Lecture 7

The Network Layer
Data Plane

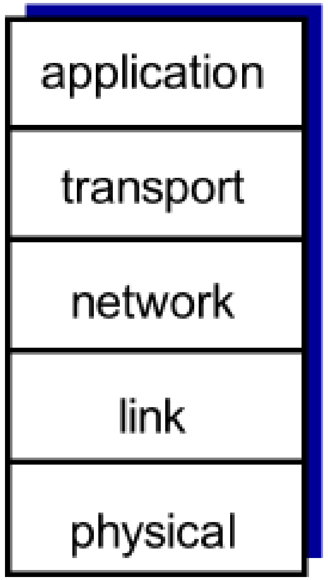
Subjects of today:

- The Role of the Network Layer
- The Router
- Internet Protocol Addressing
- Beyond IPv4, NAT and IPv6
- Generalized forwarding in brief

7.1 The Role of the Network Layer

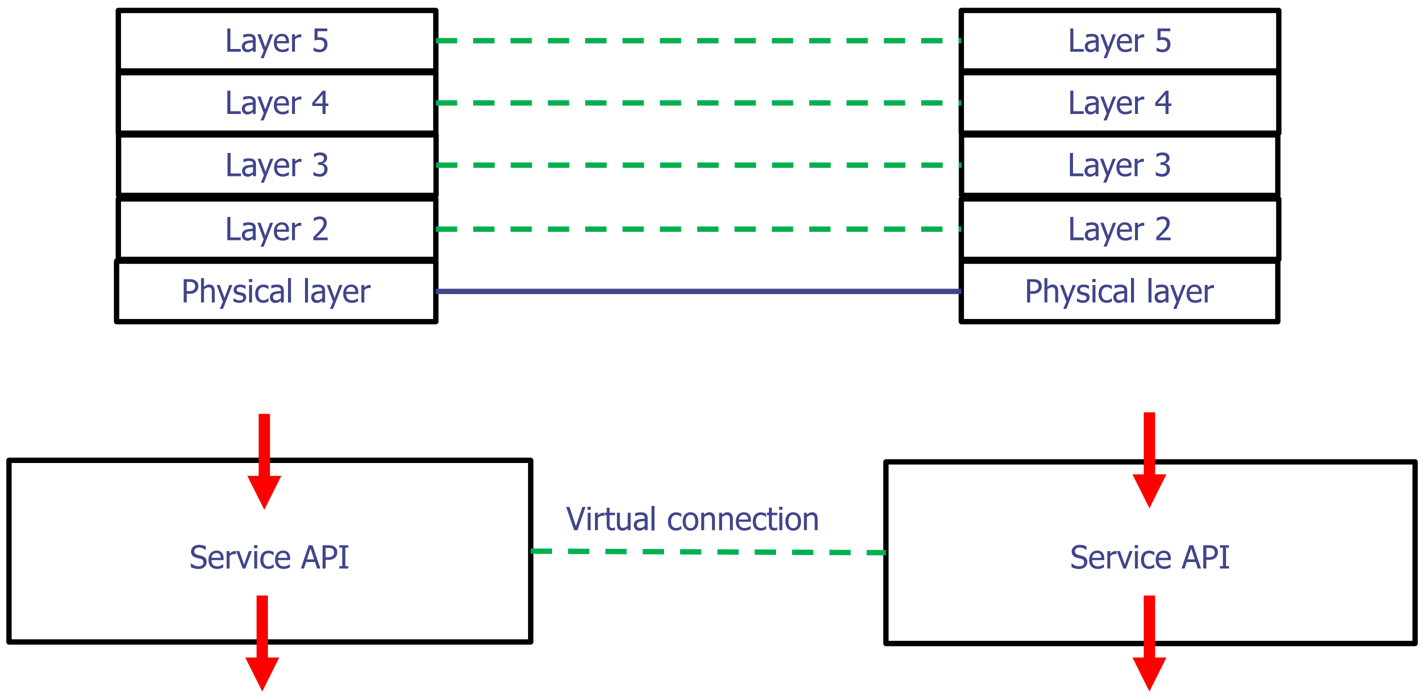
The Internet Protocol Stack

- *application:* supporting network applications
  - FTP, SMTP, HTTP
- *transport:* process-process data transfer
  - TCP, UDP
- *network:* routing of datagrams from source to destination
  - IP, routing protocols
- *link:* data transfer between neighboring network elements
  - Ethernet, 802.111 (WiFi), PPP
- *physical:* bits "on the wire"

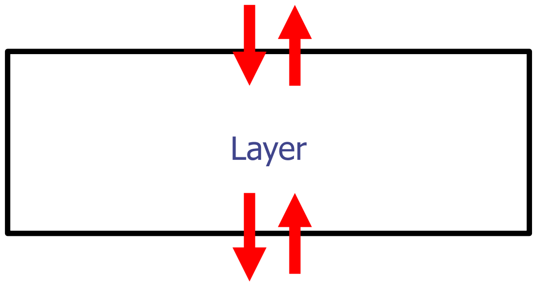


Introduction 1-60

Layered protocol stack

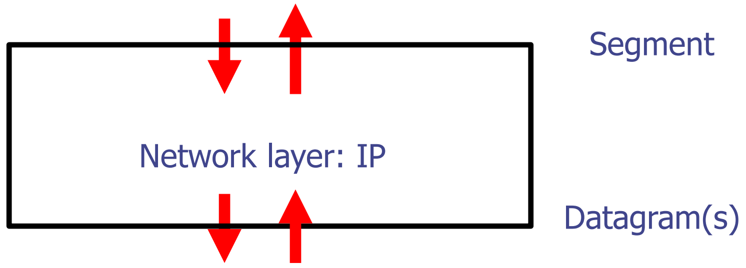


Every layer must...



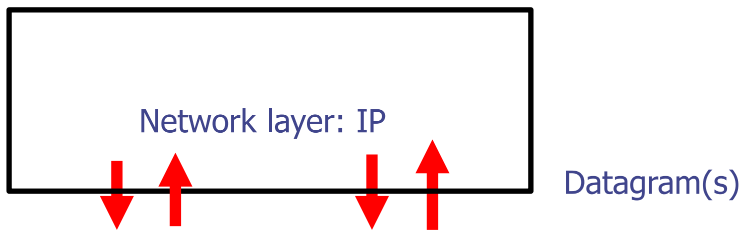
- offer services to an upper layer.
- comply with agreed protocols.
- utilize services from the underlying layer.

The Network Layer at the host…



- Retrieves and delivers segments from/to the Transport Layer.
- Comply with network layer protocols:
  - Data plane: IP (IPv4, IPv6).
  - Control plane: E.g. ICMP.
- Sends and receives datagrams to/from other layer 3 devices through the Link Layer.

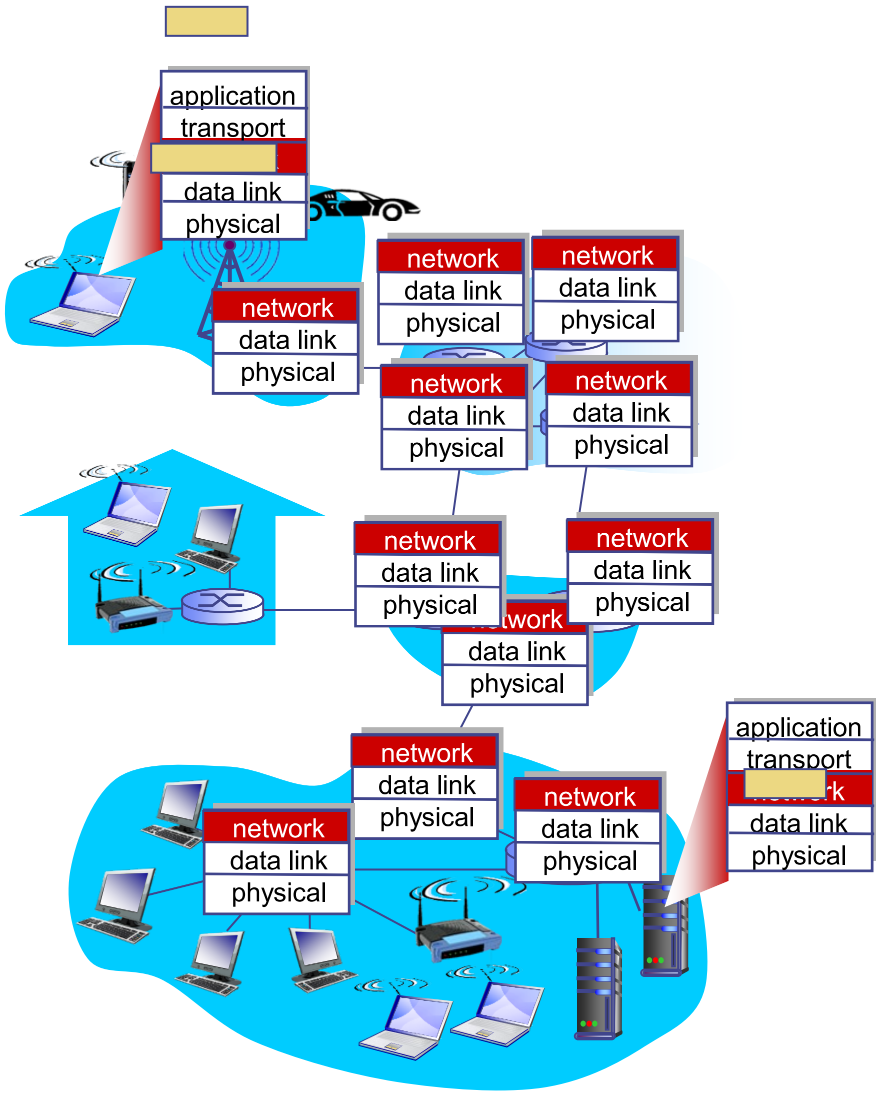
The Network Layer at the router…



- Comply with network layer protocols:
  - Data plane: IP (IPv4, IPv6).
  - Control plane: E.g. ICMP.
- Sends and receives datagrams to/from other layer 3 devices through the Link Layer.
- Forward datagrams from one interface (physical port) to another.
- Route datagrams to the “right” interface.

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



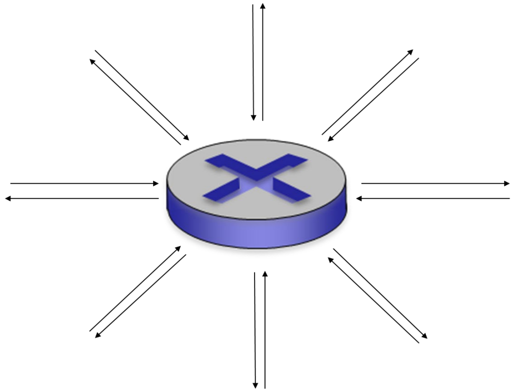
Two Key Network-Layer Functions

network-layer functions:

- *forwarding:* move packets from router's input to appropriate router output
- *routing:* determine route taken by packets from source to destination
  - *routing algorithms*

analogy: taking a trip

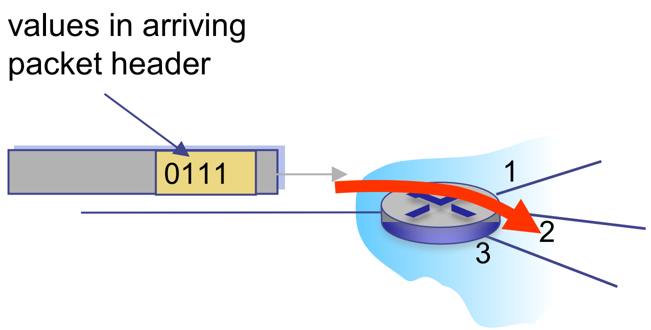
- *forwarding:* process of getting through single interchange
- *routing:* process of planning trip from source to destination



