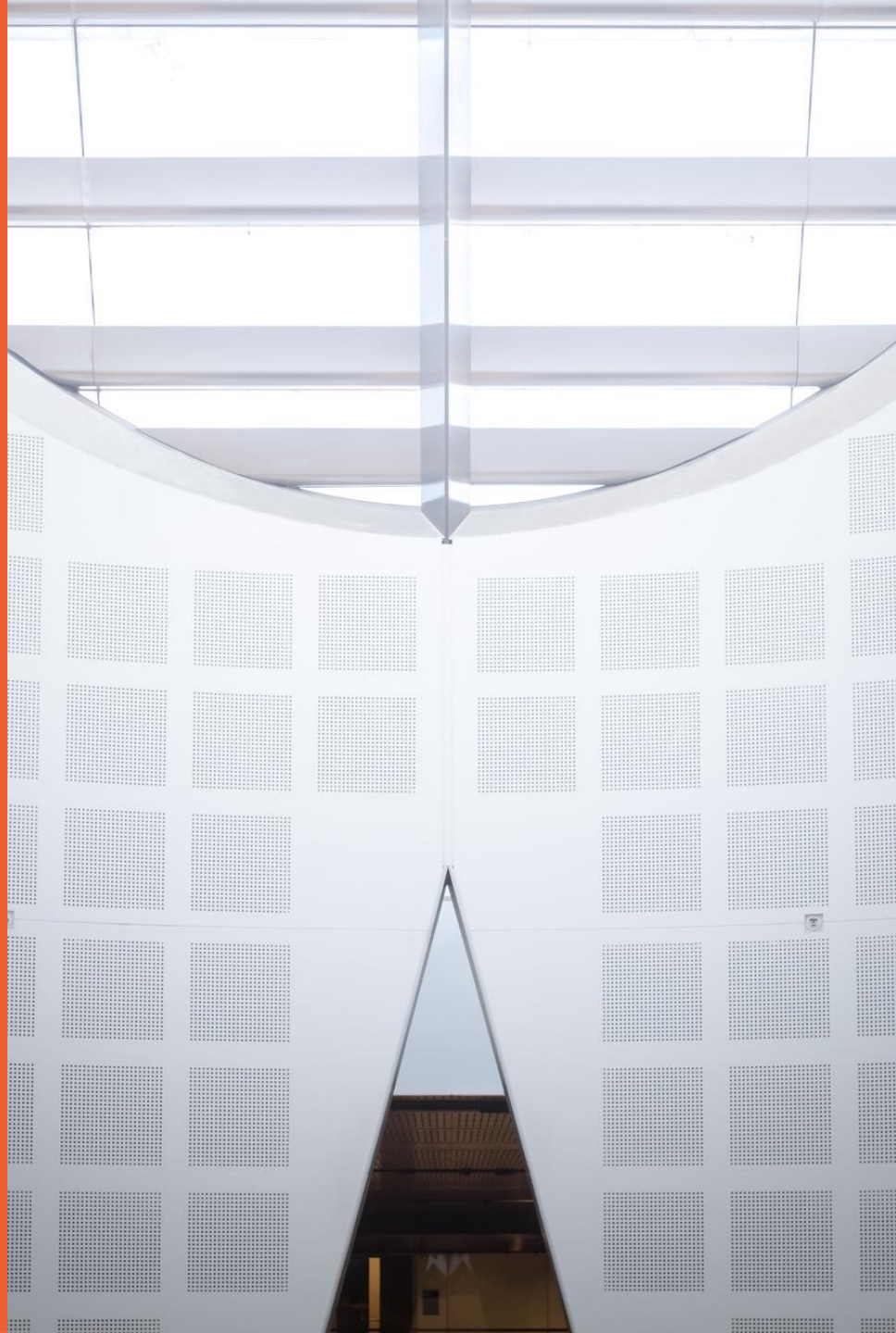
COMP9121: Design of Networks and Distributed Systems

Week 4: Network Layer 1

Wei Bao

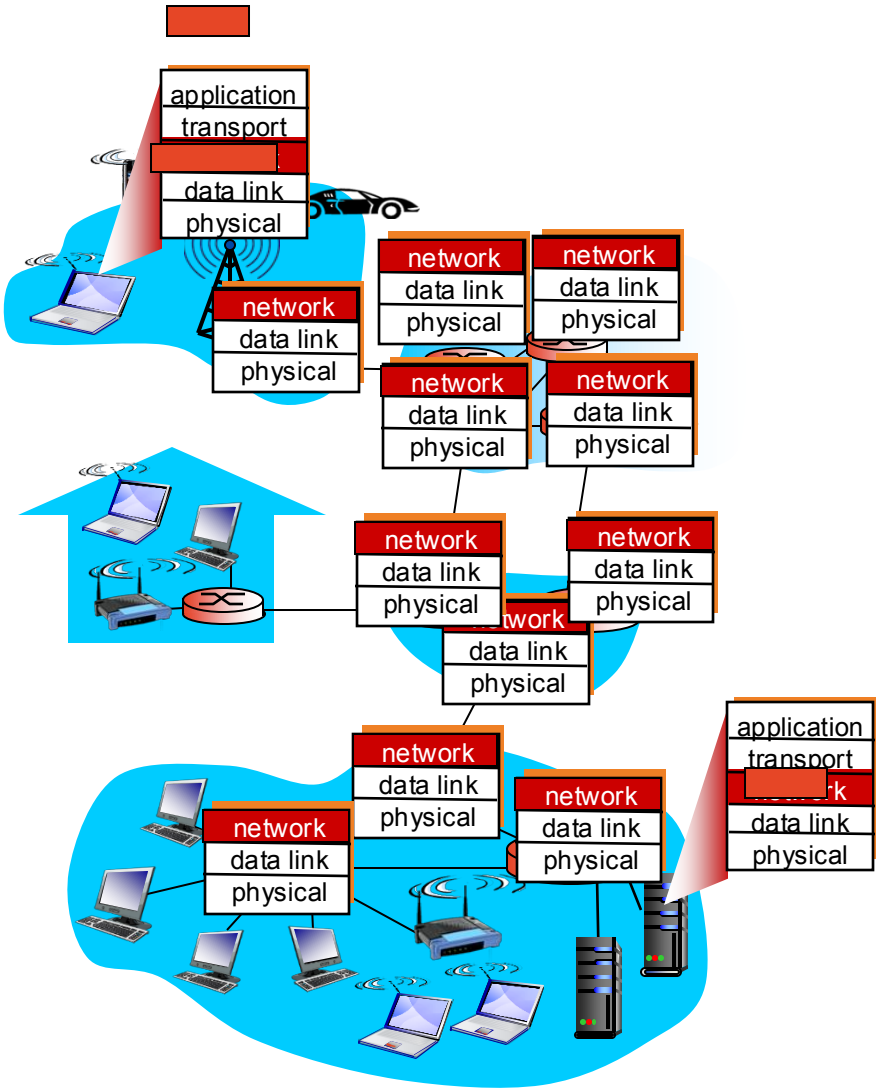
School of Computer Science

THE UNIVERSITY OF SYDNEY



Network layer

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



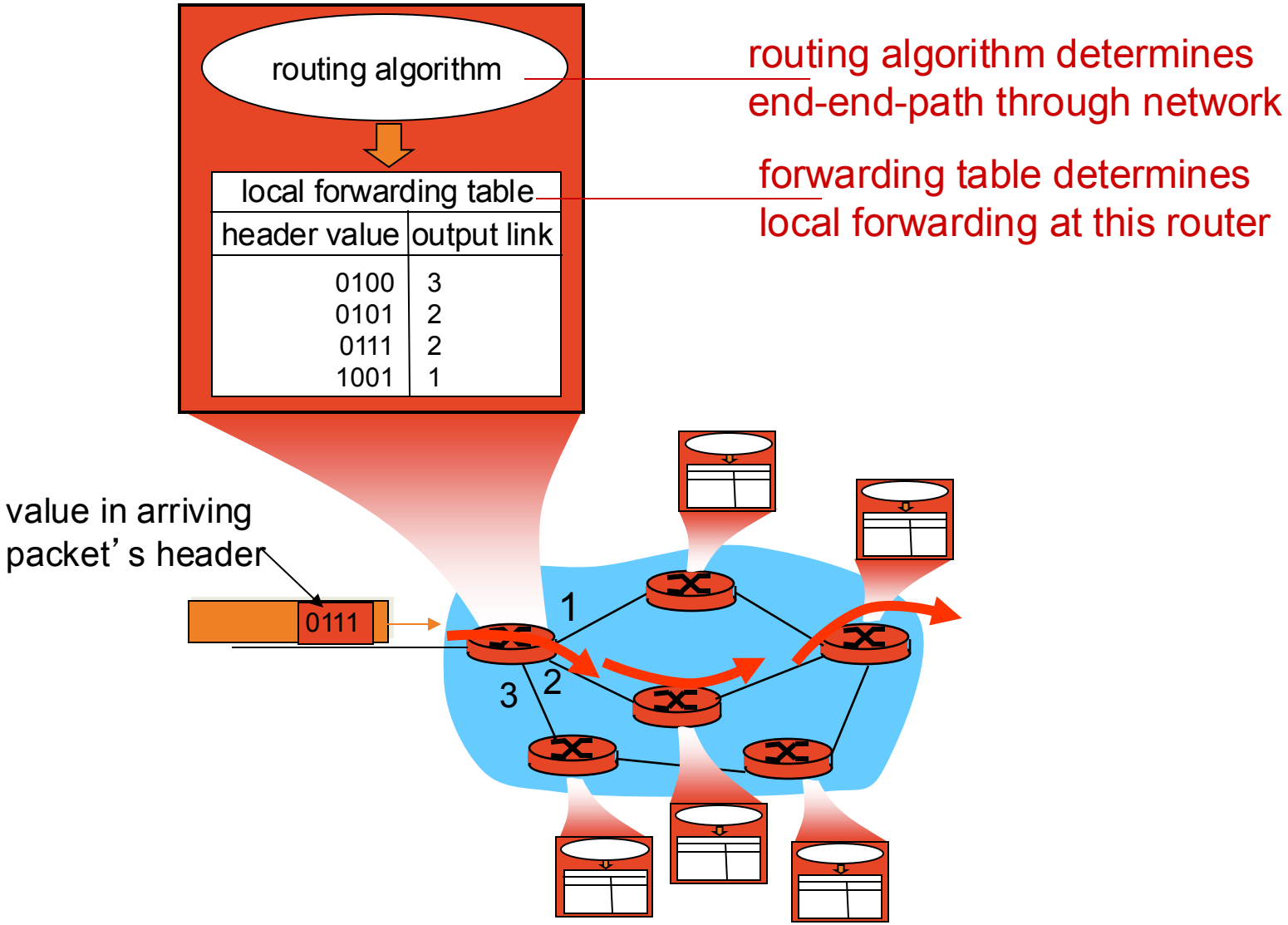
Two key network-layer functions

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

analogy:

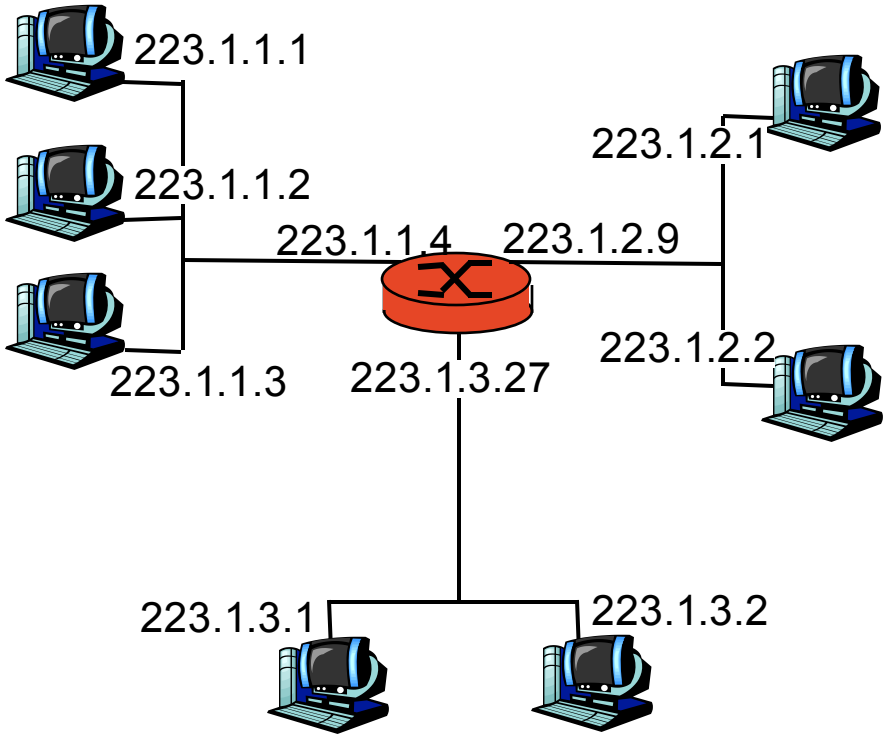
- *routing:* process of planning trip from source to dest
- *forwarding:* process of getting through single interchange

Interplay between routing and forwarding



IP Addressing

- IP address: 32-bit identifier for host, router *interface*
- Dotted decimal notation: int1.int2.int3.int4
- *interface:* connection between host/router and physical link
  - Router's typically have multiple interfaces
  - Host typically has one interface
  - IP addresses associated with each interface
- Q: What about switches or hubs?
- A: They don't have IP addresses.

