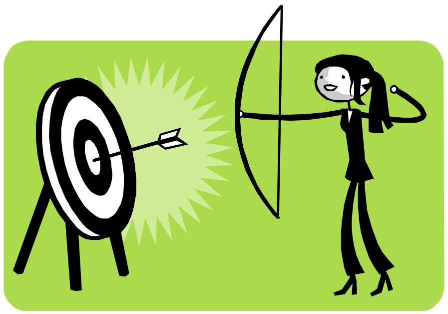
Chapter 5: Network Layer

Chapter goals:

- understand principles behind network layer services:
  - Network layer service models
  - Forwarding versus routing
  - how a router works
  - routing (path selection)
  - dealing with scale
- instantiation and implementation in the Internet



Chapter 5: Network Layer

- 5.1 Introduction
- 5.2 Virtual circuit and datagram networks
- 5.3 What's inside a router?
- 5.4 Routing algorithms:
  - Dijkstra's algorithm
  - Broadcast routing
  - Link state
  - Distance vector
  - Hierarchical routing
- 5.5 Routing in the Internet
- 5.6 IP: Internet protocol
  - IPv4 Datagram format
  - IPv4 addressing
  - IP fragment
  - NAT
  - ARP
  - ICMP
  - IPv6

Keypoints and Difficulties

Keypoints:

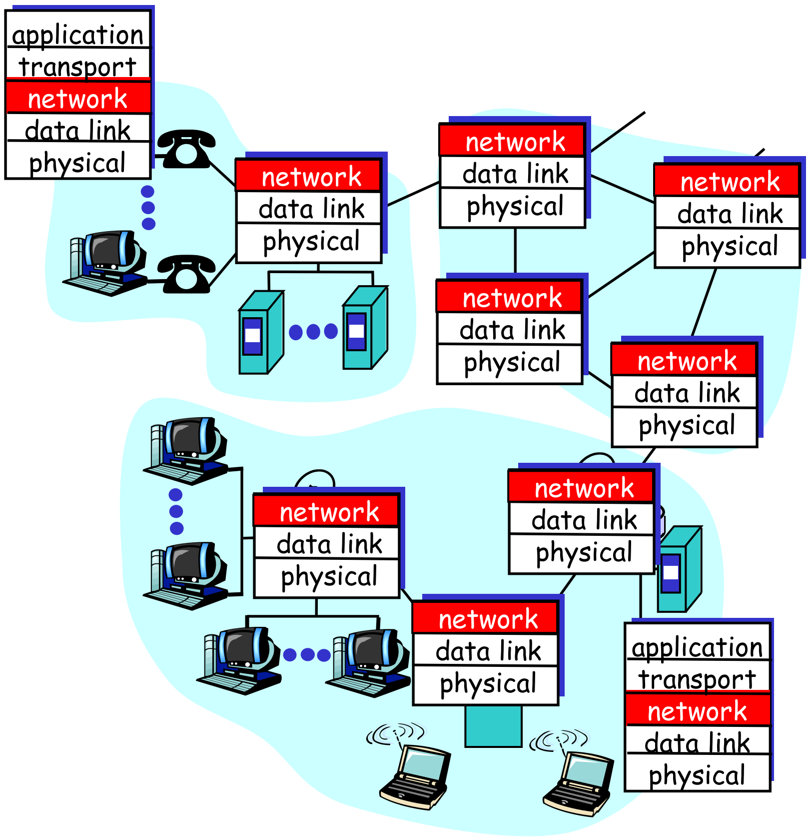
- Link state routing algorithm
- Distance vector routing algorithm
- IPv4
- IPv6

Difficulties:

- Virtual circuit and datagram networks
- Subnet and Subnet Division
- IP fragment

Network layer

- Transport packet from sending to receiving hosts
- Network layer protocols in *every* host, router
- Router examines header fields in all IP datagram passing through it



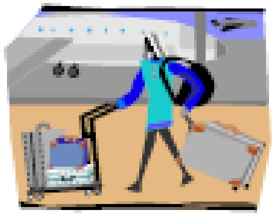
Two key Network-layer functions

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

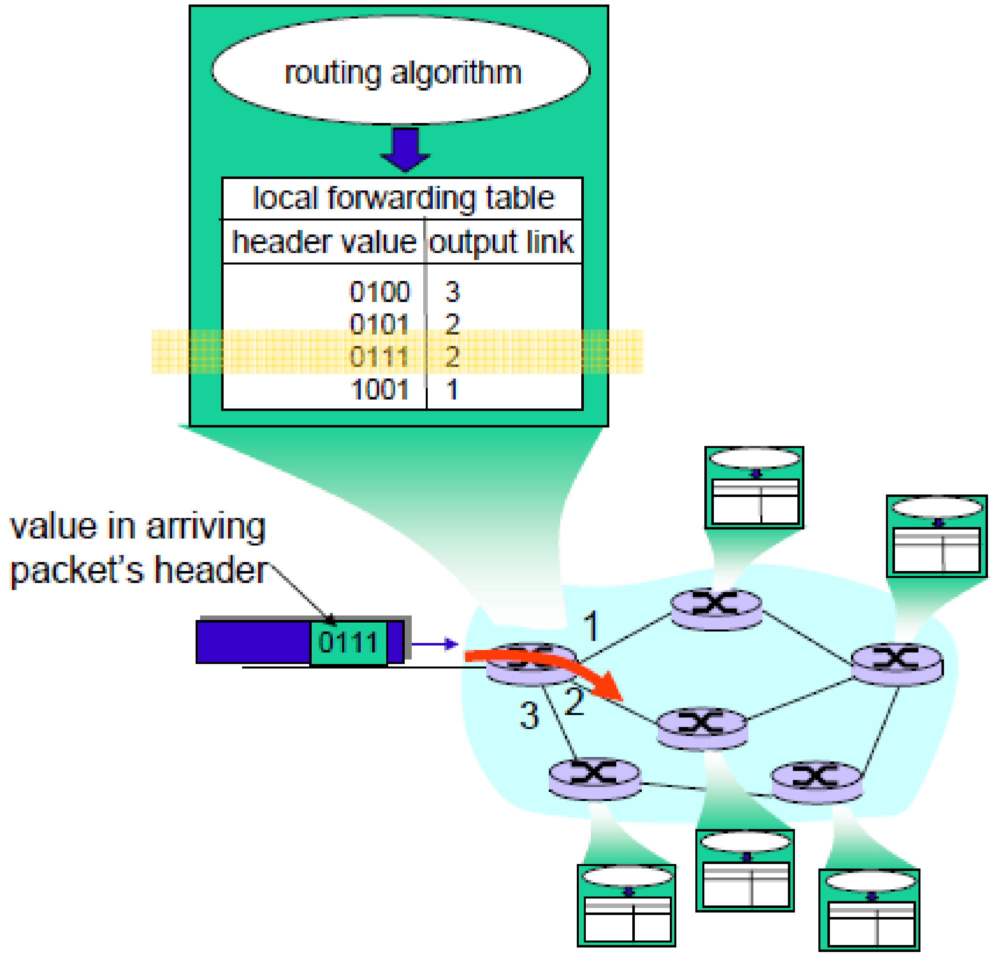
Analogy:

- *Routing:* process of planning trip from source to dest.
- *Forwarding:* process of getting through single interchange

Connection setup: some network architectures require router connection setup along path before data flows.



Interplay between routing and forwarding



Network service model

Q: What *service model* for "channel" transporting packets from sender to receiver?

service abstraction

- guaranteed bandwidth?
- preservation of inter-packet timing (no jitter)?
- loss-free delivery?
- in-order delivery?
- congestion feedback to sender?

The most important abstraction provided by network layer:

virtual circuit
or
datagram?

Network layer connection and connection-less service

- Datagram network provides network-layer connectionless service
- VC network provides network-layer connection service
- Analogous to the transport-layer services,but:
  - Service: host-to-host
  - No choice: network provides one to the other
  - Implementation: in network core

Network layer service models:

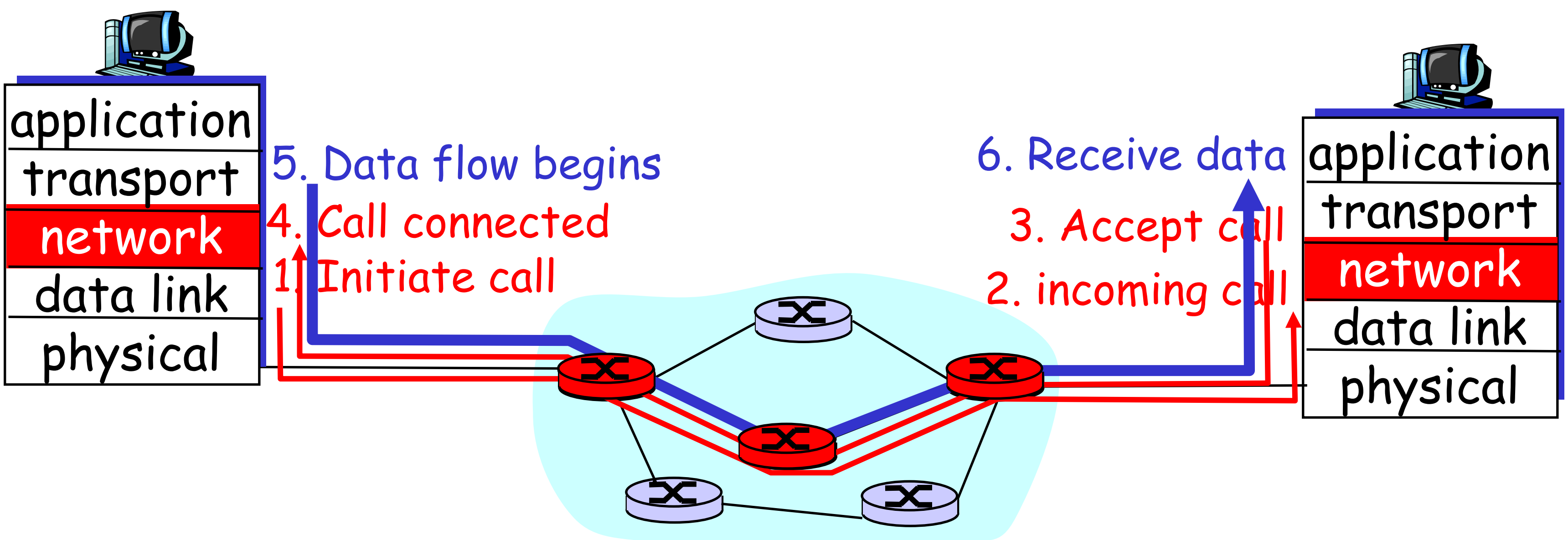
Network Architecture	Service Model	Guarantees ? Bandwidth	Loss	Order	Timing	Congestion feedback
Internet	best effort	none	no	no	no	no (inferred via loss)
ATM	CBR	constant rate	yes	yes	yes	no congestion
ATM	VBR	guaranteed rate	yes	yes	yes	no congestion
ATM	ABR	guaranteed minimum	no	yes	no	yes
ATM	UBR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

Chapter 5: Network Layer

- 5.1 Introduction
- 5.2 Virtual circuit and datagram networks
- 5.3 What's inside a router?
- 5.4 Routing algorithms:
  - Dijkstra's algorithm
  - Broadcast routing
  - Link state
  - Distance vector
  - Hierarchical routing
- 5.5 Routing in the Internet
- 5.6 IP: Internet protocol
  - IPv4 Datagram format
  - IPv4 addressing
  - IP fragment
  - NAT
  - ARP
  - ICMP
  - IPv6

Virtual circuits: signaling protocols

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



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 OD)
- *every* router on source-dest path s maintain "state" for each passing connection
  - transport-layer connection only involved two end systems
- link, router resources (bandwidth, buffers) may be *allocated* to VC
  - to get circuit-like perf.

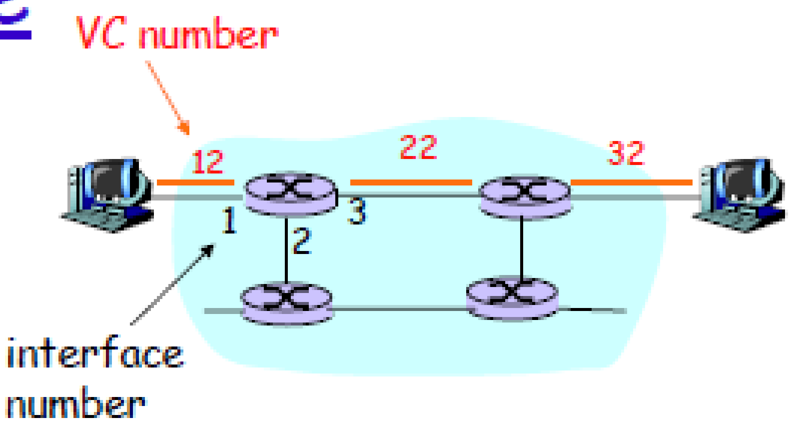
VC Implementation

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

Forwarding Table



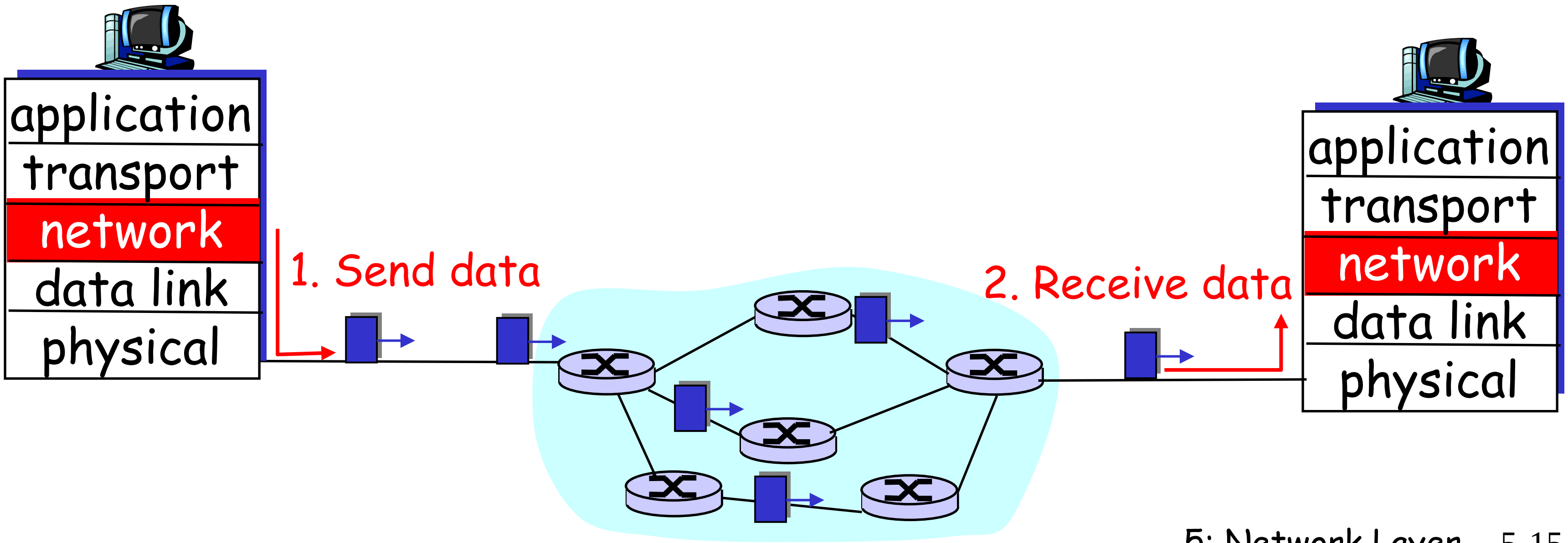
Forwarding table in northwest router:

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

Routers maintain connection state information!

Datagram networks: the Internet model

- no call setup at network layer
- routers: no state about end-to-end connections
  - no network-level concept of "connection"
- packets typically routed using destination host ID
  - packets between same source-dest pair may take different paths



Forwarding Table

4 billion possible entries

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

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?

Datagram or VC network: why?

Internet

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

ATM

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

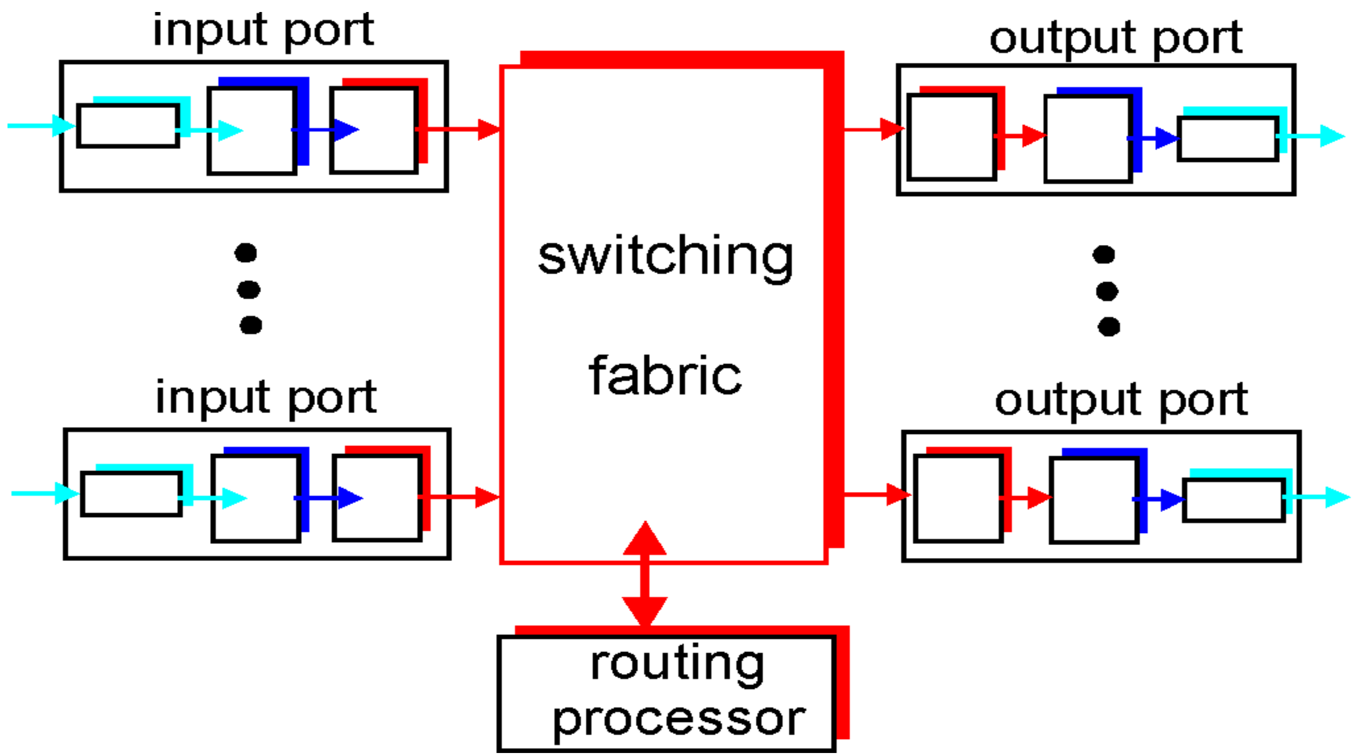
Chapter 5: Network Layer

- 5.1 Introduction
- 5.2 Virtual circuit and datagram networks
- 5.3 What's inside a router?
- 5.4 Routing algorithms:
  - Dijkstra's algorithm
  - Broadcast routing
  - Link state
  - Distance vector
  - Hierarchical routing
- 5.5 Routing in the Internet
- 5.6 IP: Internet protocol
  - IPv4 Datagram format
  - IPv4 addressing
  - IP fragment
  - NAT
  - ARP
  - ICMP
  - IPv6

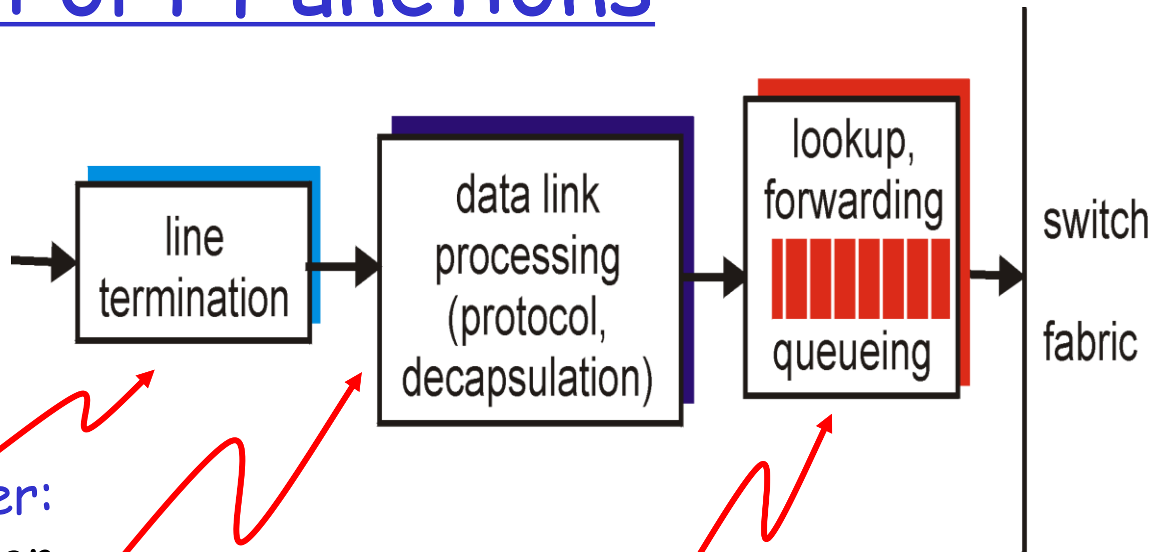
Router Architecture Overview

Two key router functions:

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



Input Port Functions



Physical layer:
bit-level reception

Data link layer:
e.g., Ethernet

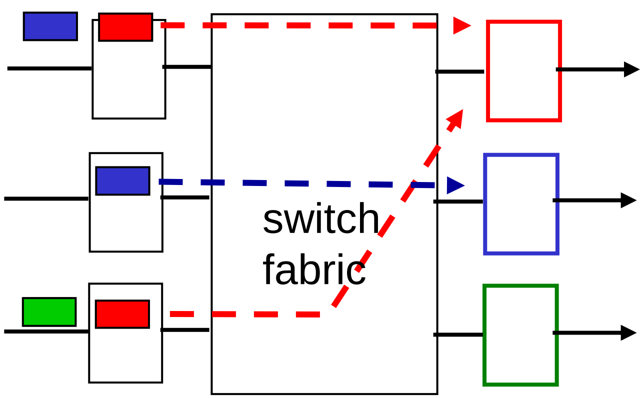
Decentralized switching:

- given datagram dest., lookup output port using routing table in input port memory
- goal: complete input port processing at 'line speed'
- queuing: if datagrams arrive faster than forwarding rate into switch fabric

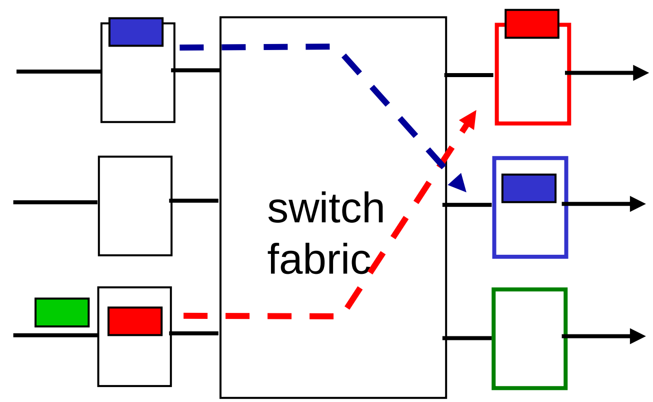
Input Port Queuing

- Fabric slower that input ports combined -> queueing may occur at input queues

  queueing delay and loss due to input buffer overflow!
- Head-of-the-Line (HOL) blocking: queued datagram at front of queue prevents others in queue from moving forward

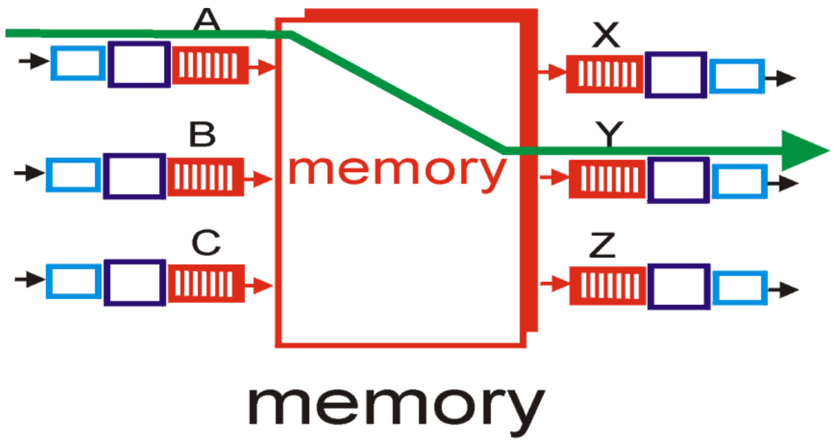


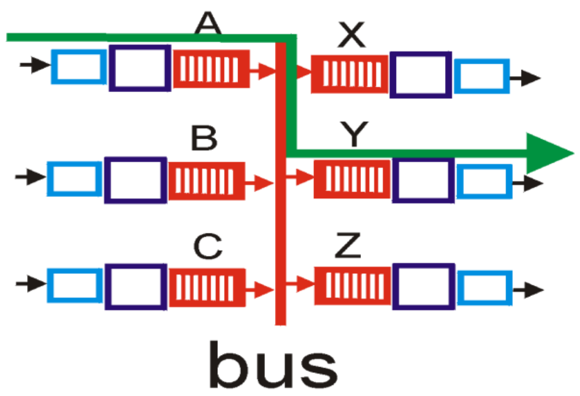
output port contention:
only one red datagram can be transferred.
lower red packet is blocked

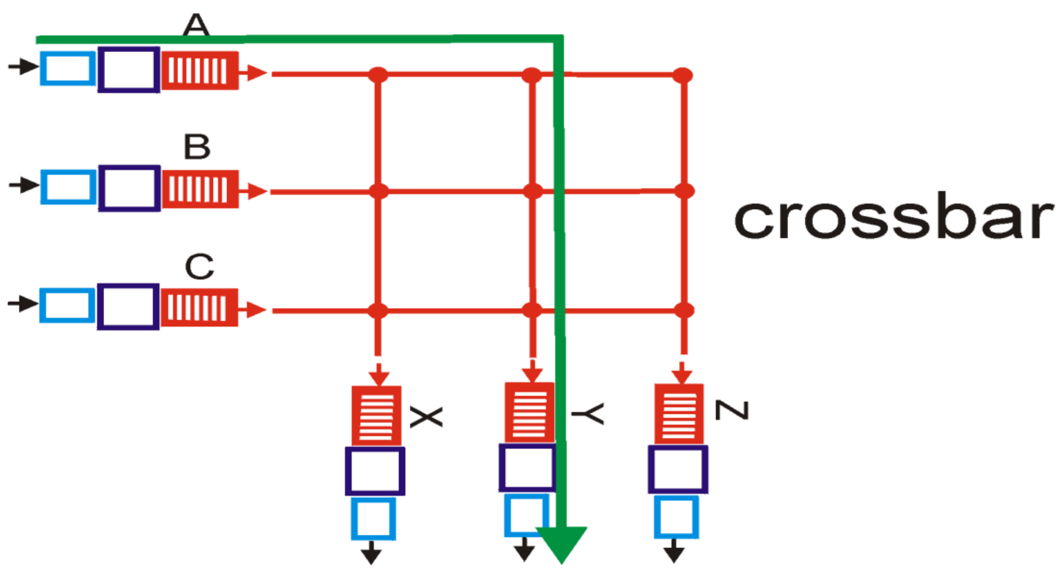


one packet time later:
green packet experiences HOL blocking

Three types of switching fabrics



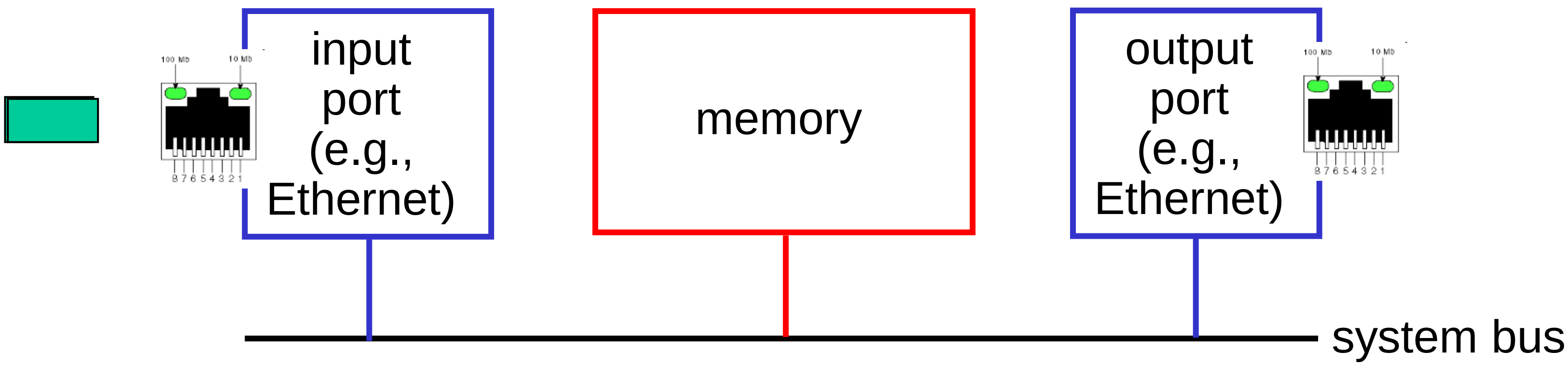




Switching Via Memory

First generation routers:

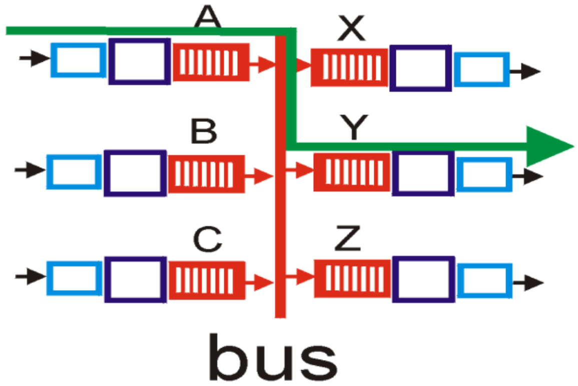
- packet copied by system's (single) CPU
- speed limited by memory bandwidth (2 bus crossings per datagram)



Modern routers:

- input port processor performs lookup, copy into memory
- Cisco Catalyst 8500

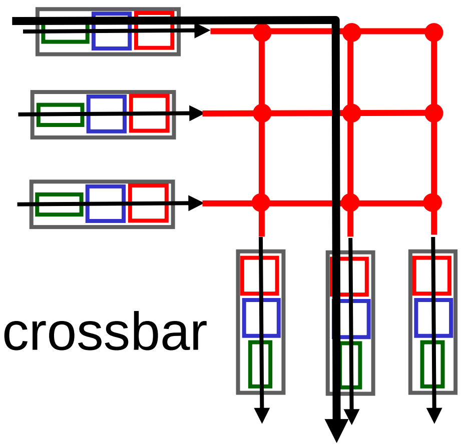
Switching Via Bus



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

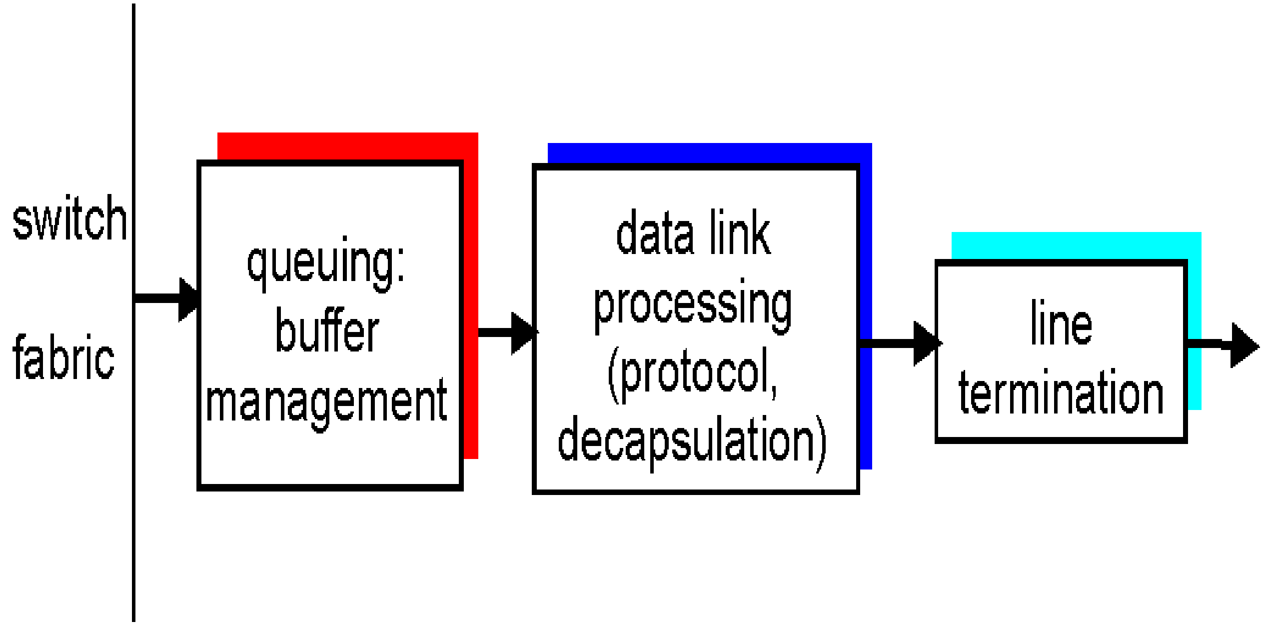
Switching Via An Interconnection Network