Network layer: Data plane vs. Control plane

Data plane

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

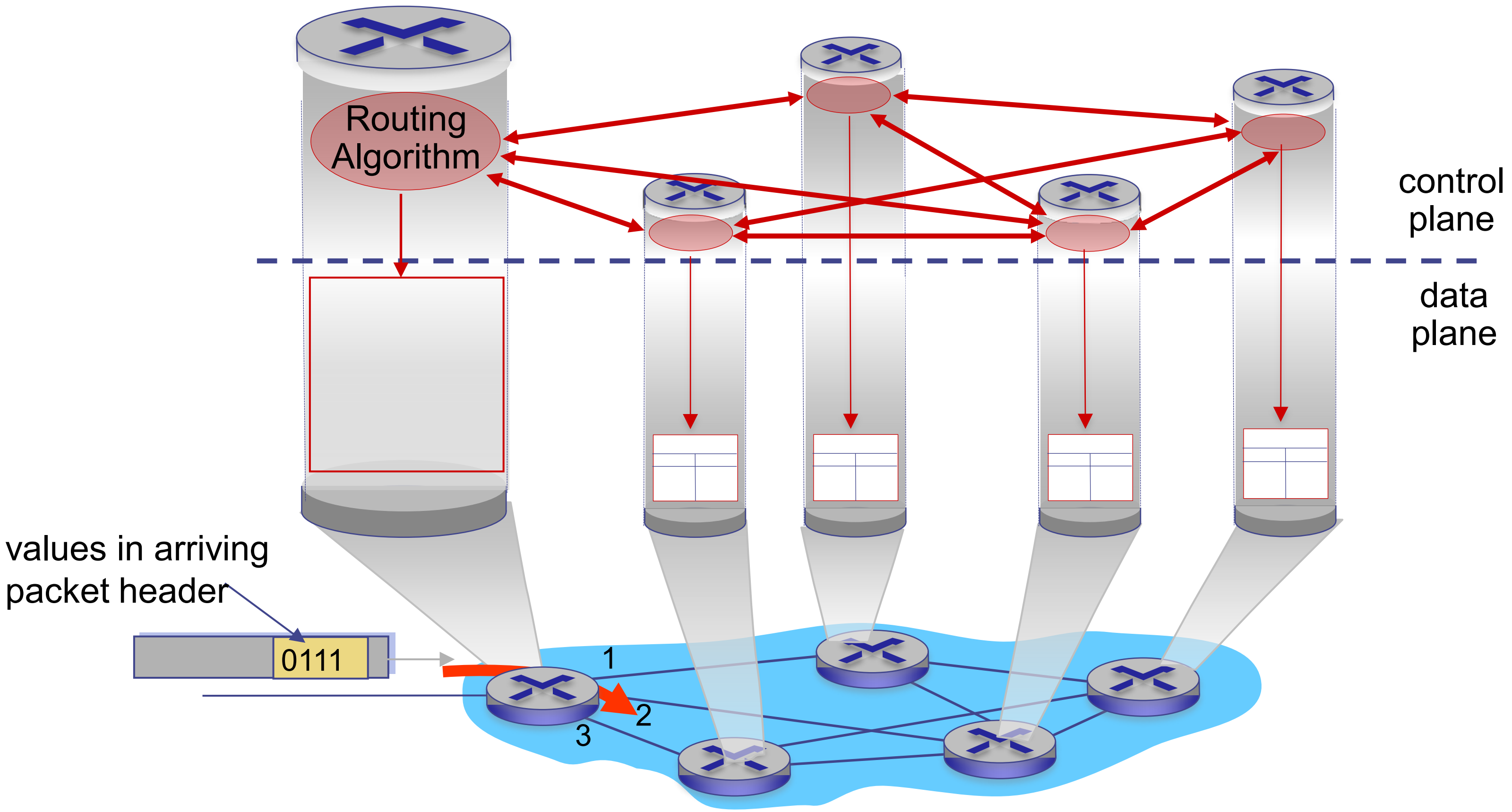


Control plane

- network-wide logic
- determines how datagram is routed among routers along end-2-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

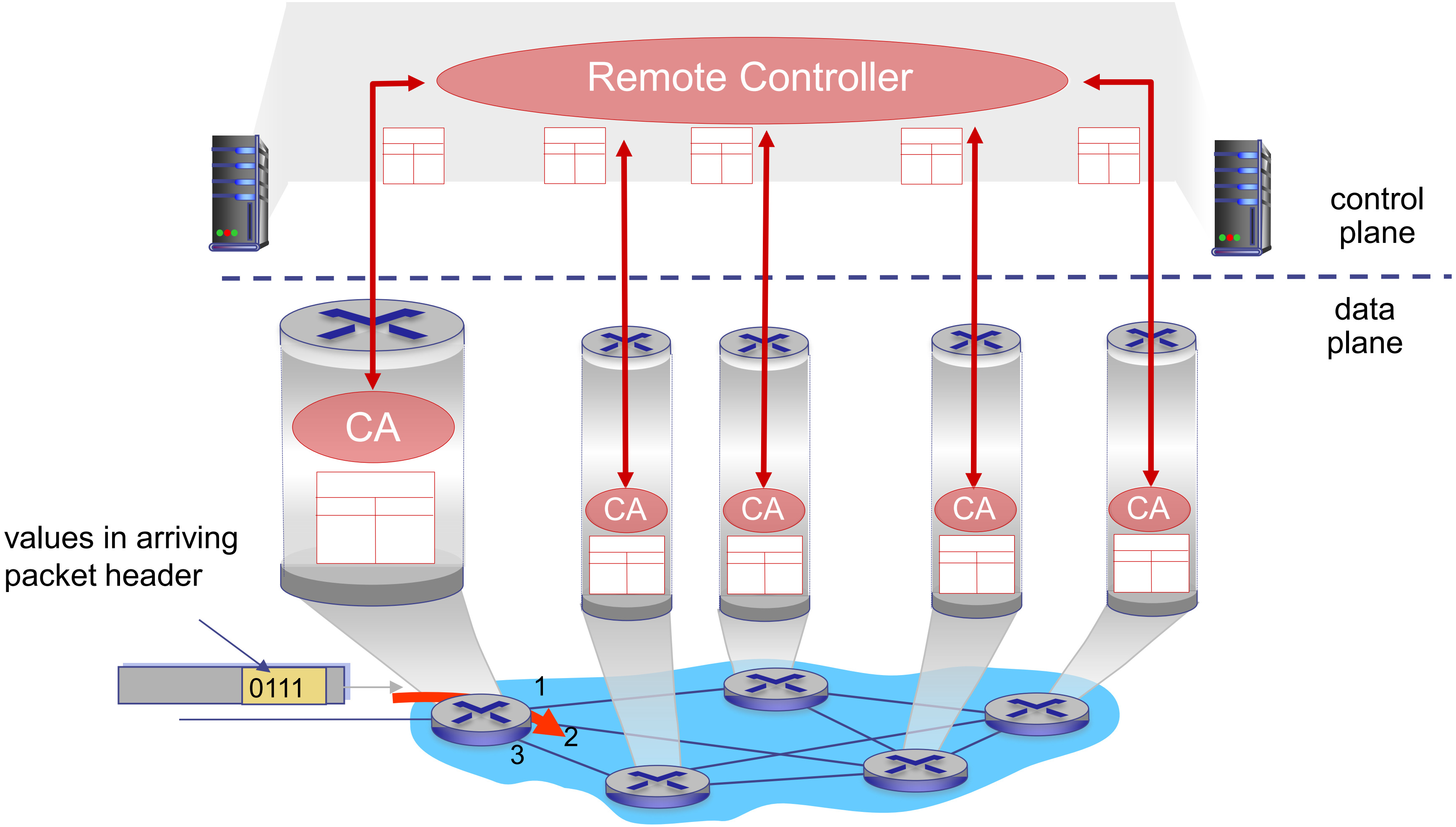
Per-router Control Plane

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



Logically Centralized Control Plane

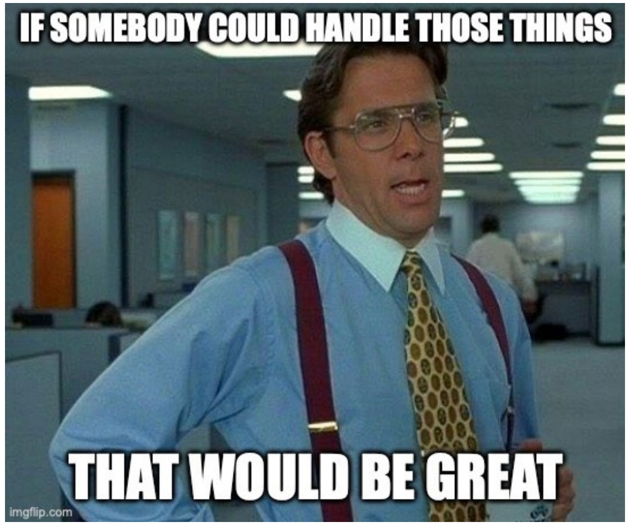
A distinct (typically remote) controller interacts with local control agents (CAs)



Network Service Model

Possibilities

- Guaranteed delivery (loss)
- Guaranteed delivery with bounded delay (time)
- In-order packet delivery
- Minimal bandwidth
- Security