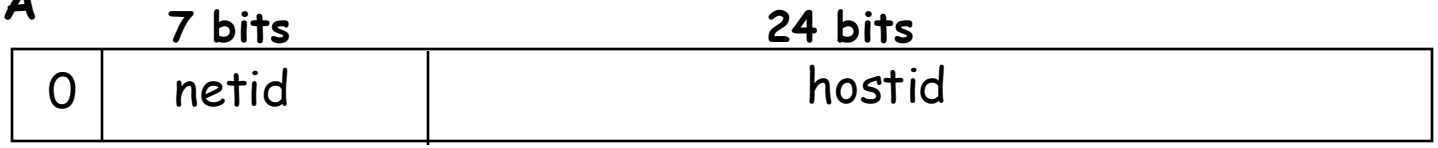


223.1.1.1 = 11011111 00000001 00000001 00000001

223 1 1 1

Classful Addresses

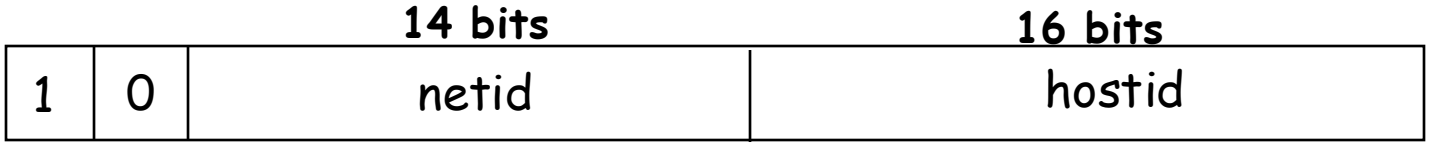
Class A



- 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

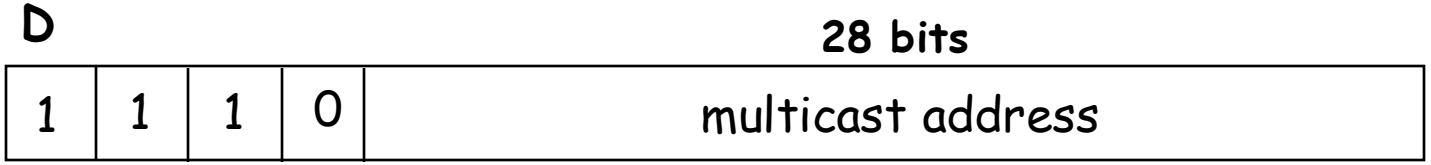
Class C

- 2 million networks with up to 254 hosts

192.0.0.0 to 223.255.255.255

Classful Addresses

Class D



224.0.0.0 to
239.255.255.255

IP Address Problems

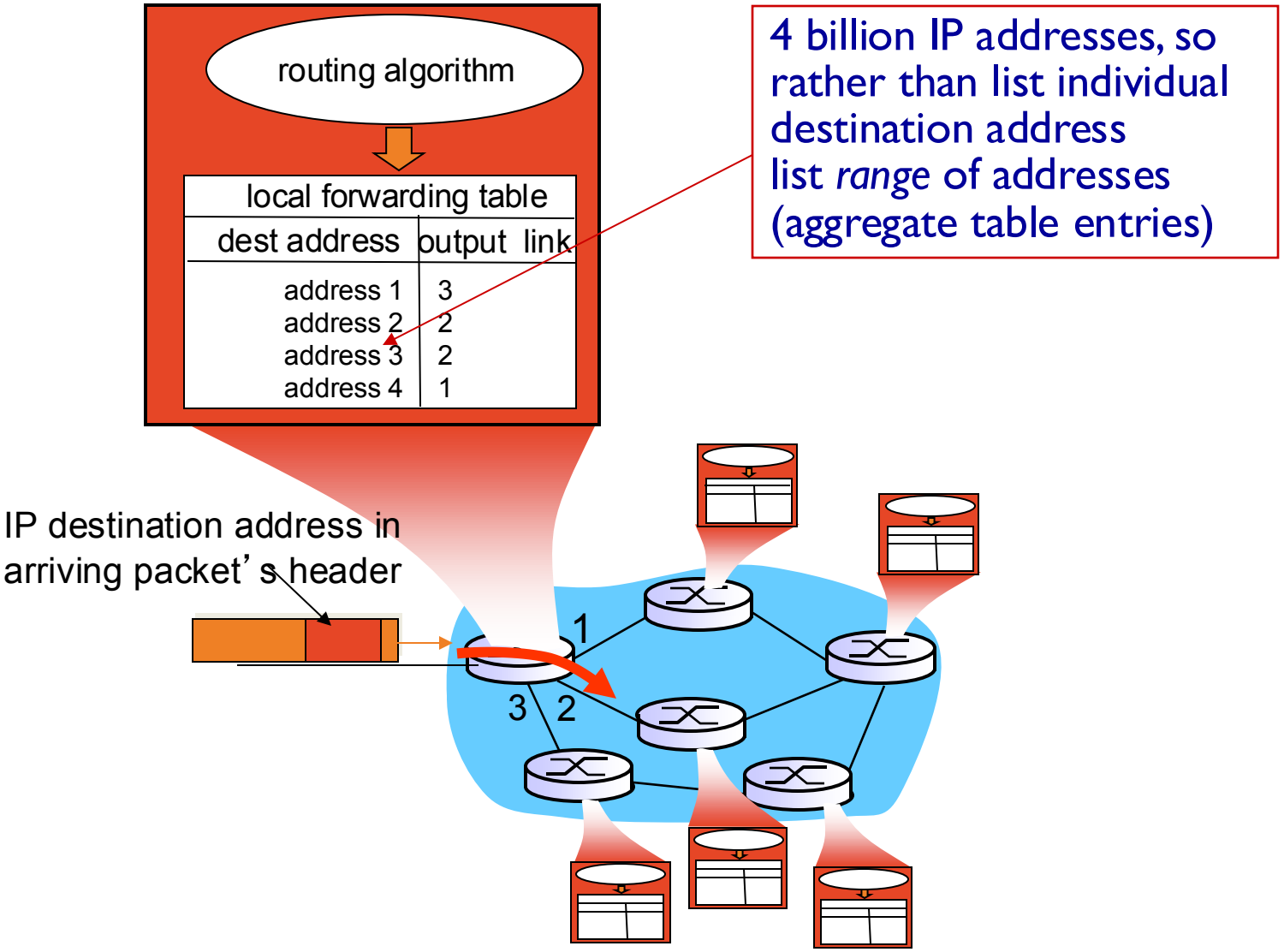
- In the 1990, two problems became apparent
  - IP addresses were being exhausted
  - IP routing tables were growing very large

- 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

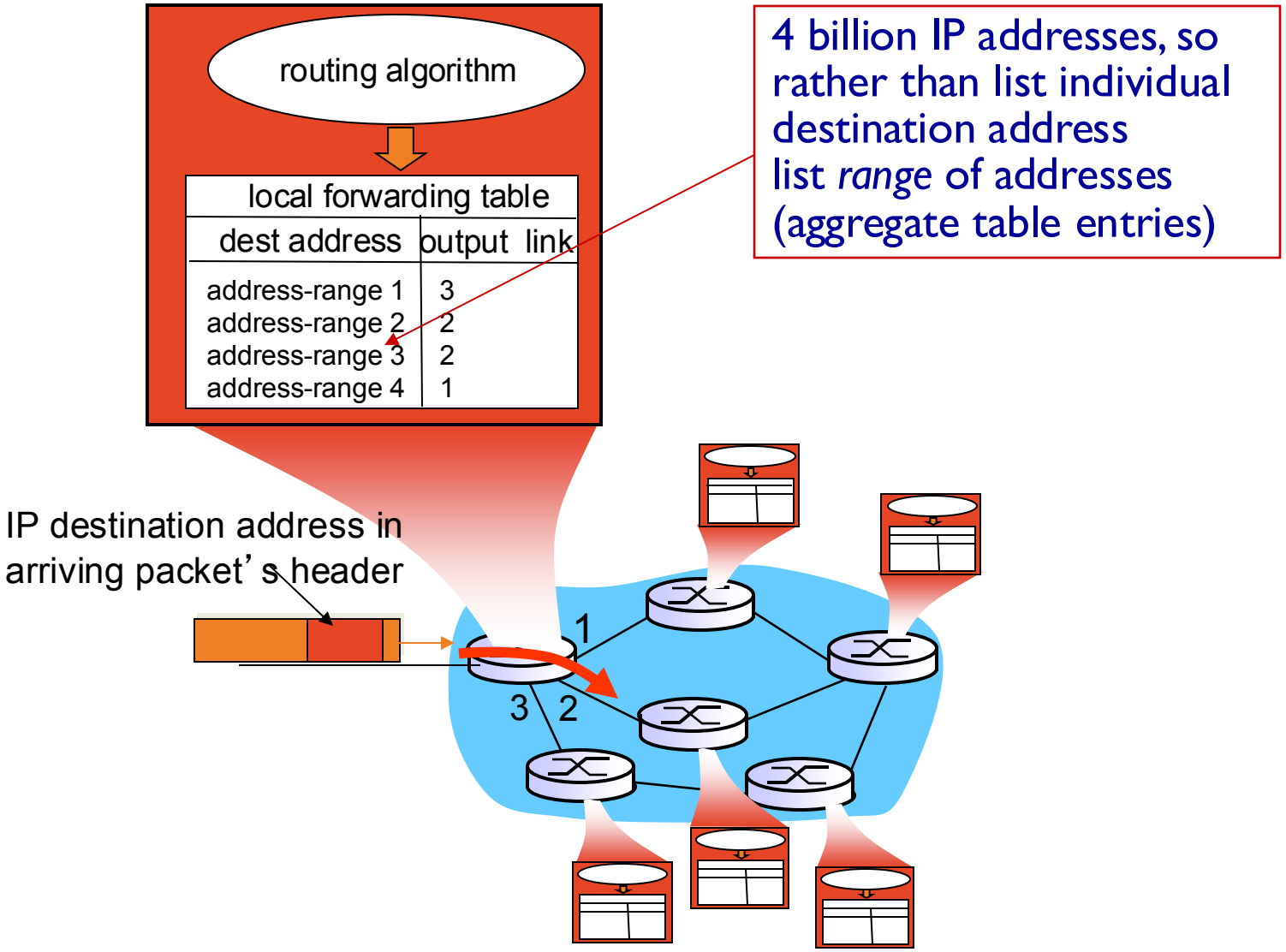
Solutions

- Short-term solution:
  - Classless Interdomain Routing (CIDR)
  - Private IP Addresses set aside for intranets
- Long-term solution:
  - IPv6 with much bigger address space
- We will discuss them later

Forwarding table



Forwarding table



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

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

Forwarding table

Destination Address Range	Link Interface
11001000 00010111 00010xxx xxxxxxxx through 11001000 00010111 00010xxx xxxxxxxx	0
11001000 00010111 00011000 xxxxxxxx through 11001000 00010111 00011000 xxxxxxxx	1
11001000 00010111 00011001 xxxxxxxx through 11001000 00010111 00011111 xxxxxxxx	2
otherwise	3

Forwarding table

Destination Address Range	Link Interface
11001000 00010111 00010xxx xxxxxxxx	0
11001000 00010111 00011000 xxxxxxxx	1
11001000 00010111 00011001 xxxxxxxx	2
otherwise	3

上面是perfetly分的情况

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

Longest prefix matching

longest prefix matching

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

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

examples:

DA: 11001000 00010111 00011000 10101010

which interface?

Longest prefix matching

longest prefix matching

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

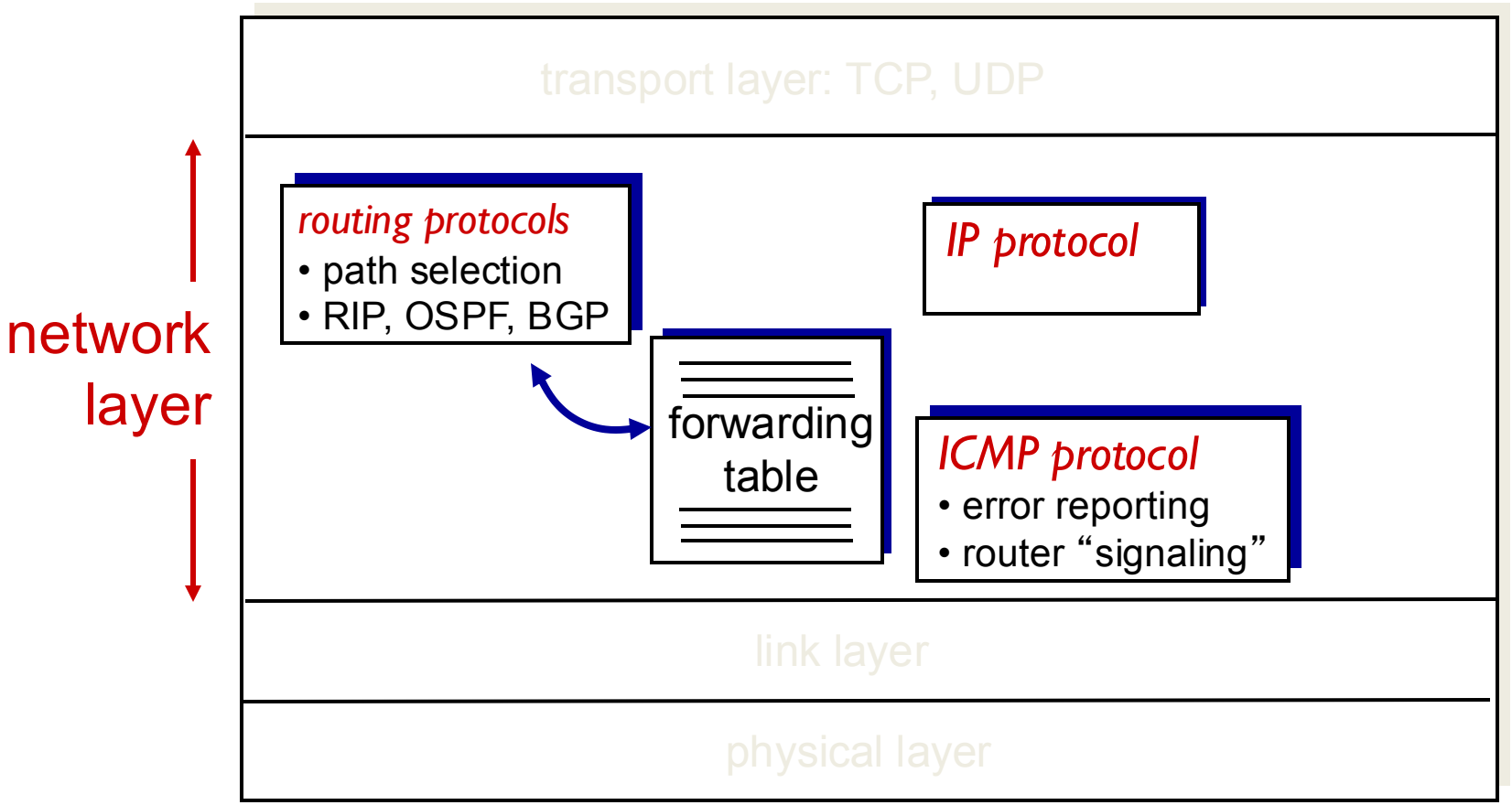
Destination Address Range	Link interface
11001000 00010111 00010*** *********	0
11001000 00010111 00011000 *********	1
11001000 00010111 00011*** *********	2
otherwise	3

examples:

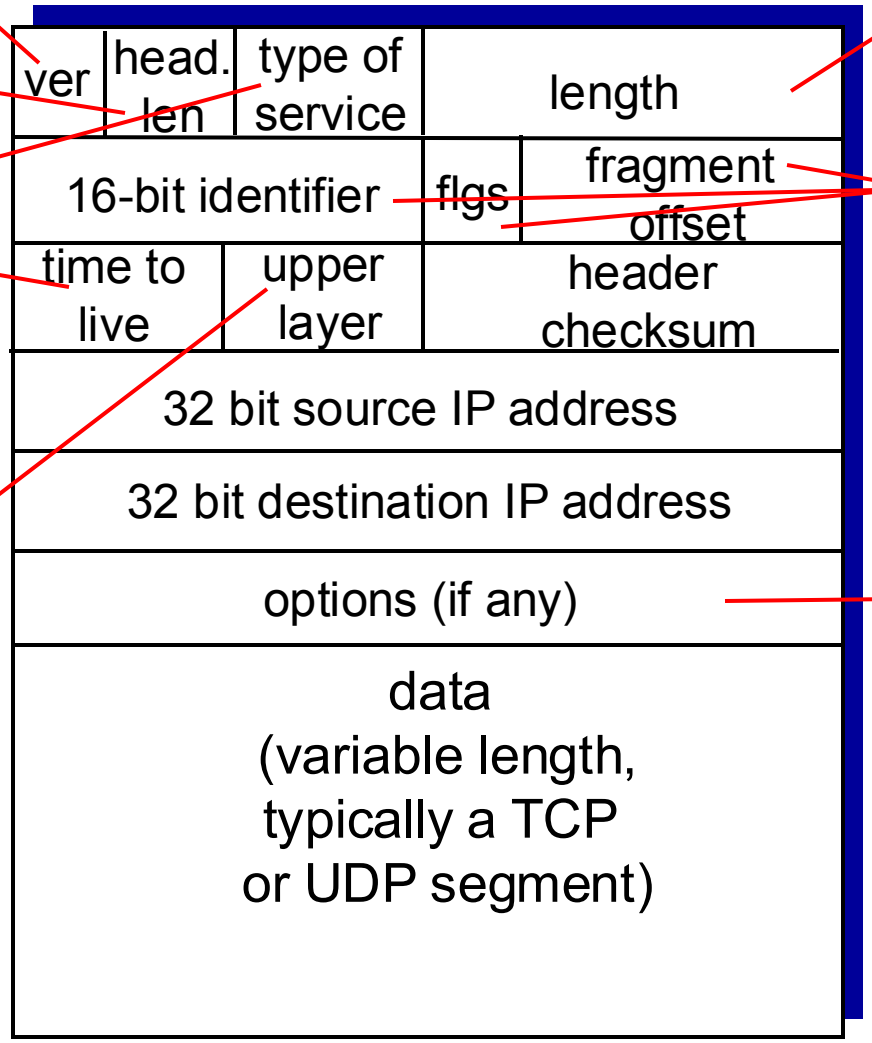
DA: 11001000 00010111 00011000 10101010 which interface?

