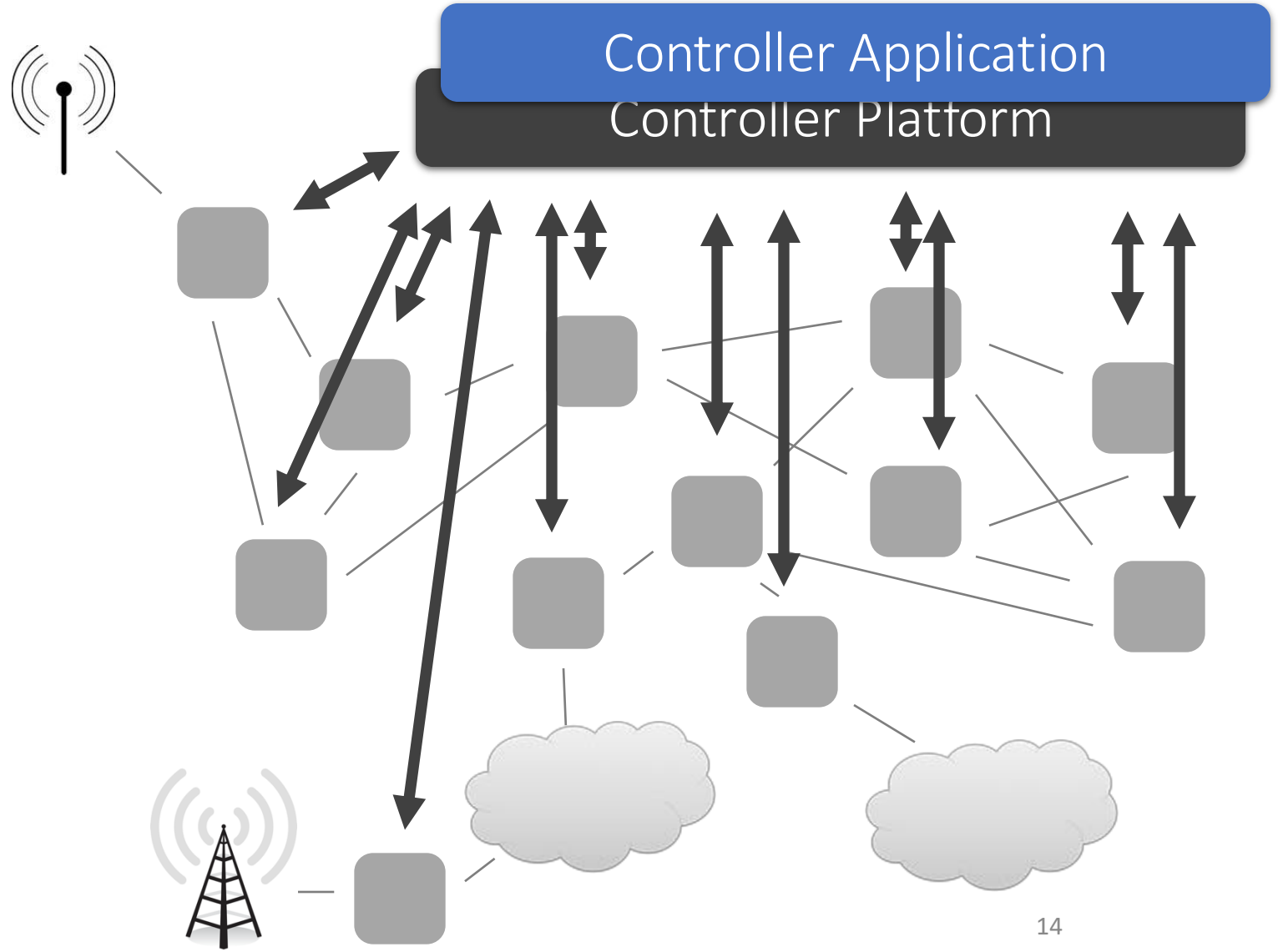


decouple control and data planes
by providing open standard API

(Logically) Centralized Controller

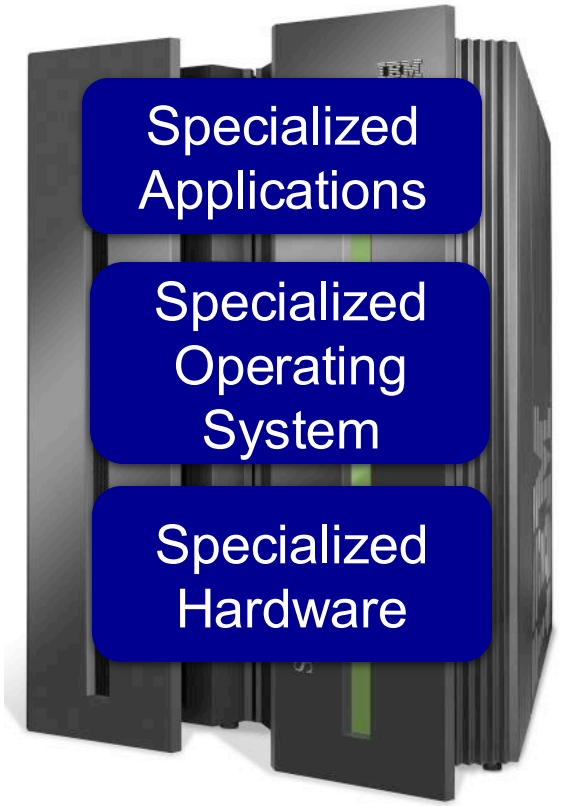


