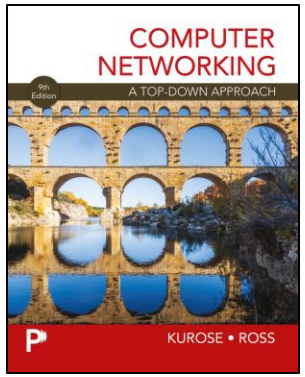


CSC 361
Computer Networks
Network Layer
(Chapter 4 – Data Plane)

Wenjun Yang

Summer 2026

Network Layer: Our Goals

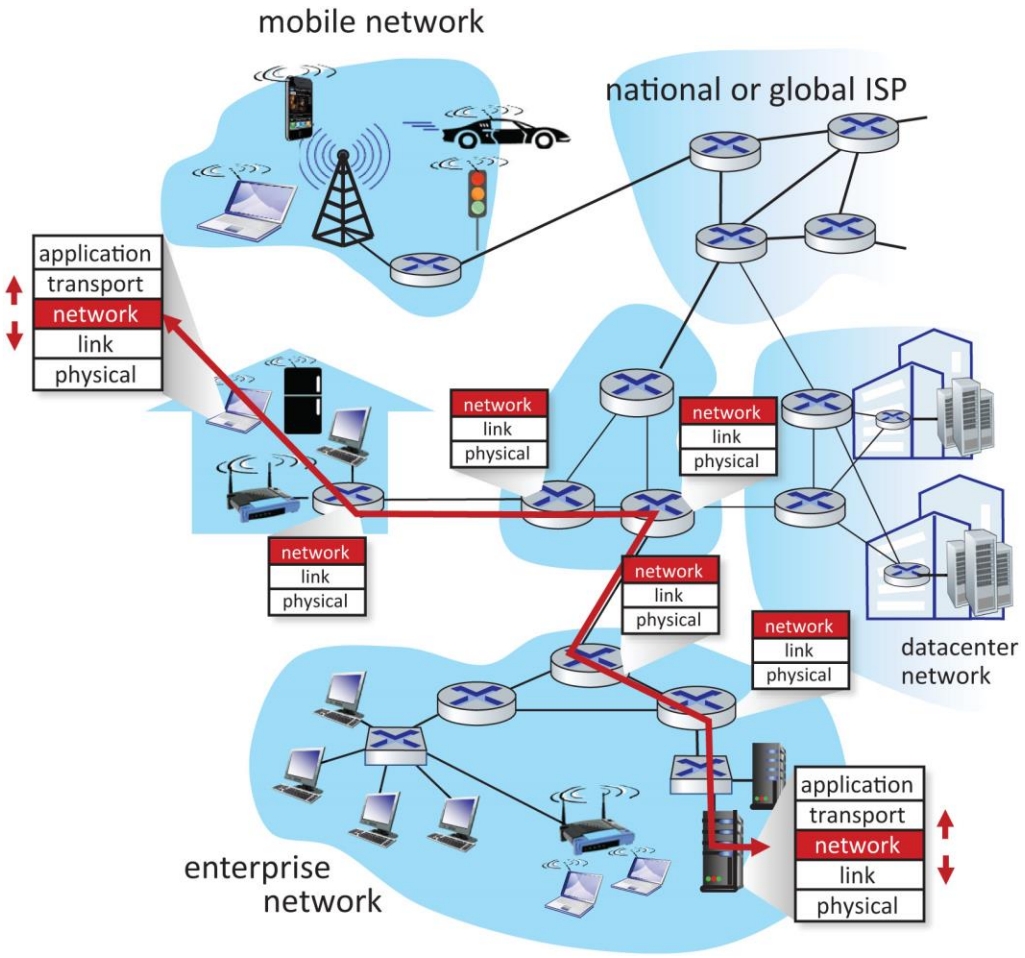
- understand principles behind network layer services, focusing on data plane:
  - network layer service models
  - forwarding versus routing
  - how a router works
  - addressing
  - generalized forwarding
  - Internet architecture
- instantiation, implementation in the Internet
  - IP protocol
  - NAT, middleboxes

Network Layer: "Data plane" Roadmap (1 of 6)

- Network layer: overview
  - data plane
  - control plane
- What's inside a router
  - input/output ports, switching
  - buffer management
  - scheduling
- IP: the Internet Protocol
  - datagram format, addressing
  - network address translation (NAT)
  - IPv6
- Generalized Forwarding
  - Match+action
  - OpenFlow
  - Middleboxes
- Internet architecture

Network-Layer Services and Protocols

- transport segment from sending to receiving host
  - sender: encapsulates segments into datagrams, passes to link layer
  - receiver: delivers segments to transport layer protocol
- network layer protocols in **every Internet device:** hosts, routers
- routers:
  - examines header fields in all IP datagrams passing through it
  - moves datagrams from input ports to output ports to transfer datagrams along end-end path



Two key Network-Layer Functions

network-layer functions:

- **forwarding:** move packets from a router's input link to appropriate router output link
- **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

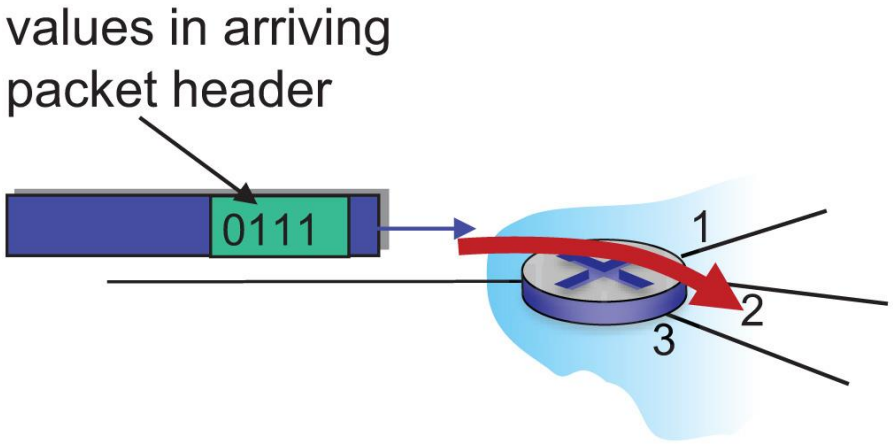
forwarding

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

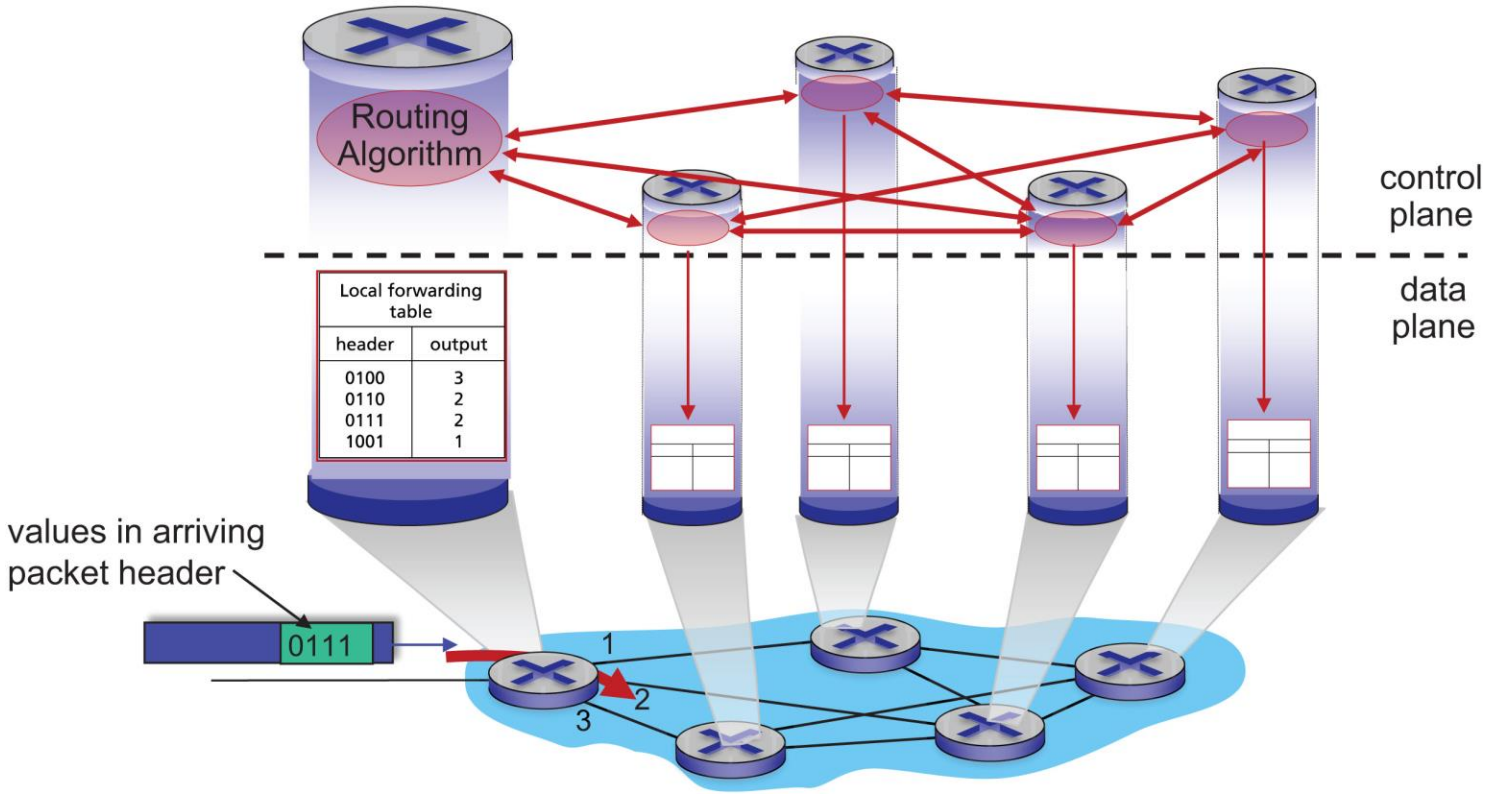


Control plane

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

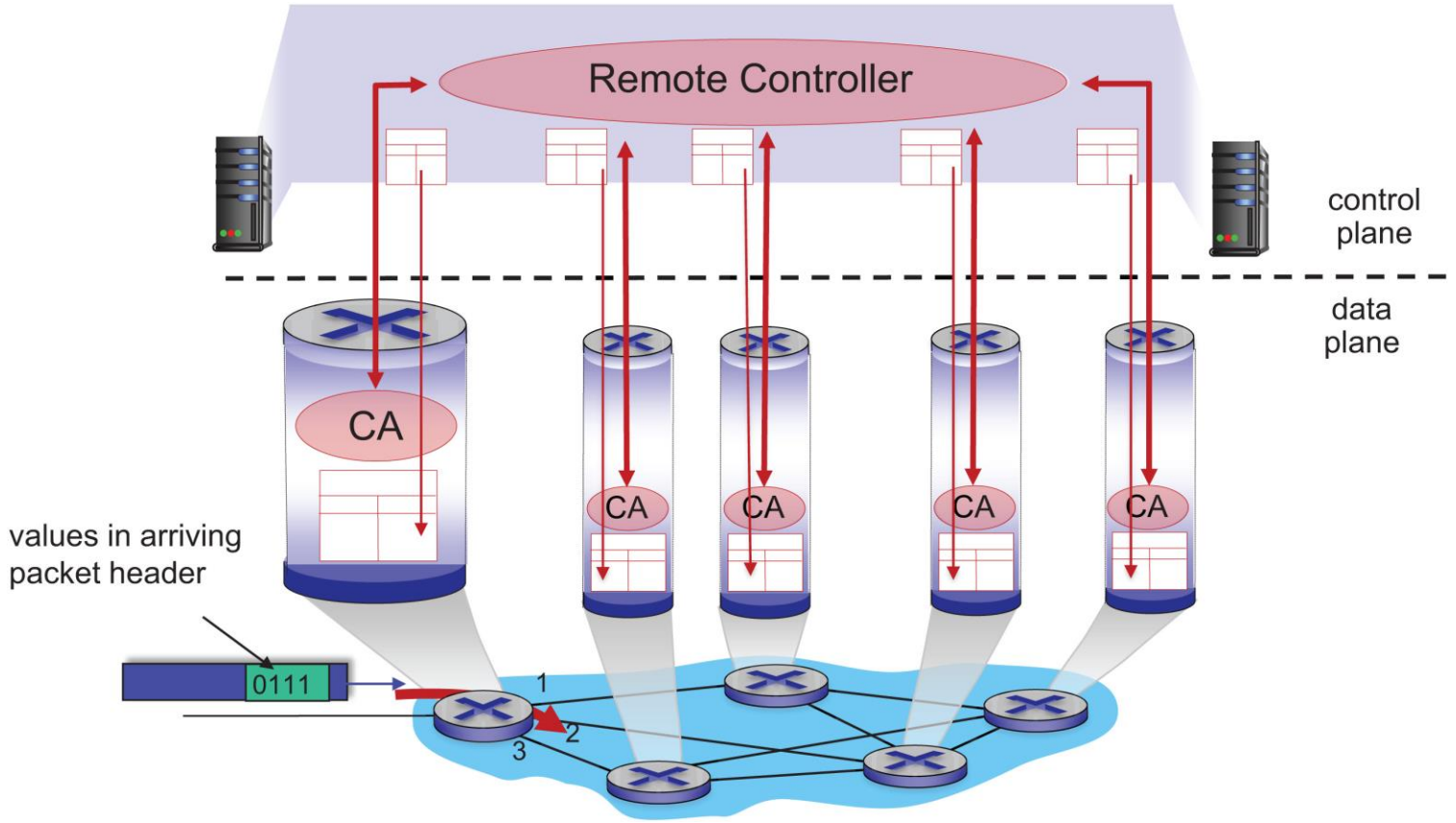
Per-Router Control Plane

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



Software-Defined Networking (SDN) Control Plane

Remote controller computes, installs forwarding tables in routers



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

example services for a **flow** of datagrams:

- in-order datagram delivery
- guaranteed minimum bandwidth to flow
- restrictions on changes in inter-packet spacing

Network-Layer Service Model (1 of 2)

Quality of Service (QoS) Guarantees ?

Network Architecture	Service Model	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

Network-Layer Service Model (2 of 2)

Quality of Service (QoS) Guarantees ?

Network Architecture	Service Model	Bandwidth	Loss	Order	Timing
Internet	best effort	none	no	no	no
ATM	Constant Bit Rate	Constant rate	yes	yes	yes
ATM	Available Bit Rate	Guaranteed min	no	yes	no
Internet	Intserv Guaranteed (RFC 1633)	yes	yes	yes	yes
Internet	Diffserv (RFC 2475)	possible	possibly	possibly	no

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
- congestion control of "elastic" services helps

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

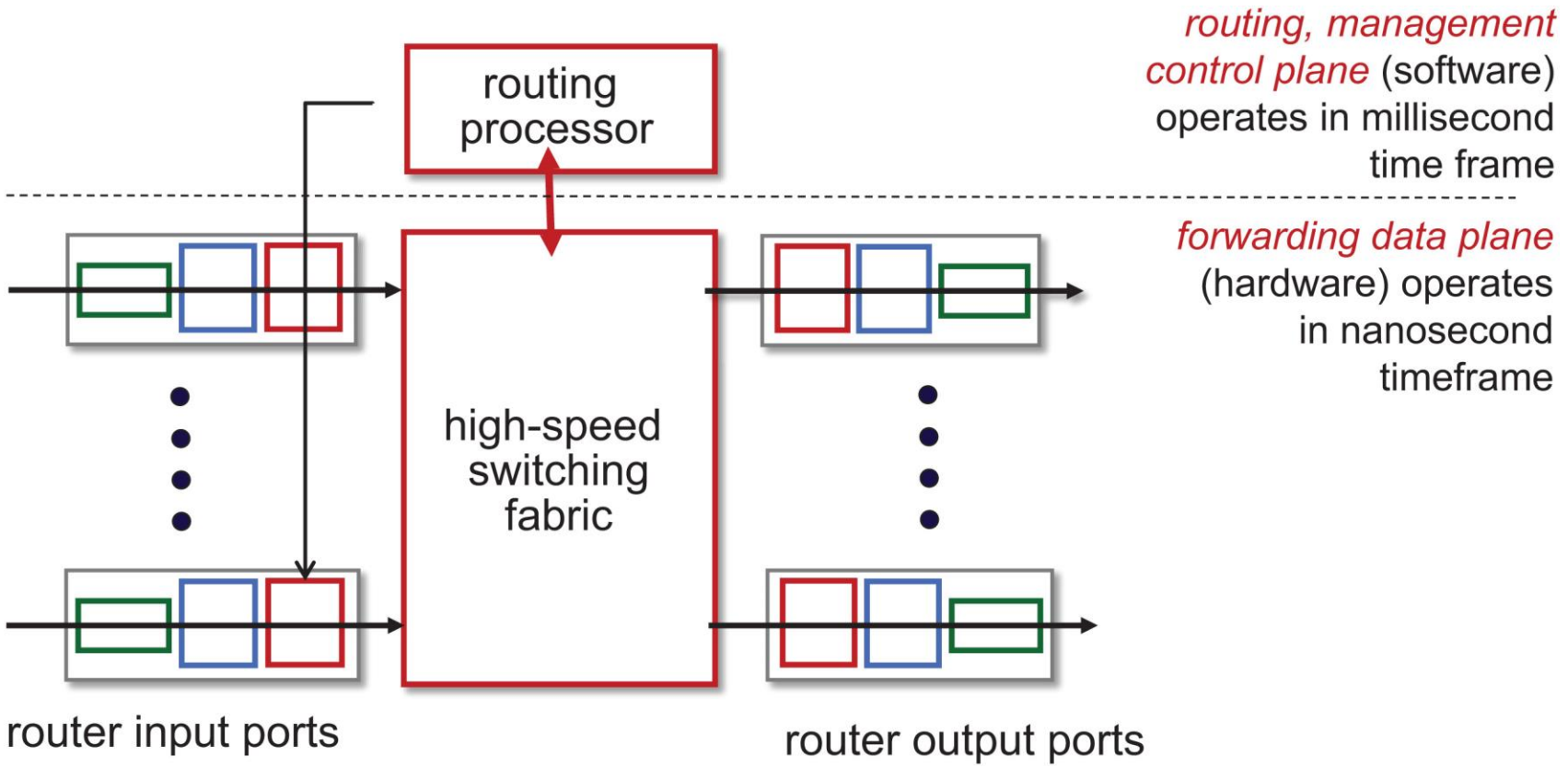
Network Layer: "Data plane" Roadmap (2 of 6)

- Network layer: overview
  - data plane
  - control plane
- What's inside a router
  - input/output ports, switching
  - buffer management
  - scheduling
- IP: the Internet Protocol
  - datagram format, addressing
  - network address translation (NAT)
  - IPv6