The network layer

host, router network layer functions:



IP datagram format



IP protocol version number

header length (bytes)

"type" of data

max number remaining hops (decremented at each router)

Why useful?

upper layer protocol to deliver payload to

total datagram length (bytes)

for fragmentation/ reassembly

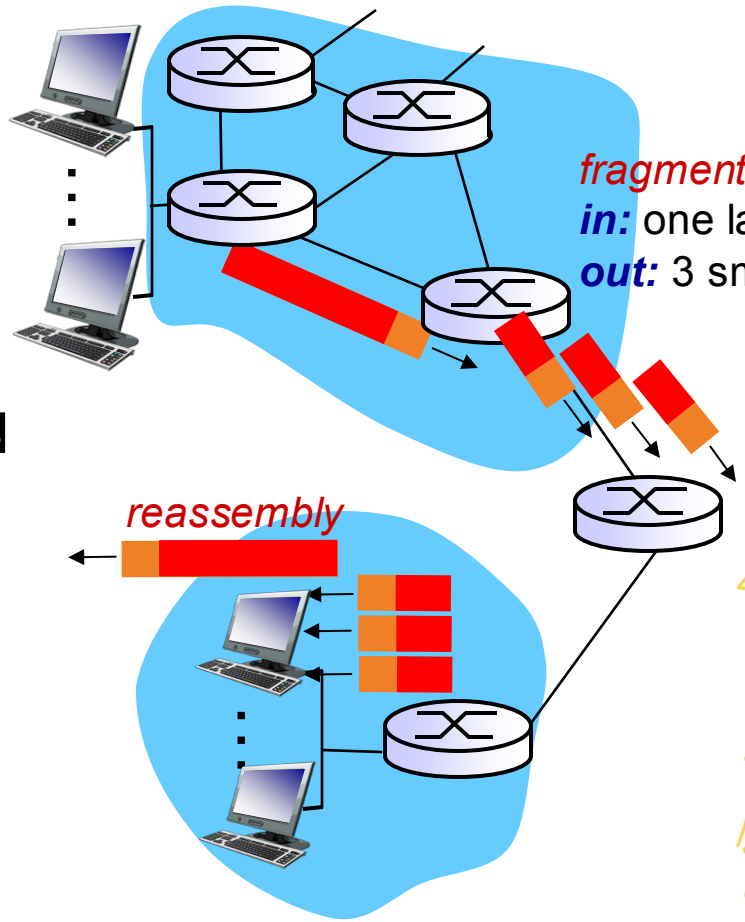
e.g. timestamp, record route taken, specify list of routers to visit.

how much overhead?

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

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

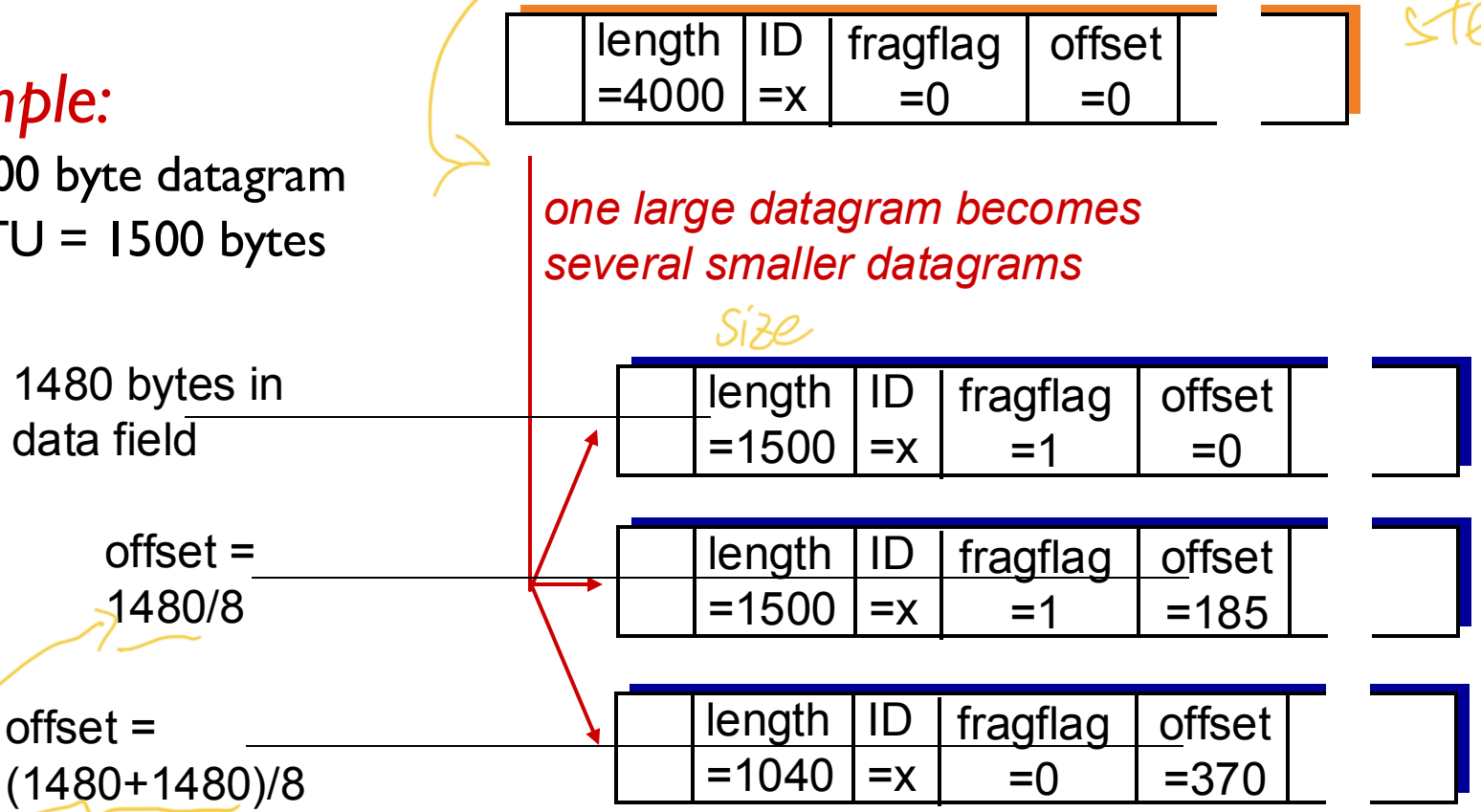


IP fragmentation, reassembly

example:

- 4000 byte datagram
- MTU = 1500 bytes

one large datagram becomes several smaller datagrams



- 3980 byte payload = 1480 +1480 +1020 bytes

Classless Interdomain Routing (CIDR)

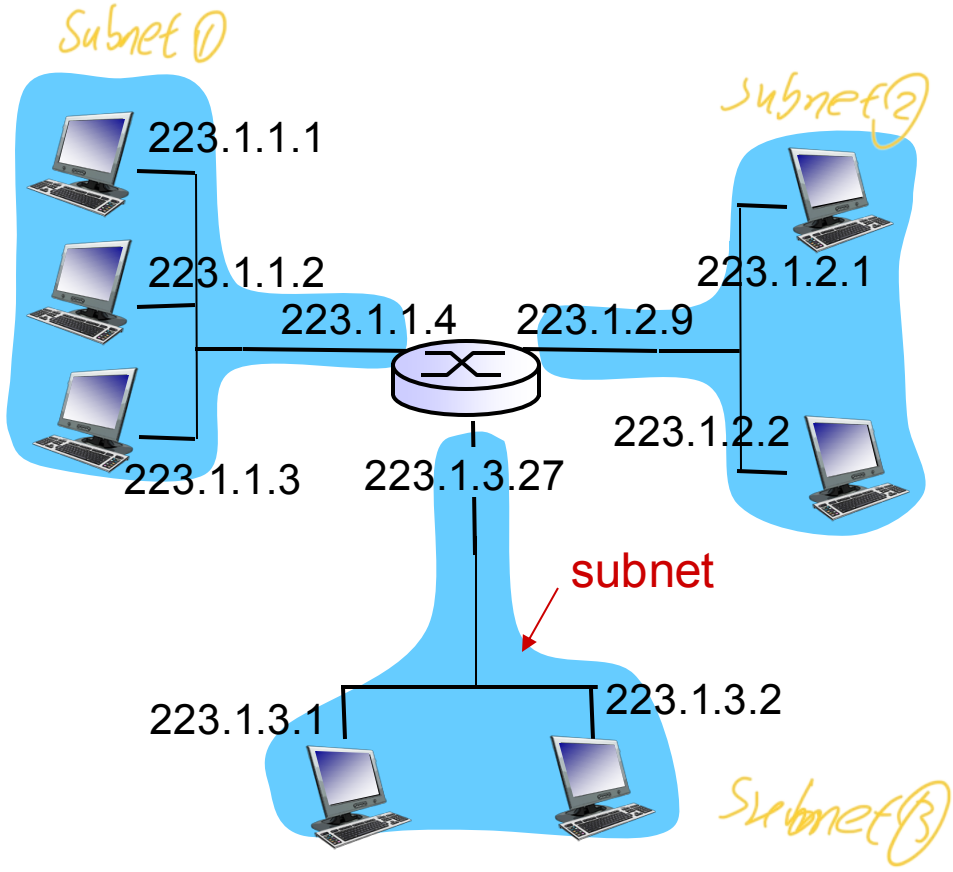
- Class A, B, and C address structure inefficient
- Class B too large for most organizations, but future proof
- Class C too small
- Rate of class B allocation implied exhaustion by 1994

- Solution: Classless Interdomain Routing (CIDR)

Subnets

- *What's a subnet ?*
  - can physically reach each other *without intervening router*
- IP address:
  - subnet part - high order bits
  - host part - low order bits
  - device interfaces with same subnet part of IP address