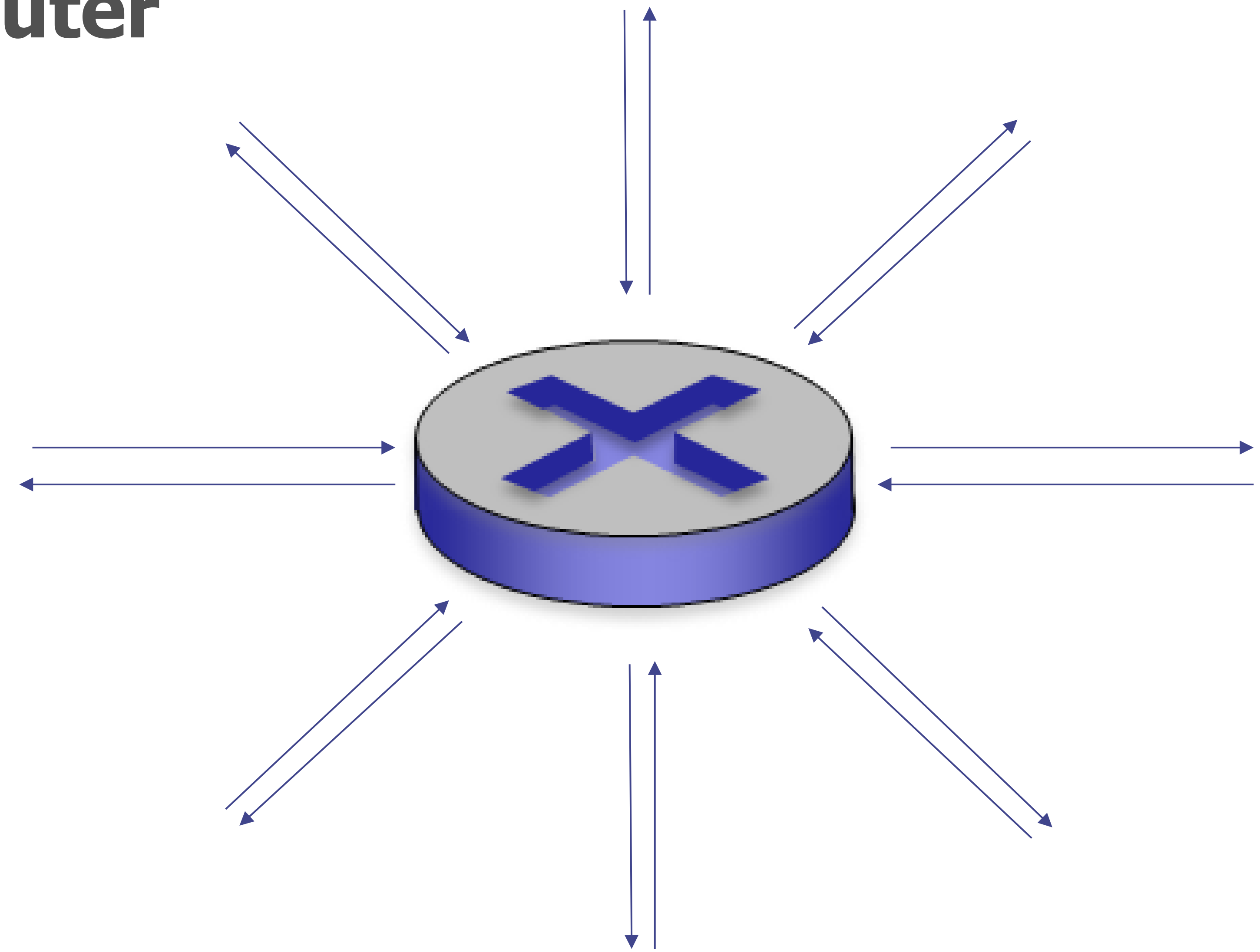


Actual

- Best-effort service…

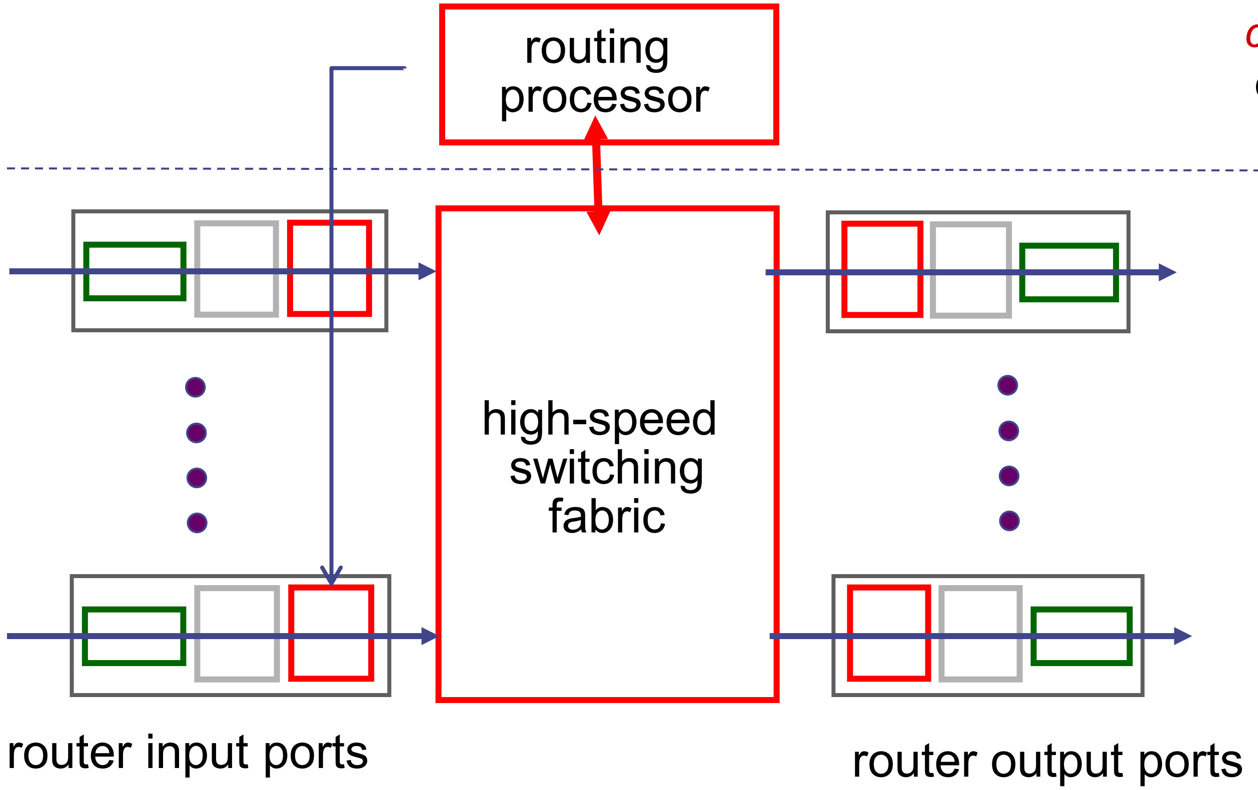
7.2 The Router

The Router



Router Architecture Overview

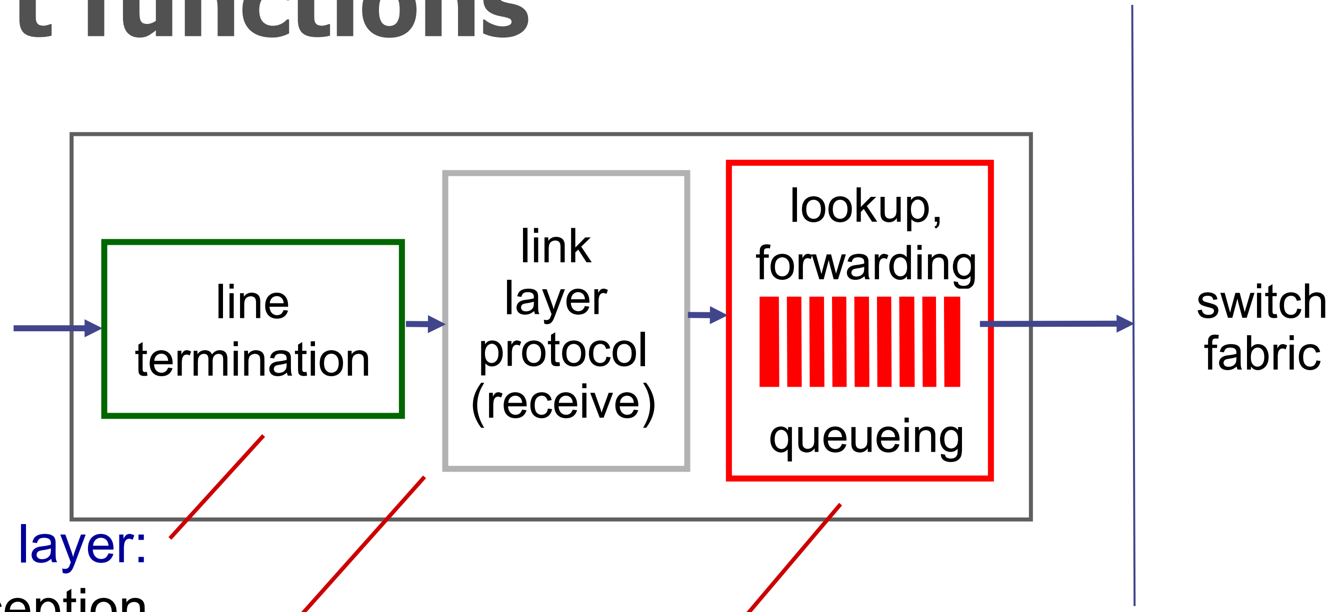
- high-level view of generic router architecture:



routing, management control plane (software) operates in millisecond time frame

forwarding data plane (hardware) operates in nanosecond timeframe

Input port functions



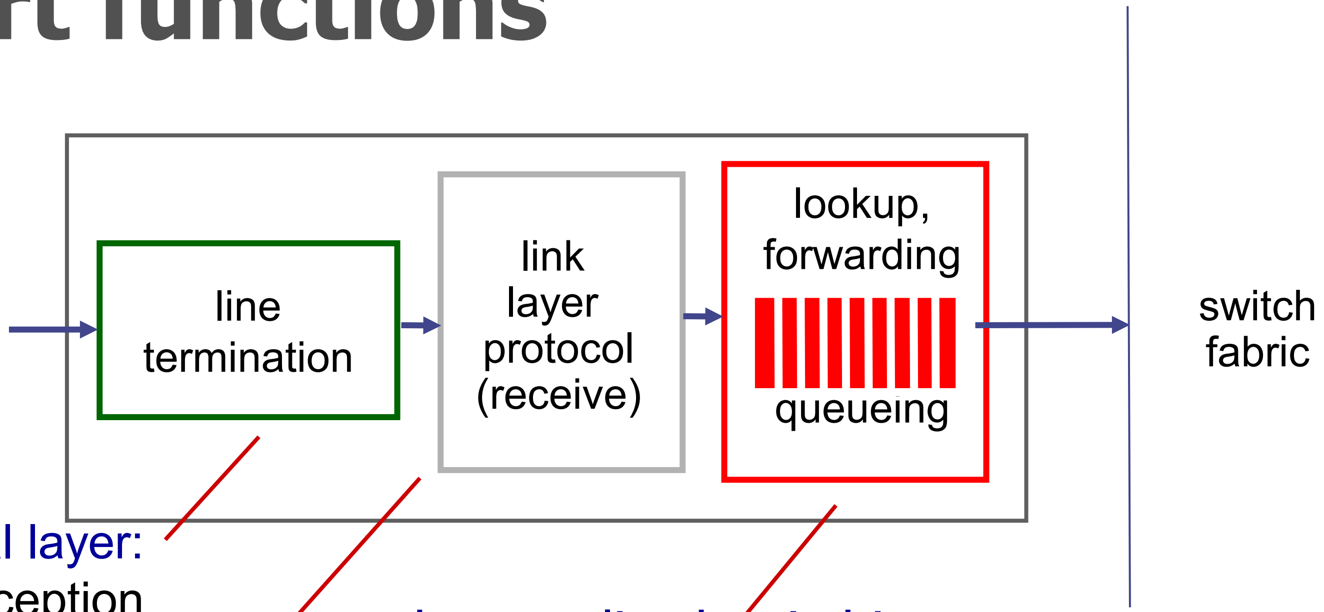
physical layer:
bit-level reception

data link layer:
e.g., Ethernet
see chapter 6

decentralized switching*:*

- using **header field** values, **lookup** output port using forwarding table in input port memory *("match plus action")*
- **goal**: complete input port processing at 'line speed'
- queuing: if datagrams arrive faster than forwarding rate into switch fabric

Input port functions



physical layer:
bit-level reception

data link layer:
e.g., Ethernet
see 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

Destination-based forwarding

forwarding table

Destination Address Range	Link Interface
11001000 00010111 00010000 00000000 through 11001000 00010111 00010111 11111111	0
11001000 00010111 00011000 00000000 through 11001000 00010111 00011000 11111111	1
11001000 00010111 00011001 00000000 through 11001000 00010111 00011111 11111111	2
otherwise	3

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

Longest prefix matching

longest prefix matching

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:

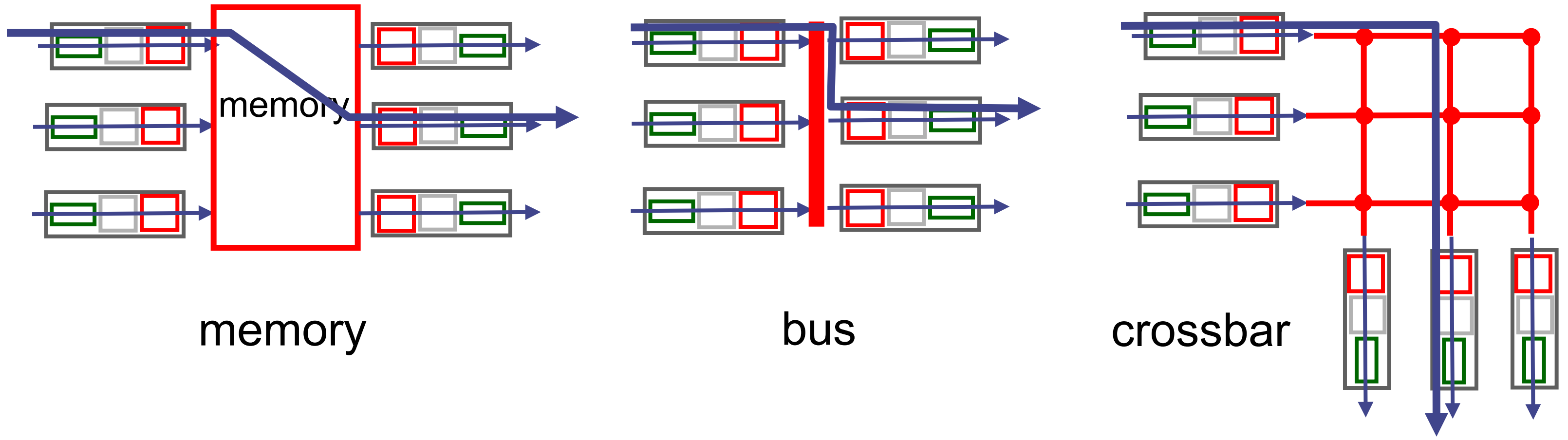
DA: 11001000 00010111 00010110 10100001 which interface?

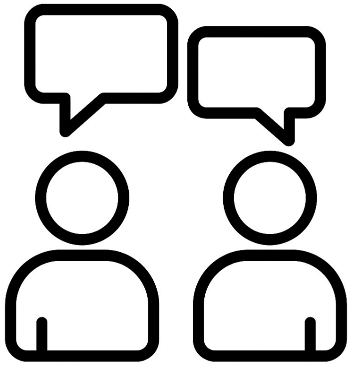
DA: 11001000 00010111 00011000 10101010 which interface?

SDU

Switching fabrics

- transfer packet from input buffer to appropriate output buffer
- switching rate: rate at which packets can be transfer from inputs to outputs
- three types of switching fabrics:



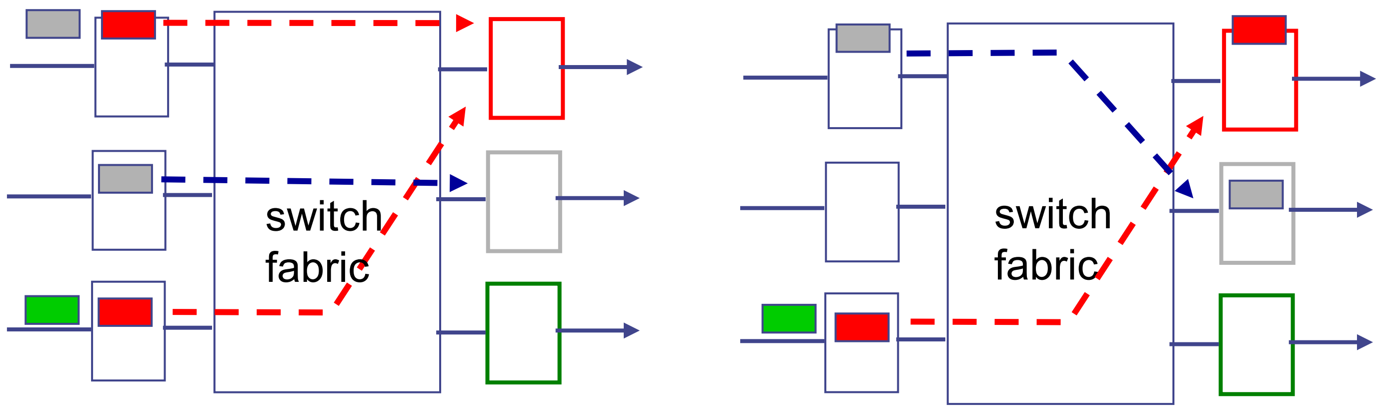


What are the drawbacks of each method?

