Chapter 4
Network Layer

Computer Networking: A Top Down Approach Featuring the Internet, 2nd edition.
Jim Kurose, Keith Ross
Addison-Wesley, July 2002.

Chapter 4: Network Layer

Chapter goals:

- understand principles behind network layer services:
  - routing (path selection)
  - dealing with scale
  - how a router works
  - advanced topics: IPv6, mobility
- instantiation and implementation in the Internet

Overview:

- network layer services
- routing principles: path selection
- hierarchical routing
- IP
- Internet routing protocols
  - intra-domain
  - inter-domain
- what's inside a router?
- IPv6
- mobility

Chapter 4 roadmap

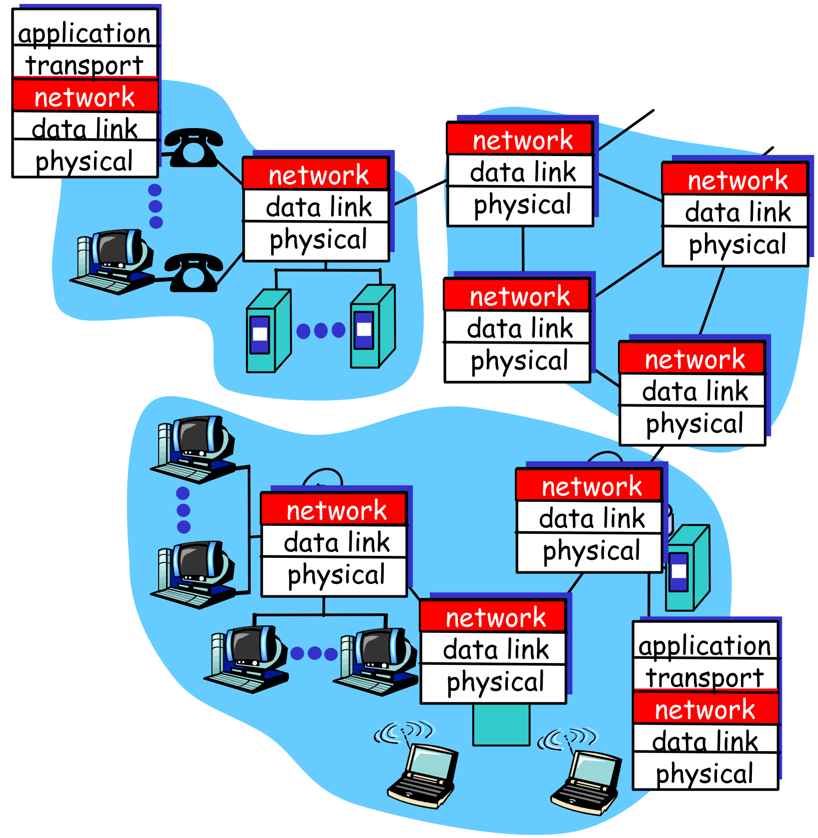
4.1 Introduction and Network Service Models
4.2 Routing Principles
4.3 Hierarchical Routing
4.4 The Internet (IP) Protocol
4.5 Routing in the Internet
4.6 What's Inside a Router
4.7 IPv6
4.8 Multicast Routing
4.9 Mobility

Network layer functions

- transport packet from sending to receiving hosts
- network layer protocols in *every* host, router

three important functions:

- *path determination:* route taken by packets from source to dest. *Routing algorithms*
- *forwarding:* move packets from router's input to appropriate router output
- *call setup:* some network architectures require router call setup along path before data flows



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?

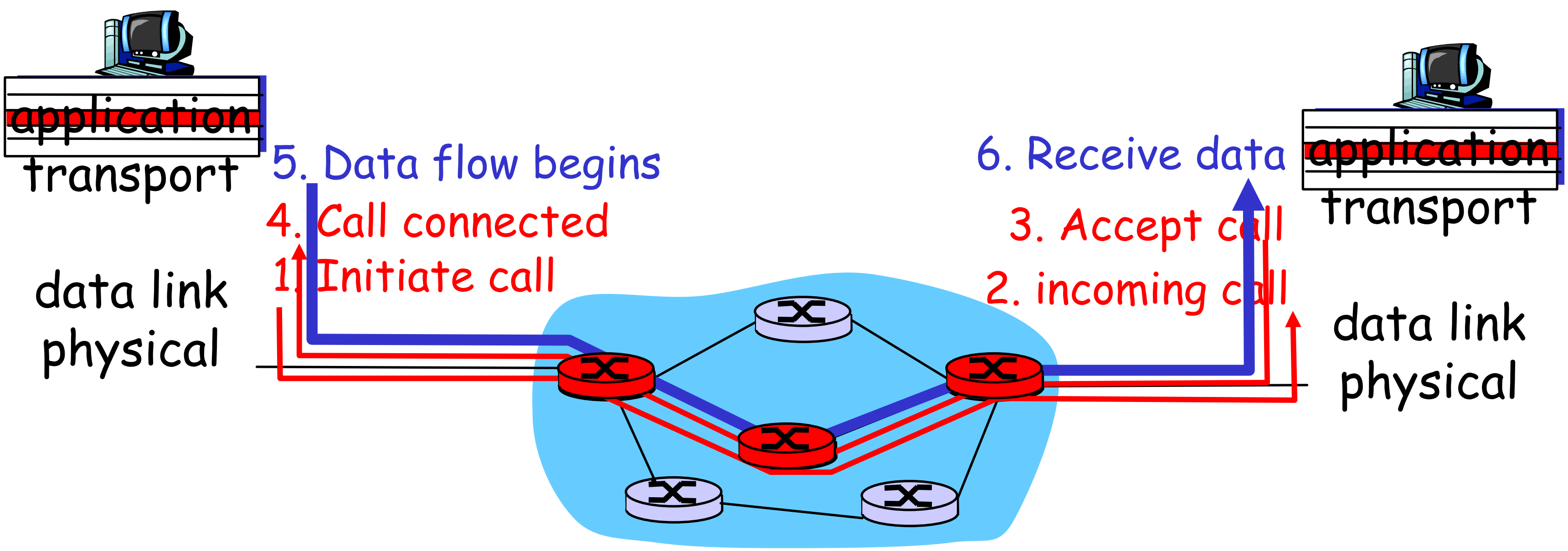
Virtual circuits

"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 ID)
- *every* router on source-dest path maintains "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.

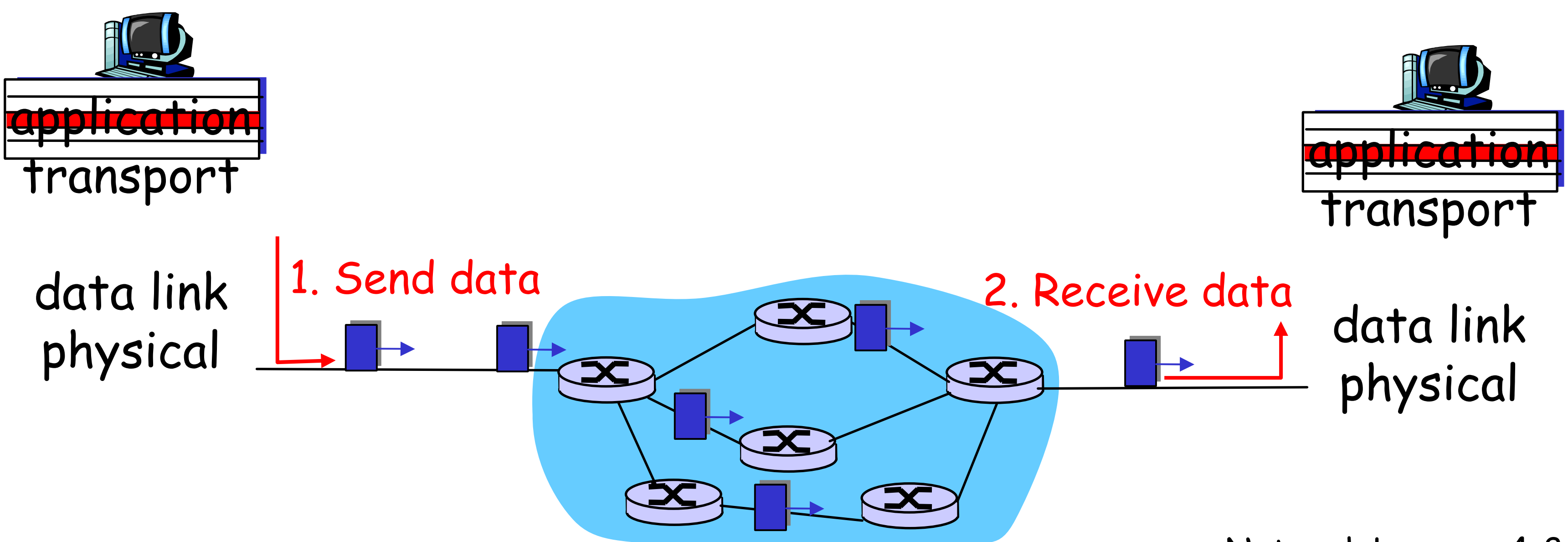
Virtual circuits: signaling protocols

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



Datagram networks: the Internet model

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



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

- Internet model being extended: Intserv, Diffserv
  - Chapter 6

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

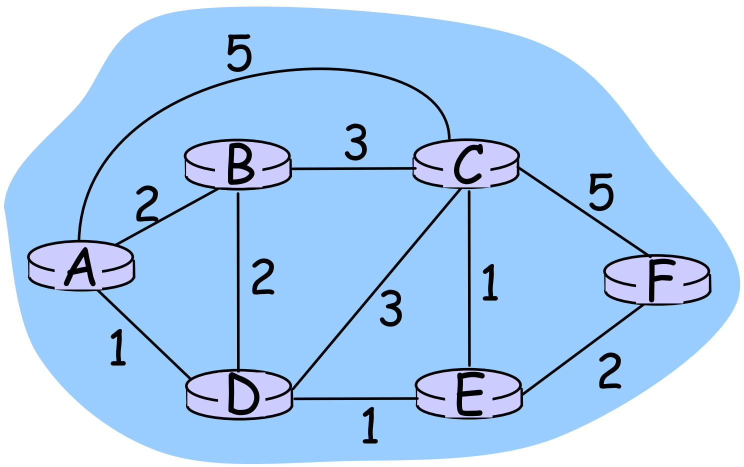
- evolved from telephony
- human conversation:
  - strict timing, reliability requirements
  - need for guaranteed service
- "dumb" end systems
  - telephones
  - complexity inside network

Chapter 4 roadmap

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

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

A Link-State Routing Algorithm

Dijkstra's algorithm

- net topology, link costs known to all nodes
  - accomplished via "link state broadcast"
  - all nodes have same info
- computes least cost paths from one node ('source") to all other nodes
  - gives routing table for that node
- iterative: after k iterations, know least cost path to k dest.'s

Notation:

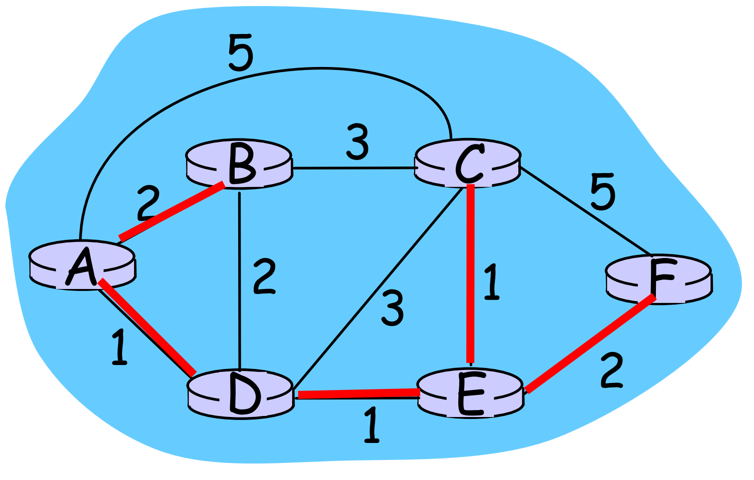
- c(i,j): link cost from node i to j. cost infinite if not direct neighbors
- D(v): current value of cost of path from source to dest. V
- p(v): predecessor node along path from source to v, that is next v
- N: set of nodes whose least cost path definitively known

Dijkstra's Algorithm

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

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	start N	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	A	2,A	5,A	1,A	infinity	infinity
1	AD	2,A	4,D		2,D	infinity
2	ADE	2,A	3,E			4,E
3	ADEB		3,E			4,E
4	ADEBC					4,E
5	ADEBCF					



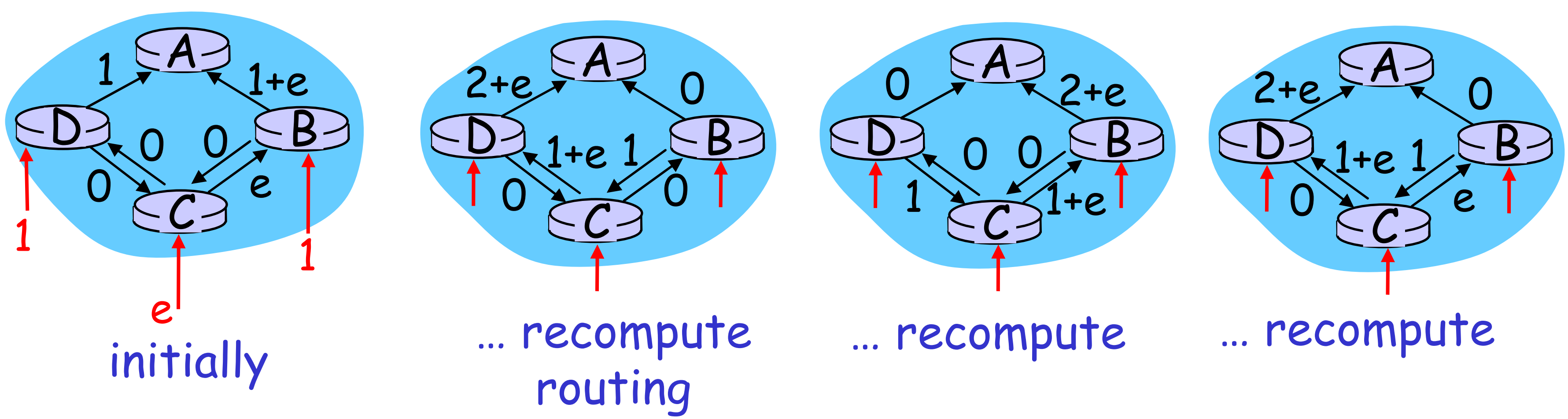
Dijkstra's algorithm, discussion

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



Distance Vector Routing Algorithm

iterative:

- continues until no nodes exchange info.
- *self-terminating*: no "signal" to stop

asynchronous:

- nodes need *not* exchange info/iterate in lock step!

distributed:

- each node communicates *only* with directly-attached neighbors

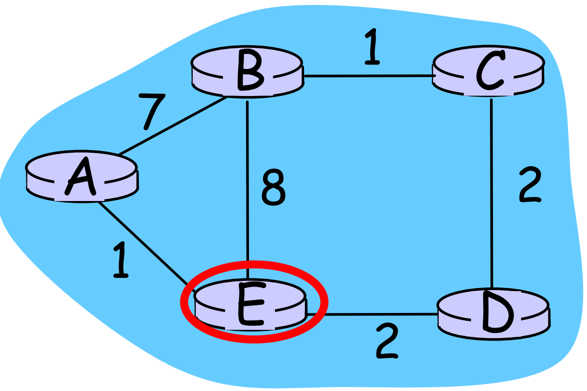
Distance Table data structure

- each node has its own
- row for each possible destination
- column for each directly-attached neighbor to node
- example: in node X, for dest. Y via neighbor Z:

$$D^X(Y,Z) = \text{distance } \textit{from} \text{ X } \textit{to} \text{ Y, } \textit{via} \text{ Z as next hop}$$

$$= c(X,Z) + \min_w \{D^Z(Y,w)\}$$

Distance Table: example



A	
A	0
B	6
C	5
D	3

B	
A	6
B	0
C	1
D	3

D	
A	3
B	3
C	2
D	0

$$D^E(C,D) = c(E,D) + \min_w \{D^D(C,w)\} = 2+2 = 4$$

$$D^E(A,D) = c(E,D) + \min_w \{D^D(A,w)\} = 2+3 = 5$$

loop!

$$D^E(A,B) = c(E,B) + \min_w \{D^B(A,w)\} = 8+6 = 14$$

loop!

cost to destination via

$D^E()$	A	B	D
A	1	14	5
B	7	8	5
C	6	9	4
D	4	11	2

destination

Distance table gives routing table

cost to destination via

$D^E()$	A	B	D
A	1	14	5
B	7	8	5
C	6	9	4
D	4	11	2

destination

Outgoing link to use, cost

A	A,1
B	D,5
C	D,4
D	D,4

destination

Distance table → Routing table

Distance Vector Routing: overview

Iterative, asynchronous: each local iteration caused by:
- local link cost change
- message from neighbor: its least cost path change from neighbor

Distributed:
- each node notifies neighbors *only* when its least cost path to any destination changes
  - neighbors then notify their neighbors if necessary

Each node:

wait for (change in local link cost or msg from neighbor)

recompute distance table

if least cost path to any dest has changed, *notify* neighbors

Distance Vector Algorithm:

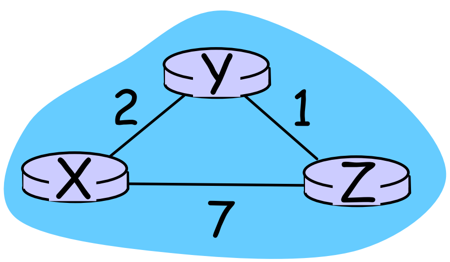
At all nodes, X:

Initialization:
 for all adjacent nodes v:
    $D^X(*,v)$ = infinity /* the * operator means "for all rows" */
    $D^X(v,v) = c(X,v)$
 for all destinations, y
    send $\min_w D^X(y,w)$ to each neighbor /* w over all X's neighbors */

Distance Vector Algorithm (cont.):

```
loop
  wait (until I see a link cost change to neighbor V
        or until I receive update from neighbor V)

  if (c(X,V) changes by d)
    /* change cost to all dest's via neighbor v by d */
    /* note: d could be positive or negative */
    for all destinations y:  D^X(y,V) =  D^X(y,V) + d

  else if (update received from V wrt destination Y)
    /* shortest path from V to some Y has changed  */
    /* V has sent a new value for its  min_w DV(Y,w) */
    /* call this received new value is "newval"     */
    for the single destination y: D^X(Y,V) = c(X,V) + newval

  if we have a new min_w D^X(Y,w)for any destination Y
     send new value of min_w D^X(Y,w) to all neighbors

forever
```

Distance Vector Algorithm: example



D^X (cost via)

dest	Y	Z
Y	2	∞
Z	∞	7

D^Y (cost via)

dest	X	Z
X	2	∞
Z	∞	1

D^Z (cost via)

dest	X	Y
X	7	∞
Y	∞	1

D^X (cost via)

dest	Y	Z
Y	2	8
Z	3	7

D^Y (cost via)

dest	X	Z
X	2	8
Z	9	1

D^Z (cost via)

dest	X	Y
X	7	3
Y	9	1

D^X (cost via)

dest	Y	Z
Y		
Z		

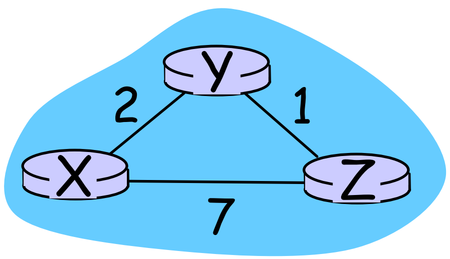
D^Y (cost via)

dest	X	Z
X		
Z		

D^Z (cost via)

dest	X	Y
X		
Y		

Distance Vector Algorithm: example



D^X (dest)	cost via Y	cost via Z
Y	2	∞
Z	∞	7

D^Y (dest)	cost via X	cost via Z
X	2	∞
Z	∞	1

D^Z (dest)	cost via X	cost via Y
X	7	∞
Y	∞	1

D^X (dest)	cost via Y	cost via Z
Y	2	8
Z	3	7

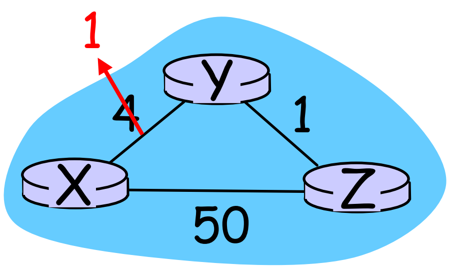
$$D^X(Y,Z) = c(X,Z) + \min_w\{D^Z(Y,w)\}$$
$$= 7+1 = 8$$

$$D^X(Z,Y) = c(X,Y) + \min_w\{D^Y(Z,w)\}$$
$$= 2+1 = 3$$

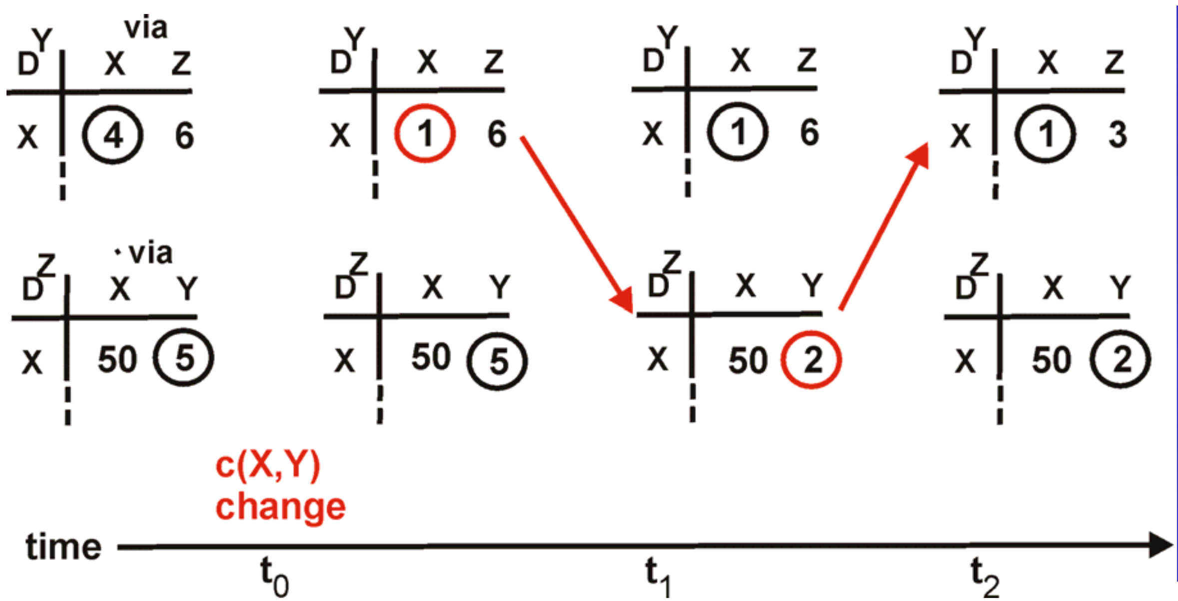
Distance Vector: link cost changes

Link cost changes:

- node detects local link cost change
- updates distance table (line 15)
- if cost change in least cost path, notify neighbors (lines 23,24)

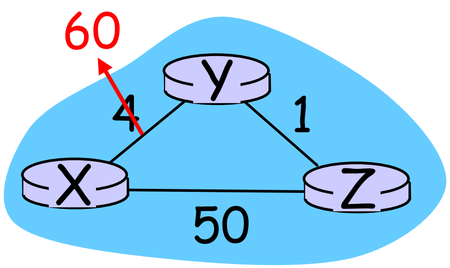


"good news travels fast"



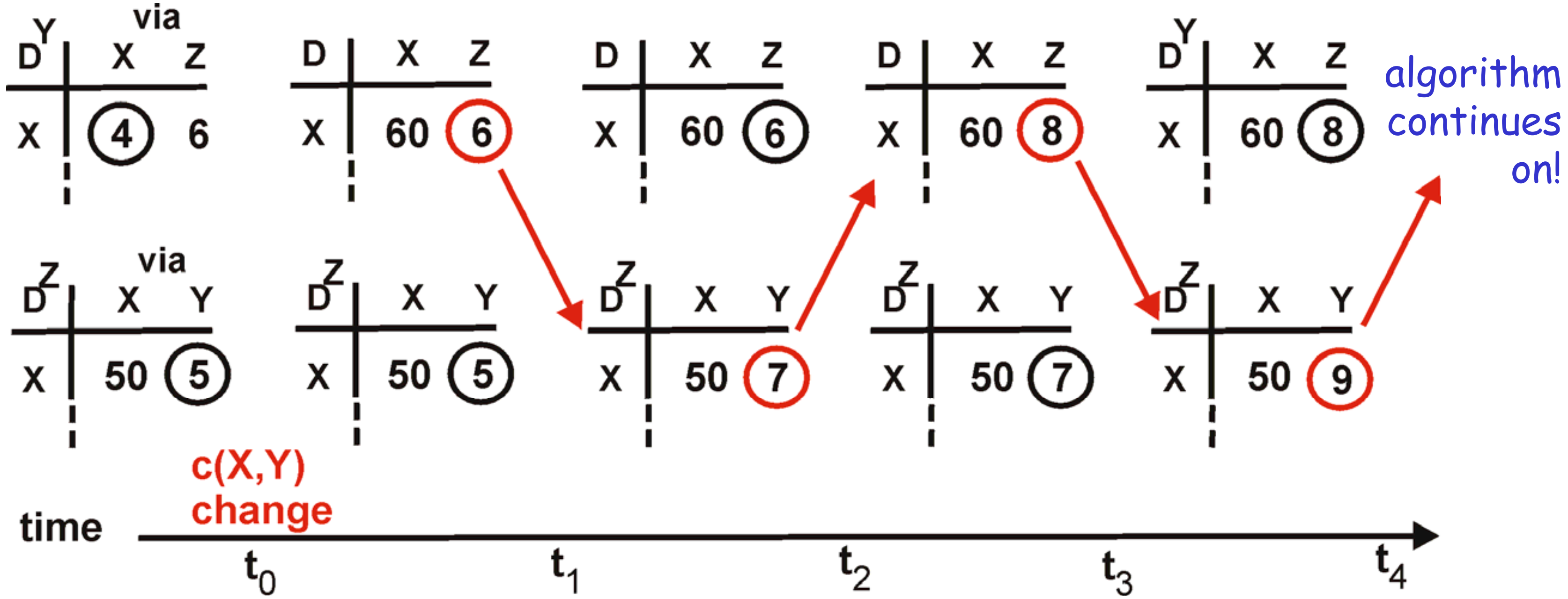
algorithm terminates

Distance Vector: link cost changes



Link cost changes:

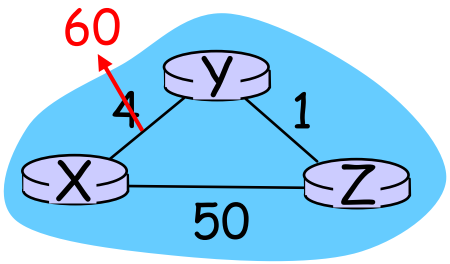
- good news travels fast
- bad news travels slow - "count to infinity" problem!

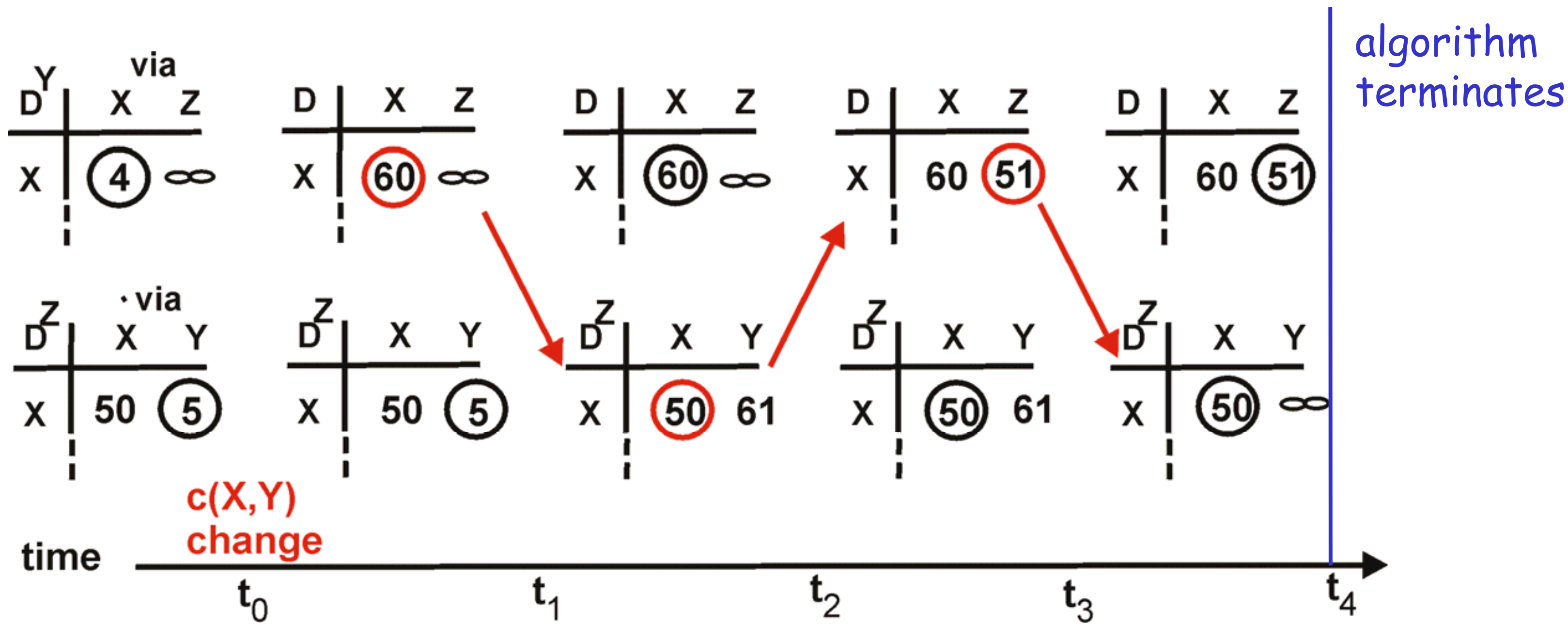


Distance Vector: poisoned reverse

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?