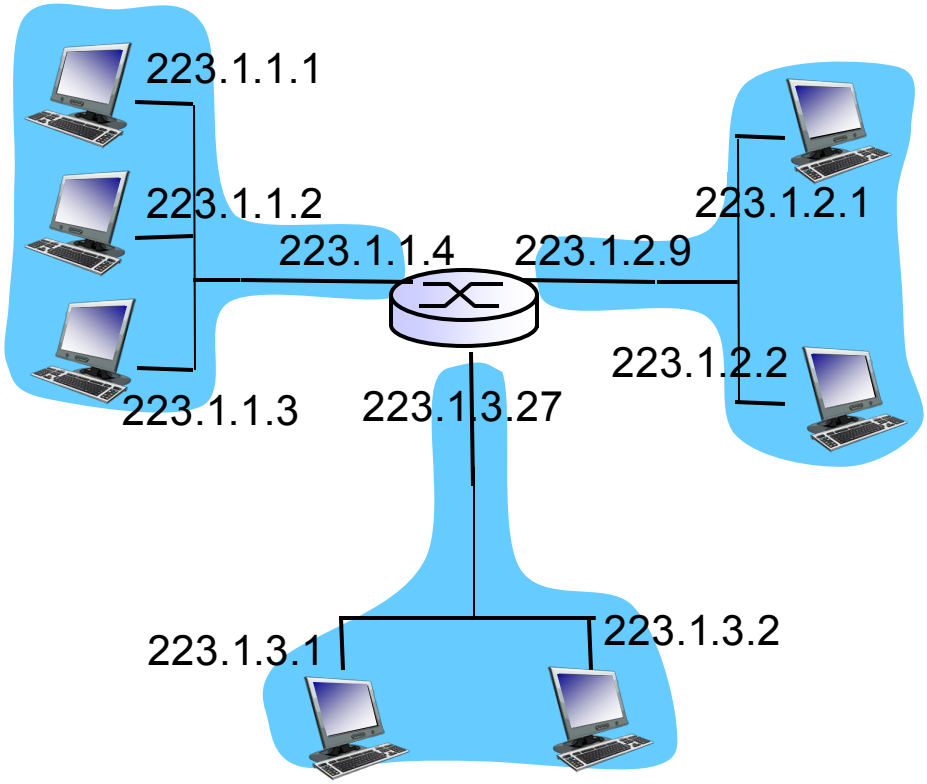
Subnet bits	Host bits



network consisting of 3 subnets

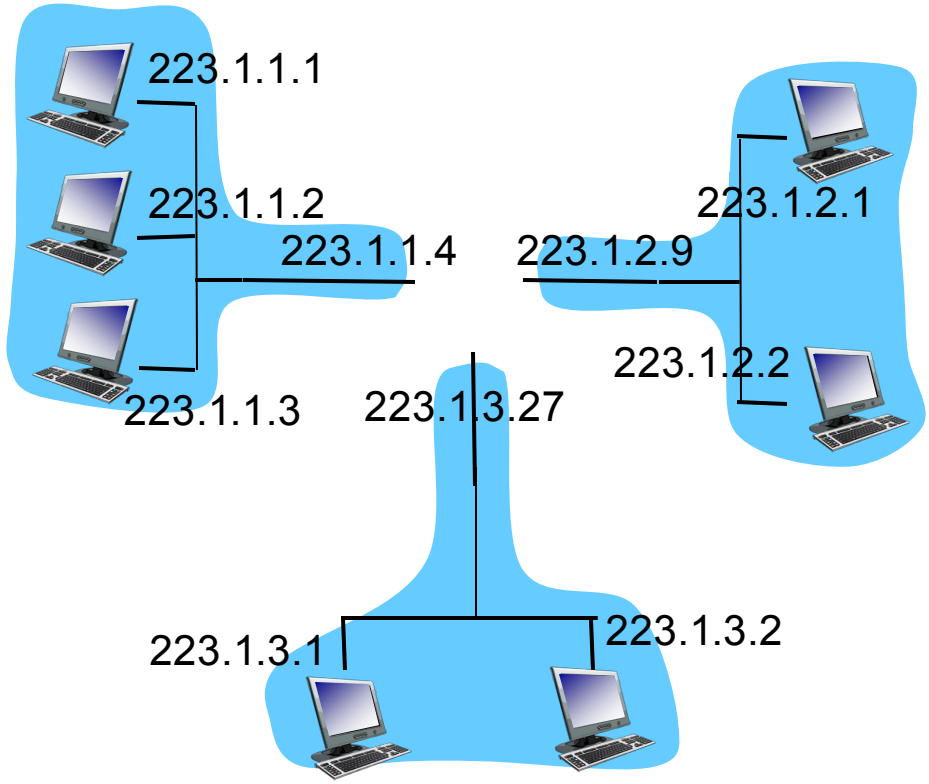
Subnets

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



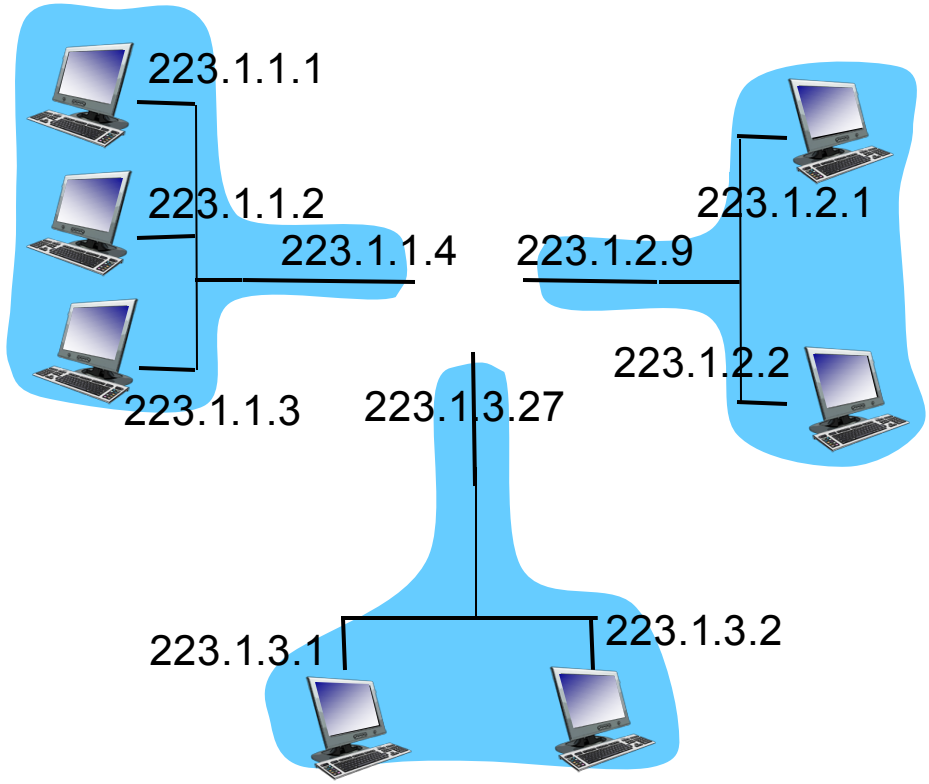
Subnets

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



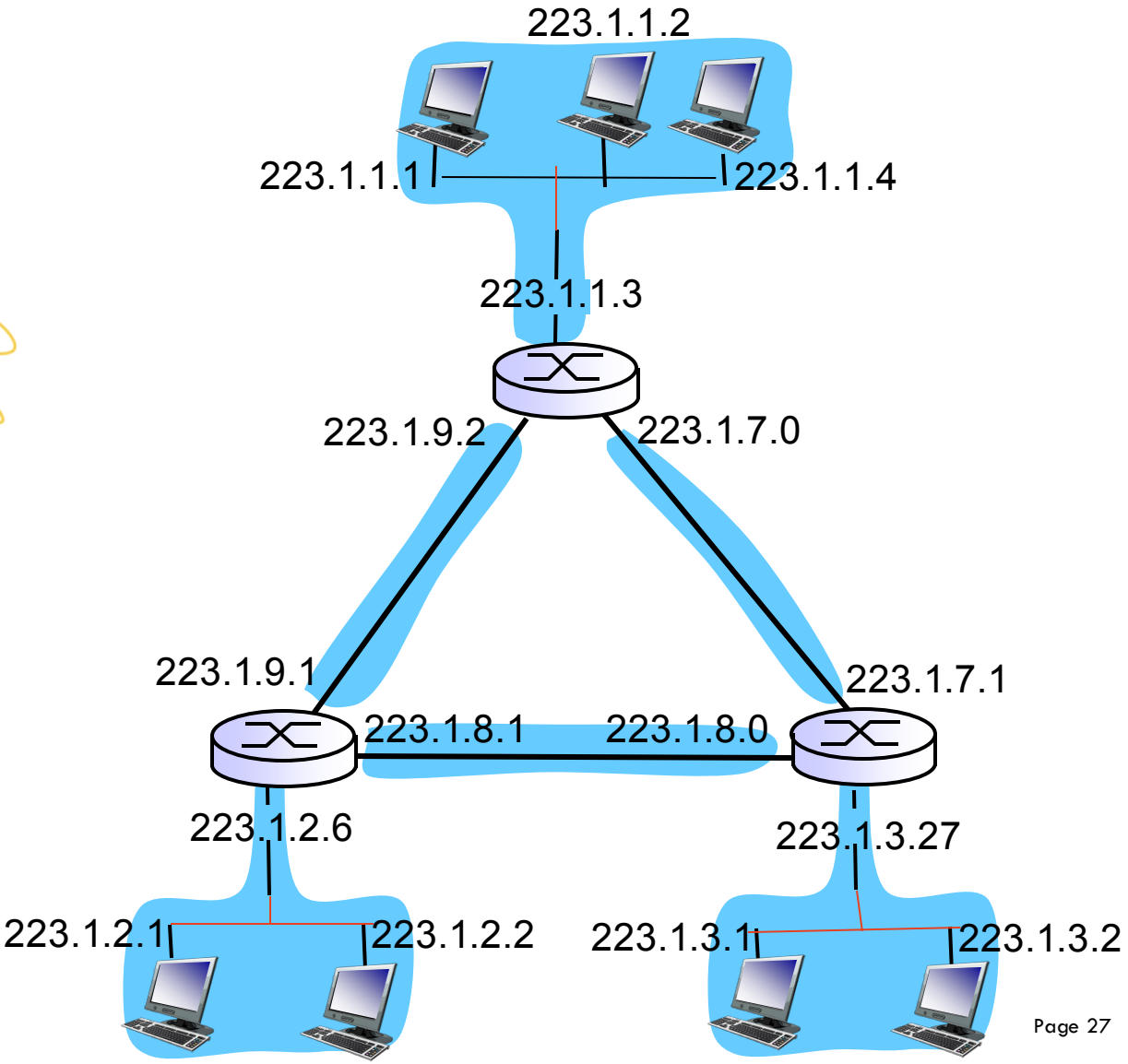
Subnets

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



Subnets

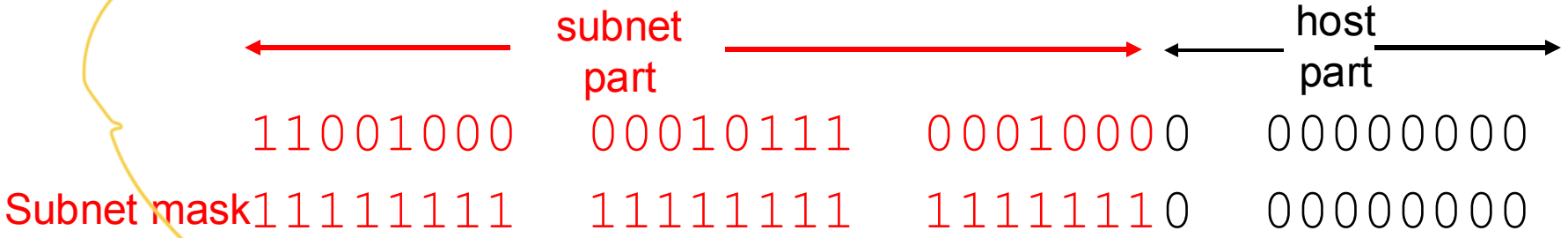
how many subnets?



IP addressing: CIDR

CIDR: Classless InterDomain Routing

- subnet portion of address of arbitrary length
- address format: a.b.c.d/x, where x is # bits in subnet portion of address
- subnet mask: all "1"s in subnet portion, all "0"s in host portion



200.23.16.0/23

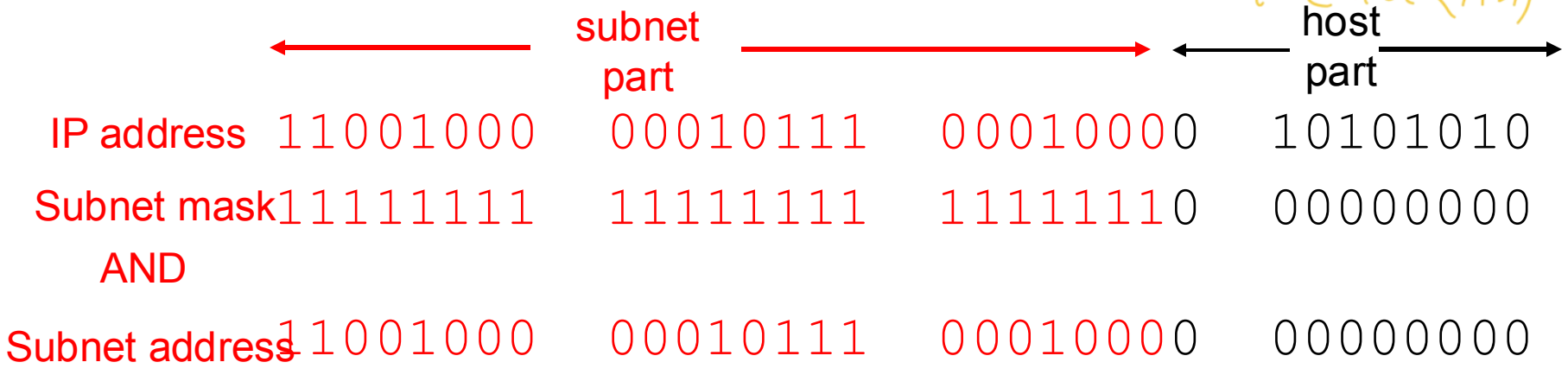
200.23.16.0 with subnet mask 255.255.254.0

IP addressing: CIDR

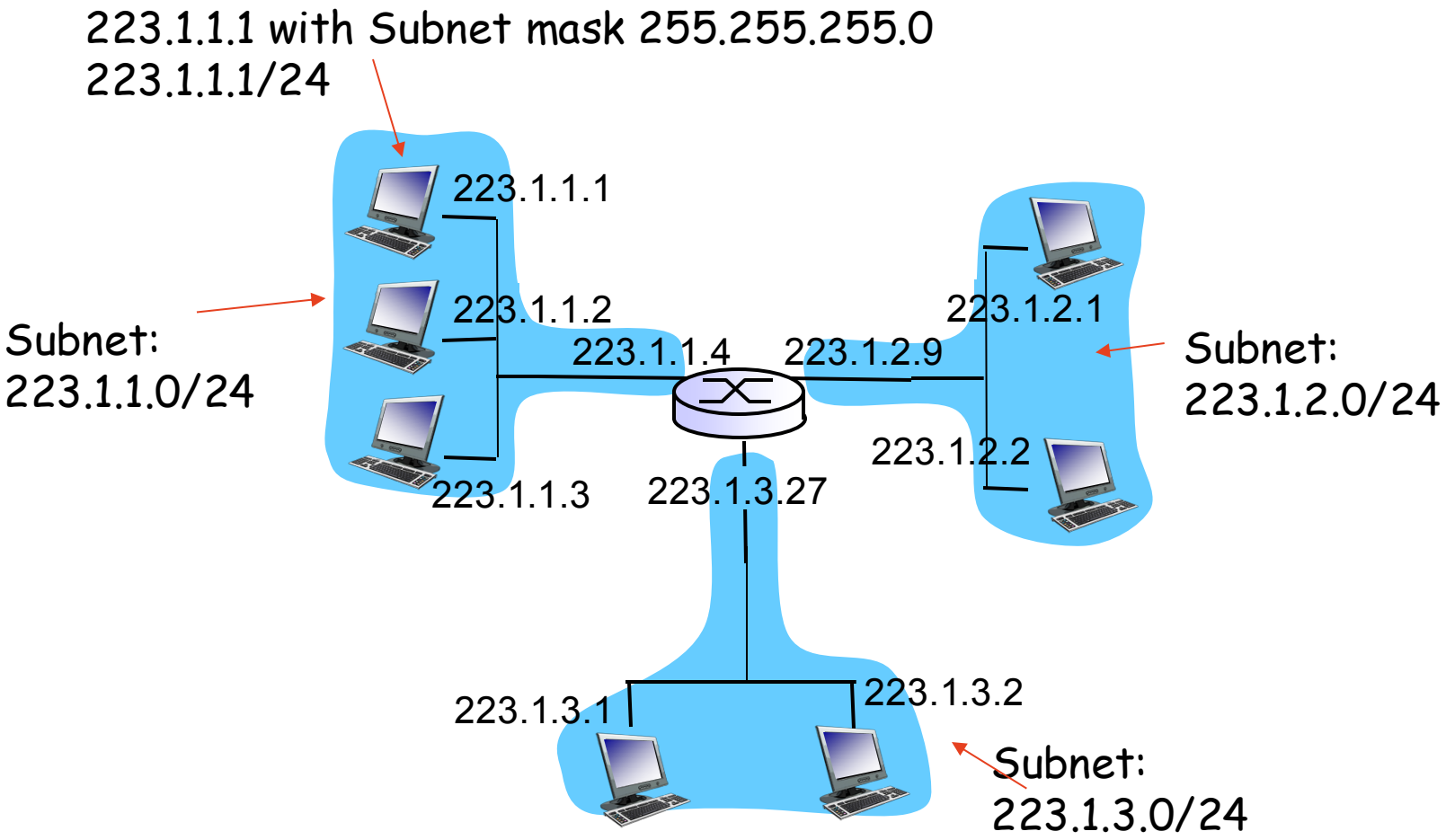
为什么需要 subnet mask:
do "and" operation

可以filter host部分
(知道Host是什么)

Subnet mask:



Subnets


223.1.1.1 with Subnet mask 255.255.255.0
223.1.1.1/24
223.1.1.1
223.1.1.2
223.1.1.3
223.1.1.4
223.1.2.9
223.1.2.1
223.1.2.2
223.1.3.27
223.1.3.1
223.1.3.2
Subnet:
223.1.1.0/24
Subnet:
223.1.2.0/24
Subnet:
223.1.3.0/24

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

why we need?

