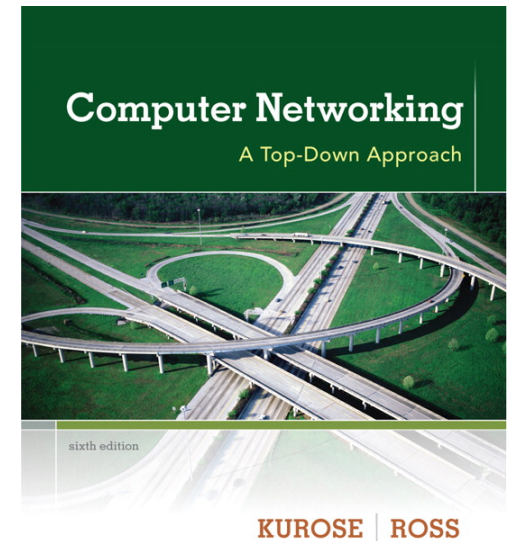


# Chapter 4

## Network Layer

Reti degli Elaboratori  
Canale ALProf.ssa Chiara  
Petrioli  
a.a. 2018/2019

We thank for the support material Prof. Kurose-Ross  
All material copyright 1996-2012  
© J.F Kurose and K.W. Ross, All Rights Reserved



*Computer  
Networking: A Top  
Down Approach*  
6<sup>th</sup> edition  
Jim Kurose, Keith Ross  
Addison-Wesley  
March 2012

# Chapter 4: network layer

## *chapter goals:*

- ❖ understand principles behind network layer services:
  - network layer service models
  - forwarding versus routing
  - how a router works
  - routing (path selection)
  - broadcast, multicast
- ❖ instantiation, implementation in the Internet

# Chapter 4: outline

## 4.1 introduction

## 4.2 virtual circuit and datagram networks

## 4.3 what's inside a router

## 4.4 IP: Internet Protocol

- datagram format
- IPv4 addressing
- ICMP
- IPv6

## 4.5 routing algorithms

- link state
- distance vector
- hierarchical routing

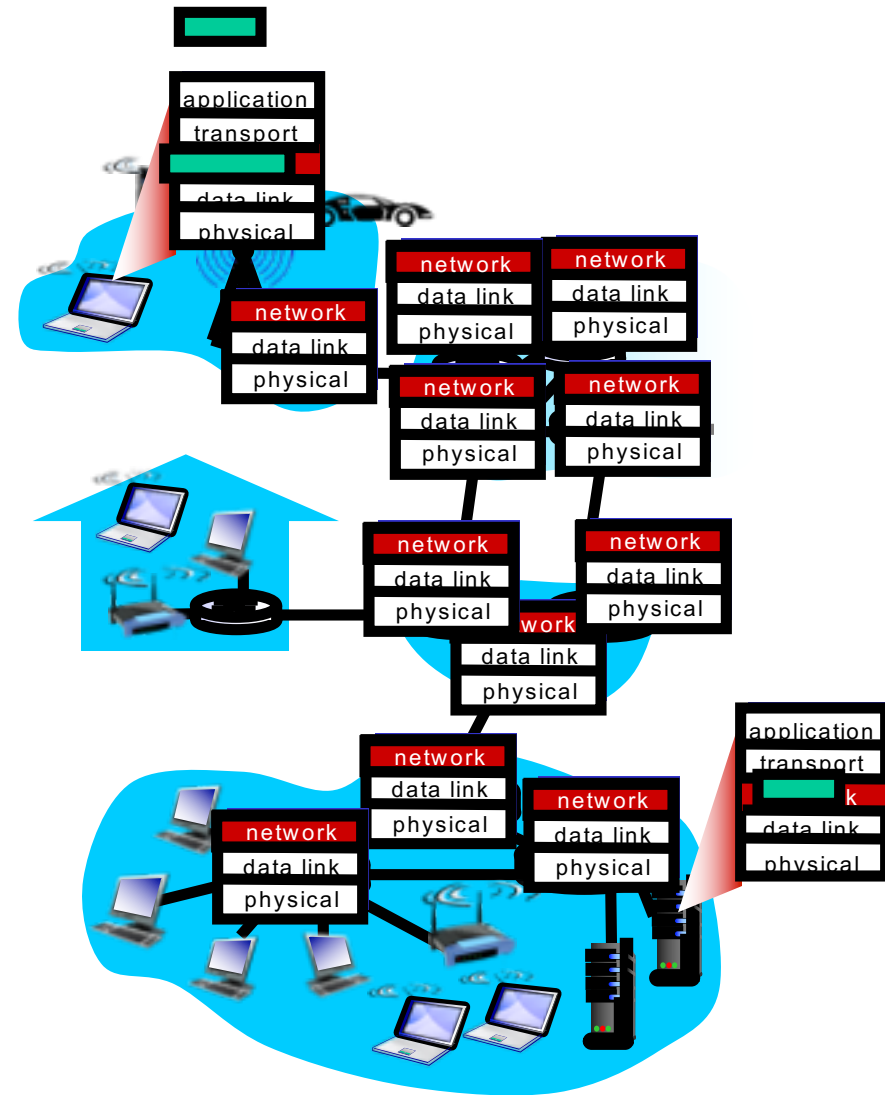
## 4.6 routing in the Internet

- RIP
- OSPF
- BGP

## 4.7 broadcast and multicast routing

# Network layer

- ❖ transport segment from sending to receiving host
- ❖ on sending side encapsulates segments into datagrams
- ❖ on receiving side, delivers segments to transport layer
- ❖ network layer protocols in *every* host, router
- ❖ router examines header fields in all IP datagrams passing through it



# Two key network-layer functions

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

- ❖ *routing*: determine route taken by packets from source to dest.

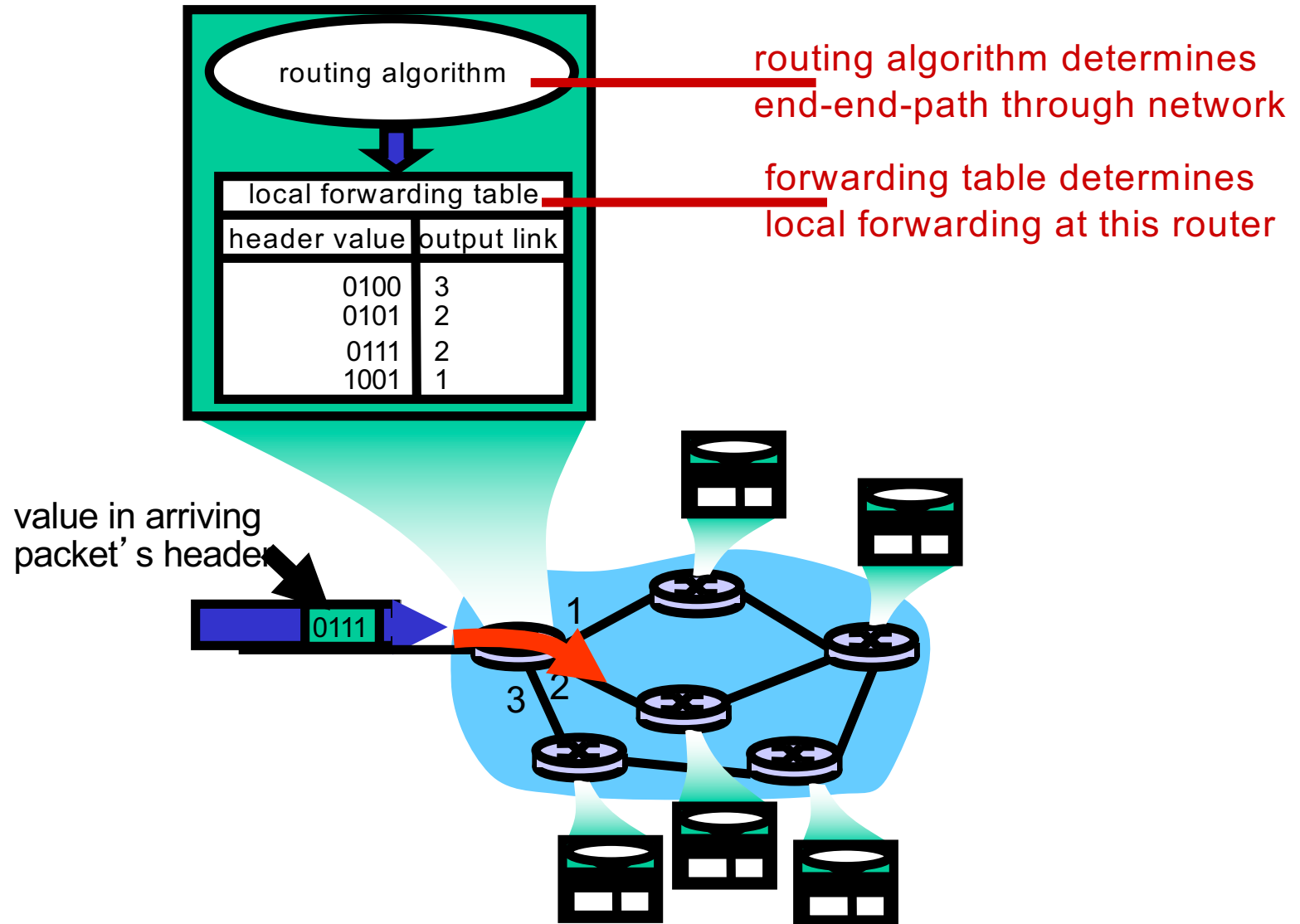
  - *routing algorithms*

*analogy:*

- ❖ *routing*: process of planning trip from source to dest

- ❖ *forwarding*: process of getting through single interchange

# Interplay between routing and forwarding



# Connection setup

- ❖ 3<sup>rd</sup> important function in *some* network architectures:
  - ATM, frame relay, X.25
- ❖ before datagrams flow, two end hosts *and* intervening routers establish virtual connection
  - routers get involved
- ❖ network vs transport layer connection service:
  - *network*: between two hosts (may also involve intervening routers in case of VCs)
  - *transport*: between two processes

# 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 models:

Network Architecture	Service Model	Guarantees ?				Congestion feedback
		Bandwidth	Loss	Order	Timing	
Internet	best effort	none	no	no	no	no (inferred via loss)
ATM	CBR	constant rate	yes	yes	yes	no congestion
ATM	VBR	guaranteed rate	yes	yes	yes	no congestion
ATM	ABR	guaranteed minimum	no	yes	no	yes
ATM	UBR	none	no	yes	no	no

# Chapter 4: outline

## 4.1 introduction

## 4.2 virtual circuit and datagram networks

## 4.3 what's inside a router

## 4.4 IP: Internet Protocol

- datagram format
- IPv4 addressing
- ICMP
- IPv6

## 4.5 routing algorithms

- link state
- distance vector
- hierarchical routing

## 4.6 routing in the Internet

- RIP
- OSPF
- BGP

## 4.7 broadcast and multicast routing

# Connection, connection-less service

- ❖ *datagram* network provides network-layer *connectionless* service
- ❖ *virtual-circuit* network provides network-layer *connection* service
- ❖ analogous to TCP/UDP connection-oriented / connectionless transport-layer services, but:
  - *service*: host-to-host
  - *no choice*: network provides one or the other
  - *implementation*: in network core

# Virtual circuits

“source-to-dest path behaves much like telephone circuit”

- performance-wise
- network actions along source-to-dest path

- ❖ call setup, teardown for each call *before* data can flow
- ❖ each packet carries VC identifier (not destination host address)
- ❖ every router on source-dest path maintains “state” for each passing connection
- ❖ link, router resources (bandwidth, buffers) may be *allocated* to VC (dedicated resources = predictable service)

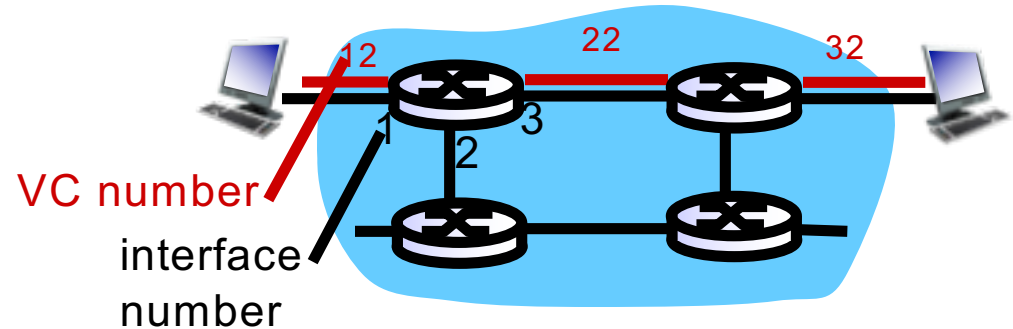
# VC implementation

*a VC consists of:*

1. *path* from source to destination
  2. *VC numbers*, one number for each link along path
  3. *entries in forwarding tables* in routers along path
- ❖ packet belonging to VC carries VC number (rather than dest address)
  - ❖ VC number can be changed on each link.
    - new VC number comes from forwarding table

# VC forwarding table

*forwarding table in  
northwest router:*

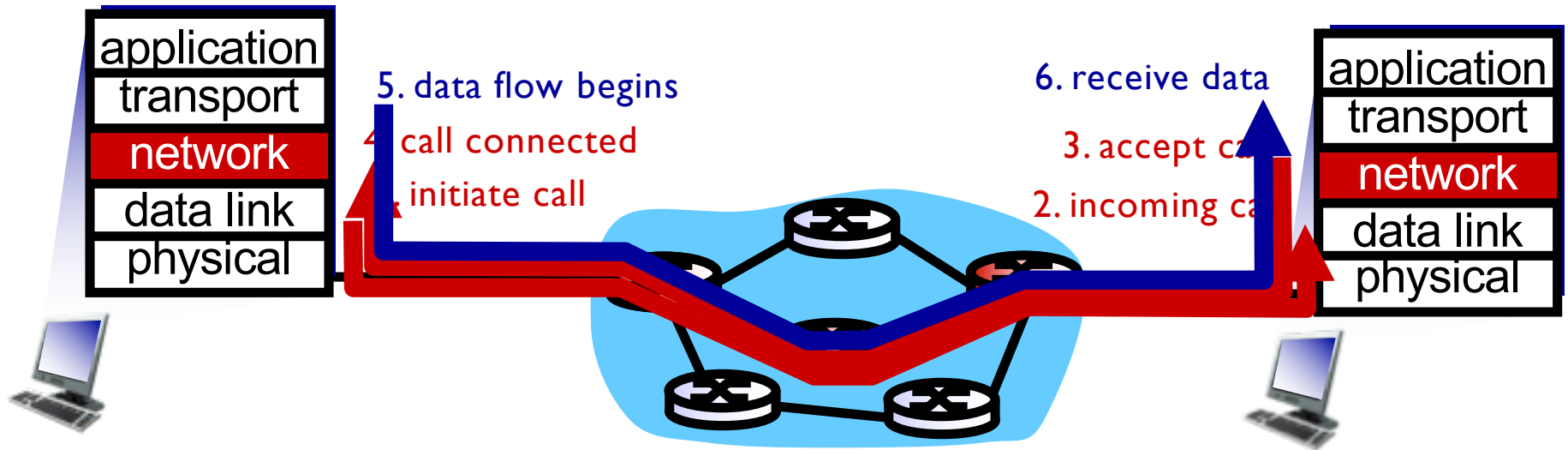


Incoming interface	Incoming VC #	Outgoing interface	Outgoing VC #
1	12	3	22
2	63	1	18
3	7	2	17
1	97	3	87
...	...	...	...

