ITH508 컴퓨터망

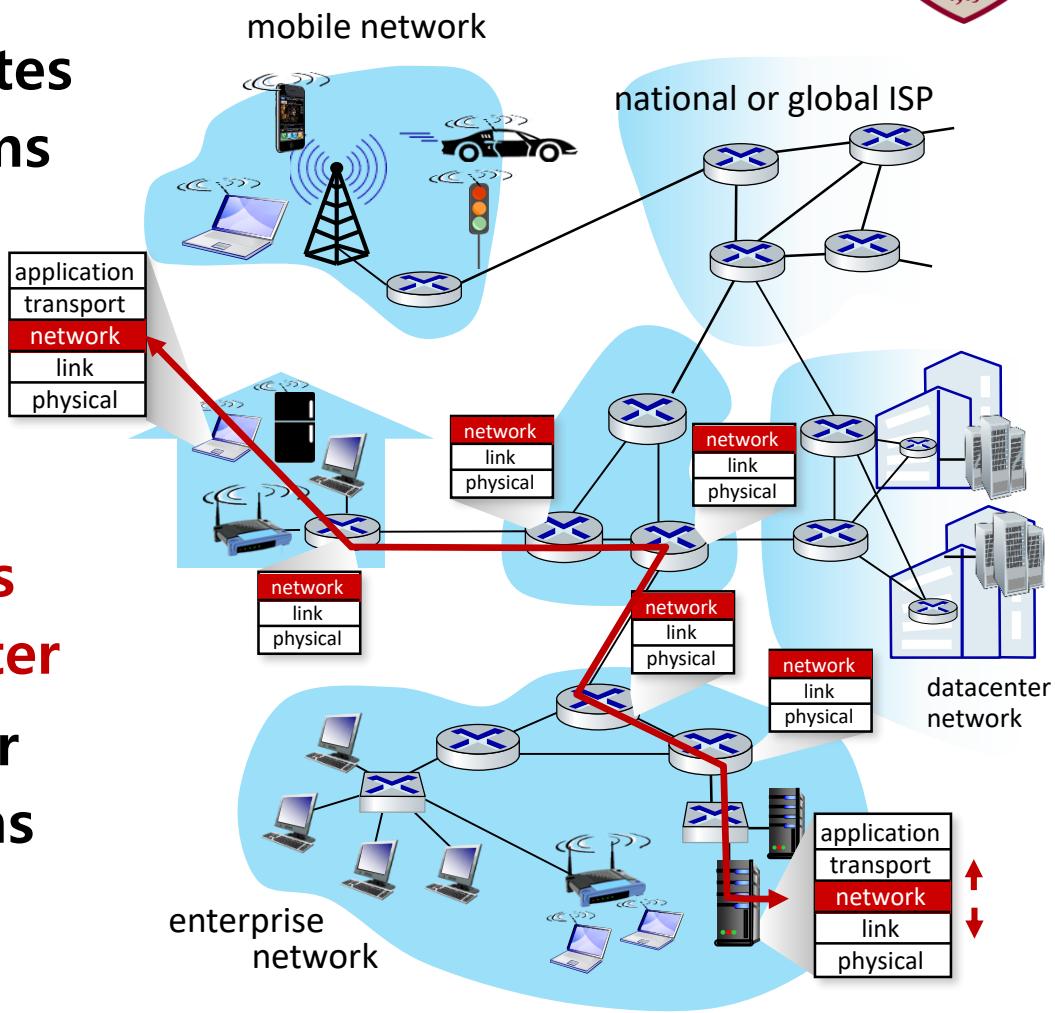
Network Layer Data Plane

Hwangnam Kim
hnkim@korea.ac.kr
School of Electrical Engineering
Korea University

NETWORK LAYER

Network Layer

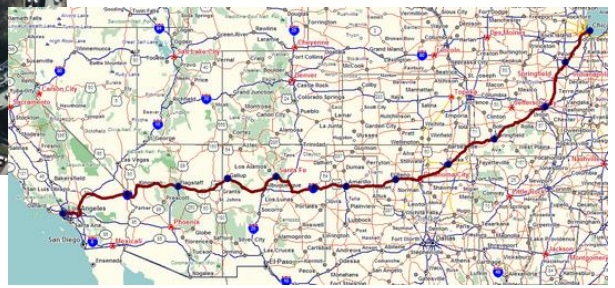
- Sending side encapsulates segments into datagrams
- Receiving side delivers segments to transport layer
- Network layer protocols exist in *every* host, router
- Router examines header fields in all IP datagrams passing through it



Network-Layer Functions

- ***Forwarding***
  - **Move packets** from router's input to appropriate router output
- ***Routing***
  - **Determine route** taken by packets from source to destination
  - *Routing algorithms*

forwarding

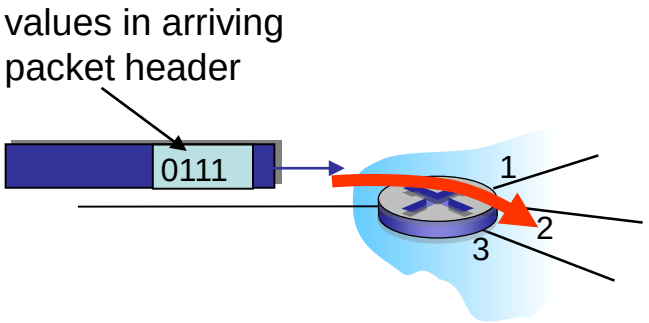


routing

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

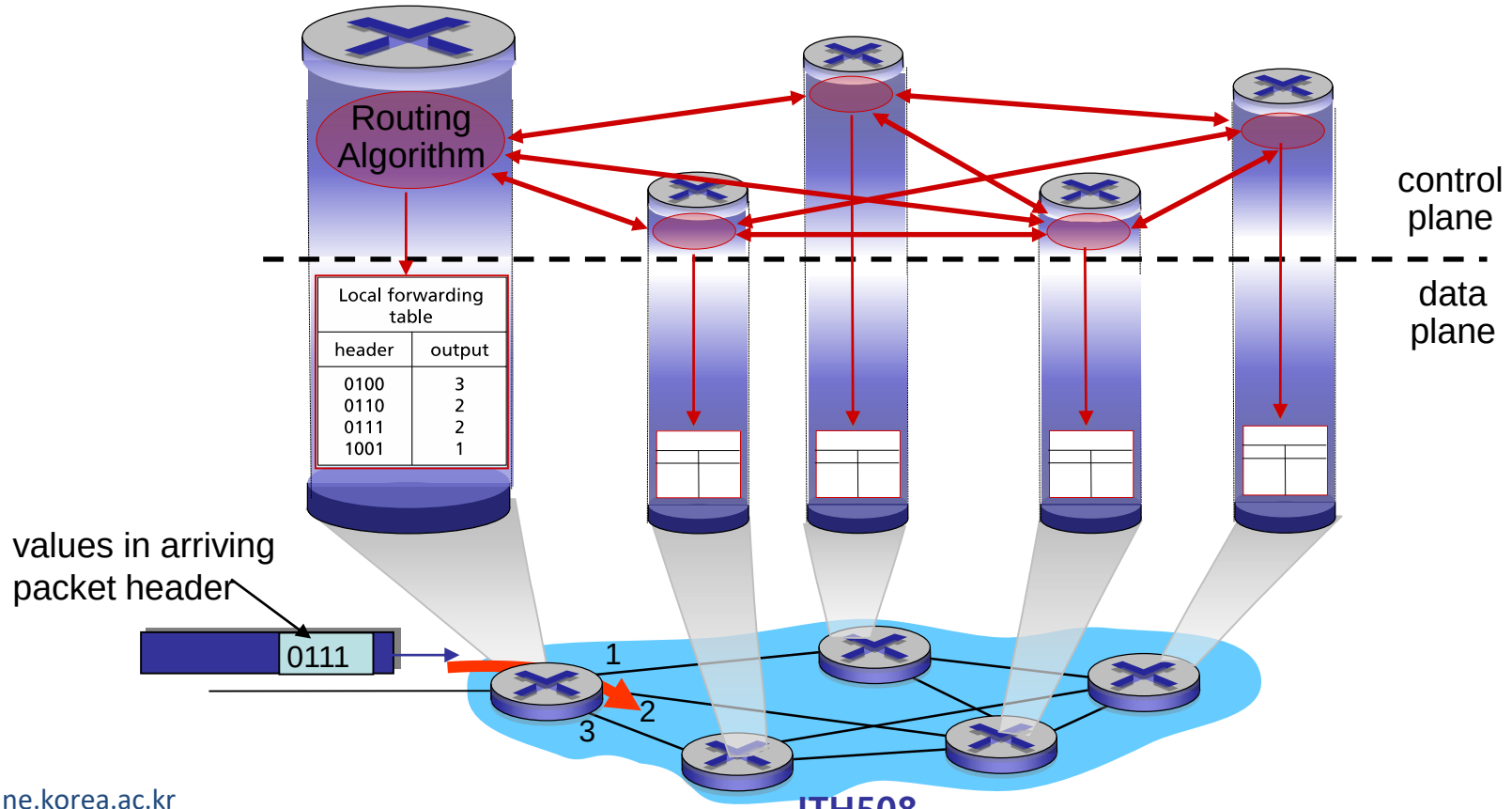


Control plane

- 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**

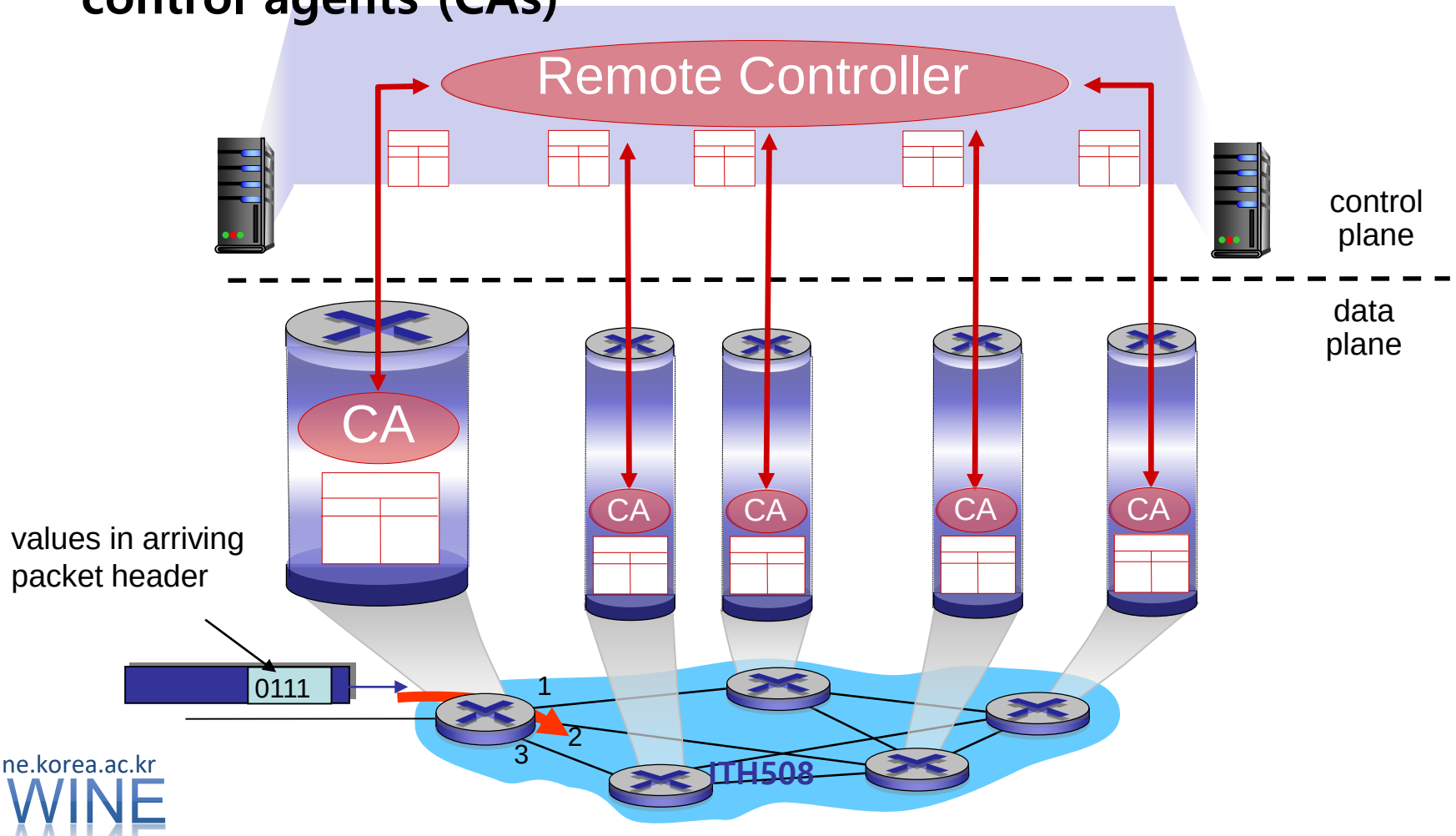
Per-Router Control Plane

- **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)**

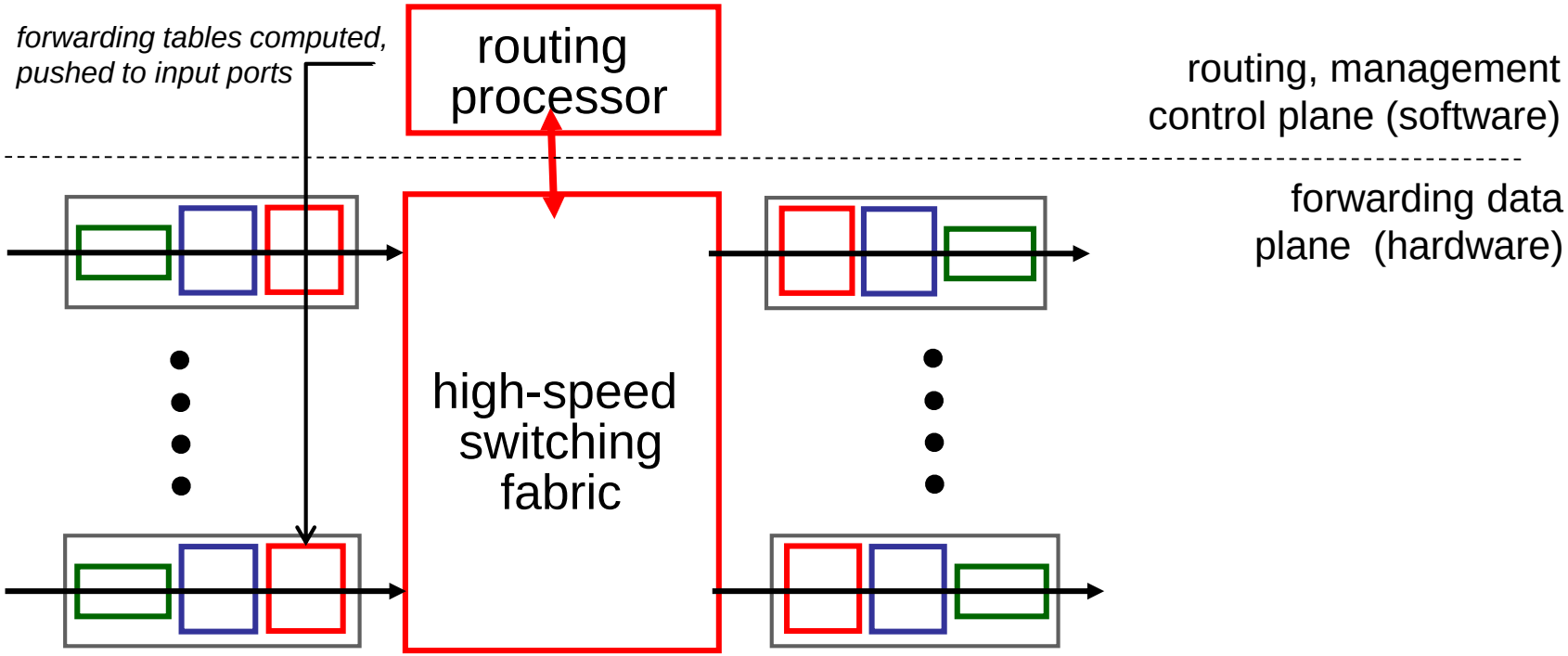


(Switching) Internal Router Architecture

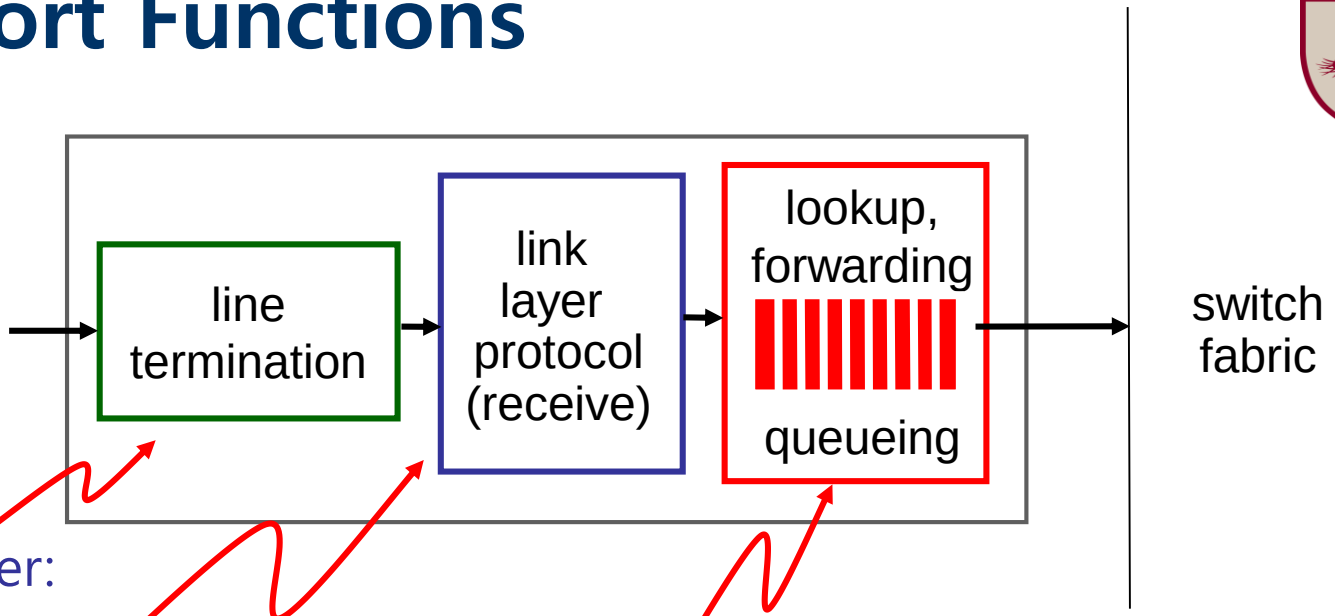
Router Architecture Overview

Two key router functions:

- **Run routing algorithms/protocol (RIP, OSPF, BGP)**
- ***Forwarding* datagrams from incoming to outgoing link**



Input Port Functions



Physical layer:
bit-level reception

Data link layer:
e.g., Ethernet

Decentralized switching:

- **Given datagram destination, lookup output port using forwarding table in input port memory**
- **Goal**
  - Complete input port processing at 'line speed'
- **Queuing: if datagrams arrive faster than forwarding rate into switch fabric**

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!