DHCP: Dynamic Host Configuration Protocol

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

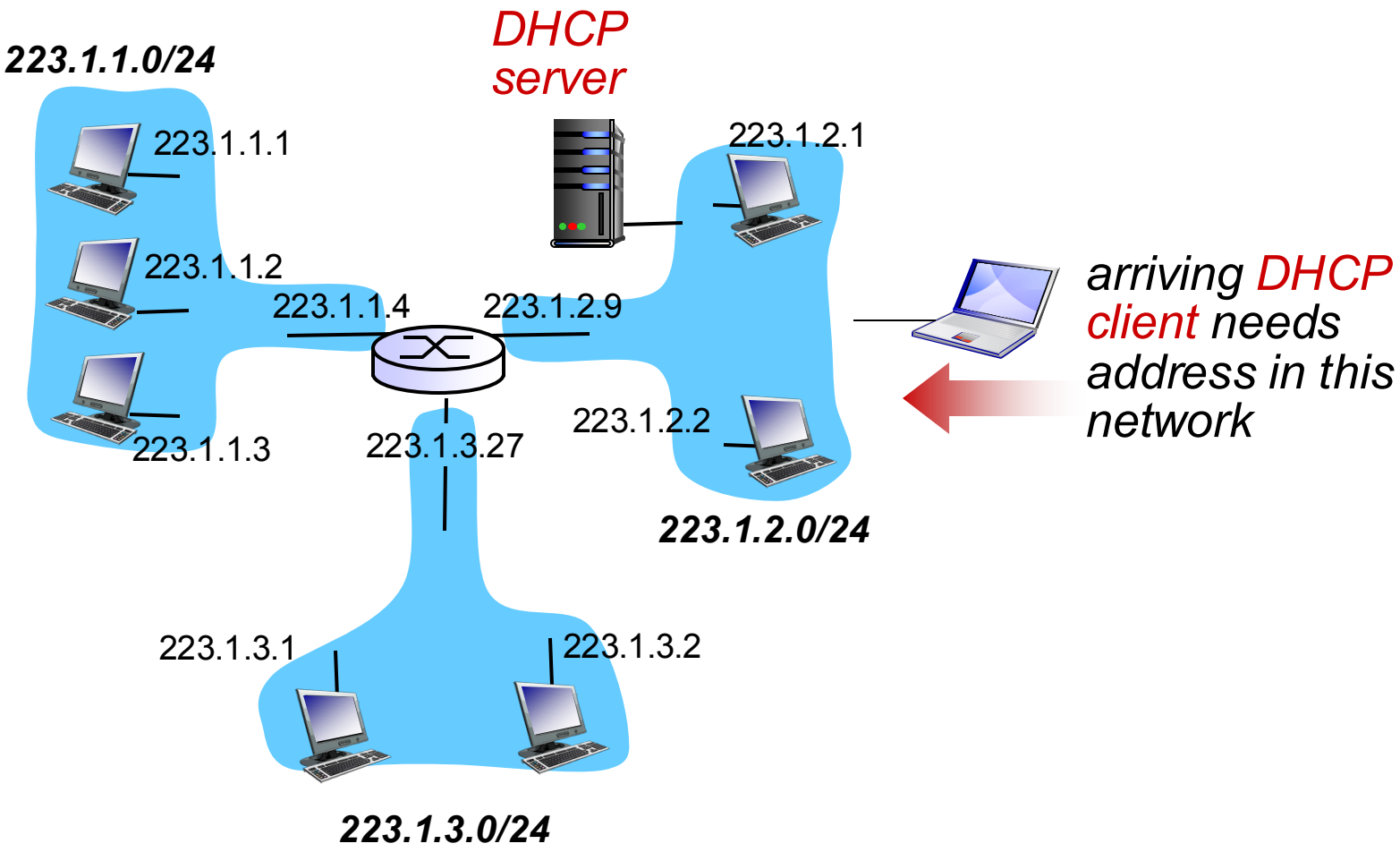
- can renew its lease on address in use
- allows reuse of addresses (only hold address while connected/"on")
- support for mobile users who want to join network (more shortly)

DHCP overview:

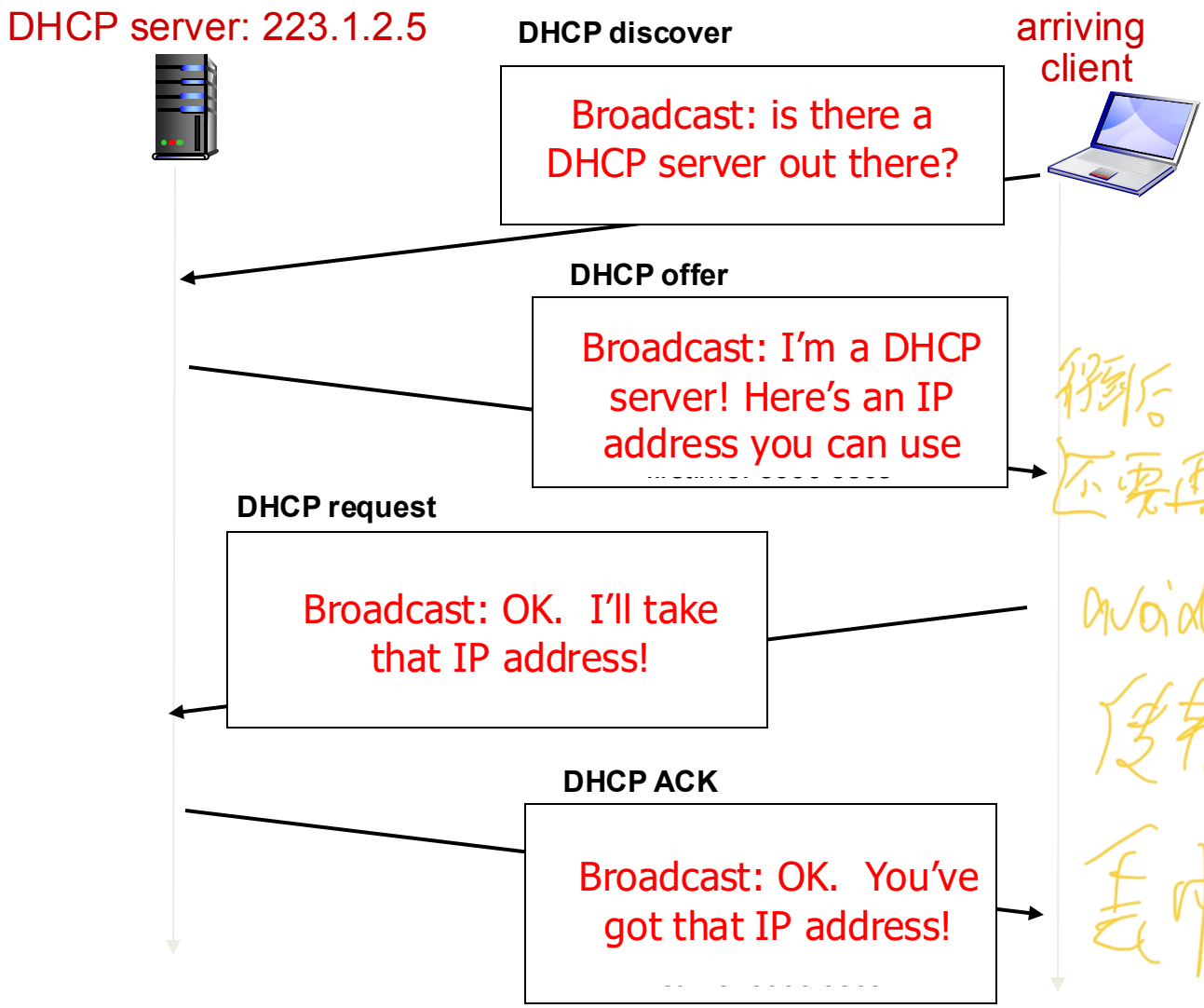
- host broadcasts "DHCP discover" msg [optional]
- DHCP server responds with "DHCP offer" msg [optional]
- host requests IP address: "DHCP request" msg
- DHCP server sends address: "DHCP ack" msg

4 steps

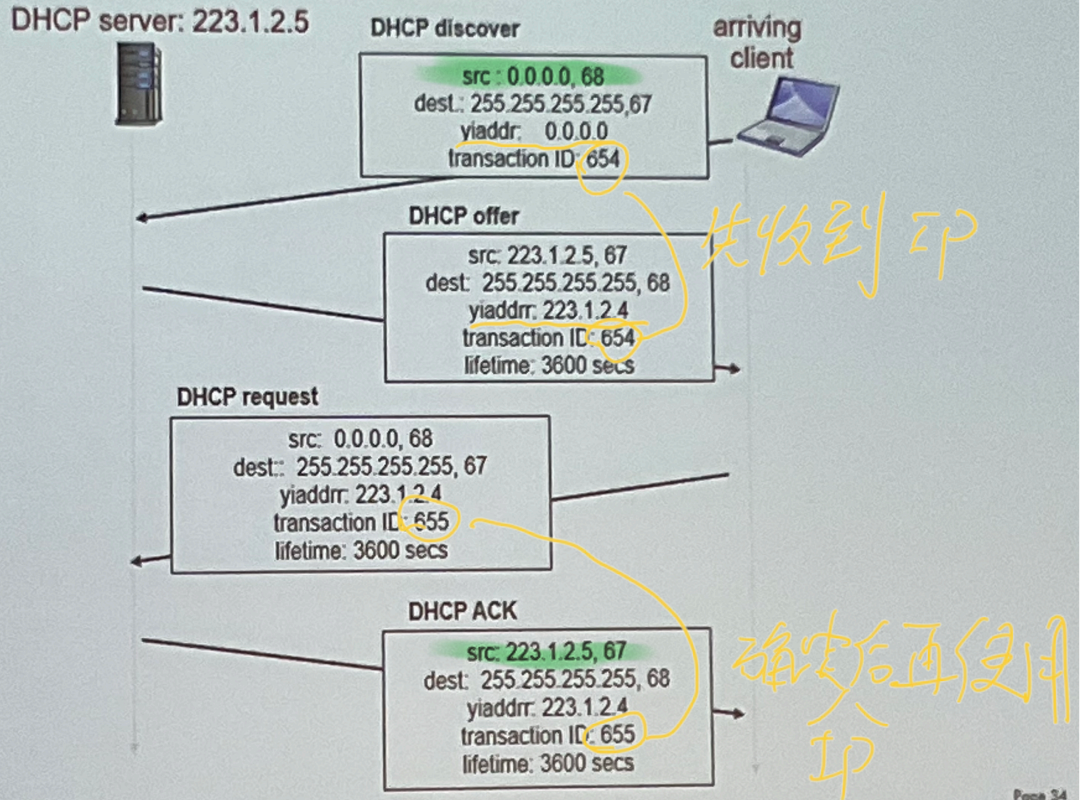
DHCP client-server scenario



DHCP client-server scenario



DHCP client-server scenario

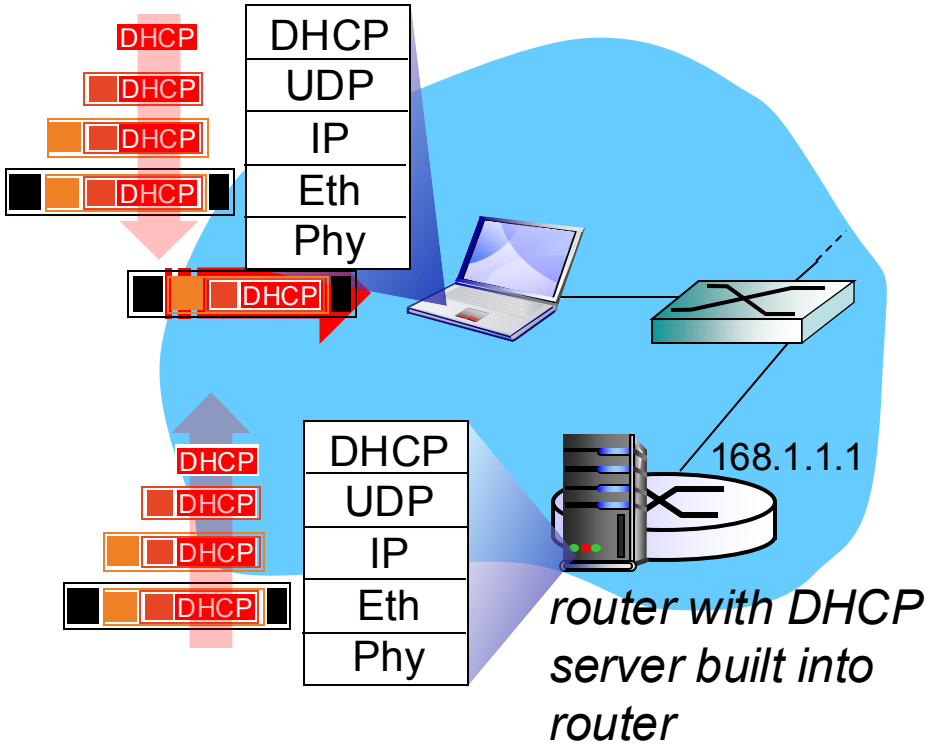


DHCP: more than IP addresses

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

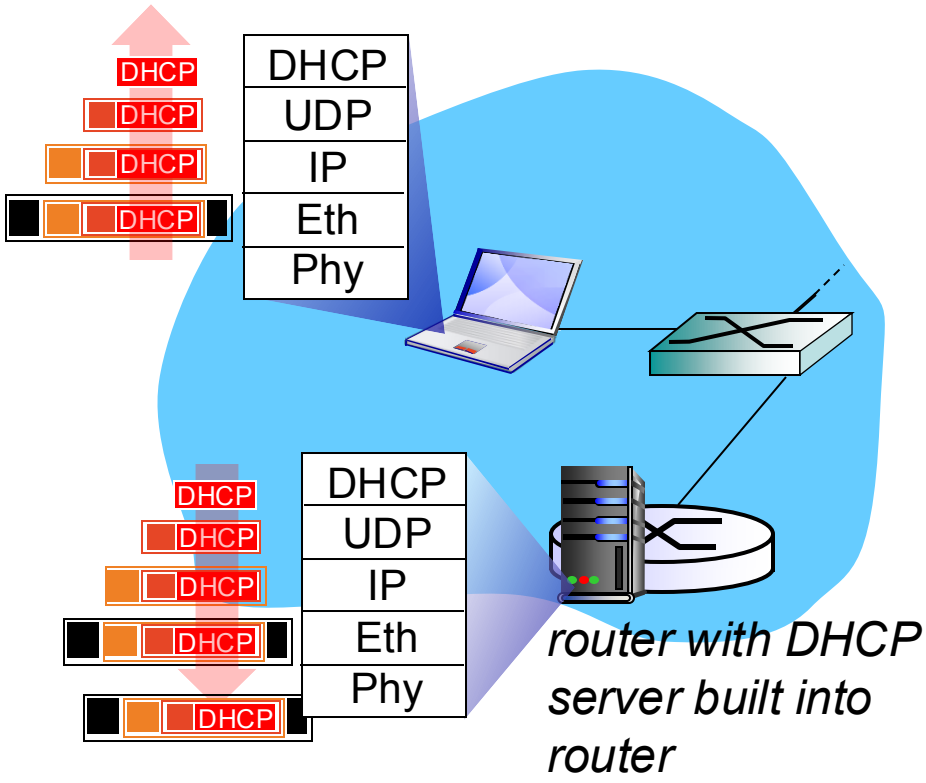
- address of first-hop router for client
- name and IP address of DNS sever
- network mask (indicating network versus host portion of address)