***VC routers maintain connection state information!***

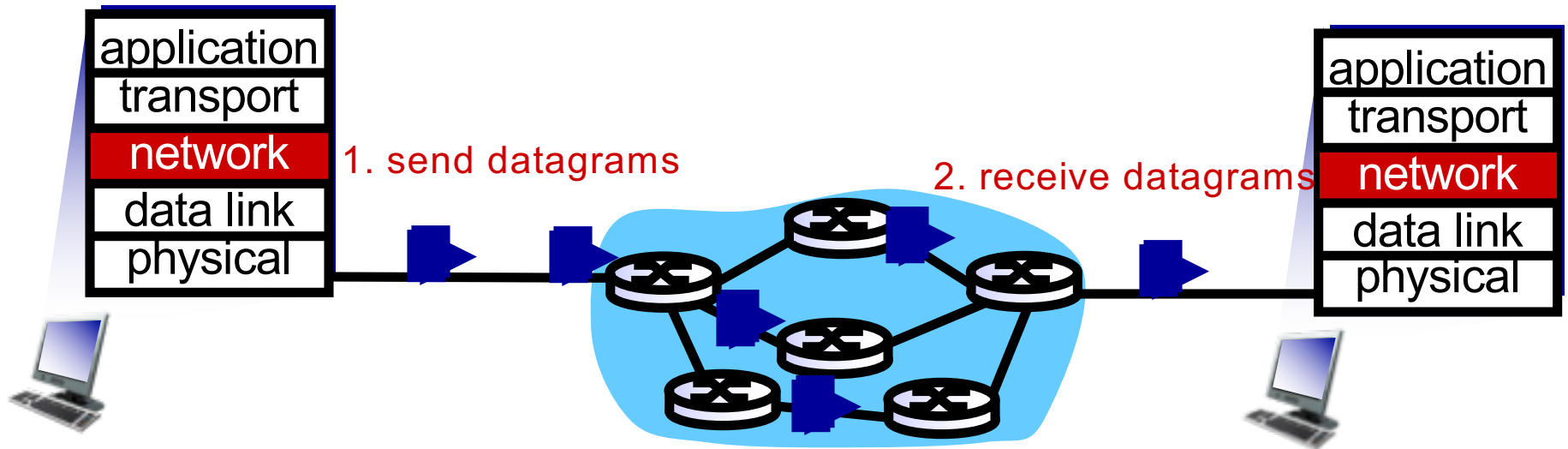
# Virtual circuits: signaling protocols

- ❖ used to setup, maintain teardown VC
- ❖ used in ATM, frame-relay, X.25
- ❖ not used in today's Internet



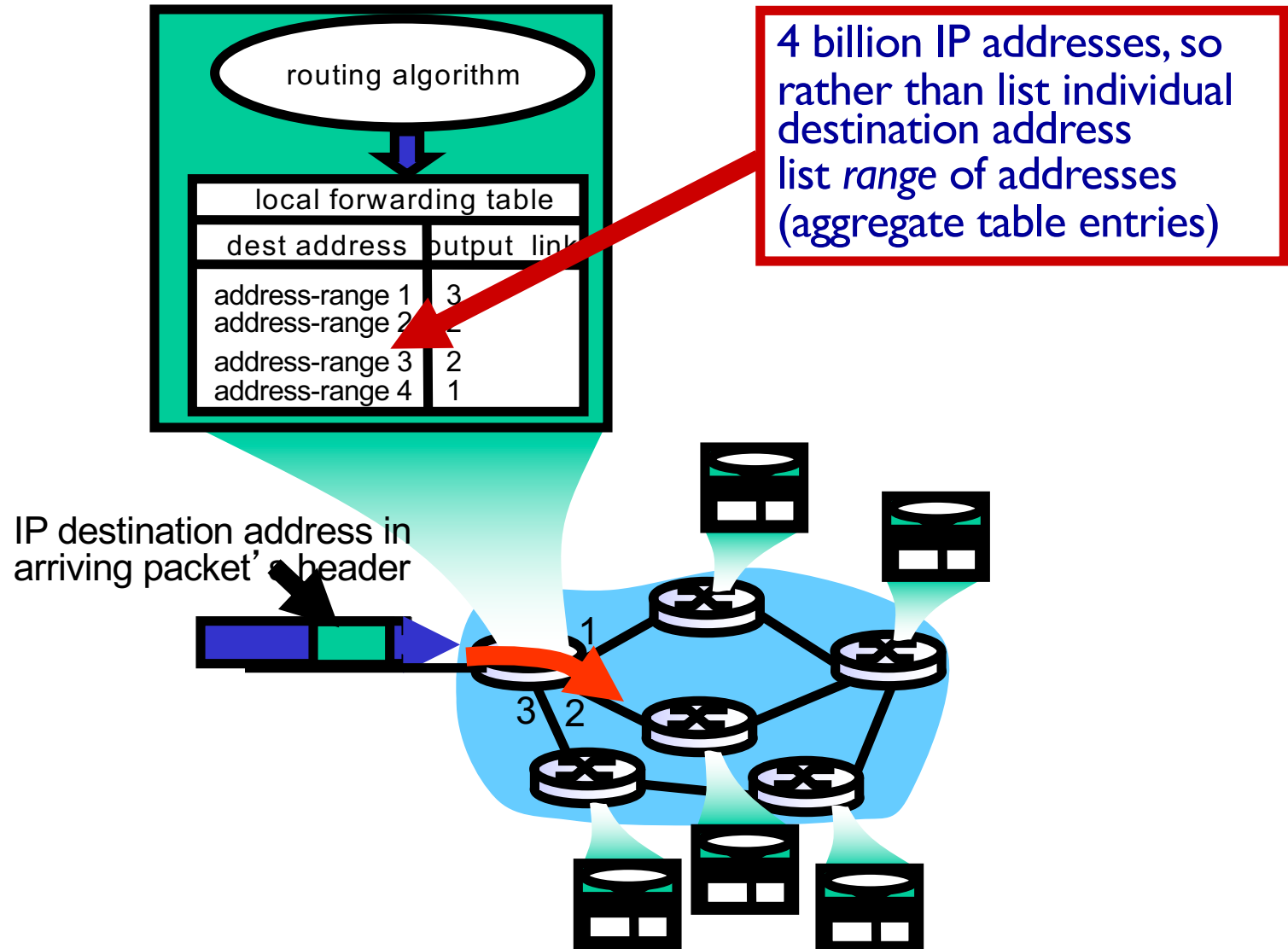
# Datagram networks

- ❖ no call setup at network layer
- ❖ routers: no state about end-to-end connections
  - no network-level concept of “connection”
- ❖ packets forwarded using destination host address





# Datagram forwarding table



# Datagram forwarding table

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

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

# Datagram or VC network: why?

## *Internet (datagram)*

- ❖ data exchange among computers
  - “elastic” service, no strict timing req.
- ❖ many link types
  - different characteristics
  - uniform service difficult
- ❖ “smart” end systems (computers)
  - can adapt, perform control, error recovery
  - **simple inside network, complexity at “edge”**

## *ATM (VC)*

- ❖ evolved from telephony
- ❖ human conversation:
  - strict timing, reliability requirements
  - need for guaranteed service
- ❖ “dumb” end systems
  - telephones
  - **complexity inside network**

# Chapter 4: outline

## 4.1 introduction

## 4.2 virtual circuit and datagram networks

## 4.3 what's inside a router

## 4.4 IP: Internet Protocol

- datagram format
- IPv4 addressing
- ICMP
- IPv6

## 4.5 routing algorithms

- link state
- distance vector
- hierarchical routing

## 4.6 routing in the Internet

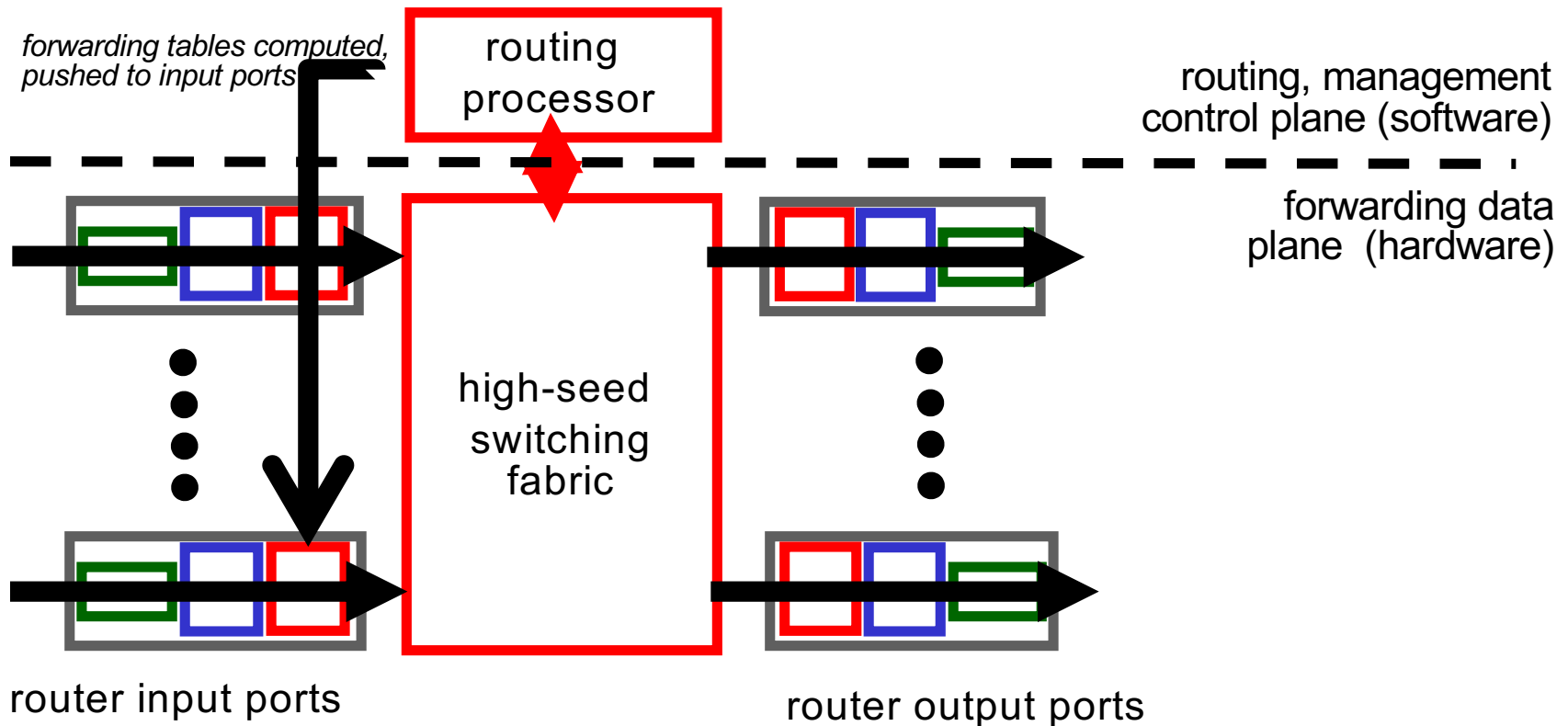
- RIP
- OSPF
- BGP

## 4.7 broadcast and multicast routing

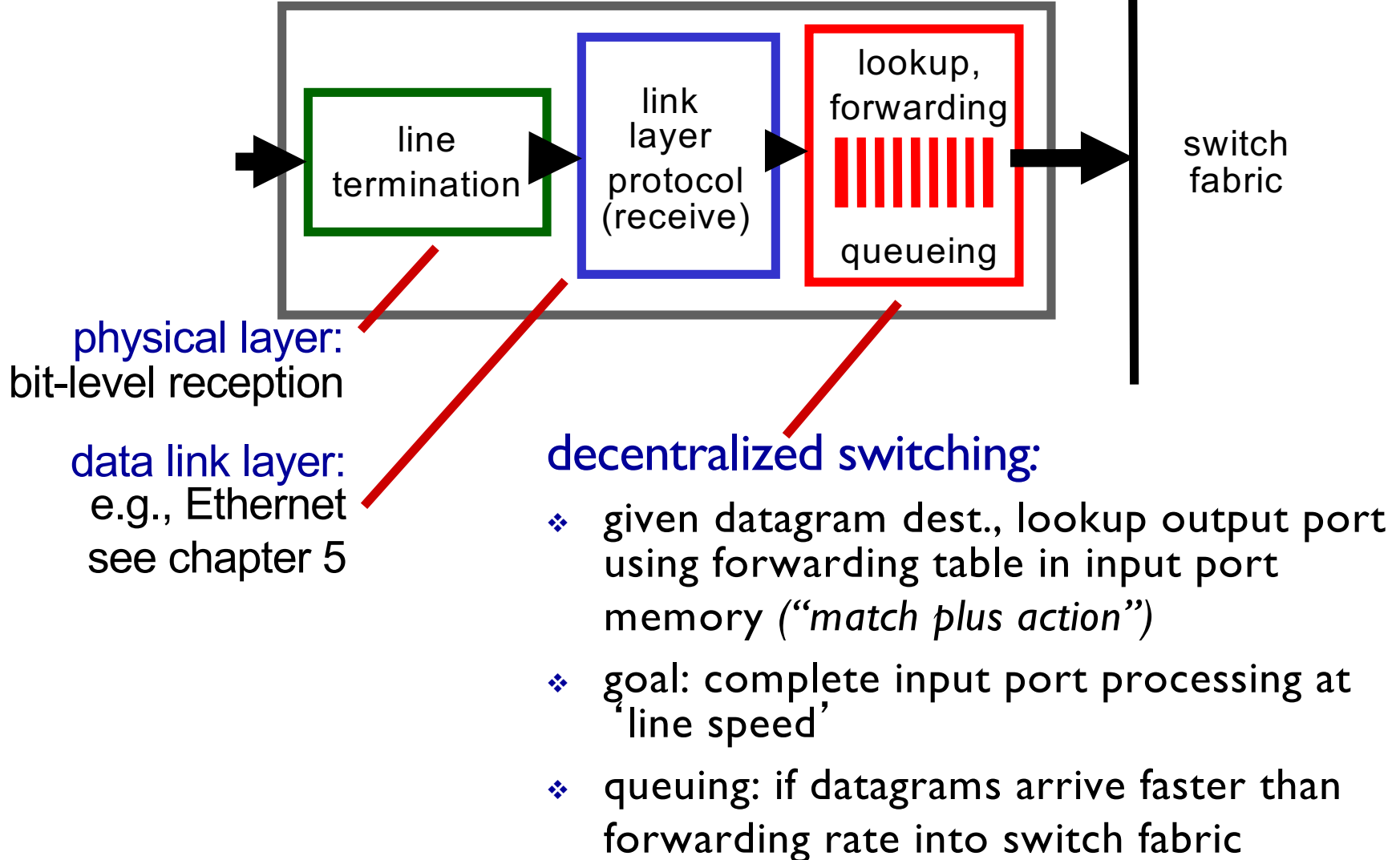
# Router architecture overview

two key router functions:

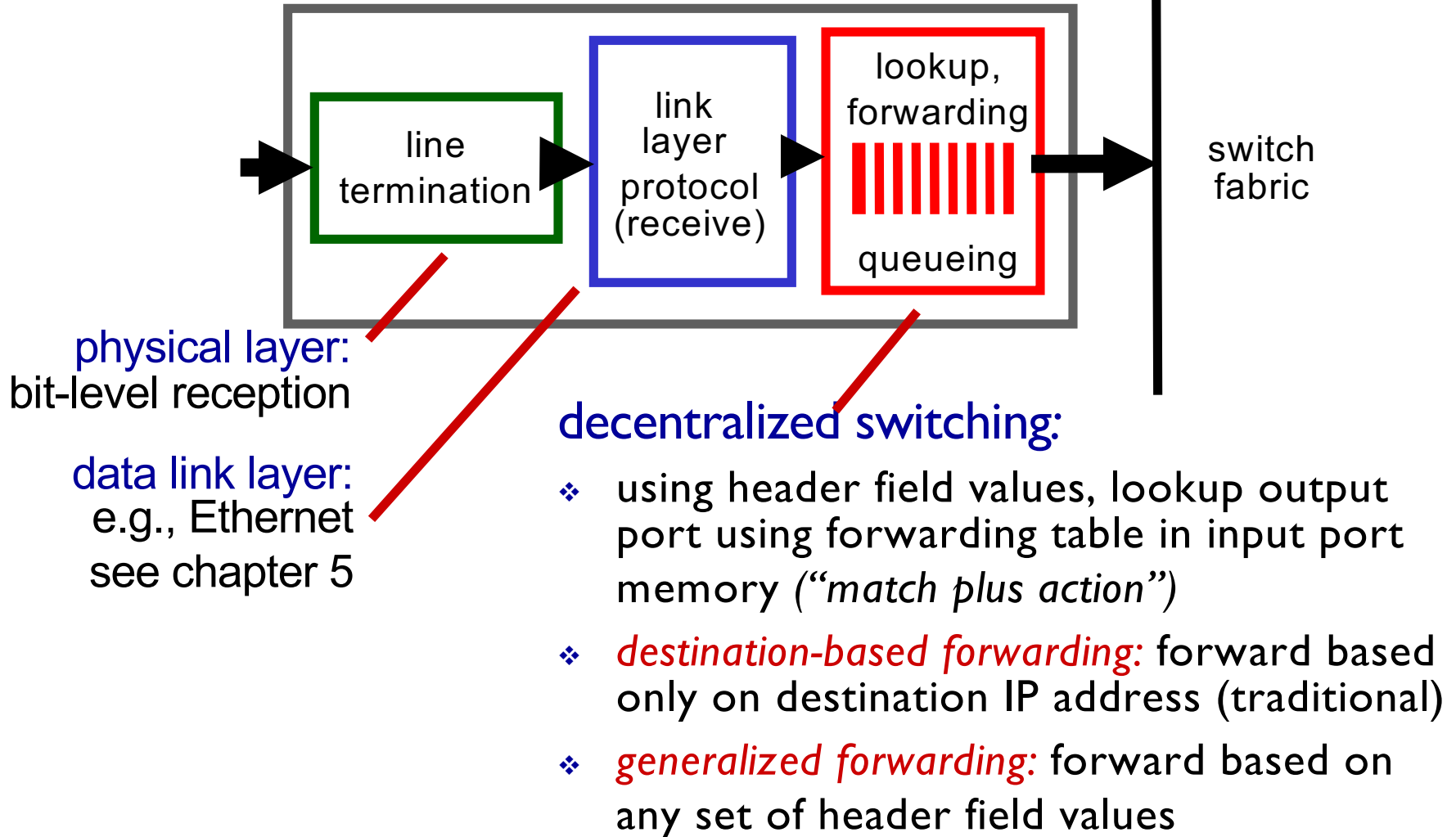
- ❖ run routing algorithms/protocol (RIP, OSPF, BGP)
- ❖ *forwarding* datagrams from incoming to outgoing link



# Input port functions



# Input port functions



# Destination-based forwarding

<i>forwarding table</i>	
Destination Address Range	Link Interface
11001000 00010111 00010000 00000000 through 11001000 00010111 00010111 11111111	0
11001000 00010111 00011000 00000000 through 11001000 00010111 00011000 11111111	1
11001000 00010111 00011001 00000000 through 11001000 00010111 00011111 11111111	2
otherwise	3

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



# Longest prefix matching

## *longest prefix matching*

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

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

examples:

DA: 11001000 00010111 00010110 10100001

which interface?