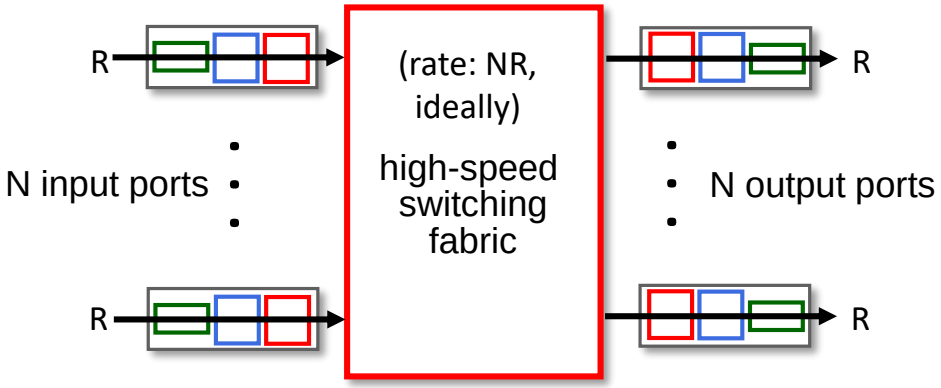
examples:

11001000 00010111 00010110 10100001 which interface?

11001000 00010111 00011000 10101010 which interface?

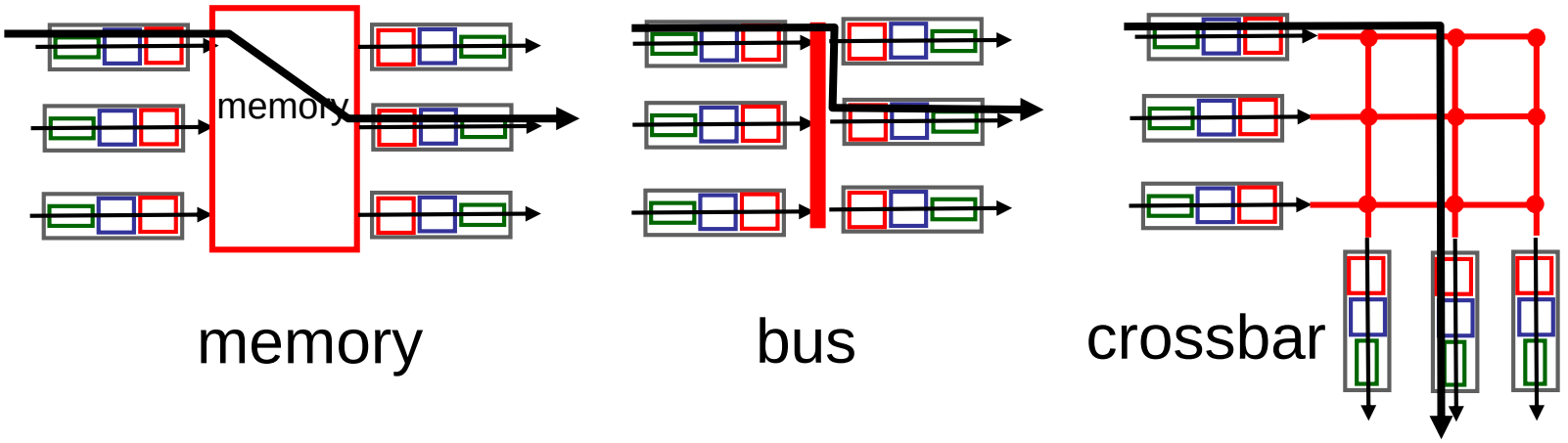
Switching Fabrics

- **Transfer packet from input link to appropriate output link**
- **Switching rate: rate at which packets can be transfer from inputs to outputs**
  - Often measured as multiple of input/output line rate
  - N inputs: switching rate N times line rate desirable



Three Types of Switching Fabrics

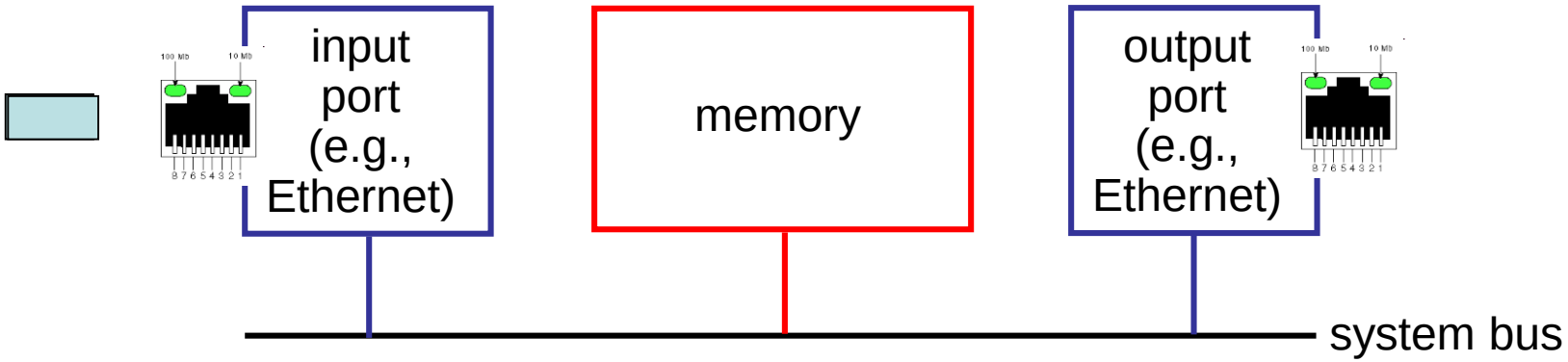
- **Transfer packet from input buffer to appropriate output buffer**
  - **Switching rate**: rate at which packets can be transfer from inputs to outputs
- **Often measured as multiple of input/output line rate**
- **N inputs: switching rate N times line rate desirable**
- **Three types of switching fabrics**



Switching via Memory

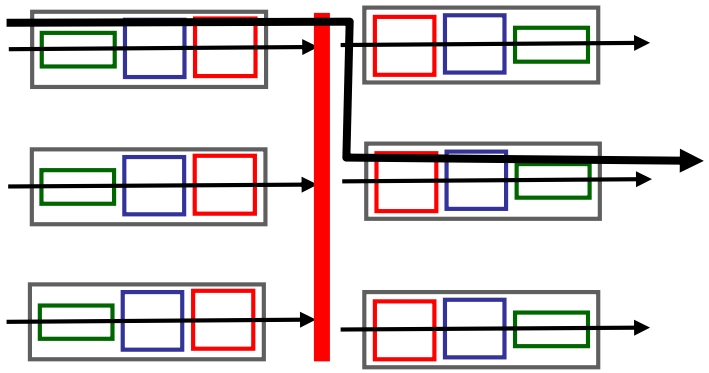
First Generation Routers:

- Traditional computers with switching under direct control of CPU
- Packet copied to system's memory
- Speed limited by memory bandwidth (2 bus crossings per datagram)



Switching via Bus

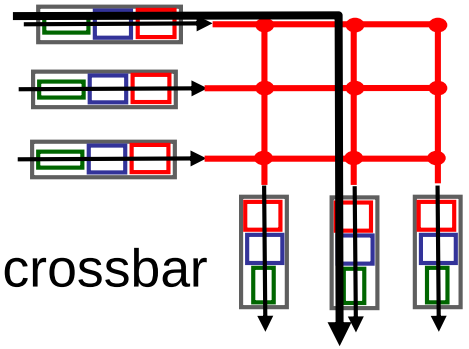
- Datagram from input port memory to output port memory via a shared bus
- **Bus contention:** switching speed limited by bus bandwidth



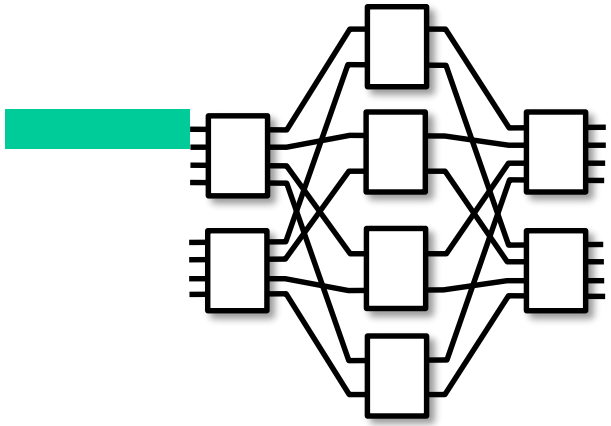
bus

Switching via Interconnection Network

- Crossbar, Clos networks, other interconnection nets initially developed to connect processors in multiprocessor



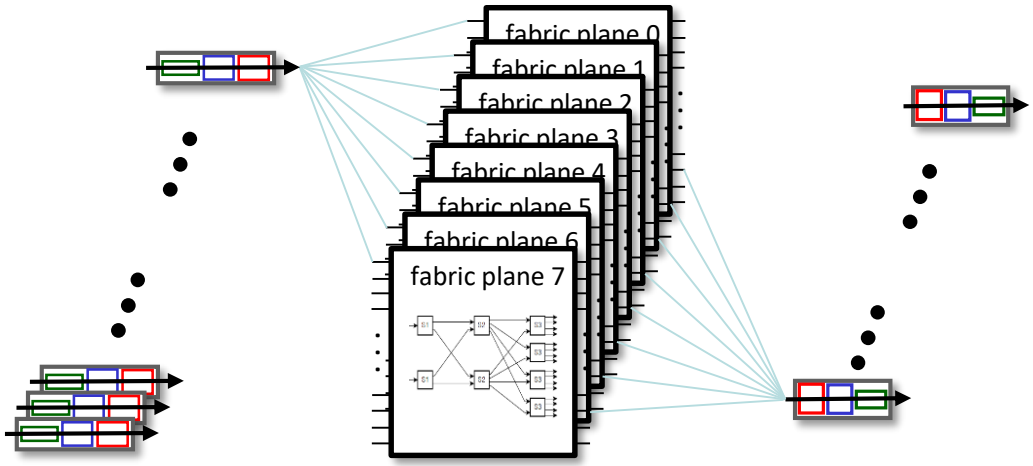
- Multistage switch
  - $n \times n$ switch from multiple stages of smaller switches
- Exploiting parallelism:
  - Fragment datagram into fixed length cells on entry
  - Switch cells through the fabric, reassemble datagram at exit



8x8 multistage switch
built from smaller-sized switches

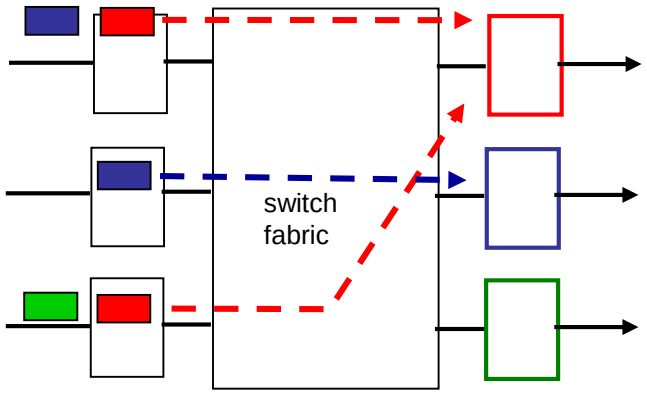
Switching via Interconnection Network

- **Scaling, using multiple switching "planes" in parallel:**
  - Speedup, scaleup via parallelism
- **Cisco CRS router:**
  - Basic unit: 8 switching planes
  - Each plane: 3-stage interconnection network
  - Up to 100's Tbps switching capacity

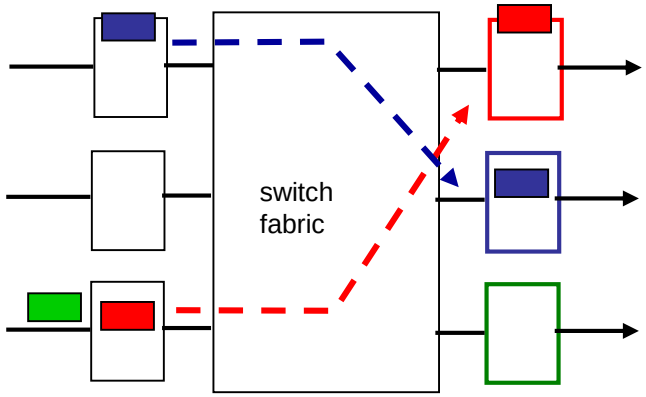


Input Port Queuing

- **Fabric slower than input ports combined**
  → Queueing may occur at input queues
- **Head-of-the-Line (HOL) blocking: queued datagram at front of queue prevents others in queue from moving forward**
- **Queueing delay and loss *due to input buffer overflow!***

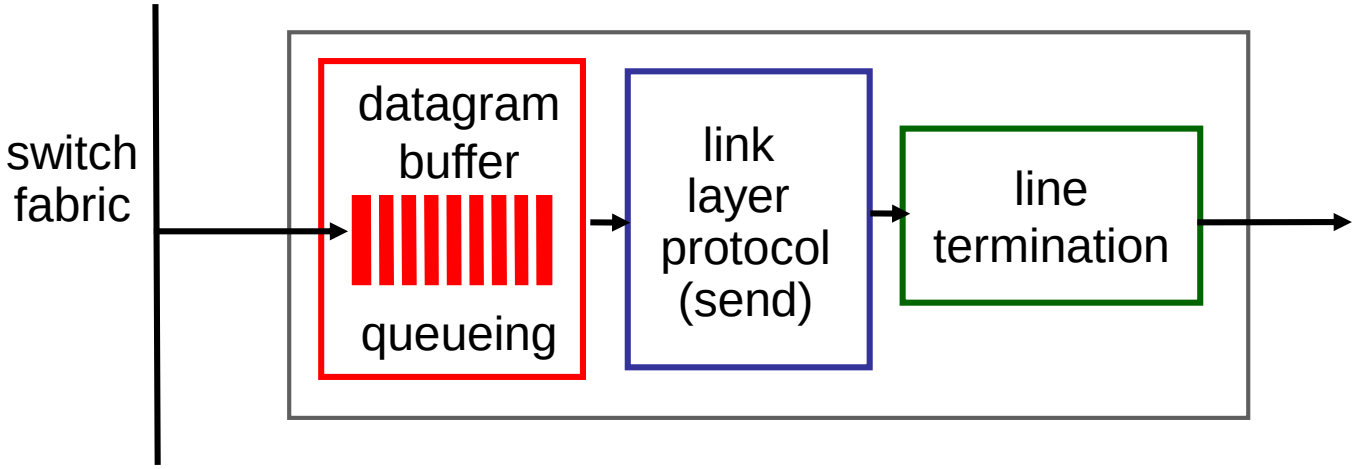


output port contention:
only one red datagram can be transferred.
lower red packet is blocked