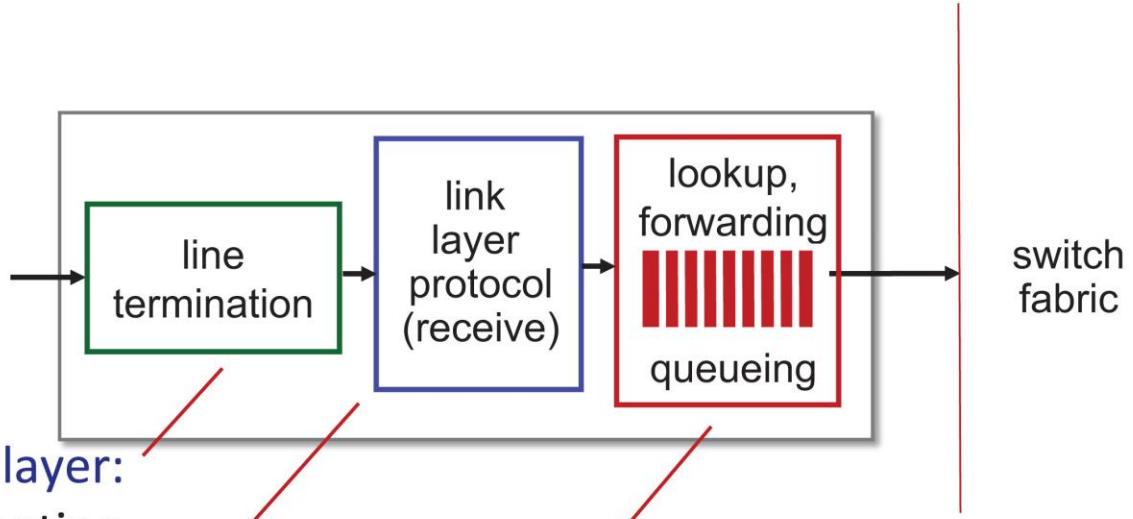
- Generalized Forwarding
  - Match+action
  - OpenFlow
  - Middleboxes
- Internet architecture

Router Architecture Overview

high-level view of generic router architecture:



Input Port Functions (1 of 2)



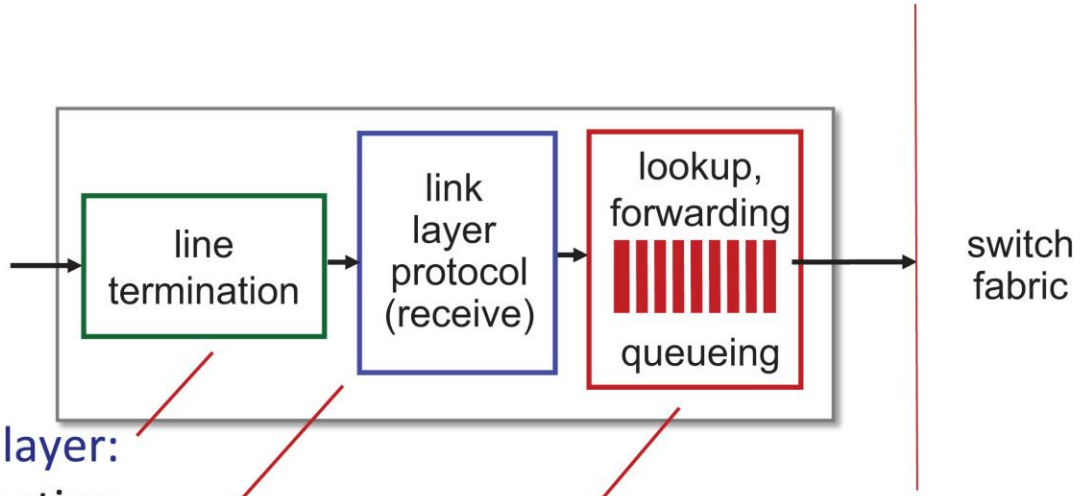
physical layer:
bit-level reception

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")*
- goal: complete input port processing at 'line speed'
- input port queuing: if datagrams arrive faster than forwarding rate into switch fabric

Input Port Functions (2 of 2)



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

Destination-Based Forwarding

forwarding table

Destination Address Range	Link Interface
11001000 00010111 00010000 00000000 through 11001000 00010111 00010000 00000100	0
through 11001000 00010111 00010000 00000111	3
11001000 00010111 00011000 11111111	
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 (1 of 5)

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 (2 of 5)

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 (3 of 5)

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 (4 of 5)

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?

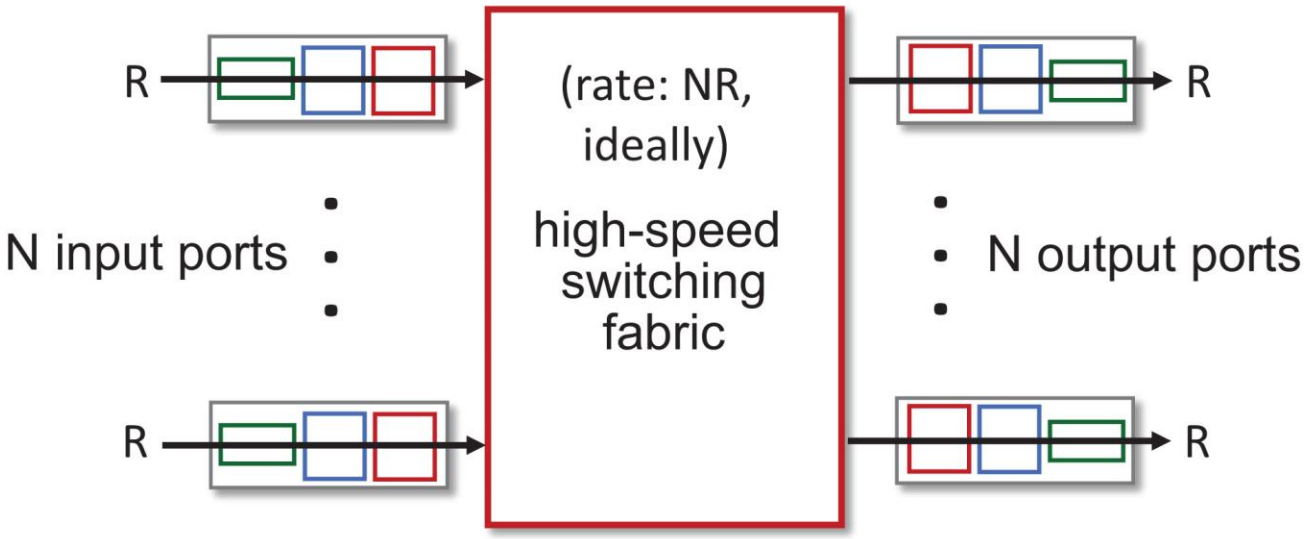
11001000 00010111 00011000 10101010 which interface?

Longest Prefix Matching (5 of 5)

- we'll see **why** longest prefix matching is used shortly, when we study addressing

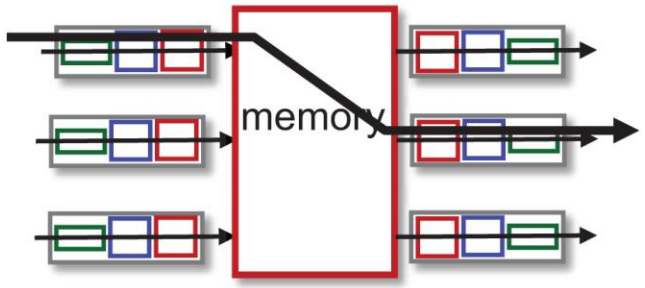
Switching Fabrics (1 of 2)

- 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

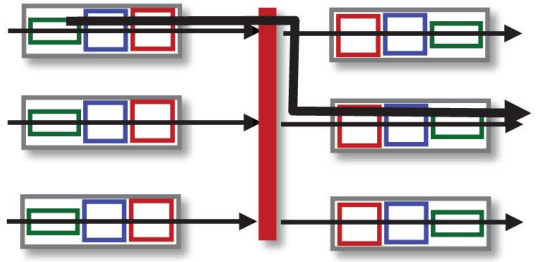


Switching Fabrics (2 of 2)

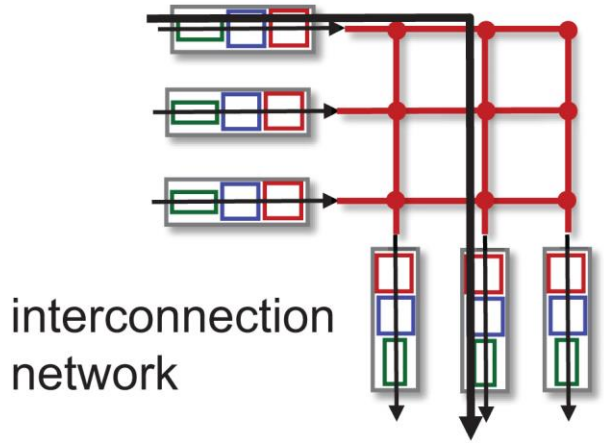
- transfer packet from input link to appropriate output link
- switching rate: rate at which packets can be transfer from inputs to outputs
  - often measured as multiple of input/output line rate
  - N inputs: switching rate N times line rate desirable
- three major types of switching fabrics:



memory



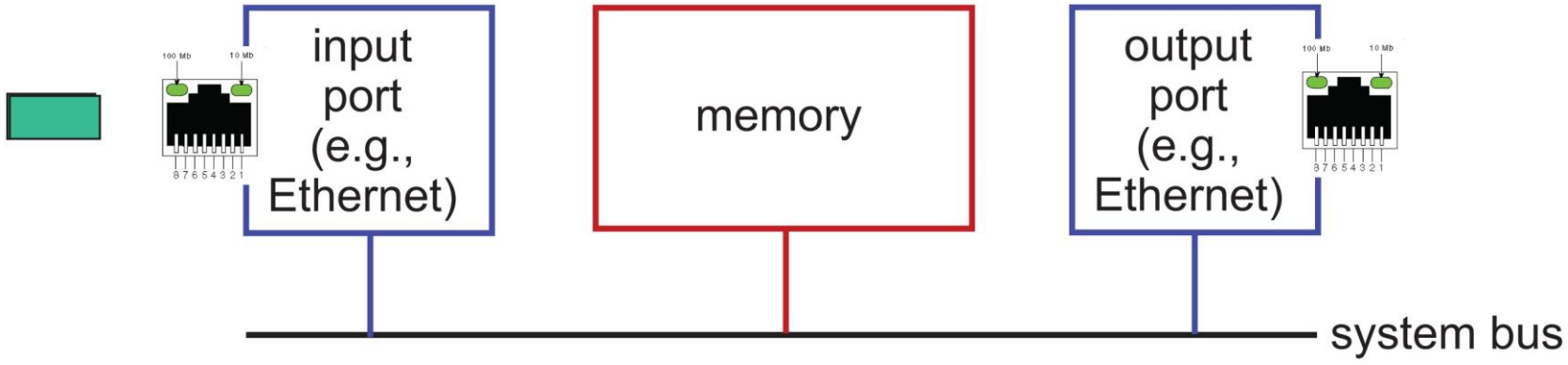
bus



Switching Via Memory

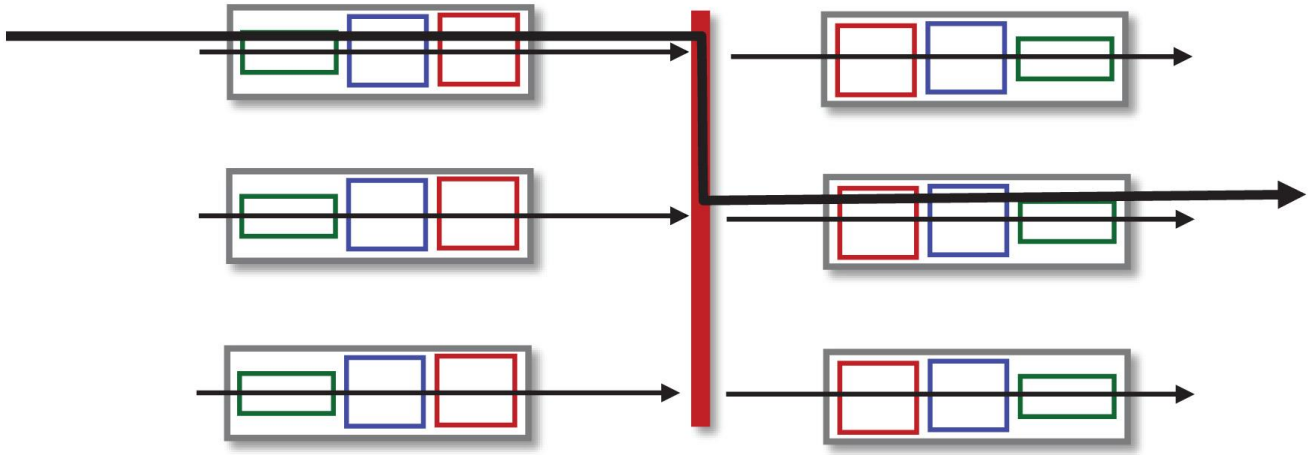
first generation routers:

- 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)



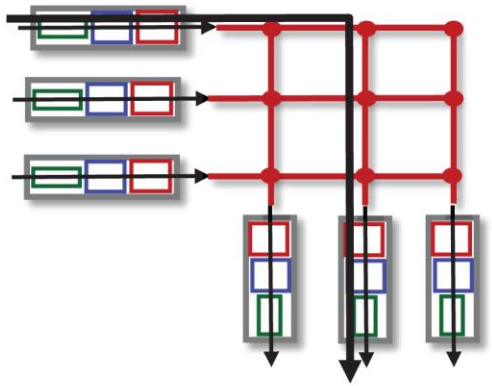
Switching Via a Bus

- datagram from input port memory to output port memory via a shared bus
- **bus contention:** switching speed limited by bus bandwidth
- 32 Gbps bus, Cisco 5600: sufficient speed for access routers

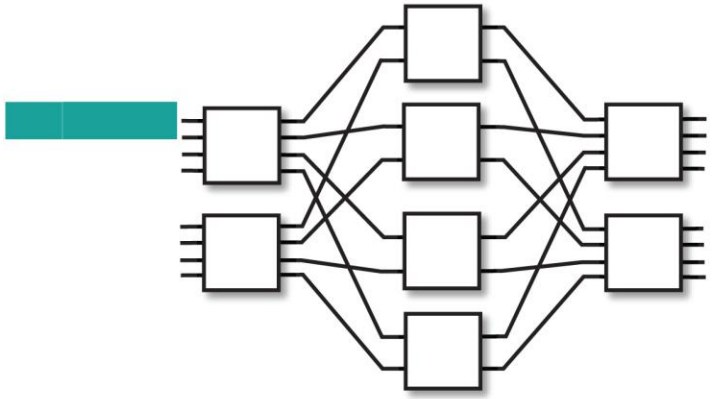


Switching Via Interconnection Network (1 of 2)

- Crossbar, Clos networks, other interconnection nets initially developed to connect processors in multiprocessor
- multistage switch: **nxn** 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



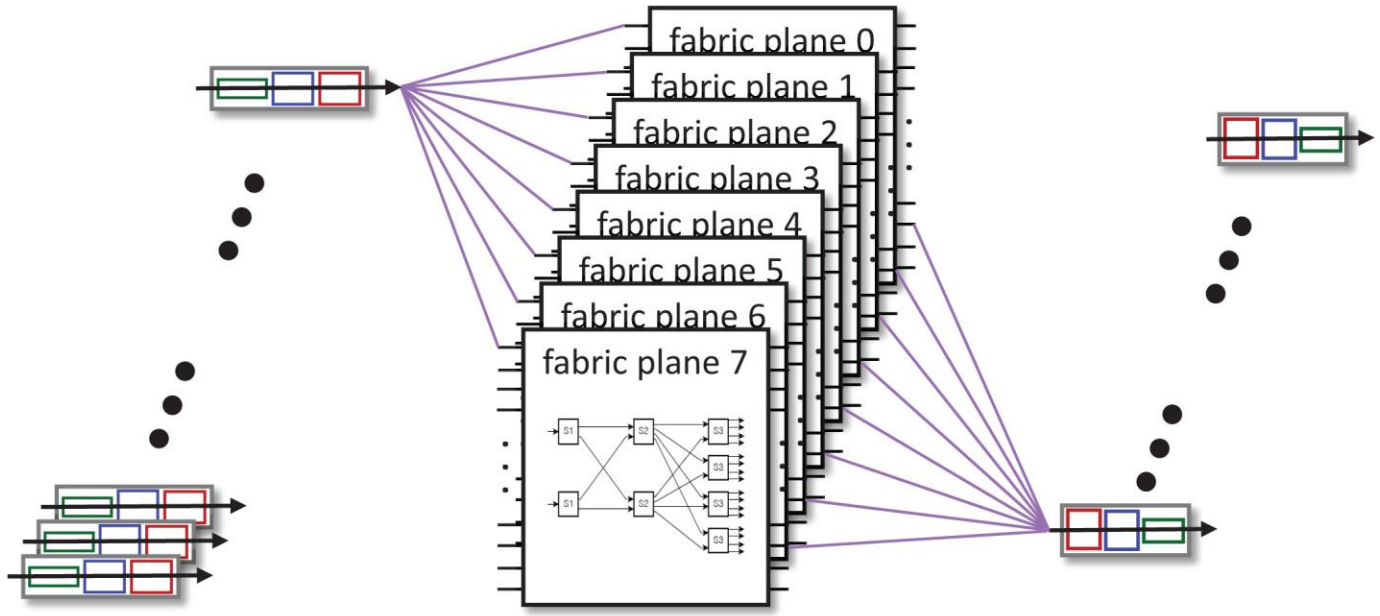
3x3 crossbar



8x8 multistage switch
built from smaller-sized switches

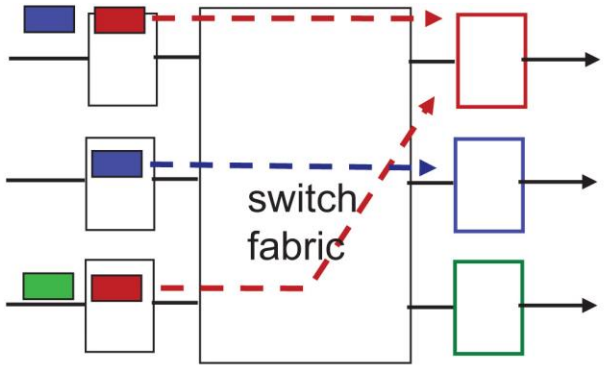
Switching Via Interconnection Network (2 of 2)