DA: 11001000 00010111 00011000 10101010

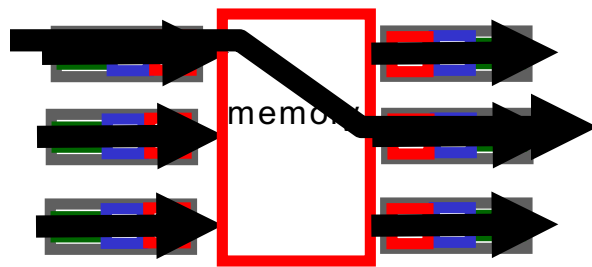
which interface?

# Longest prefix matching

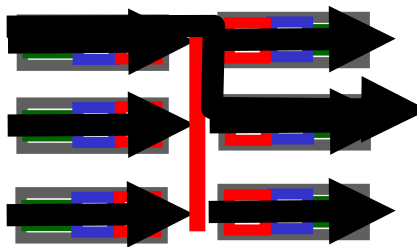
- ❖ we'll see *why* longest prefix matching is used shortly, when we study addressing
- ❖ longest prefix matching: often performed using ternary content addressable memories (TCAMs)
  - *content addressable*: present address to TCAM: retrieve address in one clock cycle, regardless of table size
  - Cisco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

# Switching fabrics

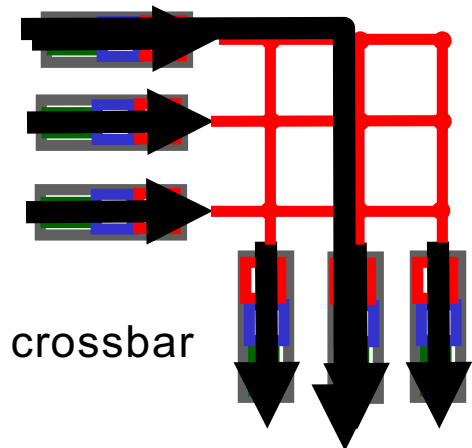
- ❖ transfer packet from input buffer to appropriate output buffer
- ❖ switching rate: rate at which packets can be transfer from inputs to outputs
  - often measured as multiple of input/output line rate
  - N inputs: switching rate N times line rate desirable
- ❖ three types of switching fabrics



memory



bus

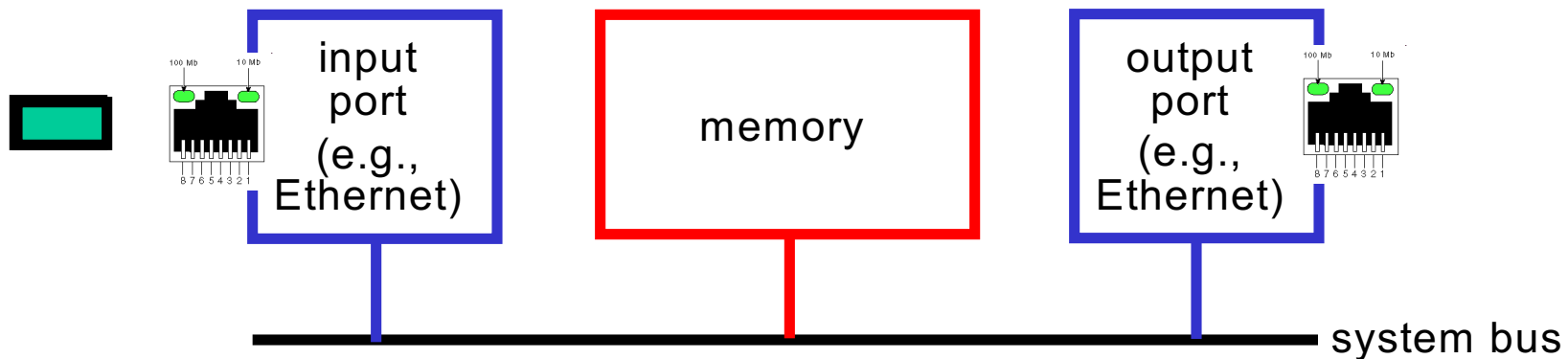


crossbar

# Switching via memory

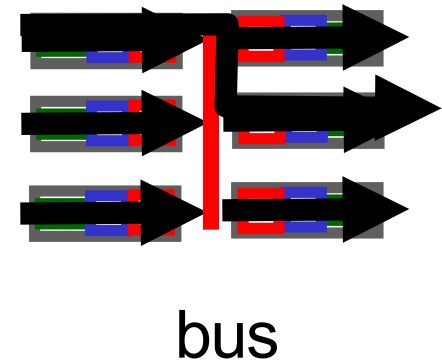
## *first generation routers:*

- ❖ 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)



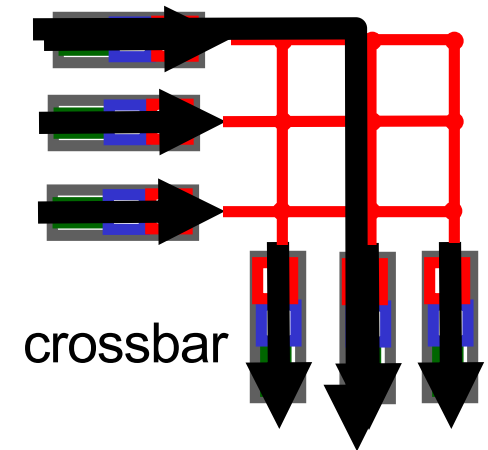
# Switching via a bus

- ❖ 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 and enterprise routers



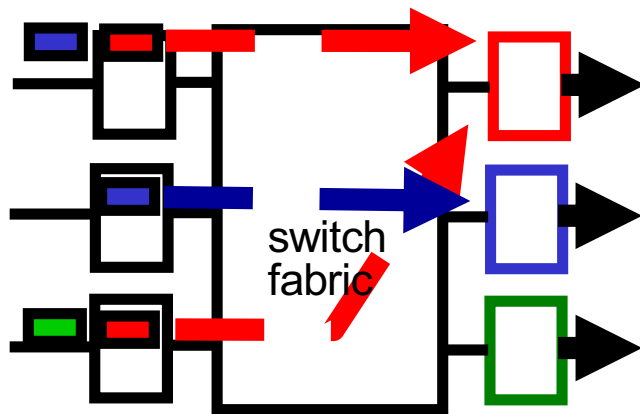
# Switching via interconnection network

- ❖ overcome bus bandwidth limitations
- ❖ banyan networks, crossbar, other interconnection nets initially developed to connect processors in multiprocessor
- ❖ advanced design: fragmenting datagram into fixed length cells, switch cells through the fabric.
- ❖ Cisco I2000: switches 60 Gbps through the interconnection network



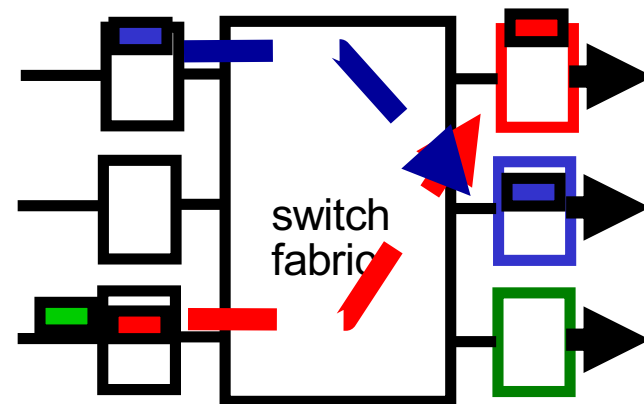
# Input port queuing

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



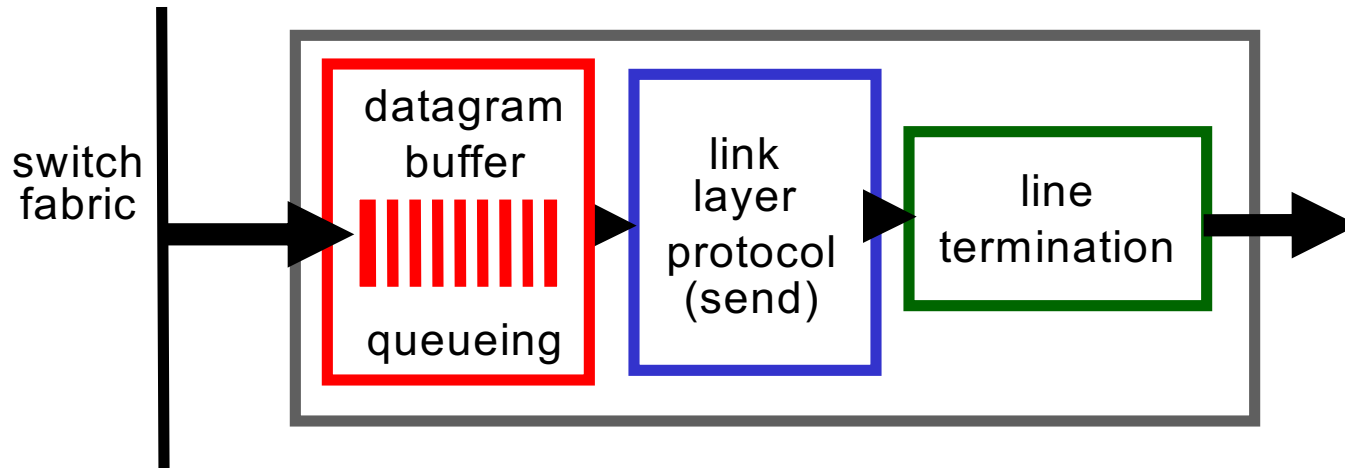
output port contention:  
only one red datagram can be  
transferred.

*lower red packet is blocked*



one packet time later:  
green packet  
experiences HOL  
blocking

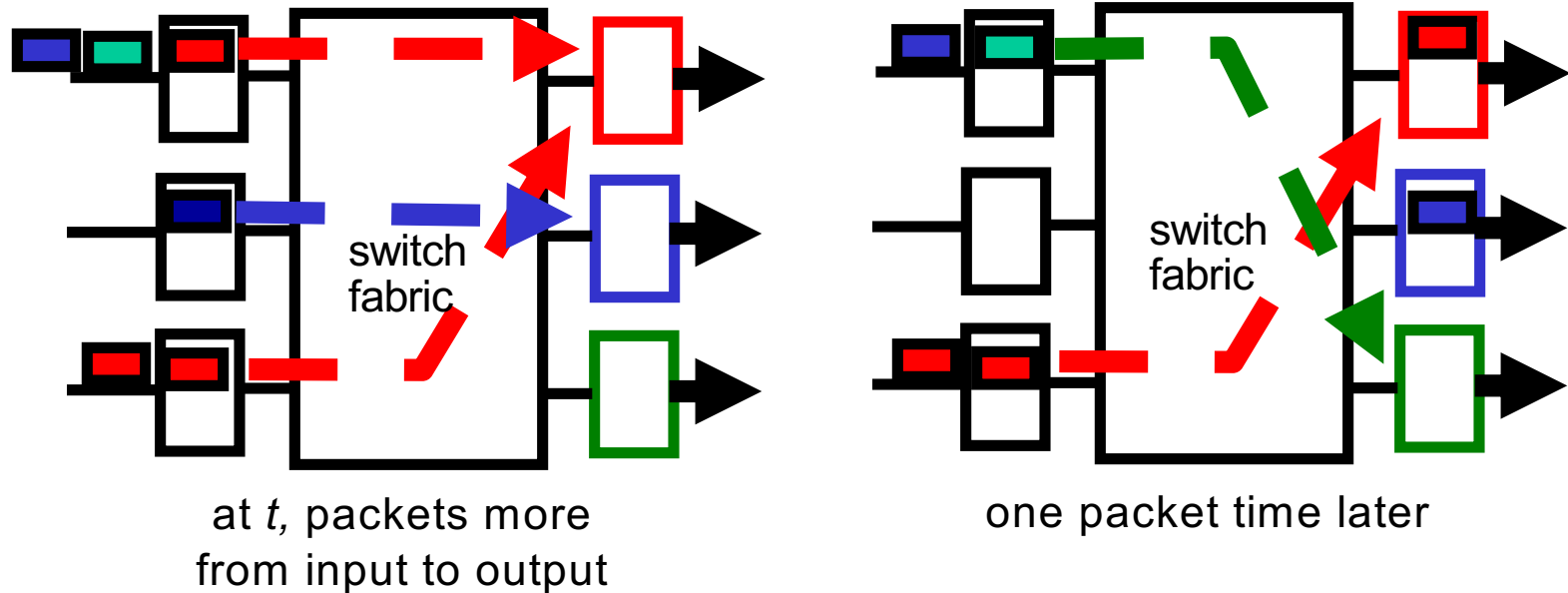
# Output ports



- ❖ *buffering* required when datagrams arrive from fabric faster than the transmission rate
- ❖ *scheduling discipline* chooses among queued datagrams for transmission



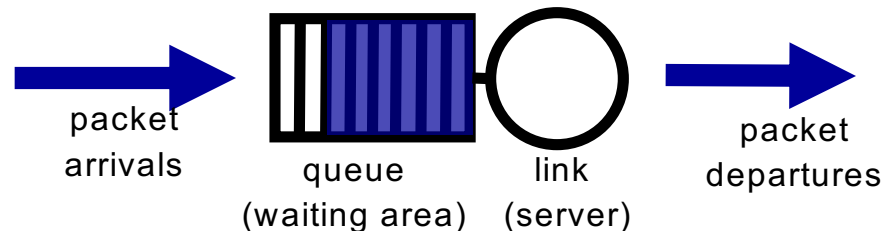
# Output port queueing



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

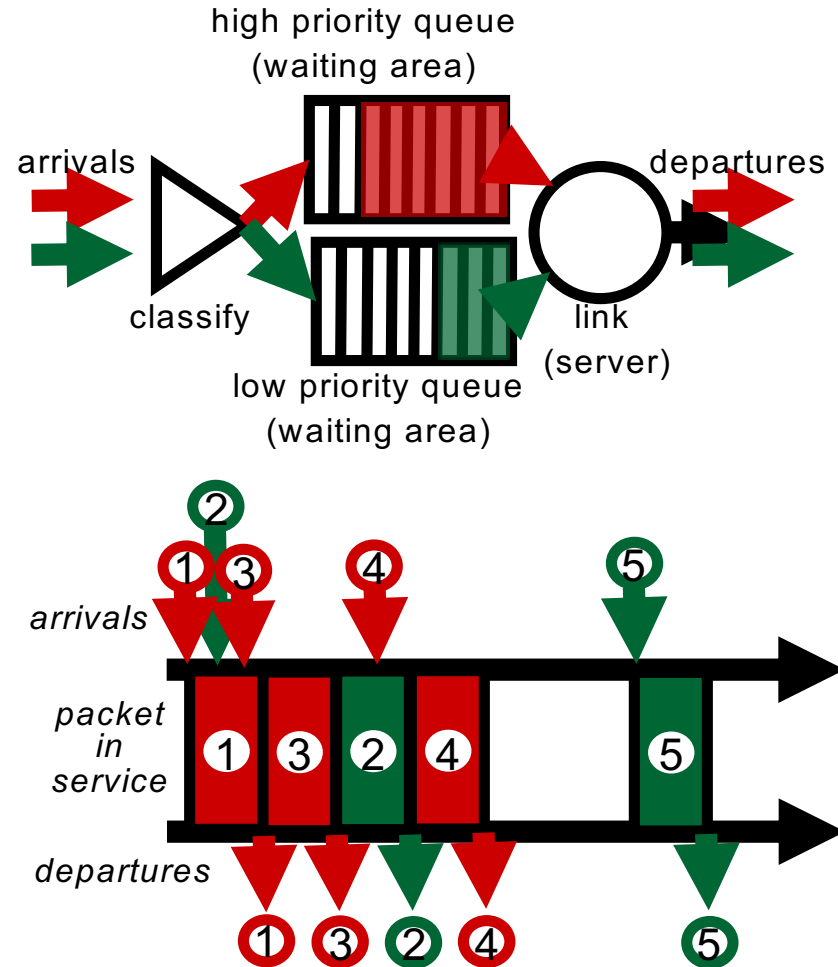
# Scheduling mechanisms

- ❖ *scheduling*: choose next packet to send on link
- ❖ *FIFO (first in first out) scheduling*: send in order of arrival to queue
  - real-world example?
  - *discard policy*: if packet arrives to full queue: who to discard?
    - *tail drop*: drop arriving packet
    - *priority*: drop/remove on priority basis
    - *random*: drop/remove randomly



# Scheduling policies: priority

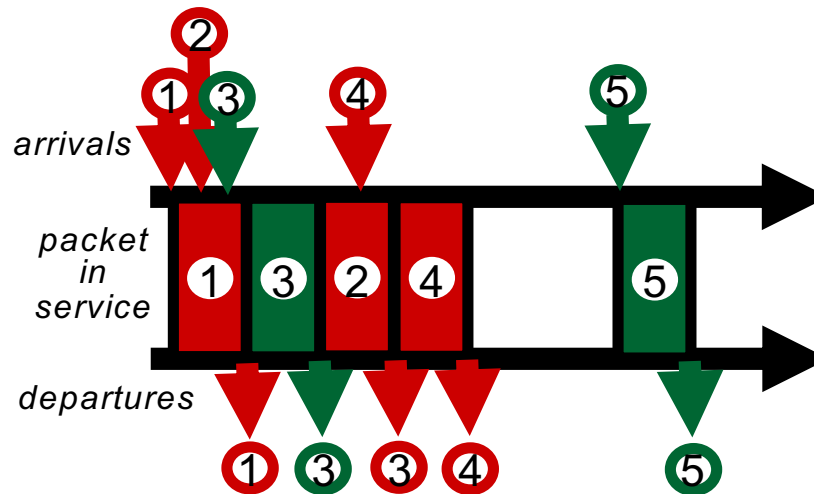
- priority scheduling*: send highest priority queued packet
- ❖ multiple *classes*, with different priorities
    - class may depend on marking or other header info, e.g. IP source/dest, port numbers, etc.
    - real world example?



