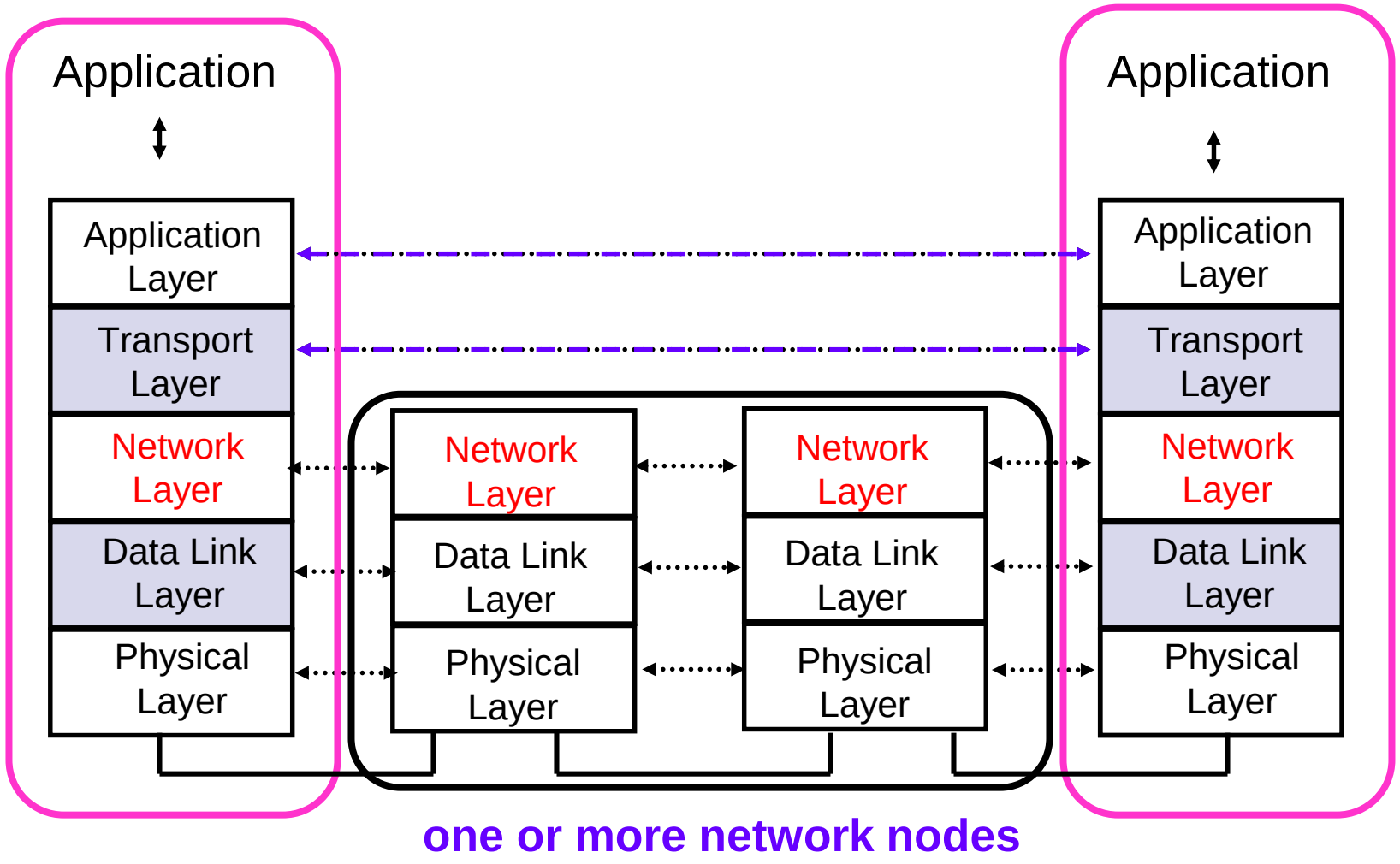
2.5 Overview of network layer

- transport layer provides process-to-process communication by relying on network layer’s host-to-host communication service

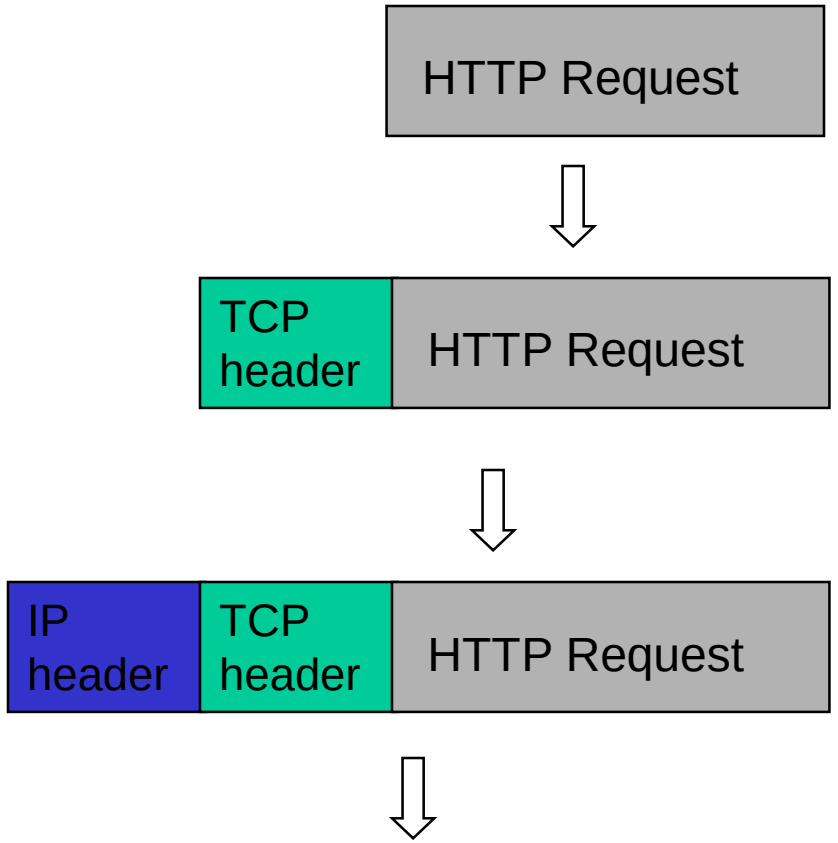
- while application layer and transport layer exist only in host, network layer exists in host and also in router


Application
Application
Application Layer
Transport Layer
Network Layer
Data Link Layer
Physical Layer
Network Layer
Data Link Layer
Physical Layer
Network Layer
Data Link Layer
Physical Layer
Application Layer
Transport Layer
Network Layer
Data Link Layer
Physical Layer
one or more network nodes

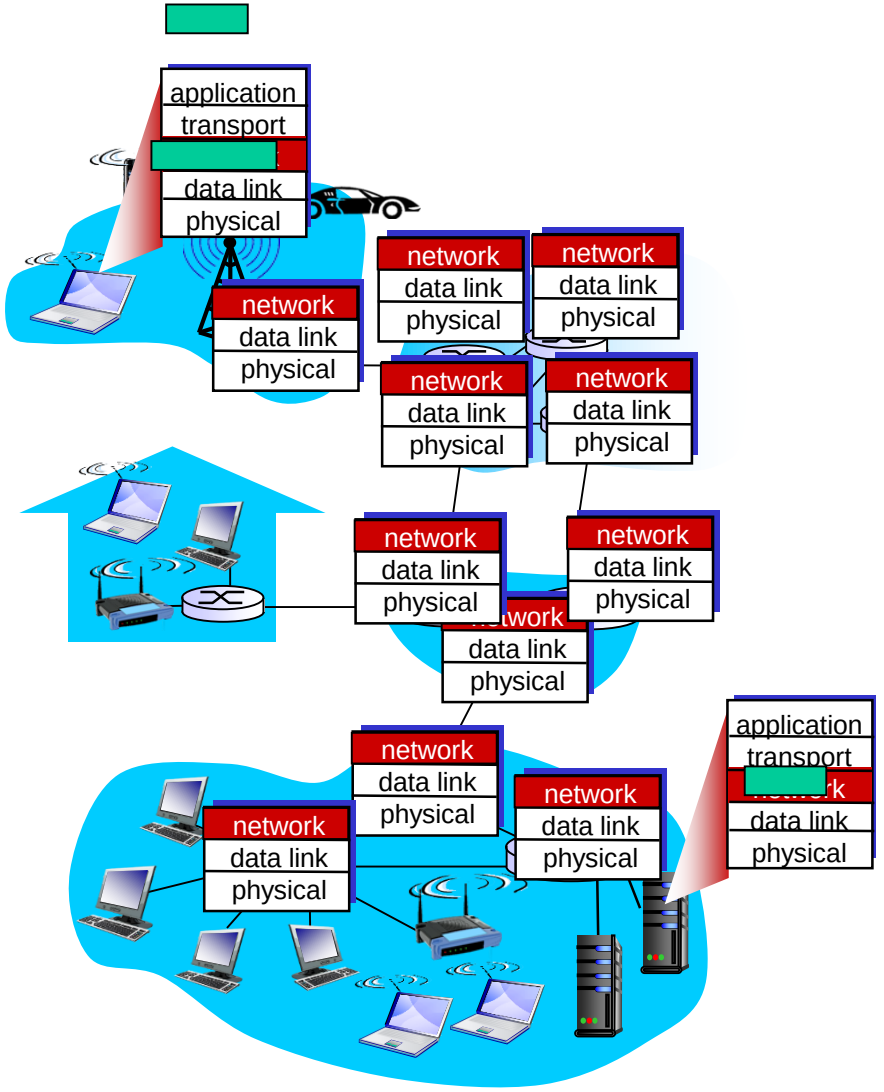
encapsulation

TCP Header contains source & destination port numbers

IP Header contains source and destination IP addresses; transport protocol type



- 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



- most complex layer in the protocol stack

- can be decomposed into 2 interacting parts – data plane, control plane

- data plane functions – per-router functions to forward a datagram arrived on one of router's input links to one of router's output links

- *control plane functions – control how a datagram is routed among routers from source to destination*

network-layer functions:

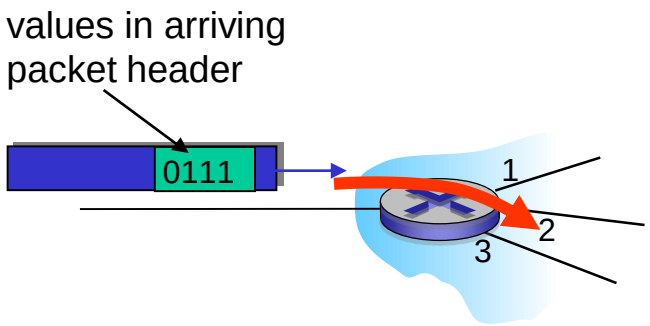
- *forwarding:* move packets from router's input to appropriate router output
- *routing:* determine route taken by packets from source to destination
  - *routing algorithms*

analogy: taking a trip

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

Data plane

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



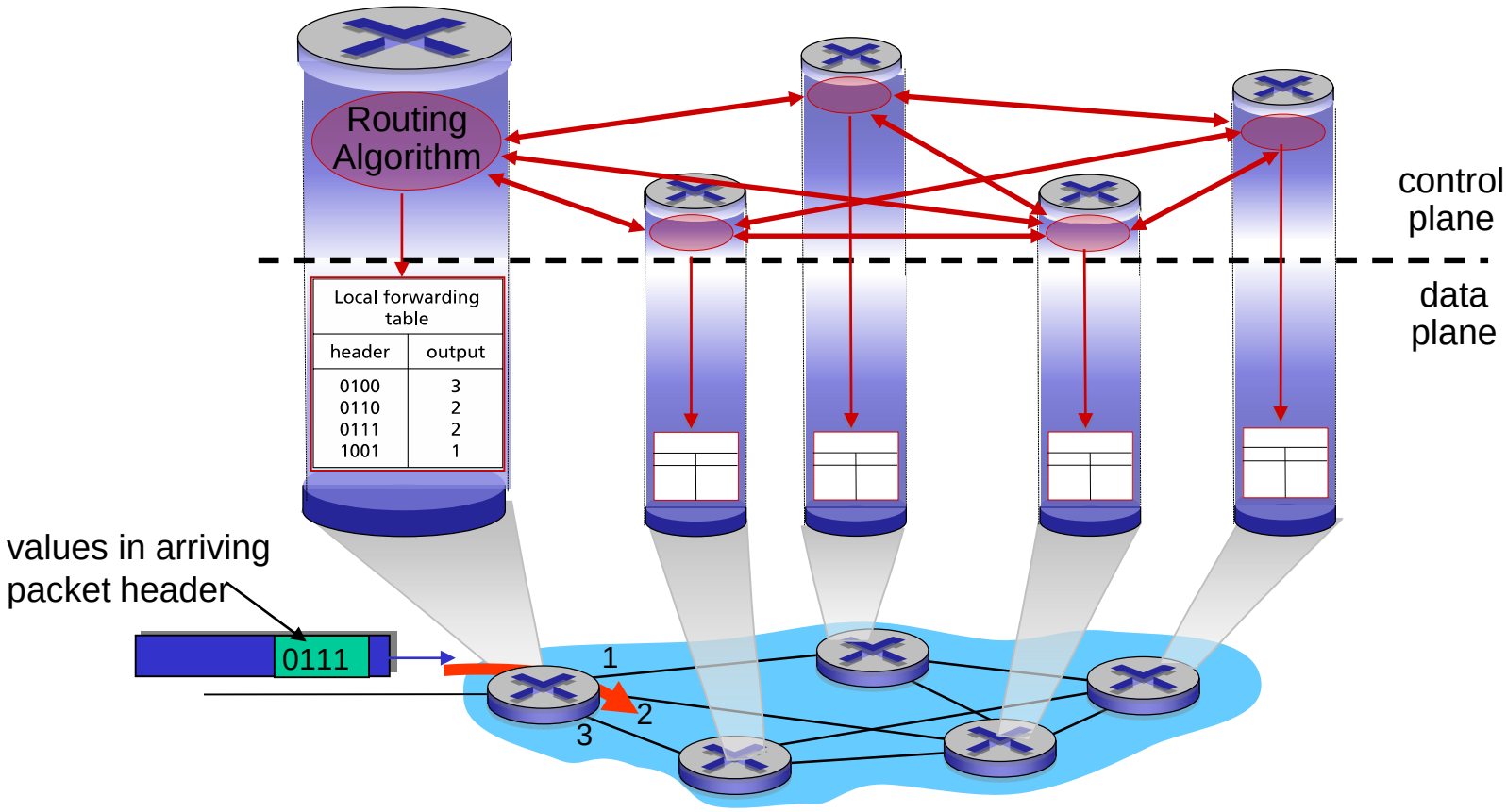
Control plane

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

traditional approach!

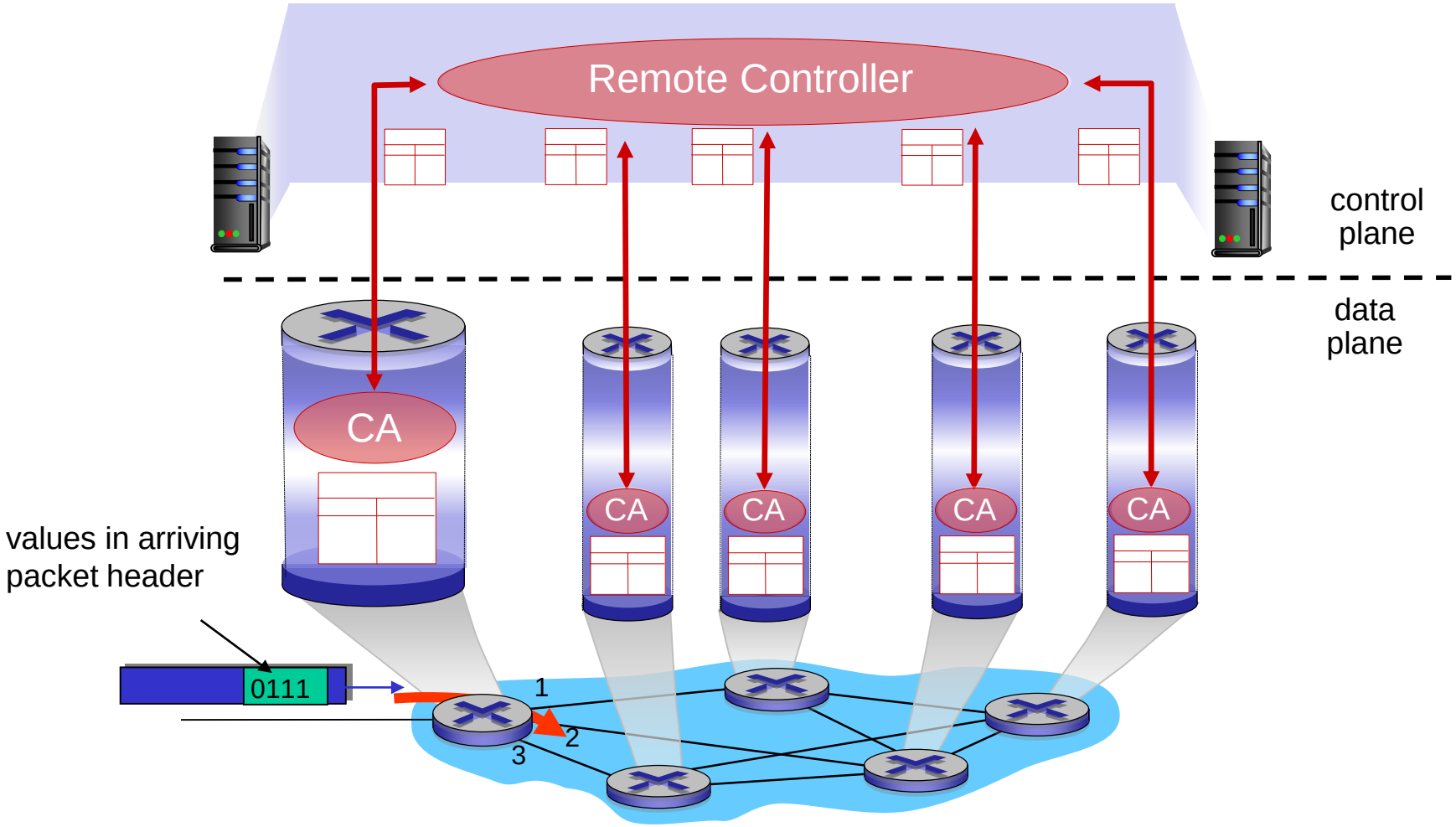
Individual routing algorithm components *in each and every router* interact in the control plane



logically centralized control plane

new approach!