- 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

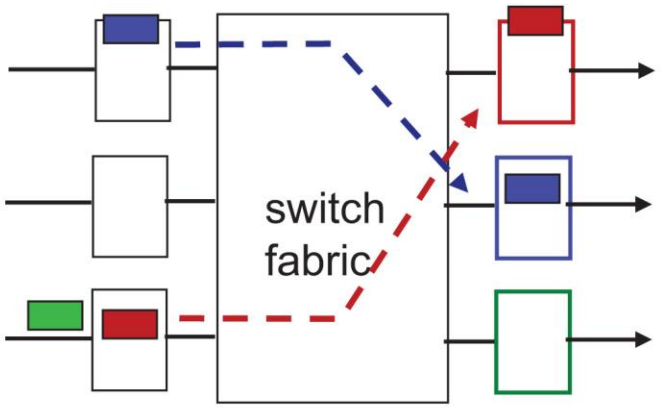


Input Port Queuing

- If switch fabric slower than input ports combined → queueing may occur at inputs 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

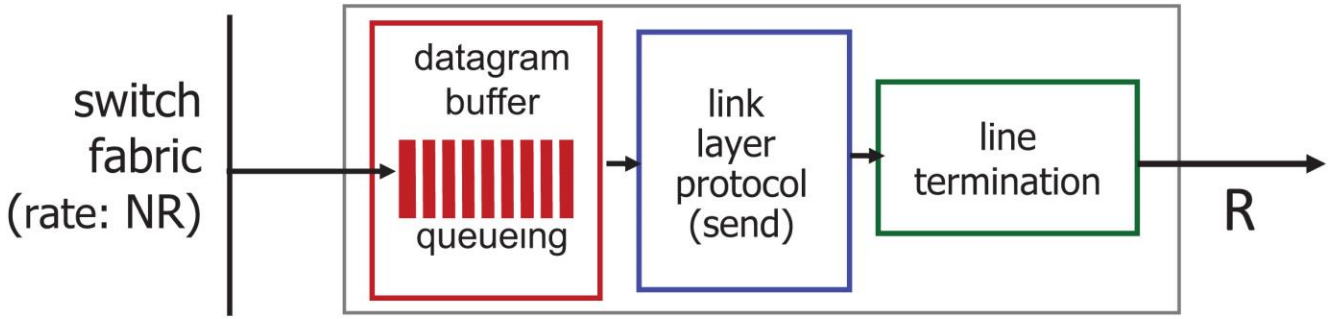


output port contention: only one red datagram can be transferred. lower red packet is *blocked*



one packet time later: green packet experiences HOL blocking

Output Port Queuing (1 of 2)



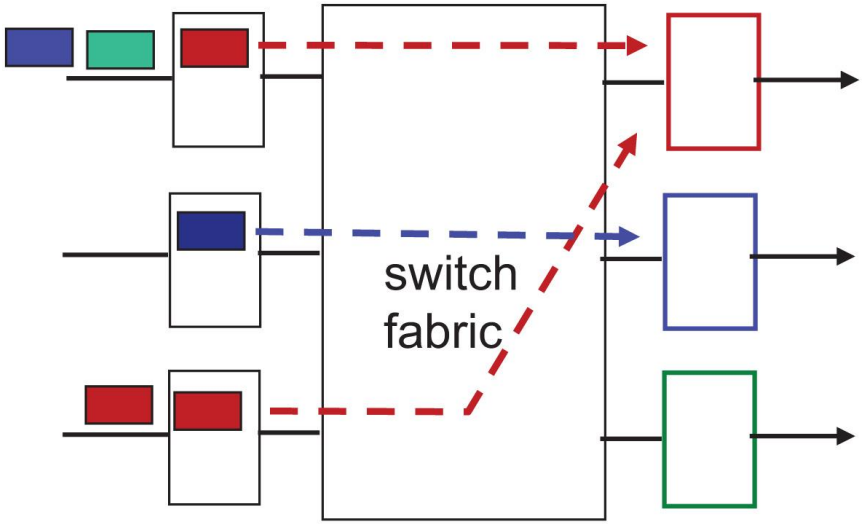
- **Buffering** required when datagrams arrive from fabric faster than link transmission rate. **Drop policy:** which datagrams to drop if no free buffers?

→ Datagrams can be lost due to congestion, lack of buffers

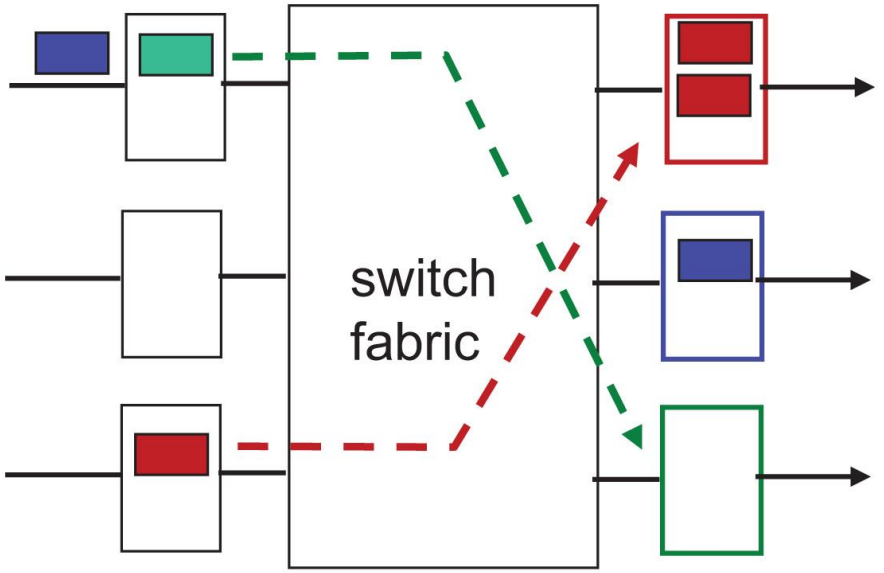
- **Scheduling discipline** chooses among queued datagrams for transmission

→ Priority scheduling – who gets best performance, network neutrality

Output Port Queuing (2 of 2)



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!

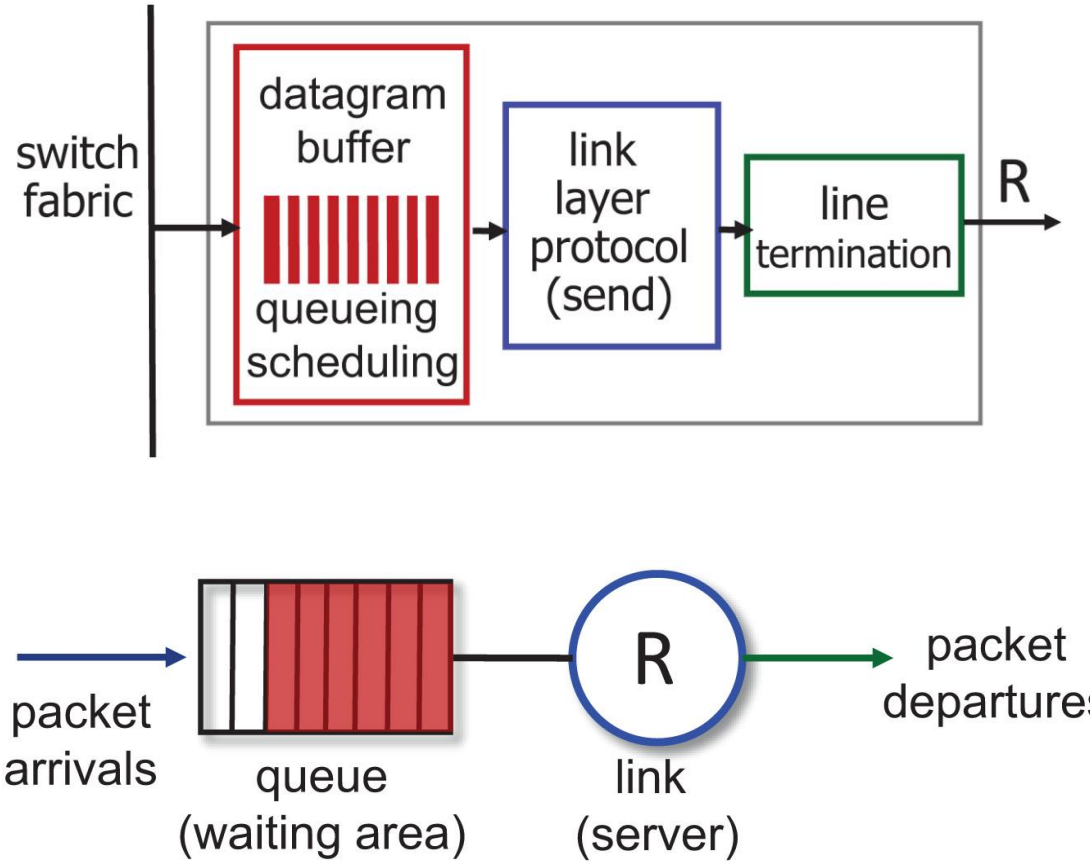
How Much Buffering?

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

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

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

Buffer Management



buffer management:

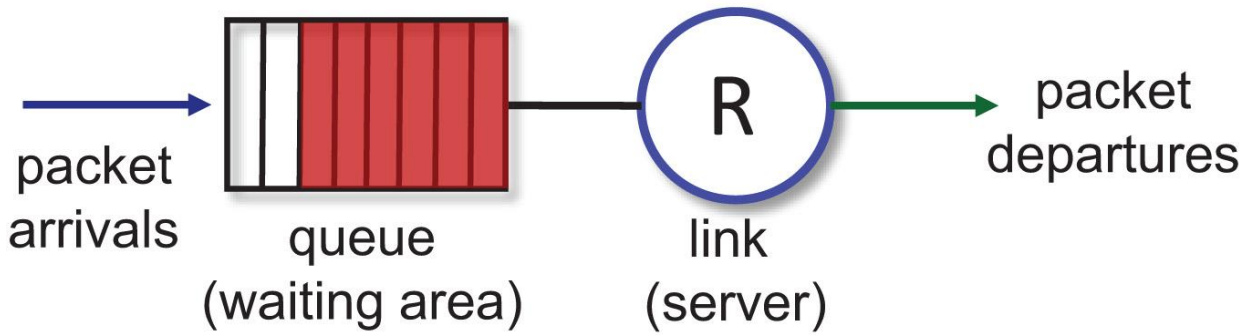
- drop: which packet to add, drop when buffers are full
  - tail drop: drop arriving packet
  - priority: drop/remove on priority basis
- marking: which packets to mark to signal congestion (ECN, RED)

Packet Scheduling: FCFS

- packet scheduling: deciding which packet to send next on link
  - first come, first served
  - priority
  - round robin
  - weighted fair queueing

- FCFS: packets transmitted in order of arrival to output port
  - also known as: First-in-first-out (FIFO)
  - real world examples?

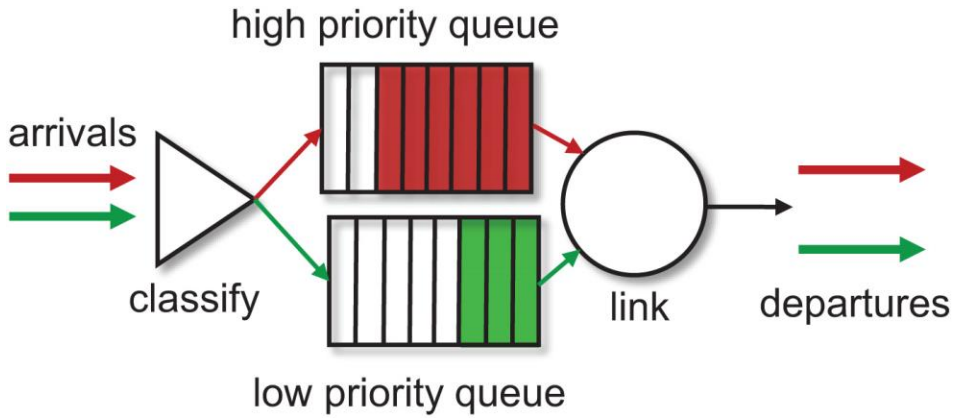
Abstraction: queue

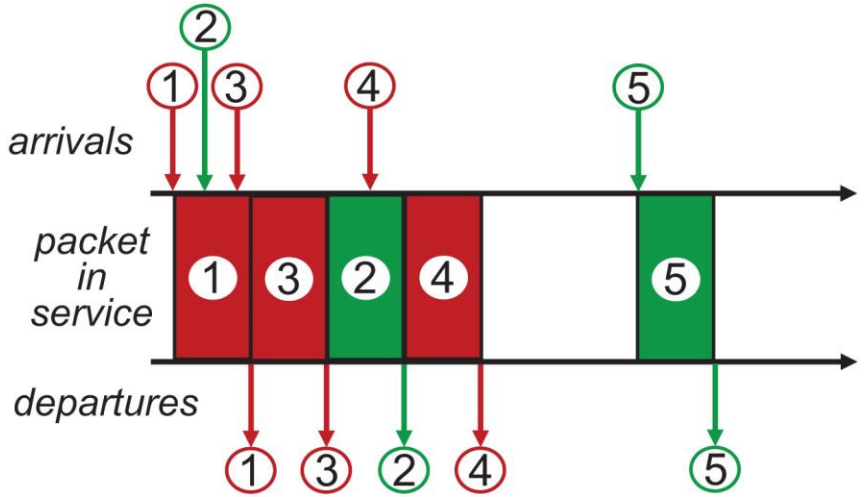


Scheduling Policies: Priority

Priority scheduling:

- arriving traffic classified, queued by class
  - any header fields can be used for classification
- send packet from highest priority queue that has buffered packets
  - FCFS within priority class

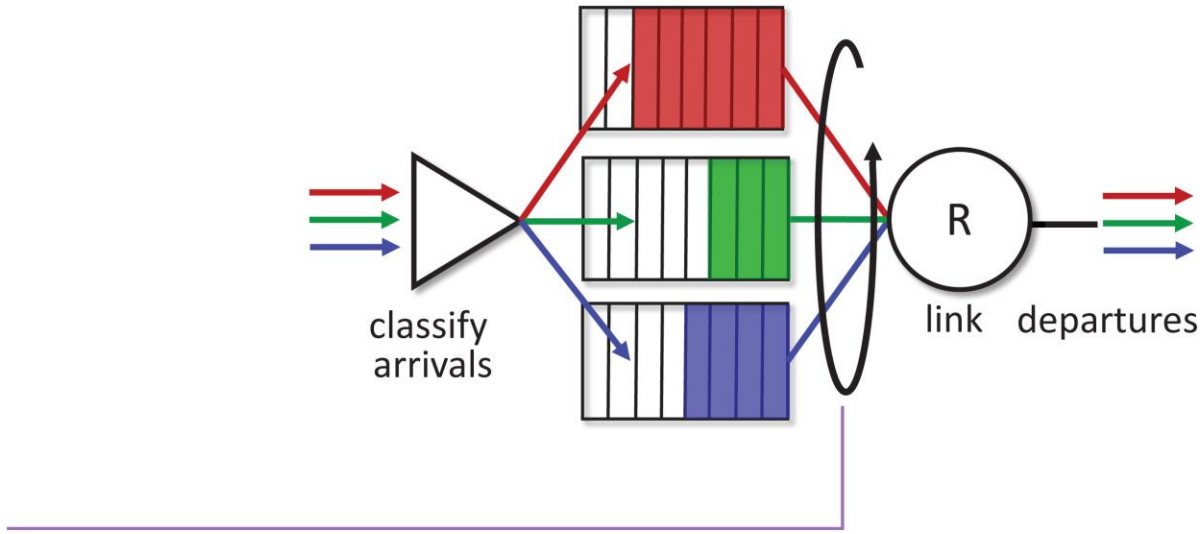




Scheduling Policies: Round Robin

Round Robin (RR) scheduling:

- arriving traffic classified, queued by class
  - any header fields can be used for classification
- server cyclically, repeatedly scans class queues, sending one complete packet from each class (if available) in turn



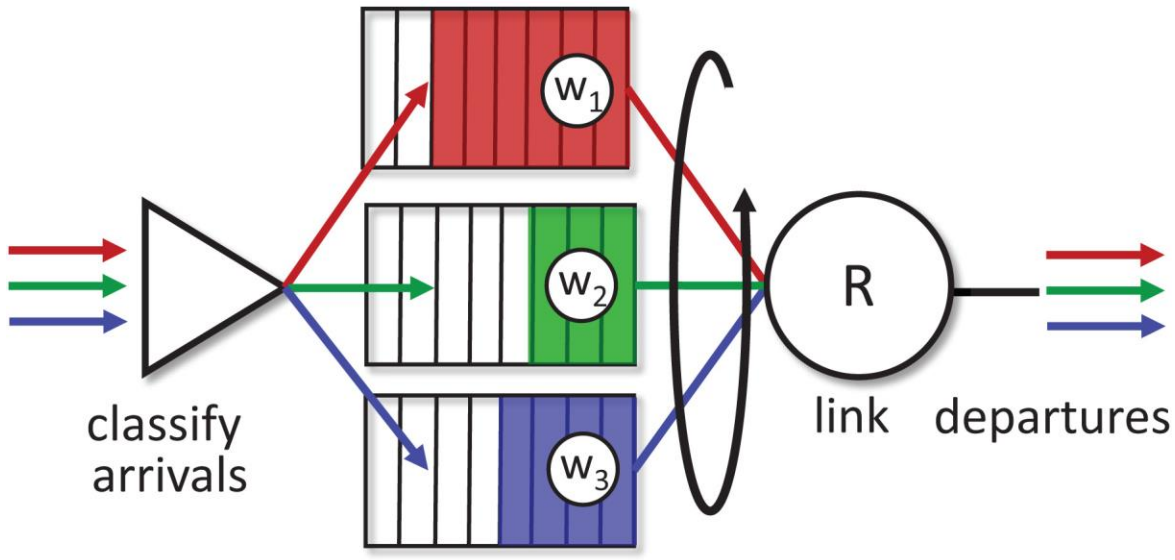
Scheduling Policies: Weighted Fair Queueing

Weighted Fair Queuing (WFQ):

- generalized Round Robin
- each class, *i*, has weight, w_i, and gets weighted amount of service in each cycle:

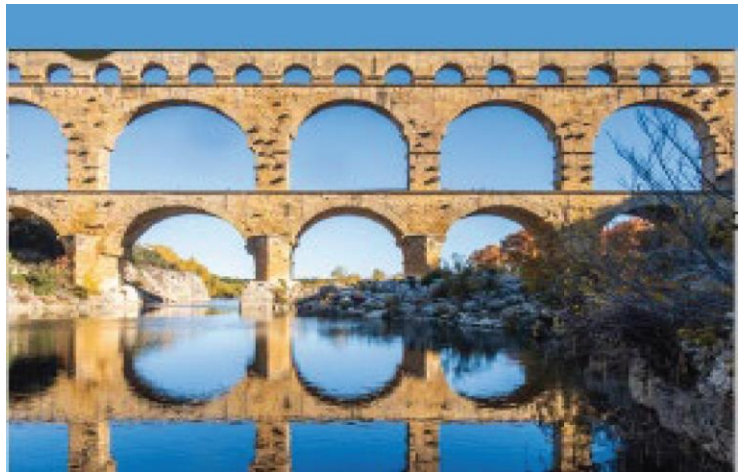
$$\frac{w_i}{\sum_j w_j}$$

- minimum bandwidth guarantee (per-traffic-class)



