Διαχείριση Δικτύων Βασισμένων στο Λογισμικό 2025 (DIT306)

Δρ. Ειρήνη Λιώτου
eliotou@hua.gr

3/4/2025

Chapter 4
Network Layer: The Data Plane

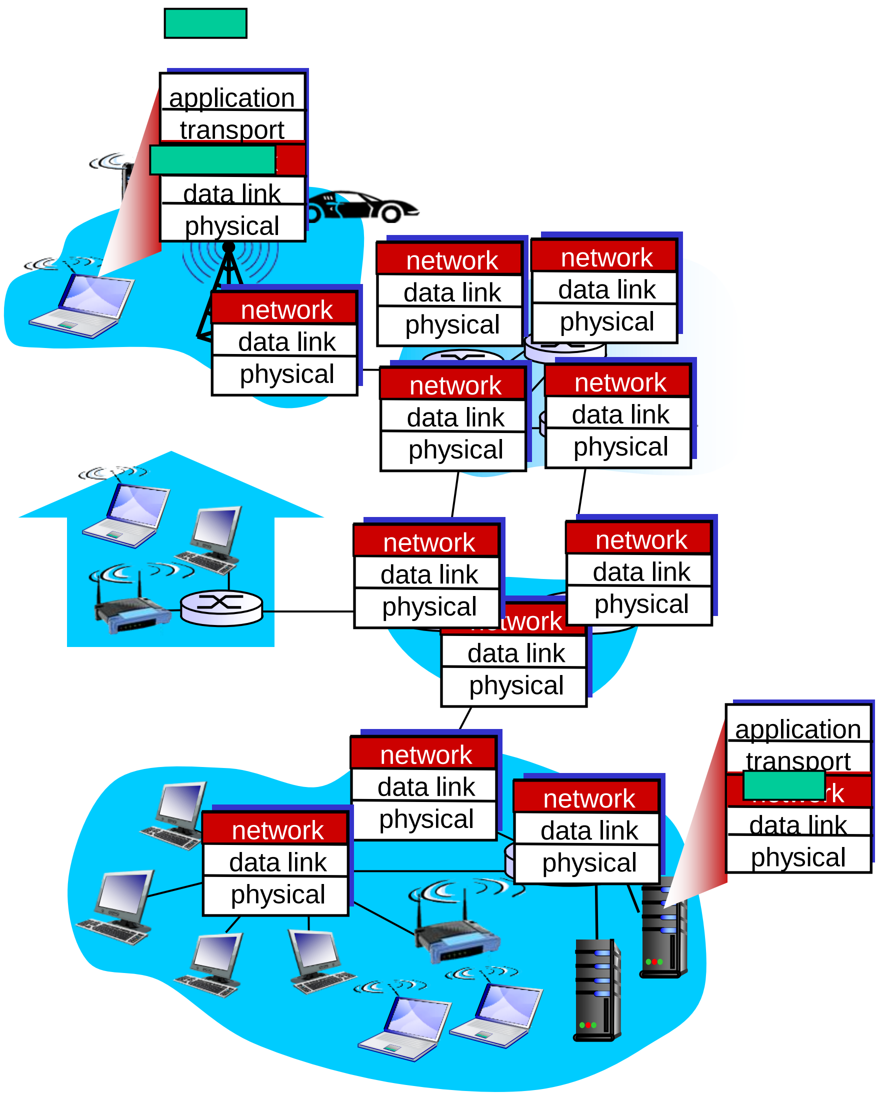
Chapter 4: outline

4.1 Overview of Network layer
- data plane
- control plane

4.4 Generalized Forward and SDN
- match
- action
- OpenFlow examples of match-plus-action in action

Network layer

- transport segment from sending to receiving host
- on sending side encapsulates segments into datagrams
- on receiving side, delivers segments to transport layer
- network layer protocols in *every* host, router
- router examines header fields in all IP datagrams passing through it



Two key network-layer functions

network-layer functions:

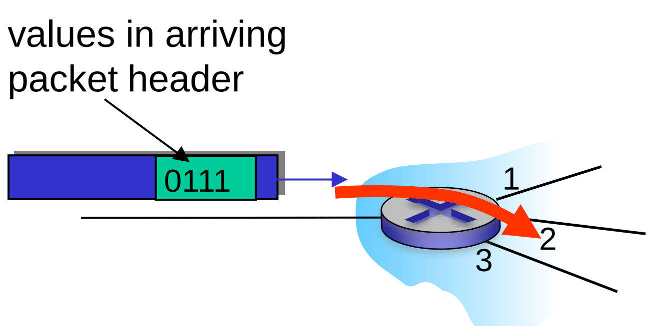
- *Forwarding (HW):* move packets from router's input to appropriate router output
- *Routing (SW):* determine route taken by packets from source to destination
  - *routing algorithms*

analogy: taking a trip

- *forwarding:* process of getting through single interchange
- *routing:* process of planning trip from source to destination

Network layer: data plane, control plane

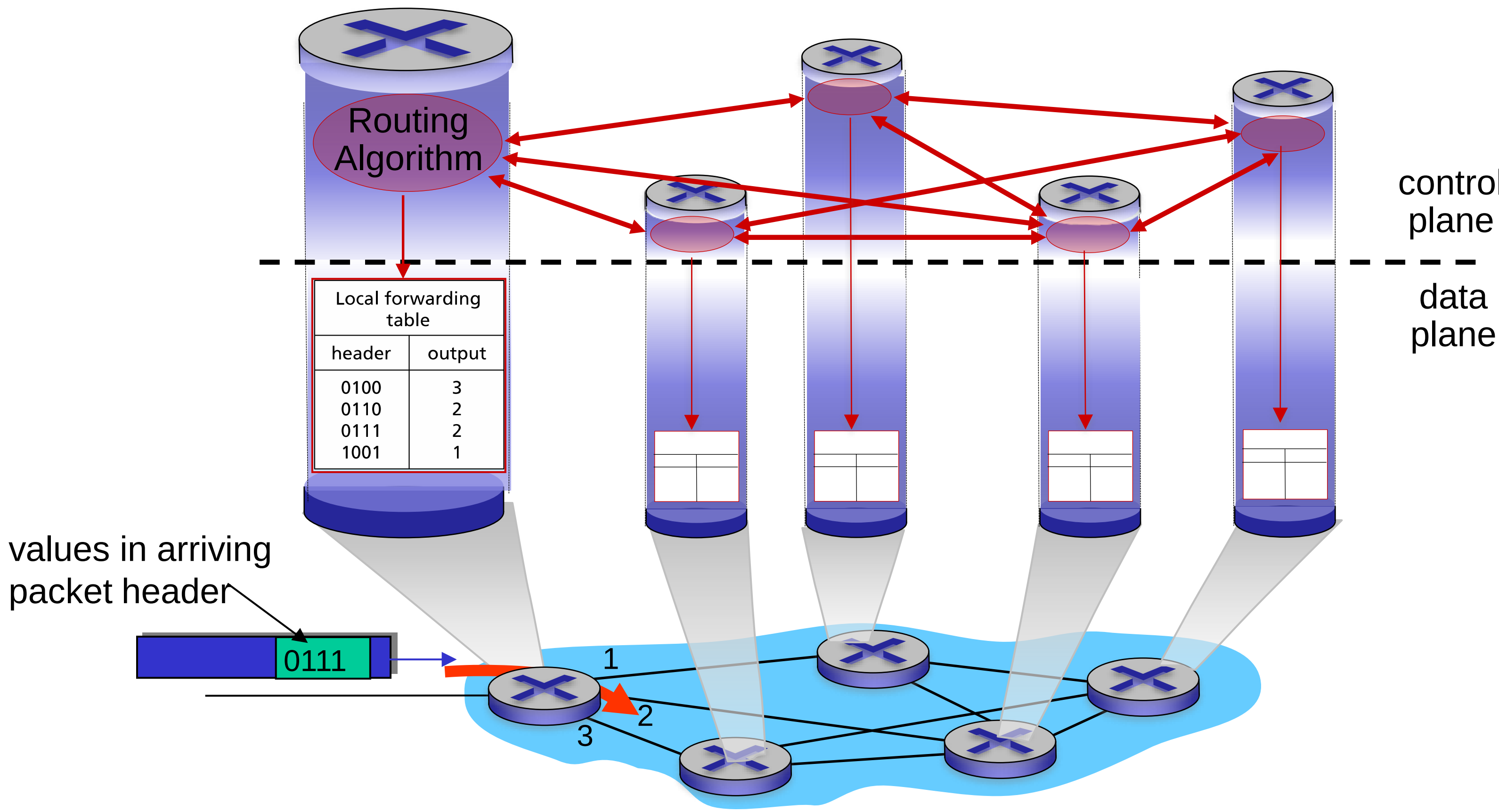
Data plane
- local, per-router function
- determines how datagram arriving on router input port is forwarded to router output port
- forwarding function