- overcome bus bandwidth limitations
- Banyan networks, crossbar, other interconnection nets initially developed to connect processors in multiprocessor
- Advanced design: fragmenting datagram into fixed length cells, switch cells through the fabric.
- Cisco 12000: switches 60 Gbps through the interconnection network



crossbar

Output Ports



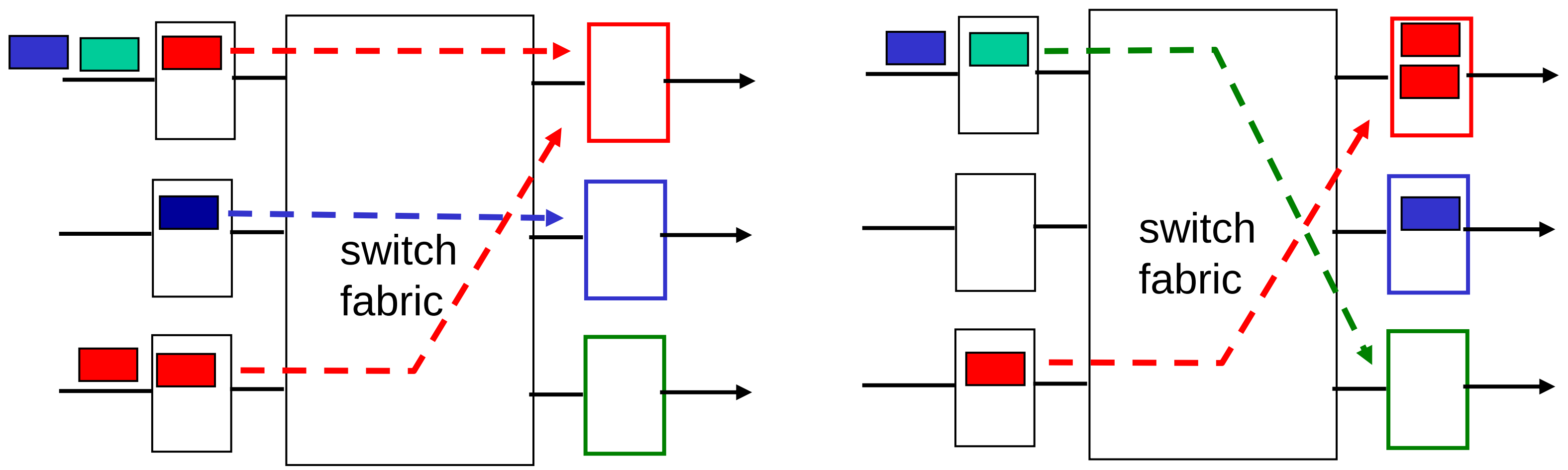
- *Buffering* required w
  fabric faster than th

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

- *Scheduling discipline* chooses among queued
  datagrams f

Priority scheduling – who gets best performance, network neutrality

Output port queueing



at *t*, packets more from input to output

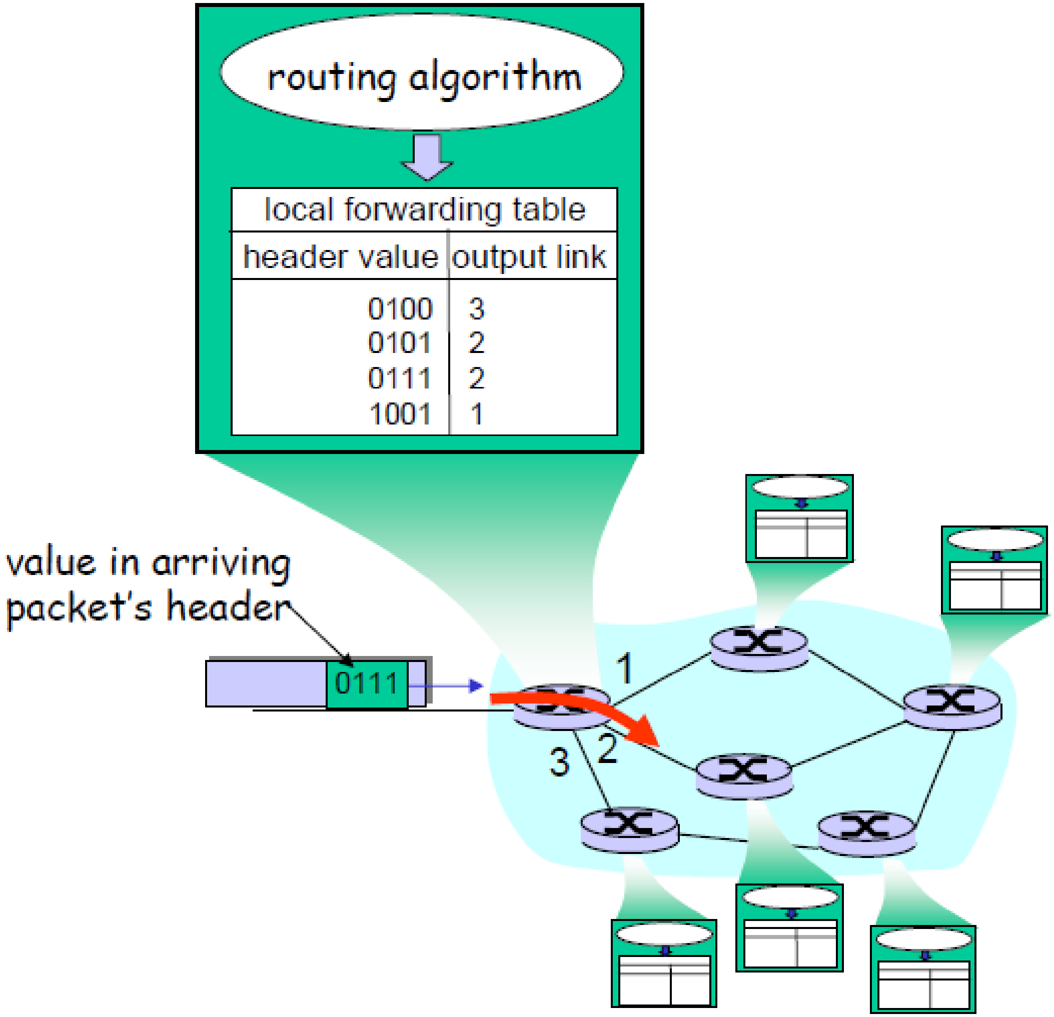
one packet time later

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

Chapter 5: Network Layer

- 5.1 Introduction
- 5.2 Virtual circuit and datagram networks
- 5.3 What's inside a router?
- 5.4 Routing algorithms:
  - Dijkstra's algorithm
  - Broadcast routing
  - Link state
  - Distance vector
  - Hierarchical routing
- 5.5 Routing in the Internet
- 5.6 IP: Internet protocol
  - IPv4 Datagram format
  - IPv4 addressing
  - IP fragment
  - NAT
  - ARP
  - ICMP
  - IPv6

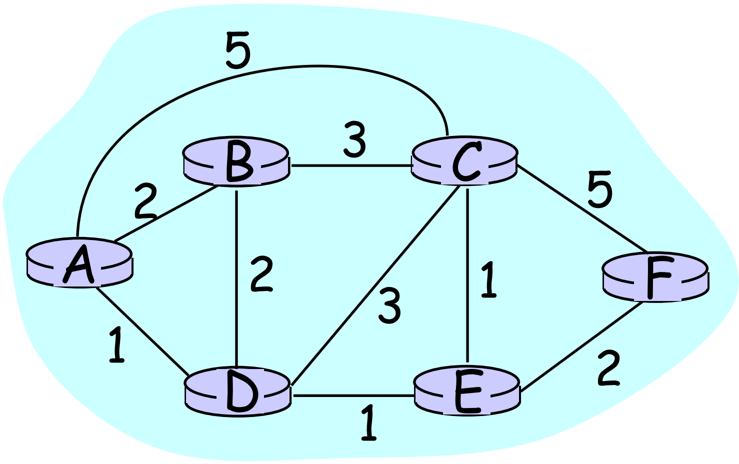
Interplay between routing and forwarding



Routing

Routing protocol

Goal: determine "good" path (sequence of routers) thru network from source to dest.

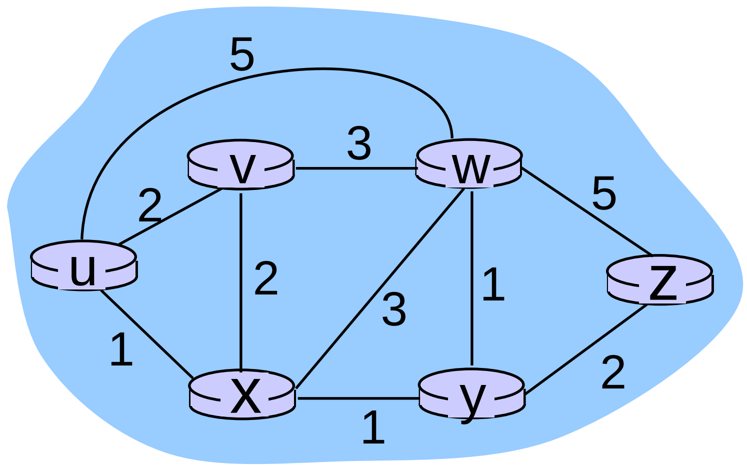


Graph abstraction for routing algorithms:

- graph nodes are routers
- graph edges are physical links
  - link cost: delay, $ cost, or congestion level

- "good" path:
  - typically means minimum cost path
  - other def's possible

Graph abstraction of the network



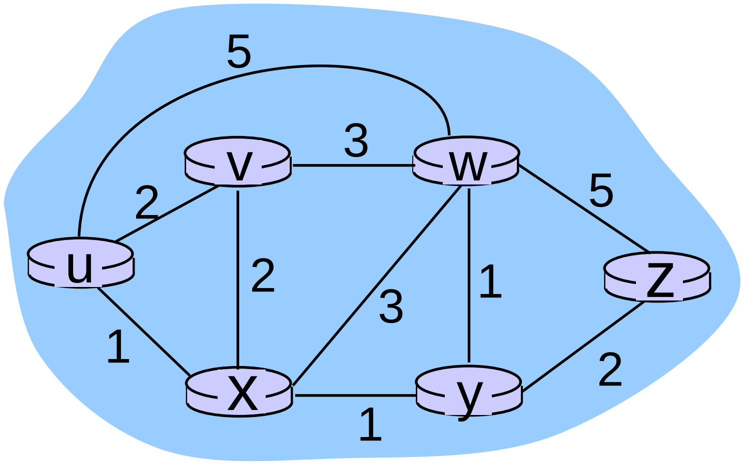
graph: G = (N,E)

N = set of routers = { u, v, w, x, y, z }

E = set of links ={ (u,v), (u,x), (v,x), (v,w), (x,w), (x,y), (w,y), (w,z), (y,z) }

aside: graph abstraction is useful in other network contexts, e.g., P2P, where *N* is set of peers and *E* is set of TCP connections

Graph abstraction: costs



c(x,x') = cost of link (x,x')
e.g., c(w,z) = 5

cost could always be 1, or inversely related to bandwidth, or inversely related to congestion

cost of path $(x_1, x_2, x_3,\ldots, x_p) = c(x_1,x_2) + c(x_2,x_3) + \ldots + c(x_{p-1},x_p)$

key question: what is the least-cost path between u and z ?
routing algorithm: algorithm that finds that least cost path

Routing Algorithm classification

Global or decentralized information?

Global:

- all routers have complete topology, link cost info
- "link state" algorithms

Decentralized:

- router knows physically-connected neighbors, link costs to neighbors
- iterative process of computation, exchange of info with neighbors
- "distance vector" algorithms

Static or dynamic?

Static:

- routes change slowly over time

Dynamic:

- routes change more quickly
  - periodic update
  - in response to link cost changes

Dijkstra's algorithm

- Net topology, link costs known to all nodes
- Computes least cost paths from one node(source) to all other nodes
- Iterative: after k iterations, know least cost path to k dest.'s

Notation:

- c(x,y): link cost from node x to y; = ∞ if not direct neighbors
- D(v): current value of cost of path from source to dest. V
- p(v): predecessor node along path from source to v
- N': set of nodes whose least cost path definitively known

Dijkstra's algorithm

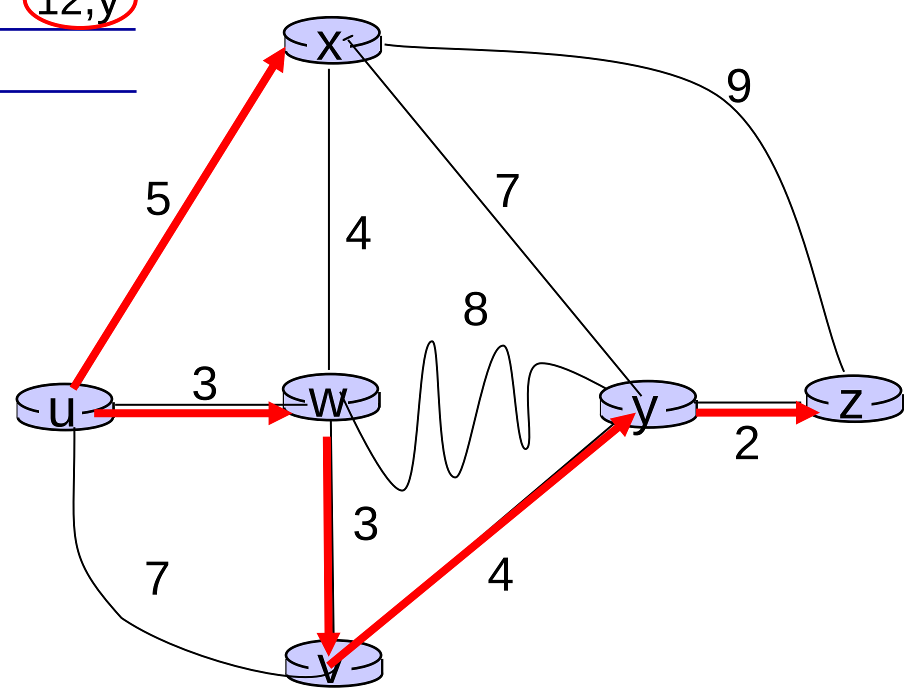
```
Initialization:
  N' = {u}
  for all nodes v
    if v adjacent to u
        then D(v) = c(u,v)
    else D(v) = ∞

Loop
  find w not in N' such that D(w) is a minimum
  add w to N'
  update D(v) for all v adjacent to w and not in N' :
      D(v) = min( D(v), D(w) + c(w,v) )
  /* new cost to v is either old cost to v or known
   shortest path cost to w plus cost from w to v */
until all nodes in N'
```

Dijkstra's algorithm: example

Step	N'	D(v) p(v)	D(w) p(w)	D(x) p(x)	D(y) p(y)	D(z) p(z)
0	u	7,u	3,u	5,u	∞	∞
1	uw	6,w		5,u	11,w	∞
2	uwx	6,w			11,w	14,x
3	uwxv				10,v	14,x
4	uwxvy					12,y
5	uwxvyz					

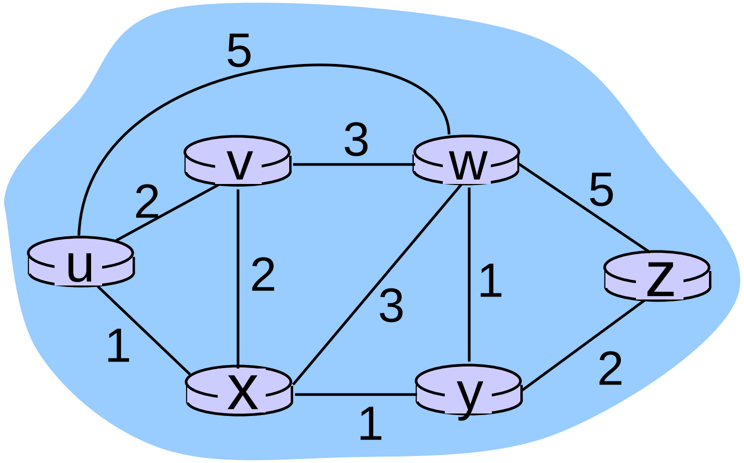
notes:

- construct shortest path tree by tracing predecessor nodes
- ties can exist (can be broken arbitrarily)



Dijkstra's algorithm: another example

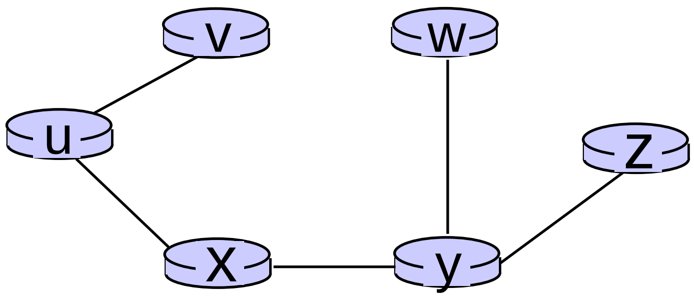
Step	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					



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

Dijkstra's algorithm: example (2)

resulting shortest-path tree from u:

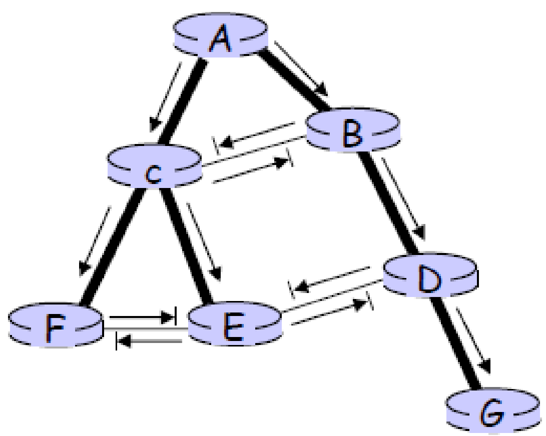


resulting forwarding table in u:

destination	link
v	(u,v)
x	(u,x)
y	(u,x)
w	(u,x)
z	(u,x)

Broadcast routing

- Flooding: Deliver packets from source to all other nodes
  - Broadcast storm

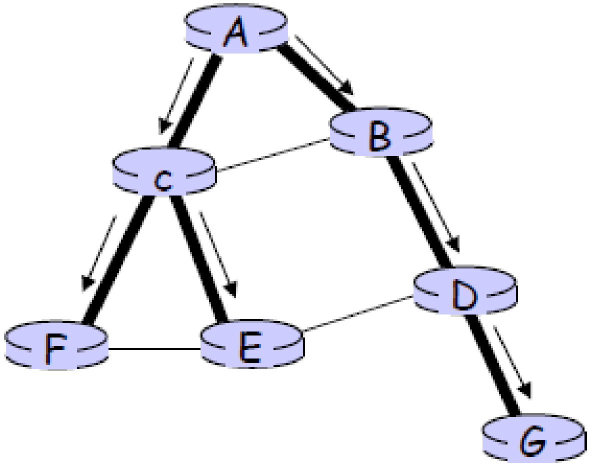


RFP: Reverse path forwarding

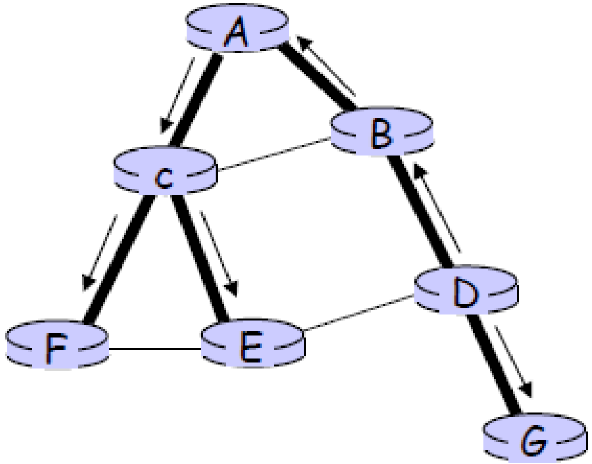
- Controlled flooding:
  - Sequence-number-controlled flooding: add *ID + broadcast sequence number* into the broadcast packets
  - reverse path forwarding, RPF
- Be able to avoid the broadcast storm

Broadcast routing

- Spanning-tree broadcast
  - Suppose you have a connected undirected graph
  - …then a spanning tree of the graph is a connected subgraph in which there are no cycles
- Be able to avoid the transmission of redundancy broadcast packets



(a) Broadcast initiated at A



(b) Broadcast initiated at D

Link state algorithm

Broadcast routing + Dijkstra's algorithm

- Having each node broadcast link-state packets to all other nodes → all nodes have an identical and complete view of the network.
- Using Dijkstra's algorithm compute the least-cost path from one node to all other nodes in the network
- If a link cost changes, re-broadcast and re-compute

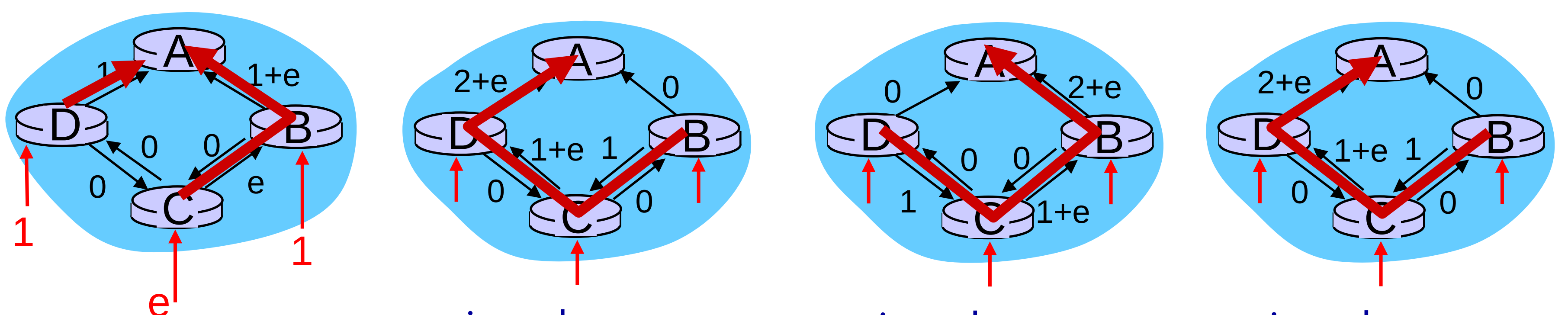
Link state algorithm,discussion

Algorithm complexity: n nodes

- Each iteration: need to check all nodes, w, not in N
- N(n+1)/2 comparisons:$O(n^2)$
- More efficient implementations possible: O(nlogn)

Oscillations possible:

- E.g., link cost = amount of carried traffic



initially

given these costs, find new routing…. resulting in new costs

given these costs, find new routing…. resulting in new costs

given these costs, find new routing…. resulting in new costs

Distance vector algorithm

Bellman-Ford equation (dynamic programming)

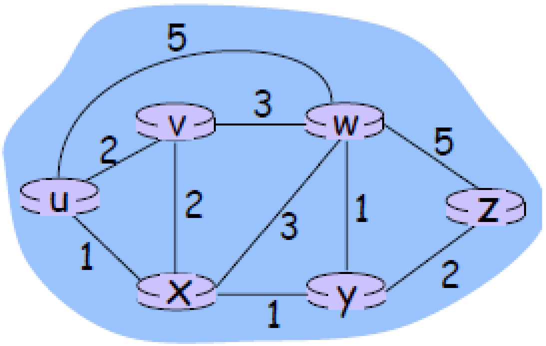
Define

$d_x(y)$:= cost of least-cost path from x to y

Then

$d_x(y) = \min_v \{c(x,v) + d_v(y)\}$

where min is taken over all neighbors v of x



For example:

Clearly, $d_v(z)=5$, $d_w(z)=3$, $d_x(z)=3$

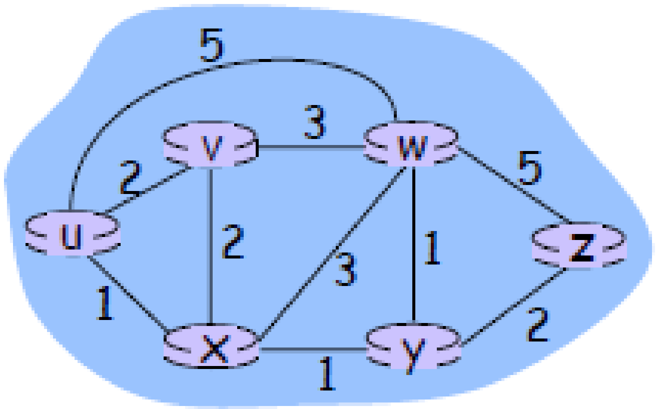
B-F equation says:

$d_u(z) = \min \{ c(u,v) + d_v(z),$
$c(u,x) + d_x(z),$
$c(u,w) + d_w(z) \}$
$= \min(2+5, 1+3, 5+3) = 4$

Distdance vector algorithm

- $D_x(y)$ = estimate of least cost from x to y
  - x maintains distance vector $D_x = [D_x(y): y \in N]$
- node x:
  - knows cost to each neighbor v: $c(x,v)$
  - maintains its neighbors' distance vectors. For each neighbor v, x maintains $D_v = [D_v(y): y \in N]$

Bellman-Ford Example



Distance vectors stored at node x

from \ Cost to	x	y	z	u	v	w
x	**0**	**1**	**3**	**1**	**2**	**2**
y	1	0	2	2	3	1
u	1	2	4	0	2	5
v	2	3	5	3	0	3
w	2	1	3	5	3	0

Routing table at node x

destination	y	z	u	v	w
hop, cost	y,1	y,1	u,1	v,2	y,2

Distdance vector algorithm

Basic idea:

- Each node periodically sends its own distance vector estimate to neighbors
- When a node x receives new DV estimate from neighbor, it updates its own DV using B-F equation

$D_x(y) \leftarrow \min_v\{c(x,v)+D_v(y)\}$ for each node $y \in N$

- Under minor, natural conditions, the estimate $D_x(y)$ converge to the actual least cost $d_x(y)$

Distance vector algorithm

Iterative, asynchronous:

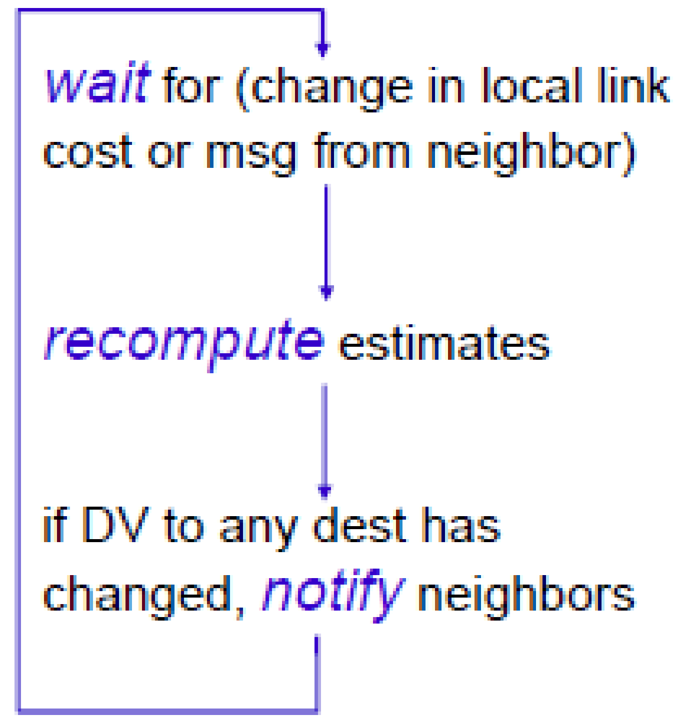
Each local iteration caused by:

- Local link cost change
- DV update message from neighbor

Distributed:

- Each node notifies neighbors only when its DV change
- Neighbors then notify their neighbors if necessary

Each node:



$$D_x(y) = \min\{c(x,y) + D_y(y),\ c(x,z) + D_z(y)\}$$
$$= \min\{2+0\ ,\ 7+1\} = 2$$

$$D_x(z) = \min\{c(x,y) + D_y(z),\ c(x,z) + D_z(z)\}$$
$$= \min\{2+1\ ,\ 7+0\} = 3$$

node x table

cost to

from	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

cost to

from	x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

node y table

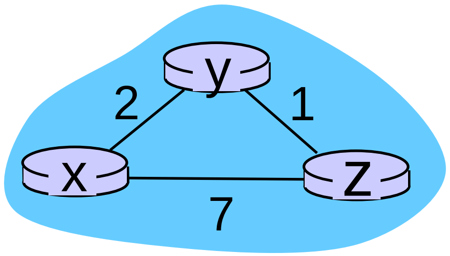
cost to

from	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

node z table

cost to

from	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0



time

$$D_x(y) = \min\{c(x,y) + D_y(y),\ c(x,z) + D_z(y)\} = \min\{2+0\ ,\ 7+1\} = 2$$

$$D_x(z) = \min\{c(x,y) + D_y(z),\ c(x,z) + D_z(z)\} = \min\{2+1\ ,\ 7+0\} = 3$$

node x table

from \ cost to	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

node y table

from \ cost to	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

from \ cost to	x	y	z
x	0	2	7
y	2	0	1
z	7	1	0

from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

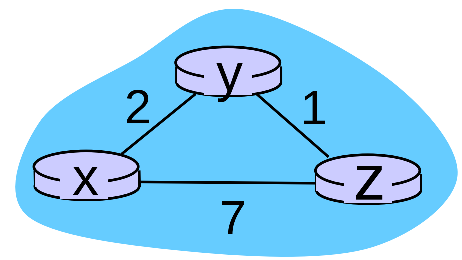
node z table

from \ cost to	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

from \ cost to	x	y	z
x	0	2	7
y	2	0	1
z	3	1	0

from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

time

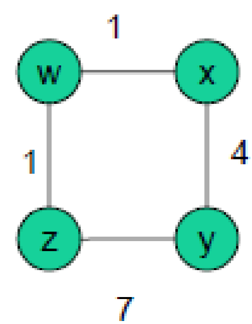


Distance vector: link cost changes

Link cost changes:

- Node detects local link cost change
- Updates routing info, recalculates distance vector
- If DV changes, notigy neighbors

"good news travels fast"



"good news travels fast, bad news travels slow"

node w table

from \ cost to	x	y	z	w
w	1	5	1	0
x	0	4	2	1
z	2	6	0	1

node x table

from \ cost to	x	y	z	w
x	0	4	2	1
w	1	5	1	0
y	4	0	6	5

node y table

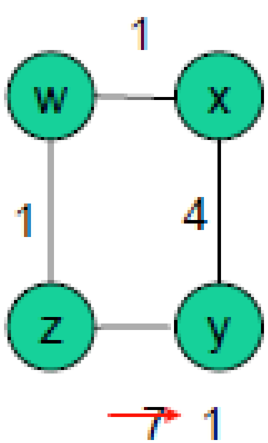
from \ cost to	x	y	z	w
y	4	0	6	5
x	0	4	2	1
z	2	6	0	1

node z table

from \ cost to	x	y	z	w
z	2	6	0	1
w	1	5	1	0
y	4	0	6	5

"good news travels fast"

Algorithm converges in 3 steps.



node w table

from \ cost to	x	y	z	w
w	1	5	1	0
x	0	4	2	1
z	2	6	0	1

from \ cost to	x	y	z	w
w	1	2	1	0
x	0	4	2	1
z	2	1	0	1

from \ cost to	x	y	z	w
w	1	2	1	0
x	0	4	2	1
z	2	1	0	1

node x table

from \ cost to	x	y	z	w
x	0	4	2	1
w	1	5	1	0
y	4	0	6	5

from \ cost to	x	y	z	w
x	0	4	2	1
w	1	5	1	0
y	3	0	1	2

from \ cost to	x	y	z	w
x	0	3	2	1
w	1	2	1	0
y	3	0	1	2

node y table

from \ cost to	x	y	z	w
y	3	0	1	2
x	0	4	2	1
z	2	6	0	1

from \ cost to	x	y	z	w
y	3	0	1	2
x	0	4	2	1
z	2	1	0	1

from \ cost to	x	y	z	w
y	3	0	1	2
x	0	4	2	1
z	2	1	0	1

node z table

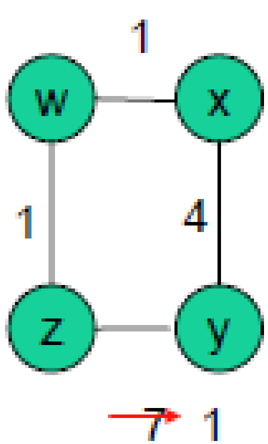
from \ cost to	x	y	z	w
z	2	1	0	1
w	1	5	1	0
y	4	0	6	5

from \ cost to	x	y	z	w
z	2	1	0	1
w	1	5	1	0
y	3	0	1	2

from \ cost to	x	y	z	w
z	2	1	0	1
w	1	2	1	0
y	3	0	1	2

"good news travels fast"

Algorithm converges in 3 steps.



node w table

from \ cost to	x	y	z	w
w	1	2	1	0
x	0	4	2	1
z	2	1	0	1

from \ cost to	x	y	z	w
w	1	2	1	0
x	0	3	2	1
z	2	1	0	1

node x table

from \ cost to	x	y	z	w
x	0	3	2	1
w	1	2	1	0
y	3	0	1	2

from \ cost to	x	y	z	w
x	0	3	2	1
w	1	2	1	0
y	3	0	1	2

node y table

from \ cost to	x	y	z	w
y	3	0	1	2
x	0	4	2	1
z	2	1	0	1

from \ cost to	x	y	z	w
y	3	0	1	2
x	0	3	2	1
z	2	1	0	1

node z table

from \ cost to	x	y	z	w
z	2	1	0	1
w	1	2	1	0
y	3	0	1	2

from \ cost to	x	y	z	w
z	2	1	0	1
w	1	2	1	0
y	3	0	1	2

Count to infinity problem

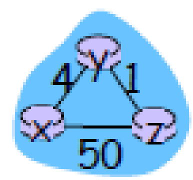
Link cost changes:

- good news travels fast
- Bad news travels slow - "count to infinity" problem!