Network Layer: "Data plane" Roadmap (3 of 6)

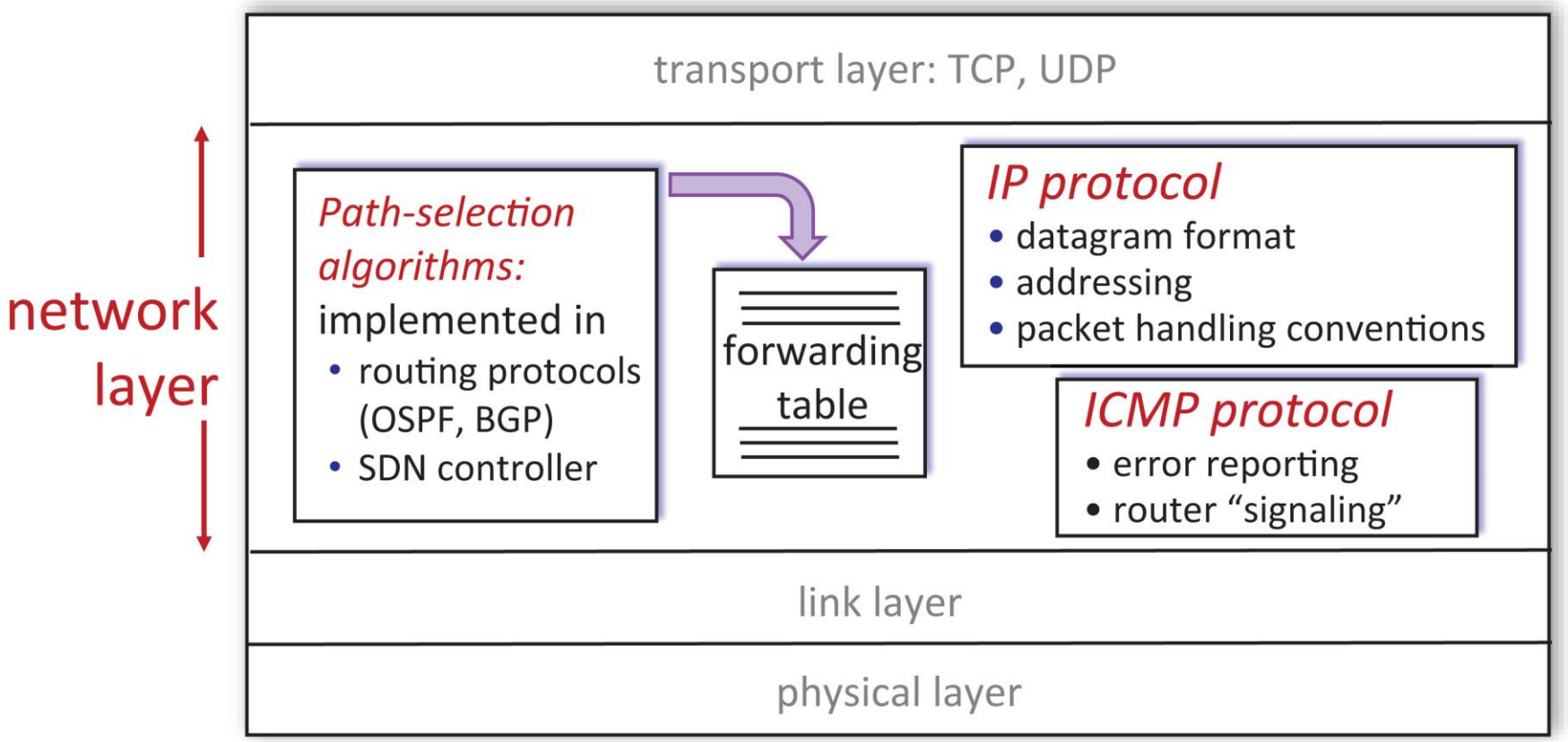
- Network layer: overview
  - data plane
  - control plane
- What's inside a router
  - input/output ports, switching
  - buffer management
  - scheduling
- IP: the Internet Protocol
  - datagram format, addressing
  - network address translation (NAT)
  - IPv6



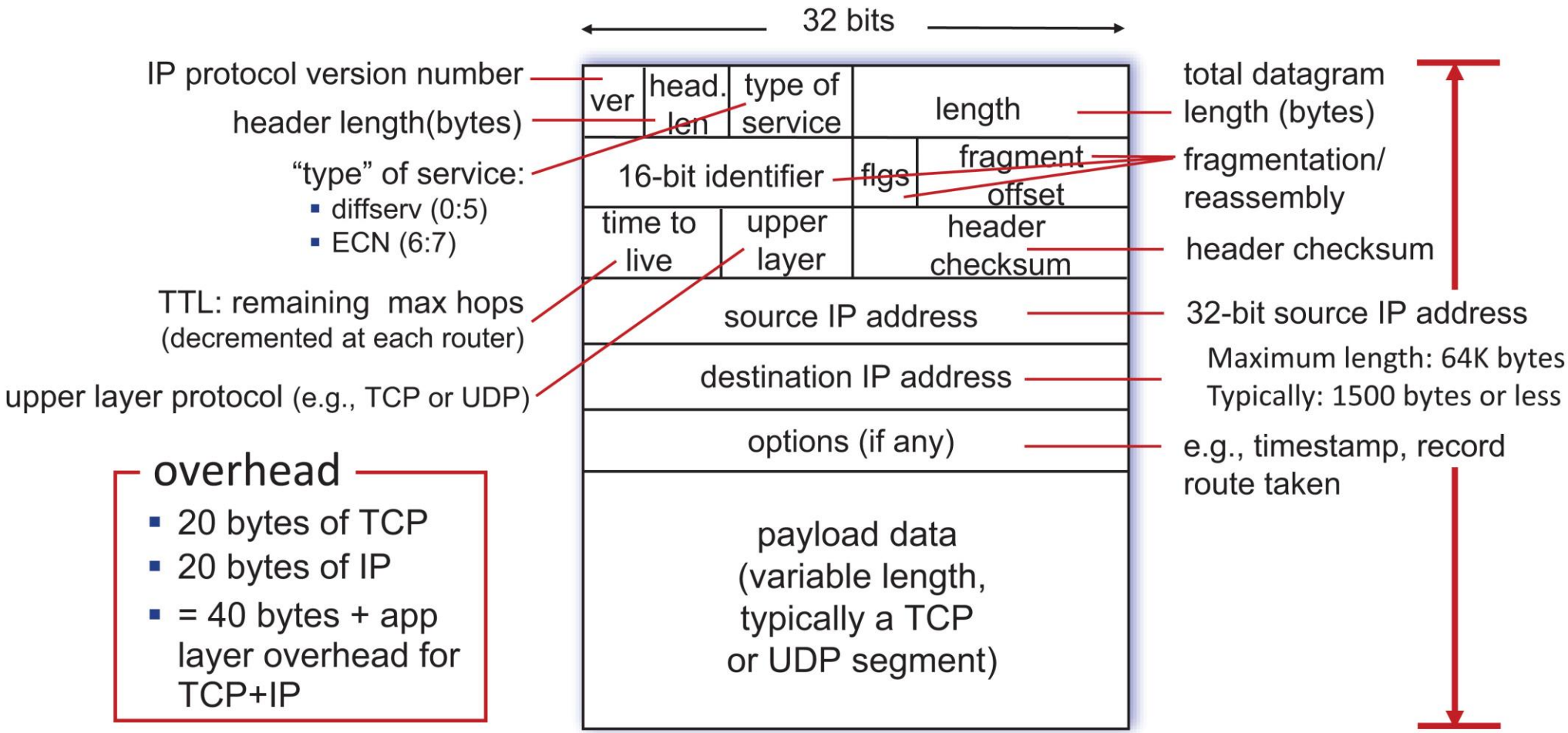
- Generalized Forwarding
  - Match+action
  - OpenFlow
  - Middleboxes
- Internet architecture

Network Layer: Internet

host, router network layer functions:



IP Datagram Format



overhead

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

Header fields

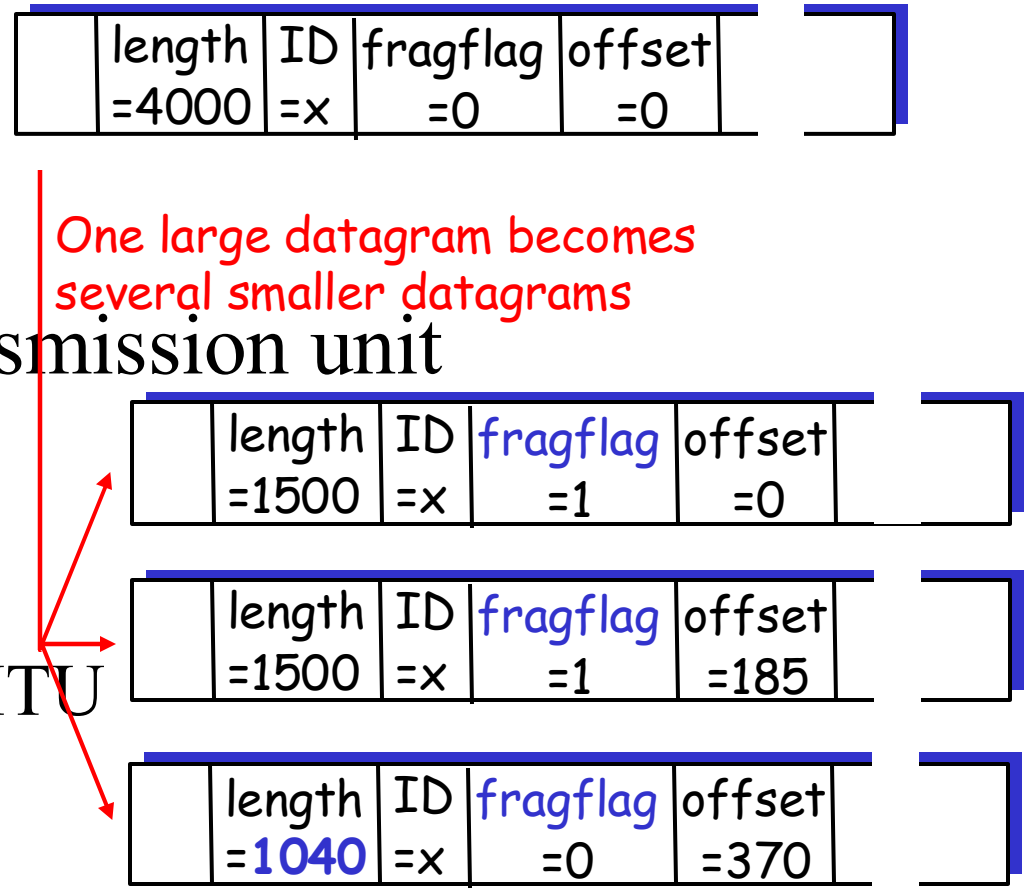
- IP header checksum (16-bit)
  - TCP/IP-style checksum
  - cover IP header (and option) only
- Protocol ID (8-bit)
  - TCP(6), UDP(17); /etc/protocols
- TTL: time-to-live (8-bit)
  - decrement by each router
  - drop if TTL=0

Header field: more

- Total length (16-bit)
  - byte counter
- IHL: IP header length (4-bit)
  - 4-byte counter
- Identification (16-bit)
- Fragment offset (13-bit)
  - 8-byte offset
  - DF: don’t fragment; MF: more fragment(s)

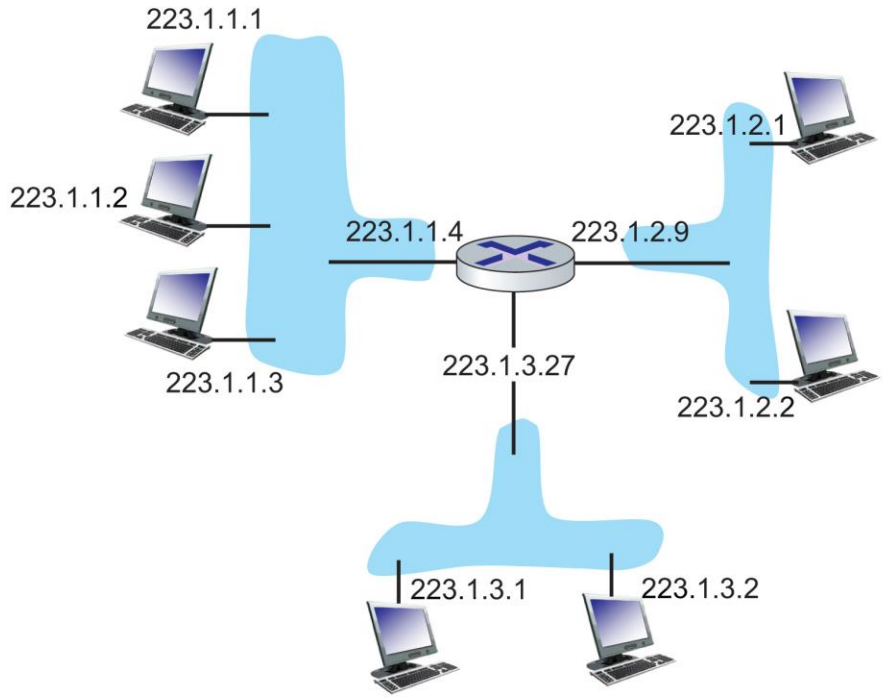
Fragment and reassemble

- IP packet length
  - 2^{16}-1 bytes
- MTU: maximum transmission unit
  - Ethernet: 1500 bytes
- Fragment
  - when total length > MTU
- Reassemble
  - only at destination
- PMTU discovery



IP Addressing: Introduction (1 of 2)

- IP address: 32-bit identifier associated with each host or 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)



dotted-decimal IP address notation:

223.1.1.1 = 11011111 00000001 00000001 00000001

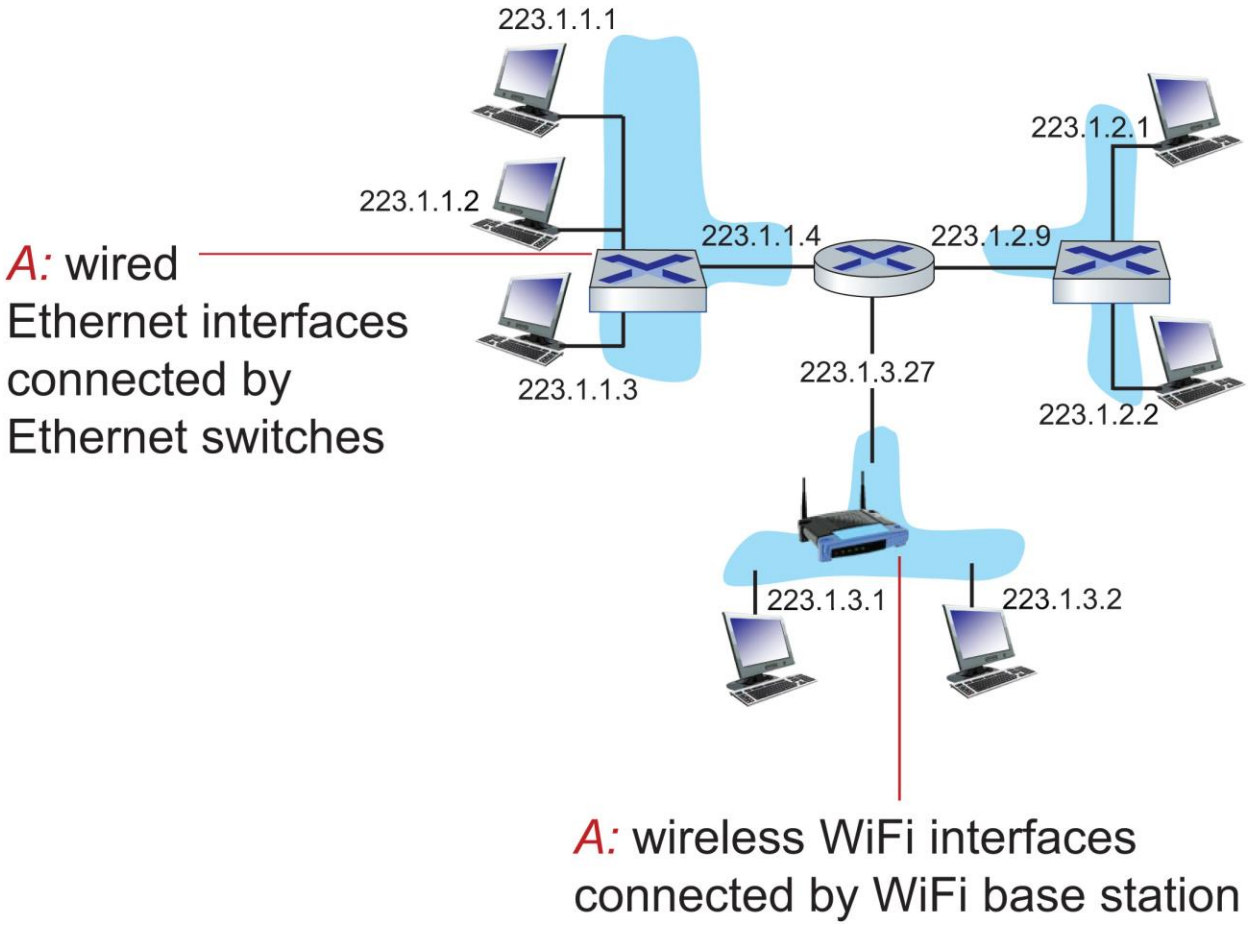
223 1 1 1

IP Addressing: Introduction (2 of 2)

Q: how are interfaces actually connected?