Comparison of LS and DV algorithms

Message complexity

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

Speed of Convergence

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

Robustness: what happens if router malfunctions?

LS:

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

DV:

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

Chapter 4 roadmap

4.1 Introduction and Network Service Models
4.2 Routing Principles
4.3 Hierarchical Routing
4.4 The Internet (IP) Protocol
4.5 Routing in the Internet
4.6 What's Inside a Router
4.7 IPv6
4.8 Multicast Routing
4.9 Mobility

Hierarchical Routing

Our routing study thus far - idealization

- all routers identical
- network "flat"

… *not* true in practice

scale: with 200 million destinations:

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

administrative autonomy

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

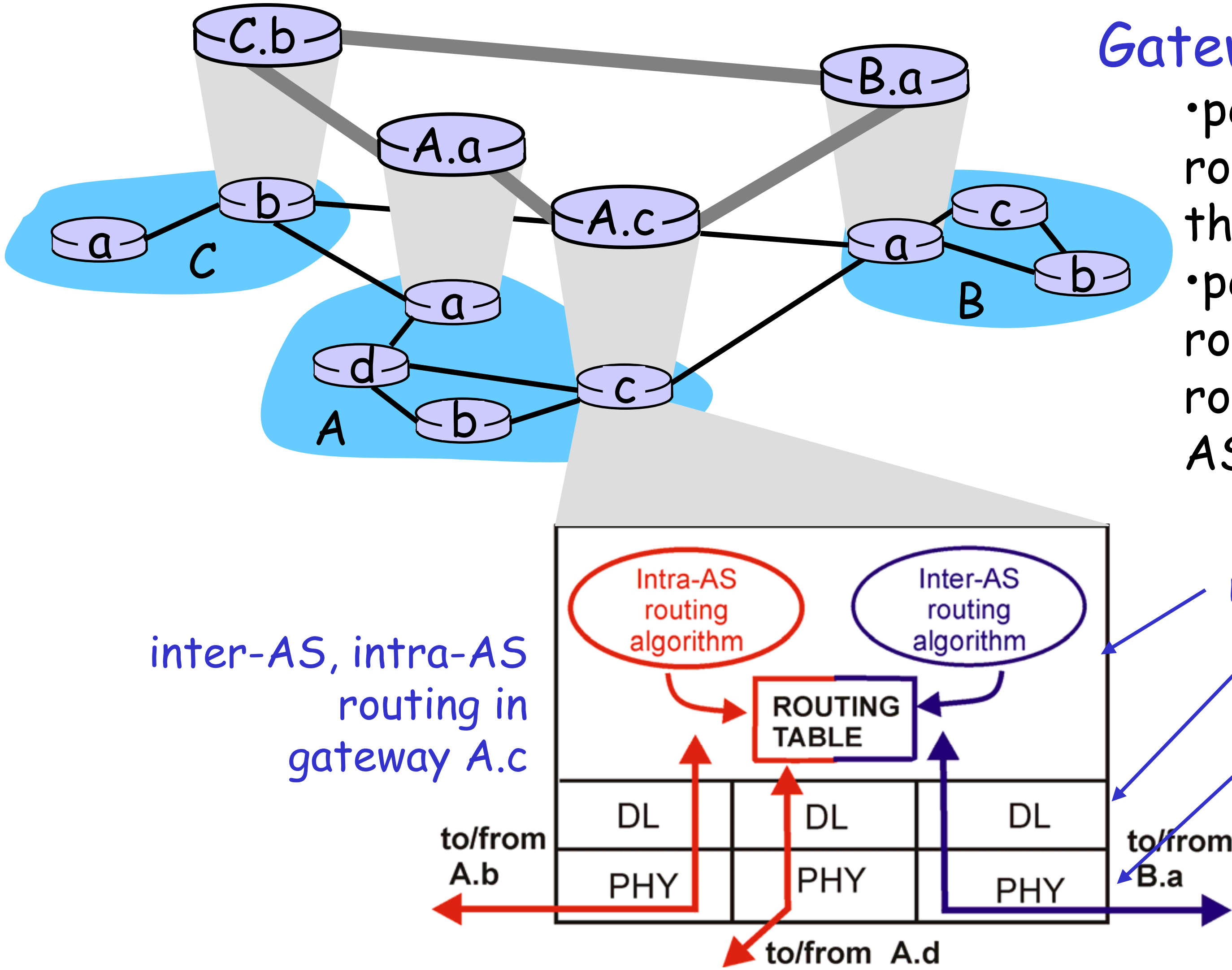
Hierarchical Routing

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

gateway 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

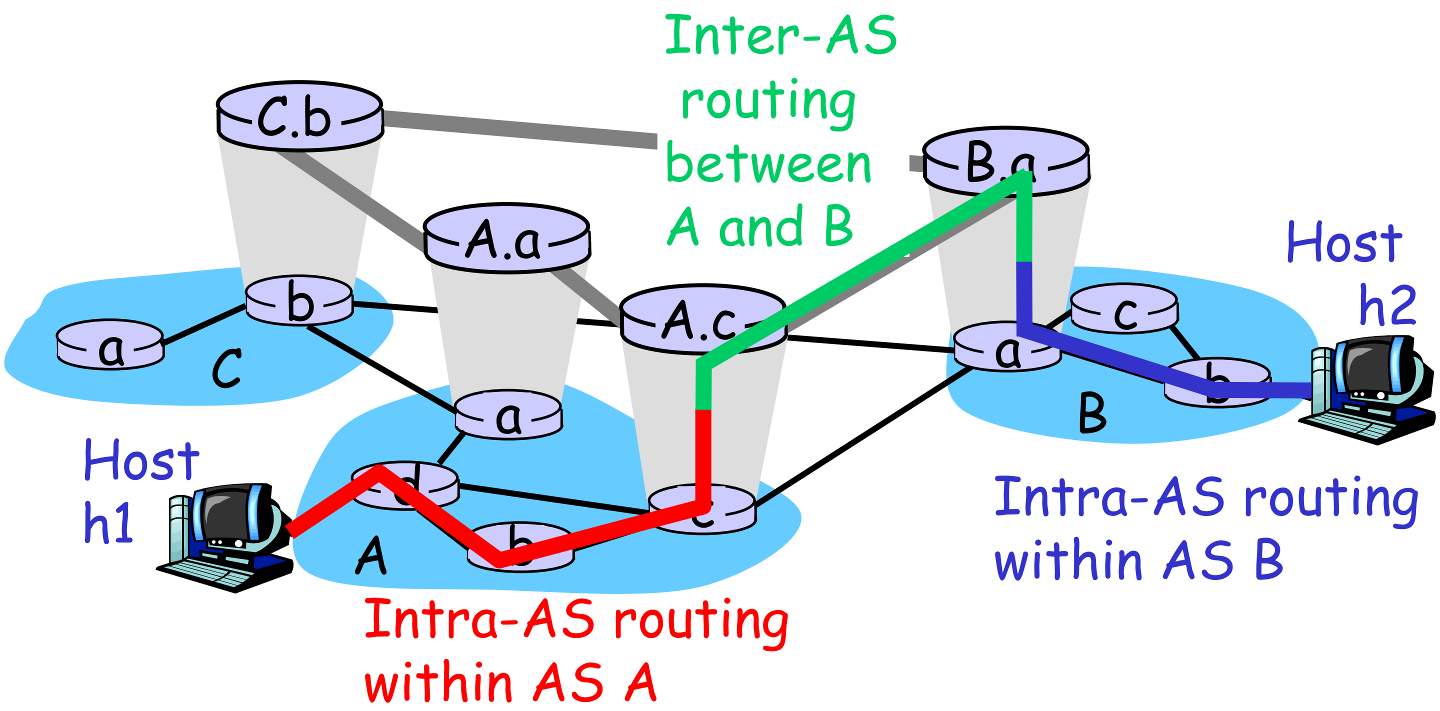
Intra-AS and Inter-AS routing



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

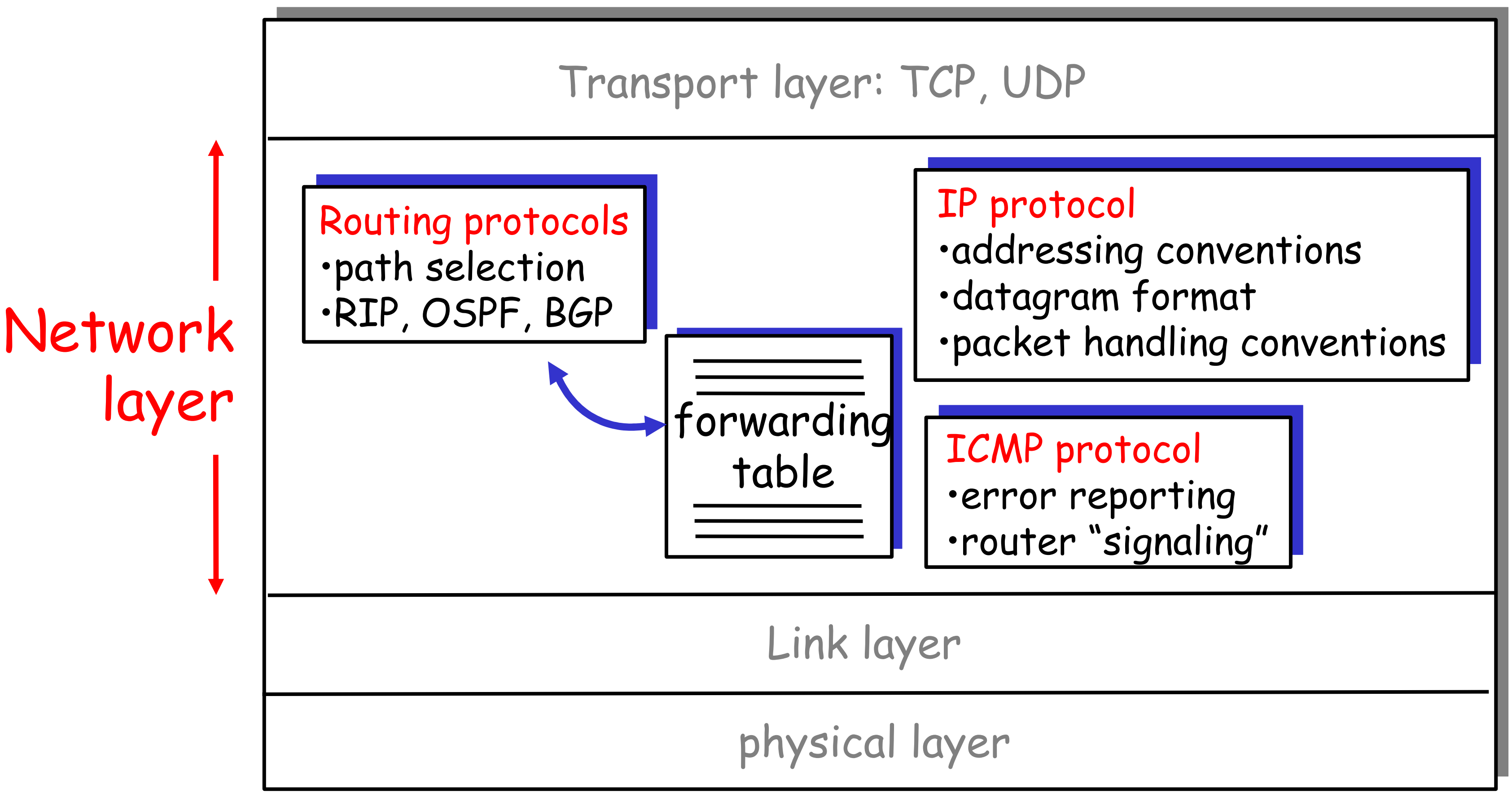
Chapter 4 roadmap

4.1 Introduction and Network Service Models
4.2 Routing Principles
4.3 Hierarchical Routing
4.4 The Internet (IP) Protocol
- 4.4.1 IPv4 addressing
- 4.4.2 Moving a datagram from source to destination
- 4.4.3 Datagram format
- 4.4.4 IP fragmentation
- 4.4.5 ICMP: Internet Control Message Protocol
- 4.4.6 DHCP: Dynamic Host Configuration Protocol
- 4.4.7 NAT: Network Address Translation

4.5 Routing in the Internet
4.6 What's Inside a Router
4.7 IPv6
4.8 Multicast Routing
4.9 Mobility

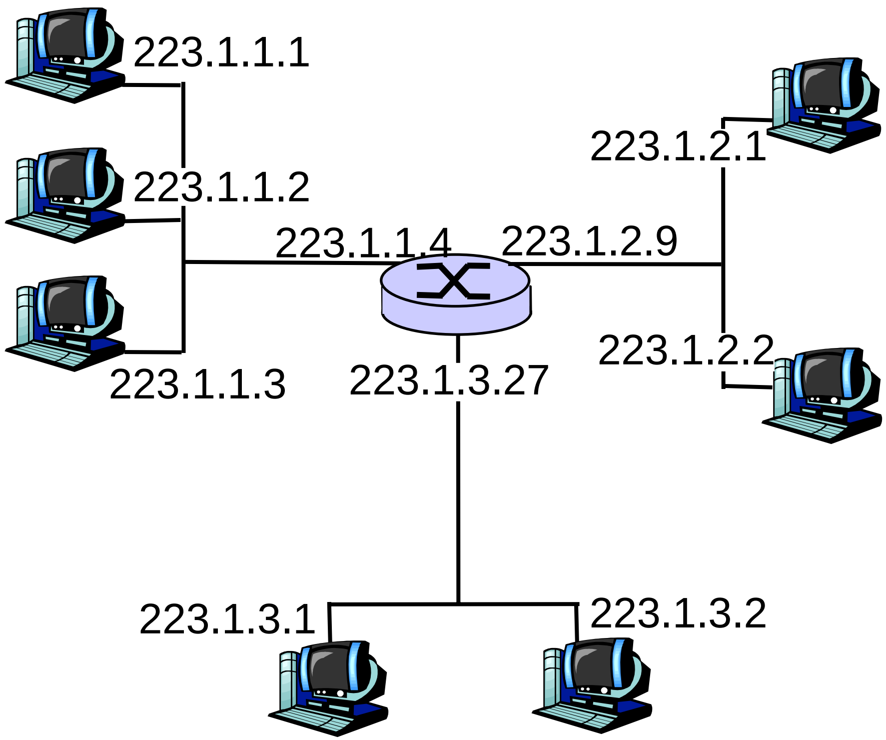
The Internet Network layer

Host, router network layer functions:



IP Addressing: introduction

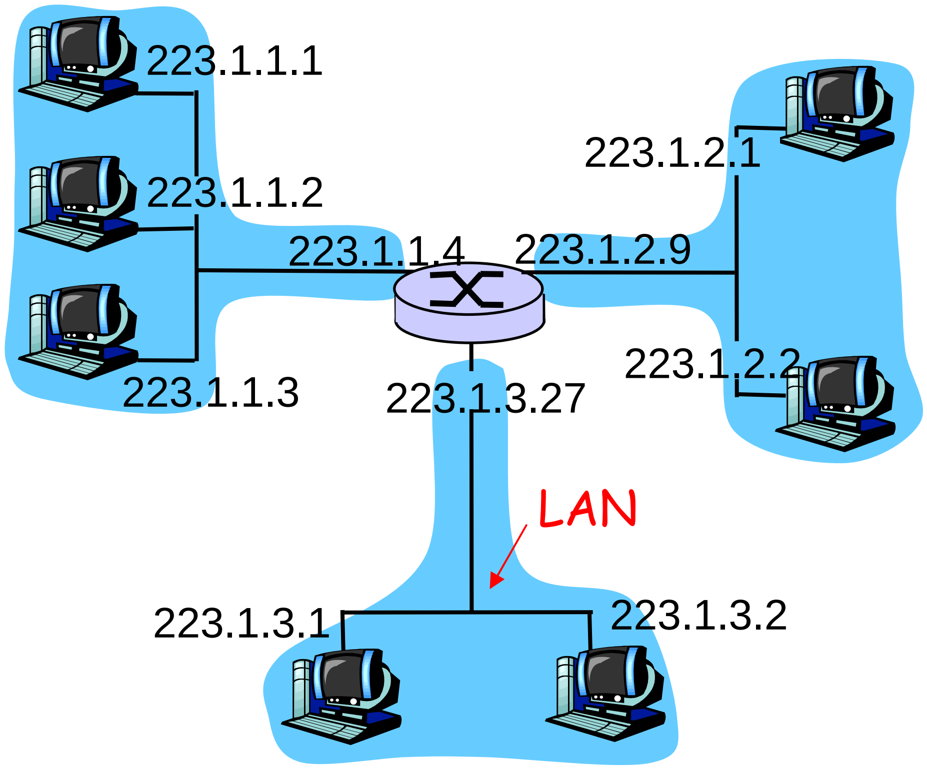
- IP address: 32-bit identifier for host, router *interface*
- *interface:* connection between host/router and physical link
  - router's typically have multiple interfaces
  - host may have multiple interfaces
  - IP addresses associated with each interface



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

IP Addressing

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

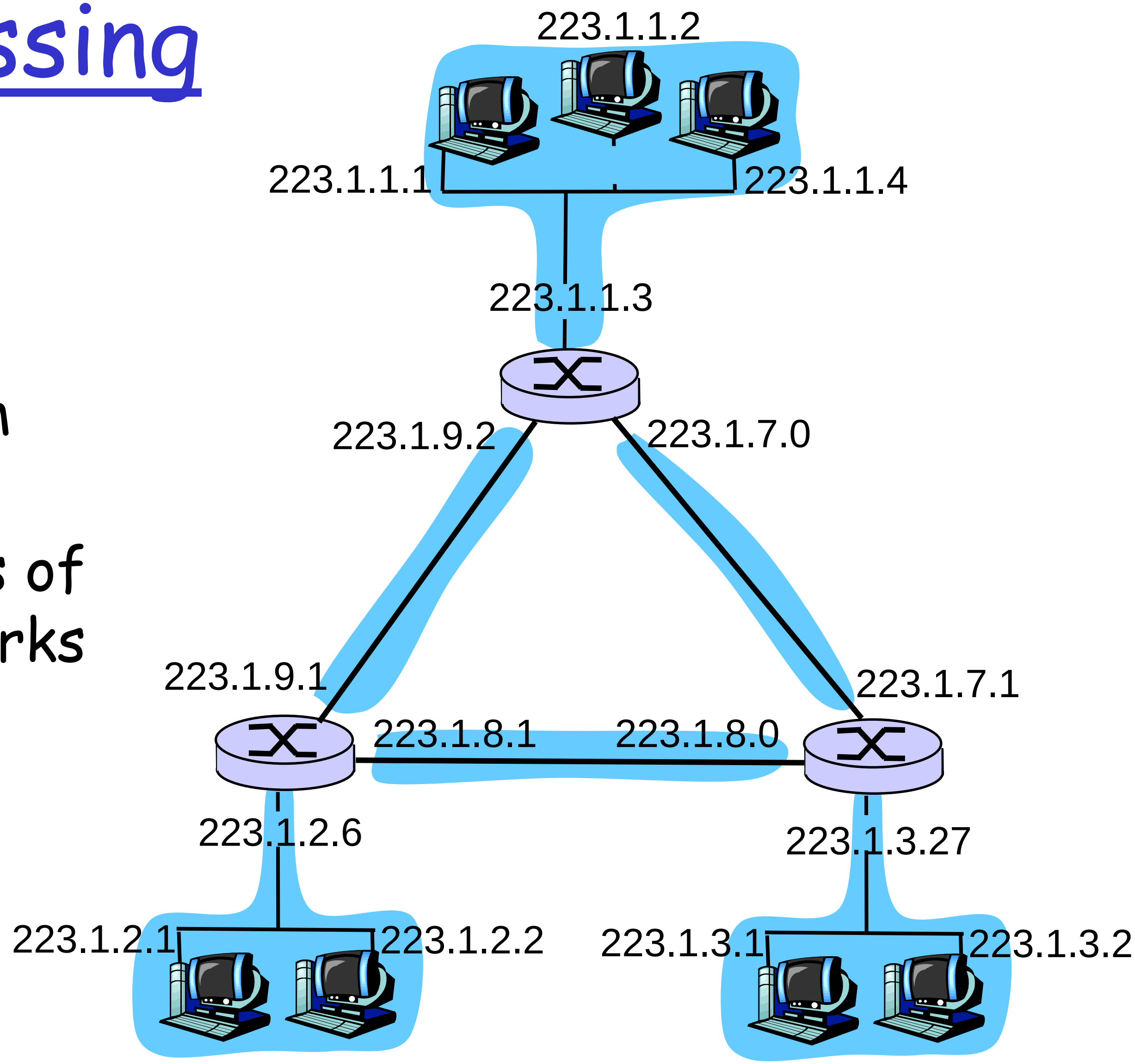


network consisting of 3 IP networks (for IP addresses starting with 223, first 24 bits are network address)

IP Addressing

How to find the networks?

- Detach each interface from router, host
- create "islands of isolated networks

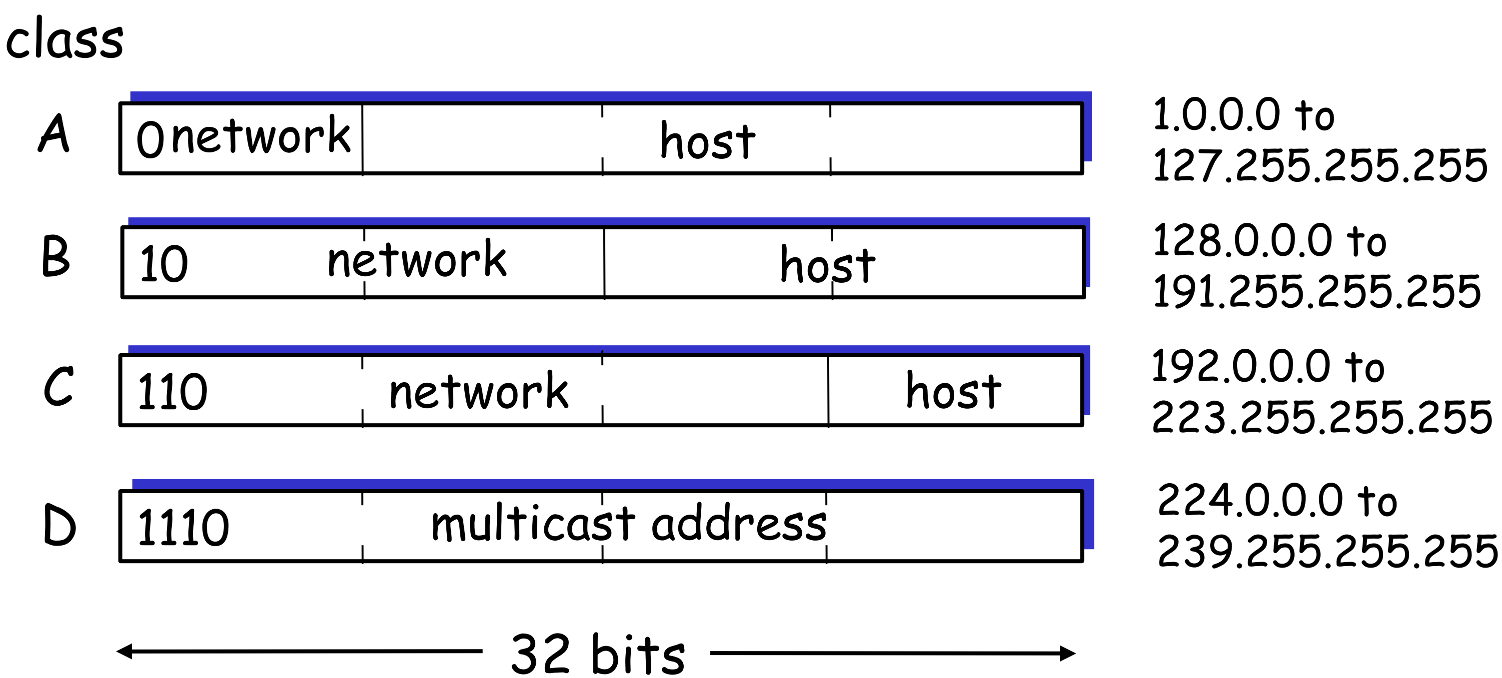


Interconnected system consisting of six networks

IP Addresses

given notion of “network”, let’s re-examine IP addresses:

“class-full” addressing:



Subnetting

- Problem 1: Any network with need for more than 255 hosts, needed class B addresses, or get many class C addresses
- Problem 2: Each new network implies additional entry in forwarding table → large table
- Solution:
  - Share one network number between several networks.

…Subnetting

- Made most sense for large corporations or campuses
- Corporation networks share 1 network number
- Number of other networks *within* the corporation, using subnet masks
  - E.g. a class B address, is shared among 8 networks, by using a 19-bit "subnet mask" (255.255.224.0 = 11111111 11111111 11100000 00000000)
  - I.e. subnet addresses are defined by 1st 19 bits of the IP address. → Host part now has a "subnet" part in it.
- Class B network address continues to be advertised to the rest of the Internet, subnetting only used "within campus"

IP addressing: CIDR

- 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
- CIDR: Classless InterDomain Routing
  - network portion of address of arbitrary length
  - address format: a.b.c.d/x, where x is # bits in network portion of address

Class: pronounce this as CIDER

network part | host part

11001000 00010111 00010000 00000000

200.23.16.0/23

CIDR vs Subnetting?

- Subnetting:
  - Proposed and used under classfull addressing
- CIDR: Fully classless
- Routing table entries are now:

Network address, subnet mask, Interface

IP addresses: how to get one?

Q: How does *host* get IP address?

- hard-coded by system admin in a file
  - Wintel: 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"

(more shortly)

IP addresses: how to get one?

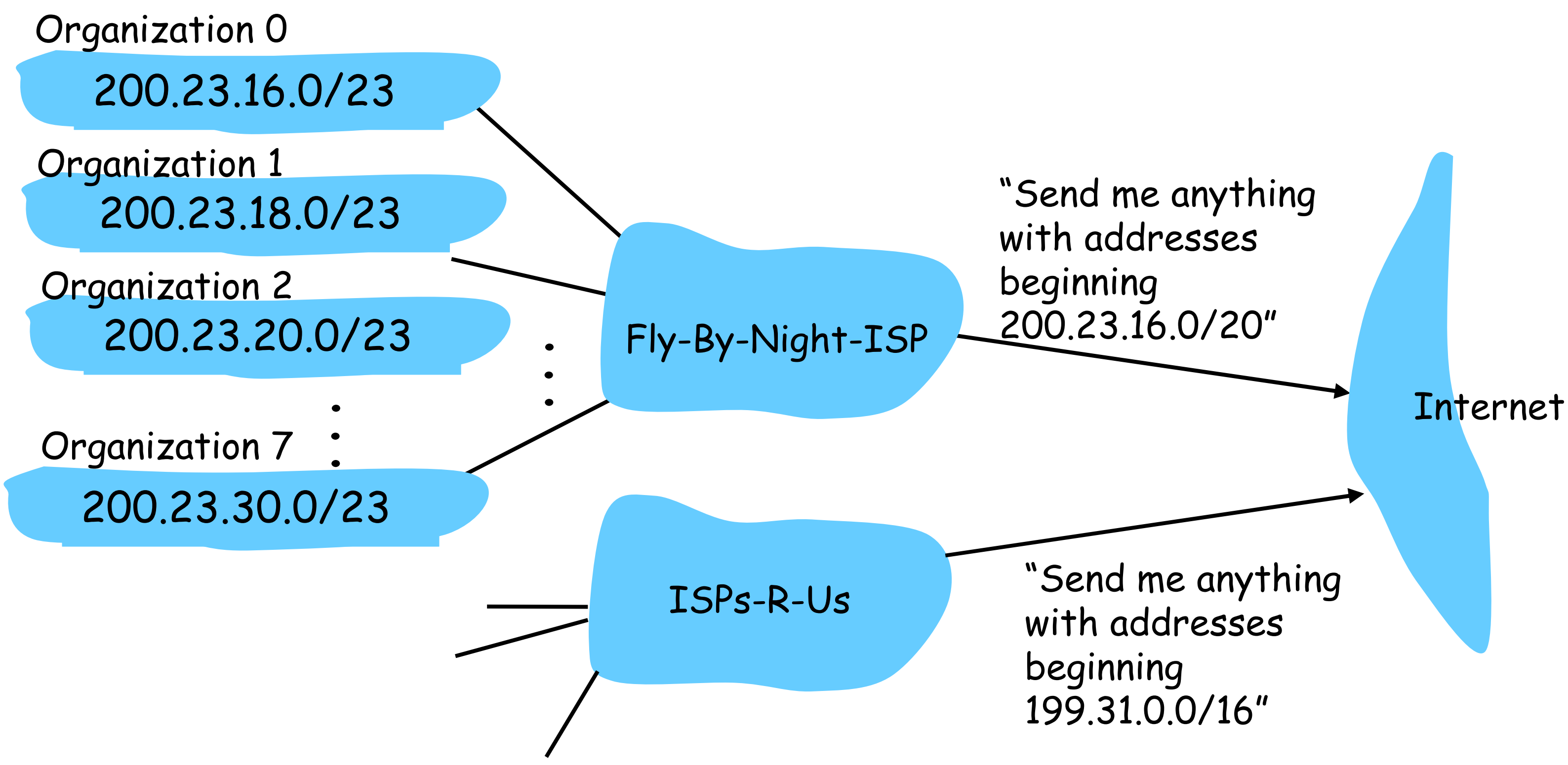
Q: How does *network* get network part of IP addr?

