

Chapter 4

Network Layer: Data Plane

A note on the use of these PowerPoint slides:

We're making these slides freely available to all (faculty, students, readers). They're in PowerPoint form so you see the animations; and can add, modify, and delete slides (including this one) and slide content to suit your needs. They obviously represent a *lot* of work on our part.

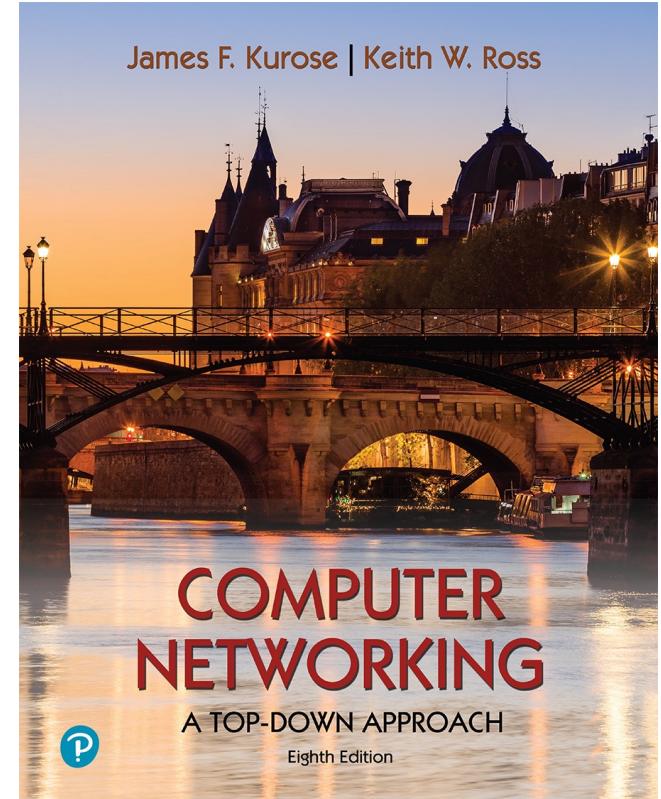
In return for use, we only ask the following:

- If you use these slides (e.g., in a class) that you mention their source (after all, we'd like people to use our book!)
- If you post any slides on a www site, that you note that they are adapted from (or perhaps identical to) our slides, and note our copyright of this material.

For a revision history, see the slide note for this page.

Thanks and enjoy! JFK/KWR

All material copyright 1996-2020
J.F Kurose and K.W. Ross, All Rights Reserved



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

Objective

Build a *system* of local area networks to transfer data from one node to a node, that is not (necessarily) physically connected

A really large system ...

1. Service Modell
2. Forwarding
3. Scalability
4. Addressing

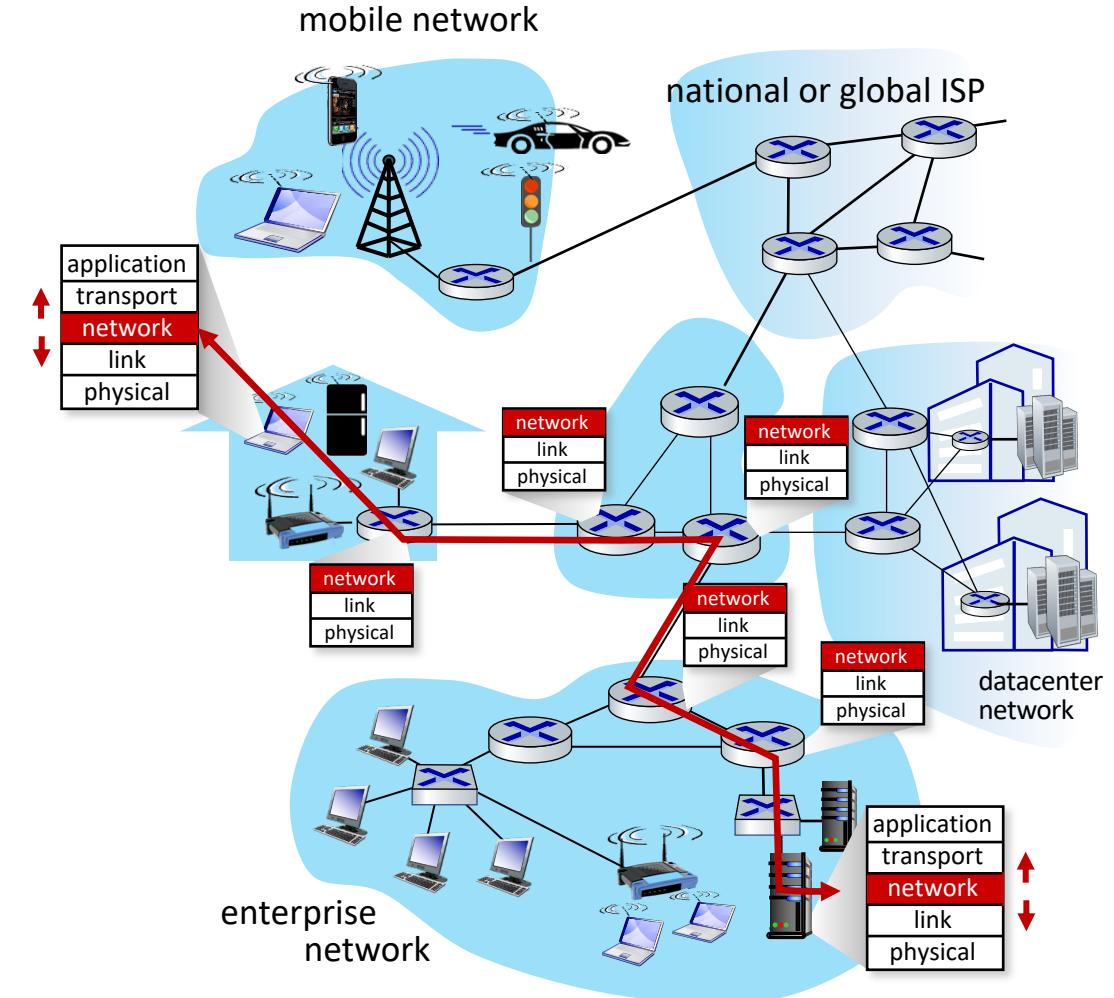
Network layer: “data plane” roadmap

- Network layer: overview
 - data plane
 - control plane
- Network layer service model
- What's inside a router
 - input ports, switching, output ports
 - buffer management, scheduling
- Forwarding of datagrams
- IP: the Internet Protocol
 - datagram format
 - addressing



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

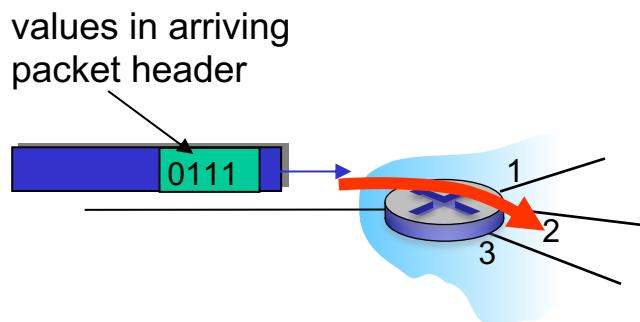


routing

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

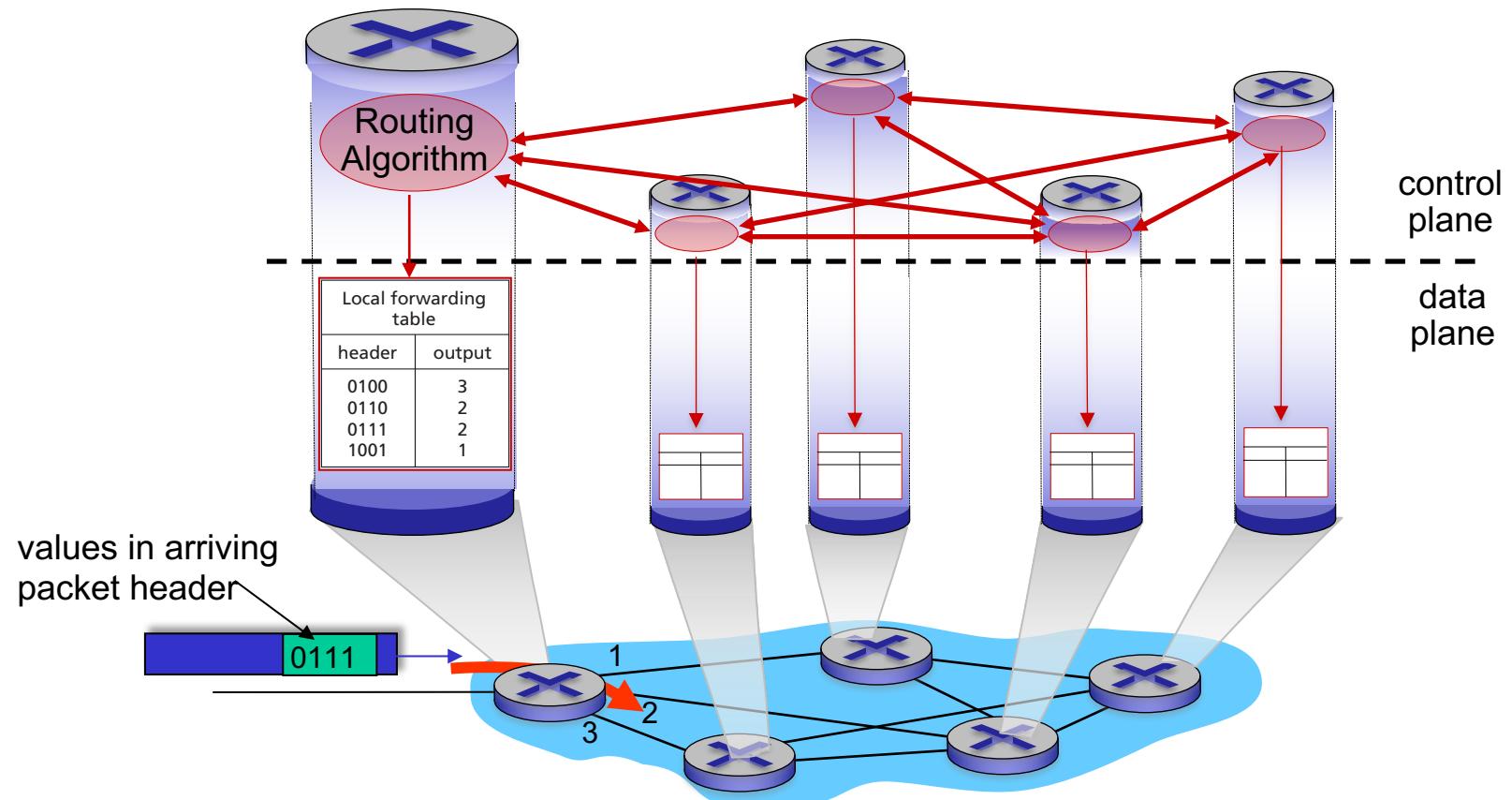


Control plane

- *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

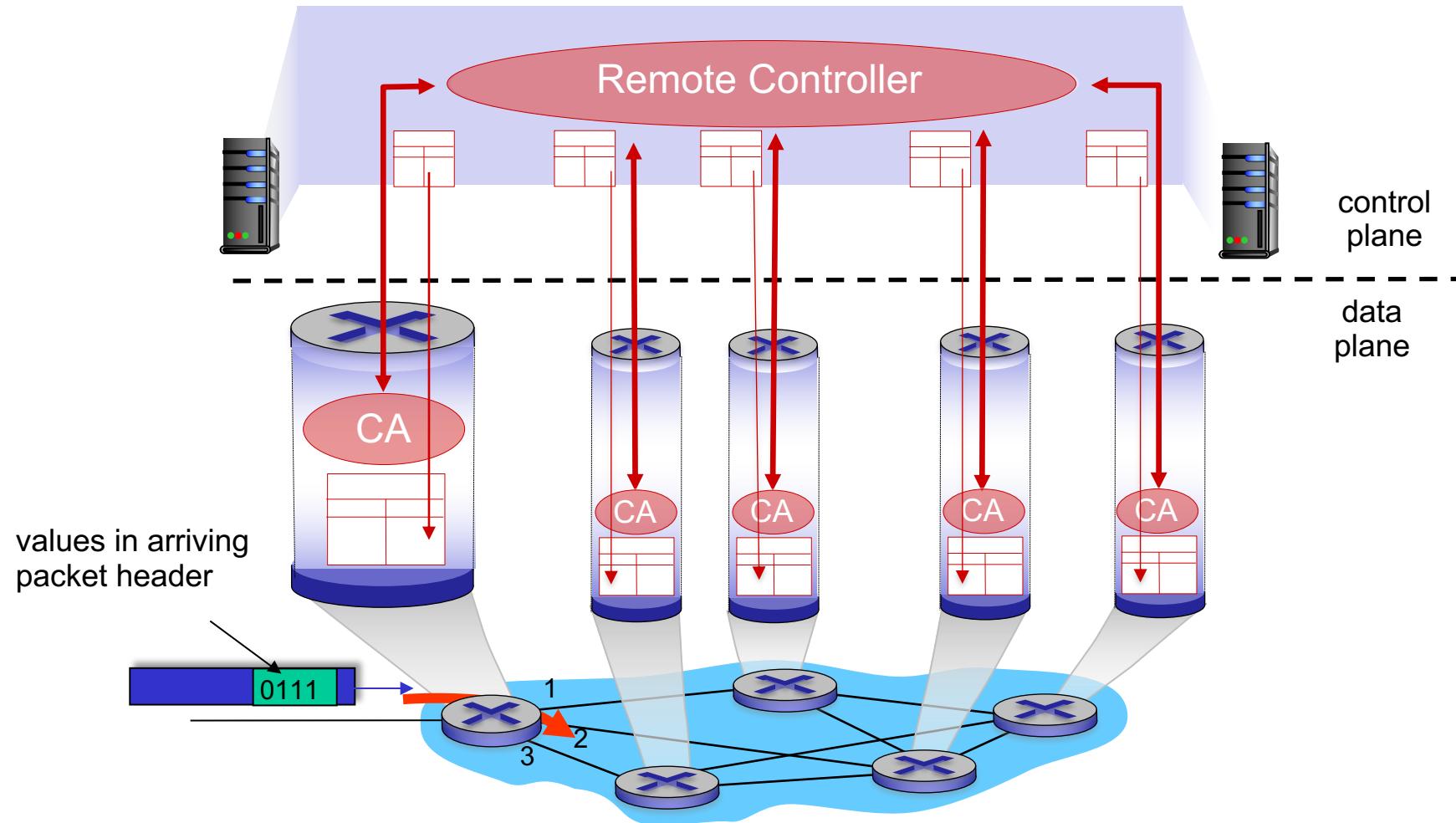
Per-router control plane

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 layer: “data plane” roadmap

- Network layer: overview
 - data plane
 - control plane
- Network layer service model
- What's inside a router
 - input ports, switching, output ports
 - buffer management, scheduling
- Forwarding of datagrams
- IP: the Internet Protocol
 - datagram format
 - addressing



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

Network Architecture	Service Model	Quality of Service (QoS) Guarantees ?			
		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

Network Architecture	Service Model	Quality of Service (QoS) Guarantees ?			
		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