DHCP: example



- connecting laptop needs its IP address, addr of first-hop router, addr of DNS server: use DHCP
- DHCP request encapsulated in UDP, encapsulated in IP, encapsulated in Ethernet frame
- Ethernet frame broadcast (dest: FFFFFFFFFFFF) on LAN, received at router running DHCP server
- Ethernet demuxed to IP demuxed, UDP demuxed to DHCP

DHCP: example



- 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

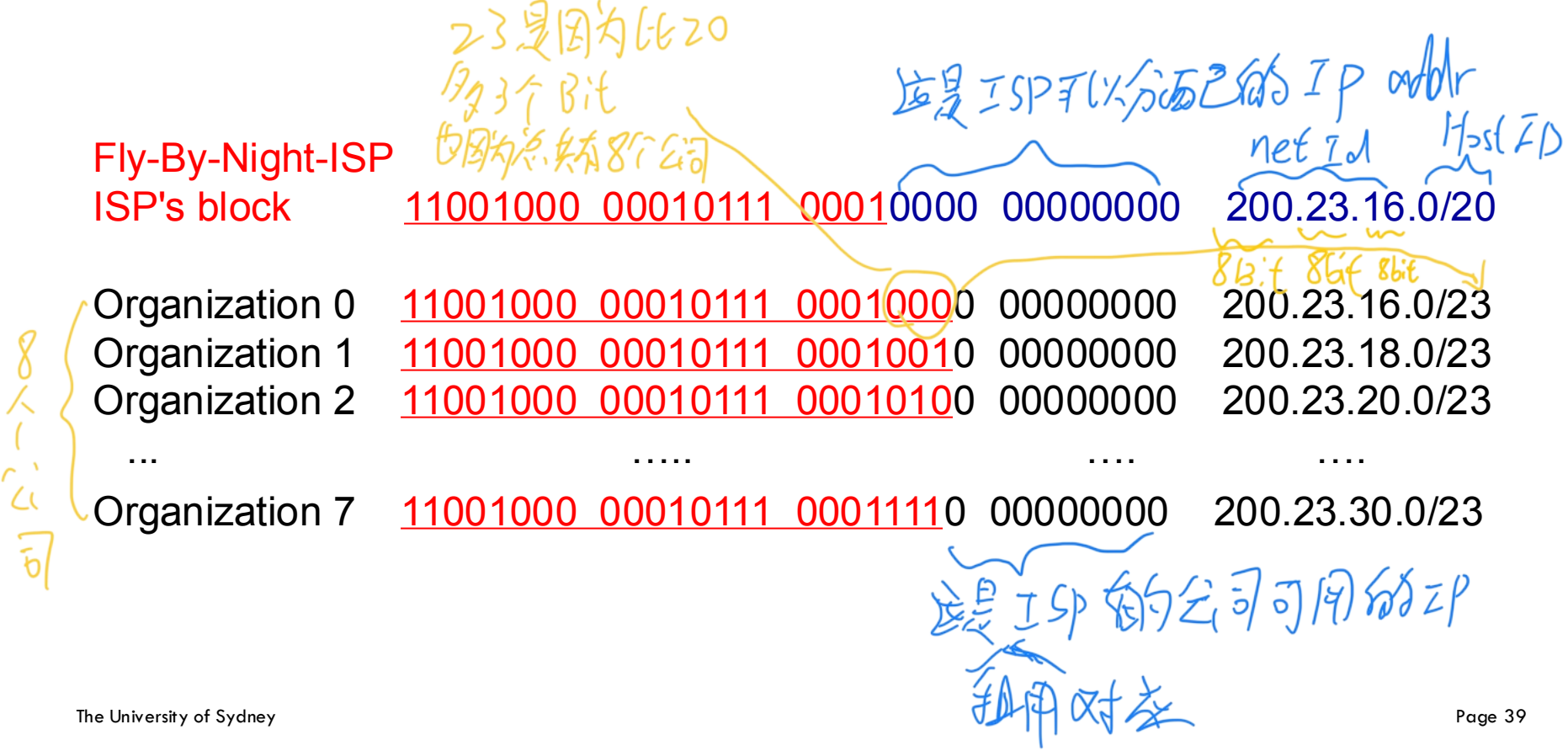
DHCP: One more comment

- Textbook shows that DHCP offer and DHCP ACK use "broadcast addresses"
- In reality, the client can choose either "unicast" mode or "broadcast" mode in the "BROADCAST flag" bit.
  - If broadcast.
    - DHCP offer. destination IP: 255.255.255.255; destination MAC: FFFFFFFFFFFF
    - DHCP ACK. destination IP: 255.255.255.255; destination MAC: FFFFFFFFFFFF
  - If unicast. X not include in this course, but it is True
    - DHCP offer. destination IP: Allocated IP; destination MAC: client's MAC address
    - DHCP ACK. destination IP: Allocated IP; destination MAC: client's MAC address
- DHCP discover and DHCP request should always broadcast
  - destination IP: 255.255.255.255; destination MAC: FFFFFFFFFFFF

IP addresses: how to get one?

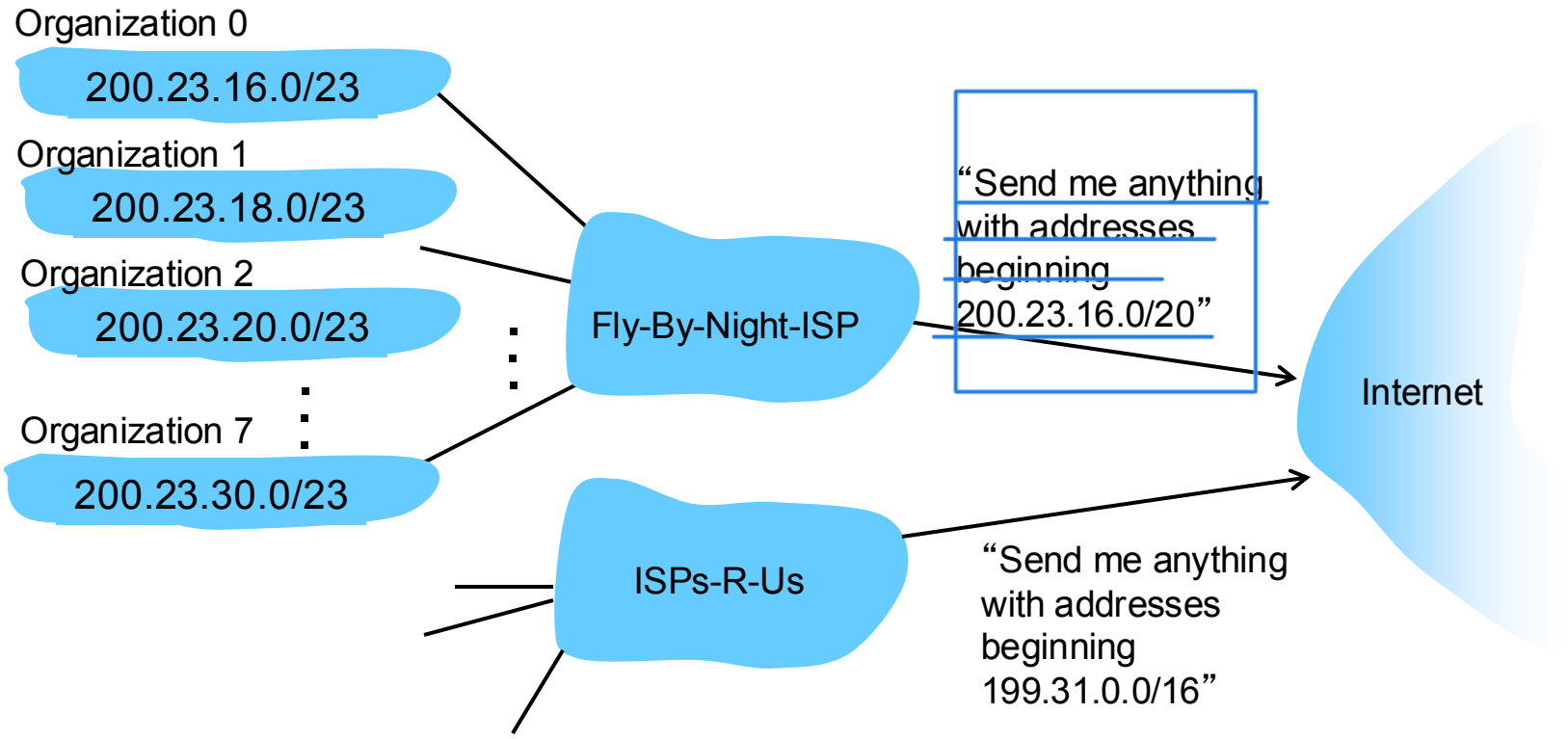
Q: how does *network* get subnet part of IP addr?

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



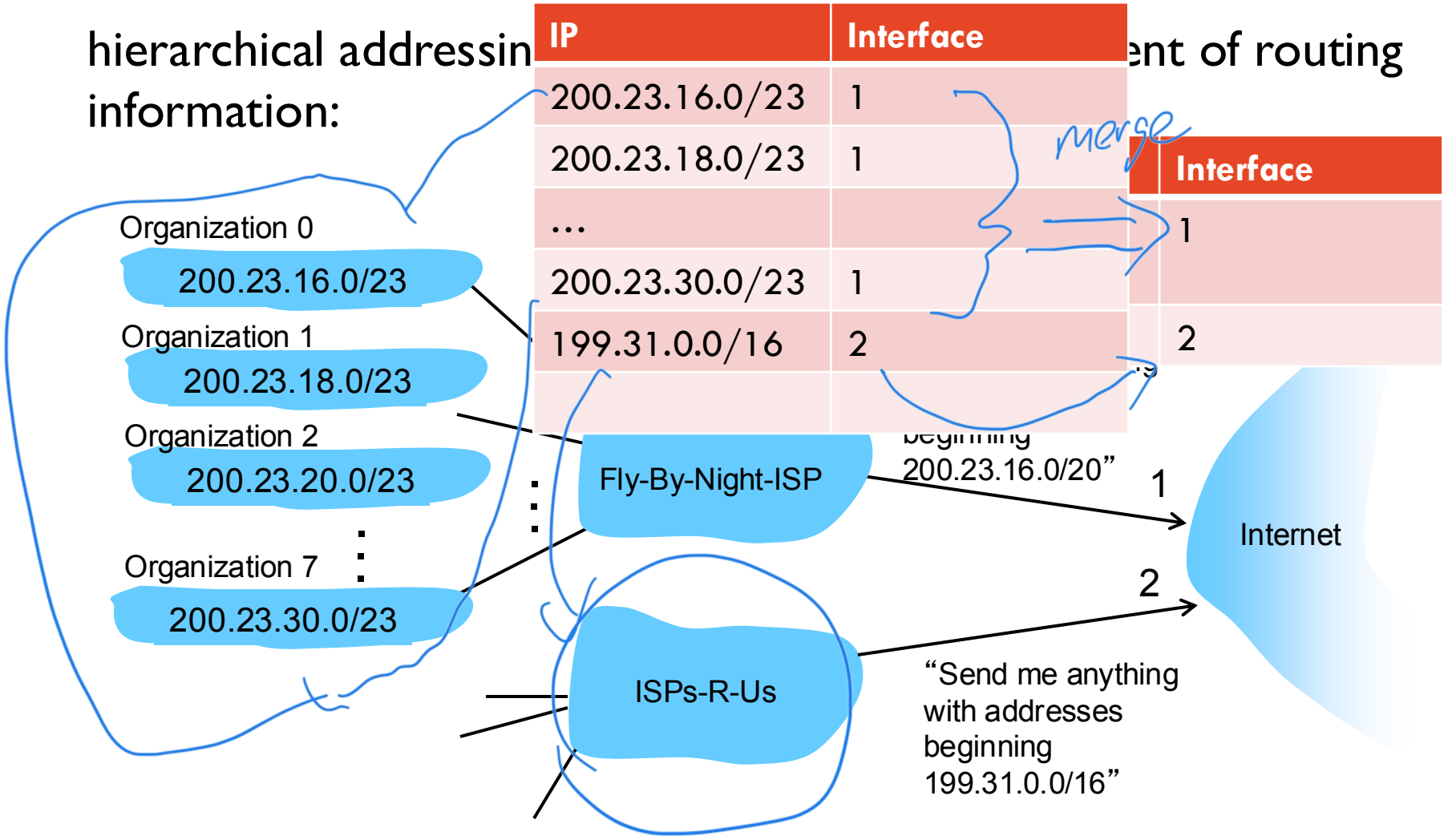
Hierarchical addressing: route aggregation

hierarchical addressing allows efficient advertisement of routing information:



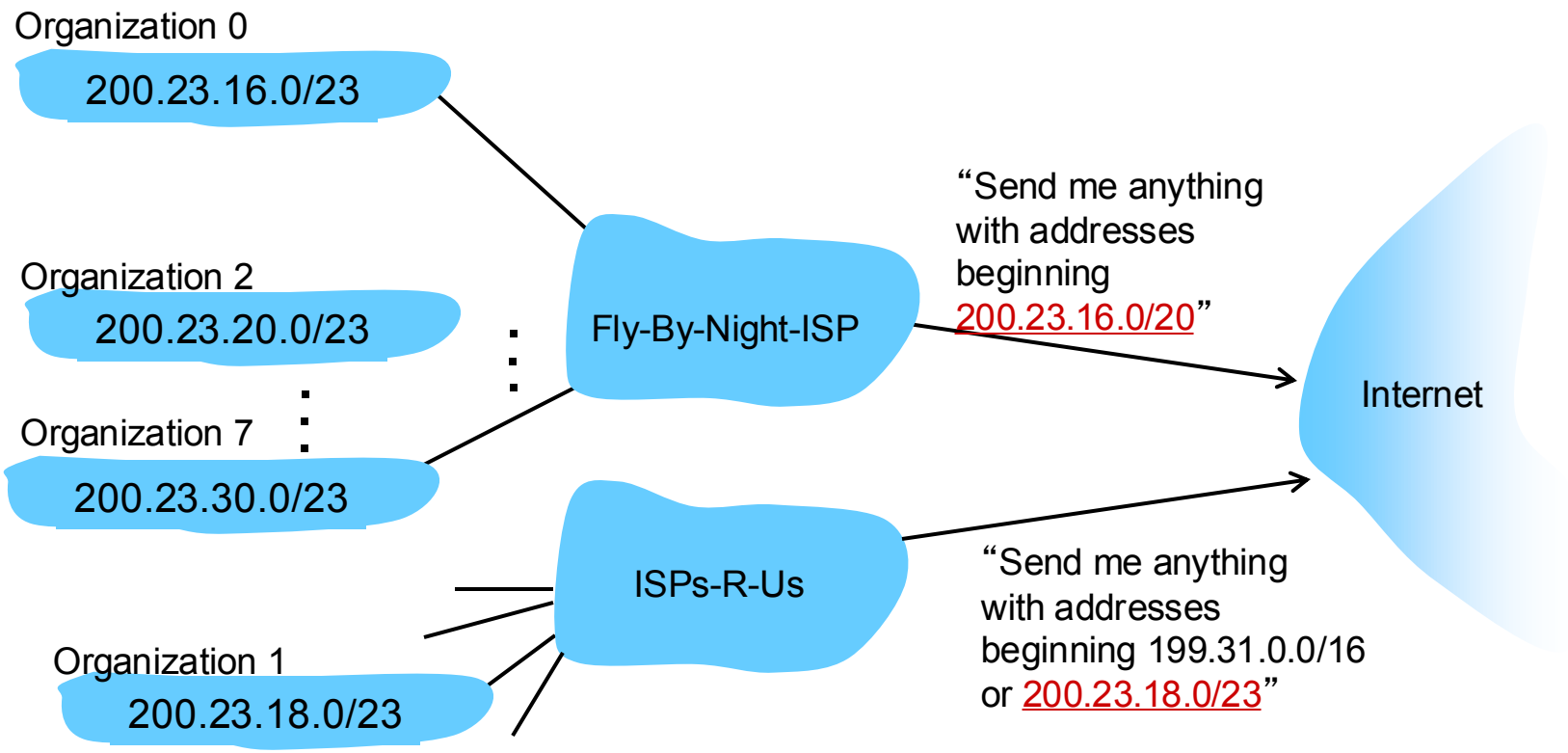
Hierarchical addressing: route aggregation

hierarchical addressin ent of routing information:



Hierarchical addressing: more specific routes

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

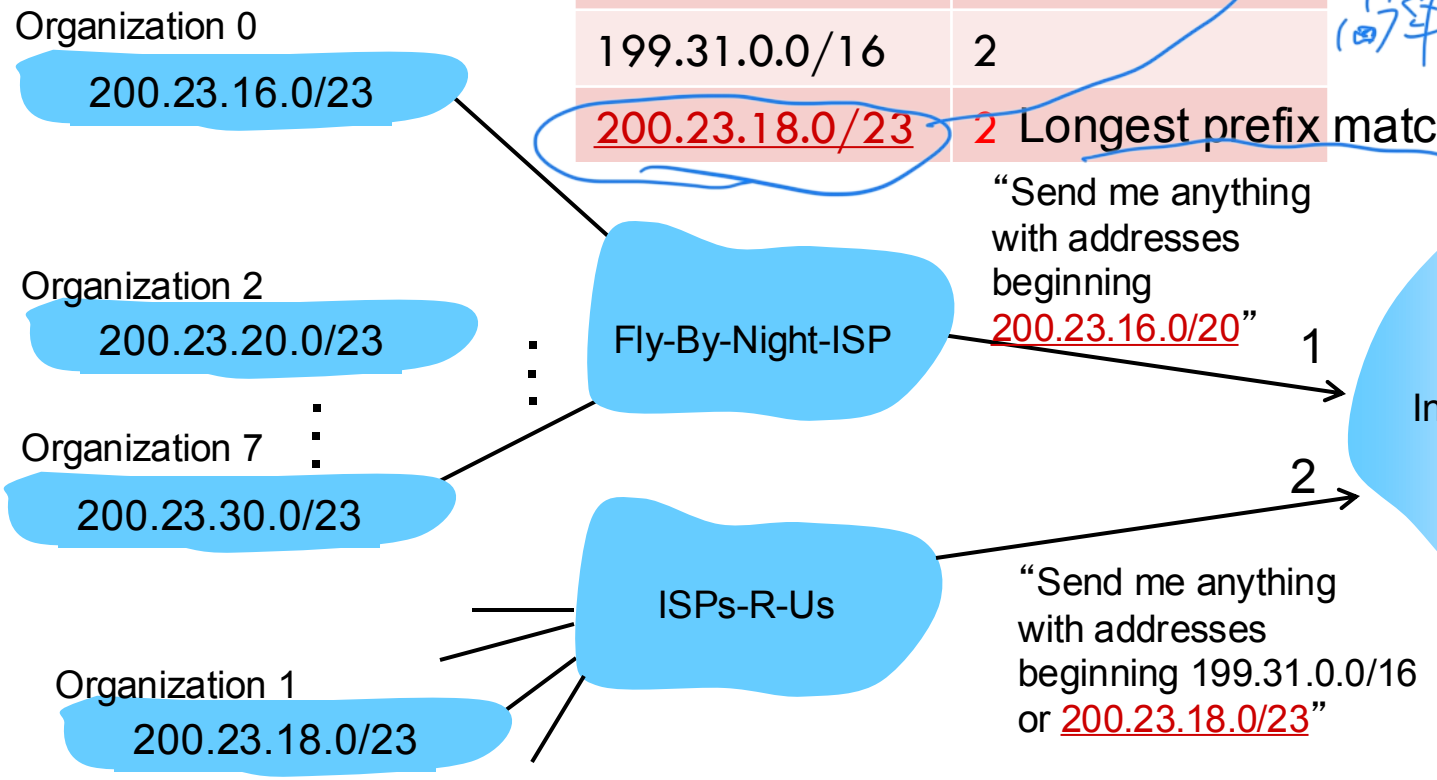


Hierarchical addressing: more specific routes

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

IP	Interface
200.23.16.0/20	1
199.31.0.0/16	2
200.23.18.0/23	2

Longest prefix match 200.23.18.1



IP addressing: the last word...

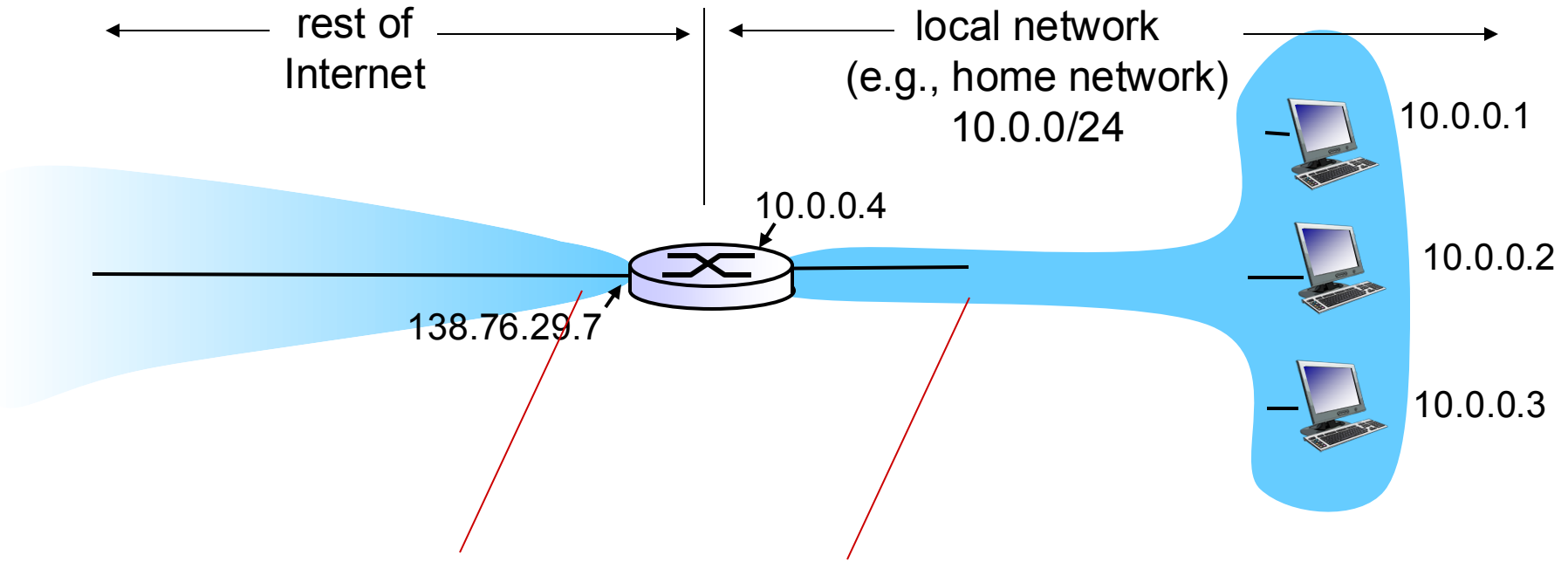
Q: how does an ISP get block of addresses?

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

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

Control by ICANN

NAT: network address translation



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

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

NAT: network address translation

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

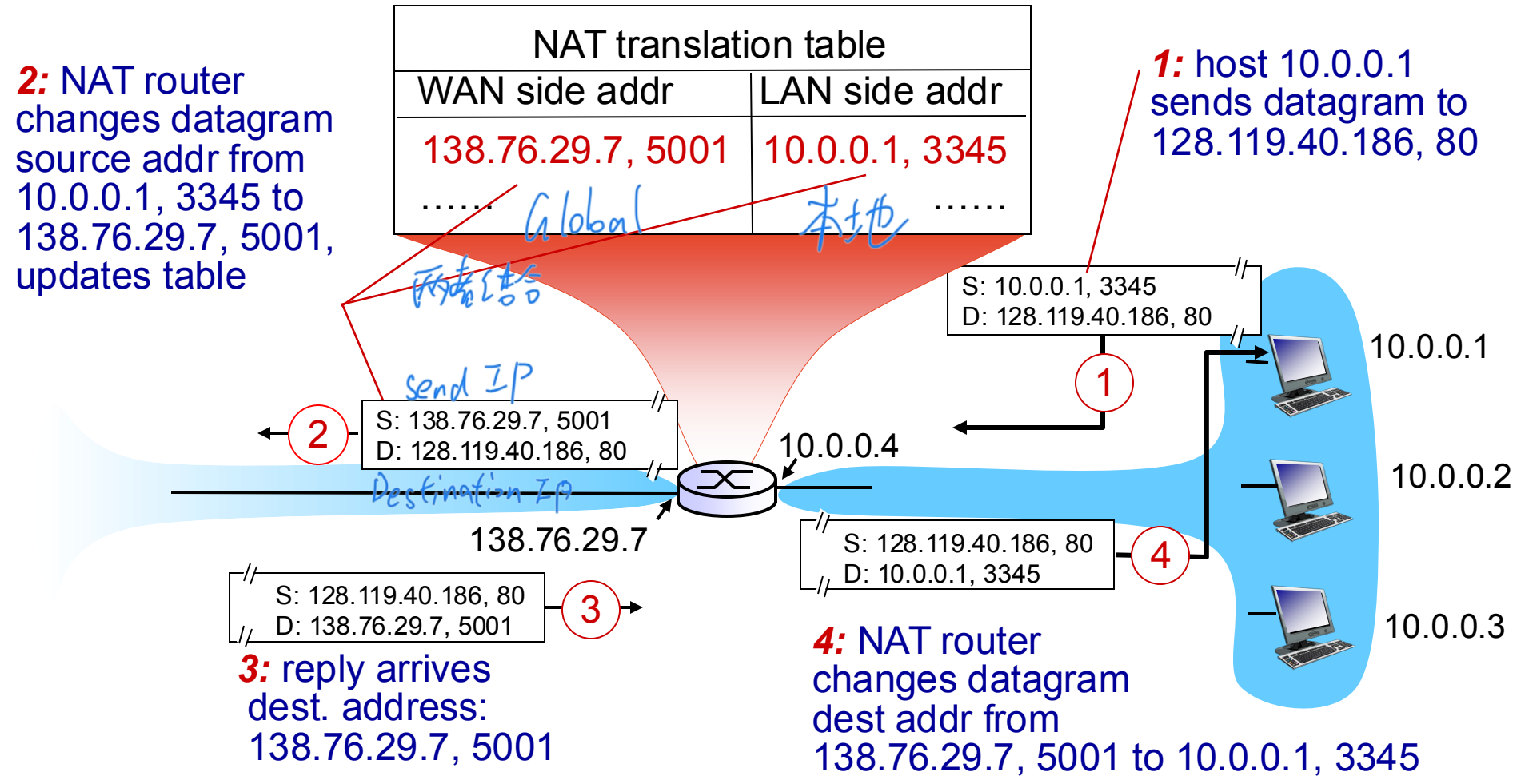
- range of addresses not needed from ISP: just one IP address for all devices
- can change addresses of devices in local network without notifying outside world
- can change ISP without changing addresses of devices in local network
- devices inside local net not explicitly addressable, visible by outside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

NAT: network address translation

implementation: NAT router must:

- *outgoing datagrams: replace* (source IP address, port #) of every outgoing datagram to (NAT IP address, new port #)
- *remember (in NAT translation table)* every (source IP address, port #) to (NAT IP address, new port #) translation pair
- *incoming datagrams: replace* (NAT IP address, new port #) in dest fields of every incoming datagram with corresponding (source IP address, port #) stored in NAT table

