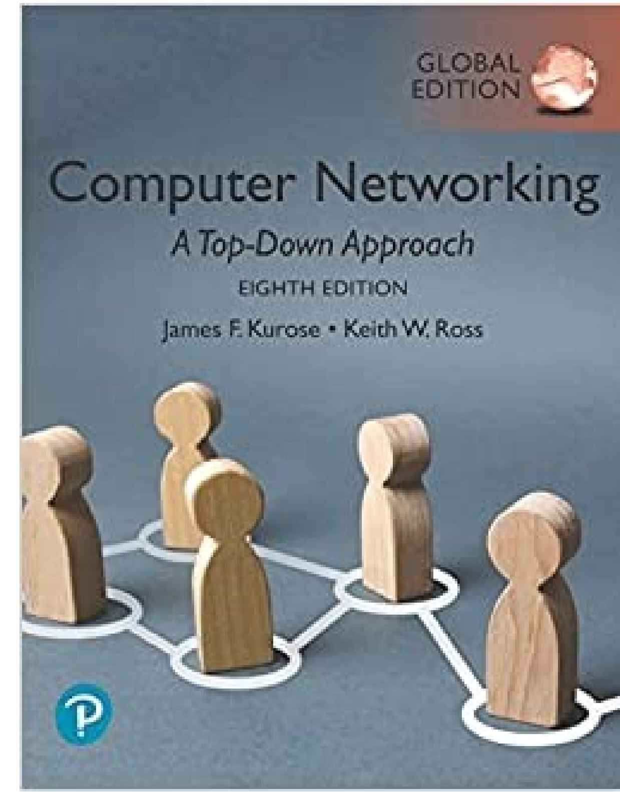
Chapter 4
Network Layer: The Data Plane part 2

School of Computing
Gachon Univ.
Joon Yoo



Computer Networking: A Top-Down Approach
8th edition (Global edition)
Jim Kurose, Keith Ross
Pearson, 2021

Many slides from J.F Kurose and K.W. Ross

가천대학교
Gachon University

Chapter 4: outline

4.1 Overview of Network layer
- data plane
- control plane

4.2 What' s inside a router

4.3 IP: Internet Protocol
- datagram format
- fragmentation
- IPv4 addressing
- network address translation
- IPv6

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

Question

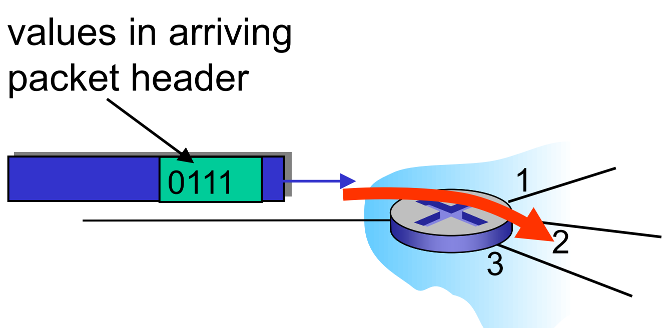
What happens inside a router?

Cisco Catalyst 4506-E Switch

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

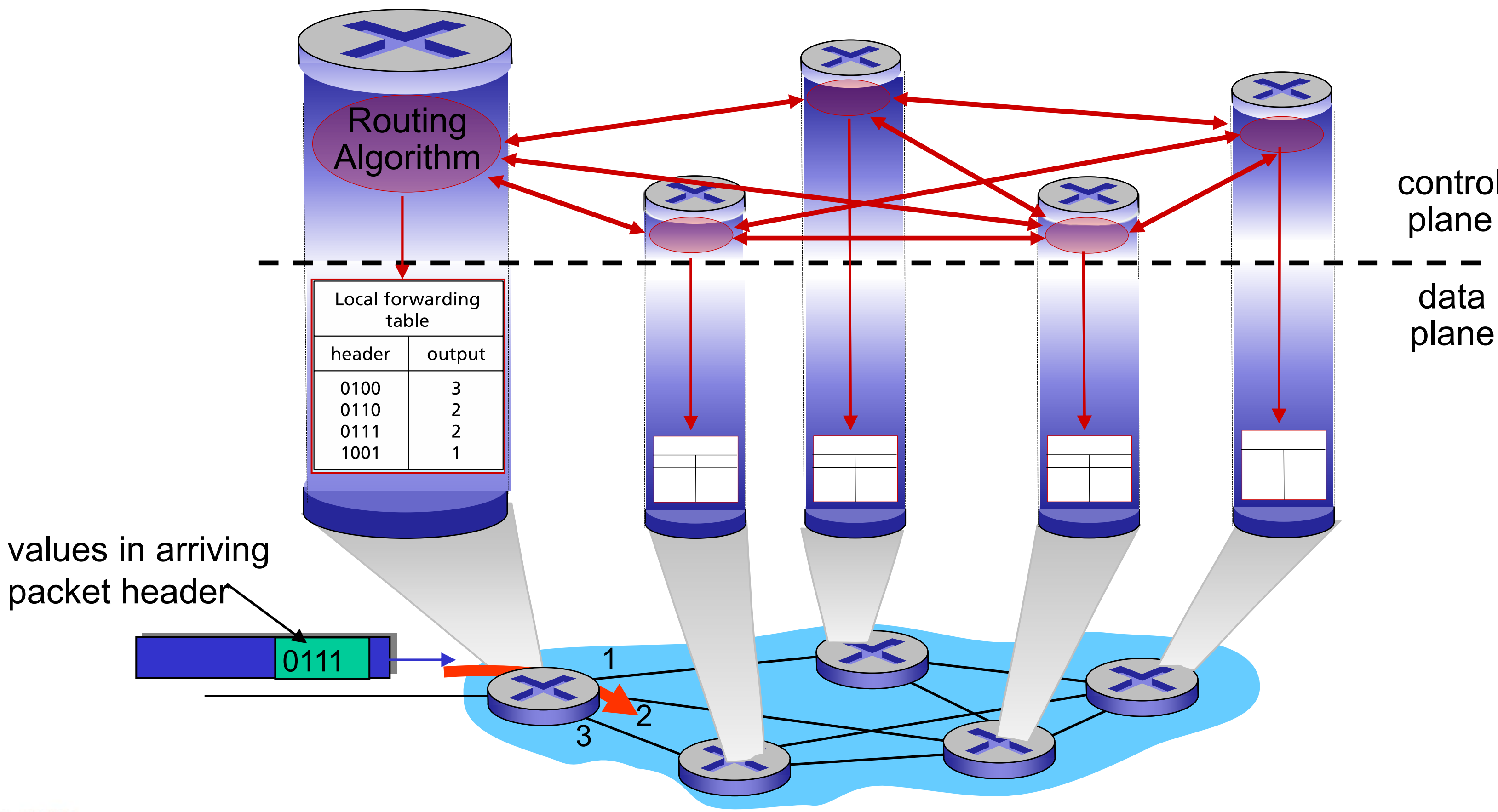


Control plane

- network-wide logic
- determines how datagram is routed among routers along end-end path from source host to destination host

Per-router control plane

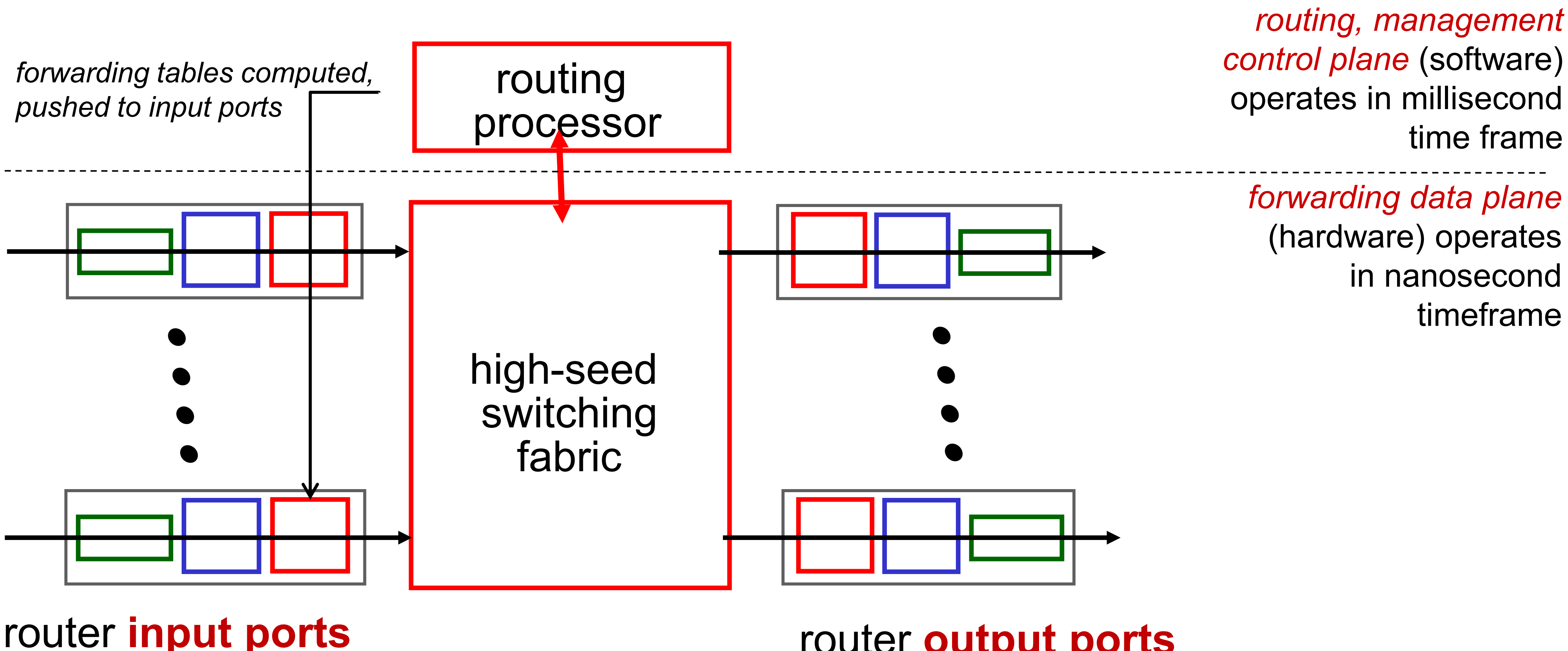
Individual routing algorithm components *in each and every router* interact in the control plane