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



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:

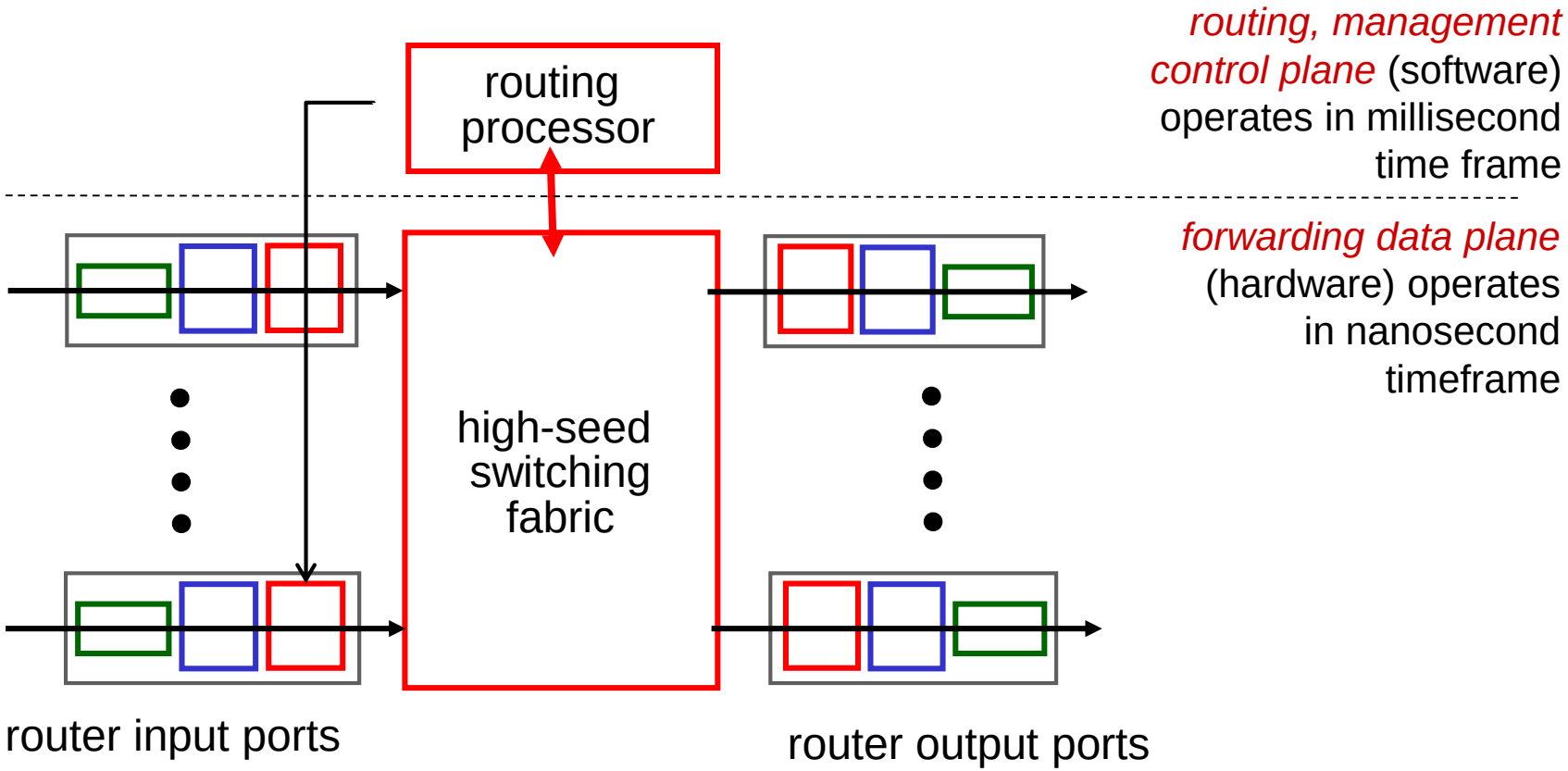
- in-order datagram delivery
- guaranteed minimum bandwidth to flow
- restrictions on changes in inter-packet spacing

Network 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

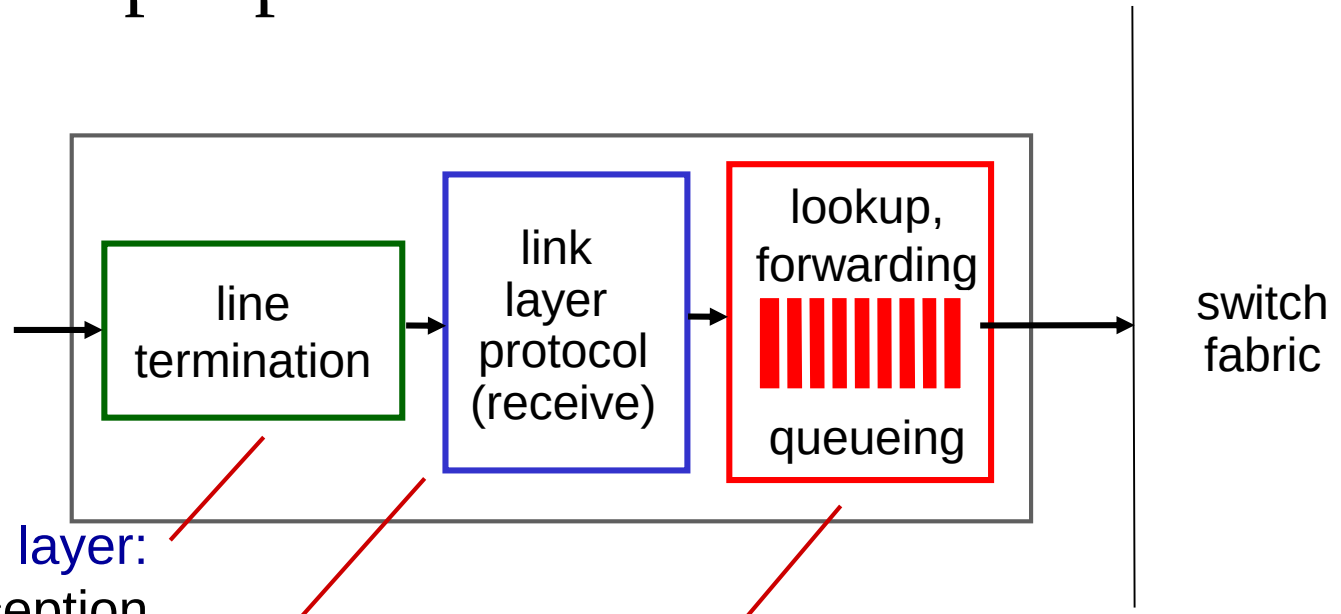
ATM (Asynchronous Transfer Mode) networks are broadband integrated services networks

2.6 What's inside a router

- high-level view of generic router architecture:



input port functions

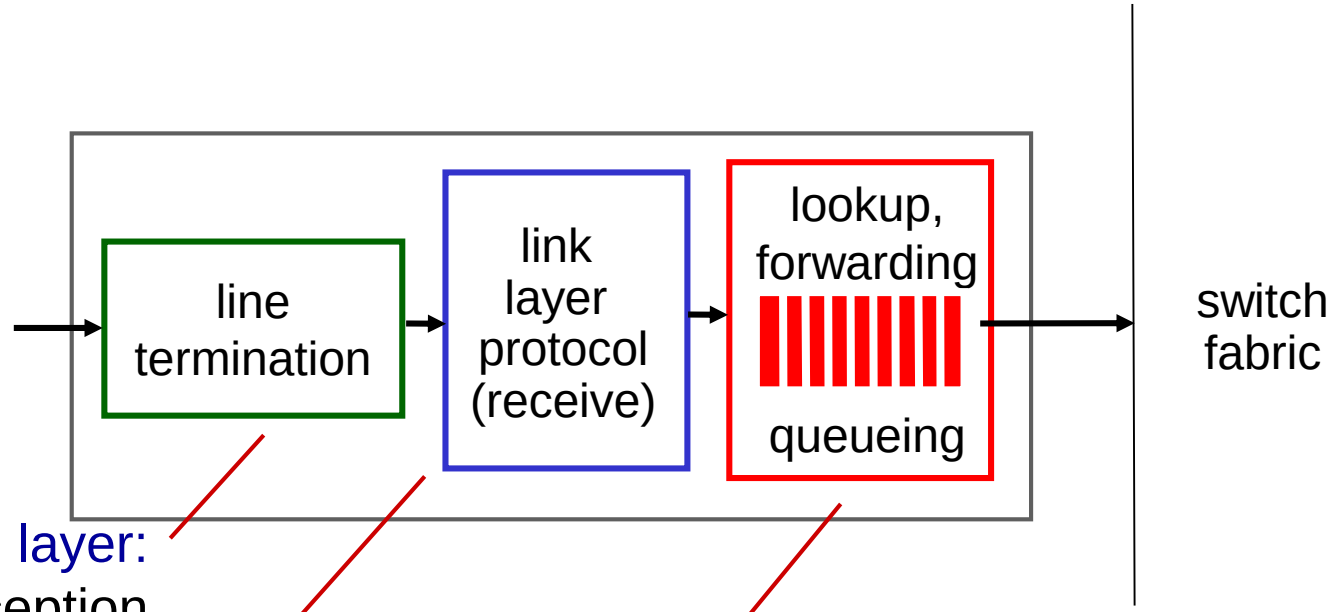


physical layer:
bit-level reception

data link layer:
e.g., Ethernet
see chapter 5

decentralized switching:

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



physical layer:
bit-level reception

data link layer:
e.g., Ethernet
see chapter 5

decentralized switching:

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

destination-based forwarding

forwarding table

Destination Address Range	Link Interface

`11001000 00010111 00010000 00000000`
through
`11001000 00010111 00010111 11111111`
	0

`11001000 00010111 00011000 00000000`
through
`11001000 00010111 00011000 11111111`
	1

`11001000 00010111 00011001 00000000`
through
`11001000 00010111 00011111 11111111`
	2

otherwise	3

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

longest prefix matching

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:

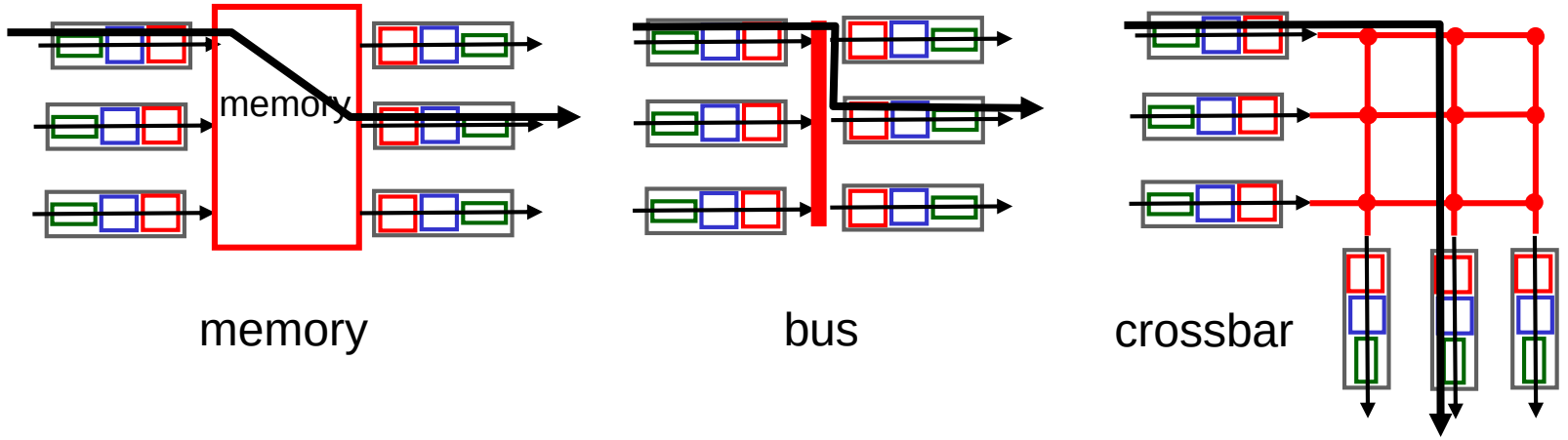
DA: 11001000 00010111 00010110 10100001 which interface?

DA: 11001000 00010111 00011000 10101010 which interface?

- 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 hold upwards ~1M routing table entries in TCAM

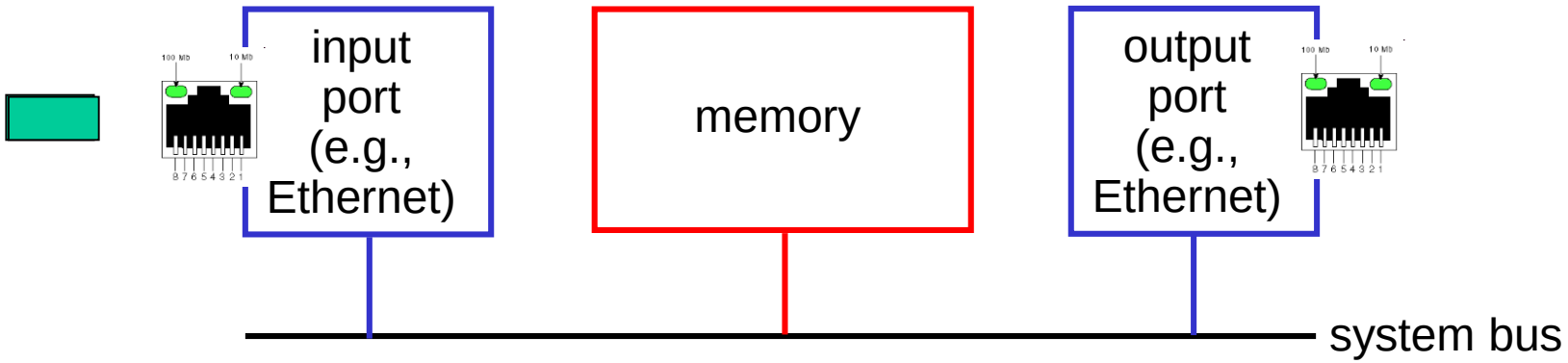
switching fabrics

- 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

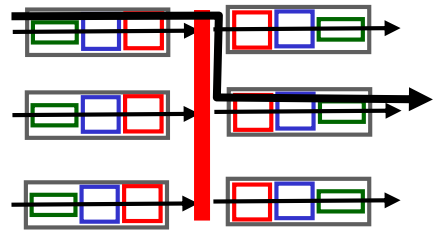


first generation routers:

- 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)

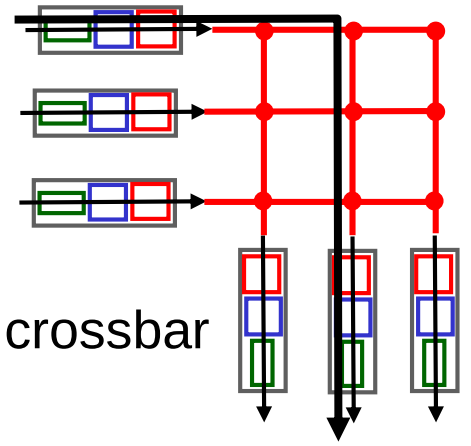


- datagram from input port memory to output port memory via a shared bus
- *bus contention:* switching speed limited by bus bandwidth
- 32 Gbps bus, Cisco 5600: sufficient speed for access and enterprise routers



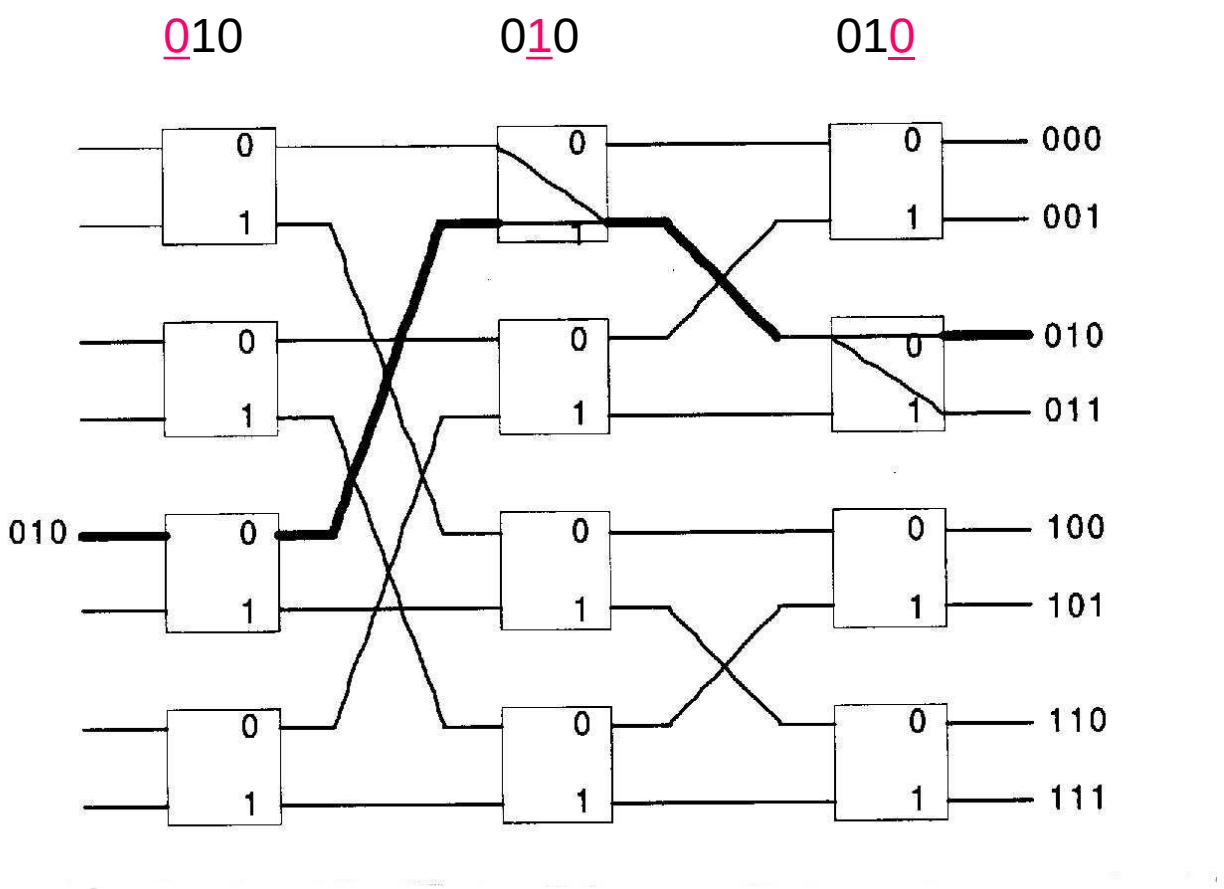
bus

- 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 12000: switches 60 Gbps through the interconnection network



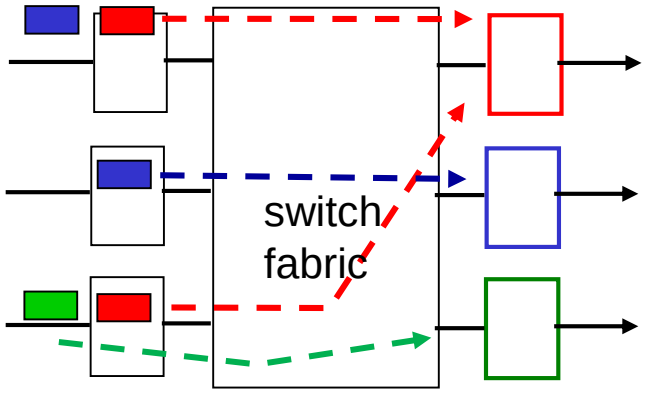
- A switch fabric has the ability to route packets to proper output ports, based on the physical output port addresses that are attached in the front of each packet
- Banyan switch is one of the switch fabrics that has the self-routing property (see next slide)