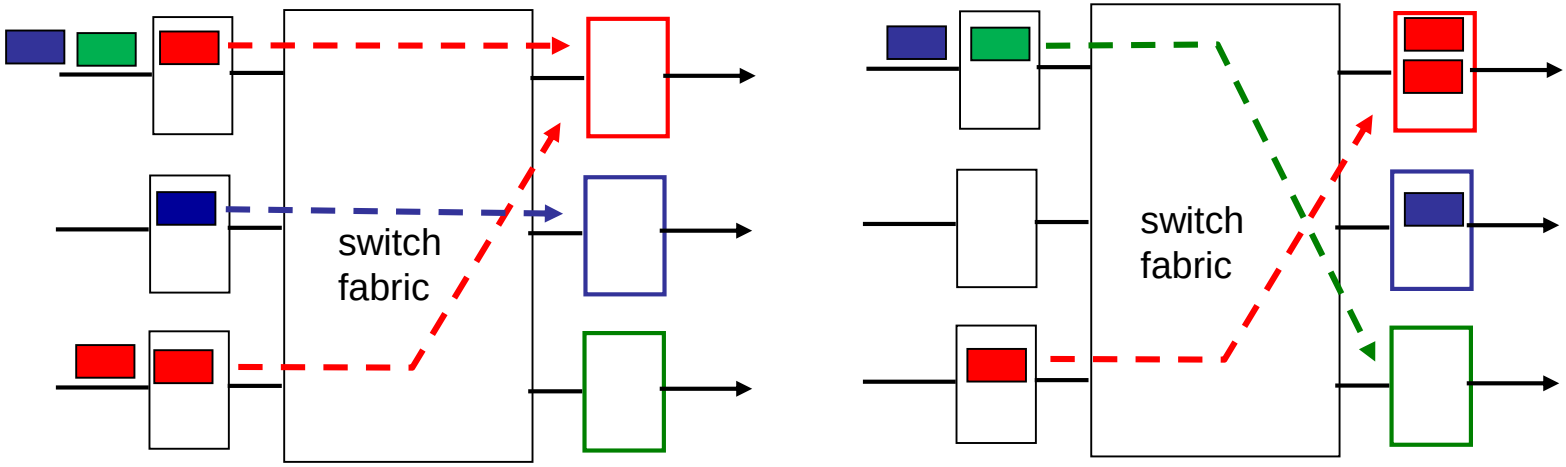
one packet time later: green packet experiences HOL blocking

Output Ports



- **Buffering required when datagrams arrive from fabric faster than the transmission rate.**
  - Datagrams can be lost due to congestion, lack of buffers
- **Scheduling discipline chooses among queued datagrams for transmission**
  - Priority scheduling – who gets best performance, network neutrality

Output Port Queuing



at *t*, packets more from input to output

one packet time later

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

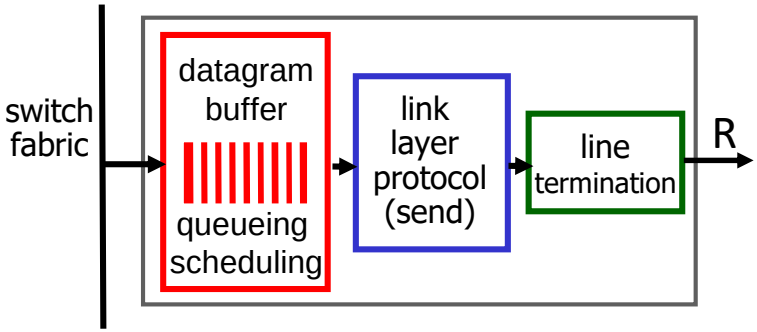
How Much Buffering?

- **RFC 3439 rule of thumb: average buffering equal to "typical" RTT (say 250 msec) times link capacity C**
  - e.g., C = 10 Gpbs link: 2.5 Gbit buffer
- **Recent recommendation: with N flows, buffering equal to**

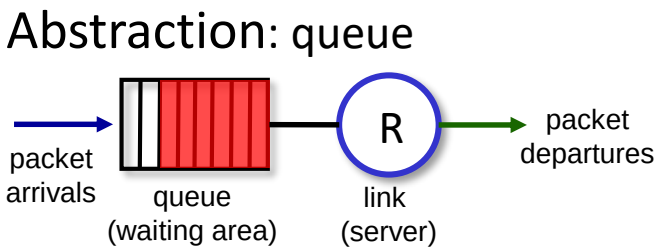
$$\frac{RTT \cdot C}{\sqrt{N}}$$

- **Too much buffering can increase delays (particularly in home routers)**
  - Long RTTs: poor performance for realtime apps, sluggish TCP response
  - Recall delay-based congestion control: "keep bottleneck link just full enough (busy) but no fuller"

Buffer Management

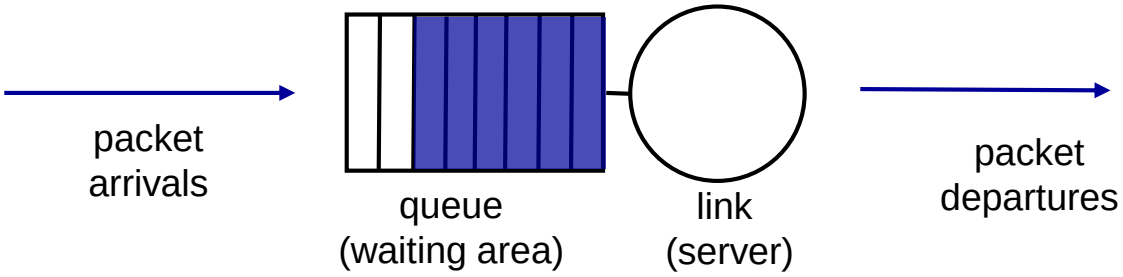


- **Buffer management:**
  - Drop: which packet to add, drop when buffers are full
    - Tail drop: drop arriving packet
    - Priority: drop/remove on priority basis
  - Marking: which packets to mark to signal congestion (ECN, RED)



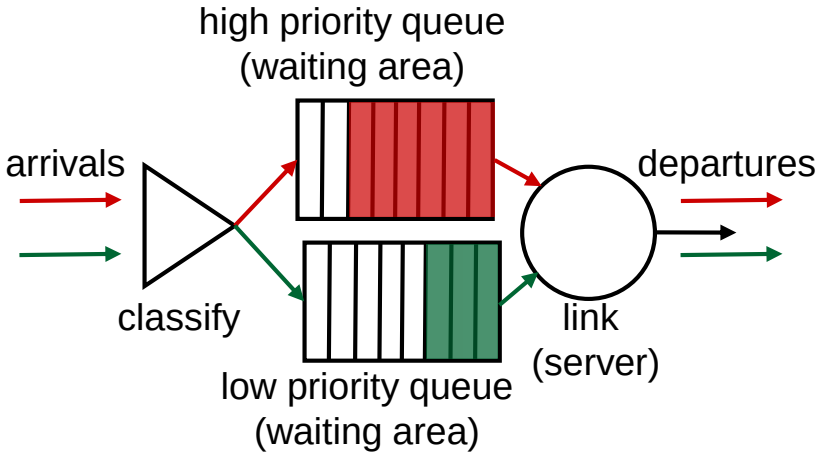
Scheduling

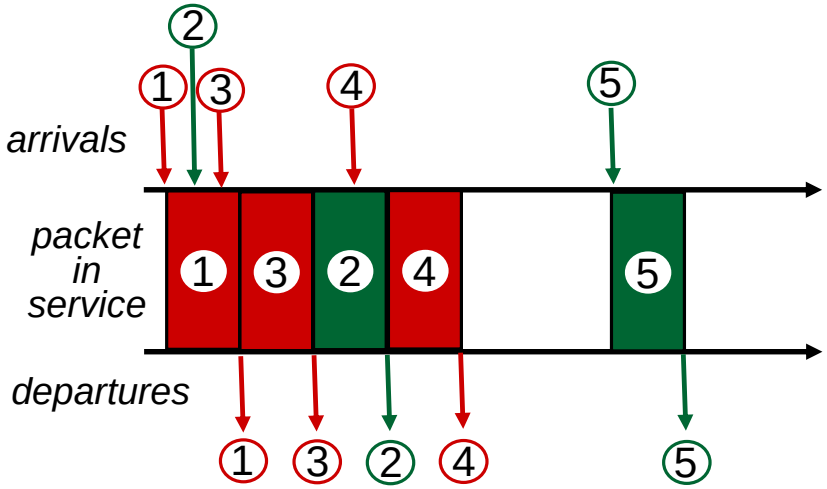
- **Scheduling**
  - Choose next packet to send on link
- **FIFO (first in first out) scheduling: send in order of arrival to queue**
  - Discard policy: if packet arrives to full queue
    - Tail drop: drop arriving packet
    - Priority: drop/remove on priority basis
    - Random: drop/remove randomly



Scheduling Policies: Priority

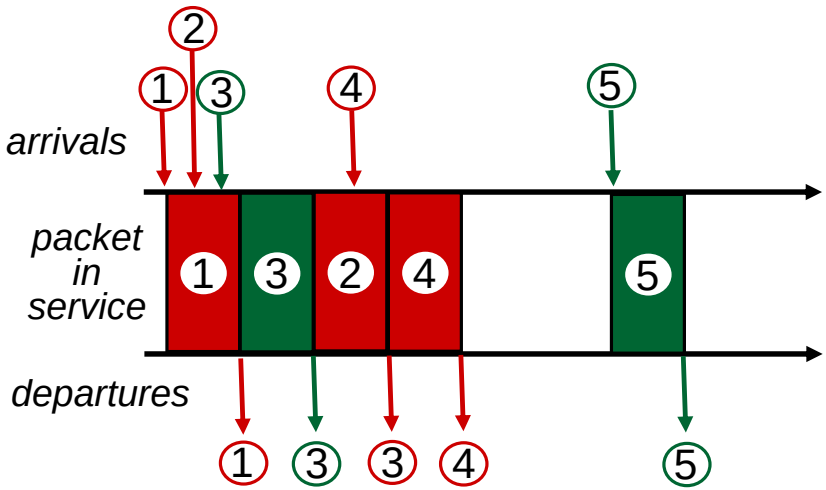
- **Priority scheduling: send highest priority queued packet**
  - Arriving traffic classified, queued by class
    - Any header fields can be used for classification
  - Multiple classes, with different priorities
- **Send packet from highest priority queue that has buffered packets**
  - FCFS within priority class

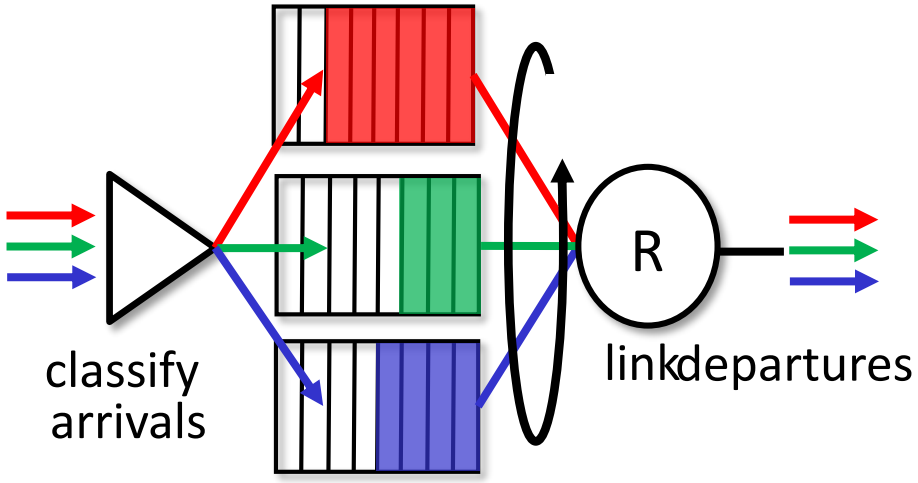




Scheduling Policies

- **Round Robin (RR) scheduling:**
  - Multiple classes
  - Server cyclically, repeatedly scans class queues, sending one complete packet from each class (if available) in turn

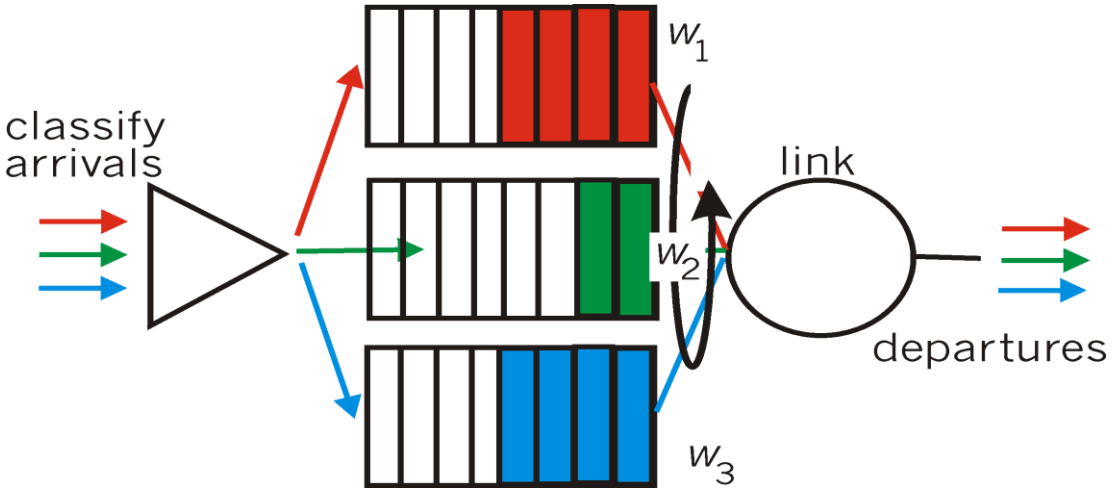




Scheduling Policies

- **Weighted Fair Queuing (WFQ):**
  - Generalized Round Robin
  - Each class gets weighted amount of service in each cycle
    - each class, *i*, has weight, w_i, and gets weighted amount of service in each cycle:

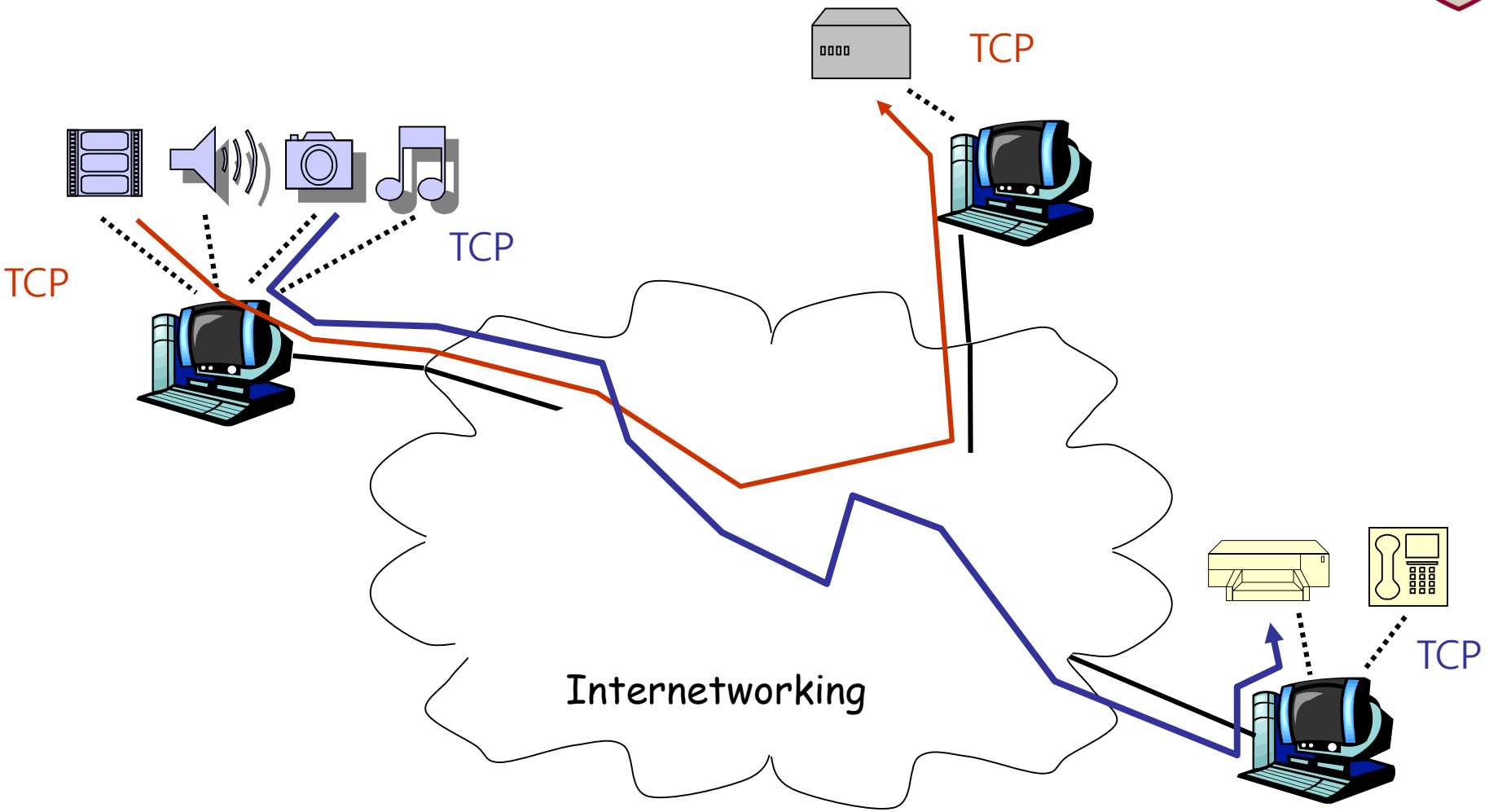
$$\frac{w_i}{\sum_j w_j}$$



IP Protocol

INTERNETWORKING

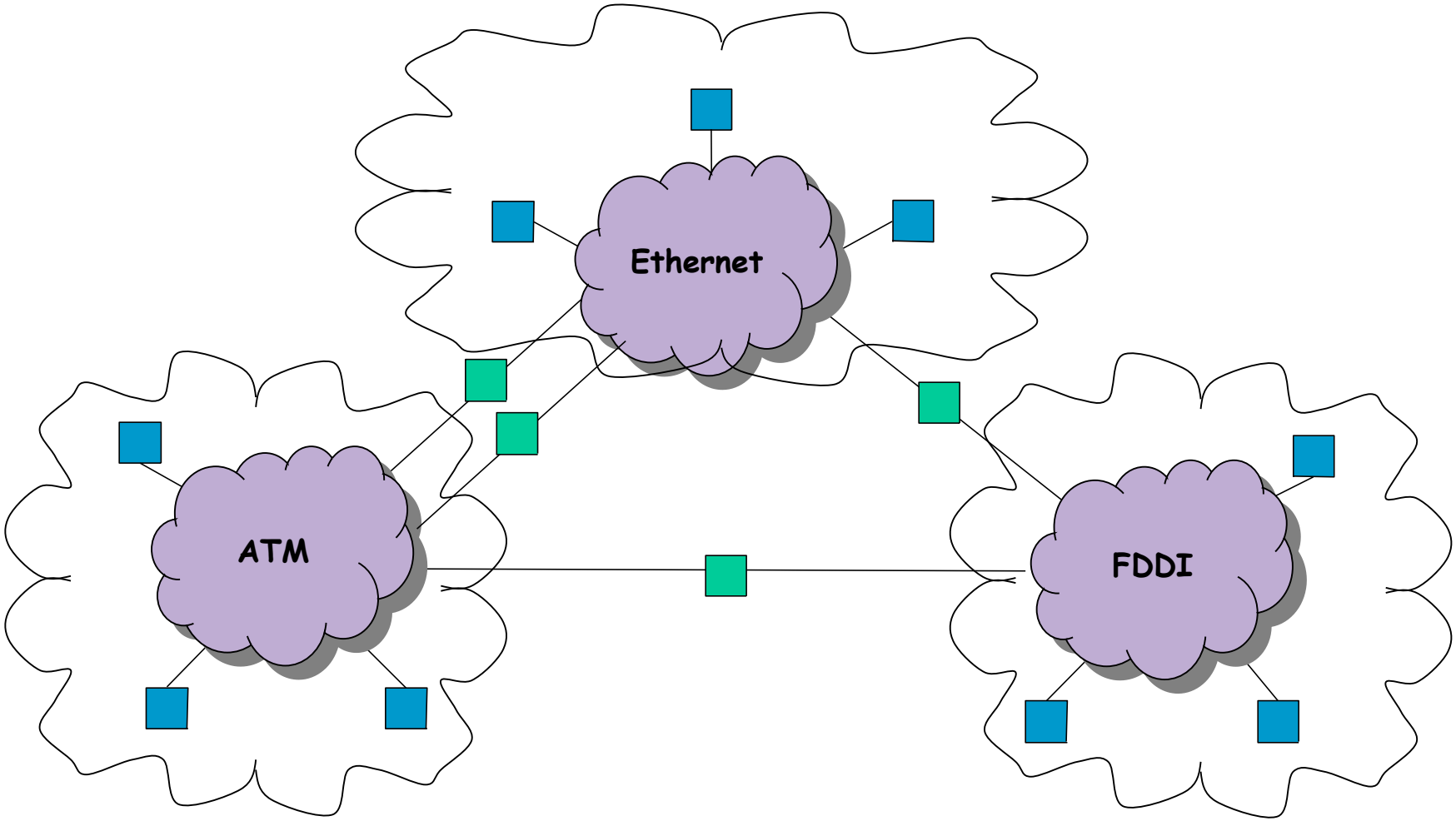
Internetworking Concept



Internetworking Basic

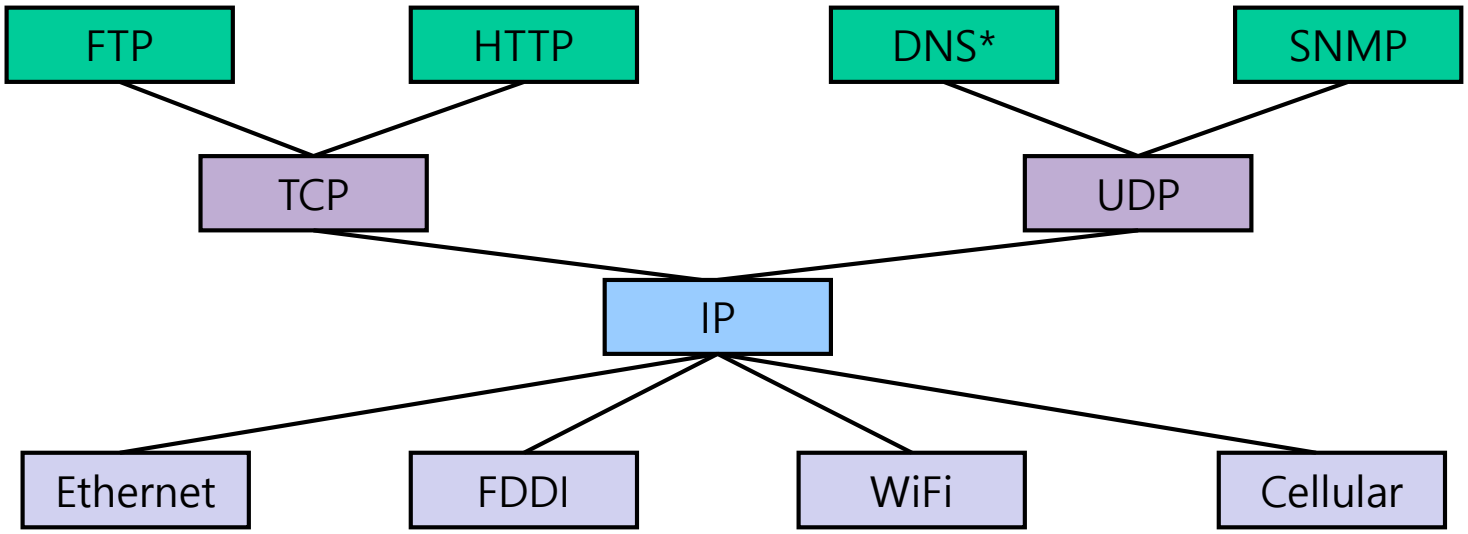
- **What is an internetwork**
  - Illusion of a single (direct link) network
  - Built on a set of **distributed heterogeneous networks**
  - Abstraction typically supported by software
- **Properties**
  - Supports **heterogeneity**
    - **Hardware, OS, network type, and topology** independent
  - Scales to global connectivity
- **The Internet is the specific global internetwork that grew out of ARPANET**

Internetworking



Internet Protocol (IP)

- **Network-level protocol for the Internet**
- **Operates on all hosts and routers**
  - Routers are nodes connecting distinct networks to the Internet



IP Role

- Overview of message transmission
- Host addressing and address translation
- Datagram forwarding
- Fragmentation and reassembly
- Error reporting/control messages
- Dynamic configuration
- Protocol extensions through tunneling

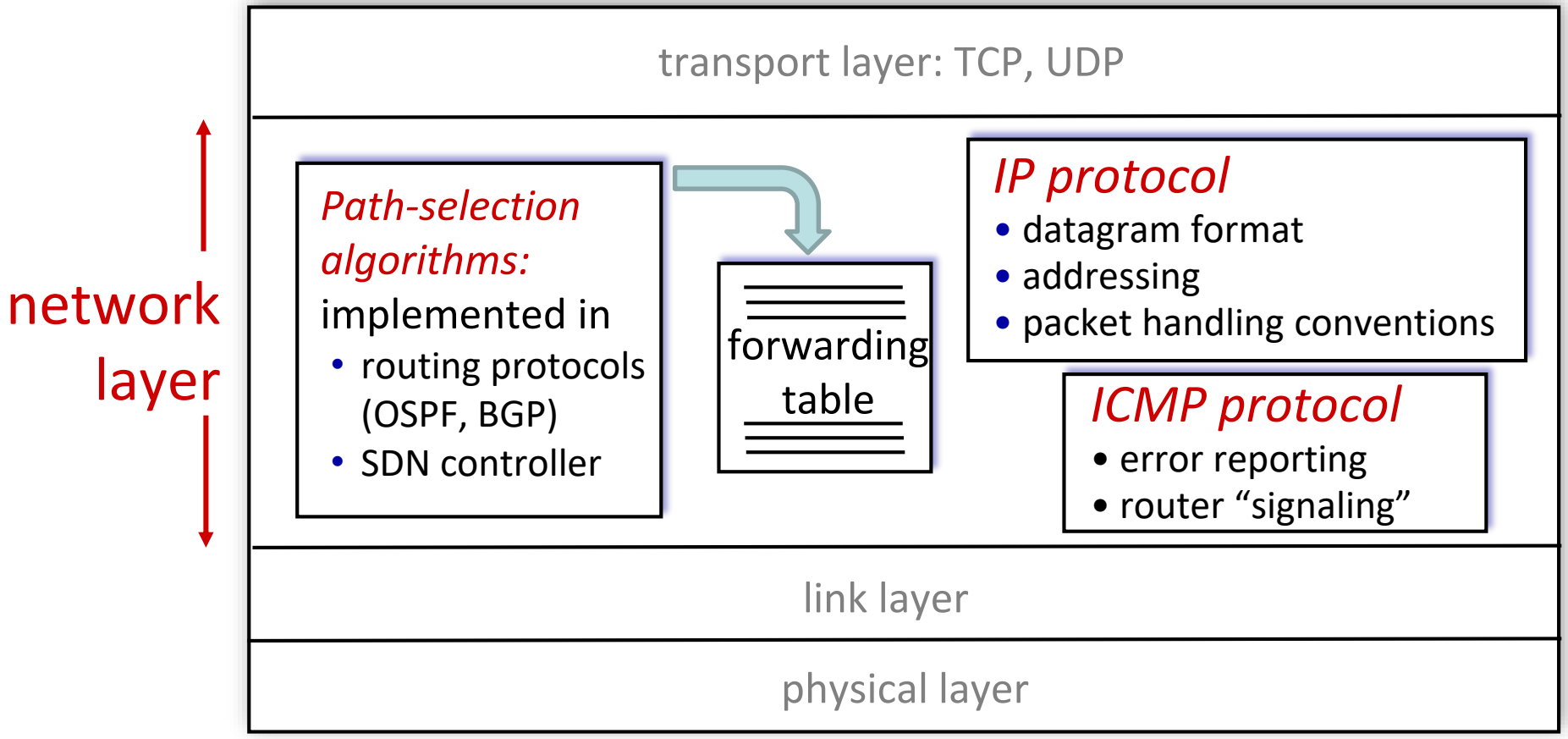
note: congestion control not handled by IP

IP Service Model

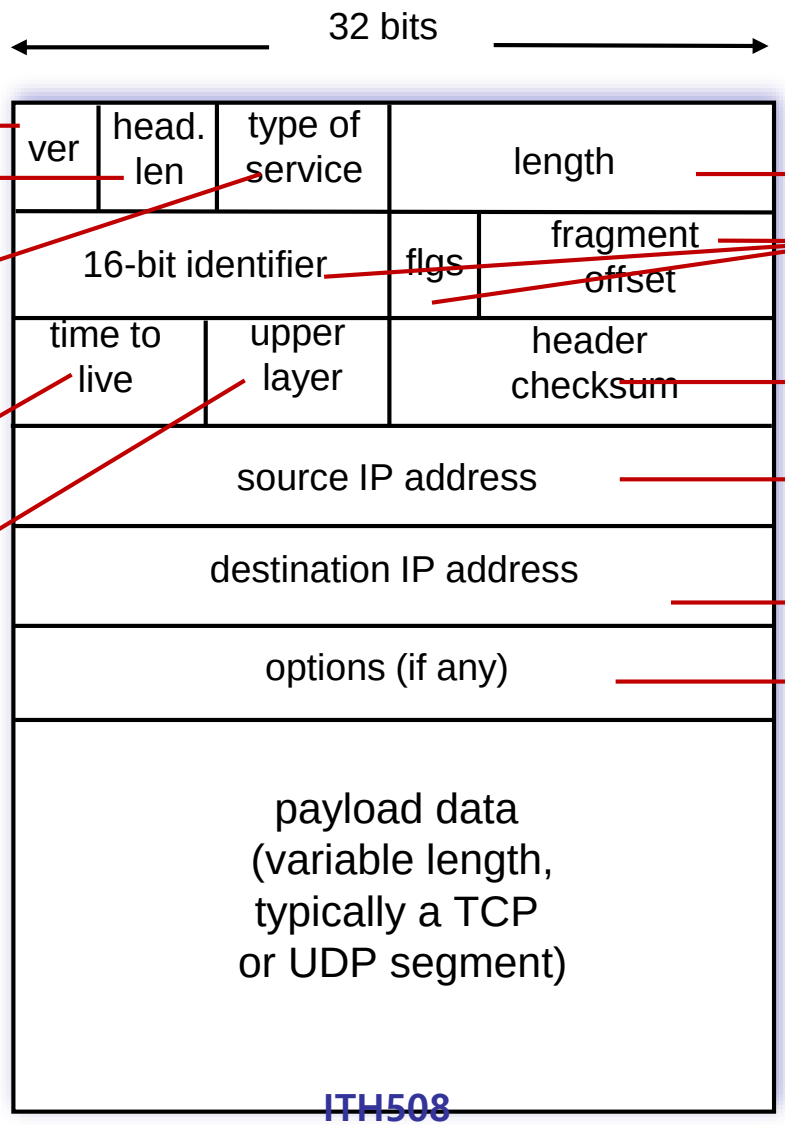
- **Service provided to transport layer (TCP, UDP)**
  - Global name space
  - Host-to-host connectivity (connectionless)
  - Best-effort packet delivery
- **Not in IP service model**
  - Delivery guarantees on bandwidth, delay or loss
- **Delivery failure modes**
  - Packet delayed for a very long time
  - Packet loss
  - Packet delivered more than once
  - Packets delivered out of order

IP Layer

Host, router network layer functions:



IP Datagram Format



IP protocol version number

header length(bytes)

"type" of service:
- diffserv (0:5)
- ECN (6:7)

TTL: remaining max hops (decremented at each router)

upper layer protocol (e.g., TCP or UDP)