Protocols ➔ Applications

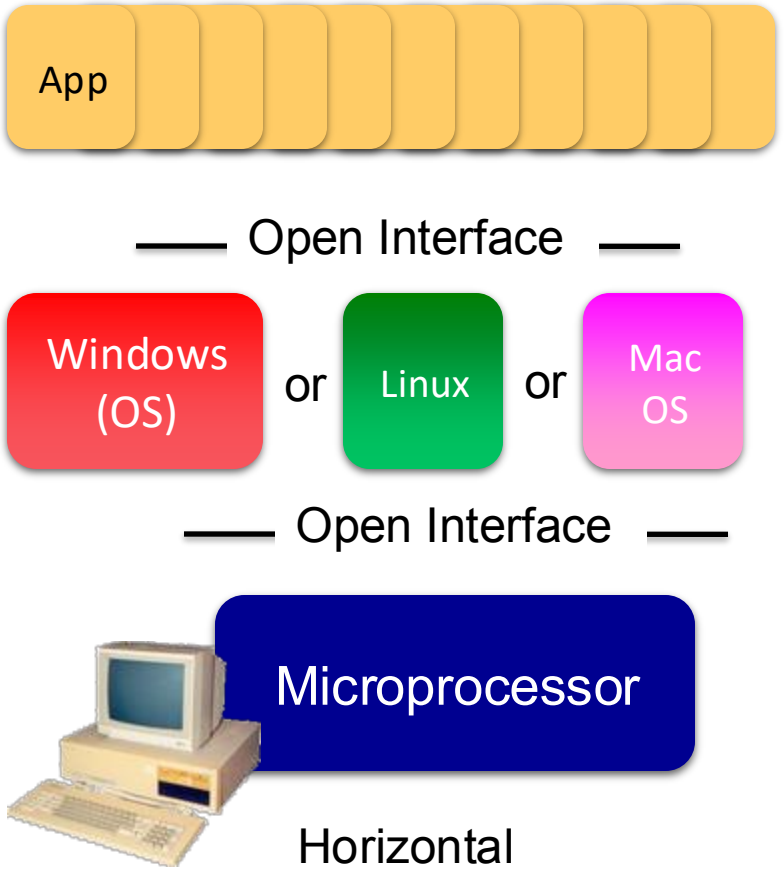


A Helpful Analogy: Computer Systems

(From Nick McKeown's talk "Making SDN Work" at the Open Networking Summit, April 2012)