- for example: 44 iterations before algorithm stabilizes

"bad news travels slow"

"count to infinity" problem



node x table

cost to

from	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

node y table

cost to

from	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

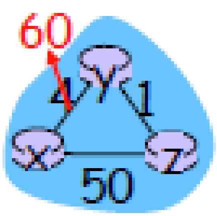
node z table

cost to

from	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

"bad news travels slow"

Algorithm converges in 44 steps.



Cost of link xy changes

node x table

from \ cost to	x	y	z
x	0	51	50
y	4	0	1
z	5	1	0

node x table

from \ cost to	x	y	z
x	0	51	50
y	6	0	1
z	5	1	0

node x table

from \ cost to	x	y	z
x	0	51	50
y	6	0	1
z	7	1	0

node x table

from \ cost to	x	y	z
x	0	51	50
y	8	0	1
z	7	1	0

node y table

from \ cost to	x	y	z
x	0	4	5
y	6	0	1
z	5	1	0

node y table

from \ cost to	x	y	z
x	0	51	50
y	6	0	1
z	5	1	0

node y table

from \ cost to	x	y	z
x	0	51	50
y	8	0	1
z	7	1	0

node y table

from \ cost to	x	y	z
x	0	51	50
y	8	0	1
z	7	1	0

. . . . . .

node z table

from \ cost to	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

node z table

from \ cost to	x	y	z
x	0	51	50
y	6	0	1
z	7	1	0

node z table

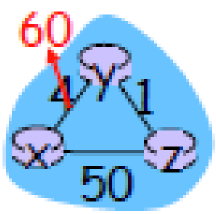
from \ cost to	x	y	z
x	0	51	50
y	6	0	1
z	7	1	0

node z table

from \ cost to	x	y	z
x	0	51	50
y	8	0	1
z	9	1	0

"bad news travels slow"

Algorithm converges in 44 steps.



Cost of link xy changes

node x table

from \ cost to	x	y	z
x	0	51	50
y	48	0	1
z	49	1	0

node x table

from \ cost to	x	y	z
x	0	51	50
y	50	0	1
z	49	1	0

node x table

from \ cost to	x	y	z
x	0	51	50
y	50	0	1
z	50	1	0

node x table

from \ cost to	x	y	z
x	0	51	50
y	51	0	1
z	50	1	0

node y table

from \ cost to	x	y	z
x	0	51	50
y	50	0	1
z	49	1	0

node y table

from \ cost to	x	y	z
x	0	51	50
y	50	0	1
z	49	1	0

node y table

from \ cost to	x	y	z
x	0	51	50
y	51	0	1
z	50	1	0

node y table

from \ cost to	x	y	z
x	0	51	50
y	51	0	1
z	50	1	0

node z table

from \ cost to	x	y	z
x	0	51	50
y	48	0	1
z	49	1	0

node z table

from \ cost to	x	y	z
x	0	51	50
y	50	0	1
z	50	1	0

node z table

from \ cost to	x	y	z
x	0	51	50
y	50	0	1
z	50	1	0

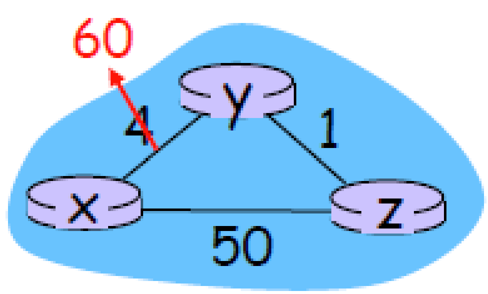
node z table

from \ cost to	x	y	z
x	0	51	50
y	51	0	1
z	50	1	0

How to solve count to infinity problem?

Poisoned reverse:

- If Z routes through Y to get to X:
- Z tells Y its (Z's) distance to X is infinite (so Y won't route to X via Z)
- Will this completely solve count to infinity problem?



node x table

from \ cost to	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

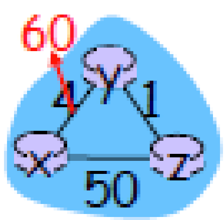
node y table

from \ cost to	x	y	z
x	0	4	∞
y	4	0	1
z	∞	1	0

node z table

from \ cost to	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

Algorithm converges in 3 steps.



Cost of link xy changes to 60

node x table

from \ cost to	x	y	z
x	0	51	50
y	4	0	1
z	5	1	0

node x table

from \ cost to	x	y	z
x	0	51	50
y	60	0	1
z	5	1	0

node x table

from \ cost to	x	y	z
x	0	51	50
y	60	0	1
z	50	1	0

node x table

from \ cost to	x	y	z
x	0	51	50
y	51	0	1
z	50	1	0

node y table

from \ cost to	x	y	z
x	0	4	∞
y	60	0	1
z	∞	1	0

node y table

from \ cost to	x	y	z
x	0	51	50
y	60	0	1
z	∞	1	0

node y table

from \ cost to	x	y	z
x	0	51	50
y	51	0	1
z	50	1	0

node y table

from \ cost to	x	y	z
x	0	51	50
y	51	0	1
z	50	1	0

node z table

from \ cost to	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

node z table

from \ cost to	x	y	z
x	0	∞	50
y	60	0	1
z	50	1	0

node z table

from \ cost to	x	y	z
x	0	∞	50
y	60	0	1
z	50	1	0

node z table

from \ cost to	x	y	z
x	0	∞	50
y	∞	0	1
z	50	1	0

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

Hierarchical Routing

Our routing study thus far - idealization

- all routers identical
- network "flat"

… *not* true in practice

scale: with billions of 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

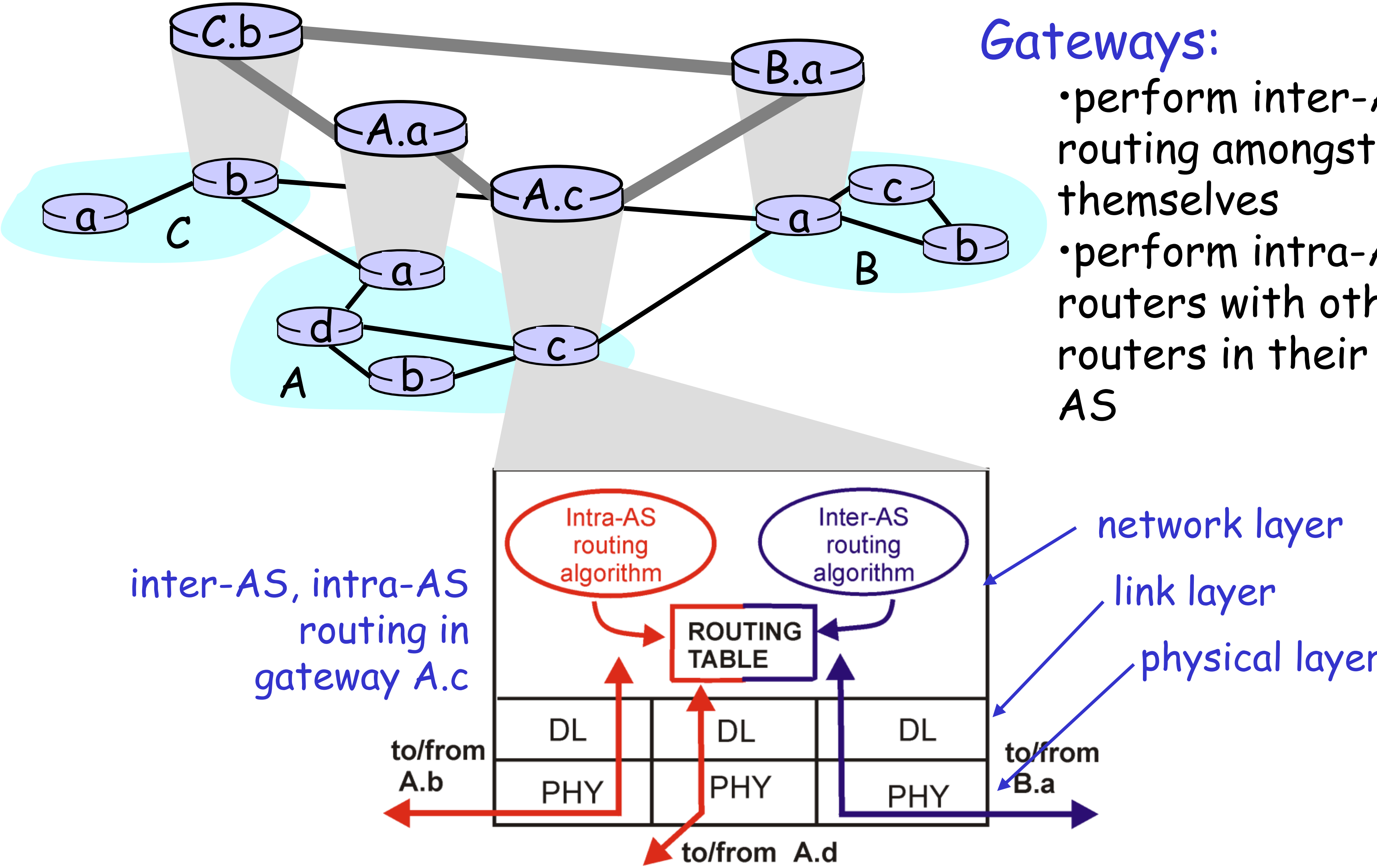
Hierarchical Routing

- 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 routers

- special routers in AS
- run intra-AS routing protocol with all other routers in AS
- *also* responsible for routing to destinations outside AS
  - run *inter-AS routing* protocol with other gateway routers

Intra-AS and Inter-AS routing

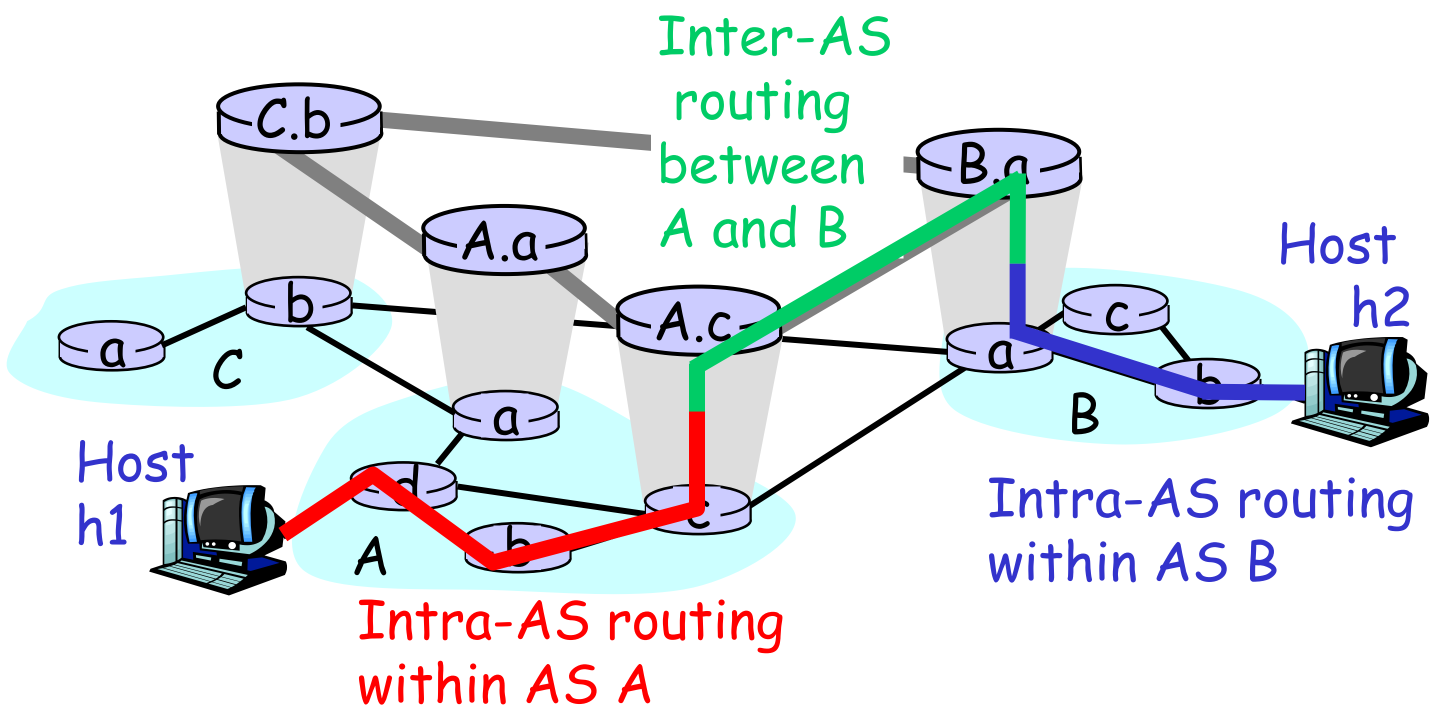


inter-AS, intra-AS routing in gateway A.c

Gateways:

- perform inter-AS routing amongst themselves
- perform intra-AS routers with other routers in their AS

Intra-AS and Inter-AS routing



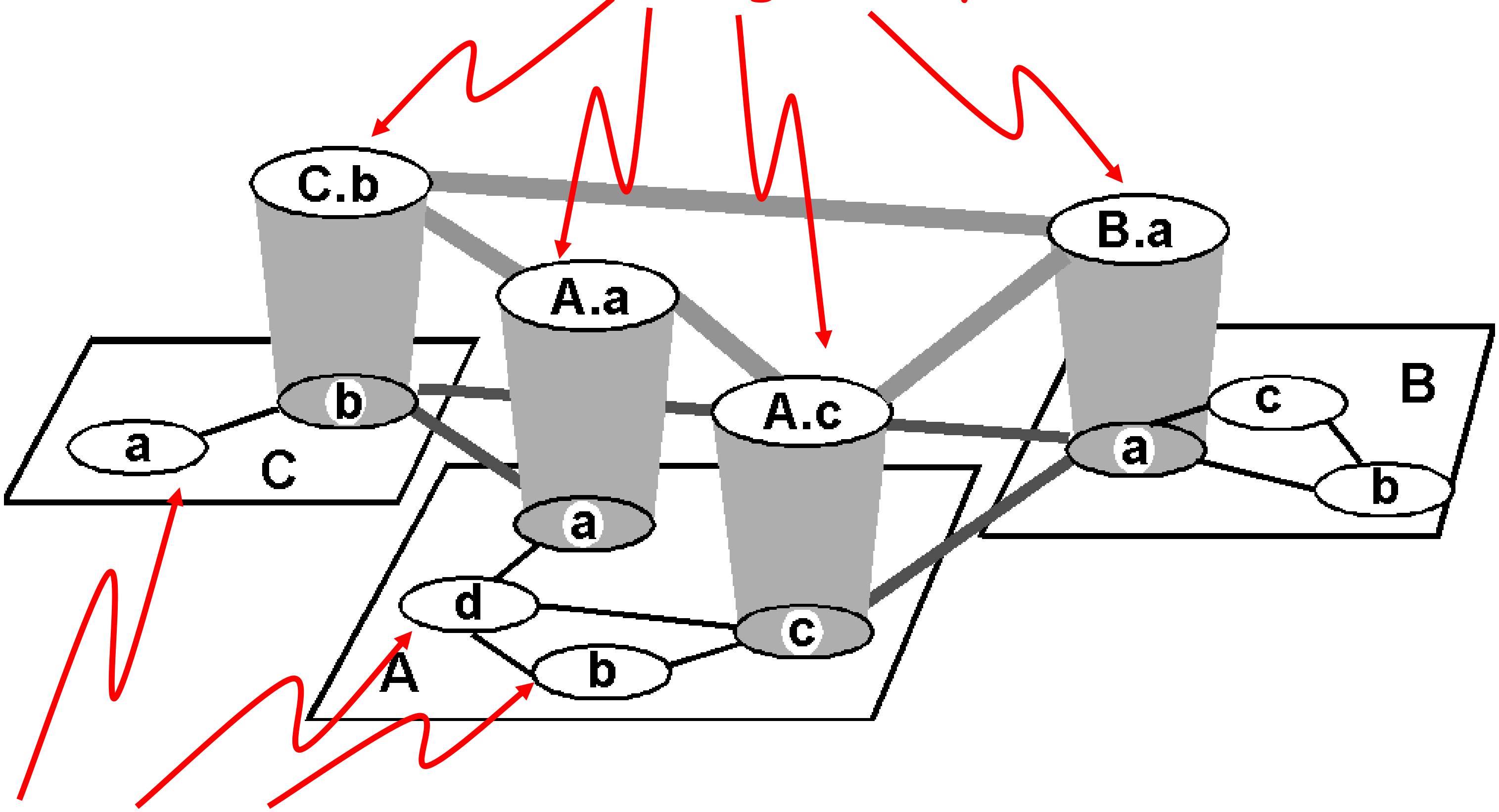
- We'll examine specific inter-AS and intra-AS Internet routing protocols shortly

Routing in the Internet

- The Global Internet consists of Autonomous Systems (AS) interconnected with each other:
  - **Stub AS**: small corporation
  - **Multihomed AS**: large corporation (no transit)
  - **Transit AS**: provider
- Two-level routing:
  - **Intra-AS**: administrator is responsible for choice
  - **Inter-AS**: unique standard

Internet AS Hierarchy

Inter-AS border (exterior gateway) routers



Intra-AS interior (gateway) routers

Chapter 5: Network Layer

- 5.1 Introduction
- 5.2 Virtual circuit and datagram networks
- 5.3 What's inside a router?
- 5.4 Routing algorithms:
  - Dijkstra's algorithm
  - Broadcast routing
  - Link state
  - Distance vector
  - Hierarchical routing
- 5.5 Routing in the Internet
- 5.6 IP: Internet protocol
  - IPv4 Datagram format
  - IPv4 addressing
  - IP fragment
  - NAT
  - ARP
  - ICMP
  - IPv6

Intra-AS Routing

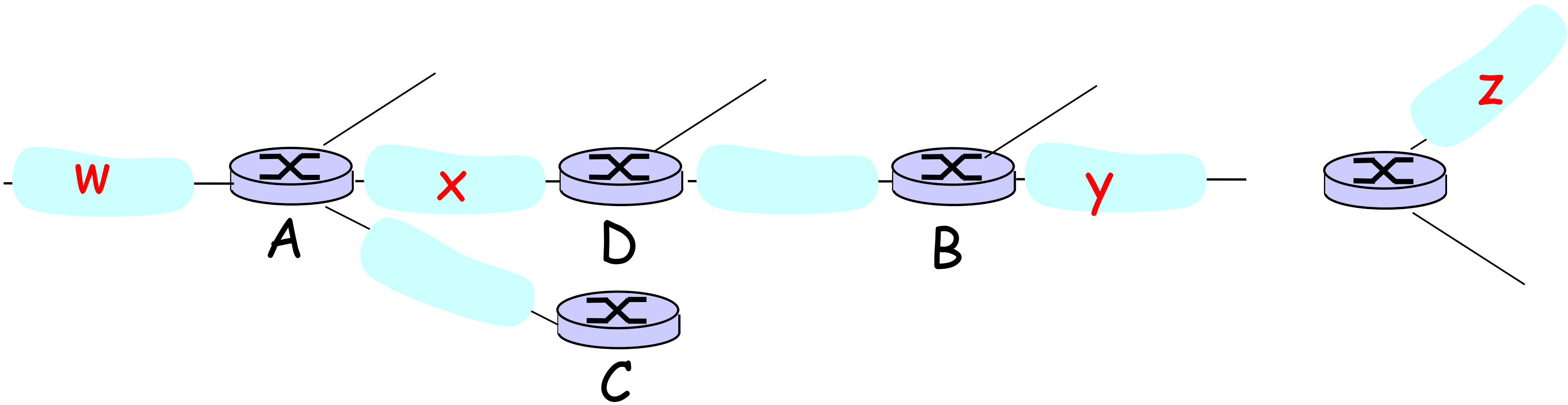
- Also known as Interior Gateway Protocols (IGP)
- Most common intra-AS routing protocols:
  - RIP: Routing Information Protocol
  - OSPF: Open Shortest Path First
  - IGRP: Interior Gateway Routing Protocol (Cisco proprietary for decades, until 2016)

RIP (Routing Information Protocol)

- Distance vector algorithm
- Included in BSD-UNIX Distribution in 1982
- Distance metric: # of hops (max = 15 hops)
  - *Can you guess why?*

- Distance vectors: exchanged every 30 sec via Response Message (also called **advertisement**)
- Each advertisement: route to up to 25 destination nets

RIP (Routing Information Protocol)



Destination Network	Next Router	Num. of hops to dest.
w	A	2
y	B	2
z	B	7
x	--	1
….	….	….

Routing table in D

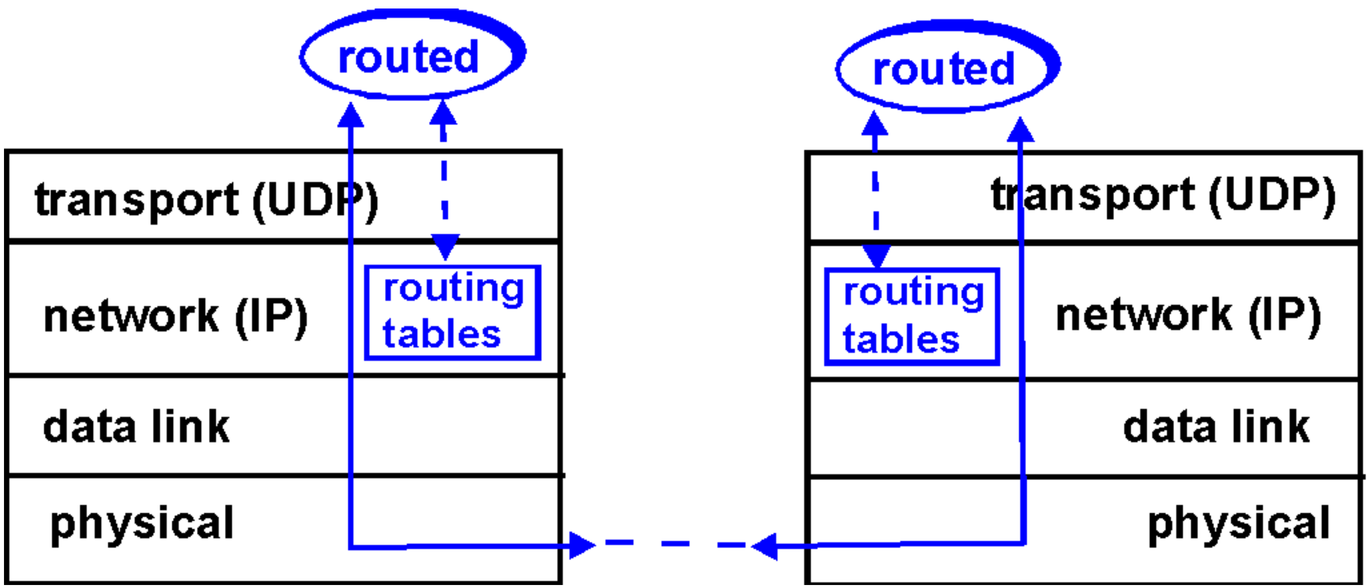
RIP: Link Failure and Recovery

If no advertisement heard after 180 sec --> neighbor/link declared dead

- routes via neighbor invalidated
- new advertisements sent to neighbors
- neighbors in turn send out new advertisements (if tables changed)
- link failure info quickly propagates to entire net
- poison reverse used to prevent ping-pong loops (infinite distance = 16 hops)

RIP Table processing

- RIP routing tables managed by **application-level** process called route-d (daemon)
- advertisements sent in UDP packets, periodically repeated



RIP Table example (continued)

Router: *giroflee.eurocom.fr*

Destination	Gateway	Flags	Ref	Use	Interface
127.0.0.1	127.0.0.1	UH	0	26492	lo0
192.168.2.	192.168.2.5	U	2	13	fa0
193.55.114.	193.55.114.6	U	3	58503	le0
192.168.3.	192.168.3.5	U	2	25	qaa0
224.0.0.0	193.55.114.6	U	3	0	le0
default	193.55.114.129	UG	0	143454	

- Three attached class C networks (LANs)
- Router only knows routes to attached LANs
- Default router used to "go up"
- Route multicast address: 224.0.0.0
- Loopback interface (for debugging)

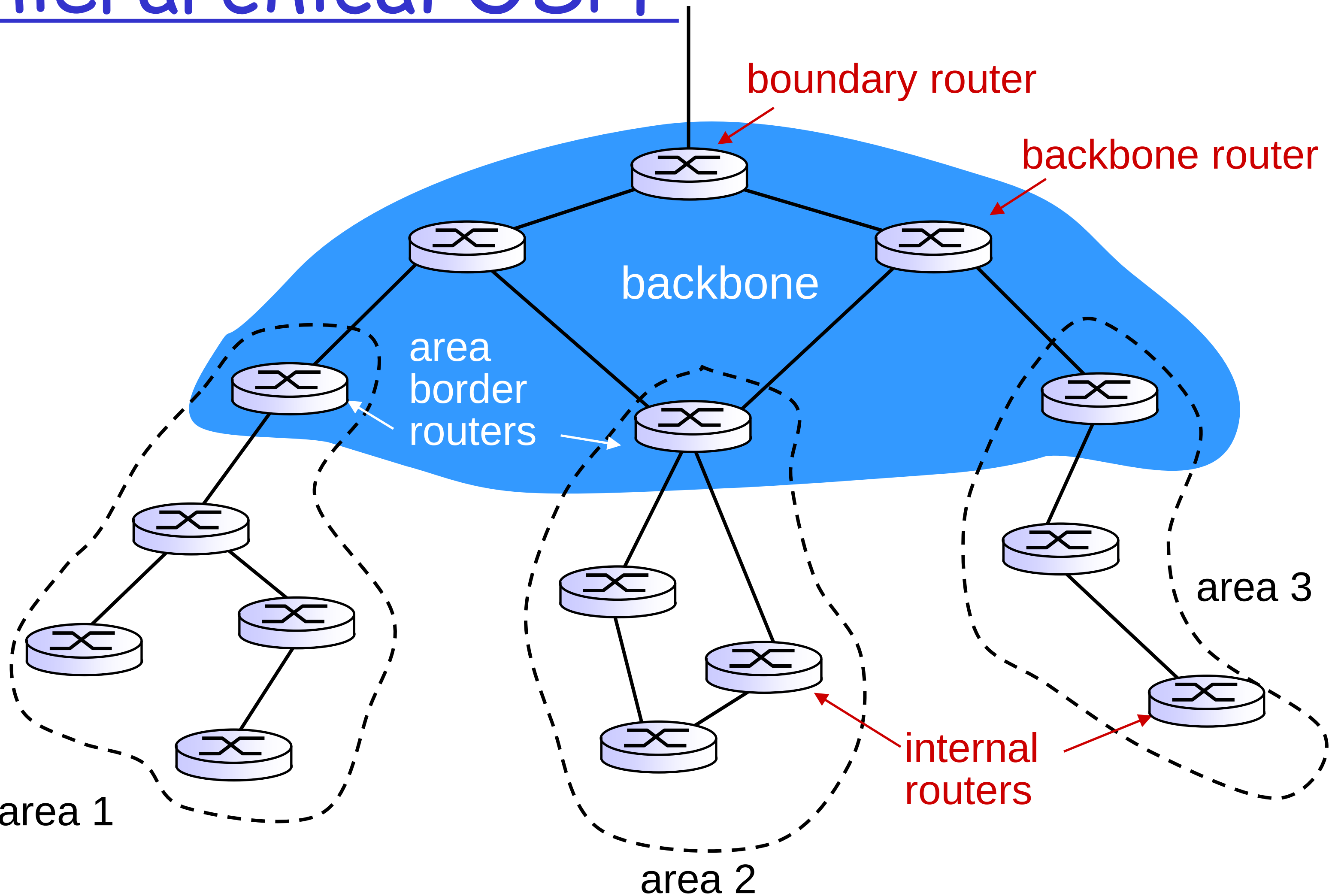
OSPF (Open Shortest Path First)

- "open": publicly available
- Uses Link -State algorithm
  - LS packet dissemination
  - Topology map at each node
  - Route computation using Dijkstra's algorithm
- OSPF advertisements are disseminated to entire AS (via flooding)
- Link state advertisements are periodically updated
- OSPF advertisements in OSPF messages are carried directly by IP

OSPF “advanced” features (not in RIP)

- Security: all OSPF messages authenticated (to prevent malicious intrusion); TCP connections used
- Multiple same-cost paths allowed (only one path in RIP)
- For each link, multiple cost metrics for different TOS (eg, satellite link cost set “low” for best effort; high for real time)
- Integrated uni- and multicast support:
  - Multicast OSPF (MOSPF) uses same topology data base as OSPF
- Hierarchical OSPF in large domains.

Hierarchical OSPF



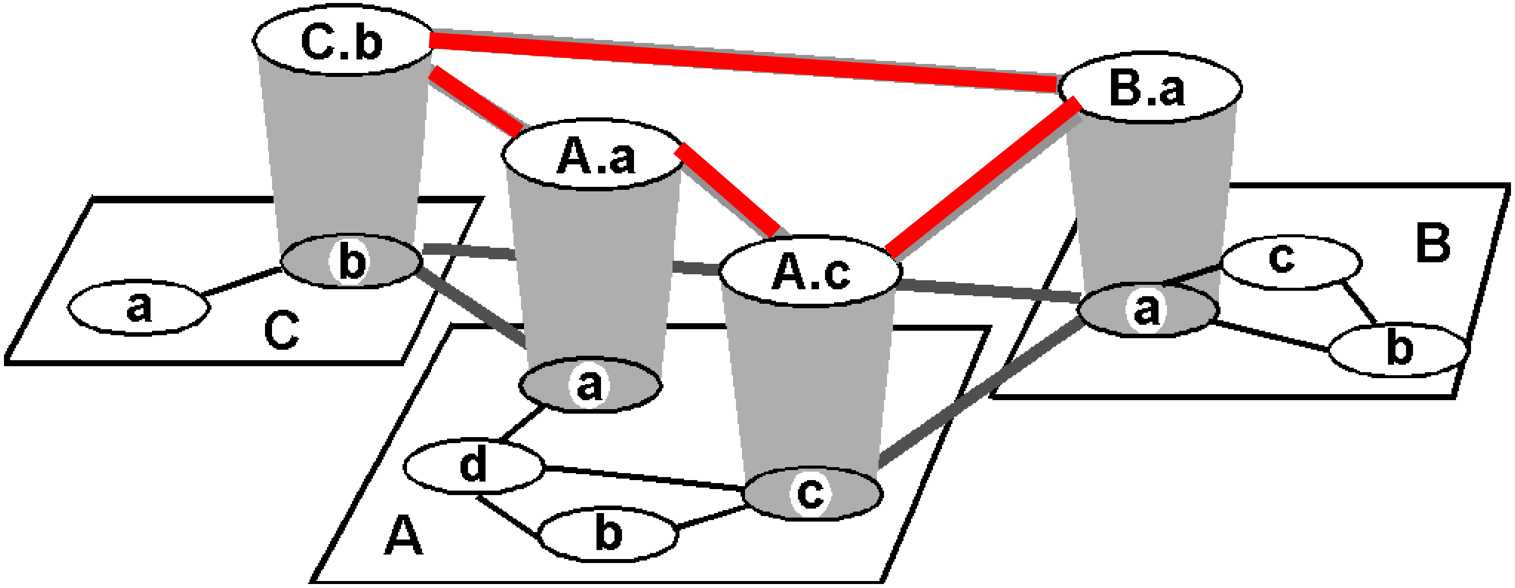
Hierarchical OSPF

- Two-level hierarchy: local area, backbone.
  - Link-state advertisements only in area
  - each nodes has detailed area topology; only know direction (shortest path) to nets in other areas.
- **Area border routers:** "summarize" distances to nets in own area, advertise to other Area Border routers.
- **Backbone routers:** run OSPF routing limited to backbone.
- **Boundary routers:** connect to other ASs.