Maximum length: 64K bytes
Typically: 1500 bytes or less

total datagram length (bytes)

fragmentation/ reassembly

header checksum

32-bit source IP address

32-bit destination IP address

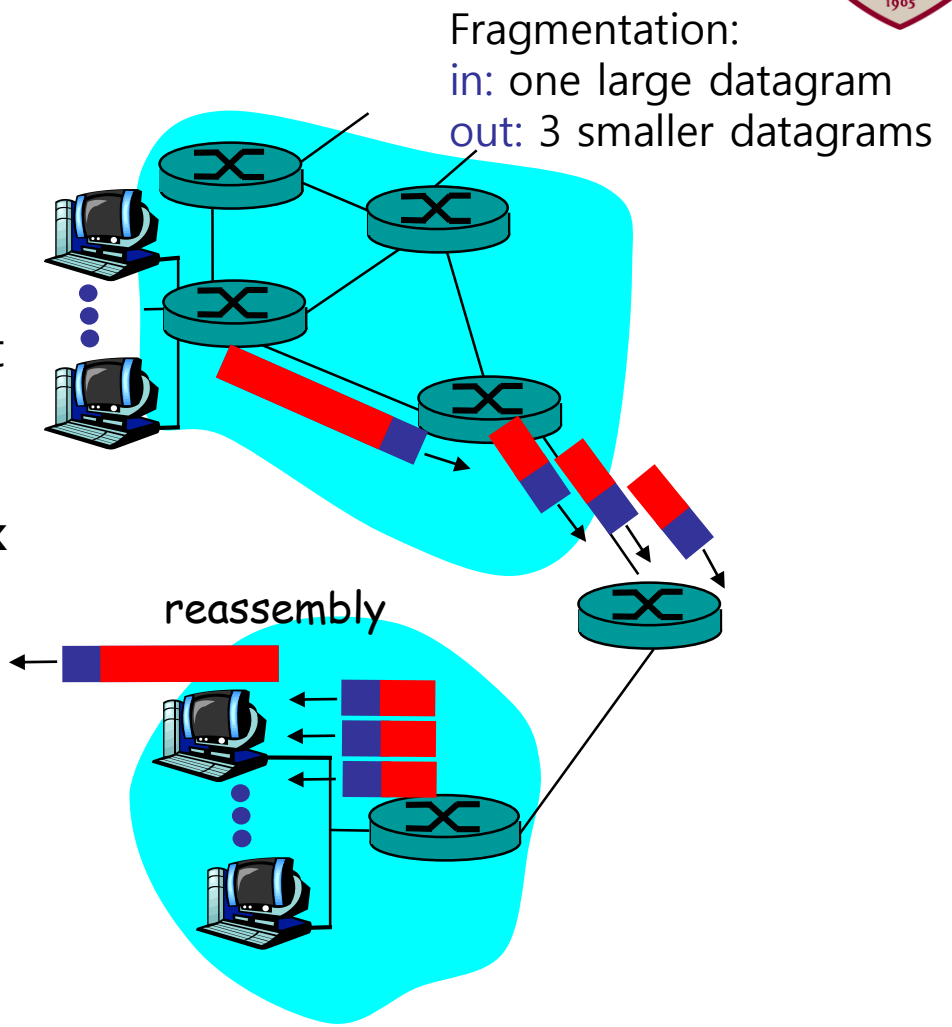
e.g., timestamp, record route taken

overhead

- 20 bytes of TCP
- 20 bytes of IP
- = 40 bytes + app layer overhead for TCP+IP

IP Fragmentation & Reassembly

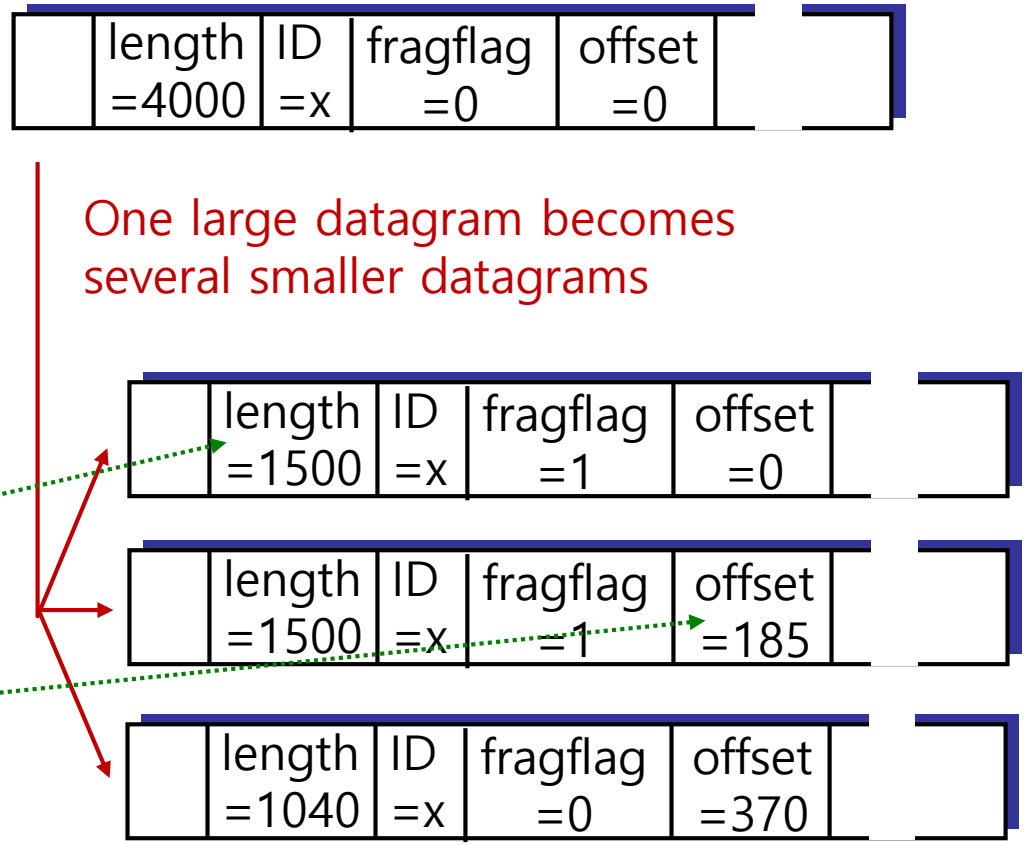
- **Network links have MTU (maximum transfer size) - largest possible link-level frame.**
  - Different link types, different MTUs
- **Large IP datagram divided ("fragmented") within network**
  - One datagram becomes several datagrams
  - "Reassembled" only at final destination
  - IP header bits used to identify, order related fragments



IP Fragmentation and Reassembly

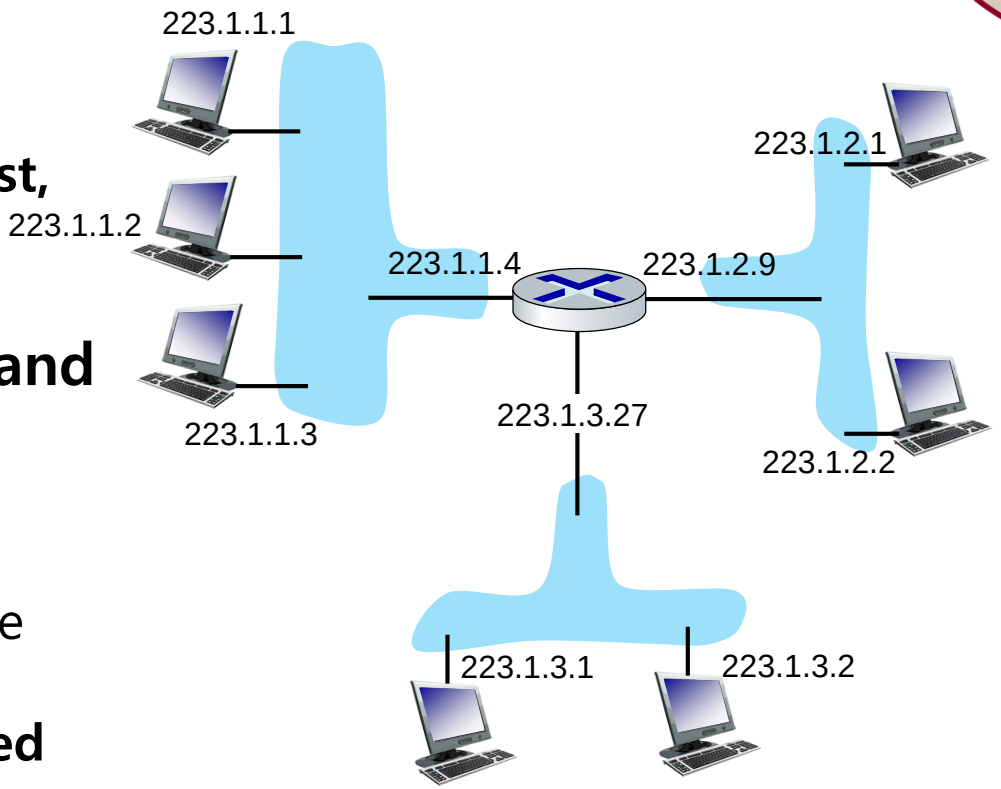
Example

- 4000 byte datagram
- MTU = 1500 bytes



IP Addressing: Introduction

- **IP address:**
  - 32-bit identifier for **host, router *interface***
- ***Interface:* connection between host/router and physical link**
  - Router's typically have multiple interfaces
  - Host may have multiple interfaces
  - **IP addresses associated with each interface**



dotted-decimal IP address notation:

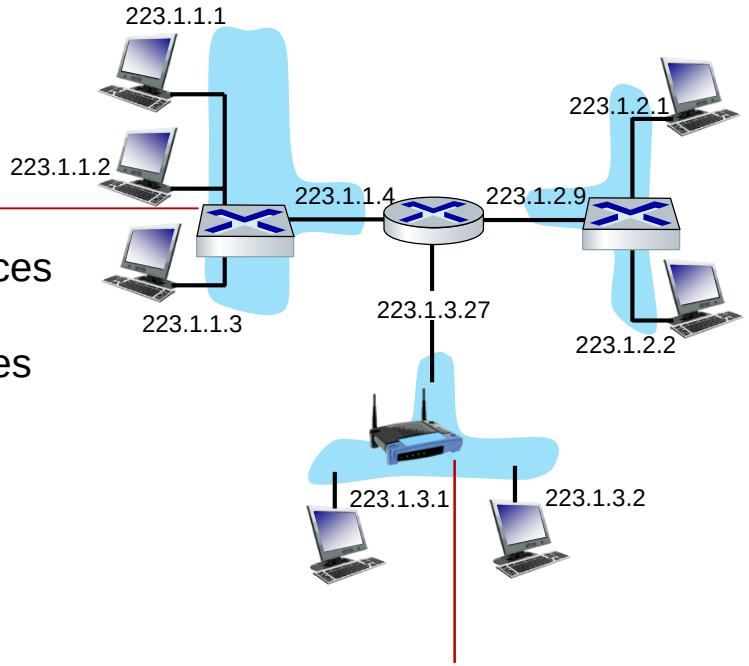
223.1.1.1 = 11011111 00000001 00000001 00000001

223 1 1 1

IP addressing: introduction

Q: how are interfaces actually connected?

A: we'll learn about that in chapters 6, 7



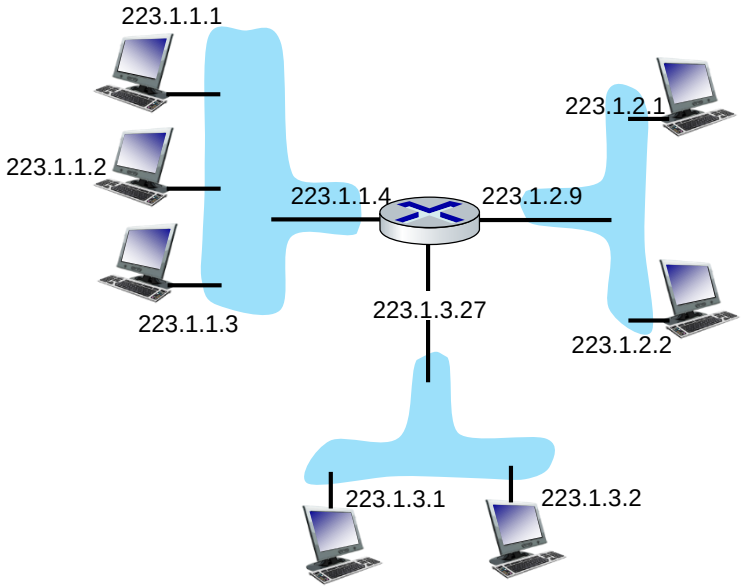
A: wired Ethernet interfaces connected by Ethernet switches

A: wireless WiFi interfaces connected by WiFi base station

For now: don't need to worry about how one interface is connected to another (with no intervening router)

Subnets

- *What's a subnet ?*
  - Device interfaces that can physically reach each other without passing through an intervening router
- IP addresses have structure:
  - Subnet part: devices in same subnet have common high order bits
  - Host part: remaining low order bits

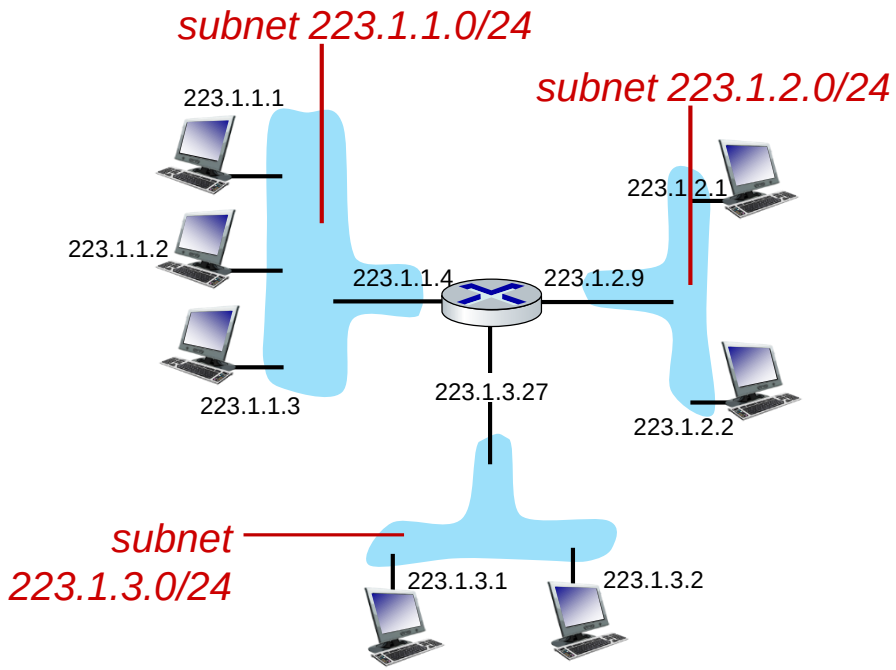


network consisting of 3 subnets

Subnets

Recipe for defining subnets:

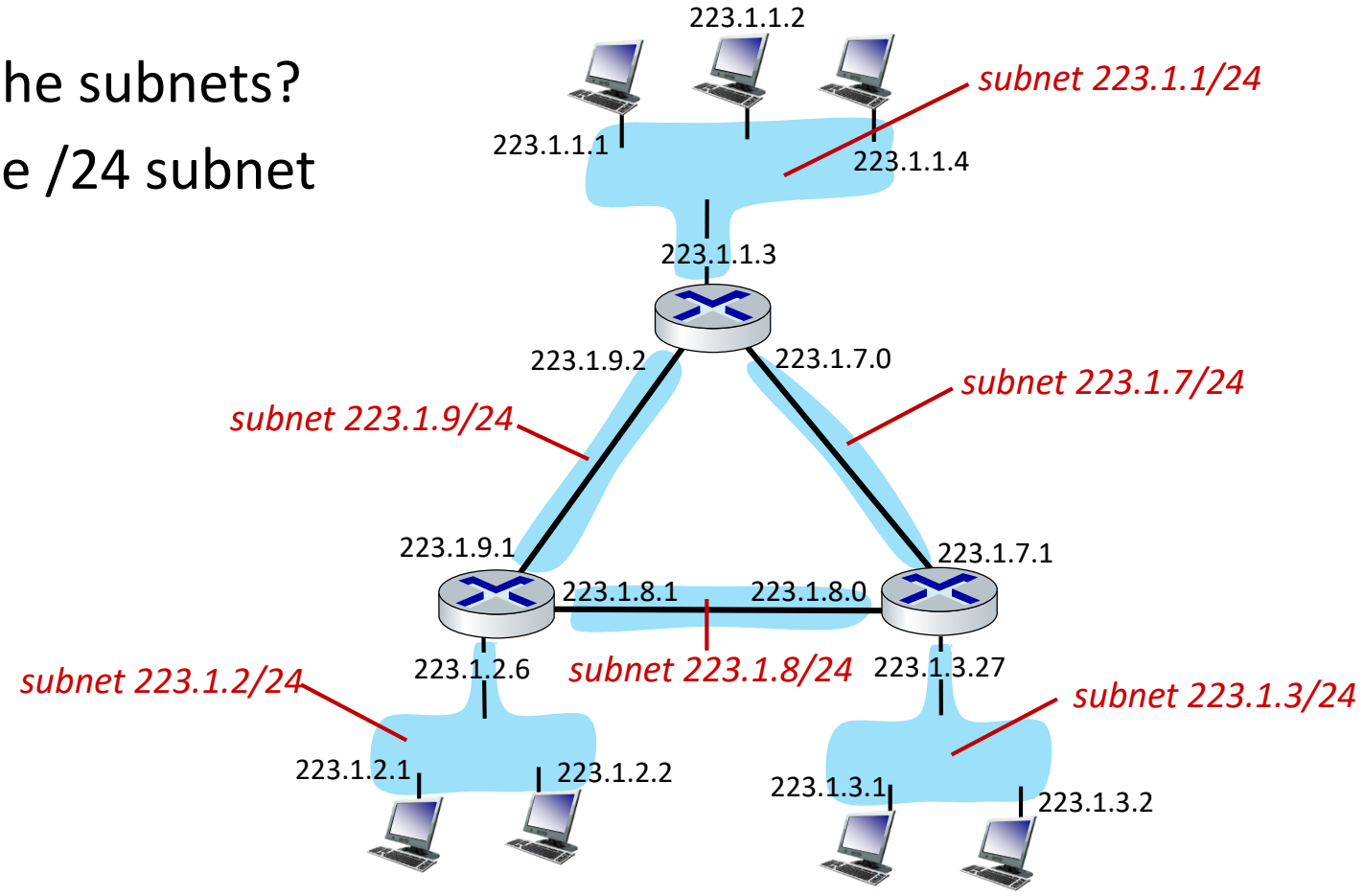
- Detach each interface from its host or router, creating "islands" of isolated networks
- Each isolated network is called a *subnet*



subnet mask: /24
(high-order 24 bits: subnet part of IP address)

Subnets

- where are the subnets?
- what are the /24 subnet addresses?



IP Addressing: CIDR

CIDR: Classless InterDomain Routing

- Subnet portion of address of arbitrary length
- Address format: a.b.c.d/x, where x is # bits in subnet portion of address

200.23.16.0/23

IP Addresses: How to Get?