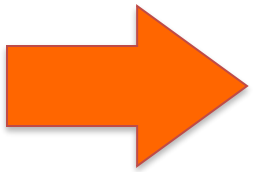


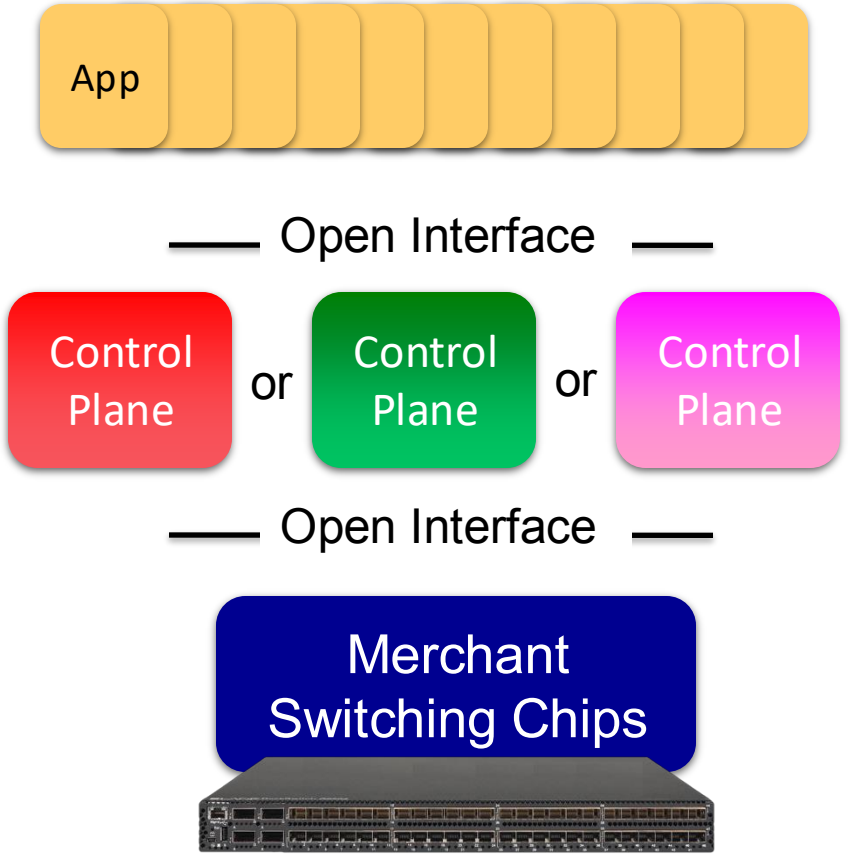
Vertically integrated
Closed, proprietary
Slow innovation
Small industry



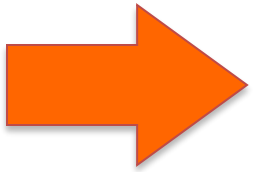
Horizontal
Open interfaces
Rapid innovation
Huge industry