IGRP (Interior Gateway Routing Protocol)

- CISCO proprietary; successor of RIP (mid 80s)
- Distance Vector, like RIP
- several cost metrics (delay, bandwidth, reliability, load etc)
- uses TCP to exchange routing updates
- Loop-free routing via Distributed Updating Alg. (DUAL) based on *diffused computation*

Inter-AS routing



Internet inter-AS routing: BGP

- **BGP (Border Gateway Protocol):** *the* de facto inter-domain routing protocol
- **Path Vector** protocol:
  - similar to Distance Vector protocol
  - each Border Gateway broadcast to neighbors (peers) *entire path* (I.e, sequence of ASs) to destination
  - E.g., Gateway X may send its path to dest. Z:

Path (X,Z) = X,Y1,Y2,Y3,…,Z

Internet inter-AS routing: BGP

- BGP (Border Gateway Protocol): *the* de facto standard
- BGP provides each AS a means to:
  - Obtain subnet reachability information from neighboring ASs.
  - Propagate reachability information to all AS-internal routers.
  - Determine good routes to subnets based on reachability information and policy.
- Allows subnet to advertise its existence to the rest of Internet: "I am here"

BGP route selection

- Router may learn about more than 1 route to some prefix. Router must select route
- Elimination rules:
  - Local preference value attribute: policy decision
  - Shortest AS-PATH
  - Closest NEXT-HOP router: hot potato routing
  - Additional criteria

Internet inter-AS routing: BGP

Suppose: gateway X send its path to peer gateway W

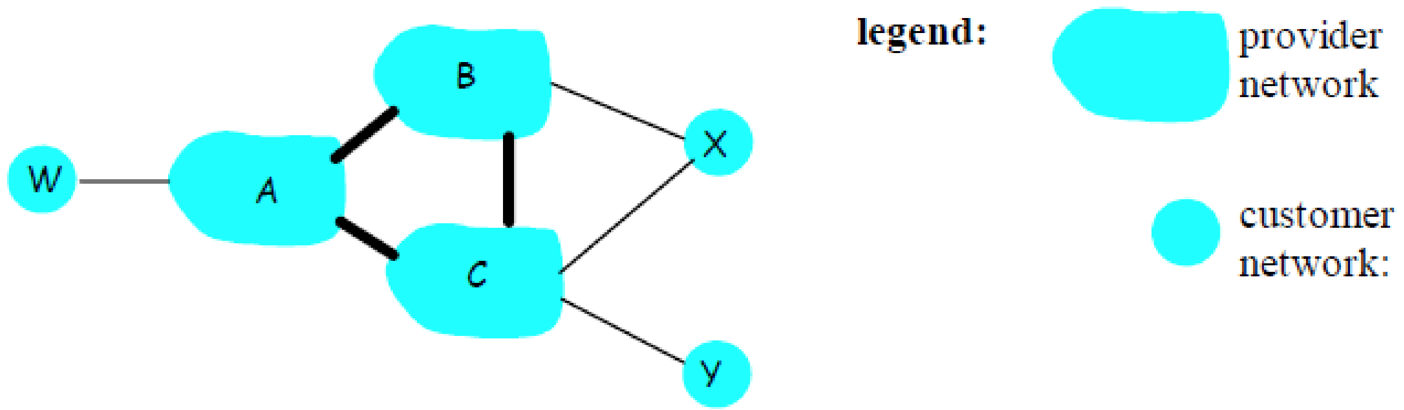
- W may or may not select path offered by X
  - cost, policy (don't route via competitors AS), loop prevention reasons.
- If W selects path advertised by X, then:

  Path (W,Z) = w, Path (X,Z)
- Note: X can control incoming traffic by controling it route advertisements to peers:
  - e.g., don't want to route traffic to Z -> don't advertise any routes to Z

Internet inter-AS routing: BGP

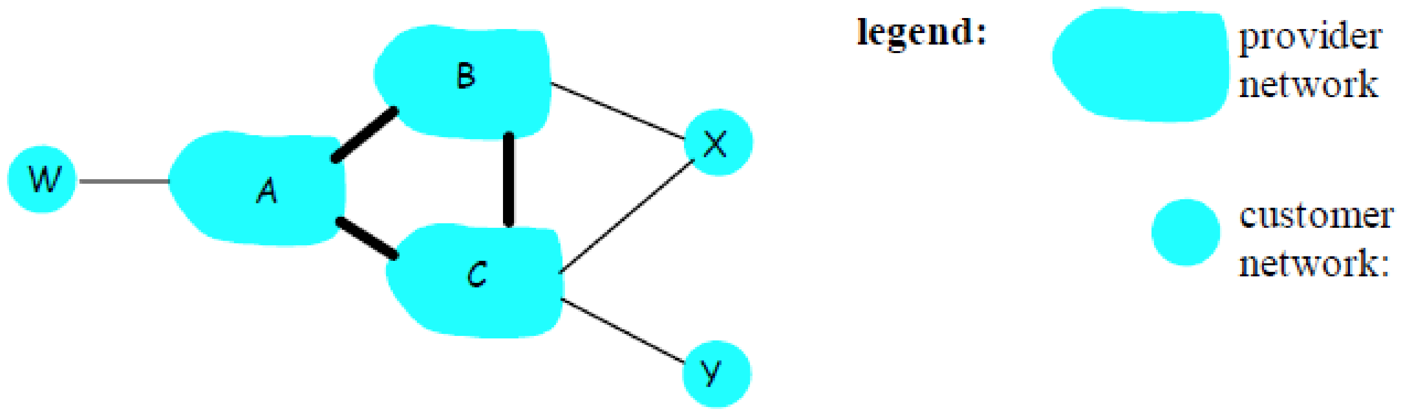
- BGP messages exchanged using TCP.
- BGP messages:
  - OPEN: opens TCP connection to peer and authenticates sender
  - UPDATE: advertises new path (or withdraws old)
  - KEEPALIVE keeps connection alive in absence of UPDATES; also ACKs OPEN request
  - NOTIFICATION: reports errors in previous msg; also used to close connection

BGP Routing Policy



- A,B,C are provider networks
- X,W,Y are customer (of provider networks)
- X is dual-homed: attached to two networks
  - X does not want to route from B via X to C
  - ...so X will not advertise to B a route to C

BGP Routing Policy



- A advertises to B the path AW
- B advertises to X the path BAW
- Should B advertise to C the path BAW?
  - No way! B gets no "revenue" for routing CBAW since neither W nor C are B's customers
  - B wants to force C to route to W via A
  - B wants to route only to/from its customers!

Why different Intra- and 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

Exercises-1

1.The protocols for directly encapsulating RIP、OSPF and BGP messages are

A. TCP、UDP、IP

B. TCP、IP、UDP

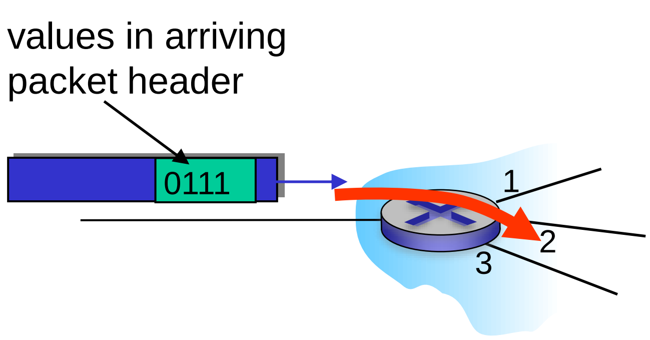
C. UDP、TCP、IP

D. UDP、IP、TCP

*Network layer: data plane, control plane

Data plane

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

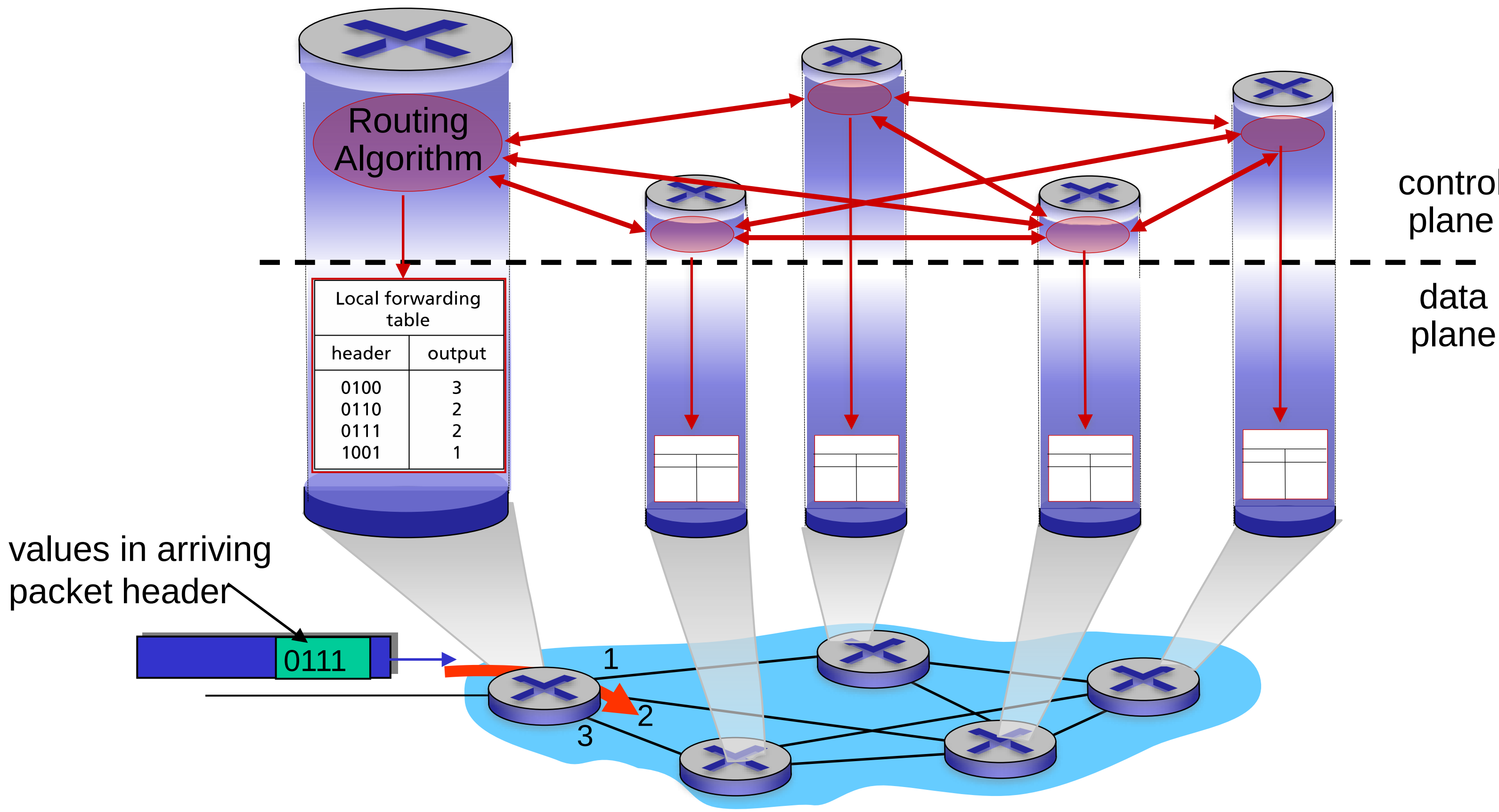


Control plane

- 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

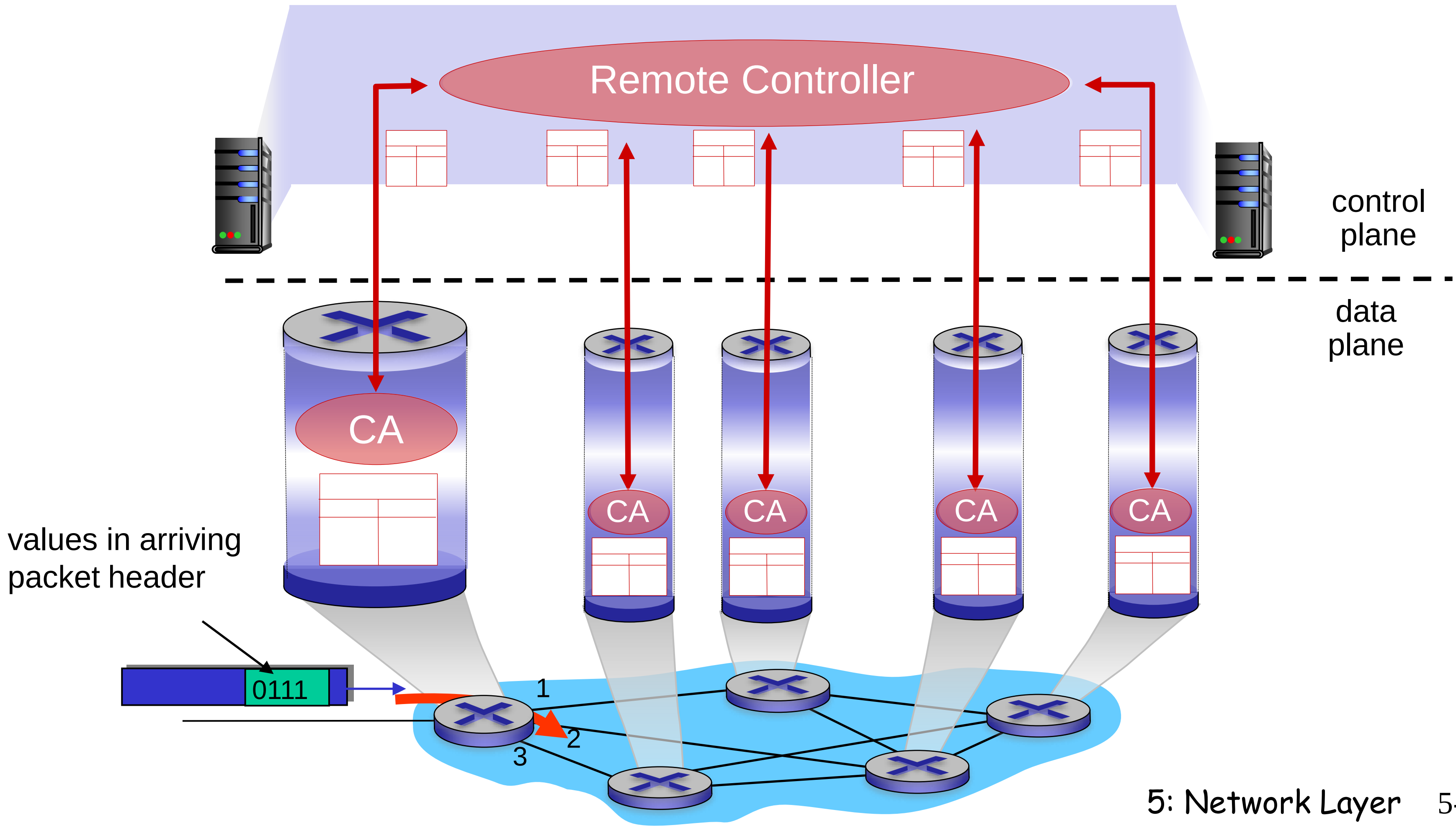
*Per-router control plane

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



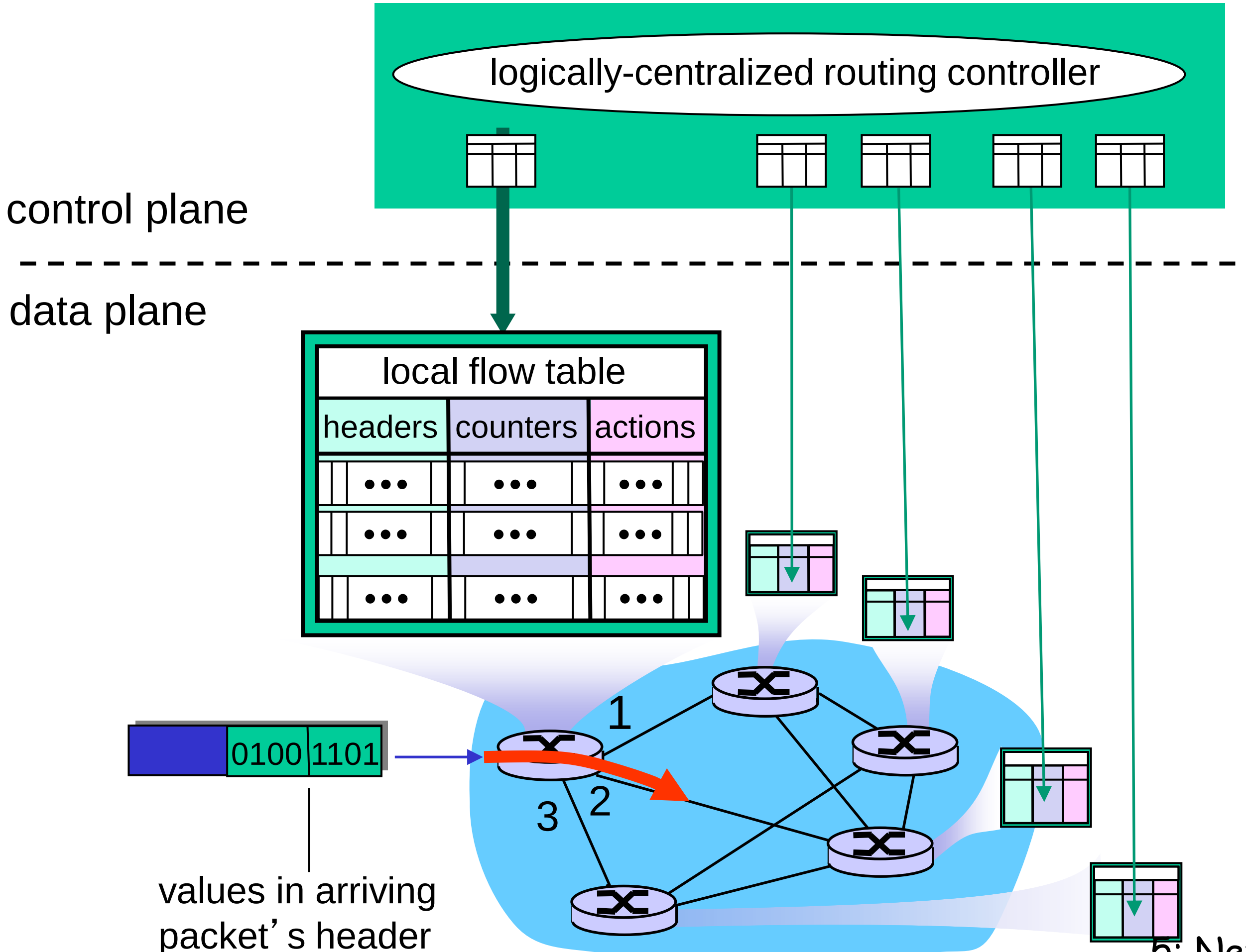
*Logically centralized control plane

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

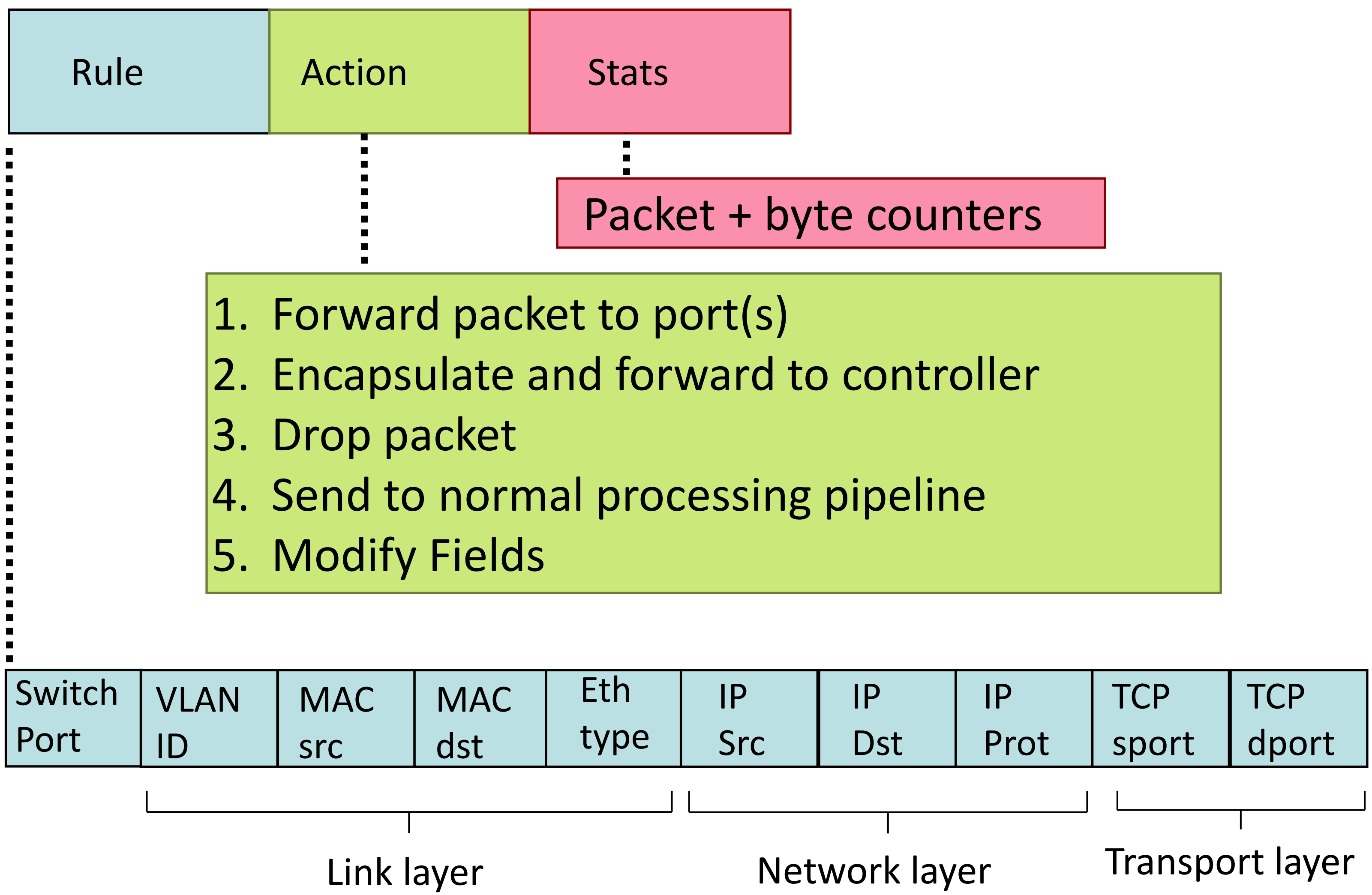


*Generalized Forwarding and SDN

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



*OpenFlow: Flow Table Entries



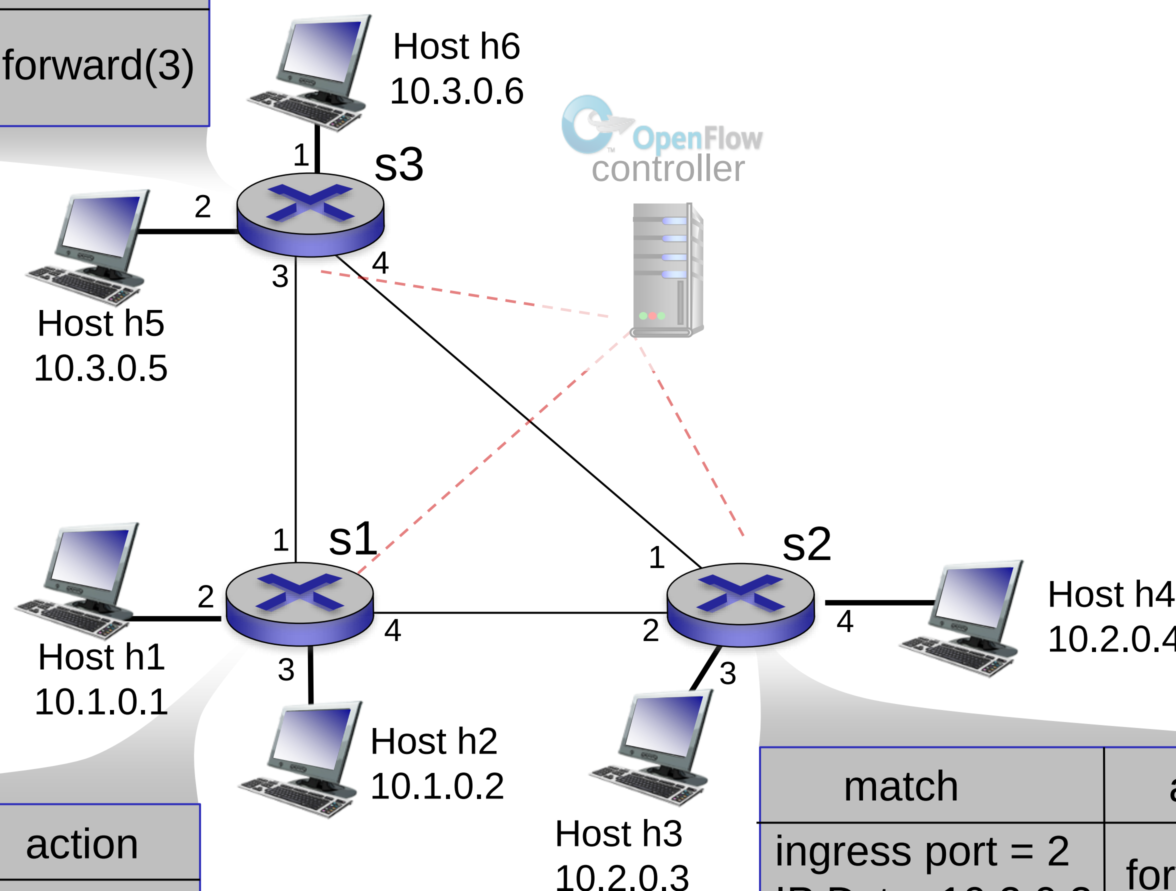
*OpenFlow abstraction

- *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

*OpenFlow example

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)

Homework

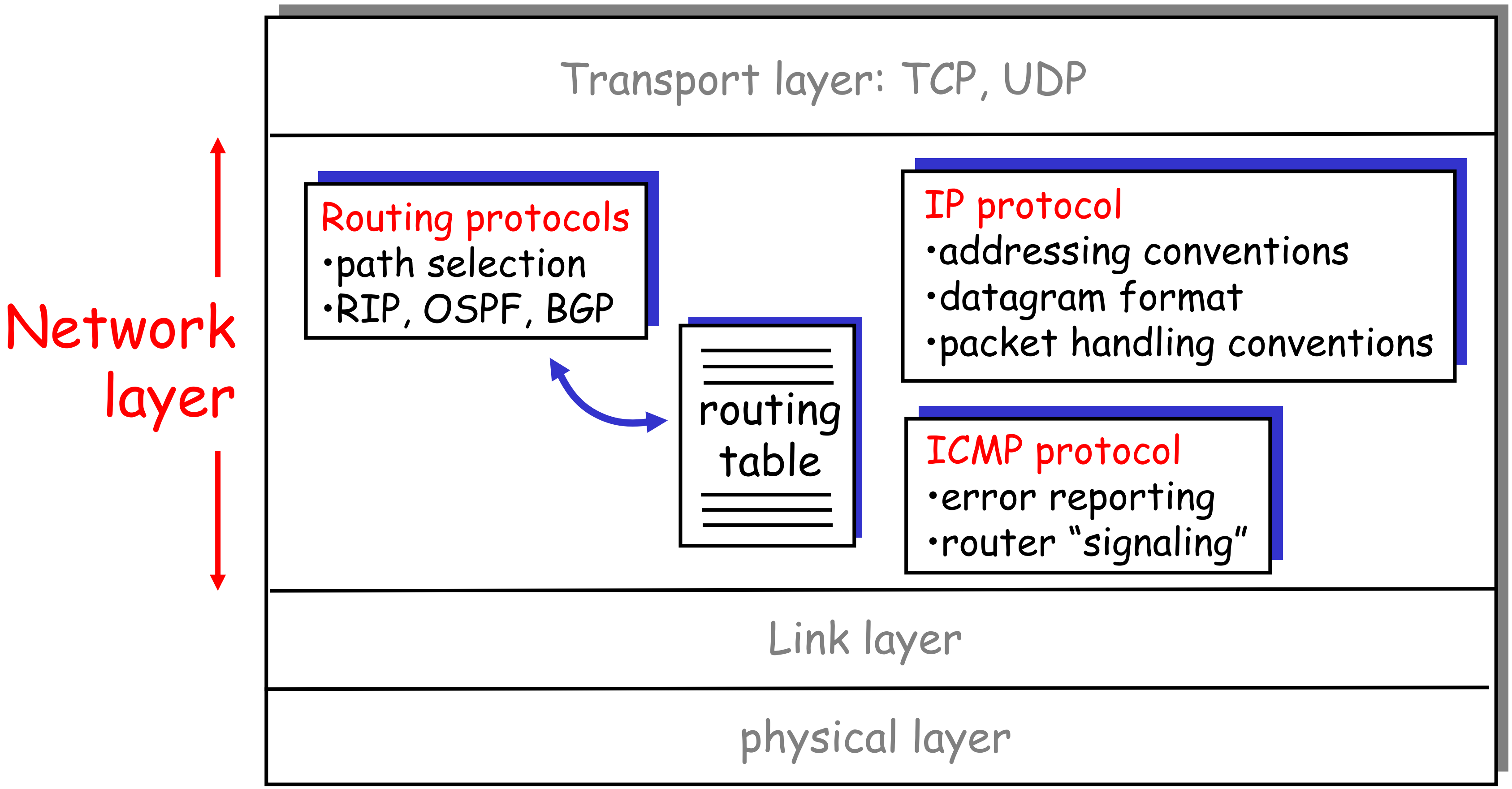
- P455:R4,R5
- P457:P3,P5
- P458:P11
- P396:P14

Chapter 5: Network Layer

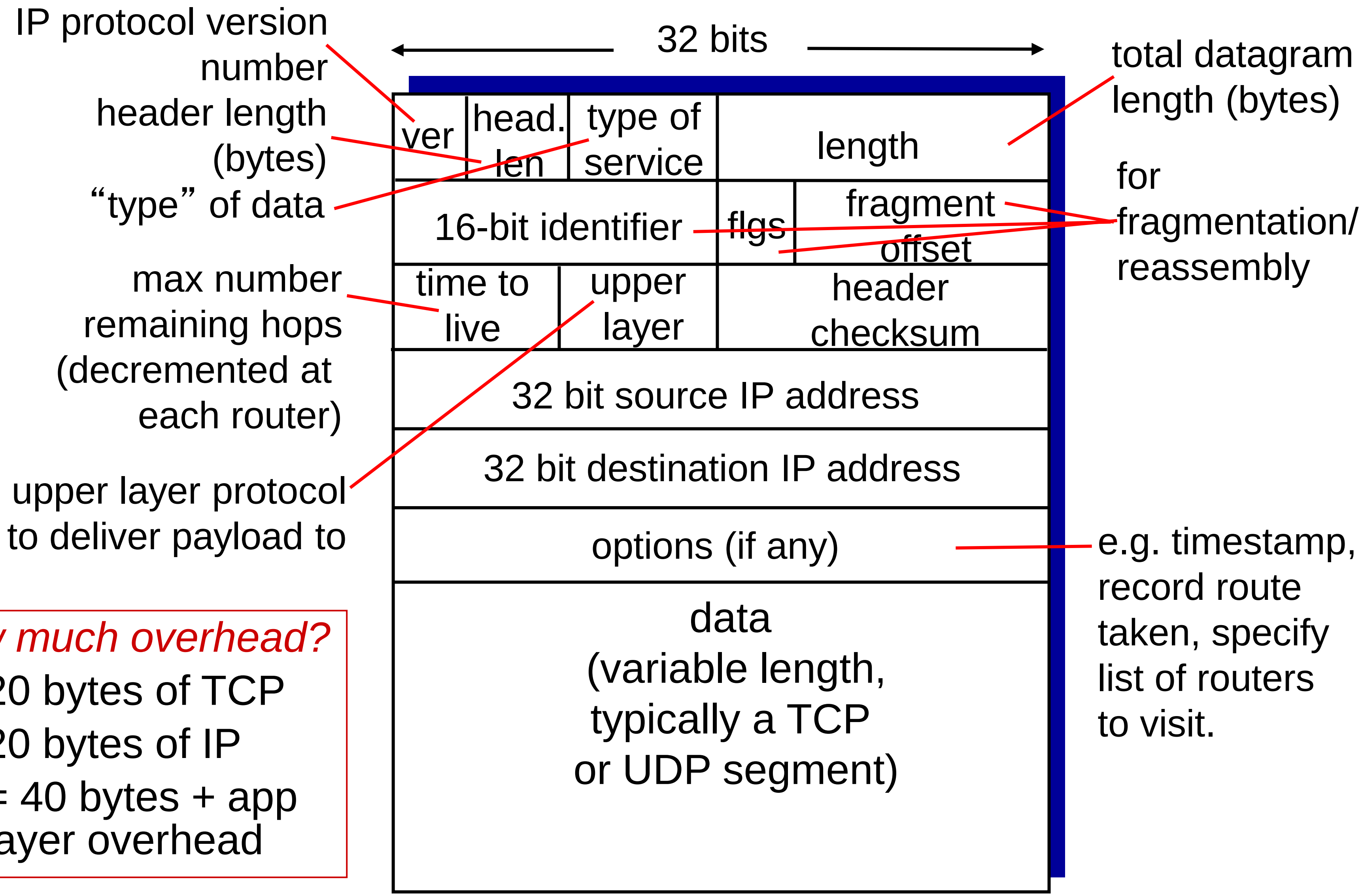
- 5.1 Introduction
- 5.2 Virtual circuit and datagram networks
- 5.3 What's inside a router?
- 5.4 Routing algorithms:
  - Dijkstra's algorithm
  - Broadcast routing
  - Link state
  - Distance vector
  - Hierarchical routing
- 5.5 Routing in the Internet
- 5.6 IP: Internet protocol
  - IPv4 Datagram format
  - IPv4 addressing
  - IP fragment
  - NAT
  - ARP
  - ICMP
  - IPv6

The Internet Network layer

Host, router network layer functions:

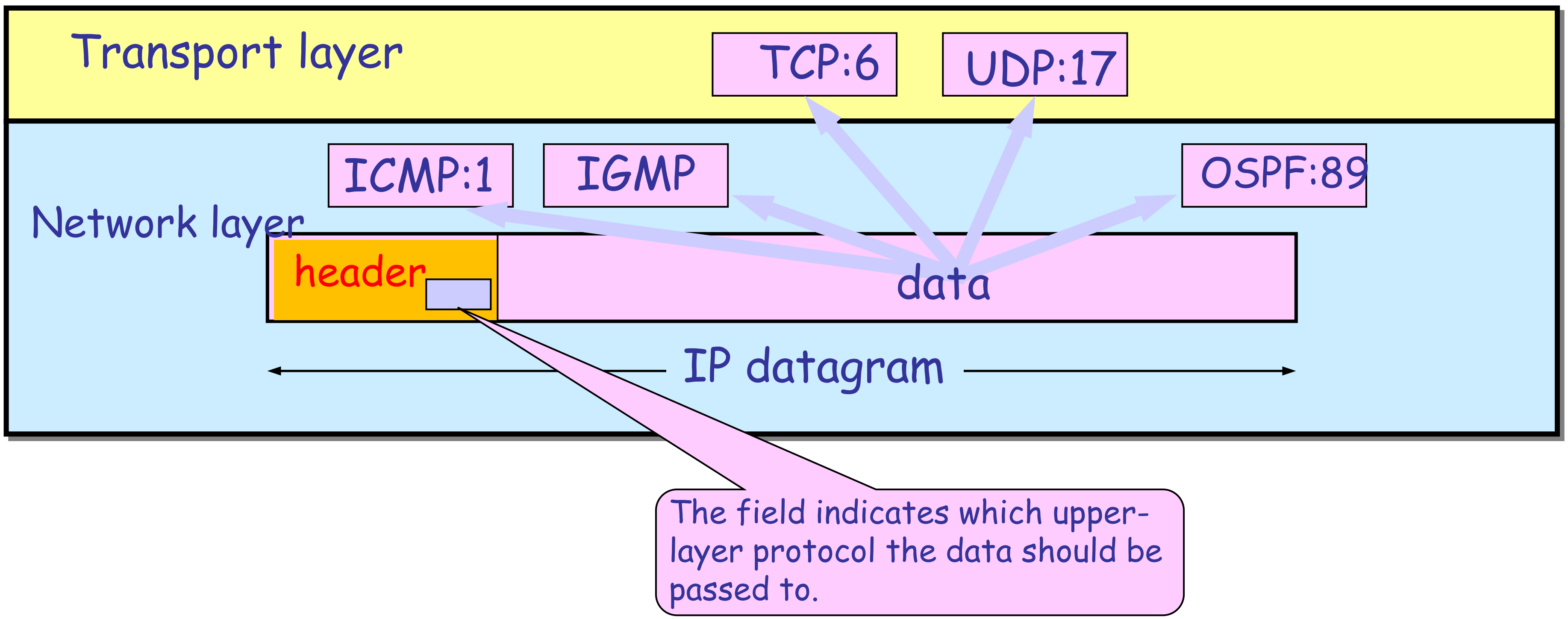


IP datagram format

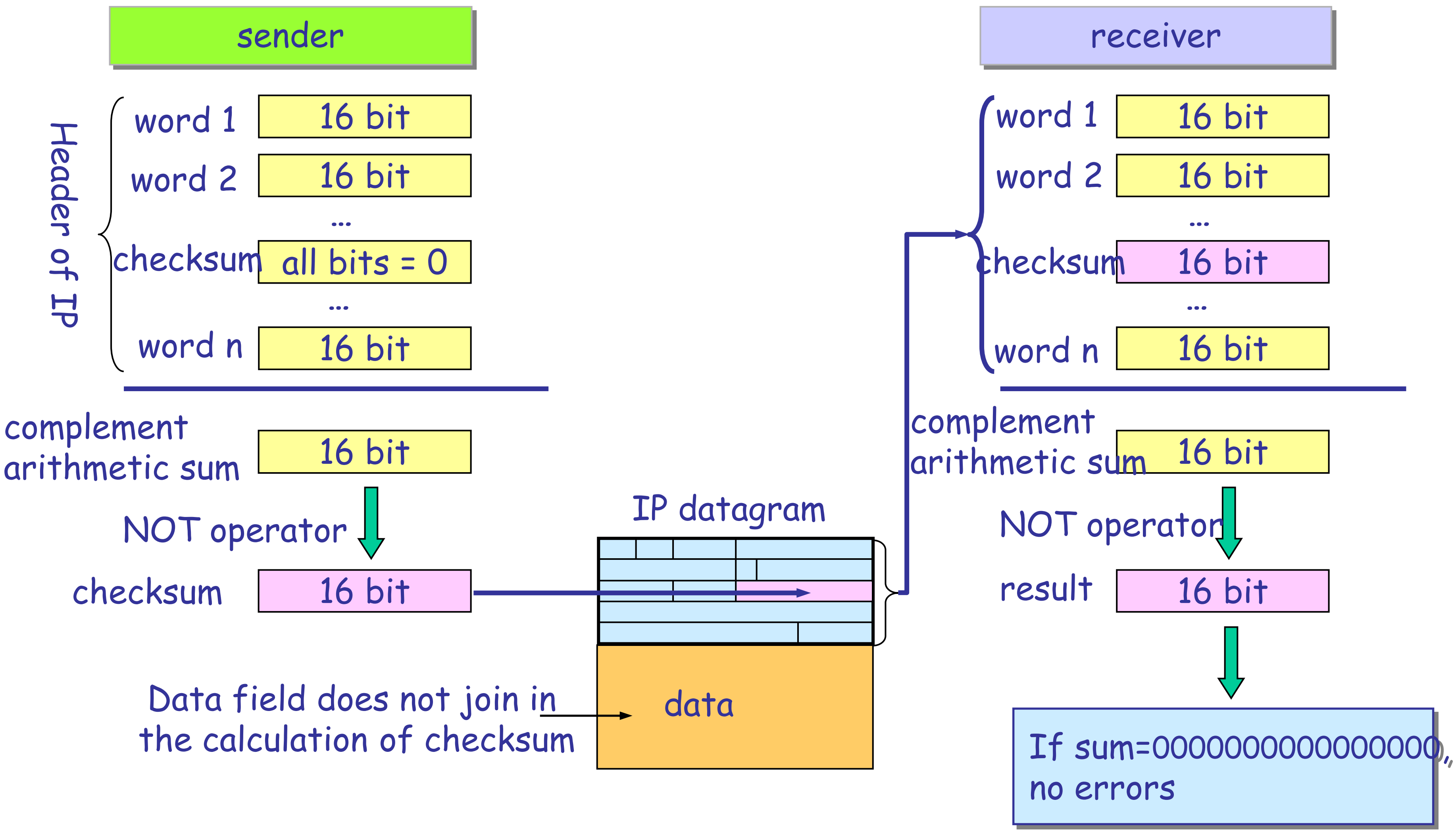


how much overhead?
- 20 bytes of TCP
- 20 bytes of IP
- = 40 bytes + app layer overhead

Upper-layer protocol field



Header checksum



Internet checksum: example

example: add two 16-bit integers

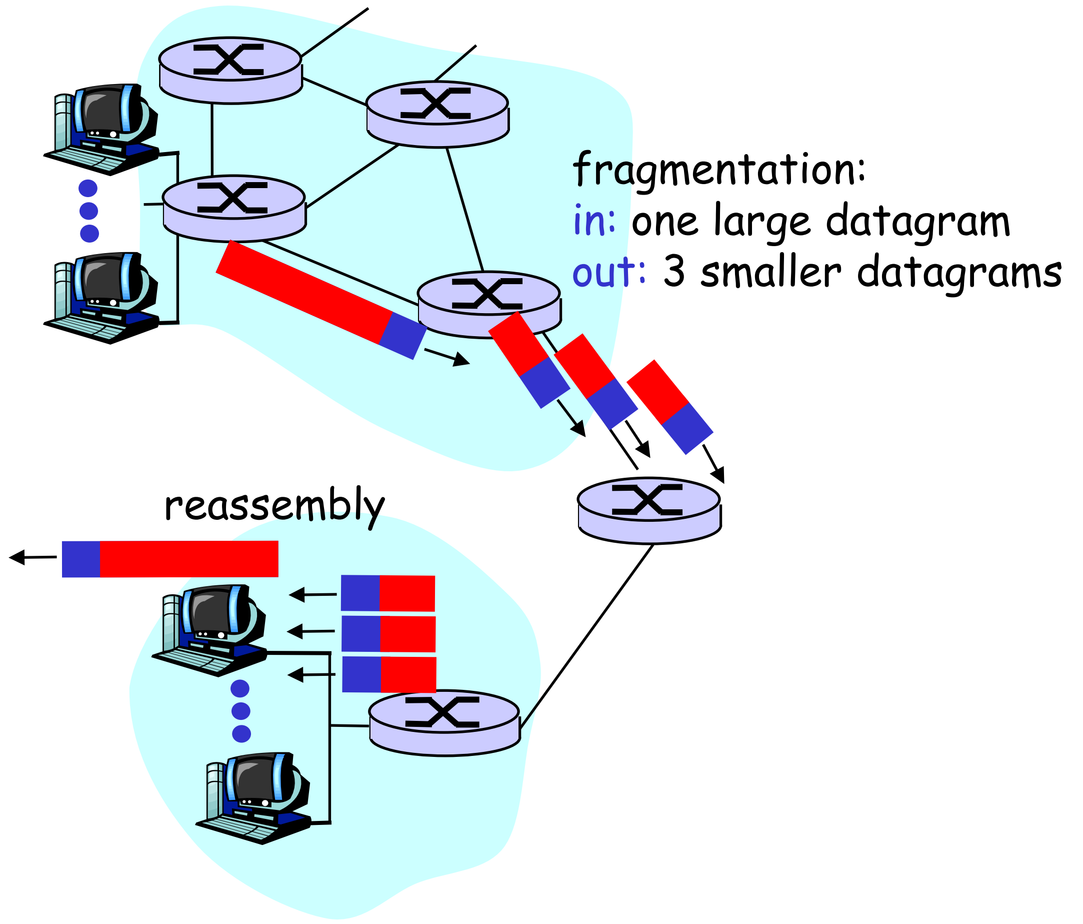
		1	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0
		1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
wraparound	1	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1
sum		1	0	1	1	1	0	1	1	1	0	1	1	1	1	0	0
checksum		0	1	0	0	0	1	0	0	0	1	0	0	0	0	1	1

Note: when adding numbers, a carryout from the most significant bit needs to be added to the result

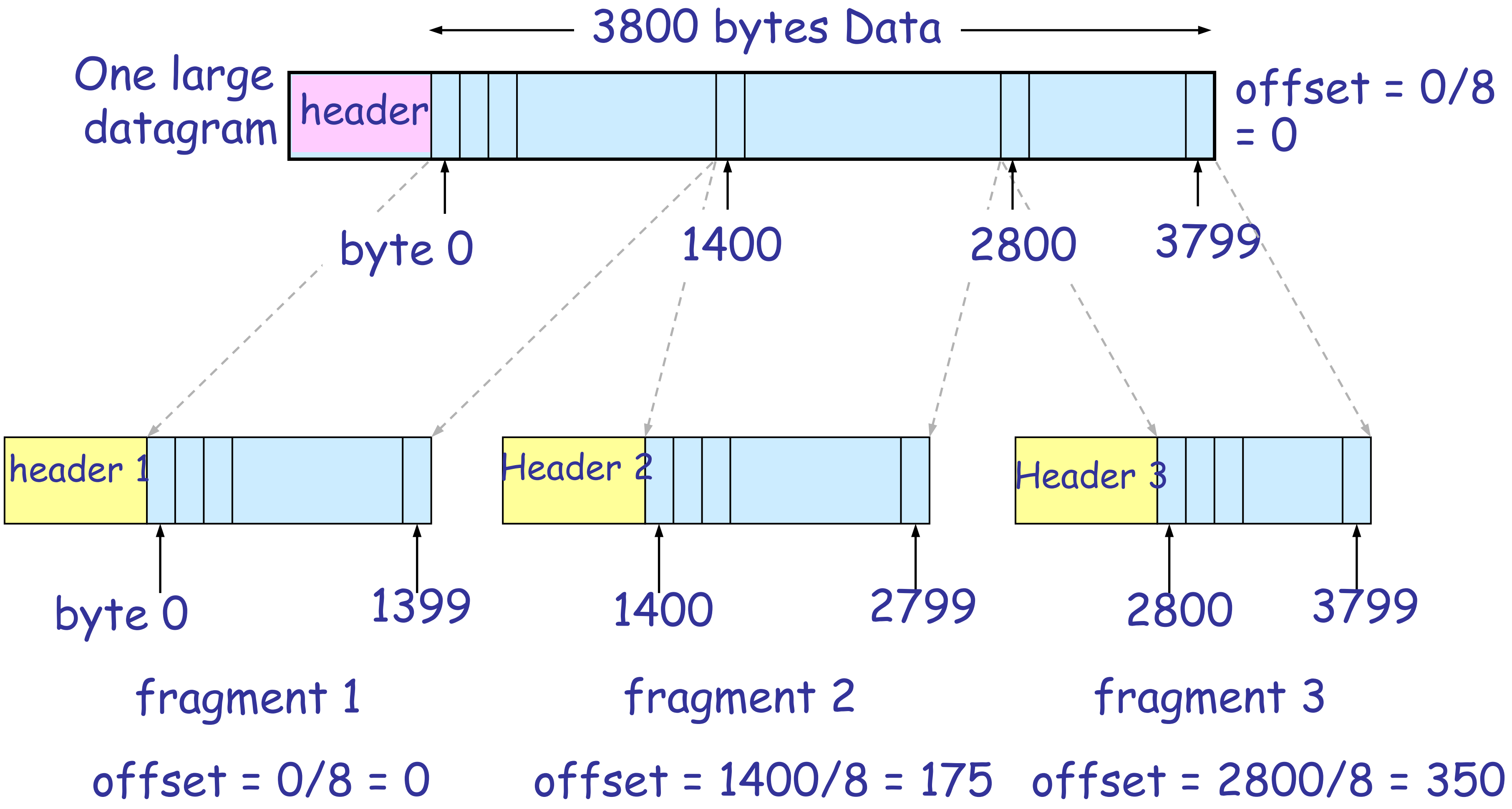
* Check out the online interactive exercises for more examples: http://gaia.cs.umass.edu/kurose_ross/interactive/

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



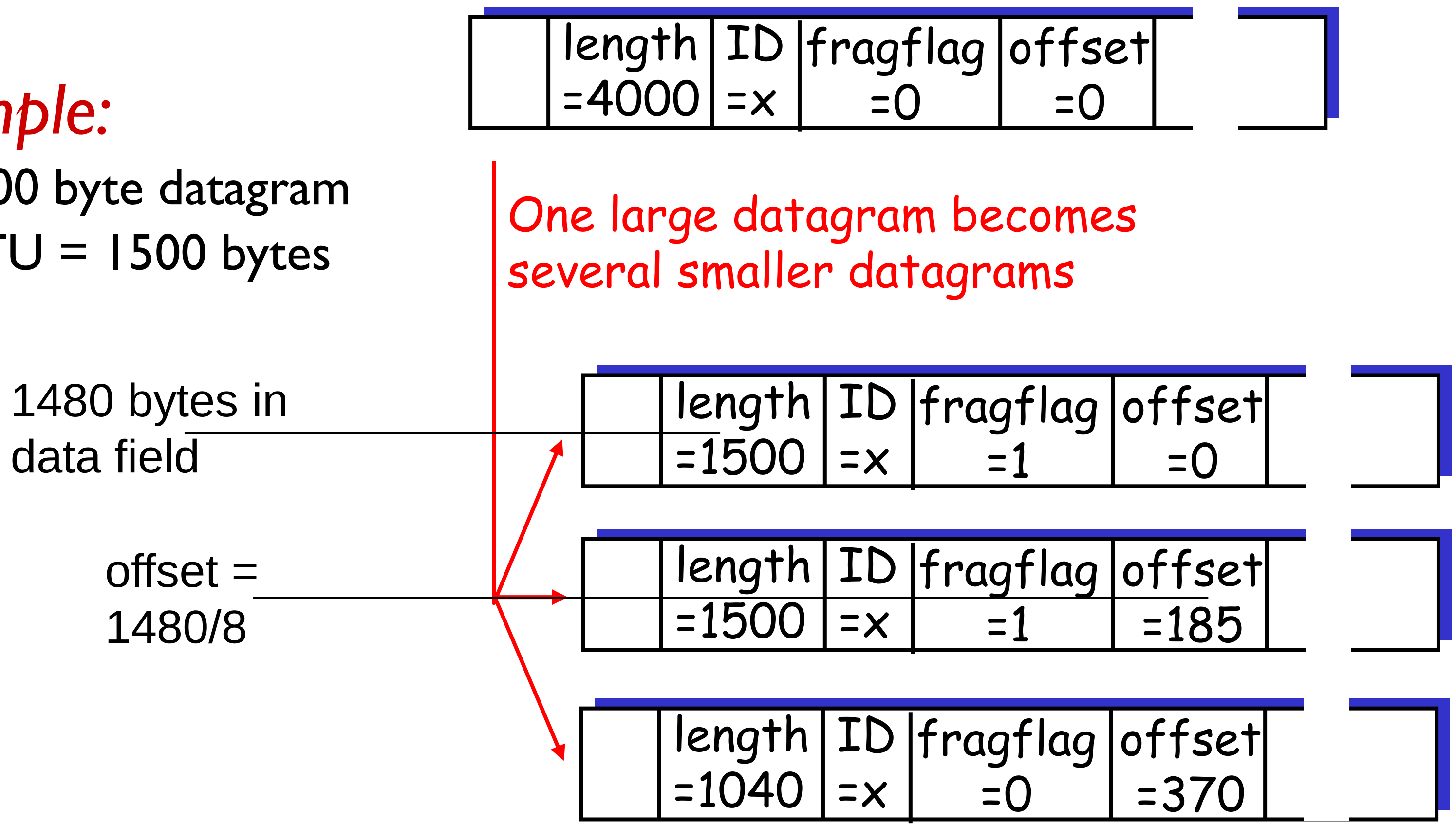
IP Fragmentation



IP Fragmentation and Reassembly

example:

- 4000 byte datagram
- MTU = 1500 bytes



IP datagram

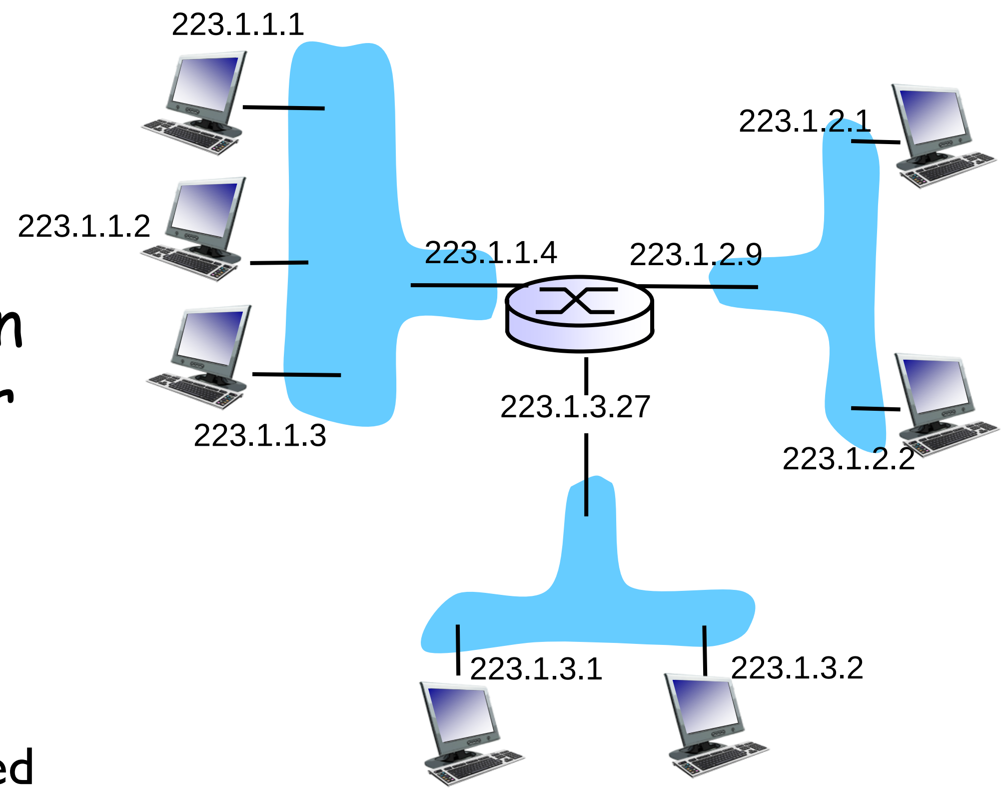
- Now assume a IP datagram is captured. The first 20 bytes are as follows:

  0x45 0x00 0x00 0x3C 0x1A 0x37 0x00
  0x00 0x80 0x01 0x6E 0x31 0xC0 0xA8
  0x01 0xD4 0xD3 0x9B 0x1C 0x41

Please try to analyze the value and meaning of each field in the IP datagram header.

IP addressing: introduction

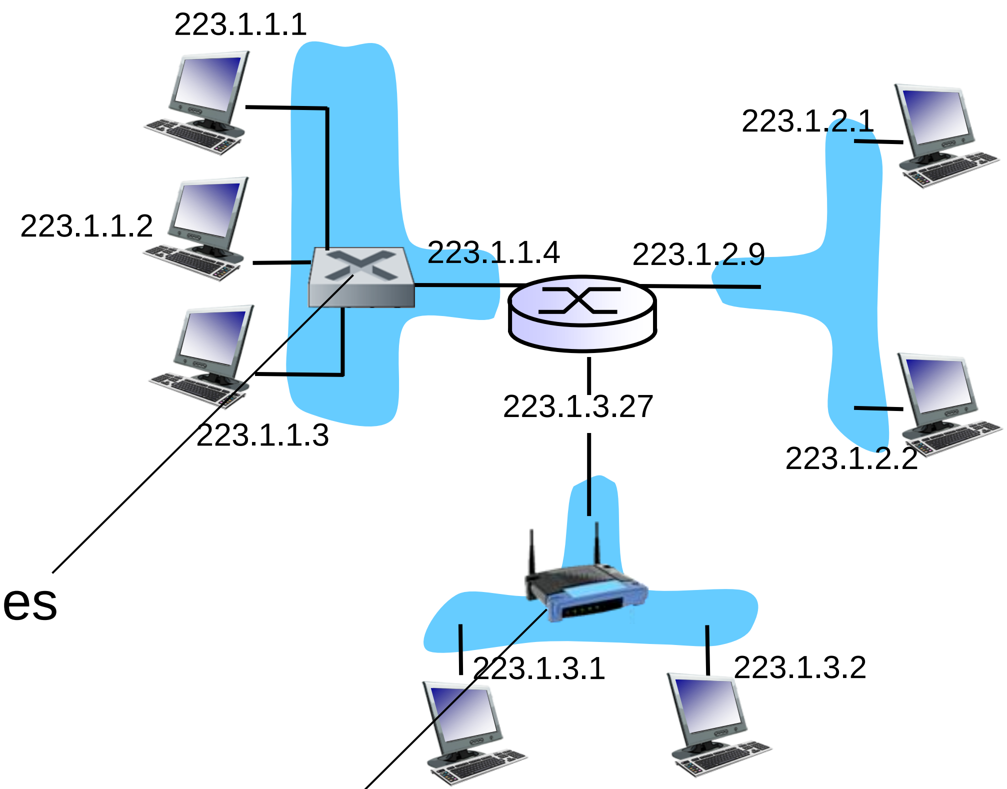
- *IP address:* 32-bit identifier for host, router *interface*
- *interface:* connection between host/router and physical link
  - router's typically have multiple interfaces
  - host typically has one or two interfaces (e.g., wired Ethernet, wireless 802.11)
- *IP addresses associated with each interface*



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

IP addressing: introduction

Q: how are interfaces actually connected?



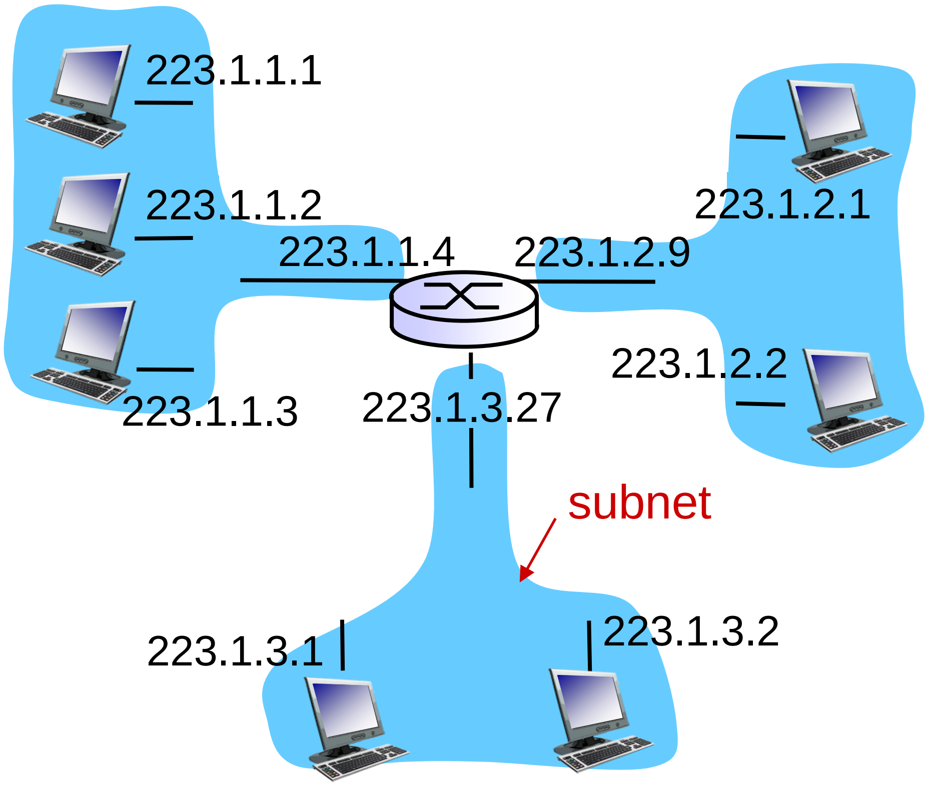
A: wired Ethernet interfaces connected by Ethernet switches

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

A: wireless WiFi interfaces connected by WiFi base station

Subnets

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

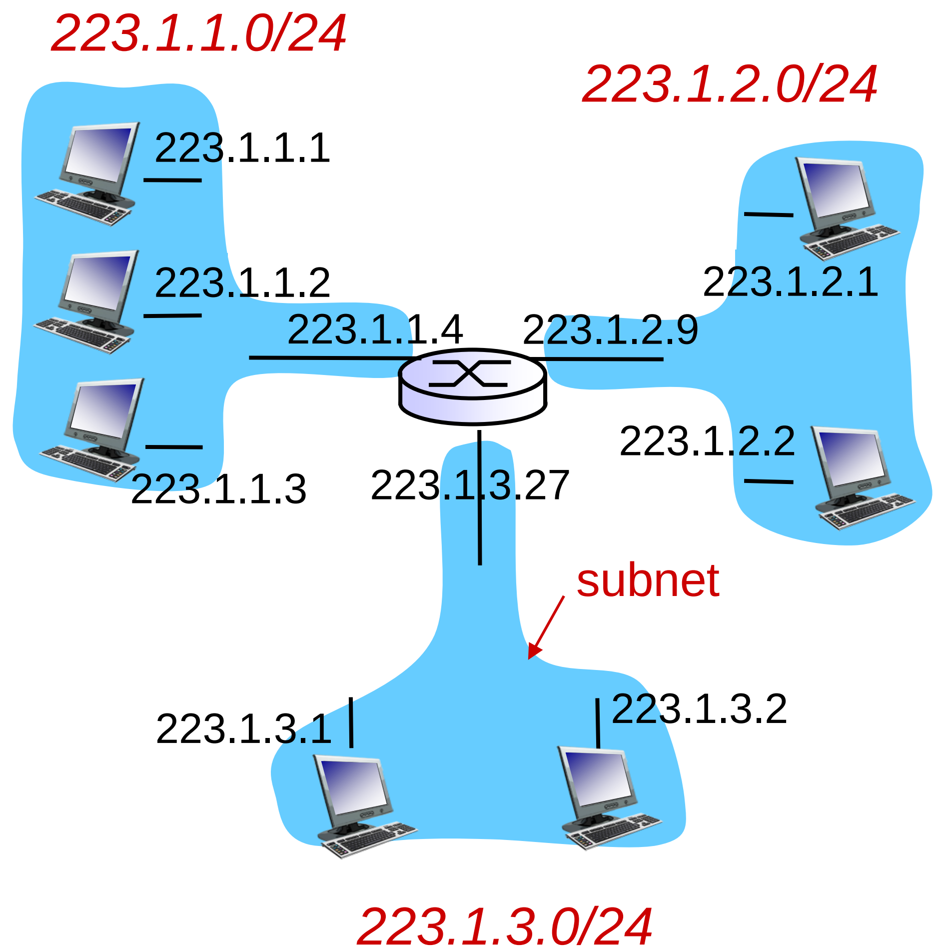


network consisting of 3 subnets

Subnets

recipe

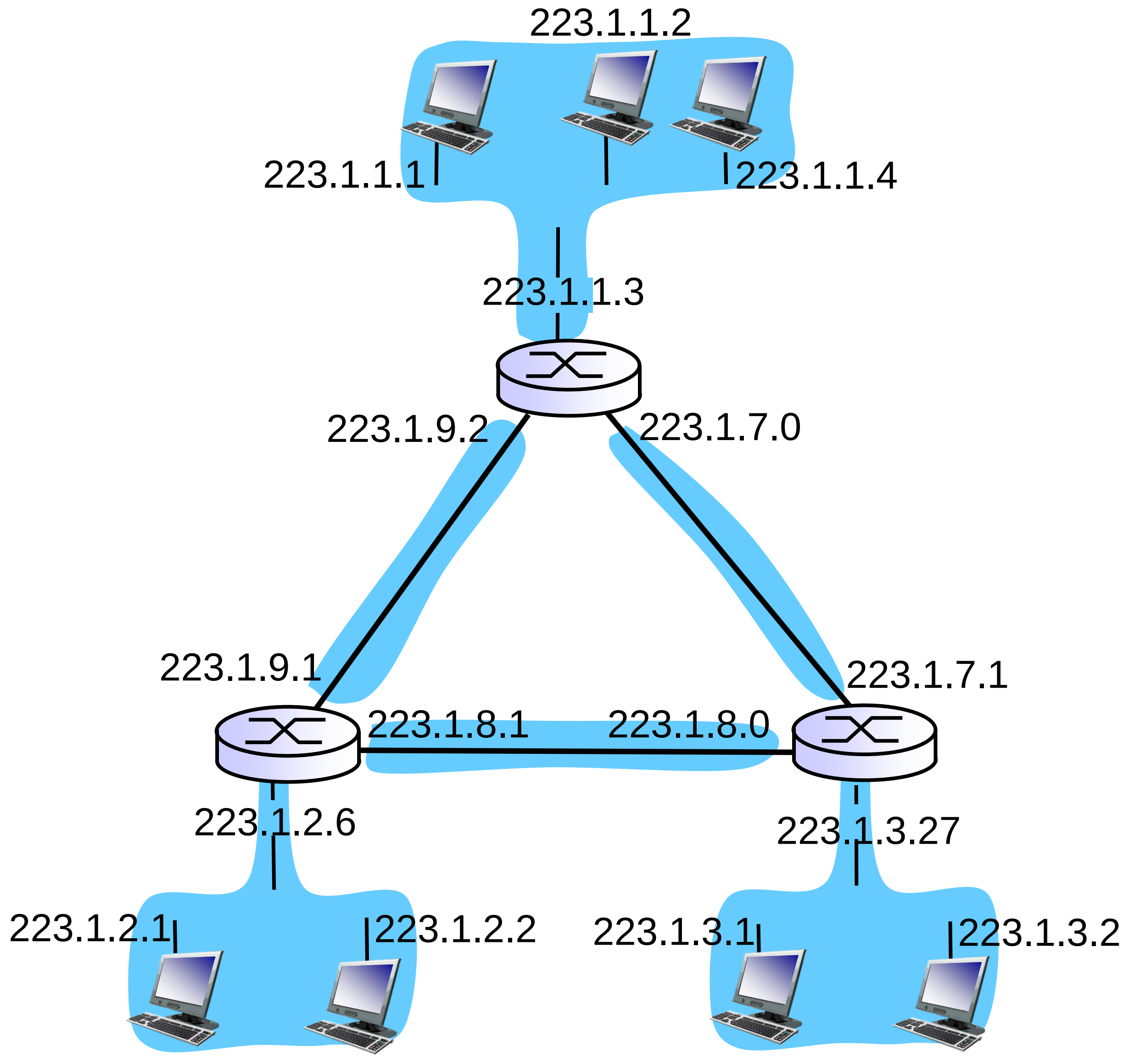
- to determine the subnets, detach each interface from its host or router, creating islands of isolated networks
- each isolated network is called a *subnet*



subnet mask: /24

Subnets

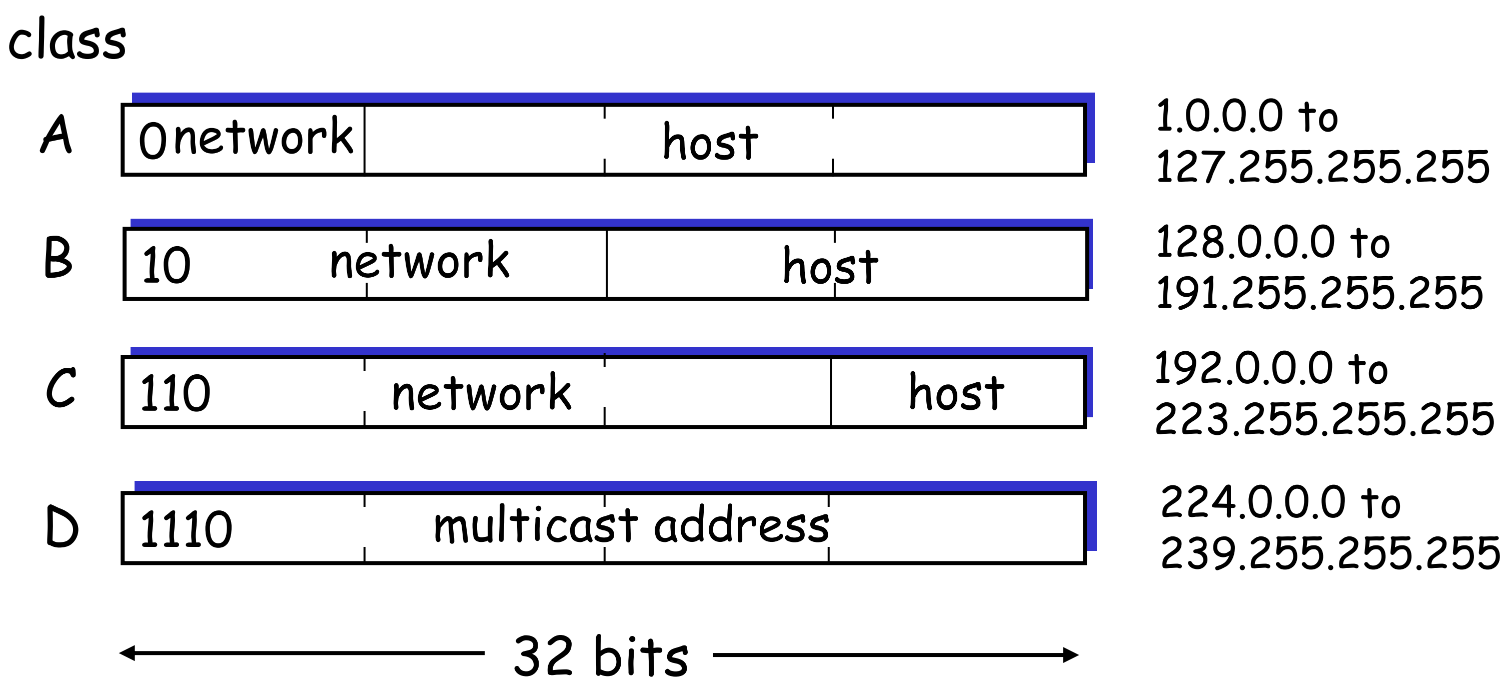
how many?



IP Addresses

given notion of "network", let's re-examine IP addresses:

"class-full" addressing:



网络号	主机号	地址类型	用途
全0	全0	本机	启动时使用
网络ID	全0	网络	标识一个网络
网络ID	全1	直接广播	在指定网上广播
全1	全1	有限广播	在本地网上广播
127	任意	回送	环回测试（127.0.0.1）

IP addressing

- Dotted-decimal notation:193.32.216.9
- classful addressing:
  - inefficient use of address space, address space exhaustion
  - e.g., class B net allocated enough addresses for 65K hosts, even if only 2K hosts in that network
- Subnetting and subnet mask

subnet part

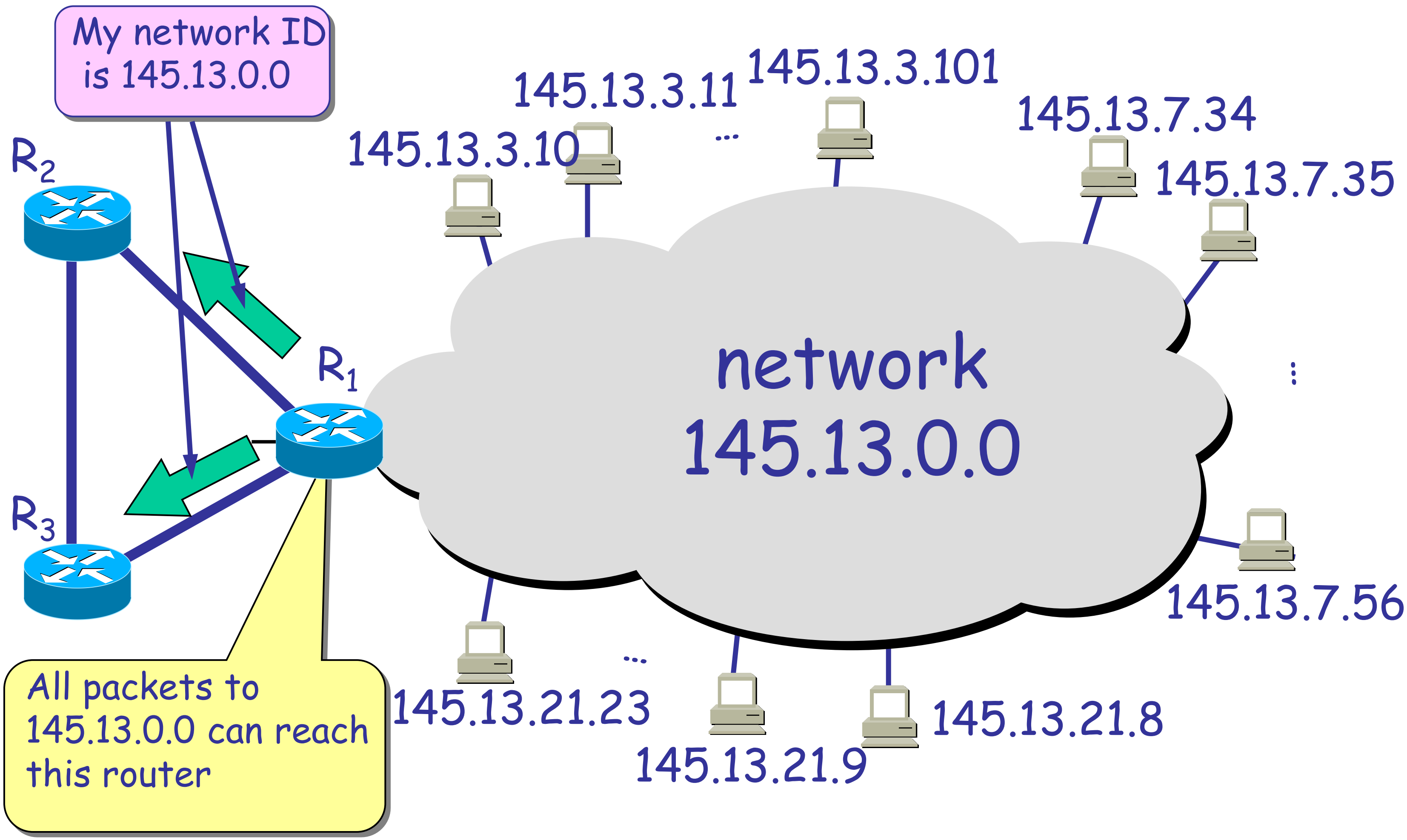
host part

network part

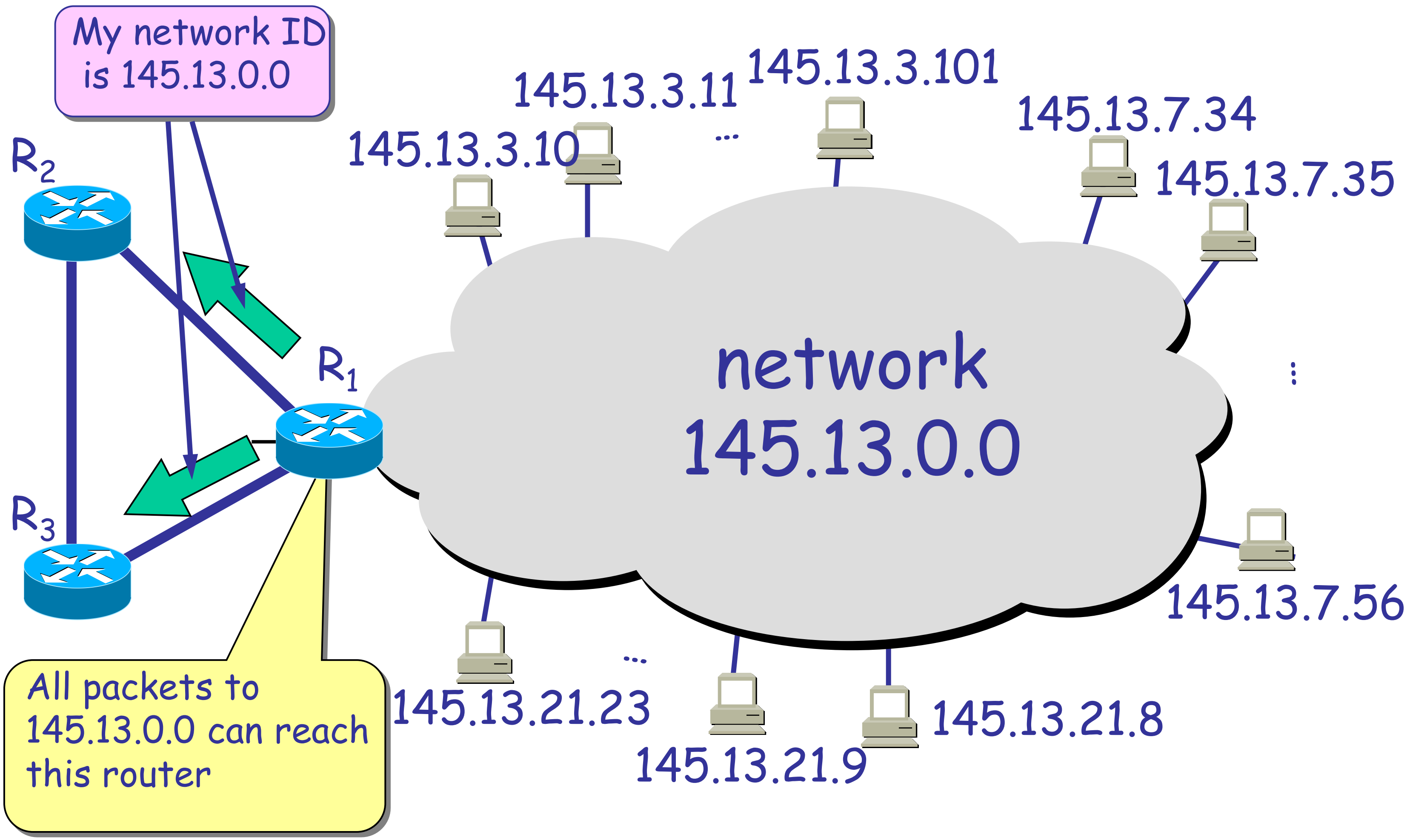
11001000 00010111 00010000 01000000

200.23.16.64/27

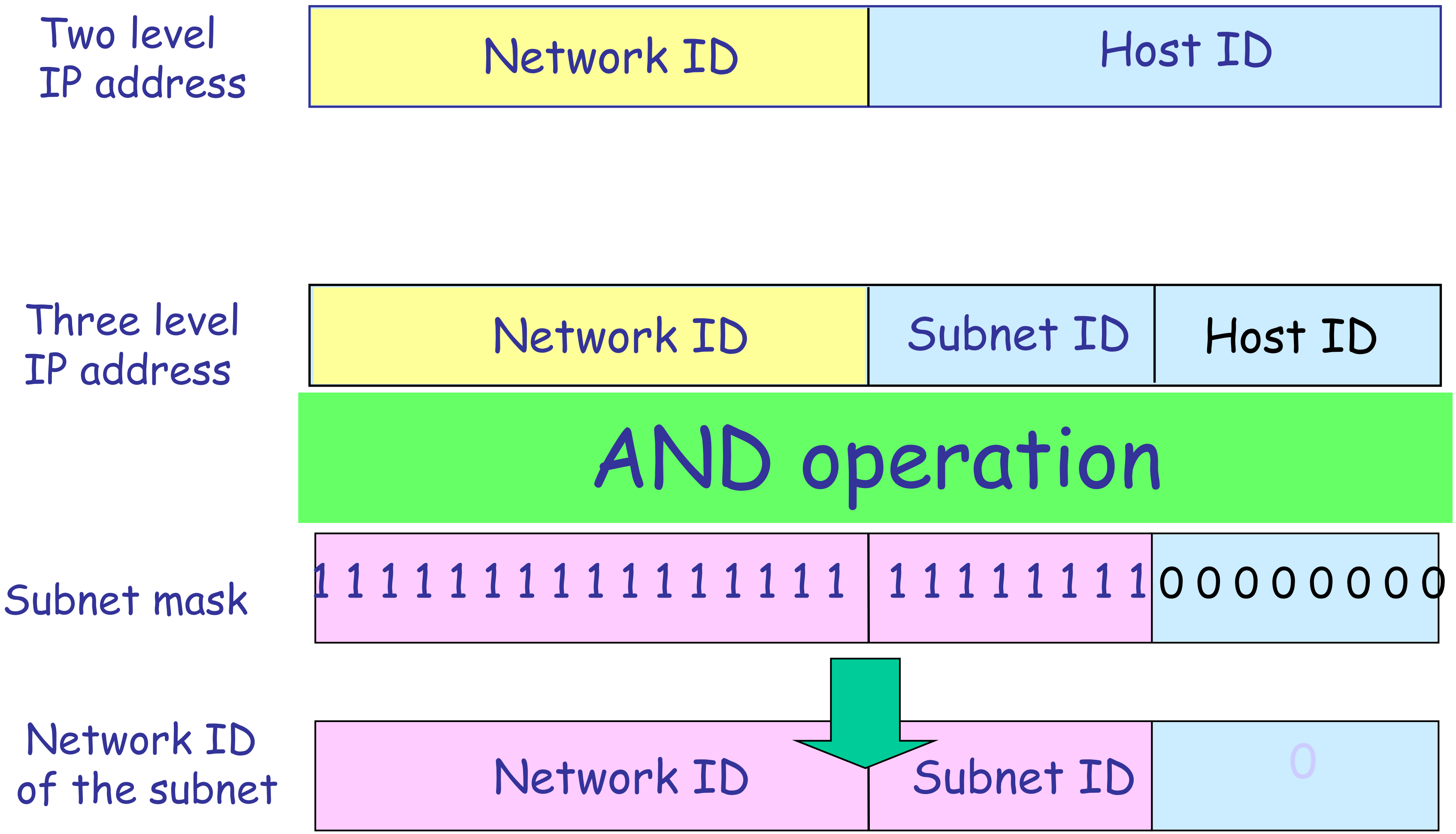
A class B network without subnet: 145.13.0.0



A network for others, the class B with 3 subnet



Subnet mask



Two level IP address
Network ID
Host ID
Three level IP address
Network ID
Subnet ID
Host ID
AND operation
Subnet mask
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0
Network ID of the subnet
Network ID
Subnet ID
0

IP addressing: CIDR

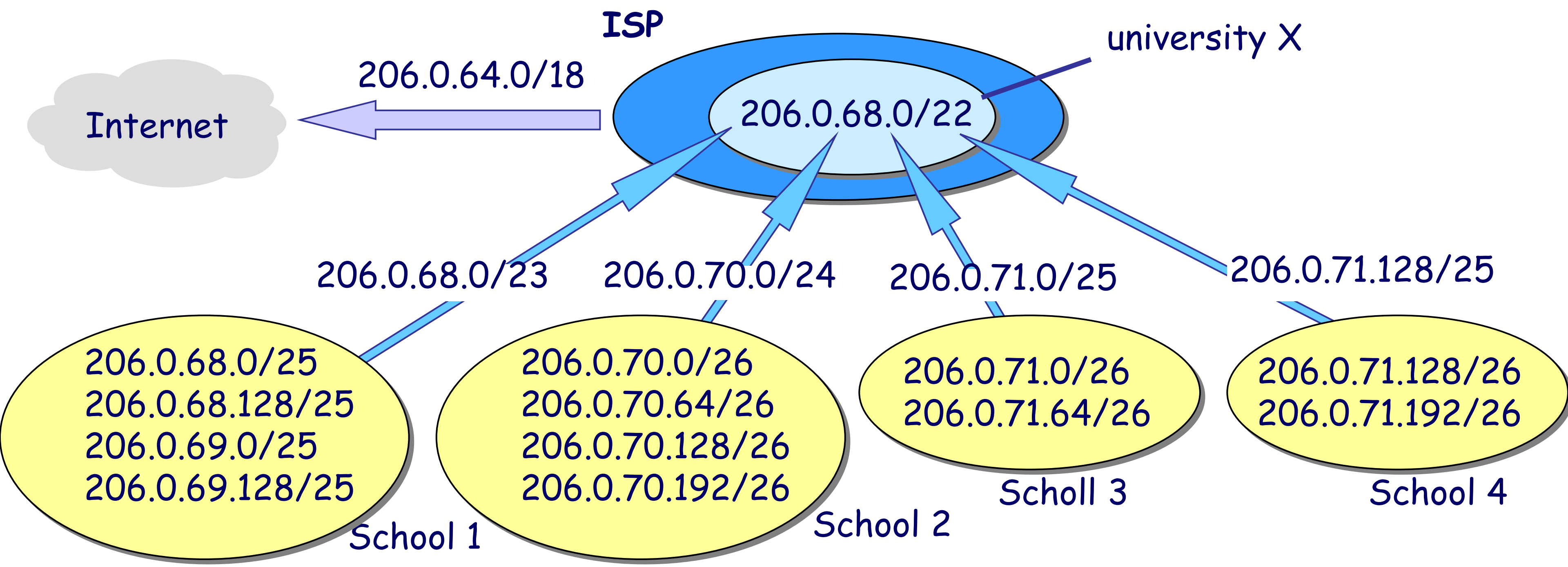
- CIDR: Classless InterDomain Routing
  - Two level address: only two parts
  - network portion of address of arbitrary length
  - address format: a.b.c.d/x, where x is # bits in network portion of address
  - Network portion is often called prefix

network part | host part

11001000 00010111 00010000 00000000

200.23.16.0/23

CIDR example



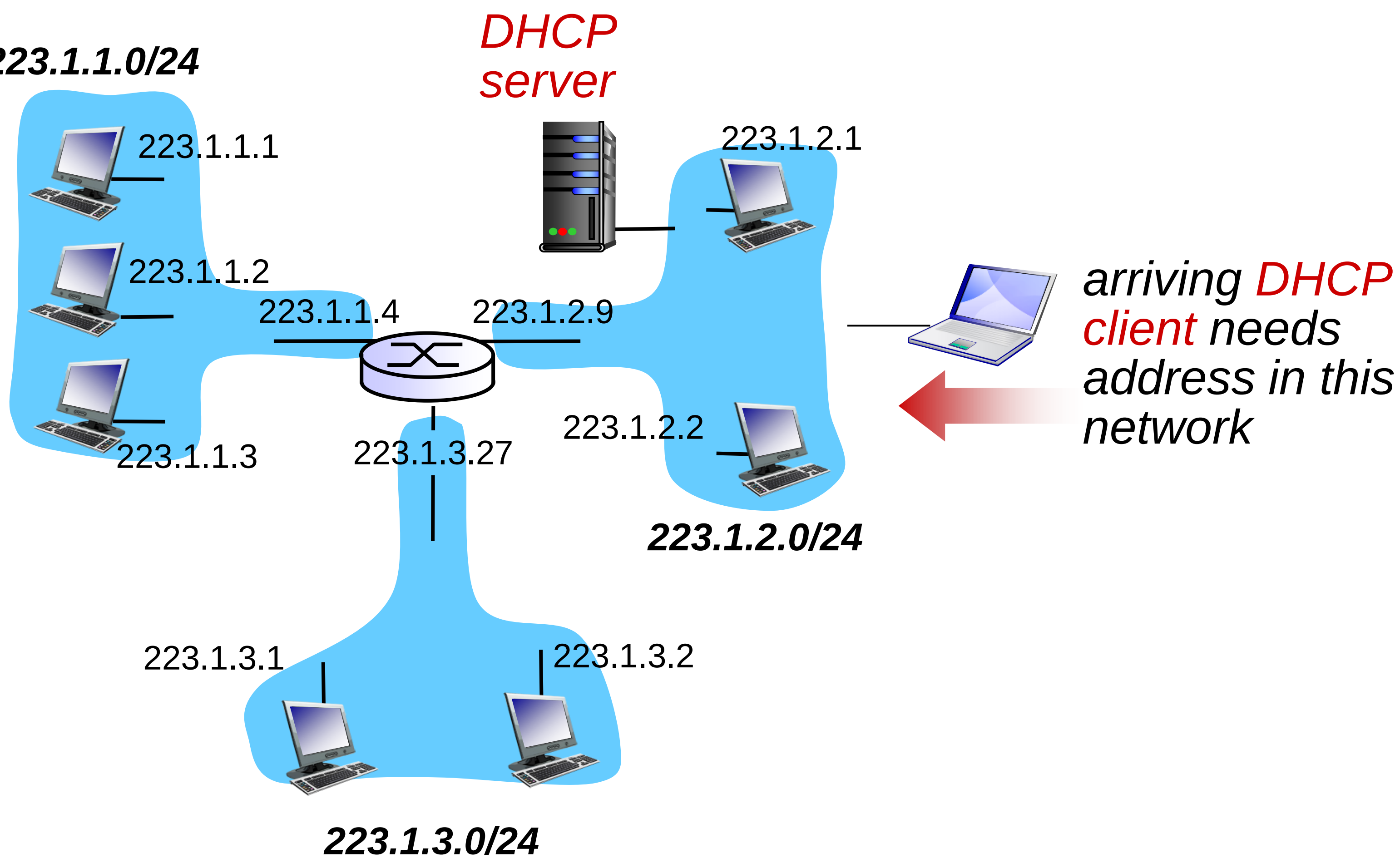
organizatio	address block	binary	number
ISP	206.0.64.0/18	11001110.00000000.01*	16384
Uni. X	206.0.68.0/22	11001110.00000000.010001*	1024
sch. 1	206.0.68.0/23	11001110.00000000.0100010*	512
Sch. 2	206.0.70.0/24	11001110.00000000.01000110.*	256
Sch.3	206.0.71.0/25	11001110.00000000.01000111.0*	128
Sch. 4	206.0.71.128/25	11001110.00000000.01000111.1*	128

IP addresses: how to get one?

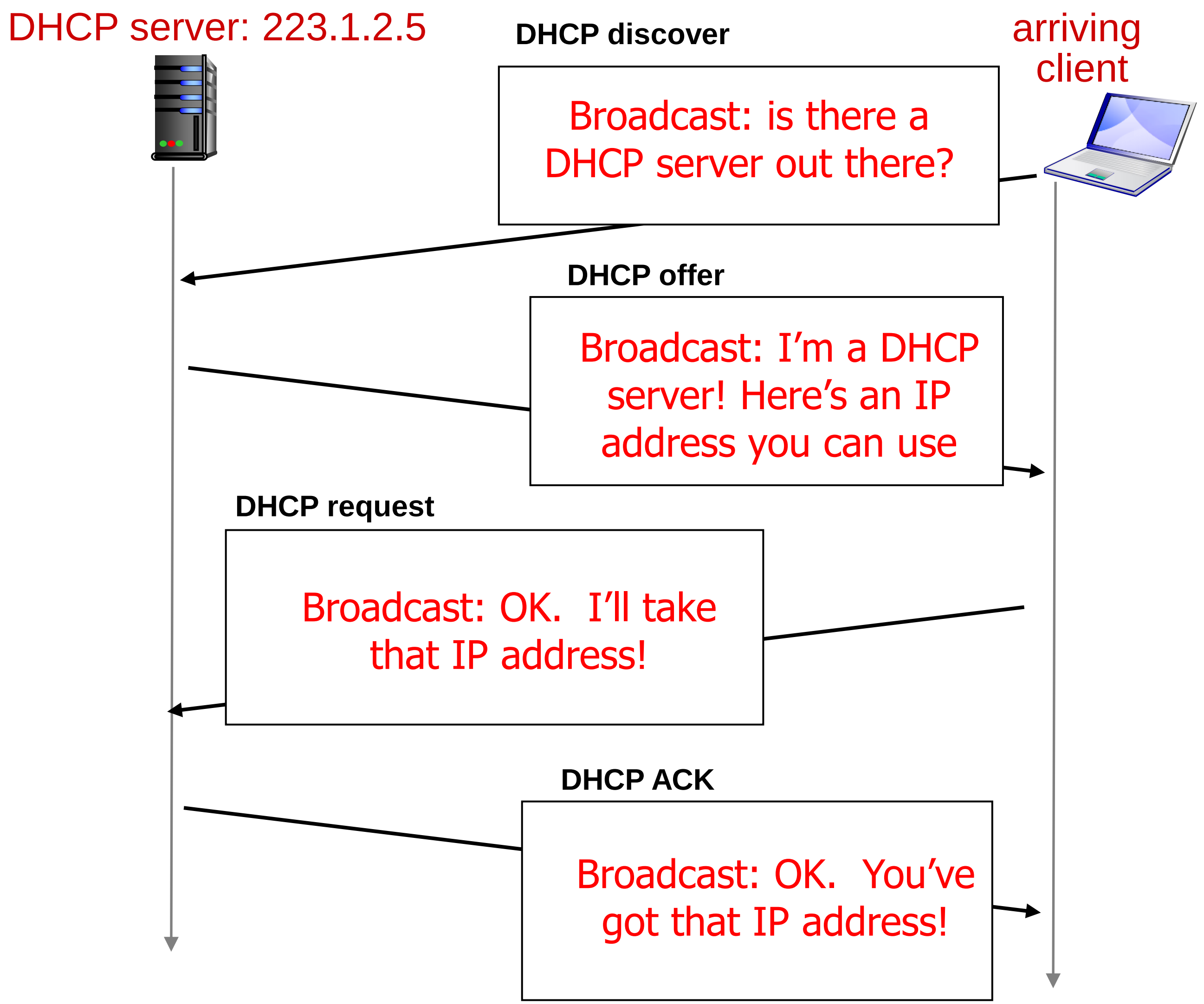
Hosts (host portion):

- hard-coded by system admin in a file
- DHCP: Dynamic Host Configuration Protocol: dynamically get address: "plug-and-play"
  - host broadcasts "DHCP discover" msg
  - DHCP server responds with "DHCP offer" msg
  - host requests IP address: "DHCP request" msg
  - DHCP server sends address: "DHCP ack" msg

DHCP client-server scenario



DHCP client-server scenario



IP addresses: how to get one?

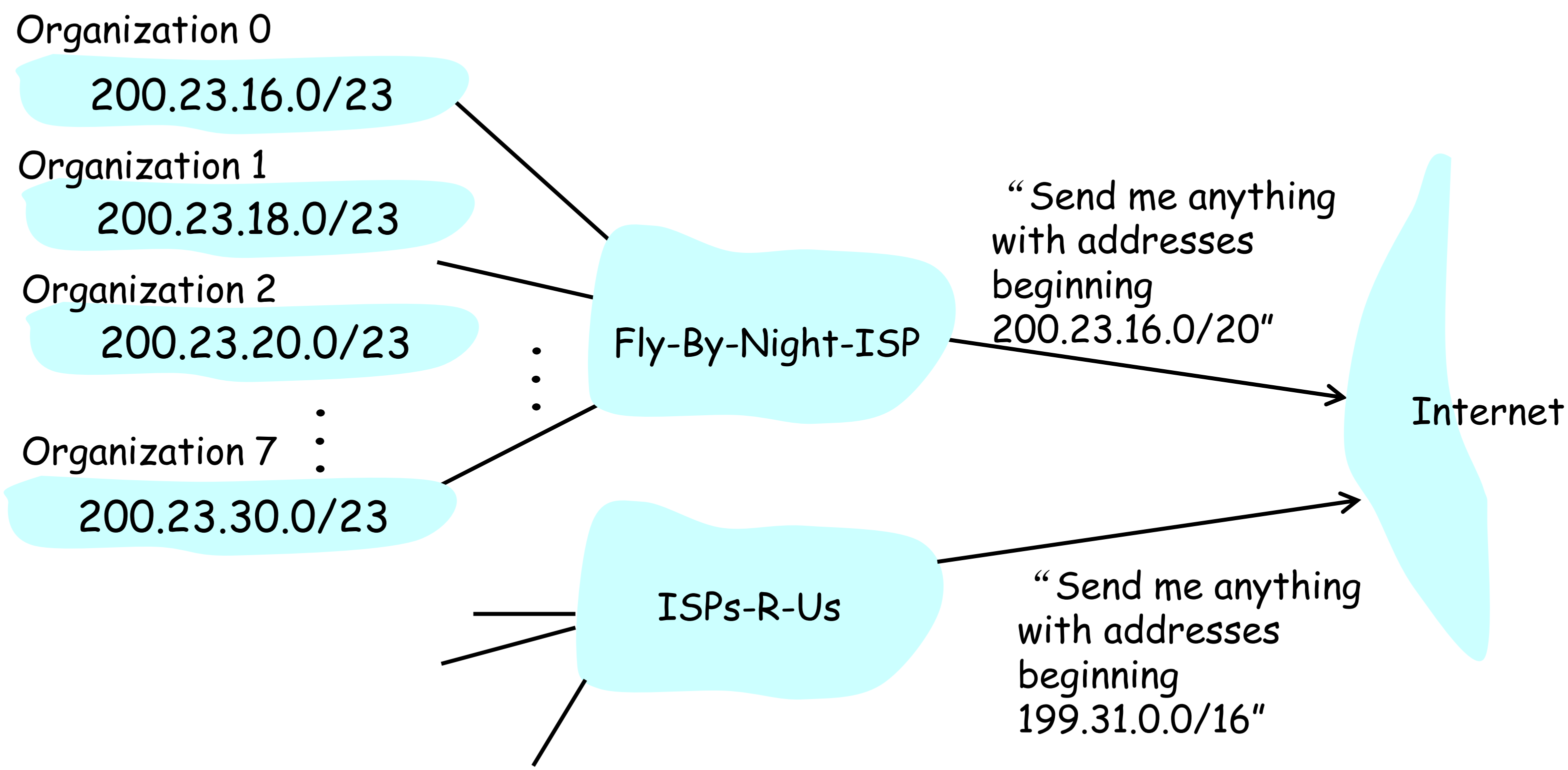
Network (network portion):

- get allocated portion of ISP's address space:

ISP's block	11001000 00010111 0001~~0000~~ 00000000		200.23.16.0/20
Organization 0	11001000 00010111 0001000~~0~~ 00000000		200.23.16.0/23
Organization 1	11001000 00010111 0001001~~0~~ 00000000		200.23.18.0/23
Organization 2	11001000 00010111 0001010~~0~~ 00000000		200.23.20.0/23
...	.....	....	....
Organization 7	11001000 00010111 0001111~~0~~ 00000000		200.23.30.0/23

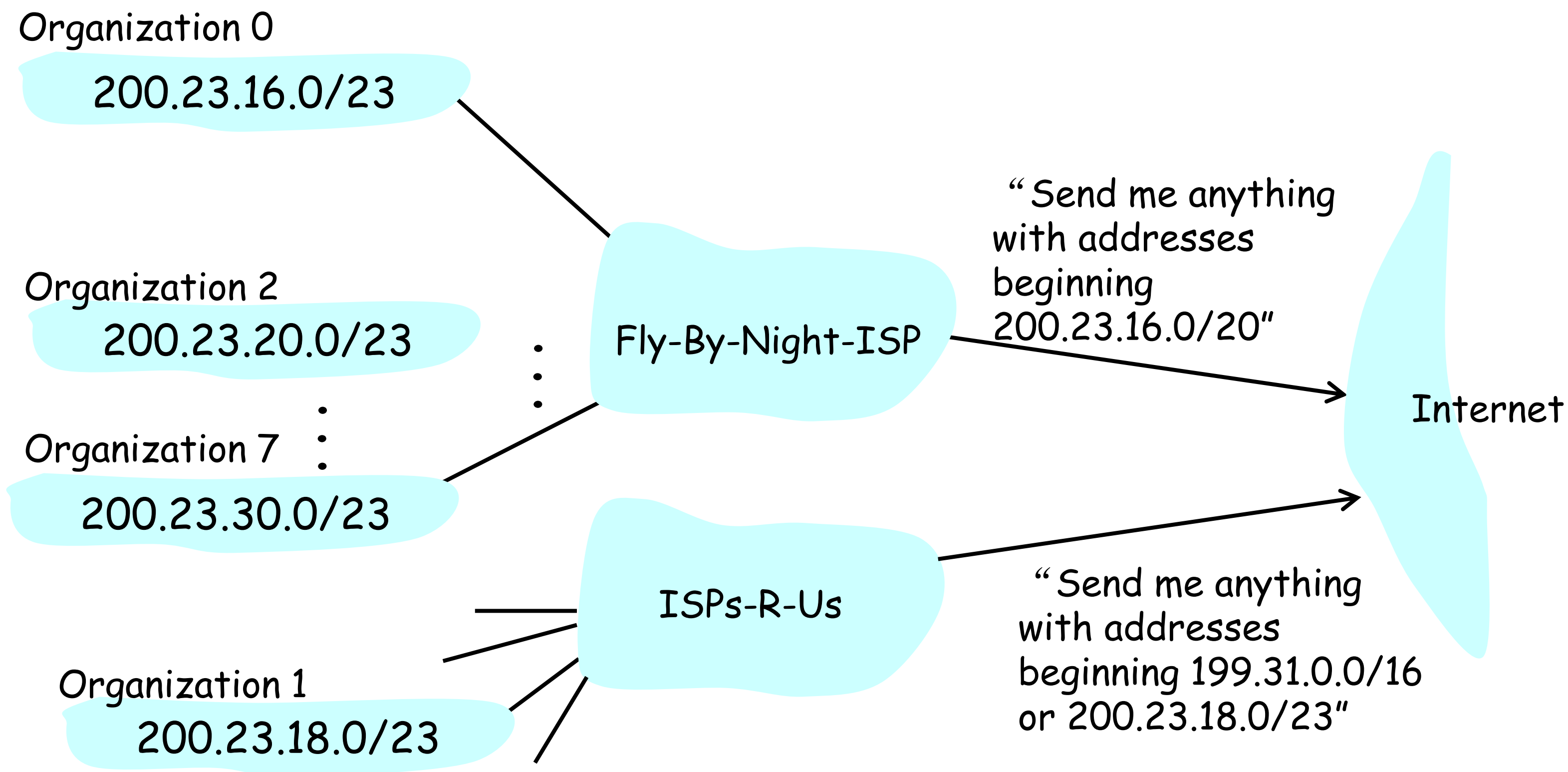
Hierarchical addressing: route aggregation

Hierarchical addressing allows efficient advertisement of routing information:



Hierarchical addressing: more specific routes

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



IP addressing: the last word...

Q: How does an ISP get block of addresses?

A: ICANN: Internet Corporation for Assigned Names and Numbers

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

Exercise-2

1. In subnet 192.168.4.0/30, the maximum number of hosts that can accept IP packets with destination address 192.168.4.3 is

A.0 B.1 C.2 D.4

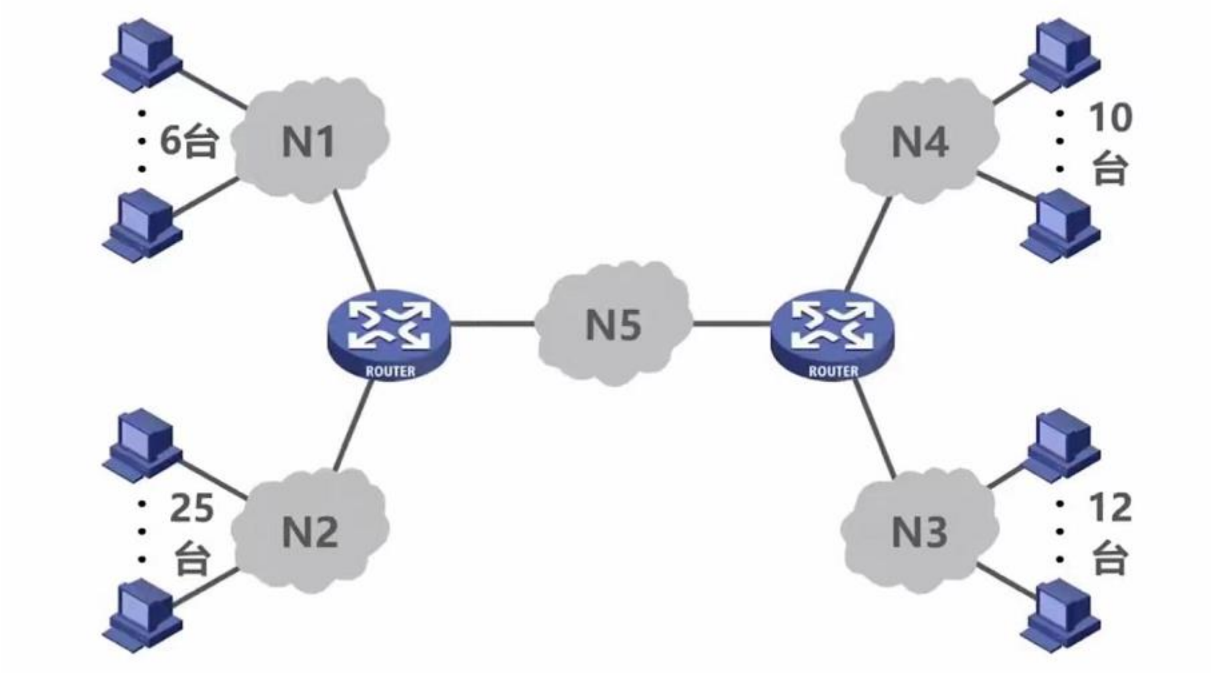
2. There are four routing table entries with the same forwarding interface in a routing table, and their destination network addresses are 35.230.32.0/21, 35.230.40.0/21, 35.230.48.0/21 and 35.230.56.0/21 respectively. After aggregating the four routes, the destination network address is

A.35.230.0.0/19 B. 35.230.0.0/20

C. 35.230.32.0/19 D. 35.230.32.0/20

Exercises-3

Assuming that the applied IP address block is 218.75.230.0/24, please assign IP addresses to the devices in the figure below using fixed length and variable length subnet masks respectively.



Exercises-3

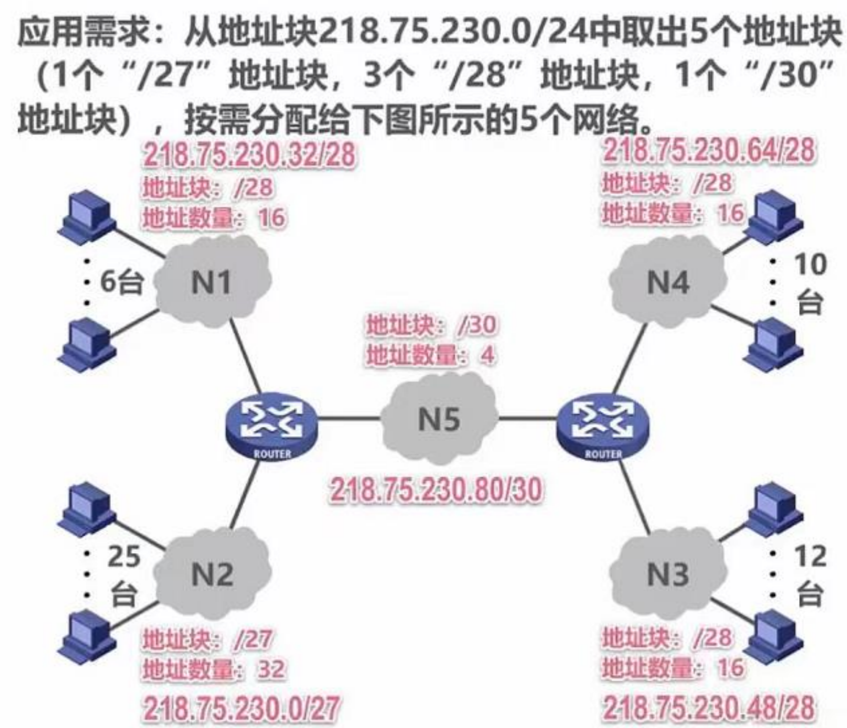
Assuming that the applied IP address block is 218.75.230.0/24, please assign IP addresses to the devices in the figure below using fixed length and variable length subnet masks respectively.

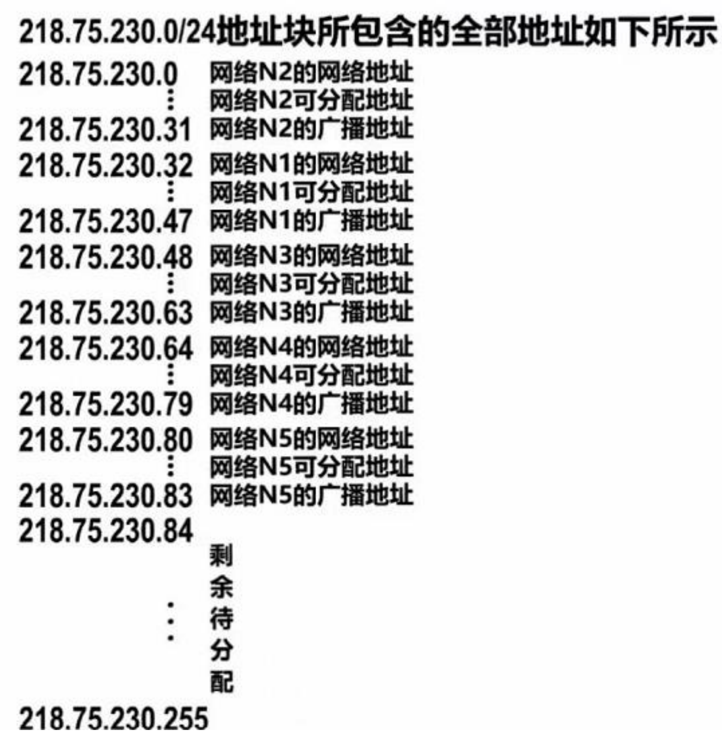
使用子网掩码255.255.255.224 对C类网络218.75.230.0进行子网划分的细节			
子网	网络地址	广播地址	该子网可分配的地址
1	218.75.230.0	218.75.230.31	218.75.230.1 ~ 218.75.230.30
2	218.75.230.32	218.75.230.63	218.75.230.33 ~ 218.75.230.62
3	218.75.230.64	218.75.230.95	218.75.230.65 ~ 218.75.230.94
4	218.75.230.96	218.75.230.127	218.75.230.97 ~ 218.75.230.126
5	218.75.230.128	218.75.230.159	218.75.230.129 ~ 218.75.230.158
6	218.75.230.160	218.75.230.191	218.75.230.161 ~ 218.75.230.190
7	218.75.230.192	218.75.230.223	218.75.230.193 ~ 218.75.230.222
8	218.75.230.224	218.75.230.255	218.75.230.225 ~ 218.75.230.254

从子网1~8中任选5个分配给左图中的N1~N5。

Exercises-3

Assuming that the applied IP address block is 218.75.230.0/24, please assign IP addresses to the devices in the figure below using fixed length and variable length subnet masks respectively.

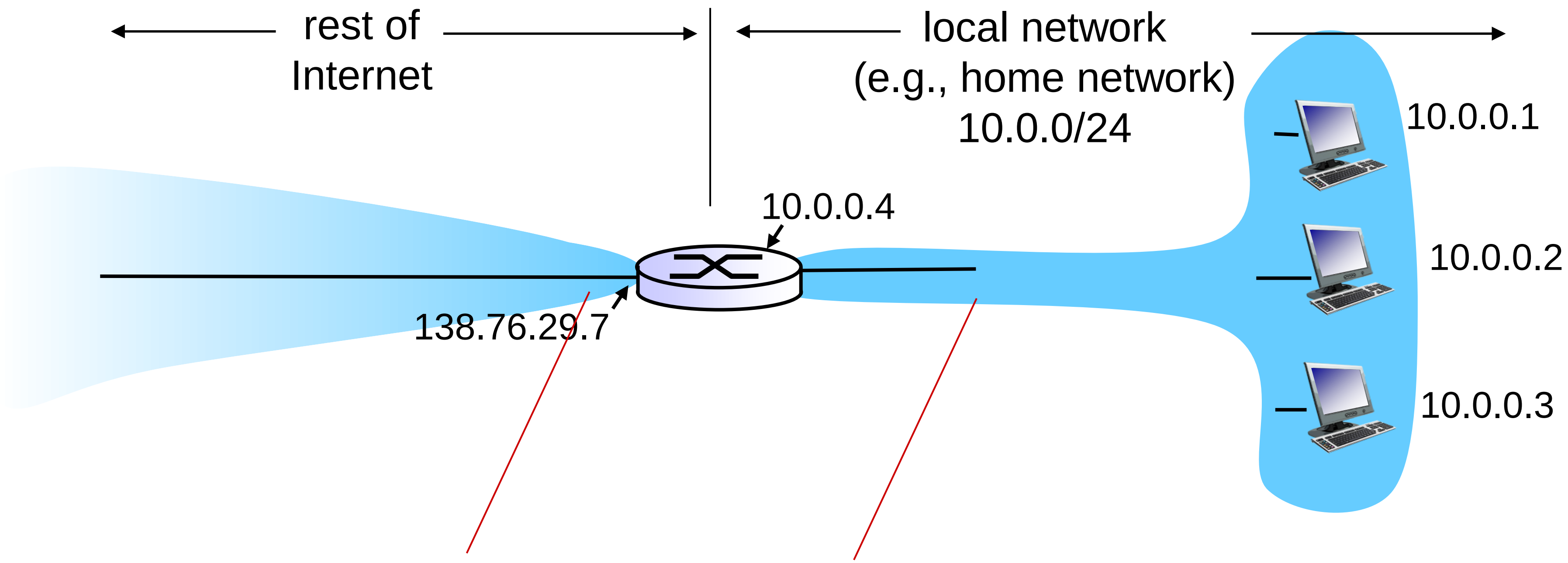




218.75.230.0/24地址块所包含的全部地址如下所示：

218.75.230.0	网络N2的网络地址
⋮	网络N2可分配地址
218.75.230.31	网络N2的广播地址
218.75.230.32	网络N1的网络地址
⋮	网络N1可分配地址
218.75.230.47	网络N1的广播地址
218.75.230.48	网络N3的网络地址
⋮	网络N3可分配地址
218.75.230.63	网络N3的广播地址
218.75.230.64	网络N4的网络地址
⋮	网络N4可分配地址
218.75.230.79	网络N4的广播地址
218.75.230.80	网络N5的网络地址
⋮	网络N5可分配地址
218.75.230.83	网络N5的广播地址
218.75.230.84	
⋮	剩余待分配
218.75.230.255	

NAT: Network Address Translation



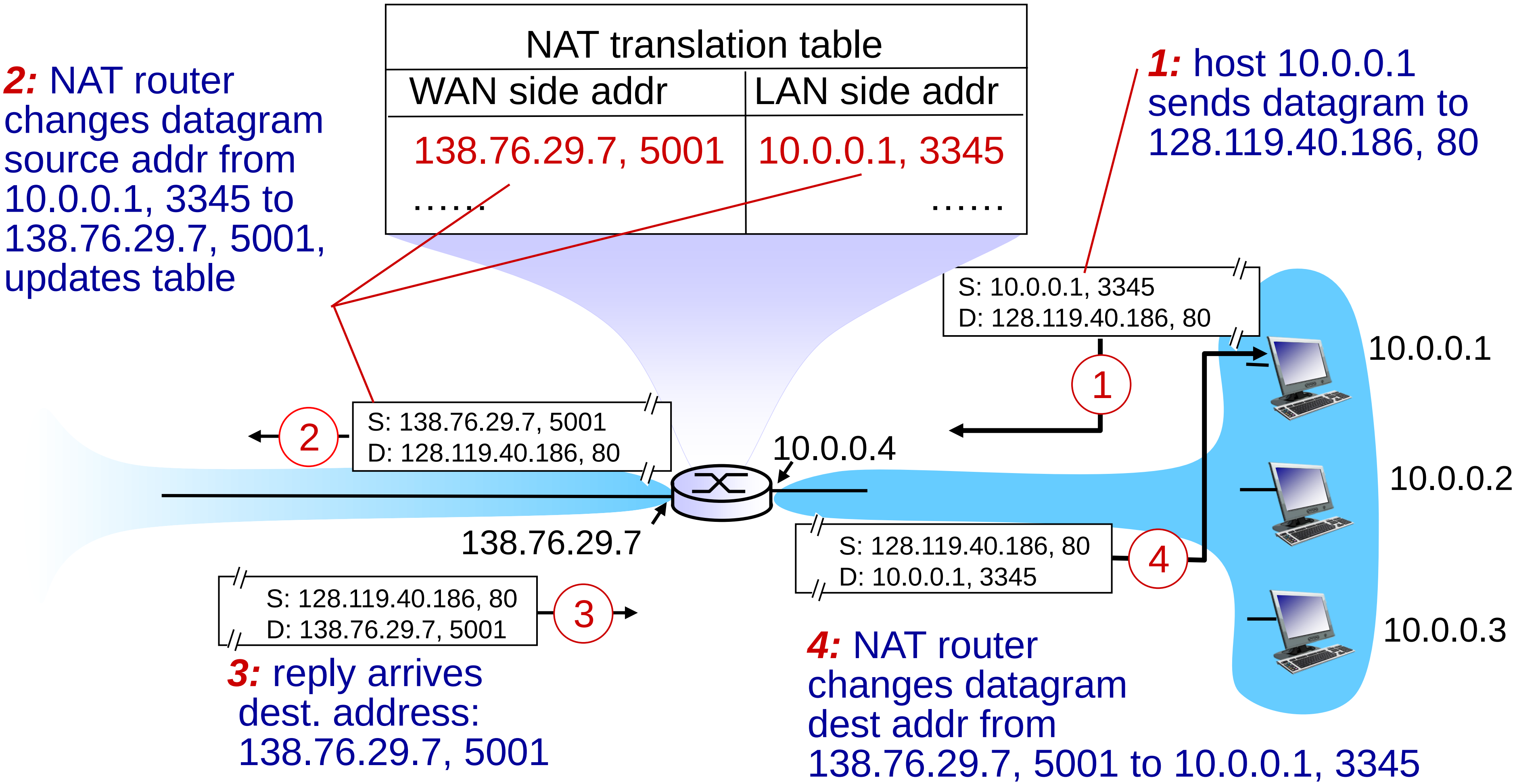
all datagrams *leaving* local network have *same* single source NAT IP address: 138.76.29.7,different source port numbers

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
- Implementation: NAT router must:
  - Outgoing datagrams: replace (source IP address, port #) to (NAT IP address, new port #)
  - Remember (in NAT translation table) every translation pair
  - Incoming datagrams: replace (NAT IP address, new port #) with correspongding (source IP address, port #) stored in NAT table

NAT: Network Address Translation



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

NAT: Network Address Translation

- Outside node cannot initiate the communication

- Reserved addresses:
  - 10.0.0.0 – 10.255.255.255
  - 172.16.0.0 – 172.31.255.255
  - 192.168.0.0 – 192.168.255.255

LAN addresses and ARP

32-bit IP address:

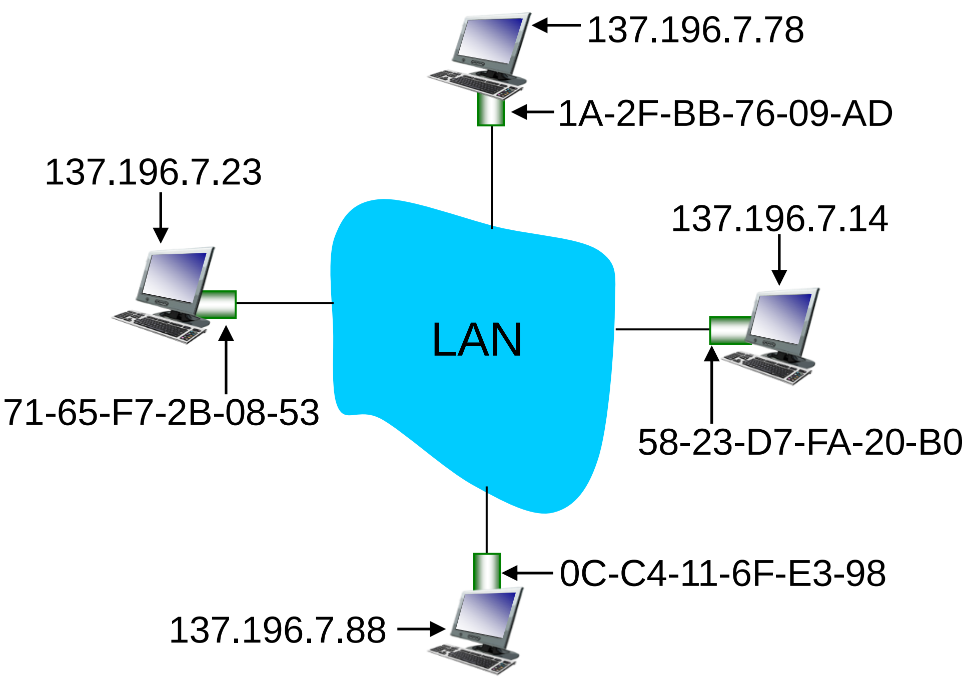
- *network-layer* address
- used to get datagram to destination network (recall IP network definition)

LAN (or MAC or physical) address:

- used to get datagram from one interface to another physically-connected interface (same network)
- 48 bit MAC address (for most LANs) burned in the adapter ROM
- e.g.: 1A-2F-BB-76-09-AD

ARP: Address Resolution Protocol

Question: how to determine MAC address of B given B's IP address?



- Each IP node (Host, Router) on LAN has ARP module, table
- ARP 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)

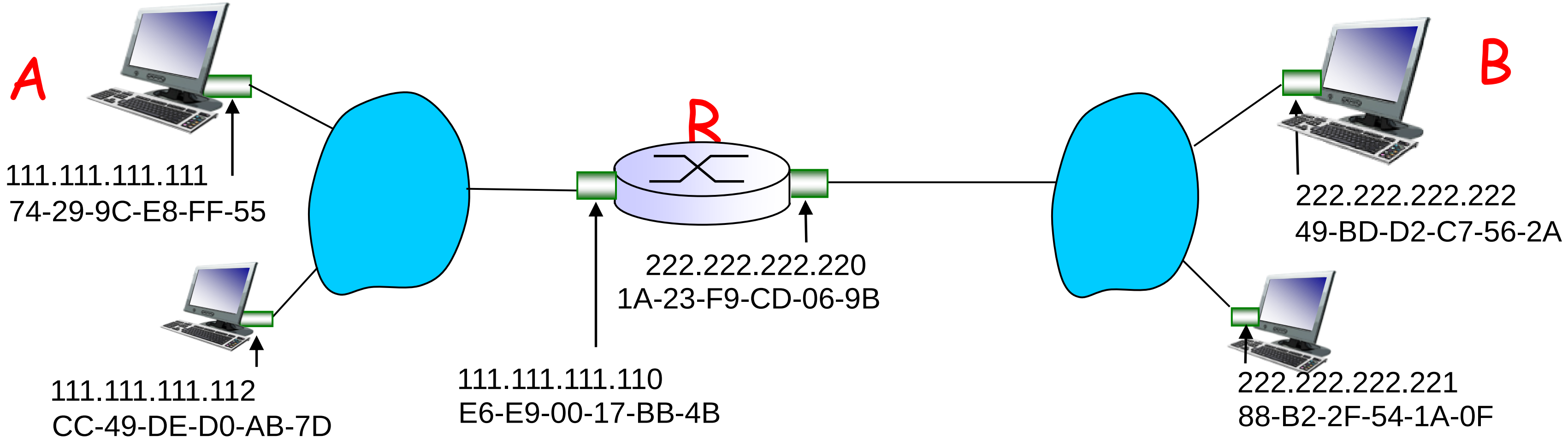
ARP protocol: same LAN

- A wants to send datagram to B
  - B's MAC address not in A's ARP table.
- A broadcasts ARP query packet, containing B's IP address
  - destination MAC address = FF-FF-FF-FF-FF-FF
  - all nodes on LAN receive ARP query
- B receives ARP packet, replies to A 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*

Routing to another LAN

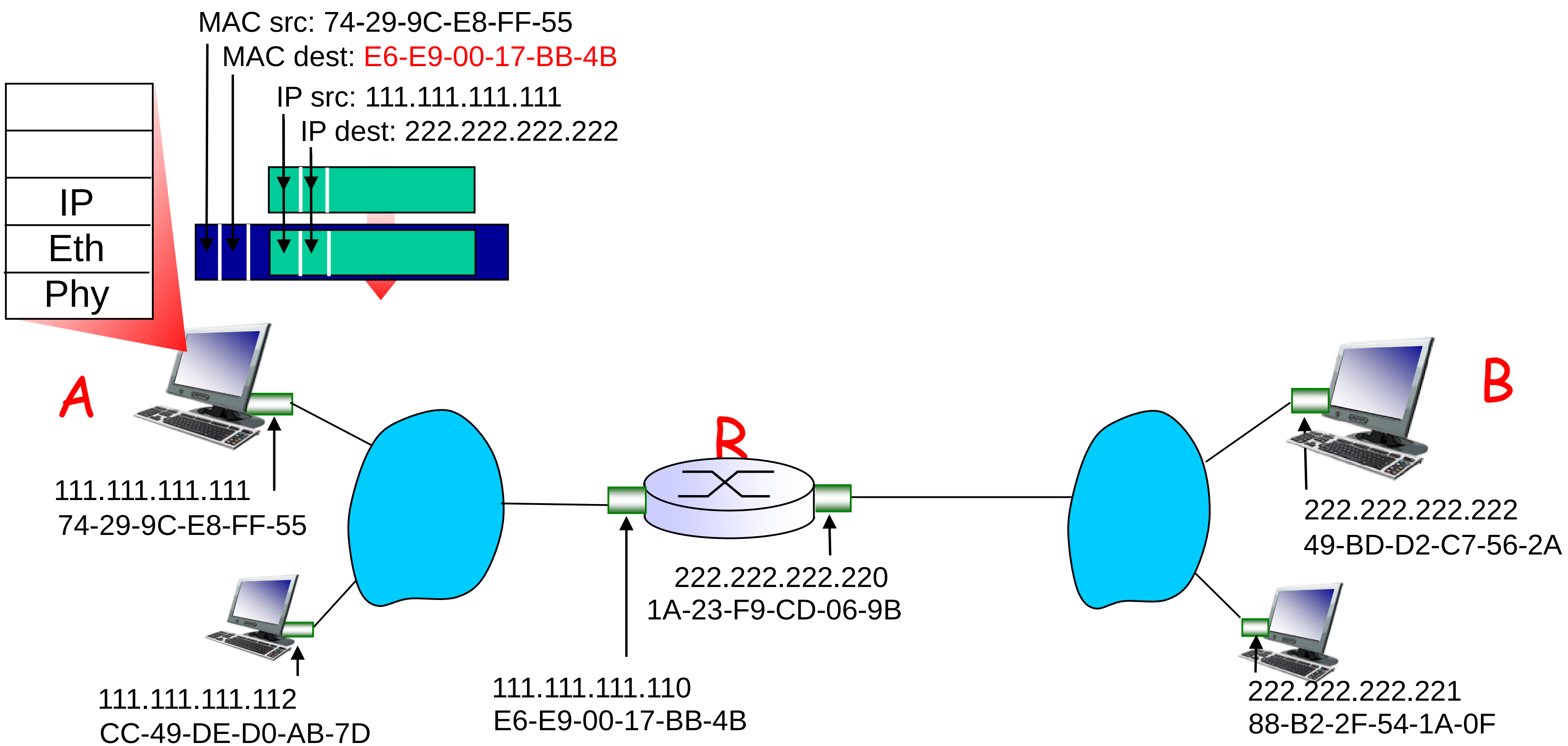
walkthrough: send datagram from A to B via R

- 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?)



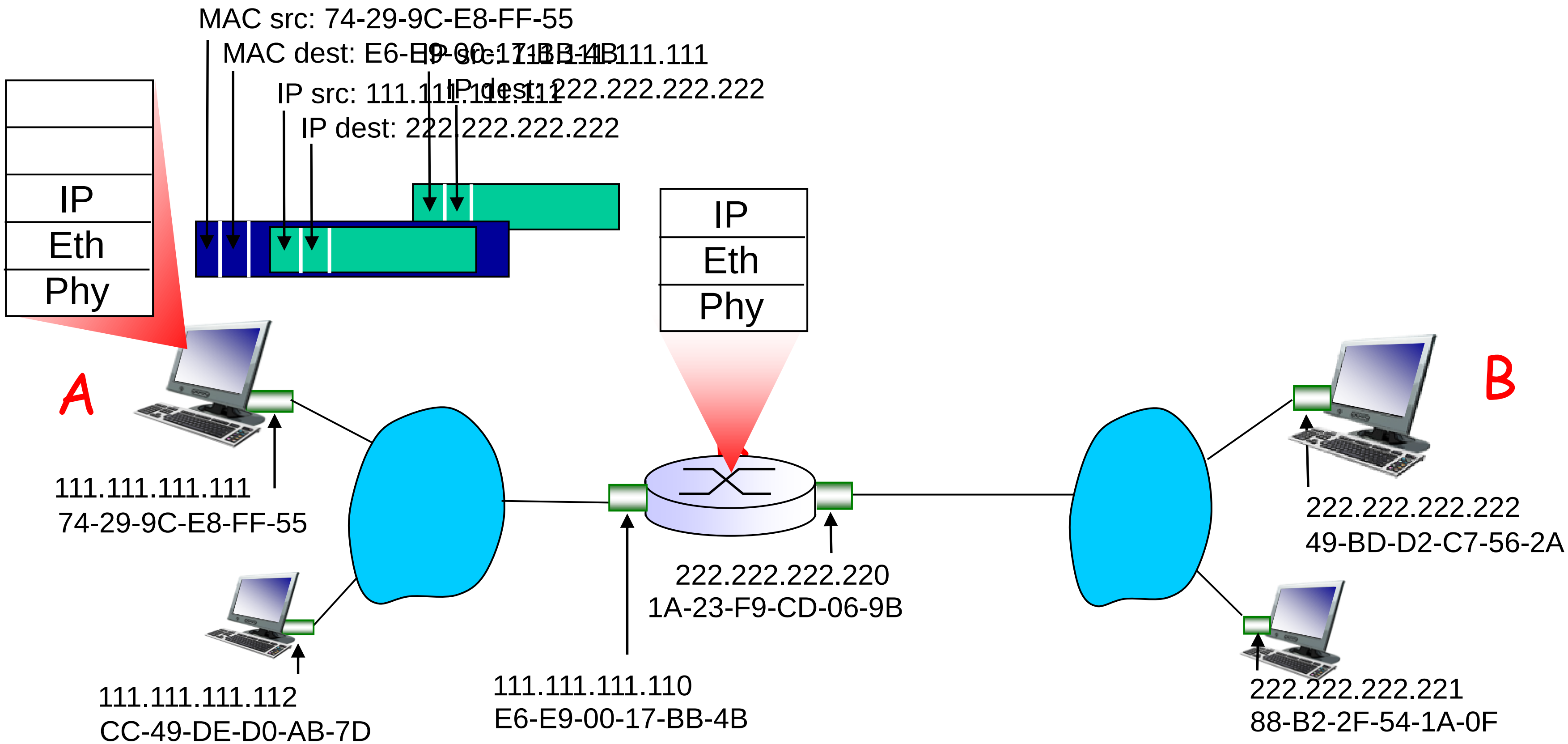
Routing to another LAN

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



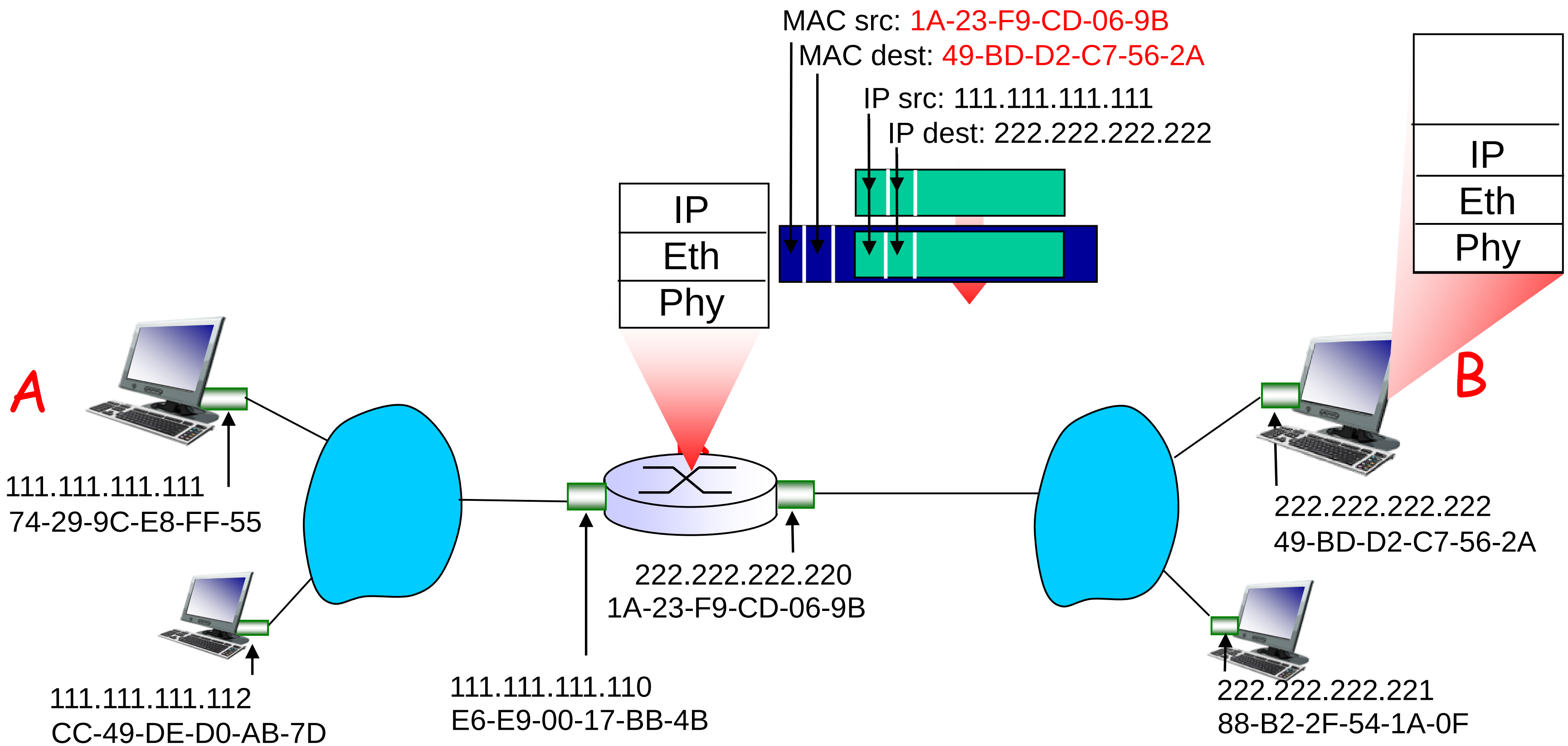
Routing to another LAN

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



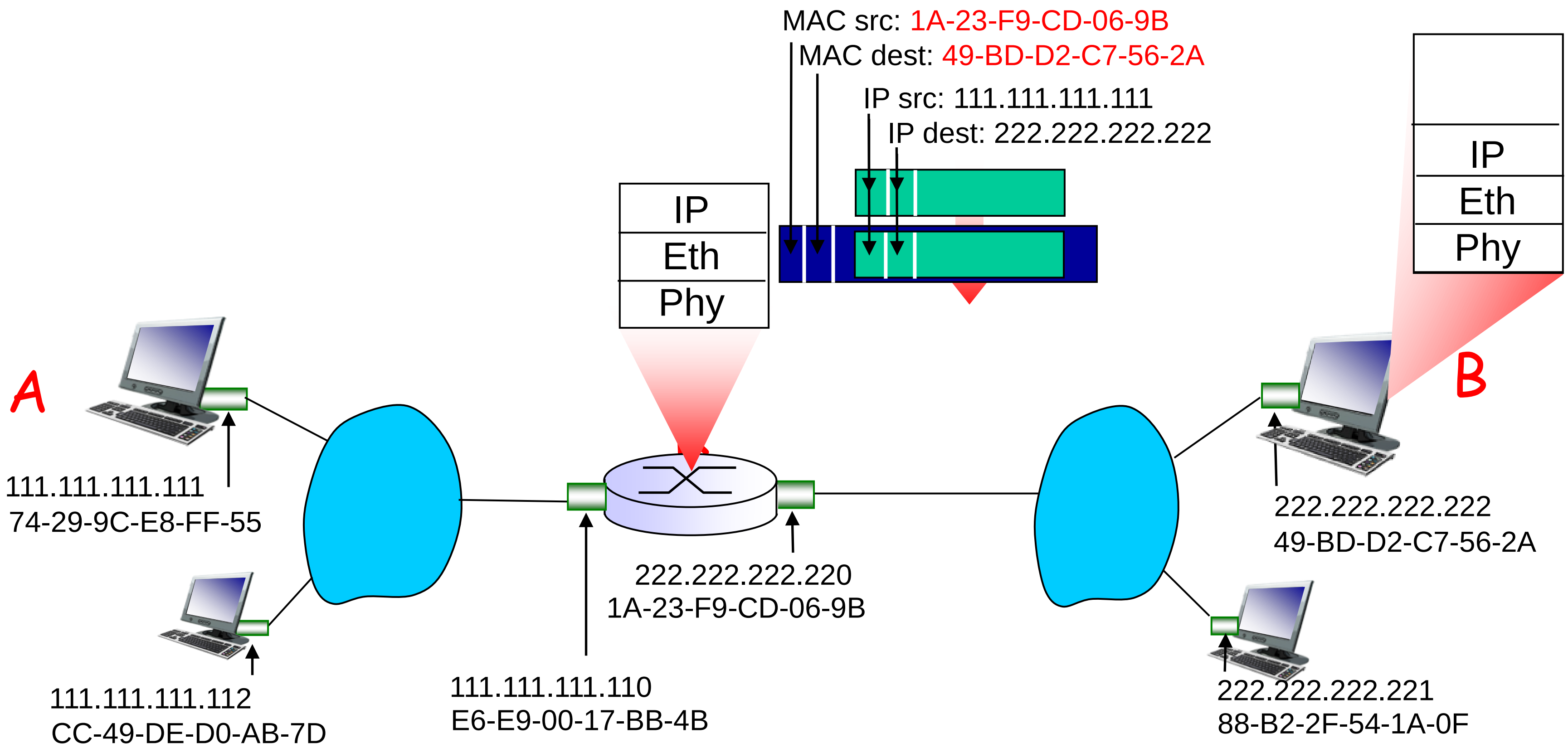
Routing to another LAN

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



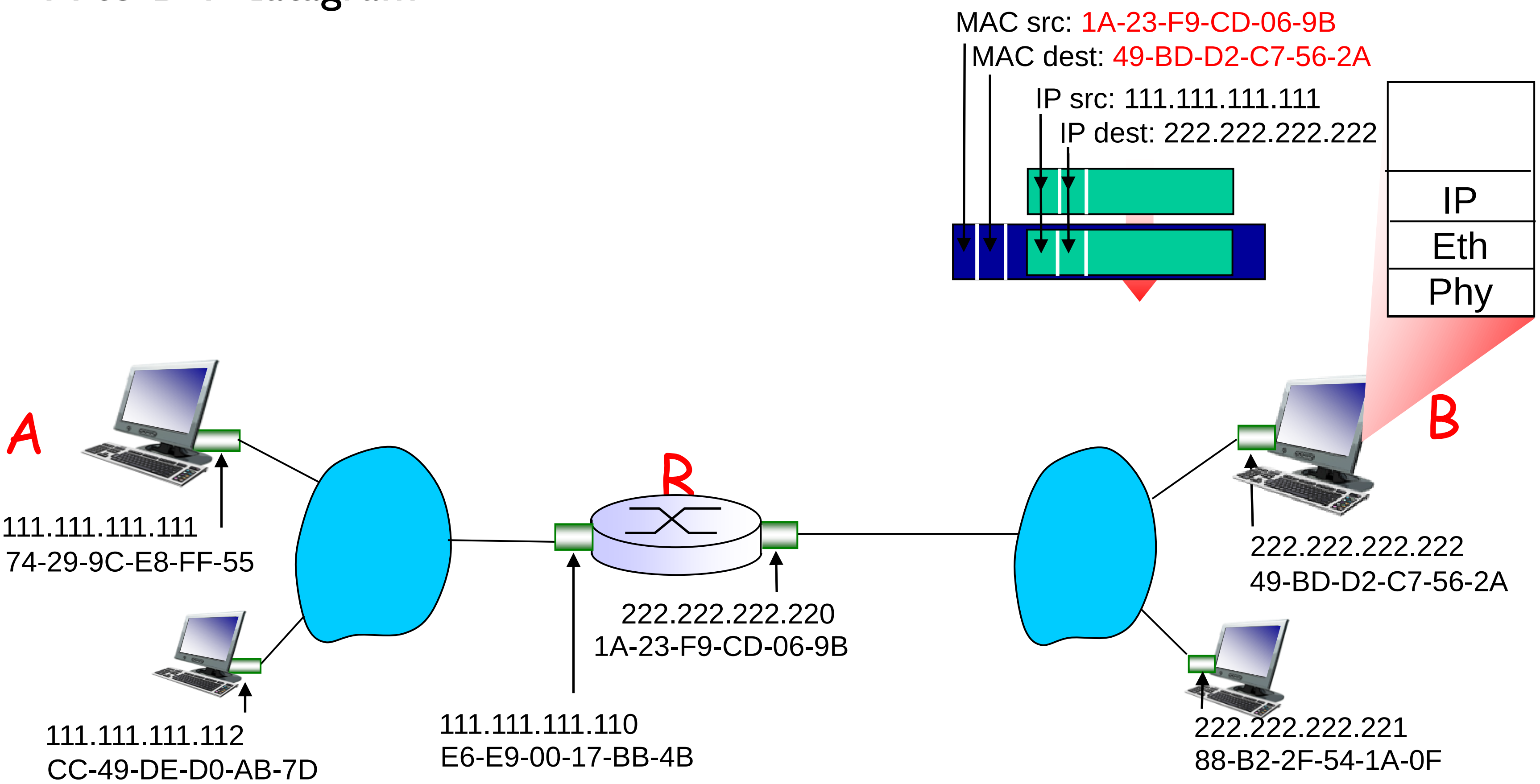
Routing to another LAN

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



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



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

Exercises-4

Figure 1 shows the network topology, and Figure 2 shows the hex-content of the first 80 bytes of an Ethernet frame of the host for a Web request.

(1) What is the IP address of the Web server? What is the MAC address of the default gateway of this host?

(2) When the IP packet encapsulated in this frame is forwarded through router R, which fields in the IP packet need to be modified？

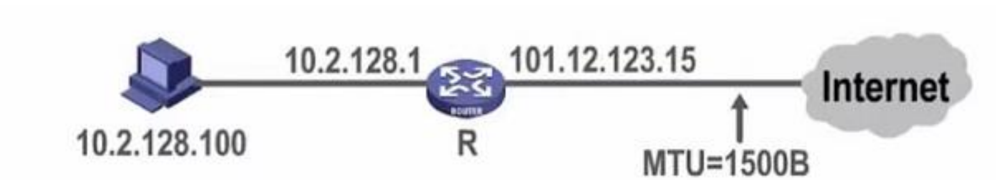


Fig. 1

```
00 21 27 21 51 ee 00 15   c5 c1 5e 28 08 00 45 00
01 ef 11 3b 40 00 80 06   ba 9d 0a 02 80 64 40 aa
62 20 04 ff 00 50 e0 e2   00 fa 7b f9 f8 05 50 18
fa f0 1a c4 00 00 47 45   54 20 2f 72 66 63 2e 68
74 6d 6c 20 48 54 54 50   2f 31 2e 31 0d 0a 41 63
```

Fig. 2

ICMP: Internet Control Message Protocol

- used by hosts, routers, gateways to communication 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

ICMP:brief summary

- ICMP is the control sibling of IP
- ICMP is used by IP and uses IP as network layer protocol
- ICMP is used for ping, traceroute, and path MTU discovery
  - Ping: uses ICMP Echo request/reply messages
  - Path MTU discovery
    - Send a large IP datagram with"No fragment" bit set
    - Reduce size until success (No ICMP message received)

ping

```
C:\Documents and Settings\XXR>ping mail.sina.com.cn

Pinging mail.sina.com.cn [202.108.43.230] with 32 bytes of data:

Reply from 202.108.43.230: bytes=32 time=368ms TTL=242
Reply from 202.108.43.230: bytes=32 time=374ms TTL=242
Request timed out.
Reply from 202.108.43.230: bytes=32 time=374ms TTL=242

Ping statistics for 202.108.43.230:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
    Minimum = 368ms, Maximum = 374ms, Average = 372ms
```

traceroute

```
C:\Documents and Settings\XXR>tracert mail.sina.com.cn

Tracing route to mail.sina.com.cn [202.108.43.230]
over a maximum of 30 hops:

  1    24 ms    24 ms    23 ms  222.95.172.1
  2    23 ms    24 ms    22 ms  221.231.204.129
  3    23 ms    22 ms    23 ms  221.231.206.9
  4    24 ms    23 ms    24 ms  202.97.27.37
  5    22 ms    23 ms    24 ms  202.97.41.226
  6    28 ms    28 ms    28 ms  202.97.35.25
  7    50 ms    50 ms    51 ms  202.97.36.86
  8   308 ms   311 ms   310 ms  219.158.32.1
  9   307 ms   305 ms   305 ms  219.158.13.17
 10   164 ms   164 ms   165 ms  202.96.12.154
 11   322 ms   320 ms  2988 ms  61.135.148.50
 12   321 ms   322 ms   320 ms  freemail43-230.sina.com [202.108.43.230]

Trace complete.
```

Traceroute and ICMP

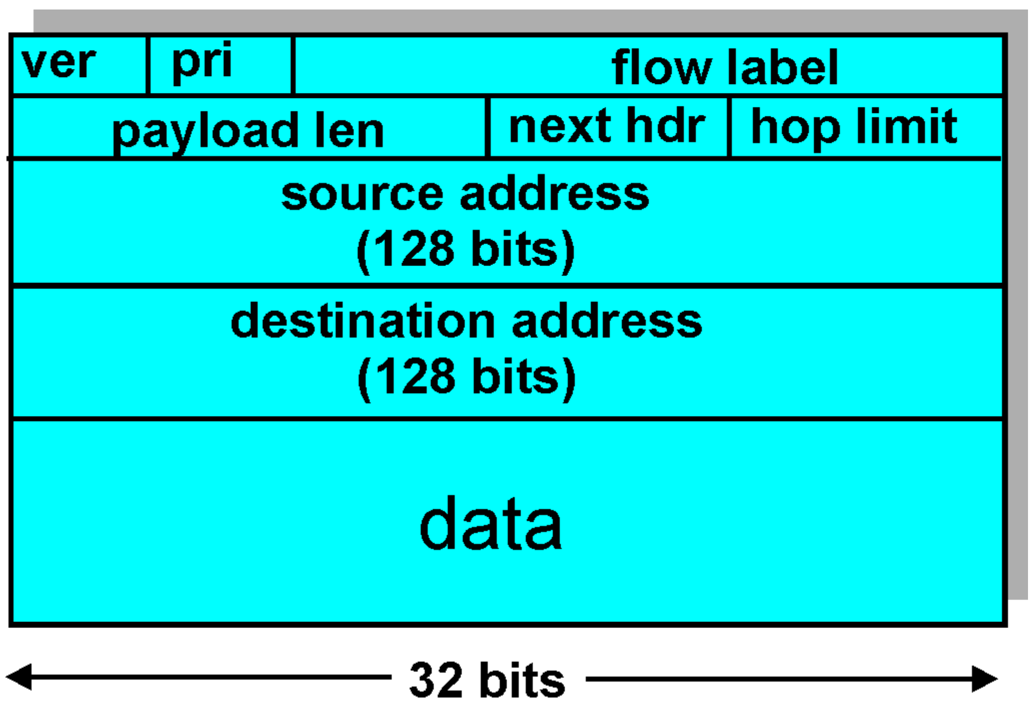
- Source sends series of UDP segments to dest.
  - First has TTL=1
  - Second has TTL=2, etc.
  - Unlikely port number
- When nth datagram arrives to nth router:
  - Router discards datagram
  - And sends to source an ICMP message (type 11, code 0)
  - Message includes name of router & IP address
- When ICMP message arrives, source calculates RTT
- Traceroute does this 3 times
- Stopping criterion
- UDP segment eventually arrives at destination host
- Destination returns ICMP "host unreachable" packet (type 3, code 3)
- When source gets this ICMP, stops

IPv6

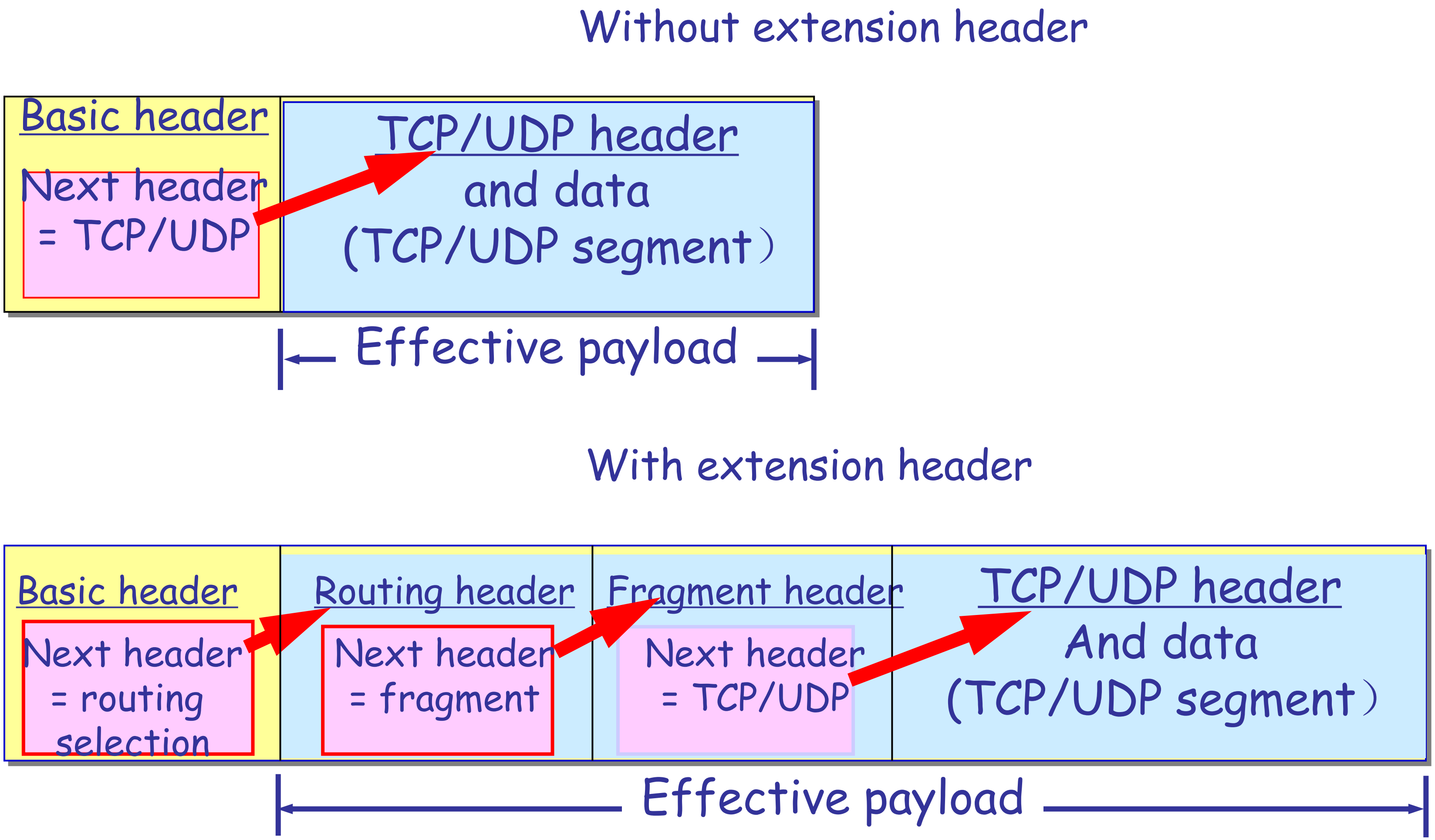
- Initial motivation: 32-bit address space completely allocated by 2008.
- Additional motivation:
  - header format helps speed processing/forwarding
  - header changes to facilitate QoS
  - new "anycast" address: route to "best" of several replicated servers
- IPv6 datagram format:
  - fixed-length 40 byte header
  - no fragmentation allowed

IPv6 Header (Cont)

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



Next header


Without extension header
Basic header
Next header = TCP/UDP
TCP/UDP header and data (TCP/UDP segment)
Effective payload
With extension header
Basic header
Next header = routing selection
Routing header
Next header = fragment
Fragment header
Next header = TCP/UDP
TCP/UDP header And data (TCP/UDP segment)
Effective payload

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

IPv6 address

- Three types: unicast, multicast, anycast
- Colon hexadecimal notation:

68E6:8C64:FFFF:FFFF:0:1180:960A:FFFF

- Zero compression:

FF05:0:0:0:0:0:0:B3 == FF05::B3 ;

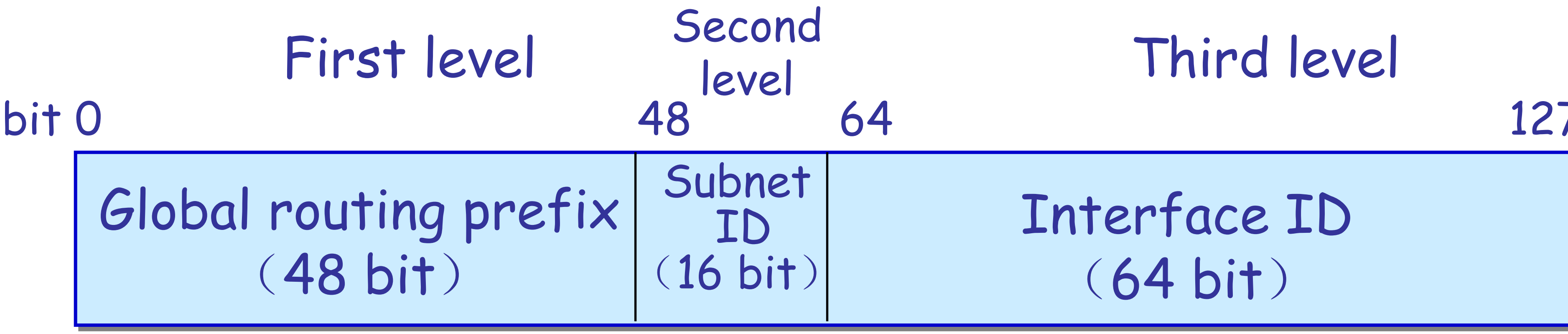
0:0:0:0:0:0:128.10.2.1 == ::128.10.2.1

12AB:0000:0000:CD30:0000:0000:0000:0000/60

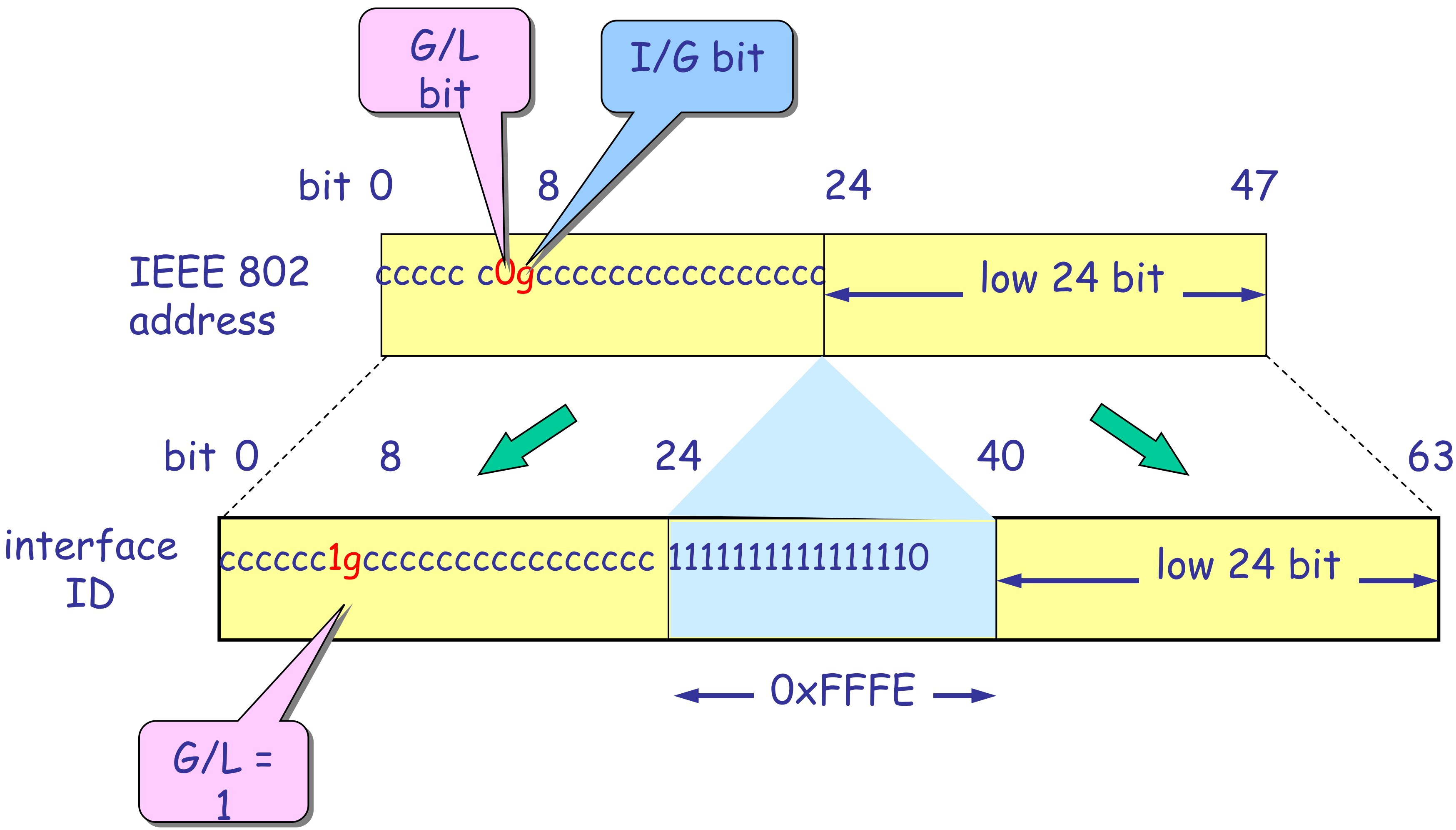
== 12AB::CD30:0:0:0:0/60

== 12AB:0:0:CD30::/60

Unicast address



EUI-64

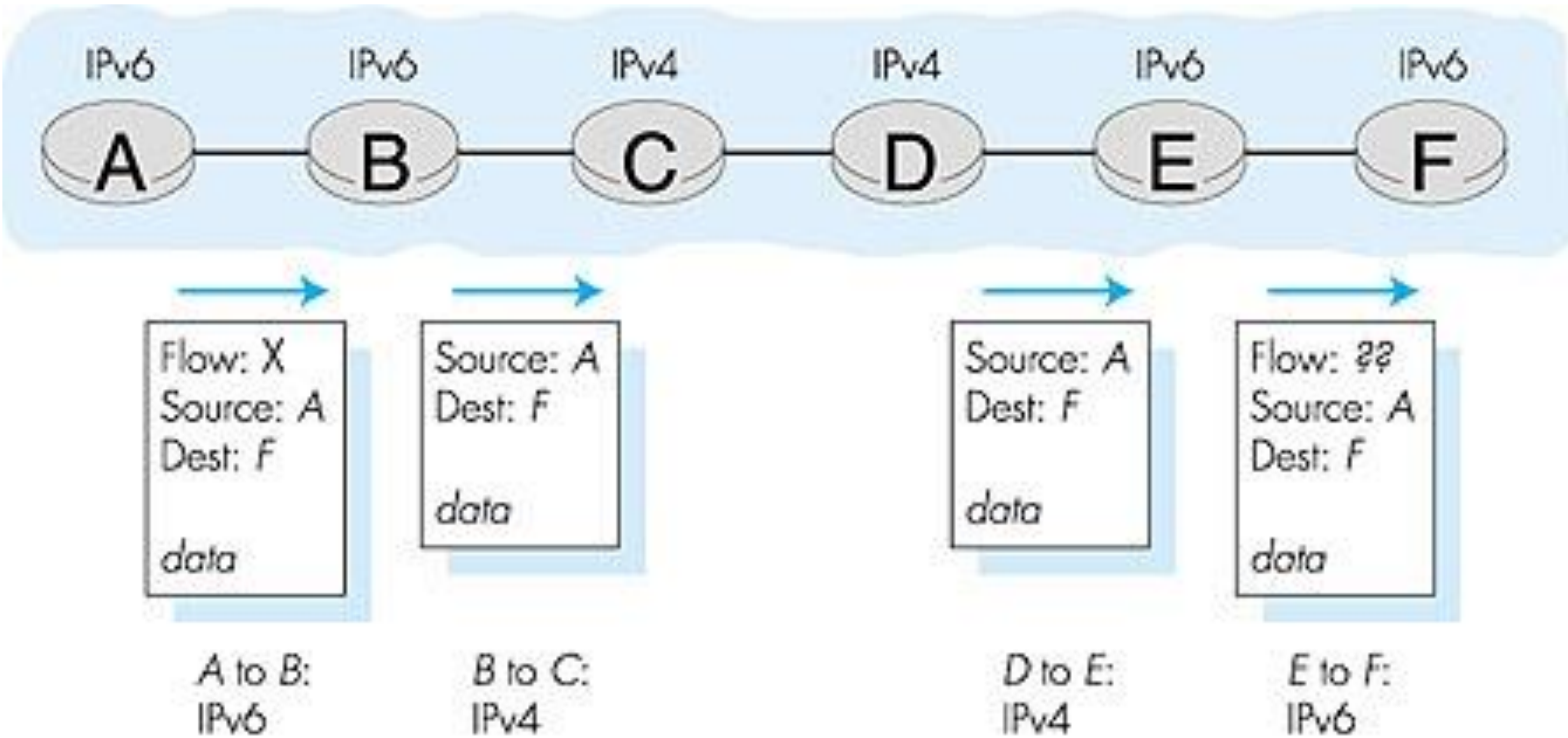

G/L bit
I/G bit
bit 0
8
24
47
IEEE 802 address
ccccc c0gcccccccccccccccc
low 24 bit
bit 0
8
24
40
63
interface ID
cccccc1gcccccccccccccccc
1111111111111110
low 24 bit
0xFFFE
G/L = 1

Example:
host MAC address: 00:0C:85:AB:50:01
insert FFFE:
00:0C:85:FF:FE:AB:50:01
set G/L bit:
02:0C:85:FF:FE:AB:50:01
host EUI-64 is:
020C:85FF:FEAB:5001
router advertise the 64bit network prefix
Host unicast address: prefix+EUI-64

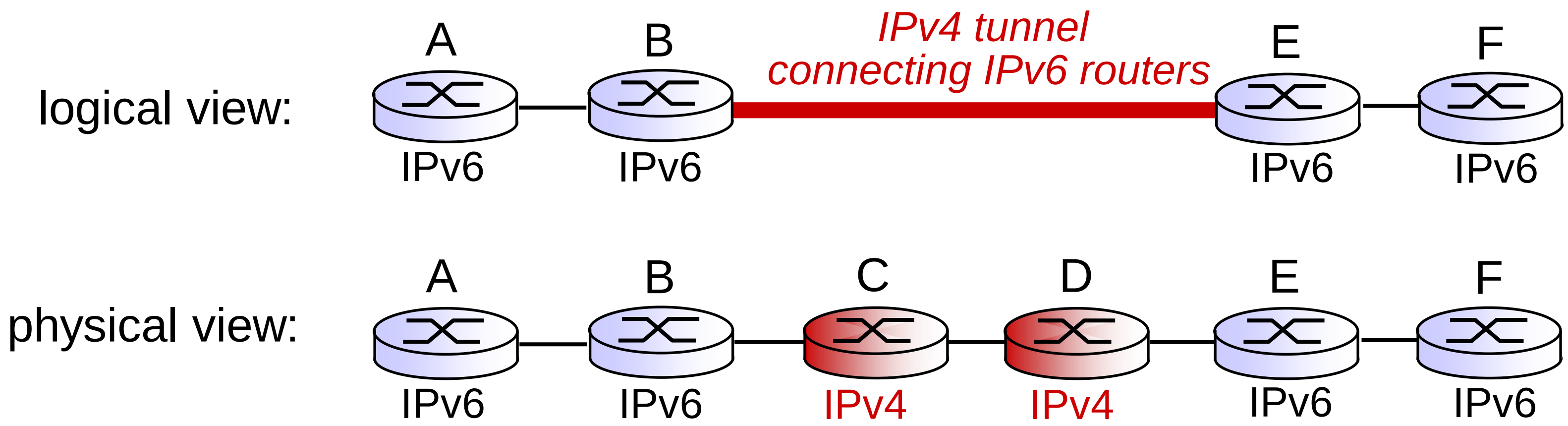
Transition From IPv4 To IPv6

- Not all routers can be upgraded simultaneous
  - no "flag days"
  - How will the network operatewith mixed IPv4 and IPv6 routers?
- Two proposed approaches:
  - *Dual Stack*: some routers with dual stack (v6, v4) can "translate" between formats
  - *Tunneling*: IPv6 carried as payload n IPv4 datagram among IPv4 routers

Dual Stack Approach

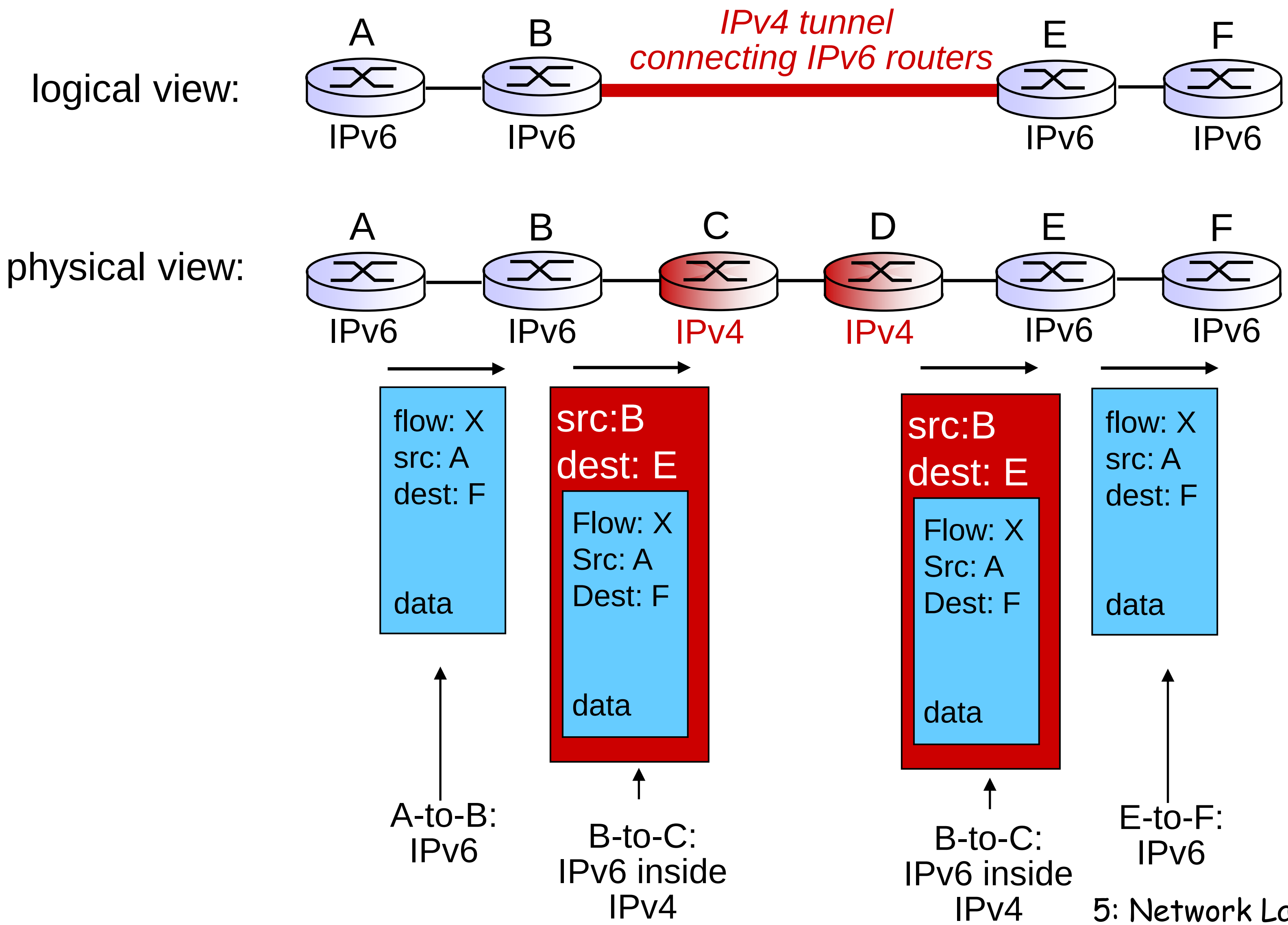


Tunneling



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


IPv4 tunnel
connecting IPv6 routers
logical view:
A
B
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

Summary

- Virtual circuit and datagram networks
- Routing algorithms:
  - Dijkstra's algorithm
  - Broadcast routing
  - Link state
  - Distance vector ("count to infinity" problem)
- Routing in the Internet (RIP, OSPF, BGP)
- IP: Internet protocol
  - IPv4 Datagram format
  - IPv4 addressing
  - IP fragment
  - NAT
  - ARP
  - ICMP
  - IPv6
  - From IPv4 to IPv6

Homework

- P455:R4,R5
- P457:P3,P5
- P458:P11
- P396:P14
- P395:P8,P11,P12