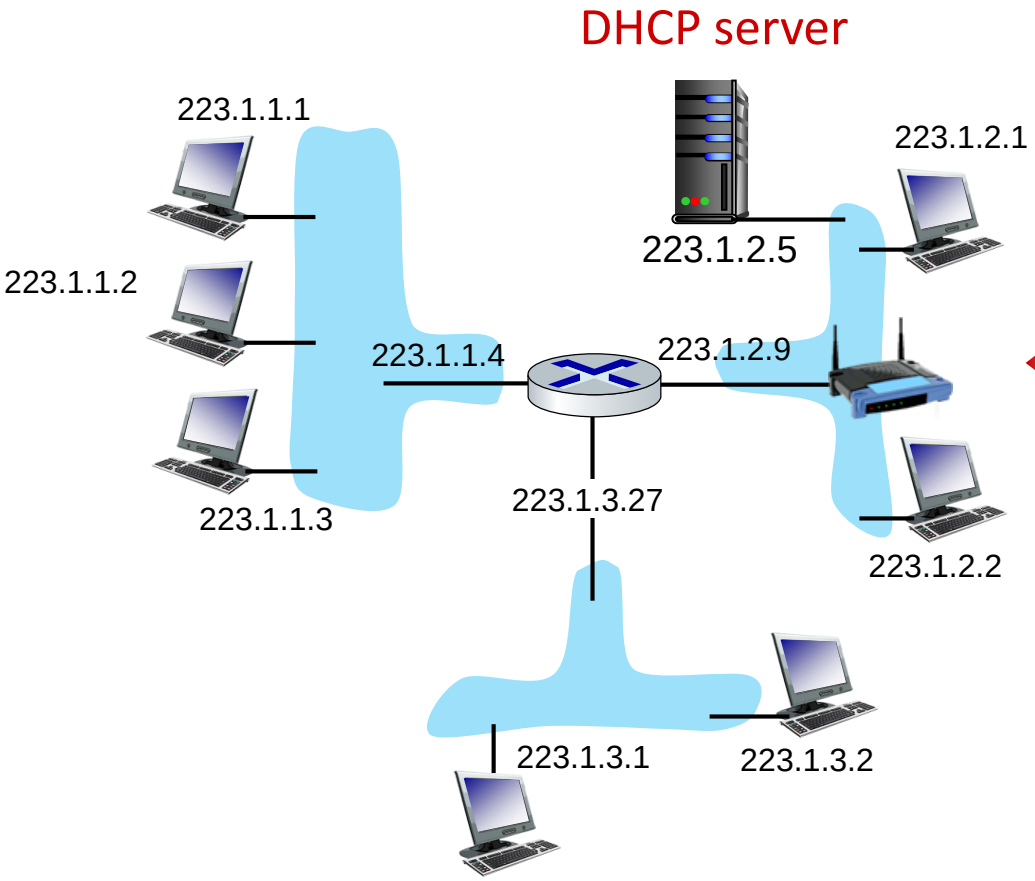
Q: How does *host* get IP address?

- **Hard-coded by system admin in a file**
  - Win: control-panel -> network -> configuration -> TCP/IP -> properties
  - UNIX: /etc/rc.config
- **DHCP: Dynamic Host Configuration Protocol: dynamically get address from as server**
  - "plug-and-play"

DHCP: Dynamic Host Configuration Protocol

- **Goal: allow host to dynamically obtain its IP address from network server when it joins network**
  - Can renew its lease on address in use
  - Allows reuse of addresses (only hold address while connected/"on")
  - Support for mobile users who want to join network (more shortly)
- **DHCP overview:**
  - Host broadcasts "DHCP discover" msg [optional]
  - DHCP server responds with "DHCP offer" msg [optional]
  - Host requests IP address: "DHCP request" msg
  - DHCP server sends address: "DHCP ack" msg

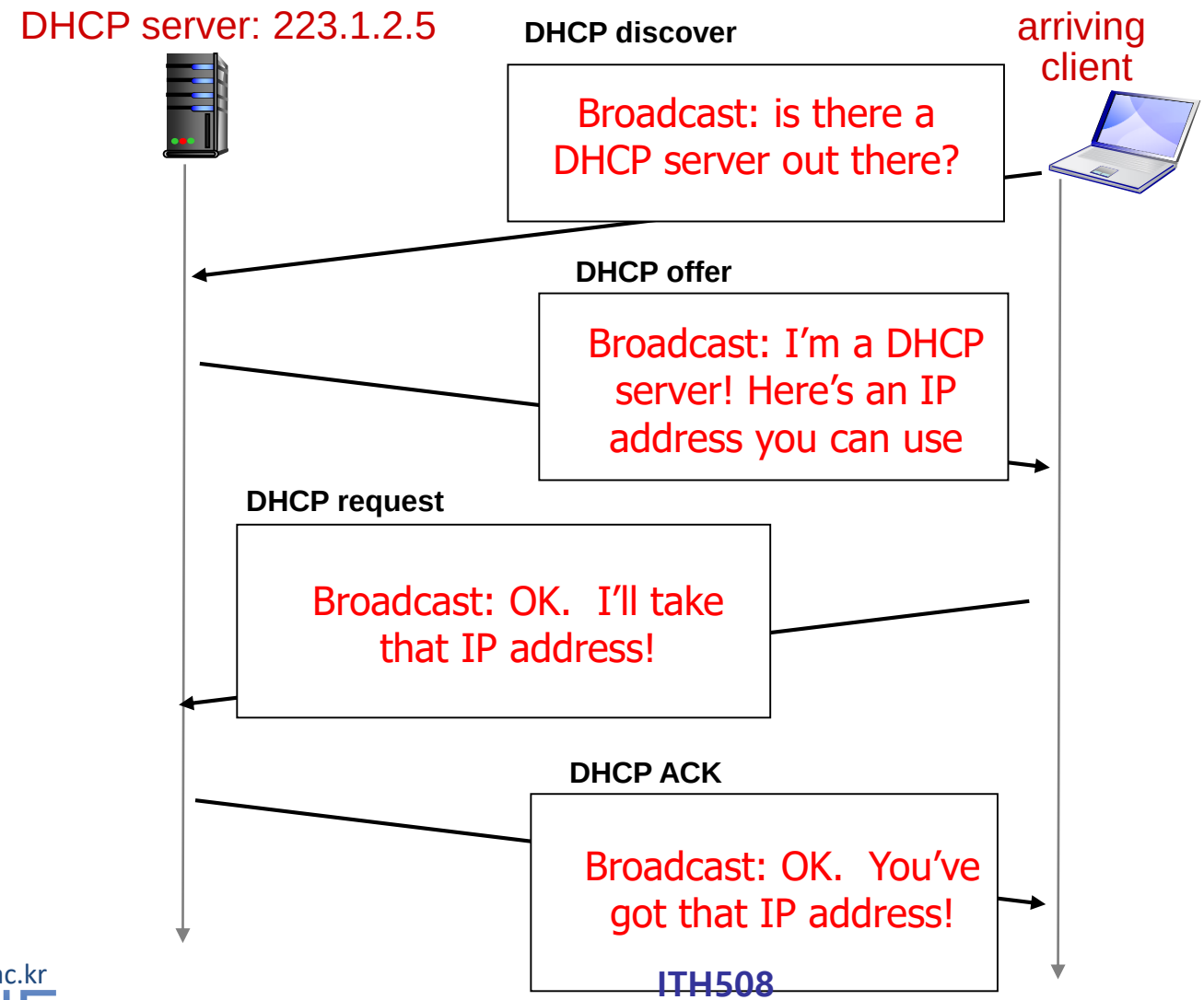
DHCP Scenario



Typically, DHCP server will be co-located in router, serving all subnets to which router is attached

arriving DHCP client needs address in this network

DHCP Scenario



IP Addresses: How to Get?

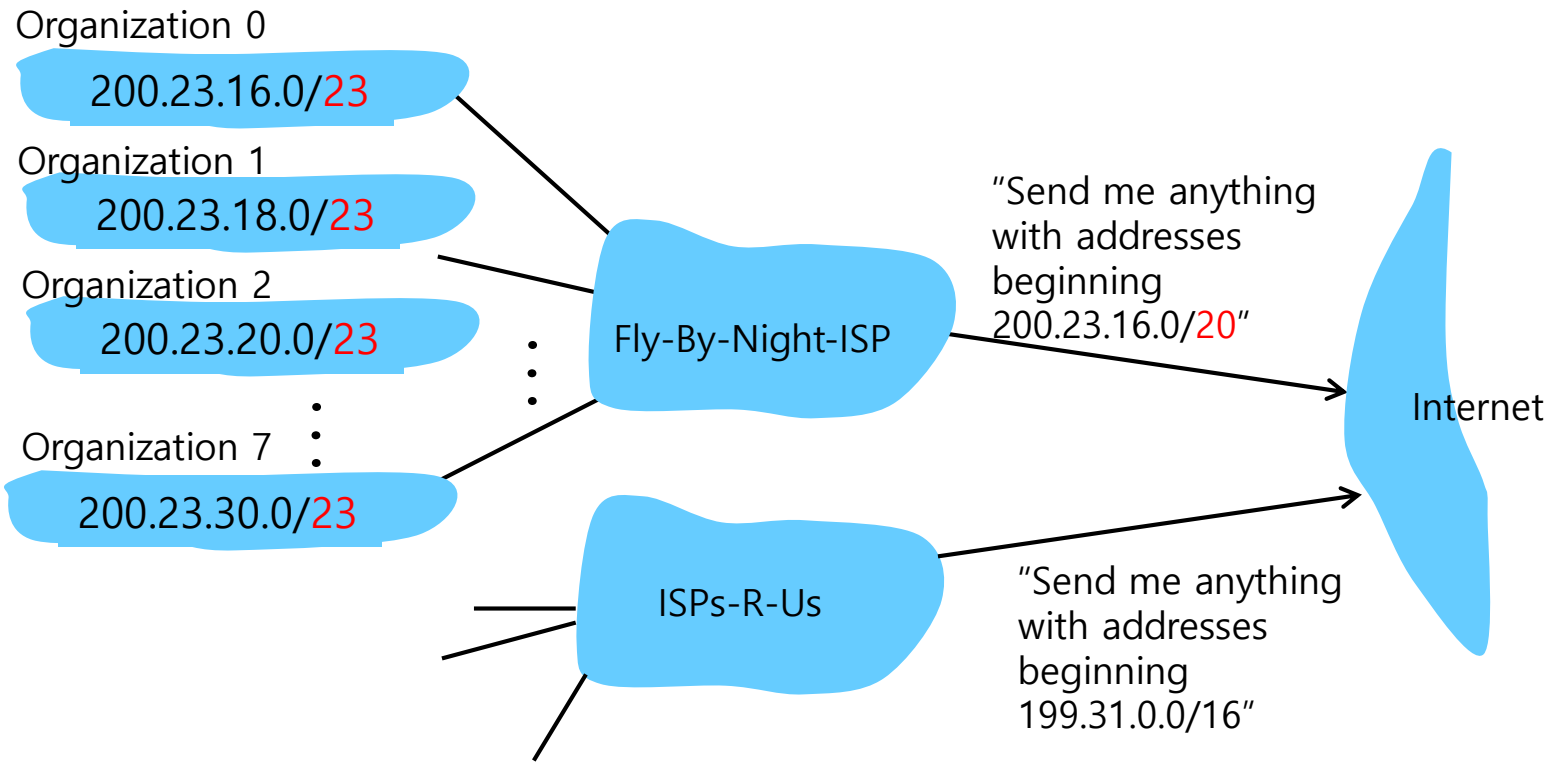
Q: How does *network* get subnet part of IP address?

A: gets allocated portion of its provider ISP's address space

ISP's block	11001000 00010111 00010000	00000000	200.23.16.0/20
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

Hierarchical Addressing: Route Aggregation

Hierarchical addressing allows **efficient advertisement of routing information:**



IP Addressing

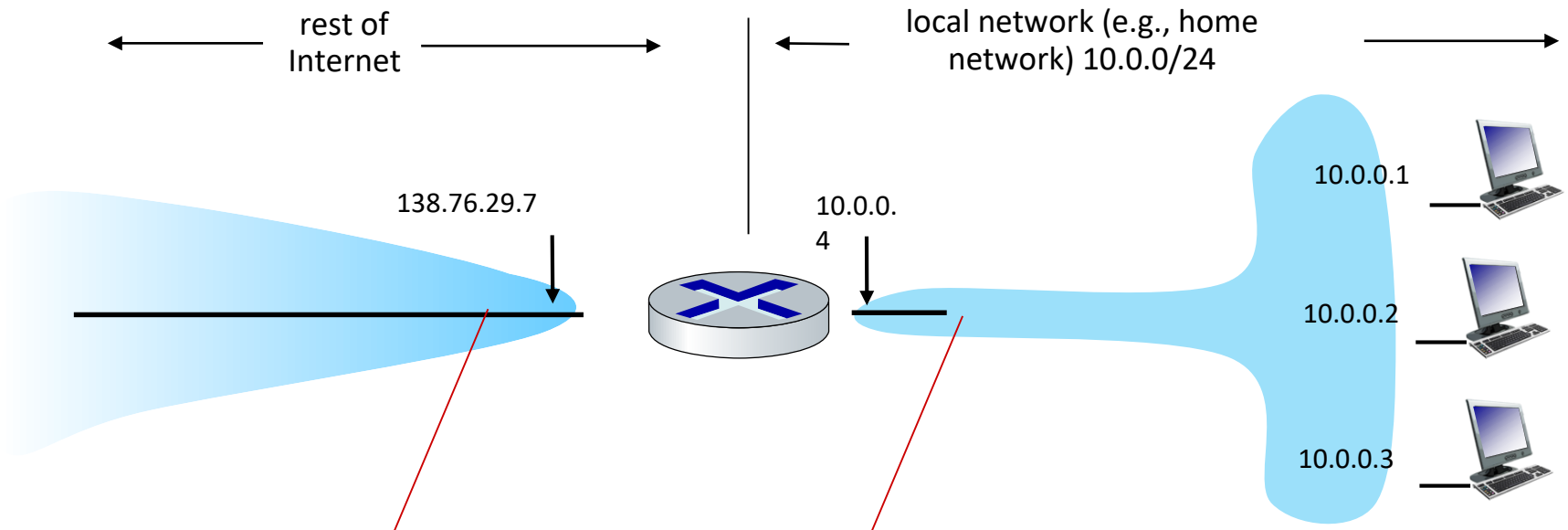
Q: How does an ISP get block of addresses?

A: ICANN: Internet Corporation for Assigned Names and Numbers

- Allocates addresses
- Manages DNS
- Assigns domain names, resolves disputes

NAT: Network Address Translation

NAT: all devices in local network share just one IPv4 address as far as outside world is concerned



all datagrams *leaving* local network have *same* source NAT IP address: 138.76.29.7, but *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

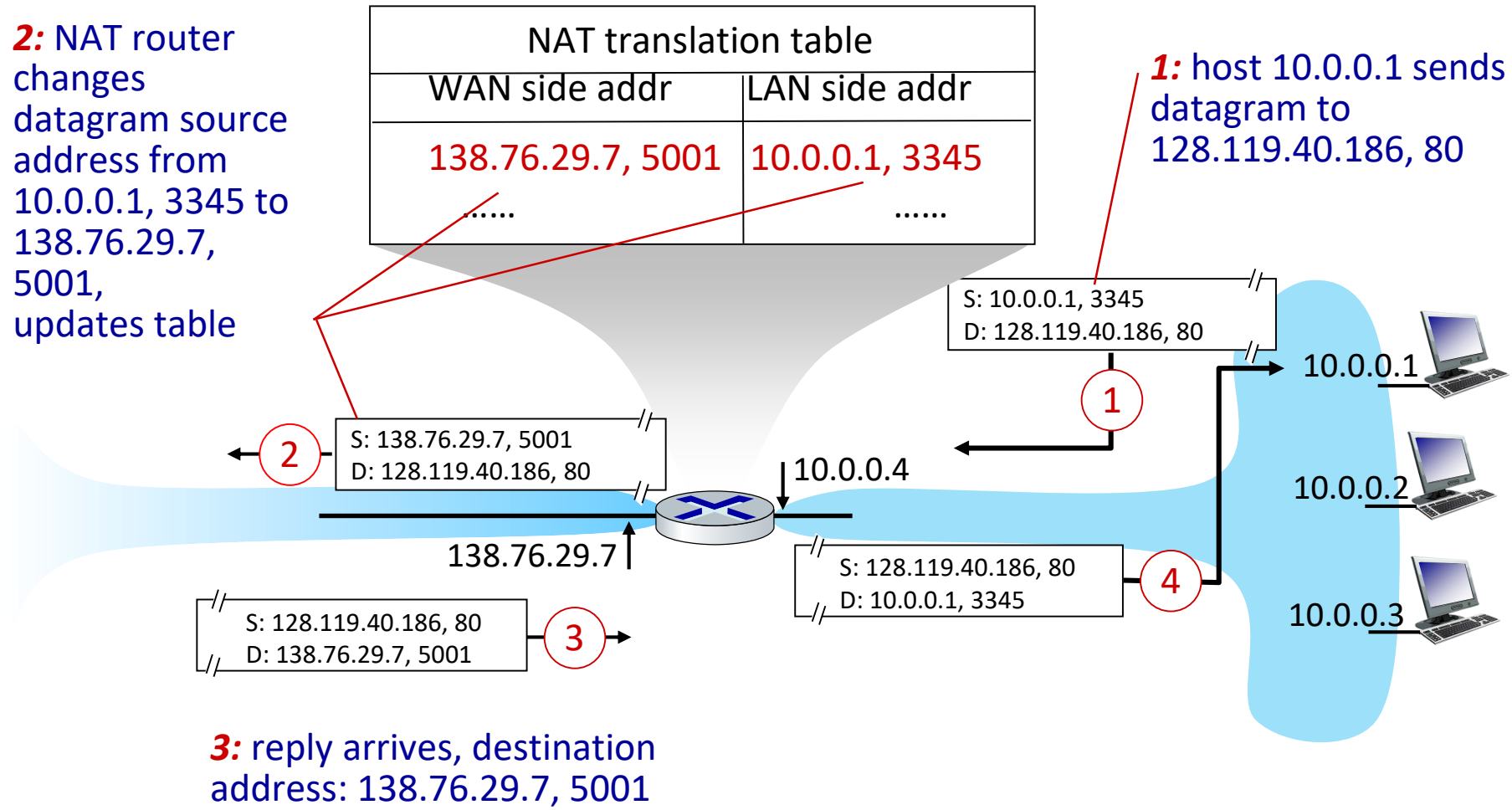
- **Motivation: local network uses just one IP address as far as outside word is concerned:**
  - **No need to be allocated range of addresses** from ISP
    - Just one IP address is used for all devices
  - **Can change addresses of devices in local network** without notifying outside world
  - **Can change ISP** without changing addresses of devices in local network
  - **Devices inside local net not explicitly addressable**, visible by outside world (a security plus).

