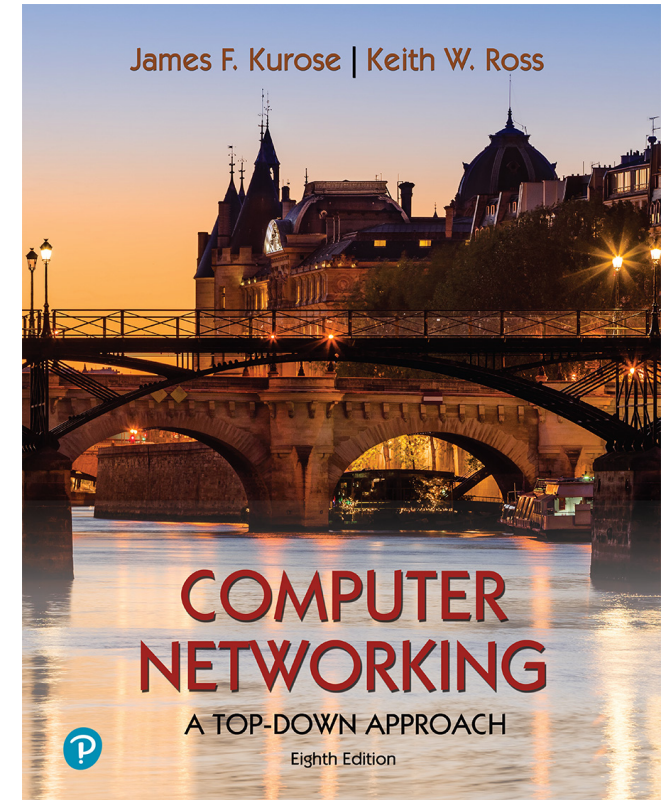


Chapter 4

Network Layer: Data Plane



Computer Networking: A Top-Down Approach

8th edition

Jim Kurose, Keith Ross
Pearson, 2020

Acknowledgement: Based on the textbook's website:
https://gaia.cs.umass.edu/kurose_ross/index.php

Network layer: “data plane” roadmap

- Network layer: overview
 - data plane
 - control plane
- What’s inside a router
 - input ports, switching, output ports
 - buffer management, scheduling
- IP: the Internet Protocol
 - datagram format
 - addressing
 - network address translation
 - IPv6
- Generalized Forwarding, SDN
 - Match+action
 - OpenFlow: match+action in action
- Middleboxes



Generalized forwarding: match plus action

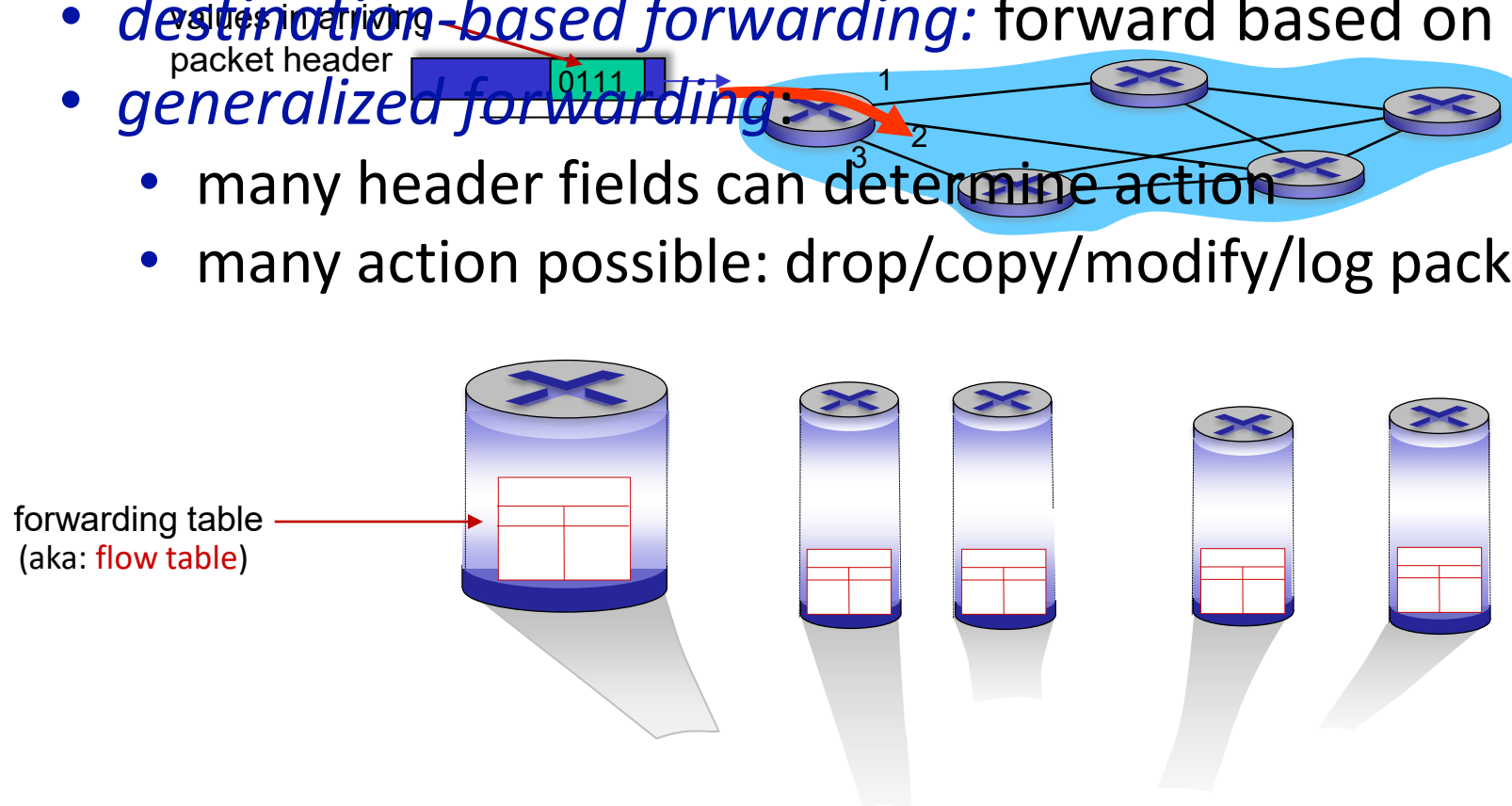
Review: each router contains a **forwarding table** (aka: **flow table**)

- “**match plus action**” abstraction: match bits in arriving packet, take action

- *destination-based forwarding*: forward based on dest. IP address

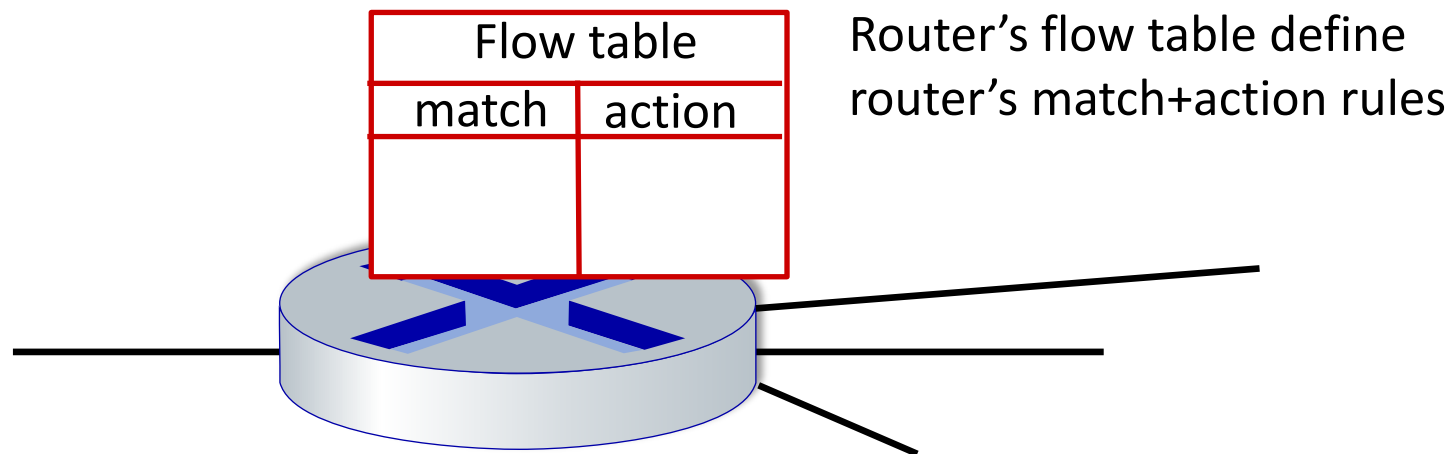
- *generalized forwarding*:

- many header fields can determine action
- many action possible: drop/copy/modify/log packet