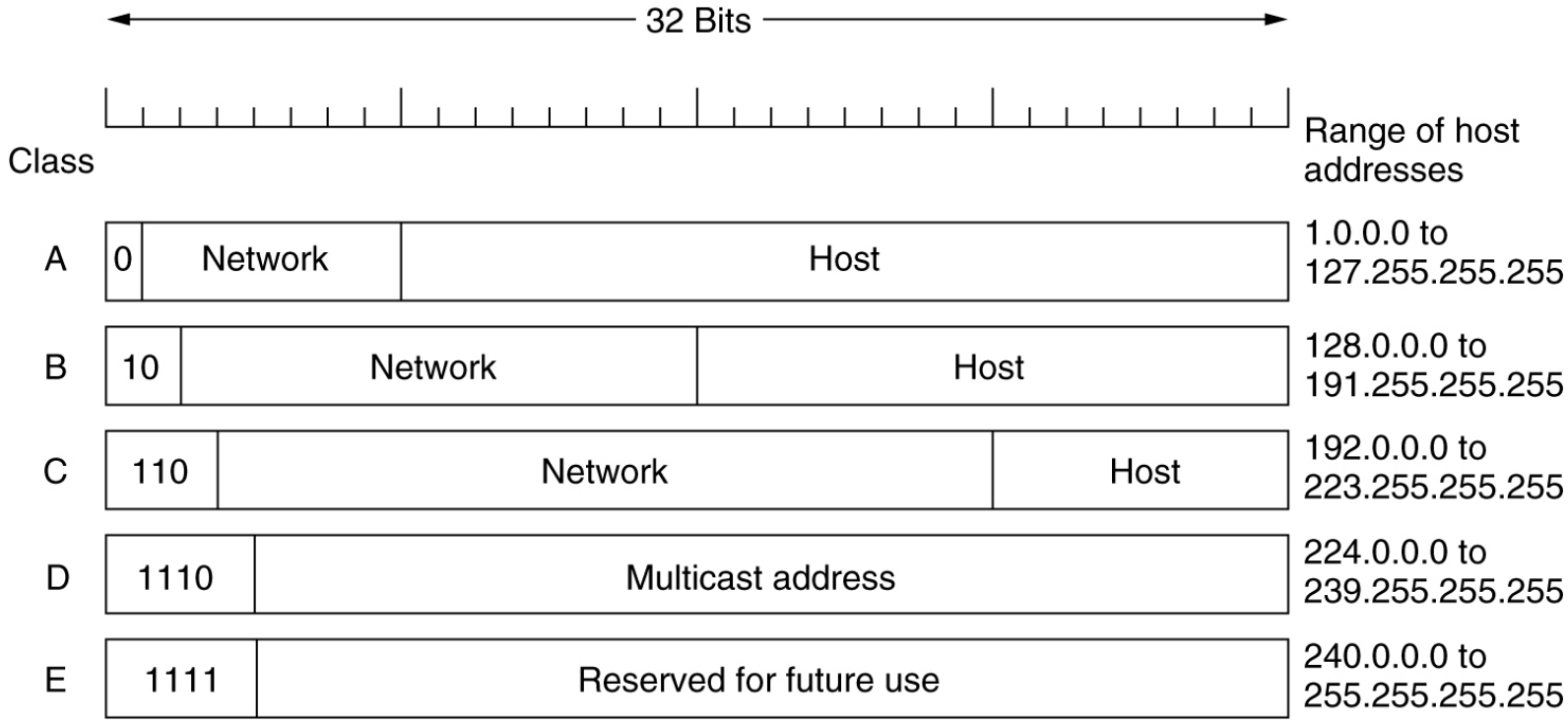
A: we'll learn about that in chapters 6, 7

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



IP address

- Address classes

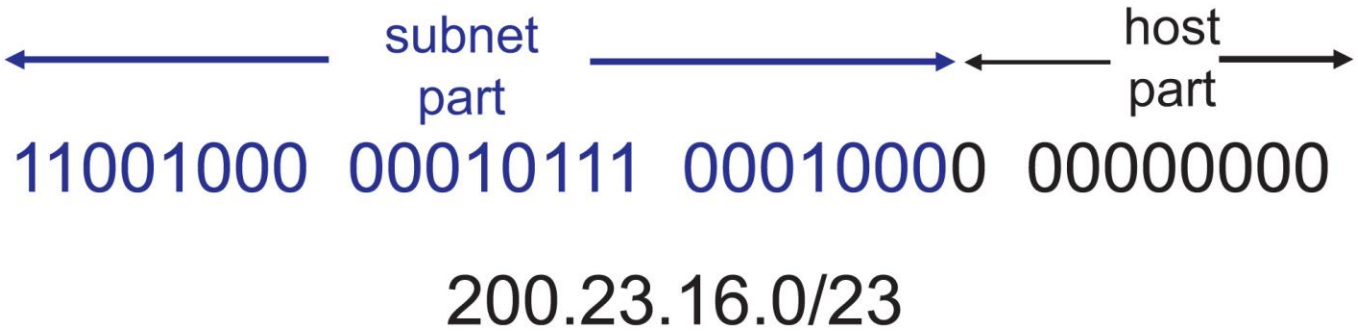


IP address: more

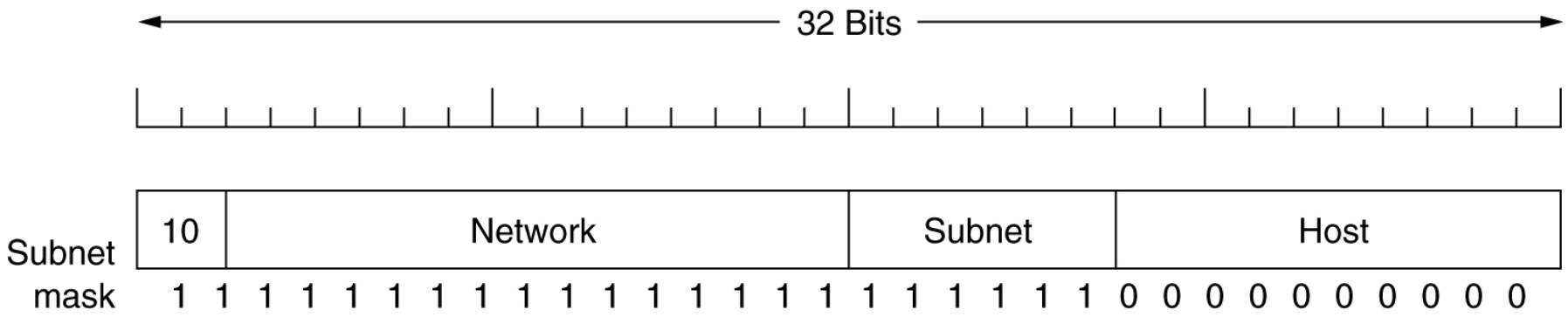
- Problem with “address classes”
  - too big a Class A network
  - too (many) small Class C networks

IP Addressing: CIDR

- CIDR: Classless InterDomain Routing (pronounced "cider")
  - subnet portion of address of arbitrary length
  - address format: a.b.c.d/x, where x is # bits in subnet portion of address

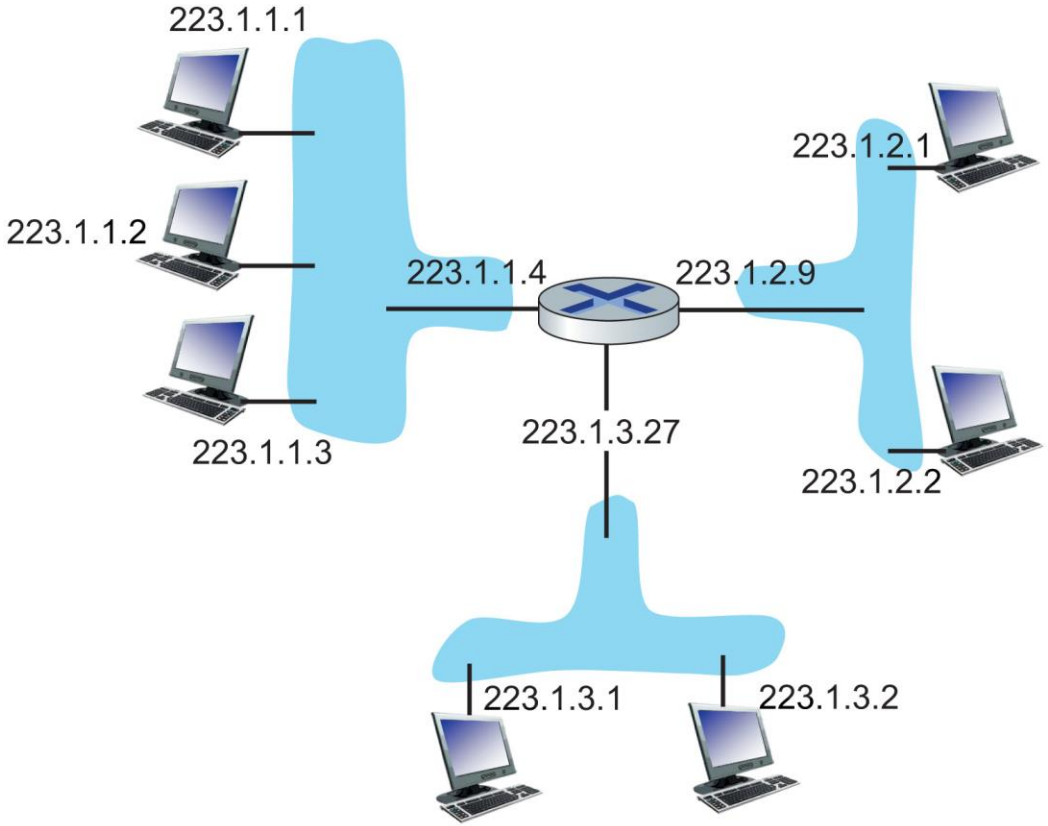


or



Subnets (1 of 3)

- 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

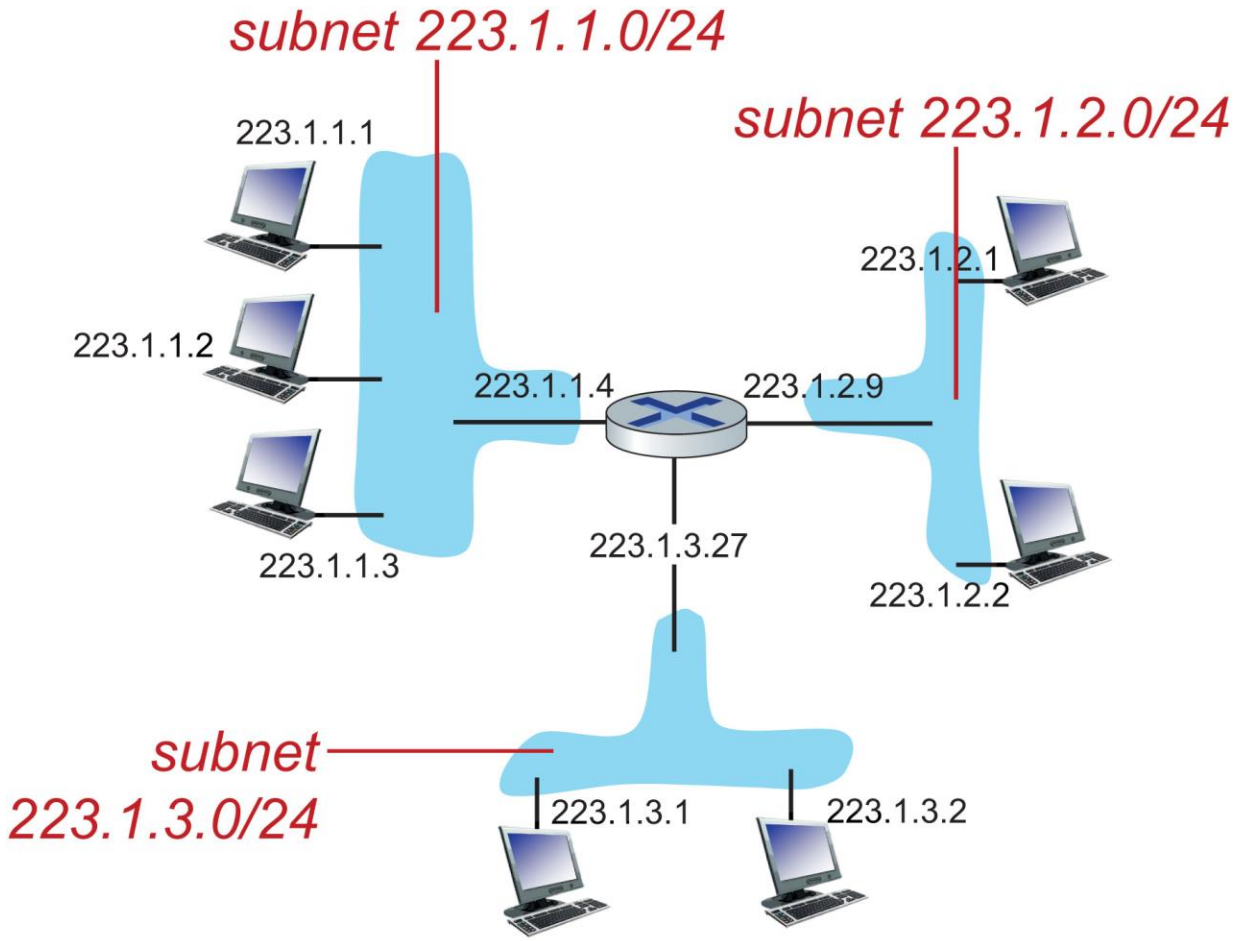


network consisting of 3 subnets

Subnets (2 of 3)

Recipe for defining subnets:

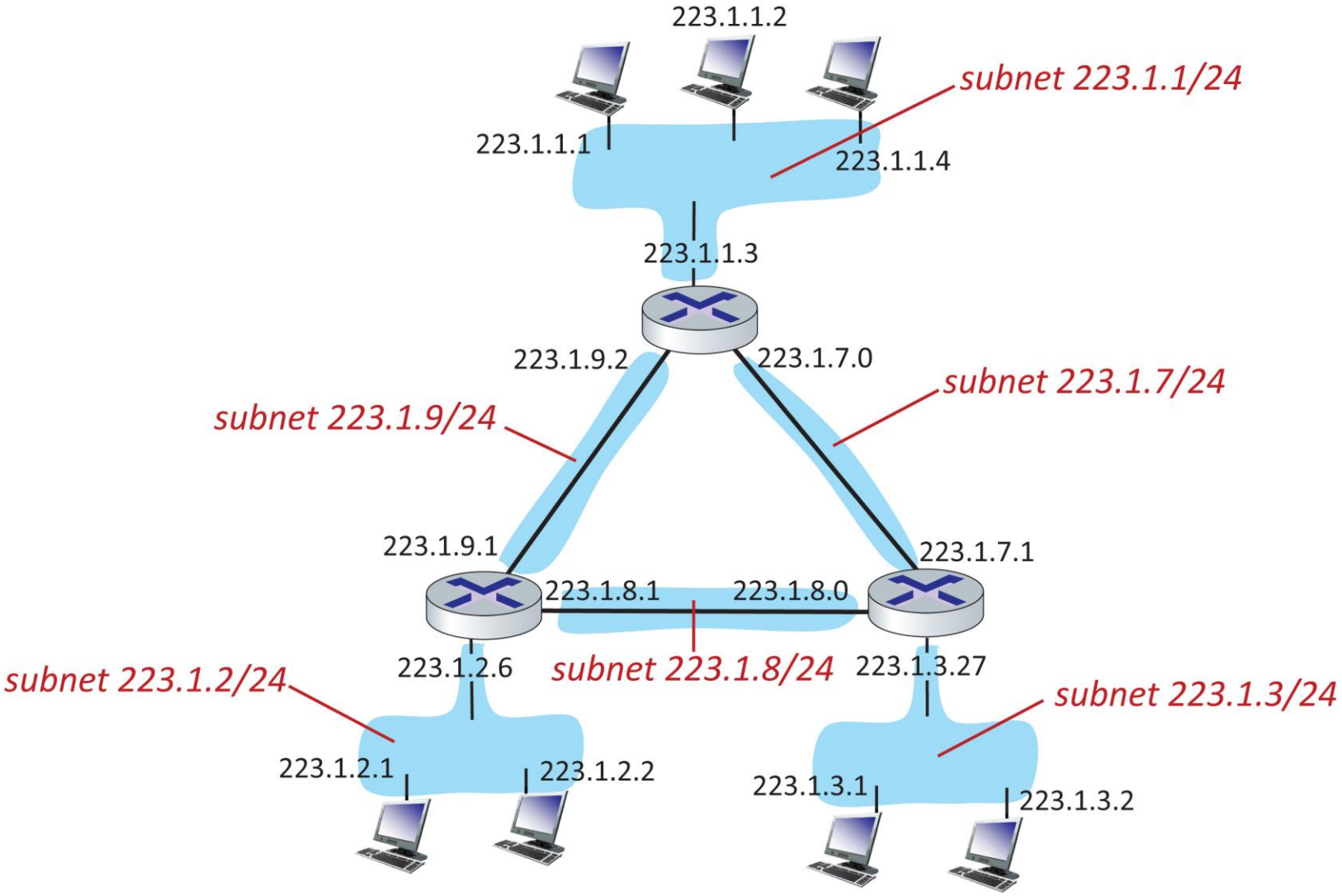
- detach each interface from its host or router, creating "islands" of isolated networks
- each isolated network is called a **subnet**



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

Subnets (3 of 3)

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



UVic's IP space

- UVicNet
  - Class B: 142.104.0.0/16
- UVic EngNet
  - network address: 142.104.96.0
  - network mask: 255.255.224.0
    - 142.104.96.0/19
  - subnet test
    - net_add & net_mask ?= host_add & net_mask
    - host_A_add & net_mask ?= host_B_add & net_mask

IP Addresses: How to Get One? (1 of 2)

That's actually two questions:

1. Q: How does a **host** get IP address within its network (host part of address)?
2. Q: How does a **network** get IP address for itself (network part of address)

How does **host** get IP address?

- hard-coded by sysadmin in config file (e.g., /etc/rc.config in UNIX)
- DHCP: Dynamic Host Configuration Protocol: dynamically get address from as server
  - "plug-and-play"

DHCP: Dynamic Host Configuration Protocol

goal: host **dynamically obtains** IP address from network server when it "joins" network

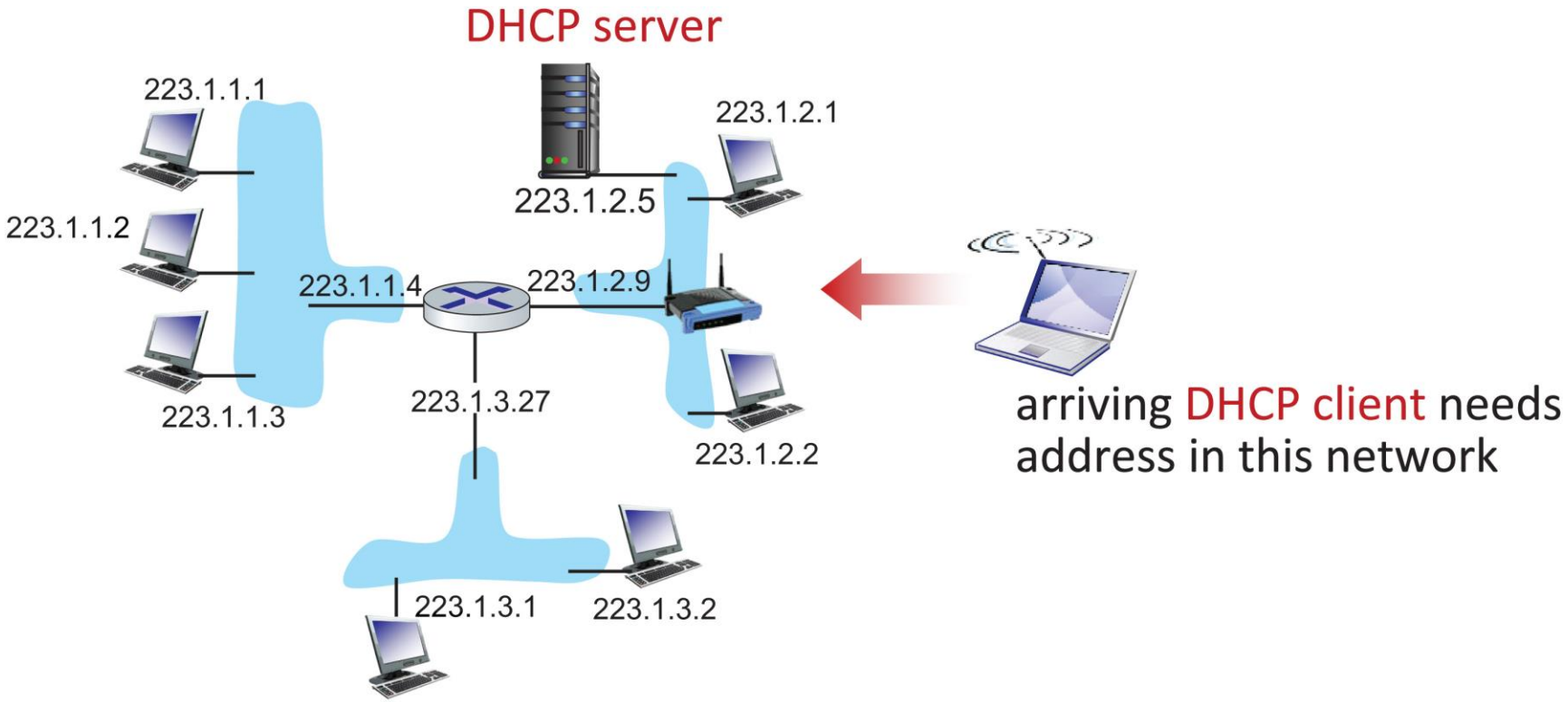
- can renew its lease on address in use
- allows reuse of addresses (only hold address while connected/on)
- support for mobile users who join/leave network

DHCP overview:

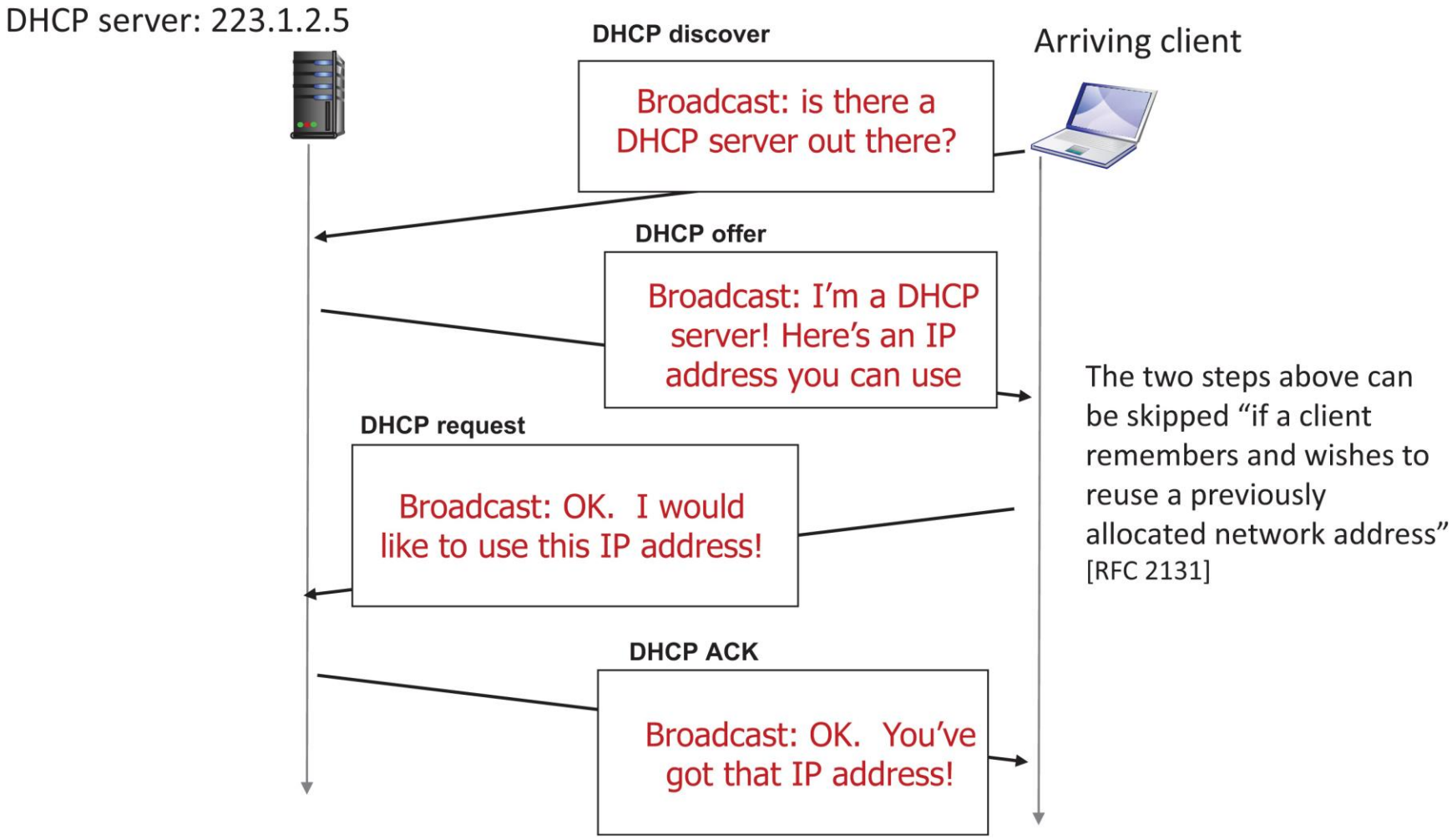
- host broadcasts DHCP discover msg [optional]
- DHCP server responds with DHCP offer msg [optional]
- host requests IP address: DHCP request msg
- DHCP server sends address: DHCP ack msg

DHCP Client-Server Scenario (1 of 2)

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



DHCP Client-Server Scenario (2 of 2)

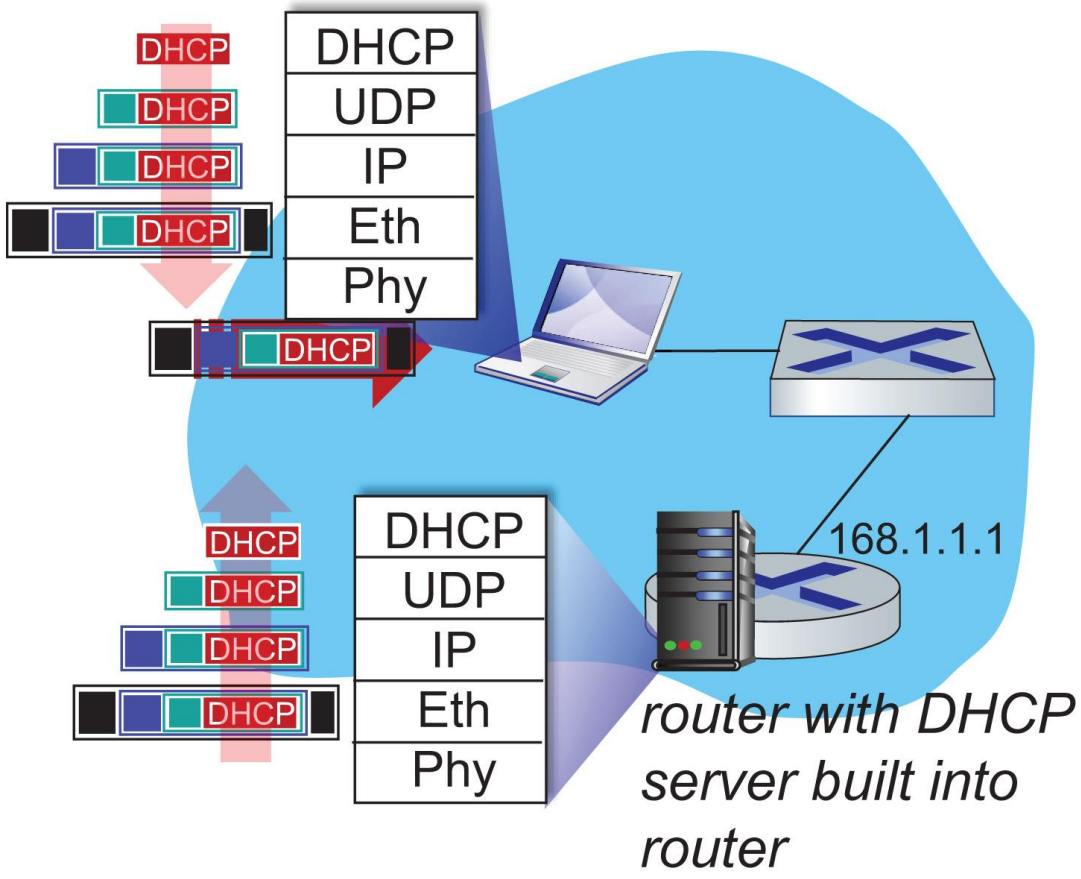


The two steps above can be skipped "if a client remembers and wishes to reuse a previously allocated network address" [RFC 2131]

DHCP: More than IP Addresses

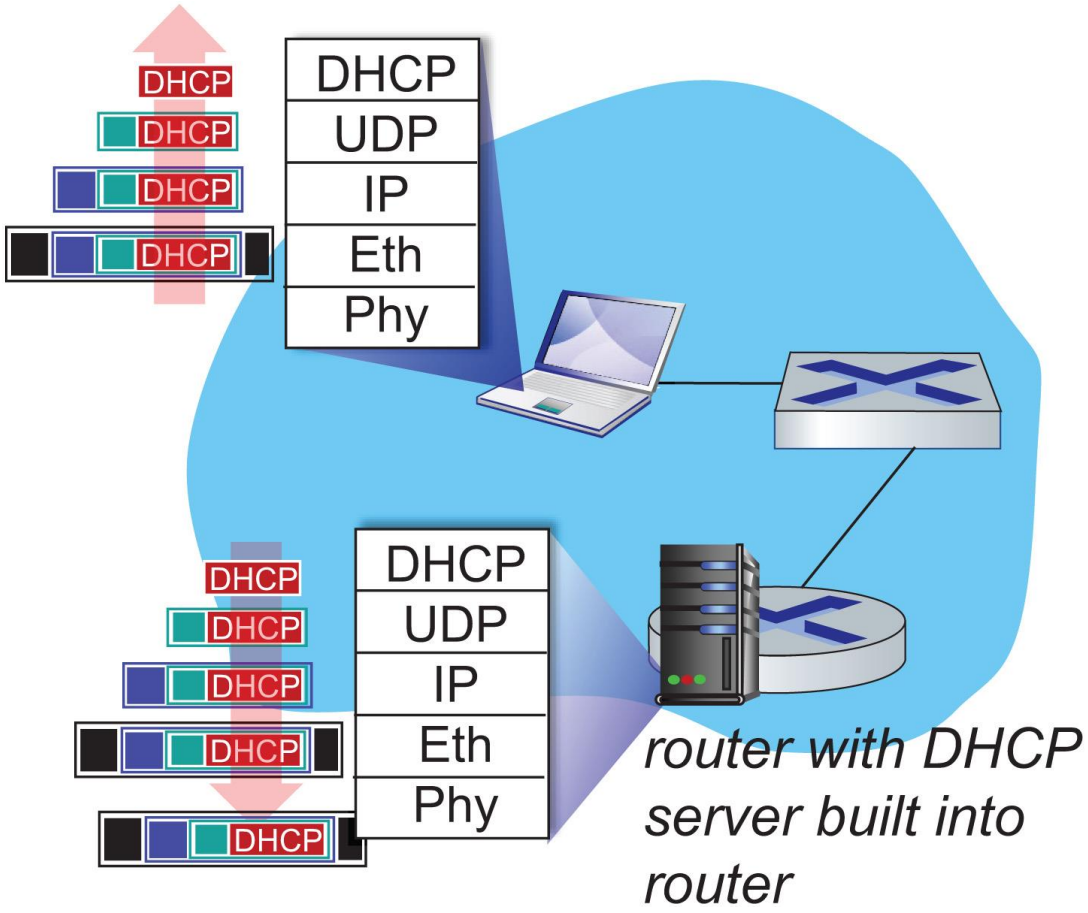
- DHCP can return more than just allocated IP address on subnet:
  - address of first-hop router for client
  - name and IP address of DNS sever
  - network mask (indicating network versus host portion of address)

DHCP: Example (1 of 2)



- Connecting laptop will use DHCP to get IP address, address of first-hop router, address of DNS server.
- DHCP REQUEST message encapsulated in UDP, encapsulated in IP, encapsulated in Ethernet
- Ethernet frame broadcast (dest: FFFF FFFFFFFF) on LAN, received at router running DHCP server
- Ethernet de-mux'ed to IP de-mux'ed, UDP de-mux'ed to DHCP

DHCP: Example (2 of 2)



- DCP server formulates DHCP ACK containing client's IP address, IP address of first-hop router for client, name & IP address of DNS server
- encapsulated DHCP server reply forwarded to client, de-muxing up to DHCP at client
- client now knows its IP address, name and IP address of DNS server, IP address of its first-hop router

IP Addresses: How to Get One? (2 of 2)

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

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

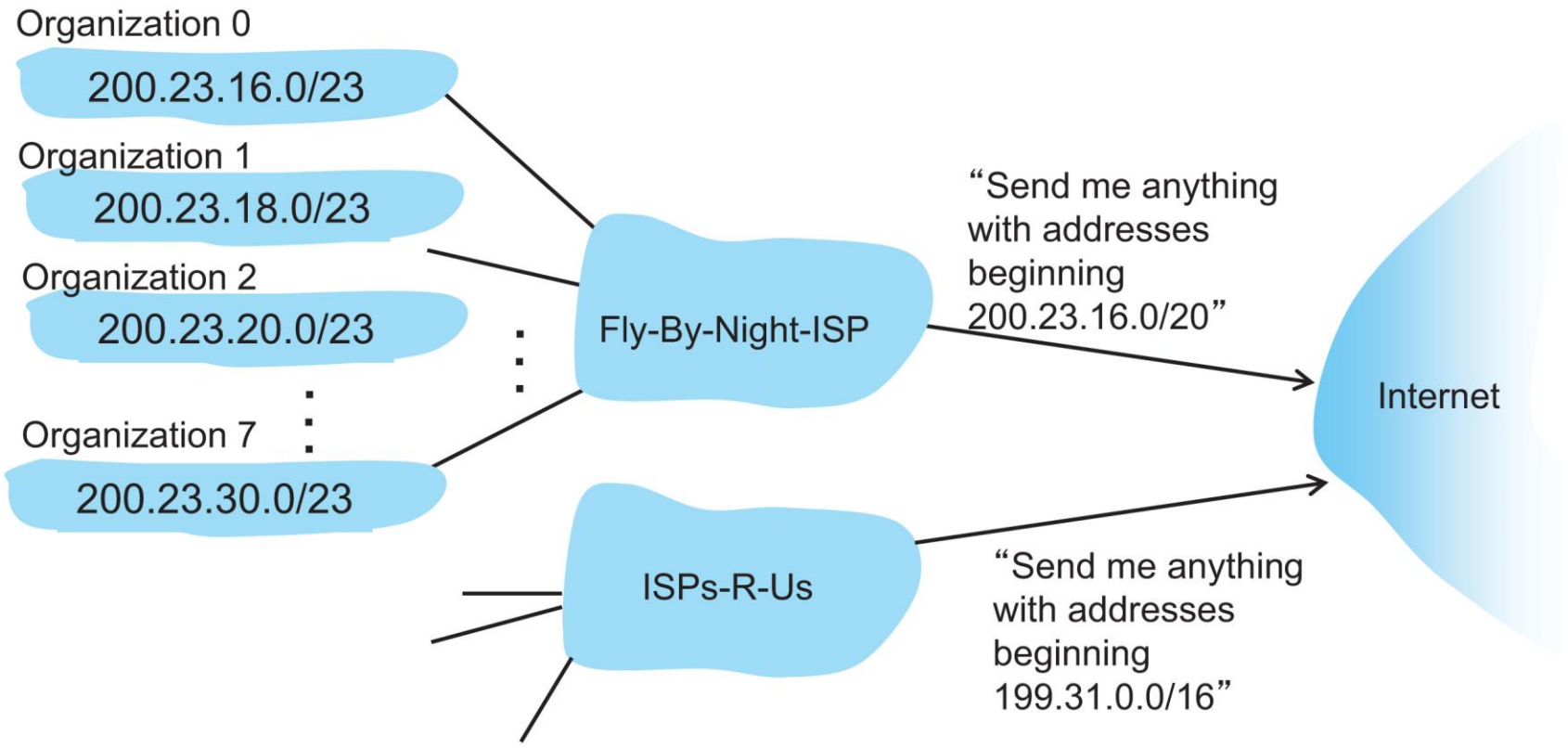
ISP's block	11001000	00010111	00010000	00000000	200.23.16.0/20

ISP can then allocate out its address space in 8 blocks:

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

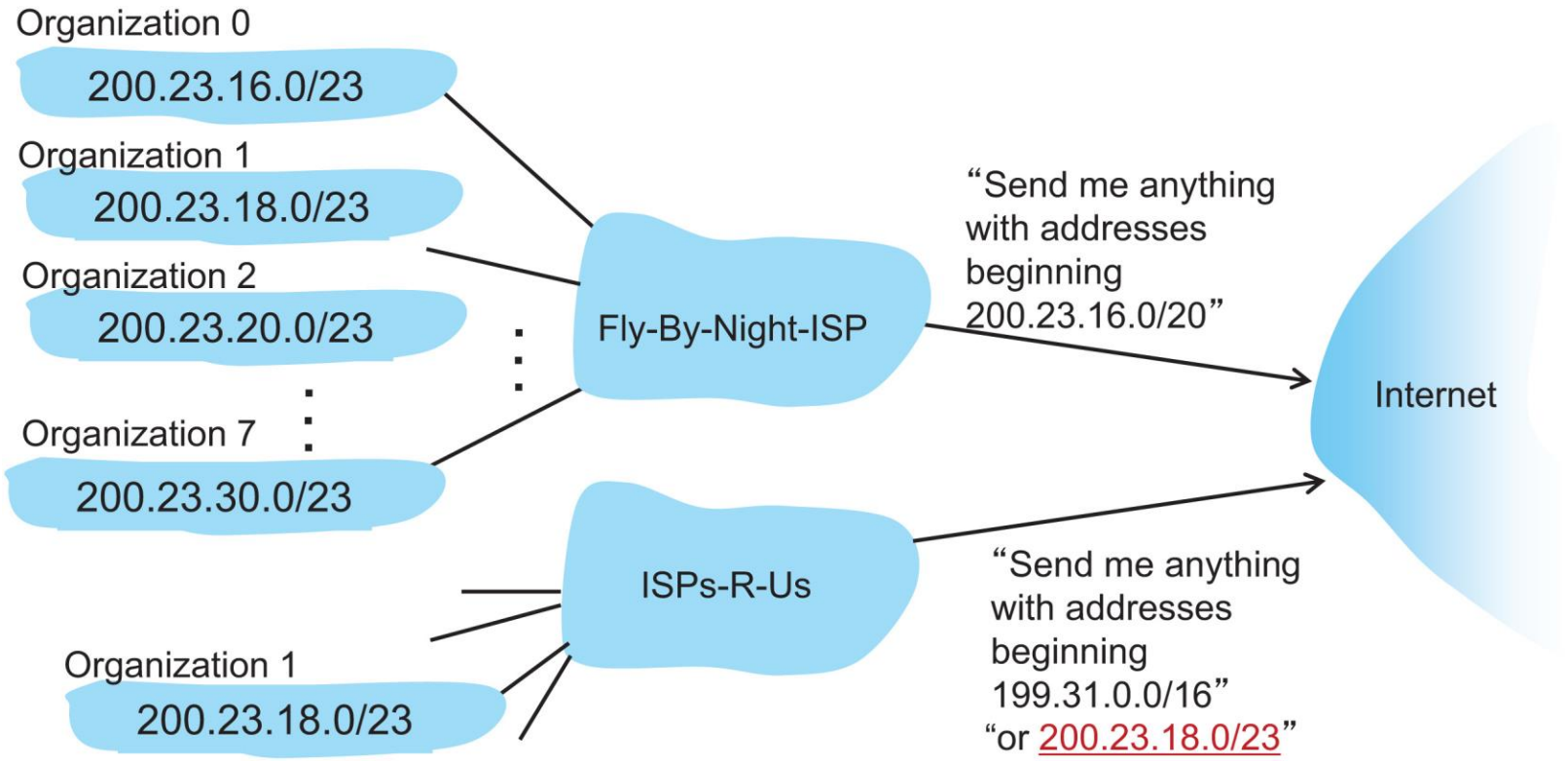
Hierarchical Addressing: Route Aggregation

hierarchical addressing allows efficient advertisement of routing information:



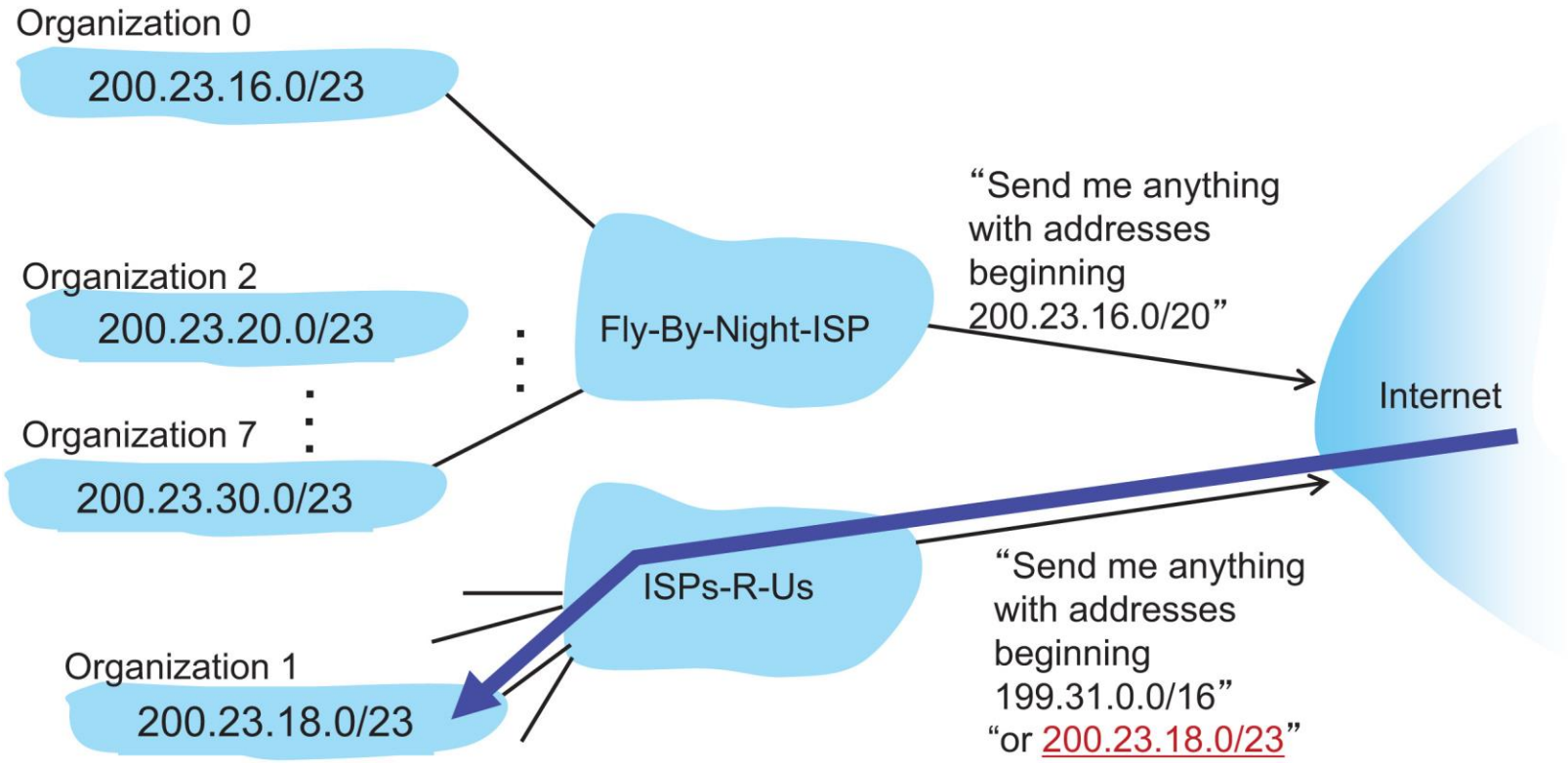
Hierarchical Addressing: More Specific Routes (1 of 2)

- Organization 1 moves from Fly-By-Night-ISP to ISPs-R-Us
- ISPs-R-Us now advertises a more specific route to Organization 1



Hierarchical Addressing: More Specific Routes (2 of 2)

- Organization 1 moves from Fly-By-Night-ISP to ISPs-R-Us
- ISPs-R-Us now advertises a more specific route to Organization 1



IP Addressing: Last Words ...

Q: how does an ISP get block of addresses?

A: ICANN: Internet Corporation for Assigned Names and Numbers

http://www.icann.org/

- allocates IP addresses, through 5 regional registries (RRs) (who may then allocate to local registries)
- manages DNS root zone, including delegation of individual TLD (.com, .edu , ...) management

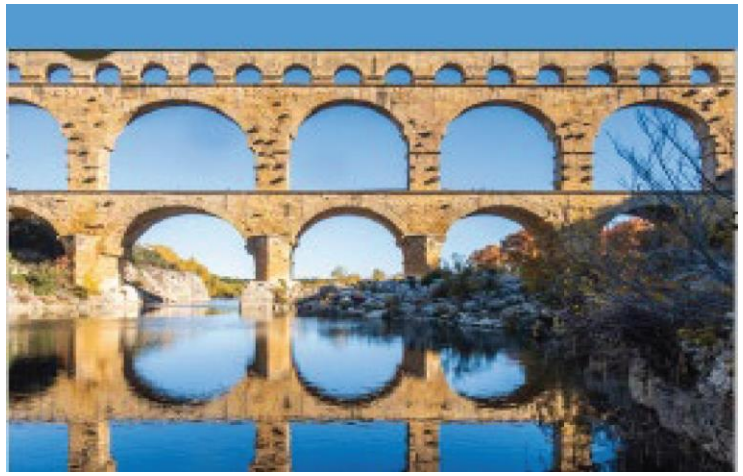
Q: are there enough 32-bit IP addresses?

- ICANN allocated last chunk of IPv4 addresses to RRs in 2011
- NAT (next) helps IPv4 address space exhaustion
- IPv6 has 128-bit address space

"Who the hell knew how much address space we needed?" Vint Cerf (reflecting on decision to make IPv4 address 32 bits long)

Network Layer: "Data Plane" Roadmap (4 of 6)

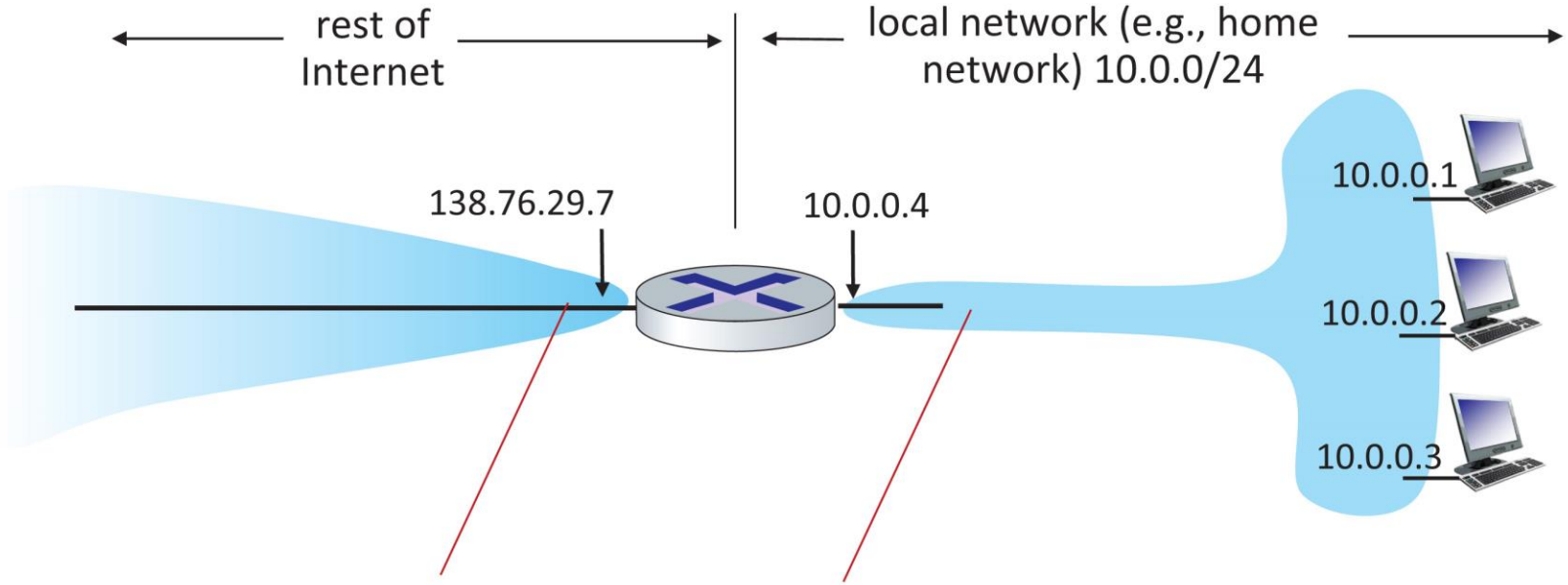
- Network layer: overview
  - data plane
  - control plane
- What's inside a router
  - input/output ports, switching
  - buffer management
  - scheduling
- IP: the Internet Protocol
  - datagram format, addressing
  - network address translation (NAT)
  - IPv6



- Generalized Forwarding
  - Match+action
  - OpenFlow
  - Middleboxes
- Internet architecture

NAT: Network Address Translation (1 of 5)

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 (2 of 5)

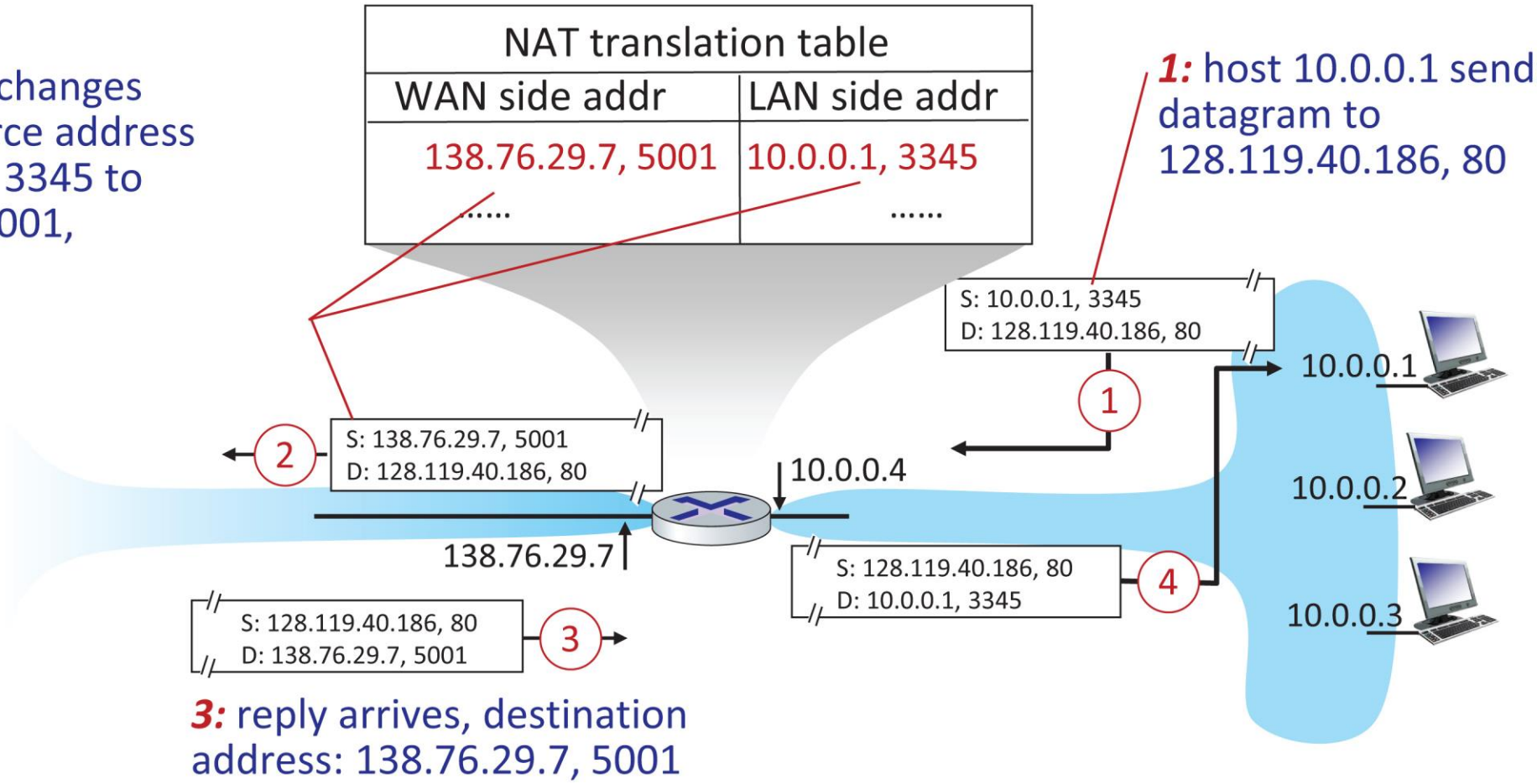
- all devices in local network have 32-bit addresses in a "private" IP address space (10/8, 172.16/12, 192.168/16 prefixes) that can only be used in local network
- advantages:
  - just one IP address needed from provider ISP for all devices
  - can change addresses of host in local network without notifying outside world
  - can change ISP without changing addresses of devices in local network
  - security: devices inside local net not directly addressable, visible by outside world

NAT: Network Address Translation (3 of 5)

implementation: NAT router must (transparently):

- 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 (4 of 5)



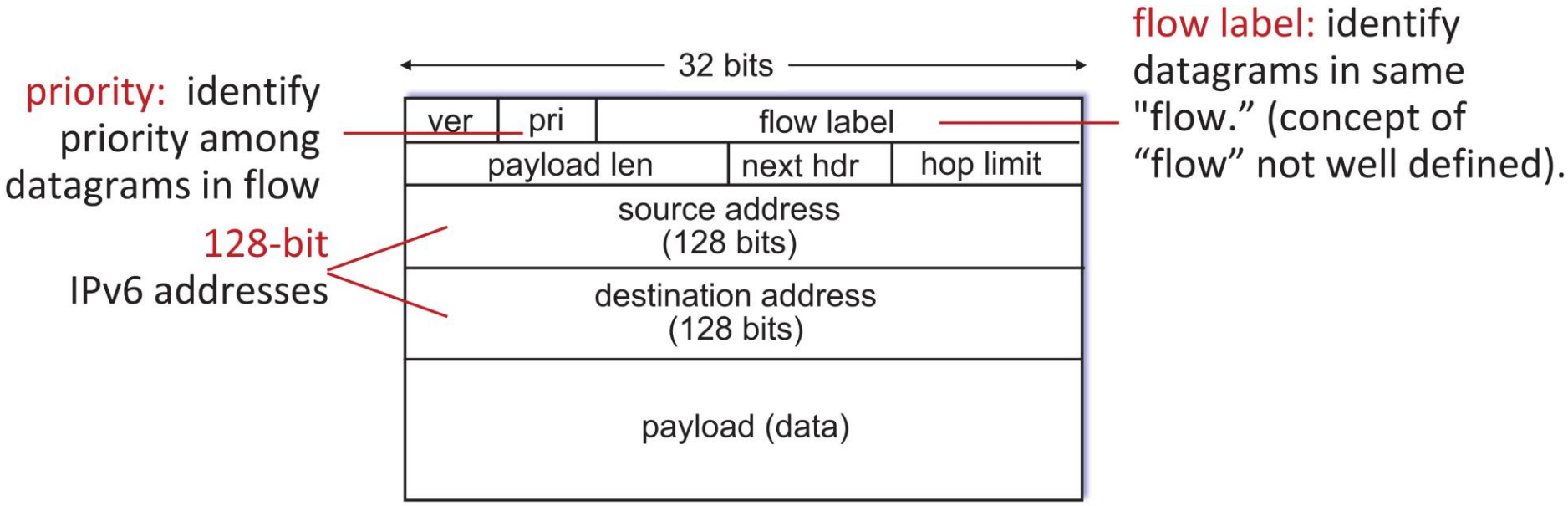
NAT: Network Address Translation (5 of 5)

- NAT has been controversial:
  - routers "should" only process up to layer 3
  - address "shortage" should be solved by IPv6
  - violates end-to-end argument (port # manipulation by network-layer device)
  - NAT traversal: what if client wants to connect to server behind NAT?
- but NAT is here to stay:
  - extensively used in home and institutional nets, 4G/5G cellular nets

IPv6: Motivation

- initial motivation: 32-bit IPv4 address space would be completely allocated
- additional motivation:
  - speed processing/forwarding: 40-byte fixed length header
  - enable different network-layer treatment of “flows”

IPv6 Datagram Format

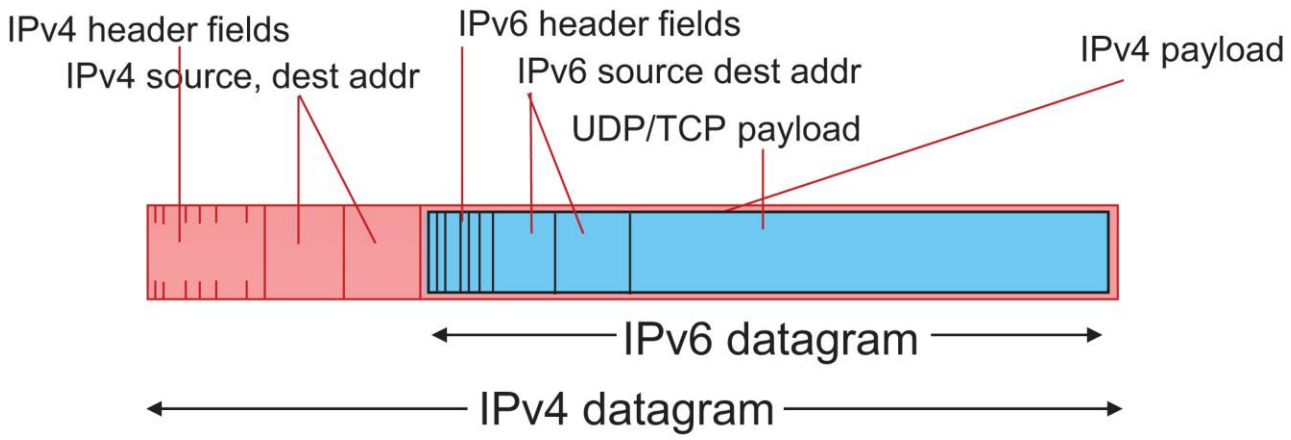


What's missing (compared with IPv4):

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

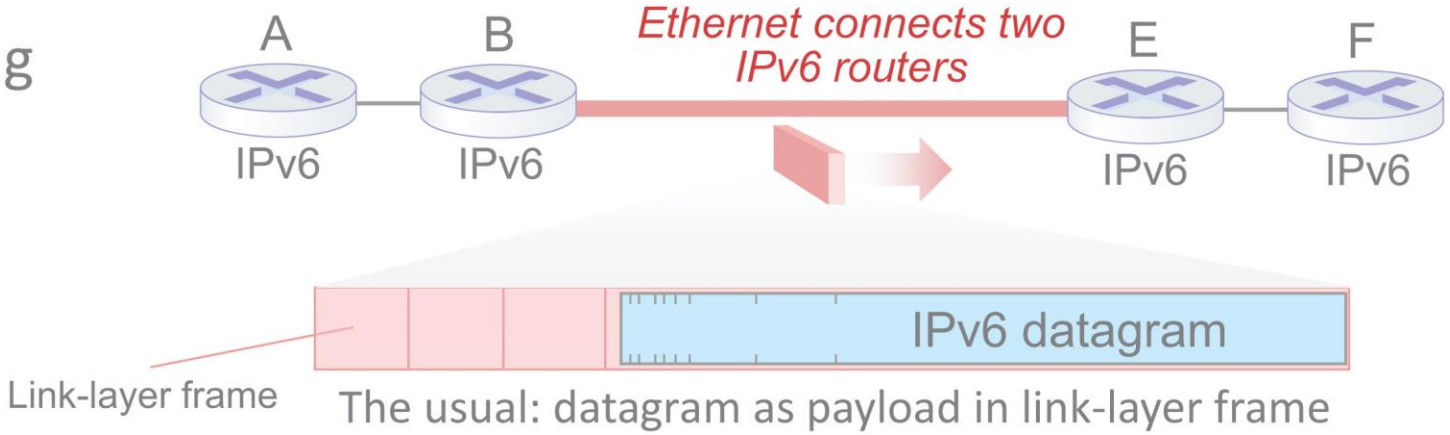
Transition From IPv4 to IPv6

- 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 (“packet within a packet”)
  - tunneling used extensively in other contexts (4G/5G)

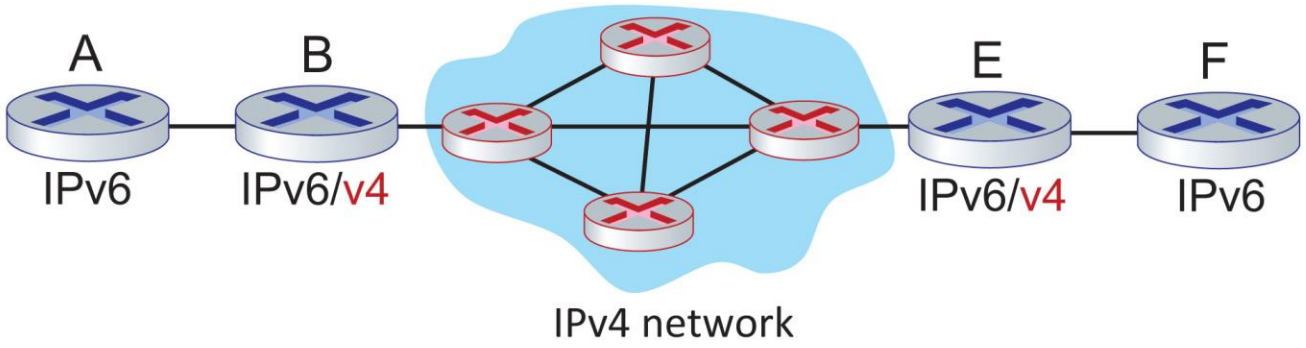


Tunneling and Encapsulation (1 of 2)

Ethernet connecting two IPv6 routers:

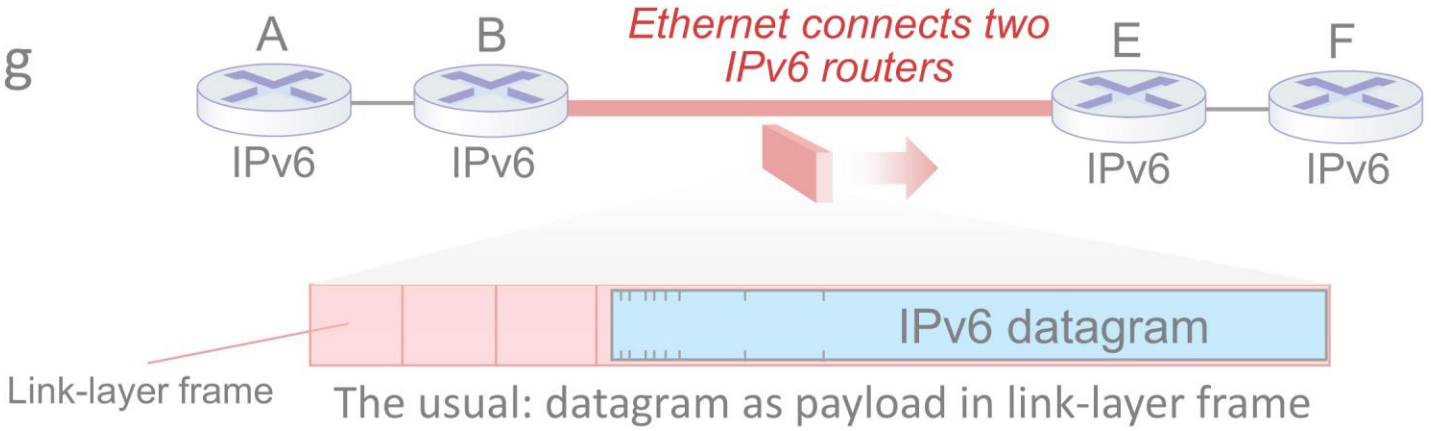


IPv4 network connecting two IPv6 routers

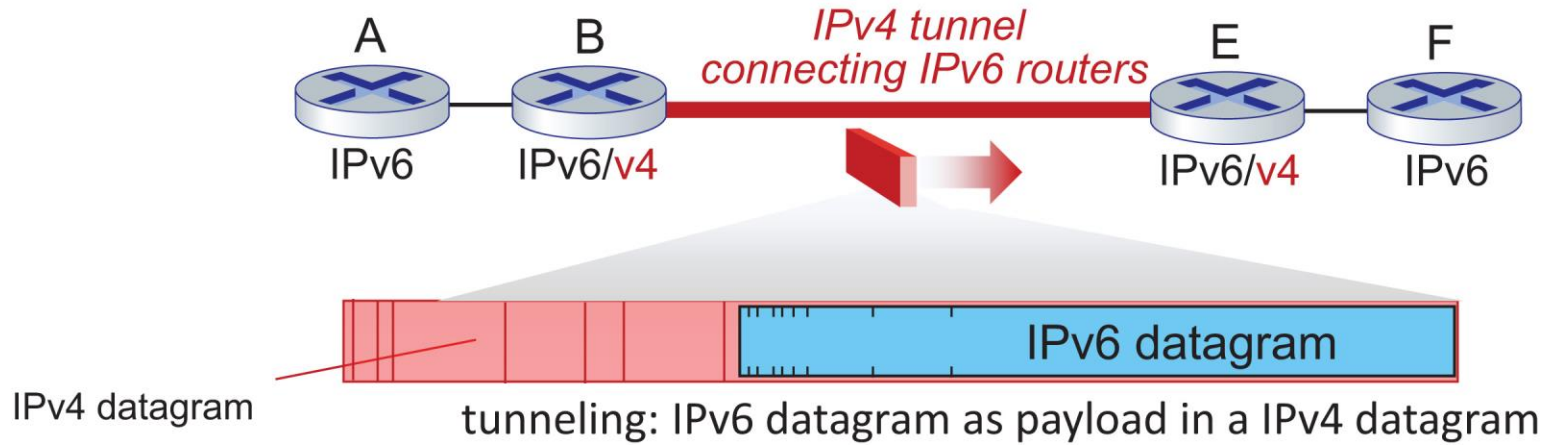


Tunneling and Encapsulation (2 of 2)

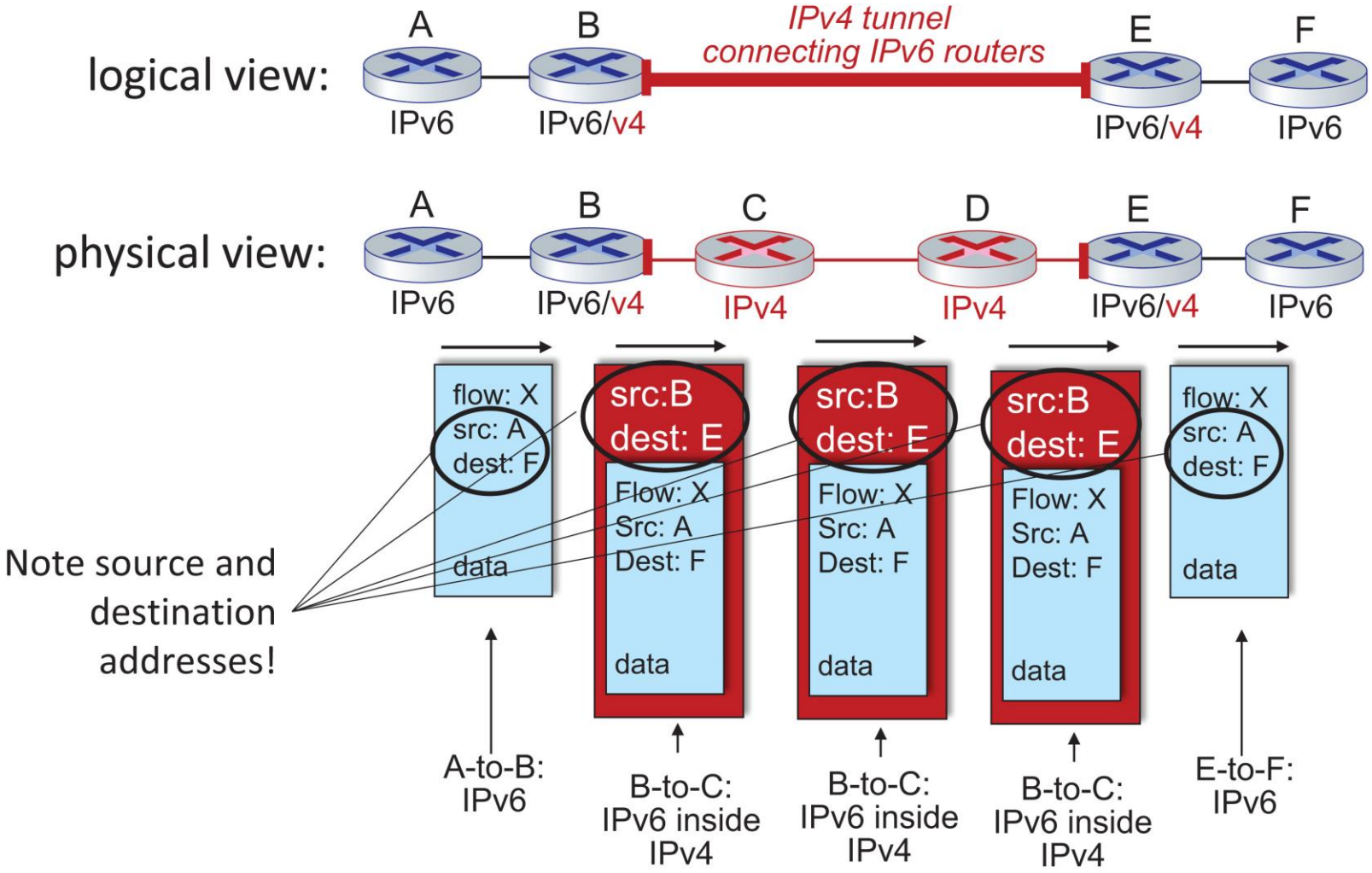
Ethernet connecting two IPv6 routers:



IPv4 tunnel connecting two IPv6 routers

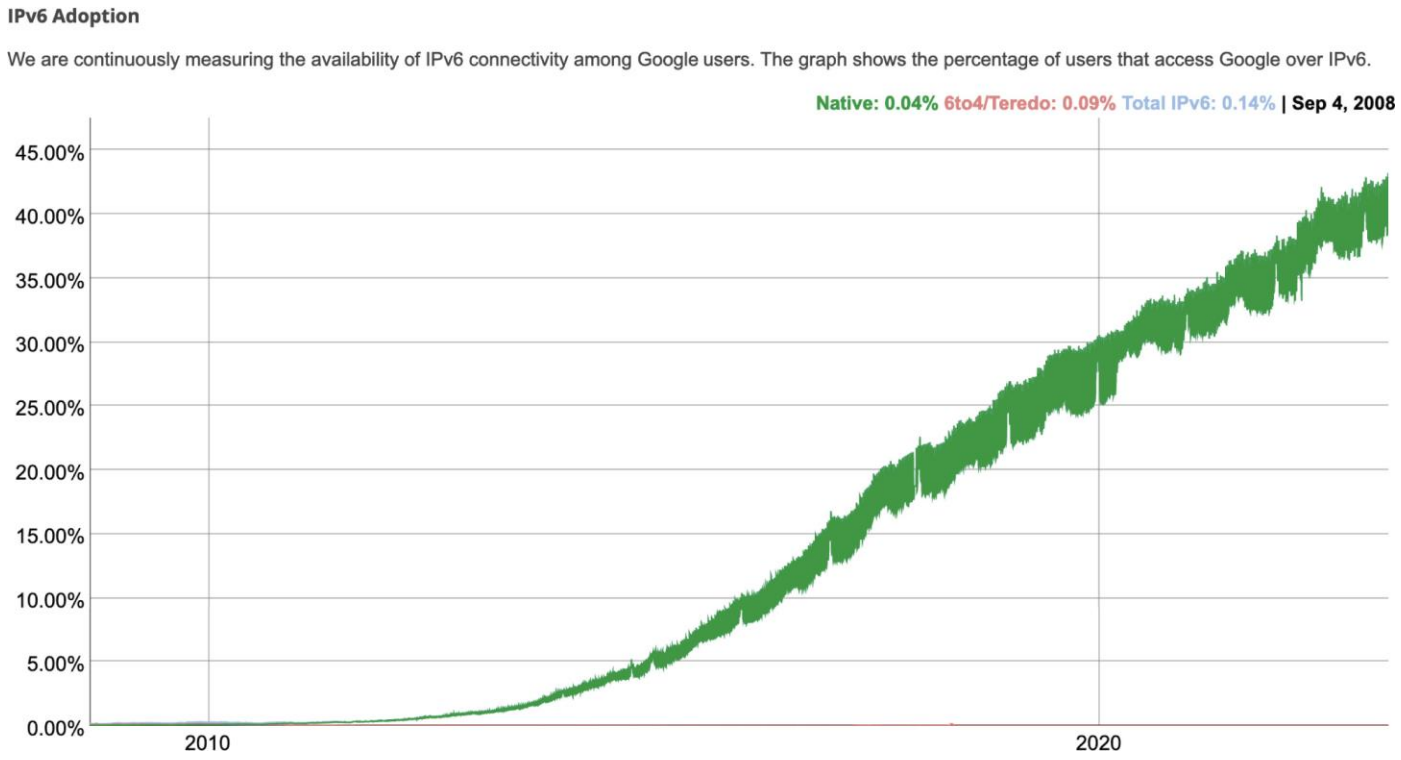


Tunneling


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

IPv6: Adoption (1 of 2)

- Google [1]: ~ 40 % of clients access services via IPv6 (2023)
- NIST: 1/ 3 of all US government domains are IPv6 capable



IPv6: Adoption (2 of 2)

- Google [1]: ~ 40 % of clients access services via IPv6 (2023)
- 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?

1 https://www.google.com/intl/en/ipv6/statistics.html