Chapter 4
Network Layer: Data Plane

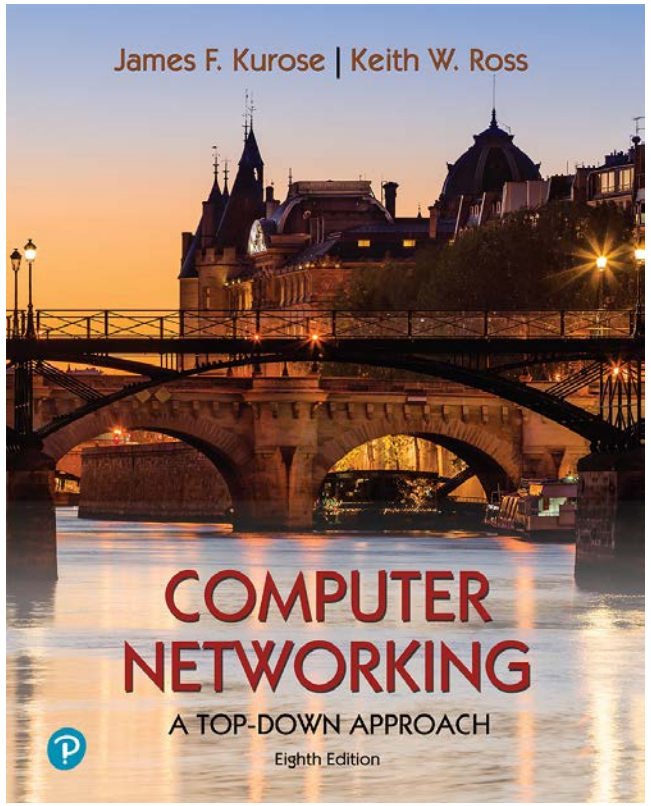
A note on the use of these PowerPoint slides:

We're making these slides freely available to all (faculty, students, readers). They're in PowerPoint form so you see the animations; and can add, modify, and delete slides (including this one) and slide content to suit your needs. They obviously represent a *lot* of work on our part. In return for use, we only ask the following:

- If you use these slides (e.g., in a class) that you mention their source (after all, we'd like people to use our book!)
- If you post any slides on a www site, that you note that they are adapted from (or perhaps identical to) our slides, and note our copyright of this material.

For a revision history, see the slide note for this page.

Thanks and enjoy! JFK/KWR



Computer Networking: A Top-Down Approach

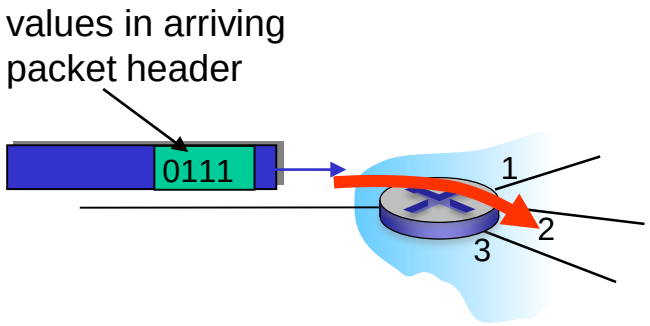
8th edition
Jim Kurose, Keith Ross
Pearson, 2020

--

Network layer: data plane, control plane

Data plane:

- *local*, per-router function
- determines how datagram arriving on router input port is forwarded to router output port