Network Elements



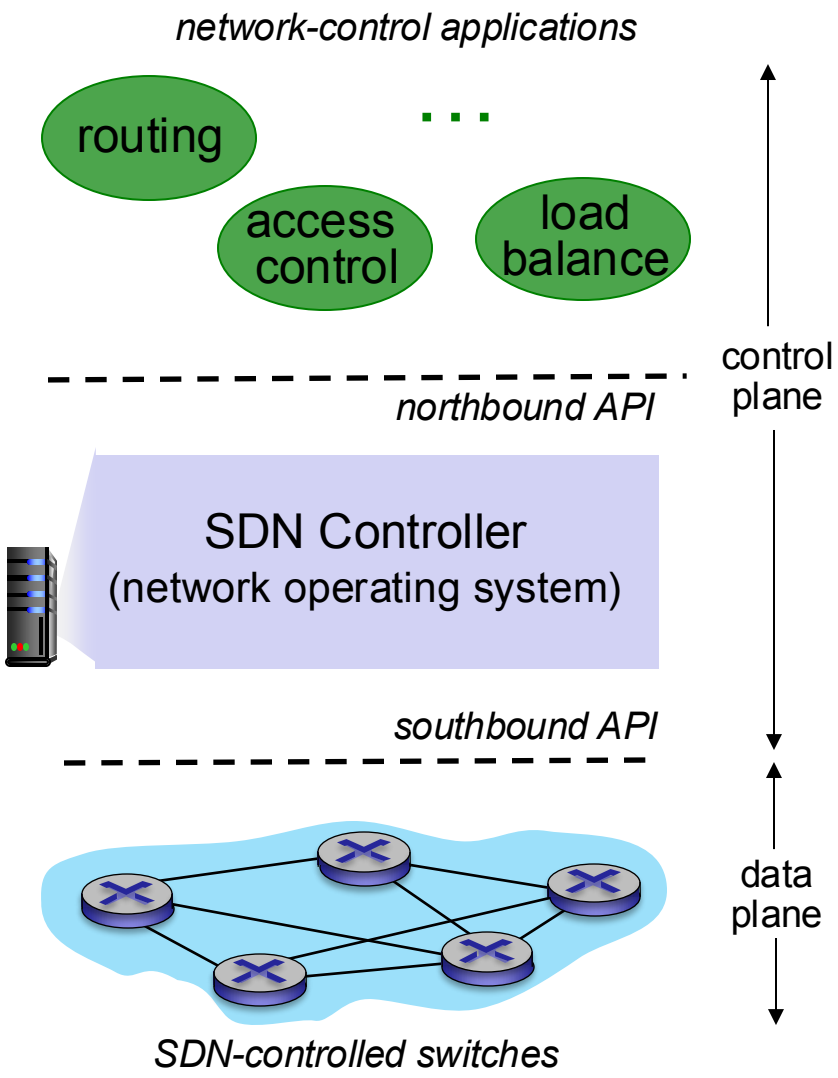


Vertically integrated
Closed, proprietary
Slow innovation



Horizontal
Open interfaces
Rapid innovation

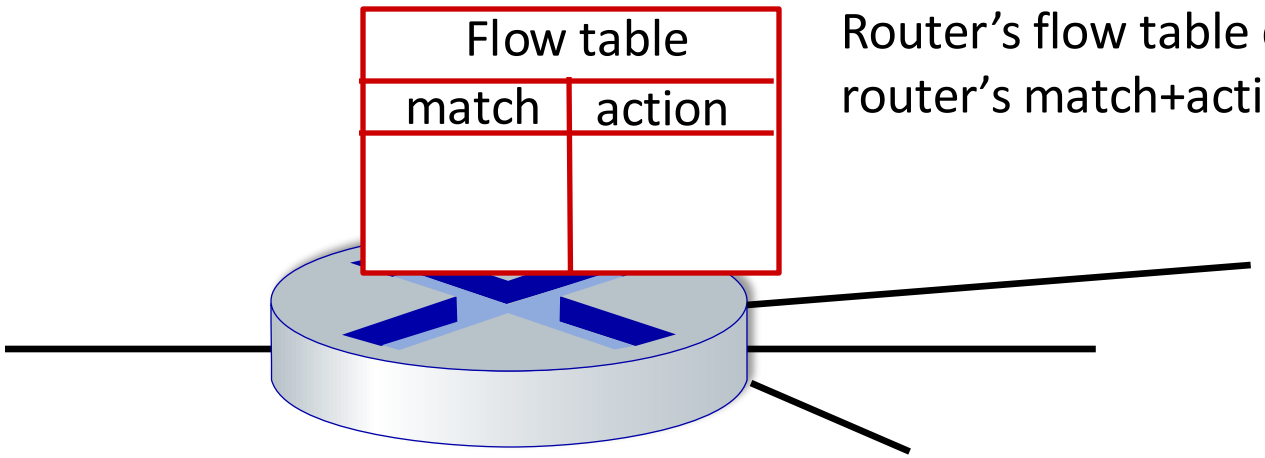
Software-defined Network



OpenFlow: Most Popular Southbound API

Flow table abstraction

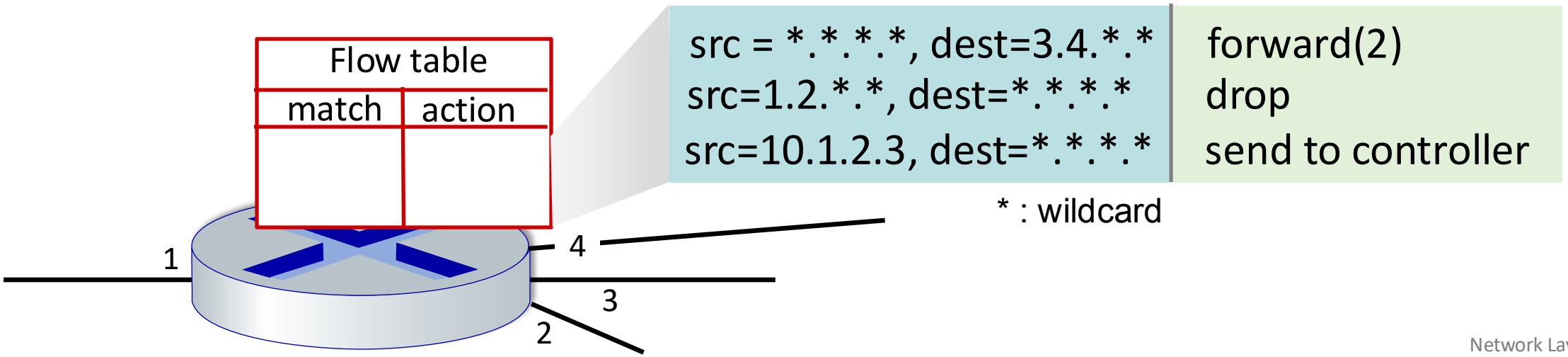
- flow: defined by header field values (in link-, network-, transport-layer fields)
- generalized forwarding: simple packet-handling rules
  - match: pattern values in packet header fields
  - actions: for matched packet: drop, forward, modify, matched packet or send matched packet to controller
  -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