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 address
  - ***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 destination fields of every incoming datagram with corresponding (source IP address, port #) stored in NAT table

NAT: Network Address Translation



NAT: Network Address Translation

- **16-bit port-number field:**
  - 60,000 simultaneous connections with a single LAN-side address!
- **NAT is controversial:**
  - Routers should only process up to layer 3
  - Violates end-to-end argument
    - NAT possibility must be taken into account by app designers, eg, P2P applications
  - Address shortage should instead be solved by IPv6

ICMP: Internet Control Message Protocol

- **Used by hosts & routers to communicate network-level information**
  - **Error reporting**: unreachable host, network, port, protocol
  - **Echo request/reply** (used by ping)
- **Network-layer "above" IP:**
  - ICMP msgs carried in IP datagrams
- **ICMP message: type, code plus first 8 bytes of IP datagram causing error**

Type	Code	description
0	0	echo reply (ping)
3	0	dest. network unreachable
3	1	dest host unreachable
3	2	dest protocol unreachable
3	3	dest port unreachable
3	6	dest network unknown
3	7	dest host unknown
4	0	source quench (congestion control - not used)
8	0	echo request (ping)
9	0	route advertisement
10	0	router solicitation (discovery)
11	0	TTL expired
12	0	bad IP header

Real Internet Delays and Routes Revisited

traceroute: mailserver.cs.uiuc.edu to portal.korea.ac.kr

```
traceroute to portal.korea.ac.kr (163.152.6.19), 30 hops max, 40 byte packets
 1  dcsgw-dept (128.174.252.129)  1.253 ms  4.447 ms  0.731 ms
 2  uiuc-node1siebel-net.gw.uiuc.edu (128.174.1.185)  1.912 ms  0.642 ms  0.675 ms
 3  * * *
 4  130.126.0.14 (130.126.0.14)  1.444 ms  1.066 ms  0.942 ms
 5  130.126.0.30 (130.126.0.30)  0.969 ms  1.063 ms  0.933 ms
 6  t-dmzo.gw.uiuc.edu (130.126.0.70)  1.004 ms  1.020 ms  1.374 ms
 7  192.17.10.46 (192.17.10.46)  4.334 ms  4.548 ms  3.804 ms
 8  206.220.240.166 (206.220.240.166)  4.125 ms  10.570 ms  3.867 ms
 9  192.203.116.9 (192.203.116.9)  195.091 ms  192.615 ms  196.139 ms
10  apii-juniper-ge1-0-0-1036.jp.apan.net (203.181.248.226)  190.498 ms  191.076 ms  190.460 ms
11  203.181.249.161 (203.181.249.161)  192.779 ms  192.425 ms  192.874 ms
12  192.168.99.1 (192.168.99.1)  193.692 ms  193.713 ms  193.740 ms
13  192.168.66.2 (192.168.66.2)  193.756 ms  193.755 ms  193.673 ms
14  sgp-tgp.koren21.net (203.255.248.161)  193.683 ms  193.878 ms  201.024 ms
15  ku-sgp.koren21.net (203.255.248.205)  211.900 ms  203.850 ms  202.772 ms
16  * * *
17  * * *
18  * * *
19  * * *
20  * * *
21  * * *
22  * * *
23  * * *
24  * * *
25  * * *
26  * * *
27  * * *
28  * * *
29  * * *
30  * * *
```

Three delay measurements

trans-oceaniclink

* means no response (probe lost, router not replying)

Traceroute and ICMP

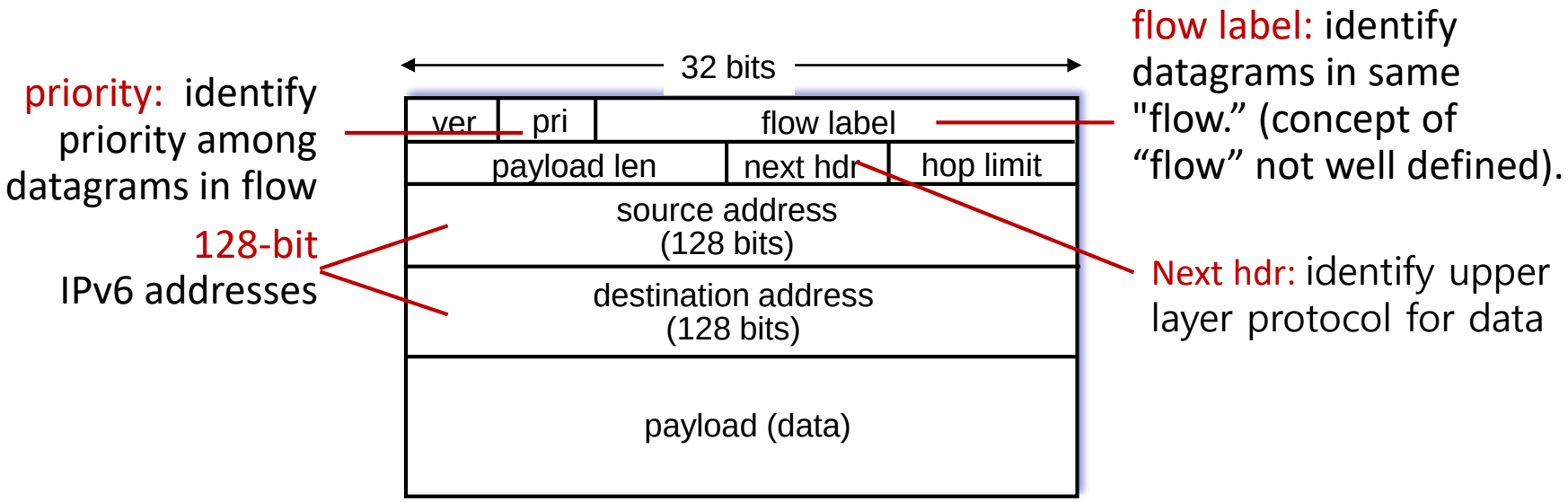
- **Source sends series of UDP segments to destination**
  - First has TTL =1
  - Second has TTL=2, etc.
  - Unlikely port number
- **When nth datagram arrives to nth router:**
  - Router discards datagram
  - And **sends to source an ICMP message** (type 11, code 0)
  - Message includes name of router& IP address
- **When ICMP message arrives, source calculates RTT**
- **Traceroute does this 3 times**
- **UDP segment eventually arrives at destination host**
- **Destination returns ICMP "host unreachable" packet (type 3, code 3)**
- **When source gets this ICMP, stops.**

IPv6

- **Initial motivation: 32-bit address space soon to be completely allocated.**
- **Additional motivation:**
  - Header format helps **speed processing/forwarding**
  - Header changes to **facilitate QoS**

  IPv6 datagram format:
  - Fixed-length 40 byte header
  - No fragmentation allowed

IPv6 Datagram Format



What's missing (compared with IPv4):

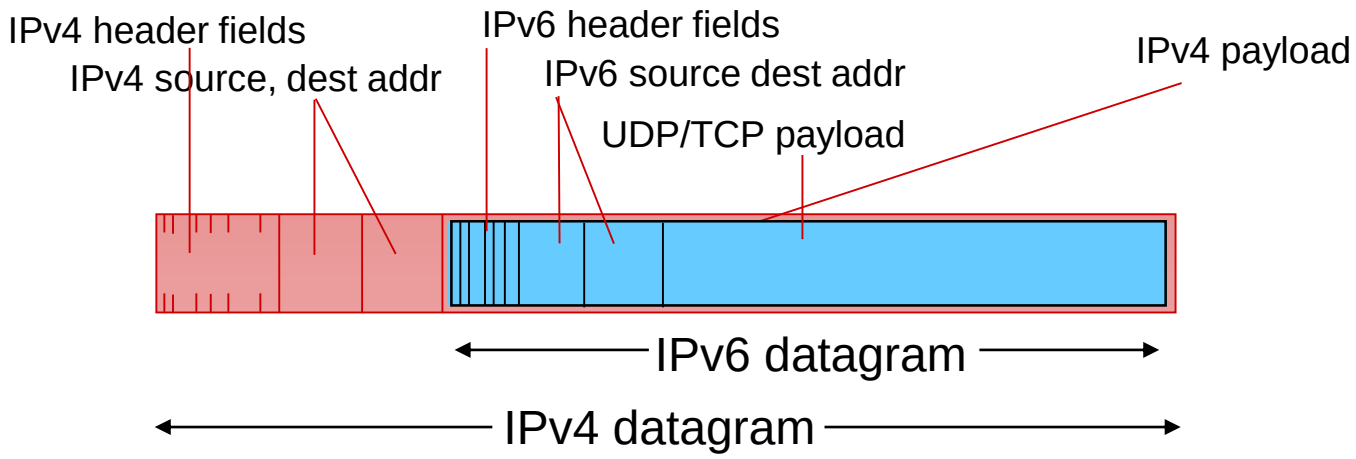
- no checksum (to speed processing at routers)
- no fragmentation/reassembly
- no options (available as upper-layer, next-header protocol at router)

Other Changes from IPv4

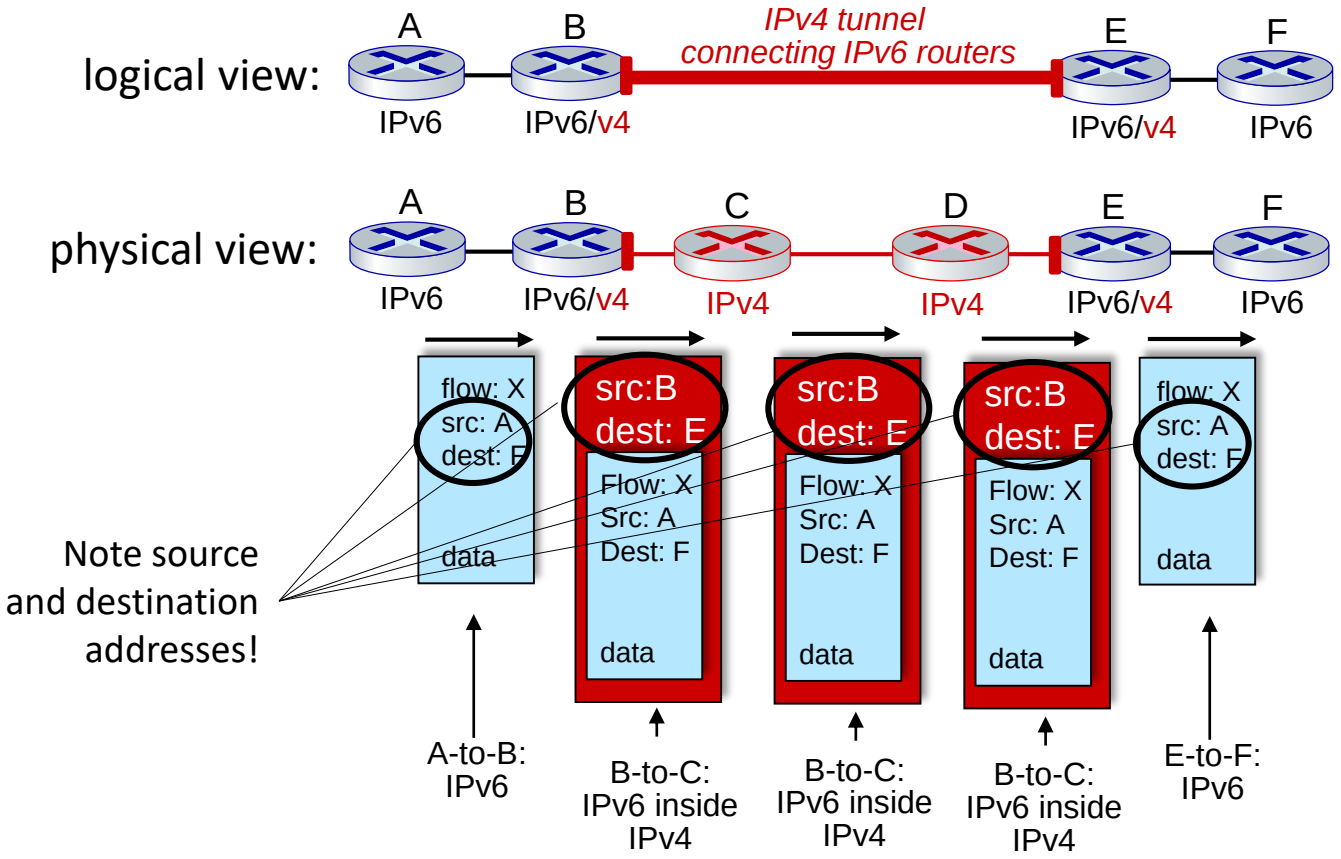
- **Checksum:** removed entirely to reduce processing time at each hop
- **Options:** allowed, but outside of header, indicated by "Next Header" field
- **ICMPv6:** new version of ICMP
  - Additional message types, e.g. "Packet Too Big"
  - Multicast group management functions

Transition From IPv4 To IPv6

- **Not all routers can be upgraded simultaneous**
  - No "flag days"
  - How will the network operate with mixed IPv4 and IPv6 routers?
- ***Tunneling:*** **IPv6 carried as payload in IPv4 datagram among IPv4 routers**

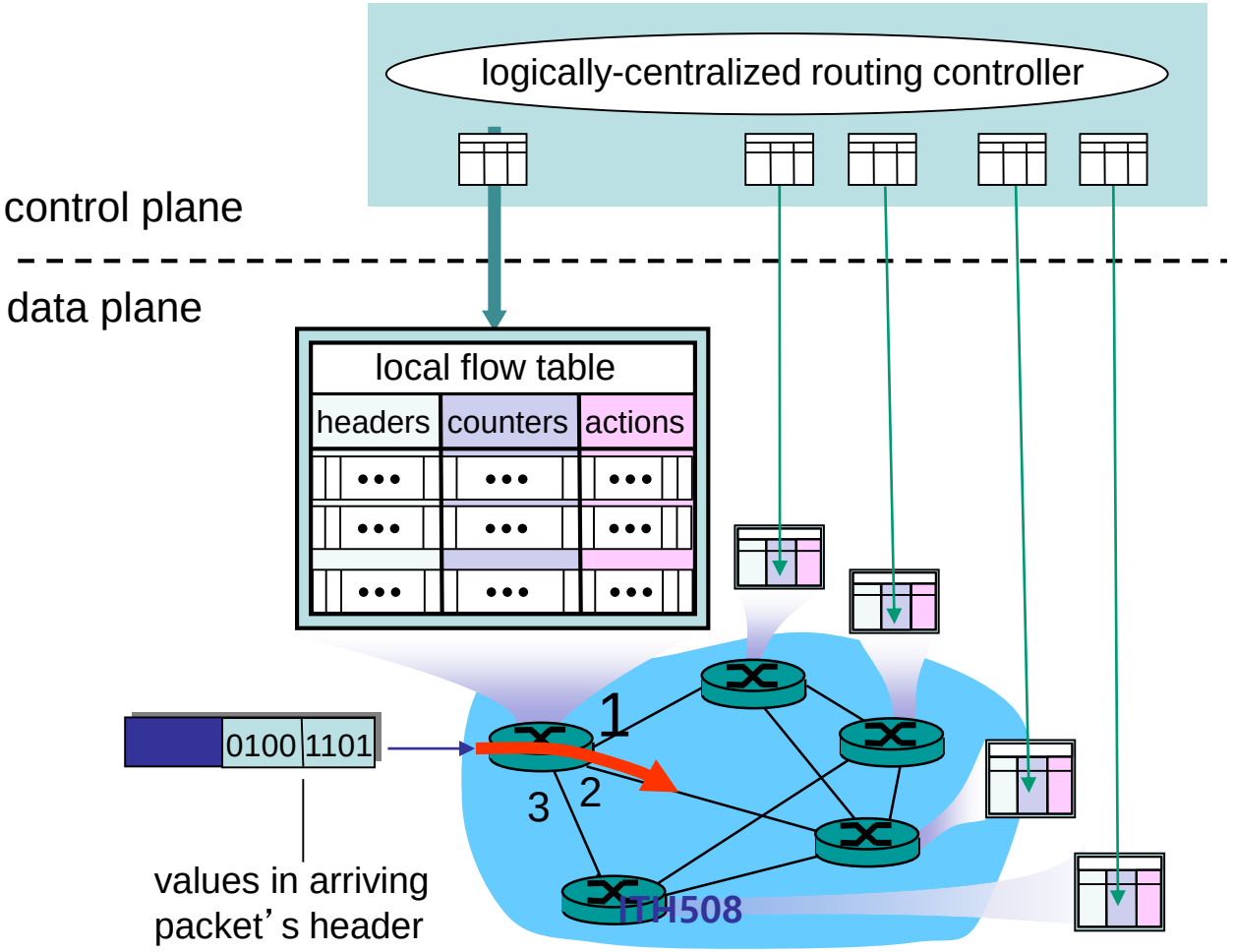


Tunneling


logical view:
A
B
IPv4 tunnel
connecting IPv6 routers
E
F
IPv6
IPv6/v4
IPv6/v4
IPv6
physical view:
A
B
C
D
E
F
IPv6
IPv6/v4
IPv4
IPv4
IPv6/v4
IPv6
Note source
and destination
addresses!
flow: X
src: A
dest: F
data
src:B
dest: E
Flow: X
Src: A
Dest: F
data
src:B
dest: E
Flow: X
Src: A
Dest: F
data
src:B
dest: E
Flow: X
Src: A
Dest: F
data
flow: X
src: A
dest: F
data
A-to-B:
IPv6
B-to-C:
IPv6 inside
IPv4
B-to-C:
IPv6 inside
IPv4
B-to-C:
IPv6 inside
IPv4
E-to-F:
IPv6

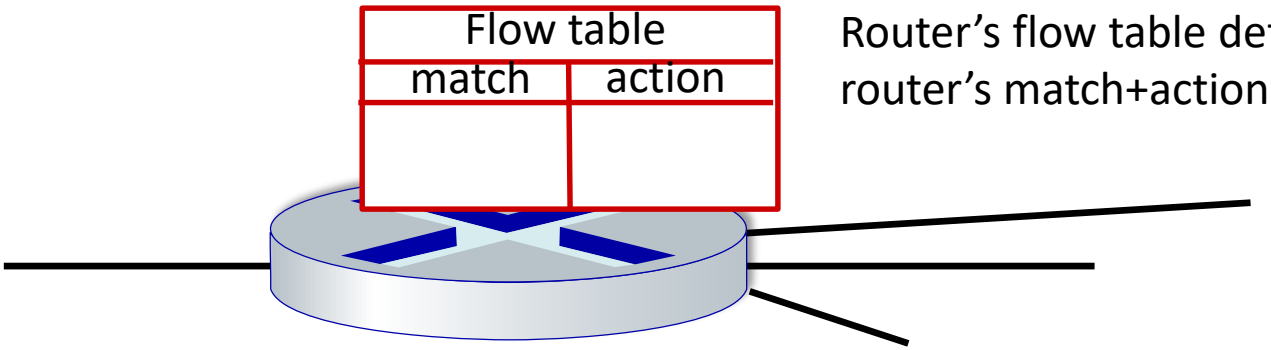
Generalized Forwarding and SDN

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



OpenFlow Data Plane Abstraction

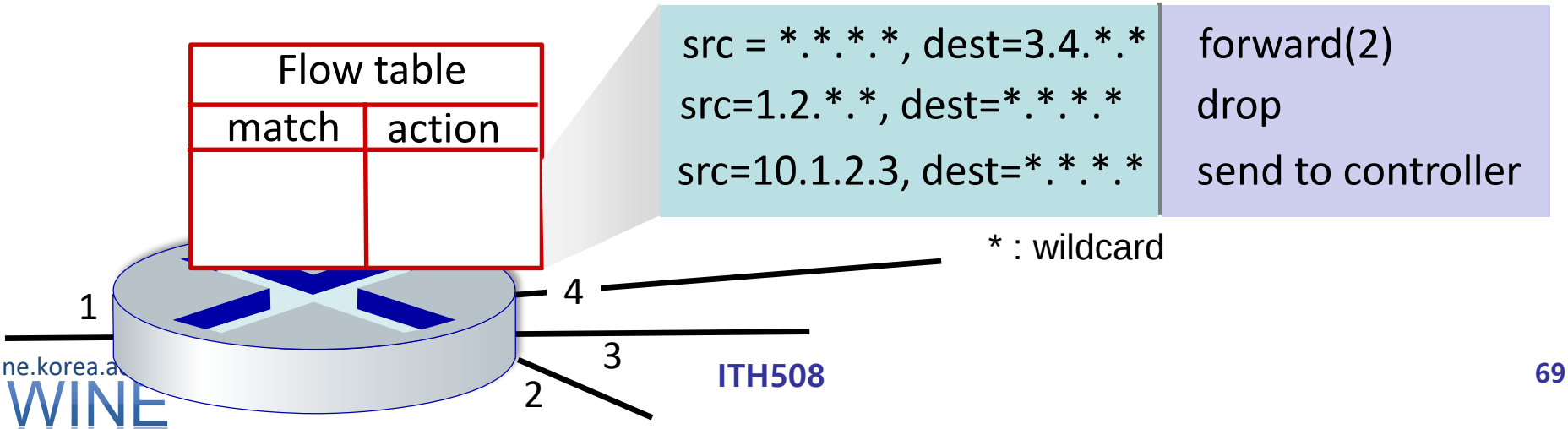
- ***Flow*: defined by header fields**
- **Generalized forwarding: simple packet-handling rules**
  - *Pattern:* match values in packet header fields
  - *Actions: for matched packet:* drop, forward, modify, matched packet or send matched packet to controller
  - *Priority*: disambiguate overlapping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



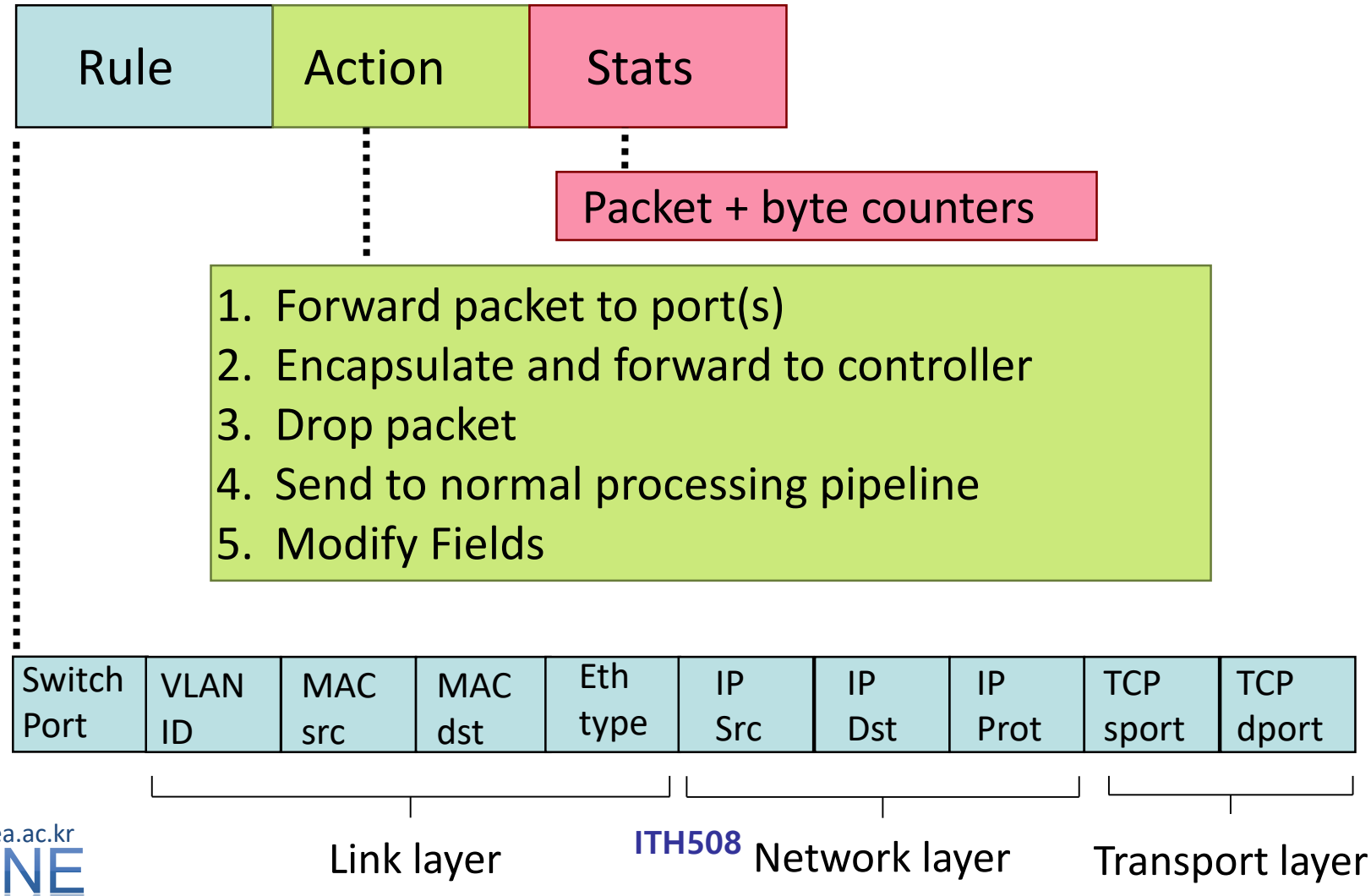
Flow table in a router (computed and distributed by controller) define router's match+action rules

OpenFlow Data Plane Abstraction

- ***Flow*: defined by header fields**
- **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



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	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 3

OpenFlow Abstraction

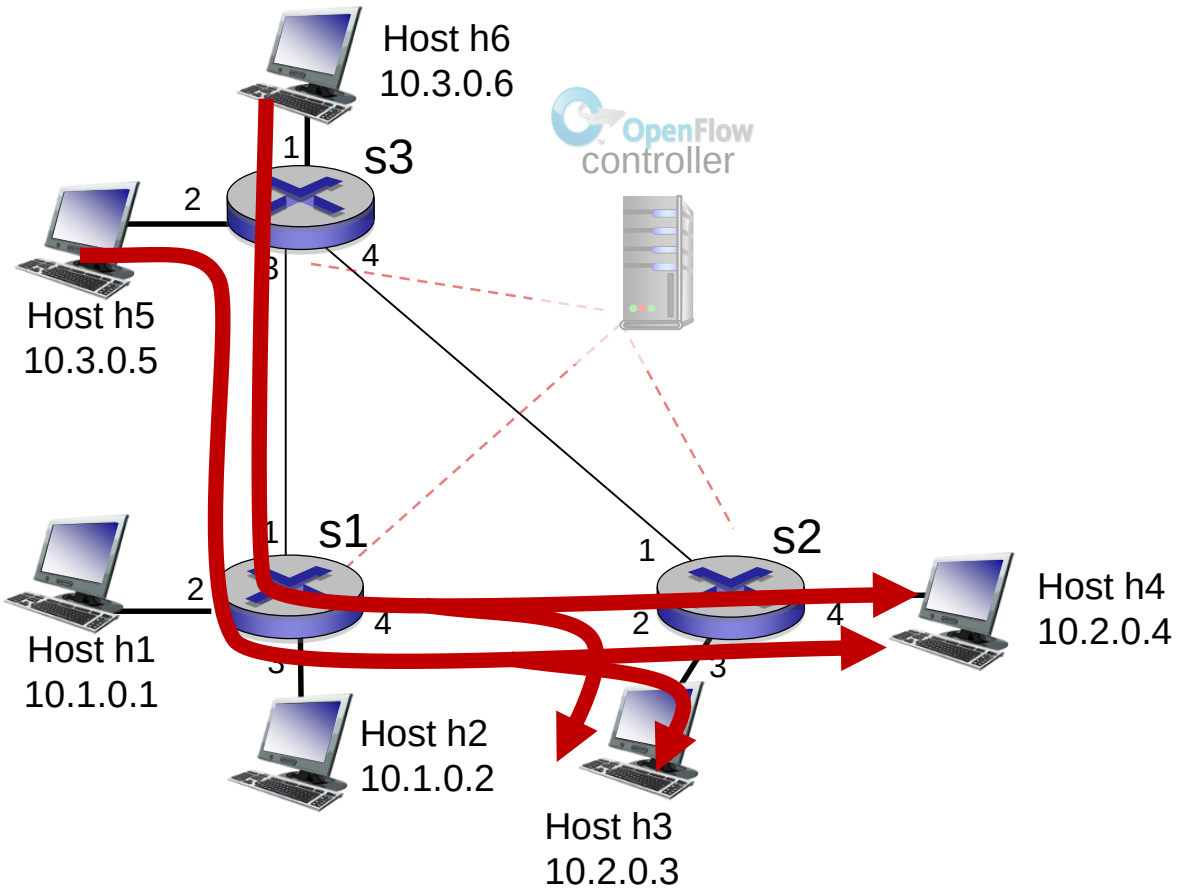
- *match+action:* unifies different kinds of devices

- **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

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

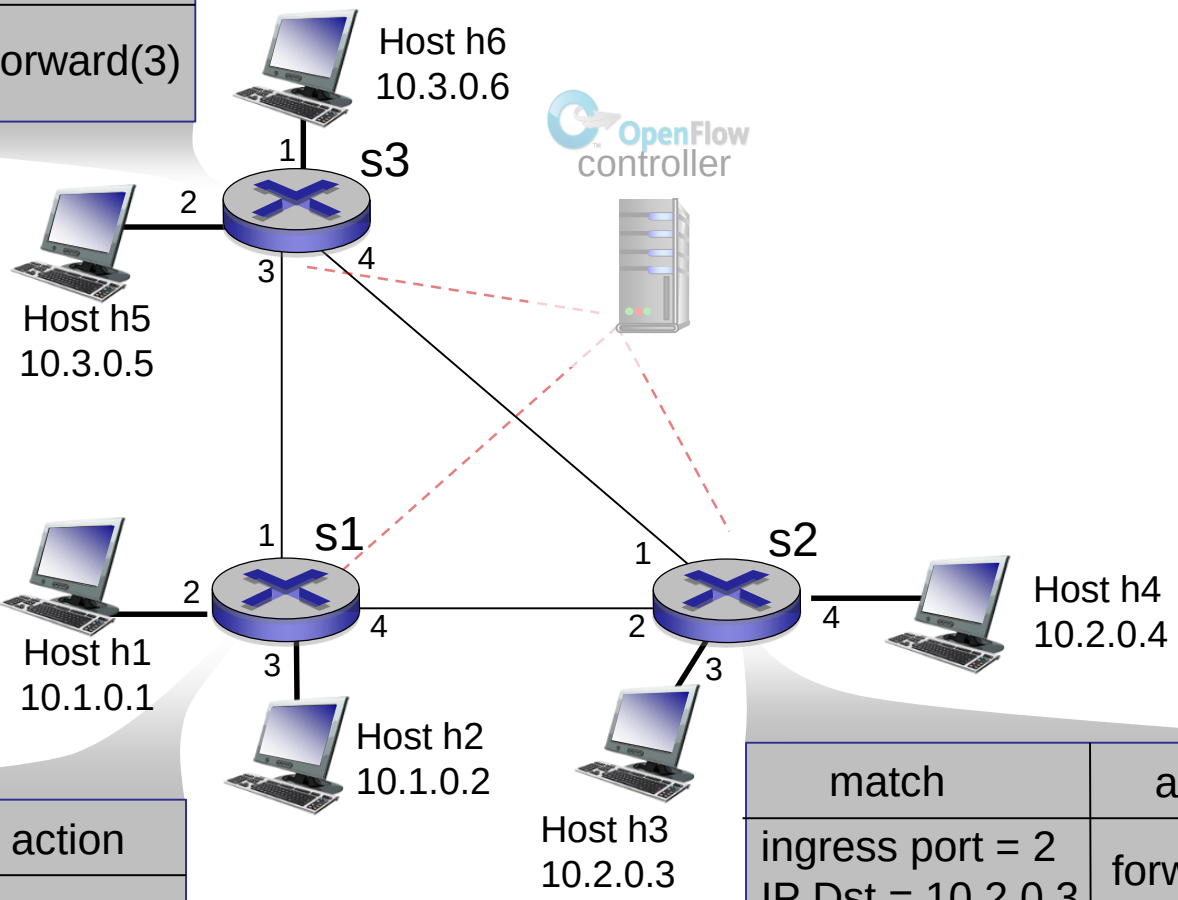
- datagrams from hosts h5 and h6 should be sent to h3 or h4, via s1 and from there to s2



OpenFlow Example

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)