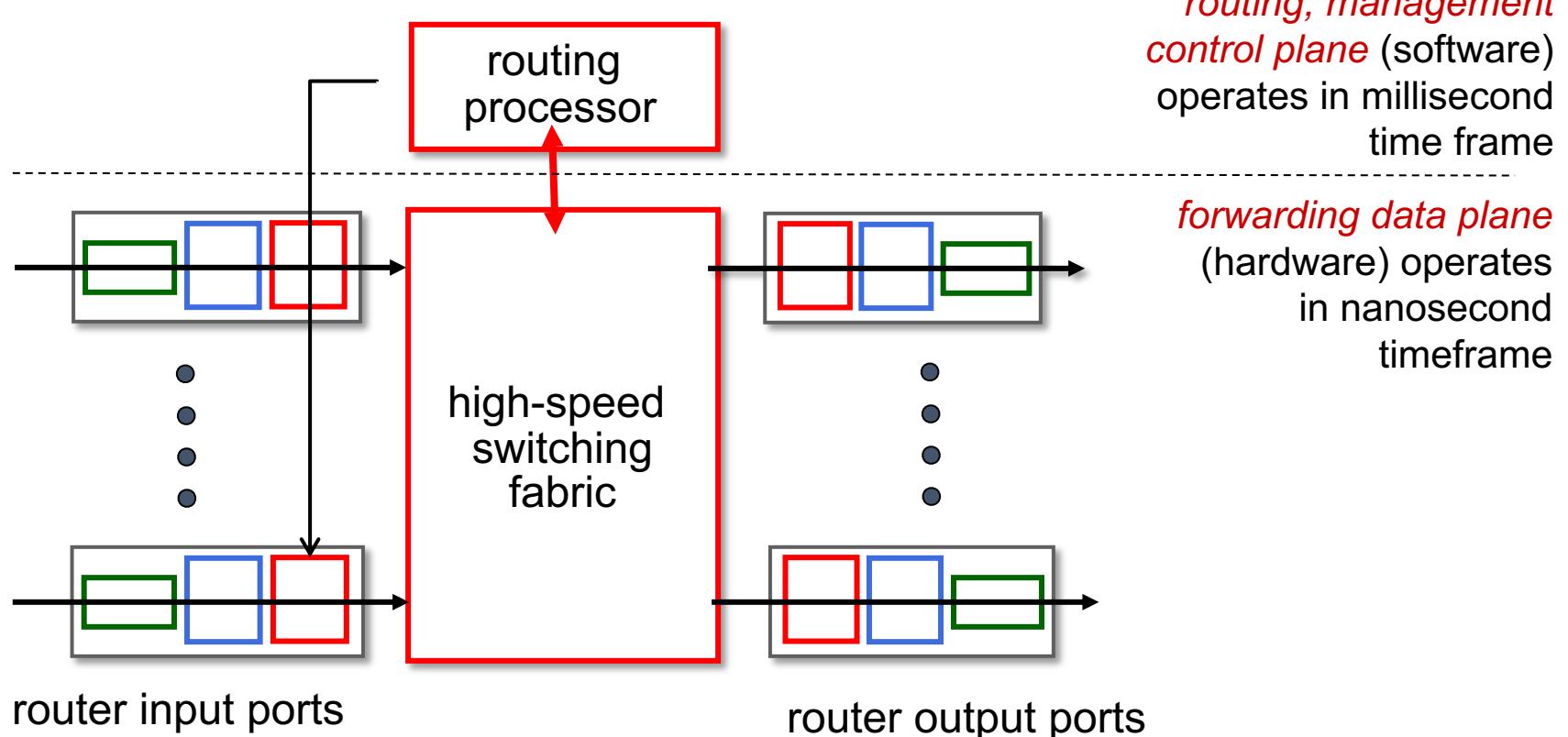
Network layer: “data plane” roadmap

- Network layer: overview
 - data plane
 - control plane
- Network layer service model
- What's inside a router
 - input ports, switching, output ports
 - buffer management, scheduling
- Forwarding of datagrams
- IP: the Internet Protocol
 - datagram format
 - addressing

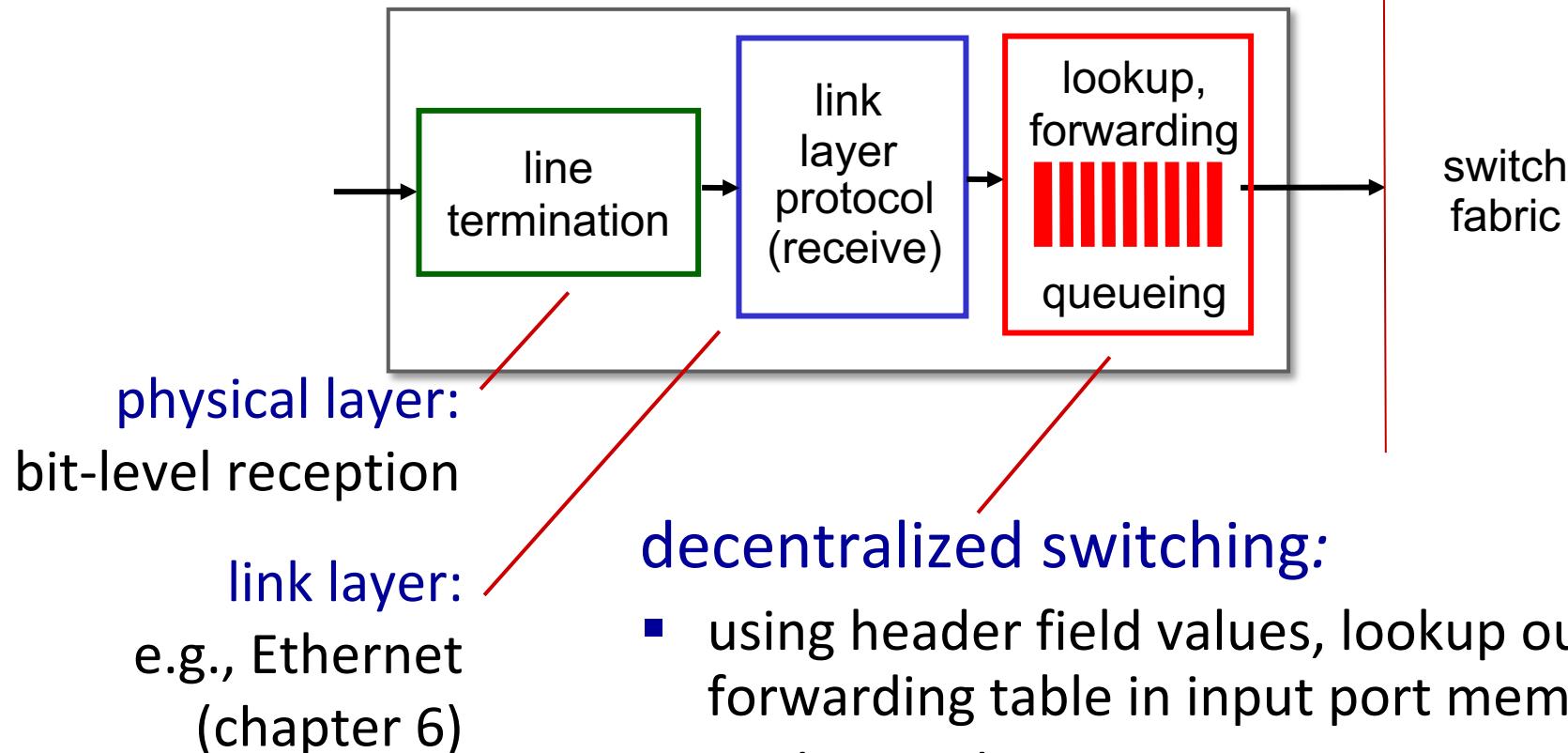


Router architecture overview

high-level view of generic router architecture:



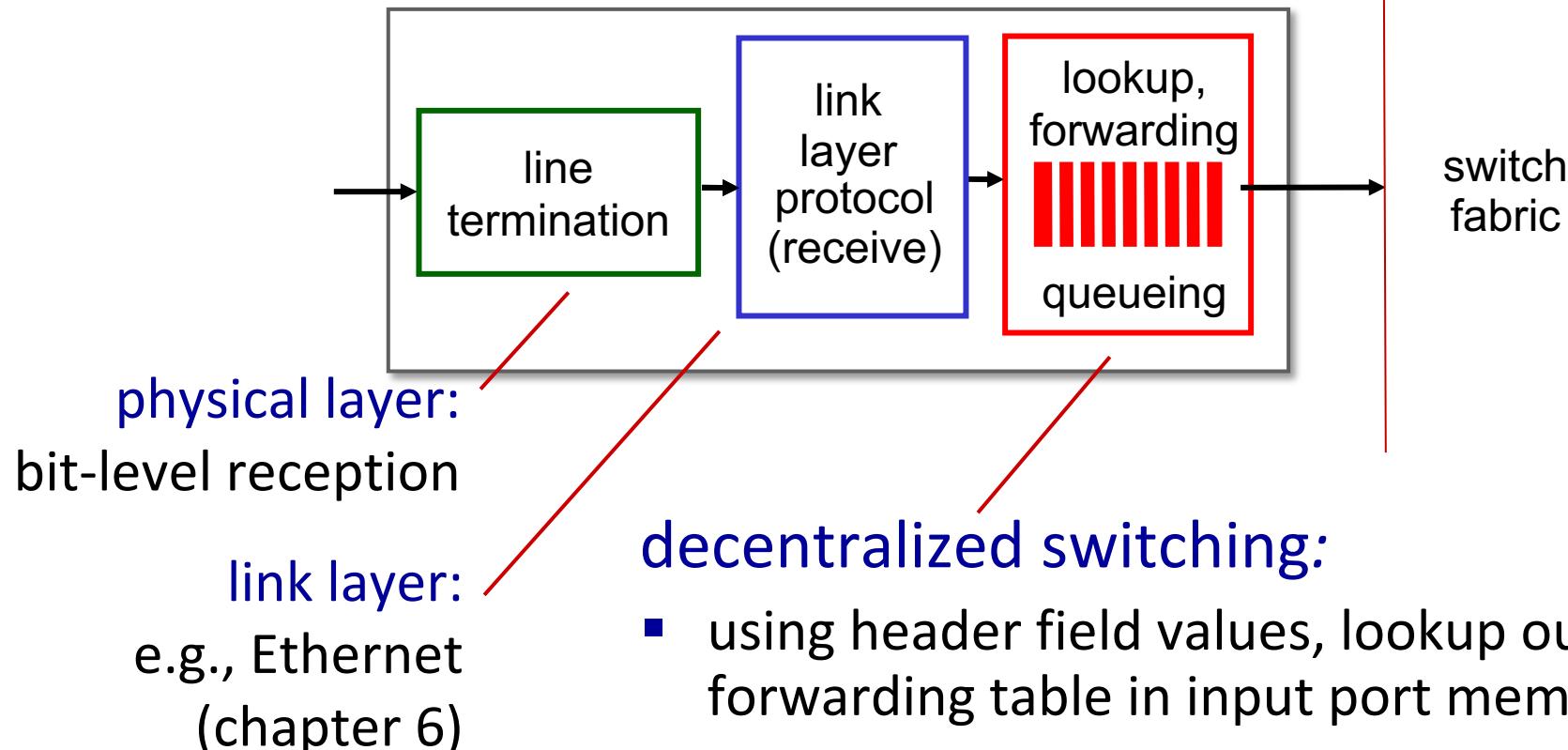
Input port functions



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

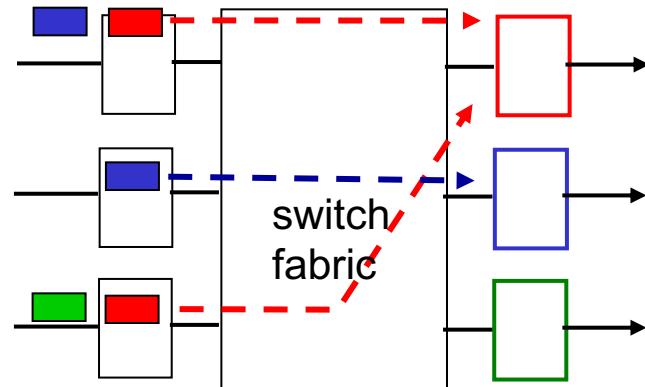


decentralized switching:

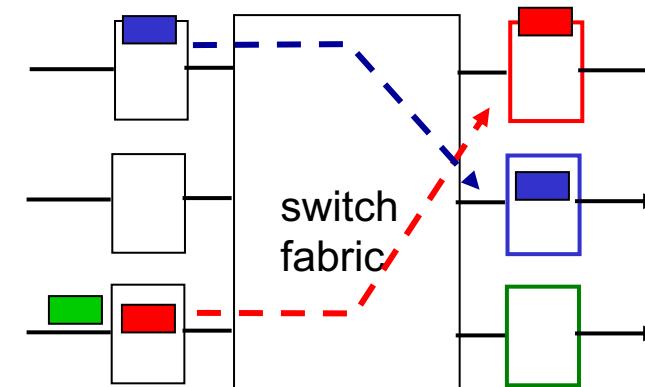
- using header field values, lookup output port using forwarding table in input port memory ("*match plus action*")
- **destination-based forwarding**: forward based only on destination IP address (traditional)
- **generalized forwarding**: forward based on any set of header field values