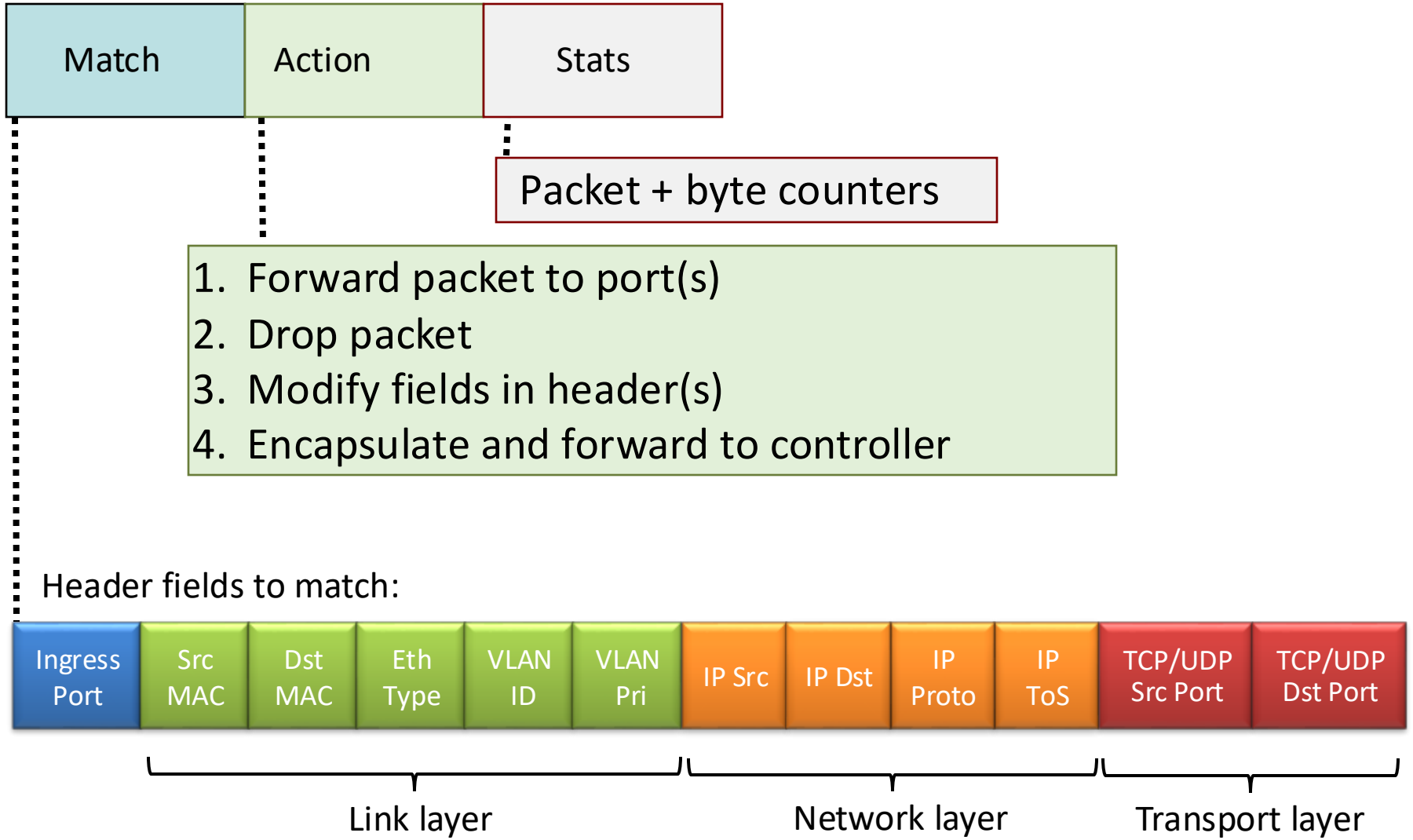
Router's flow table define router's match+action rules