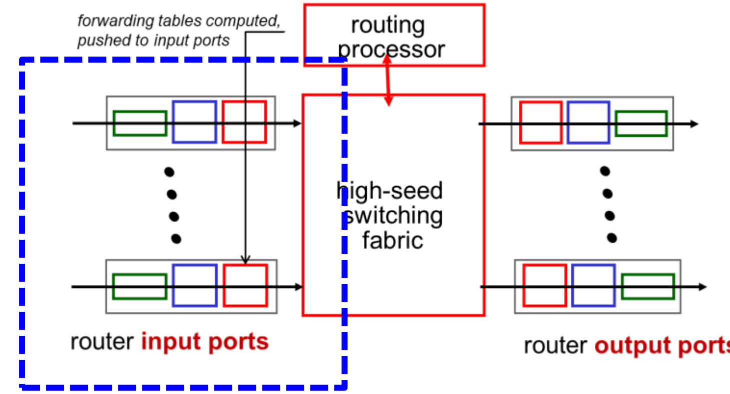
Router architecture overview

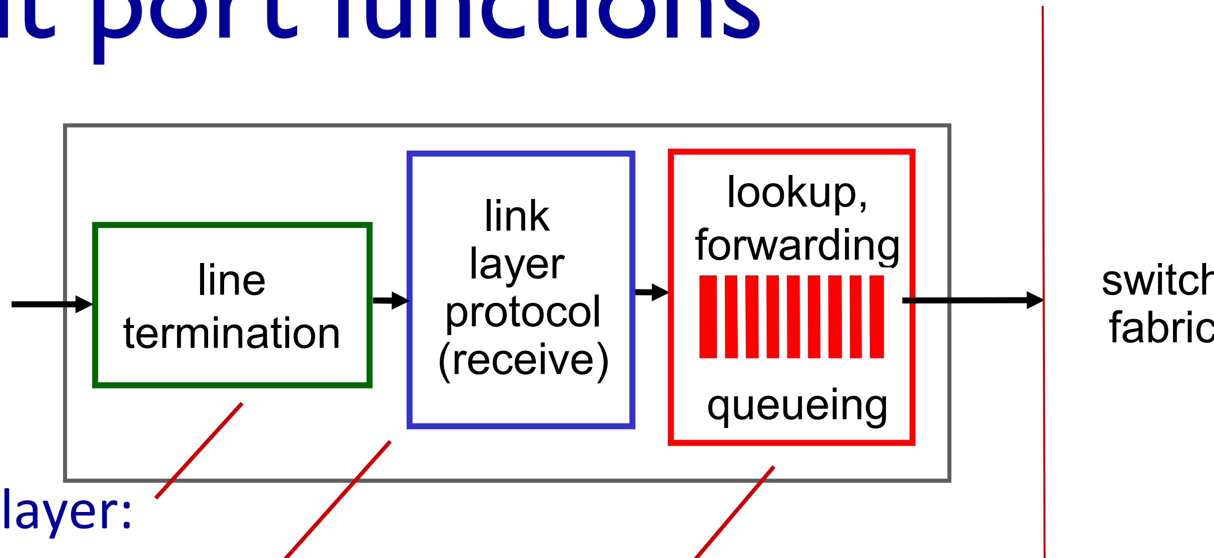
two key router functions:

- run routing algorithms/protocol (RIP, OSPF, BGP) – discussed in Ch. 5
- *forwarding* datagrams from incoming to outgoing link



Input port functions





physical layer:
bit-level reception

link layer:
e.g., Ethernet
(chapter 6)

decentralized switching:

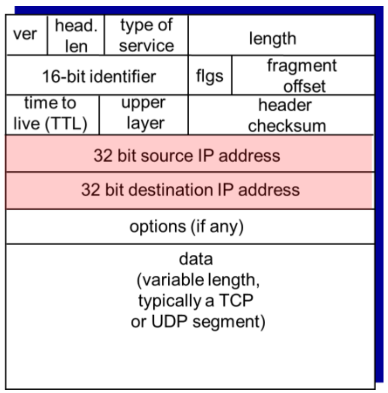
- using header field values, lookup output port using forwarding table in input port memory *("match plus action")*
- destination-based forwarding: forward based only on destination IP address (traditional)
- generalized forwarding: forward based on any set of header field values

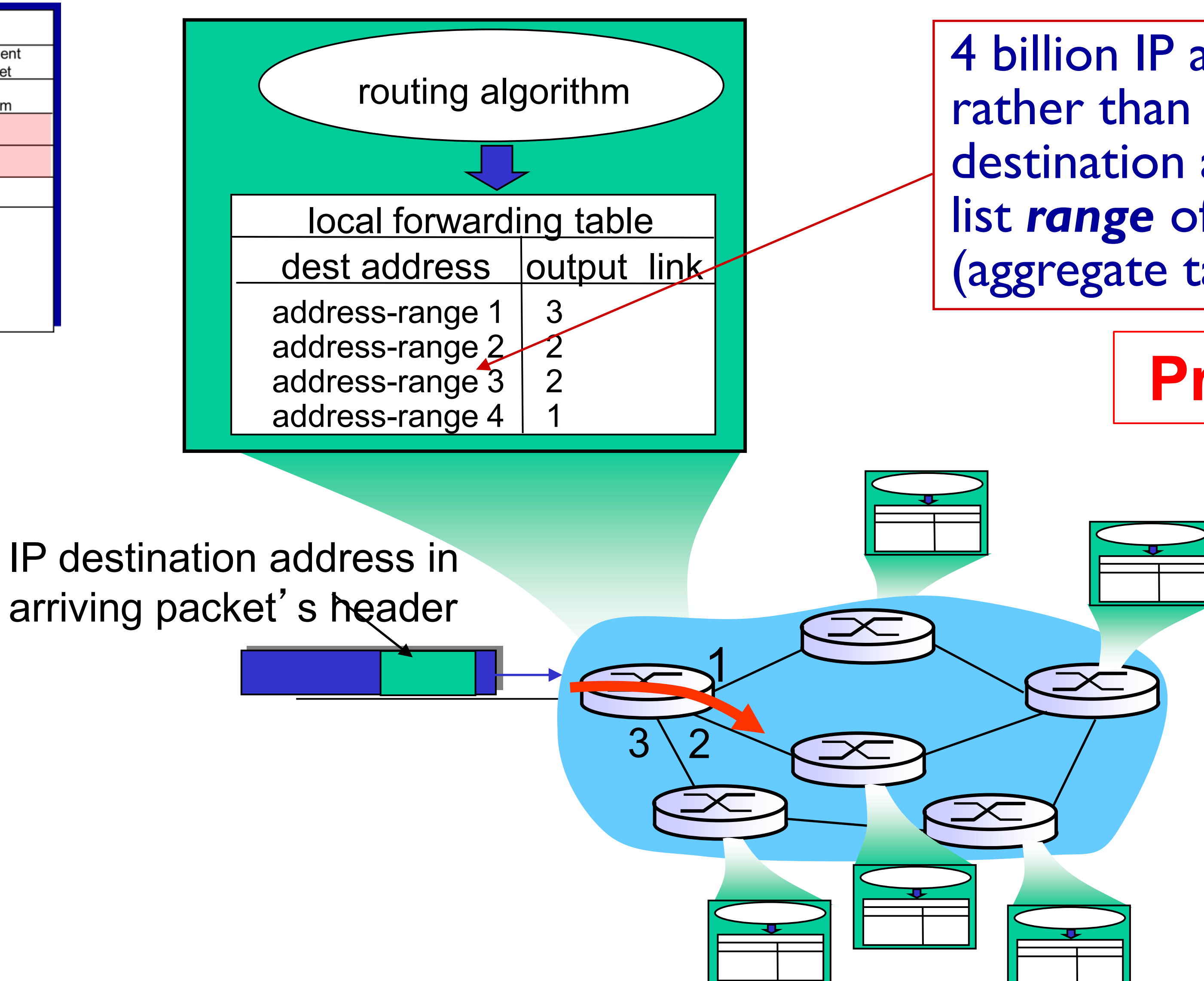
가천대학교
Gachon University

Hop-by-Hop Packet Forwarding

- Each router has a forwarding table
  - Maps destination address to output port
- Upon receiving a packet
  - Inspect the destination address in the header
  - Index into the table
  - Determine the output port
  - Forward the packet out that output port
- Then, the next router in the path repeats