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

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

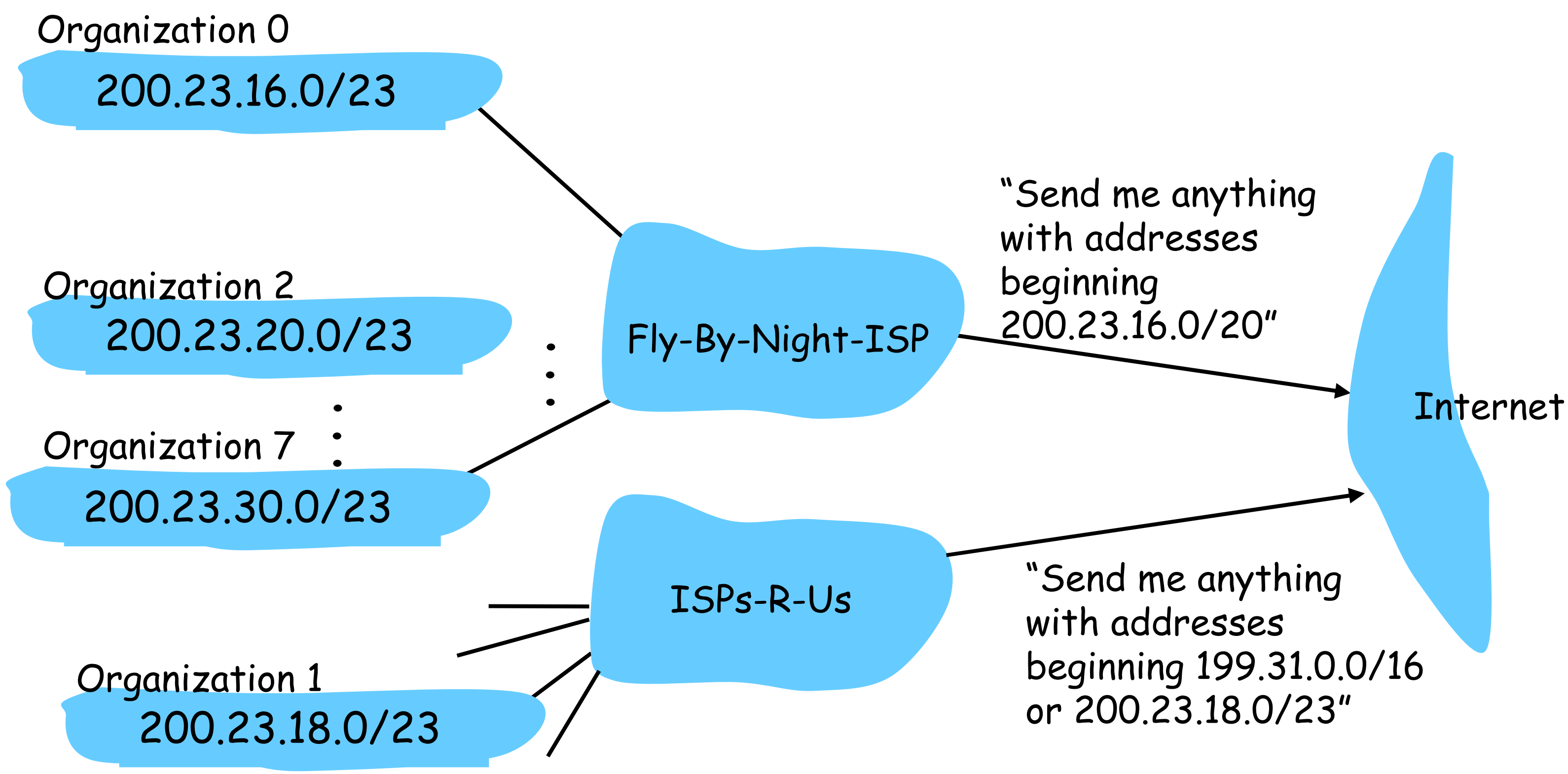
Hierarchical addressing: route aggregation

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

Getting a datagram from source to dest.

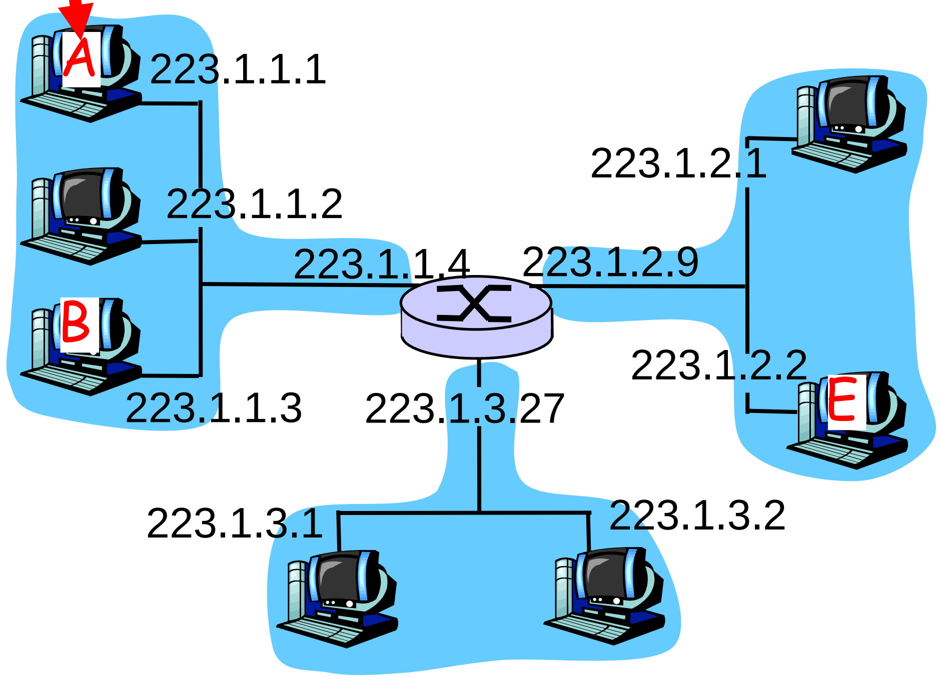
forwarding table in A

Dest. Net.	next router	Nhops
223.1.1		1
223.1.2	223.1.1.4	2
223.1.3	223.1.1.4	2

IP datagram:

misc fields	source IP addr	dest IP addr	data

- datagram remains unchanged, as it travels source to destination
- addr fields of interest here



Getting a datagram from source to dest.

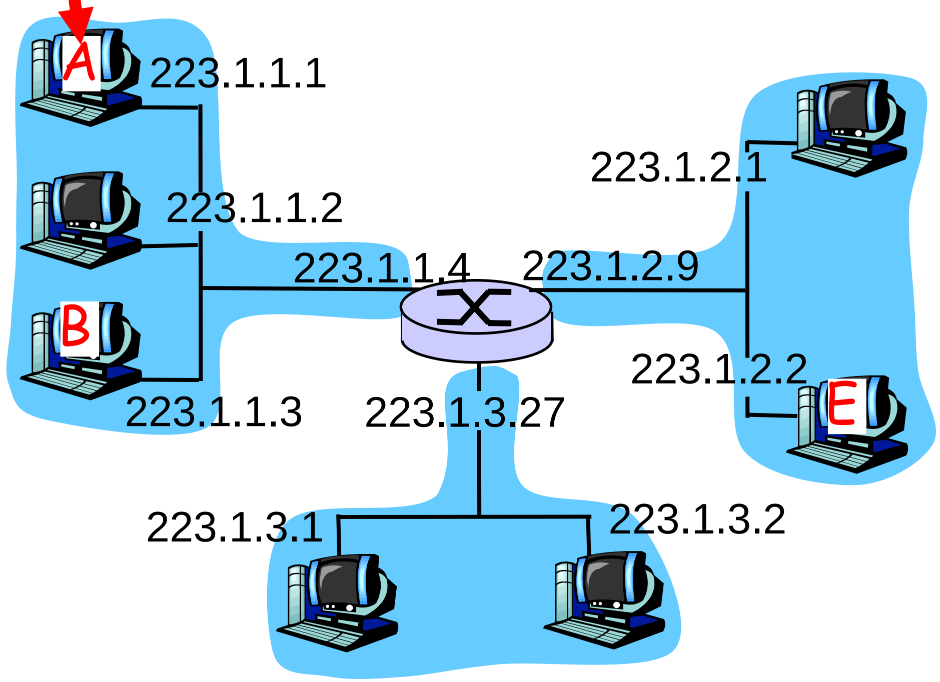
misc fields	223.1.1.1	223.1.1.3	data

forwarding table in A

Dest. Net.	next router	Nhops
223.1.1		1
223.1.2	223.1.1.4	2
223.1.3	223.1.1.4	2

Starting at A, send IP datagram addressed to B:

- look up net. address of B in forwarding table
- find B is on same net. as A
- link layer will send datagram directly to B inside link-layer frame
  - B and A are directly connected

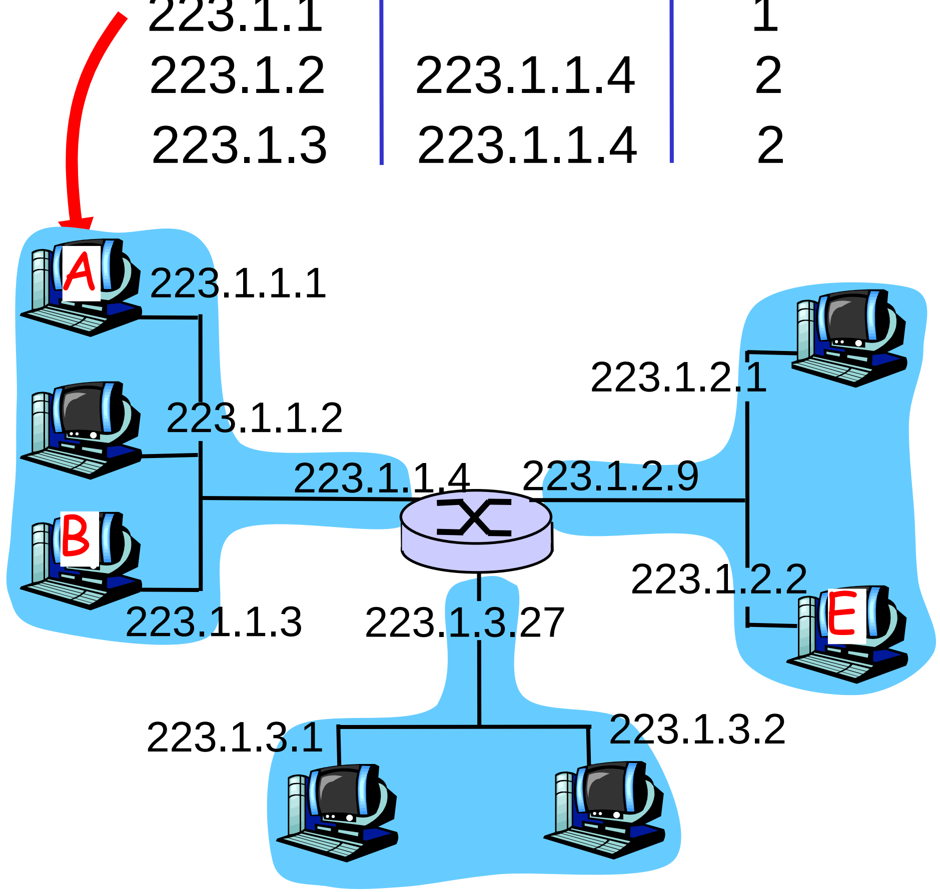


Getting a datagram from source to dest.

misc fields	223.1.1.1	223.1.2.3	data

forwarding table in A

Dest. Net.	next router	Nhops
223.1.1		1
223.1.2	223.1.1.4	2
223.1.3	223.1.1.4	2



Starting at A, dest. E:

- look up network address of E in forwarding table
- E on *different* network
  - A, E not directly attached
- routing table: next hop router to E is 223.1.1.4
- link layer sends datagram to router 223.1.1.4 inside link-layer frame
- datagram arrives at 223.1.1.4
- continued…..

Getting a datagram from source to dest.

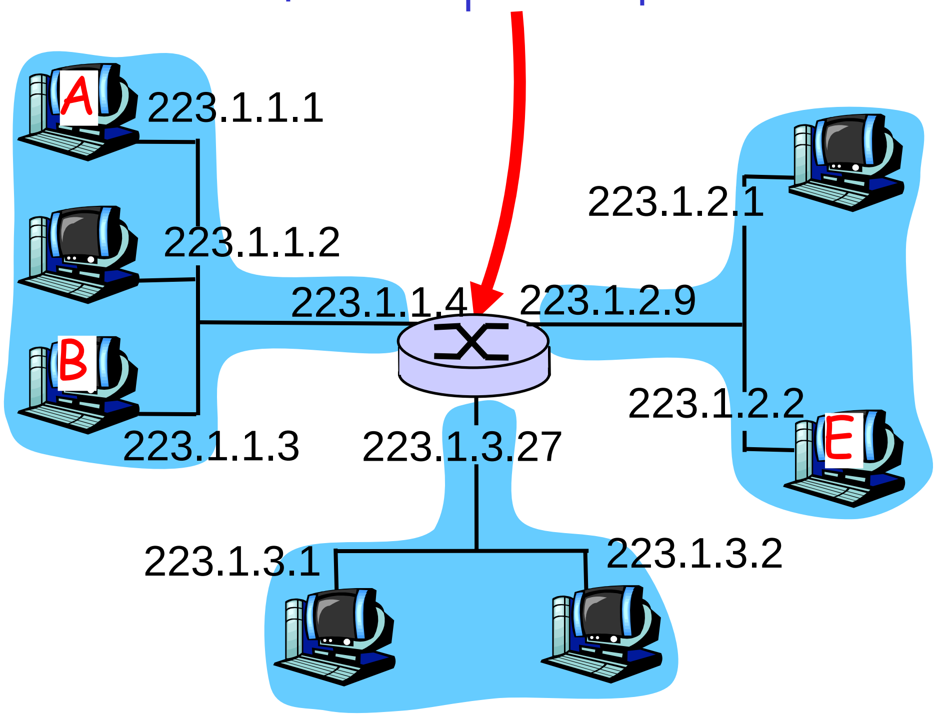
misc fields	223.1.1.1	223.1.2.3	data

forwarding table in router

Dest. Net	router	Nhops	interface
223.1.1	-	1	223.1.1.4
223.1.2	-	1	223.1.2.9
223.1.3	-	1	223.1.3.27

Arriving at 223.1.4, destined for 223.1.2.2

- look up network address of E in router's forwarding table
- E on *same* network as router's interface 223.1.2.9
  - router, E directly attached
- link layer sends datagram to 223.1.2.2 inside link-layer frame via interface 223.1.2.9
- datagram arrives at 223.1.2.2!!! (hooray!)