8x8 Banyan switch

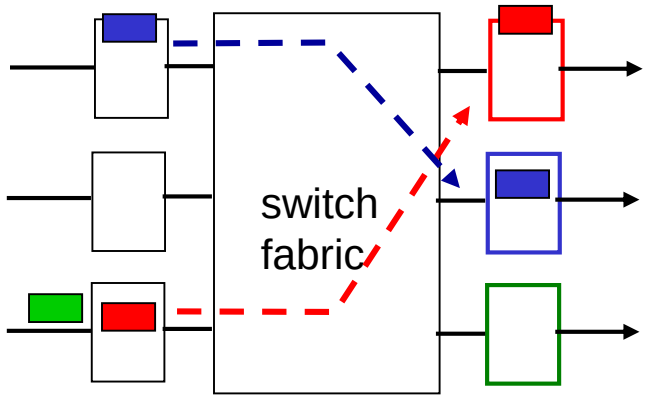


input port queuing

- 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

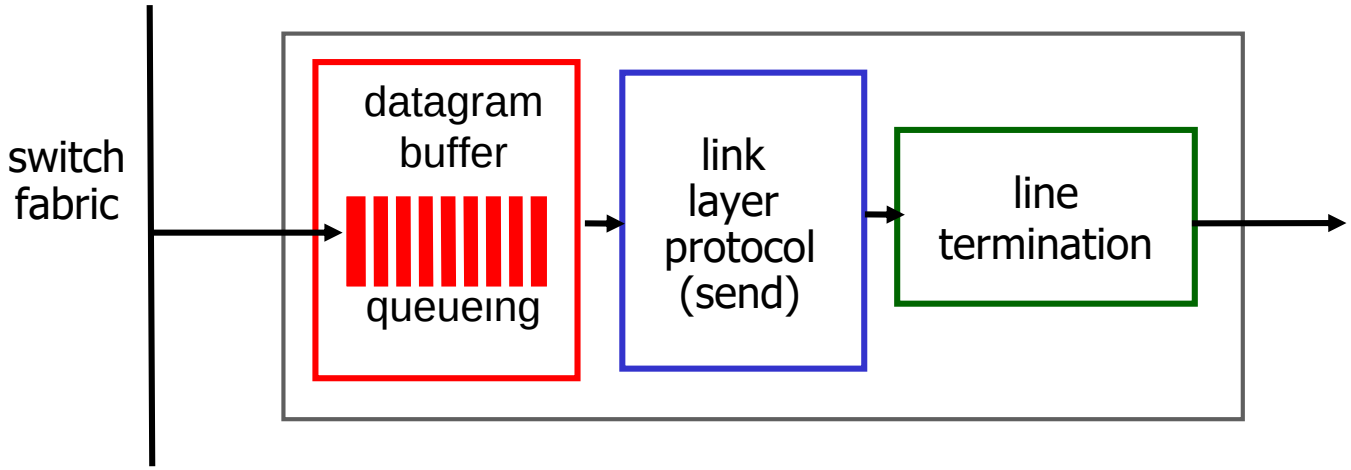


output port contention:
only one red datagram can be transferred.
lower red packet is blocked



one packet time later: lower red packet is transferred, so green packet experiences HOL blocking

output ports



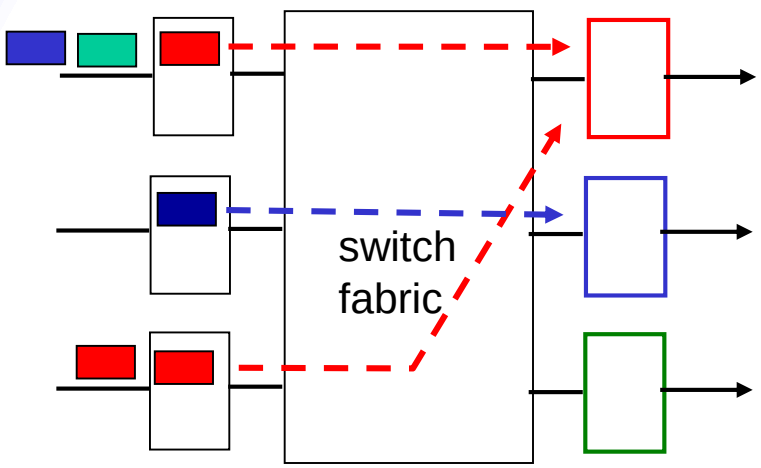
- *buffering* requir
  from fabric fast
  rate

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

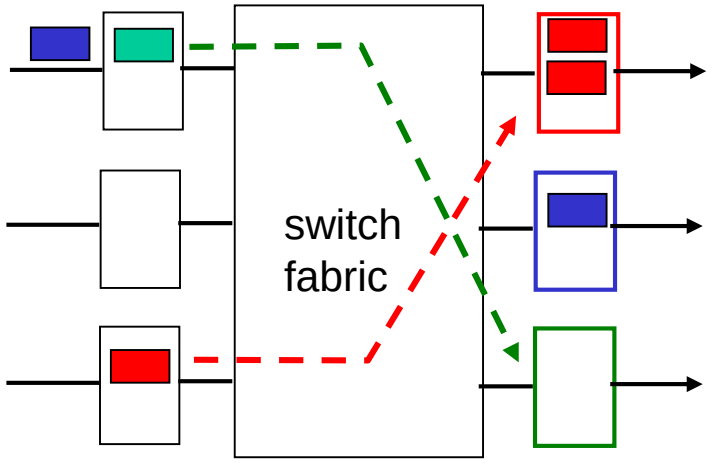
- *scheduling*
  queued dat

Priority scheduling – who gets best performance, network neutrality

output port queueing



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!*

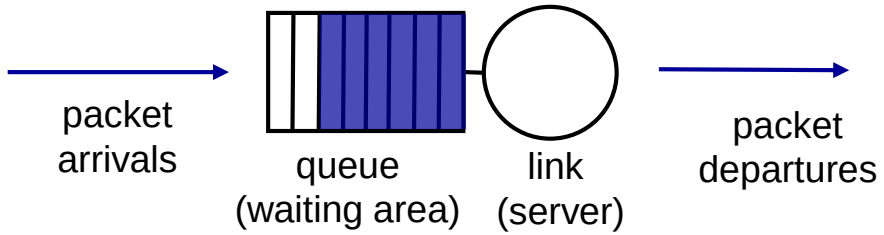
How much buffering?

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

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

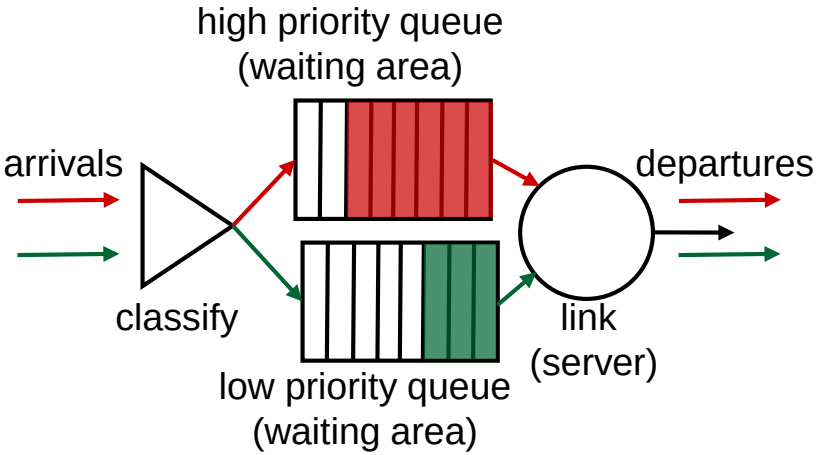
scheduling mechanisms

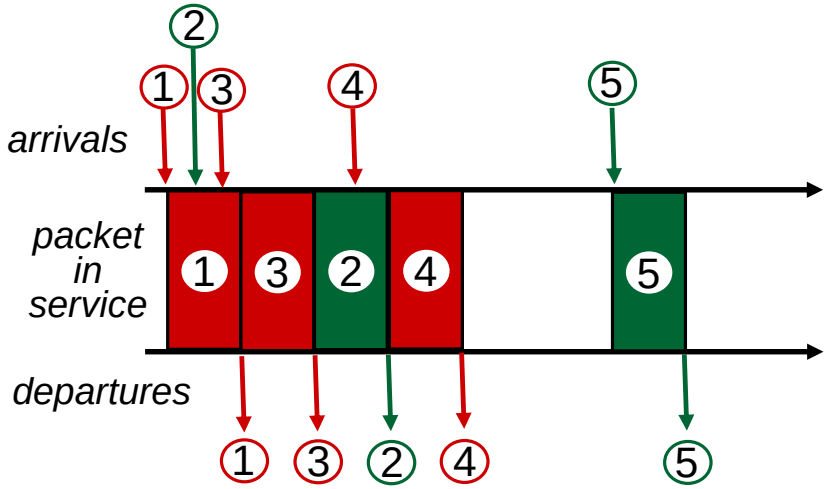
- *scheduling:* choose next packet to send on link
- *first in first out (FIFO) scheduling:* send in order of arrival to queue
  - real-world example?
  - *discard policy:* if packet arrives to full queue: who to discard?
    - *tail drop:* drop arriving packet
    - *priority:* drop/remove on priority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



priority scheduling: send highest priority queued packet

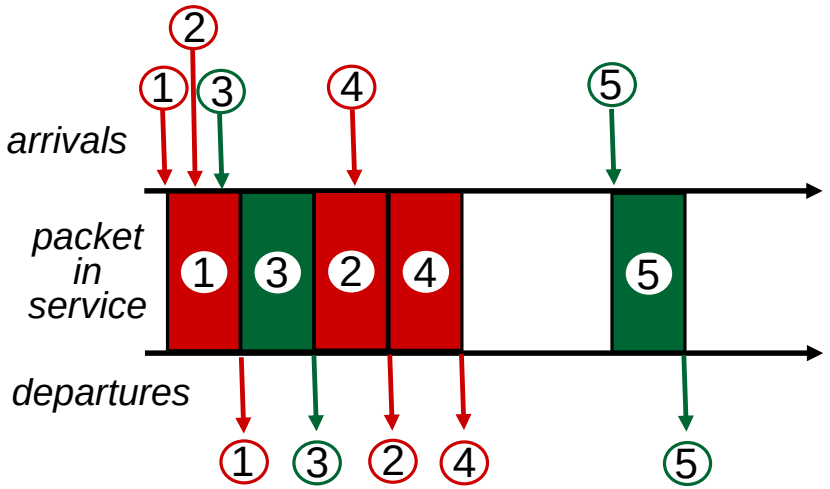
- multiple *classes*, with different priorities
  - class may depend on marking or other header info, e.g. IP source/dest, port numbers, etc.
  - real world example?





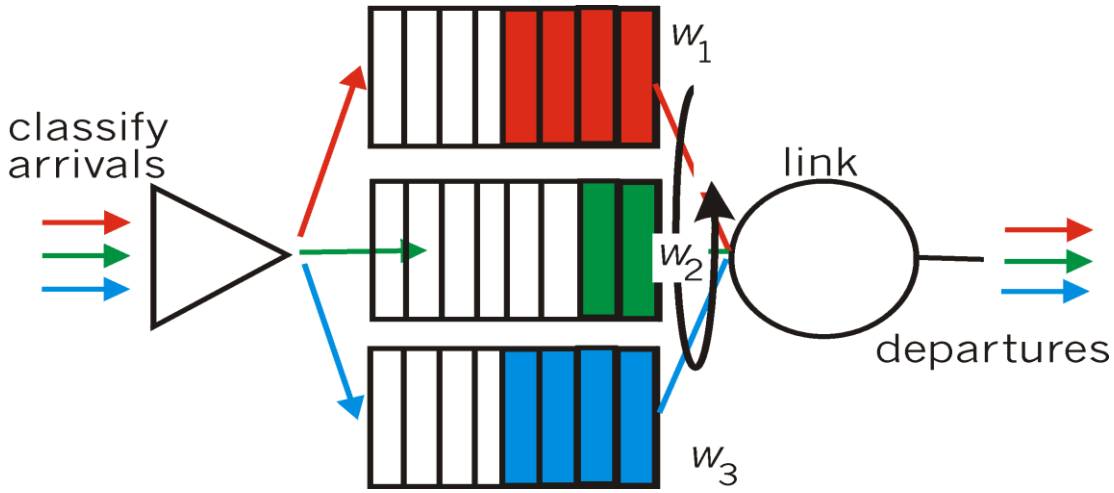
Round Robin (RR) scheduling:

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

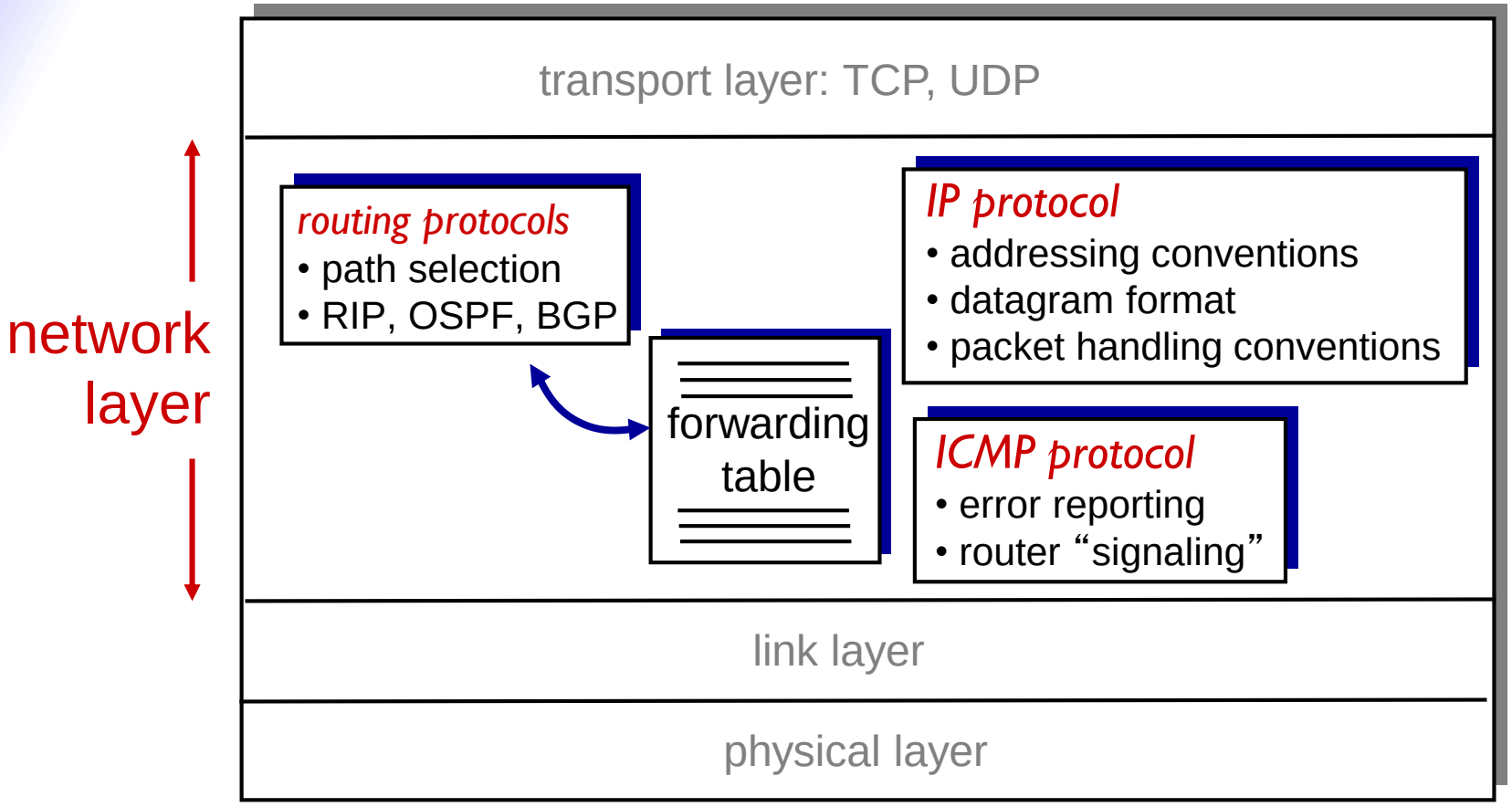


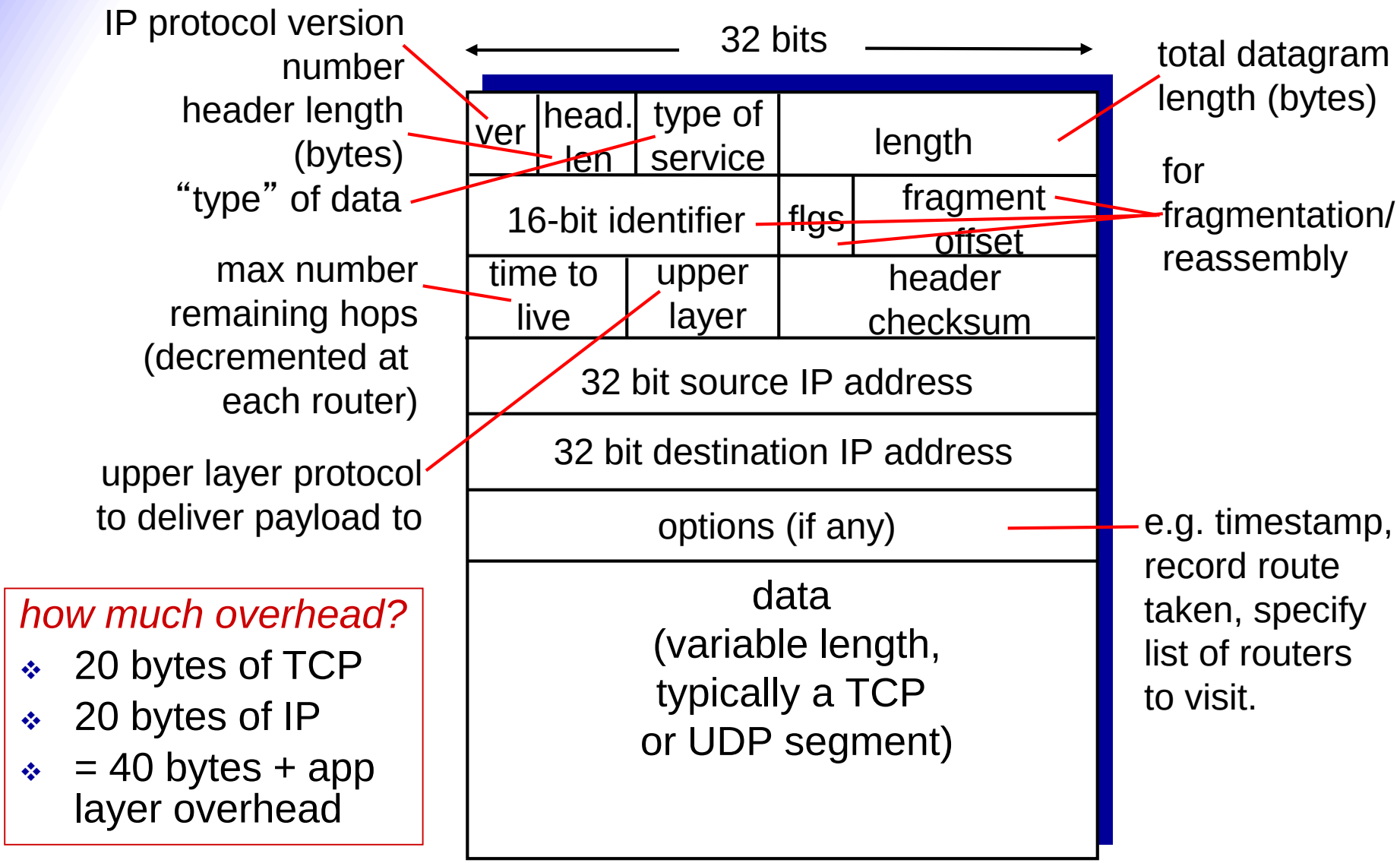
Weighted Fair Queuing (WFQ):

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



2.7 Internet Protocol (IP)



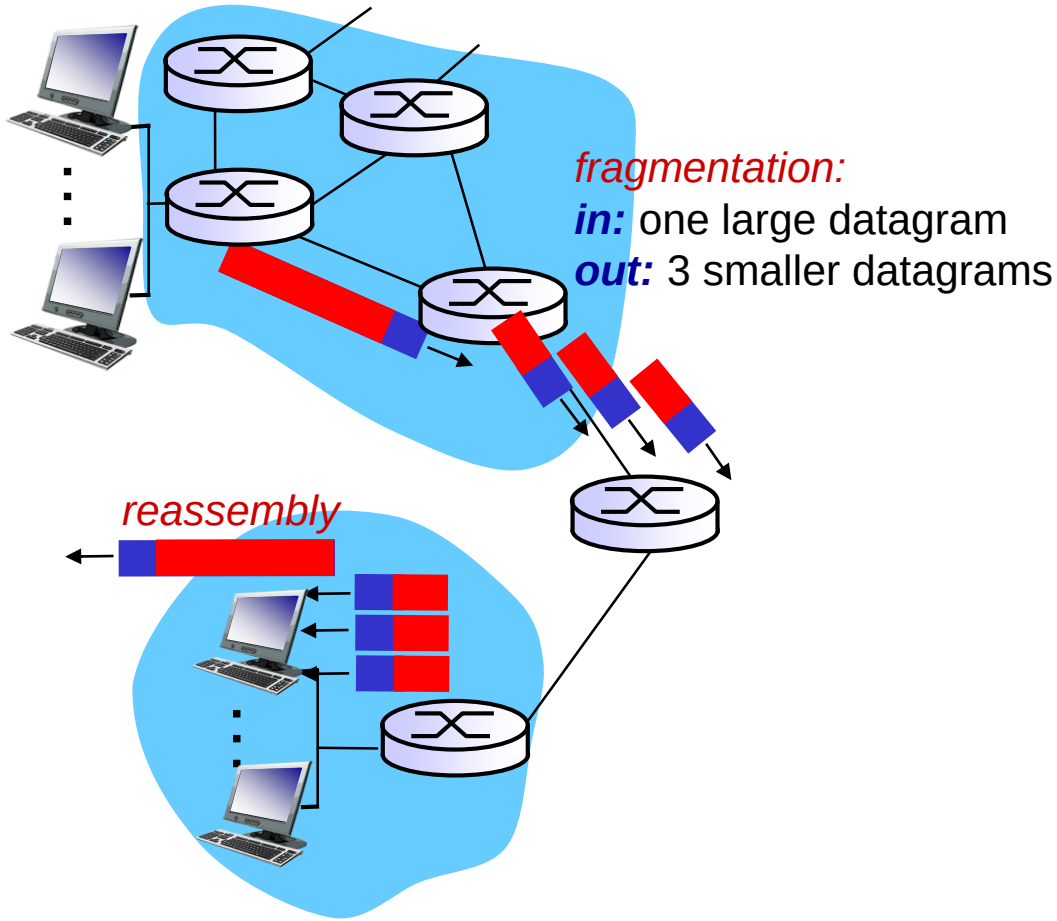


how much overhead?