NAT: network address translation



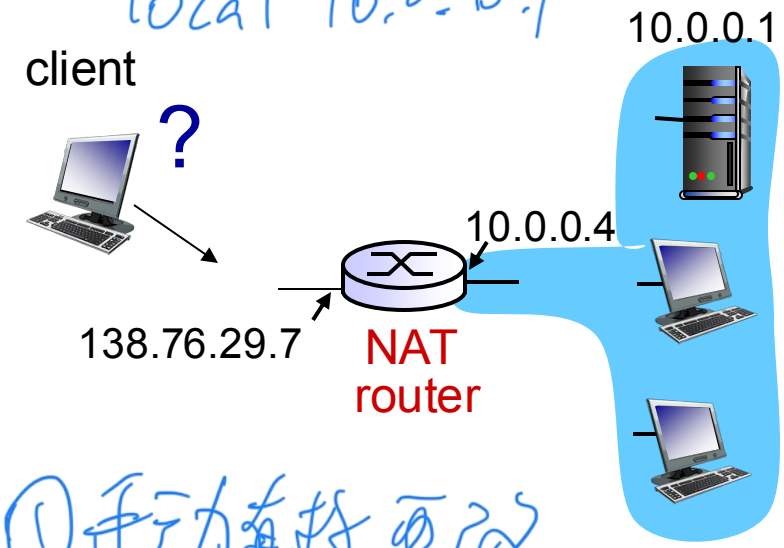
NAT: network address translation

违反了 layer 之间互相独立的原则

- 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

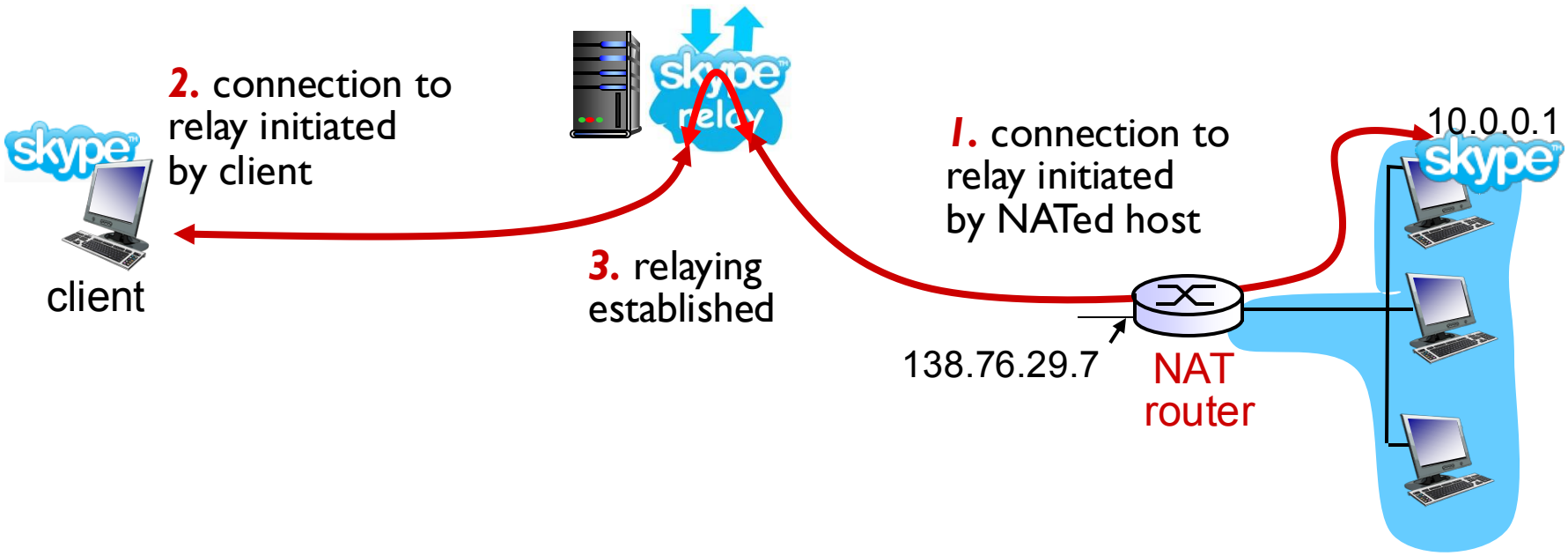
NAT traversal problem

- 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 80) 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



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)
8	0	echo request (ping)
9	0	route advertisement
10	0	router discovery
11	0	TTL expired
12	0	bad IP header

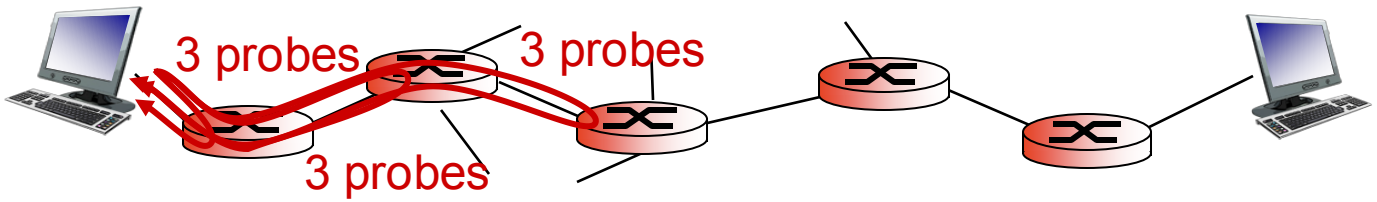
Traceroute and ICMP

- source sends series of UDP segments to dest
  - first set has TTL =1 time to leave
  - 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:

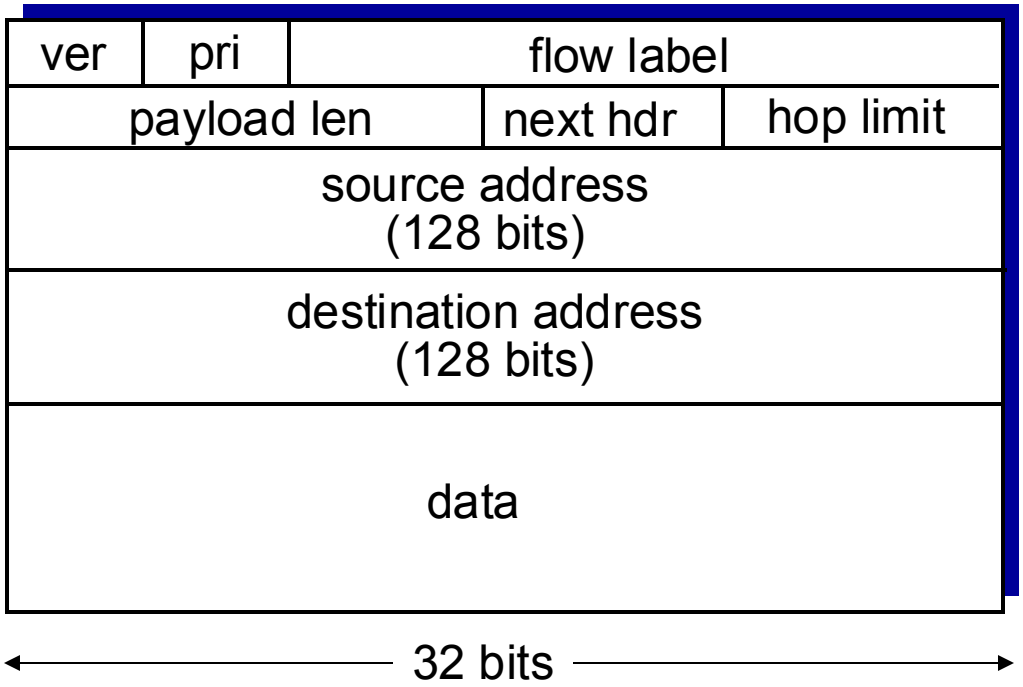
- fixed-length 40 byte header
- no fragmentation allowed

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



IPV4 is 20 bits?

IPv6 address format

Example: *2001:0db8:85a3:0000:0000:8a2e:0370:7334*

Format: 128 bits, 16bit/field (4 hexadecimal digits), 8 fields, separated by ":", lower-case preferred

Compression:

Leading zeros in each 16-bit field are suppressed

2001:db8:85a3:0:0:8a2e:370:7334

Longest consecutive all-zero fields (at least 2 all-zero fields) are replaced by "::"

2001:db8:85a3::8a2e:370:7334

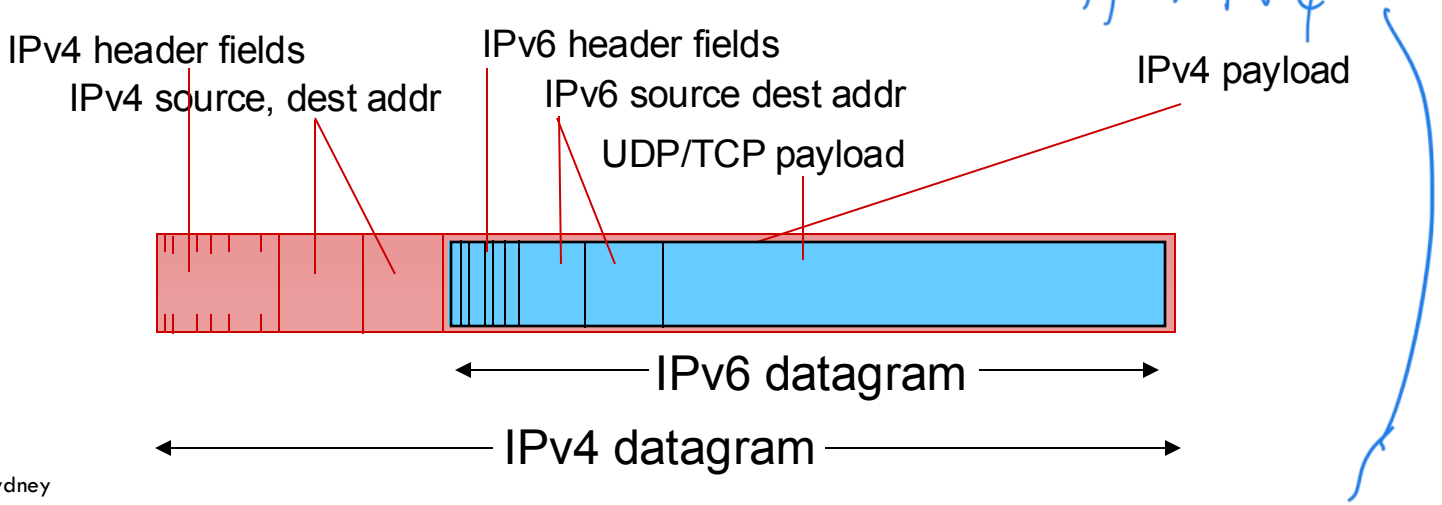
Other changes from IPv4

- *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"

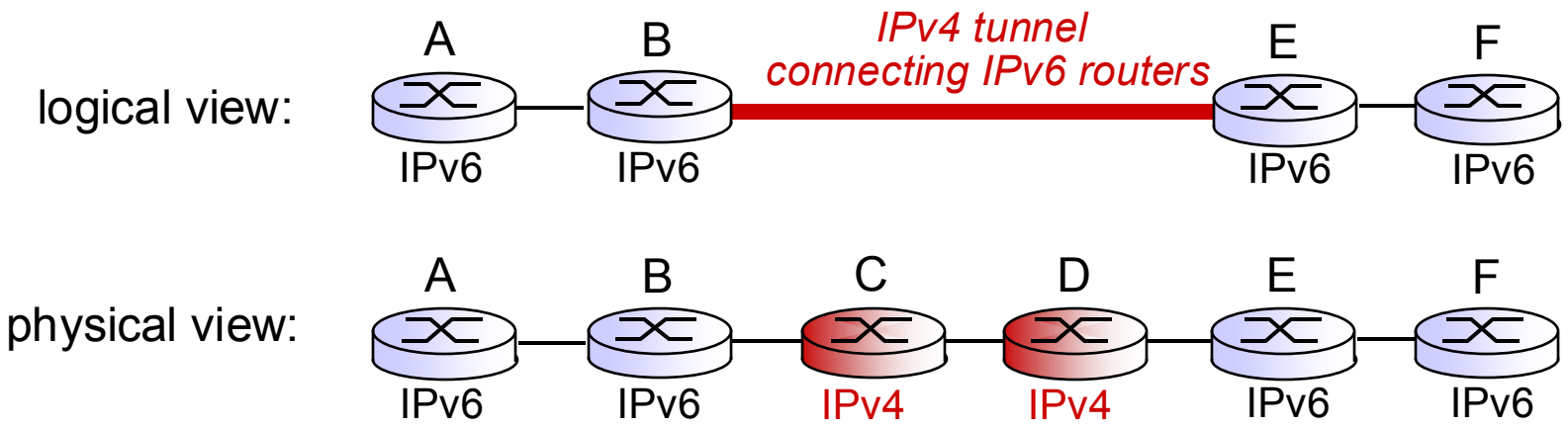
Transition from IPv4 to IPv6

- 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

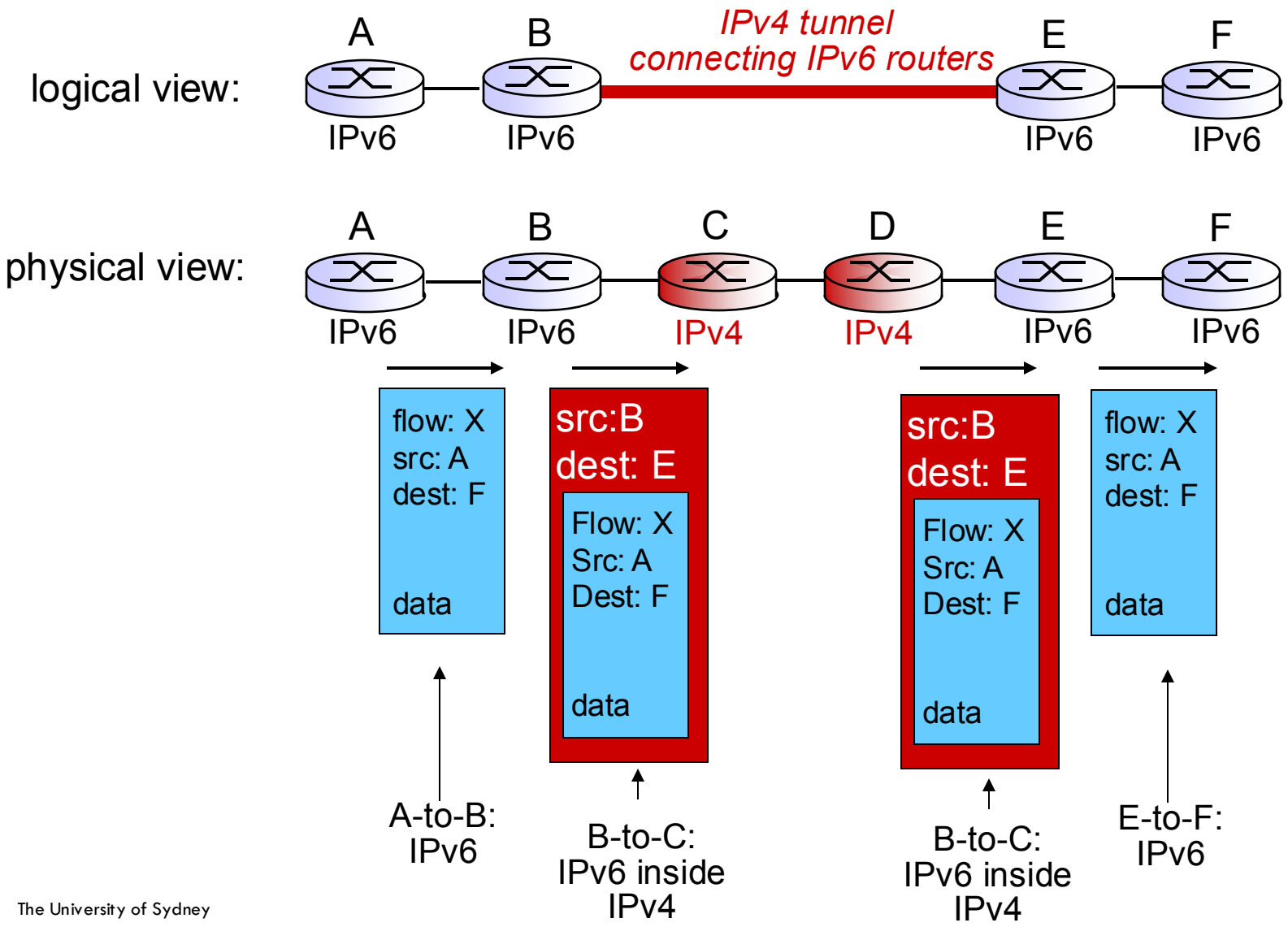
有些Router不认识IPv6的选项
加IPv4



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