Forwarding Ex. with Subnet Masks

- Routing Table:

SubnetNumber	SubnetMask	NextHop
128.96.170.0	255.255.254.0	Intface 0
128.96.168.0	255.255.254.0	Intface 1
128.96.166.0	255.255.254.0	R2
128.96.164.0	255.255.252.0	R3
Default		R4

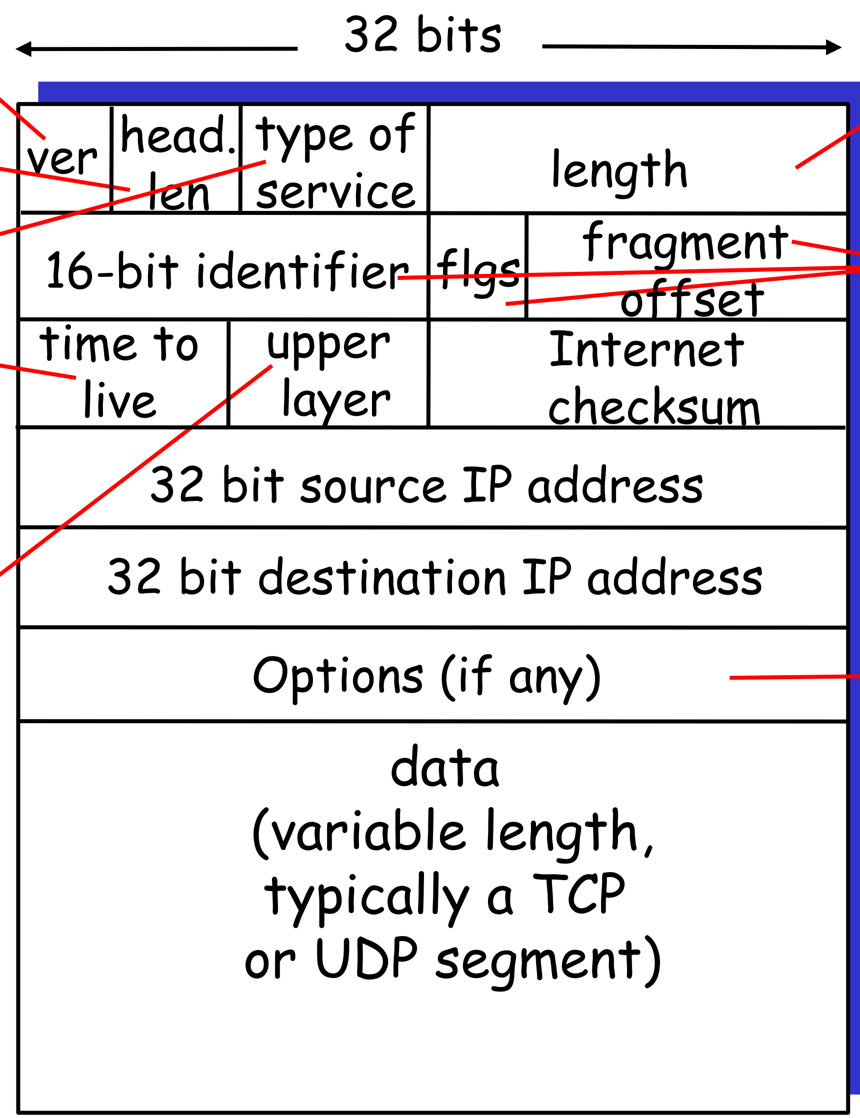
1. 128.96.171.92 Interface 0
2. 128.96.167.151 R2
3. 128.96.163.151 R4
4. 128.96.169.192 Interface 1
5. 128.96.165.121 R3

Forwarding Ex. with Subnet Masks

SubnetNumber	SubnetMask	NextHop
128.96.170.0 (128.96.170.0 – 128.96.171.255)	255.255.254.0 8+8+7=23 bits net (9 bits host)	Intface 0
128.96.169.0 128.96.1010101?.???????? 128.96.168.0-128.96.169.255	255.255.254.0 23 bits net/9 bits host	Intface 1
128.96.166.0 128.96.166.0-128.96.167.255	255.255.254.0 23 bits net/9 bits host	R2
128.96.164.0 128.96.164.0-128.96.167.255	255.255.252.0 22 bits net/ 10 bits host	R3
Default		R4

1. 128.96.171.92: Iface0
2. 128.96.167.151: R2&R3 so R2
3. 128.96.163.151: R4
4. 128.96.169.192: I fac1
5. 128.96.165.121: R3

IP datagram format

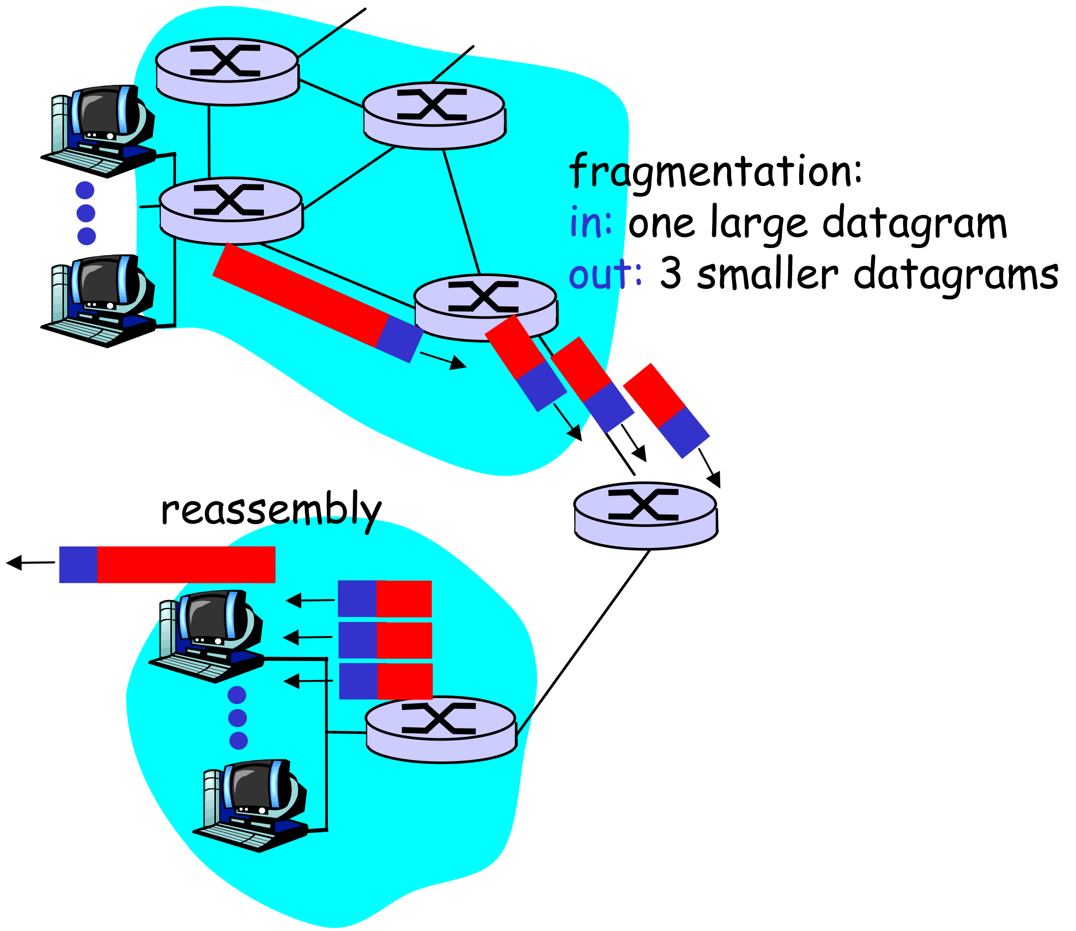


<u>how much overhead with TCP?</u>

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

IP Fragmentation & Reassembly

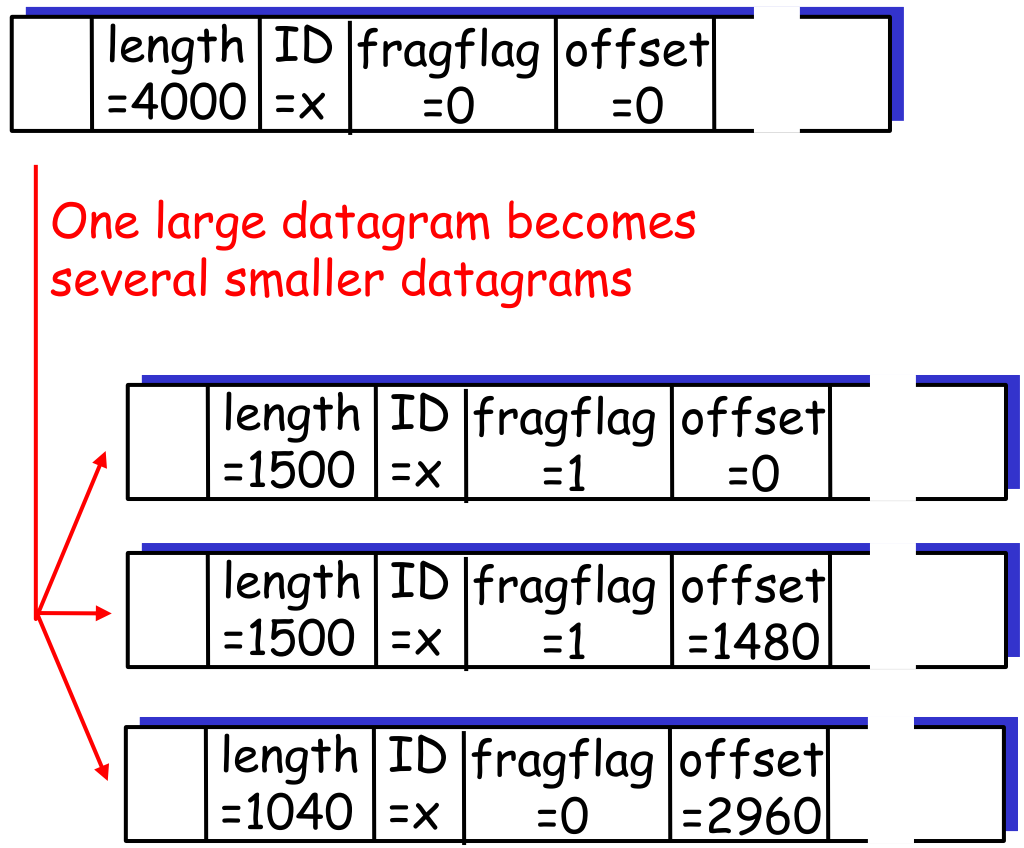
- network links have MTU (max.transfer size) - largest possible link-level frame.
  - different link types, different MTUs
- Design choice: datagram size = smallest MTU (problems?)
- large IP datagram divided ("fragmented") within net
  - one datagram becomes several datagrams
  - "reassembled" only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



IP Fragmentation and Reassembly

Example

- 4000 byte datagram
- MTU = 1500 bytes



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 Examples

- ICMP-Redirect: Router R1 can send back to host H that R2 is a better router for some destination
- Trace-route: Implemented using ICMP, and the TTL field. How?
  - Send a sequence of packets, starting with TTL = 1 and increasing. For TTL = n, the n^{th} router will send back an error message 11 (and its address in the source address field).
  - Timer for finding RTT

Chapter 4 roadmap

4.1 Introduction and Network Service Models
4.2 Routing Principles
4.3 Hierarchical Routing
4.4 The Internet (IP) Protocol
- 4.4.1 IPv4 addressing
- 4.4.2 Moving a datagram from source to destination
- 4.4.3 Datagram format
- 4.4.4 IP fragmentation
- 4.4.5 ICMP: Internet Control Message Protocol
- 4.4.6 DHCP: Dynamic Host Configuration Protocol
- 4.4.7 NAT: Network Address Translation

4.5 Routing in the Internet
4.6 What's Inside a Router
4.7 IPv6
4.8 Multicast Routing
4.9 Mobility

DHCP: Dynamic Host Configuration Protocol

Goal:

Allow reuse of addresses (only hold address while connected and "on"). Support many more machines this way.