SDU

Input port queuing

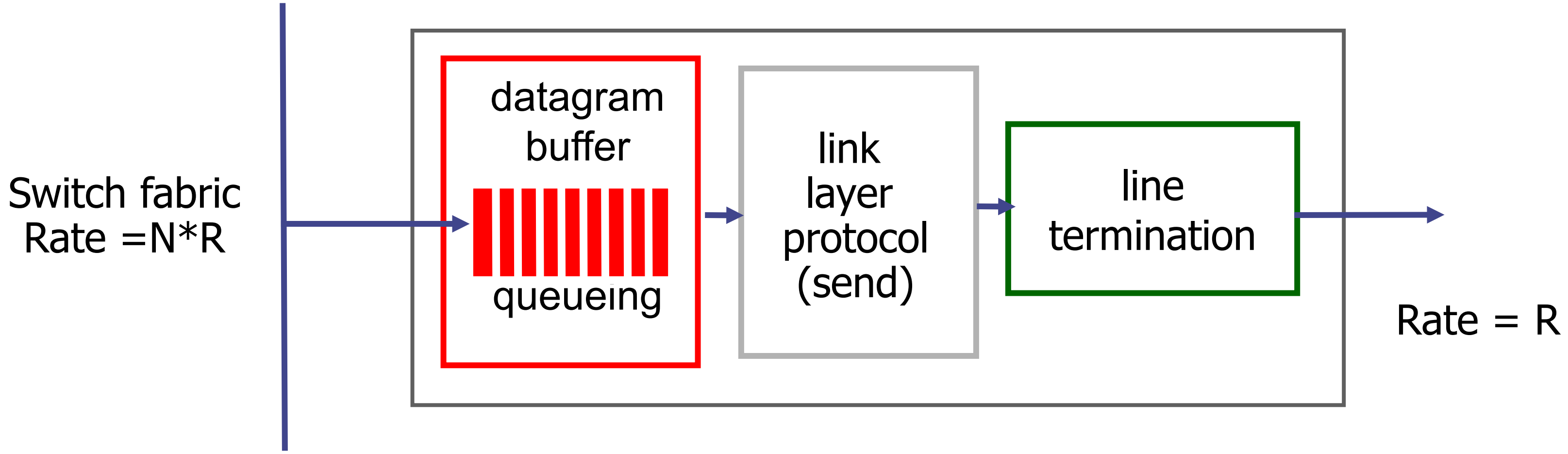
- fabric slower than input ports combined -> queueing may occur at input queues
    - *queueing delay and loss due to input buffer overflow!*
- Head-of-the-Line (HOL) blocking: queued datagram at front of queue prevents others in queue from moving forward



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

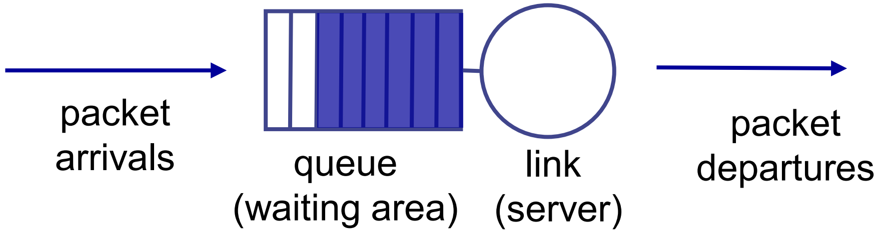
Datagram (packets) can be lost due to congestion/lack of buffers

- *scheduling discipline* chooses among queued datagrams for transmission

Priority scheduling – who gets best performance?

Scheduling Mechanisms

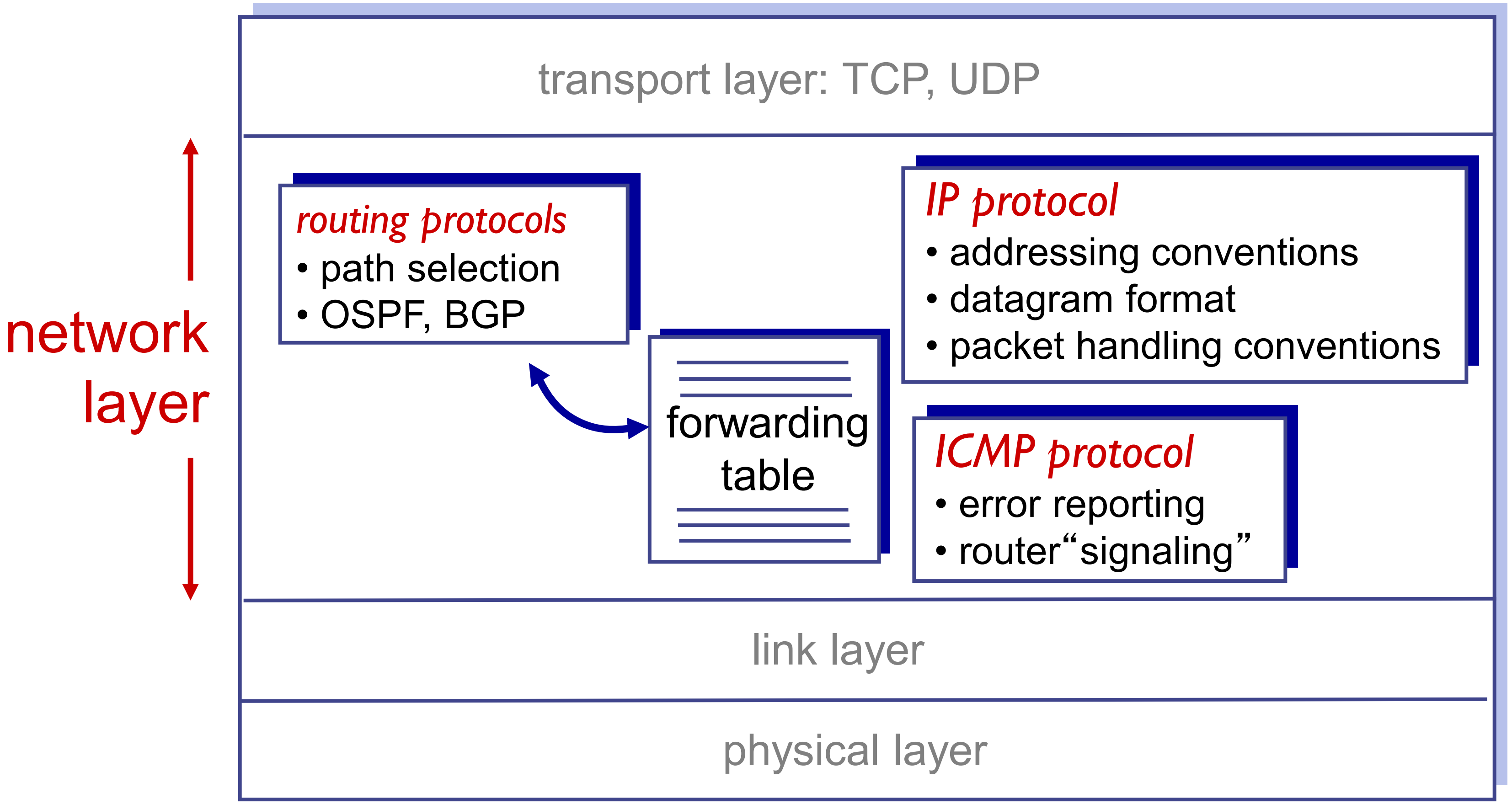
- Scheduling: choose next packet to send on link
- FIFO (first in first out) scheduling: send in order of arrival to queue
  - Discard policy: if packet arrives to full queue: who to discard?
    - Tail drop: drop arriving packet
    - Priority: drop/remove on priority basis
    - Random: drop/remove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



7.3 Internet Protocol Addressing

The Internet Network Layer

Host and router network layer functions:



IP datagram format

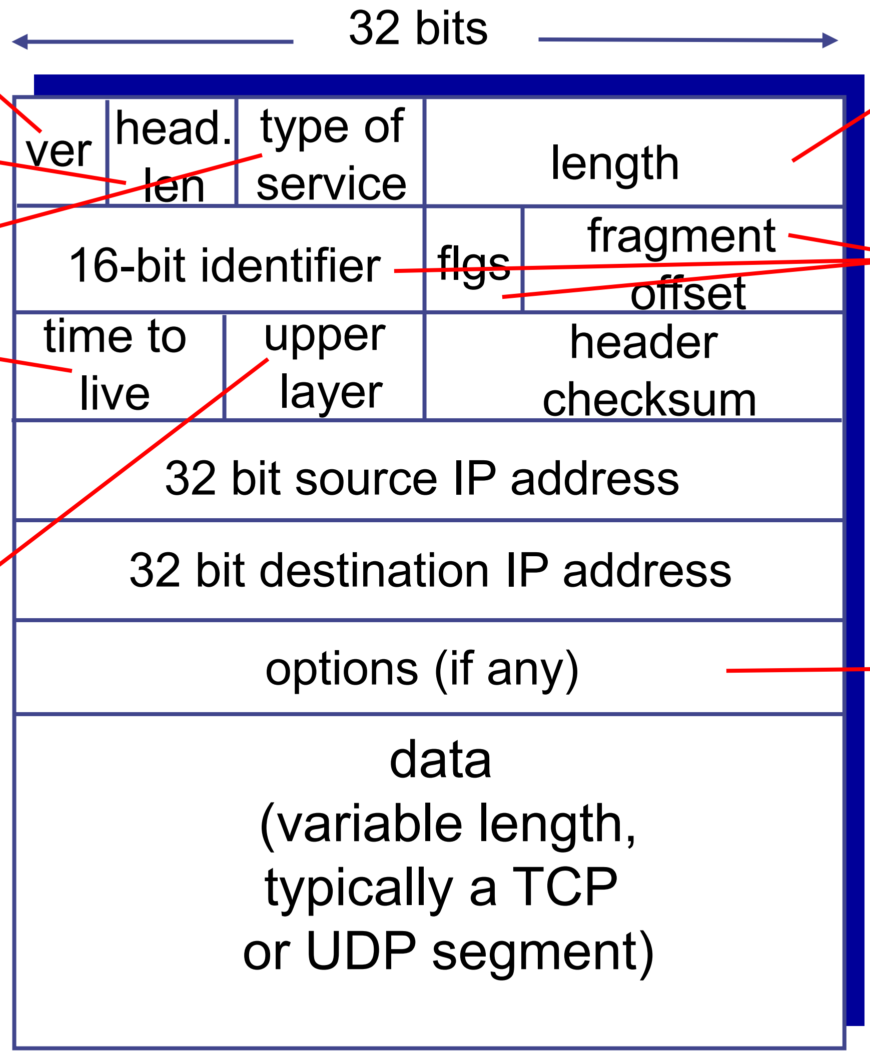
IP protocol version number

header length (Number of 32 bit words)

“type” of data

max number remaining hops (decremented at each router)

upper layer protocol to deliver payload to



total datagram length (bytes)

for fragmentation/ reassembly

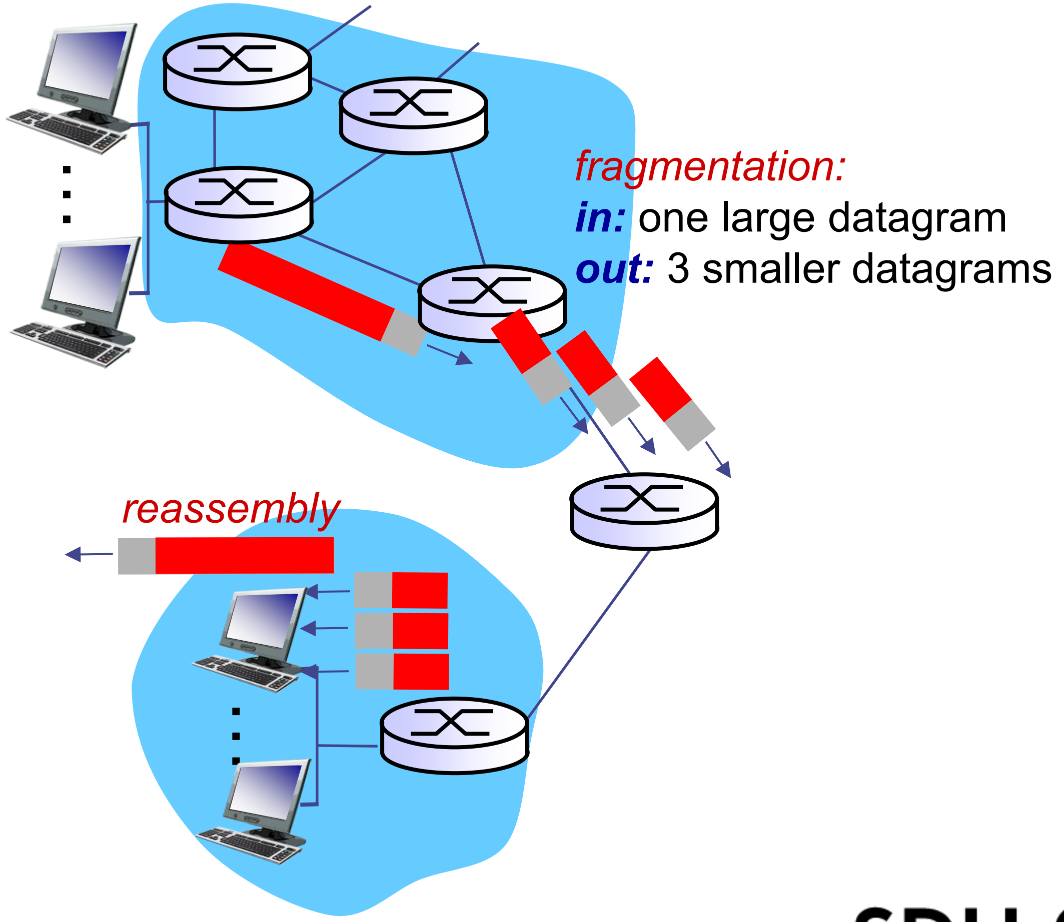
e.g. timestamp, record route taken, specify list of routers to visit.

how much overhead?