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

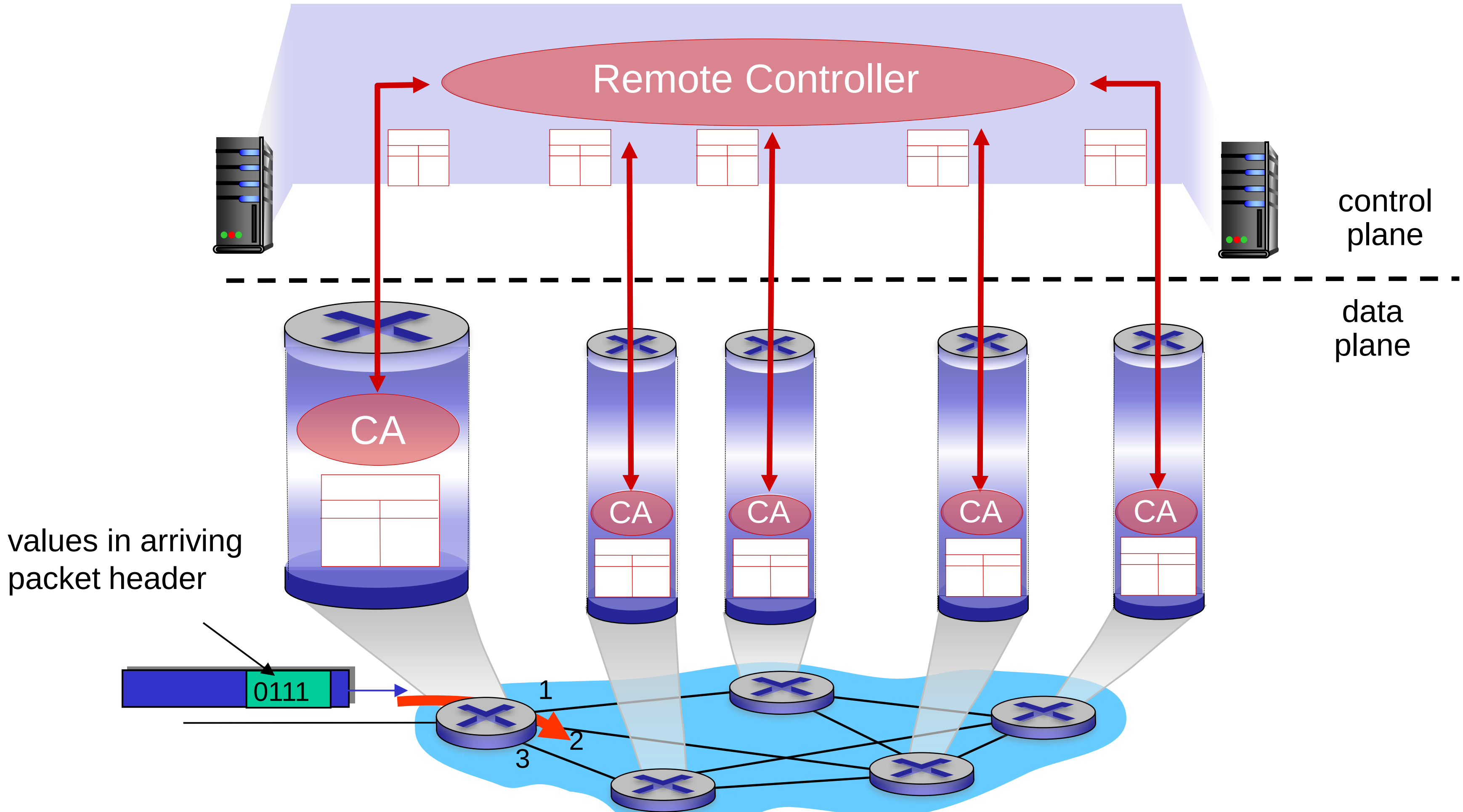
Per-router control plane

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

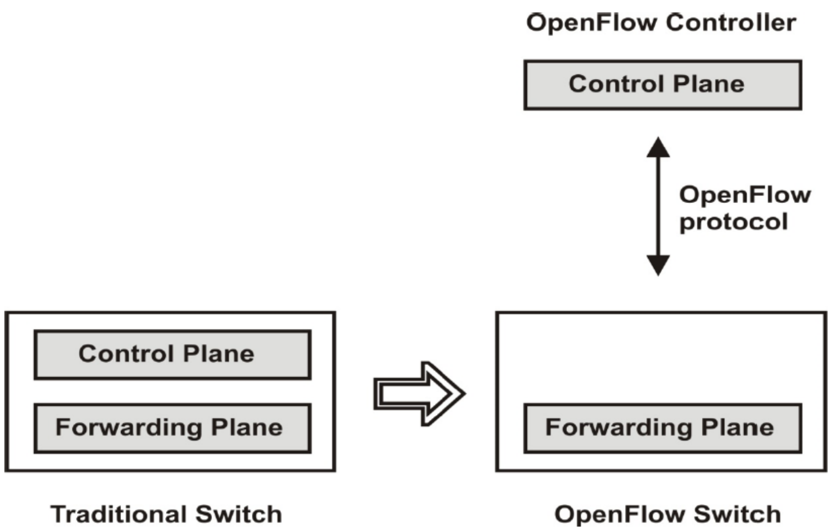


Logically centralized control plane

A distinct (typically remote) controller interacts with local control agents (CAs)



Separation of Data Plane & Control Plane



SDN has decoupled both the hardware & software parts. You can buy the hardware from one vendor or even use merchant silicon devices. Software part can be obtained from other vendors or can use Open Source control planes which are free available.

Problems in Traditional Network Devices

- They are vendor specific
- Hardware & Software is bundled together
- Very costly
- New features can only be added at the will of the vendor. Client can only request the features, vendor will decide whether to add those features or not & the time frame in which these features will become available is at the sole discretion of the vendor.
- Devices are function specific. You can not make your router behave like load balancer or make your switch behave like a firewall or vice versa.
- If your network consists of hundred of these devices, each device has to be configured individually. There is no centralized management.