Support for mobile users who want to join network (more shortly)

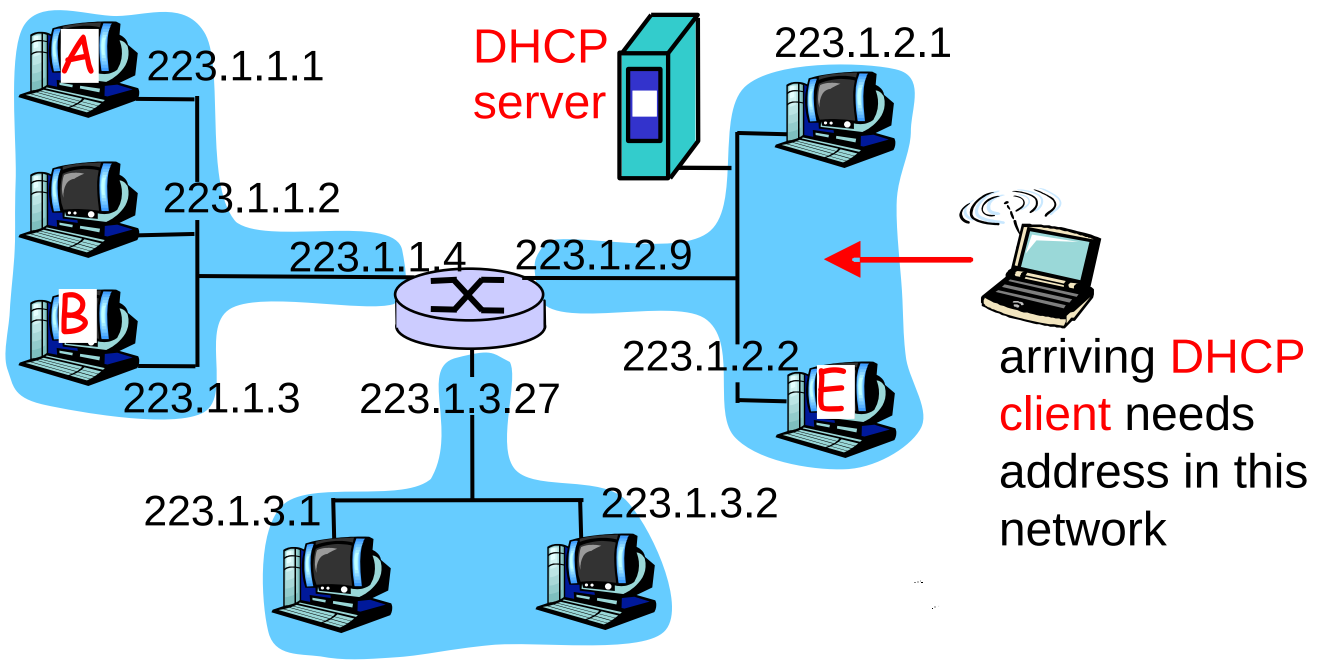
allow host to *dynamically* obtain its IP address from network server when it joins network

Can renew its lease on address in use

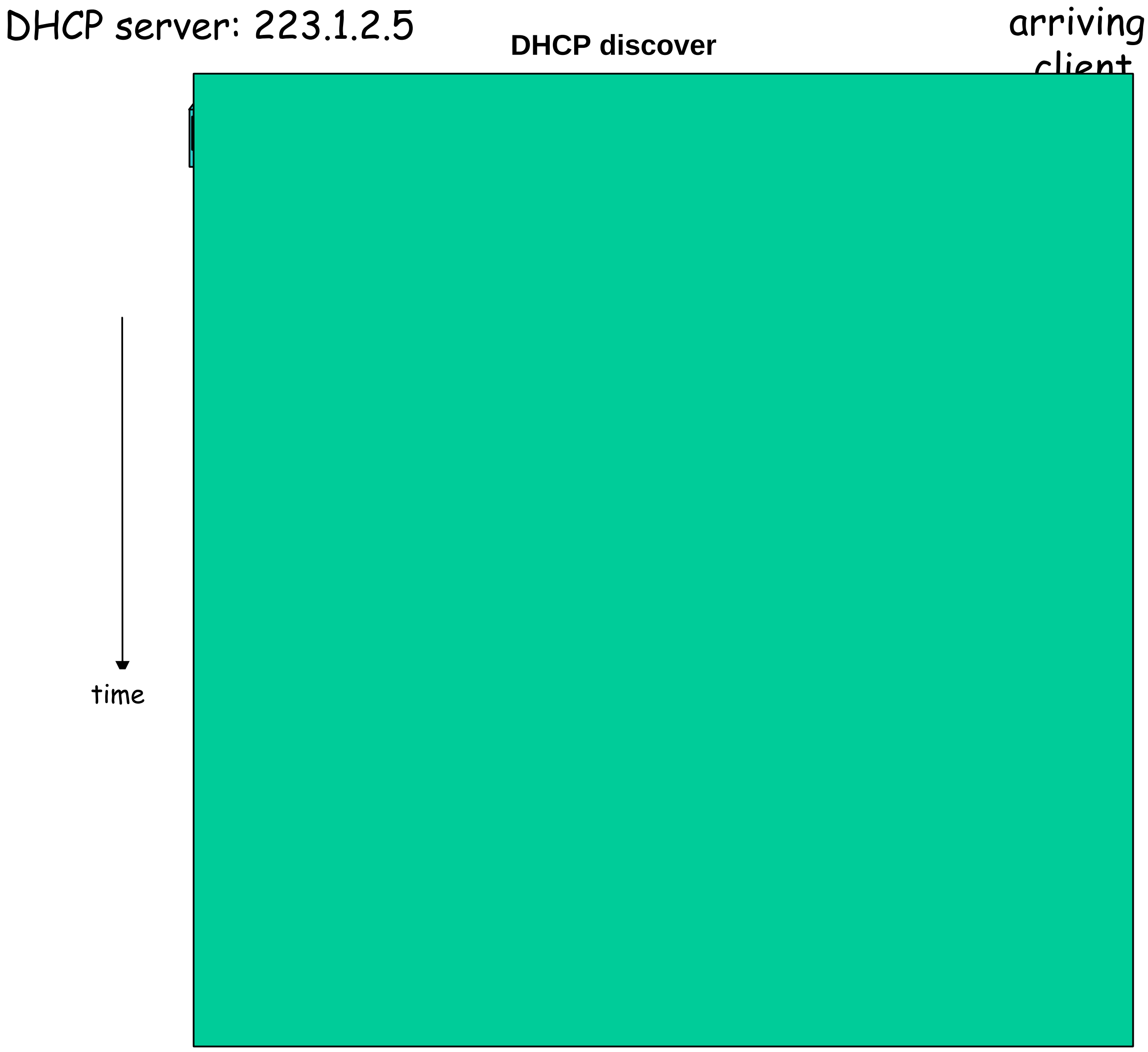
DHCP overview:

- 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



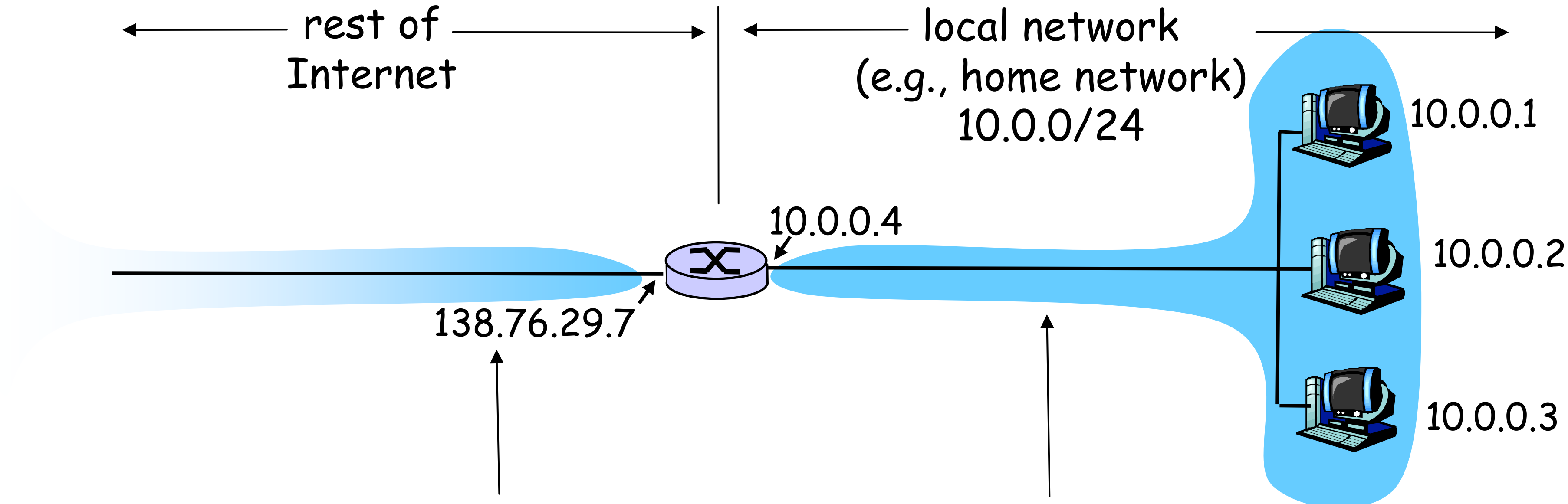
DHCP

- Network management: Easy or difficult?
  - Easier configuration
  - Harder isolation of malfunction

NAT: Network Address Translation

- IP address management within organizations should be easy –
  - Flexible w.r.t. growth of machines
  - Not encumbered by "global" addressing problems
- Solution: (Albeit HACKY)
  - NAT

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

- Local network uses just one IP address as far as outside world is concerned:
  - no need to be allocated range of addresses from ISP: - just one IP address is used for all devices
  - can change addresses of devices in local network without notifying outside world
  - can change ISP without changing addresses of devices in local network
  - devices inside local net not explicitly addressable, visible by outside world (a security plus).

NAT: Network Address Translation

Implementation: NAT router must:

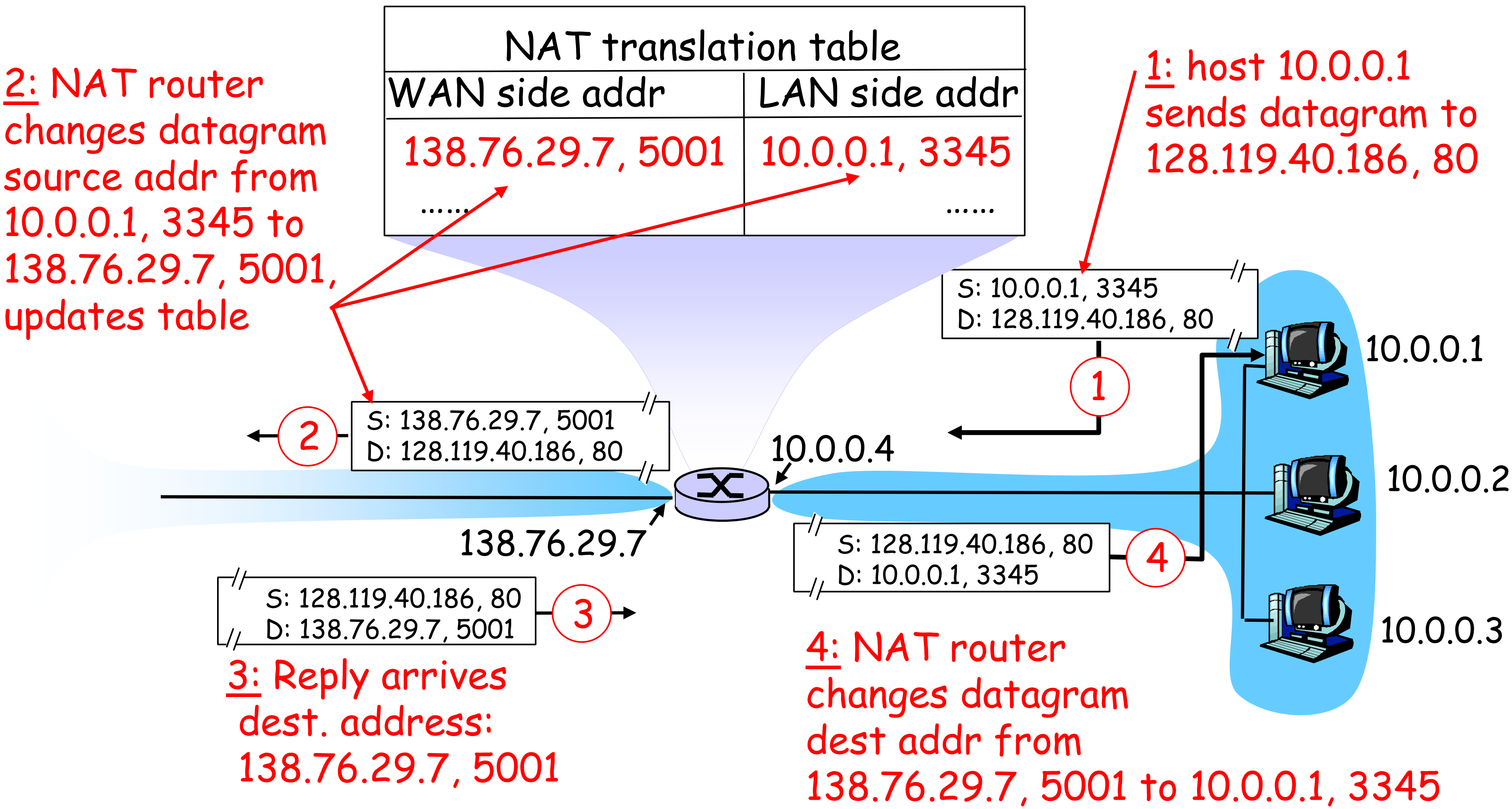
- *outgoing datagrams: replace* (source IP address, port #) of every outgoing datagram to (NAT IP address, new port #)

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

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

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

NAT: Network Address Translation



NAT: Network Address Translation

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

Chapter 4 roadmap

4.1 Introduction and Network Service Models

4.2 Routing Principles

4.3 Hierarchical Routing

4.4 The Internet (IP) Protocol

4.5 Routing in the Internet

- 4.5.1 Intra-AS routing: RIP and OSPF
- 4.5.2 Inter-AS routing: BGP

4.6 What's Inside a Router?

4.7 IPv6

4.8 Multicast Routing

4.9 Mobility

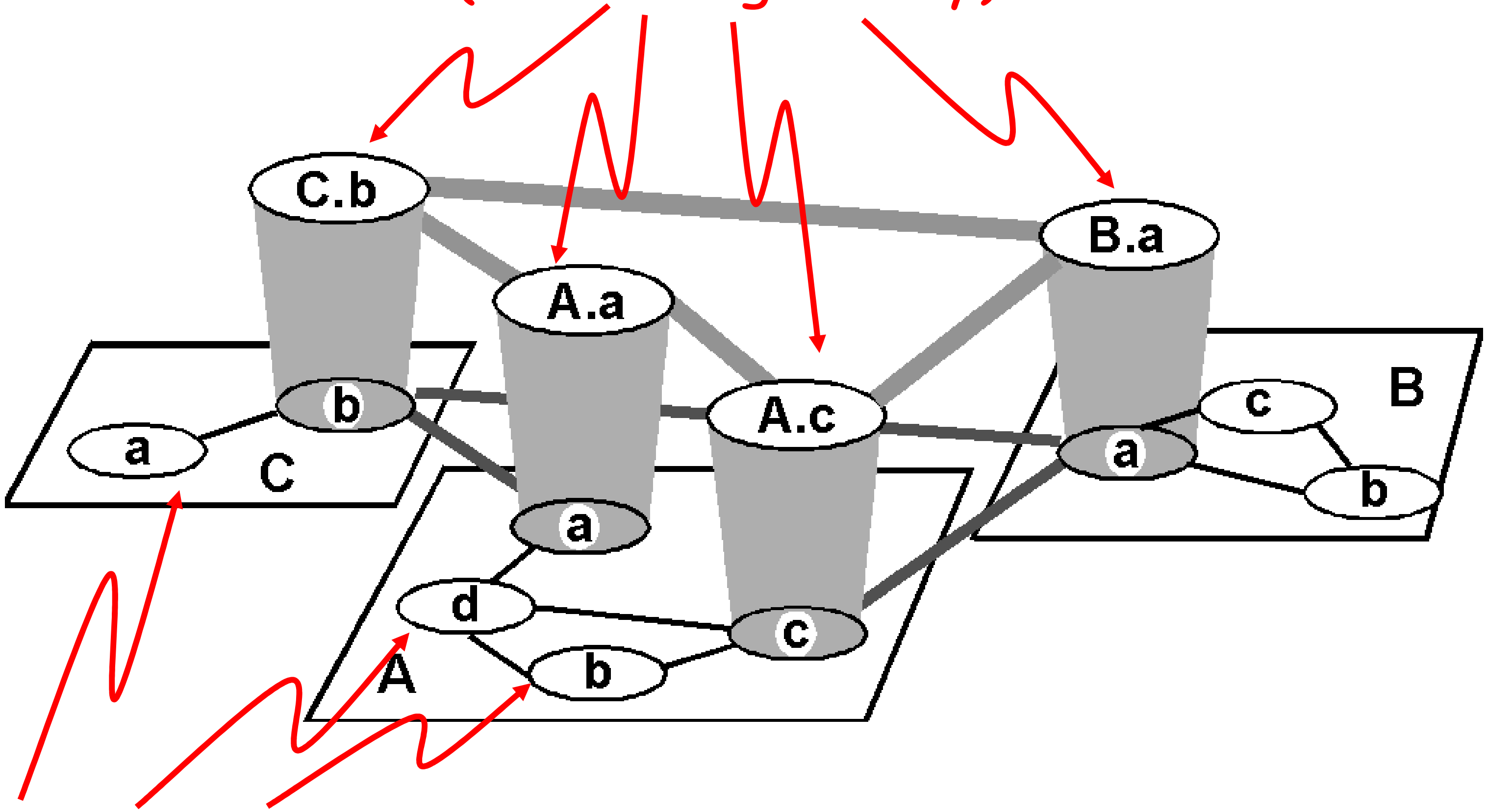
Routing in the Internet

- The Global Internet consists of Autonomous Systems (AS) interconnected with each other:
  - **Stub AS**: small corporation: one connection to other AS's
  - **Multihomed AS**: large corporation (no transit): multiple connections to other AS's
  - **Transit AS**: provider, hooking many AS's together