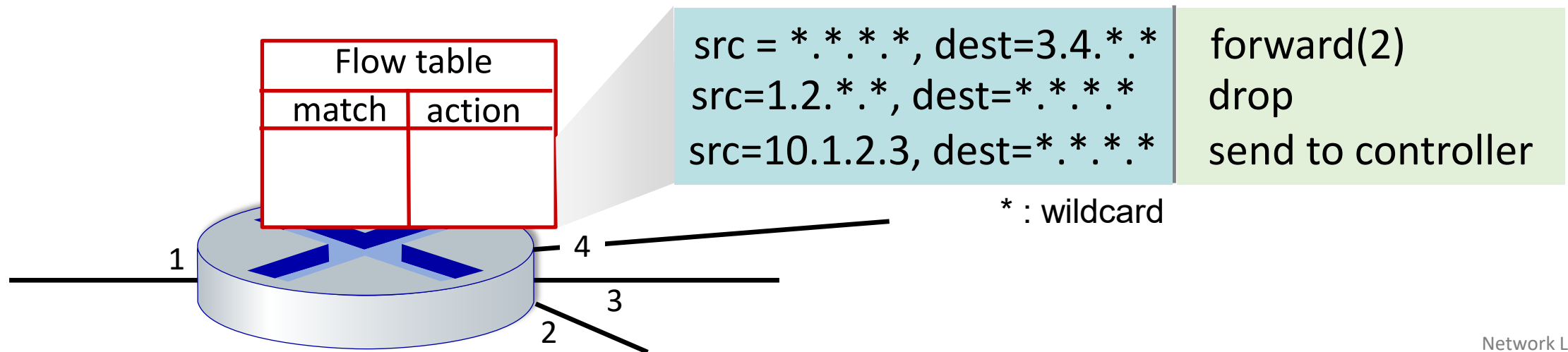
Flow table abstraction

- **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

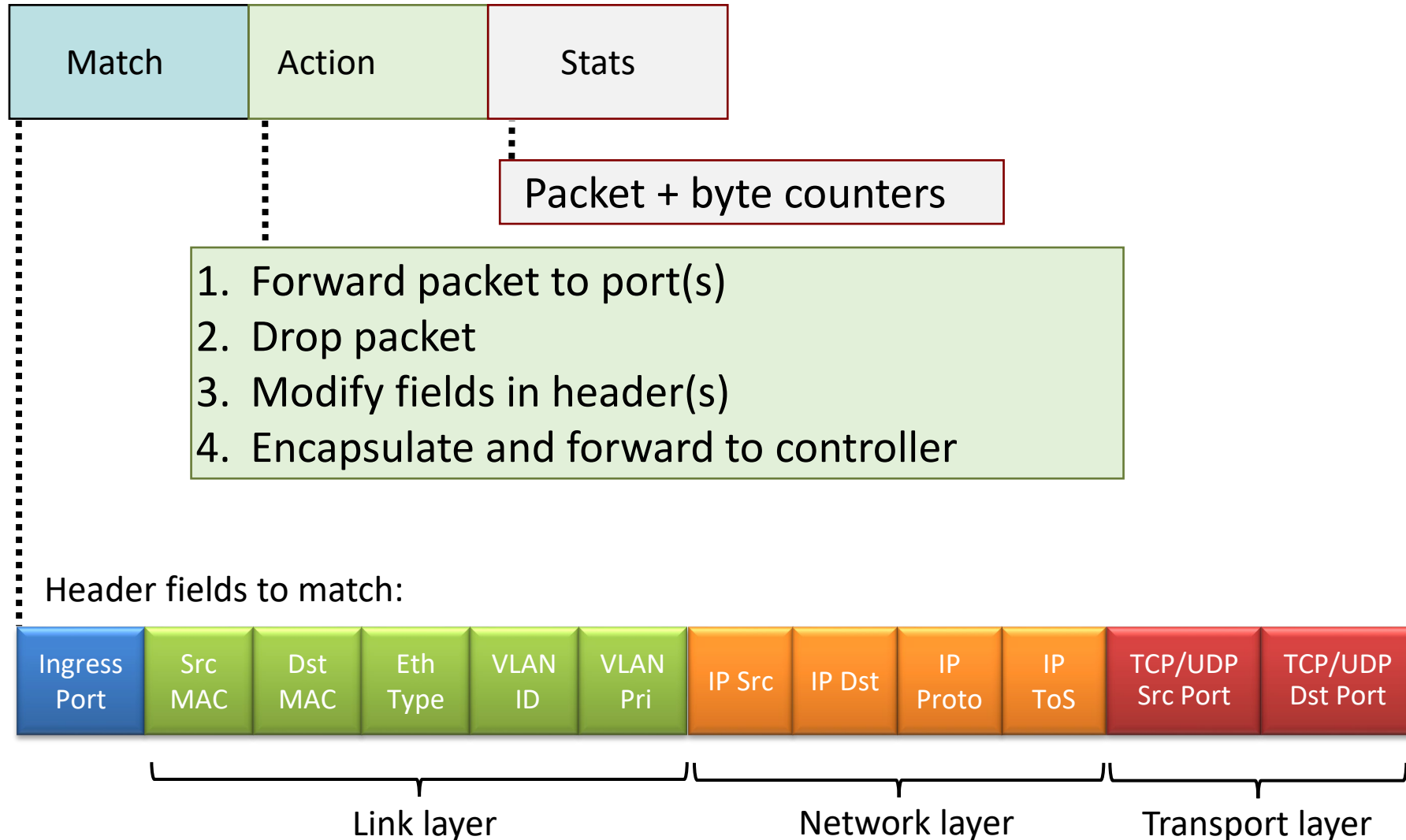


Flow table abstraction

- **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

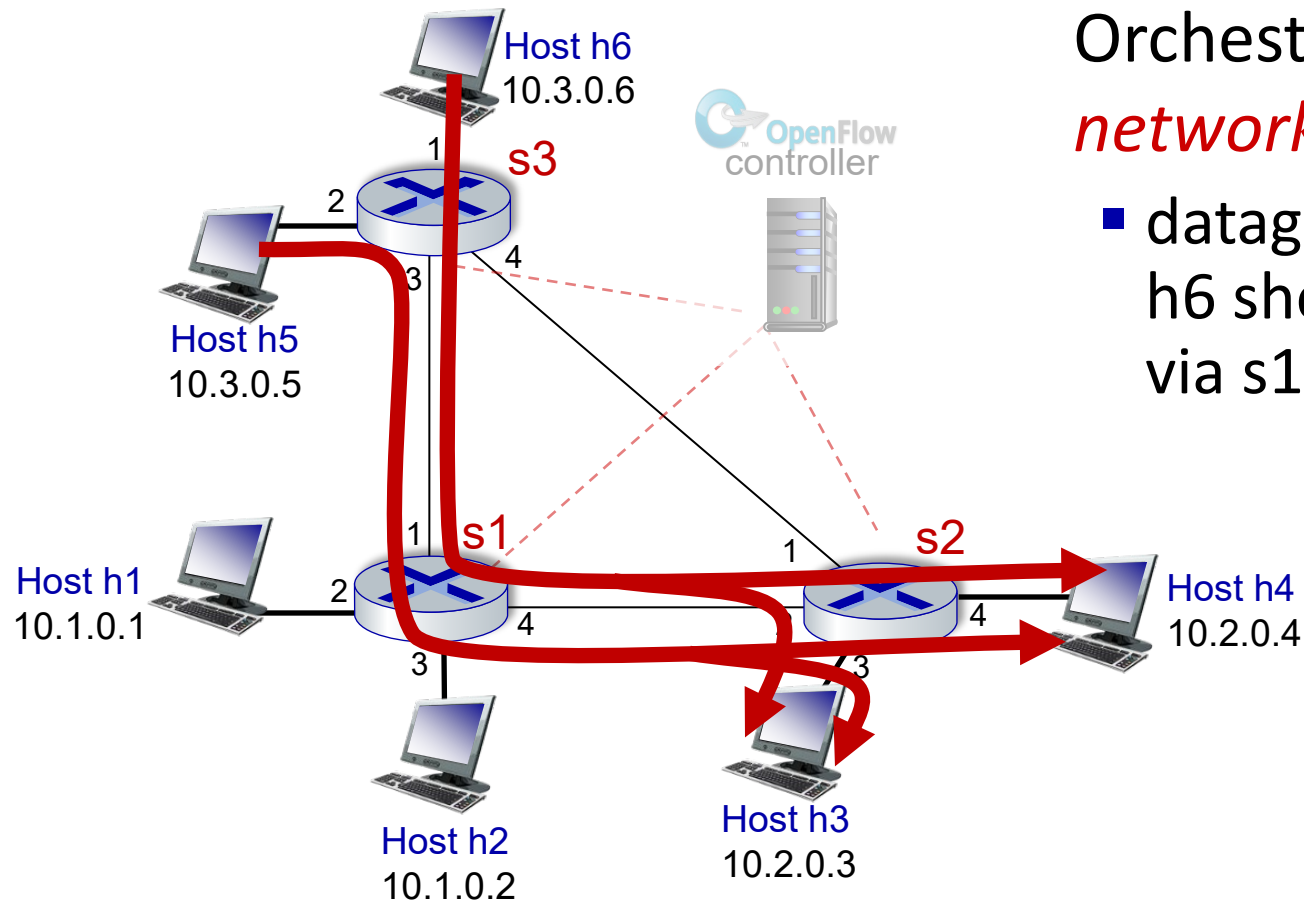
Firewall

- *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

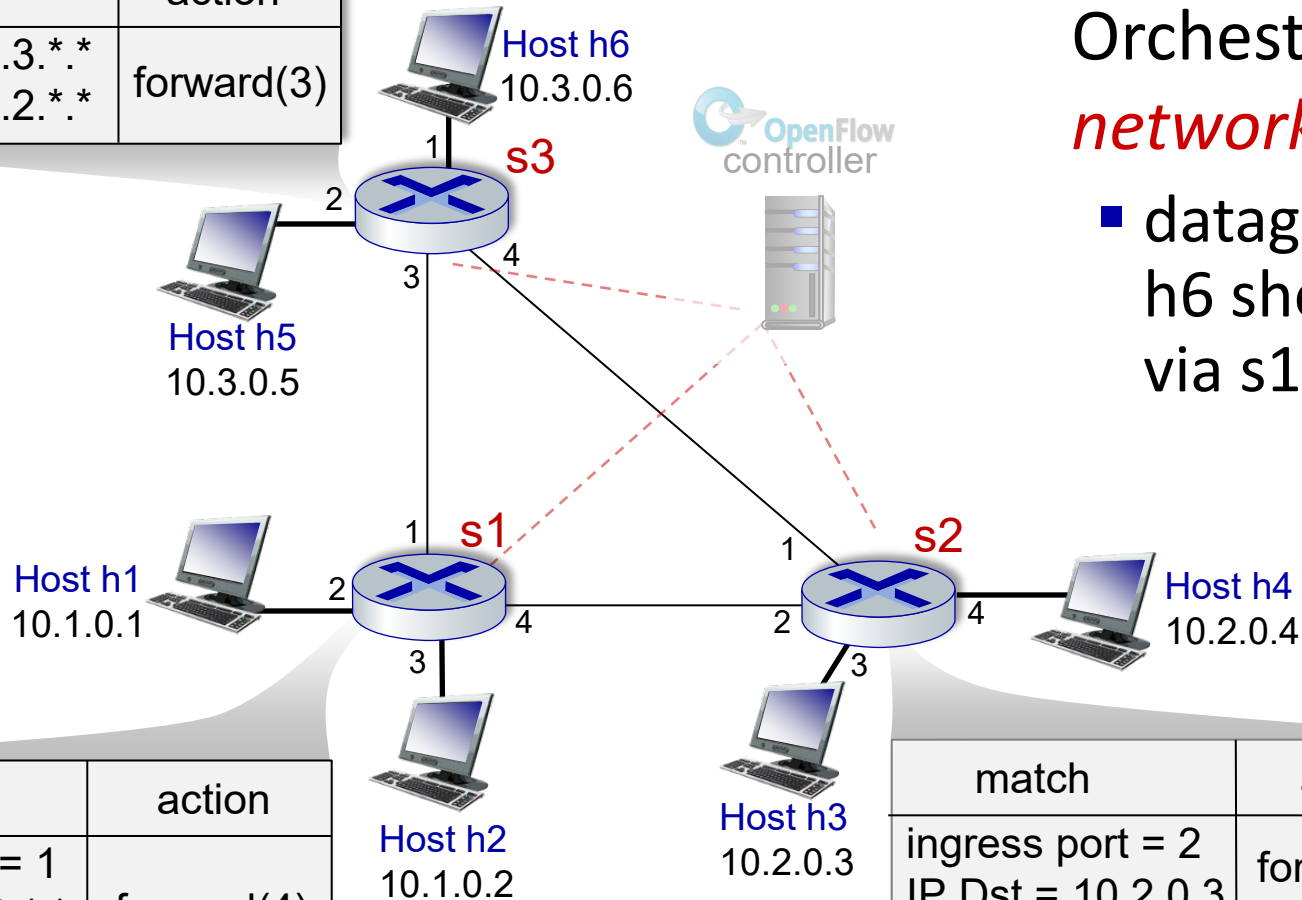


Orchestrated tables can create *network-wide* behavior, e.g.,:

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

OpenFlow example

| 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) |

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

Generalized forwarding: summary

- “match plus action” abstraction: match bits in arriving packet header(s) in any layers, take action
 - matching over many fields (link-, network-, transport-layer)
 - local actions: drop, forward, modify, or send matched packet to controller
 - “program” *network-wide* behaviors
- simple form of “network programmability”
 - programmable, per-packet “processing”
 - *historical roots*: active networking
 - *today*: more generalized programming: P4 (see p4.org).