- Innovations are very rare. Last 3 decades have not seen many innovations in networking. Whereas Compute and storage industry has seen drastic changes such as compute virtualization & storage virtualization. Networking has not been able to keep pace with other ingredients of Cloud Computing.

Network service model

Q: What *service model* for "channel" transporting datagrams from sender to receiver?

example services for individual datagrams:

- guaranteed delivery
- guaranteed delivery with less than 40 msec delay (bounded delay)

example services for a flow of datagrams:

- in-order datagram delivery
- guaranteed minimum bandwidth to flow

Network service model

Network Architecture	Service Model	Quality of Service (QoS) Guarantees ?			
		Bandwidth	Loss	Order	Timing
Internet	best effort	none	no	no	no

Internet "best effort" service model

No guarantees on:

i. successful datagram delivery to destination
ii. timing or order of delivery
iii. bandwidth available to end-end flow

Reflections on best-effort service

- simplicity of mechanism has allowed Internet to be widely deployed adopted
- sufficient provisioning of bandwidth allows performance of real-time applications (e.g., interactive voice, video) to be "good enough" for "most of the time"
- replicated, application-layer distributed services (datacenters, content distribution networks) connecting close to clients' networks, allow services to be provided from multiple locations (e.g. Netflix)
- congestion control of "elastic" services helps (TCP)