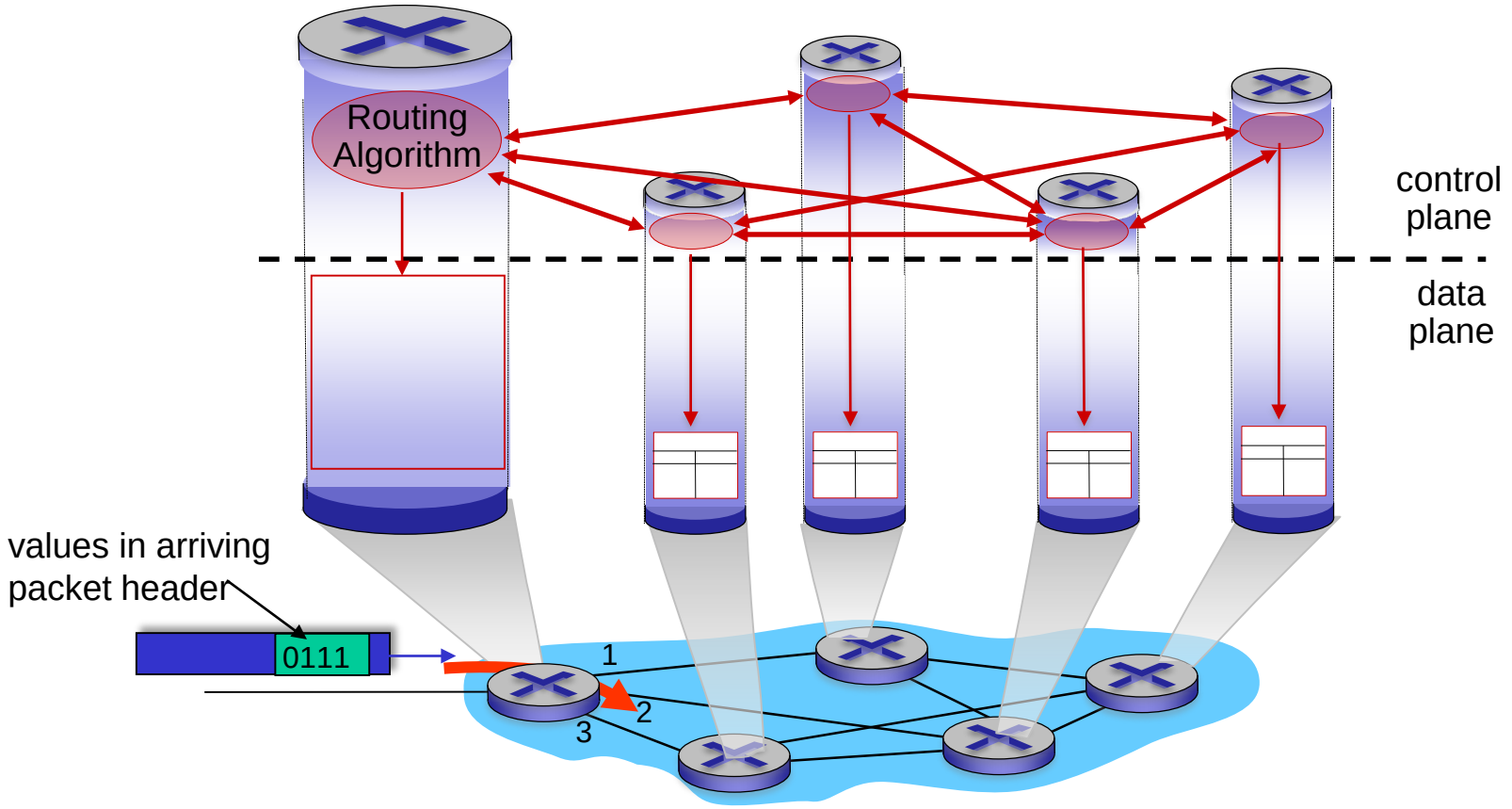
Control plane

- *network-wide* logic
- determines how datagram is routed among routers along end-end path from source host to destination host
- two control-plane approaches:
  - *traditional routing algorithms:* implemented in routers
  - *software-defined networking (SDN)*: implemented in (remote) servers