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

IP fragmentation, reassembly

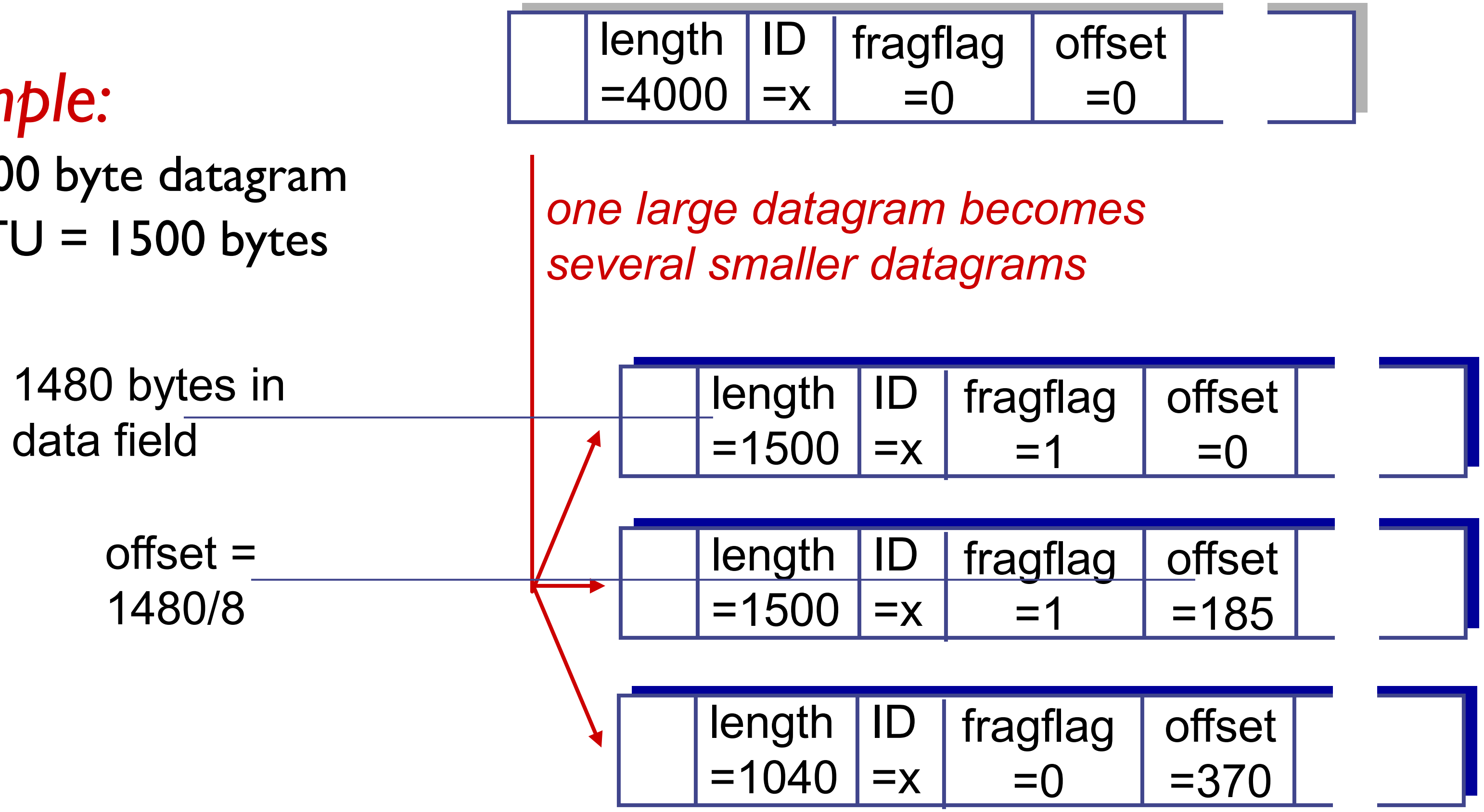
- network links have MTU (max.transfer size) - largest possible link-level frame
  - different link types, different MTUs
- large IP datagram divided ("fragmented") within net
  - one datagram becomes several datagrams
  - "reassembled" only at final destination
  - IP header bits used to identify, order related fragments

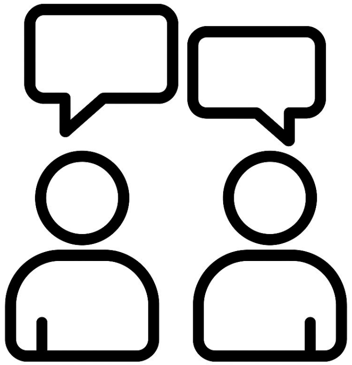


IP Fragmentation & Reassembly

example:

- 4000 byte datagram
- MTU = 1500 bytes

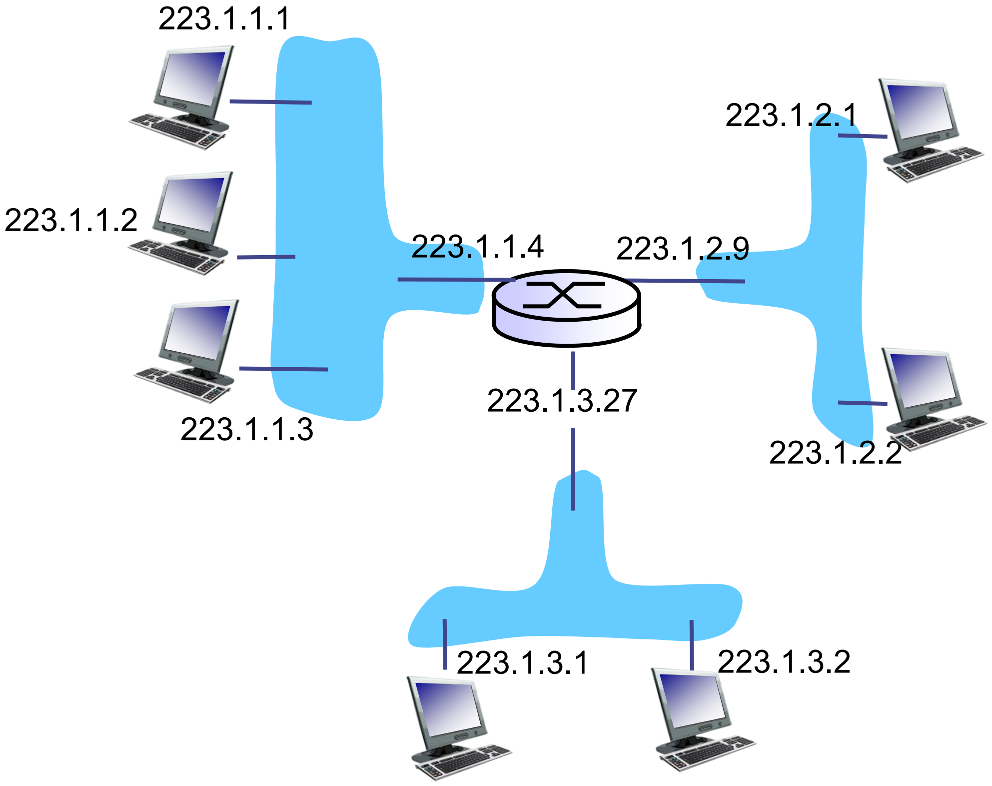




What happens if two hosts sends with the same ID?

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 typically has one or two interfaces (e.g., wired Ethernet, wireless 802.11)
- *IP addresses associated with each interface*

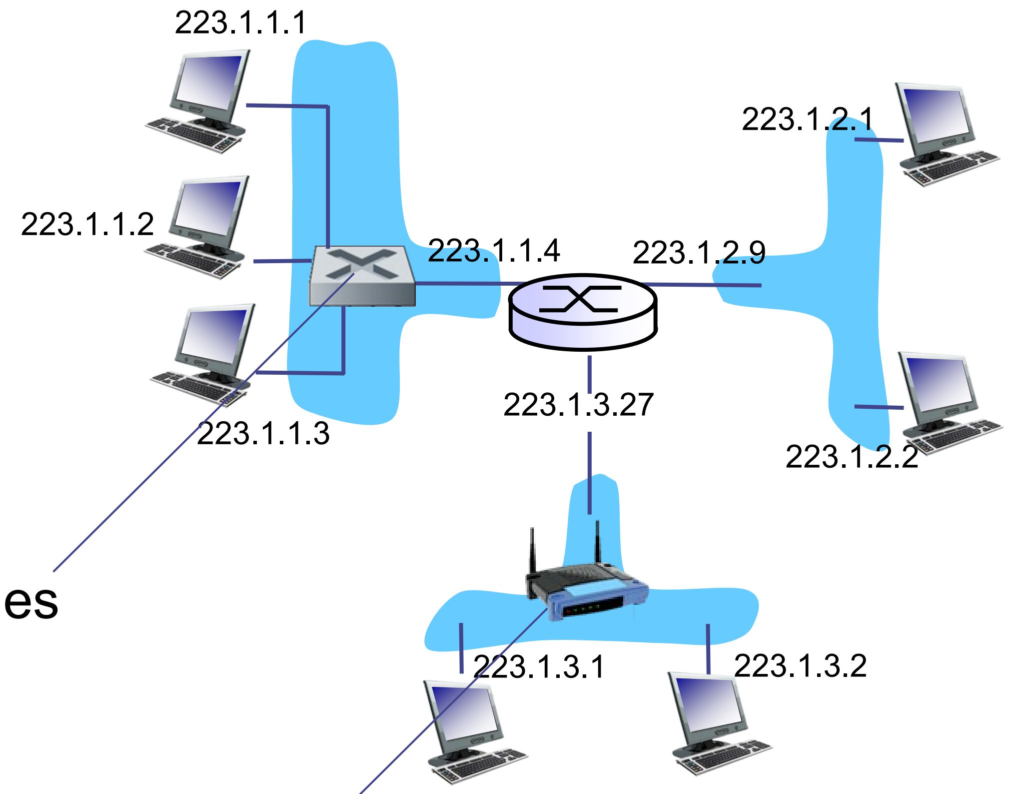


$$223.1.1.1 = \underbrace{11011111}_{223}\ \underbrace{00000001}_{1}\ \underbrace{00000001}_{1}\ \underbrace{00000001}_{1}$$

IP Addressing: Introduction

Q: how are interfaces actually connected?

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



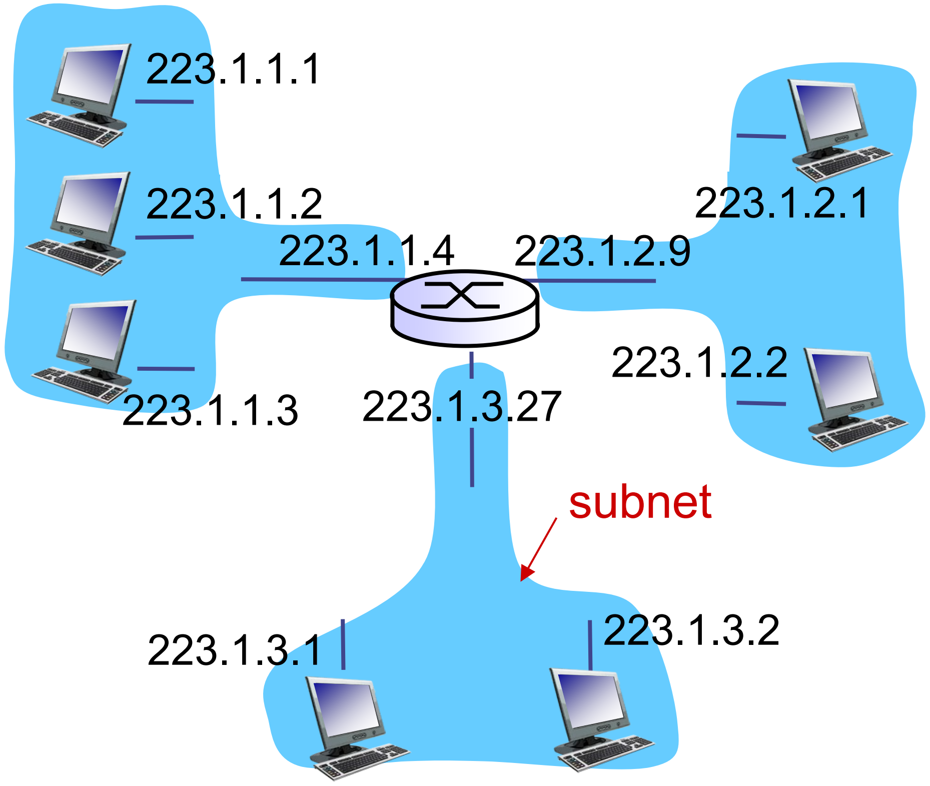
A: wired Ethernet interfaces connected by Ethernet switches

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

A: wireless WiFi interfaces connected by WiFi base station

Subnets

- IP address:
  - subnet part - high order bits
  - host part - low order bits
- *What's a subnet ?*
  - device interfaces with same subnet part of IP address
  - can physically reach each other *without intervening router*