It's hard to argue with success of best-effort service model

Chapter 4: outline

4.1 Overview of Network layer

- data plane
- control plane

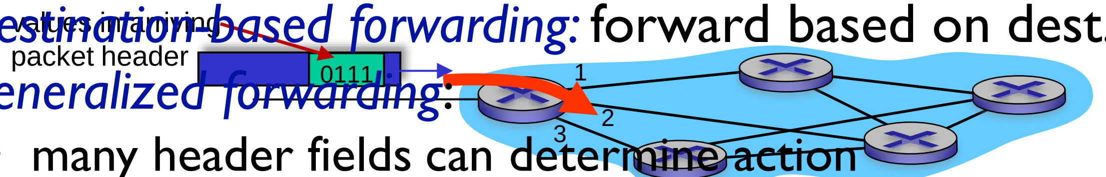
4.4 Generalized Forward and SDN

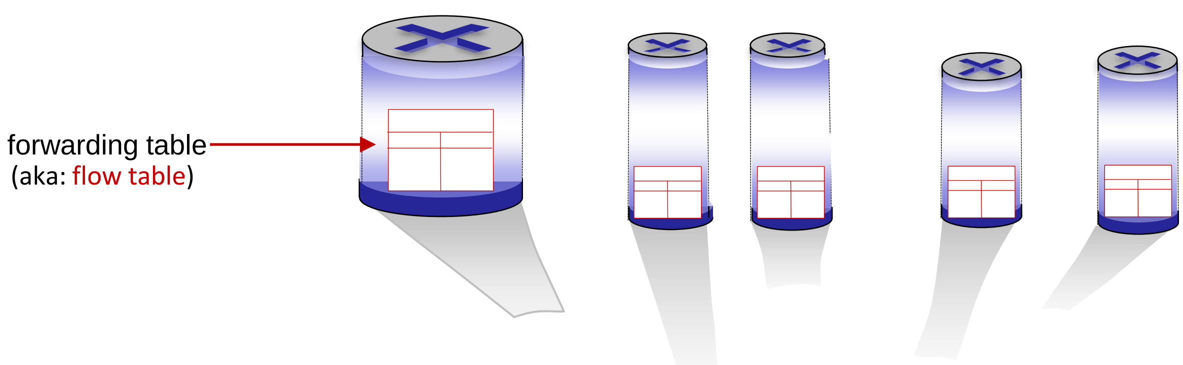
- match
- action
- OpenFlow examples of match-plus-action in action

Generalized forwarding: match plus action

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

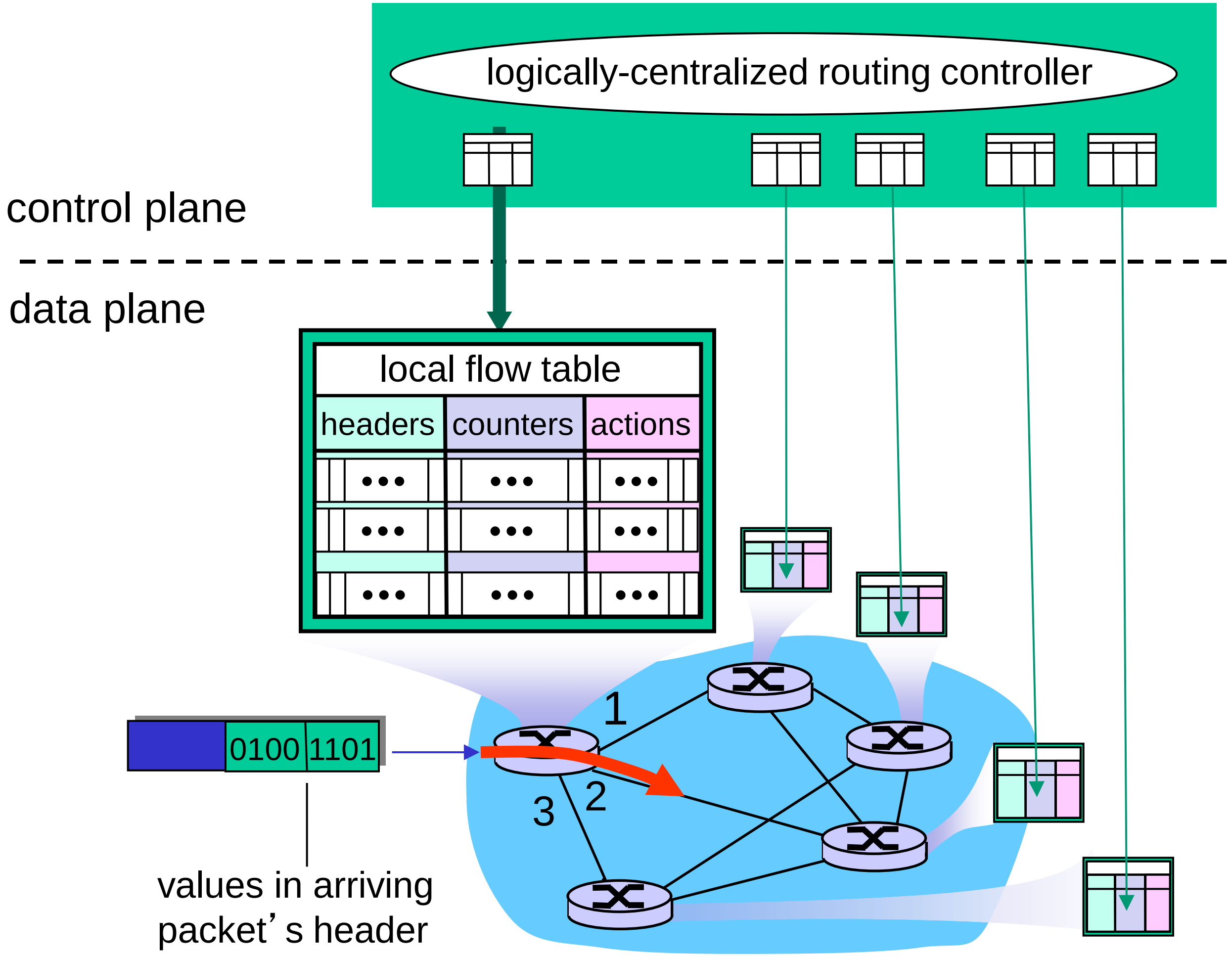
- "match plus action" abstraction: match bits in arriving packet, take action
  - *destination-based forwarding:* forward based on dest. IP address
  - *generalized forwarding:*
    - many header fields can determine action
    - many actions possible: drop/copy/modify/log packet