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

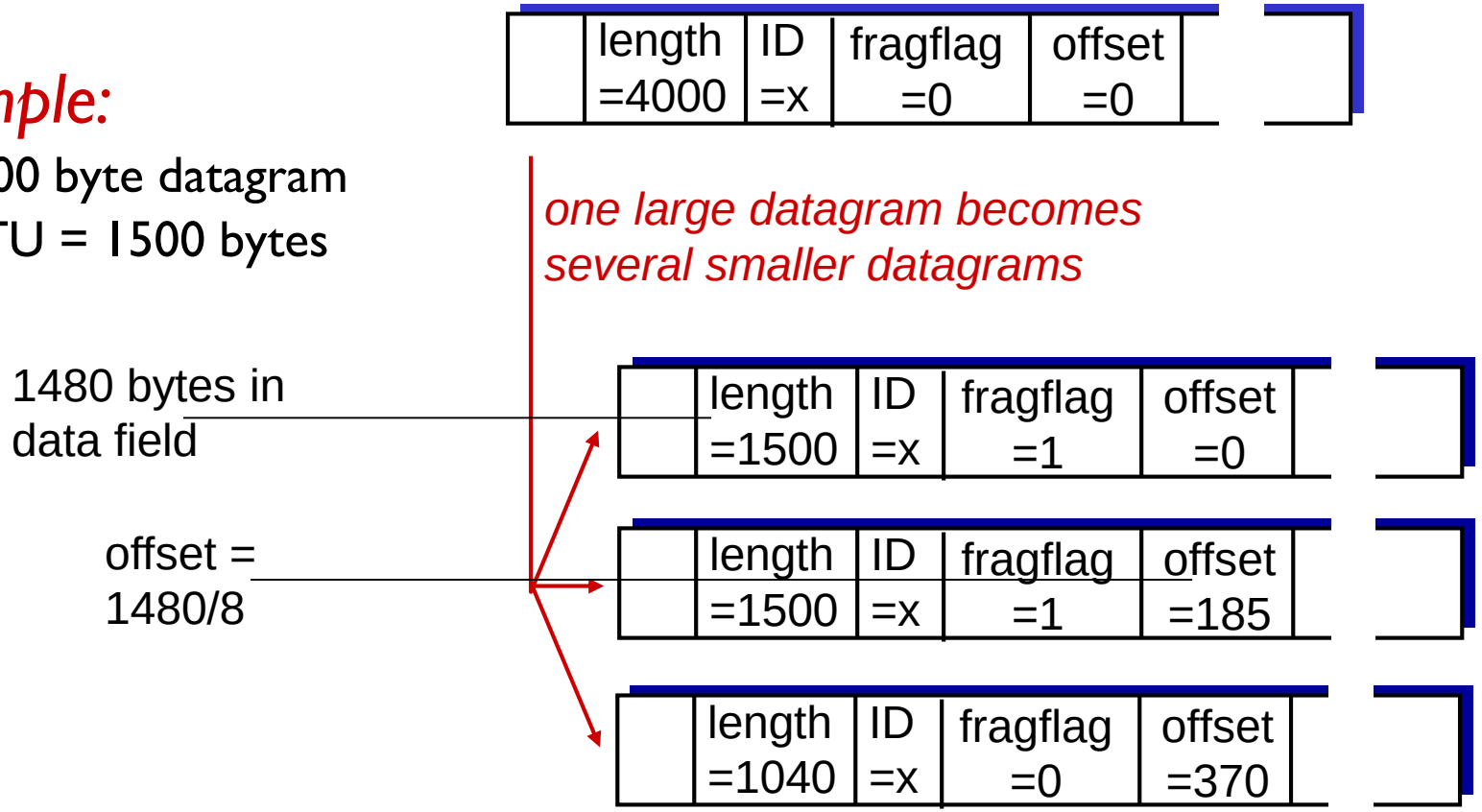
datagram fragmentation

- network links have maximum transmission unit (MTU) - 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



example:

- 4000 byte datagram
- MTU = 1500 bytes



IPv4 addressing

Internet Names

- Each host has a unique name
  - Independent of physical location
  - Facilitate memorization by humans
  - Domain Name
  - Organization under single administrative unit
- Host Name
  - Name given to host computer
- User Name
  - Name assigned to user

leongarcia@comm.utoronto.ca

Internet Addresses

- Each host has globally unique *logical* 32 bit IP address
- Separate address for each physical connection to a network
- Routing decision is done based on destination IP address
- IP address has two parts:
  - *netid* and *hostid*
  - *netid* unique
  - *netid* facilitates routing
- Dotted Decimal Notation:

  int1.int2.int3.int4

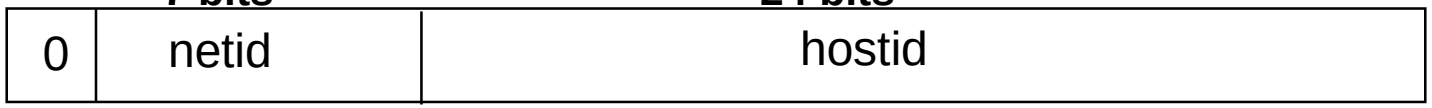
  (intj = jth octet)

IP address of 10000000 10000111 01000100 00000101
is 128.135.68.5 in dotted-decimal notation

DNS resolves IP name to IP address

classful addressing

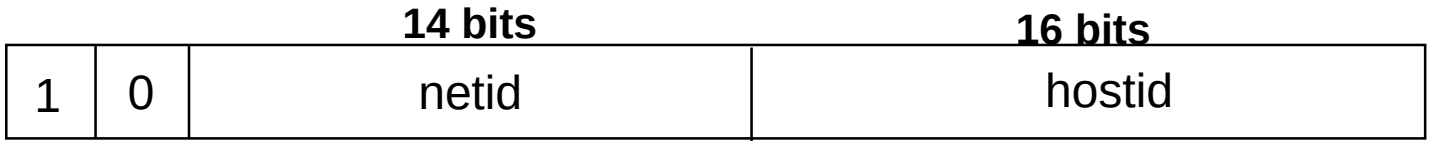
Class A



- 126 networks with up to 16 million hosts

1.0.0.0 to 127.255.255.255

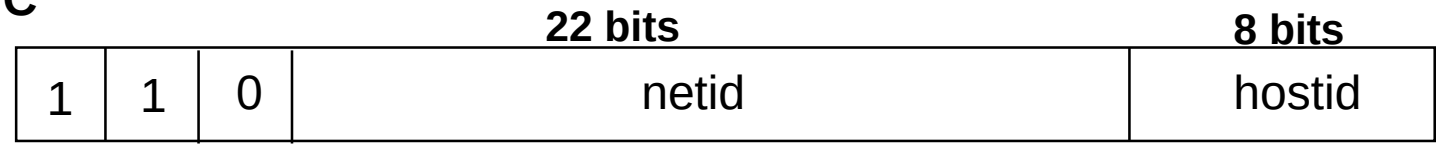
Class B



- 16,382 networks with up to 64,000 hosts

128.0.0.0 to 191.255.255.255

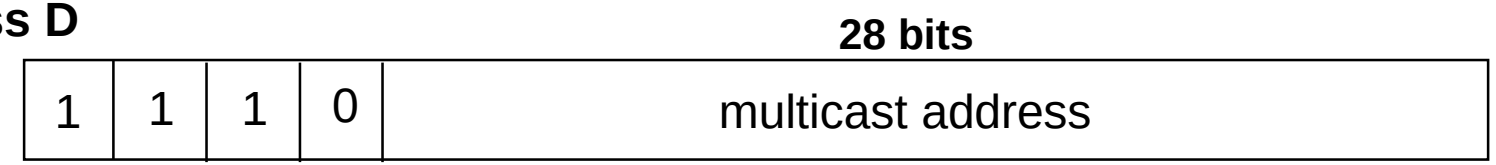
Class C



- 2 million networks with up to 254 hosts

192.0.0.0 to 223.255.255.255

Class D



224.0.0.0 to
239.255.255.255

- Up to 250 million multicast groups at the same time

first octet rule

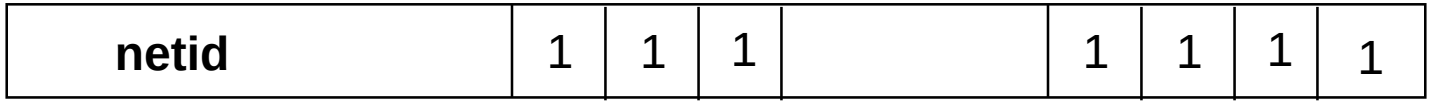
- This rule states that the class of an address can be determined by the numerical value of the first octet
- The range of class addresses is as follows, with the first octet represented in decimal:
  - 1 to 126 – Class A address
  - 128 to 191 - Class B address
  - 192 to 223 - Class C address
  - 224 to 239 - Class D address
- Note that the number 127.0.0.0 is reserved for the *loopback address*, which is used by a device to address itself internally. This technique is used to test the local device's TCP/IP stack and identify possible stack corruption.

reserved host IDs (all 0s and 1s)

Internet address used to refer to network has hostid set to all 0s

Broadcast address has hostid set to all 1s

broadcast on local network



broadcast on distant network

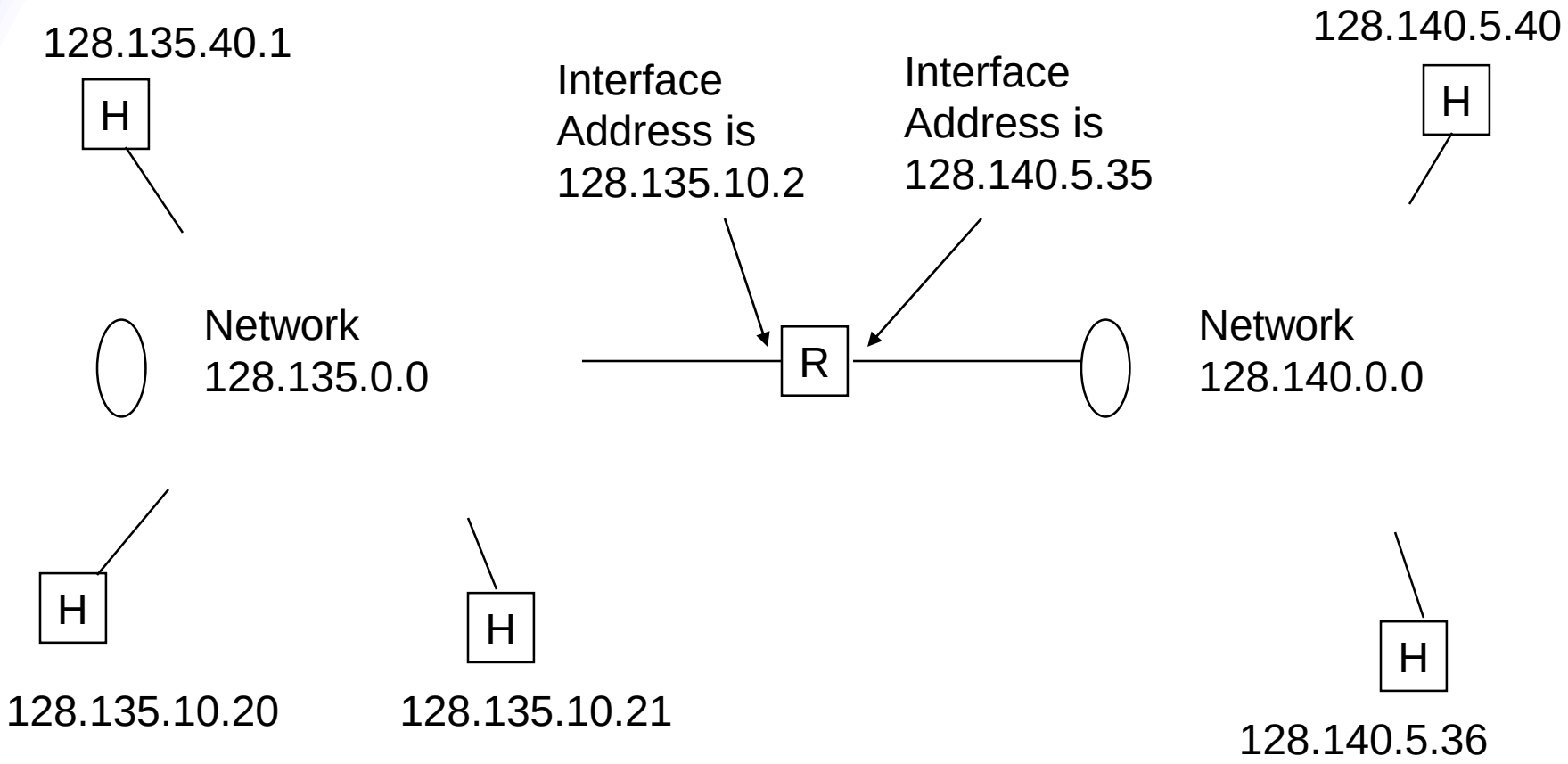
e.g. 200.99.44.0 refers to a network, host address belonging to this network ranges from 200.99.44.1 to 200.99.44.254

Therefore, if n is the number of bits in host portion of an IP address, the number of available host addresses is $2^n - 2$.

private IP addresses

- Specific ranges of IP addresses set aside for use in private networks (RFC 1918)
- Use restricted to private internets; routers in public Internet discard packets with these addresses
- Range 1: 10.0.0.0 to 10.255.255.255
- Range 2: 172.16.0.0 to 172.31.255.255
- Range 3: 192.168.0.0 to 192.168.255.255
- Network Address Translation (NAT) used to convert between private and global IP addresses

Example



Address with host ID=all 0s refers to the network
Address with host ID=all 1s refers to a broadcast packet

R = router
H = host

- Note that an IP address does **not** identify a specific computer. Instead, each IP address identifies a *connection between a computer and a network* (usually referred as an *interface*).
- Thus, a router (practically a computer which has multiple connections) must be assigned with one IP address for each interface

- Address mask is used to distinguish the network portion from the host portion of an IP address
- Also in dotted-decimal format, with all network bits set to one
- The standard or *default* masks for Class A, B, and C are:
  - A – 255.0.0.0
  - B – 255.255.0.0
  - C – 255.255.255.0

Example

Find network address for 150.100.12.176

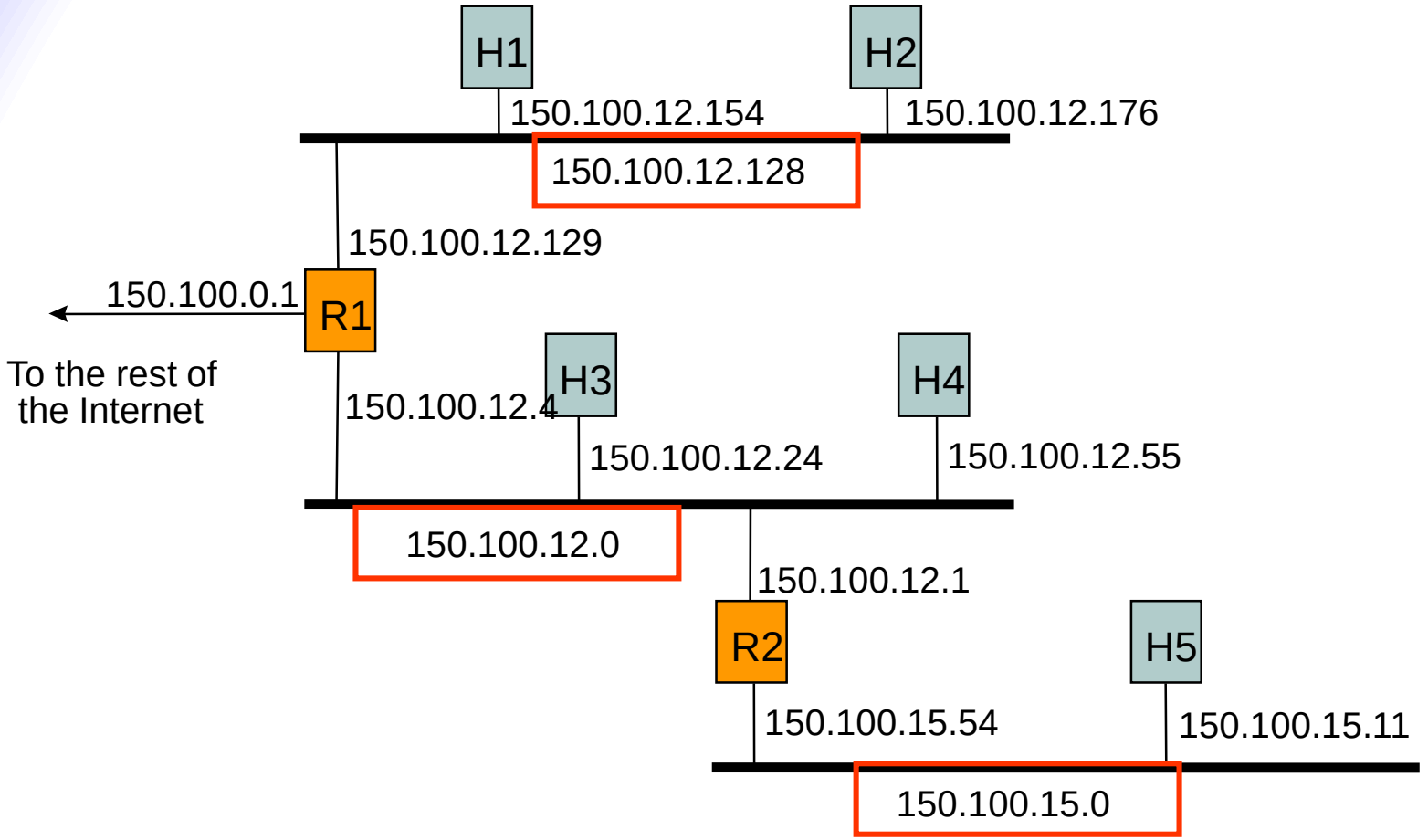
- IP add = 10010110 01100100 00001100 10110000
- Mask = 11111111 11111111 00000000 00000000
- AND = 10010110 01100100 00000000 00000000
- Subnet = 150.100.0.0

This may seem trivial, but it is useful for *subnetting*

- Subnet addressing introduces another hierarchical level
- Transparent to remote networks
- Simplifies management of multiplicity of LANs
- Masking used to find subnet number

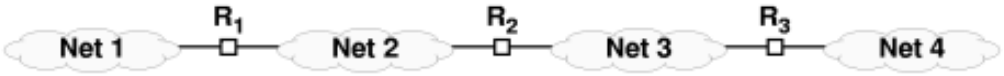
Example

- Organization has Class B address (16 host ID bits) with network ID: 150.100.0.0
- Create subnets with up to 100 hosts each
  - 7 bits sufficient for each subnet
  - 16-7=9 bits for subnet ID
- Apply subnet mask to IP addresses to find corresponding subnet
  - Example: Find subnet for 150.100.12.176
  - IP add = 10010110 01100100 00001100 10110000
  - Mask = 11111111 11111111 11111111 10000000
  - AND = 10010110 01100100 00001100 10000000
  - Subnet = 150.100.12.128
  - Subnet address used by routers within organization


H1
H2
150.100.12.154
150.100.12.176
150.100.12.128
150.100.12.129
150.100.0.1
R1
To the rest of
the Internet
H3
H4
150.100.12.4
150.100.12.24
150.100.12.55
150.100.12.0
150.100.12.1
R2
H5
150.100.15.54
150.100.15.11
150.100.15.0

conceptual forwarding table

- Each router keeps routing information in a *forwarding table*
- The table below shows conceptually the contents of forwarding table of router R_2
- Since R_2 directly connects to Net 2 and Net 3, it can deliver a datagram to any destination attached to those networks
- When a datagram is destined to Net 4, R_2 sends the datagram to R_3



(a)

Destination	Next Hop
net 1	R_1
net 2	deliver direct
net 3	deliver direct
net 4	R_3

(b)

actual forwarding tables

- In practice, an IP forwarding table includes a mask in each entry
- Forwarding table below is for the center router
- For a datagram with destination *D*, each entry *i* is checked such that

$$if\ (\ (Mask[i]\ \&\ D) == Destination\ [i])$$
$$forward\ to\ NextHop[i]\ ;$$

- Try $D = 192.4.10.0$, what is the next hop address?

(a)

Destination	Mask	Next Hop
30.0.0.0	255.0.0.0	40.0.0.7
40.0.0.0	255.0.0.0	deliver direct
128.1.0.0	255.255.0.0	deliver direct
192.4.10.0	255.255.255.0	128.1.0.9

(b)

IP address problems

- In the 1990, two problems became apparent
  - IP addresses were being exhausted
  - IP routing tables were growing very large
- IP Address Exhaustion
  - Class A, B, and C address structure inefficient
    - Class B too large for most organizations, but future proof
    - Class C too small
    - Rate of class B allocation implied exhaustion by 1994
- IP forwarding table size
  - Growth in number of networks in Internet reflected in # of table entries
    - From 1991 to 1995, routing tables doubled in size every 10 months
    - Stress on router processing power and memory allocation
- Short-term solution:
- Classless Interdomain Routing (CIDR), RFC 1518
- New allocation policy (RFC 2050)
- Private IP Addresses set aside for intranets
- Long-term solution: IPv6 with much bigger address space

classless addressing

- The idea of subnetting is generalized: the division between prefix and suffix can occur on an arbitrary bit boundary
- Besides allowing subnetting, networks can be aggregated to form a *supernet*
- CIDR notation is used for classless addressing, eg, the subnet 150.100.15.0 with subnet mask 255.255.255.0 can be represented as 150.100.15.0/24

supernetting

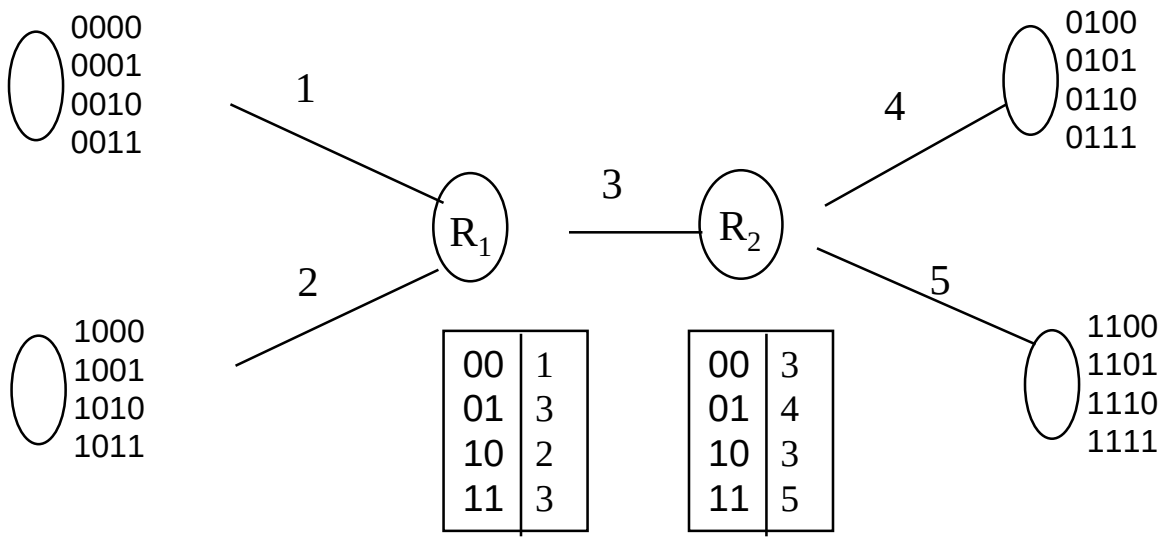
- Summarize a contiguous group of class C addresses using variable-length mask
- Example: 200.158.16.0/20
  - IP Address (200.158.16.0) & mask length (20)
  - IP add = 11001000 10011110 00010000 00000000
  - Mask = 11111111 11111111 11110000 00000000
  - Contains 16 Class C blocks:
  - From 11001000 10011110 00010000 00000000
  - i.e. 200.158.16.0
  - Up to 10010110 10011110 00011111 00000000
  - i.e. 200.158.31.0

classless inter-domain routing

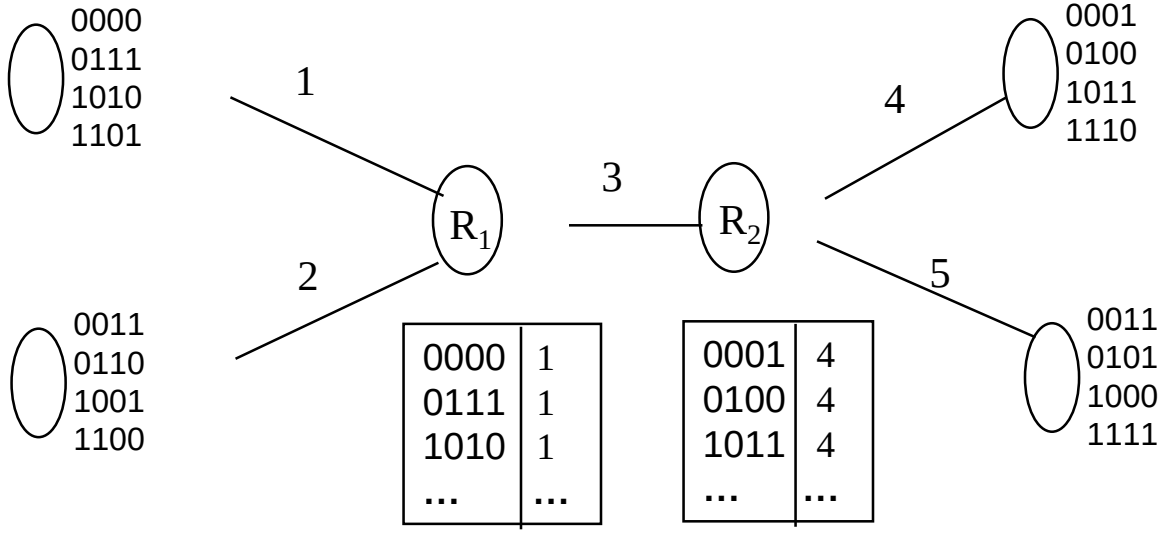
- CIDR deals with Routing Table Explosion Problem
  - Networks represented by prefix and mask
  - Pre-CIDR: Network with range of 16 contiguous class C blocks requires 16 entries
  - Post-CIDR: Network with range of 16 contiguous class C blocks requires 1 entry
- Solution: *Route according to prefix of address*, not class
  - Routing table entry has <IP address, network mask>
  - Example: 192.32.136.0/21
  - `11000000 00100000 10001000 00000001` min address
  - `11111111 11111111 11111--- --------` mask
  - `11000000 00100000 10001--- --------` IP prefix
  - `11000000 00100000 10001111 11111110` max address
  - `11111111 11111111 11111--- --------` mask
  - `11000000 00100000 10001--- --------` same IP prefix

hierarchical routing and table efficiency

(a)

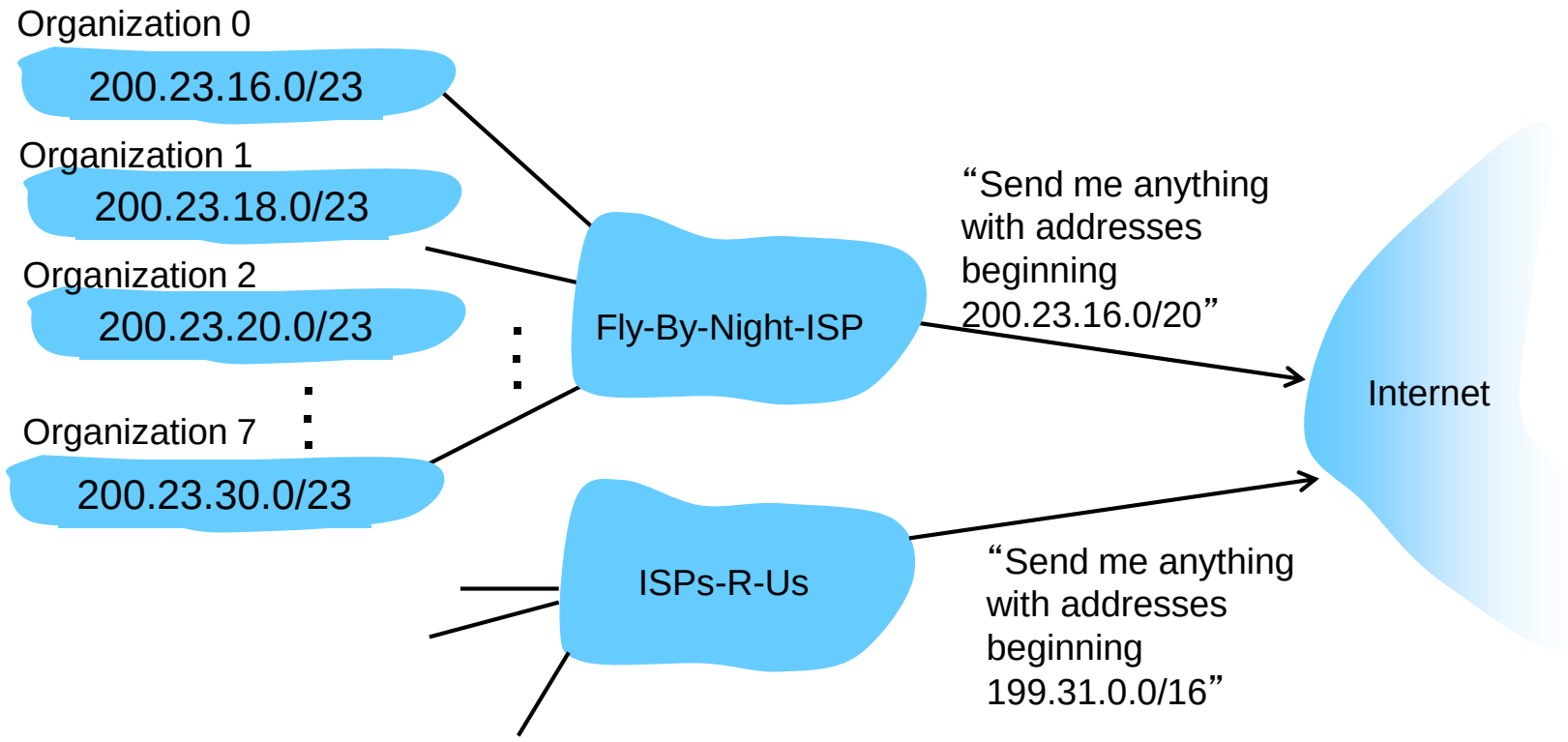


(b)



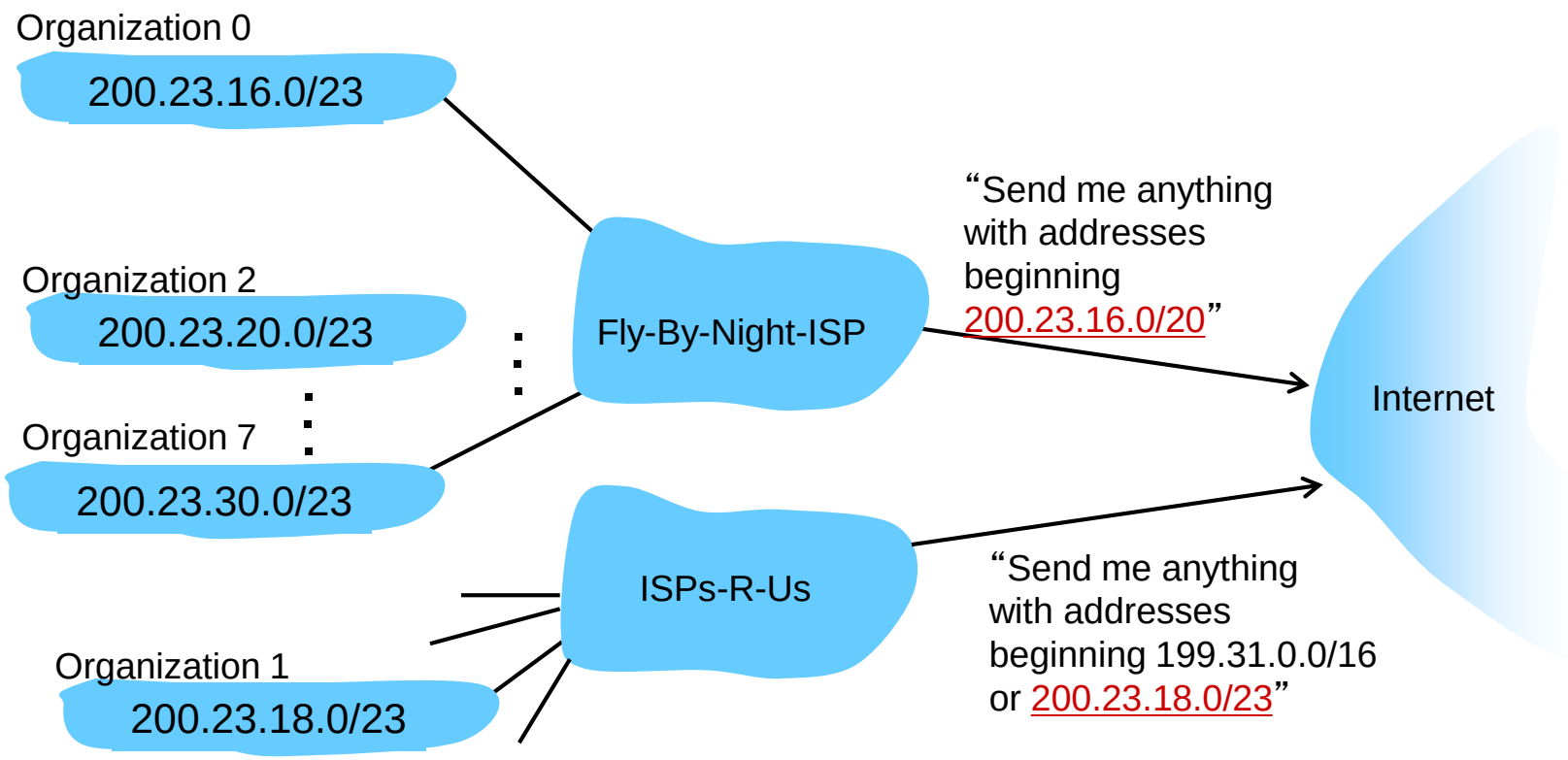
hierarchical addressing: route aggregation

hierarchical addressing allows efficient advertisement of routing information:



hierarchical addressing: more specific routes

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



longest prefix match

- CIDR impacts routing and forwarding
- Forwarding tables and routing protocols must carry IP address and mask
- Multiple entries may match a given IP destination address
- Example: routing table may contain
  - 205.100.0.0/22 which corresponds to a given supernet
  - 205.100.0.0/20 which results from aggregation of a larger number of destinations into a supernet
  - Packet must be routed using the *more specific route*, that is, the longest prefix match

new address allocation policy

- Class A and B assigned only for clearly demonstrated need
- *Consecutive* blocks of class C assigned (up to 64 blocks)
  - All IP addresses in the range have a common **prefix**, and every address with that prefix is within the range
  - Arbitrary prefix length for network ID improves efficiency
- Lower half of class C space assigned to regional authorities
  - More hierarchical allocation of addresses
  - Service provider to customer

Address Requirement	Address Allocation
< 256	1 Class C
256<,<512	2 Class C
512<,<1024	4 Class C
1024<,<2048	8 Class C
2048<,<4096	16 Class C
4096<,<8192	32 Class C
8192<,<16384	64 Class C

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

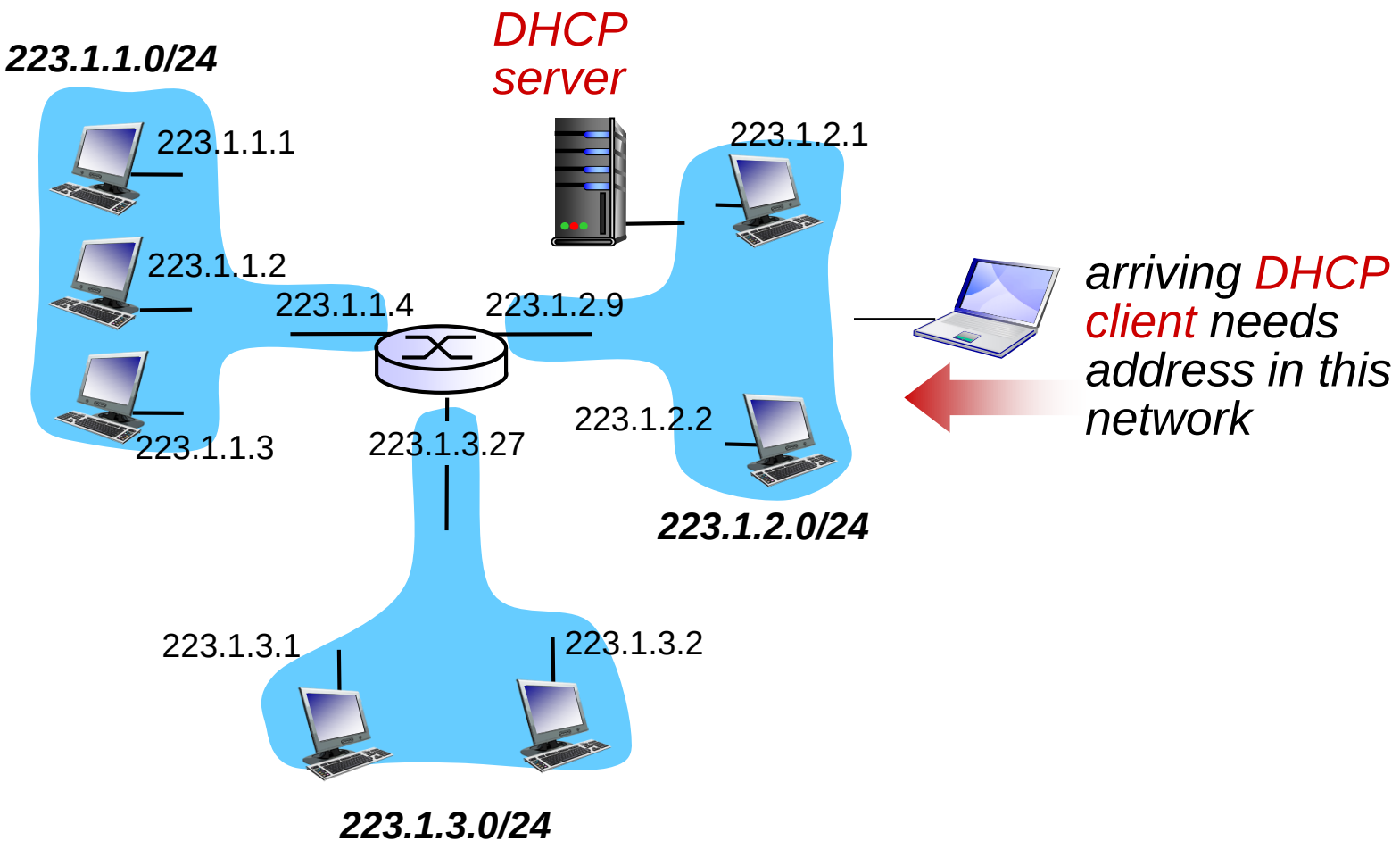
- hard-coded by system admin in a file
  - Windows: control-panel->network->configuration->tcp/ip->properties
  - UNIX: /etc/rc.config
- Dynamic Host Configuration Protocol (DHCP): dynamically get address from as server
  - "plug-and-play"

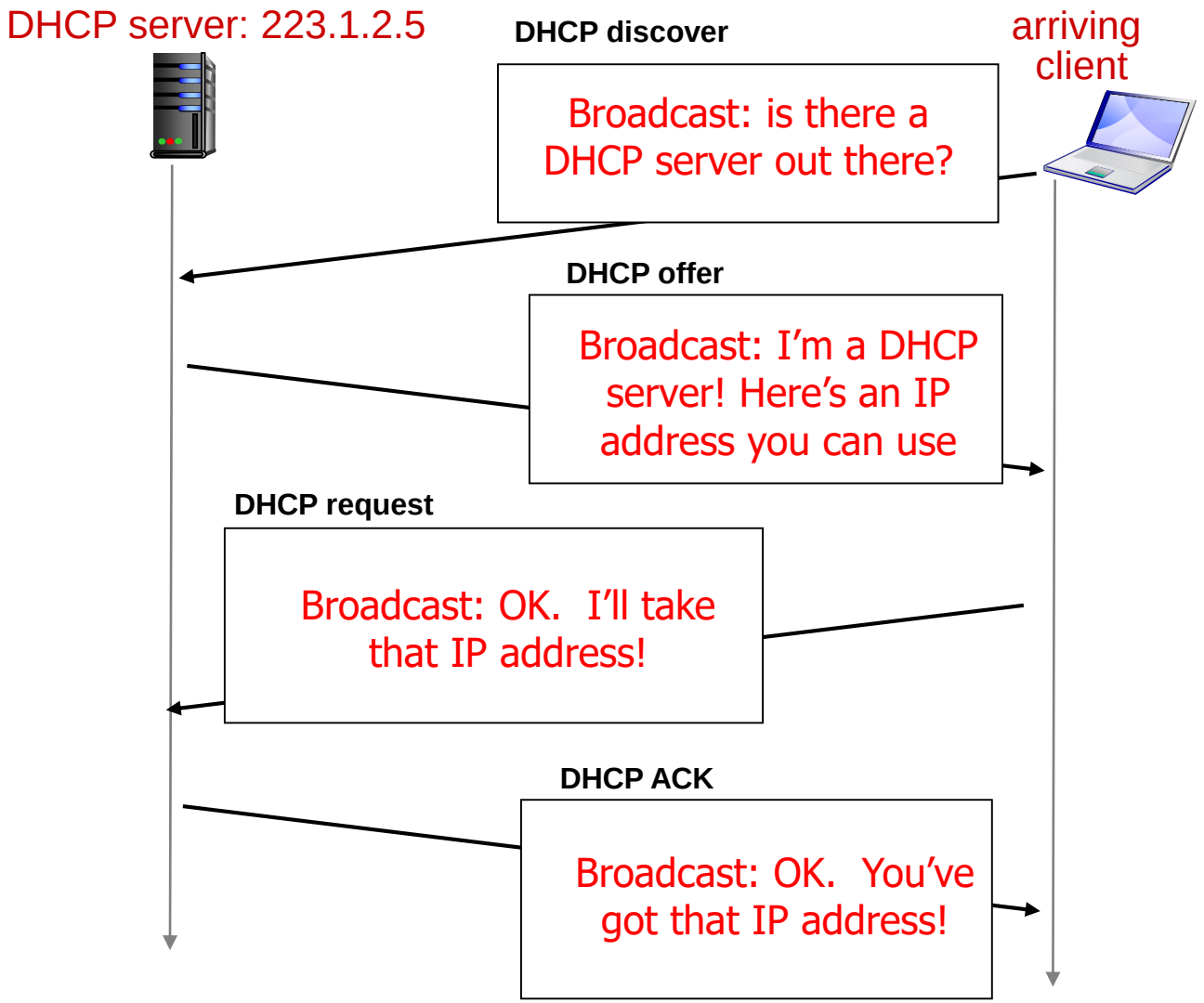
goal: allow host to *dynamically* obtain its IP address from network server when it joins network

- 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

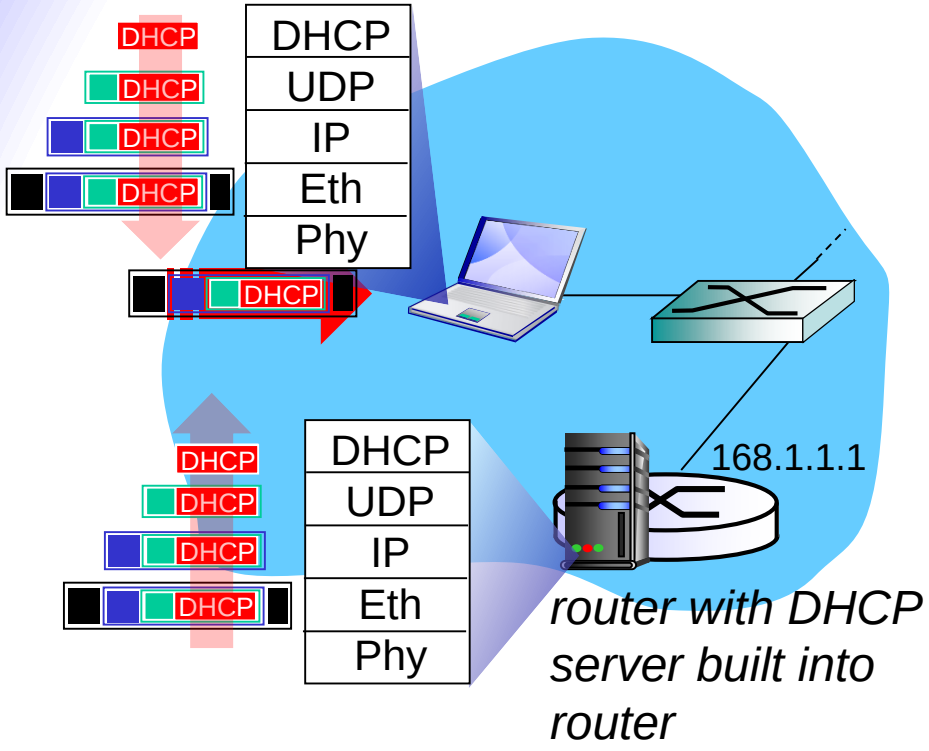

223.1.1.0/24
DHCP server
223.1.1.1
223.1.2.1
223.1.1.2
223.1.1.4
223.1.2.9
arriving DHCP client needs address in this network
223.1.1.3
223.1.3.27
223.1.2.2
223.1.2.0/24
223.1.3.1
223.1.3.2
223.1.3.0/24


DHCP server: 223.1.2.5
DHCP discover
arriving client
Broadcast: is there a DHCP server out there?
DHCP offer
Broadcast: I'm a DHCP server! Here's an IP address you can use
DHCP request
Broadcast: OK. I'll take that IP address!
DHCP ACK
Broadcast: OK. You've got that IP address!

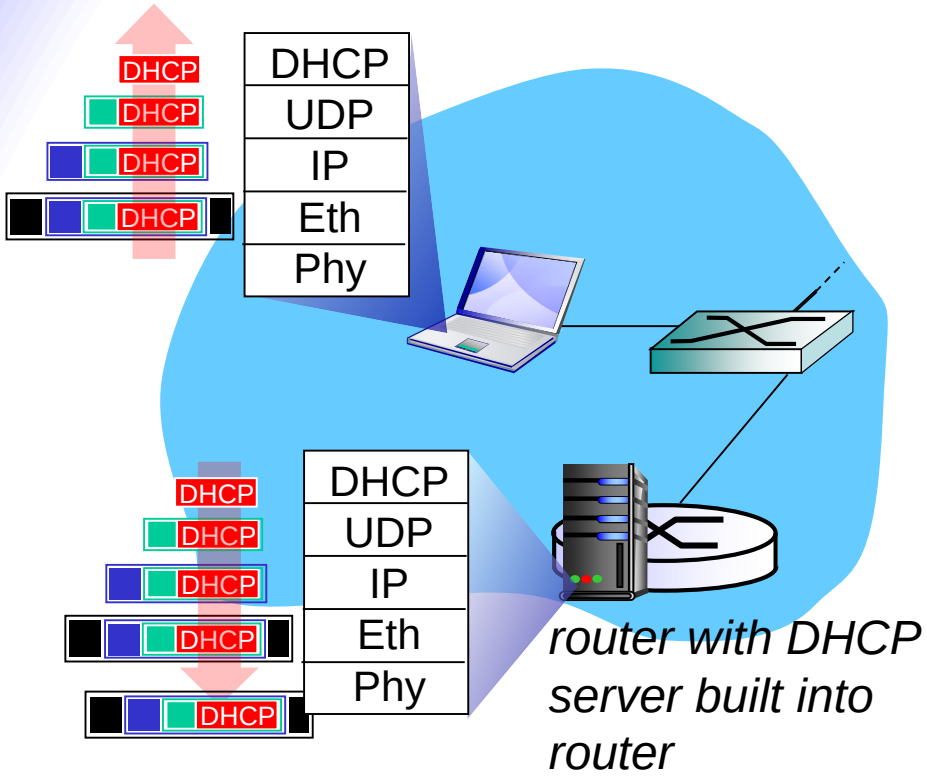
DHCP can return more than just allocated IP address on subnet:

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

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



- DHCP 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

request

Message type: **Boot Request (1)**
Hardware type: Ethernet
Hardware address length: 6
Hops: 0
Transaction ID: 0x6b3a11b7
Seconds elapsed: 0
Bootp flags: 0x0000 (Unicast)
Client IP address: 0.0.0.0 (0.0.0.0)
Your (client) IP address: 0.0.0.0 (0.0.0.0)
Next server IP address: 0.0.0.0 (0.0.0.0)
Relay agent IP address: 0.0.0.0 (0.0.0.0)
Client MAC address: Wistron_23:68:8a (00:16:d3:23:68:8a)
Server host name not given
Boot file name not given
Magic cookie: (OK)
Option: (t=53,l=1) **DHCP Message Type = DHCP Request**
Option: (61) Client identifier
    Length: 7; Value: 010016D323688A;
    Hardware type: Ethernet
    Client MAC address: Wistron_23:68:8a (00:16:d3:23:68:8a)
Option: (t=50,l=4) Requested IP Address = 192.168.1.101
Option: (t=12,l=5) Host Name = "nomad"
Option: (55) Parameter Request List
    Length: 11; Value: 010F03062C2E2F1F21F92B
    1 = Subnet Mask; 15 = Domain Name
    3 = Router; 6 = Domain Name Server
    44 = NetBIOS over TCP/IP Name Server
    ……

reply

Message type: **Boot Reply (2)**
Hardware type: Ethernet
Hardware address length: 6
Hops: 0
Transaction ID: 0x6b3a11b7
Seconds elapsed: 0
Bootp flags: 0x0000 (Unicast)
Client IP address: 192.168.1.101 (192.168.1.101)
Your (client) IP address: 0.0.0.0 (0.0.0.0)
Next server IP address: 192.168.1.1 (192.168.1.1)
Relay agent IP address: 0.0.0.0 (0.0.0.0)
Client MAC address: Wistron_23:68:8a (00:16:d3:23:68:8a)
Server host name not given
Boot file name not given
Magic cookie: (OK)
Option: (t=53,l=1) DHCP Message Type = DHCP ACK
Option: (t=54,l=4) Server Identifier = 192.168.1.1
Option: (t=1,l=4) Subnet Mask = 255.255.255.0
Option: (t=3,l=4) Router = 192.168.1.1
Option: (6) Domain Name Server
    Length: 12; Value: 445747E2445749F244574092;
    IP Address: 68.87.71.226;
    IP Address: 68.87.73.242;
    IP Address: 68.87.64.146
Option: (t=15,l=20) Domain Name = "hsd1.ma.comcast.net."

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

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

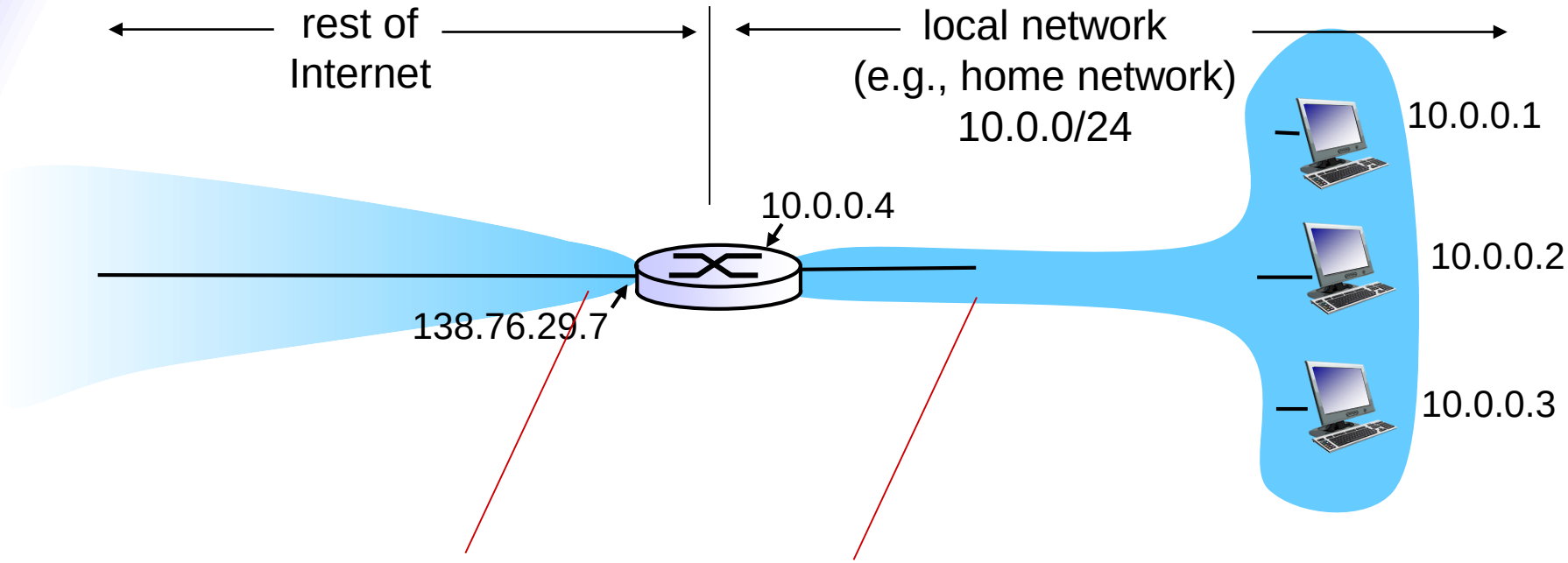
ISP's block	11001000 00010111 00010000	00000000	200.23.16.0/20
Organization 0	11001000 00010111 00010000	00000000	200.23.16.0/23
Organization 1	11001000 00010111 00010010	00000000	200.23.18.0/23
Organization 2	11001000 00010111 00010100	00000000	200.23.20.0/23
...	.....	....	....
Organization 7	11001000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

Q: how does an ISP get block of addresses?

A: Internet Corporation for Assigned Names and Numbers (ICANN)
http://www.icann.org/

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

network address translation (NAT)



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)

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)

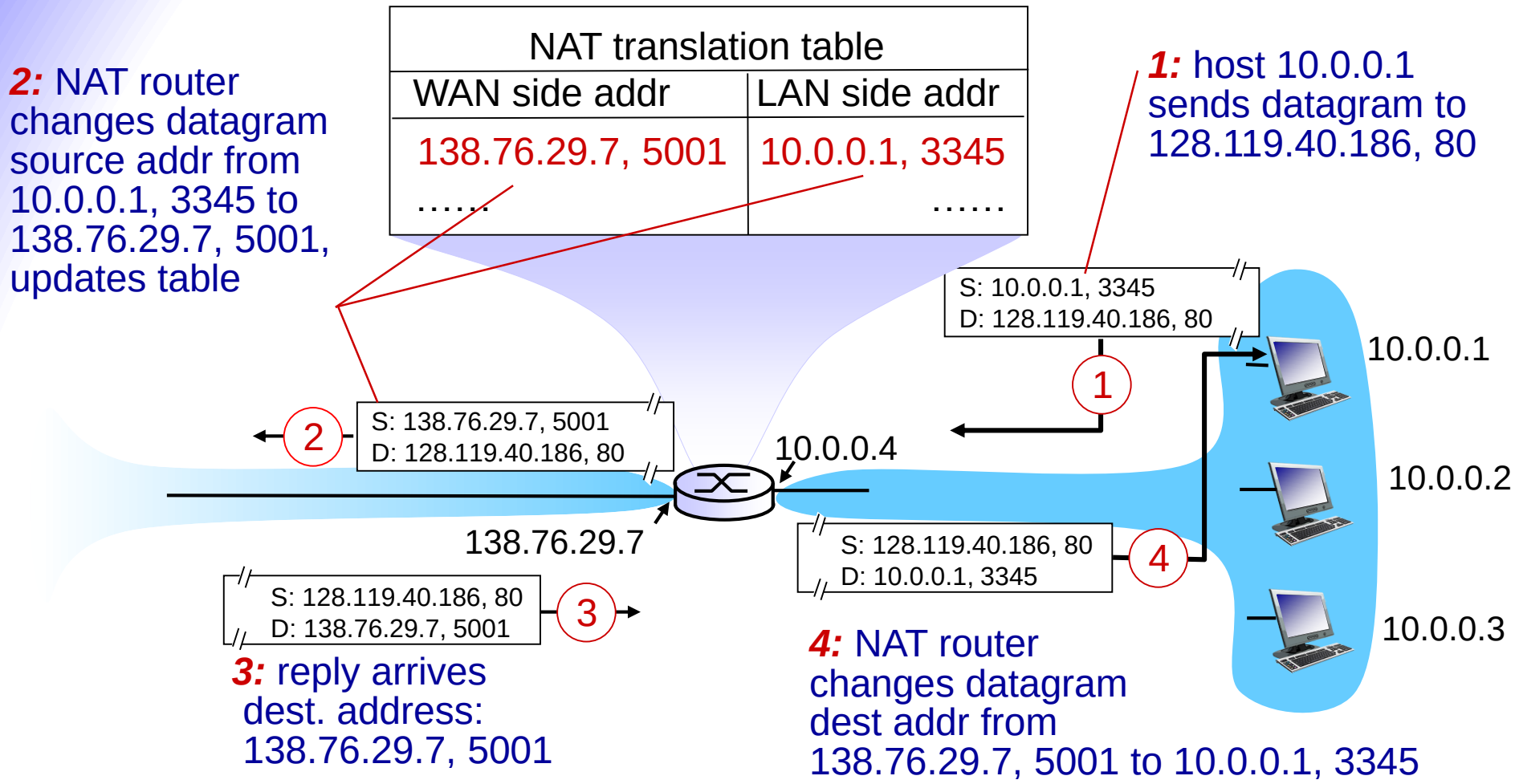
implementation: NAT router must:

- *outgoing datagrams: replace* (source IP address, port number) of every outgoing datagram to (NAT IP address, new port number)

  . . . remote clients/servers will respond using (NAT IP address, new port number) as destination address

- *remember (in NAT translation table)* every (source IP address, port number) to (NAT IP address, new port number) translation pair

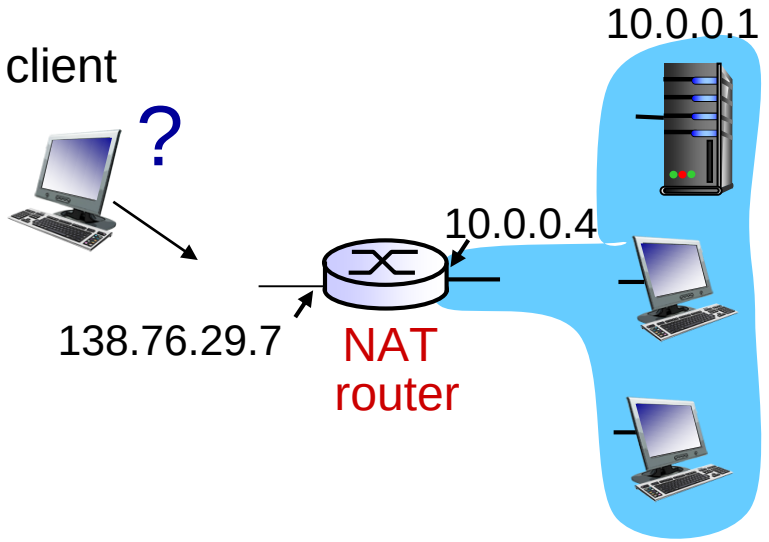
- *incoming datagrams: replace* (NAT IP address, new port number) in destinatiion fields of every incoming datagram with corresponding (source IP address, port number) stored in NAT table



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

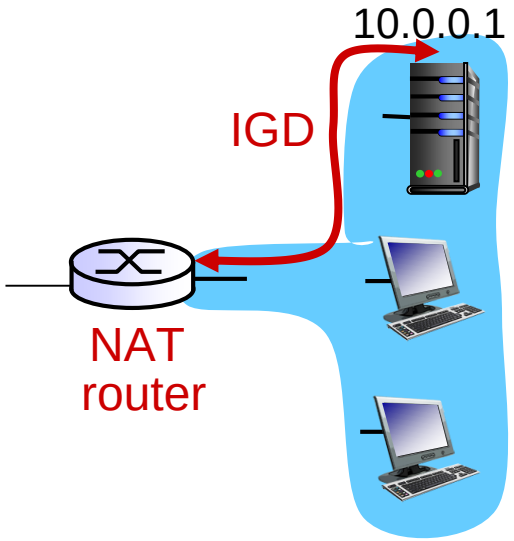
- 16-bit port-number field:
  - 60,000 simultaneous connections with a single LAN-side address!
- NAT is controversial:
  - routers should only process up to layer 3
  - address shortage should be solved by IPv6
  - violates end-to-end argument
    - NAT possibility must be taken into account by app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

- client wants to connect to server with address 10.0.0.1
  - server address 10.0.0.1 local to LAN (client cannot use it as destination address)
  - 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., (123.76.29.7, port 2500) always forwarded to 10.0.0.1 port 25000

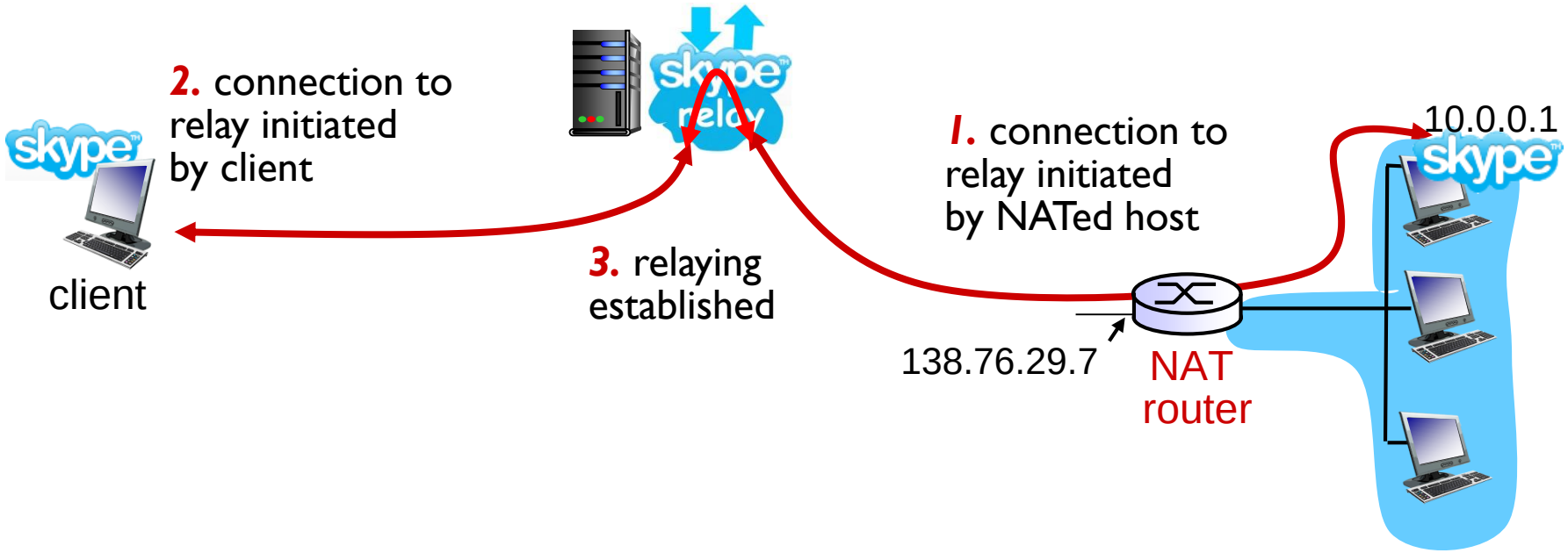


- *solution 2:* Universal Plug and Play (UPnP) Internet Gateway Device (IGD) Protocol. Allows NATed host to:
  - learn public IP address (138.76.29.7)
  - add/remove port mappings (with lease times)

  i.e., automate static NAT port map configuration



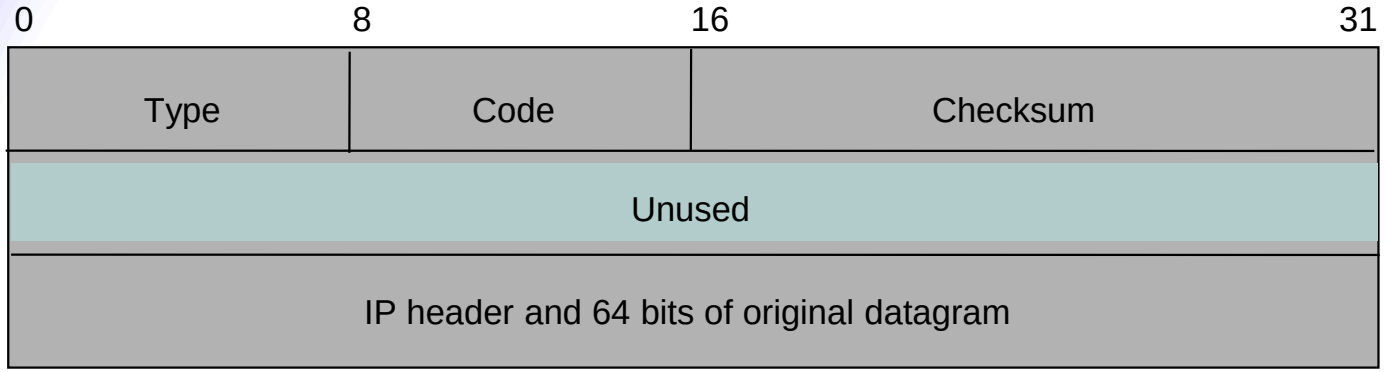
- *solution 3:* relaying (used in Skype)
  - NATed client establishes connection to relay
  - external client connects to relay
  - relay bridges packets between to connections



Internet Control Message Protocol (ICMP)

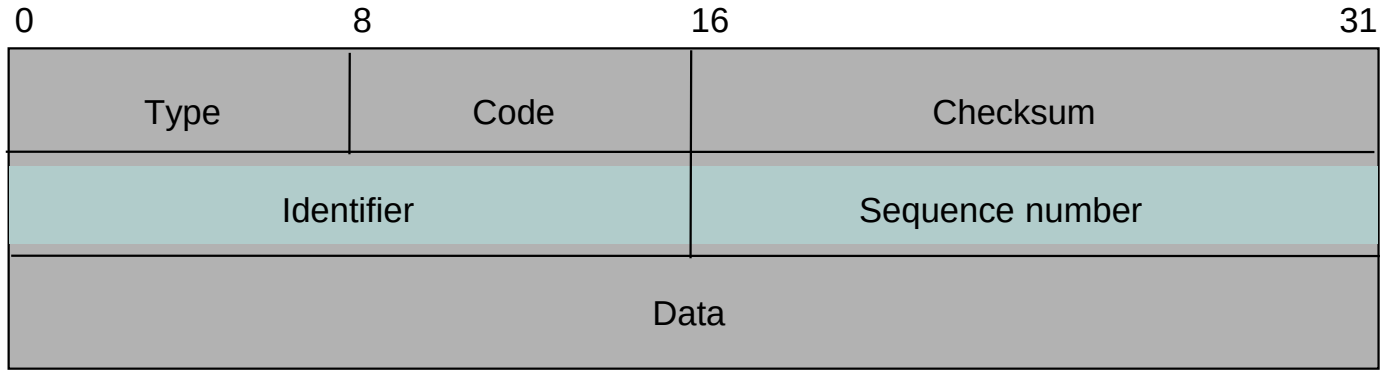
- RFC 792; Encapsulated in IP packet (protocol type = 1)
- Handles error and control messages
- If router cannot deliver or forward a packet, it sends an ICMP “host unreachable” message to the source
- If router receives packet that should have been sent to another router, it sends an ICMP “redirect” message to the sender; Sender modifies its routing table
- ICMP “router discovery” messages allow host to learn about routers in its network and to initialize and update its routing tables
- ICMP echo request and reply facilitate diagnostic and used in “ping”

basic error message format



- *Type* of message: e.g.: Type=3, Destination Unreachable
- Code: purpose of message, e.g.: for Type=3
  - 0 Network Unreachable;
  - 1 Host Unreachable
  - 2 Protocol Unreachable
  - 3 Port Unreachable
  - 4 Fragmentation needed
  - 5 Source route failed
- IP header and 64 bits of original datagram
  - To match ICMP message with original data in IP packet

echo request and echo reply message format



- Echo request: type=8; Echo reply: type=0
  - Destination replies with echo reply by copying data in request onto reply message
- Sequence number to match reply to request
- ID to distinguish between different sessions using echo services
- Used in PING

IPv6

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

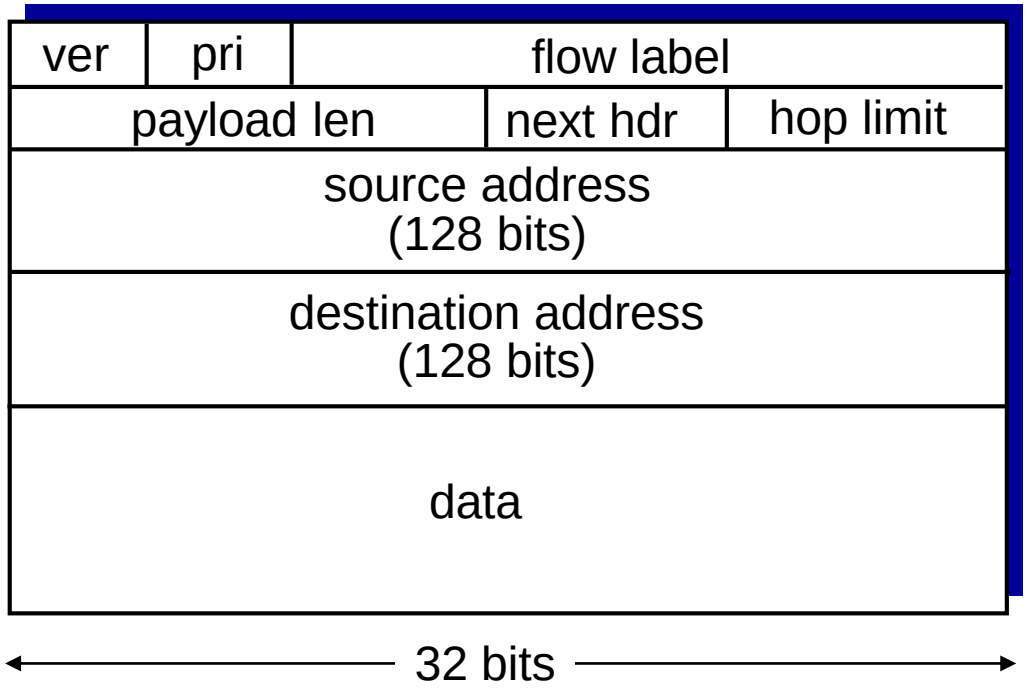
IPv6 datagram format:

- fixed-length 40 byte header
- no fragmentation allowed

priority: identify priority among datagrams in flow

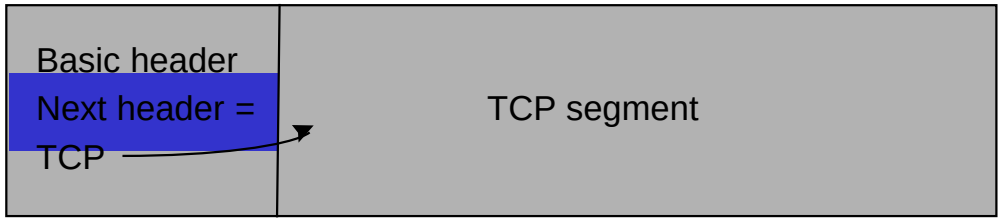
flow Label: identify datagrams in same "flow."
(concept of"flow" not well defined).

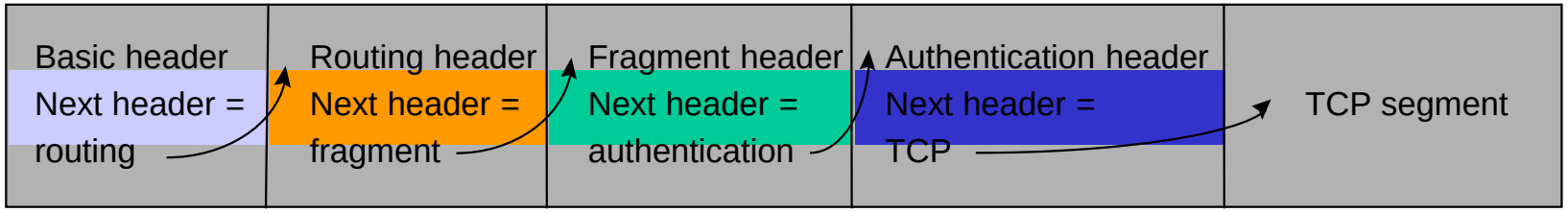
next header: identify upper layer protocol for data



- Address categories
  - Unicast: single network interface
  - Multicast: group of network interfaces, typically at different locations. Packet sent to all.
  - Anycast: group of network interfaces. Packet sent to only one interface in group, e.g. nearest.
- Hexadecimal notation
  - Groups of 16 bits represented by 4 hex digits
  - Separated by colons
    - 4BF5:AA12:0216:FEBC:BA5F:039A:BE9A:2176
  - Shortened forms:
    - 4BF5:0000:0000:0000:BA5F:039A:000A:2176
    - To 4BF5:0:0:0:BA5F:39A:A:2176
    - To 4BF5::BA5F:39A:A:2176
  - Mixed notation:
    - ::FFFF:128.155.12.198

Daisy chains of extension headers

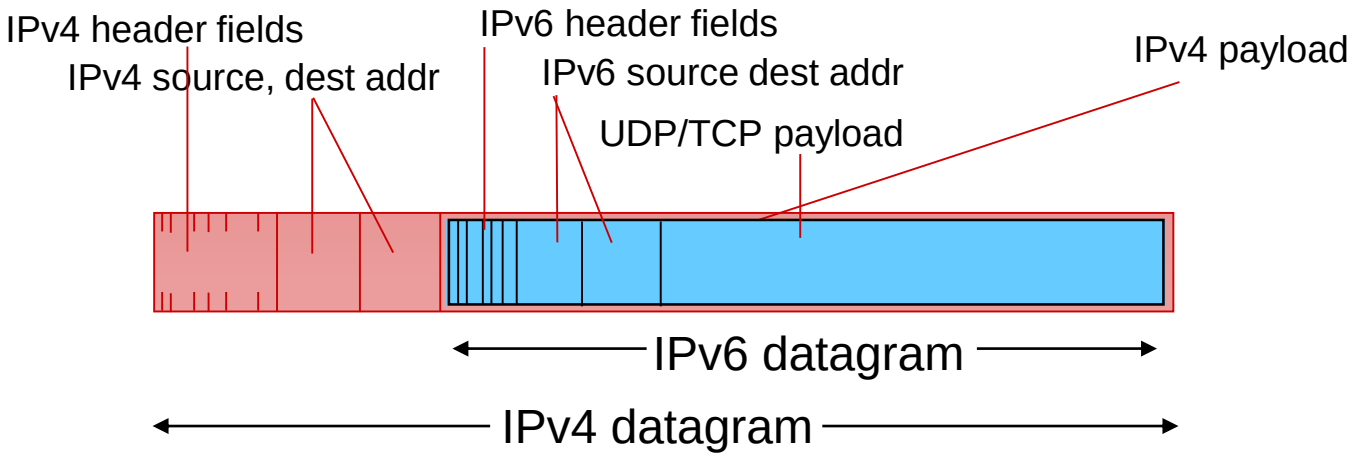


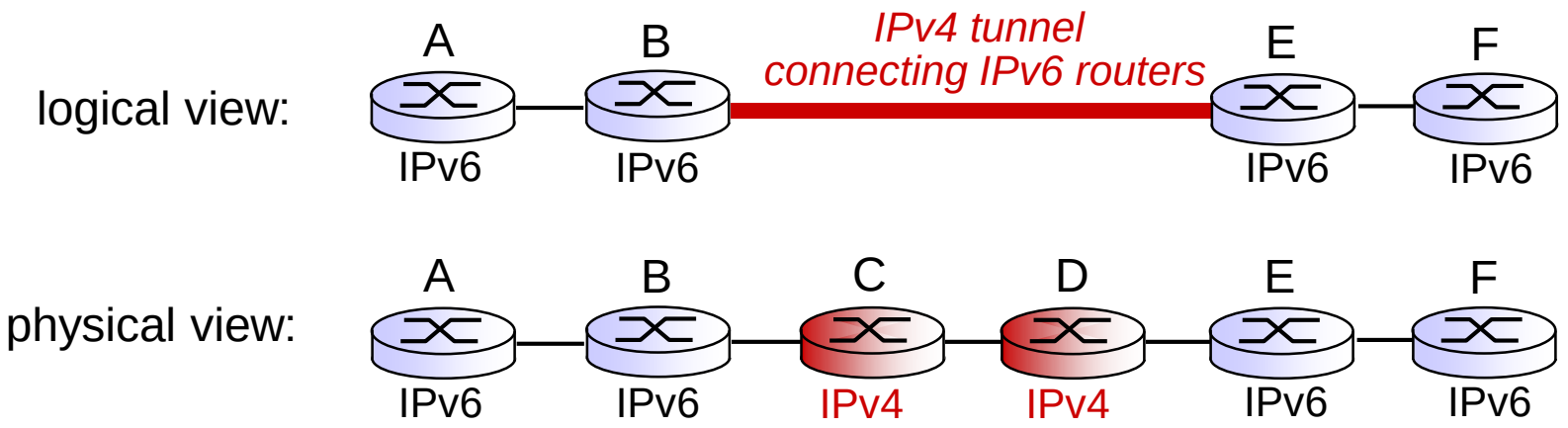


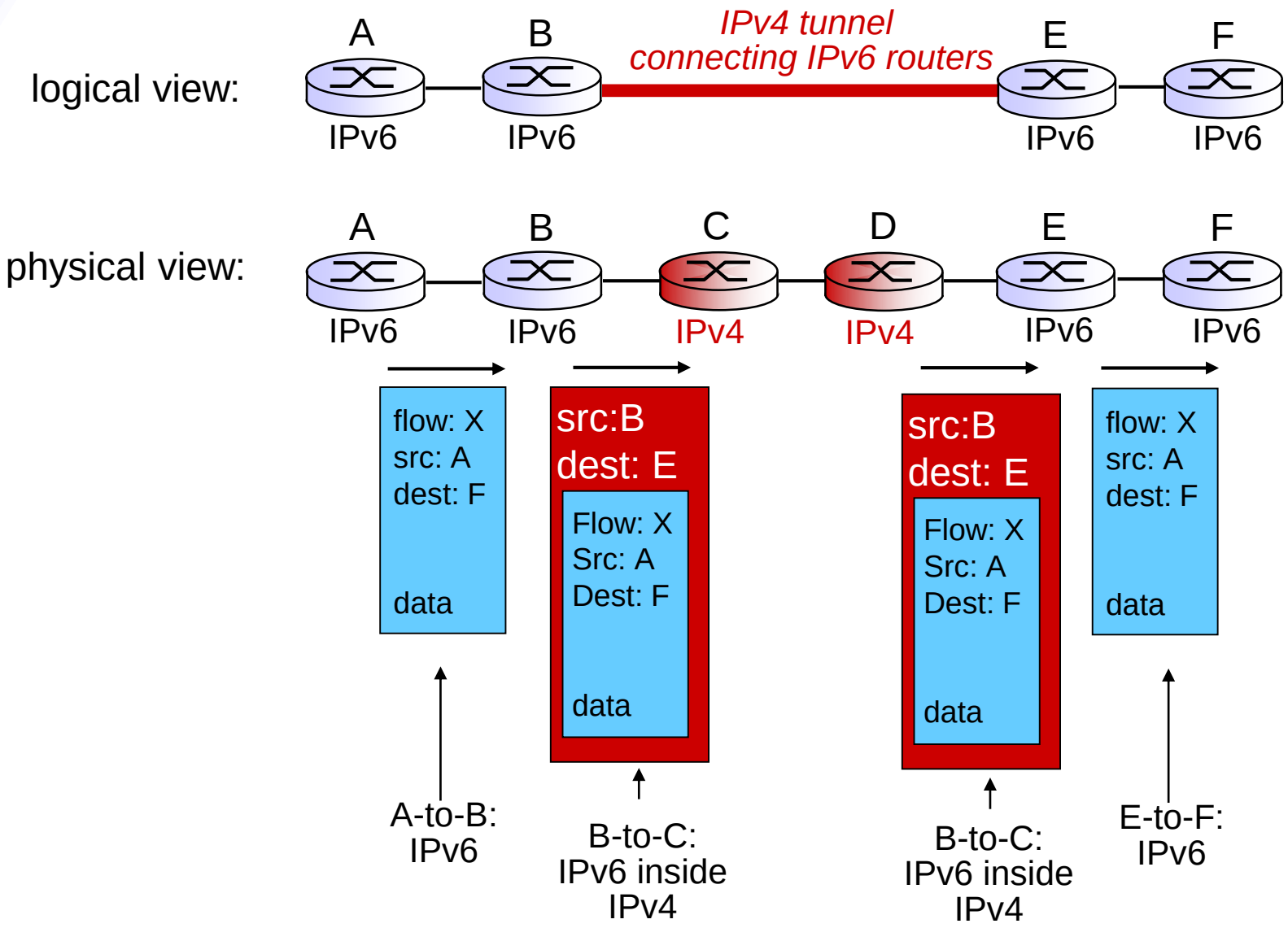
- Extension headers processed in order of appearance

- *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

- 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




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

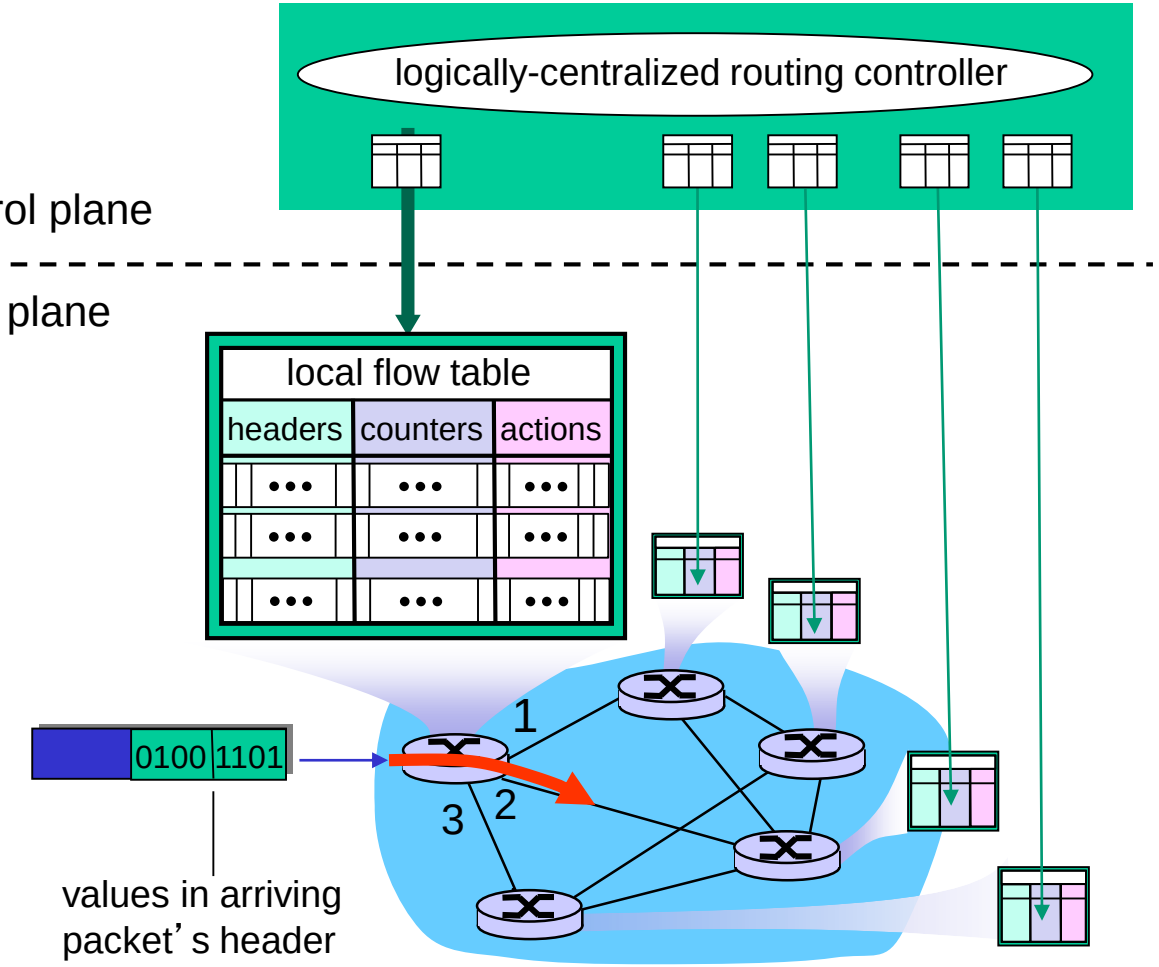

logical view:
A
B
IPv4 tunnel
connecting IPv6 routers
E
F
IPv6
IPv6
IPv6
IPv6
physical view:
A
B
C
D
E
F
IPv6
IPv6
IPv4
IPv4
IPv6
IPv6
flow: X
src: A
dest: F
data
src:B
dest: E
Flow: X
Src: A
Dest: F
data
src:B
dest: E
Flow: X
Src: A
Dest: F
data
flow: X
src: A
dest: F
data
A-to-B:
IPv6
B-to-C:
IPv6 inside
IPv4
B-to-C:
IPv6 inside
IPv4
E-to-F:
IPv6

- Google: 8% of clients access services via IPv6
- NIST: 1/3 of all US government domains are IPv6 capable

- *Long (long!) time for deployment, use*
  - 20 years and counting!
  - think of application-level changes in last 20 years: WWW, Facebook, streaming media, Skype, …
  - *Why?*

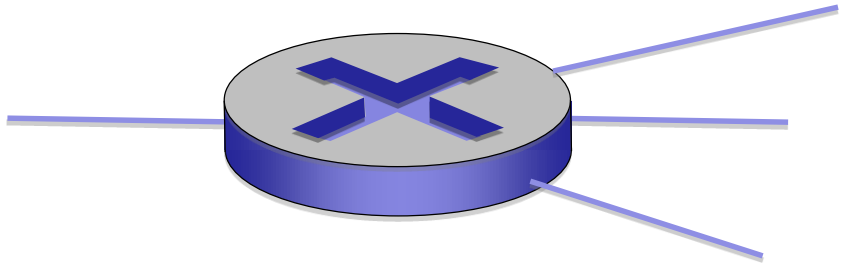
2.8 Generalized forwarding and Software-Defined Networking (SDN)

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



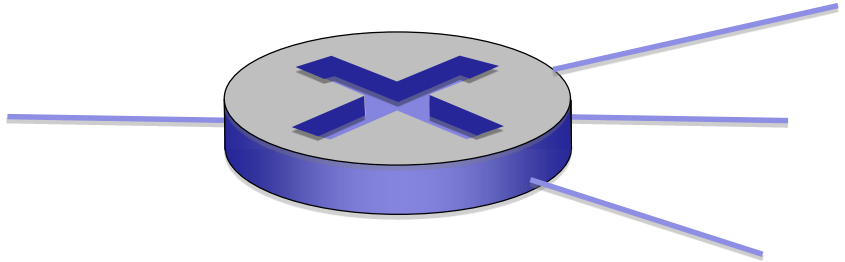
OpenFlow data plane abstraction

- *flow*: defined by header fields
- generalized forwarding: simple packet-handling rules
  - *Pattern:* match values in packet header fields
  - *Actions: for matched packet:* drop, forward, modify, matched packet or send matched packet to controller
  - *Priority*: disambiguate overlapping patterns
  - *Counters:* #bytes and #packets



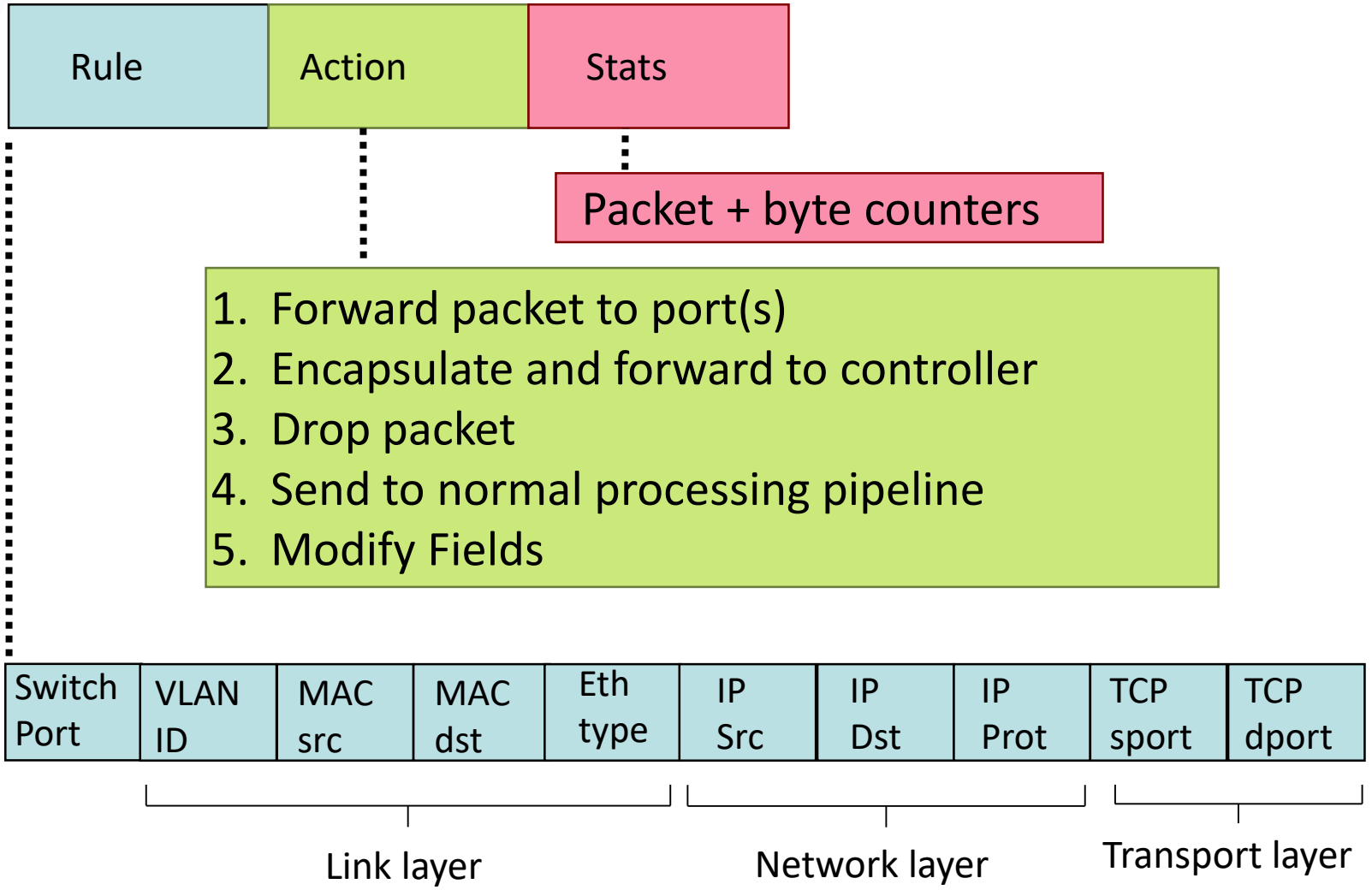
Flow table in a router (computed and distributed by controller) define router's match+action rules

- *flow*: defined by header fields
- generalized forwarding: simple packet-handling rules
  - *Pattern:* match values in packet header fields
  - *Actions: for matched packet:* drop, forward, modify, matched packet or send matched packet to controller
  - *Priority*: disambiguate overlapping patterns
  - *Counters:* #bytes and #packets



* : wildcard

1. src=1.2.*.*, dest=3.4.5.* → drop
2. src = *.*.*.*, dest=3.4.*.* → forward(2)
3. src=10.1.2.3, dest=*.*.*.* → send to controller


Rule
Action
Stats
Packet + byte counters
1. Forward packet to port(s)
2. Encapsulate and forward to controller
3. Drop packet
4. Send to normal processing pipeline
5. Modify Fields
Switch Port
VLAN ID
MAC src
MAC dst
Eth type
IP Src
IP Dst
IP Prot
TCP sport
TCP dport
Link layer
Network layer
Transport layer

Examples

Destination-based forwarding:

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	*	*	*	*	51.6.0.8	*	*	*	port6

IP datagrams destined to IP address 51.6.0.8 should be forwarded to router output port 6

Firewall:

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Forward
*	*	*	*	*	*	*	*	*	22	drop

do not forward (block) all datagrams destined to TCP port 22

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Forward
*	*	*	*	*	128.119.1.1	*	*	*	*	drop

do not forward (block) all datagrams sent by host 128.119.1.1

Destination-based layer 2 (switch) forwarding:

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	22:A7:23: 11:E1:02	*	*	*	*	*	*	*	*	port3

layer 2 frames from MAC address 22:A7:23:11:E1:02 should be forwarded to output port 3

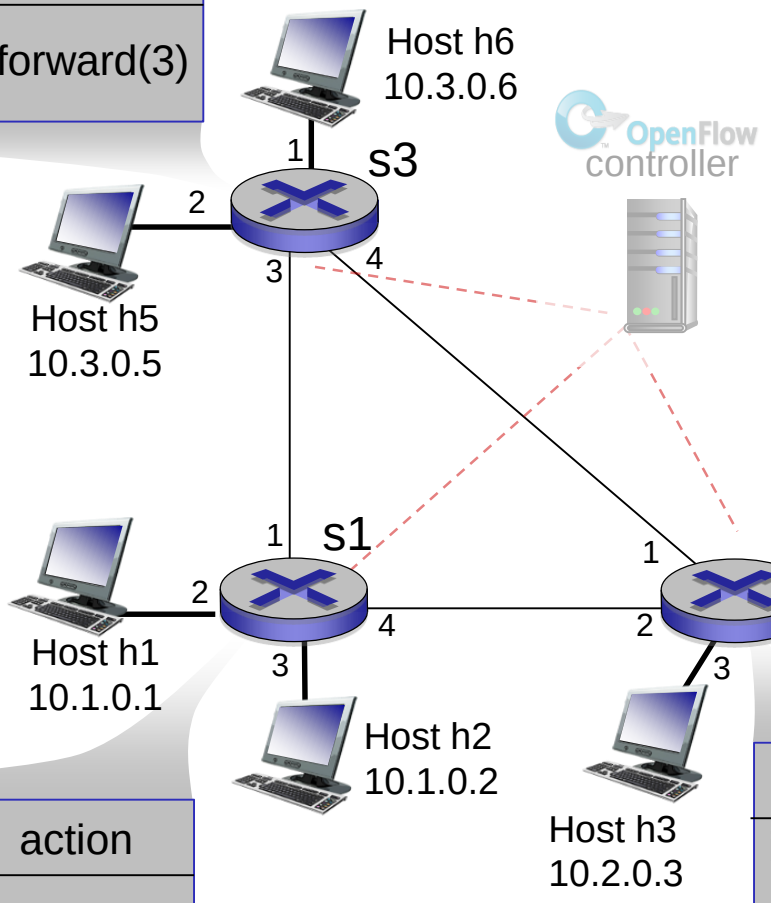
OpenFlow abstraction

- *match+action:* unifies different kinds of devices

- Router
  - *match:* longest destination IP prefix
  - *action:* forward out a link
- Switch
  - *match:* destination MAC address
  - *action:* forward or flood
- Firewall
  - *match*: IP addresses and TCP/UDP port numbers
  - *action:* permit or deny
- NAT
  - *match:* IP address and port
  - *action:* rewrite address and port

Example: datagrams from hosts h5 and h6 should be sent to h3 or h4, via s1 and from there to s2

match	action
IP Src = 10.3.*.* IP Dst = 10.2.*.*	forward(3)



match	action
ingress port = 1 IP Src = 10.3.*.* IP Dst = 10.2.*.*	forward(4)

match	action
ingress port = 2 IP Dst = 10.2.0.3	forward(3)
ingress port = 2 IP Dst = 10.2.0.4	forward(4)

Question: how do forwarding tables (destination-based forwarding) or flow tables (generalized forwarding) computed?

Summary

- network layer protocol – data plane
- router
- IPv4 and IPv6
- generalized forwarding and SDN

Reference

Chapter 4, Computer Networking:
A Top-Down Approach