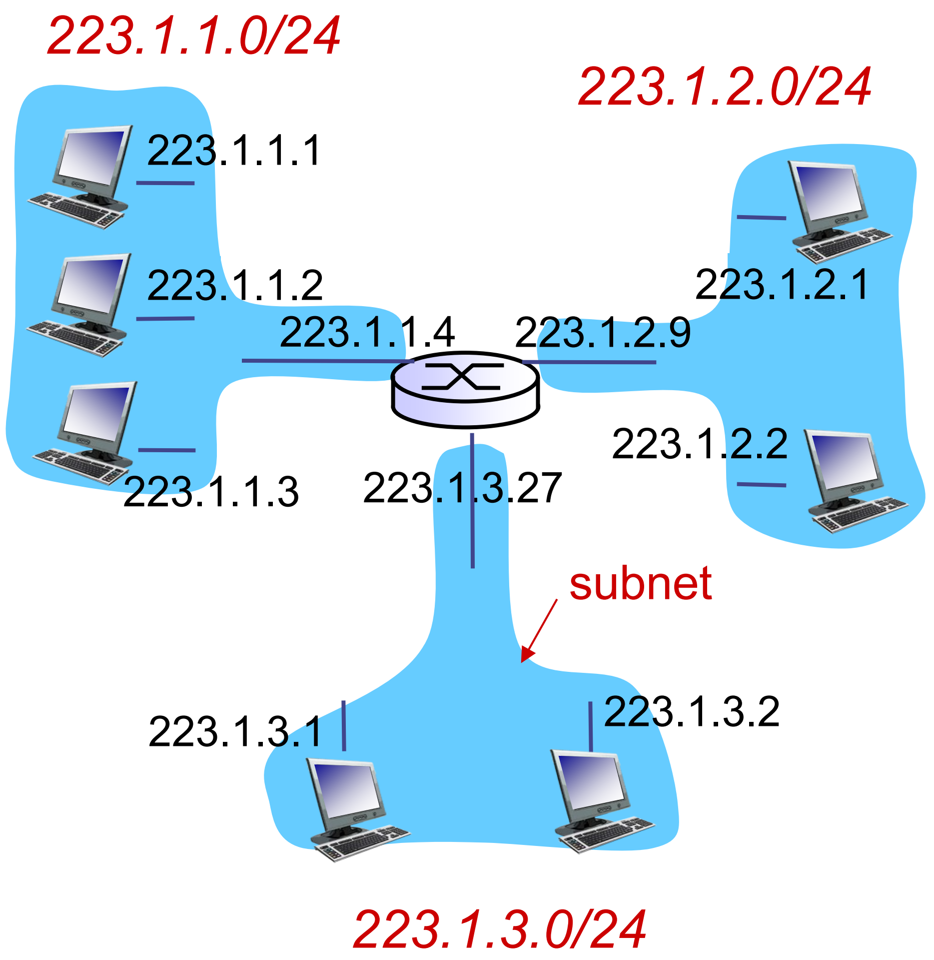


network consisting of 3 subnets

Subnets

recipe

- to determine the subnets, detach each interface from its host or router, creating islands of isolated networks
- each isolated network is called a *subnet*



subnet mask: /24

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

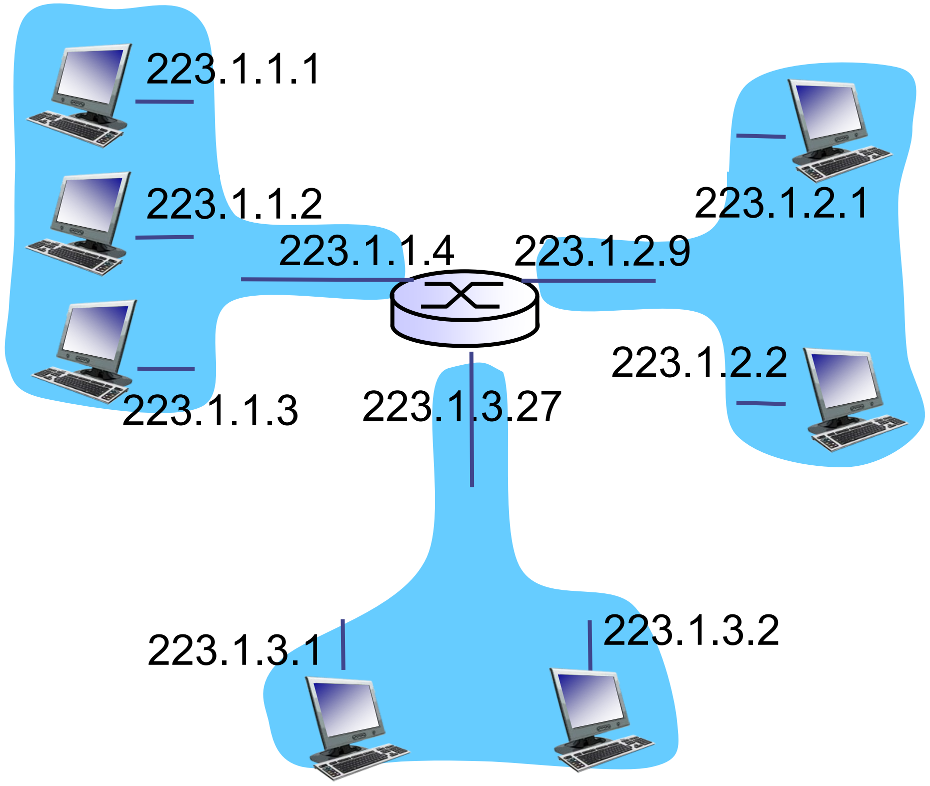
subnet part   host part

11001000 00010111 00010000 00000000

200.23.16.0/23

Subnets again...

How many subnets if the subnet mask was: /16?



IP addresses: How to get one?

- Hard-coded by system admin in a file
- DHCP: **D**ynamic **H**ost **C**onfiguration **P**rotocol: dynamically get address from as server
  - "plug-and-play"

DHCP: Dynamic Host Configuration Protocol

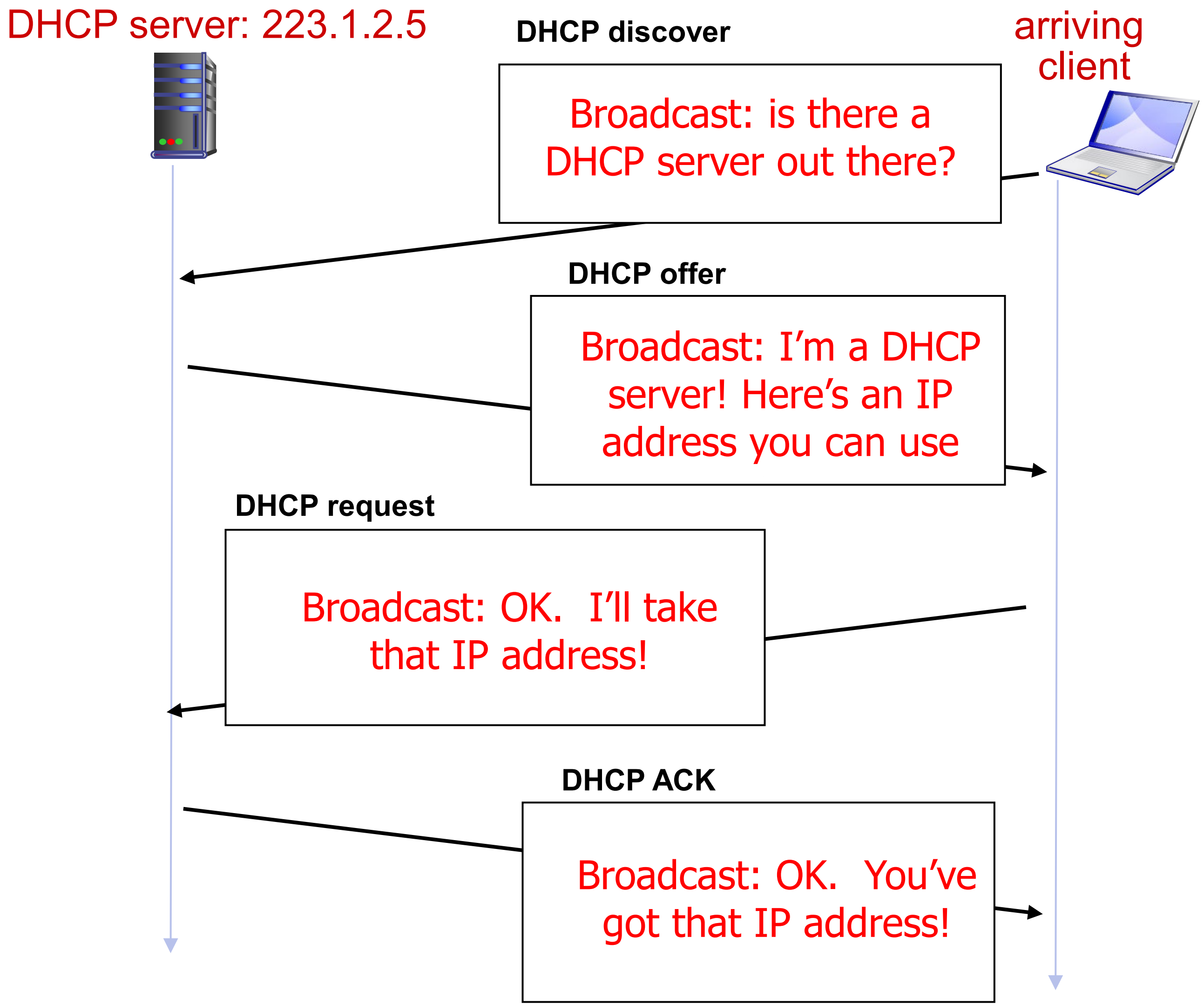
Goal: allow host to *dynamically* obtain its IP address from network server when it joins network

- 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

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 client-server scenario



DHCP: more than IP addresses

DHCP can return more than just an allocated IP address on subnet:

- address of first-hop router for client
- name and IP address of DNS sever
- network mask (indicating network versus host portion of address)

IP addresses: how to get one?

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

A: Gets the 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

IP addressing: the last word...

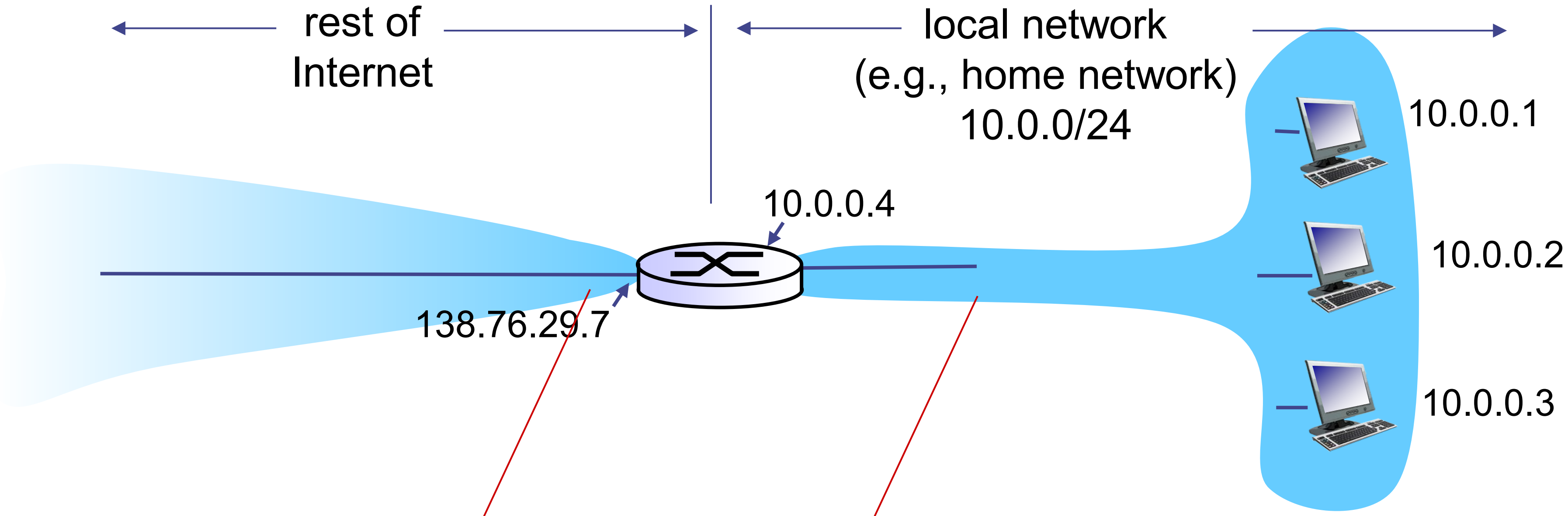
Q: How does an ISP get block of addresses?