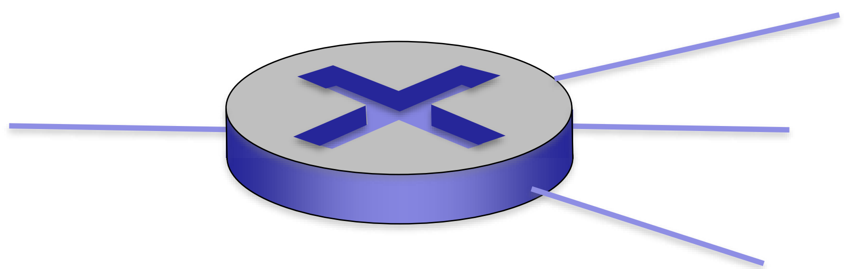
Generalized Forwarding and SDN

Each router contains a *flow table* that is computed and distributed by a *logically centralized* routing controller



OpenFlow data plane abstraction

- *flow*: defined by header fields values (in link-, network-, transport-layer fields
- generalized forwarding: simple packet-handling rules
  - *Pattern:* match 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

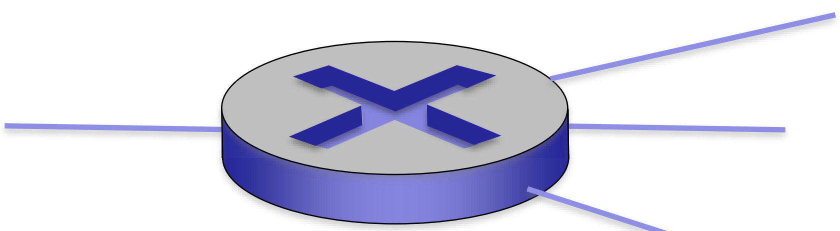


* : wildcard

Flow table in a router (computed and distributed by controller) define router's match+action rules