Flow table abstraction

- flow: defined by header 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



OpenFlow: flow table entries



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

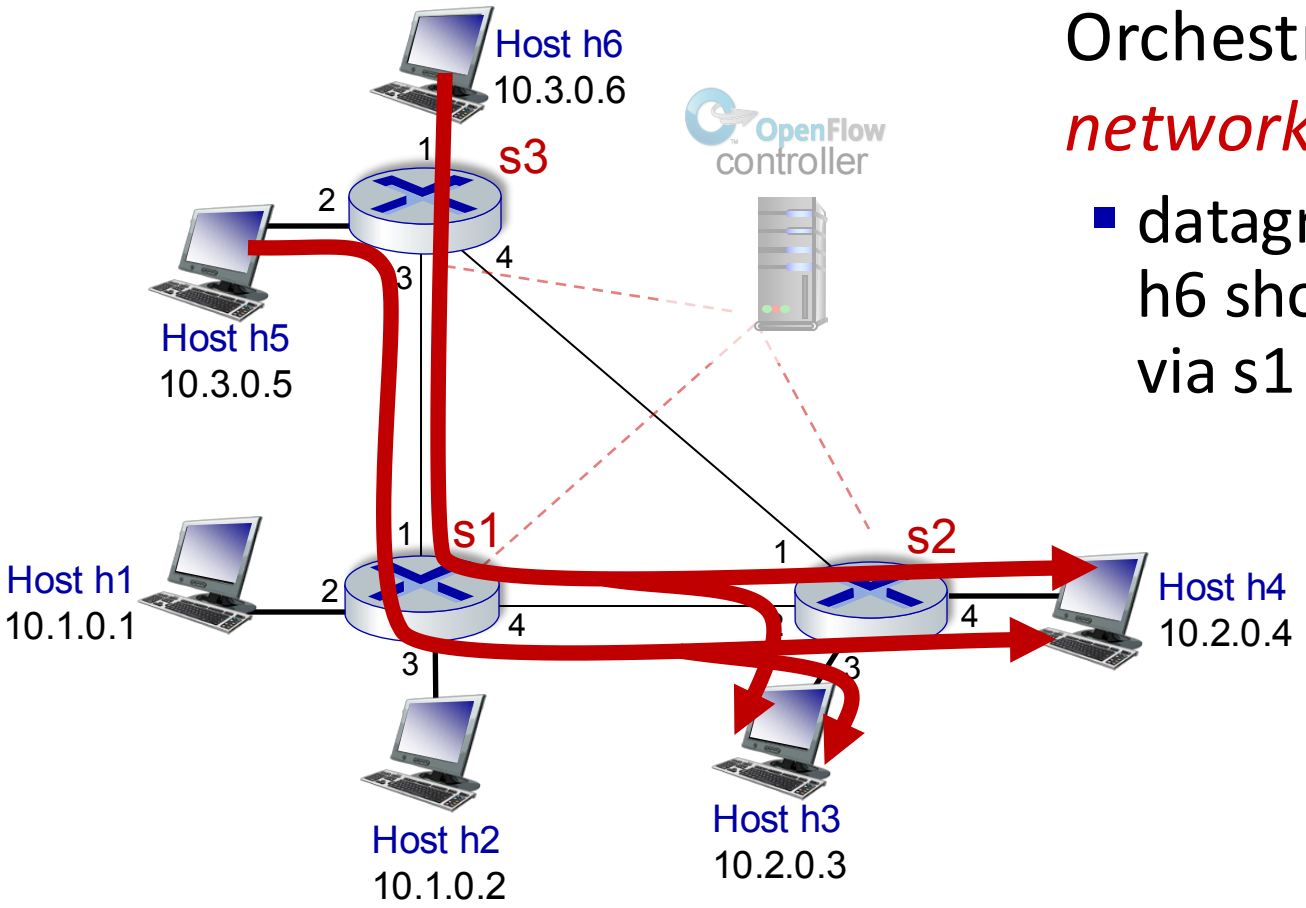
Firewall

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

NAT