Input port queuing

- If switch 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



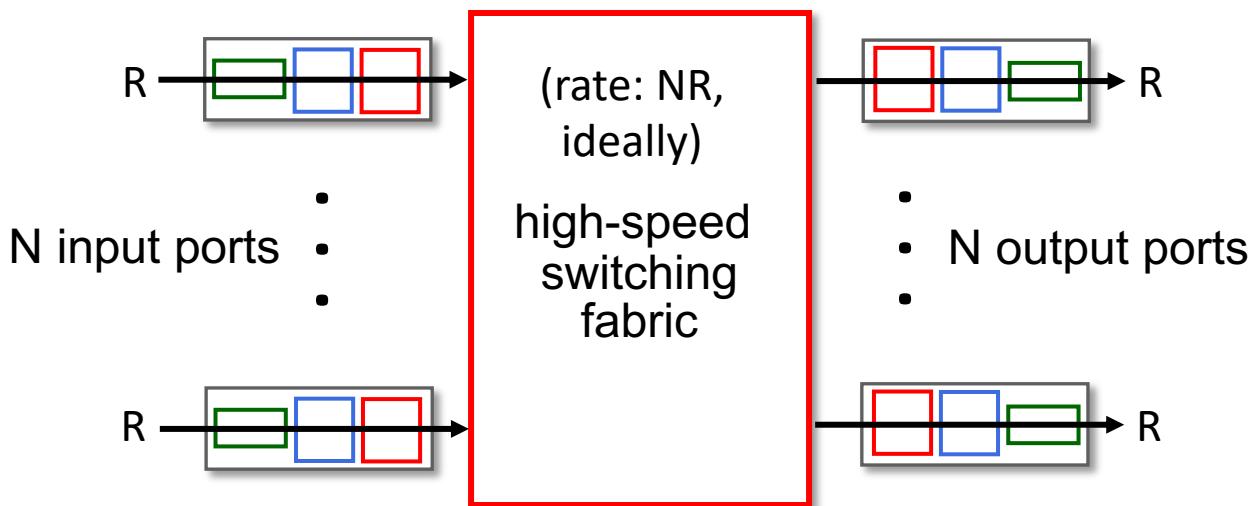
output port contention: only one red datagram can be transferred. lower red packet is *blocked*



one packet time later: green packet experiences HOL blocking

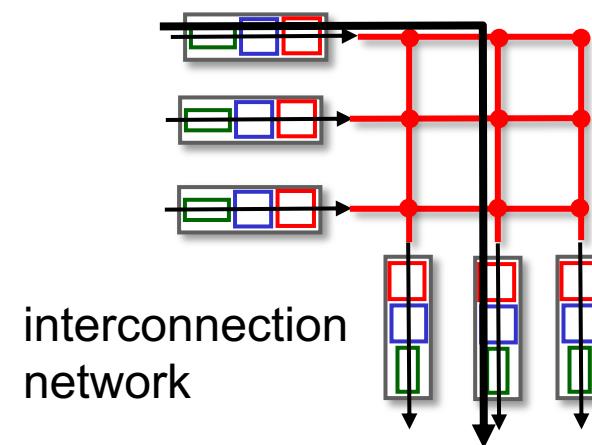
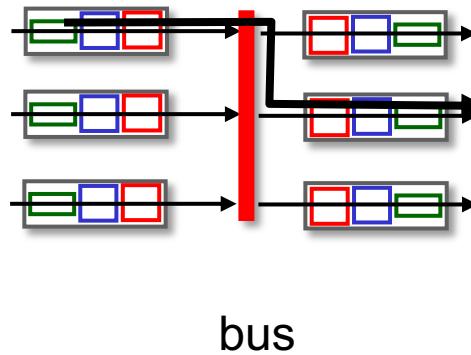
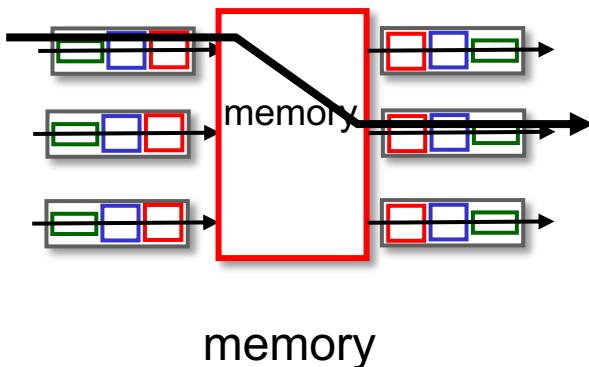
Switching fabrics

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



Switching fabrics

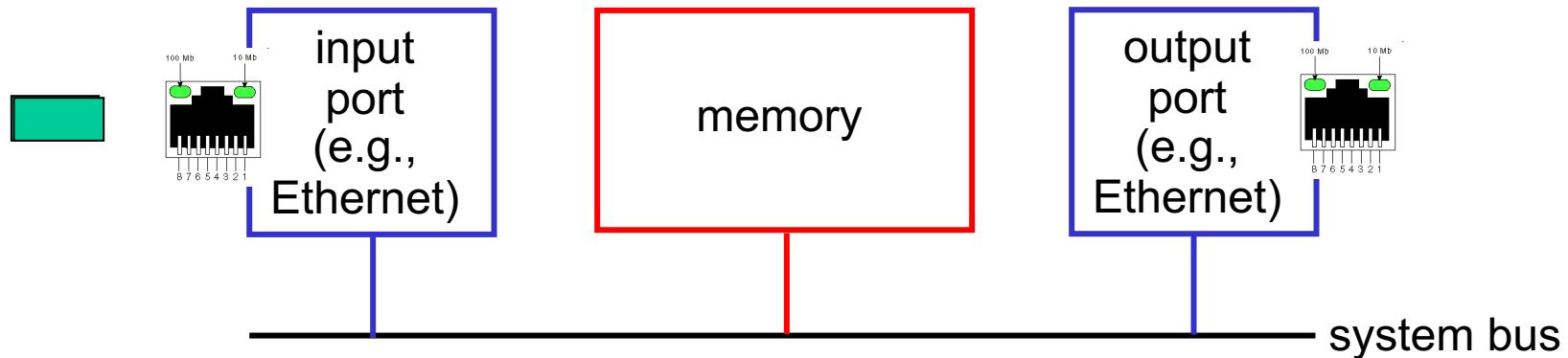
- transfer packet from input link to appropriate output link
- **switching rate:** rate at which packets can be transferred from inputs to outputs
 - often measured as multiple of input/output line rate
 - N inputs: switching rate N times line rate desirable
- three major types of switching fabrics:



Switching via memory

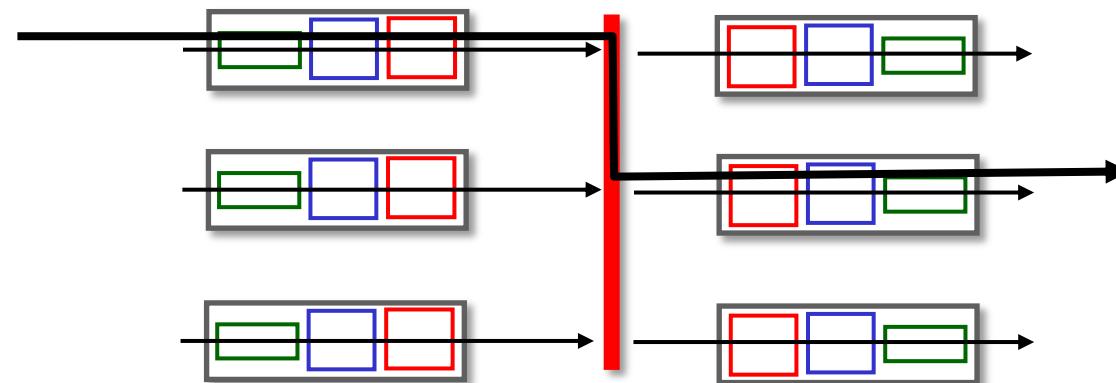
first generation routers:

- 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)



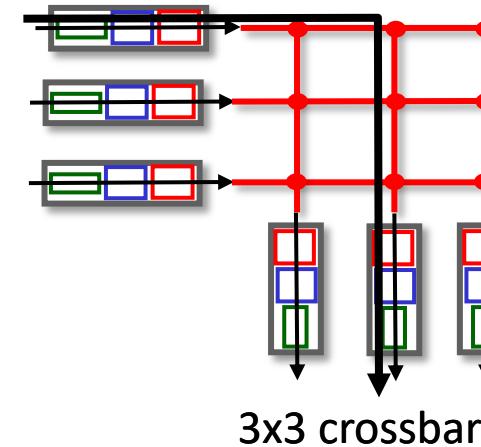
Switching via a bus

- 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

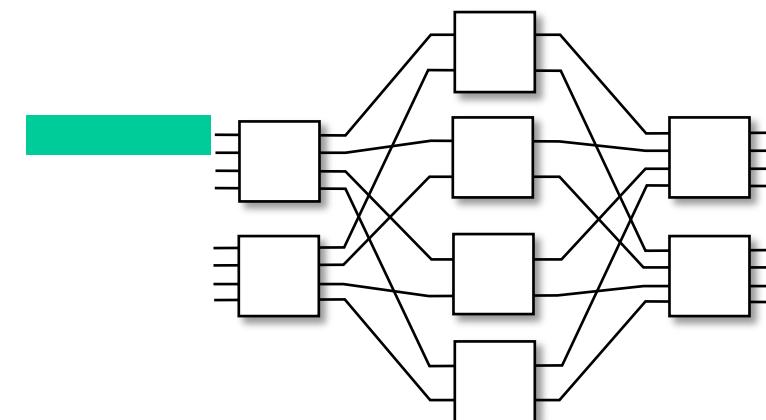


Switching via interconnection network

- 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



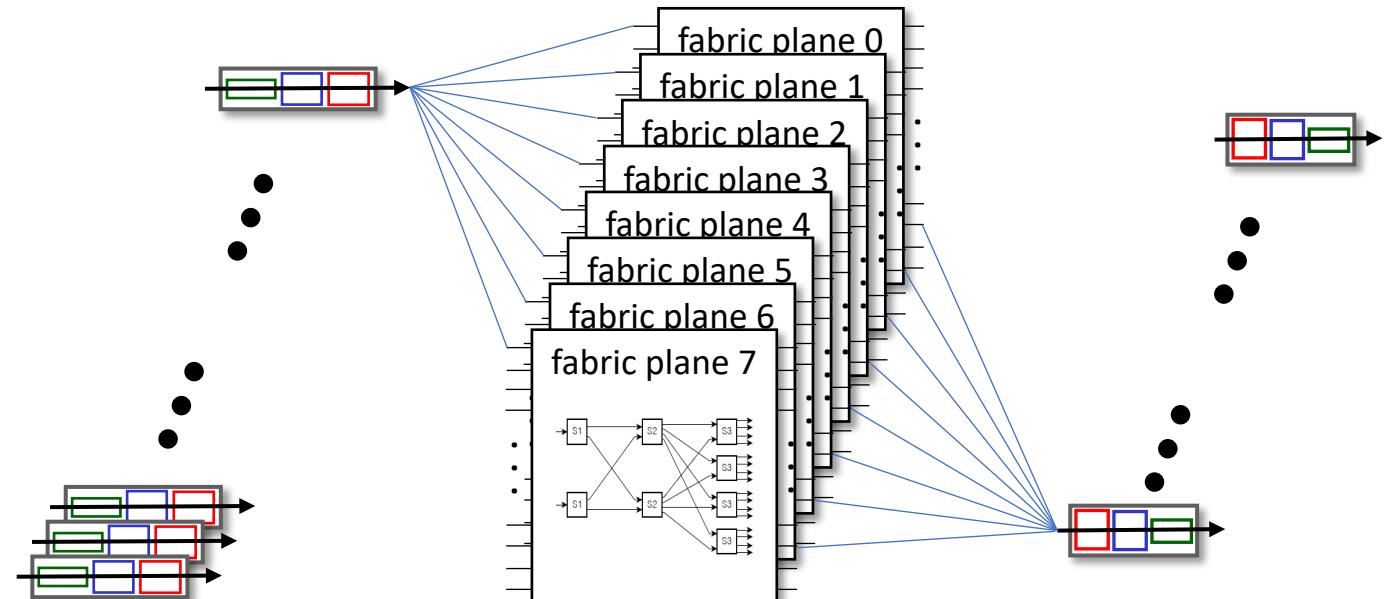
3x3 crossbar



8x8 multistage switch
built from smaller-sized switches

Switching via interconnection network

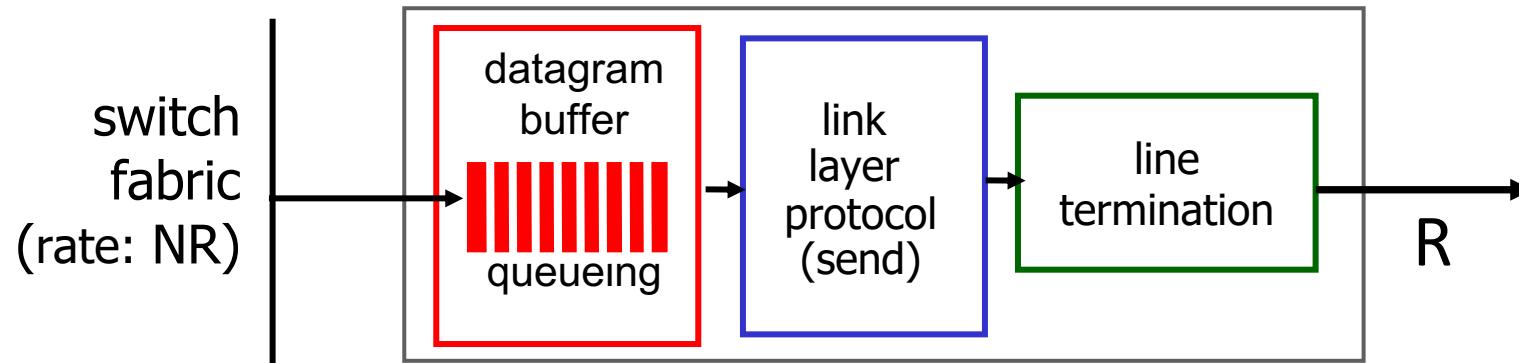
- scaling, using multiple switching “planes” in parallel:
 - speedup, scaleup via parallelism
- Cisco CRS router:
 - basic unit: 8 switching planes
 - each plane: 3-stage interconnection network
 - up to 100's Tbps switching capacity



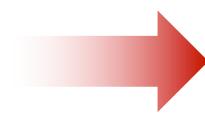
Output port queuing



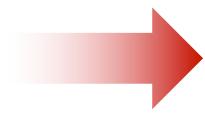
This is a really important slide



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

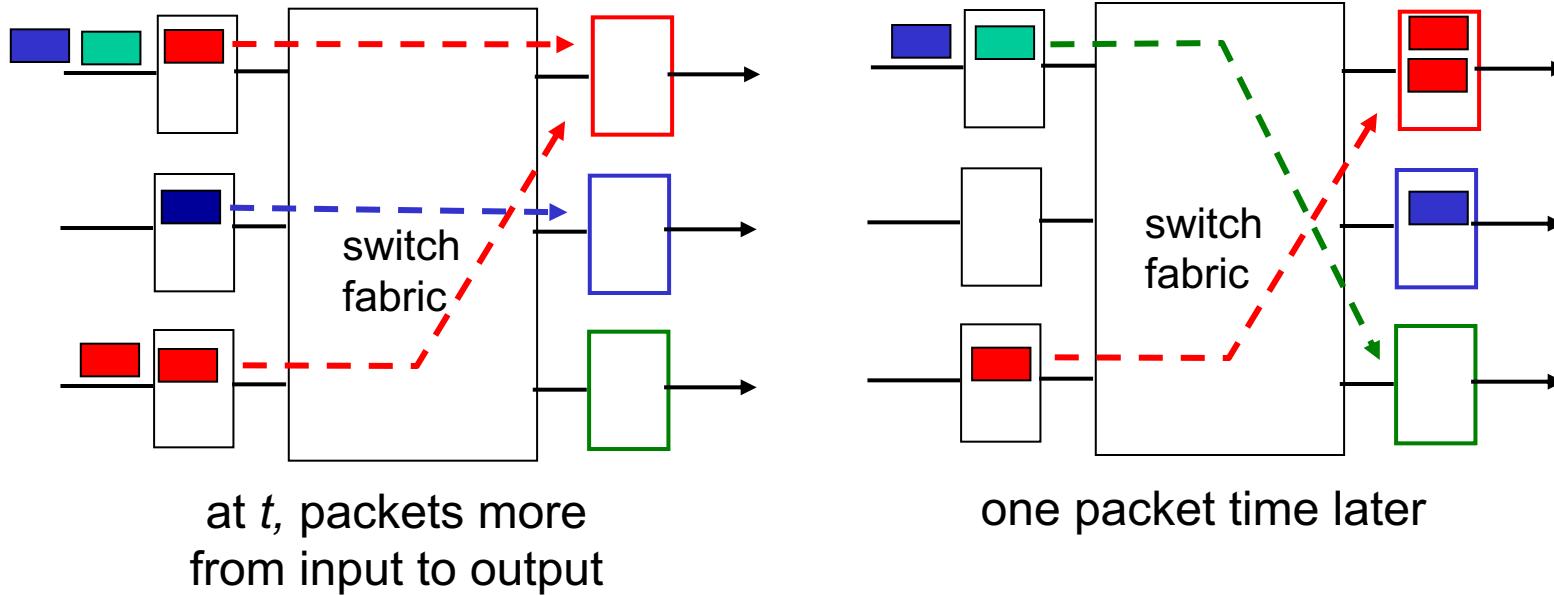


Datagrams can be lost due to congestion, lack of buffers



Priority scheduling – who gets best performance, network neutrality

Output port queuing



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

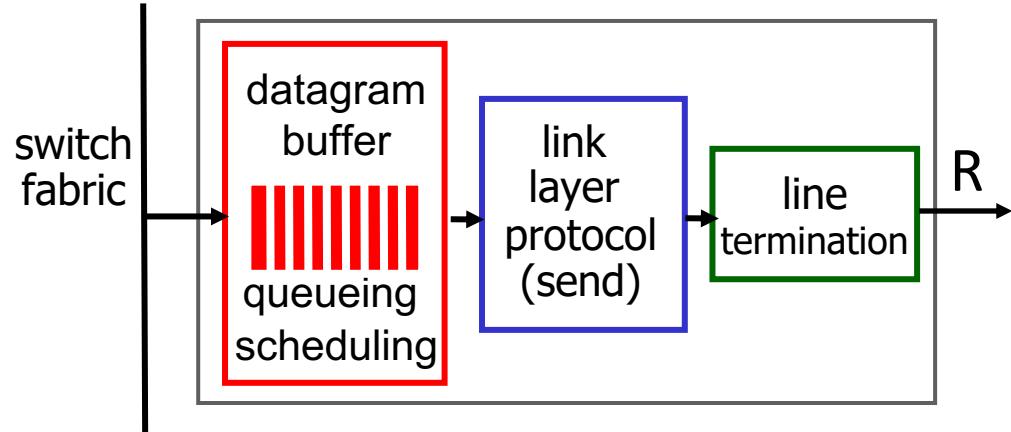
How much buffering?

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

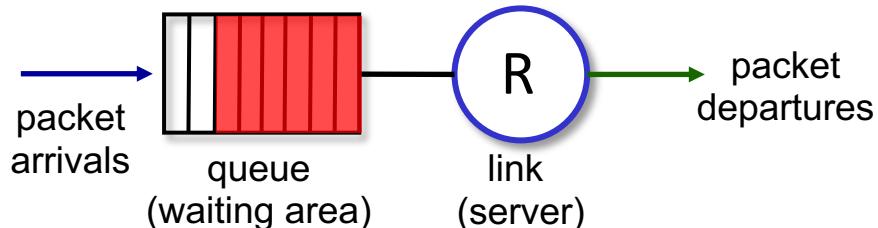
$$\frac{\text{RTT} \cdot C}{\sqrt{N}}$$

- but *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



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: FCFS

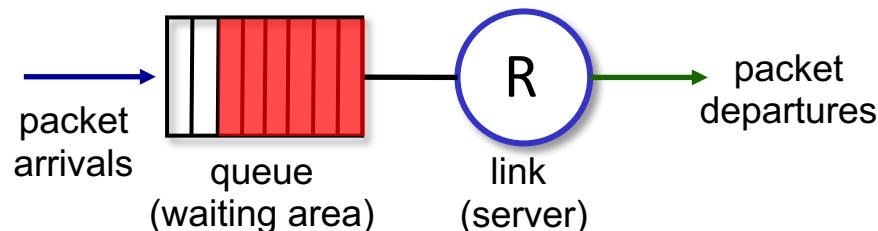
packet scheduling: deciding which packet to send next on link

- 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?

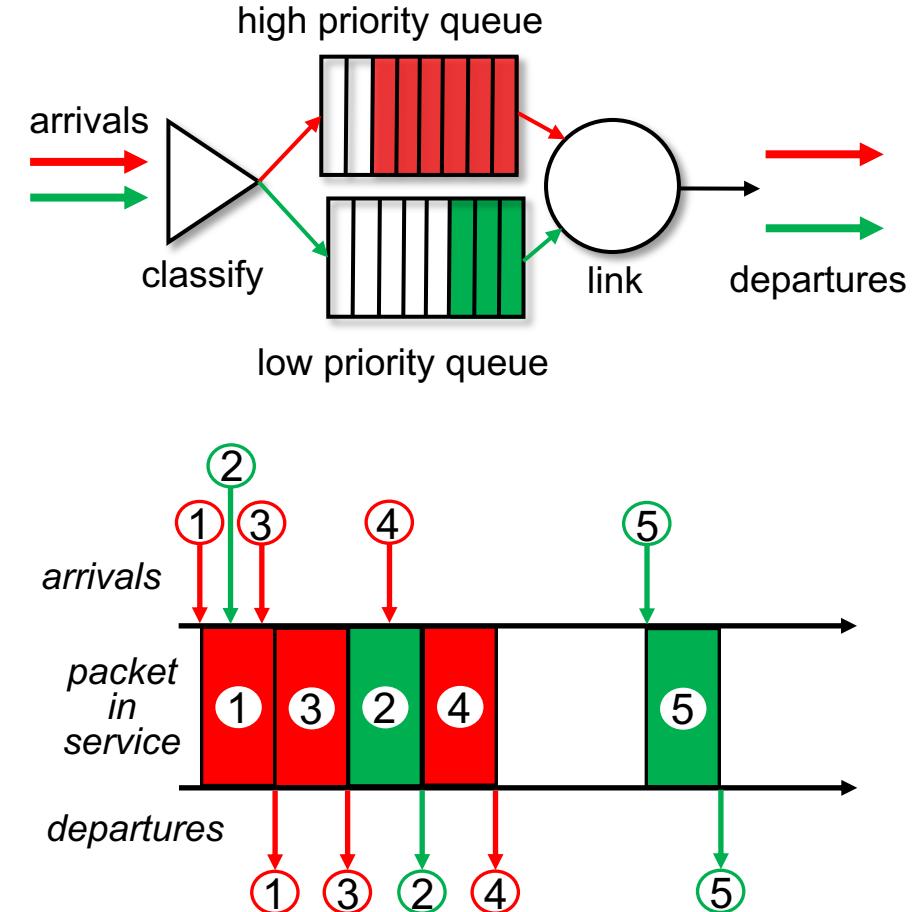
Abstraction: queue



Scheduling policies: priority

Priority scheduling:

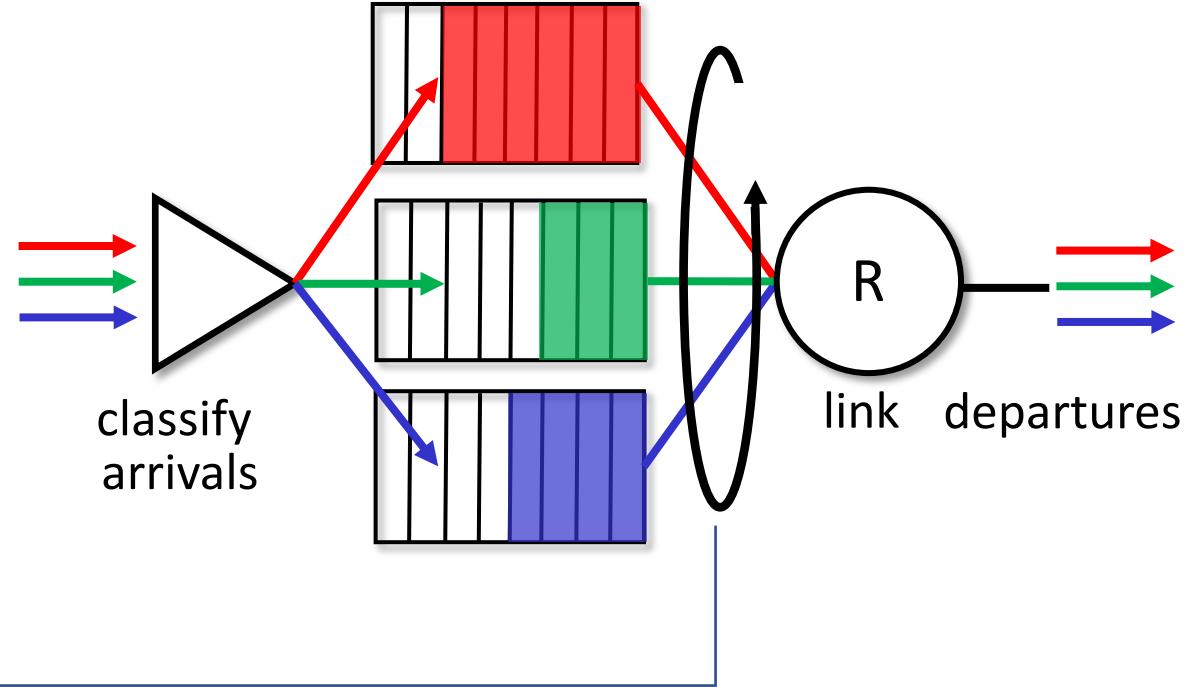
- arriving traffic classified, queued by class
 - any header fields can be used for classification
- send packet from highest priority queue that has buffered packets
 - FCFS within priority class



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



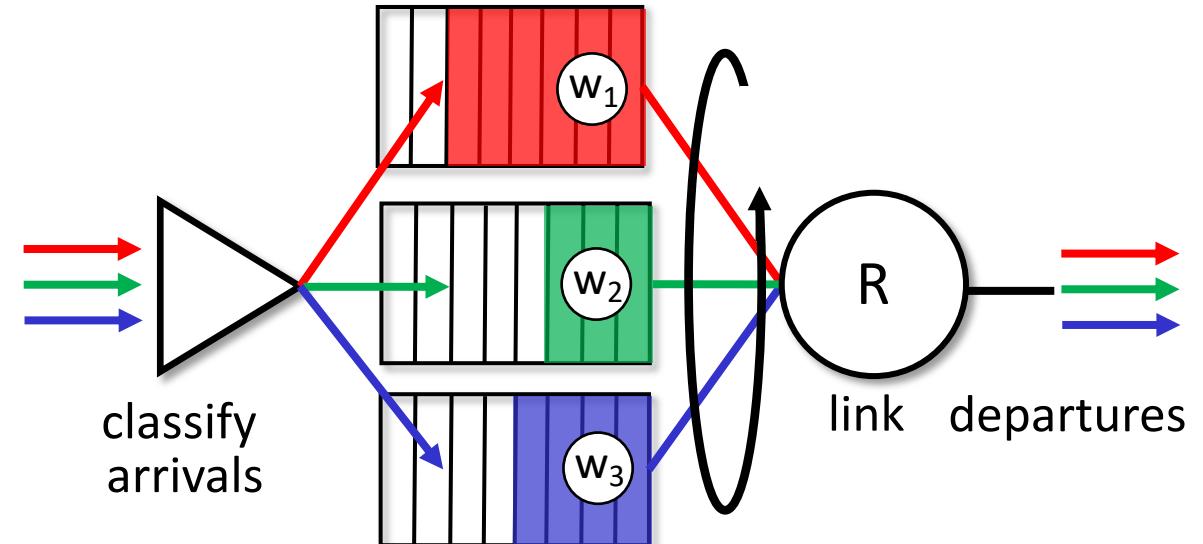
Scheduling policies: weighted fair queueing

Weighted Fair Queueing (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

- Network layer: overview
 - data plane
 - control plane
- Network layer service model
- What's inside a router
 - input ports, switching, output ports
 - buffer management, scheduling
- **Forwarding of datagrams**
- IP: the Internet Protocol
 - datagram format
 - addressing



Destination-based forwarding

- The **forwarding table** associates every **destination address** with an **output interface**

Destination Address	Link Interface
4211	1
1267	2
7382	0
2632	1
3782	1
3428	0
3727	1
5422	2

Q: what is the big problem of this approach?

Destination-based forwarding

- The **forwarding table** associates every **destination address range** with an output interface
 - Aggregate multiple addresses into a single association

Destination Address Range	Link Interface
1 – 1.449.999	0
1.450.000 – 2.999.999	1
3.000.000 – 4.250.999	2
otherwise	3

- Great scalability! But does that fit reality?

Destination-based forwarding

- The **forwarding table** associates every **destination address range** with an output interface
 - Multiple ranges can be associated with one interface

Destination Address Range	Link Interface
1 – 1.449.999	0
1.450.000 – 2.999.999	1
3.000.000 – 3.199.999	2
3.200.000 – 3.249.999	0
3.250.000 – 4.250.999	2
otherwise	3

Q: how should address range be defined?

Hierarchical addressing: route aggregation

hierarchical addressing allows efficient advertisement of routing information:

Organization 0

1 – 1.449.999

Organization 1

1.450.000 – 2.999.999

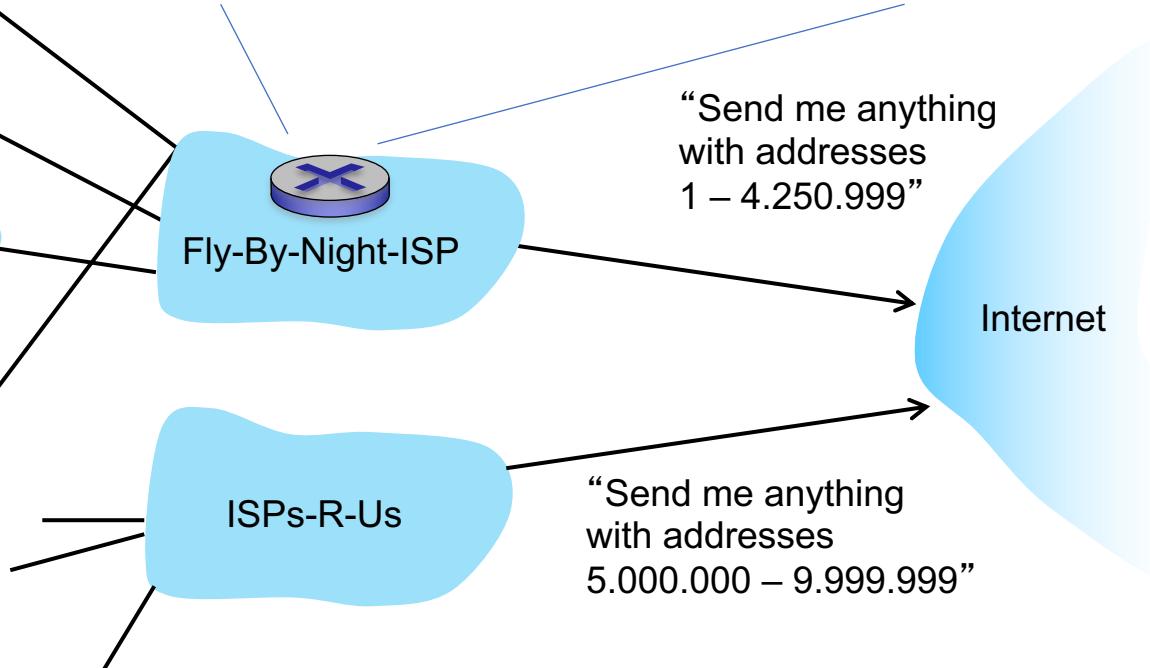
Organization 2

3.000.000 – 3.199.999

Organization 7

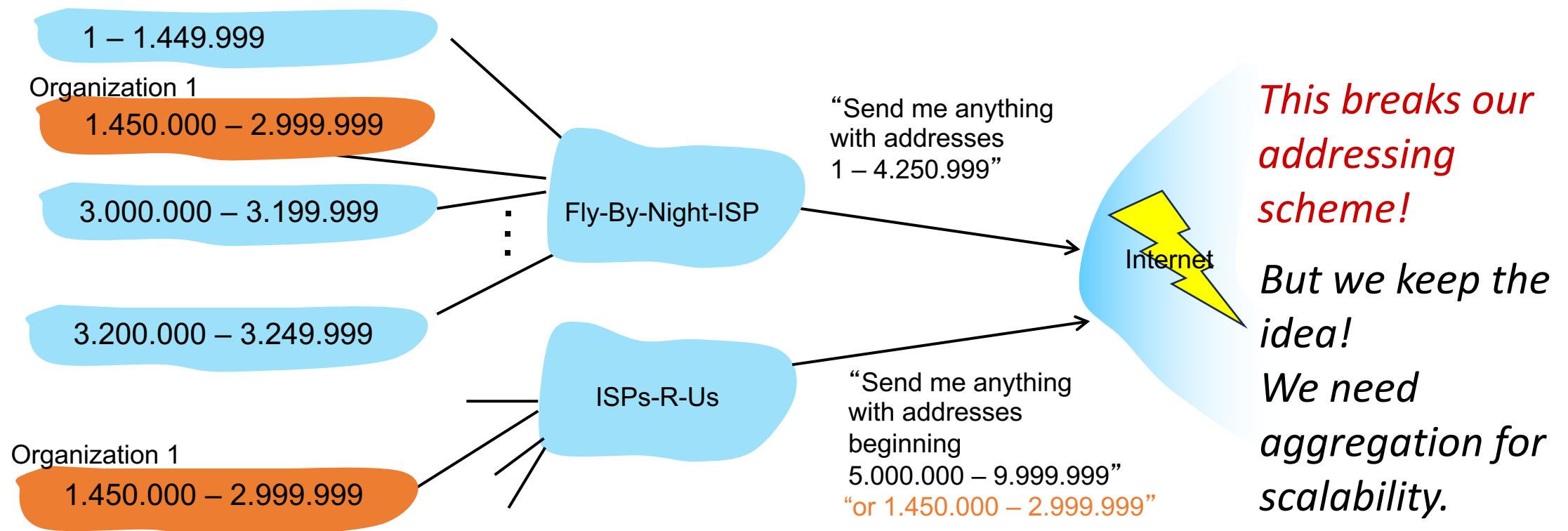
3.200.000 – 3.249.999

Destination Address Range	Link Interface
1 – 1.449.999	0
1.450.000 – 2.999.999	1
3.000.000 – 3.199.999	2
3.200.000 – 3.249.999	0
3.250.000 – 4.250.999	2
otherwise	3



Hierarchical addressing: more specific routes

- Organization 1 moves from Fly-By-Night-ISP to ISPs-R-Us taking its address with it



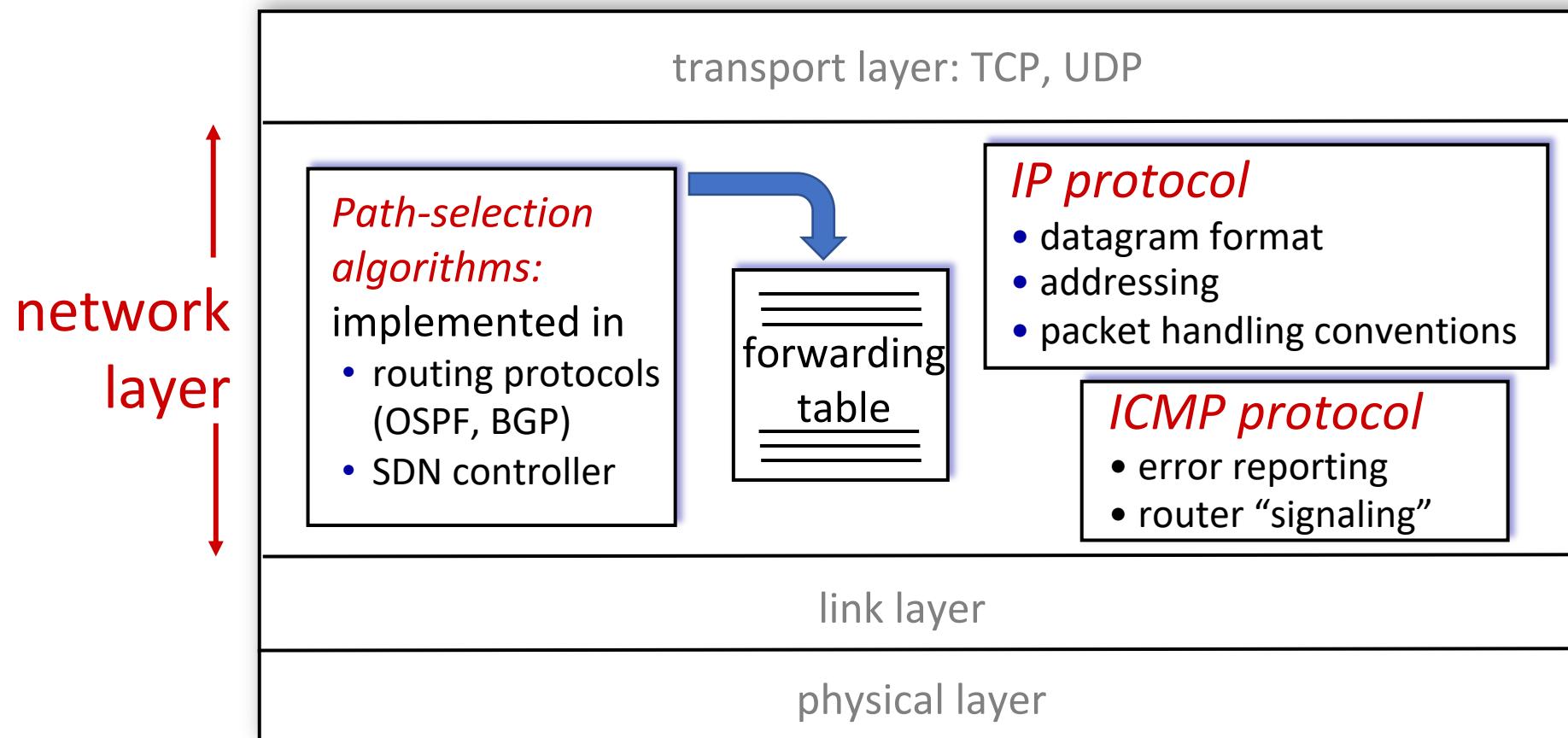
Network layer: “data plane” roadmap

- Network layer: overview
 - data plane
 - control plane
- Network layer service model
- What's inside a router
 - input ports, switching, output ports
 - buffer management, scheduling
- IP: the Internet Protocol
 - datagram format
 - addressing

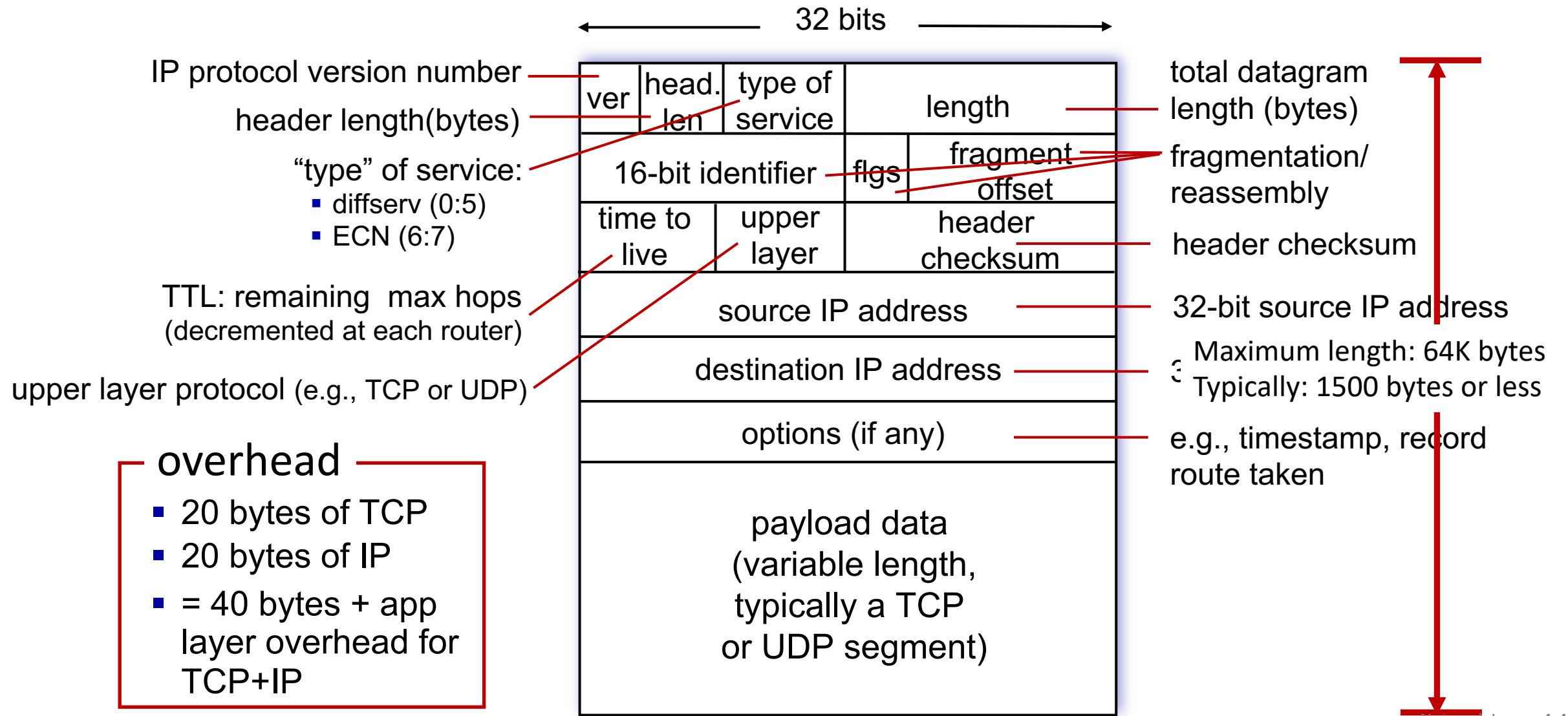


Network Layer: Internet

host, router network layer functions:

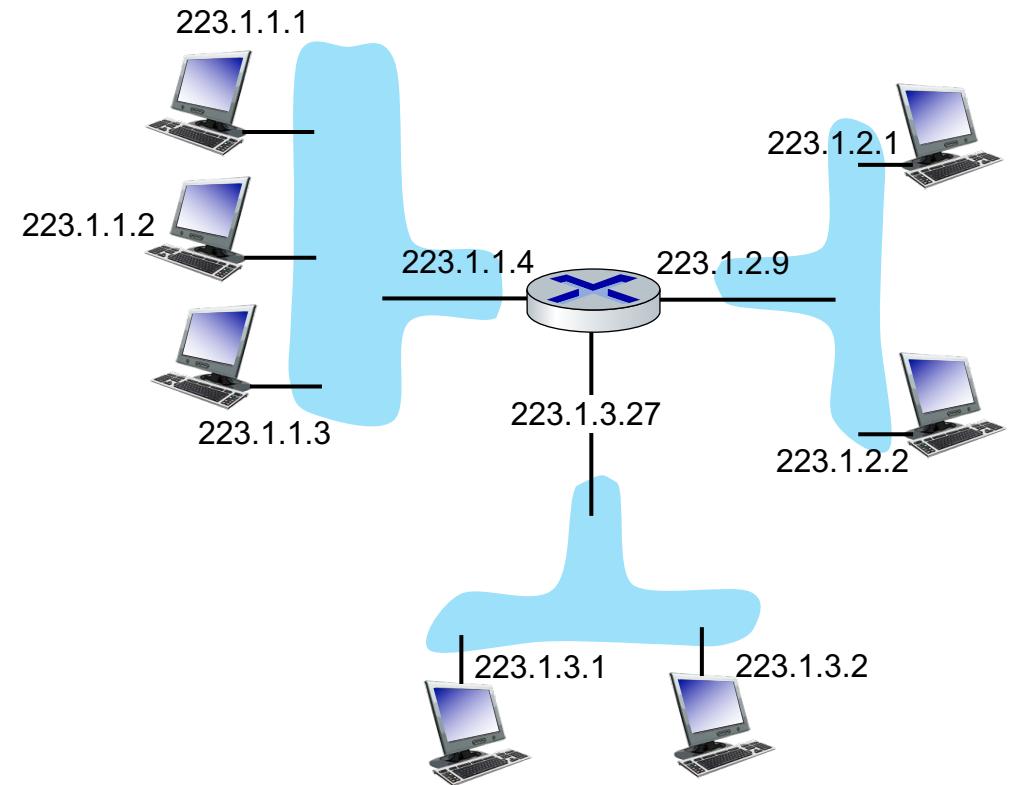


IP Datagram format



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)



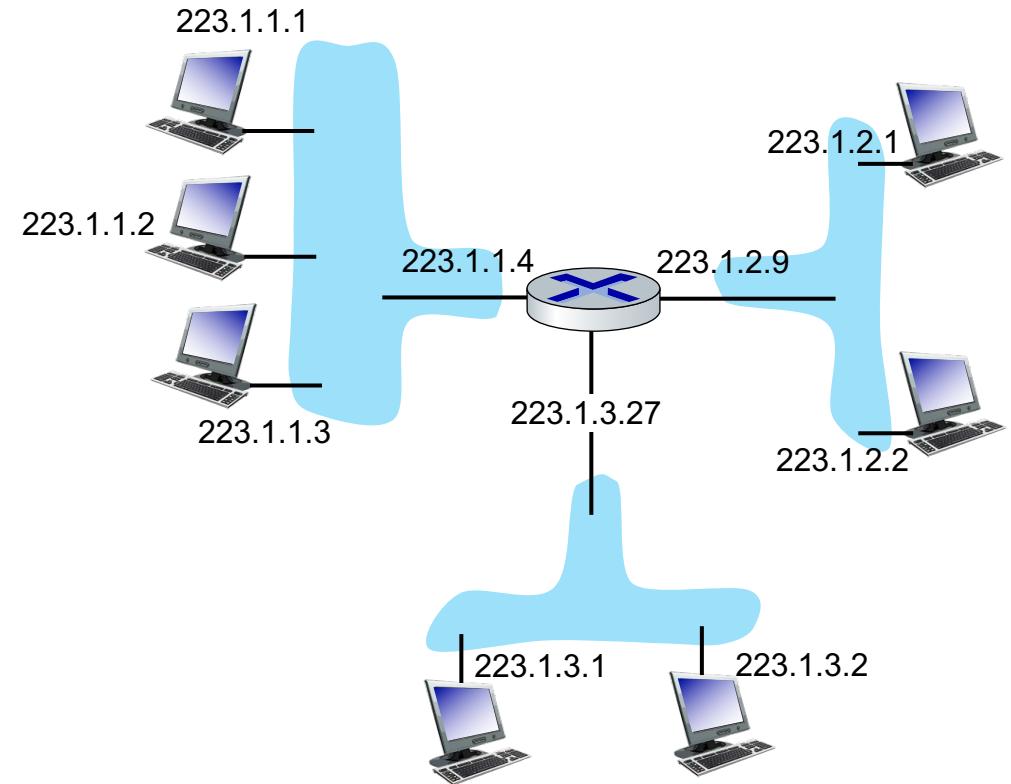
dotted-decimal IP address notation:

223.1.1.1 =

11011111	00000001	00000001	00000001
223	1	1	1

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)



dotted-decimal IP address notation:

223.1.1.1 =
 | | |
 223 1 1 1

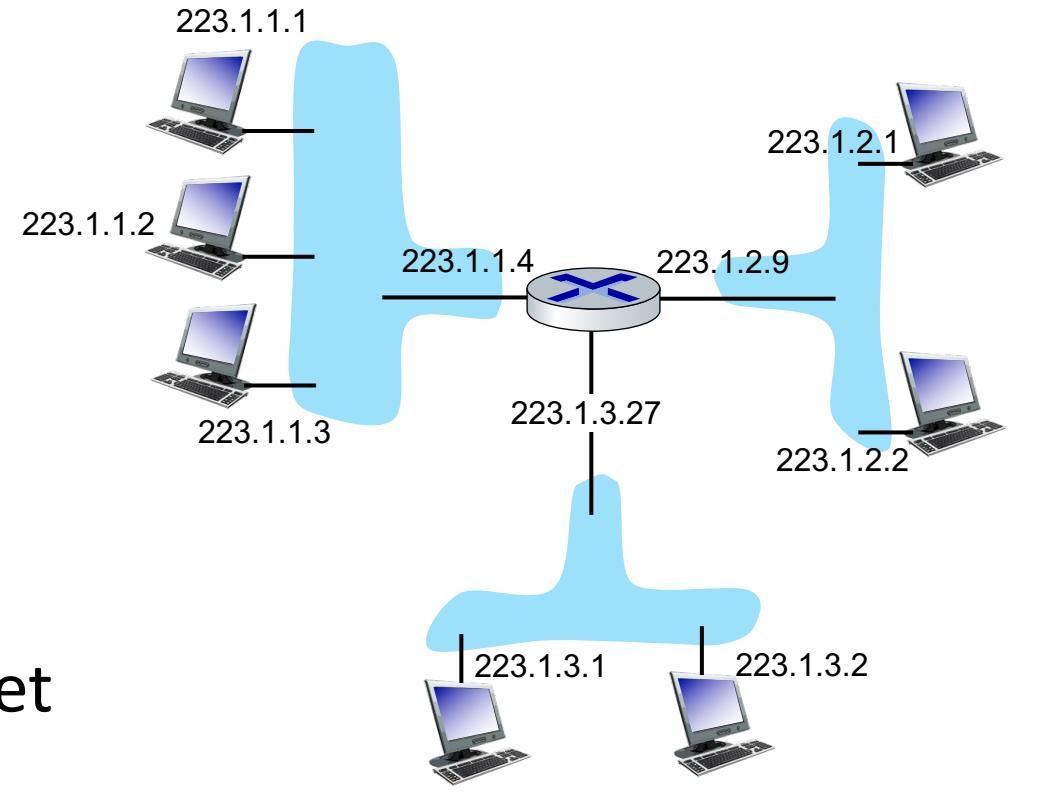
Subnets

- *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

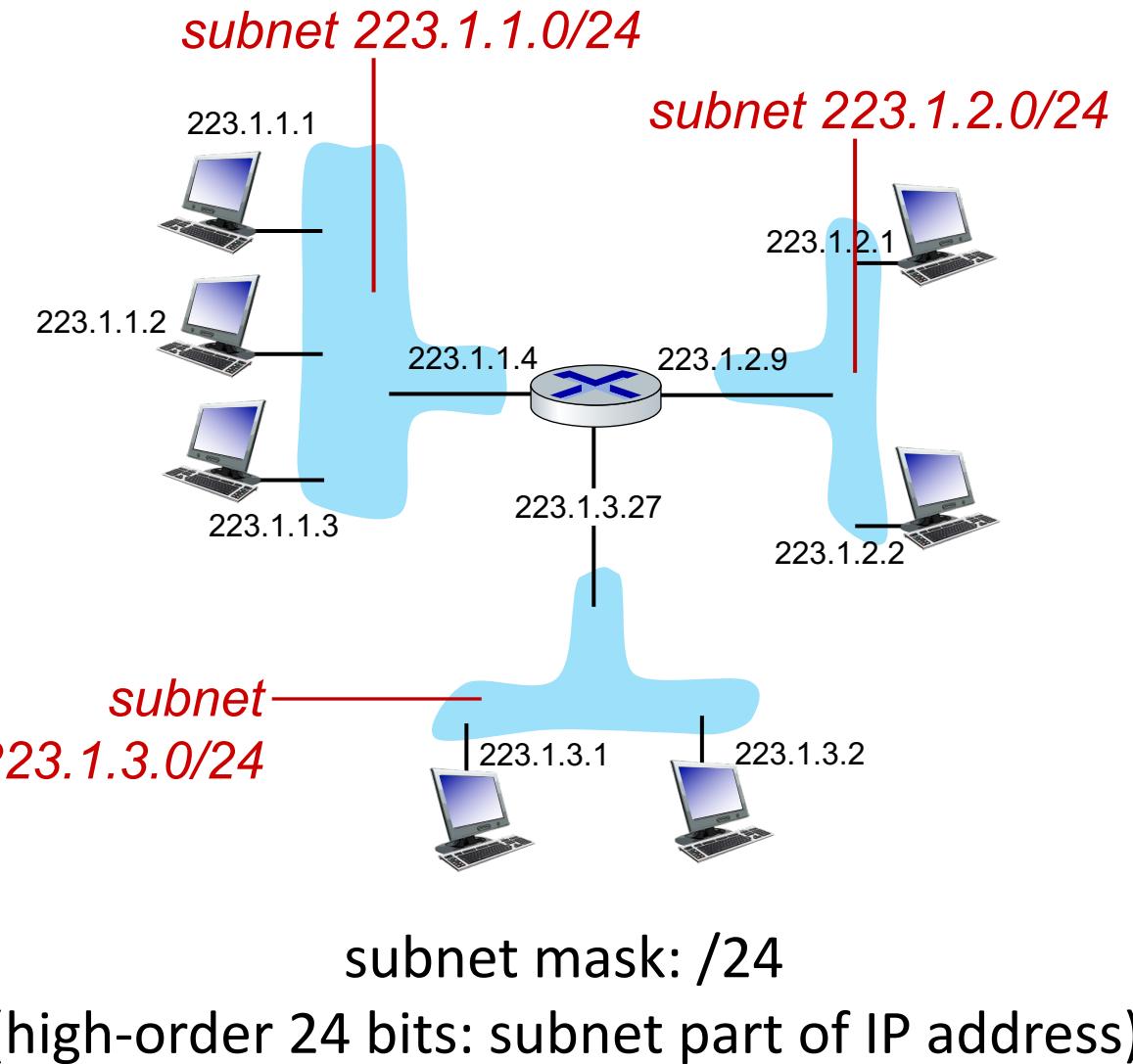


network consisting of 3 subnets

Subnets

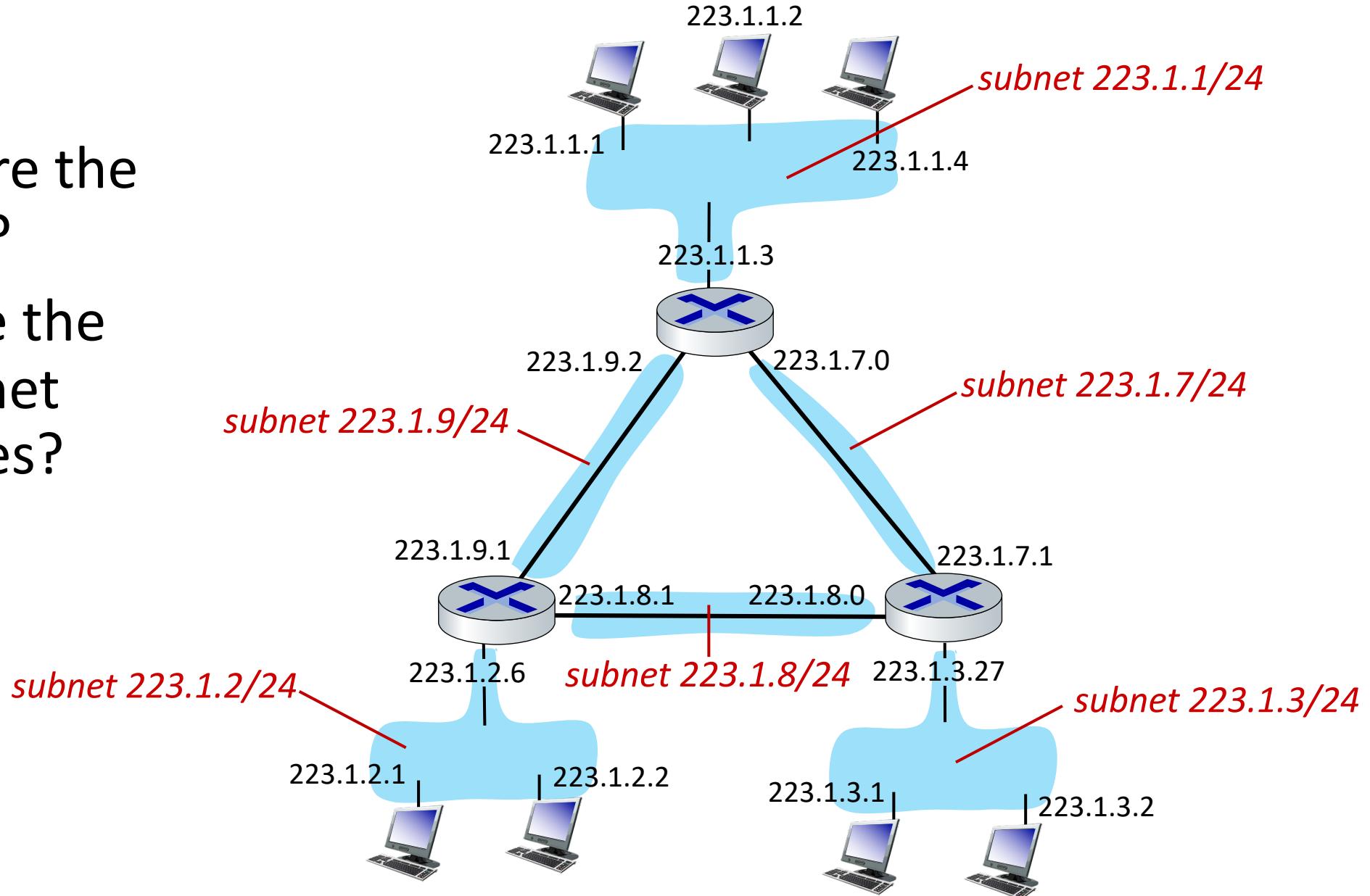
Recipe for defining subnets:

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



Subnets

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



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



IP addresses: how to get one?

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 server
 - “plug-and-play”

DHCP: Dynamic Host Configuration Protocol

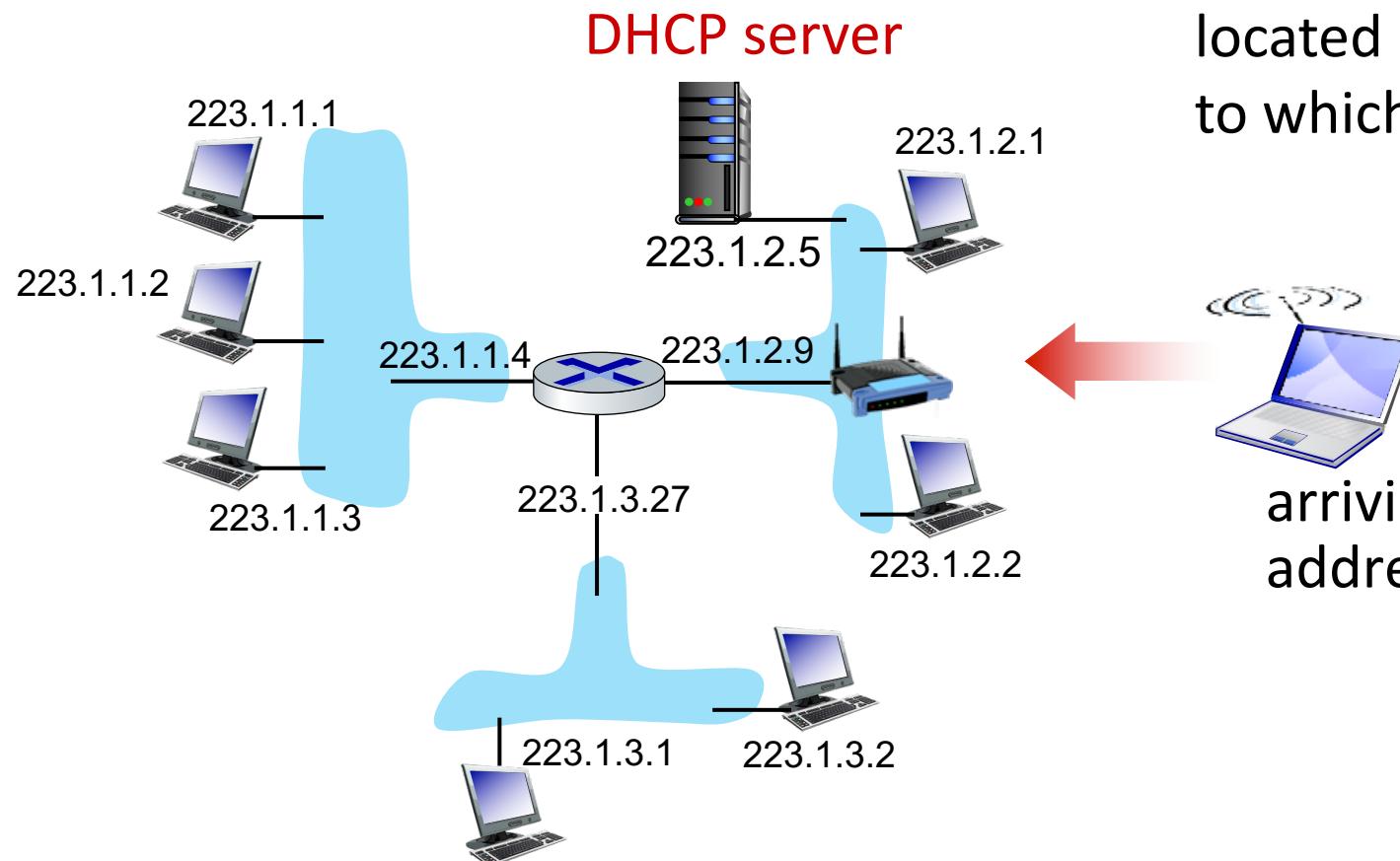
goal: host *dynamically* obtains IP address from network server when it “joins” network