---

Per-router control plane

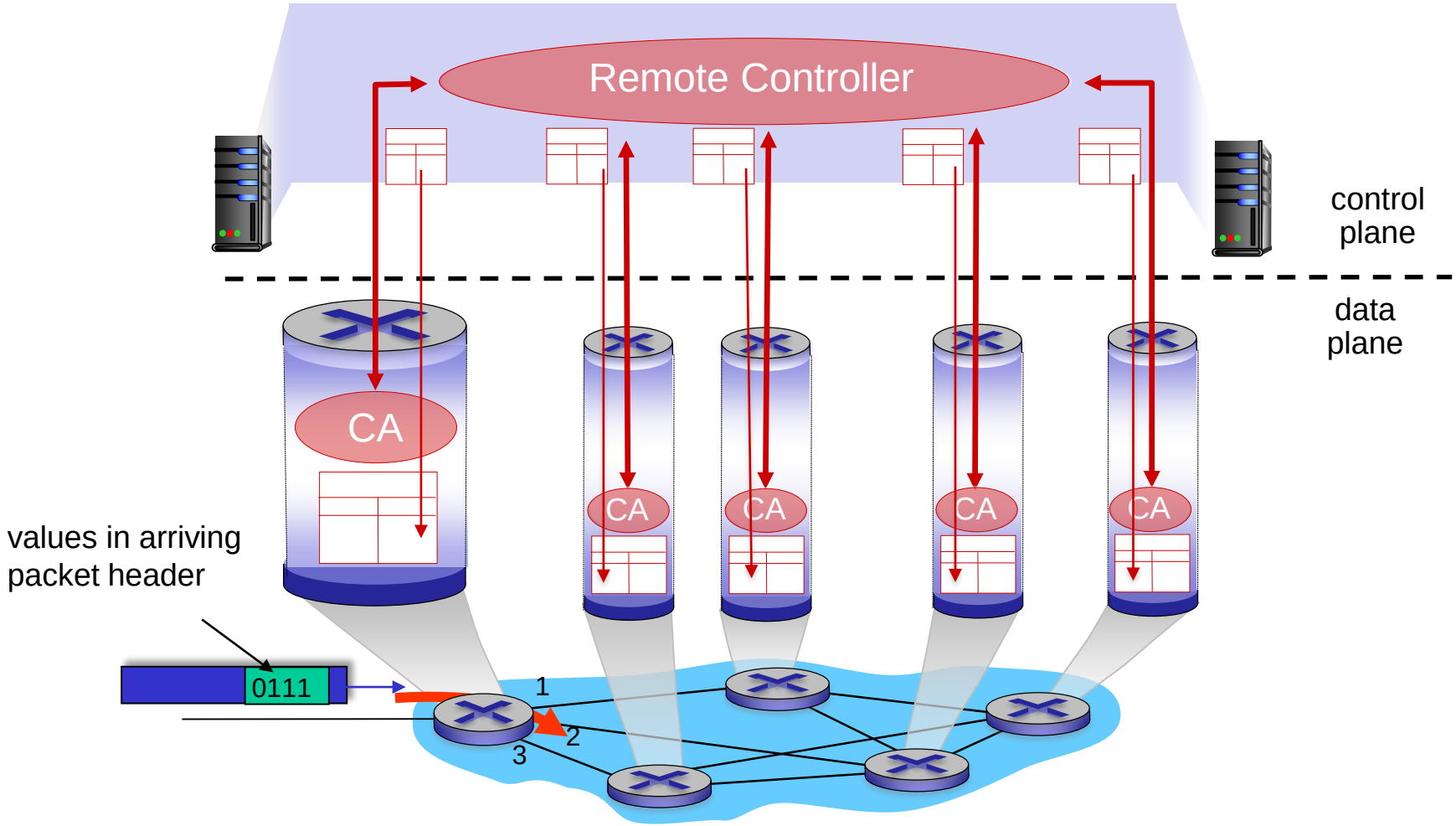
Individual routing algorithm components *in each and every router* interact in the control plane