OpenFlow data plane abstraction

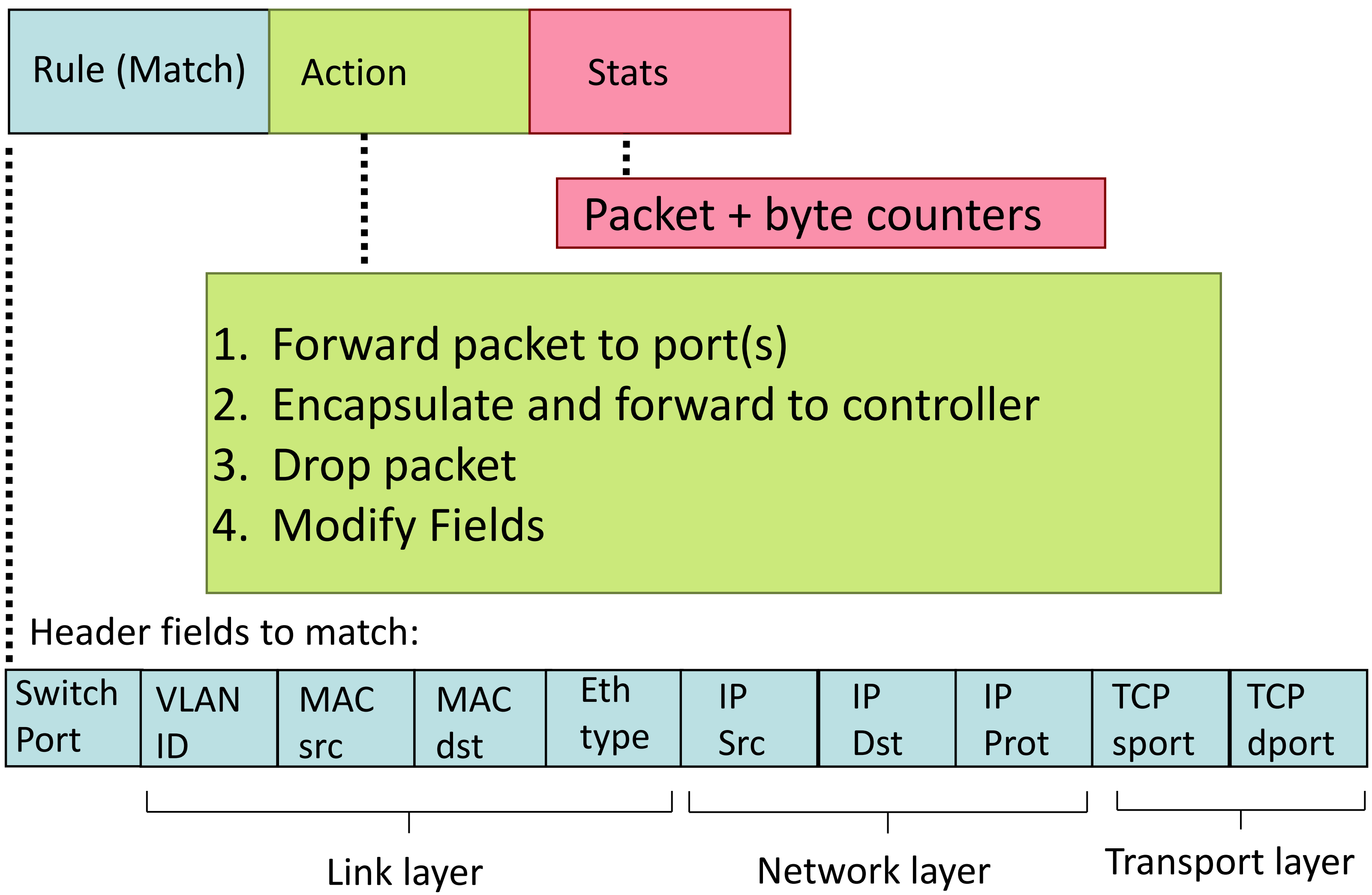
- *flow*: defined by header fields values (in link-, network-, transport-layer fields
- generalized forwarding: simple packet-handling rules
  - *Pattern:* match values in packet header fields
  - *Actions for matched packet:* drop, forward, modify matched packet or send matched packet to controller
  - *Priority:* disambiguate overlapping patterns
  - *Counters:* #bytes and #packets



* : wildcard

1. src=1.2.*.*, dest=3.4.5.* → drop
2. src = *.*.*.*, dest=3.4.*.* → forward(2)
3. src=10.1.2.3, dest=*.*.*.* → send to controller

OpenFlow: Flow Table Entries



Examples

Destination-based forwarding:

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	Action
*	*	*	*	*	*	*	*	*	22	drop

