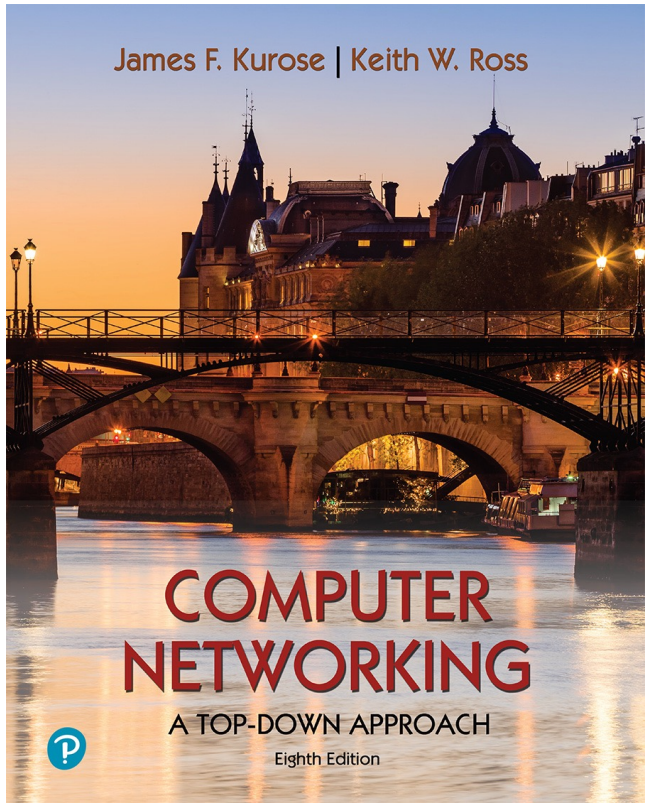
Chapter 4
Network Layer: Data Plane

Yaxiong Xie

Department of Computer Science and Engineering
University at Buffalo, SUNY

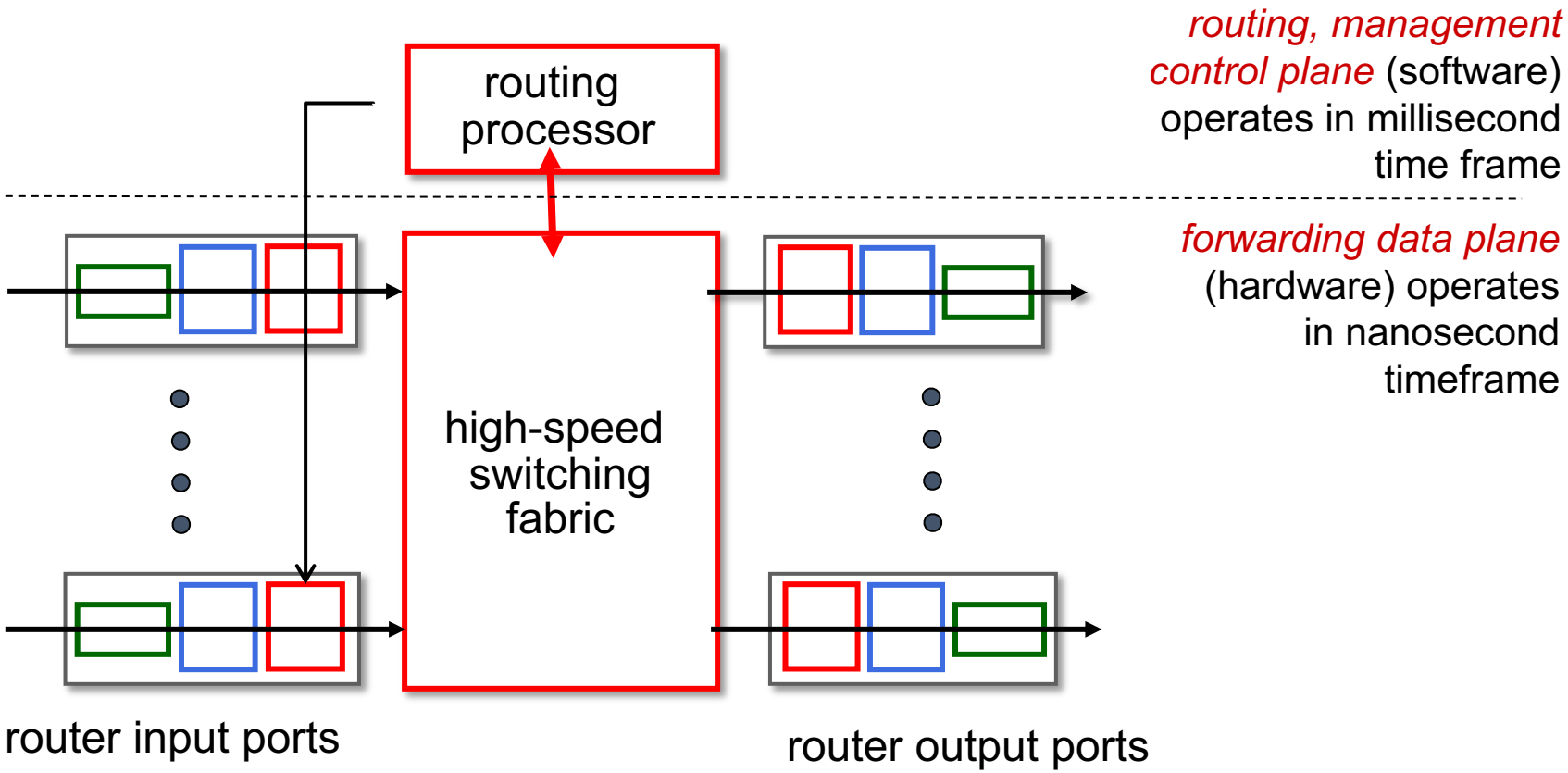


Computer Networking: A Top-Down Approach
8th edition
Jim Kurose, Keith Ross
Pearson, 2020

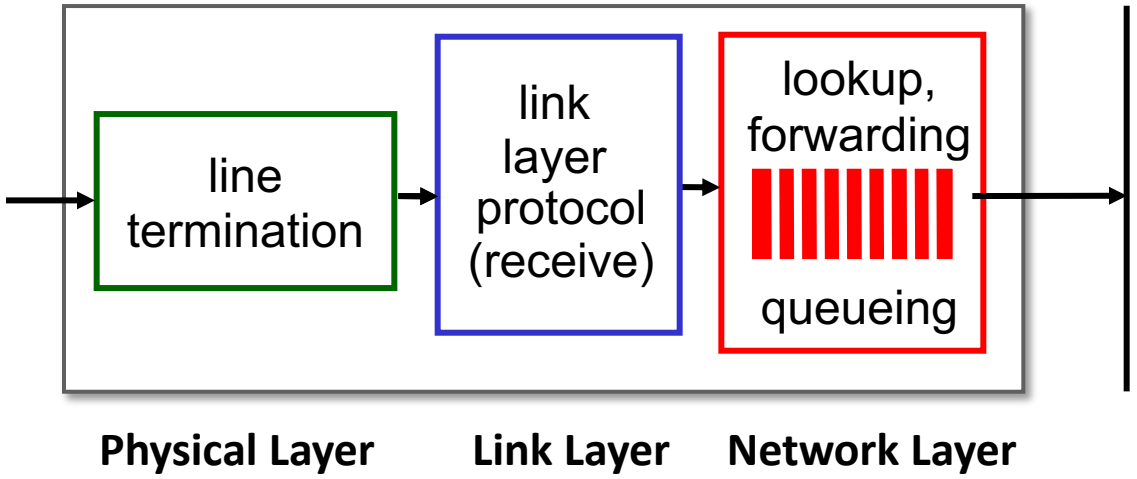
Adapted from the slides of the book's authors

Router architecture overview

high-level view of generic router architecture:

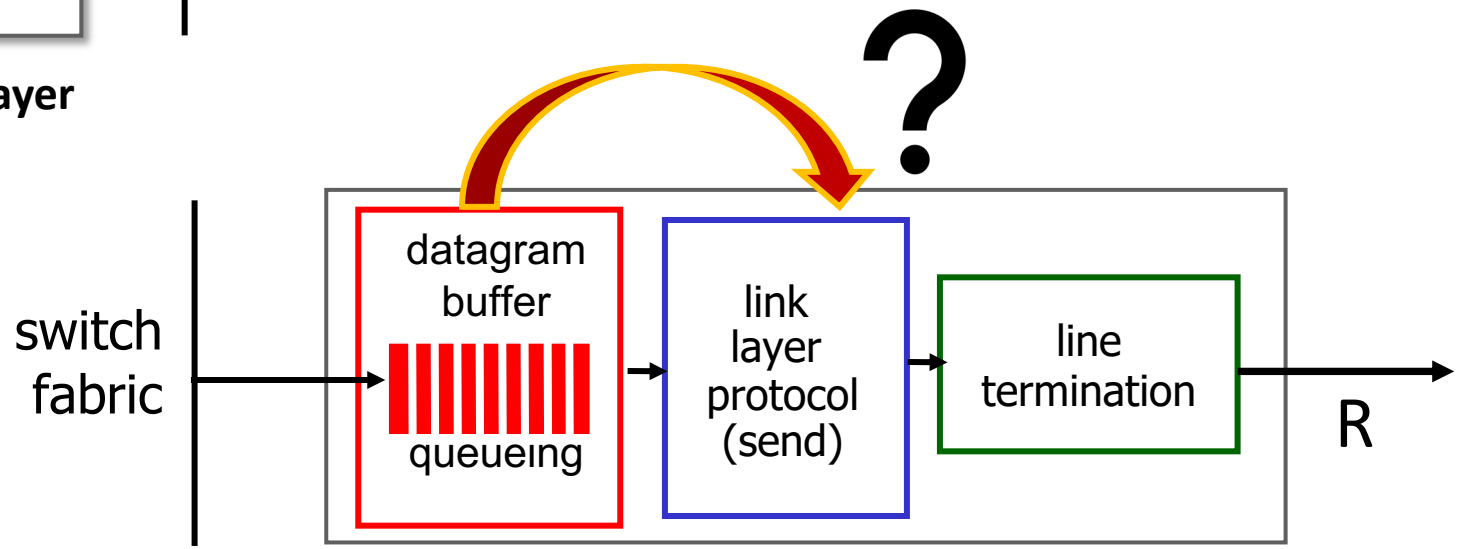


Output port queuing

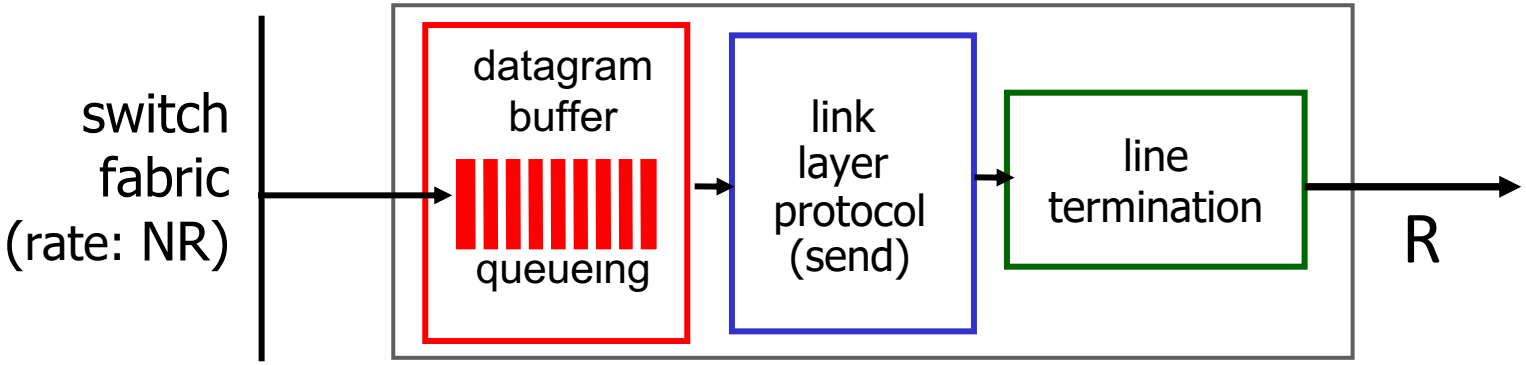


Architecture of Input Port

Architecture of Output Port



Output port queuing



This is a really important slide

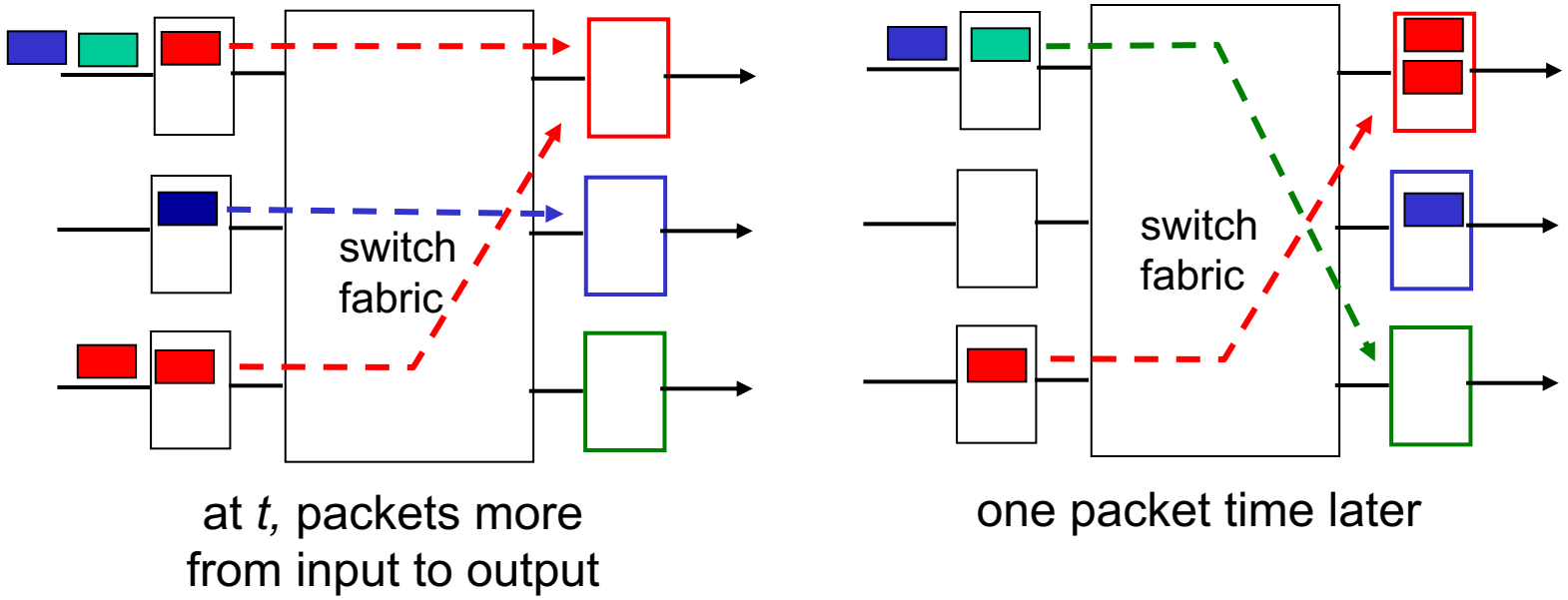
- *Buffering* required when datagrams arrive from fabric faster than link transmission rate. *Drop policy:* which datagrams to drop if no free buffers?

→ 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



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?

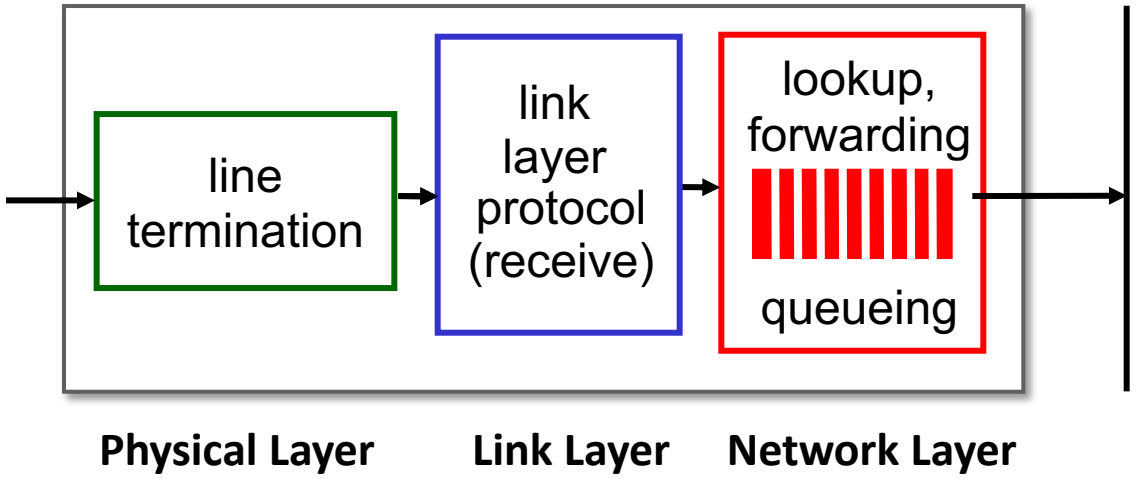
- RFC 3439 rule of thumb: average buffering equal to “typical” RTT (say 250 msec) times link capacity C
  - e.g., C = 10 Gbps link: 2.5 Gbit buffer
- more recent recommendation: with *N* flows, buffering equal to

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

- 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”

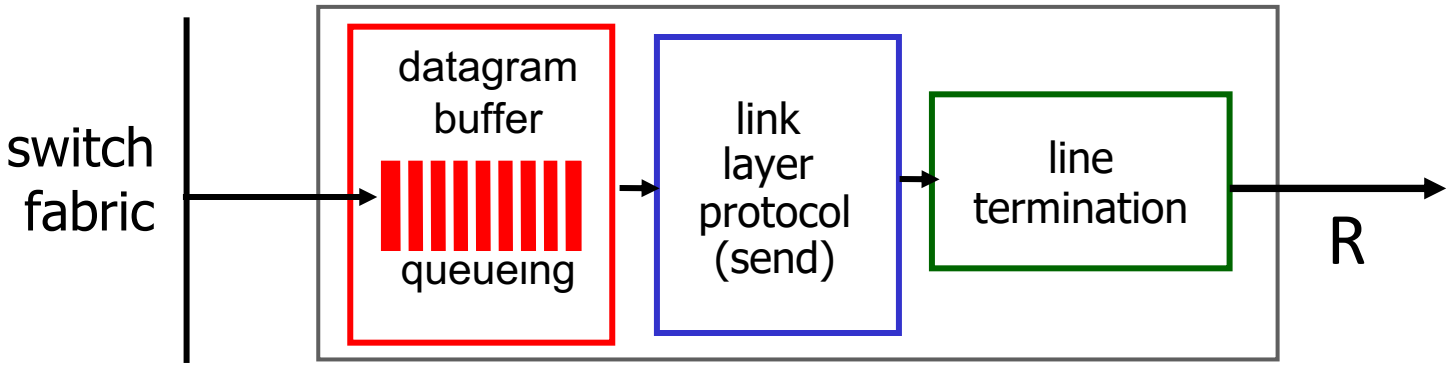
Buffer-induced Challenges and Opportunities

Output port queuing: Buffer Management

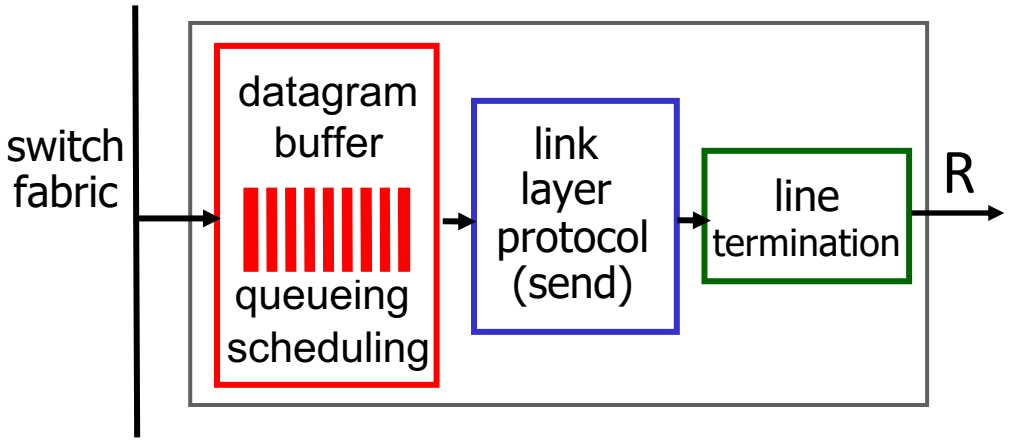


Architecture of Input Port

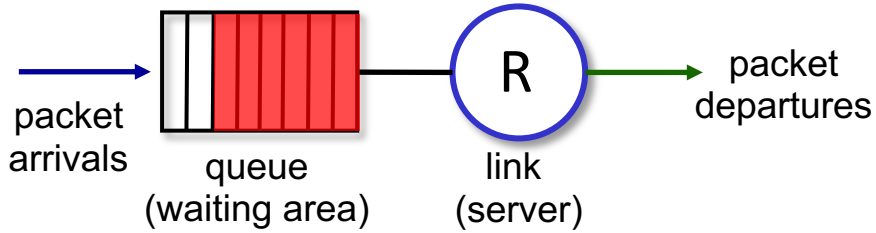
Architecture of Output Port



Buffer Management

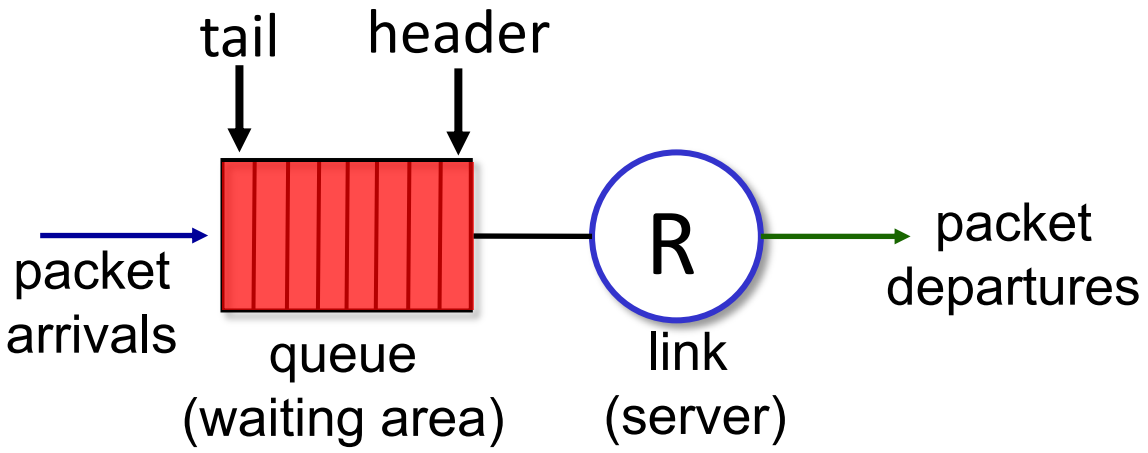


Abstraction: queue



Buffer Management

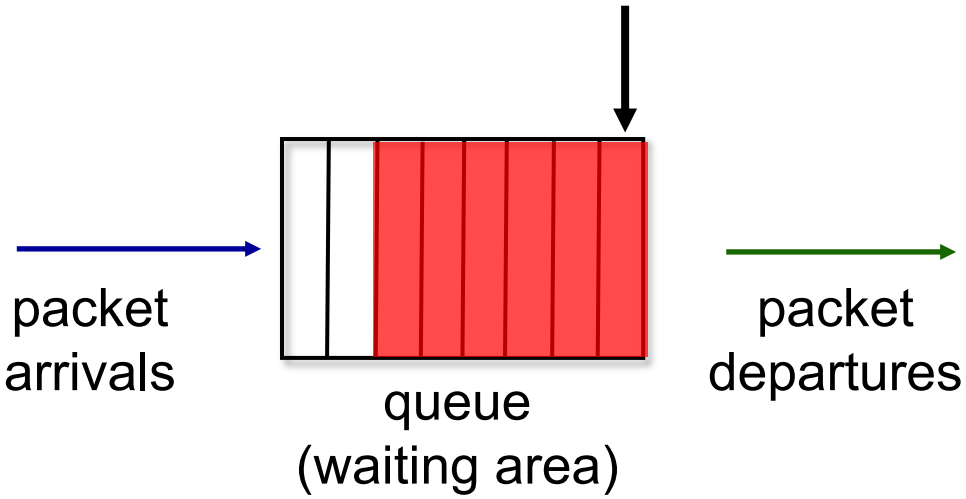
Abstraction: queue



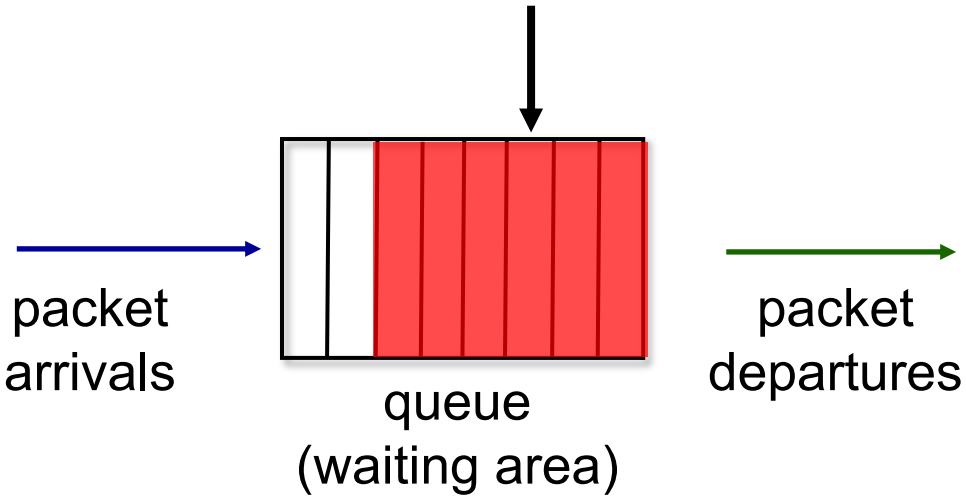
buffer management:

- 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)

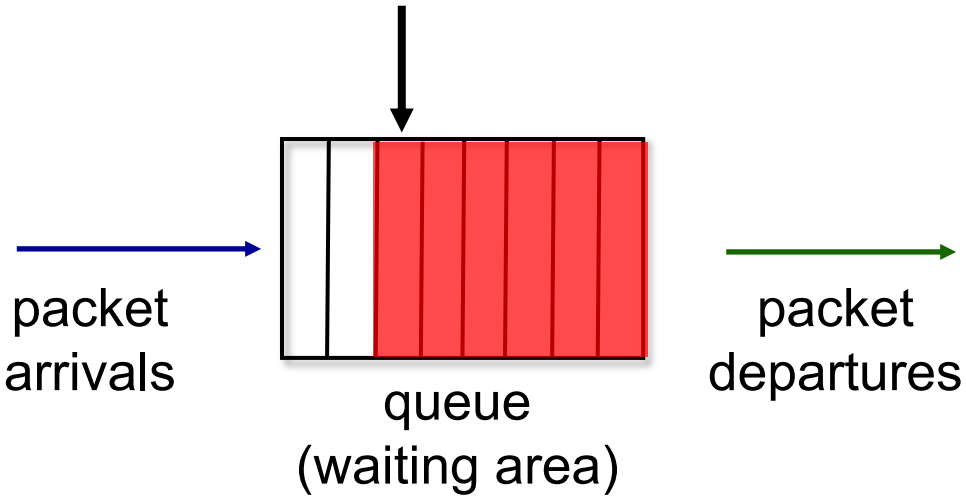
Packet Scheduling



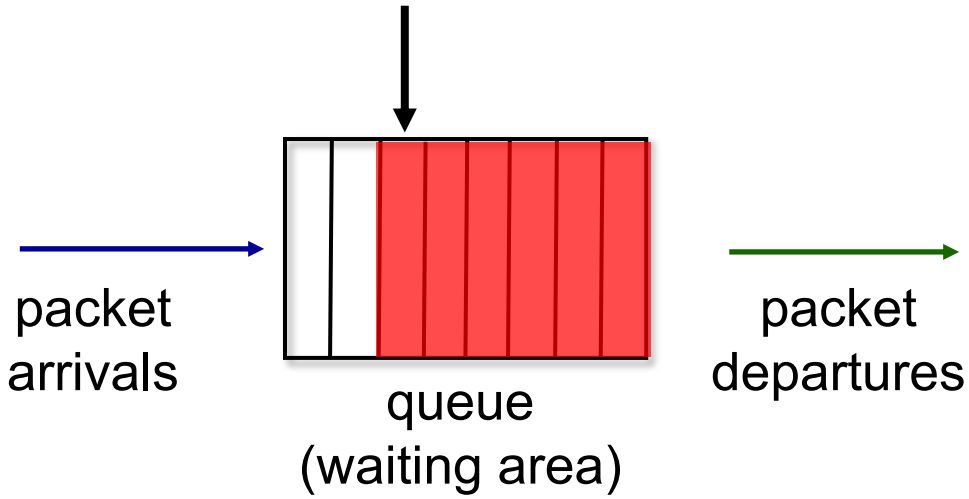
Packet Scheduling



Packet Scheduling



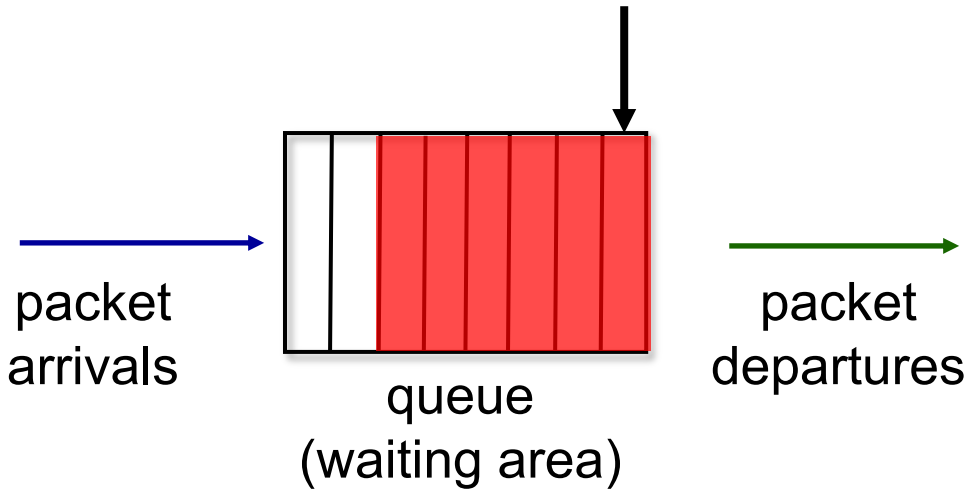
Packet Scheduling



packet scheduling: deciding which packet to send next on link

- first come, first served
- priority
- round robin
- weighted fair queueing

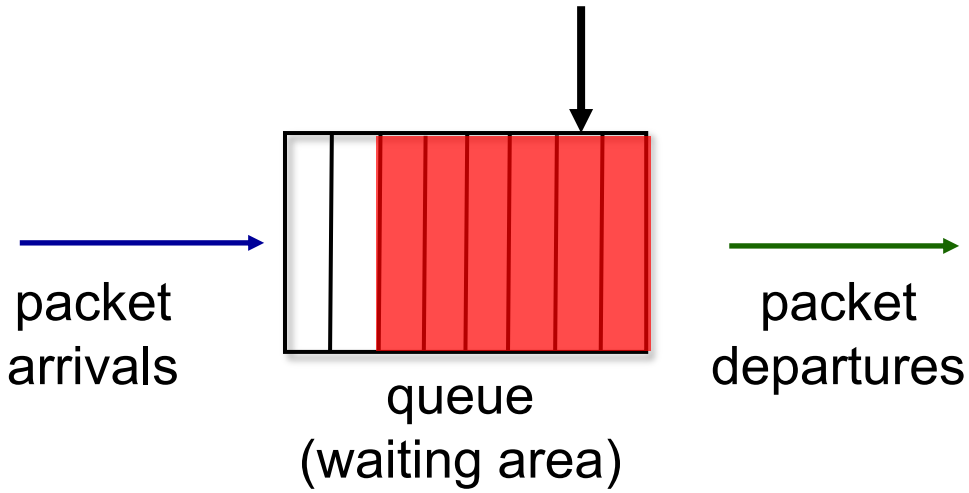
Packet Scheduling: FCFS



FCFS: packets transmitted in order of arrival to output port

- also known as: First-in-first-out (FIFO)

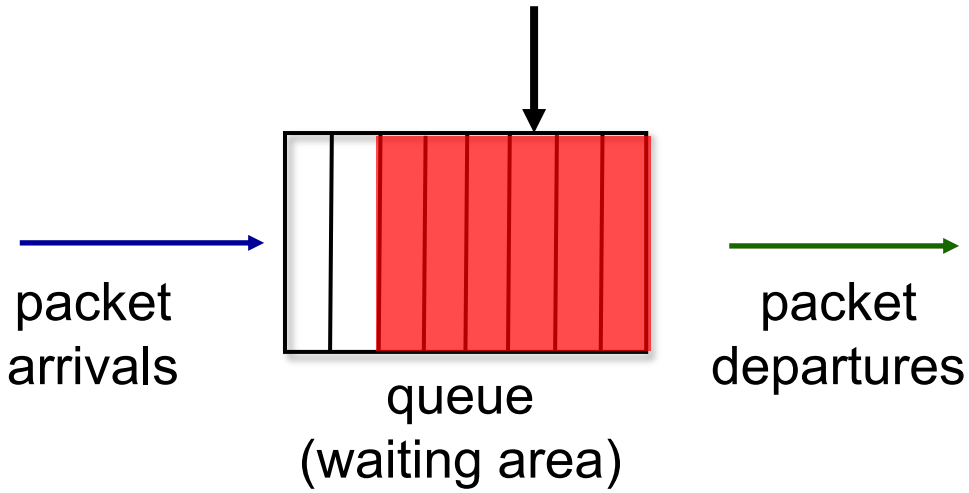
Packet Scheduling: FCFS



FCFS: packets transmitted in order of arrival to output port

- also known as: First-in-first-out (FIFO)

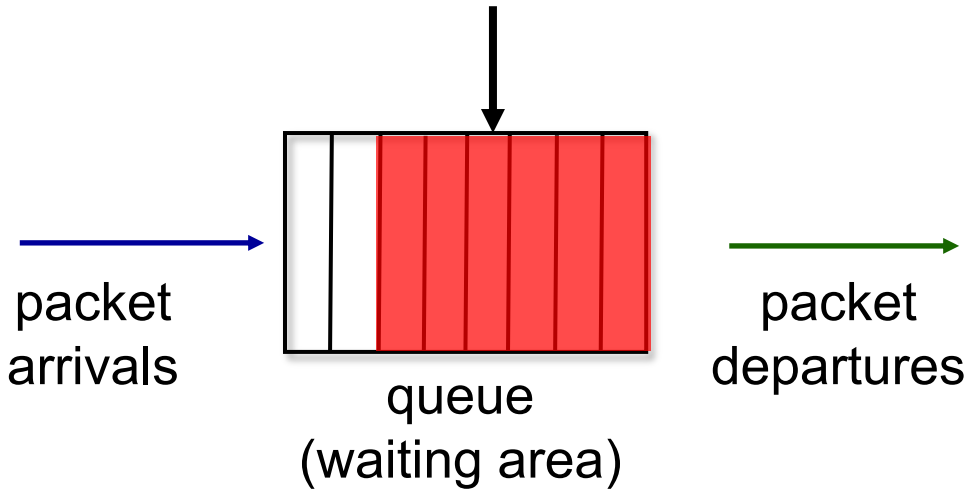
Packet Scheduling: FCFS



FCFS: packets transmitted in order of arrival to output port

- also known as: First-in-first-out (FIFO)

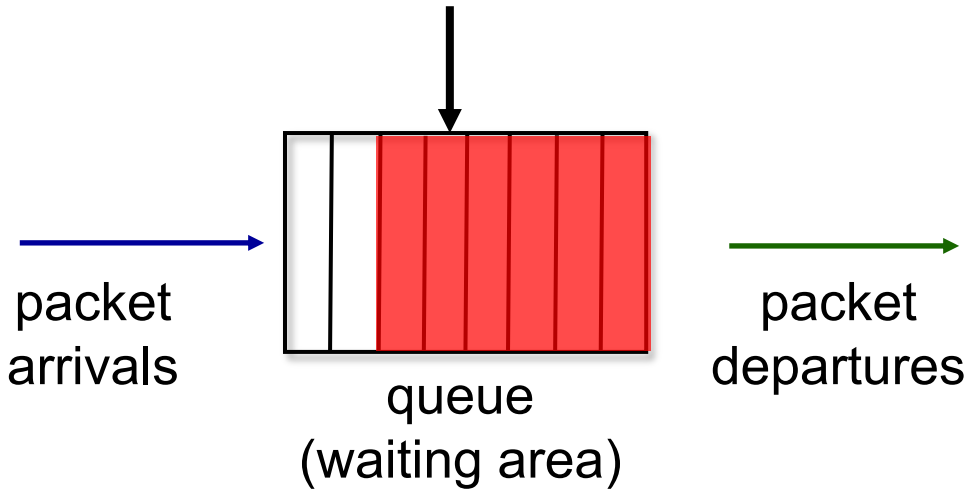
Packet Scheduling: FCFS



FCFS: packets transmitted in order of arrival to output port

- also known as: First-in-first-out (FIFO)

Packet Scheduling: FCFS



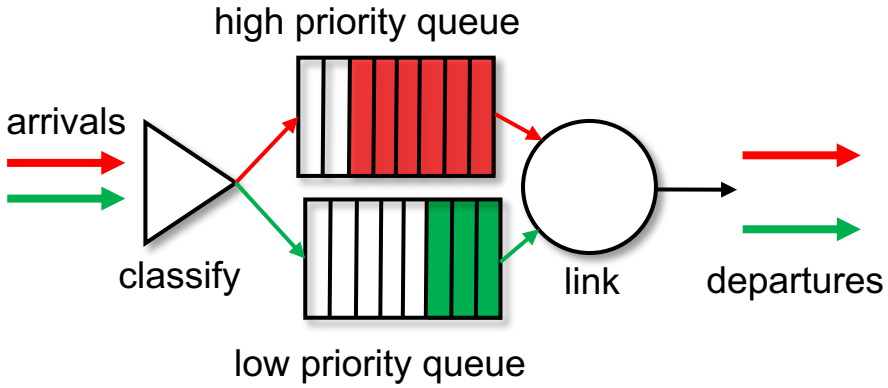
FCFS: packets transmitted in order of arrival to output port

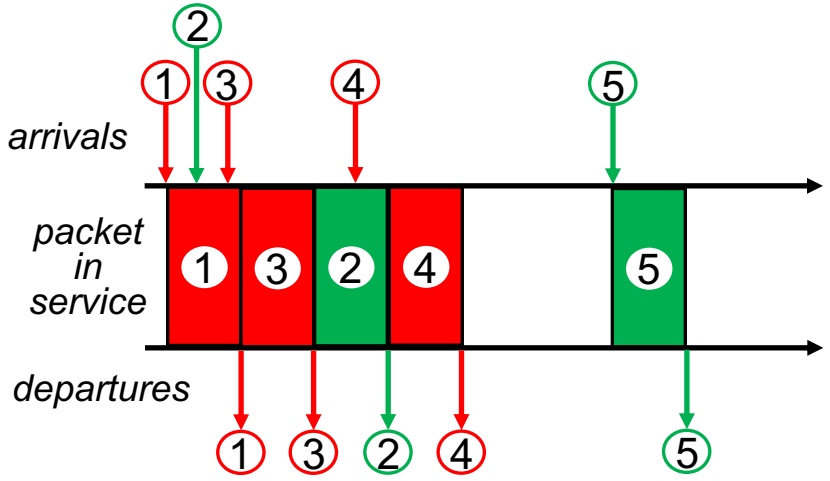
- also known as: First-in-first-out (FIFO)

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

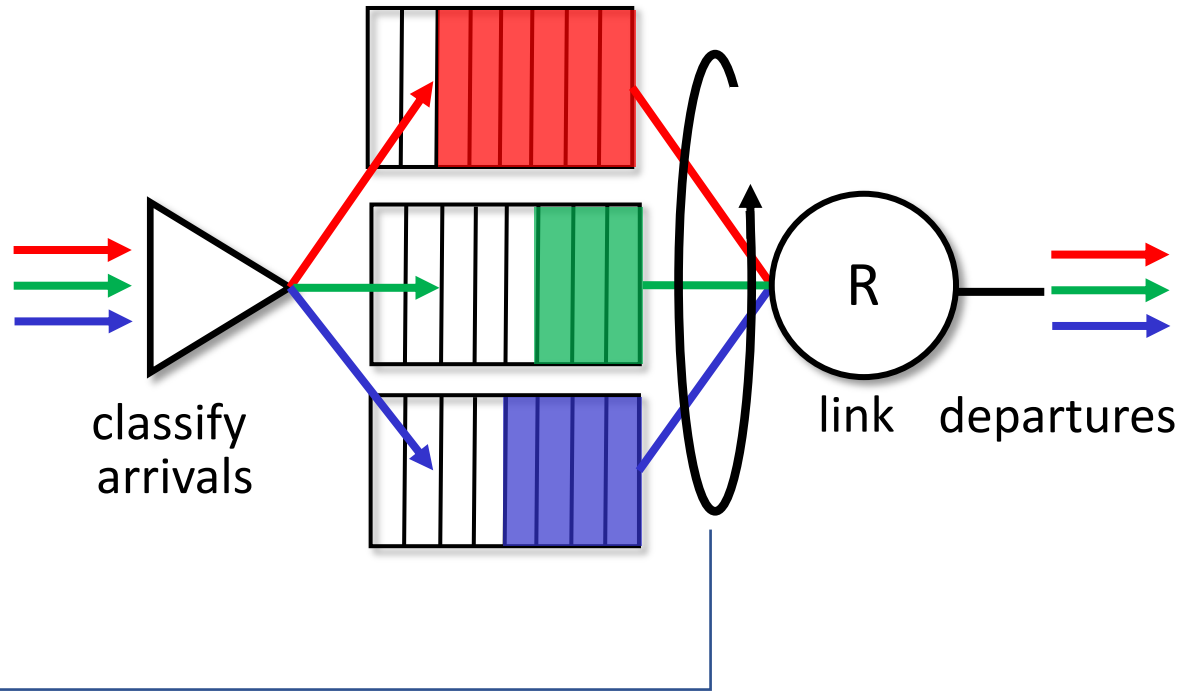




Scheduling policies: round robin

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



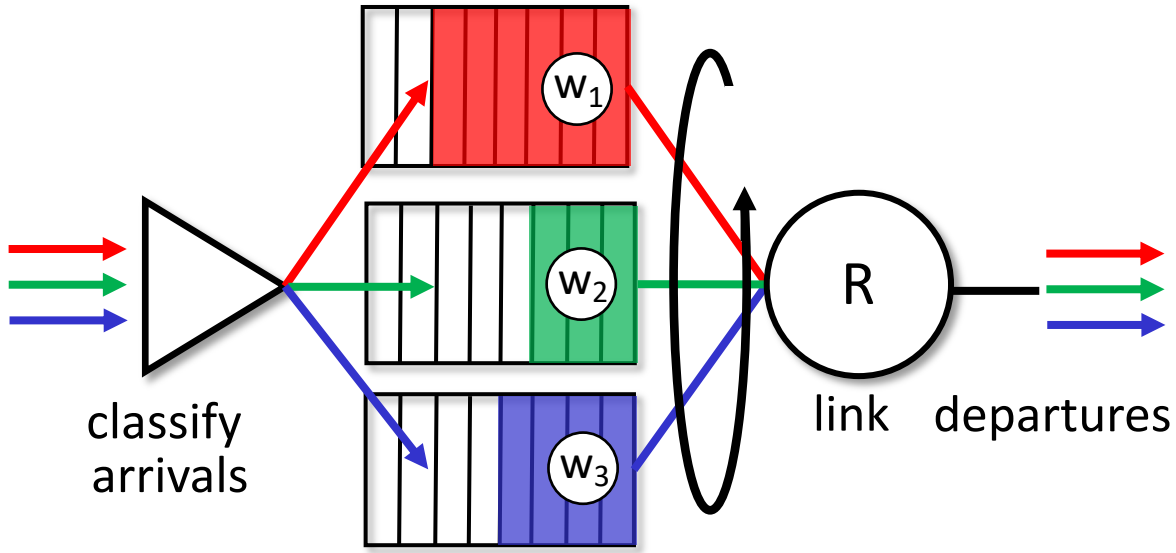
Scheduling policies: weighted fair queueing

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)



Sidebar: Network Neutrality

What is network neutrality?

- *technical:* how an ISP should share/allocation its resources
  - packet scheduling, buffer management are the *mechanisms*
- *social, economic* principles
  - protecting free speech
  - encouraging innovation, competition
- enforced *legal* rules and policies

Different countries have different "takes" on network neutrality

Sidebar: Network Neutrality

2015 US FCC *Order on Protecting and Promoting an Open Internet:* three "clear, bright line" rules:

- no blocking … "shall not block lawful content, applications, services, or non-harmful devices, subject to reasonable network management."
- no throttling … "shall not impair or degrade lawful Internet traffic on the basis of Internet content, application, or service, or use of a non-harmful device, subject to reasonable network management."
- no paid prioritization. … "shall not engage in paid prioritization"

ISP: telecommunications or information service?

Is an ISP a "telecommunications service" or an "information service" provider?

- the answer *really* matters from a regulatory standpoint!

US Telecommunication Act of 1934 and 1996:

- *Title II:* imposes "common carrier duties" on *telecommunications services*: reasonable rates, non-discrimination and *requires regulation*
- *Title I:* applies to *information services:*
  - no common carrier duties (*not regulated*)
  - but grants FCC authority "… as may be necessary in the execution of its functions"

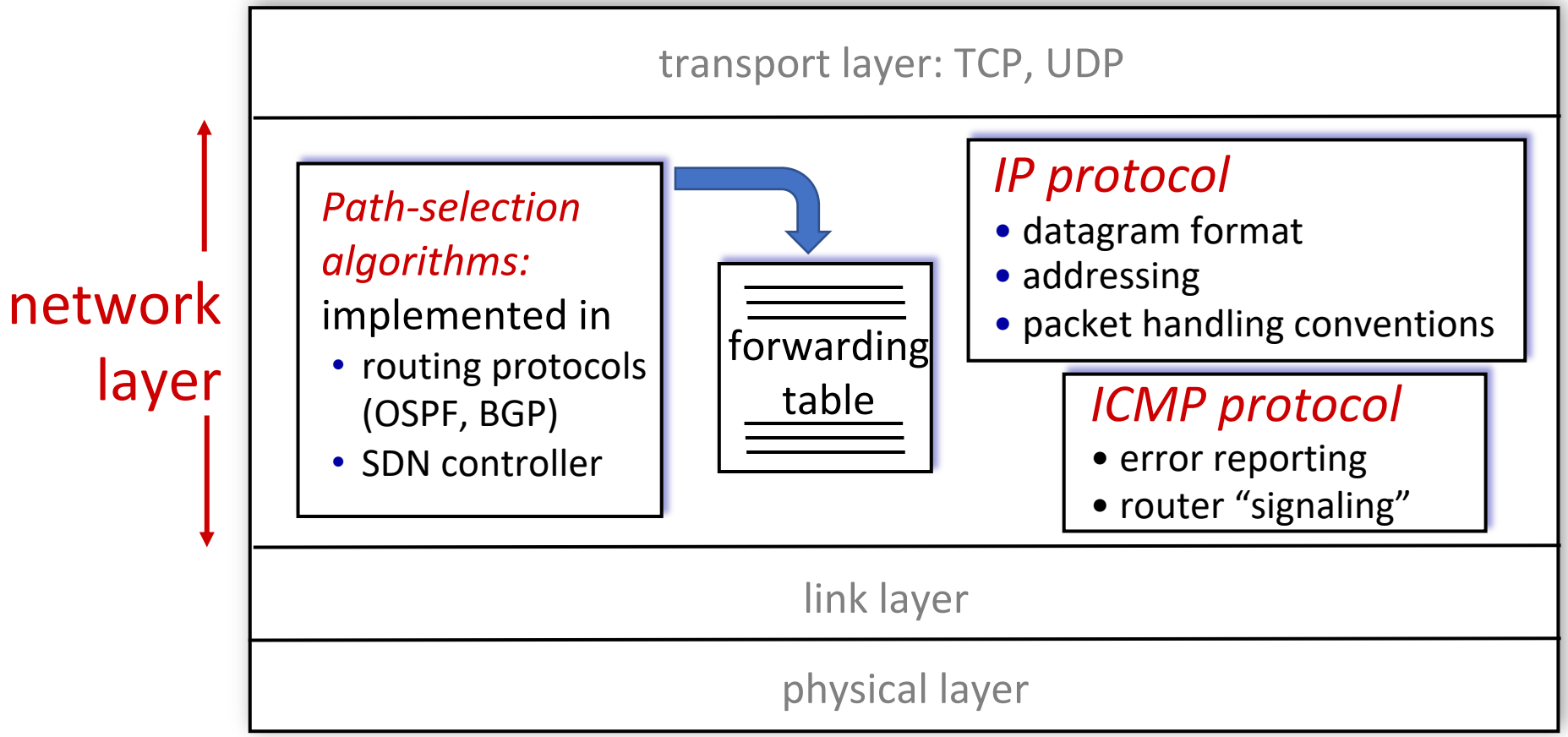
Network layer: “data plane” roadmap

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

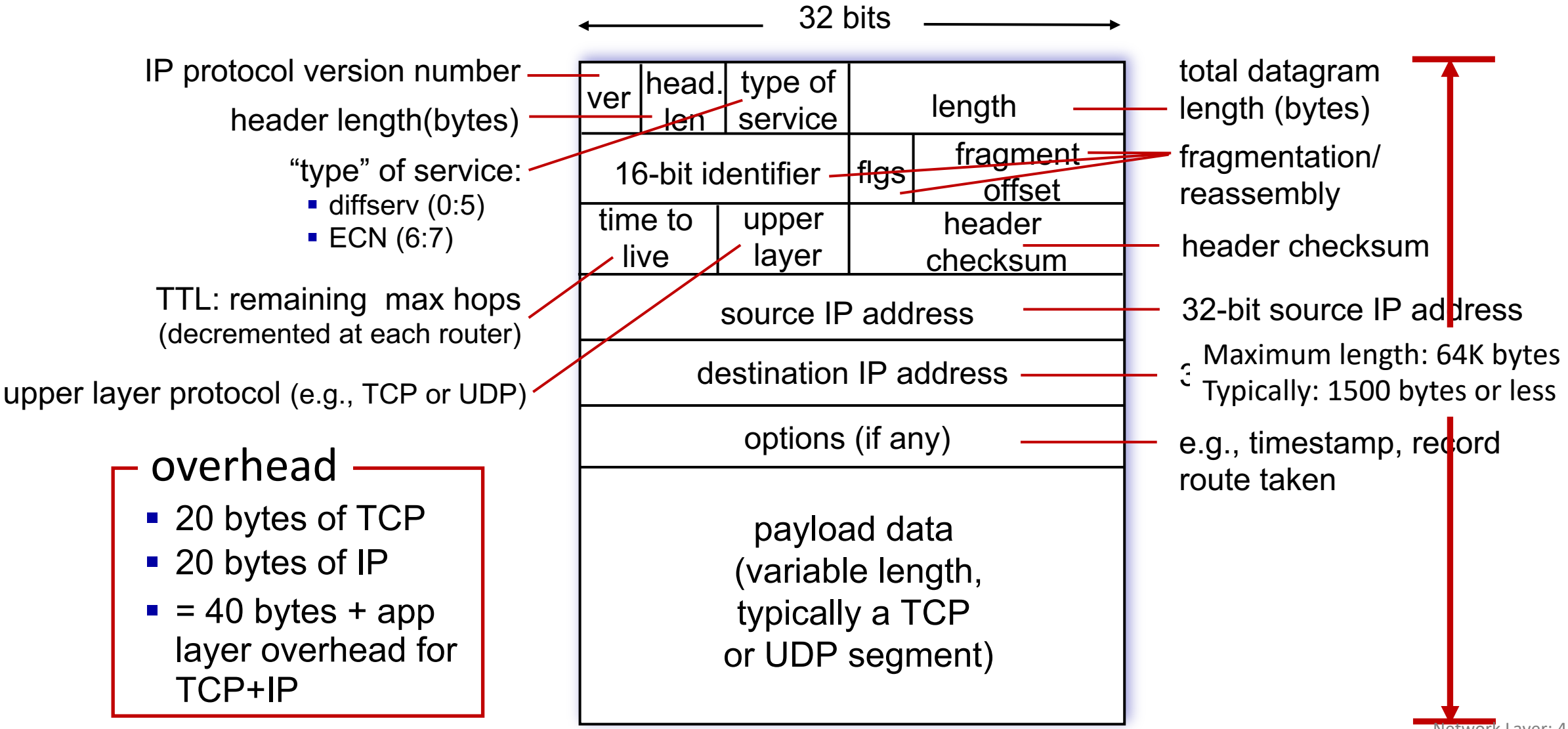
- Generalized Forwarding, SDN
  - match+action
  - OpenFlow: match+action in action
- Middleboxes

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

IP addressing: introduction

- 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)

IP addressing: Format

Dotted-decimal IP address notation: **IPv4: 4 bytes**

223.1.1.1 = 11011111 00000001 00000001 00000001

223 1 1 1

$$2^{32} = 4{,}294{,}967{,}296$$

IPv6: 16 bytes

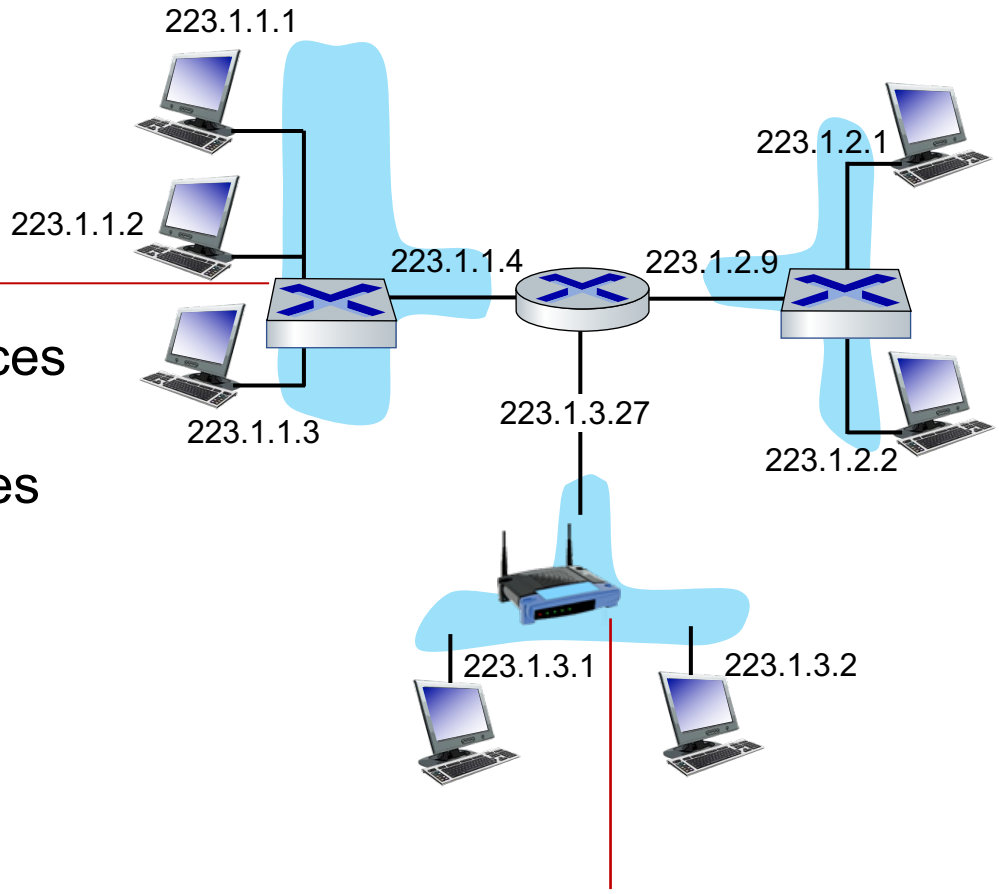
$$2^{128} = 3.4 \times 10^{38}$$

340,282,366,920,938,463,463,374,607,431,768,211,456

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

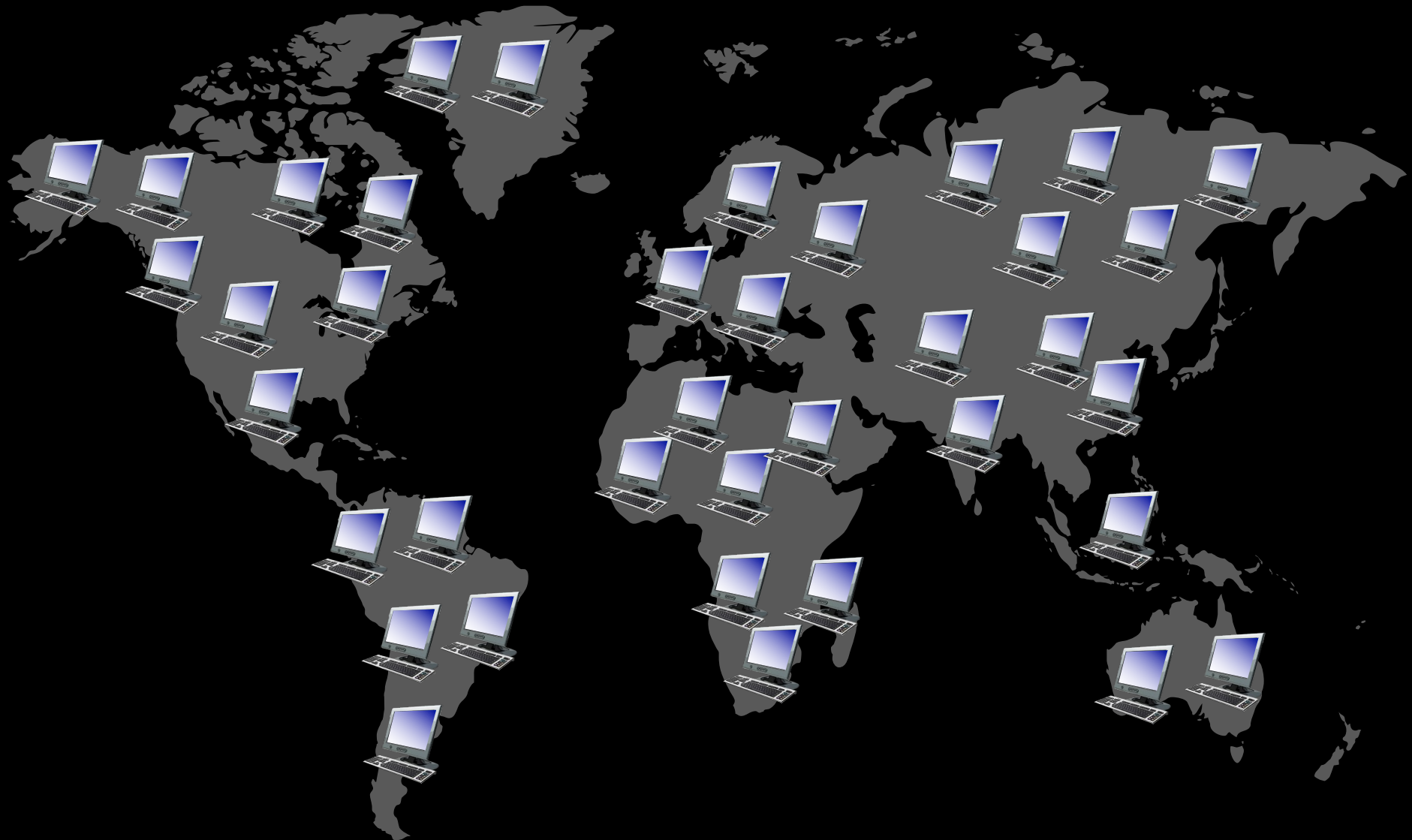
A: wireless WiFi interfaces connected by WiFi base station

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

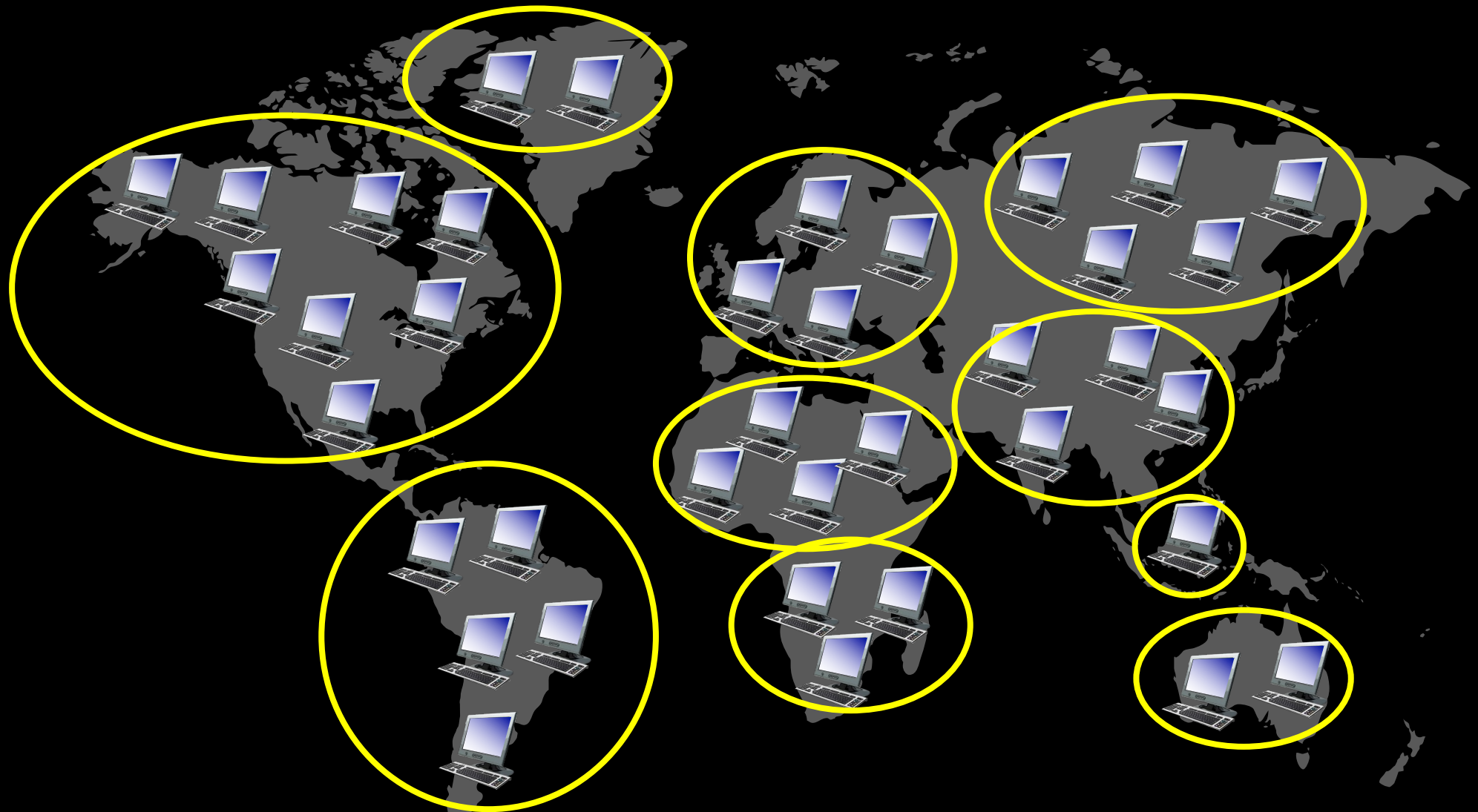
A Very Important Concept:
Subnets

Why we need Subnets?

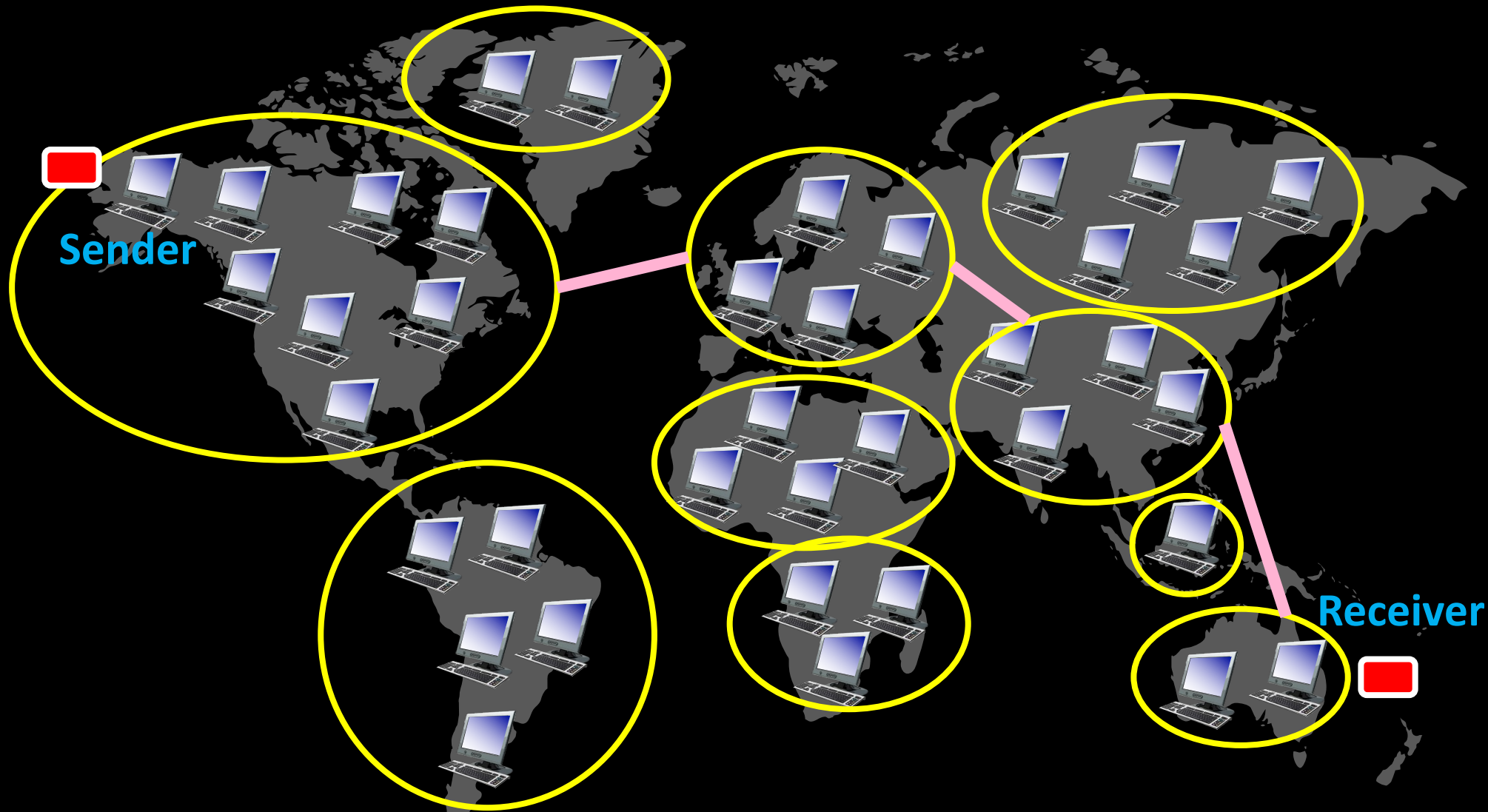
Subnets VS. Global Internet



Subnets VS. Global Internet



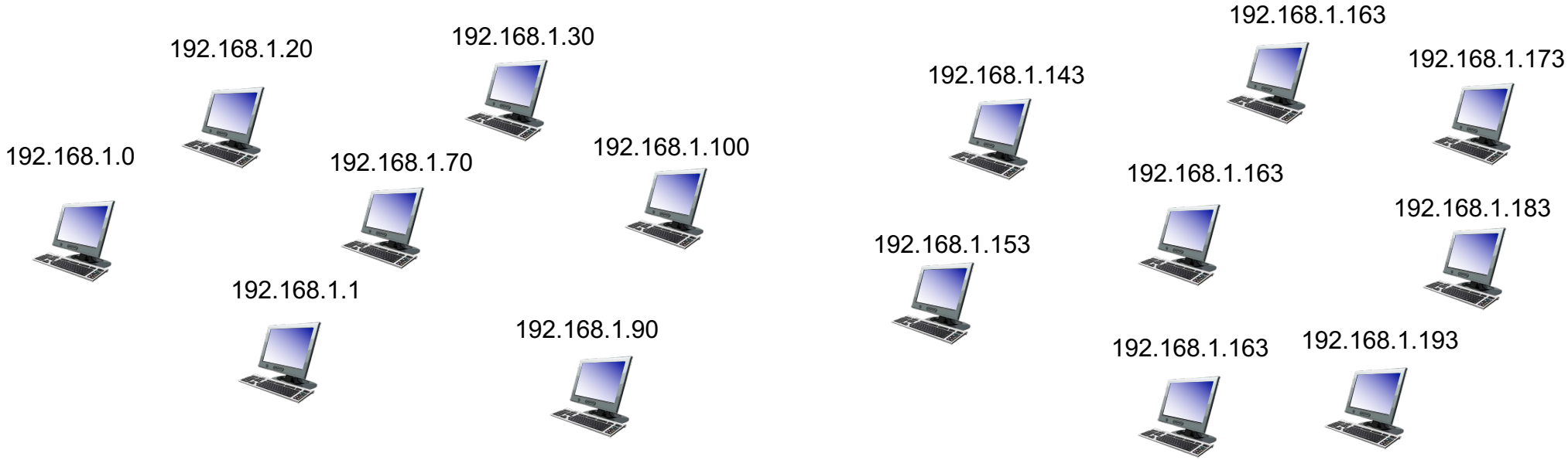
Subnets VS. Global Internet (Network of Networks)



Subnets VS. Global Internet (Network of Networks)

How should we organize/get the subnets?

Subnets

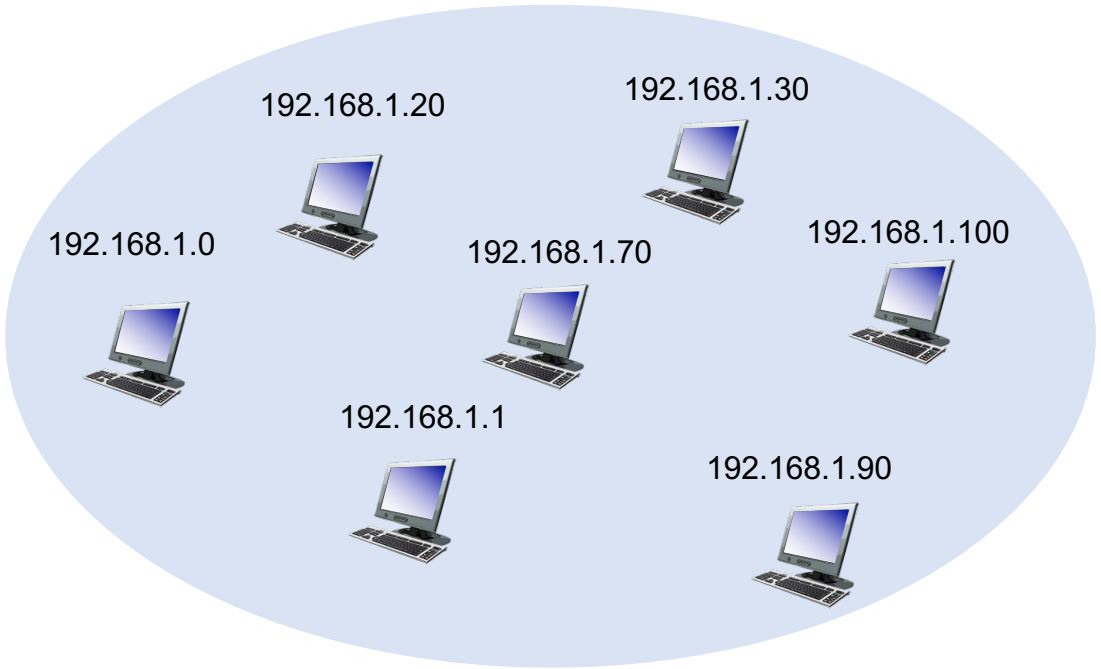


An IP network: 192.168.1.0 to 192.168.1.255

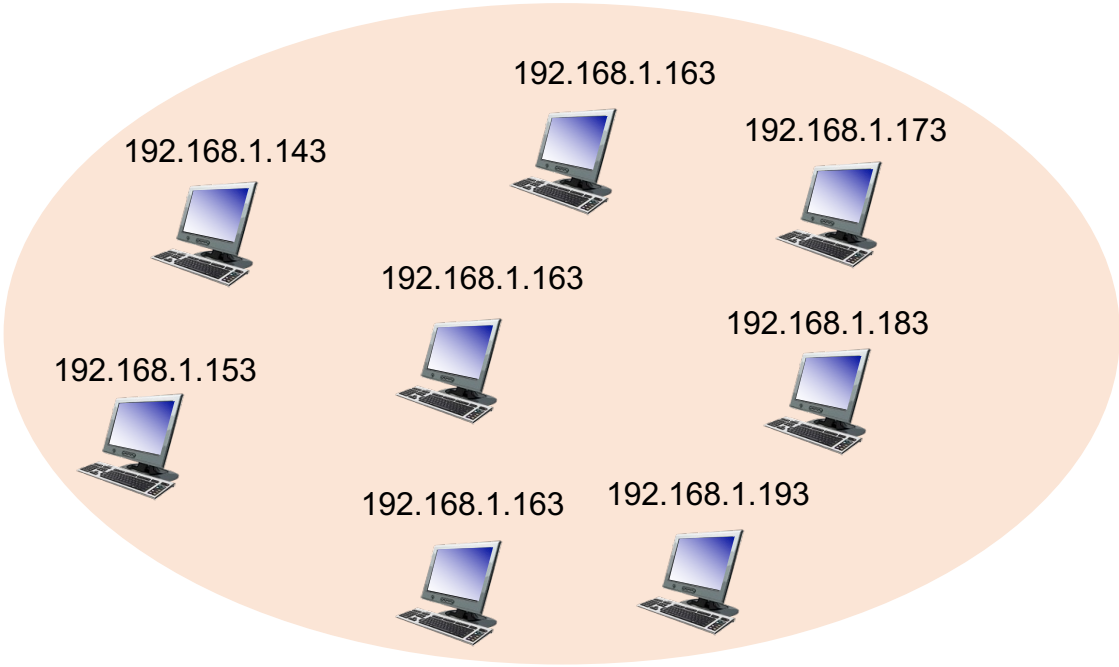
Subnets

- *What's a subnet ?*
  - Subnet is a logical subdivision of an IP network.

192.168.1.0 to 192.168.1.127



192.168.1.128 to 192.168.1.255



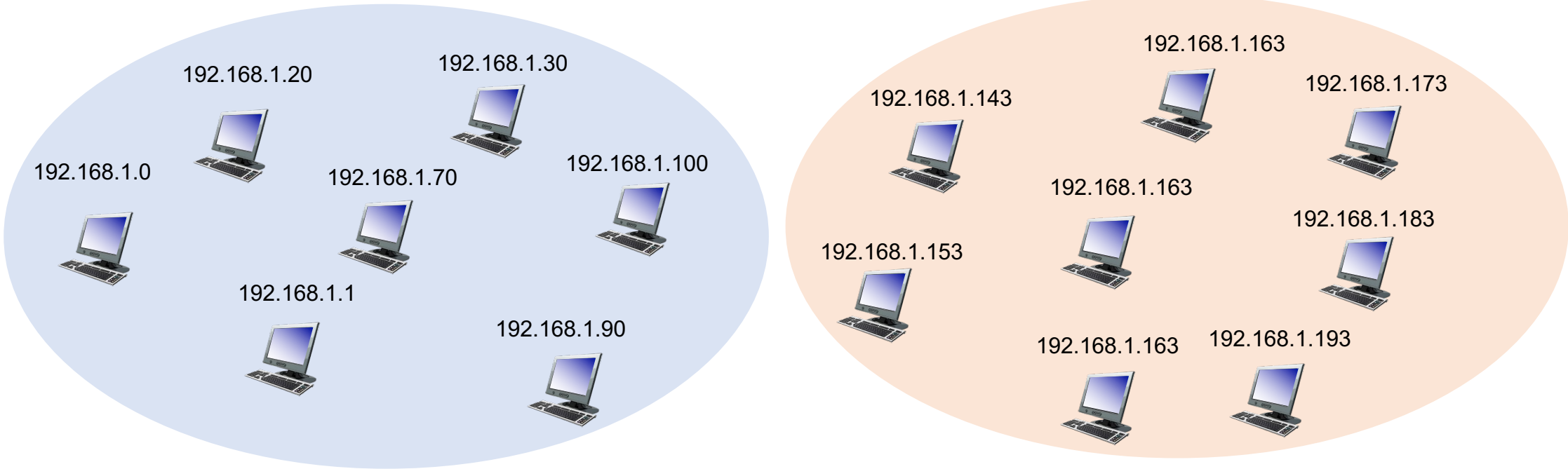
An IP network: 192.168.1.0 to 192.168.1.255

How should we represent the subnets?

Subnets Masks

192.168.1.0 to 192.168.1.127

192.168.1.128 to 192.168.1.255



An IP network: 192.168.1.0 to 192.168.1.255

Subnets Masks

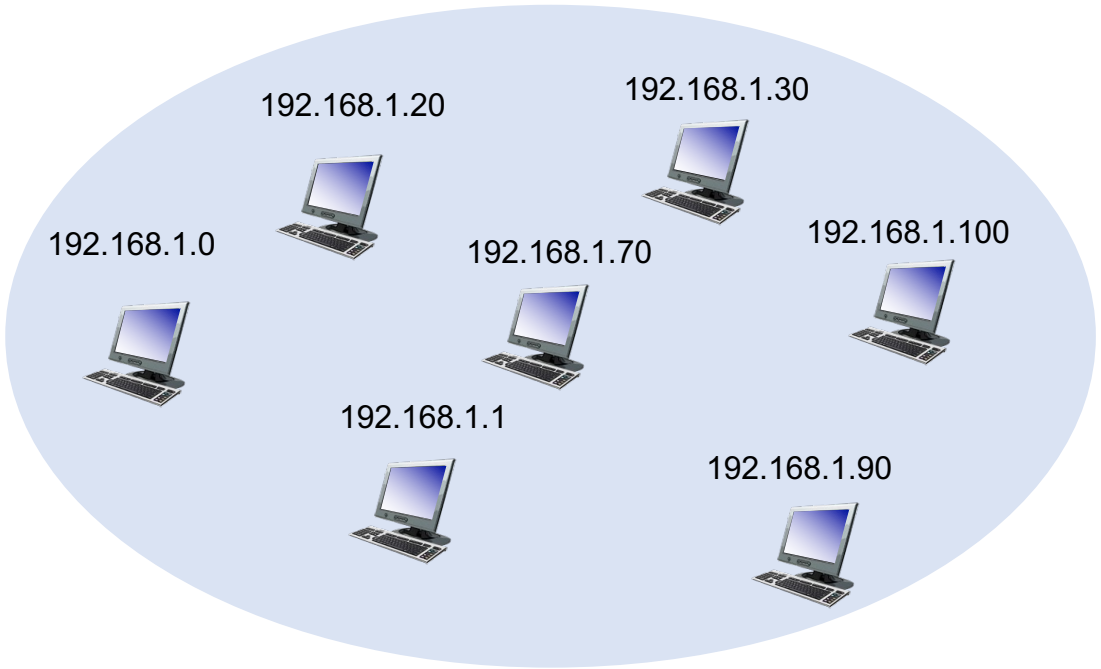
192.168.1.0

11000000. 10101000. 00000001. 00000000

192.168.1.127

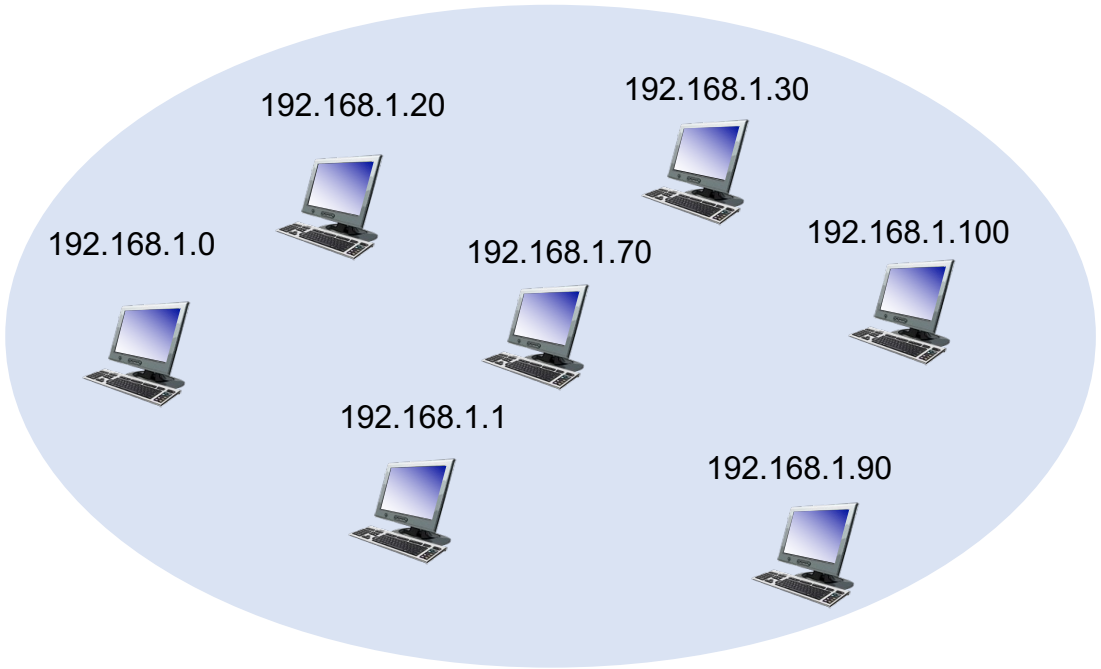
11000000. 10101000. 00000001. 01111111

192.168.1.0 to 192.168.1.127



Subnets Masks

192.168.1.0 to 192.168.1.127



Network Portion (fixed for a subnet)	Host Portion (varies with host)
11000000. 10101000. 00000001.	00000000
11000000. 10101000. 00000001.	01111111

A device with the following IP:

11000000. 10101000. 00000001.	00101001
11000000. 10101000. 00000001.	01011101
11000000. 10101000. 00000001.	01000100

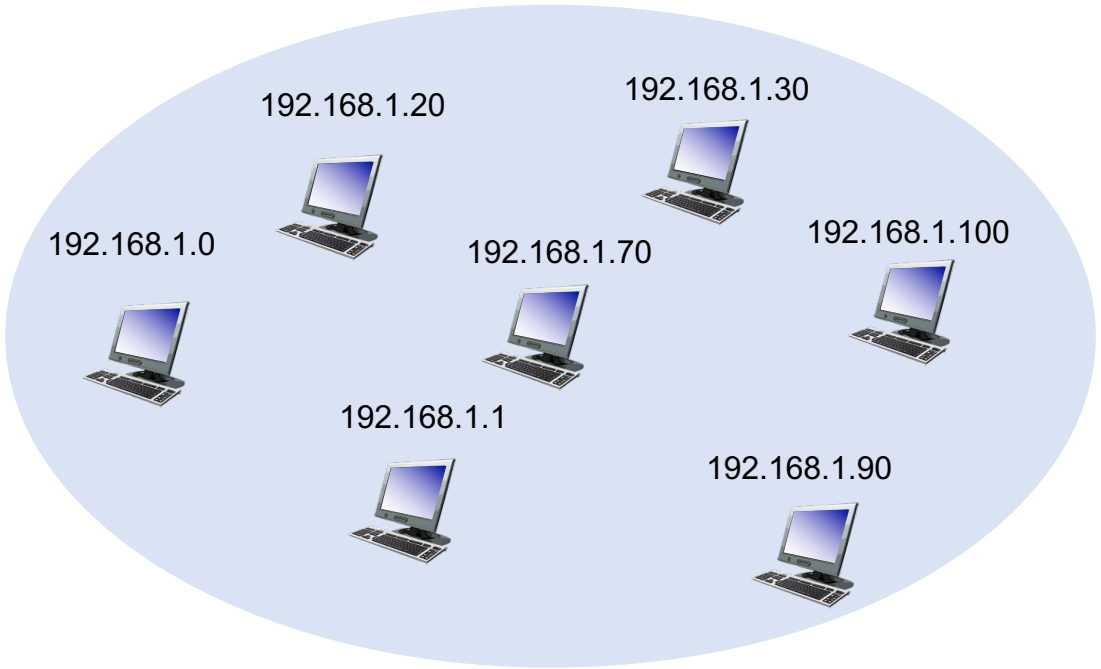
Subnets Masks

Network Portion (fixed for a subnet) | **Host Portion (varies with host)**

11000000. 10101000. 00000001. 00000000

11000000. 10101000. 00000001. 01111111

192.168.1.0 to 192.168.1.127



11111111. 11111111. 11111111. 1

Network Portion | **Host Portion**

The **sequence of 1s** in the subnet mask indicates which bits of the IP address belong to the **network portion**

The **sequence of 0s** indicates which bits belong to the **host portion**.

Subnets Masks

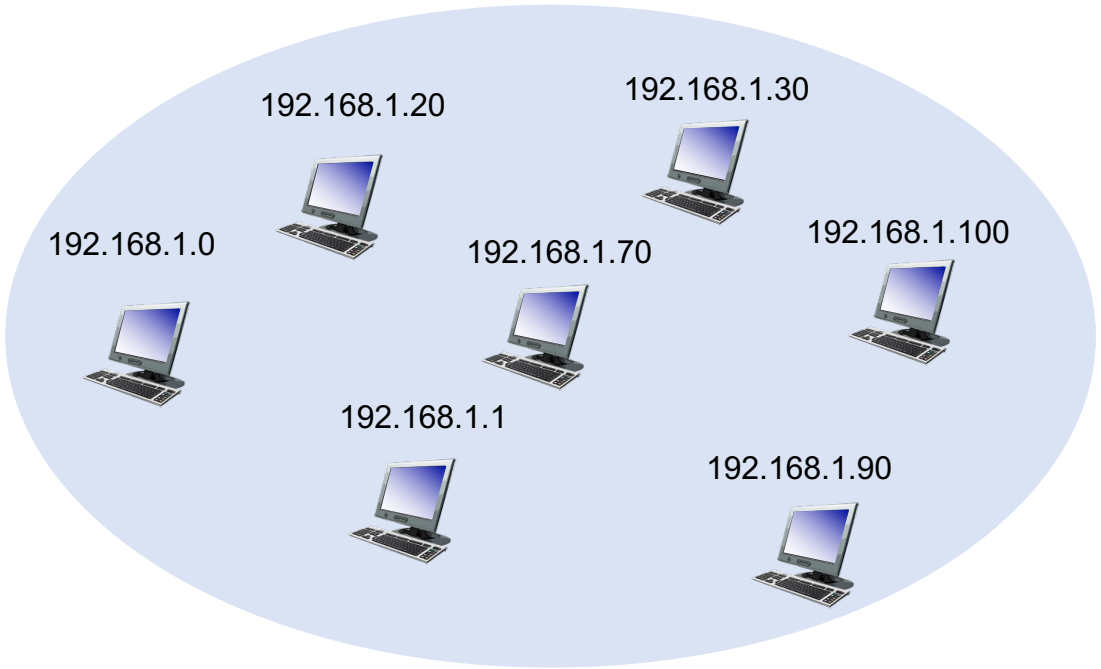
Network Portion
(fixed for a subnet)

Host Portion
(varies with host)

11000000. 10101000. 00000001. 00000000

11000000. 10101000. 00000001. 01111111

192.168.1.0 to 192.168.1.127



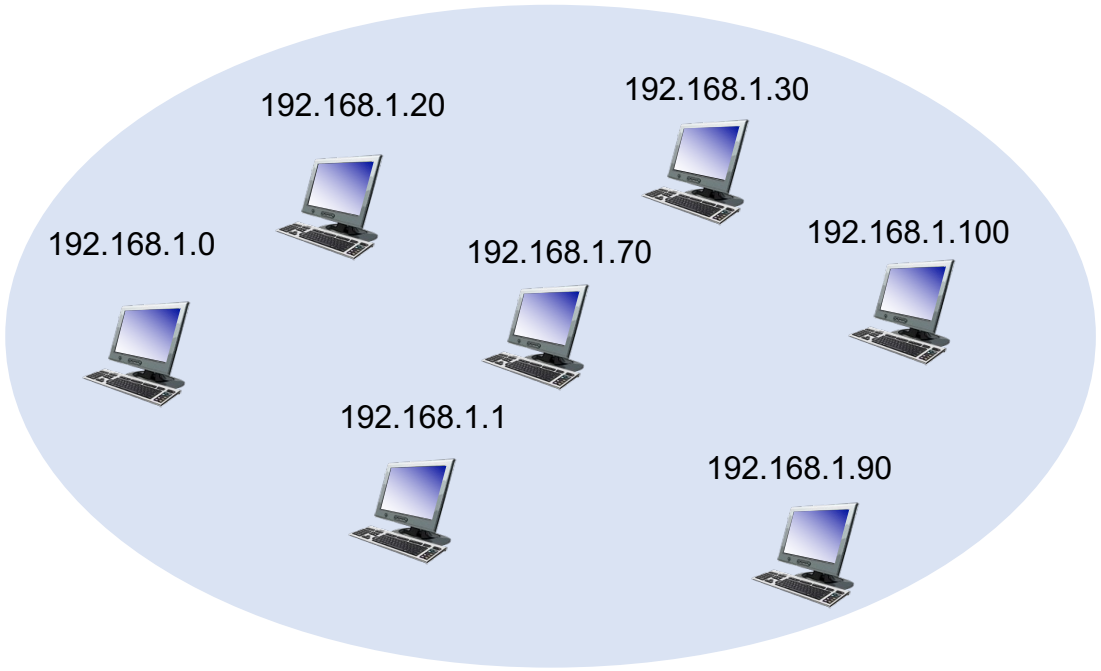
11111111. 11111111. 11111111. 10000000

Network Portion **Host Portion**

Subnet Mask: 255. 255. 255. 128

Subnets Masks

192.168.1.0 to 192.168.1.127



Network Portion (fixed for a subnet)

Host Portion (varies with host)

11000000. 10101000. 00000001. 00000000

11111111. 11111111. 11111111. 10000000

Network Portion

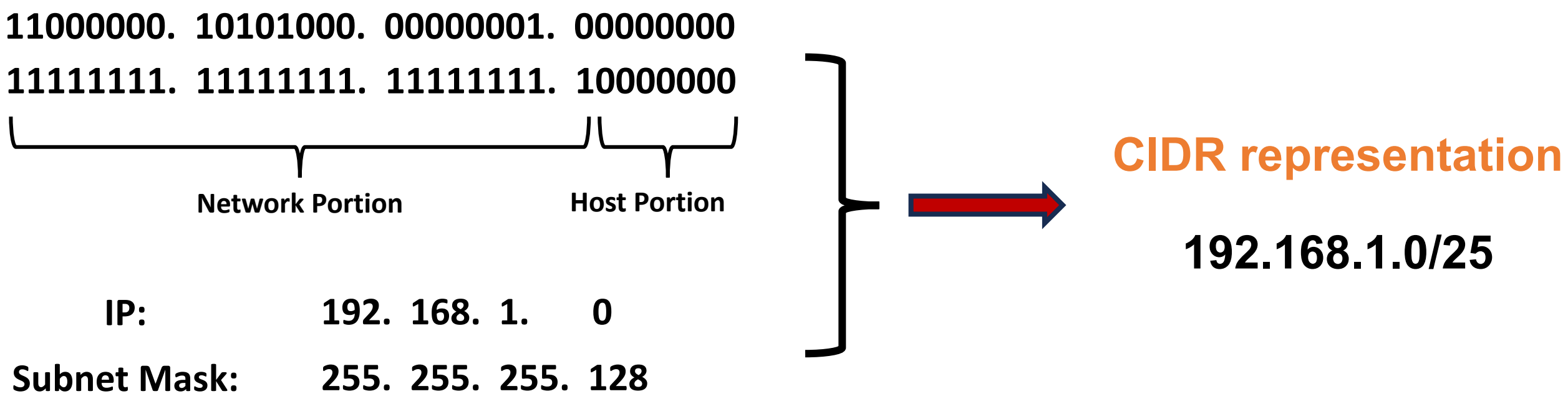
Host Portion

IP:	192. 168. 1. 0
Subnet Mask:	255. 255. 255. 128

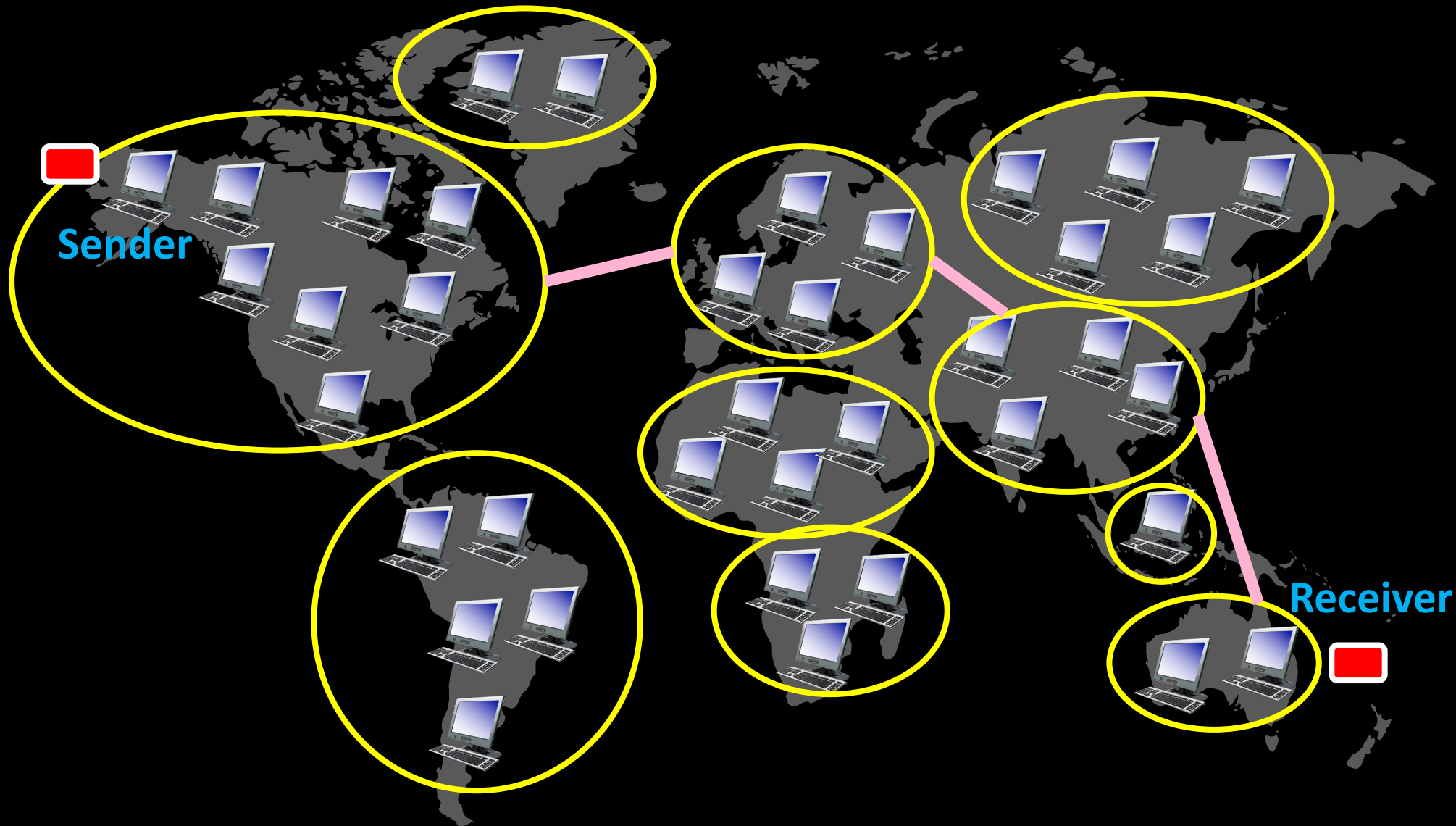
Subnet and CIDR

CIDR: Classless InterDomain Routing (pronounced “cider”)

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

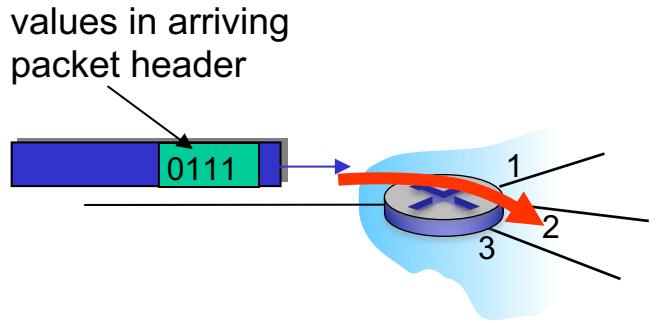


Subnets VS. Global Internet (Network of Networks)



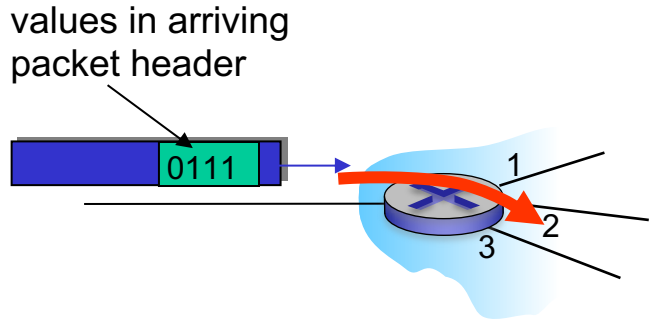
Forwarding Table

IP Address Range		Forwarding Interface
192.168.0.1	192.168.0.20	**1**
192.168.0.40	192.168.0.60	**2**
192.168.0.80	192.168.0.100	**3**



Forwarding Table

IP Address Range	Forwarding Interface
192.168.1.0/25	**1**
192.168.0.0/24	**2**
127.1.0.80/26	**3**



Any digits

11000000. 10101000. 00000001. 00000000
11111111. 11111111. 11111111. 10000000

11000000. 10101000. 00000001. 0*******

Longest prefix matching

longest prefix match

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

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

examples:

11001000 00010111 00010110 10100001 which interface?

11001000 00010111 00011000 10101010 which interface?

Longest prefix matching

longest prefix match

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

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

match!

examples:

11001000 00010111 00010110 10100001 which interface?

11001000 00010111 00011000 10101010 which interface?

Longest prefix matching

longest prefix match

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

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

match!

examples:

11001000 00010111 00010110 10100001 which interface?

11001000 00010111 00011000 10101010 which interface?

Longest prefix matching

longest prefix match

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

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

match!

examples:

11001000 00010111 00010110 10100001 which interface?

11001000 00010111 00011000 10101010 which interface?

Longest prefix matching

longest prefix match

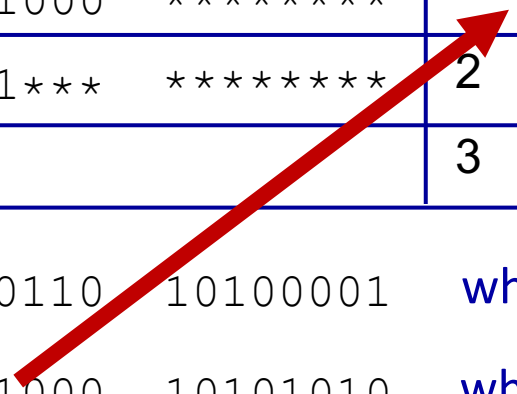
when looking for forwarding table entry for given destination address, use *longest* address prefix that matches destination address.

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

- we'll see *why* longest prefix matching is used shortly, when we study addressing
- longest prefix matching: often performed using ternary content addressable memories (TCAMs)
  - *content addressable:* present address to TCAM: retrieve address in one clock cycle, regardless of table size
  - Cisco Catalyst: ~1M routing table entries in TCAM