--

Software-Defined Networking (SDN) control plane

Remote controller computes, installs forwarding tables in routers

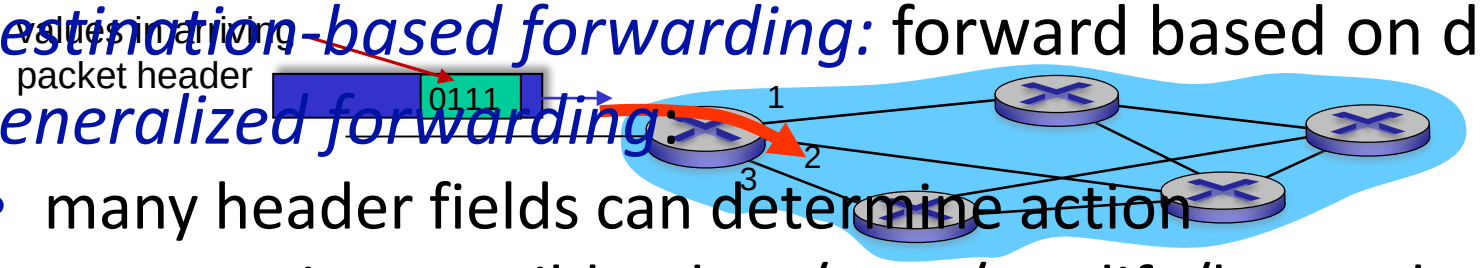


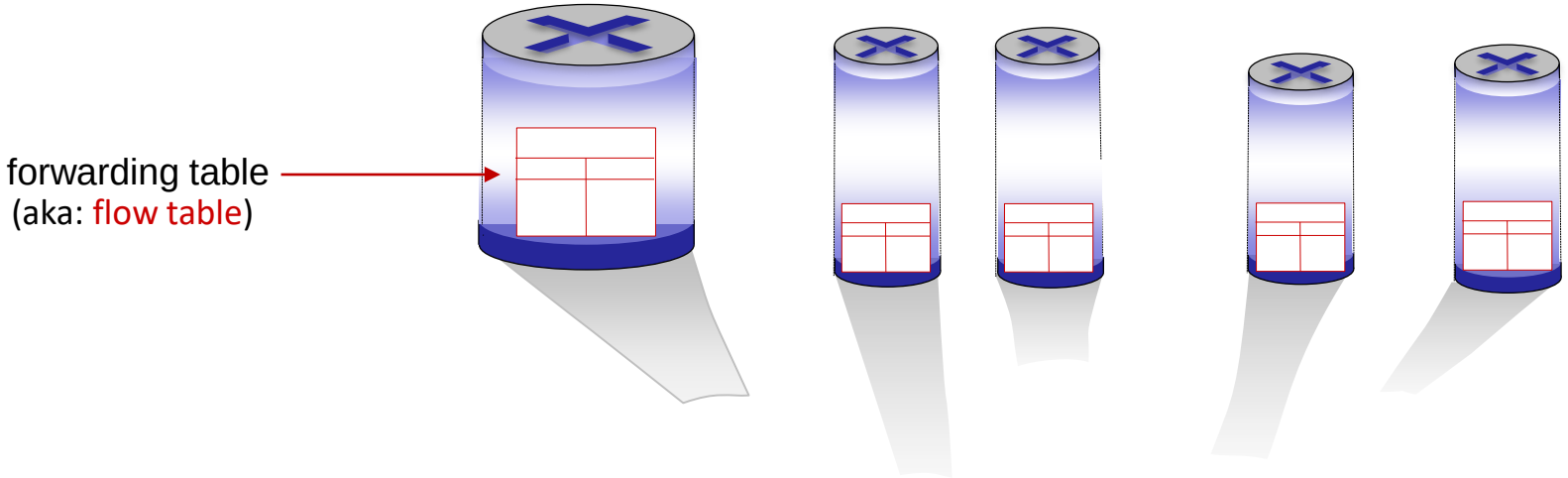
--

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

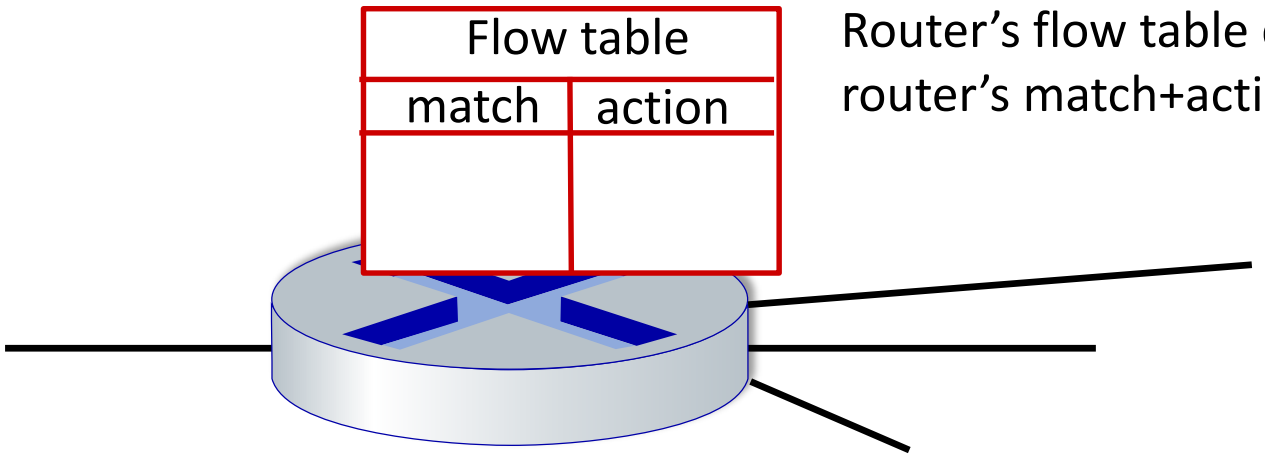




-
توی SDN فورواردینگ دیگه دستینیشن بیس نیست بلکه generalized forwarding است
generalized forwarding : لازمش اینه که شکل فورواردینگ تیبل عوض بشه - توی فورواردینگ تیبل دستینیشن بیس یک ستون داشتیم برای ادرس مقصد و یک ستون داشتیم برای پورت خروجی
اما در SDN به جای فورواردینگ تیبل یک تیبلی به اسم flow table داریم که این flow tabel چند ستون داره که بخشی از ستون ها فیلدهای هدرهای بسته هستن
ما اینجا تصمیم گیریمون هم میتونه فراتر از فورواردینگ باشه --> در روترهای سنتی ما فقط فورواردینگ میکنیم ینی اگر مچ شد فوروارد میشه به اون پورتی که تیبل میگه ولی اینجا میتونیم اکشن دلخواه داشته باشیم و چندتا اکشن بتونیم انجام بدیم اگر مچ شد که این انعطاف پذیری بالایی ایجاد میکنه
این کنترلر این الگوریتم رو اجرا میکنه و این جدول ها رو به اسم flow table هارو ایجاد میکنه برای هر روتر و براش می فرسته
یک پکت میاد و فیلدهای مورد نظر اون پکت مچ میشه با تیبل و هرکدوم که مچ شد اکشن مورد نظرش اجرا میشه حالا این اکشن میتونه فوروارد باشه مثلا یا چیزهای دیگه ای
مزیت:
توی این روش فقط دیگه محدود به الگوریتم دایکسترا نیستیم
مزیت مهم نرم افزار محور بودن کار است
و...