Datagram forwarding table



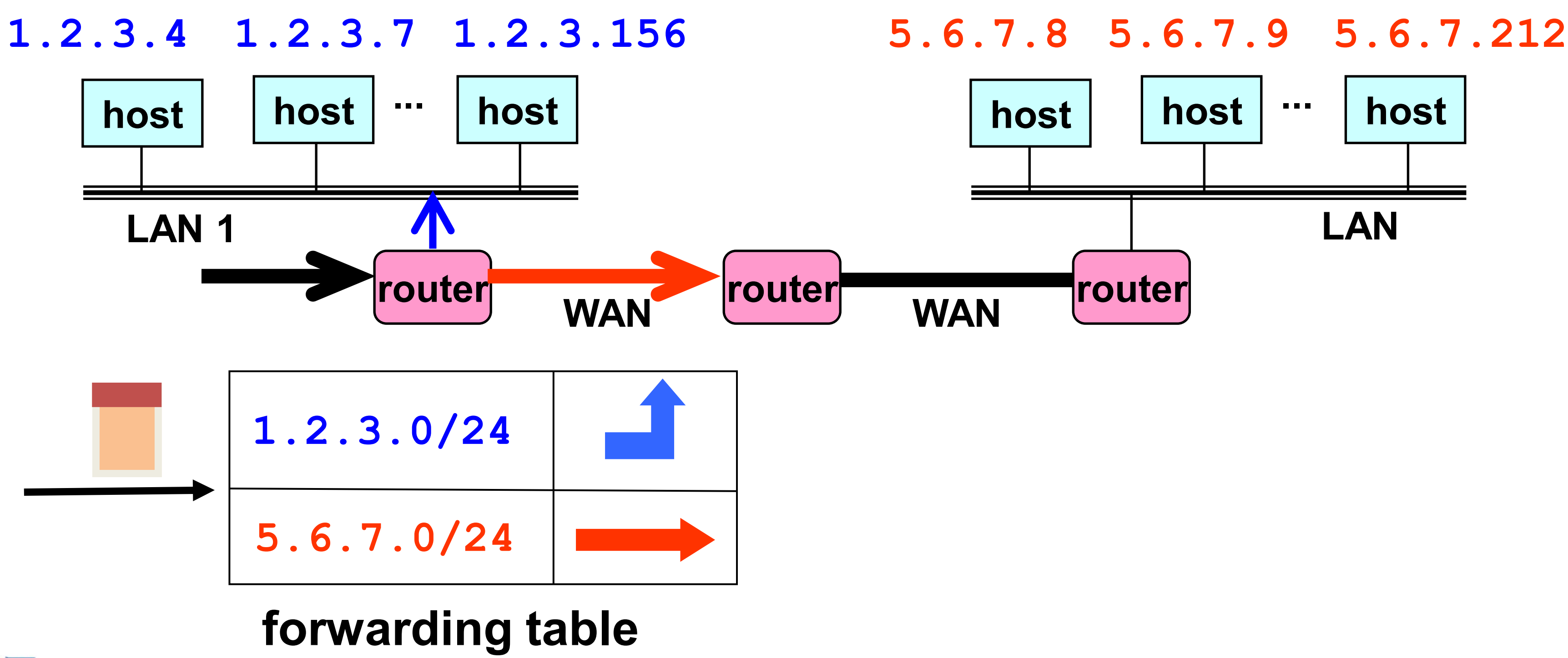


4 billion IP addresses, so rather than list individual destination address list ***range*** of addresses (aggregate table entries)

Prefix

Separate Forwarding Entry Per Prefix

- Prefix-based forwarding
  - Map the destination address to matching prefix
  - Forward to the outgoing interface



가천대학교 Gachon University

prefix matching (Ch. 4.2.1)

prefix matching

- destination address ranges are expressed as prefix (subnet address)
- match the prefix – forward the packet

Prefix	Link interface	
11001000 00010111 00010*** *********	0	200.23.16.0 / 21
11001000 00010111 00011000 *********	1	200.23.24.0 / 24
11001000 00010111 00011*** *********	2	200.23.24.0 / 21
otherwise	3	

example:

DA: 11001000 00010111 00010110 10100001 — *which interface?* — 200.23.22.161

DA: 11001111 00011111 00011000 00111111 — *which interface?* — 207.31.24.63

DA: 11001000 00010111 00011000 10101010 — *which interface?* — 200.23.24.170

Q: but what happens if ranges don't divide up so nicely?

Longest prefix matching

longest prefix match

when looking for forwarding table entry for given destination address, use *longest* address prefix that matches destination address.

Destination Address Range	Link interface
11001000 00010111 00010*** ********	0
11001000 00010111 00011000 ********	1
11001000 00010111 00011*** ********	2
otherwise	3

examples:

11001000 00010111 00010110 10100001 which interface?

11001000 00010111 00011000 10101010 which interface?

Longest prefix matching

longest prefix match

when looking for forwarding table entry for given destination address, use *longest* address prefix that matches destination address.

Destination Address Range				Link interface
11001000	00010111	00010***	********	0
11001000	00010111	00011000	********	1
11001000	00010111	00011***	********	2
otherwise				3

match!

examples:

11001000 00010111 00010110 10100001 which interface?

11001000 00010111 00011000 10101010 which interface?

Longest prefix matching

longest prefix match

when looking for forwarding table entry for given destination address, use *longest* address prefix that matches destination address.

Destination Address Range	Link interface
11001000 00010111 00010*** ********	0
11001000 00010111 00011000 ********	1
11001000 00010111 00011*** ********	2
otherwise	3

match!

examples:

11001000 00010111 00010110 10100001 which interface?

11001000 00010111 00011000 10101010 which interface?

Longest prefix matching

longest prefix match

when looking for forwarding table entry for given destination address, use *longest* address prefix that matches destination address.

Destination Address Range				Link interface
11001000	00010111	00010***	********	0
11001000	00010111	00011000	********	1
11001000	00010111	00011***	********	2
otherwise				3

match!

examples:

11001000	00010111	00010110	10100001	which interface?
11001000	00010111	00011000	10101010	which interface?

Longest prefix matching

longest prefix match

when looking for forwarding table entry for given destination address, use *longest* address prefix that matches destination address.

- **Important!:** Why do we need longest prefix matching? – next few slides

IP addresses: how can your network get them? (Ch.4.3.3)

ISP-A acquires a block IP address (200.23.16.0/20)

– how many addresses? $2^{12} = 4096$

ISP's block	11001000 00010111 00010000 00000000	200.23.16.0/20

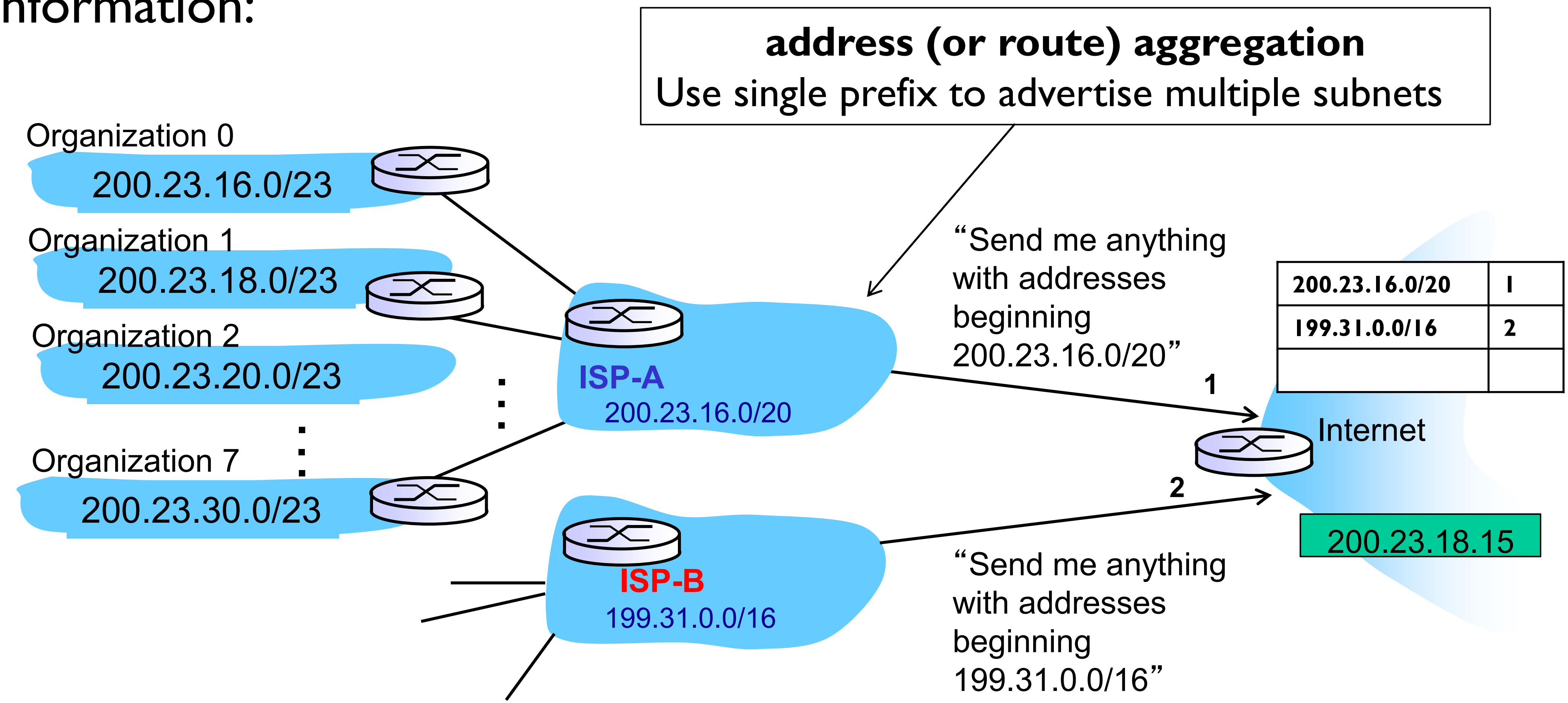
It then allocates blocks of IP addresses to 8 organizations:

Organization 0	11001000 00010111 00010000 00000000	200.23.16.0/23
Organization 1	11001000 00010111 00010010 00000000	200.23.18.0/23
Organization 2	11001000 00010111 00010100 00000000	200.23.20.0/23
…..	…. ….	
Organization 7	11001000 00010111 00011110 00000000	200.23.30.0/23

– how many addresses for each organization?

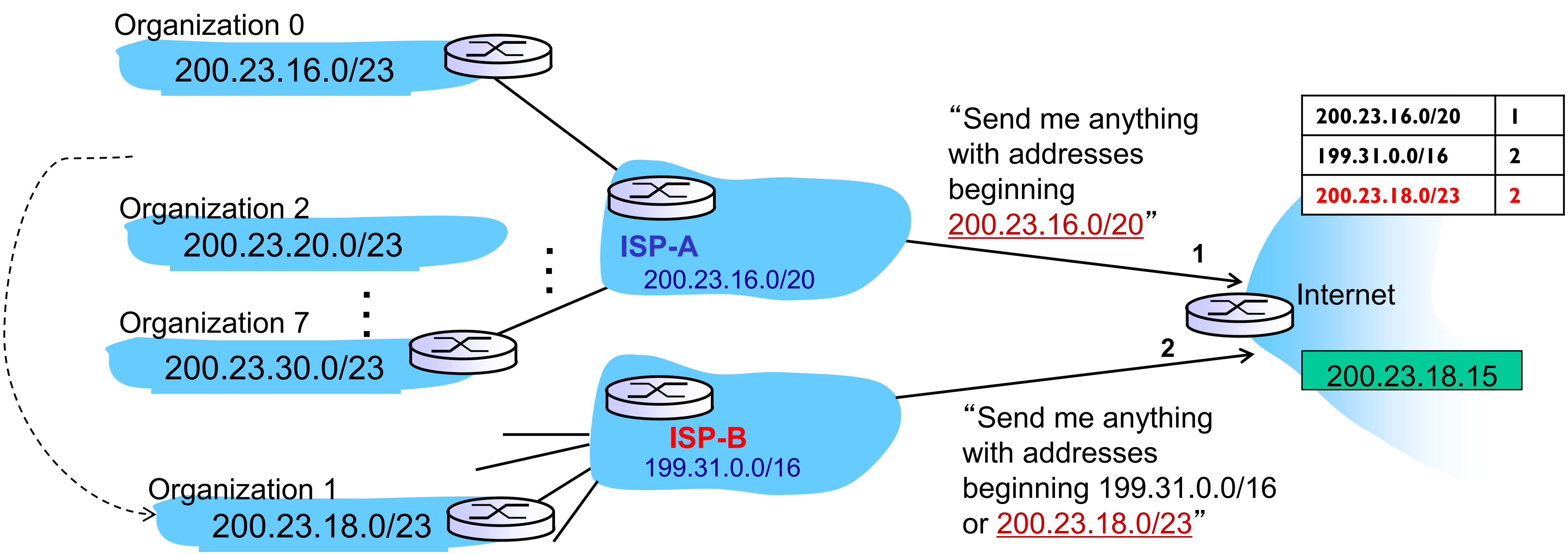
Hierarchical addressing: route aggregation

hierarchical addressing allows efficient advertisement of routing information:



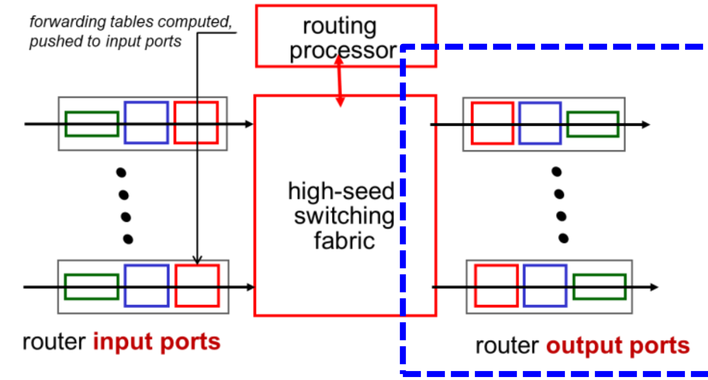
Hierarchical addressing: more specific routes

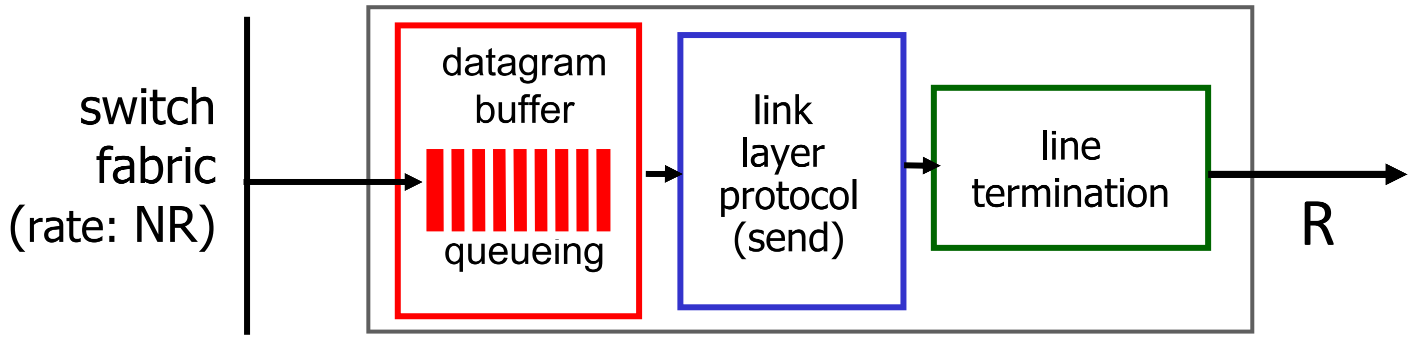
ISP-A acquires **ISP-B** and moves Organization 1 to **ISP-B**:
ISP-B has a more specific route to Organization 1
– ***longest prefix matching***



가천대학교 Gachon University

Output port queuing

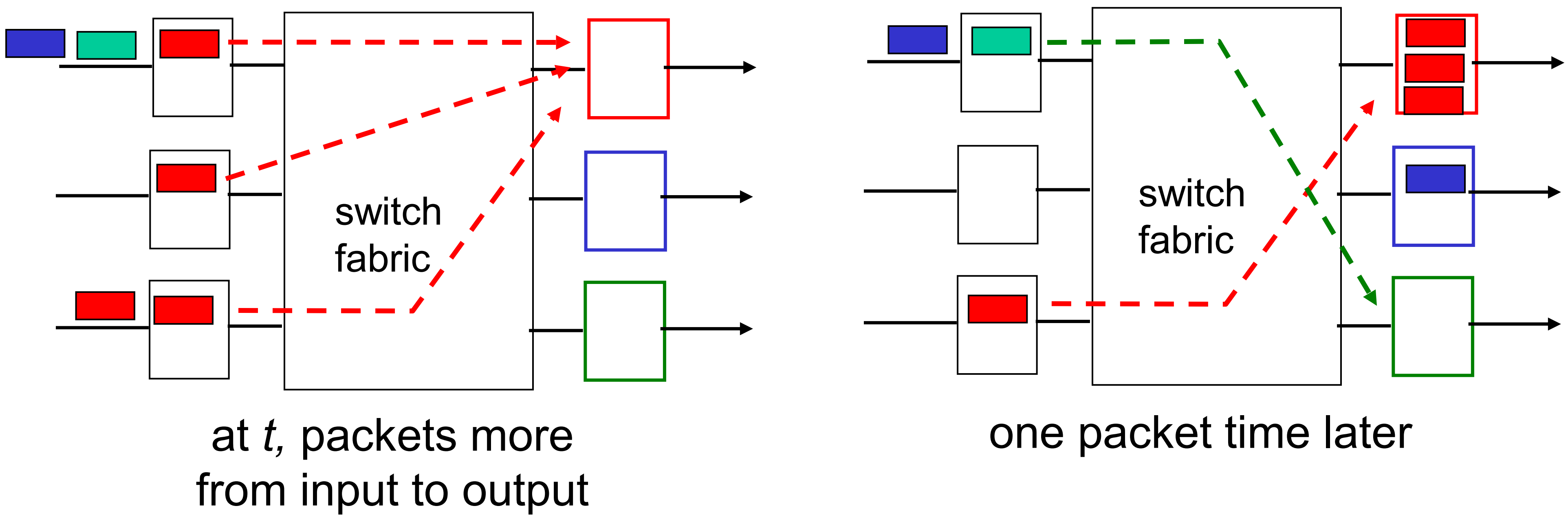




- *Buffering* required when datagrams arrive from fabric faster than link transmission rate.

Datagrams can be lost due to congestion, lack of buffers

Output port queueing



- buffering when arrival rate via switch exceeds output line speed
- *queueing (delay) and loss due to output port buffer overflow!*

Chapter 4: outline

4.1 Overview of Network layer
- data plane
- control plane

4.2 What's inside a router

4.3 IP: Internet Protocol
- datagram format
- fragmentation
- IPv4 addressing
- network address translation
- IPv6

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

Middleboxes

Middleboxes

Middlebox (RFC 3234)

"any intermediary box performing functions apart from normal, standard functions of an IP router on the data path between a source host and destination host"

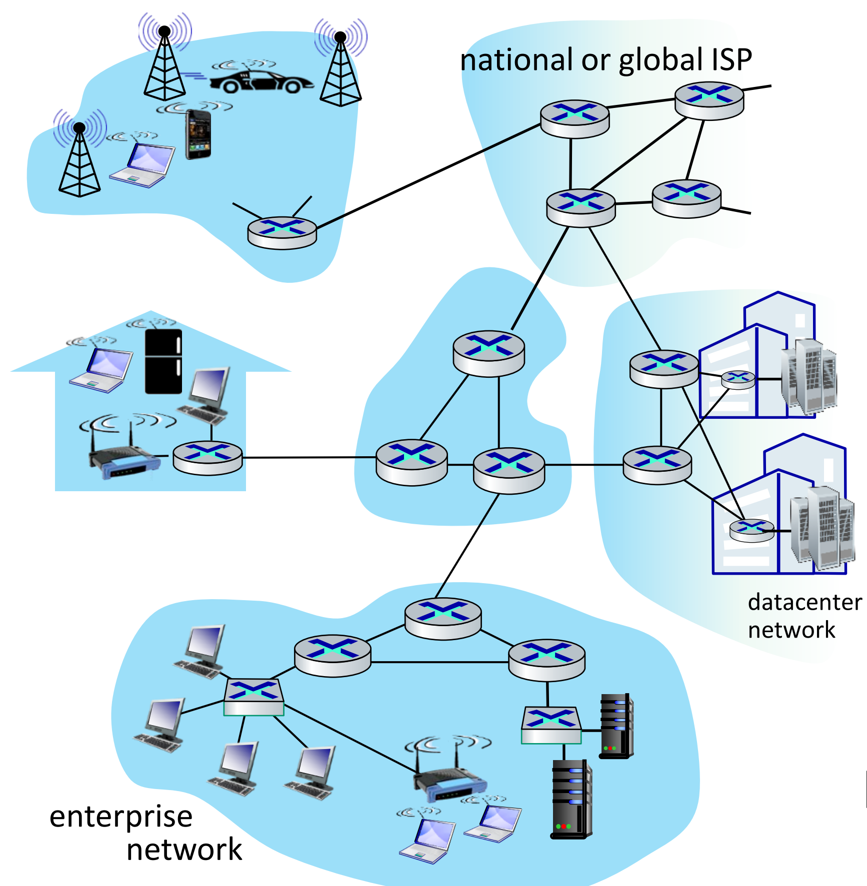
Middleboxes

- Operate at network layer but have functions different from routers
- Examples
  - NATs
  - Firewalls
  - Load balancers
  - Tunneling

Middleboxes everywhere!

NAT: home, cellular, institutional

Application-specific: service providers, institutional, CDN



Firewalls, IDS: corporate, institutional, service providers, ISPs

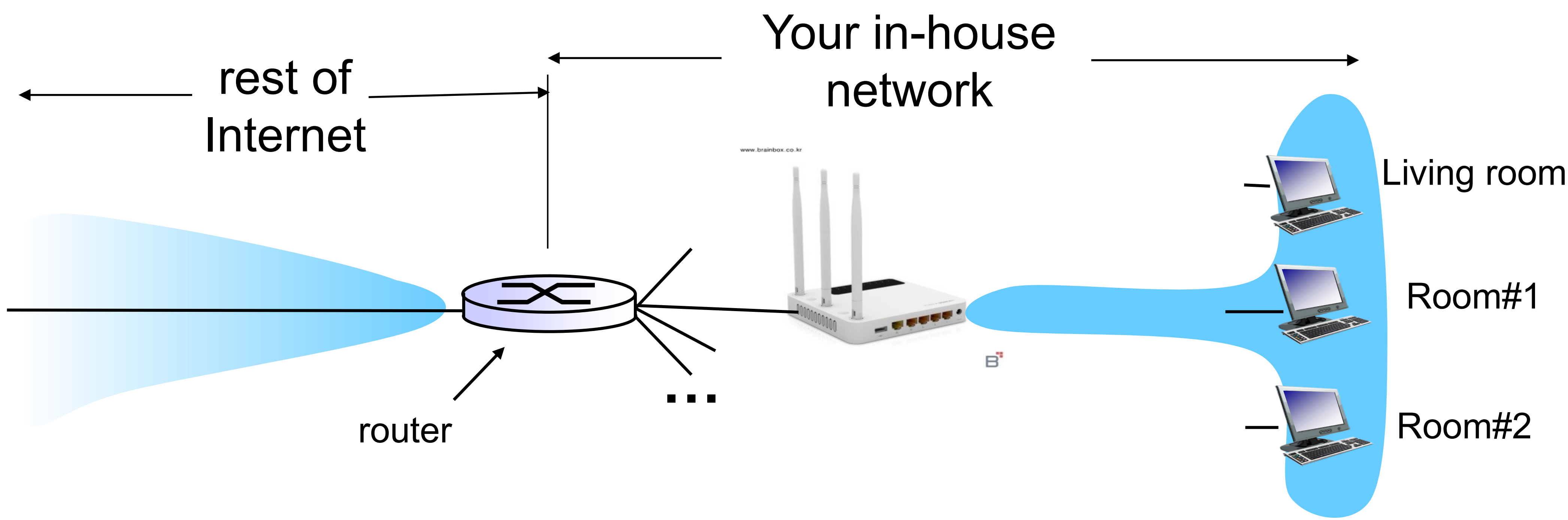
Load balancers: corporate, service provider, data center, mobile nets

Caches: service provider, mobile, CDNs

Network Address Translation (NAT)
(Ch. 4.3.4)

가천대학교
Gachon University

Question



Q: How many IPs are used in your home?
If more than 1, do you pay your ISP (e.g., KT, LG U+, SKT, Cable, …)
more for using all of them ?

History of NATs

- IP address space depletion
  - Clear in early 90s that 2^{32} addresses not enough
  - Work began on a successor to IPv4 (*aka* IPv6)
- In the meantime…
  - Share addresses among numerous devices
  - … without requiring changes to existing hosts
- Meant as a short-term remedy
  - Now: NAT is widely deployed, much more than IPv6

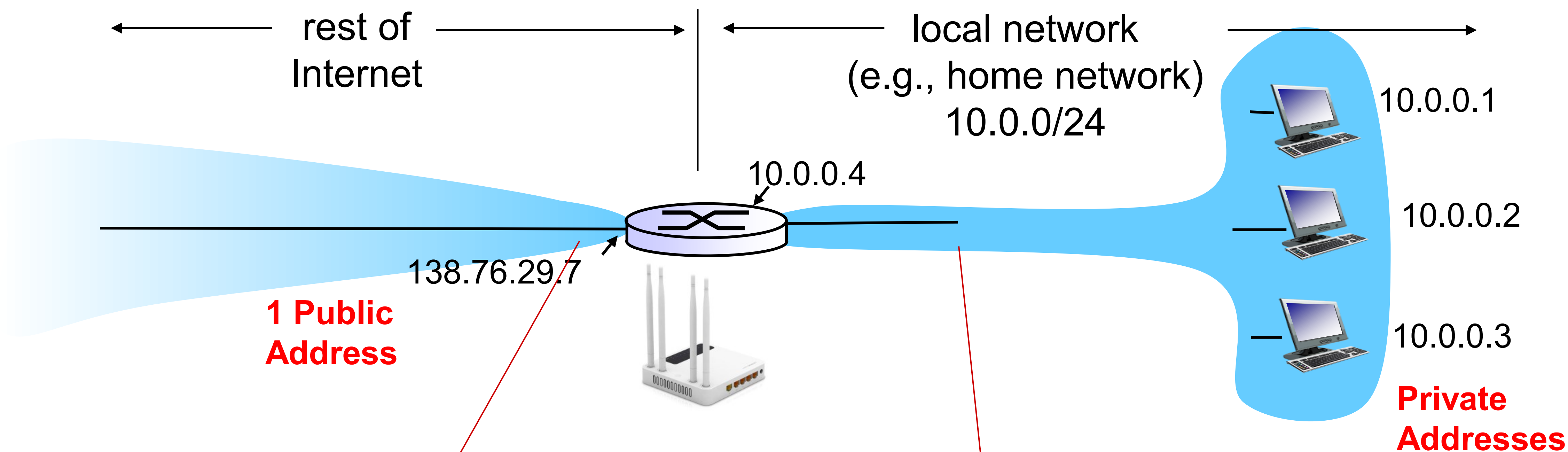
NAT: network address translation

motivation: local network uses just one IP address as far as outside world is concerned:

- just one ***public*** *(or **external**) IP address* for all devices inside local network
- devices inside the local network uses ***private*** *(or **internal**) IP addresses* provided by NAT

Summary: By using NAT, many devices can connect to the Internet by using just <u>one</u> public IP address!

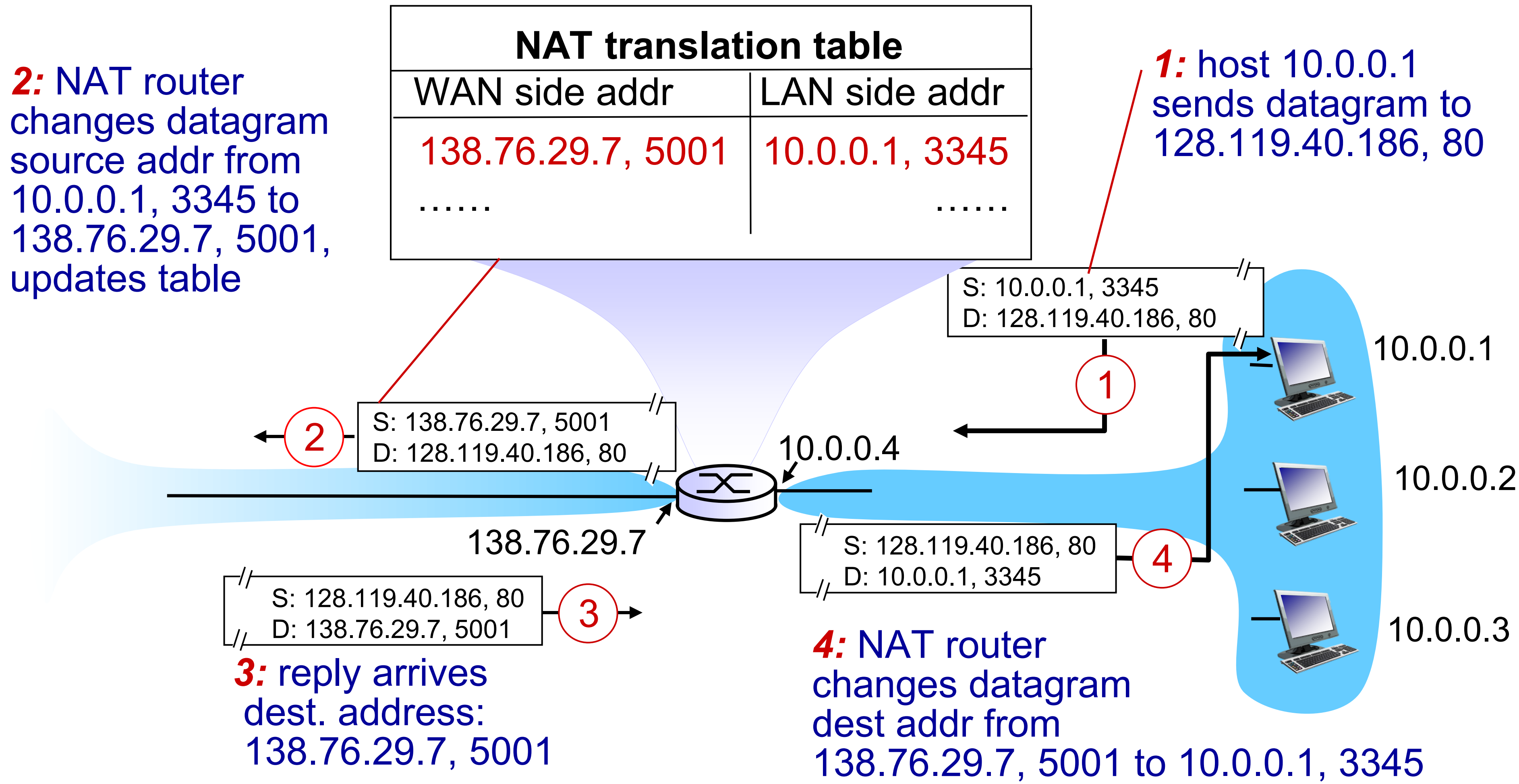
NAT: network address translation



all datagrams *leaving* (*arriving*) local network have *same* single source (destination) NAT IP address: 138.76.29.7, different source port numbers

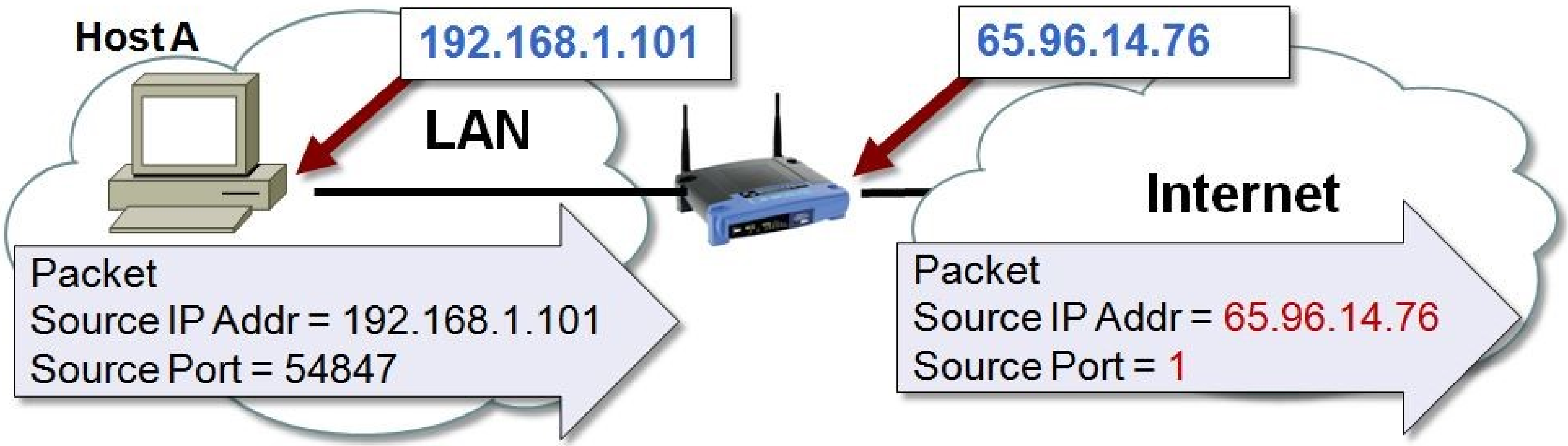
datagrams with source or destination in this network have 10.0.0/24 address for source, destination (as usual)

NAT: network address translation



Another NAT Example

See for yourself



NAT Translation Table

	Local IP Address	Source Port #		Internet IP Address	Source Port #
process X, Host A	192.168.1.101	54,847	=	65.96.14.76	1
Host B	192.168.1.103	24,123	=	65.96.14.76	2
process Y, Host A	192.168.1.101	42,156	=	65.96.14.76	3
Host C	192.168.1.102	33,543	=	65.96.14.76	4

NAT: network address translation

implementation: NAT router must:

- *outgoing datagrams: replace* (source IP address, port #) of every outgoing datagram to (NAT IP address, new port #)

  . . . remote clients/servers will respond using (NAT IP address, new port #) as destination addr

- *remember (in NAT translation table)* every (source IP address, port #) to (NAT IP address, new port #) translation pair

- *incoming datagrams: replace* (NAT IP address, new port #) in dest fields of every incoming datagram with corresponding (source IP address, port #) stored in NAT table

NAT: network address translation

- Create an entry upon seeing an outgoing packet
  - Packet with new (source addr, source port) pair
- Eventually, need to delete entries to free up #'s
  - When? If no packets arrive before a timeout
  - (At risk of disrupting a temporarily idle connection)
- Yet another example of "soft state"
  - i.e., removing state if not refreshed for a while

Private IP address

- Some parts of IP addresses have been reserved for private IP addresses (*<u>not</u>* used for global IP address)

IP address range	number of addresses	host id size
24-bit block	10.0.0.0 - 10.255.255.255	10.0.0.0/8 (255.0.0.0)
20-bit block	172.16.0.0 - 172.31.255.255	172.16.0.0/12 (255.240.0.0)
16-bit block	192.168.0.0 - 192.168.255.255	192.168.0.0/16 (255.255.0.0)

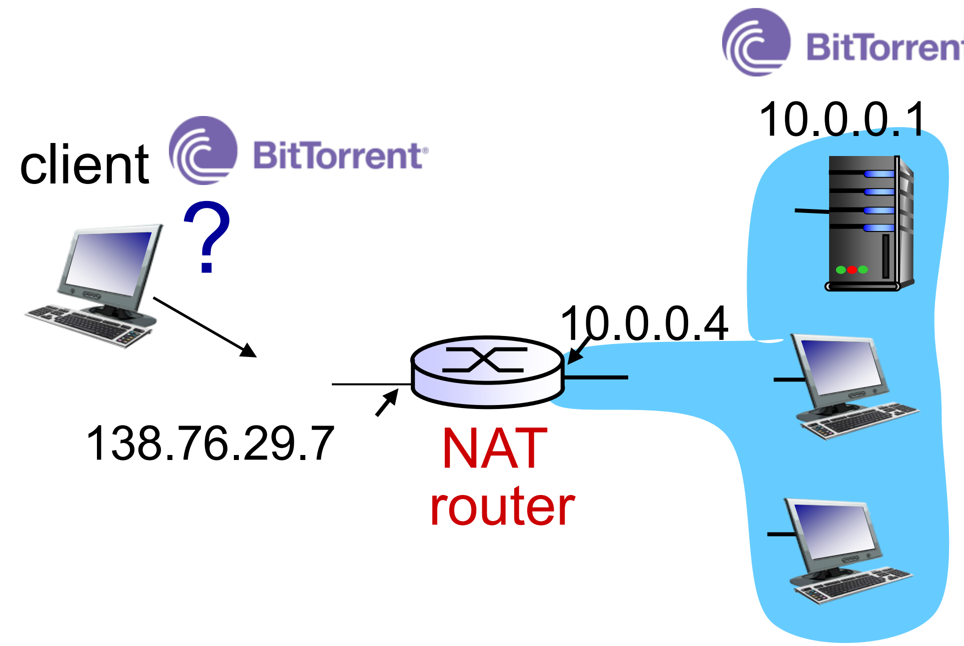
"RFC 1918: Address Allocation for Private Internets". IETF. February 1996. p. 4.

NAT discussion

- 16-bit port-number field:
  - Up to 2^{16}=65,536 simultaneous connections with a single global IP address!
- Usually NAT/DHCP work together in a local network
  - e.g., “GC_free_Wi-Fi” gives you a private addresses (via NAT) automatically by using DHCP
    - You don’t have to configure your IP address, etc.
    - You don’t need a globally unique IP address (only locally unique private address)
- devices inside local network not explicitly addressable, visible by outside world (a security plus)
  - Any problems?

NAT traversal problem

- client wants to connect to server with address 10.0.0.1
  - server address 10.0.0.1 local to LAN (client can't use it as destination addr)
  - only one externally visible NATed address: 138.76.29.7
- Solutions
  - Universal Plug and Play (UPnP) used in BitTorrent
  - Relaying used in Skype
  - Port Forwarding, DMZ, ...

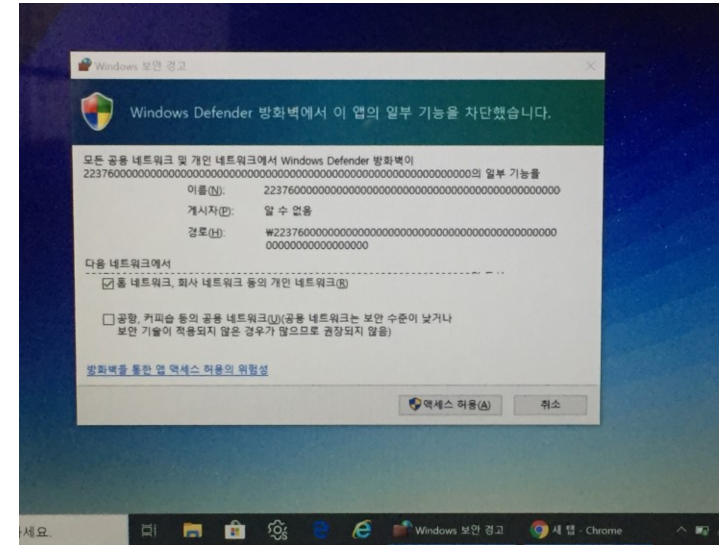


Firewalls
(Ch. 8.9.1)

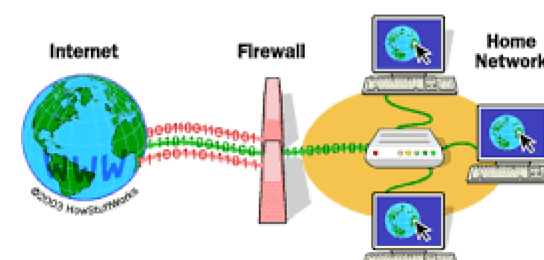
Question

What is a firewall (방화벽)?

Host-based Firewall



Network-based Firewall

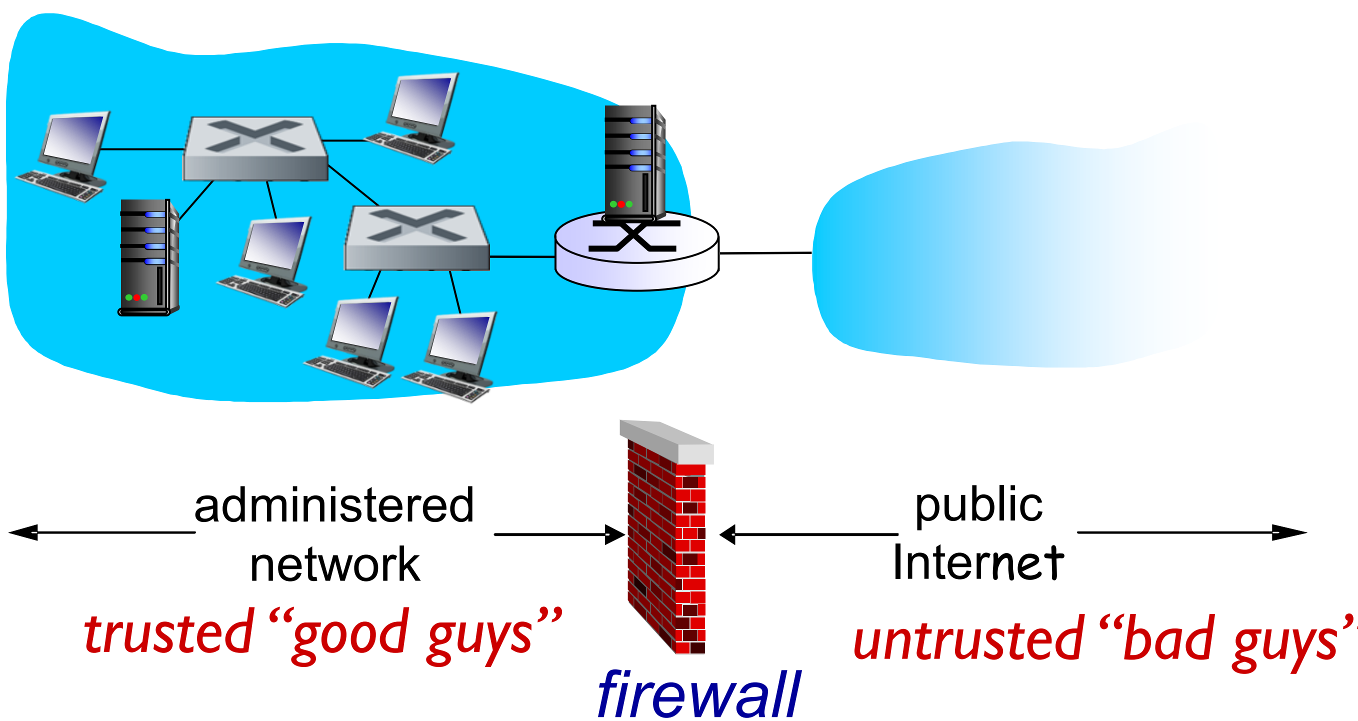


가천대학교 Gachon University

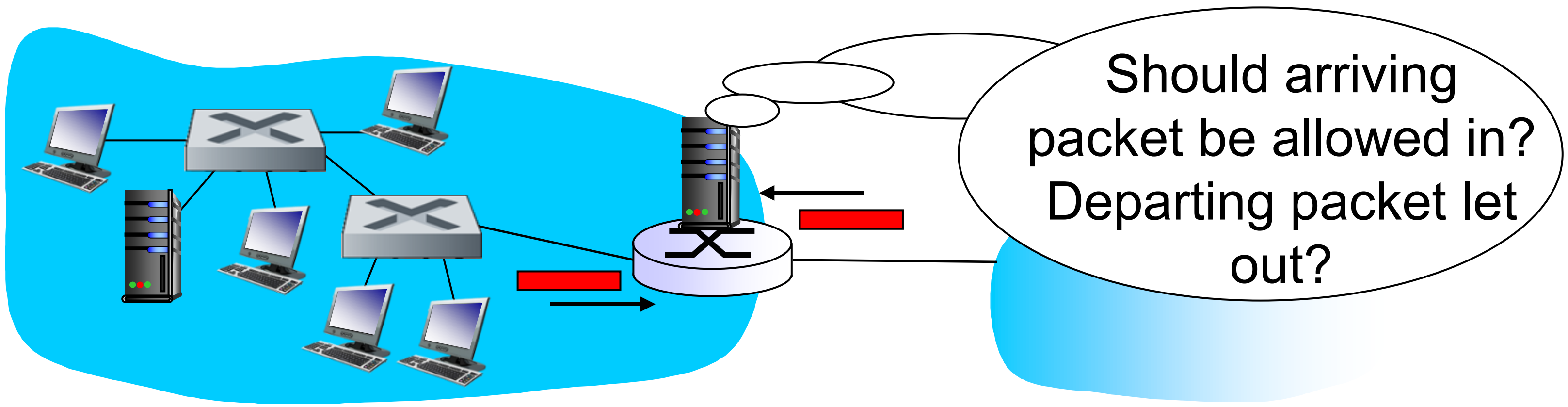
Firewalls

firewall

isolates organization's internal net from larger Internet, allowing some packets to pass, blocking others



Firewalls: Packet Filtering



- internal network connected to Internet via *network firewall*
- router *filters packet-by-packet,* decision to forward/drop packet based on:
  - source IP address, destination IP address
  - TCP/UDP source and destination port numbers
  - ...


THE GREAT (FIRE)WALL OF CHINA
Top #10 Websites Blocked in China
>60 million users
friendster
75 million users
>200 million users
86 million users
ebay
Plenty
You Tube
48.2 million users
skype
>560 million users
IMDb
Plenty
>25 million users
>500 million users

Chapter 4: outline

4.1 Overview of Network layer
- data plane
- control plane

4.2 What's inside a router

4.3 IP: Internet Protocol
- datagram format
- fragmentation
- IPv4 addressing
- network address translation
- IPv6

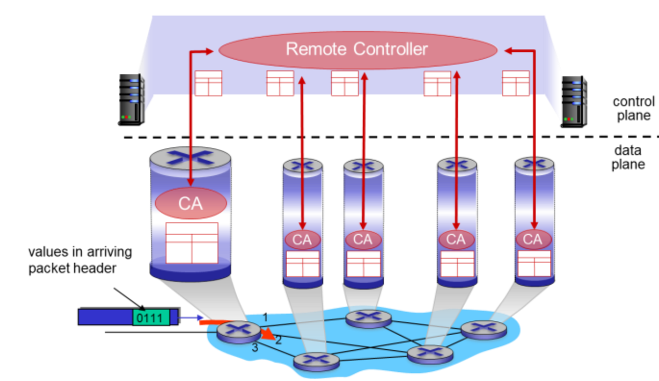
4.4 Generalized Forward and SDN
- match
- action
- OpenFlow examples of match-plus-action in action

Generalized Forwarding and SDN

- Proliferation of middleboxes, and packet switches (e.g., routers)
  - Each with own specialized hardware, software
  - Complicated and hard to manage
- Need a unified approach: Software Defined Networking (SDN)
  - A generic approach that can incorporate all the middlebox, switch, router functions

Logically centralized control plane

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



Generalized Forwarding and SDN

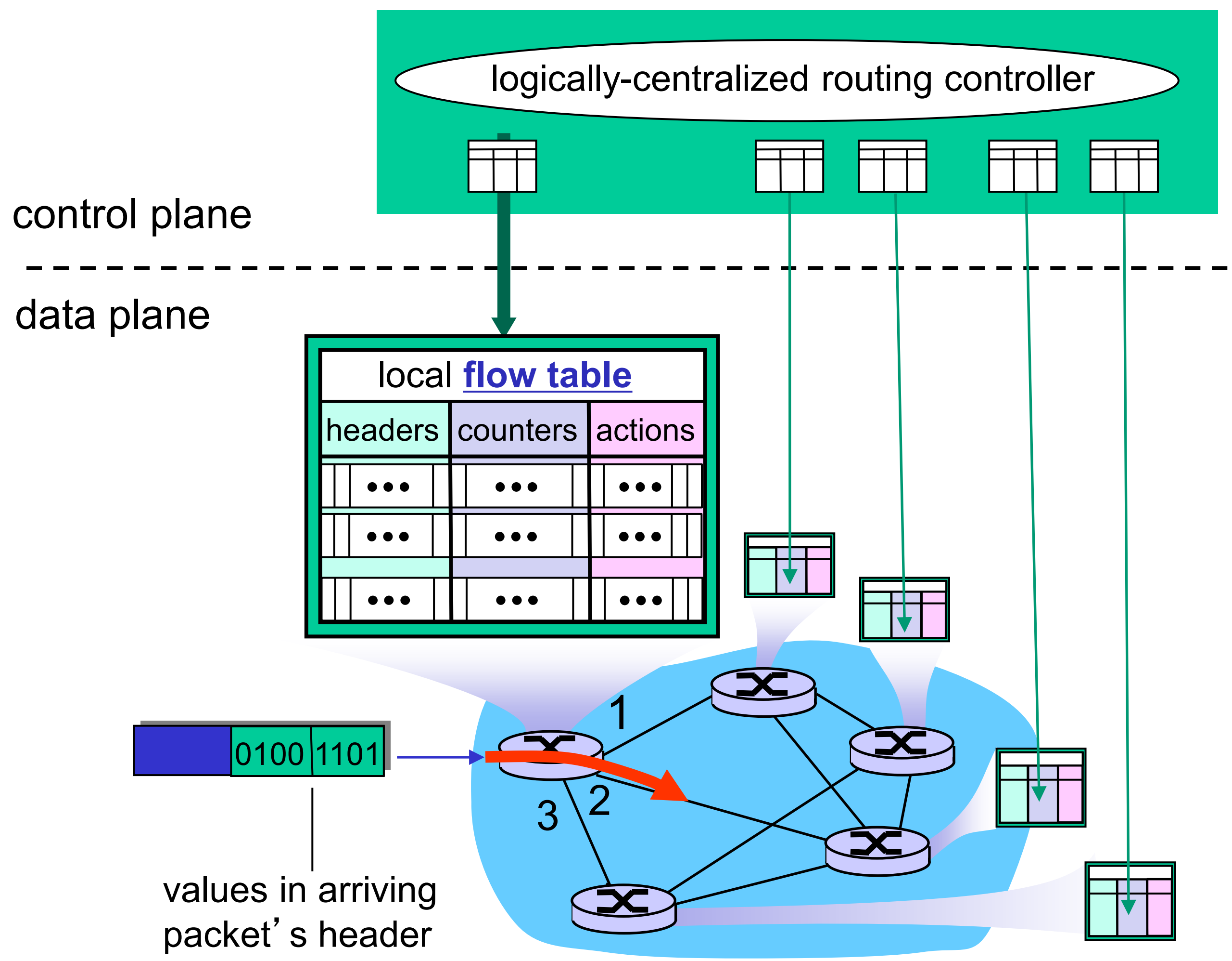
- Basics
  - Recall destination based IP forwarding
    - Match: Look up destination IP address in forwarding table
    - Action: Send packet to specified output port
  - *Generalized* "match-plus-action" approach
    - Match: multiple header fields (for different protocols at different layers in the protocol stack
    - Action:
      - Forwarding the packet to one output port (destination-based forwarding)
      - rewriting header values (NAT)
      - purposefully blocking/dropping a packet (firewall)
- **OpenFlow 1.0** standard