# Scheduling policies: still more

## *Round Robin (RR) scheduling:*

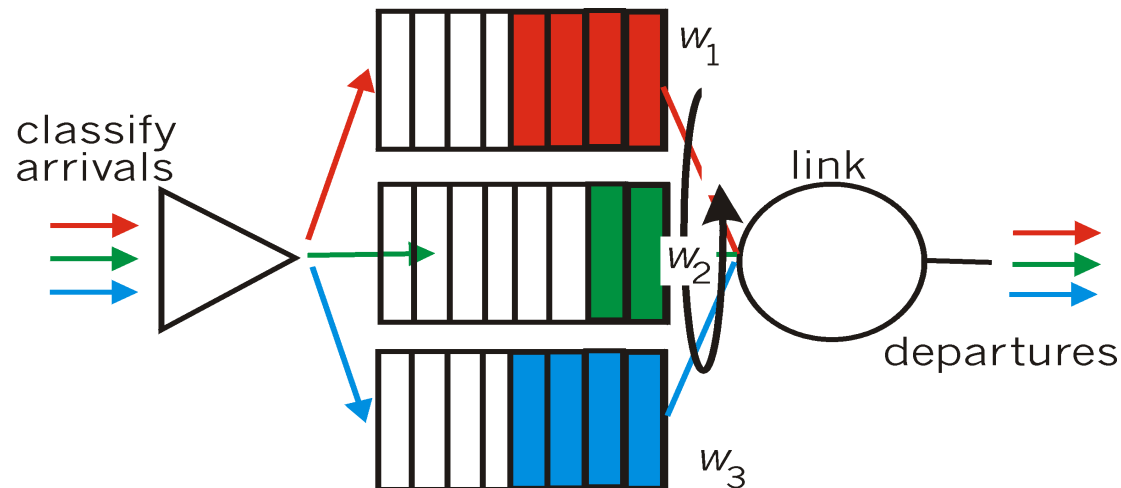
- ❖ multiple classes
- ❖ cyclically scan class queues, sending one complete packet from each class (if available)
- ❖ real world example?



# Scheduling policies: still more

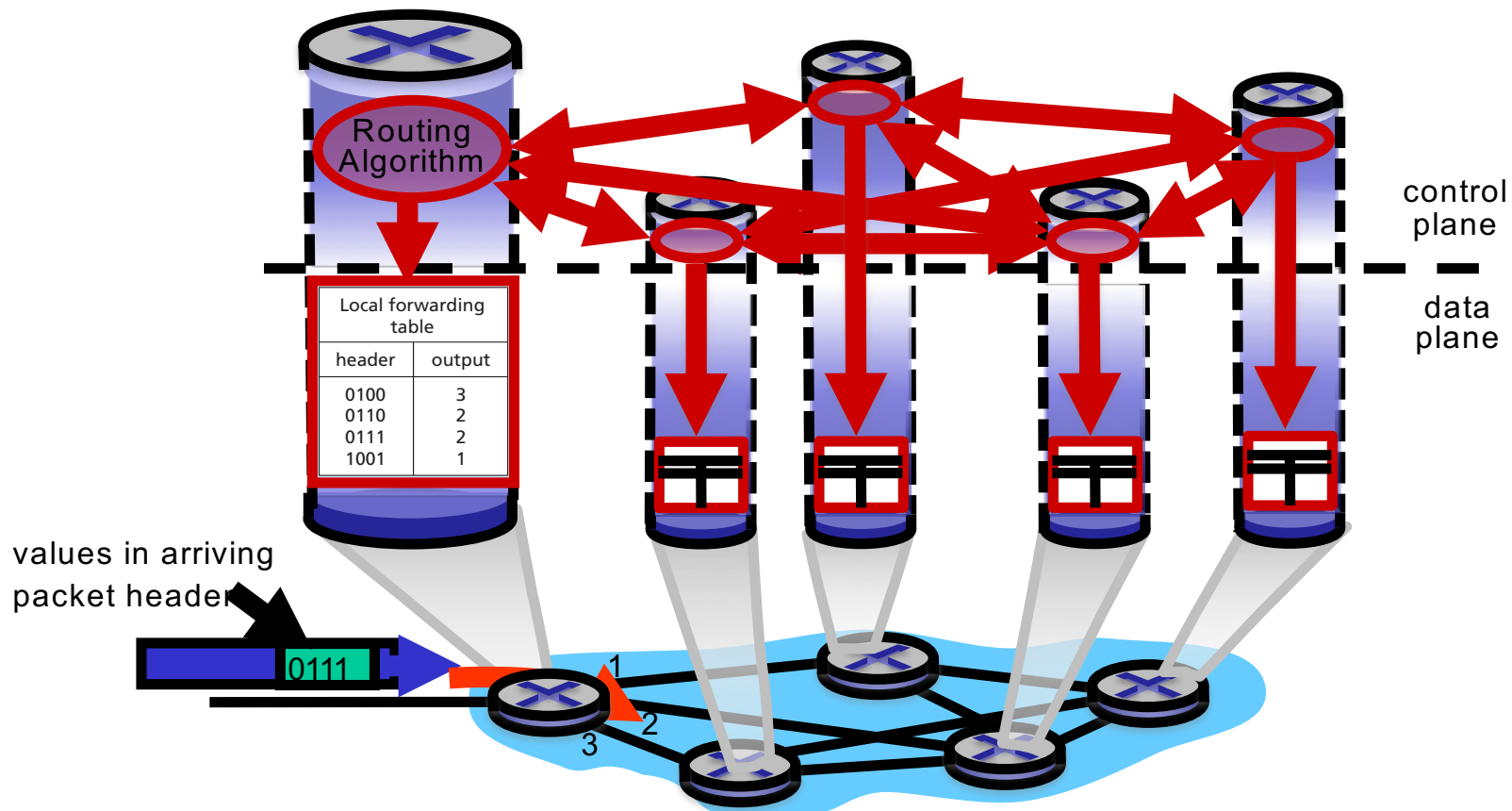
## *Weighted Fair Queuing (WFQ):*

- ❖ generalized Round Robin
- ❖ each class gets weighted amount of service in each cycle
- ❖ real-world example?



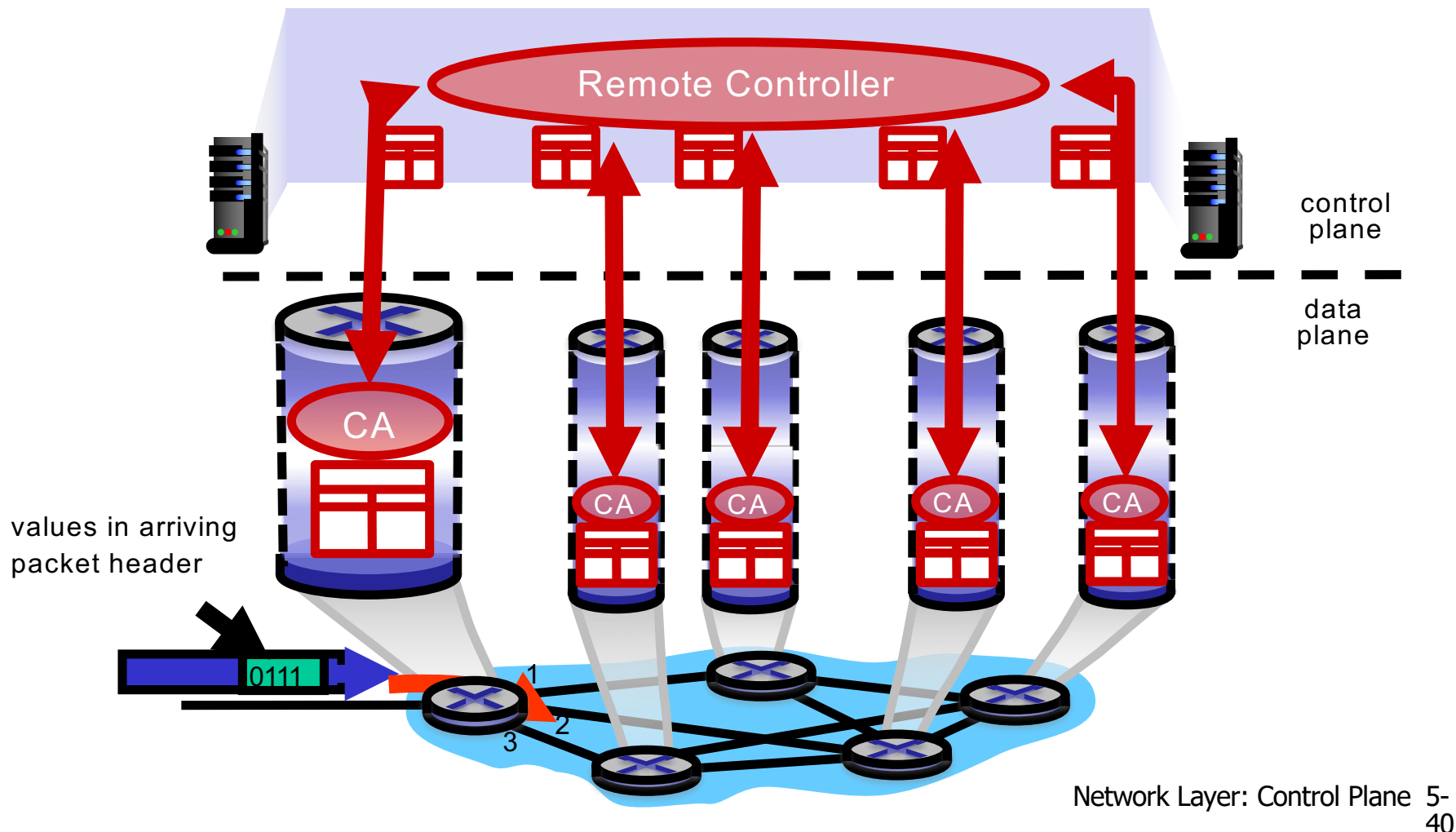
# Per-router control plane

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



# Logically centralized control plane

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



# Chapter 4: outline

## 4.1 introduction

## 4.2 virtual circuit and datagram networks

## 4.3 what's inside a router

## 4.4 IP: Internet Protocol

- datagram format
- IPv4 addressing
- ICMP
- IPv6

## 4.5 routing algorithms

- link state
- distance vector
- hierarchical routing

## 4.6 routing in the Internet

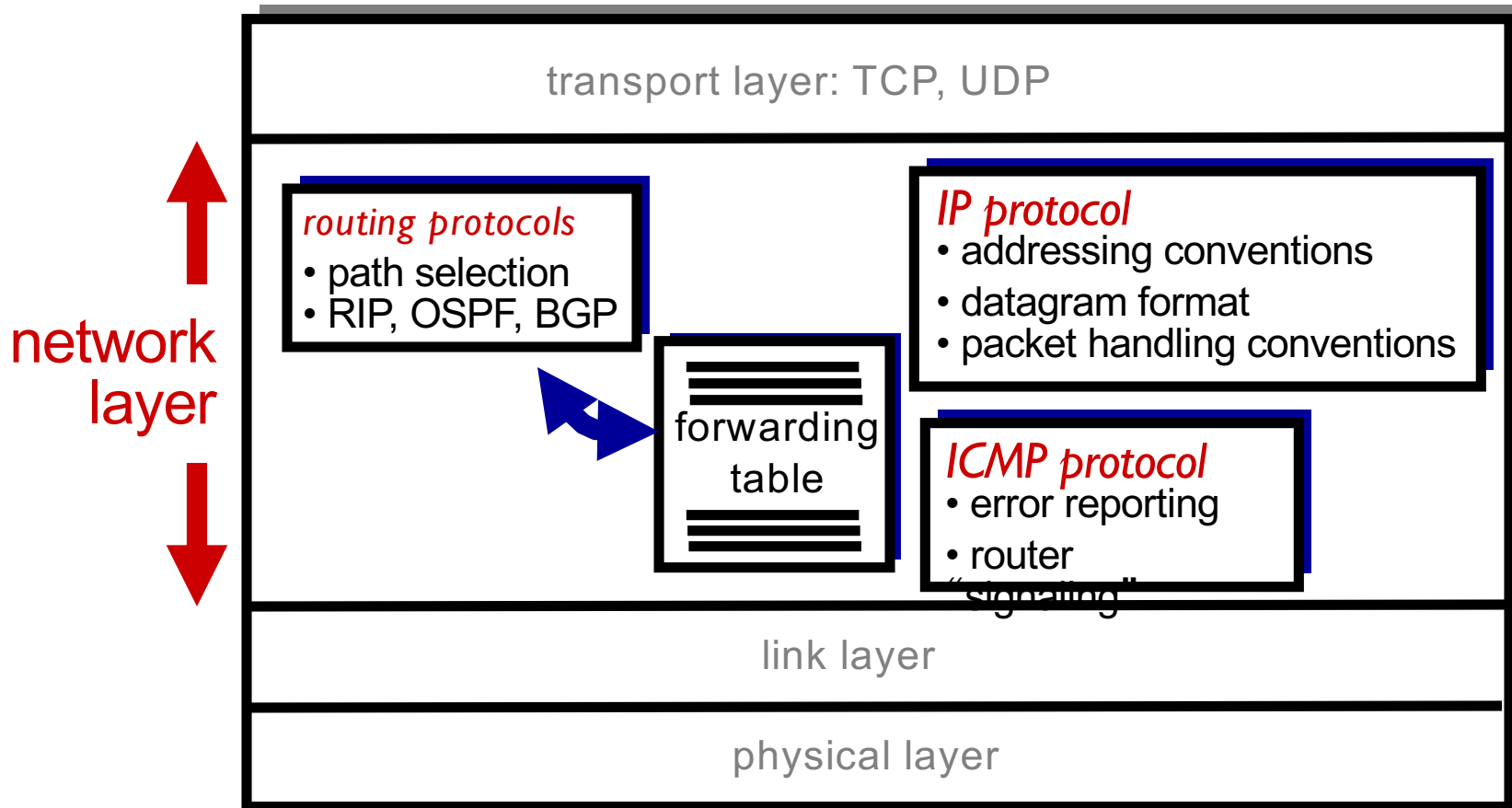
- RIP
- OSPF
- BGP

## 4.7 broadcast and multicast routing

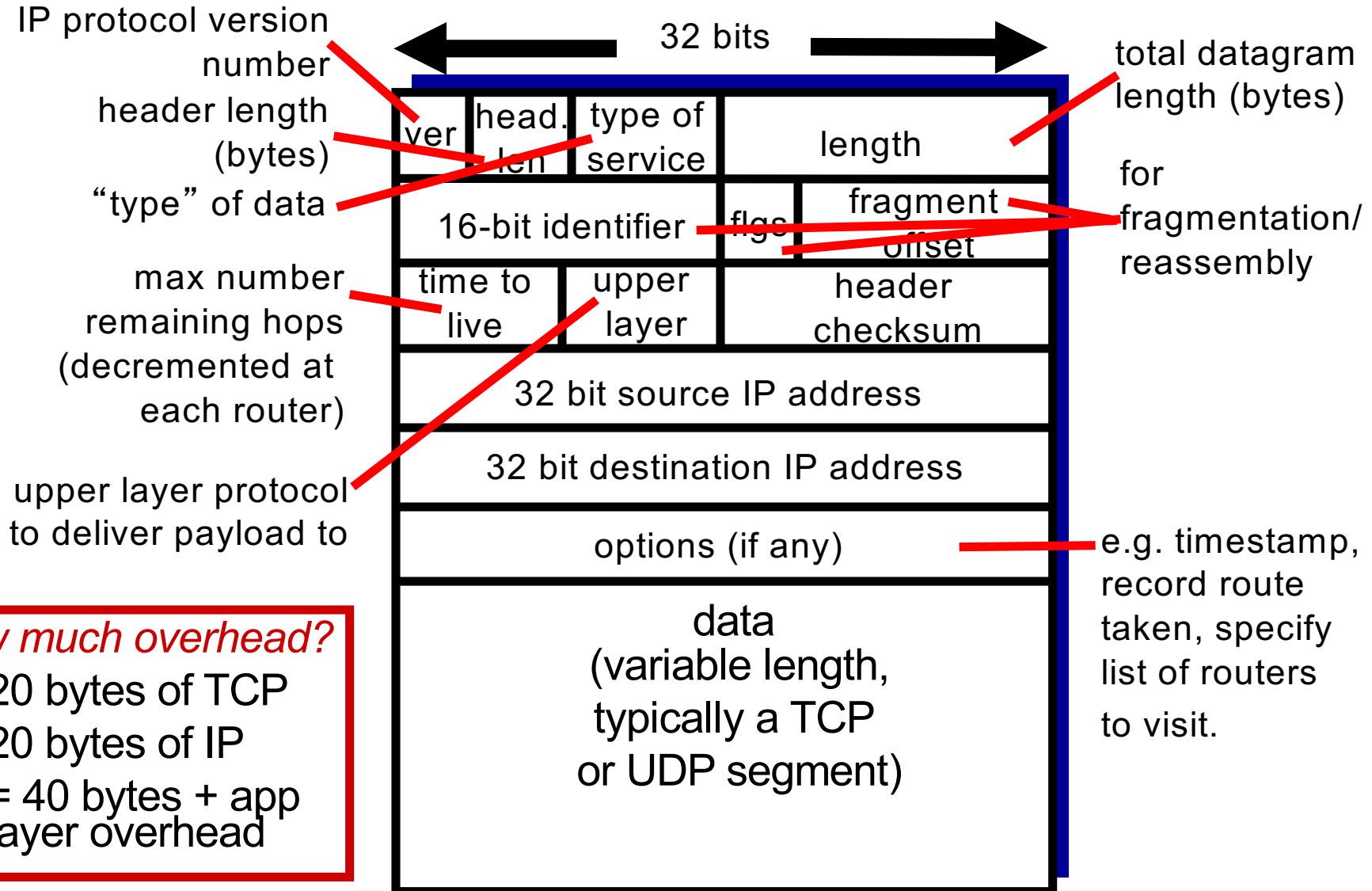


# The Internet network layer

host, router network layer functions:

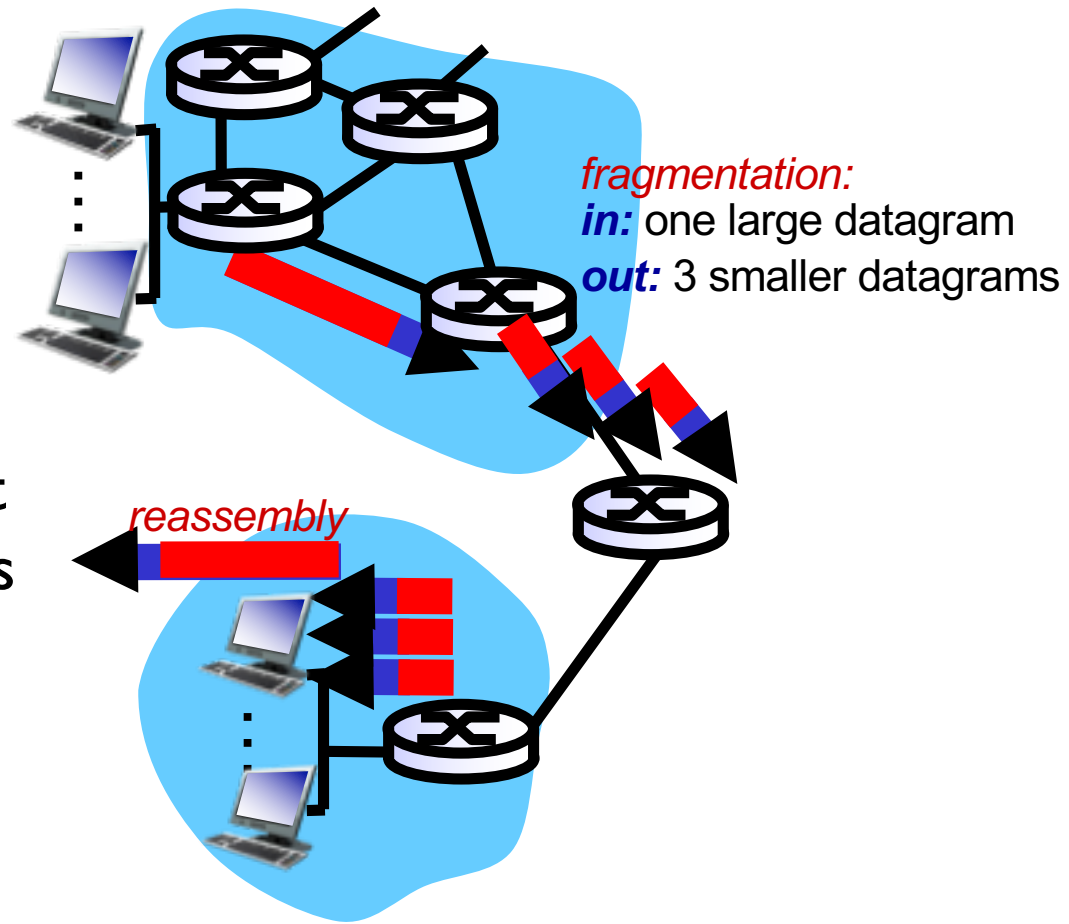


# IP datagram format



# IP fragmentation, reassembly

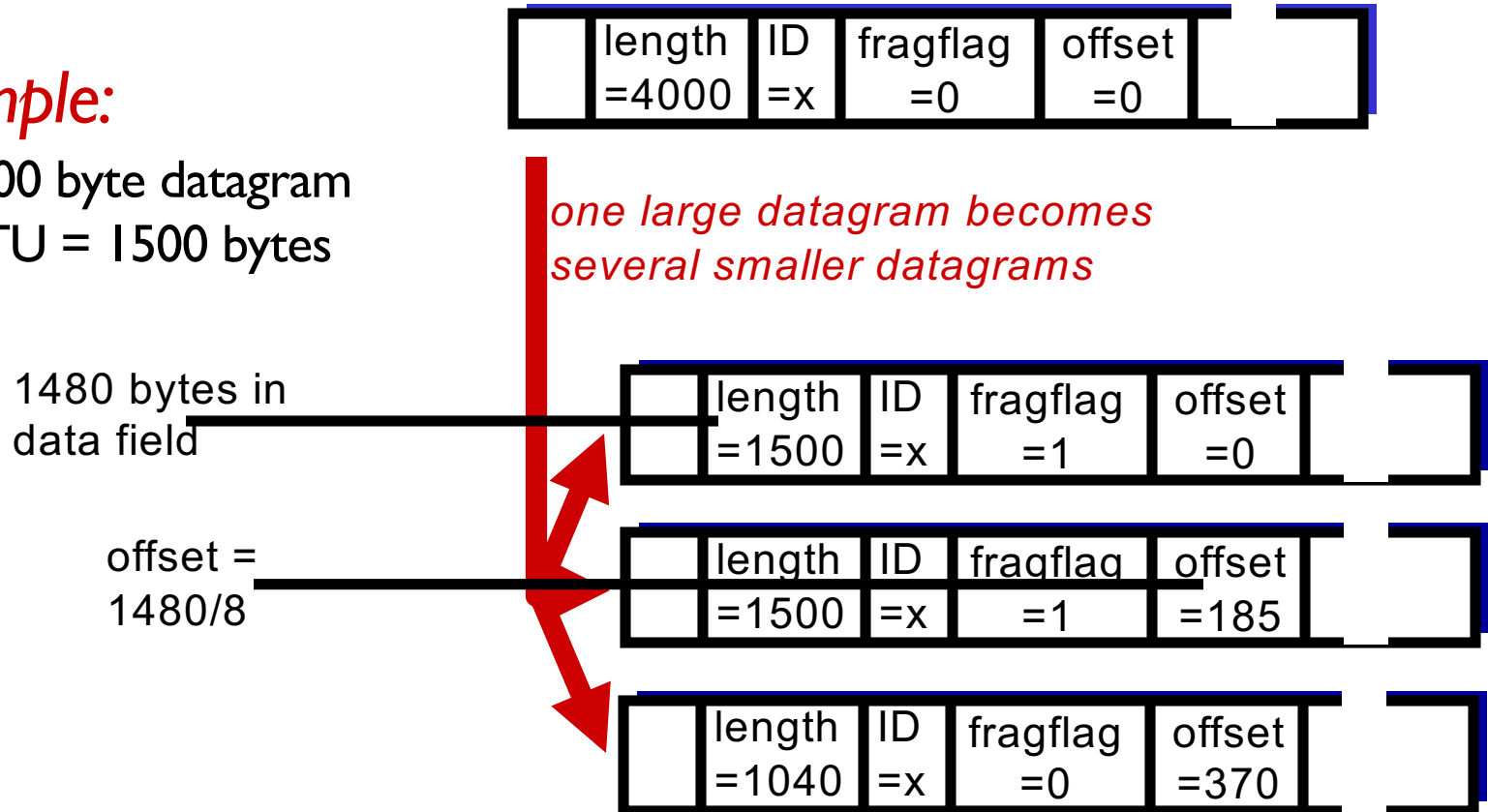
- ❖ network links have MTU (max.transfer size) - largest possible link-level frame
  - different link types, different MTUs
- ❖ large IP datagram divided (“fragmented”) within net
  - one datagram becomes several datagrams
  - “reassembled” only at final destination
  - IP header bits used to identify,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



# IP fragmentation, reassembly

## *example:*

- ❖ 4000 byte datagram
- ❖ MTU = 1500 bytes



# Chapter 4: outline

## 4.1 introduction

## 4.2 virtual circuit and datagram networks

## 4.3 what's inside a router

## 4.4 IP: Internet Protocol

- datagram format
- IPv4 addressing
- ICMP
- IPv6

## 4.5 routing algorithms

- link state
- distance vector
- hierarchical routing

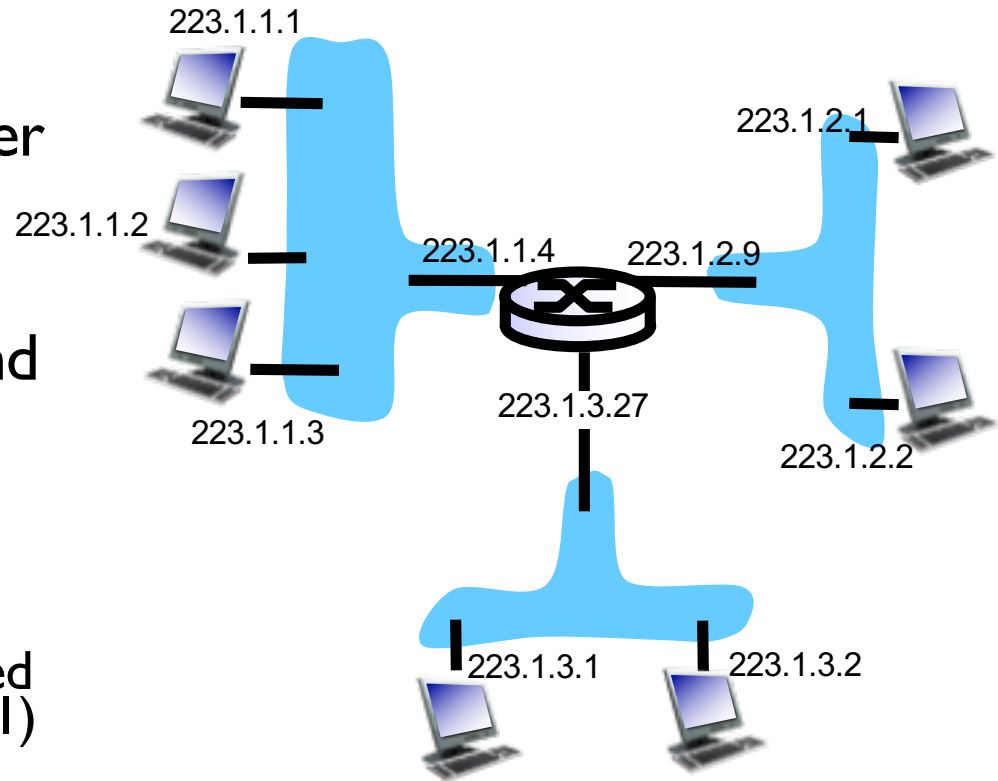
## 4.6 routing in the Internet

- RIP
- OSPF
- BGP

## 4.7 broadcast and multicast routing

# IP addressing: introduction

- ❖ **IP address:** 32-bit identifier for host, router interface
- ❖ **interface:** connection between host/router and physical link
  - router's typically have multiple interfaces
  - host typically has one or two interfaces (e.g., wired Ethernet, wireless 802.11)
- ❖ **IP addresses associated with each interface**



$$223.1.1.1 = \underbrace{11011111}_{223} \underbrace{00000001}_1 \underbrace{00000001}_1 \underbrace{00000001}_1$$

# IP addressing: introduction

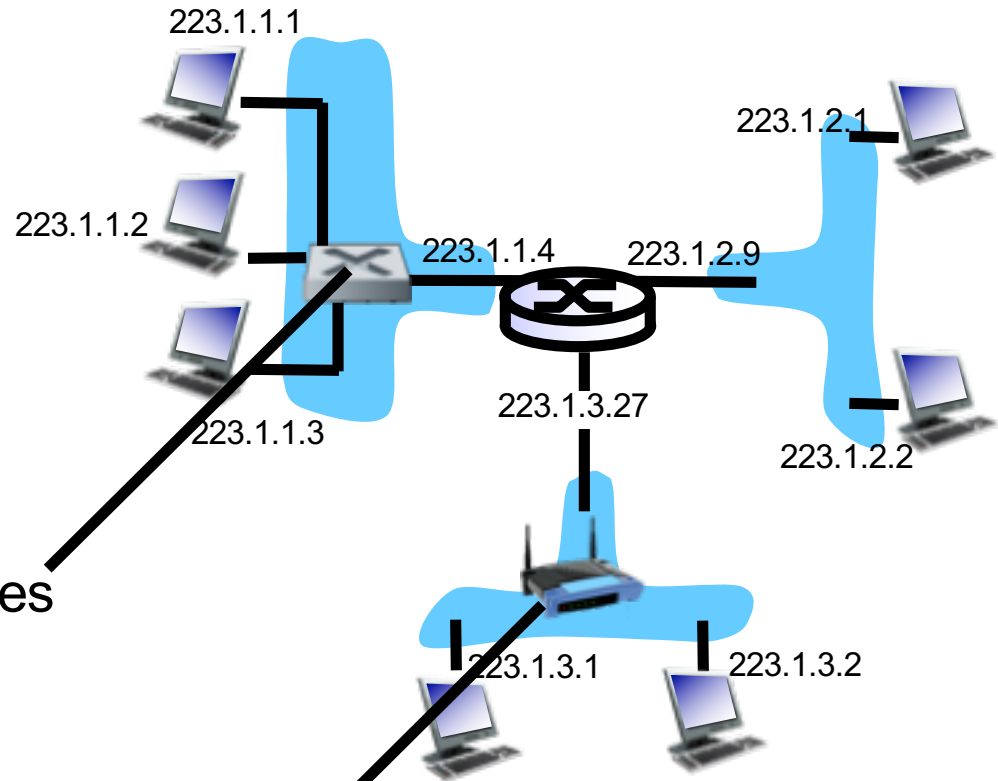
*Q: how are interfaces actually connected?*

*A: we'll learn about that in chapter 5, 6.*

*A:* wired Ethernet interfaces connected by Ethernet switches

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

*A:* wireless WiFi interfaces connected by WiFi base station



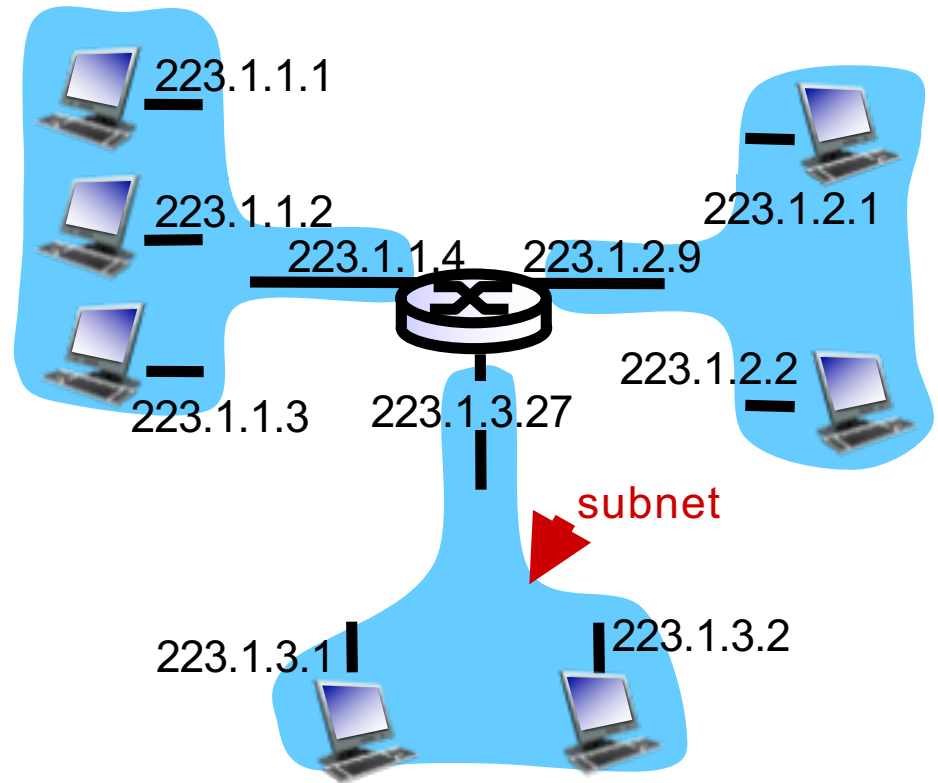
# Subnets

## ❖ IP address:

- subnet part - high order bits
- host part - low order bits

## ❖ *what's a subnet?*

- device interfaces with same subnet part of IP address
- can physically reach each other *without intervening router*