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

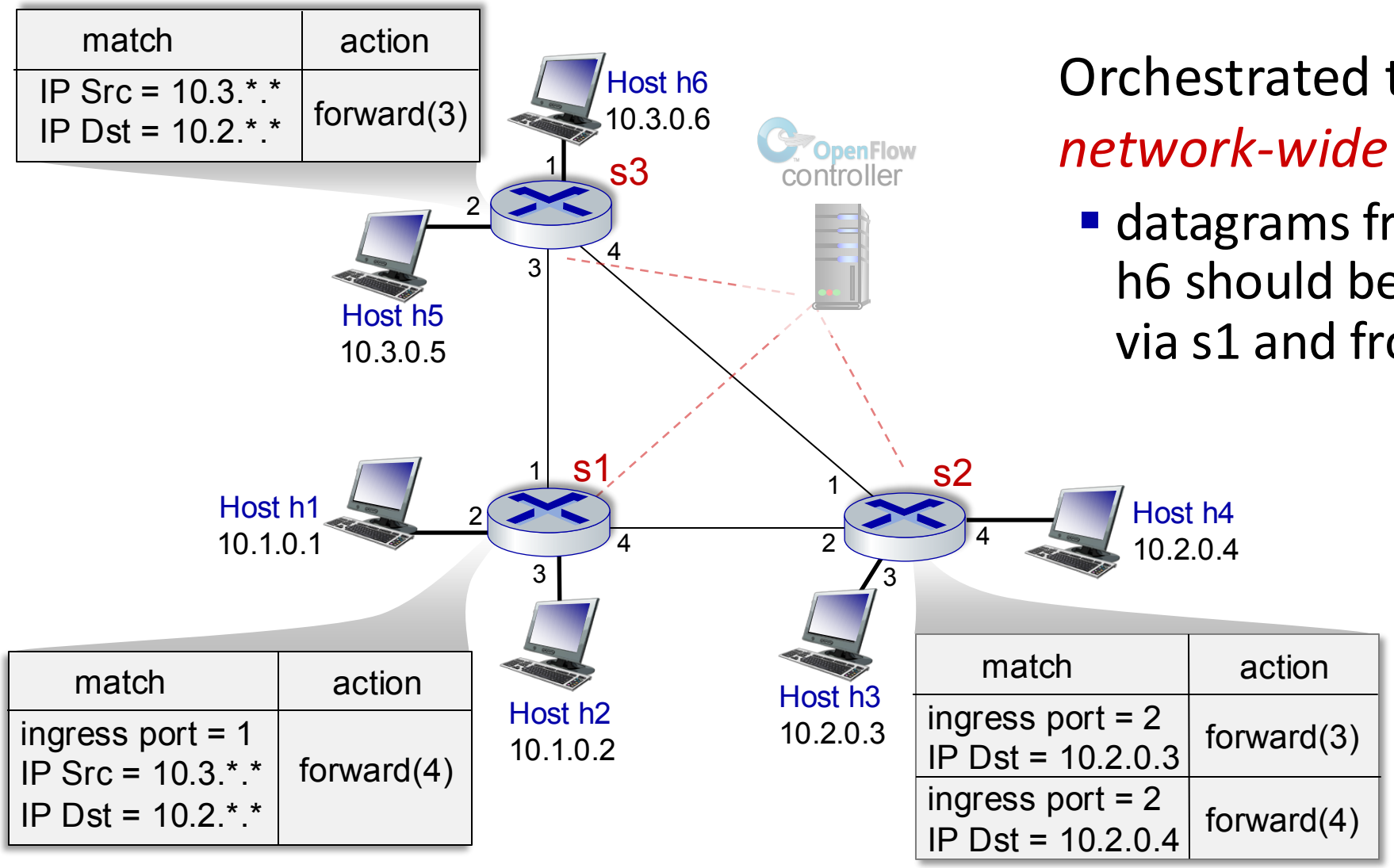
OpenFlow example



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

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

OpenFlow example



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

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