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

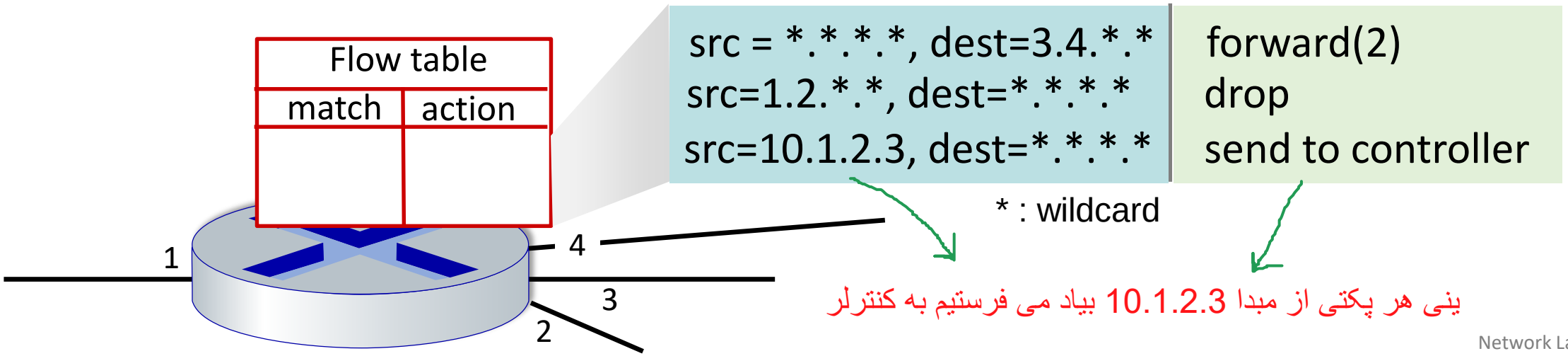


Router's flow table defines router's match+action rules

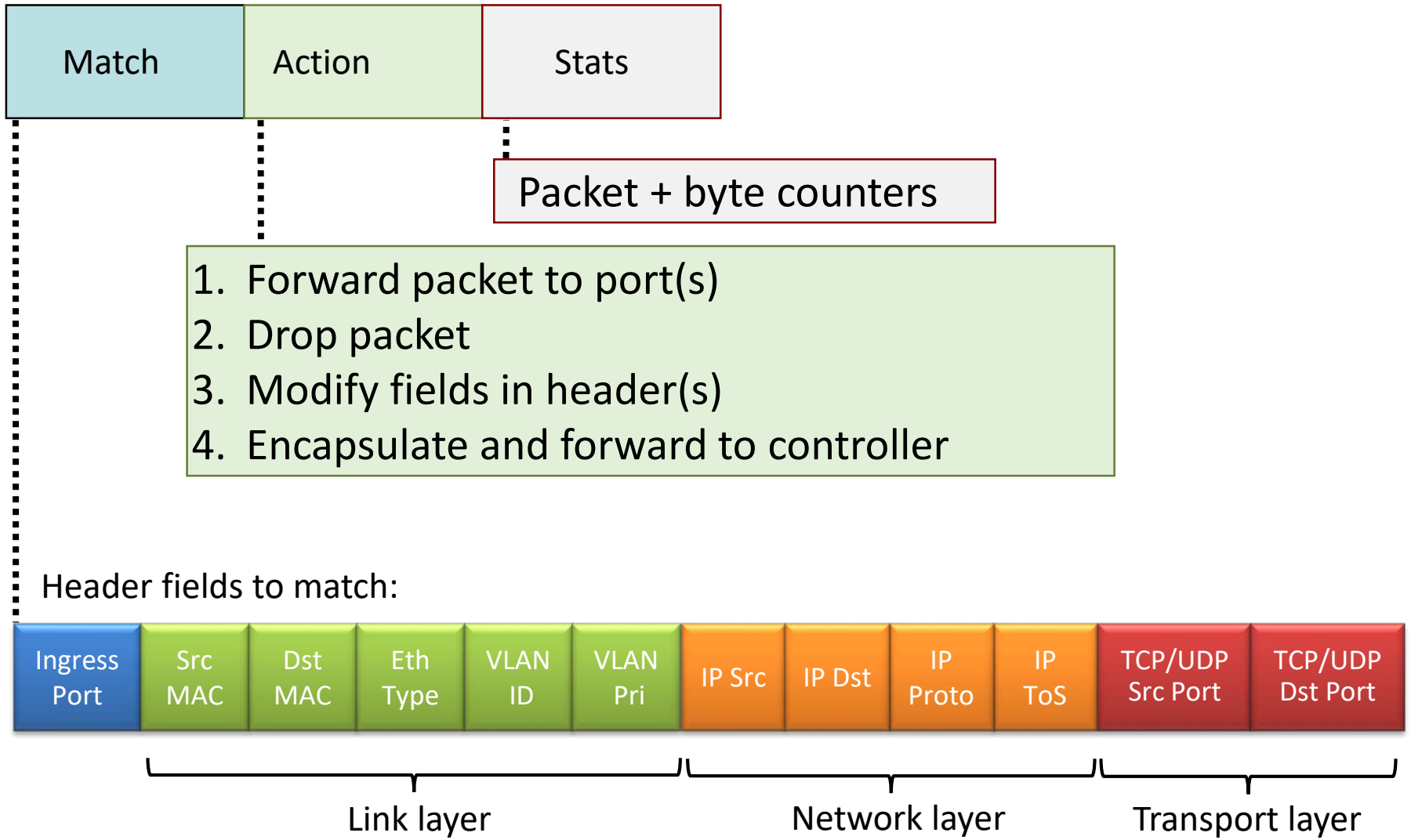
-
ساختار ما در SDN به این صورت است که:
اولا روی flow ها تصمیم میگیریم و یک flow براساس فیلدهای هدر مشخص میشه
توی openflow فرض ما اینه که برای تک تک بسته ها به تنهایی روتینگ انجام نمیدیم بلکه برای Flow انجام میدیم منتها اولین بسته که اومد براساس اون تصمیم میگیریم و بقیه بسته ها شرایطشون مثل اولین بسته است تا همه بسته ها تموم بشه
بعد generalized forwarding انجام میدیم : flow براساس یکسری فیلدهایی نوشته میشه و توی بسته باید همه اون فیلدهارو با flow مچ بکنیم و بعد اکشن ما برای پترن هایی که مچ شده اند انعطاف پذیر است و اکشن هایی که داریم اینجا: فورواردینگ، modify ،drop یا اینکه بفرستیمش به کنترلر ینی از مسیر بکشیم بیرون و بفرستیم به کنترلر و کنترلر مستقیما این پکت رو بررسی کنه و هر کاری که میخواد روش انجام بده
توی این جدول ها priority هم می تونه اعمال بشه چون مچنیگ های یک flow براساس اون فیلدهای هدر مشخص شده ممکنه یک پکت براساس فیلدهای مشخص شده با چندتا flow مچ بشه و اینجا باید اولویت داشته باشیم که کدوم مچ باید مبنای اون اکشن قرار بگیره
به ازای همه پکت های یک flow که مچ انجام میشه یکسری کانتر هم داریم که می شماریم اینارو که اینا در واقع برای مانیتورینگ می تونه استفاده بشه : counters
نکته: هر flow جدید که برقرار بشه از طریق شبکه کنترلر باید روترهایی که توی مسیرش قرار میگیرند جدول flowشون رو اپدیت بکنه و این flow رو هم اضافه کنه به اون
اینجا داریم داینامیک میریم
openflow پروتکلی است که کنترلر براساس اون با روتر ارتباط برقرار کنه و رد و بدل میکنه اطلاعات رو و هر کاری که بالا گفتیم رو انجام میده پس openflow پروتکل اجرایی همه این روش است و معماریش هم SDN بود و پروتکلش میشه openflow

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 جدول flow تعریف میکنه و ورژن های مختلف داره که ورژن اولیه اش همین است
rule: کدوم فیلدها و به ازای هر flow مشخص میکنه که چه پترنی باید داشته باشه که این فیلدها، فیلدهای لایه لینک، لایه نت ورک و لایه ترنسپرنت است و switch port ینی از کدوم پورت ورودی بسته اومده و براساس اینا مچ انجام میگیره
و مچ که انجام شد اکشن مشخص میشه توی فیلد که دسته هاشو نوشته
stats: فیلدهایی هستند که همون شمارنده هایی که اشاره کردیم ینی یه تعداد شمارنده به ازای هر Flow اپلای میشه و پکت هایی که مچ میشن رو می شماره که اینا کاربرد مانیتورینگ شبکه دارن

OpenFlow: examples

Destination-based forwarding:

این به شکل سنتی است:

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

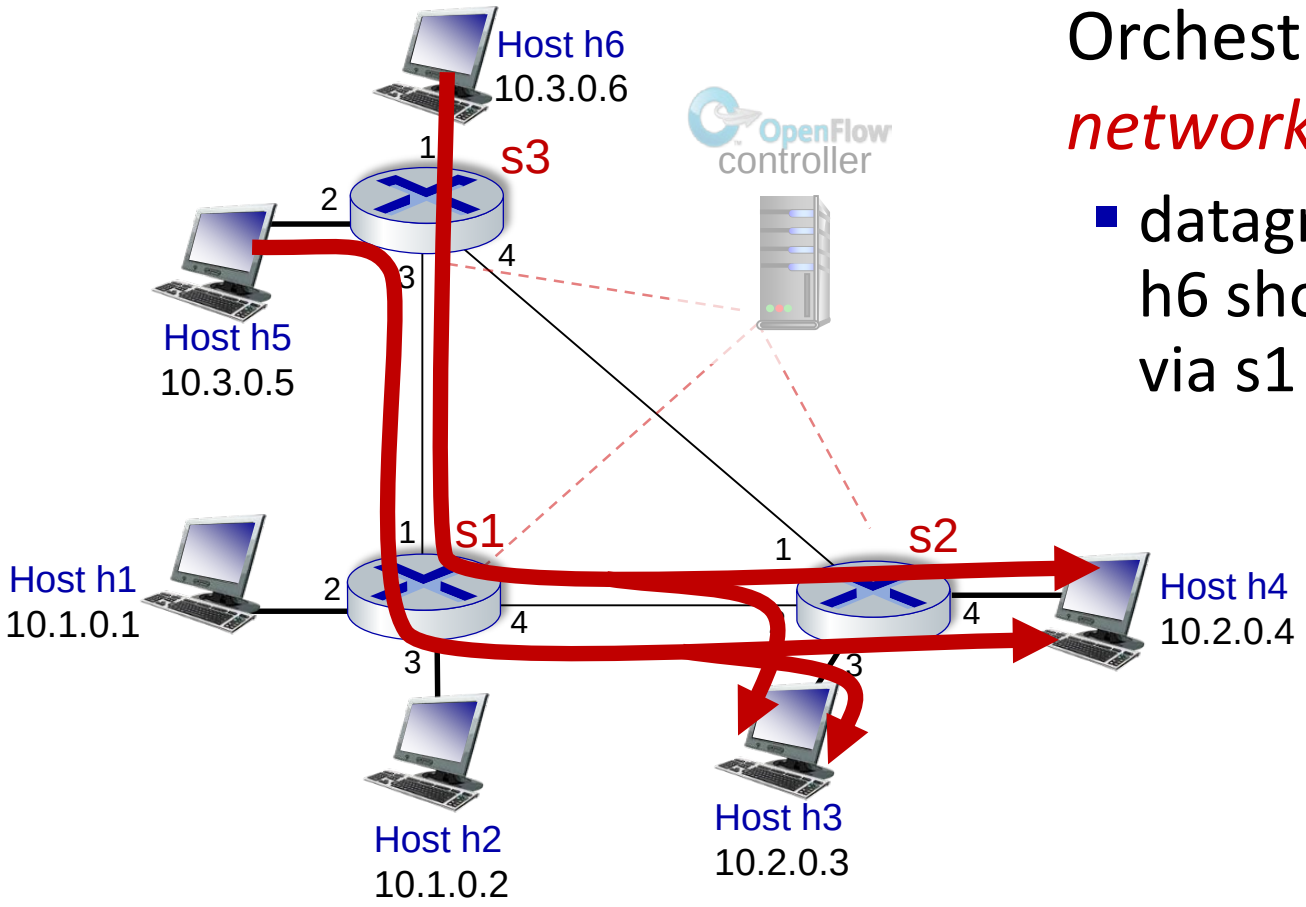
- *match*: IP addresses and TCP/UDP port numbers
- *action:* permit or deny

NAT

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

-
پس OpenFlow میاد action + match انجام میده در حالی که روترمون مچش longest destination IP prefix بود و اکشنش فوروارد به پورت خروجی که در OpenFlow هم این کارو میتونیم انجام بدیم و هم سوییچ لایه دو رو می تونیم انجام بدیم و هم این که میتونیم firewalling انجام بدیم و هر اکشن دیگری که OpenFlow اجازه میده مثلا بازم بخوایم بگیم مثل nat که اکشنش اینه که پورت نامبر عوض میکنیم

OpenFlow example

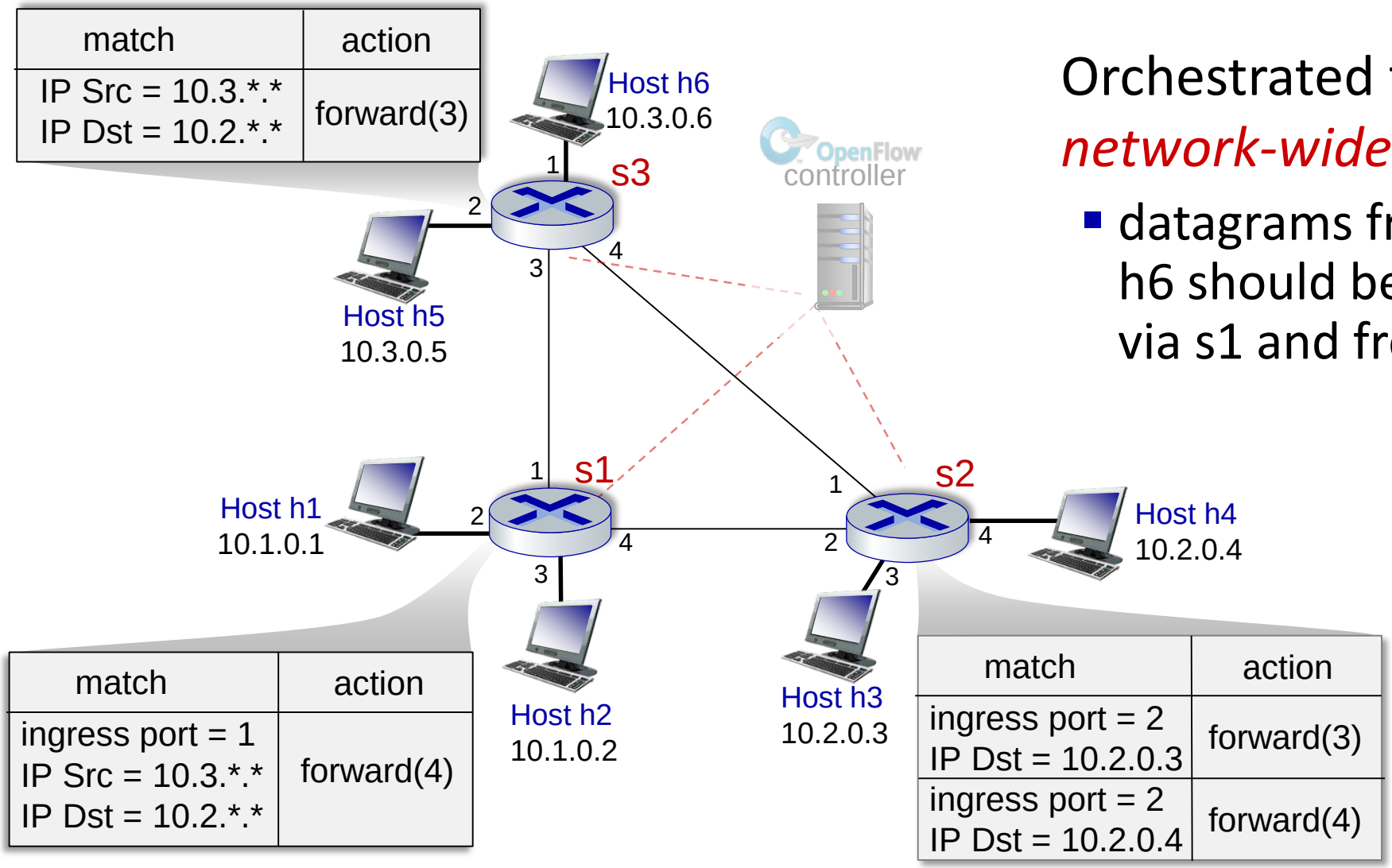


Orchestrated tables can create *network-wide* behavior, e.g.,:

- 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

--

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" n*etwork-wide* behaviors
- simple form of "network programmability"
  - programmable, per-packet "processing"
  - *historical roots:* active networking
  - *today:* more generalized programming: P4 (see p4.org).

Generalized forwarding: summary

Question: how are forwarding tables (destination-based forwarding) or flow tables (generalized forwarding) computed?

Answer: by the control plane (next chapter)