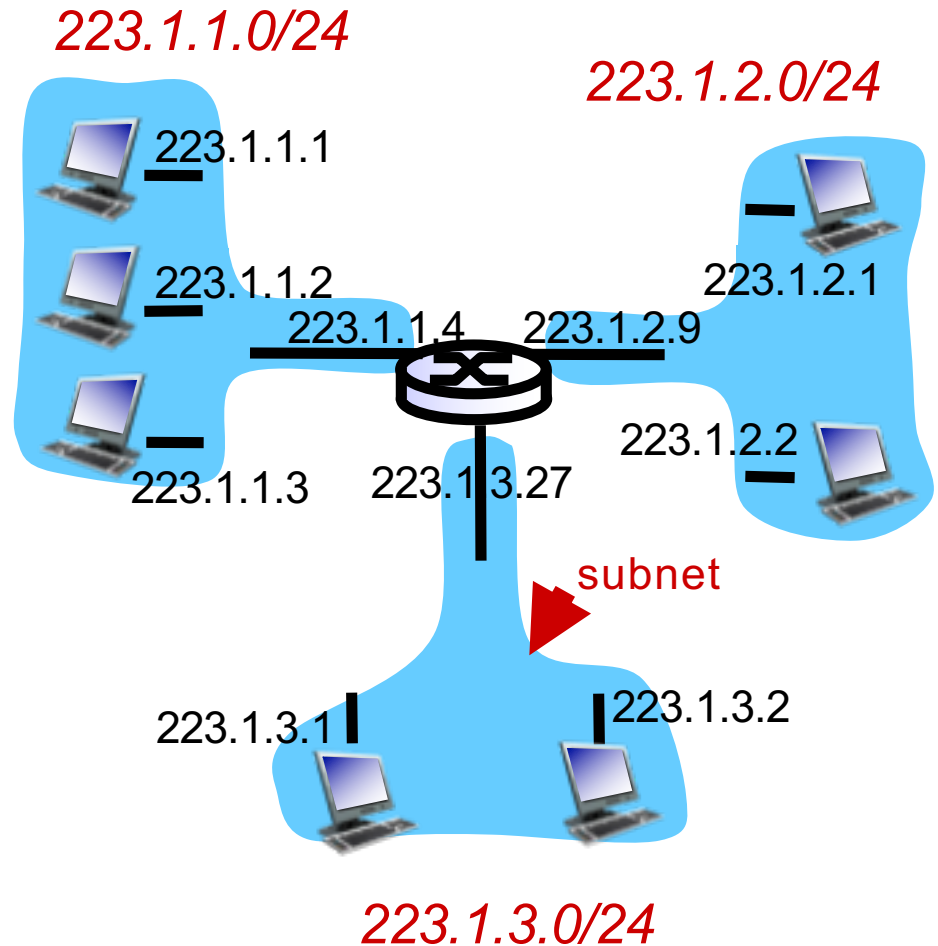
network consisting of 3 subnets



# Subnets

## *recipe*

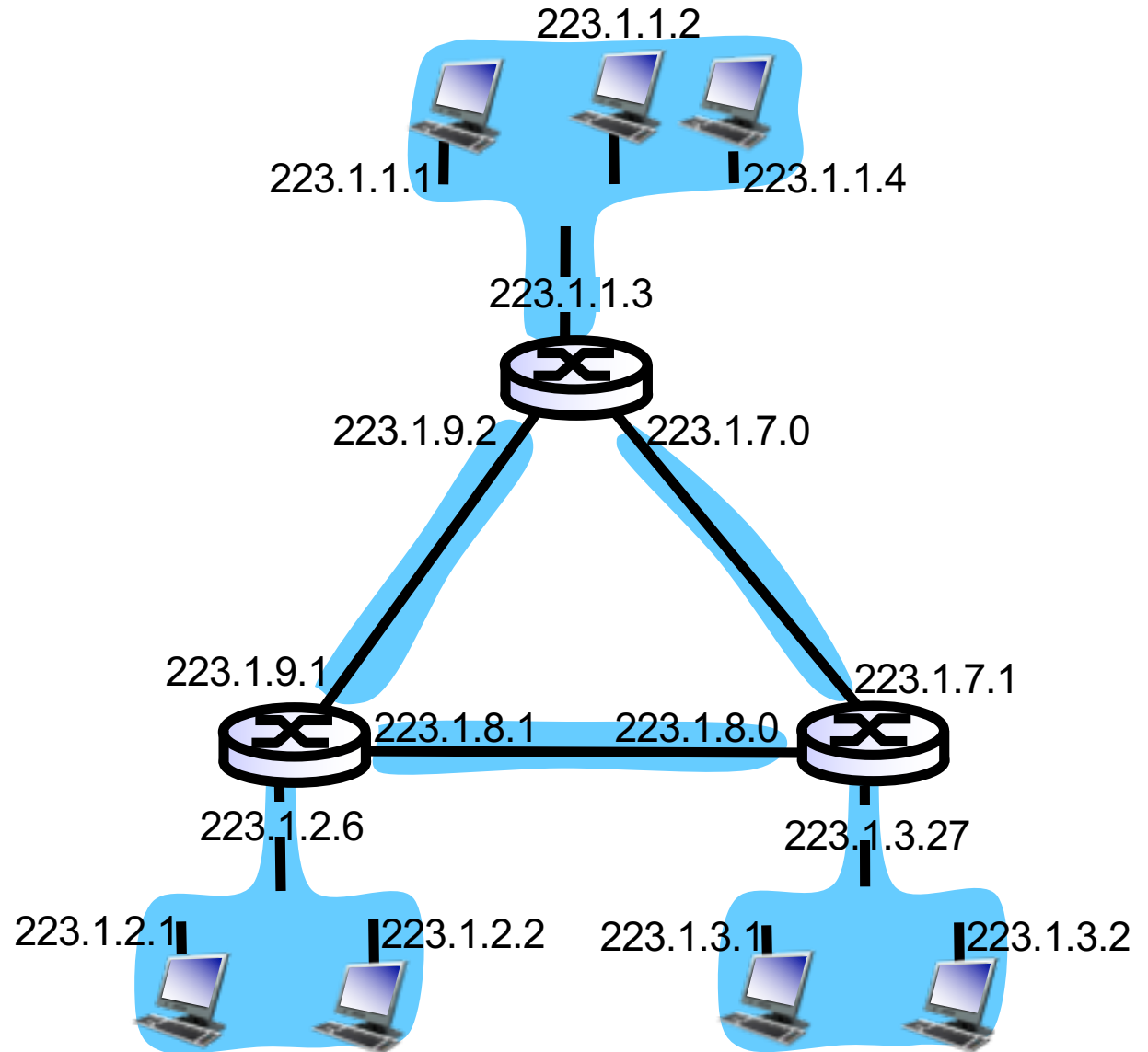
- ❖ to determine the subnets, detach each interface from its host or router, creating islands of isolated networks
- ❖ each isolated network is called a *subnet*



subnet mask: /24

# Subnets

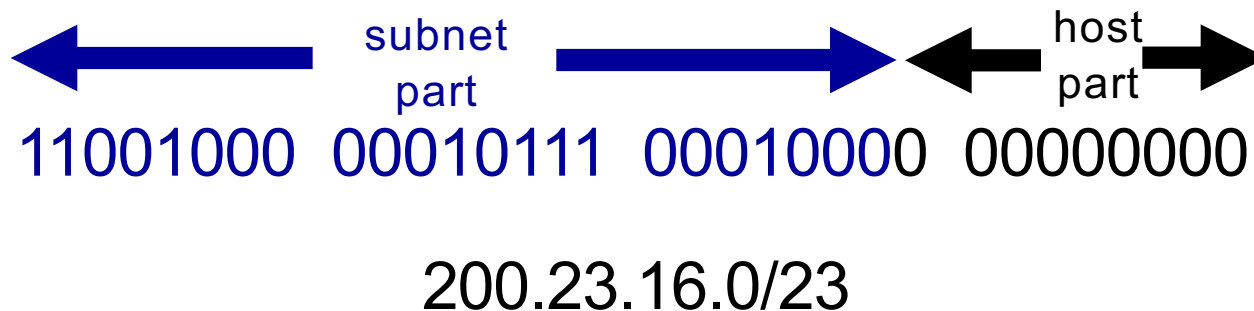
how many?



# IP addressing: CIDR

## CIDR: Classless InterDomain Routing

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



# IP addresses: how to get one?

**Q:** How does a *host* get IP address?

- ❖ hard-coded by system admin in a file
  - Windows: control-panel->network->configuration->tcp/ip->properties
  - UNIX: /etc/rc.config
- ❖ **DHCP: Dynamic Host Configuration Protocol:** dynamically get address from as server
  - “plug-and-play”

# DHCP: Dynamic Host Configuration Protocol

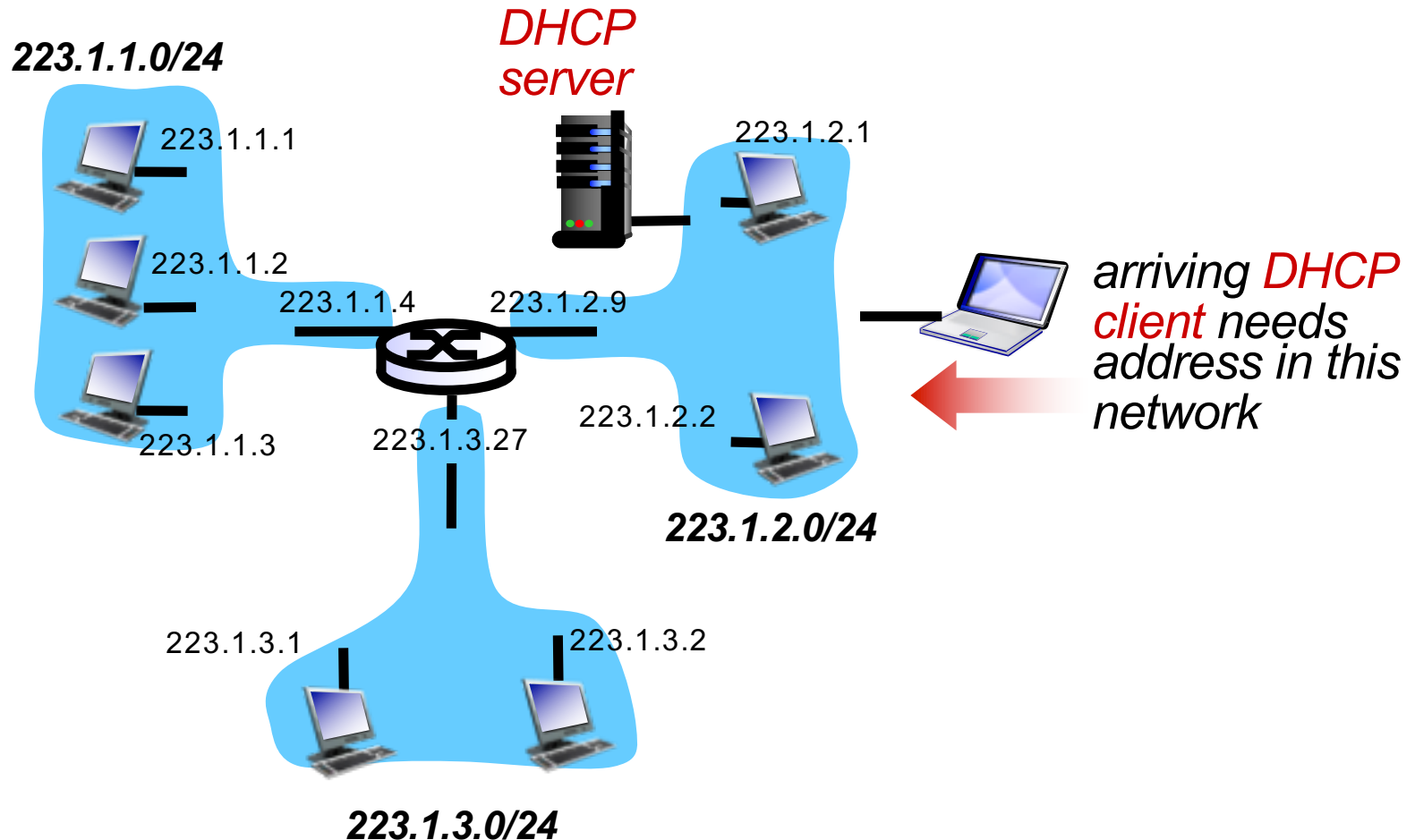
*goal:* allow host to *dynamically* obtain its IP address from network server when it joins network

- can renew its lease on address in use
- allows reuse of addresses (only hold address while connected/“on”)
- support for mobile users who want to join network (more shortly)

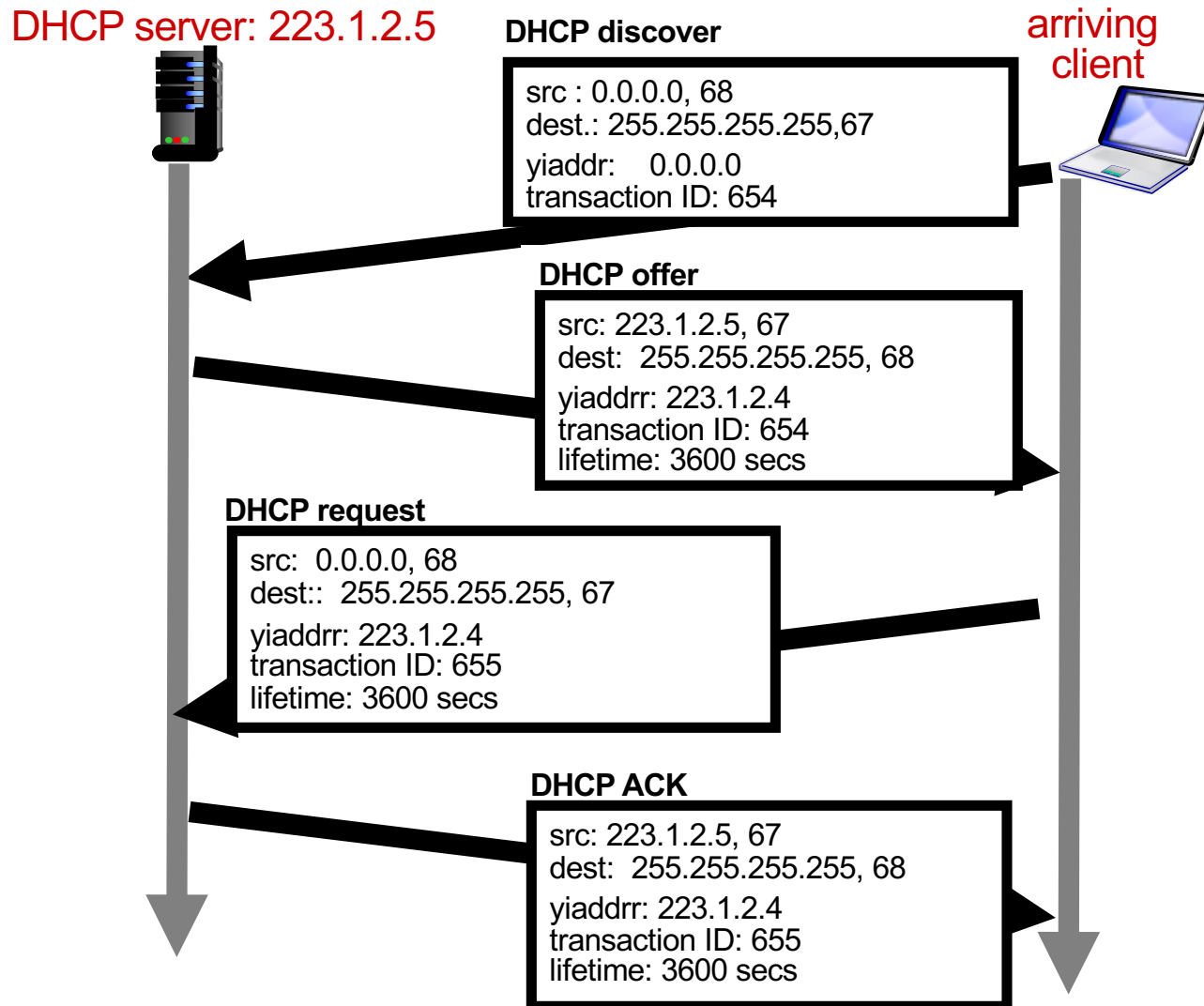
## *DHCP overview:*

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

# DHCP client-server scenario



# DHCP client-server scenario



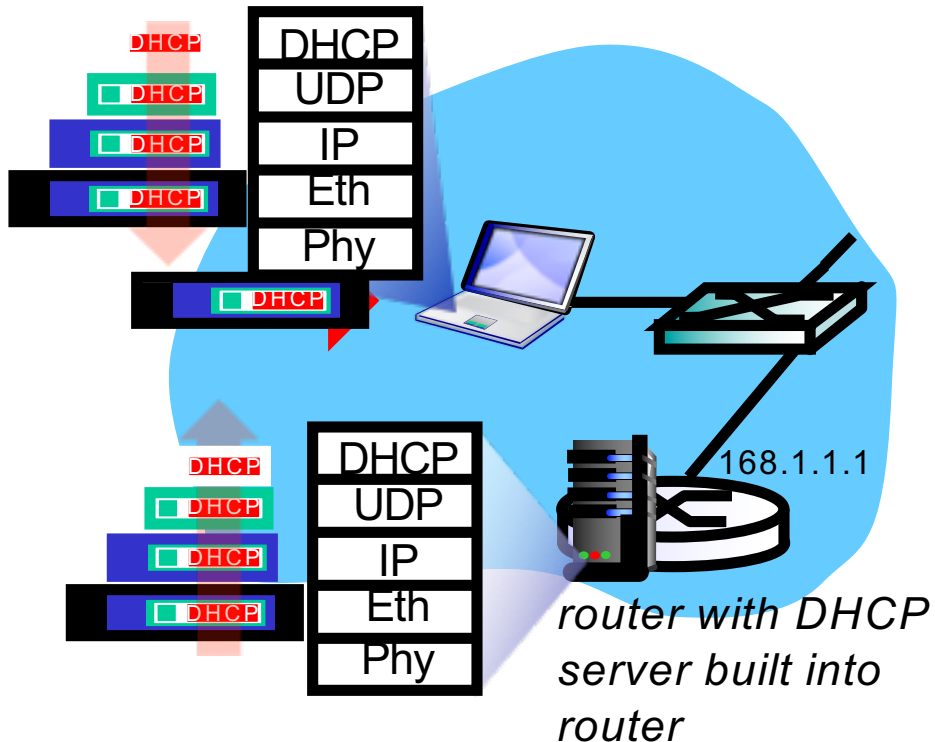
# DHCP: more than IP addresses

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)