- 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

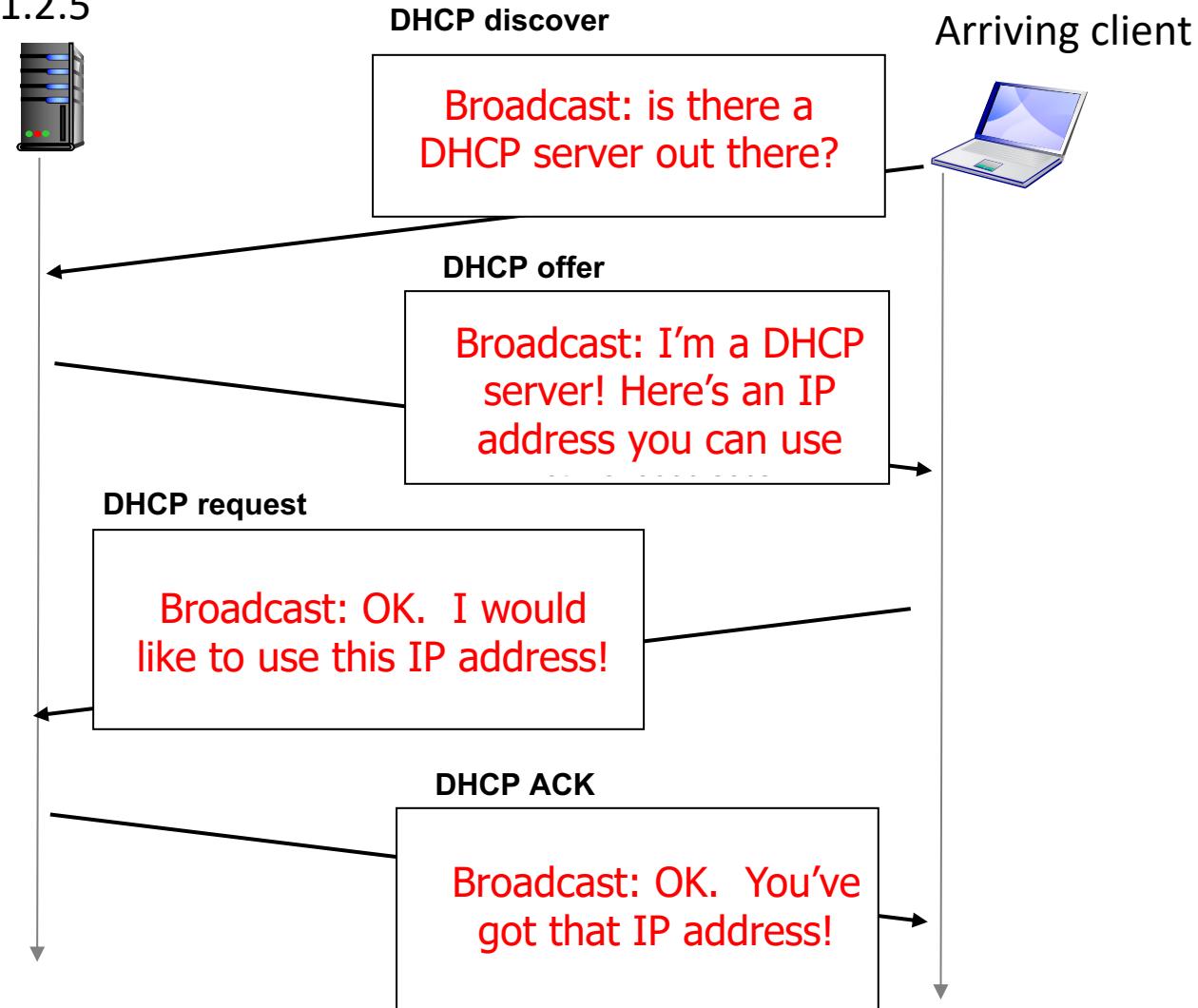


Typically, DHCP server will be co-located in router, serving all subnets to which router is attached

arriving **DHCP client** needs address in this network

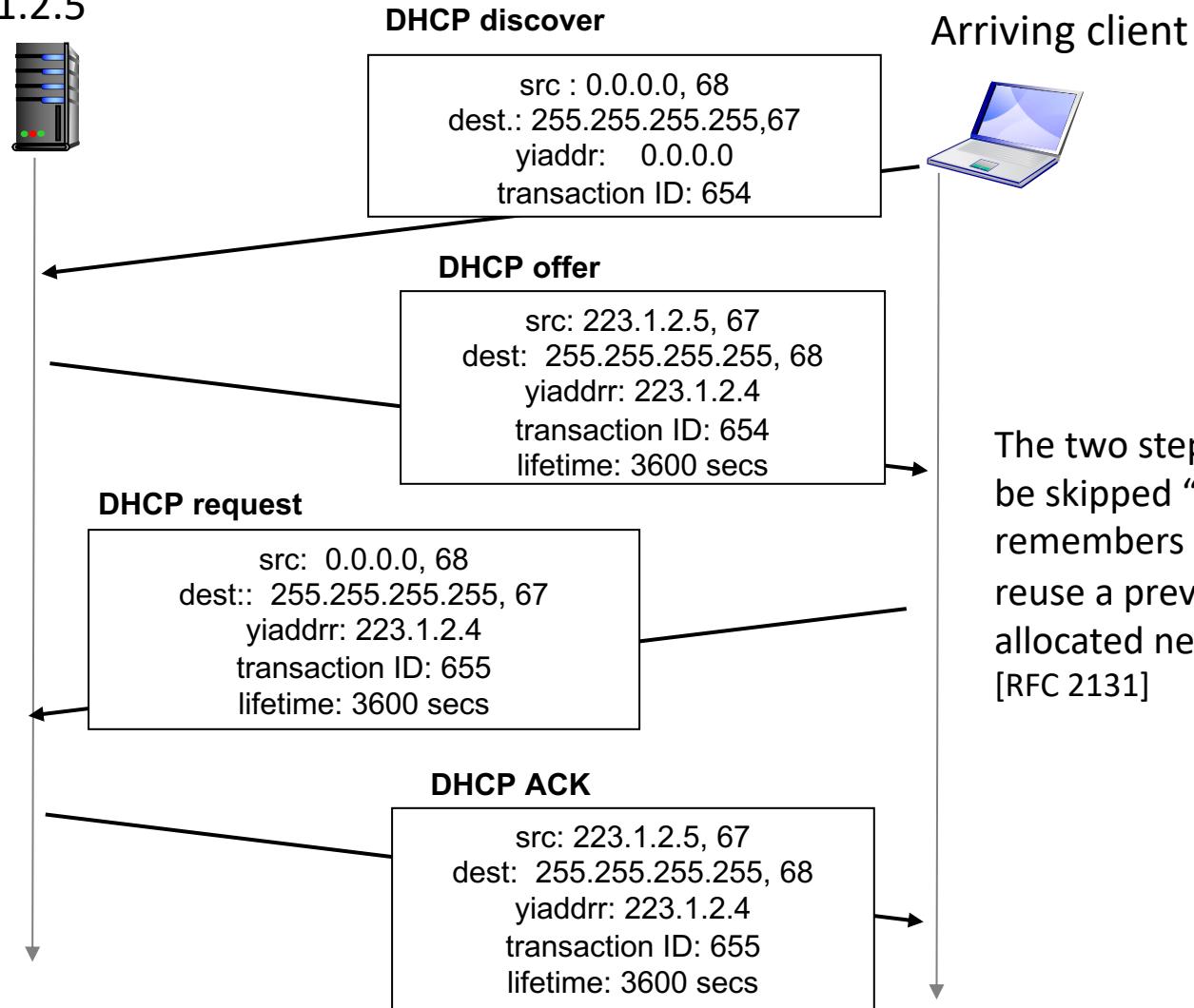
DHCP client-server scenario

DHCP server: 223.1.2.5



DHCP client-server scenario

DHCP server: 223.1.2.5

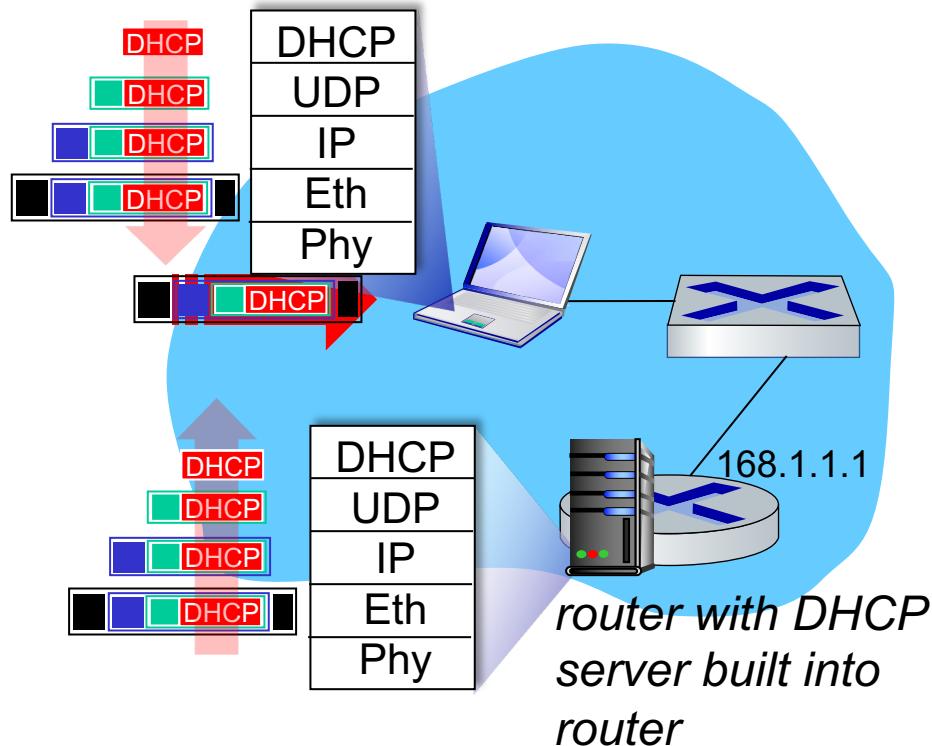


DHCP: more than IP addresses

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

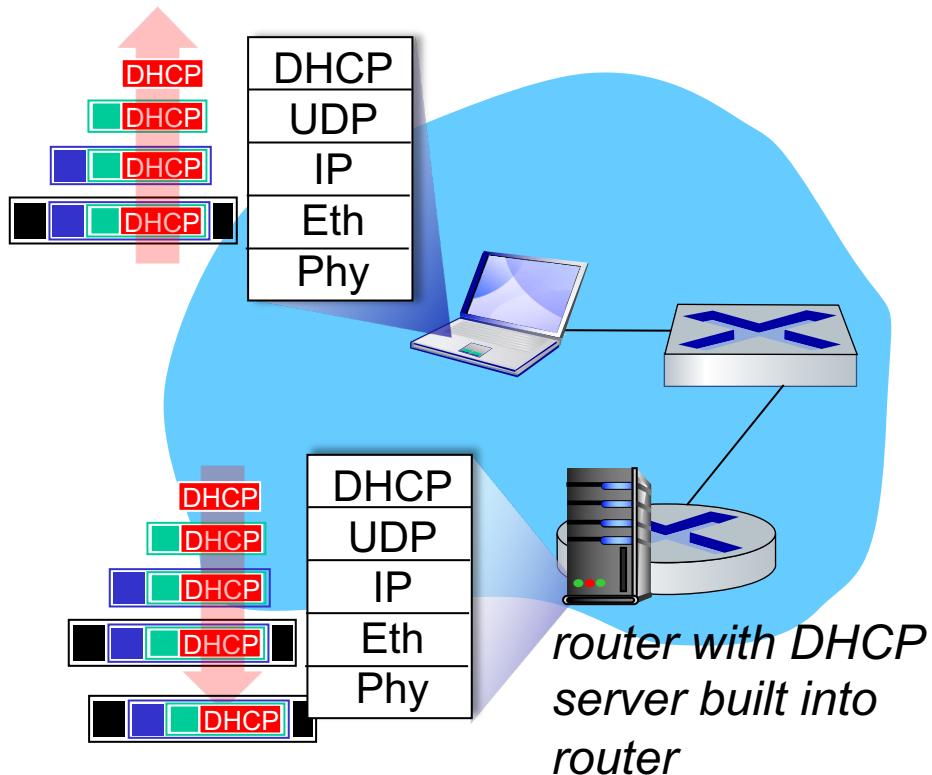
- address of first-hop router for client
- name and IP address of DNS server
- network mask (indicating network versus host portion of address)

DHCP: example



- 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: FFFFFFFFFFFF) on LAN, received at router running DHCP server
- Ethernet demux'ed to IP demux'ed, UDP demux'ed to DHCP

DHCP: example



- 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, demuxing 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?

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

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

ISP's block	<u>11001000</u> <u>00010111</u> <u>00010000</u> <u>00000000</u>	200.23.16.0/20
-------------	---	----------------

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

Organization 0	<u>11001000</u> <u>00010111</u> <u>00010000</u> <u>00000000</u>	200.23.16.0/23
----------------	---	----------------

Organization 1	<u>11001000</u> <u>00010111</u> <u>00010010</u> <u>00000000</u>	200.23.18.0/23
----------------	---	----------------

Organization 2	<u>11001000</u> <u>00010111</u> <u>00010100</u> <u>00000000</u>	200.23.20.0/23
----------------	---	----------------

...

.....

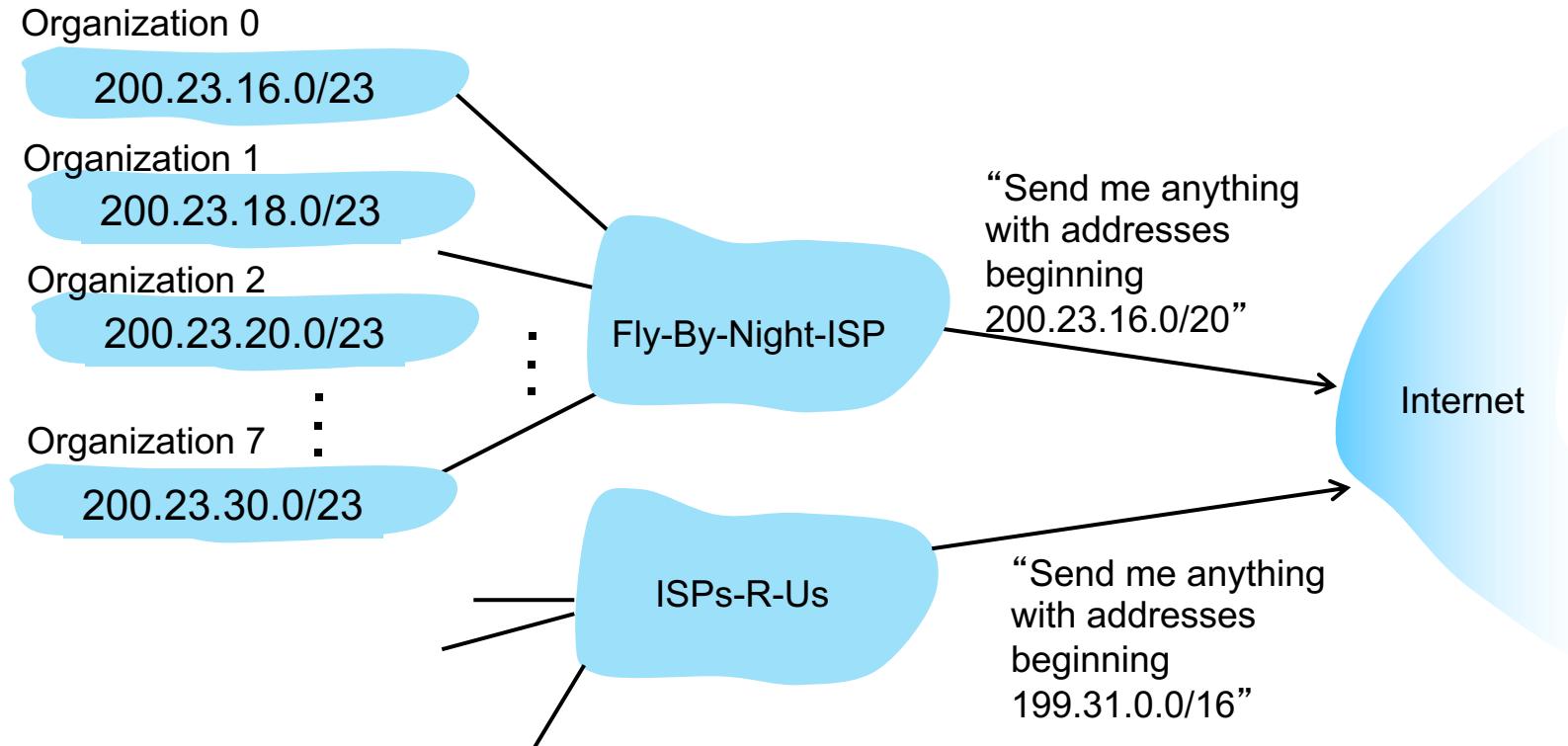
.....