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

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

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

Examples

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 3

OpenFlow abstraction

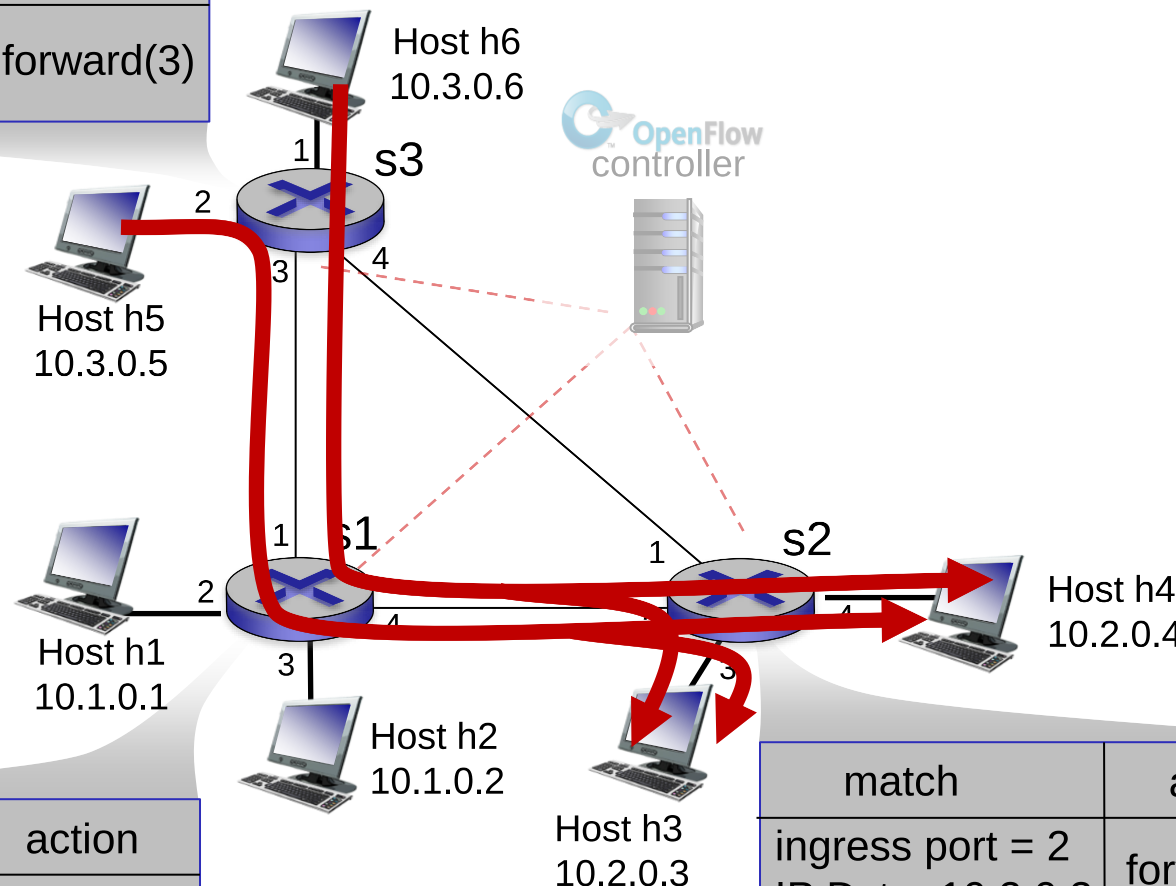
- *match+action:* unifies different kinds of devices

- 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

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)

Manipulation of Flow Table Entries for Creating Network Applications

	Ingress Port	MAC src	MAC dst	Eth type	VLAN ID	IP src	IP dst	TCP sport	TCP dport	Action
Flow Switching	Port1	00:..:01	00:..:03	0800	vlan2	10.0.0.1	10.0.0.3	35554	80	port2
Firewall	*	*	*	*	*	*	*	*	23	drop
Switching	*	*	00:..:04	* .	*	*	*	*	*	port4
Routing	*	*	*	*	*	*	10.0.0.5	*	*	port5
Load Balancer	*	*	00:..:fe	0x800	vlan1	10.0.0.3	10.0.0.254	*	80	mod_nw_dst port1
Packet In (table miss entry)	*	*	*	*	*	*	*	*	*	controller

Flow Entries in Flow Table