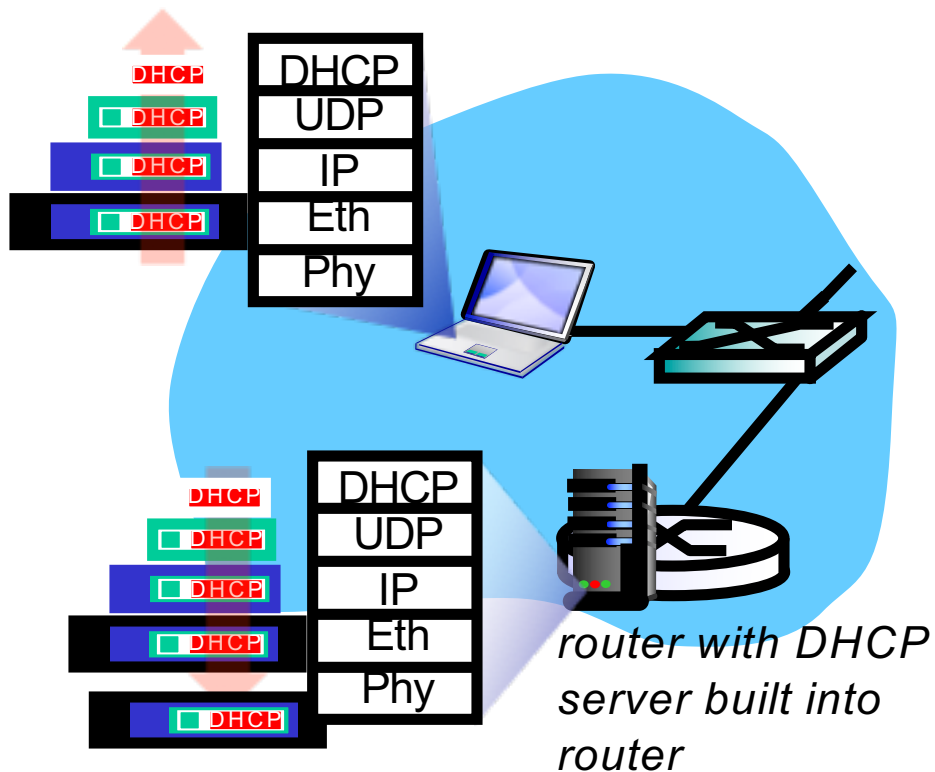


# DHCP: example



- ❖ connecting laptop needs its IP address, addr of first-hop router, addr of DNS server: use DHCP
- ❖ DHCP request encapsulated in UDP, encapsulated in IP, encapsulated in 802.1 Ethernet
- ❖ Ethernet frame broadcast (dest: FFFFFFFFFFFFFFFF) on LAN, received at router running DHCP server
- ❖ Ethernet demuxed to IP demuxed, UDP demuxed 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
- ❖ encapsulation of DHCP server, frame forwarded to client, demuxing up to DHCP at client
- ❖ client now knows its IP address, name and IP address of DSN server, IP address of its first-hop router

# IP addresses: how to get one?

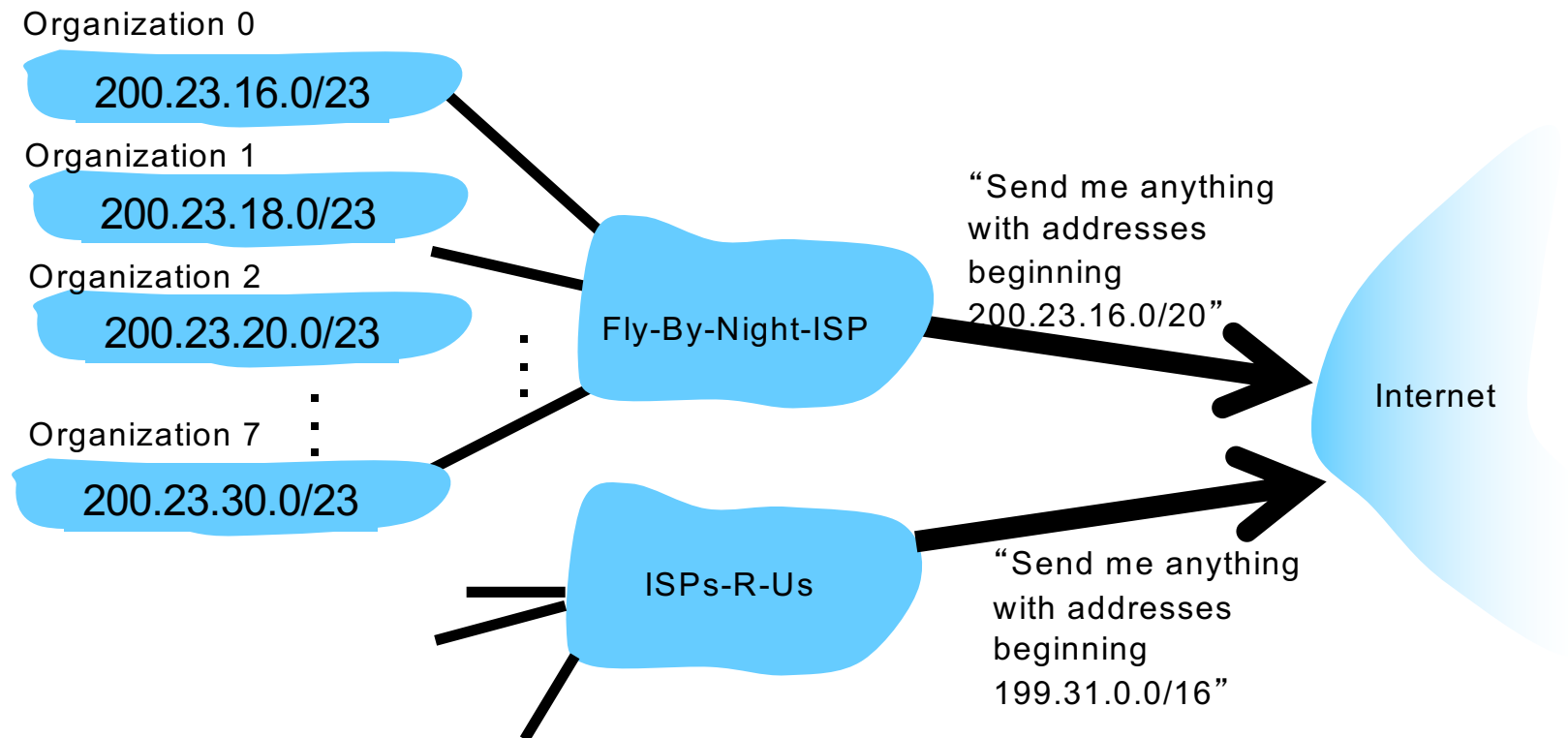
**Q:** how does *network* get subnet part of IP addr?

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

ISP's block	<u>11001000 00010111 00010000</u> 00000000	200.23.16.0/20
Organization 0	<u>11001000 00010111 00010000</u> 00000000	200.23.16.0/23
Organization 1	<u>11001000 00010111 00010010</u> 00000000	200.23.18.0/23
Organization 2	<u>11001000 00010111 00010100</u> 00000000	200.23.20.0/23
...	.....	....
Organization 7	<u>11001000 00010111 00011110</u> 00000000	200.23.30.0/23

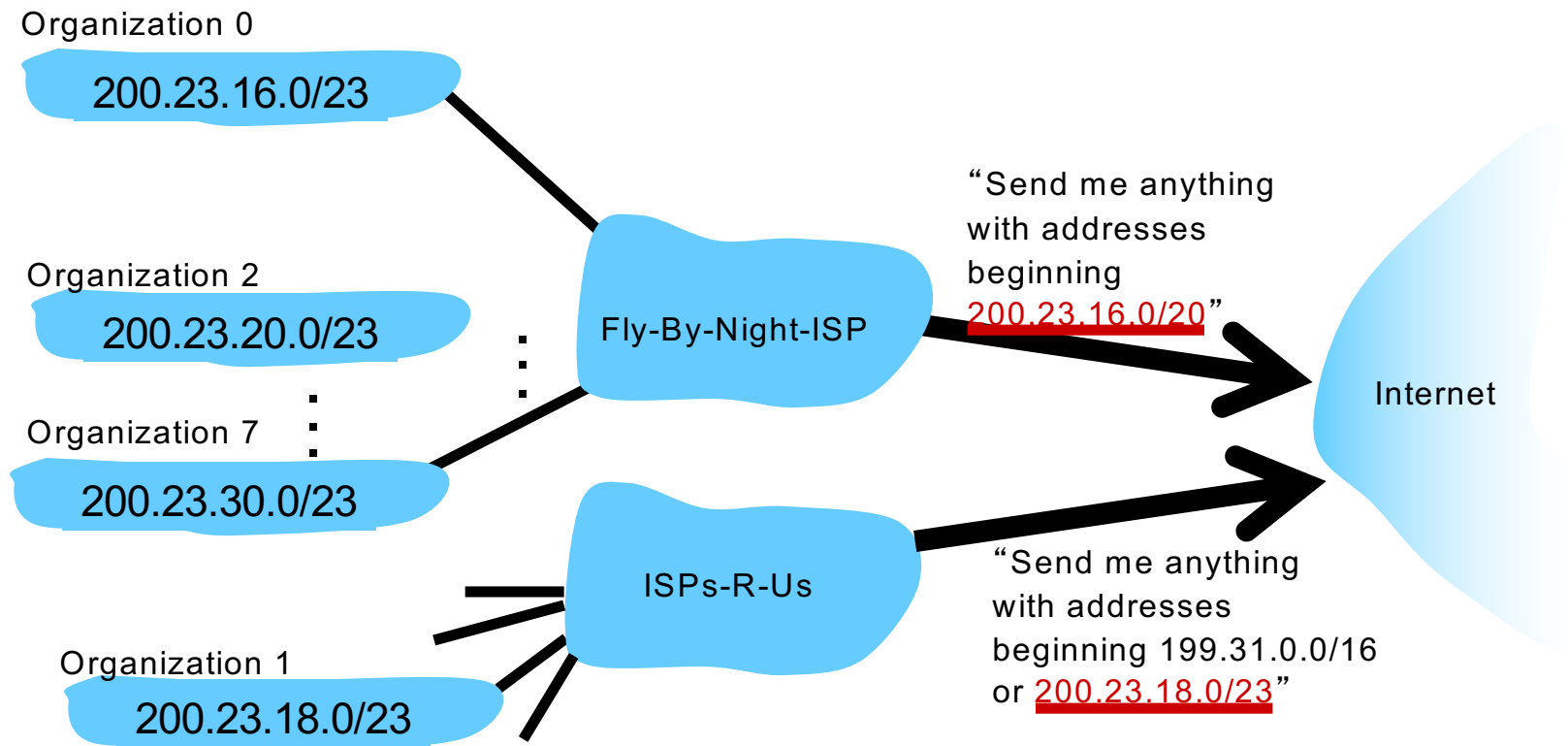
# Hierarchical addressing: route aggregation

hierarchical addressing allows efficient advertisement of routing information:



# Hierarchical addressing: more specific routes

ISPs-R-Us has a more specific route to Organization 1



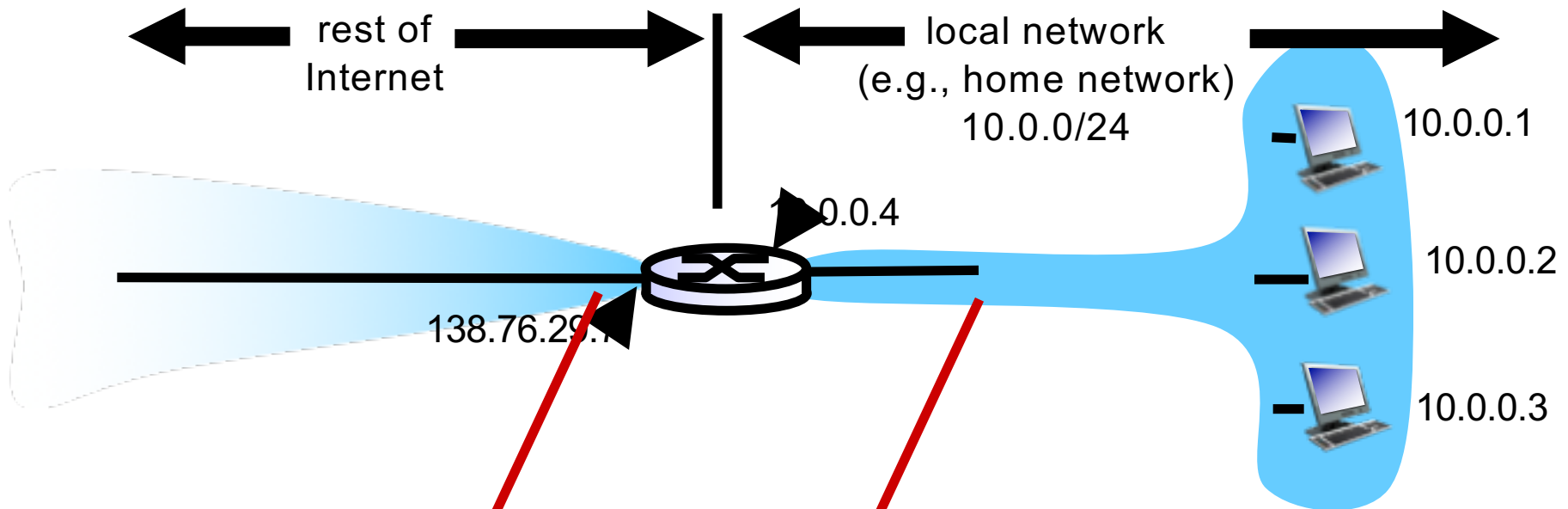
# IP addressing: the last word...