A: ICANN: Internet Corporation for Assigned Names and Numbers https://www.icann.org/

- allocates addresses
- manages DNS
- assigns domain names, resolves disputes

7.4 Beyond IPv4, NAT and IPv6

NAT: Network Address Translation



all datagrams *leaving* local network have *same* single source 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

Motivation: Local network uses just one IP address as far as outside world is concerned:

- Range of addresses not needed from ISP: just one IP address 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

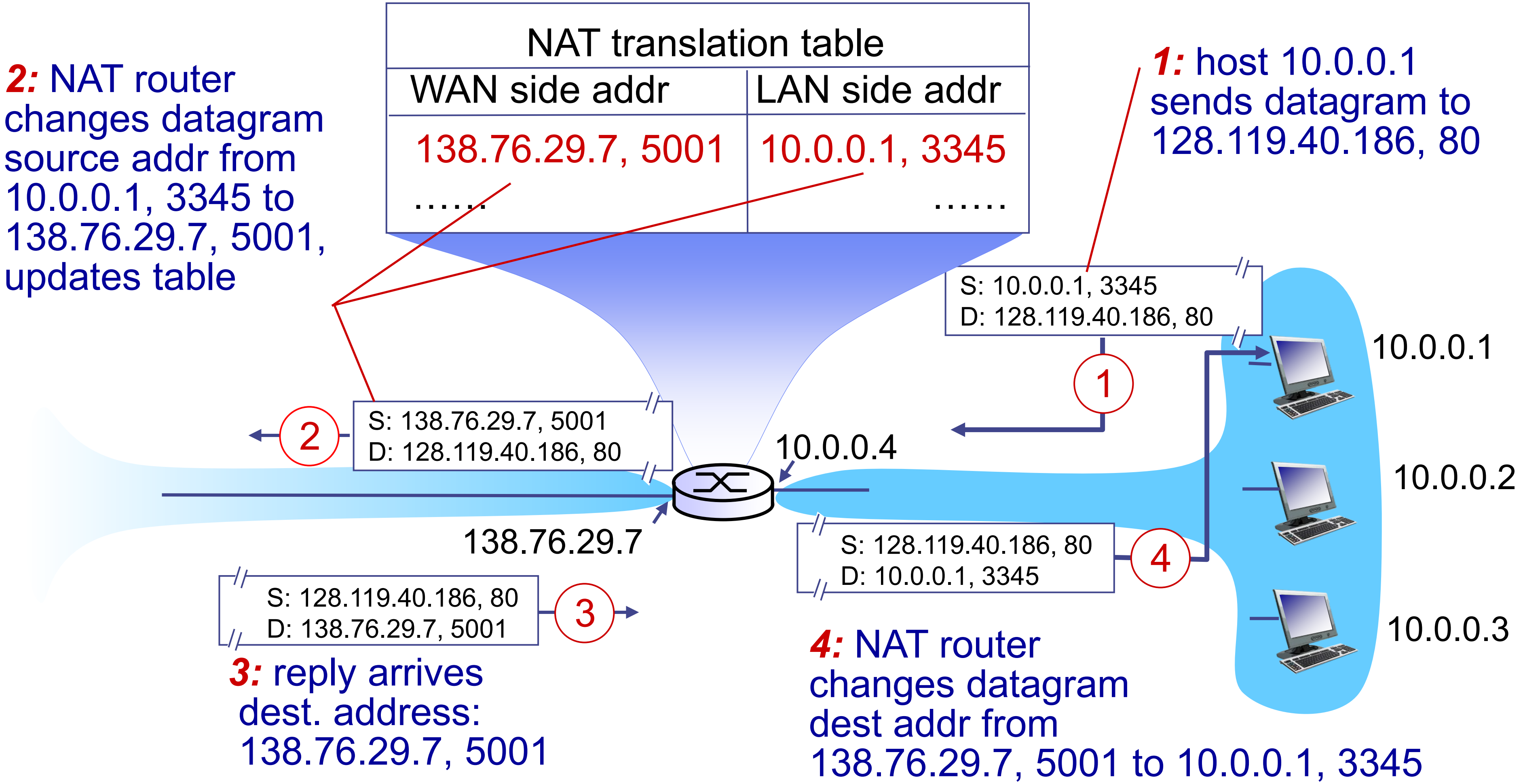
- *implementation*: NAT router must:

  - *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 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

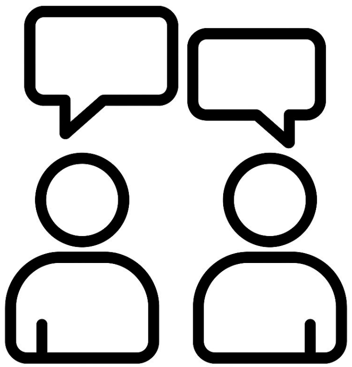
NAT: Network Address Translation



* Check out the online interactive exercises for more examples: http://gaia.cs.umass.edu/kurose_ross/interactive/

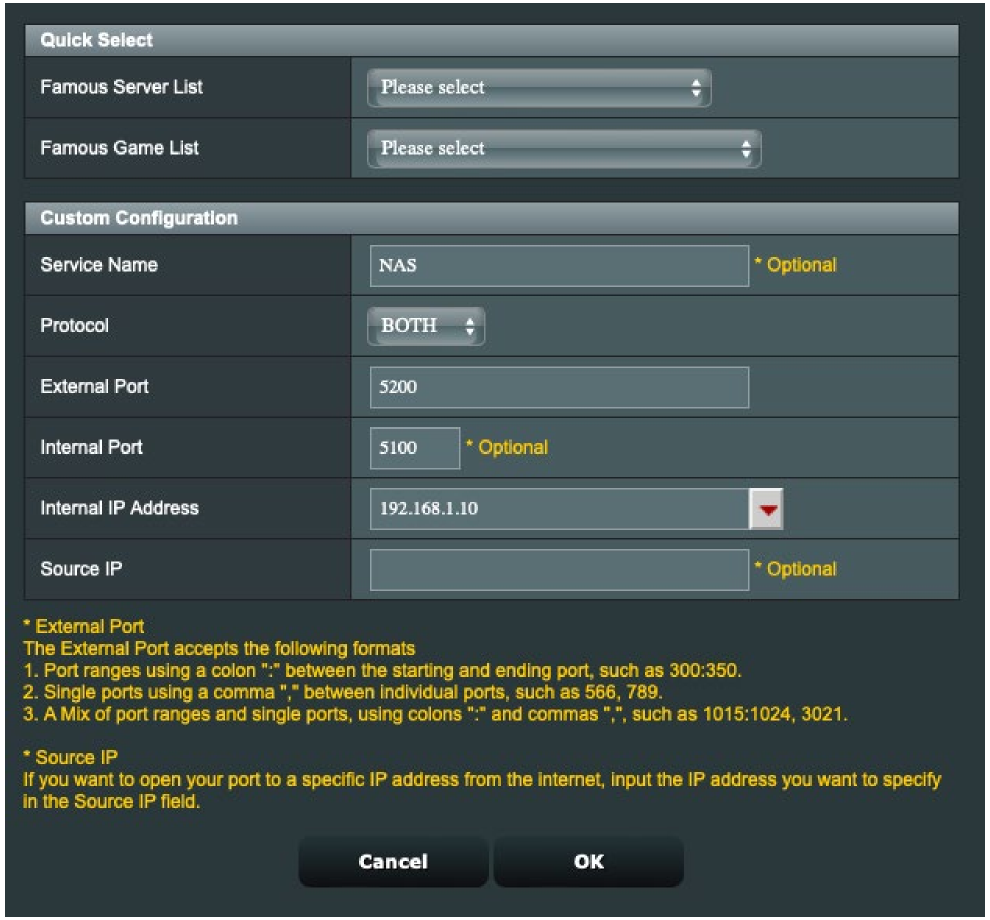
NAT: Network Address Translation

- 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
  - Address shortage should be solved by IPv6
  - Violates end-to-end argument
    - NAT possibility must be taken into account by app designers, e.g., P2P applications



What if a client wants to connect to server behind NAT?

Port Forwarding



IPv6: Motivation

- 32-bit address space completely allocated[1]
- Additional motivation:
  - header format helps speed processing/forwarding
  - header changes to facilitate QoS

IPv6 datagram format:

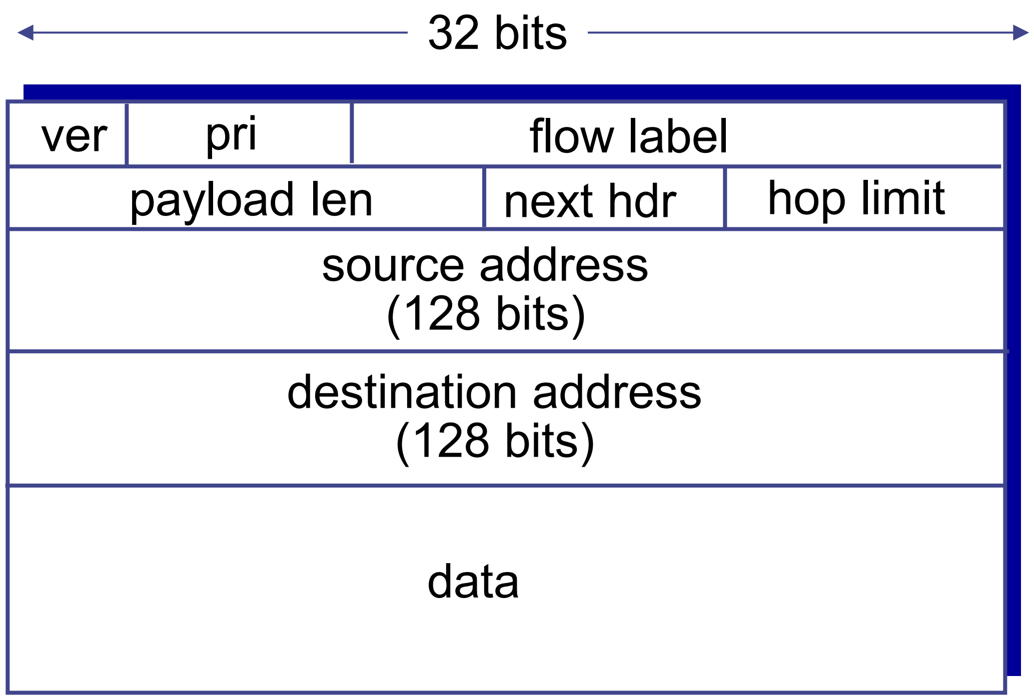
  - fixed-length 40 byte header
  - no fragmentation allowed
  - no checksum
  - no options

- *ICMPv6:* new version of ICMP
  - additional message types, e.g. "Packet Too Big"
  - multicast group management functions

[1]https://www.ripe.net/manage-ips-and-asns/ipv4/ipv4-run-out/

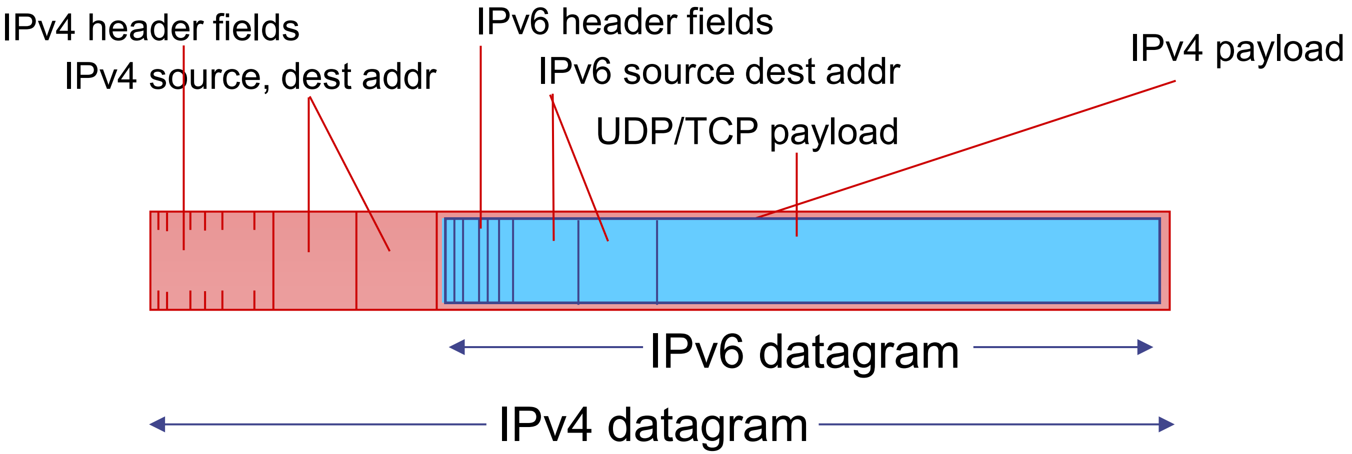
IPv6 Datagram Format

priority: identify priority among datagrams in flow
flow Label: identify datagrams in same "flow." (concept of"flow" not well defined).
next header: identify upper layer protocol for data (in the payload)

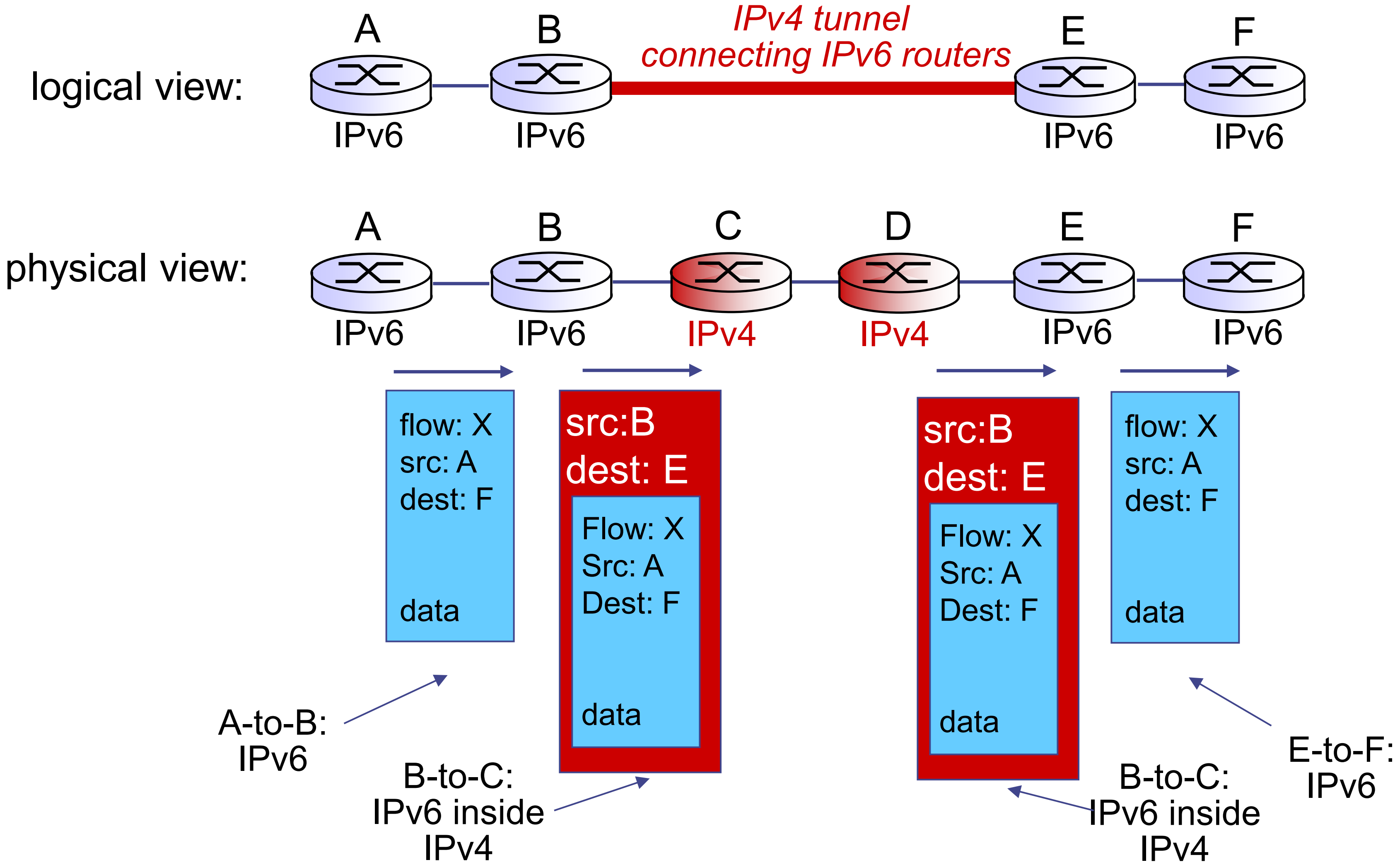


Transitioning from IPv4 to IPv6

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



Tunneling


logical view:
A
B
IPv4 tunnel
connecting IPv6 routers
E
F
IPv6
IPv6
IPv6
IPv6
physical view:
A
B
C
D
E
F
IPv6
IPv6
IPv4
IPv4
IPv6
IPv6
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
E-to-F:
IPv6

IPv6: adoption

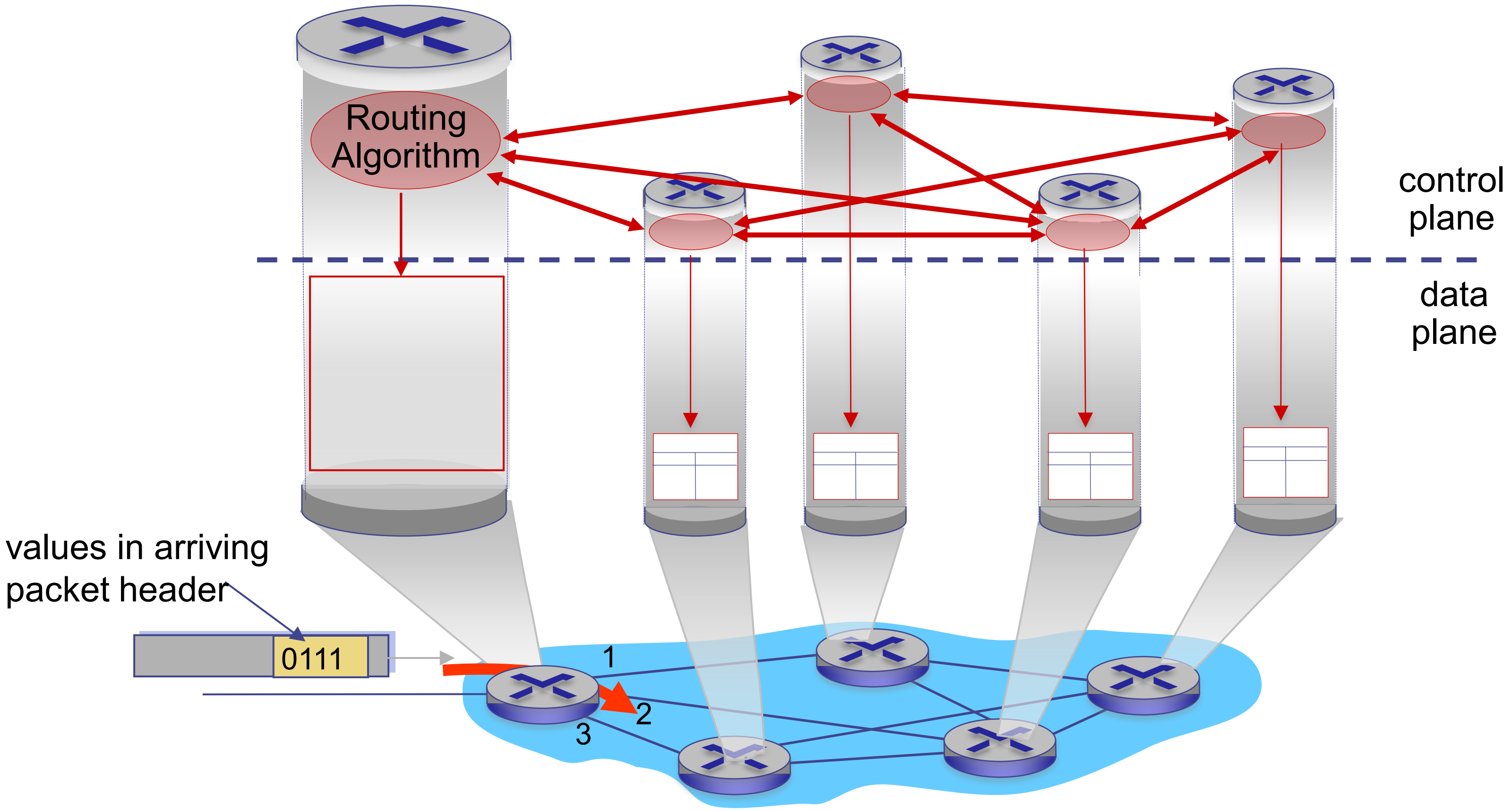
- Google: 30% of clients access services via IPv6
- NIST: 1/3 of all US government domains are IPv6 capable

- *Long (long!) time for deployment, use*
  - 25 years and counting!
  - think of application-level changes in last 25 years: WWW, social media, streaming media, gaming, telepresence, …
  - *Why?*

7.5 Generalized forwarding in brief

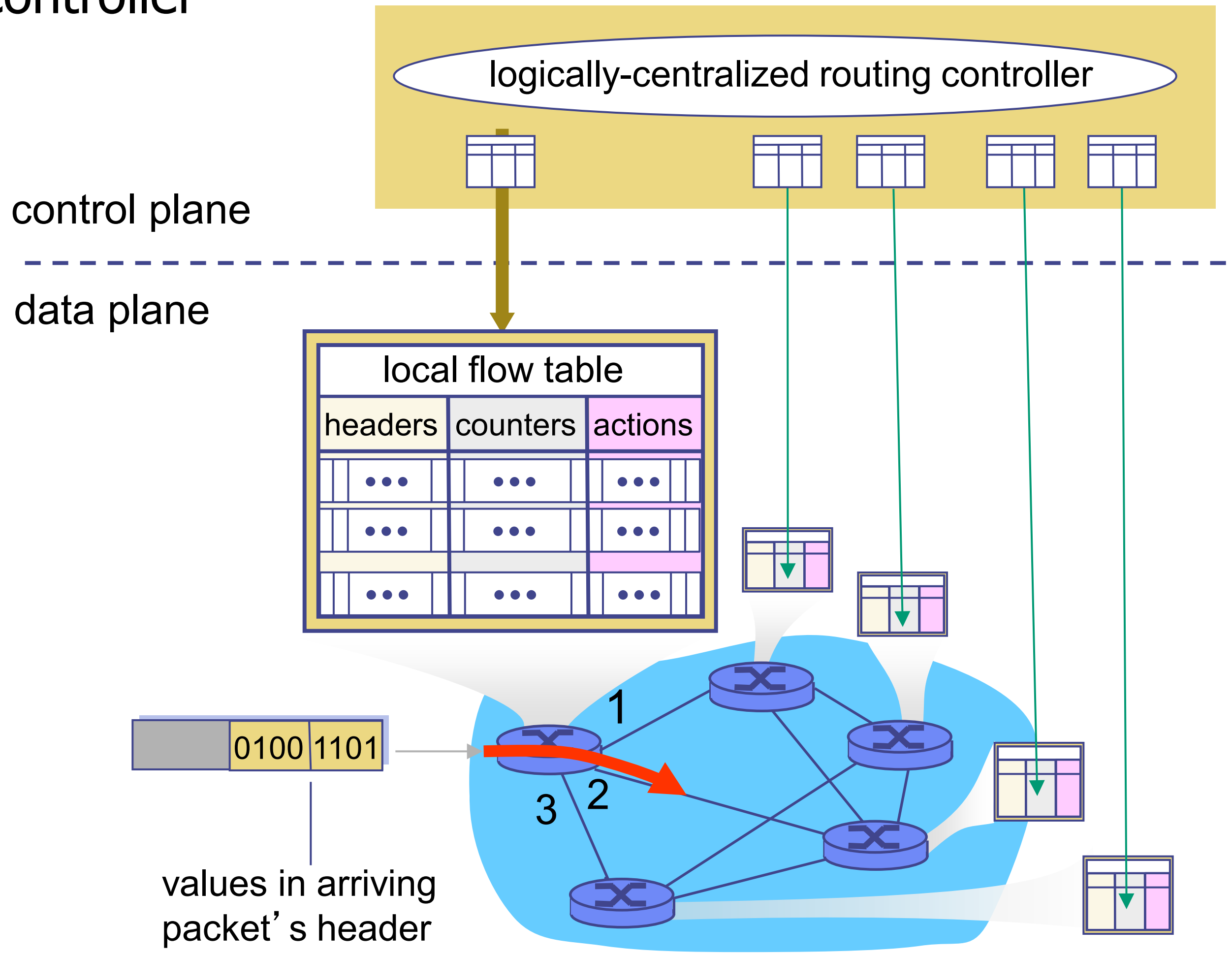
Per-router Control plane

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



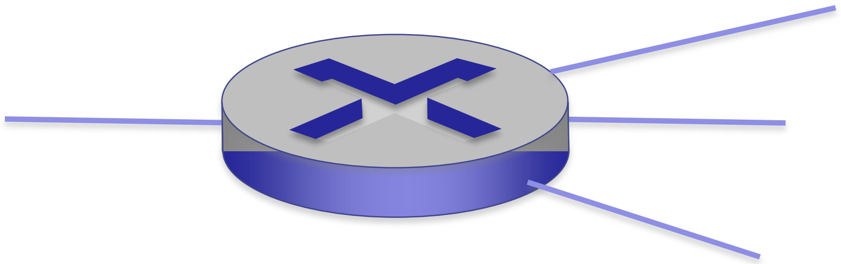
Generalized Forwarding and SDN

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



OpenFlow data plane abstraction

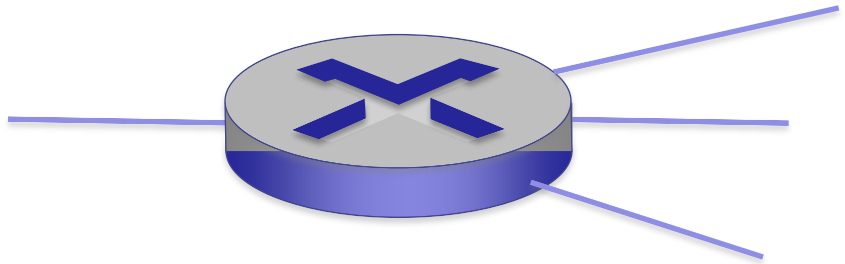
- *Flow*: defined by header fields
- Generalized forwarding: simple packet-handling rules
  - *Pattern:* **match** values in packet **header fields**
  - *Actions: for matched packet:* drop, forward, modify, matched packet or send matched packet to controller
  - *Priority*: disambiguate overlapping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



* : 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 SrcPort	TCP DstPort

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

Next session, we will talk abut the control plane of the network layer.
Read chapter 5 in the book (page 407-467).

Today we introduce the second hand-in assignment of this course, you must hand in a journal of your work with Wireshark Lab. 6 in the hand-in folder at itsLearning. Deadline is 14/12-2025.

As with the first hand-in ”Python Lab 3” the hand-in is mandatory for all that adhere to the “Data communications” T590000101 course description.

The conditions of the hand in, can be found in the “Lab 6” plan at itslearning/resources.