OpenFlow

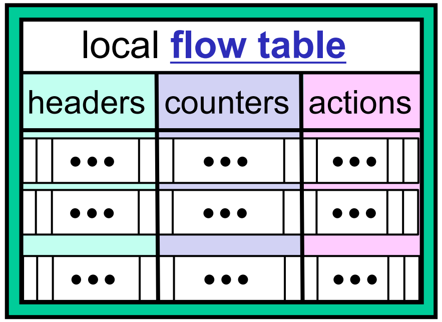
가천대학교
Gachon University

Generalized Forwarding and SDN

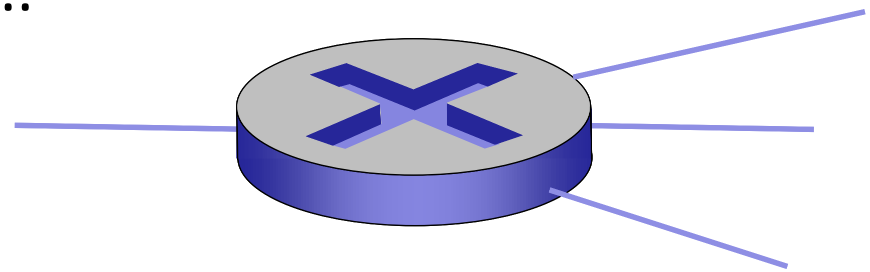
Each router contains a *flow table* that is computed and distributed by a *logically centralized routing controller*



Flow Table in OpenFlow



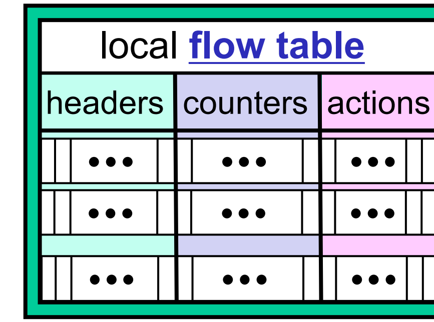
- *A* set of ***header field*** values
  - match values in packet header fields
- A set of ***counters***
  - #packets matched to flow table entries
  - #time since the table entry was last updated
- A set of ***actions*** to be taken
  - forward packet to given output port, drop the packet, rewrite selected header fields, ...

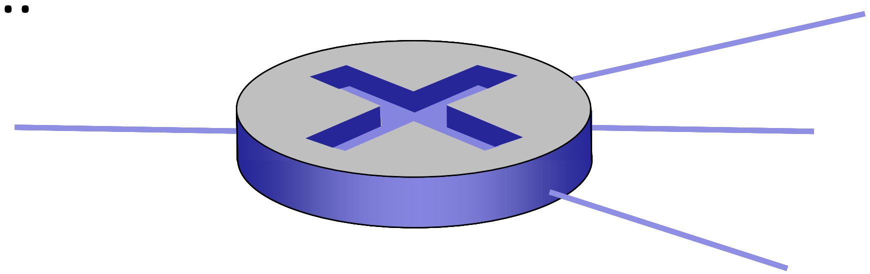


Flow table *in a router (computed and distributed by controller) define router's match+action rules*

Flow Table in OpenFlow

- A set of ***header field*** values
  - match values in packet header fields
- A set of ***counters***
  - #packets matched to flow table entries
  - #time since the table entry was last updated
- A set of ***actions*** to be taken
  - forward packet to given output port, drop the packet, rewrite selected header fields, ...





* : 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

Rule	Action	Stats

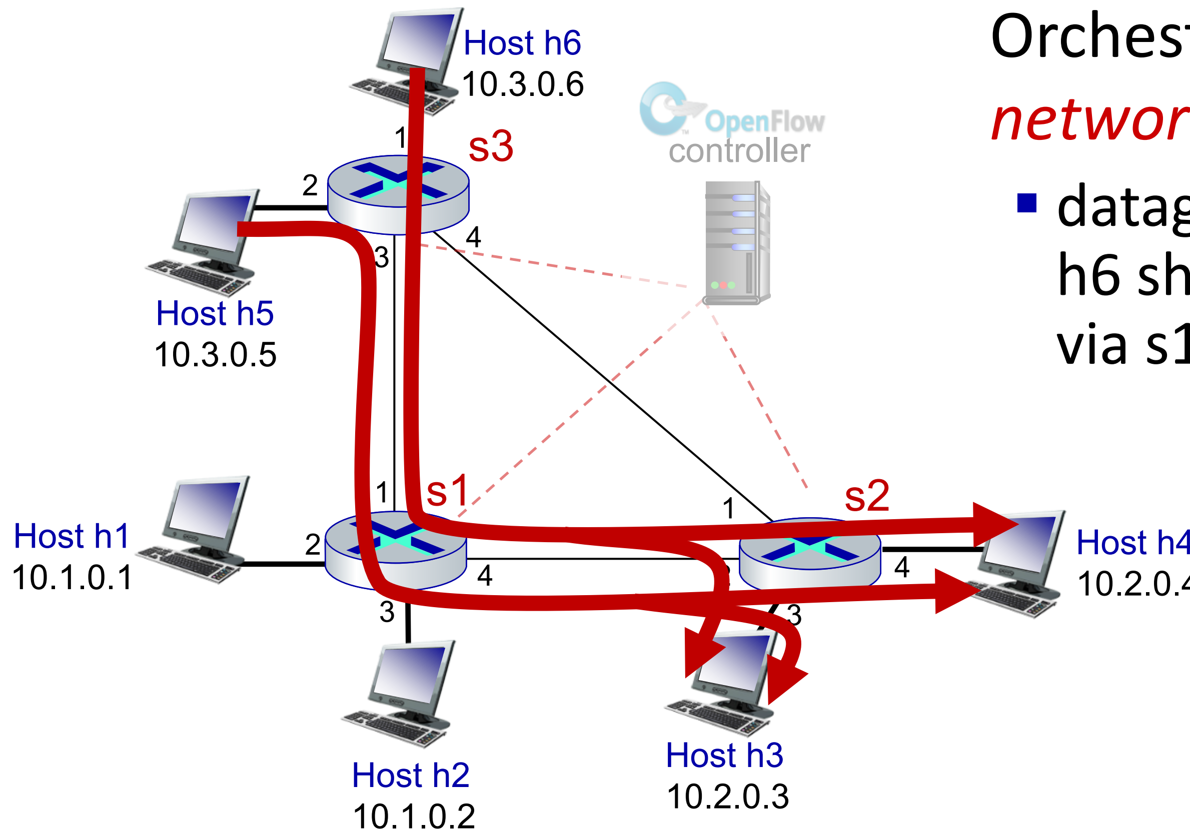
Packet + byte counters

1. Forward packet to port(s)
2. Encapsulate and forward to controller
3. Drop packet
4. Send to normal processing pipeline
5. Modify Fields

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

Link layer (VLAN ID – Eth type) · Network layer (IP Src – IP Prot) · Transport layer (TCP sport – TCP dport)

OpenFlow example

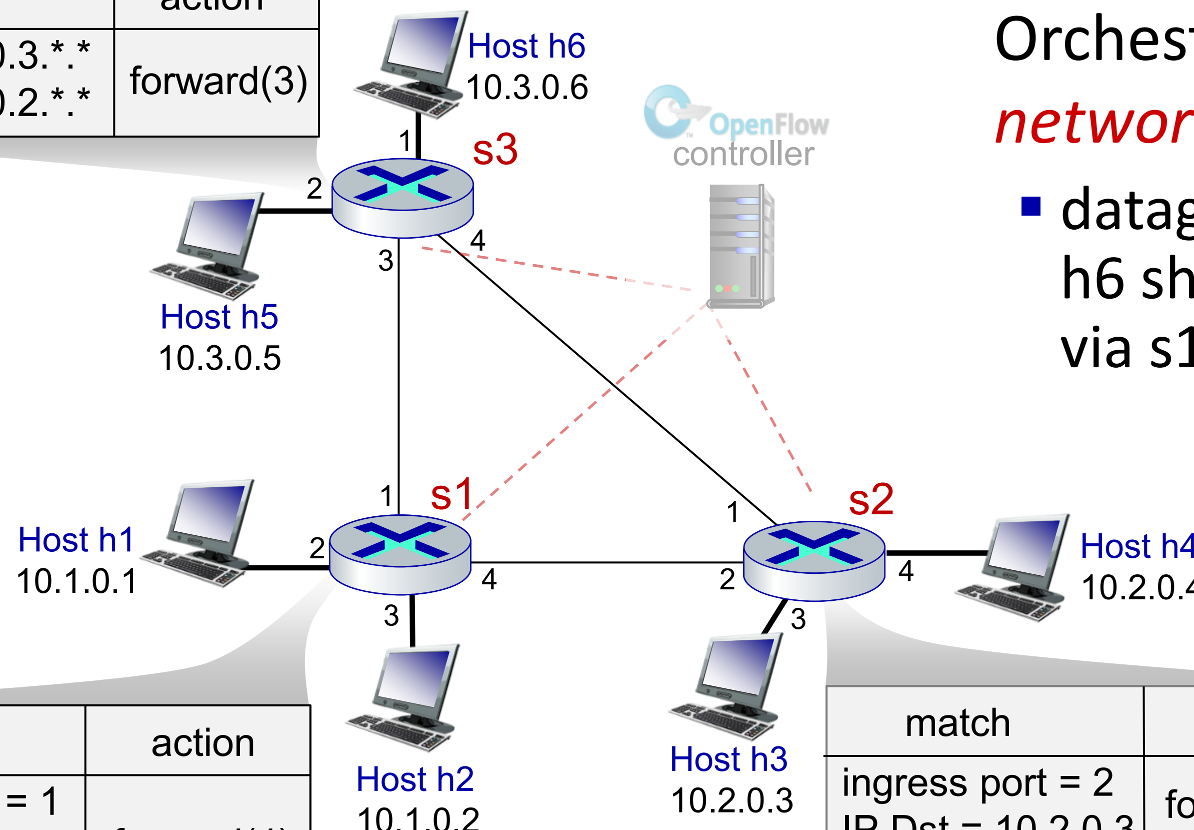


Orchestrated tables can create *network-wide* behavior, e.g.,:

- 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

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	Forward
*	*	*	*	*	*	*	*	*	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	Forward
*	*	*	*	*	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 6

Examples (NAT)

What should be the flow table for the NAT translation table shown below?
(Say that the packet is from WAN side)

NAT translation table

WAN side addr	LAN side addr
138.76.29.7, 5001	10.0.0.1, 3345
……	……

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

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

Chapter 4: *done!*

4.1 Overview of Network layer: data plane and control plane

4.2 What's inside a router

4.3 IP: Internet Protocol

- datagram format
- fragmentation
- IPv4 addressing
- NAT
- IPv6

4.4 Generalized Forward and SDN

- match plus action
- OpenFlow example

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

Answer: by the control plane (next chapter)