**Q:** how does an ISP get block of addresses?

**A: ICANN:** Internet Corporation for Assigned Names and Numbers <http://www.icann.org/>

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

# NAT: network address translation



*all* datagrams *leaving* local network have *same* single source NAT IP address: 138.76.29.7, different source port numbers

datagrams with source or destination in this network have 10.0.0/24 address for source, destination (as usual)

# NAT: network address translation

*motivation:* local network uses just one IP address as far as outside world is concerned:

- range of addresses not needed from ISP: just one IP address for all devices
- can change addresses of devices in local network without notifying outside world
- can change ISP without changing addresses of devices in local network
- devices inside local net not explicitly addressable, visible by outside world (a security plus)

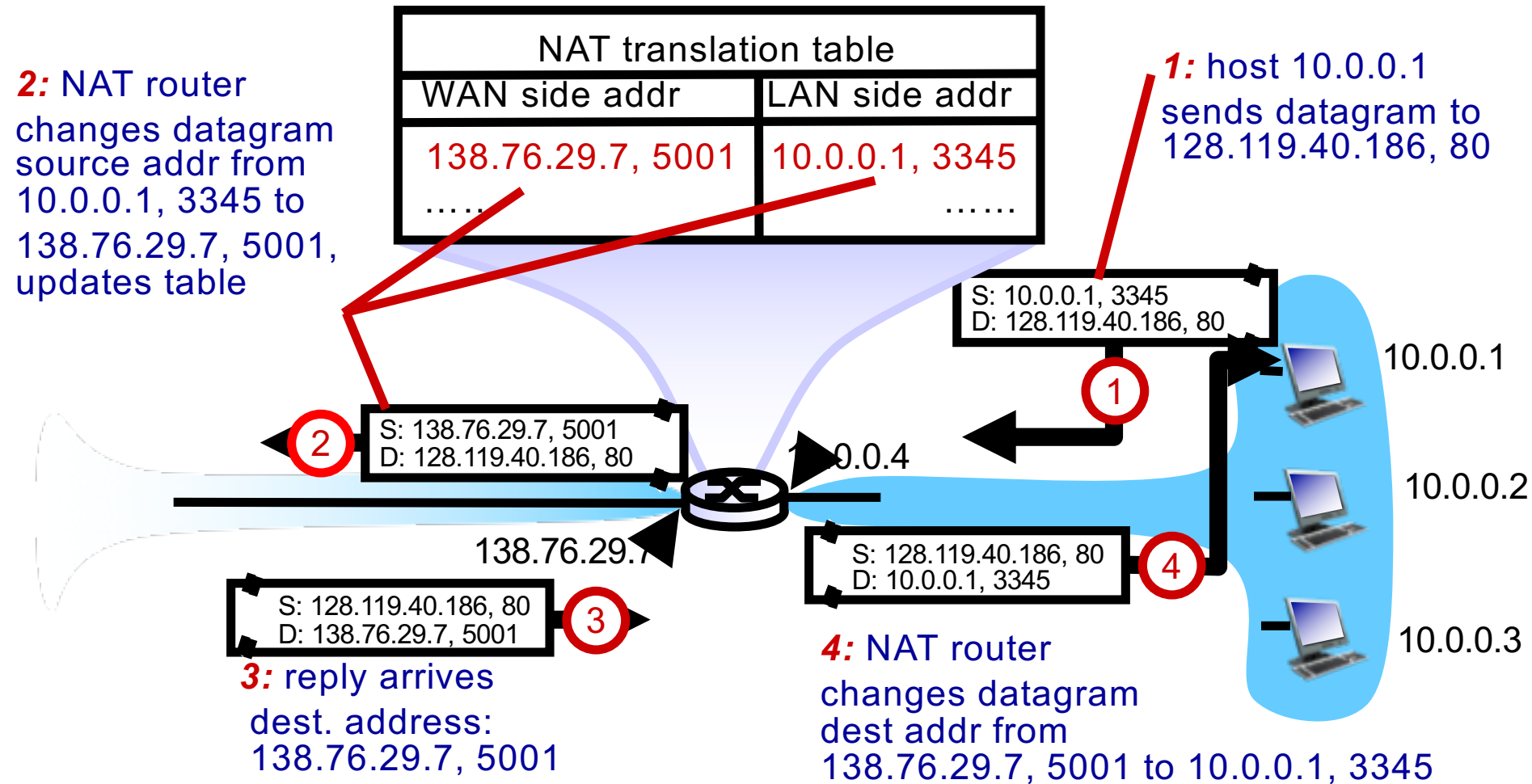


# NAT: network address translation

*implementation:* NAT router must:

- *outgoing datagrams: replace* (source IP address, port #) of every outgoing datagram to (NAT IP address, new port #) . . . remote clients/servers will respond using (NAT IP address, new port #) as destination addr
- *remember (in NAT translation table)* every (source IP address, port #) to (NAT IP address, new port #) translation pair
- *incoming datagrams: replace* (NAT IP address, new port #) in dest fields of every incoming datagram with corresponding (source IP address, port #) stored in NAT table

# NAT: network address translation

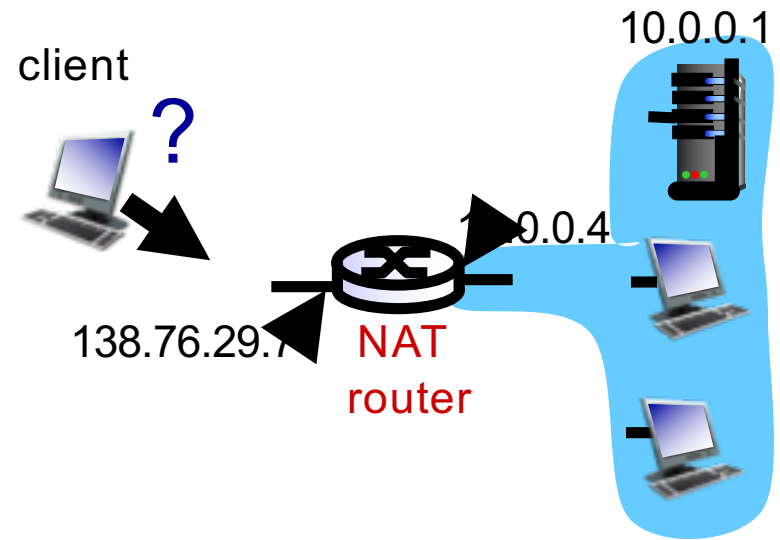


# NAT: network address translation

- ❖ 16-bit port-number field:
  - 60,000 simultaneous connections with a single LAN-side address!
- ❖ NAT is controversial:
  - routers should only process up to layer 3
  - violates end-to-end argument
    - NAT possibility must be taken into account by app designers, e.g., P2P applications
  - address shortage should instead be solved by IPv6
  - what if hosts behind the NAT are servers?

# NAT traversal problem

- ❖ client wants to connect to server with address 10.0.0.1
  - server address 10.0.0.1 local to LAN (client can't use it as destination addr)
  - only one externally visible NATed address: 138.76.29.7
- ❖ **solution 1:** statically configure NAT to forward incoming connection requests at given port to server
  - e.g., (138.76.29.7, port 2500) always forwarded to 10.0.0.1 port 25000

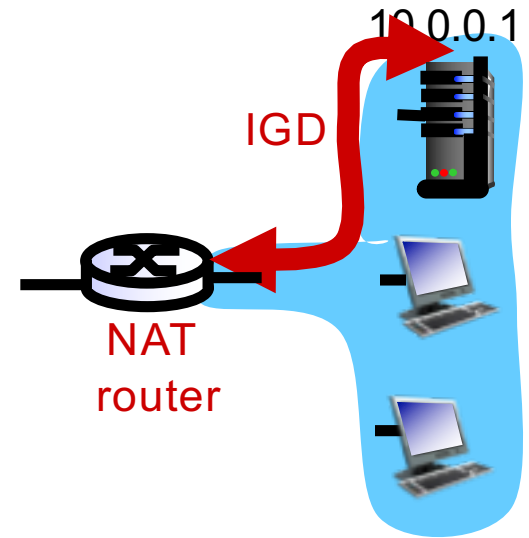


# NAT traversal problem

❖ *solution 2*: Universal Plug and Play (UPnP) Internet Gateway Device (IGD) Protocol. Allows NATed host to:

- ❖ learn public IP address (138.76.29.7)
- ❖ add/remove port mappings (with lease times)

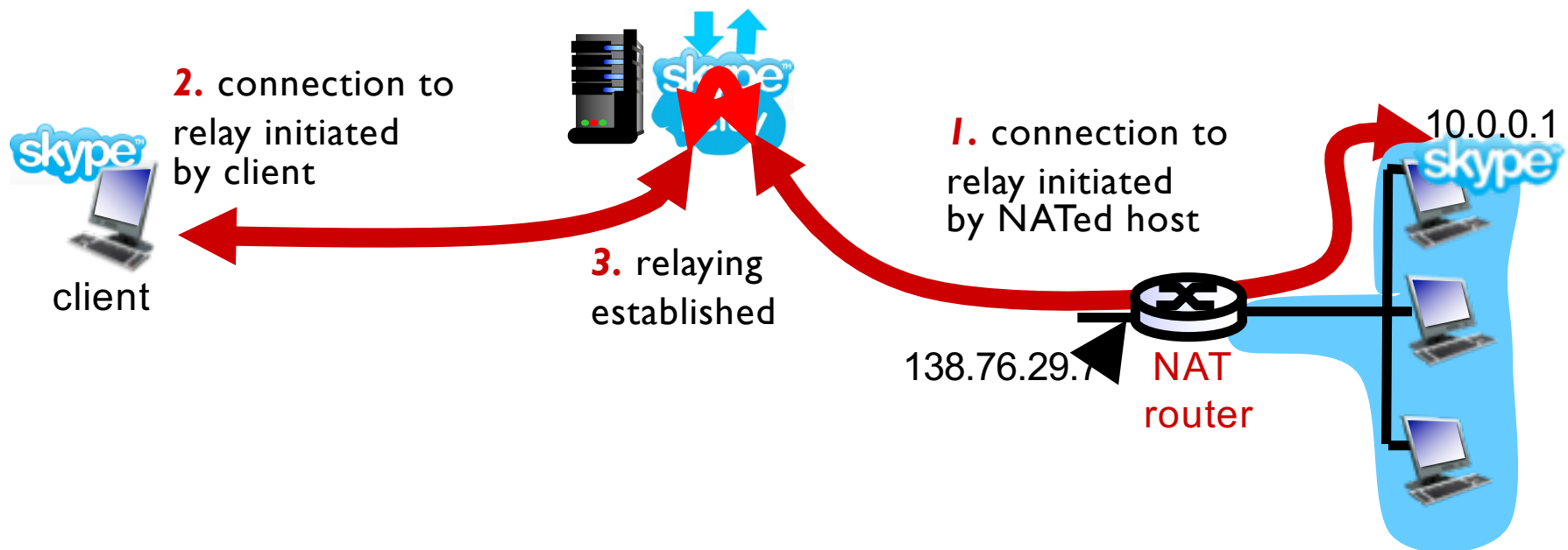
i.e., automate static NAT port map configuration



# NAT traversal problem

## ❖ *solution 3*: relaying (used in Skype)

- NATed client establishes connection to relay
- external client connects to relay
- relay bridges packets between to connections



# Chapter 4: outline

## 4.1 introduction

## 4.2 virtual circuit and datagram networks

## 4.3 what's inside a router

## 4.4 IP: Internet Protocol

- datagram format
- IPv4 addressing
- ICMP
- IPv6

## 4.5 routing algorithms

- link state
- distance vector
- hierarchical routing

## 4.6 routing in the Internet

- RIP
- OSPF
- BGP

## 4.7 broadcast and multicast routing

# IPv6: motivation

- ❖ *initial motivation*: 32-bit address space soon to be completely allocated.
- ❖ additional motivation:
  - header format helps speed processing/forwarding
  - header changes to facilitate QoS

## *IPv6 datagram format:*

- fixed-length 40 byte header
- no fragmentation allowed



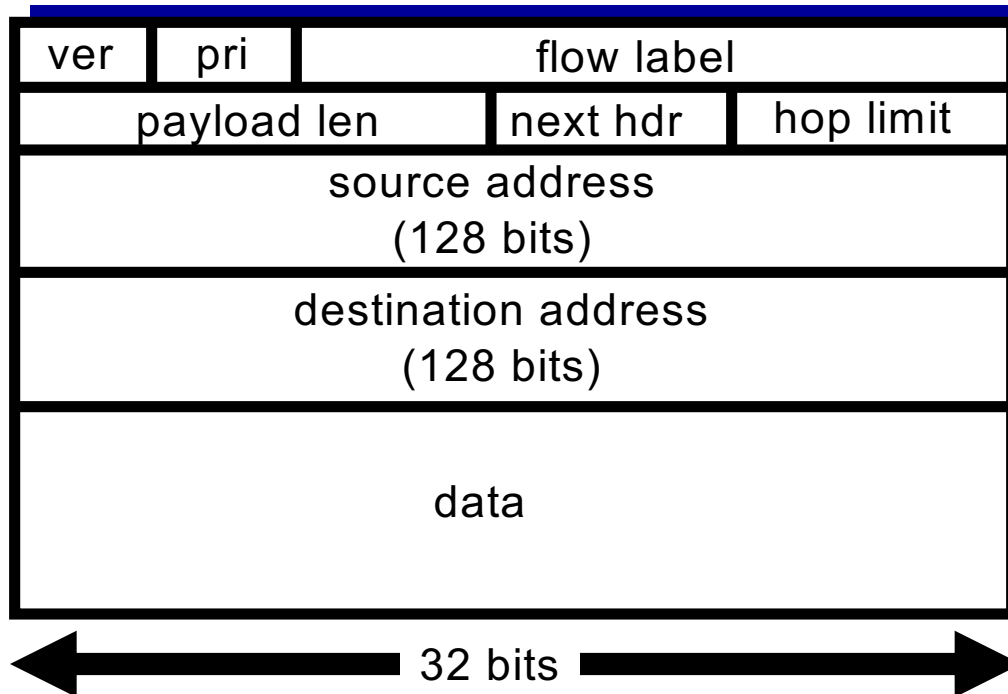
# IPv6 datagram format

*priority*: identify priority among datagrams in flow

*flow Label*: identify datagrams in same “flow.”

(concept of “flow” not well defined).

*next header*: identify upper layer protocol for data

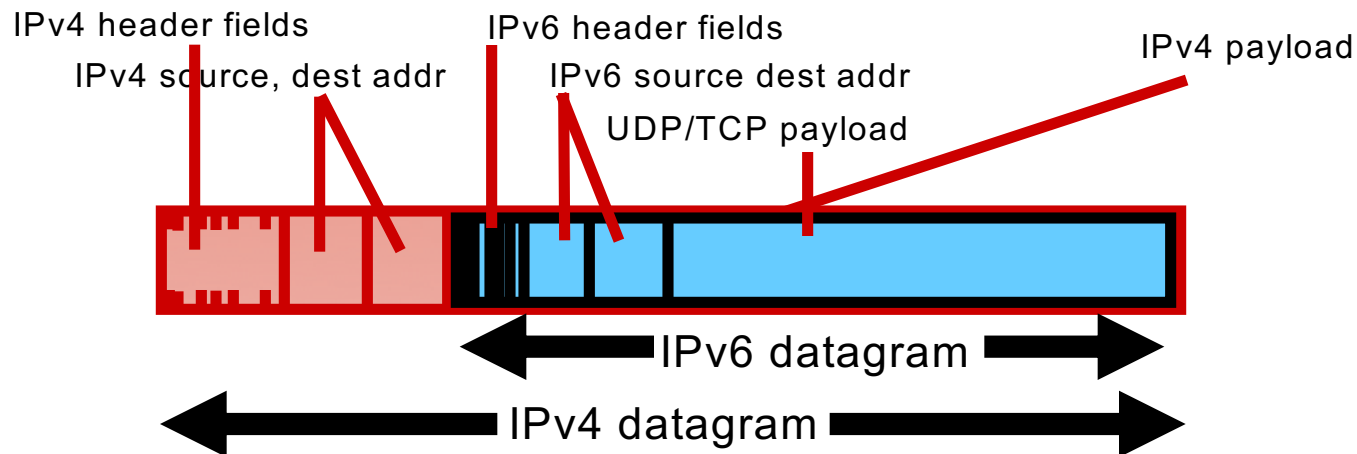


# Other changes from IPv4

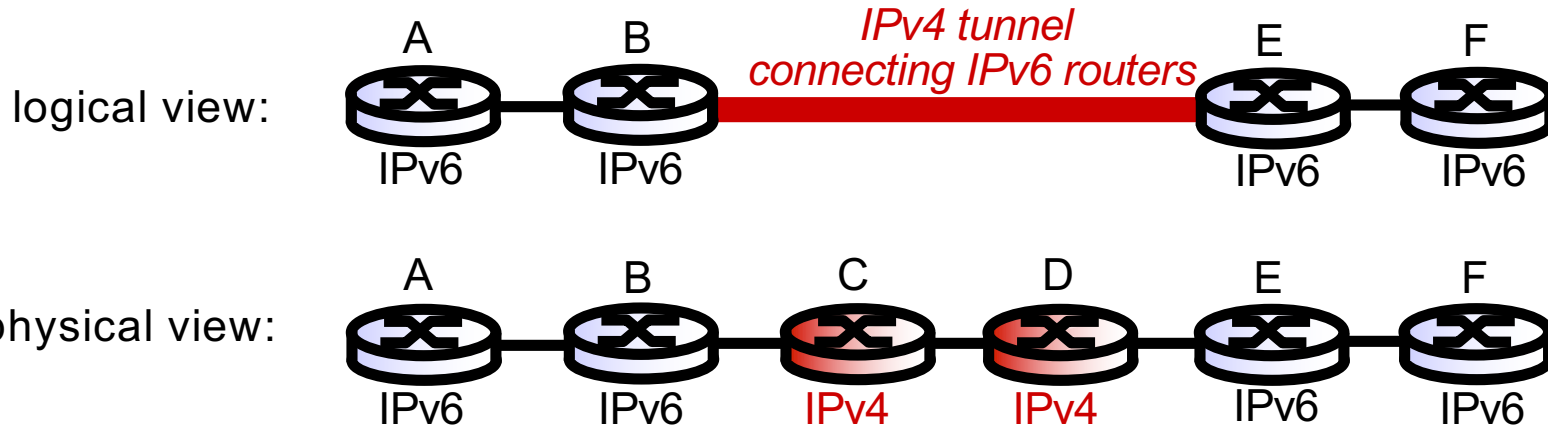
- ❖ *checksum*: removed entirely to reduce processing time at each hop
- ❖ *options*: allowed, but outside of header, indicated by “Next Header” field
- ❖ *ICMPv6*: new version of ICMP
  - additional message types, e.g. “Packet Too Big”
  - multicast group management functions

# Transition from IPv4 to IPv6

- ❖ not all routers can be upgraded simultaneously
  - no “flag days”
  - how will network operate with mixed IPv4 and IPv6 routers?
- ❖ **tunneling**: IPv6 datagram carried as *payload* in IPv4 datagram among IPv4 routers

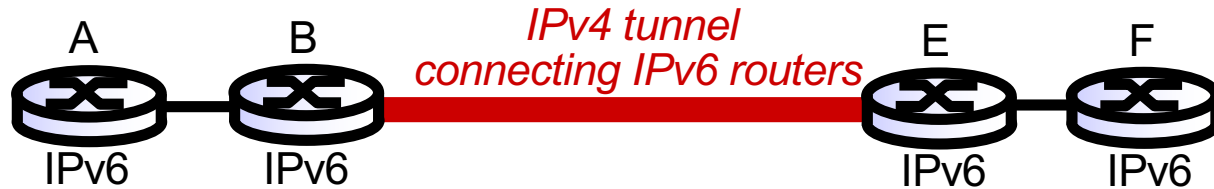


# Tunneling



# Tunneling

logical view:



physical view:

