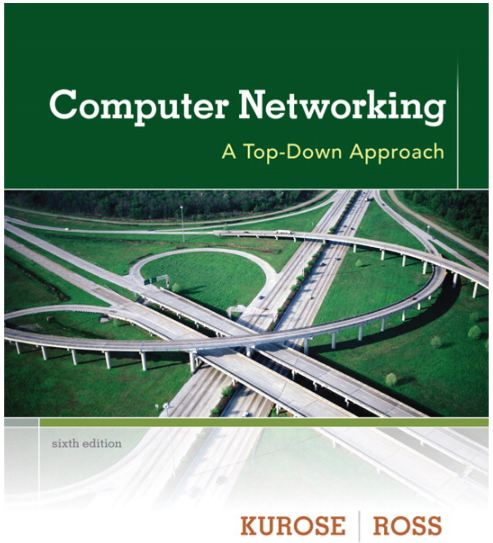
Chapter 4
Network Layer



A note on the use of these ppt 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.

Thanks and enjoy! JFK/KWR

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

Chapter 4: outline

4.1 introduction

4.2 virtual circuit and datagram networks

4.3 IP: Internet Protocol

- datagram format
- IPv4 addressing
- ICMP
- IPv6

4.4 routing algorithms

- link state
- distance vector
- hierarchical routing

4.5 routing in the Internet

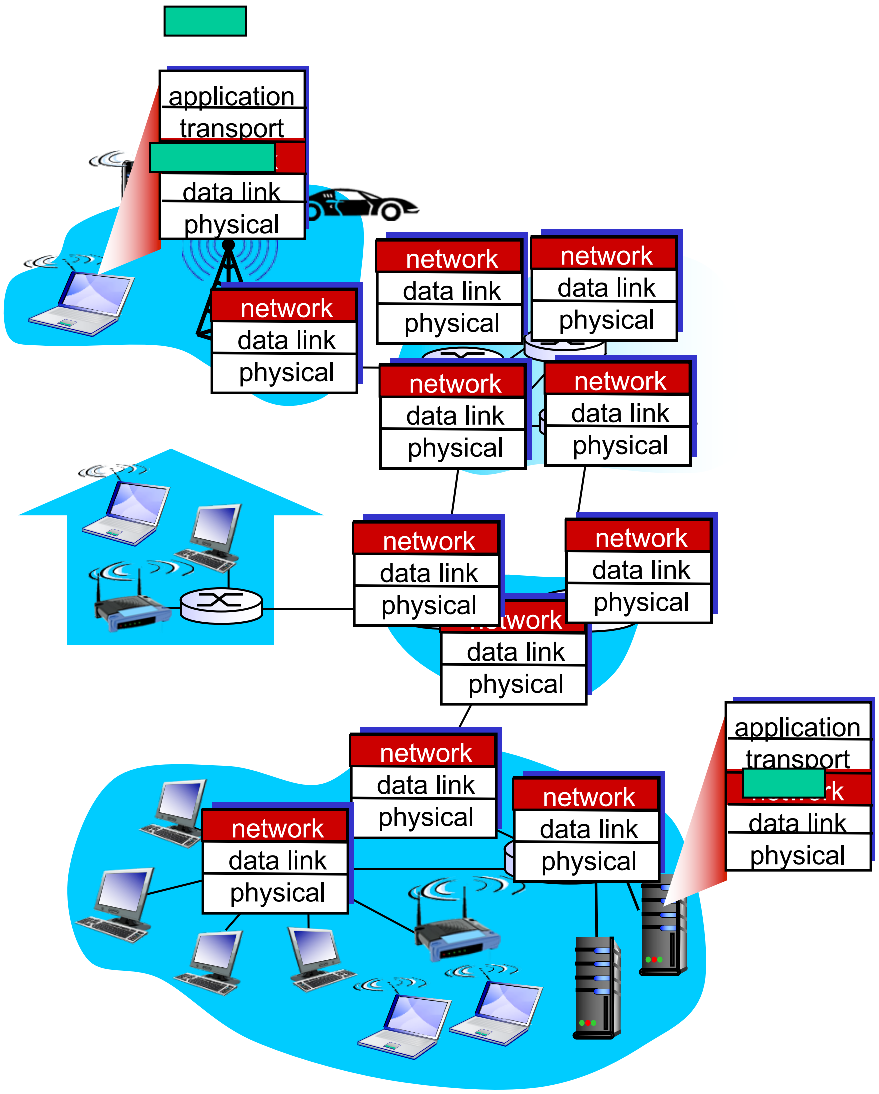
- RIP
- OSPF
- BGP

Link layer issues

- inefficient link-redundancy management
  - Spanning-tree Protocol
- filtering-database saturation on switches in large-size networks
  - route aggregation not possible on switches
- layer-2 broadcast traffic propagation
  - broadcast traffic is useful for several application but *must* be limited

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

Chapter 4: outline

4.1 introduction

4.2 virtual circuit and datagram networks

4.3 IP: Internet Protocol

- datagram format
- IPv4 addressing
- ICMP
- IPv6

4.4 routing algorithms

- link state
- distance vector
- hierarchical routing

4.5 routing in the Internet

- RIP
- OSPF
- BGP

Connection, connection-less service

- *datagram* network provides network-layer *connectionless* service
- *virtual-circuit (VC)* network provides network-layer *connection* service

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)

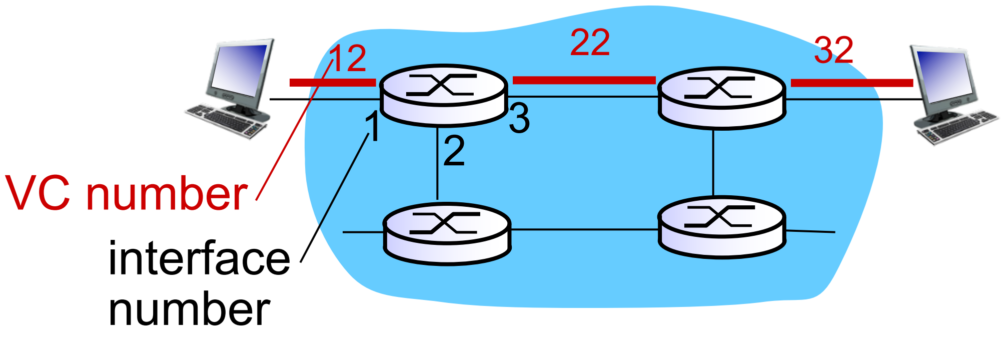
VC implementation

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



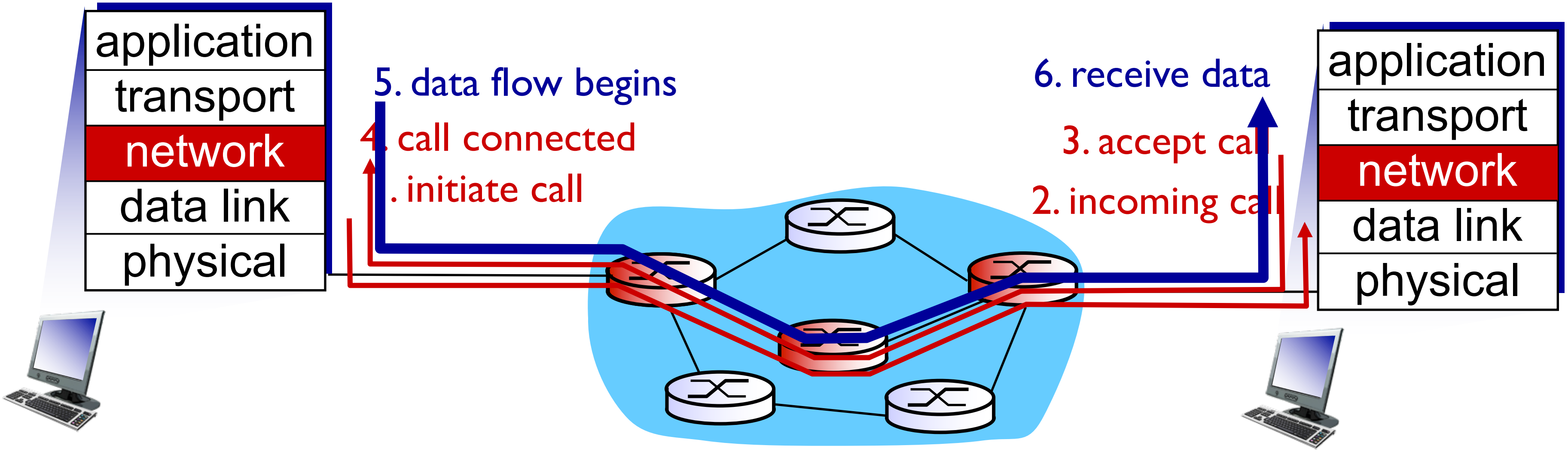
forwarding table in northwest router:

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!

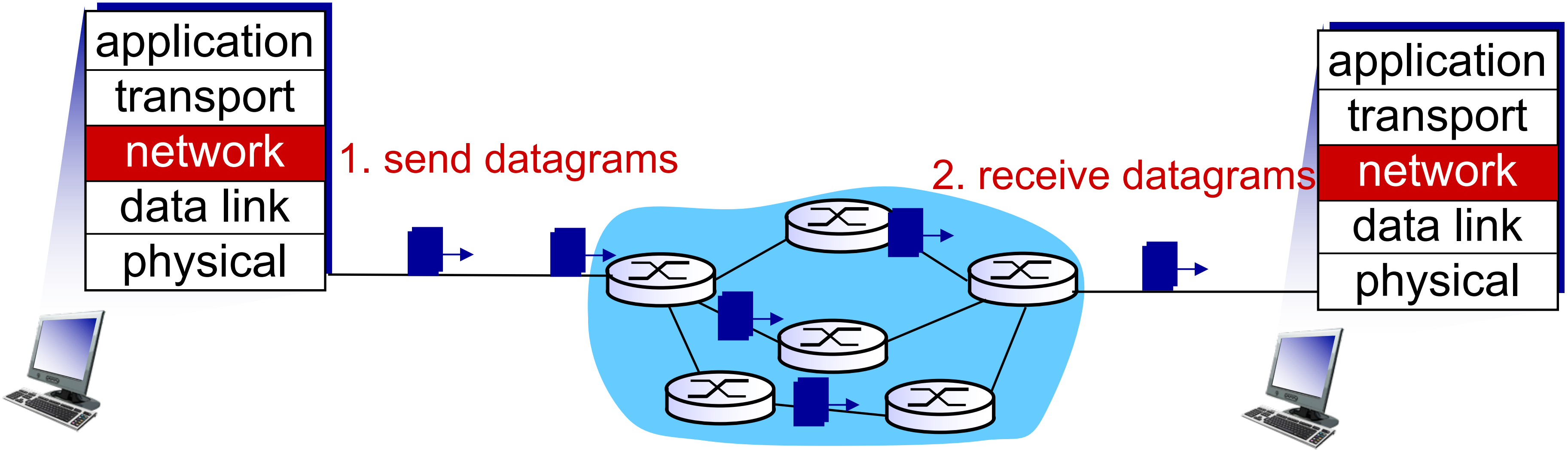
Virtual circuits: signaling protocols

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

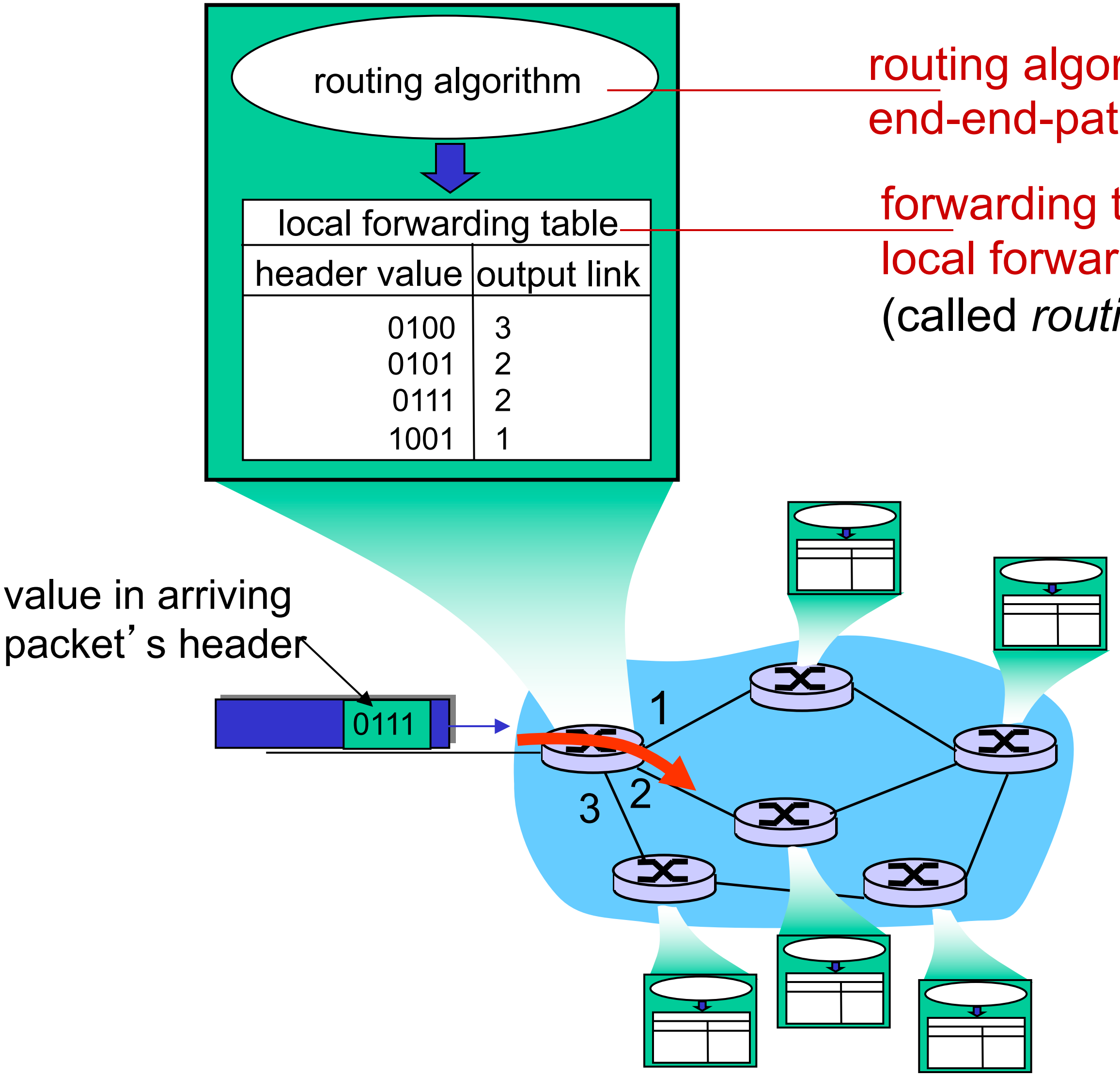


Datagram networks

- 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



local forwarding table	
header value	output link
0100	3
0101	2
0111	2
1001	1

value in arriving packet’s header

routing algorithm determines end-end-path through network

forwarding table determines local forwarding at this router (called *routing table* in IP)

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 IP: Internet Protocol

- datagram format
- IPv4 addressing
- ICMP
- IPv6

4.4 routing algorithms

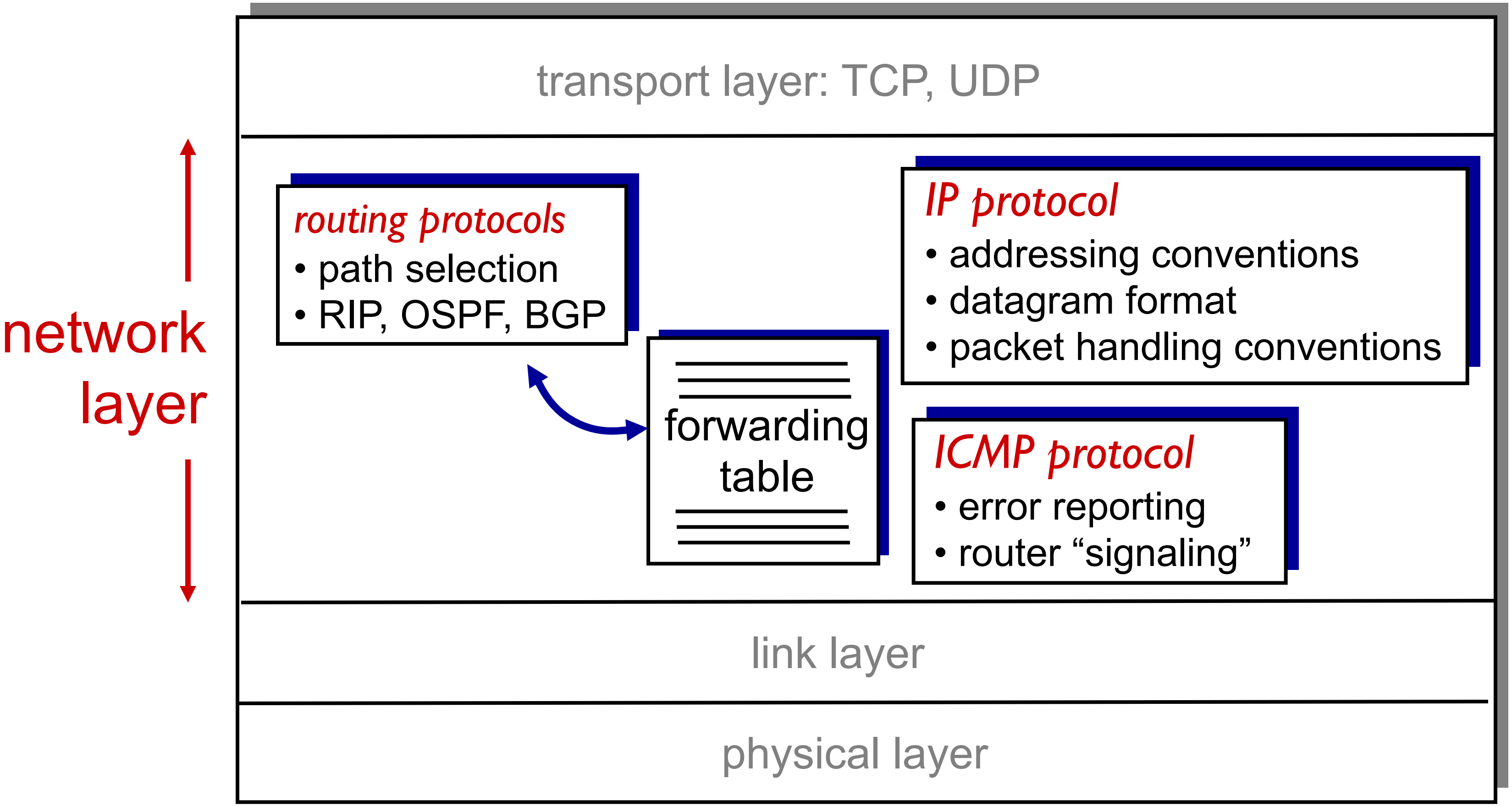
- link state
- distance vector
- hierarchical routing

4.5 routing in the Internet

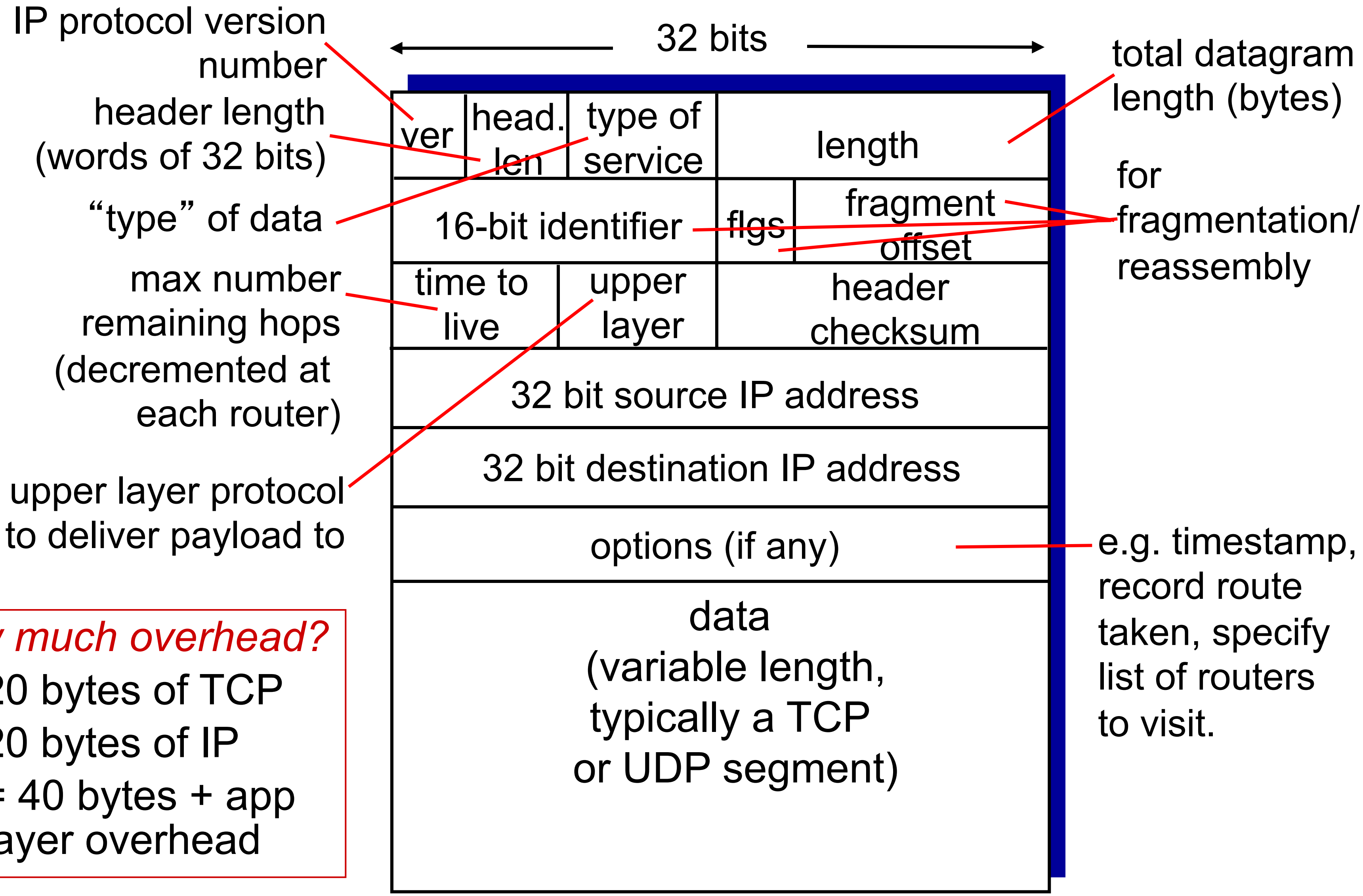
- RIP
- OSPF
- BGP

The Internet network layer

host, router network layer functions:



IP datagram format



IP protocol version number

header length (words of 32 bits)

"type" of data

max number remaining hops (decremented at each router)

upper layer protocol to deliver payload to

32 bits

ver	head. len	type of service	length
16-bit identifier		flgs	fragment offset
time to live	upper layer	header checksum	
32 bit source IP address			
32 bit destination IP address			
options (if any)			
data (variable length, typically a TCP or UDP segment)			

total datagram length (bytes)

for fragmentation/ reassembly

e.g. timestamp, record route taken, specify list of routers to visit.

how much overhead?

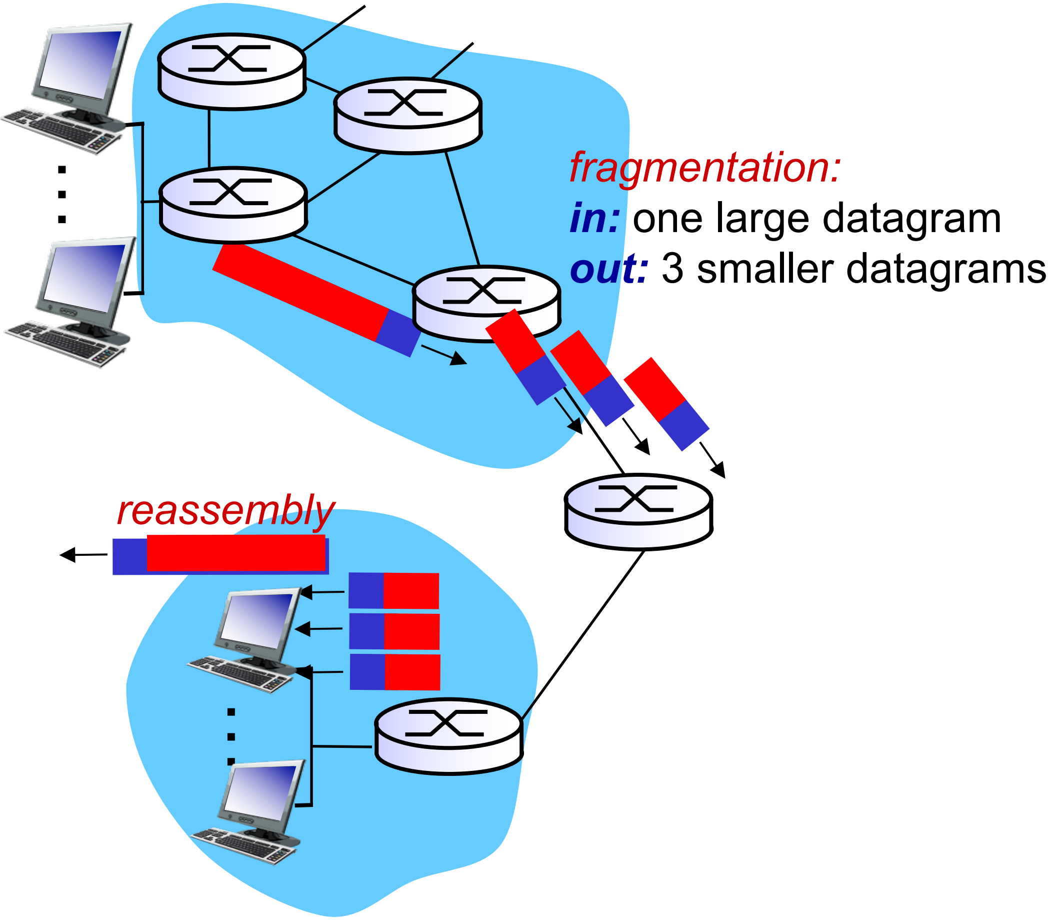
- 20 bytes of TCP
- 20 bytes of IP
- = 40 bytes + app layer overhead

Header length vs. total length

- *header length* (4 bits) provides the number of 32 bit words forming the IP header
  - required due to the possible presence of IP options
    - e.g., 20 bytes + two 32 bits options → HL = 7
- *total length* (16 bits) provides the total size of the IP datagram, including the header
  - maximum IP datagram size is $2^{16} – 1 = 65353$ bytes
  - required for properly handling fragmentation
  - also useful when padding at layer 2 is used
    - e.g., 46 bytes of Ethernet payload, TL = 40 → padding = 16 bytes

IP fragmentation, reassembly

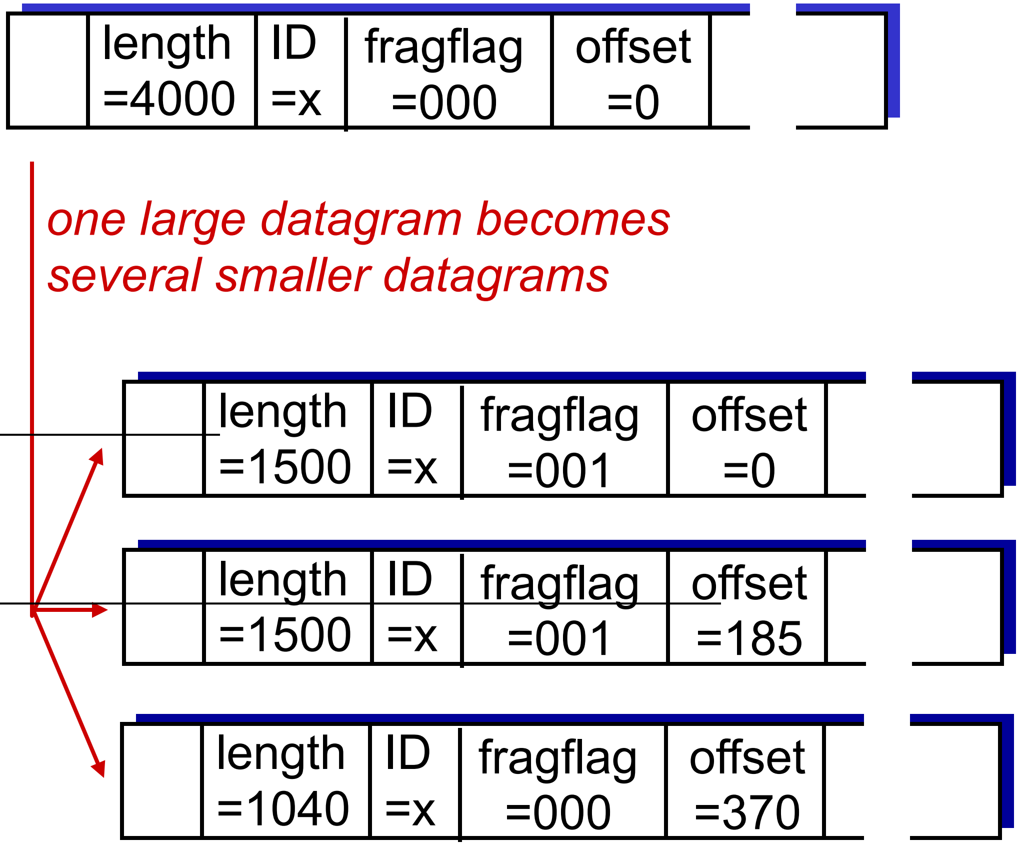
- network links have MTU (max.transfer size) - largest possible link-level frame
  - different link types, different MTUs
- large IP datagram divided ("fragmented") within net
  - one datagram becomes several datagrams
  - "reassembled" only at final destination
  - IP header bits used to identify, order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



IP fragmentation, reassembly

example:

- 4000 byte datagram
- MTU = 1500 bytes



length =4000	ID =x	fragflag =000	offset =0

one large datagram becomes several smaller datagrams

length =1500	ID =x	fragflag =001	offset =0
length =1500	ID =x	fragflag =001	offset =185
length =1040	ID =x	fragflag =000	offset =370

1480 bytes in data field

offset = 1480/8

flags (3 bits)

- 1st reserved (set to 0)
- 2nd set to 1 if don't fragment
- 3rd set to 1 if more fragments

offset: position (in multiple of 8 bytes) of the fragment in the original datagram

Header checksum

- calculated considering all the fields of the IP header
- calculated also at the destination and compared with the value carried in the datagram
  - if same value, datagram delivered to upper layers
  - if not the same, datagram considered corrupted
- re-evaluated at each router, as also TTL changes

Chapter 4: outline

4.1 introduction

4.2 virtual circuit and datagram networks

4.3 IP: Internet Protocol

- datagram format
- IPv4 addressing
- ICMP
- IPv6

4.4 routing algorithms

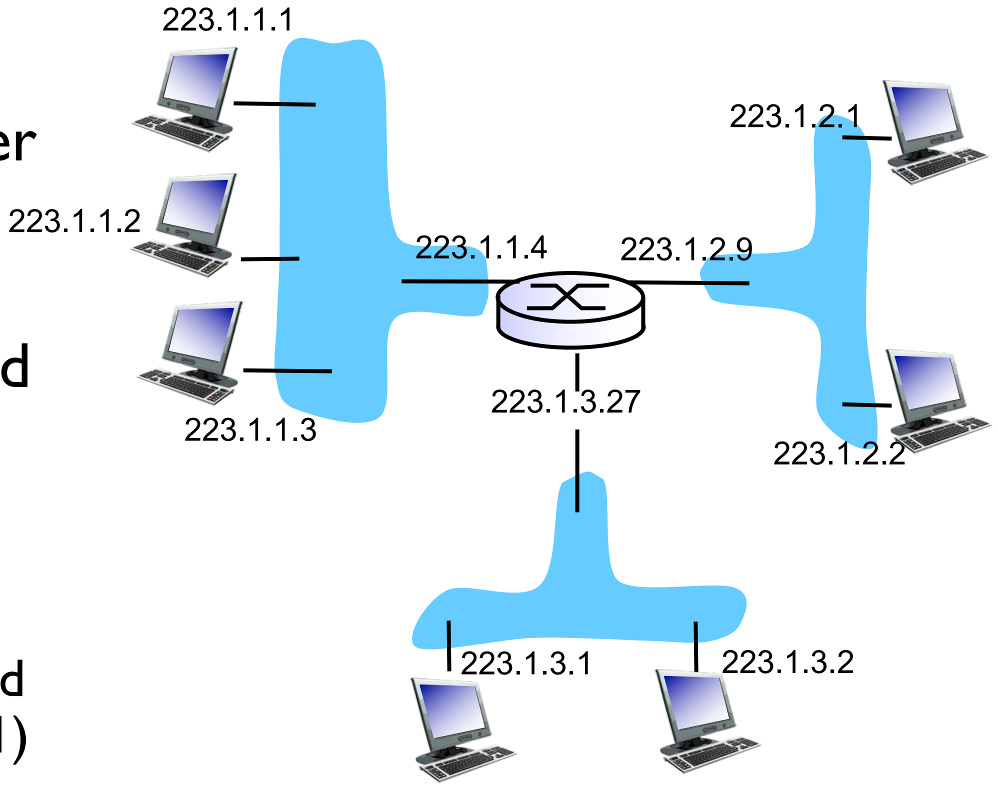
- link state
- distance vector
- hierarchical routing

4.5 routing in the Internet

- RIP
- OSPF
- BGP

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*

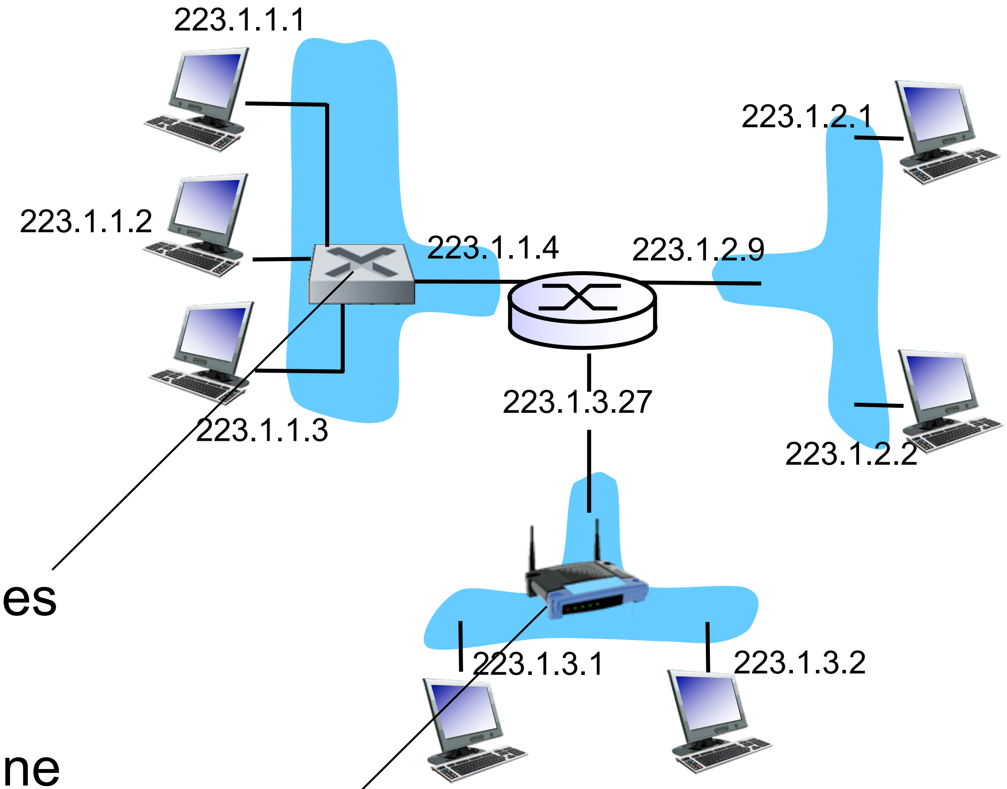


$$223.1.1.1 = \underbrace{11011111}_{223}\ \underbrace{00000001}_{1}\ \underbrace{00000001}_{1}\ \underbrace{00000001}_{1}$$

IP addressing: introduction

Q: how are interfaces actually connected?

A: by means of a proper link layer technology.



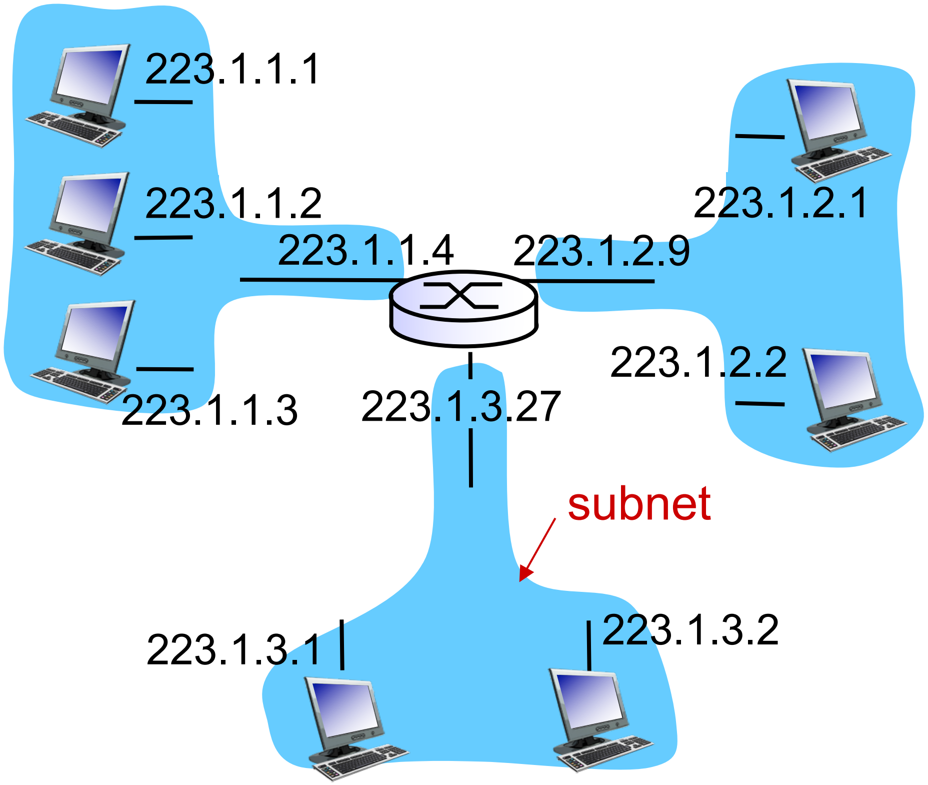
A: wired Ethernet interfaces connected by Ethernet switches

IP does not care about how one interface is connected to another (with no intervening router)

A: wireless WiFi interfaces connected by WiFi base station

Logical IP Subnets (LIS)

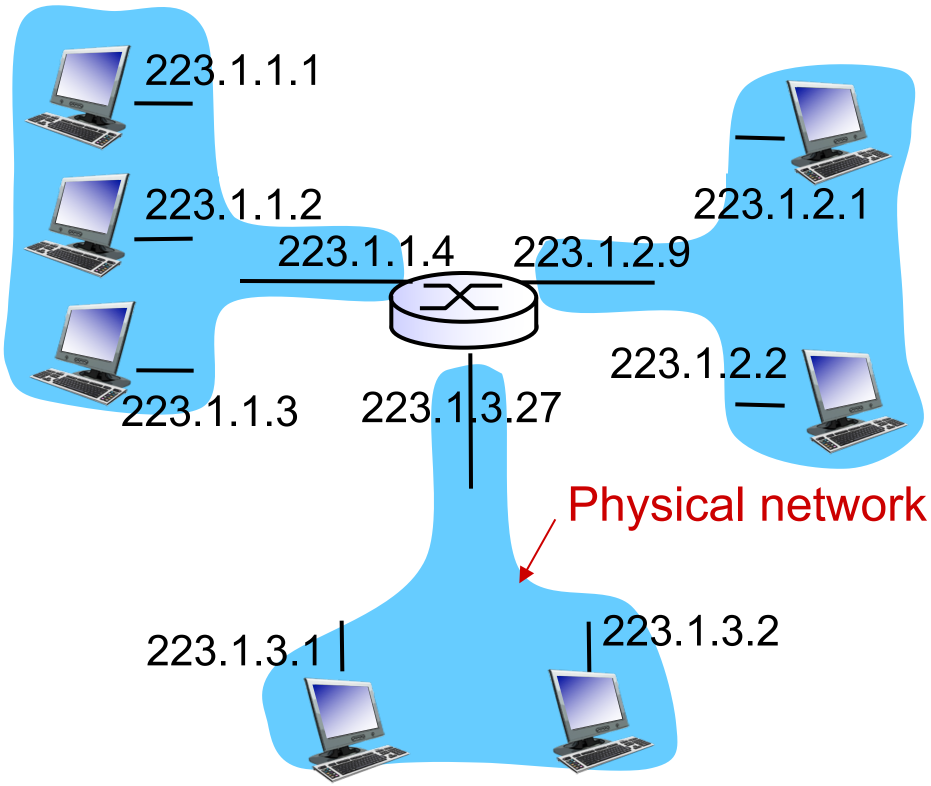
- IP address:
  - subnet part - high order bits
  - host part - low order bits
- *what ’s a subnet ?*
  - set of device interfaces with same subnet part of IP address
  - interfaces *should* physically reach each other. *Why?*



network consisting of 3 subnets

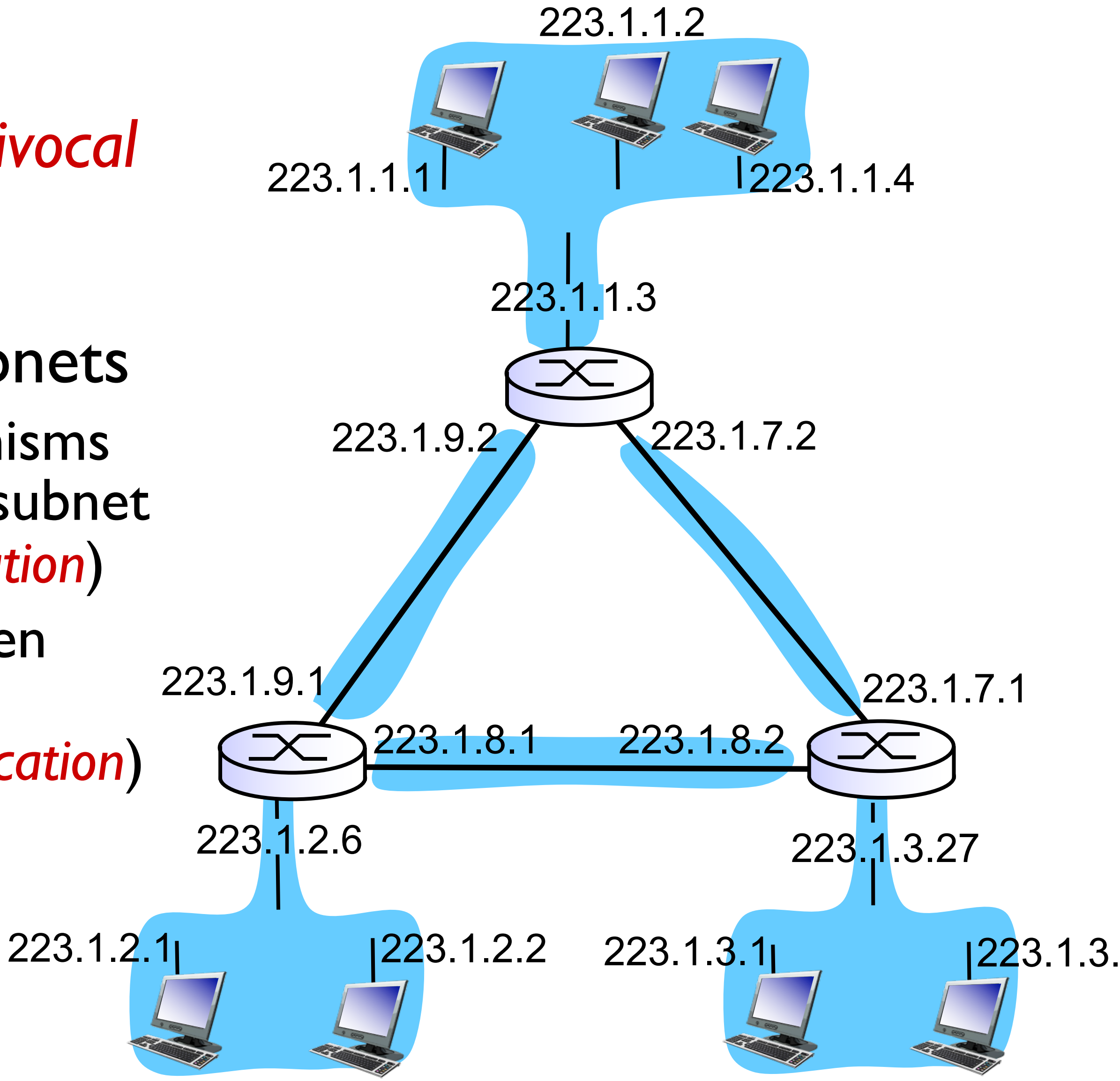
Physical networks

- set of devices that can physically reach each other *by means of link layer mechanisms*
- to determine the physical networks, detach each interface from its router, creating islands of isolated networks



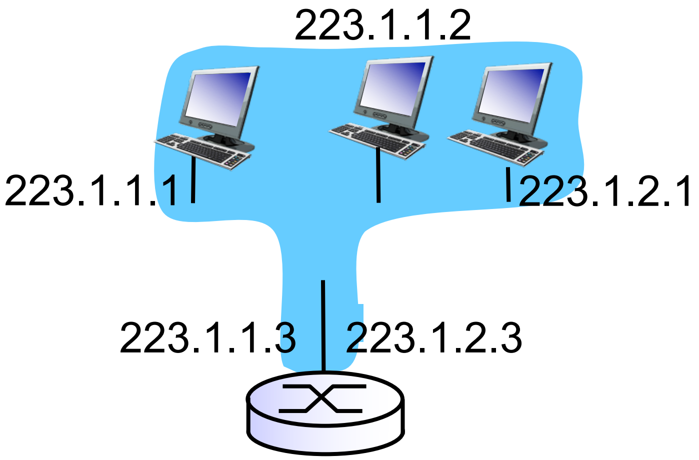
Subnets and physical networks

- IP assumes a *biunivocal correspondence* between physical networks and subnets
  - link layer mechanisms within the same subnet (*direct communication*)
  - IP routing between different subnets (*indirect communication*)

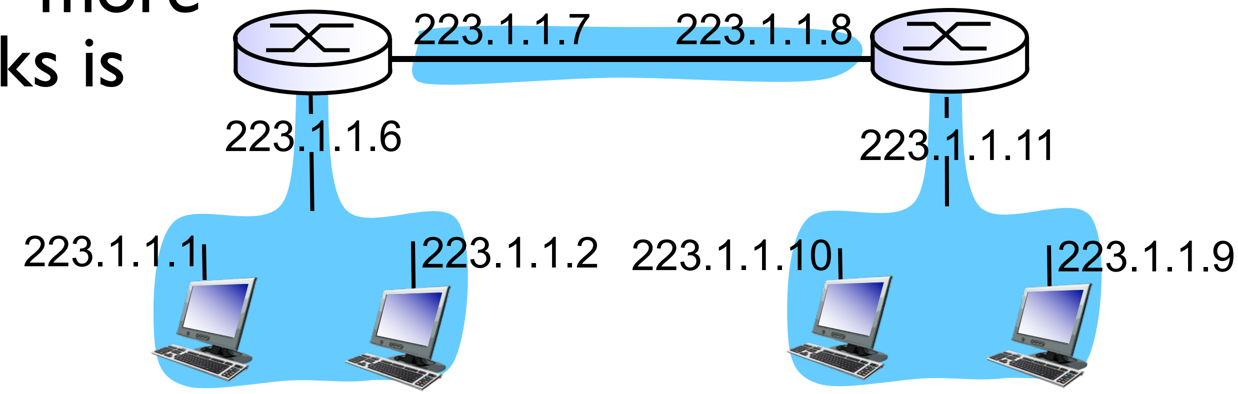


Subnets and physical networks

- more subnets over a single physical network are possible
  - *one-arm router*



- one subnet over more physical networks is possible
  - *proxy ARP*

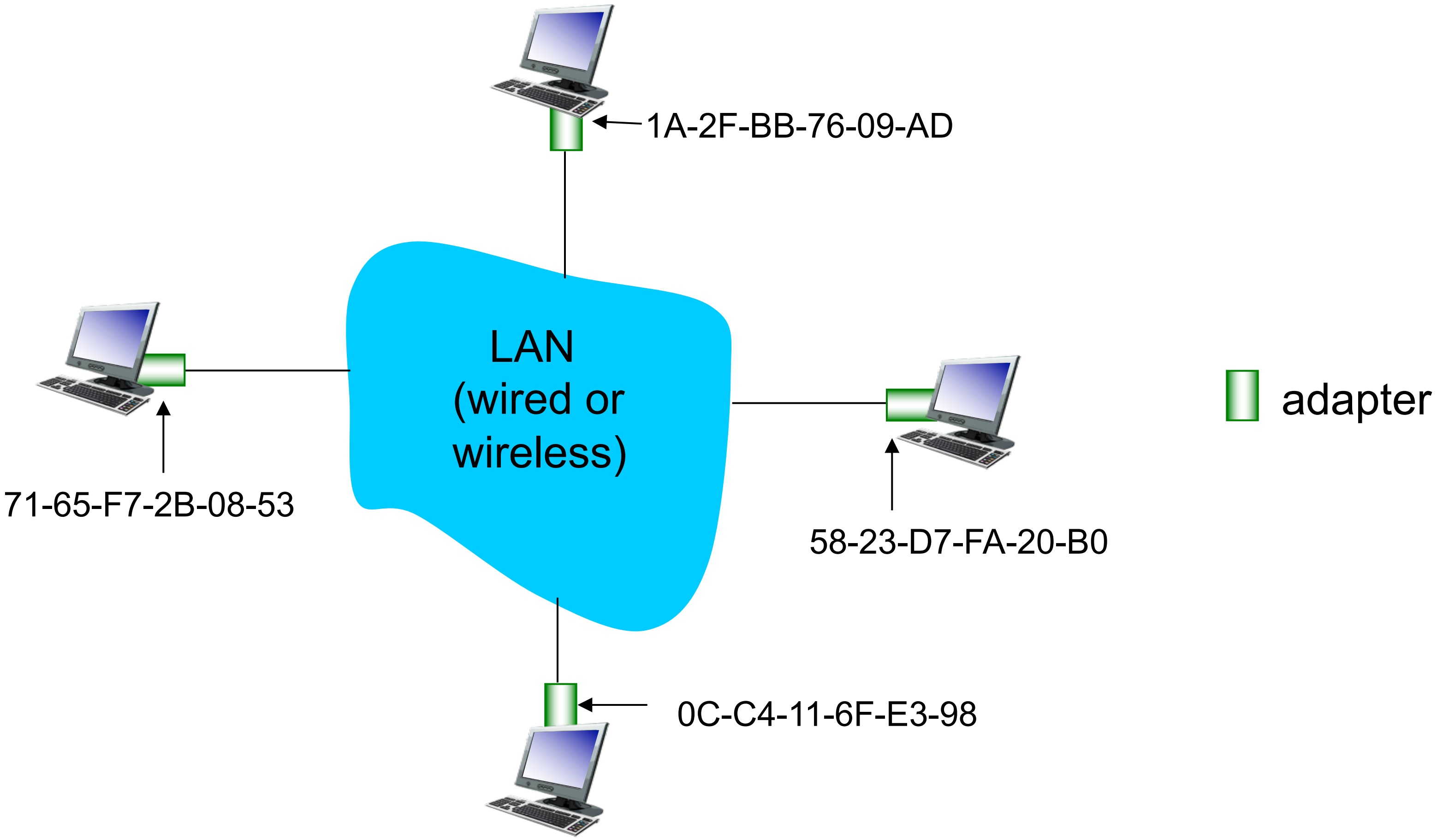


IP addressing: special addresses

Subnet	All 0s	**the (sub)network (network ID)**
All 1s		**limited broadcast (local net)**
Subnet	All 1s	**directed broadcast for net**
127	Anything (often 1)	**loopback**

LAN addresses

each adapter on LAN has unique *LAN* address

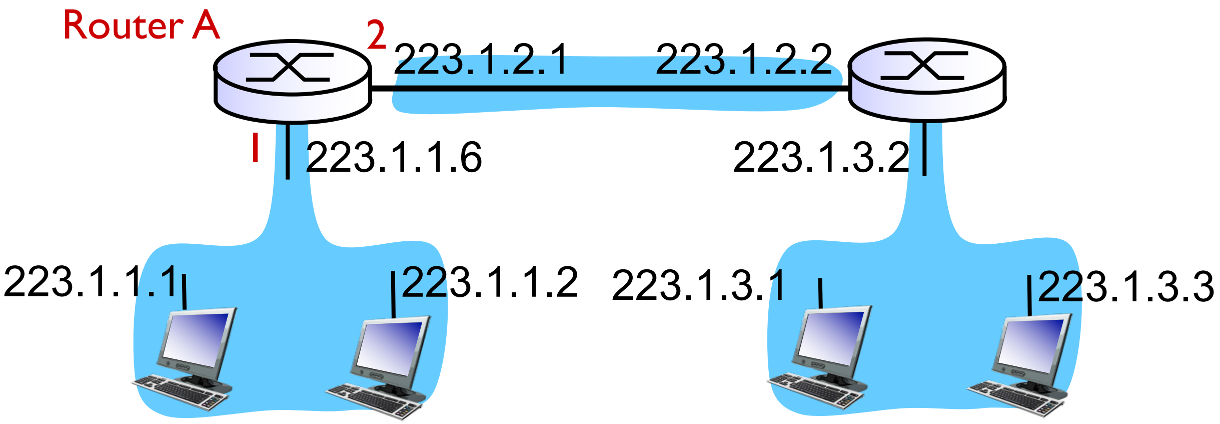


LAN addresses (more)

- MAC address allocation administered by IEEE
- manufacturer buys portion of MAC address space (to assure uniqueness)
- analogy:
  - MAC address: like Social Security Number
  - IP address: like postal address
- MAC flat address ➜ portability
  - can move LAN card from one LAN to another
- IP hierarchical address *not* portable
  - address *depends on IP subnet* to which node is attached

Subnets and IP routing

- IP addresses grouped according to their location (subnet)
- an entry on a routing table can refer to *an entire subnet* rather than to a single address!
- how can we do even better?
    - hierarchical addressing (see *later*…)

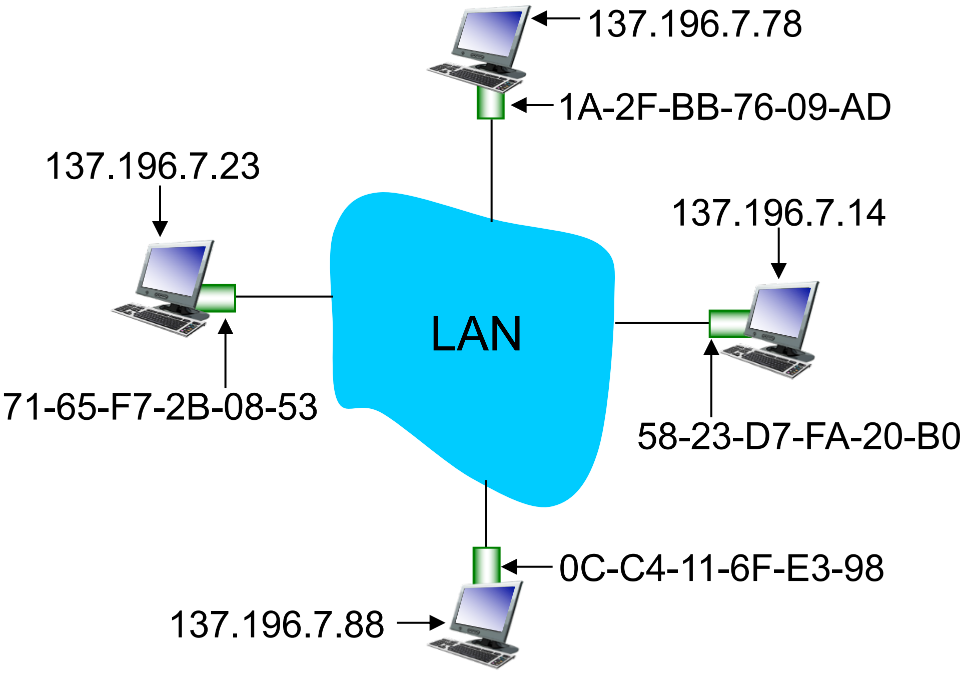


Router A routing table

destination	output link
223.1.1.0	1
223.1.2.0	2
223.1.3.0	2

ARP: address resolution protocol

Question: how to determine interface's MAC address, knowing its IP address?



ARP table: each IP node (host, router) on LAN has table

- IP/MAC address mappings for some LAN nodes:

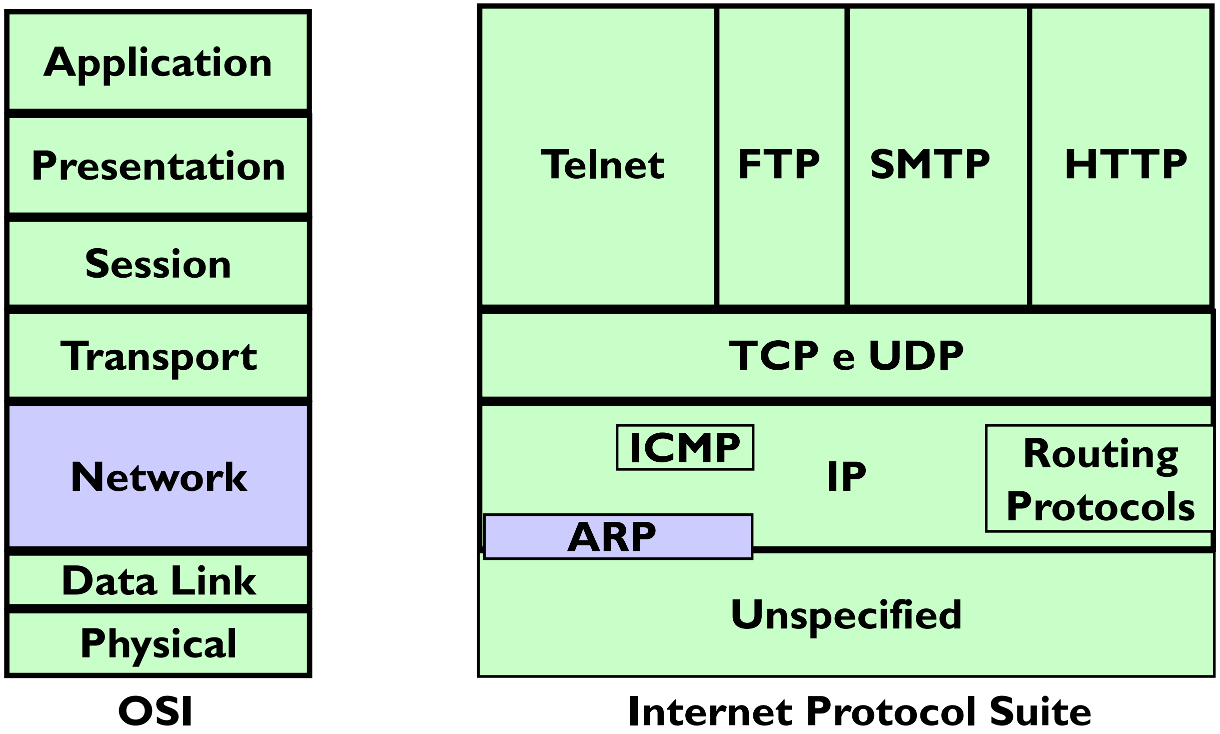
  < IP address; MAC address; TTL>

- TTL (Time To Live): time after which address mapping will be forgotten (typically 20 min)

ARP protocol: same LAN

- A wants to send datagram to B
  - B's MAC address not in A's ARP table.
- A broadcasts ARP query packet (*ARP request*), containing B's IP address
  - dest MAC address = FF-FF-FF-FF-FF-FF
  - all nodes on LAN receive ARP query
- B receives ARP packet, replies to A (*ARP reply*) with its (B's) MAC address
  - frame sent to A's MAC address (unicast)
- A caches (saves) IP-to-MAC address pair in its ARP table until information becomes old (times out)
  - soft state: information that times out (goes away) unless refreshed
- ARP is "plug-and-play":
  - nodes create their ARP tables *without intervention from net administrator*

ARP protocol (more)



- ARP packets are not IP datagrams!
  - They do not have IP source/destination, TTL, etc.
- ARP packets are payloads of link layer frames

ARP request and reply

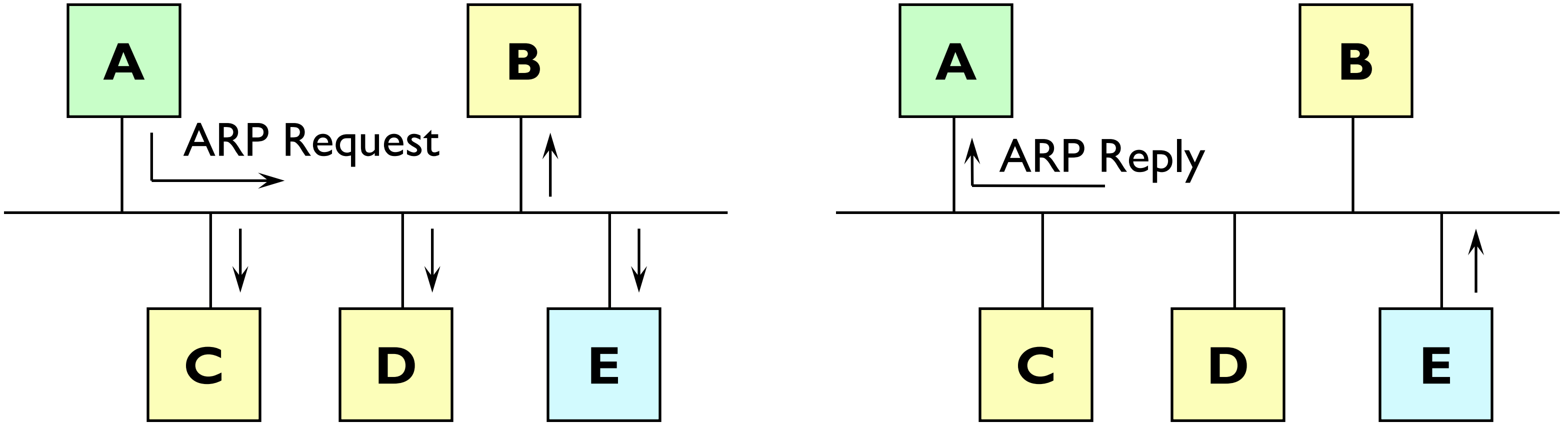
most significant fields of the link layer header *most significant fields of the ARP packet*

MAC broadcast	**MAC A**	**ARP Req**	**MAC A**	**IP A***	**??**	**IP E***

ARP Request

MAC A	**MAC E**	**ARP Reply**	**MAC E**	**IP E***	**MAC A**	**IP A***

ARP Reply

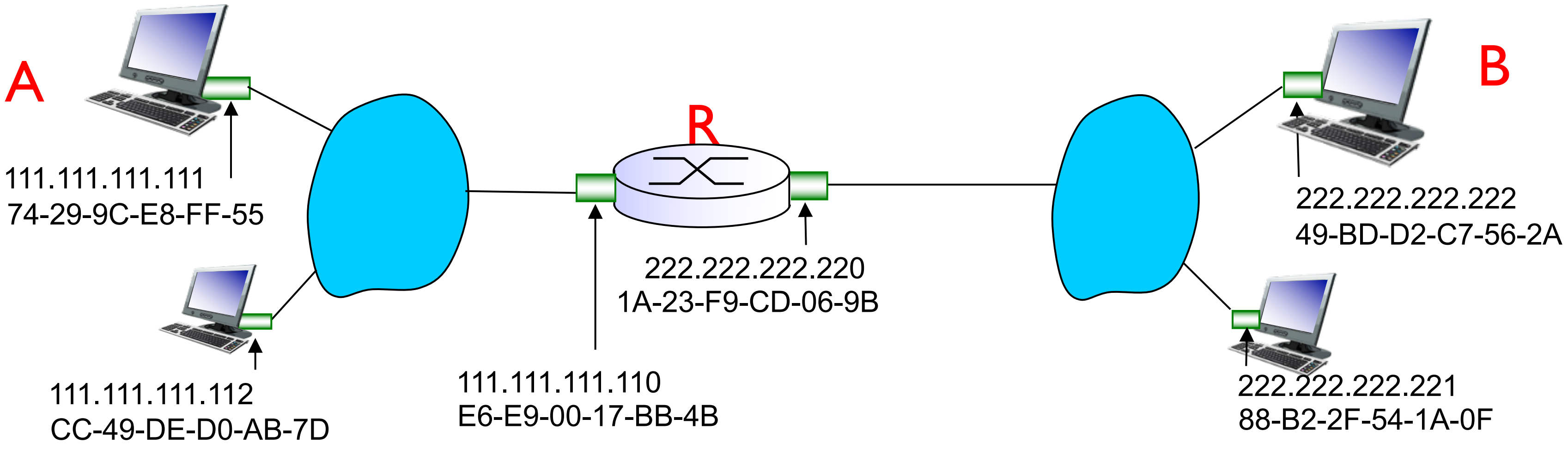


*These fields are not IP source and destination fields of an IP header!

Addressing: routing to another LAN

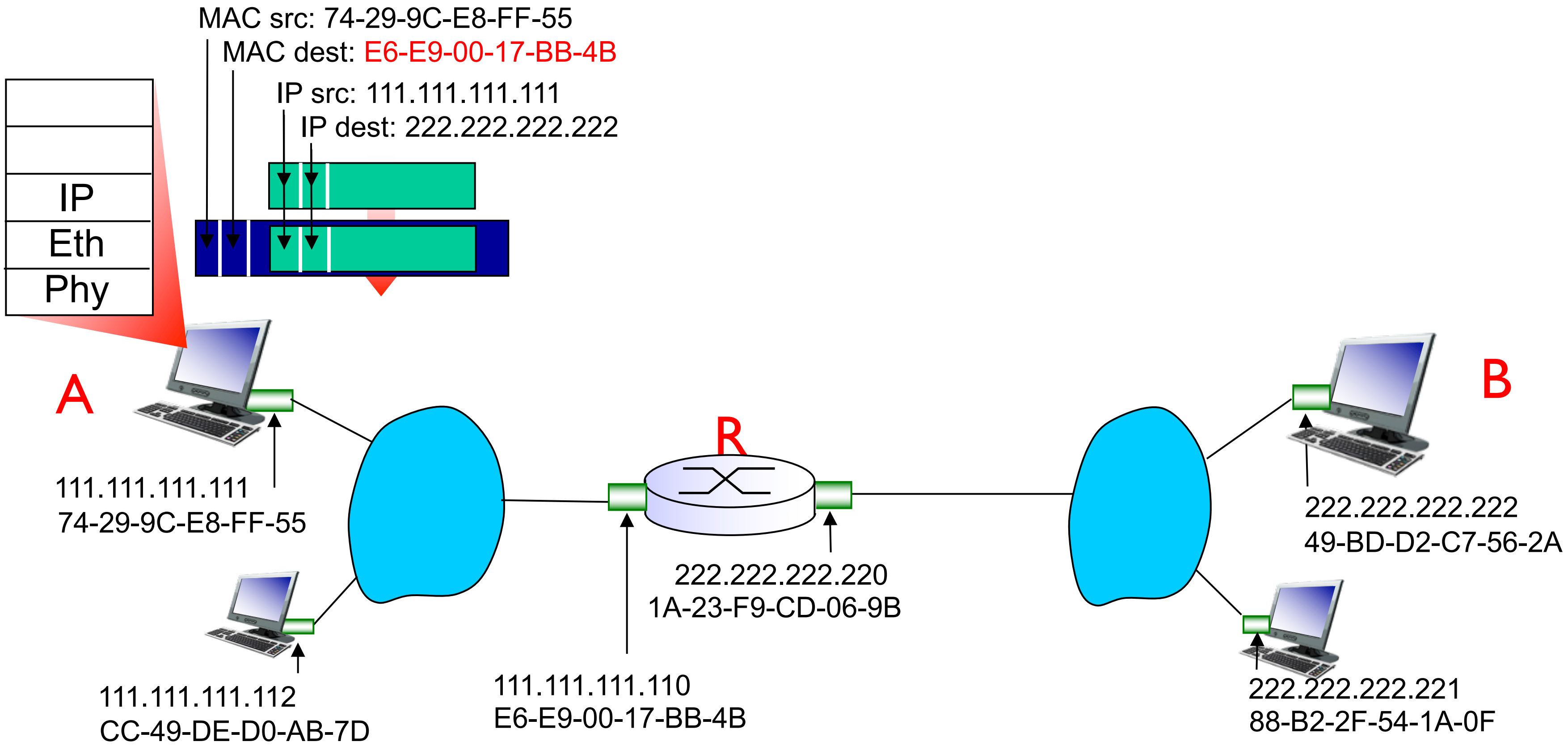
walkthrough: send datagram from A to B via R

- focus on addressing – at IP (datagram) and MAC layer (frame)
- assume A knows B's IP address
- assume A knows IP address of first hop router, R (how?)
- assume A knows R's MAC address (how?)



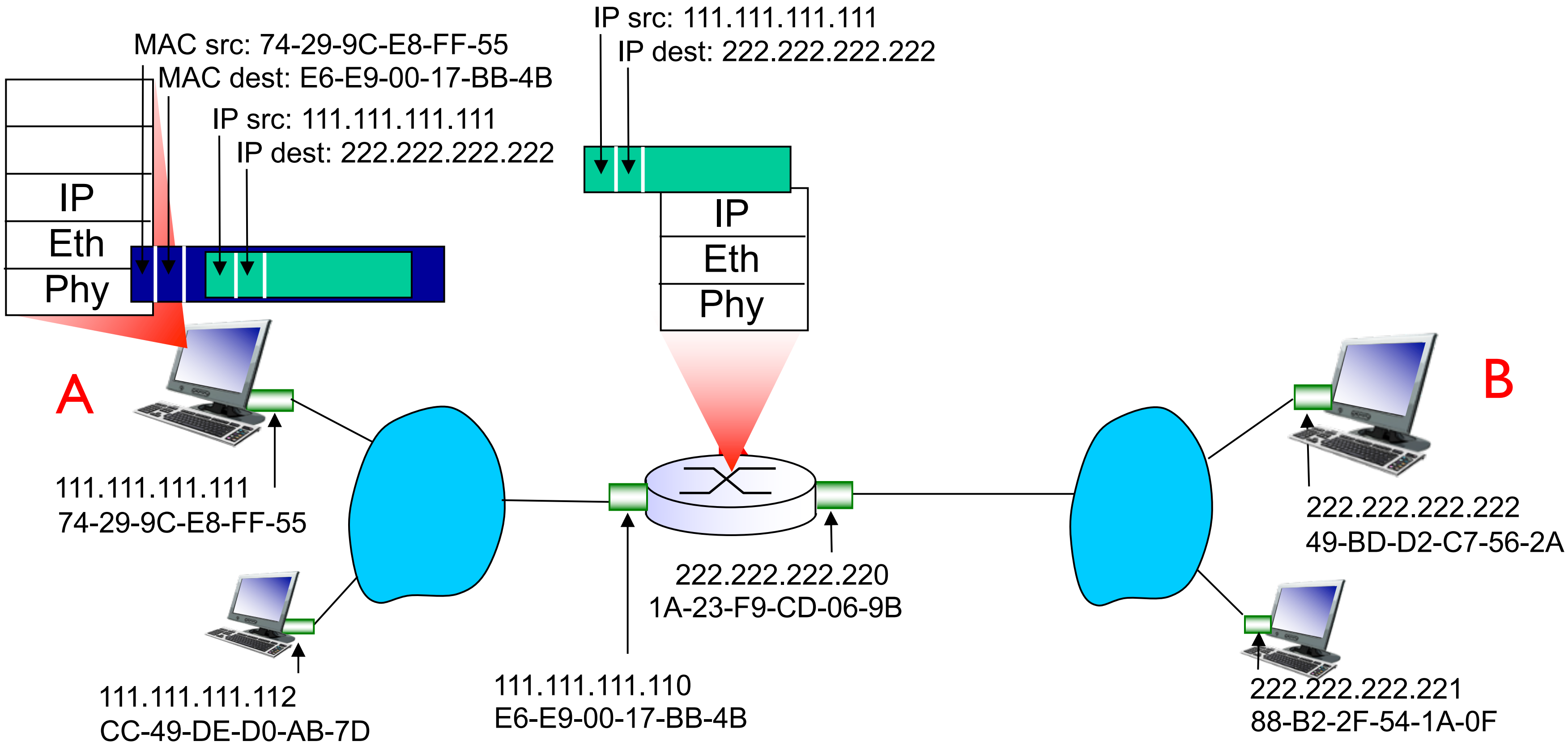
Addressing: routing to another LAN

- A creates IP datagram with IP source A, destination B
- A creates link-layer frame with R's MAC address as dest, frame contains A-to-B IP datagram



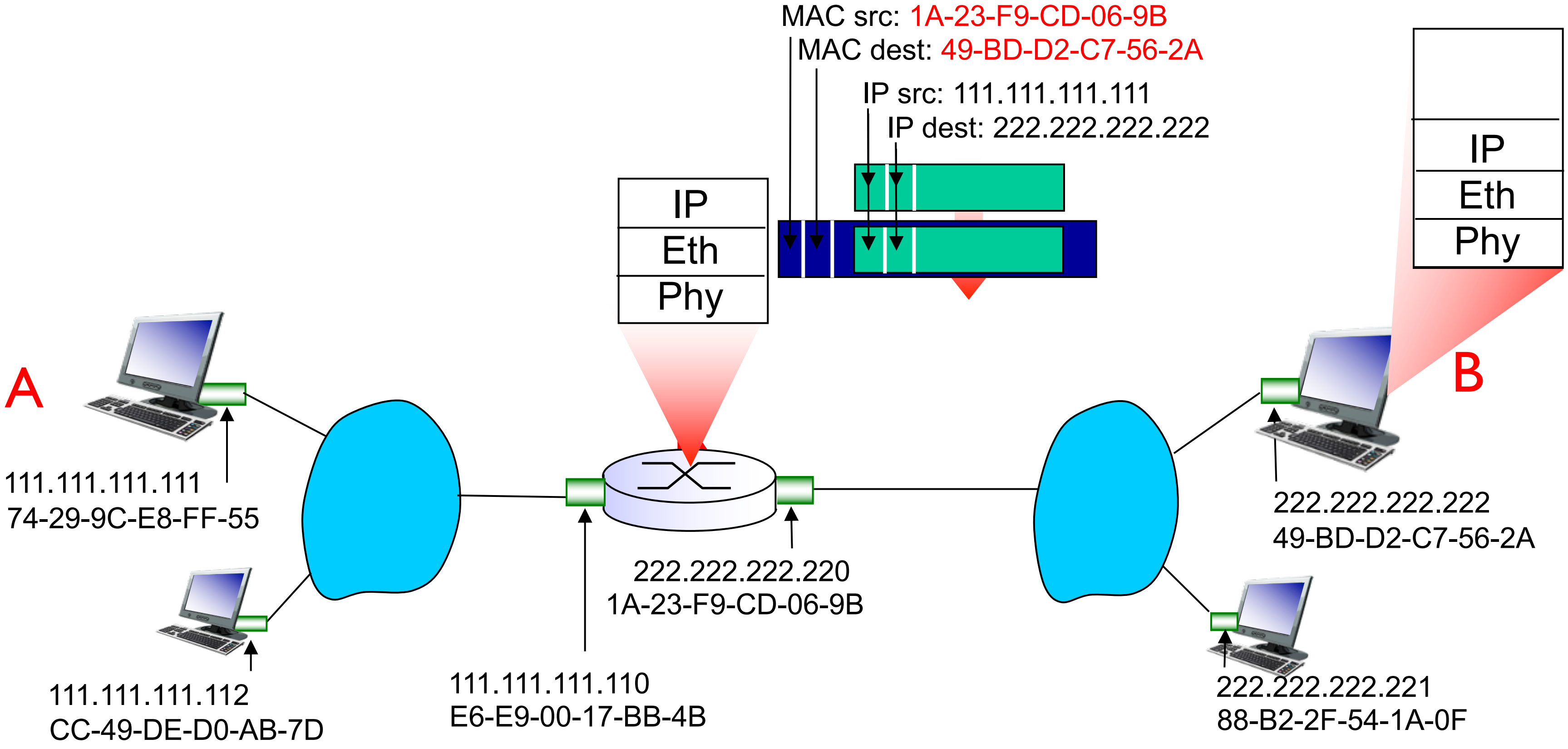
Addressing: routing to another LAN

- frame sent from A to R
- frame received at R, datagram removed, passed up to IP



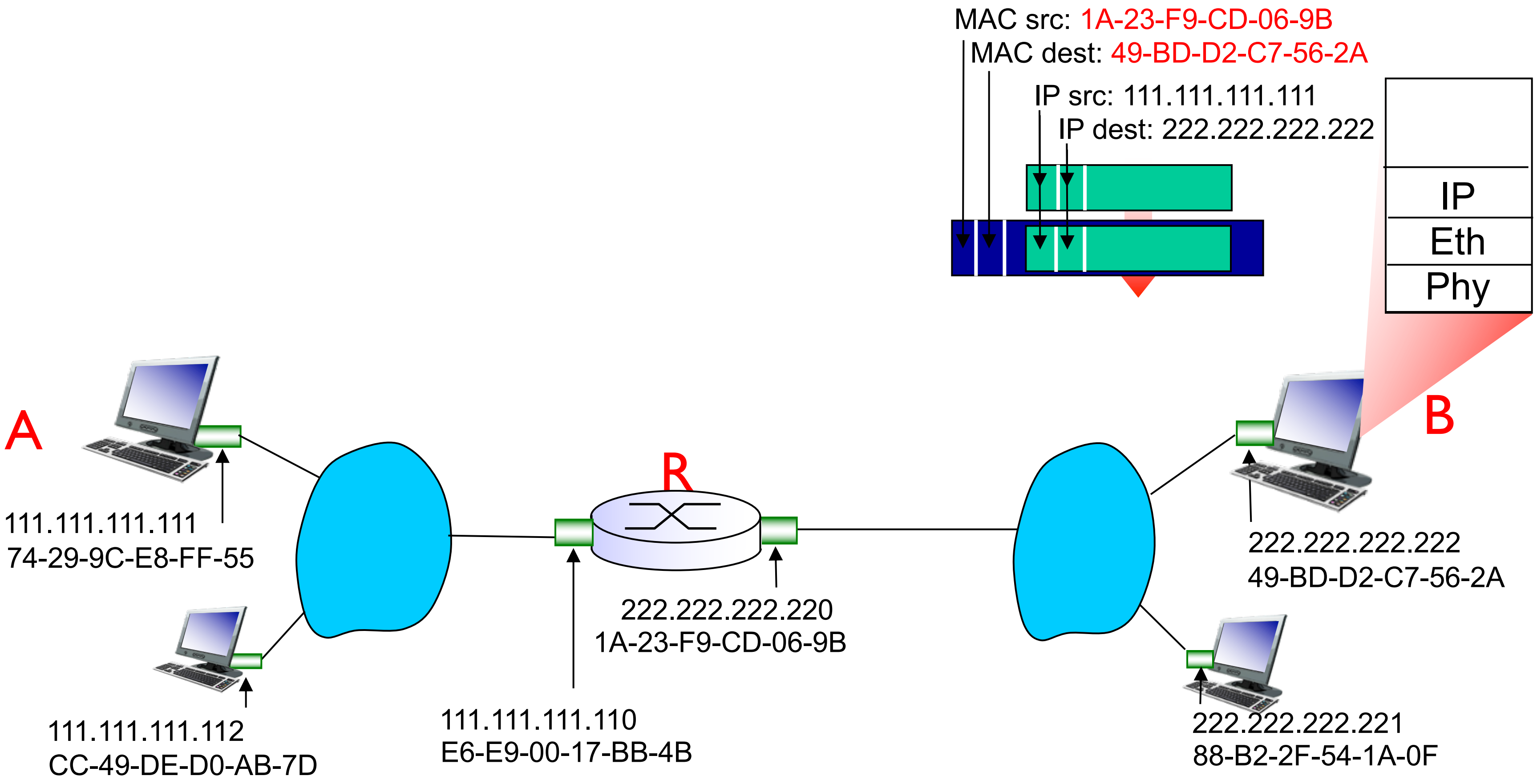
Addressing: routing to another LAN

- R forwards datagram with IP source A, destination B
- R creates link-layer frame with B's MAC address as dest, frame contains A-to-B IP datagram



Addressing: routing to another LAN

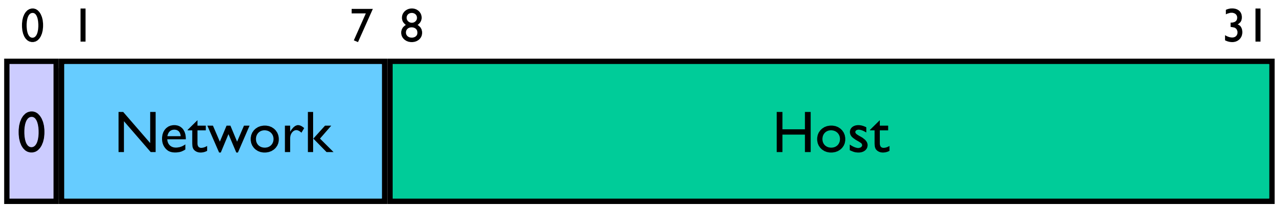
- R forwards datagram with IP source A, destination B
- R creates link-layer frame with B's MAC address as dest, frame contains A-to-B IP datagram



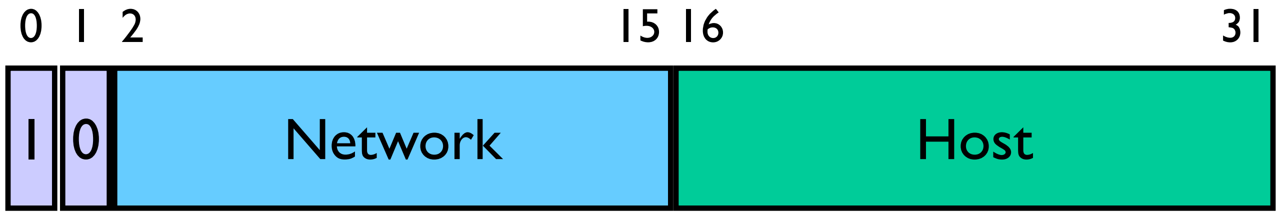
IP addressing: the entire story…

- *classful addressing:* static division of network part and host part of an IP address
  - no concept of subnet
  - only three possible sizes for IP networks
  - poor flexibility
- *subnetting:* starting from a classful addressing, define smaller IP networks called “subnets”
  - introduction of the concept of subnet and subnet mask
  - Variable Length Subnet Masking (VLSM)
- *classless addressing:* concept of IP class completely removed
  - we should refer again to IP networks (and netmasks) rather than to IP subnets (and subnet masks), but nobody cares about it!
  - Classless Inter Domain Routing (CIDR)

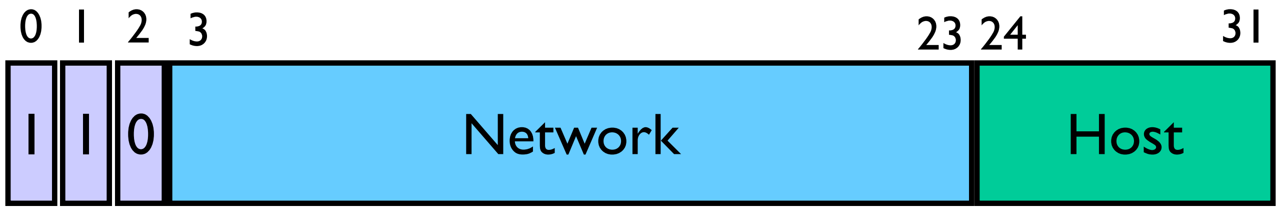
IP addressing: Classes



Class A – 128 networks – 1° byte: 0-127

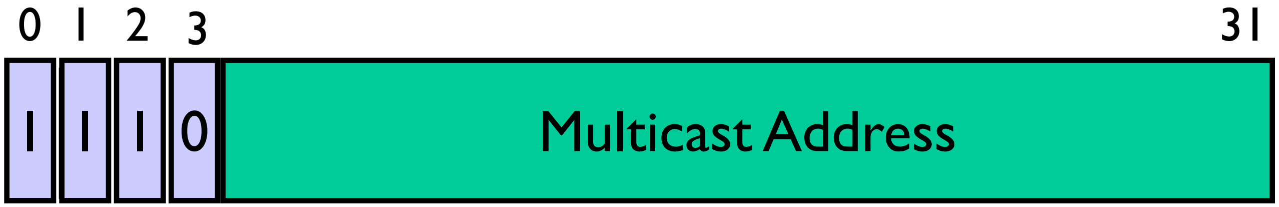


Class B – 16K networks – 1° byte: 128-191

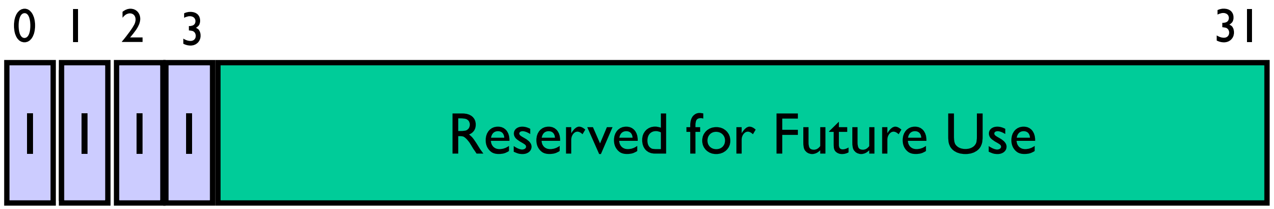


Class C – 2M networks - 1° byte: 192-223

IP addressing: Classes



Class D – 1° byte: 224-239

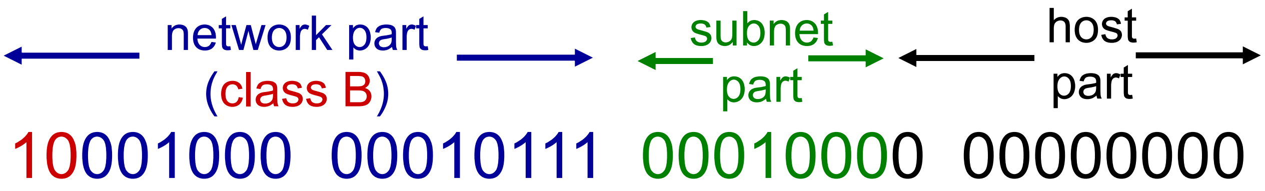


Class E – 1° byte: 240-255

IP addressing: subnetting

VLSM: Variable Length Subnet Masking

- starting from an IP network of a given class, define smaller subnets of arbitrary size
- address format: network ID + subnet mask
    - subnet mask: all '1s' in the subnet part, all '0s' in the host part



network part (class B)

subnet part

host part

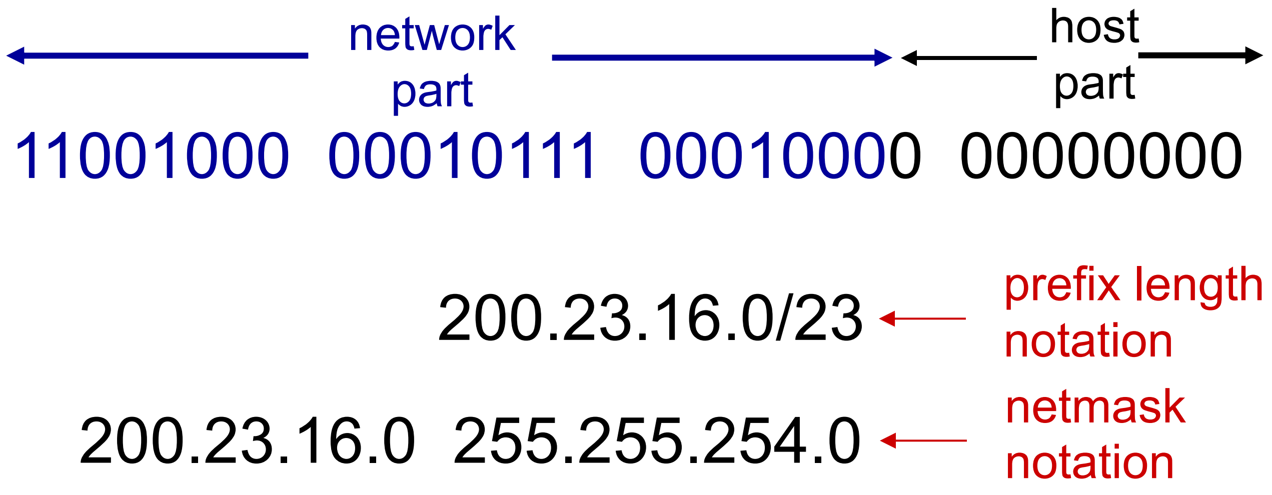
10001000 00010111 00010000 00000000

136.23.16.0 255.255.254.0 ← subnet mask notation

IP addressing: CIDR

CIDR: Classless InterDomain Routing

- network portion of address of arbitrary length
- address format: network ID + prefix length or netmask
  - prefix length: /x, where x is # bits in network portion of address
  - netmask: all '1s' in the network part, all '0s' in the host part



network part

host part

11001000 00010111 00010000 00000000

200.23.16.0/23 ← prefix length notation

200.23.16.0 255.255.254.0 ← netmask notation

IP addressing: CIDR

valid netmasks: possible values in the 4 bytes composing the address

0	0000 0000	(256)
128	1000 0000	(128)
192	1100 0000	(64)
224	1110 0000	(32)
240	1111 0000	(16)
248	1111 1000	(8)
252	1111 1100	(4)
254	1111 1110	(2)*
255	1111 1111	(1)

***not valid in the 4° byte**

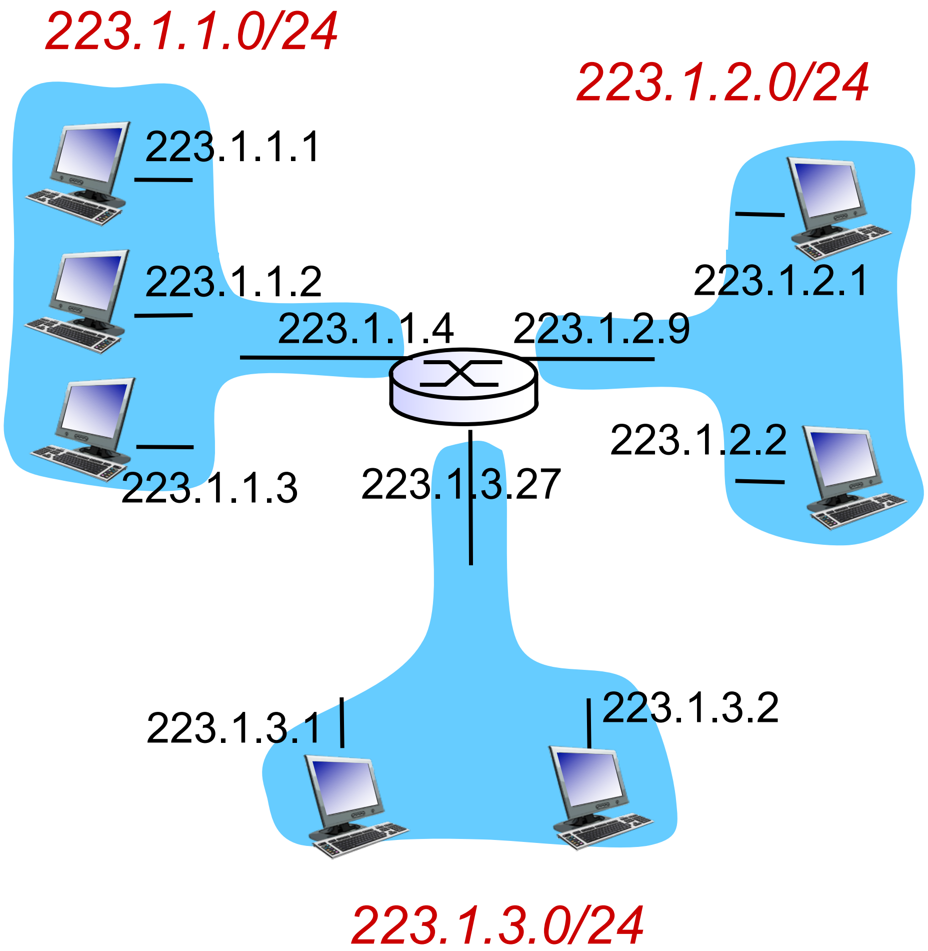
IP addressing: CIDR

some examples

- 130.192.0.0/16 – 130.192.0.0 255.255.0.0
- 130.192.0.0/24 – 130.192.0.0 255.255.255.0
- 130.192.0.0/25 – 130.192.0.0 255.255.255.128
- 130.192.2.0/23 – 130.192.2.0 255.255.254.0
- 130.192.1.4/30 – 130.192.1.4 255.255.255.252
- ~~130.192.1.0/31 – 130.192.1.0 255.255.255.254~~

Each IP network *must* contain at least the network ID and the broadcast address!

IP addressing: a real example



netmask: 255.255.255.0

IP addressing: device config

each device must be provided with

- *an IP address*
- *a netmask*
  - to *evaluate* its own network ID
  - to *infer* the network ID of the destination
- *a default gateway*, a.k.a. first-hop router
  - to perform indirect communications

Hierarchical addressing plans

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

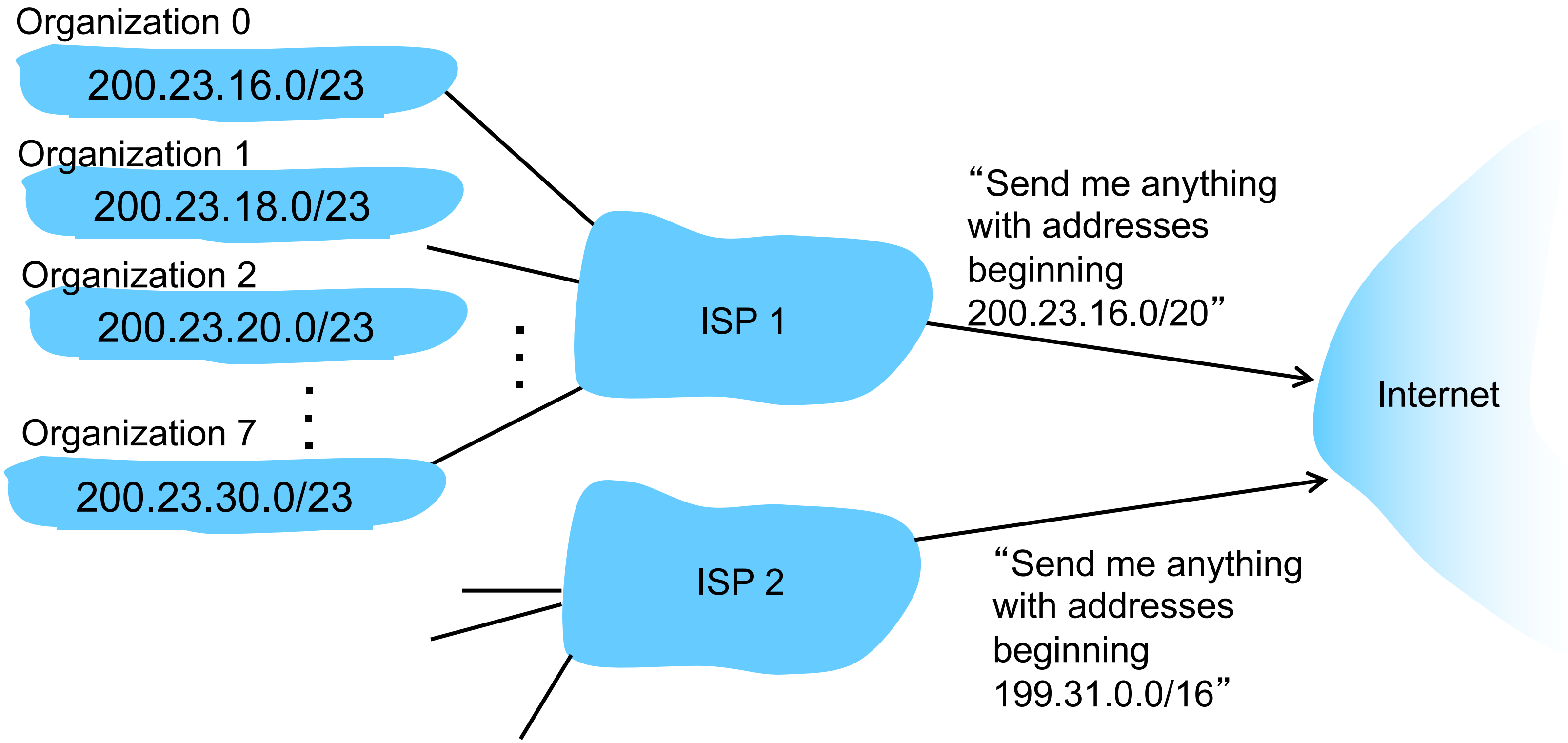
A: gets allocated portion of its provider ISP' s address space

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

This was not possible with classful addressing!

Hierarchical addressing: route aggregation

hierarchical addressing allows efficient advertisement of routing information:



"Send me anything with addresses beginning 200.23.16.0/20"

"Send me anything with addresses beginning 199.31.0.0/16"

How do IP routing really work?

Q: How do a host learn its own IP network(s)?

- *bit-wise AND* between its IP address(es) and its netmask(s)

interface IP address: 192.168.10.69

11000000 10101000 00001010 01000101

11111111 11111111 11111111 11000000

11000000 10101000 00001010 01000000

bit-wise AND result:192.168.10.64

netmask: 255.255.255.192

How do IP routing really work?

Q: How do a host infer the IP network of the destination it wants to contact?

- *bit-wise AND* between the destination IP address and *its own* netmask

destination IP address: 192.168.10.101

11000000	10101000	00001010	01100101
11111111	11111111	11111111	11000000
11000000	10101000	00001010	01000000

bit-wise AND result:192.168.10.64

netmask: 255.255.255.192

How do IP routing really work?

Q: How do a host infer the IP network of the destination it wants to contact?

- *bit-wise AND* between the destination IP address and *its own* netmask

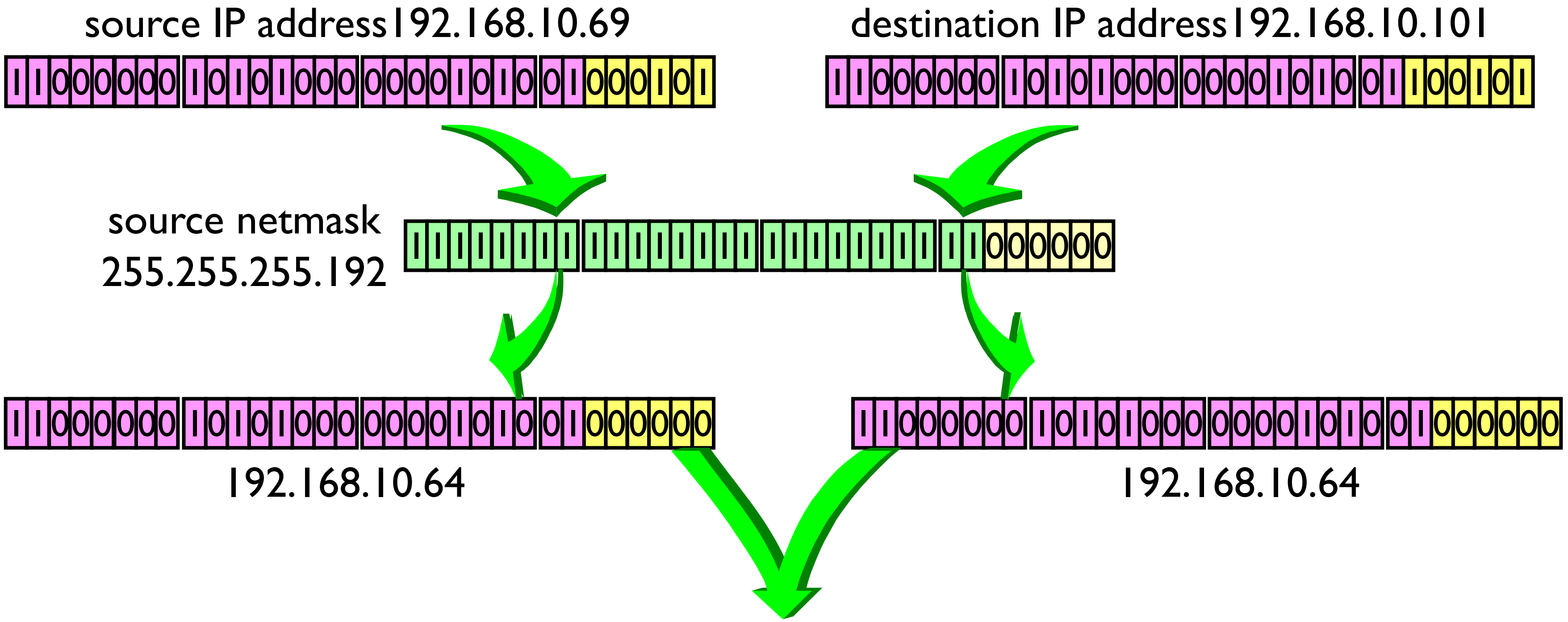
destination IP address: 192.168.10.132

```
11000000 10101000 00001010 10000100
11111111 11111111 11111111 11000000
-----------------------------------
11000000 10101000 00001010 10000000
```

bit-wise AND result:192.168.10.128

netmask: 255.255.255.192

How do IP routing really work?



source IP address192.168.10.69

11000000 10101000 00001010 01000101

destination IP address192.168.10.101

11000000 10101000 00001010 01100101

source netmask
255.255.255.192

11111111 11111111 11111111 11000000

11000000 10101000 00001010 01000000

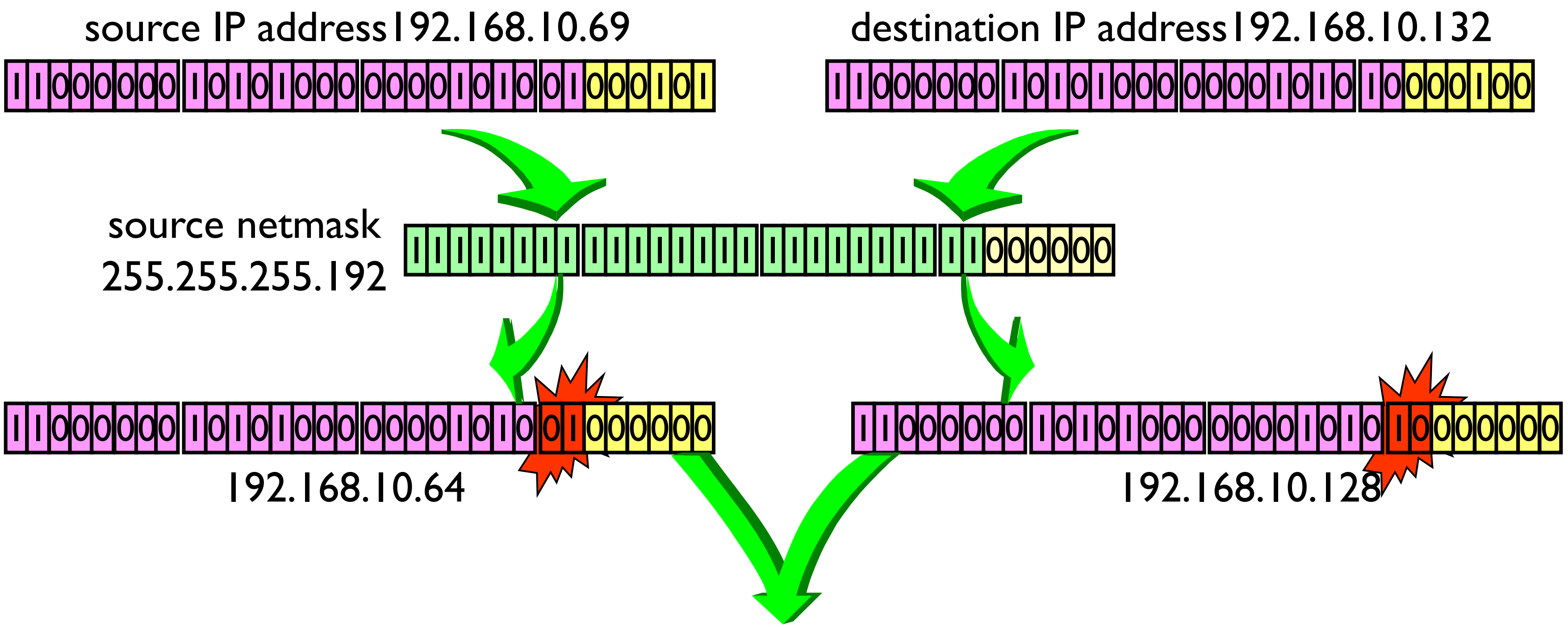
192.168.10.64

11000000 10101000 00001010 01000000

192.168.10.64

same IP network: *direct communication*

How do IP routing really work?



source IP address192.168.10.69

11000000 10101000 00001010 01000101

destination IP address192.168.10.132

11000000 10101000 00001010 10000100

source netmask
255.255.255.192

11111111 11111111 11111111 11000000

11000000 10101000 00001010 01000000

192.168.10.64

11000000 10101000 00001010 10000000

192.168.10.128

different IP network: *indirect communication*

we need a router!

How do IP routing really work?

Q: How do a router select the correct output port?

- *bit-wise AND* between the destination IP address of a packet and the netmask of each entry in the routing table, looking for a match

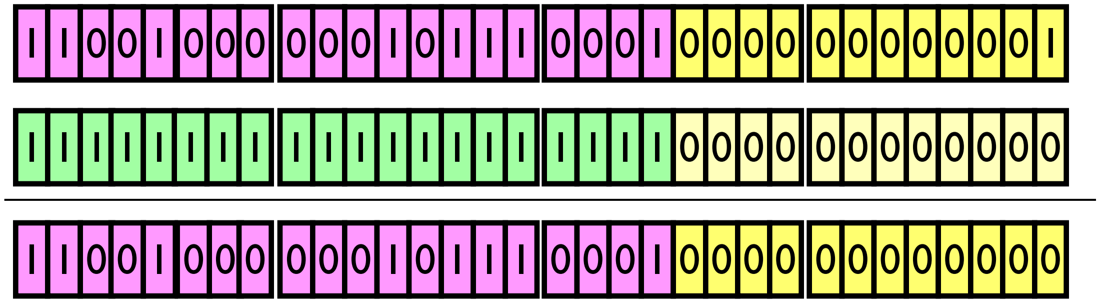
routing table	
destination	output link
200.23.16.0/20	1
199.31.0.0/16	2

How do IP routing really work?

Q: How do a router select the correct output port?

- *bit-wise AND* between the destination IP address of a packet and the netmask of each entry in the routing table, looking for a match

destination IP address: 200.23.16.1



11001000 00010111 00010000 00000001

11111111 11111111 11110000 00000000

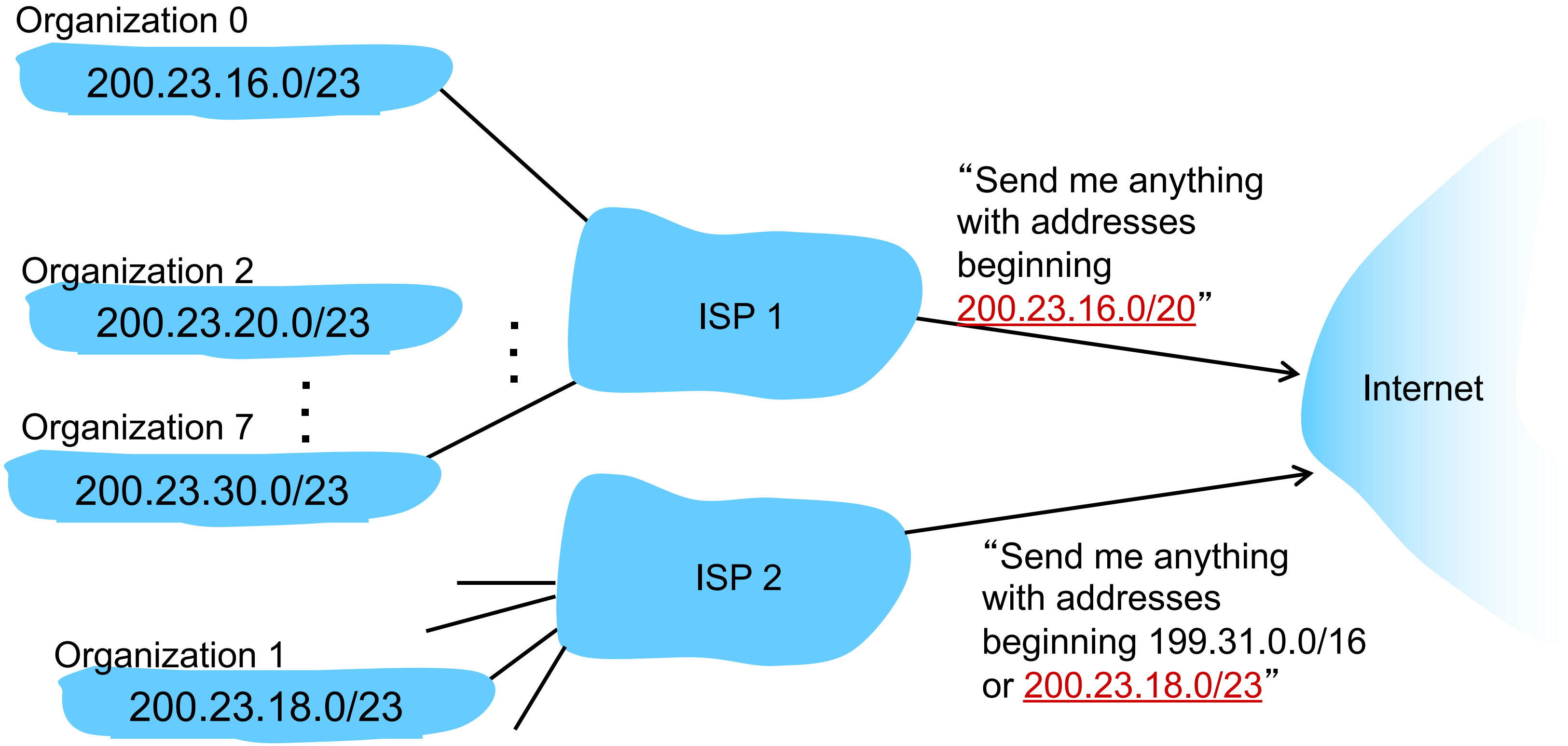
11001000 00010111 00010000 00000000

bit-wise AND result: 200.23.16.0 → match!

netmask: 255.255.240.0

Hierarchical addressing: more specific routes

ISP 2 has a more specific route to Organization 1


Organization 0
200.23.16.0/23
Organization 2
200.23.20.0/23
Organization 7
200.23.30.0/23
Organization 1
200.23.18.0/23
ISP 1
ISP 2
"Send me anything with addresses beginning 200.23.16.0/20"
Internet
"Send me anything with addresses beginning 199.31.0.0/16 or 200.23.18.0/23"

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 0001**** *********	0
11001000 00010111 0001001* *********	1
11000111 00011111 ******** *********	1
otherwise	2

examples:

DA: 11001000 00010111 00010110 10100001 *which interface?*

DA: 11001000 00010111 00010010 10101010 *which interface?*

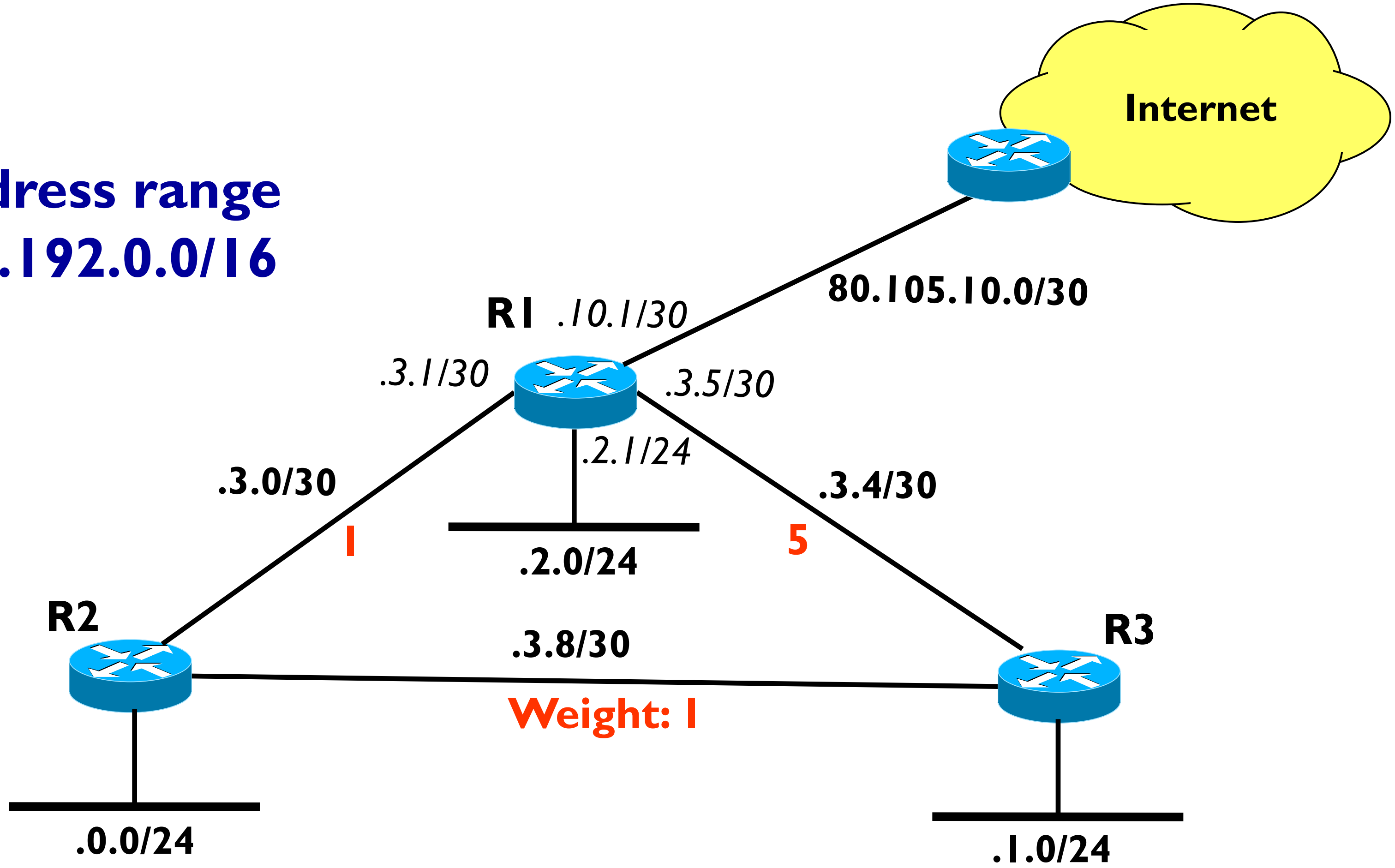
IP routing: a real routing table

there are three types of *routes*

- *direct routes*
  - networks directly connected to the router (i.e., address ranges including the router interfaces)
- *static routes*
  - manually configured routes to remote networks
- *dynamic routes*
  - automatically configured routes to remote networks
    - routing protocols
    - ICMP redirect

IP routing: example

Address range 130.192.0.0/16



IP routing: example

R1 routing table

remote
interface!!

Type	*Destination*	*Next-hop*	*Cost*
S	130.192.0.0/24	130.192.3.2	2
S	130.192.1.0/24	130.192.3.2	2
S	130.192.3.8/30	130.192.3.2	2
S	0.0.0.0/0	80.105.10.2	2
D	130.192.2.0/24	130.192.2.1	1
D	130.192.3.0/30	130.192.3.1	1
D	130.192.3.4/30	130.192.3.5	1
D	80.105.10.0/30	80.105.0.1	1

local
interface!!

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

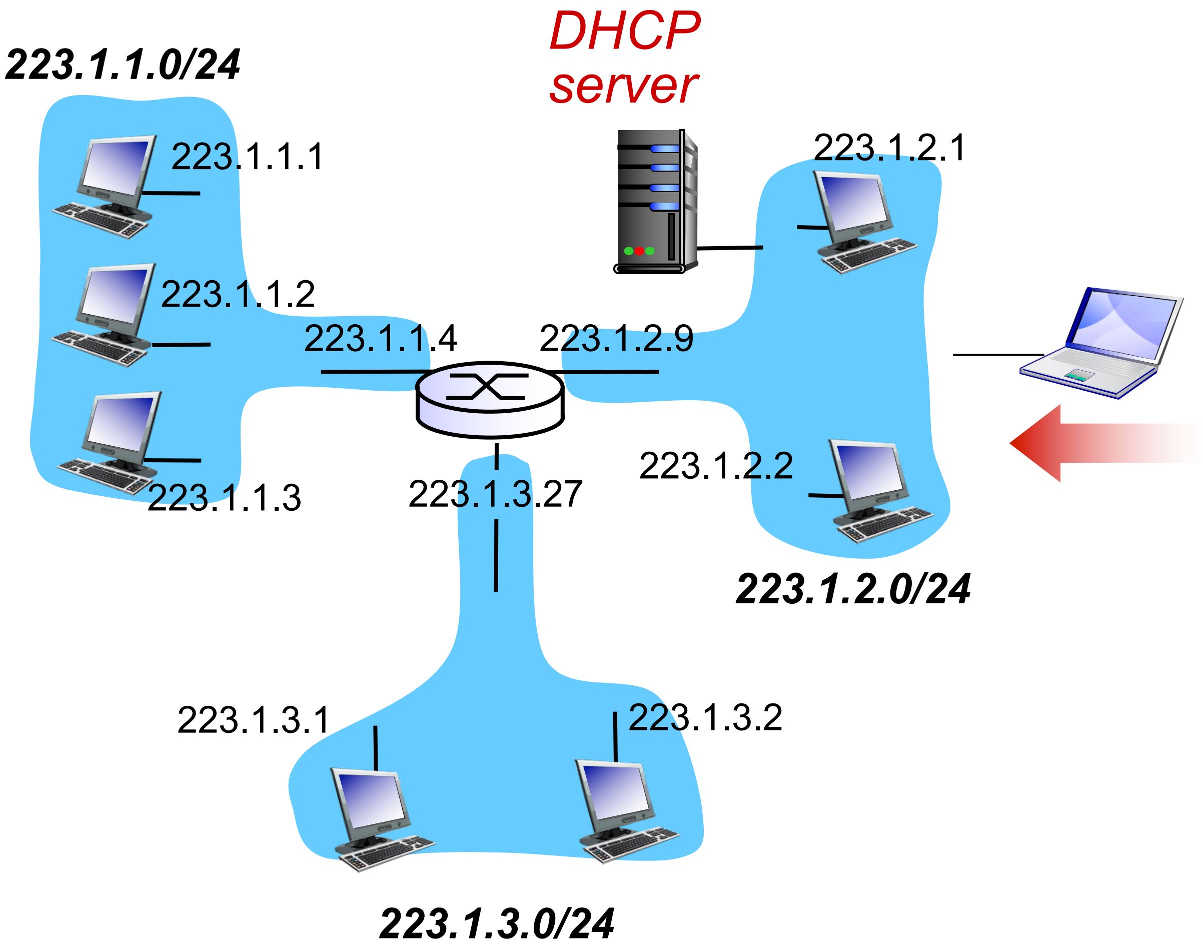
goal: allow host to *dynamically* obtain its IP address from network server when it joins network

- 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:

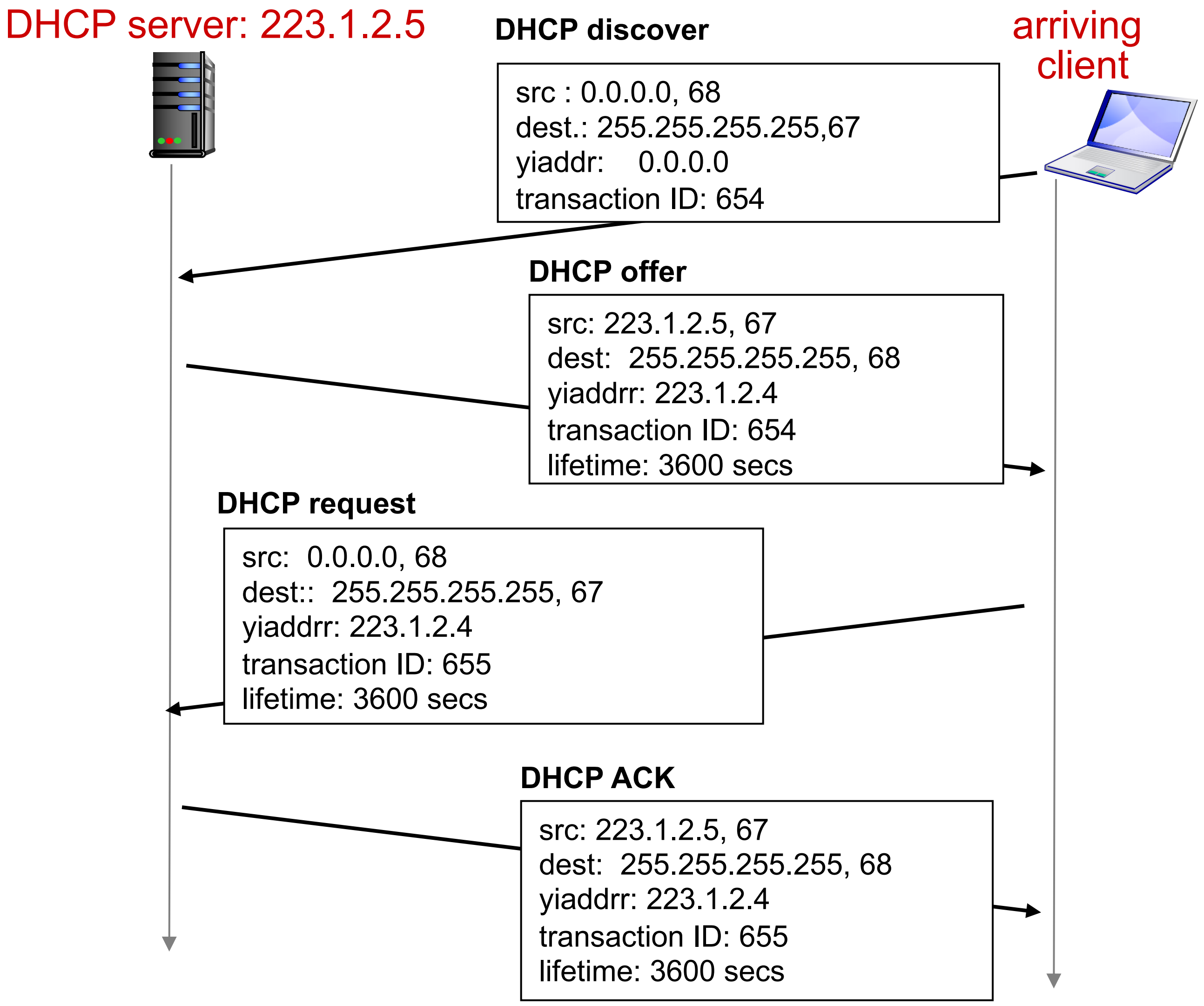
- host broadcasts "DHCP discover" msg [optional]
- DHCP server responds with "DHCP offer" msg [optional]
- host requests IP address: "DHCP request" msg
- DHCP server sends address: "DHCP ack" msg

DHCP client-server scenario



arriving DHCP client needs address in this network

DHCP client-server scenario



DHCP server: 223.1.2.5

arriving client

DHCP discover

src : 0.0.0.0, 68
dest.: 255.255.255.255,67
yiaddr: 0.0.0.0
transaction ID: 654

DHCP offer

src: 223.1.2.5, 67
dest: 255.255.255.255, 68
yiaddrr: 223.1.2.4
transaction ID: 654
lifetime: 3600 secs

DHCP request

src: 0.0.0.0, 68
dest:: 255.255.255.255, 67
yiaddrr: 223.1.2.4
transaction ID: 655
lifetime: 3600 secs

DHCP ACK

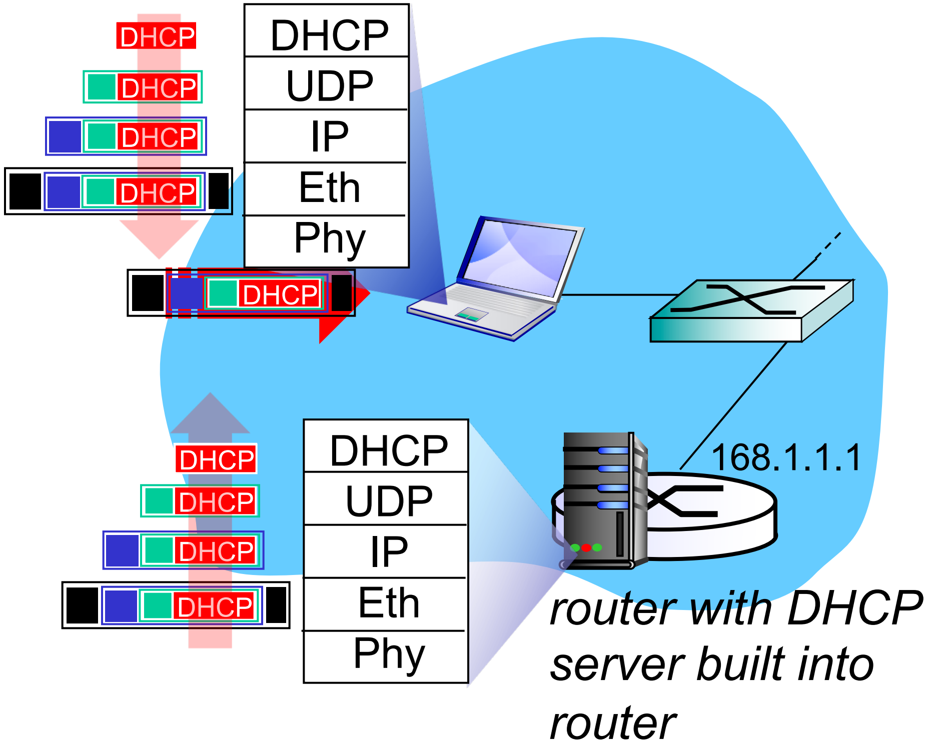
src: 223.1.2.5, 67
dest: 255.255.255.255, 68
yiaddrr: 223.1.2.4
transaction ID: 655
lifetime: 3600 secs

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)

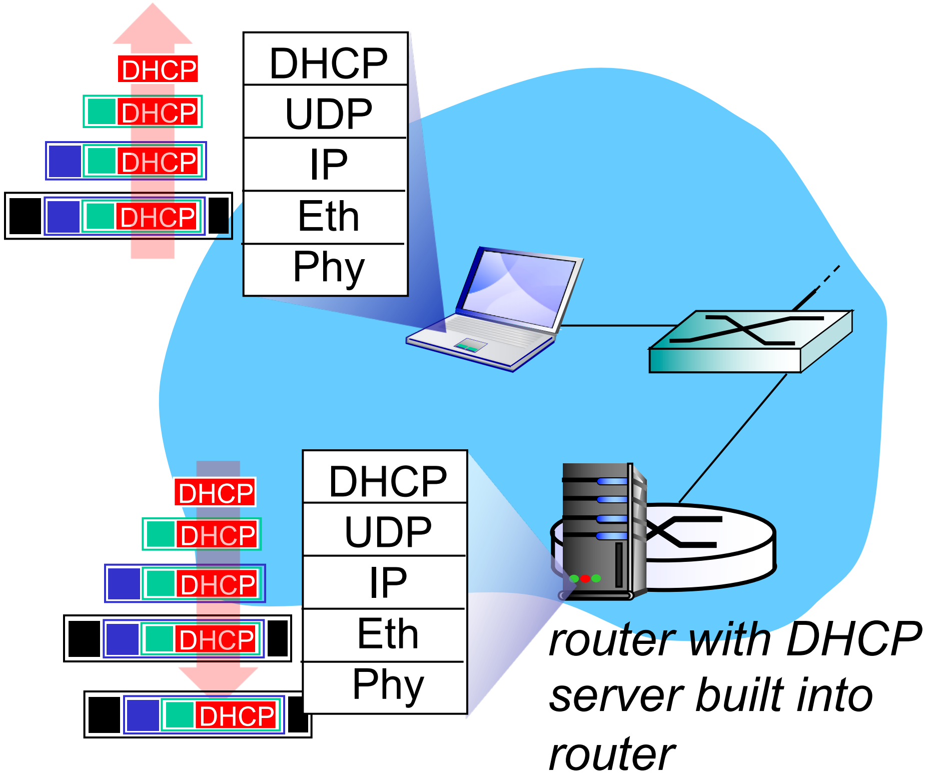
DHCP: example



router with DHCP server built into router

- 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: example



router with DHCP server built into router

- 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

DHCP: Wireshark output (home LAN)

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."
```

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/

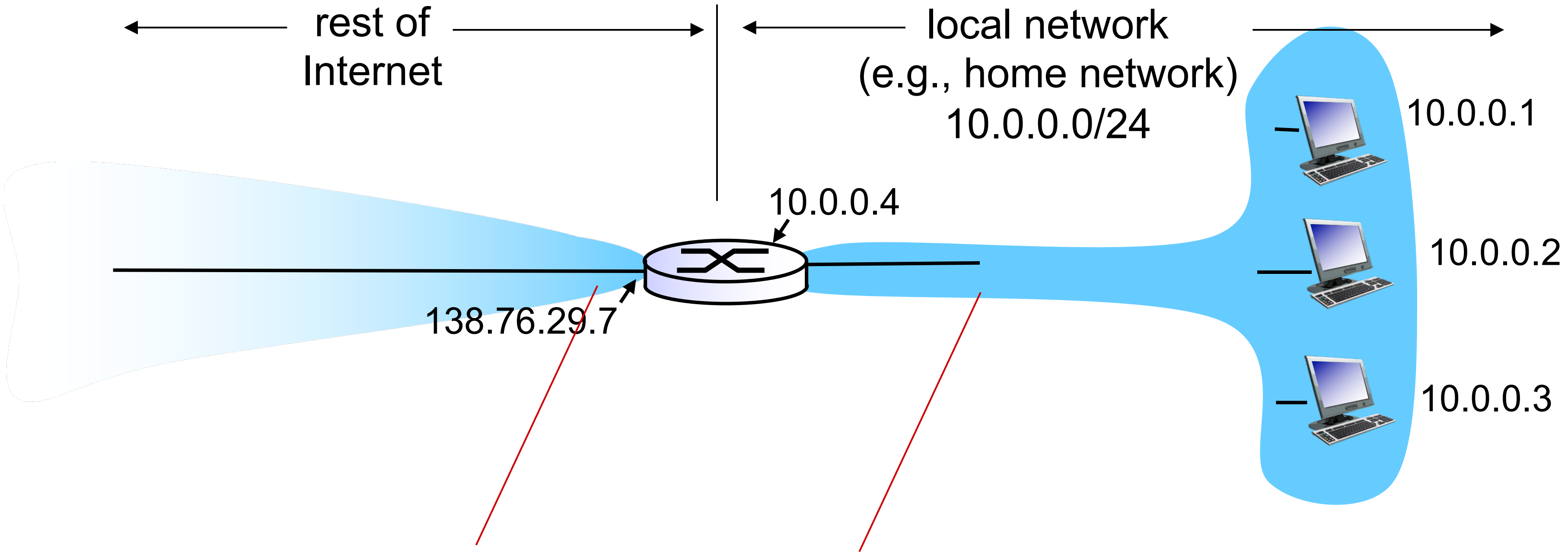
- allocates addresses
- manages DNS
- assigns domain names, resolves disputes

NAT: network address translation

- some address ranges are reserved for private networks
  - they are not announced over the Internet, so they are not reachable from remote areas of the network
  - reserved as classful networks, but still valid in CIDR

10.0.0.0 - 10.255.255.255	**1 class A network**
172.16.0.0 - 172.31.255.255	**16 class B networks**
192.168.0.0 - 192.168.255.255	**256 class C networks**

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 10.0.0.0/24 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:

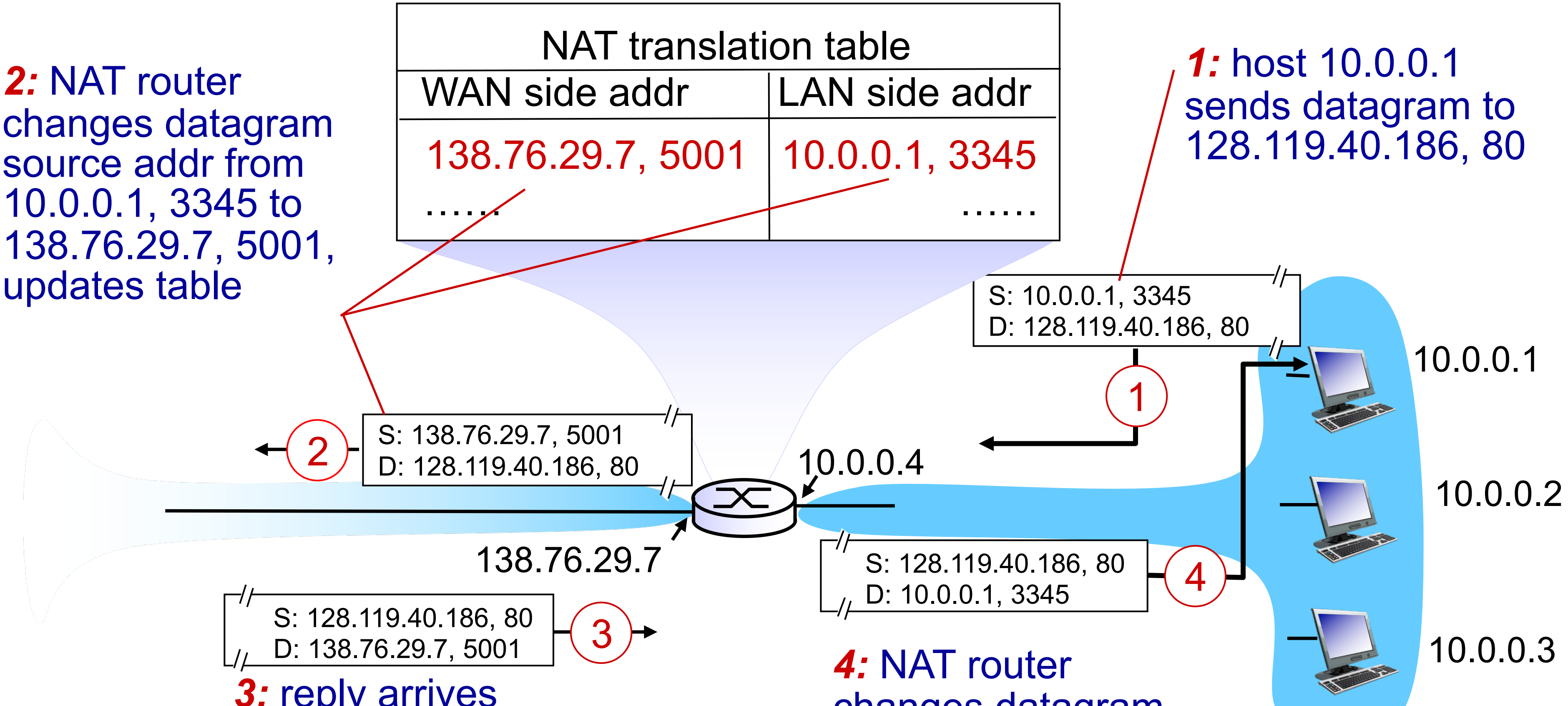
- *outgoing datagrams: replace* (source IP address, port #) of every outgoing datagram to (NAT IP address, new port #)

  . . . 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 translation table	
WAN side addr	LAN side addr
138.76.29.7, 5001	10.0.0.1, 3345
……	……

2: NAT router changes datagram source addr from 10.0.0.1, 3345 to 138.76.29.7, 5001, updates table

1: host 10.0.0.1 sends datagram to 128.119.40.186, 80

3: reply arrives dest. address: 138.76.29.7, 5001

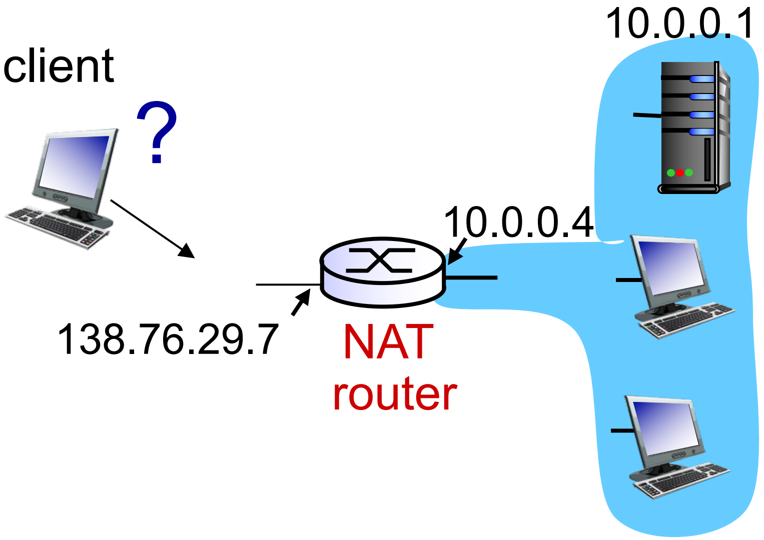
4: NAT router changes datagram dest addr from 138.76.29.7, 5001 to 10.0.0.1, 3345

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

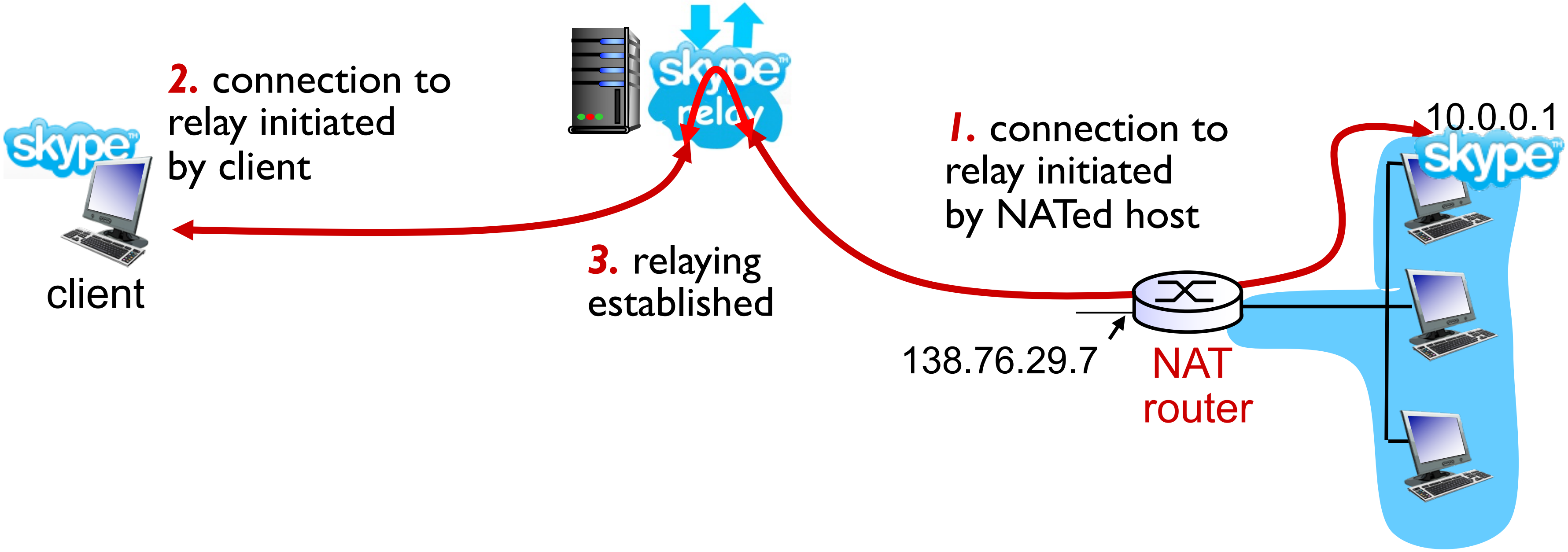
NAT traversal problem

- 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
- *solution1:* 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



NAT traversal problem

- *solution 2:* relaying (used in Skype)
  - NATed client establishes connection to relay
  - external client connects to relay
  - relay bridges packets between to connections



2. connection to relay initiated by client
client
3. relaying established
1. connection to relay initiated by NATed host
10.0.0.1
138.76.29.7
NAT router

Chapter 4: outline

4.1 introduction

4.2 virtual circuit and datagram networks

4.3 IP: Internet Protocol

- datagram format
- IPv4 addressing
- ICMP
- IPv6

4.4 routing algorithms

- link state
- distance vector
- hierarchical routing

4.5 routing in the Internet

- RIP
- OSPF
- BGP

ICMP: internet control message protocol

- used by hosts & routers to communicate network-level information
  - error reporting: unreachable host, network, port, protocol
  - echo request/reply (used by ping)
- network-layer "above" IP:
  - ICMP msgs carried in IP datagrams
- ICMP message: type, code plus first 8 bytes of IP datagram causing error

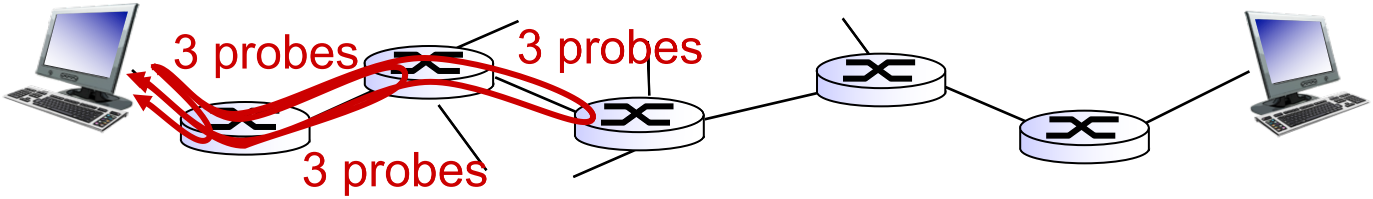
Type	Code	description
0	0	echo reply (ping)
3	0	dest. network unreachable
3	1	dest host unreachable
3	2	dest protocol unreachable
3	3	dest port unreachable
3	6	dest network unknown
3	7	dest host unknown
4	0	source quench (congestion control - not used)
5	0	redirect
8	0	echo request (ping)
9	0	route advertisement
10	0	router discovery
11	0	TTL expired (time exceeded)
12	0	bad IP header

Traceroute and ICMP

- source sends series of UDP segments to dest
  - first set has TTL =1
  - second set has TTL=2, etc.
  - unlikely port number
- when *n*th set of datagrams arrives to nth router:
  - router discards datagrams
  - and sends source ICMP messages (type 11, code 0)
  - ICMP messages includes name of router & IP address
- when ICMP messages arrives, source records RTTs

stopping criteria:

- UDP segment eventually arrives at destination host
- destination returns ICMP "port unreachable" message (type 3, code 3)
- source stops



IPv6: motivation

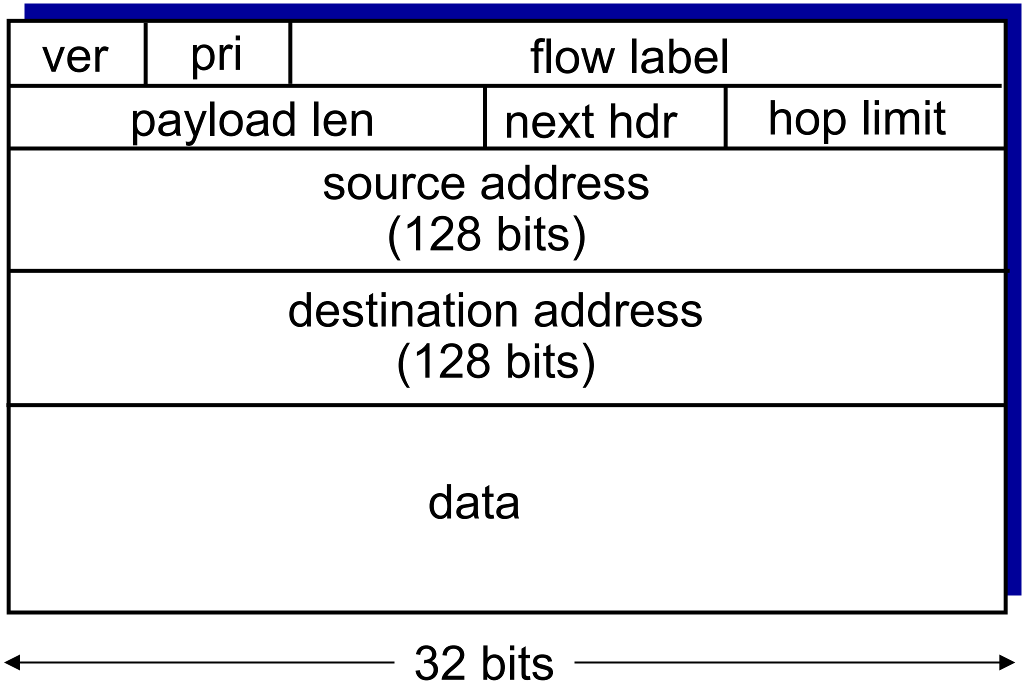
- *initial motivation:* 32-bit address space soon to be completely allocated.
- additional motivation:
  - header format helps speed processing/forwarding
  - header changes to facilitate QoS

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



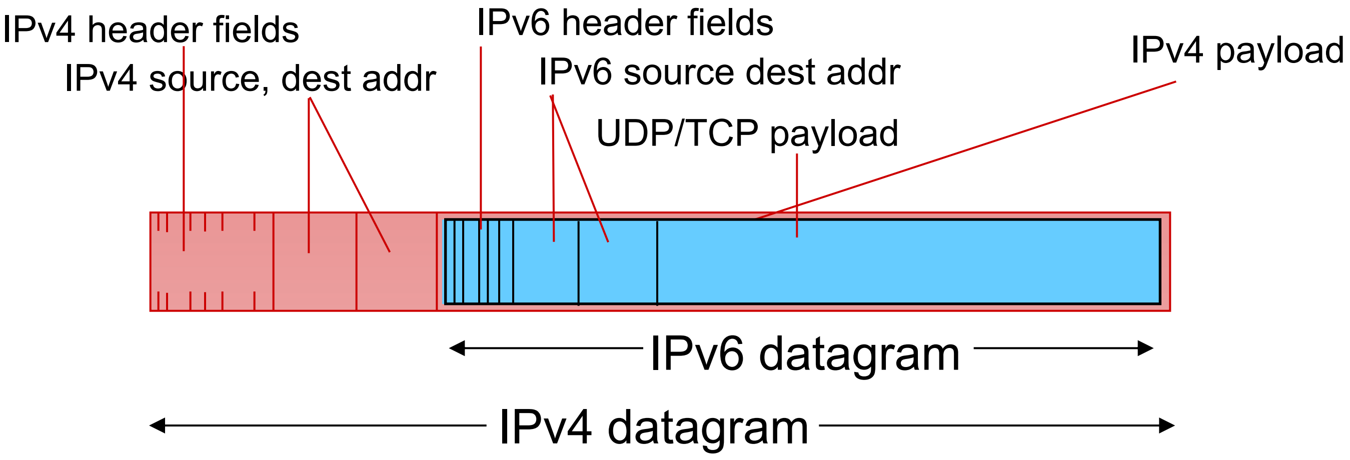
ver
pri
flow label
payload len
next hdr
hop limit
source address
(128 bits)
destination address
(128 bits)
data
32 bits

Other changes from IPv4

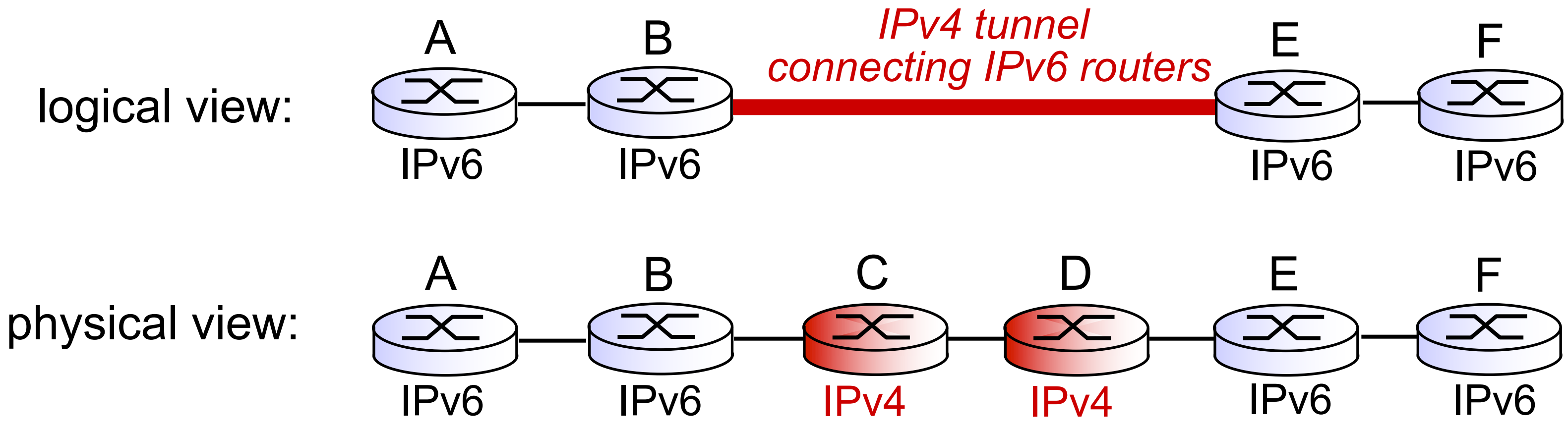
- *checksum:* removed entirely to reduce processing time at each hop
- *options:* allowed, but outside of header, indicated by "Next Header" field
- *ICMPv6:* new version of ICMP
  - additional message types, e.g. "Packet Too Big"
  - multicast group management functions

Transition from IPv4 to IPv6

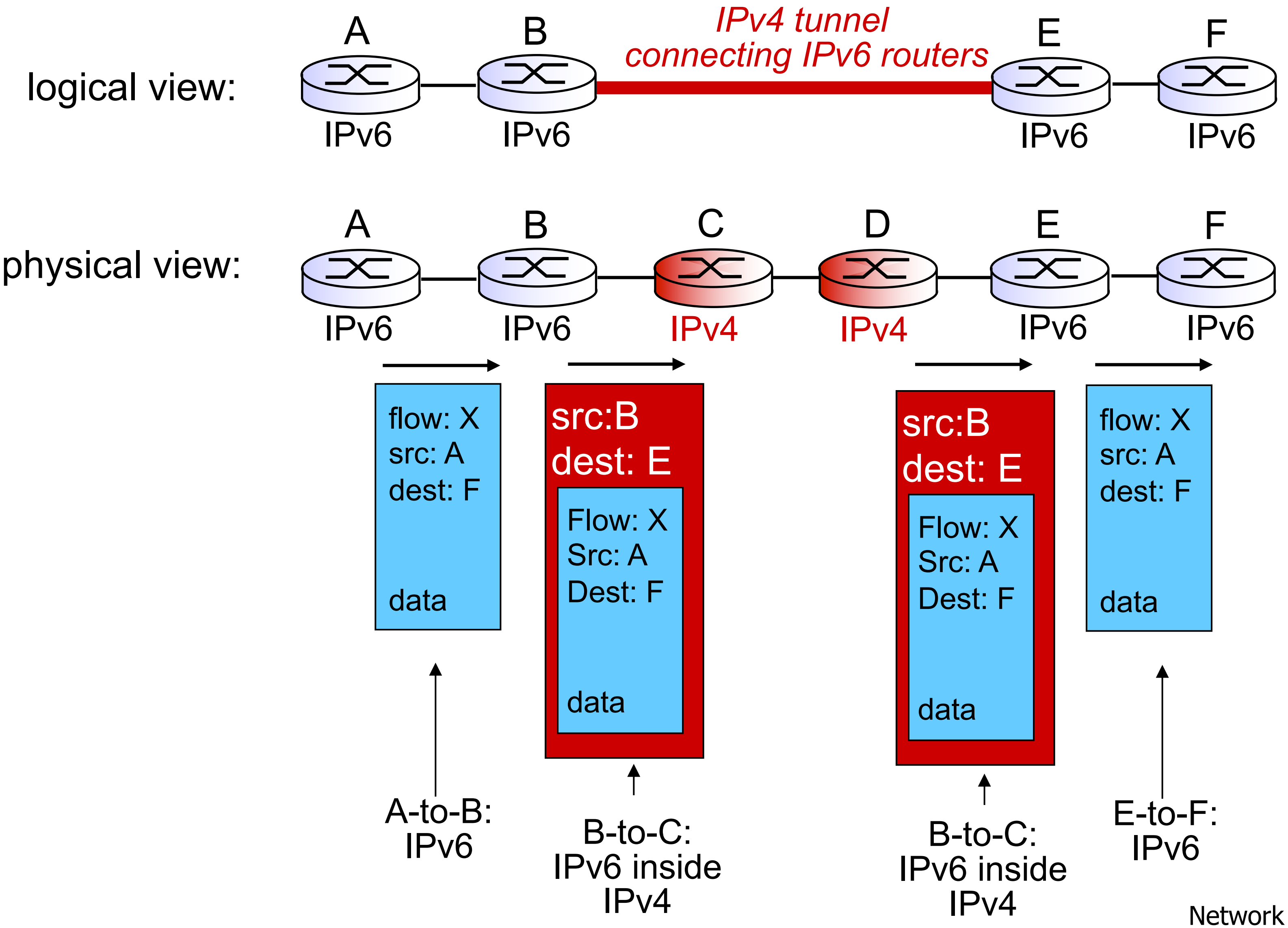
- not all routers can be upgraded simultaneously
  - no "flag days"
  - how will network operate with mixed IPv4 and IPv6 routers?
- *tunneling:* IPv6 datagram carried as *payload* in IPv4 datagram among IPv4 routers



Tunneling


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

Tunneling



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

Chapter 4: outline

4.1 introduction

4.2 virtual circuit and datagram networks

4.3 IP: Internet Protocol

- datagram format
- IPv4 addressing
- ICMP
- IPv6

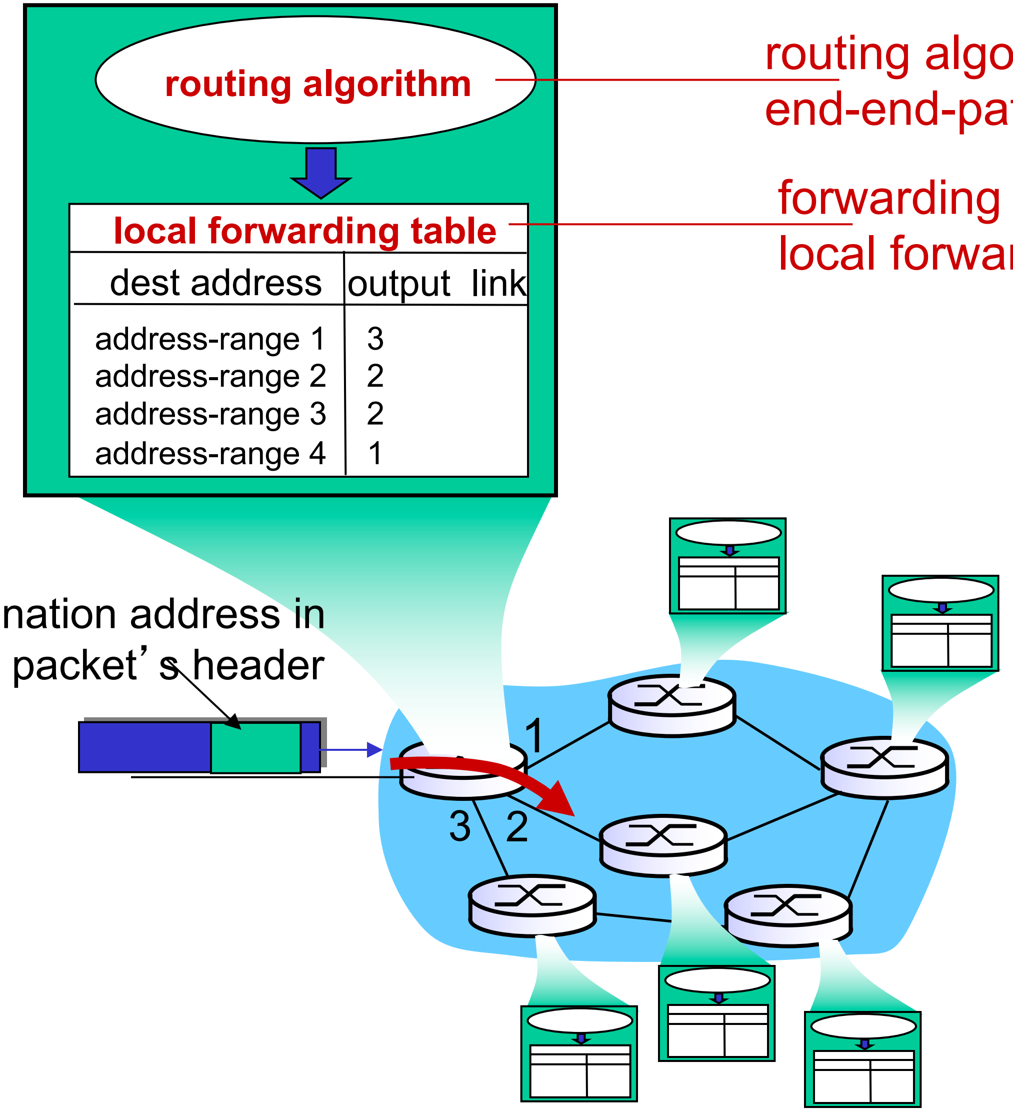
4.4 routing algorithms

- link state
- distance vector
- hierarchical routing

4.5 routing in the Internet

- RIP
- OSPF
- BGP

Interplay between routing, forwarding



routing algorithm

local forwarding table

dest address	output link
address-range 1	3
address-range 2	2
address-range 3	2
address-range 4	1

routing algorithm determines end-end-path through network

forwarding table determines local forwarding at this router

IP destination address in arriving packet’s header

Comparison of LS and DV algorithms

message complexity

- ***LS:*** with n nodes, E links, O(nE) msgs sent
- ***DV:*** exchange between neighbors only
  - convergence time varies

speed of convergence

- ***LS:*** O(n^2) algorithm requires O(nE) msgs
  - may have oscillations
- ***DV:*** convergence time varies
  - may be routing loops
  - count-to-infinity problem

robustness: what happens if router malfunctions?

LS:

- node can advertise incorrect *link* cost
- each node computes only its *own* table

DV:

- DV node can advertise incorrect *path* cost
- each node's table used by others
  - error propagate thru network

Chapter 4: outline

4.1 introduction

4.2 virtual circuit and datagram networks

4.3 IP: Internet Protocol

- datagram format
- IPv4 addressing
- ICMP
- IPv6

4.4 routing algorithms

- link state
- distance vector
- hierarchical routing

4.5 routing in the Internet

- RIP
- OSPF
- BGP

Hierarchical routing

our routing study thus far - idealization

- all routers identical
- network "flat"

… *not* true in practice

scale: with 600 million destinations:

- can't store all dest's in routing tables!
- routing table exchange would swamp links!

administrative autonomy

- internet = network of networks
- each network admin may want to control routing in its own network

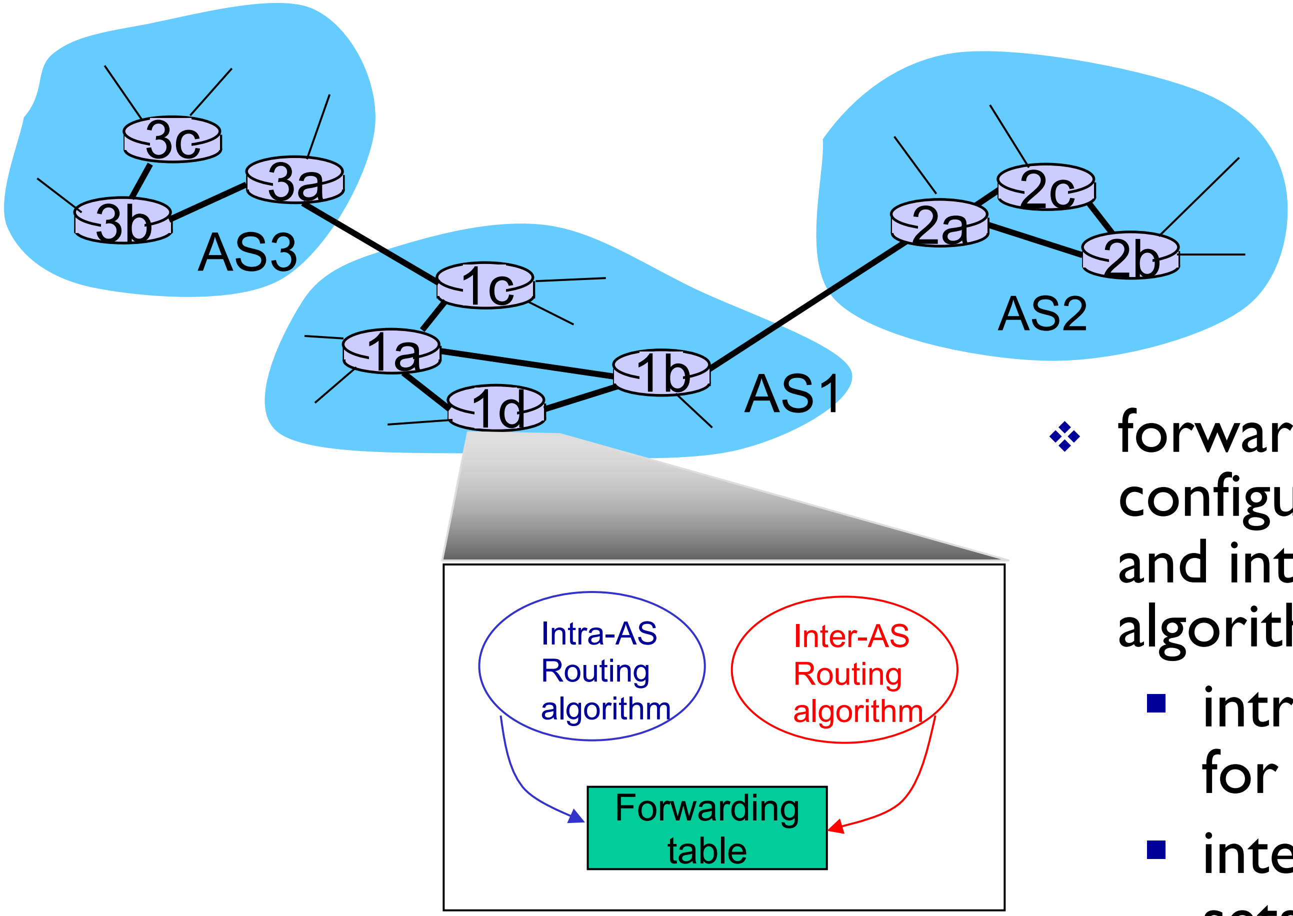
Hierarchical routing

- aggregate routers into regions, "autonomous systems" (AS)
- routers in same AS run same routing protocol
  - "intra-AS" routing protocol
  - routers in different AS can run different intra-AS routing protocol

gateway router:

- at "edge" of its own AS
- has link to router in another AS

Interconnected ASes



- forwarding table configured by both intra- and inter-AS routing algorithm
  - intra-AS sets entries for internal dests
  - inter-AS & intra-AS sets entries for external dests

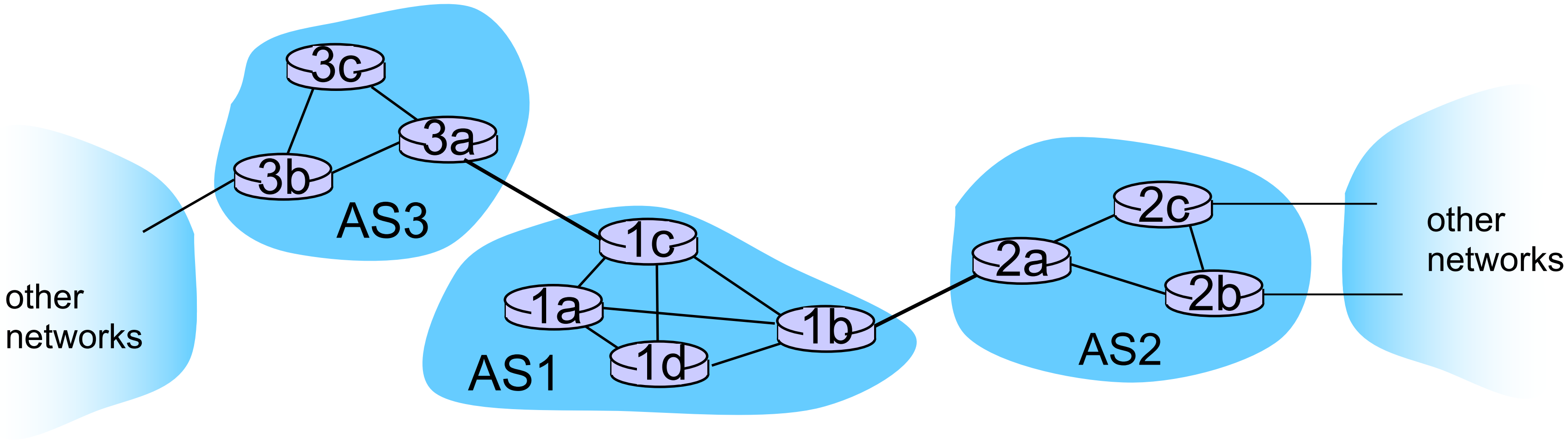
Inter-AS tasks

- suppose router in AS1 receives datagram destined outside of AS1:
  - router should forward packet to gateway router, but which one?

AS1 must:

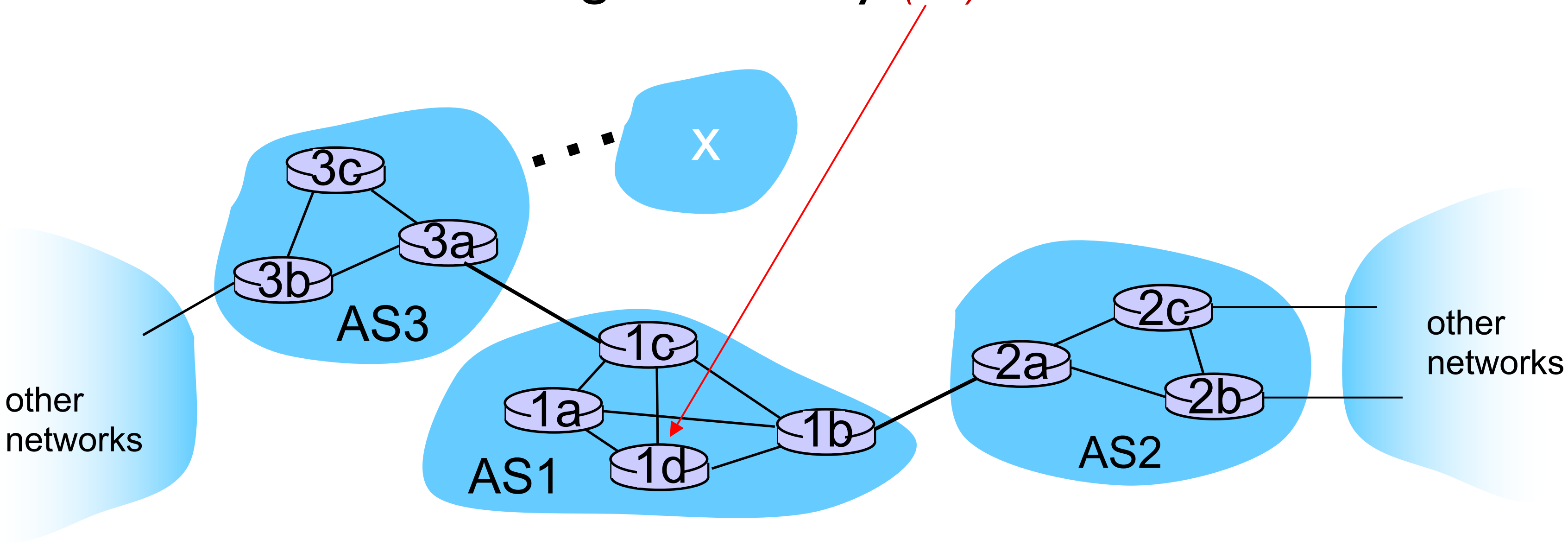
1. learn which dests are reachable through AS2, which through AS3
2. propagate this reachability info to all routers in AS1

job of inter-AS routing!



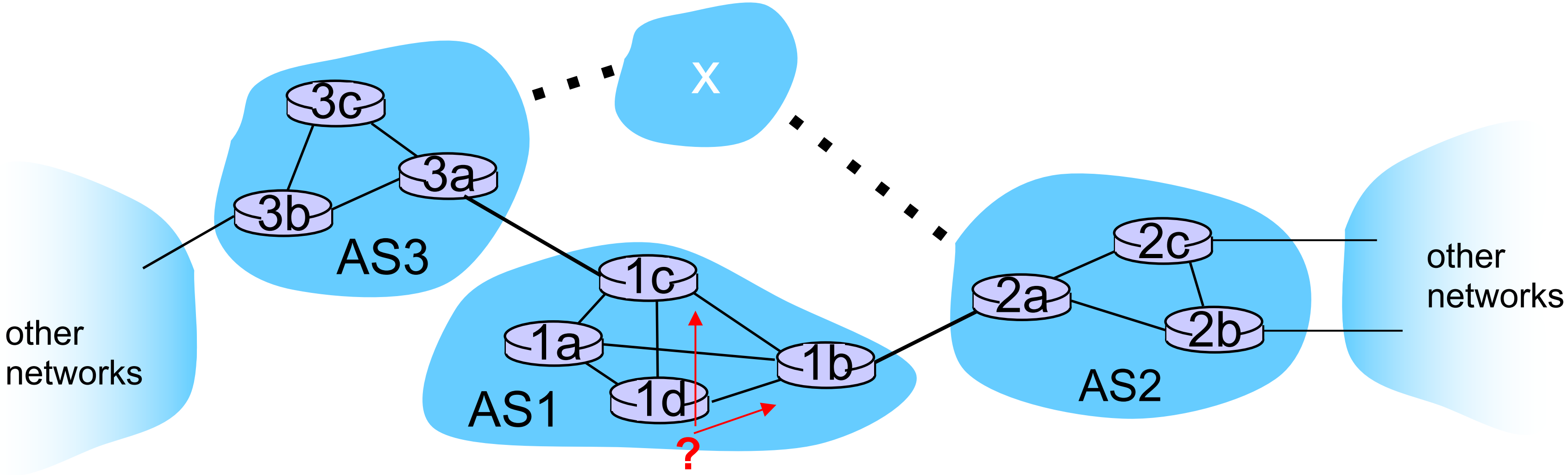
Example: setting forwarding table in router 1d

- suppose AS1 learns (via inter-AS protocol) that subnet *x* reachable via AS3 (gateway 1c), but not via AS2
  - inter-AS protocol propagates reachability info to all internal routers
- router 1d determines from intra-AS routing info that its interface *I* is on the least cost path to 1c
  - installs forwarding table entry *(x,I)*



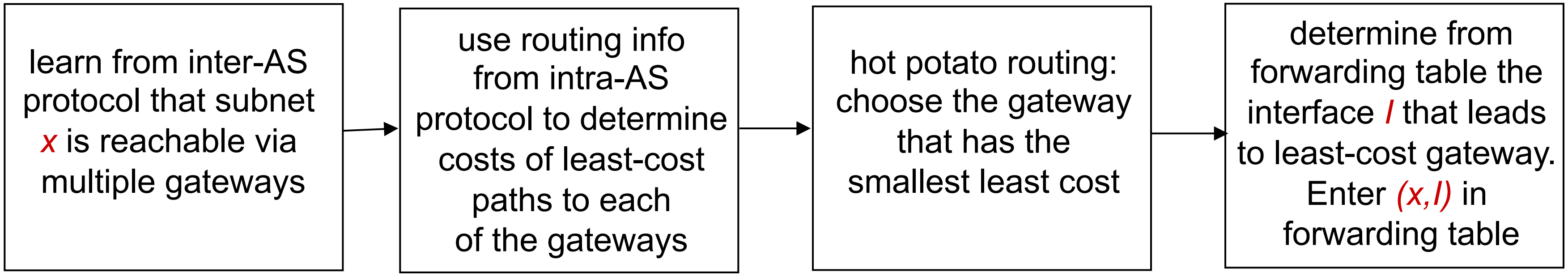
Example: choosing among multiple ASes

- now suppose AS1 learns from inter-AS protocol that subnet *x* is reachable from AS3 *and* from AS2.
- to configure forwarding table, router 1d must determine which gateway it should forward packets towards for dest x
  - this is also job of inter-AS routing protocol!



Example: choosing among multiple ASes

- now suppose AS1 learns from inter-AS protocol that subnet *x* is reachable from AS3 *and* from AS2.
- to configure forwarding table, router 1d must determine towards which gateway it should forward packets for dest x
  - this is also job of inter-AS routing protocol!
- *hot potato routing: send* packet towards closest of two routers.



learn from inter-AS protocol that subnet *x* is reachable via multiple gateways

use routing info from intra-AS protocol to determine costs of least-cost paths to each of the gateways

hot potato routing: choose the gateway that has the smallest least cost

determine from forwarding table the interface *I* that leads to least-cost gateway. Enter *(x,I)* in forwarding table

Chapter 4: outline

4.1 introduction

4.2 virtual circuit and datagram networks

4.3 IP: Internet Protocol

- datagram format
- IPv4 addressing
- ICMP
- IPv6

4.4 routing algorithms

- link state
- distance vector
- hierarchical routing

4.5 routing in the Internet

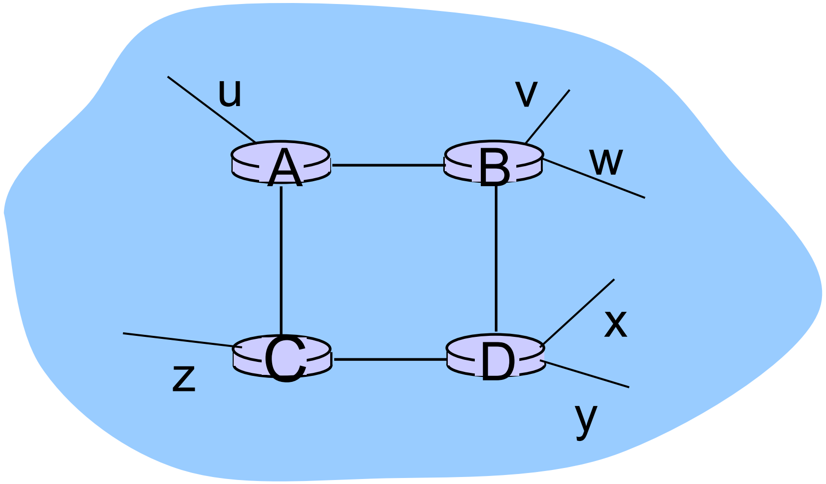
- RIP
- OSPF
- BGP

Intra-AS Routing

- also known as *interior gateway protocols (IGP)*
- most common intra-AS routing protocols:
  - RIP: Routing Information Protocol
  - OSPF: Open Shortest Path First
  - IGRP: Interior Gateway Routing Protocol (Cisco proprietary)

RIP (Routing Information Protocol)

- included in BSD-UNIX distribution in 1982
- distance vector algorithm
  - distance metric: # hops (max = 15 hops), each link has cost 1
  - DVs exchanged with neighbors every 30 sec in response message (aka advertisement)
  - each advertisement: list of up to 25 destination *subnets* *(in IP addressing sense)*



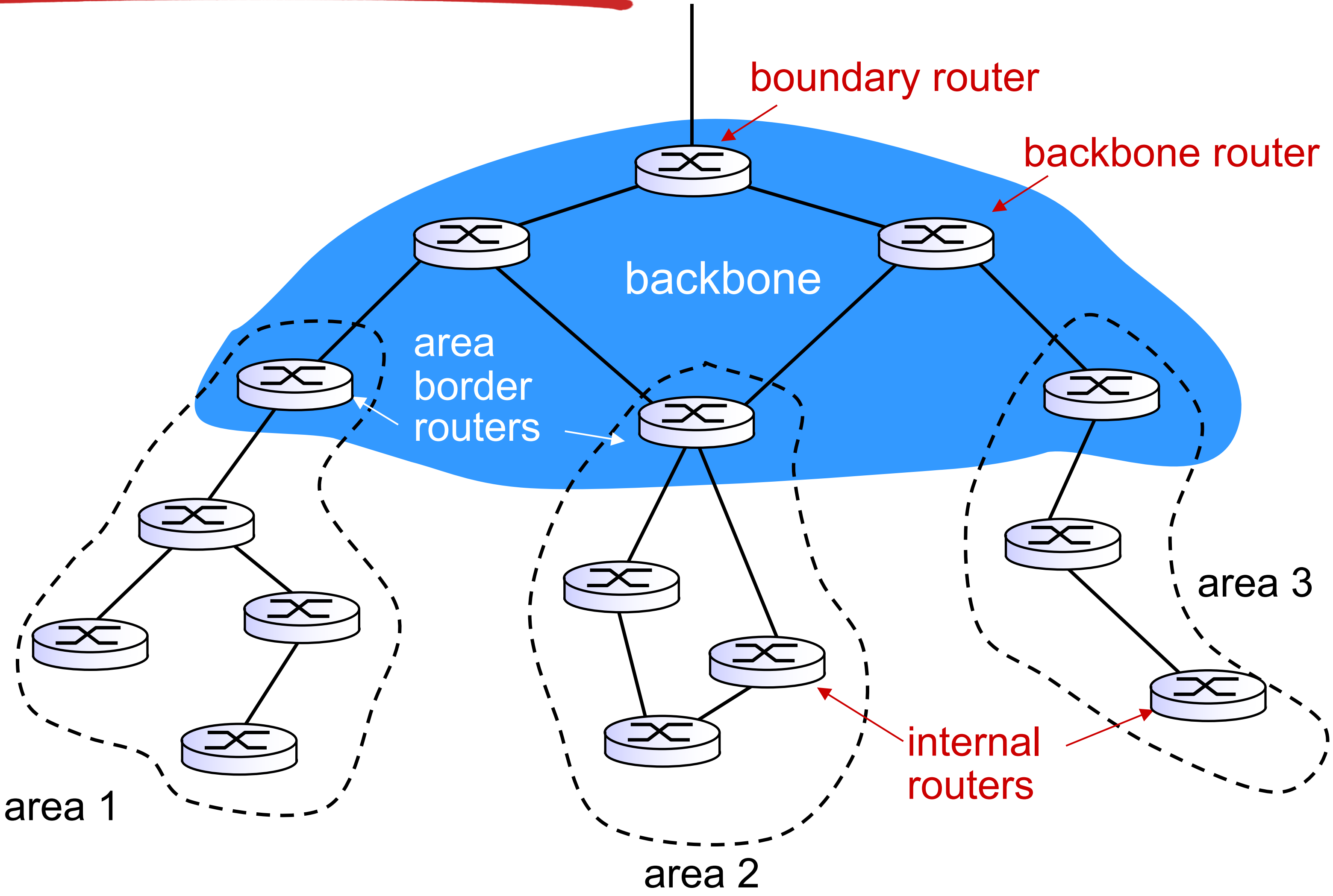
from router A to destination *subnets:*

subnet	hops
u	1
v	2
w	2
x	3
y	3
z	2

OSPF (Open Shortest Path First)

- "open": publicly available
- uses link state algorithm
  - LS packet dissemination
  - topology map at each node
  - route computation using Dijkstra's algorithm
- OSPF advertisement carries one entry per neighbor
- advertisements flooded to *entire* AS
  - carried in OSPF messages directly over IP (rather than TCP or UDP
- *IS-IS routing* protocol: nearly identical to OSPF

Hierarchical OSPF



Internet inter-AS routing: BGP

- BGP (Border Gateway Protocol): *the* de facto inter-domain routing protocol
  - "glue that holds the Internet together"
- BGP provides each AS a means to:
  - eBGP: obtain subnet reachability information from neighboring ASs.
  - iBGP: propagate reachability information to all AS-internal routers.
  - determine "good" routes to other networks based on reachability information and policy.
- allows subnet to advertise its existence to rest of Internet: *"I am here"*

Why different Intra-, Inter-AS routing ?

policy:

- inter-AS: admin wants control over how its traffic routed, who routes through its net.
- intra-AS: single admin, so no policy decisions needed

scale:

- hierarchical routing saves table size, reduced update traffic

performance:

- intra-AS: can focus on performance
- inter-AS: policy may dominate over performance