.....

Organization 7	<u>11001000</u> <u>00010111</u> <u>00011110</u> <u>00000000</u>	200.23.30.0/23
----------------	---	----------------

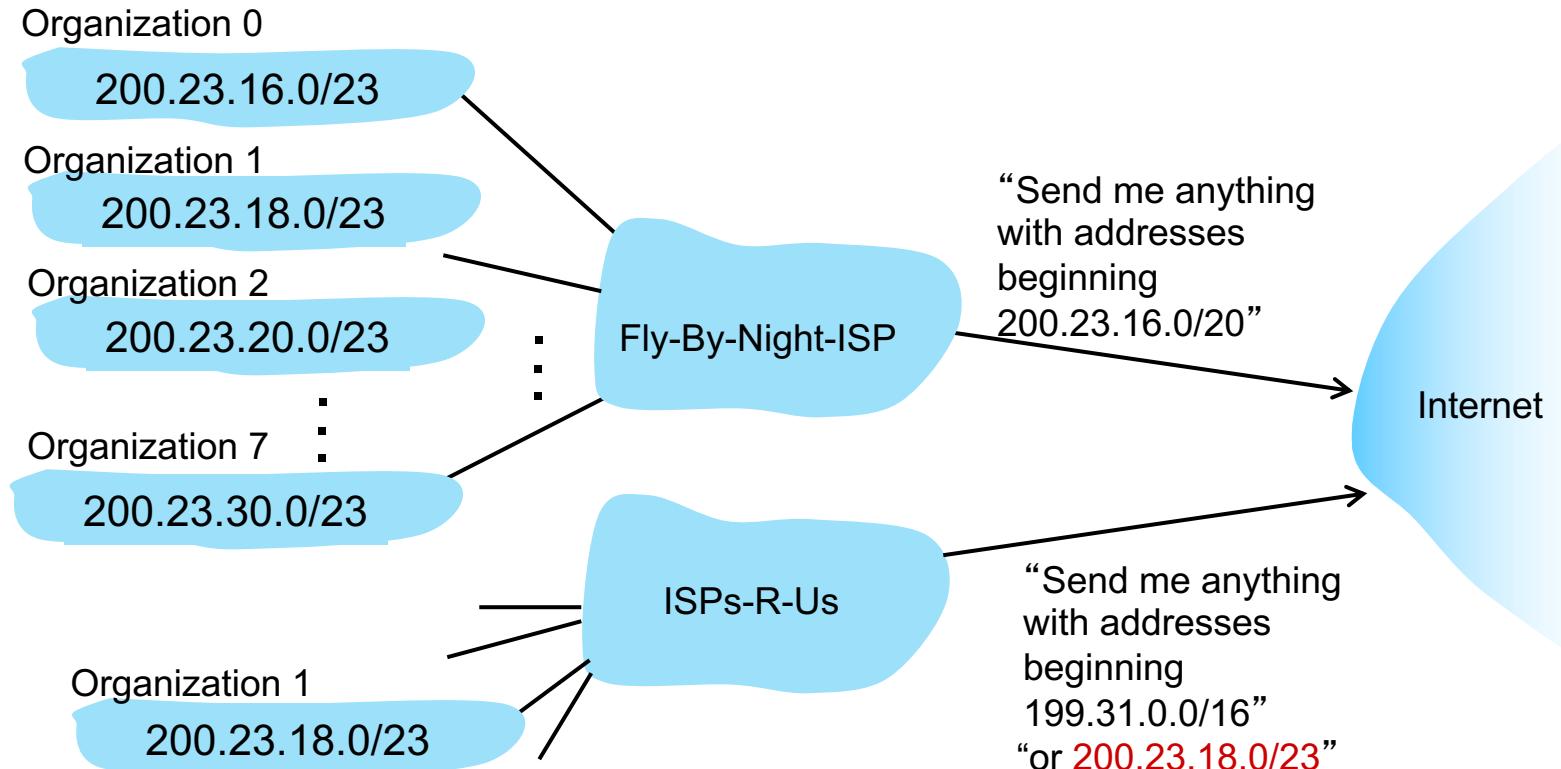
Hierarchical addressing: route aggregation

hierarchical addressing allows efficient advertisement of routing information:



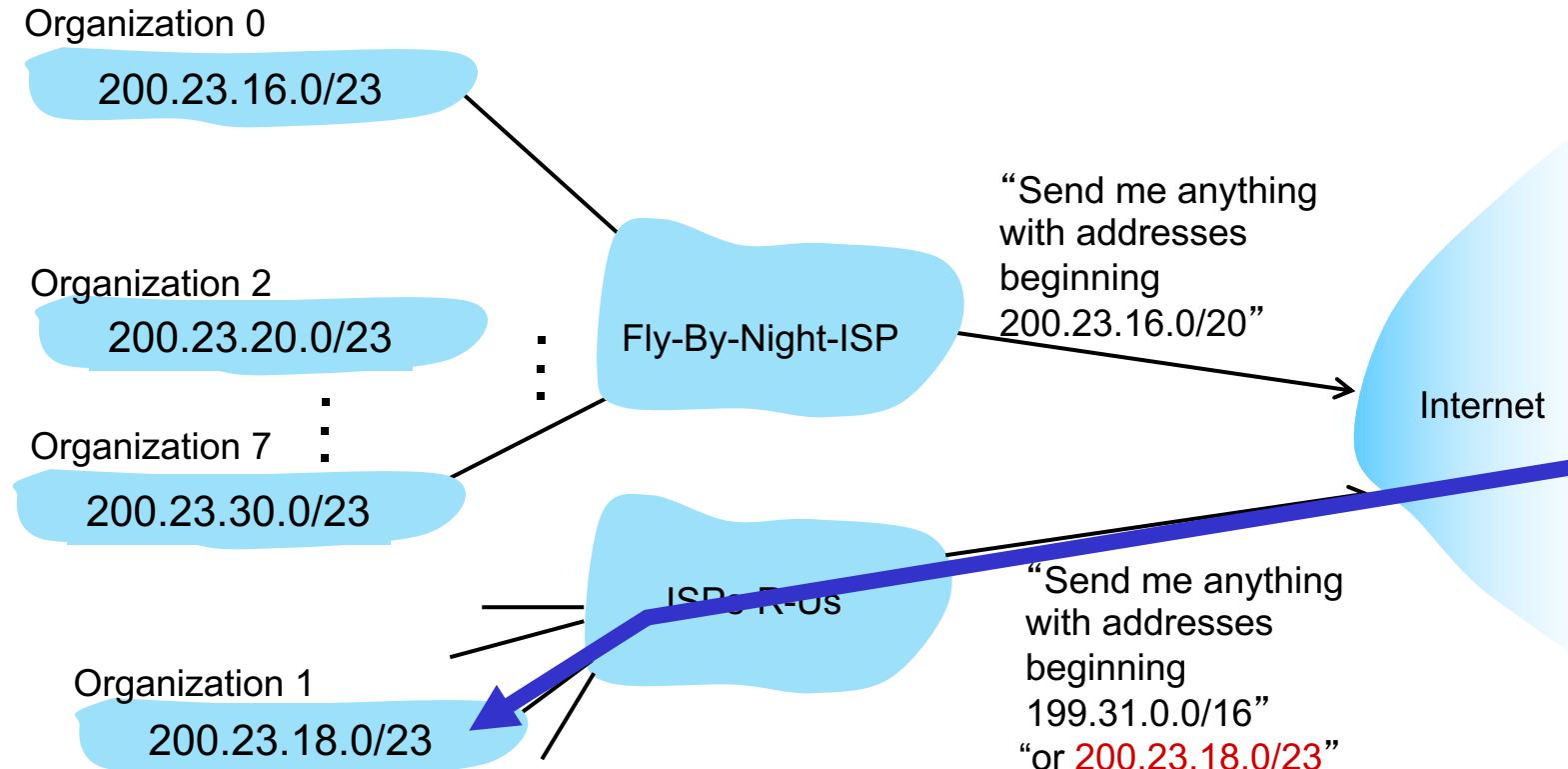
Hierarchical addressing: more specific routes

- 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

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



Destination-based forwarding

<i>forwarding table</i>	
Destination Address Range	Link Interface
11001000 00010111 00010000 00000000 through	0
11001000 00010111 00010000 00000100 through	3
11001000 00010111 00010000 00000111	
11001000 00010111 00011000 11111111	
11001000 00010111 00011001 00000000 through	2
11001000 00010111 00011111 11111111	
otherwise	3

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

Longest prefix matching

longest prefix match

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

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

examples:

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

Longest prefix matching

longest prefix match

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

Destination Address Range	Link interface
11001000 00010111 00010*****	0
11001000 00010111 00011000 *****	1
11001000 1 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

- 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

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)

More problems and solution ideas

- What problems remain?
 - Discussion in small groups (10 minutes)
 - Presentation of findings (oral, no slides)

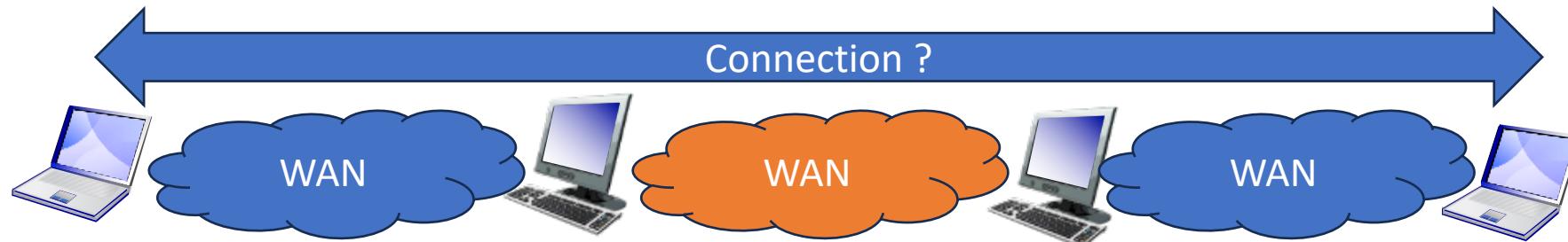
Chapter 4: Summary

- Forwarding and Routing are very different
 - Router needs to decide on which outgoing interface to send the datagram towards its destination
 - We only looked at forwarding so far, routing is coming soon.
- Network layer service
 - The Internet provides a best effort service only
- Router internals
 - Buffering at incoming and outgoing interface
- Address ranges and aggregation for scalability
 - A suitable addressing hierarchy is needed to support this
- IPv4 protocol and datagram

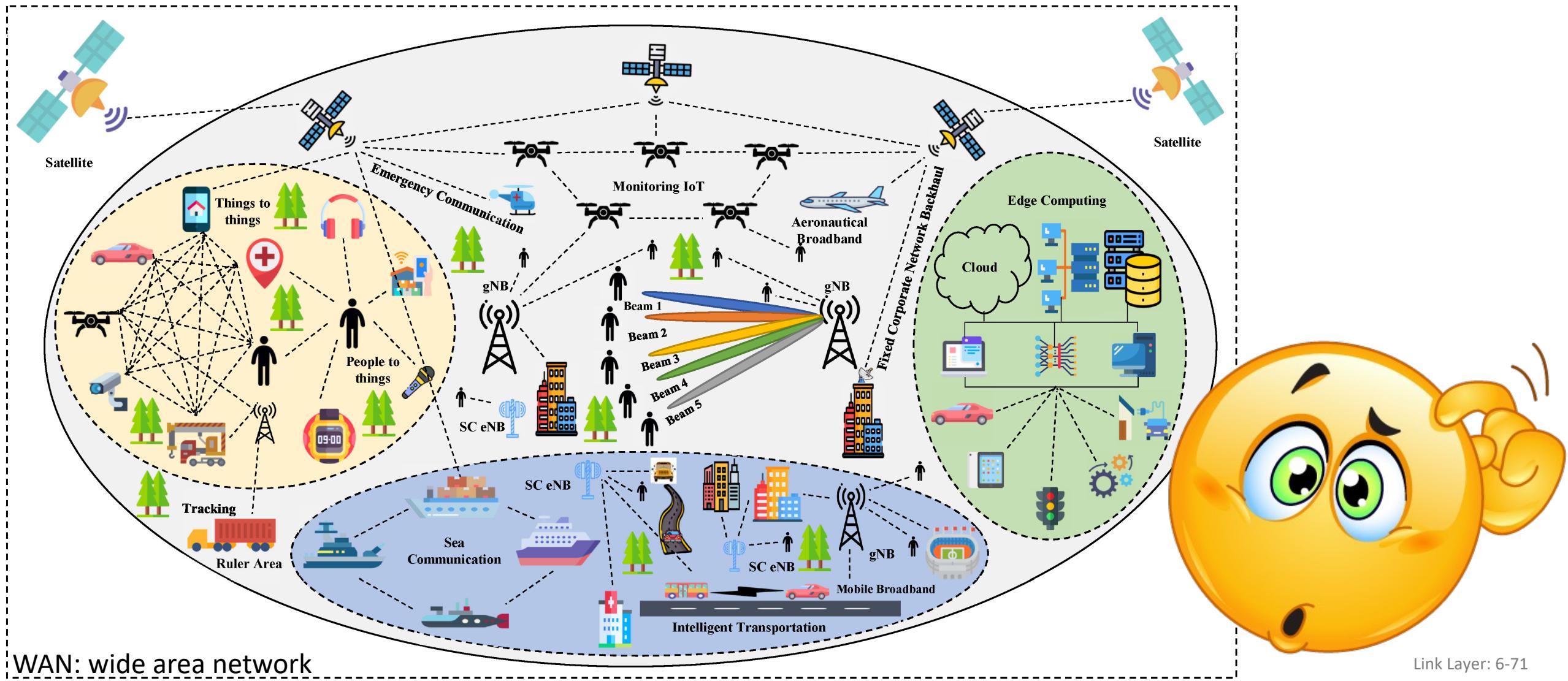
More problems and solution ideas

- What problems remain?
 - Discussion in small groups (10 minutes)

Real world application – building even larger networks



Real world application – too many devices



Topics for next lecture

1. NAT
2. IPv6
3. Tunneling