- Two-level routing:
  - **Intra-AS:** administrator responsible for choice of routing algorithm within network
  - **Inter-AS:** unique standard for inter-AS routing: BGP

Internet AS Hierarchy

Intra-AS border (exterior gateway) routers



Inter-AS interior (gateway) routers

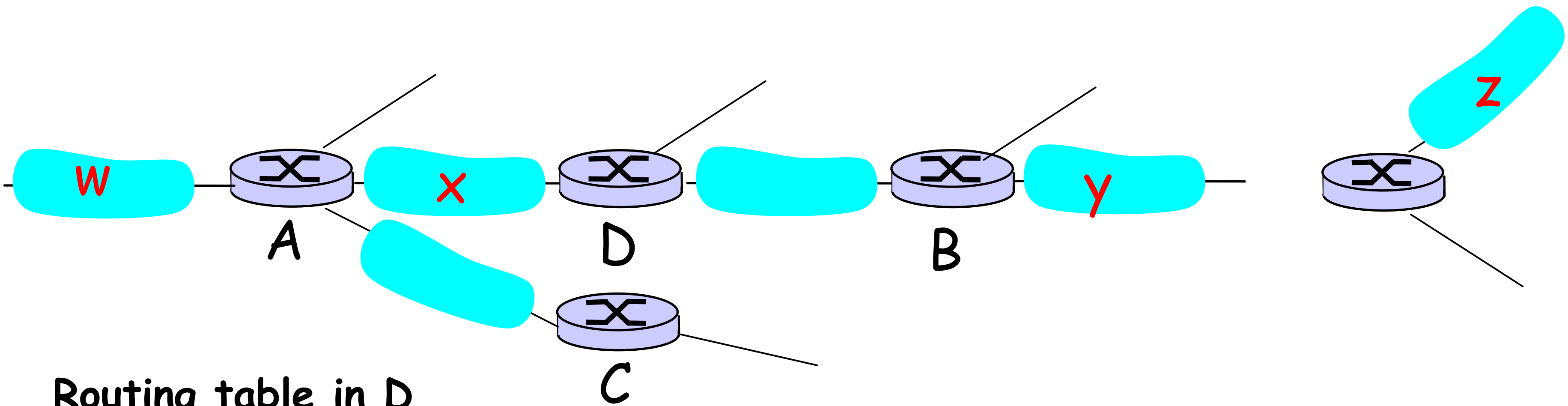
Intra-AS Routing

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

RIP (Routing Information Protocol)

- Distance vector algorithm
- Included in BSD-UNIX Distribution in 1982
- Distance metric: # of hops (max = 15 hops)
- Distance vectors: exchanged among neighbors every 30 sec via Response Message (also called **advertisement**)
- Each advertisement: list of up to 25 destination nets within AS

RIP: Example



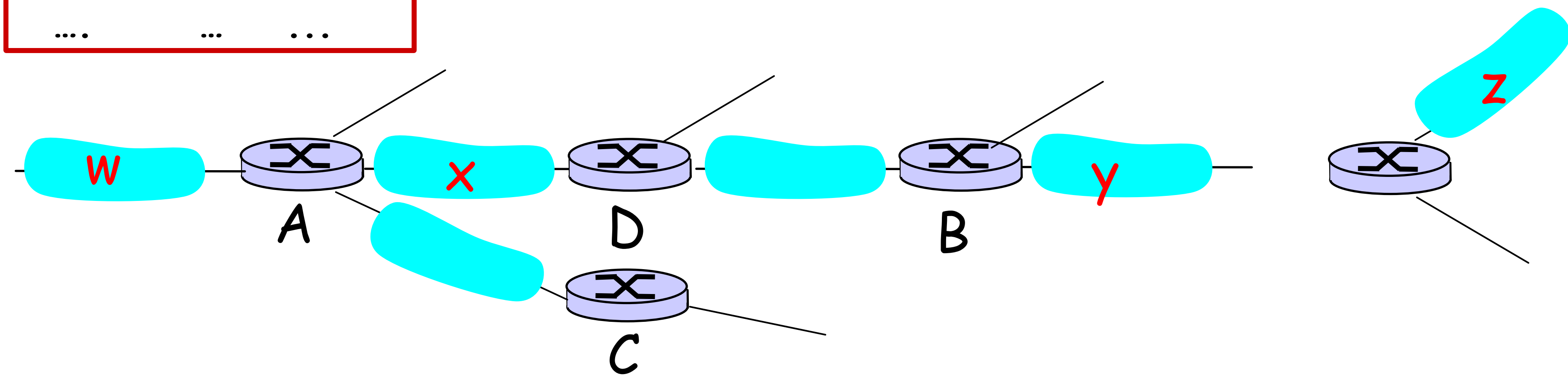
Routing table in D

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

RIP: Example

Dest	Next	hops
w	-	-
x	-	-
z	C	4
….	…	...

Advertisement from A to D



Destination Network	Next Router	Num. of hops to dest.
w	A	2
y	B	2
z	~~B~~ A	~~7~~ 5
x	--	1
….	….	....

Routing table in D

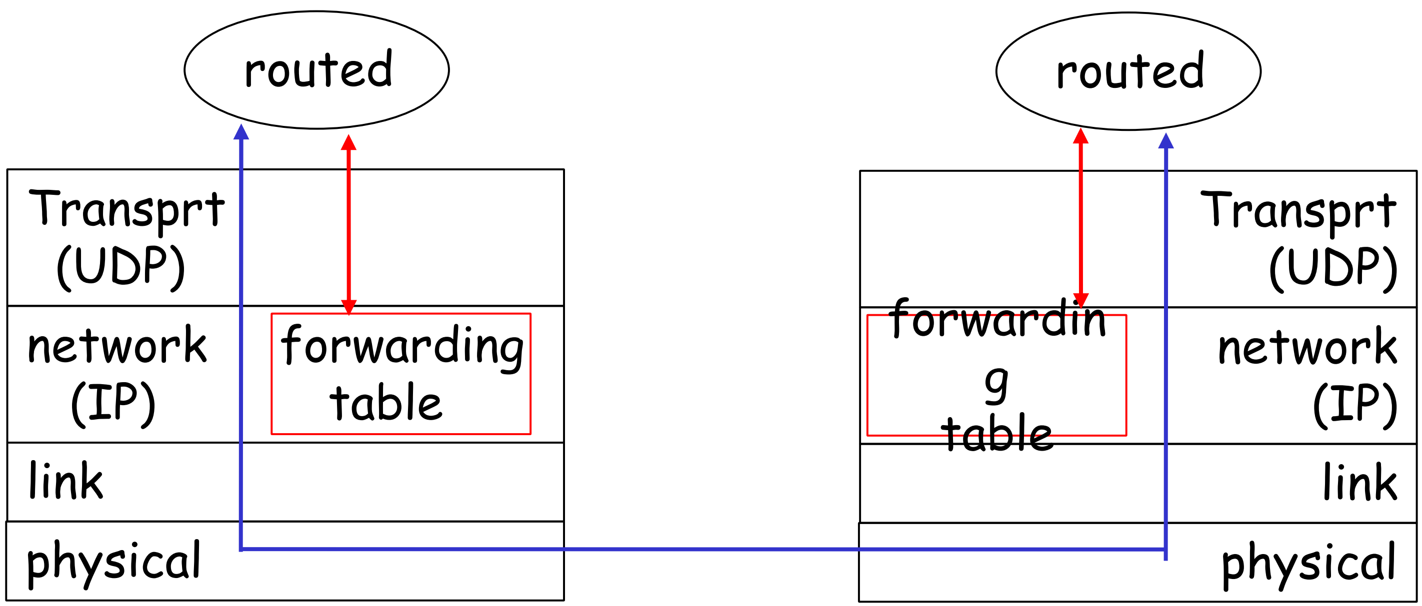
RIP: Link Failure and Recovery

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)
  - "split horizon" is when you don't send anything, poisoned reverse is when you send infinity.

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 (more later)
- Loopback interface (for debugging)

RIP Packet format

0	8	16 … 31
Command	Version	Must be zero
Family of net 1		Address of net 1
Address of net 1		
Distance to net 1		
Family of net 2		Address of net 2
Address of net 2		
Distance to net 2		

OSPF (Open Shortest Path First)

- "open": publicly available
- Uses Link State algorithm
  - LS packet dissemination
  - Topology map at each node
  - Route computation using Dijkstra's algorithm

- OSPF advertisement carries one entry per neighbor router
- Advertisements disseminated to entire AS (via flooding)
  - Carried in OSPF messages directly over IP (rather than TCP or UDP

OSPF

- Link State Packet (or Advertisement)
  - id of the node that created the LSP
  - cost of link to each directly connected neighbor
  - sequence number (SEQNO)
  - time-to-live (TTL) for this packet
- First two items for route calculation
- Last two for correct operation
  - Multiple LSPs may be traveling in the network, sequence number helps distinguish that
- LSP generated when
  - a node's link costs change
  - Periodically ("hello" message)

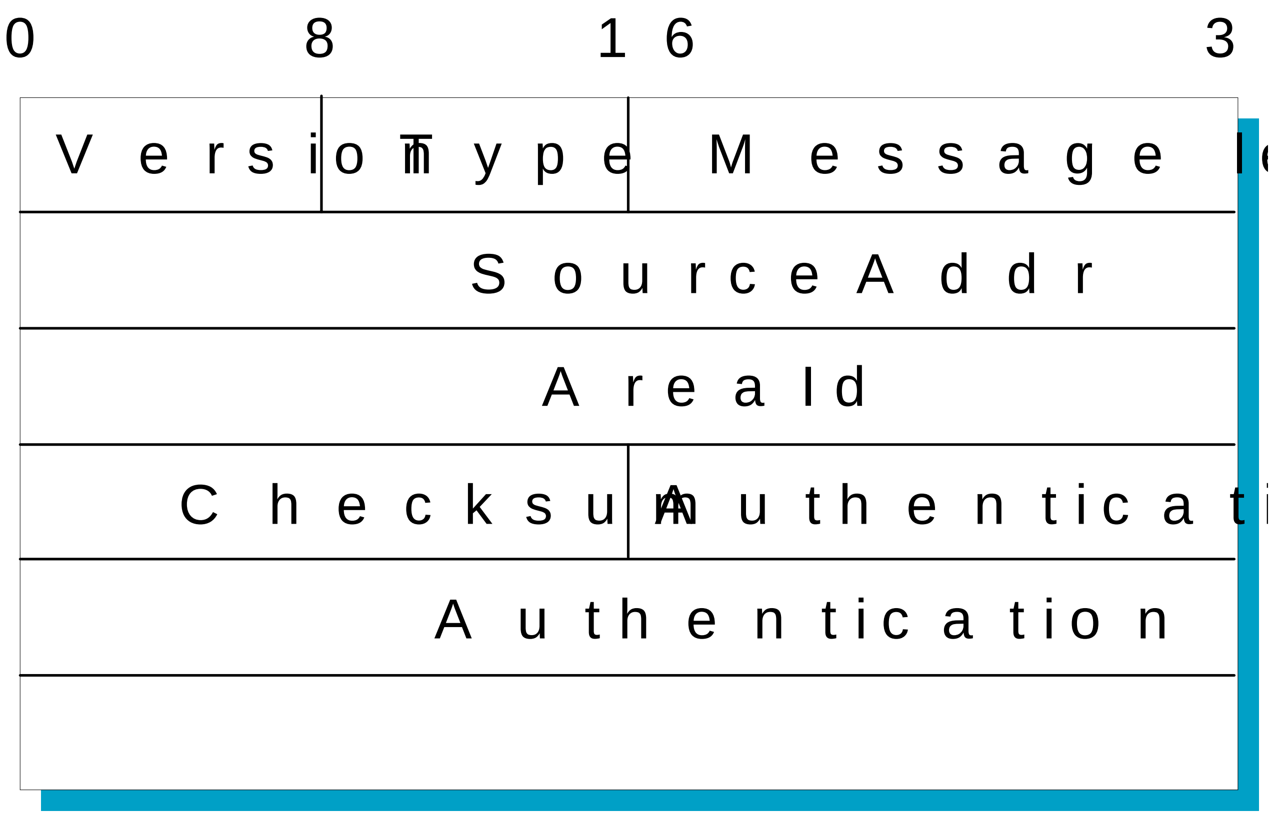
OSPF: Reliable Flooding

- store most recent LSP from each node
- forward LSP to all nodes but one that sent it
- generate new LSP periodically
    - increment SEQNO
- start SEQNO at 0 when reboot
- decrement TTL of each stored LSP
    - discard when TTL=0
        - Ensures removal of old information
- Also "age" LSP while stored at node, by decrementing TTL
    - When TTL reaches 0, re-flood network with LSP with TTL=0, this ensures deletion of the LSP

OSPF "advanced" features (not in RIP)

- Security: all OSPF messages authenticated (to prevent malicious intrusion)
- Multiple same-cost paths allowed (only one path in RIP). Can implement load-balancing.
- For each link, multiple cost metrics for different TOS (e.g., 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 (AS can be subdivided into *areas*.)

OSPF Header format



Header

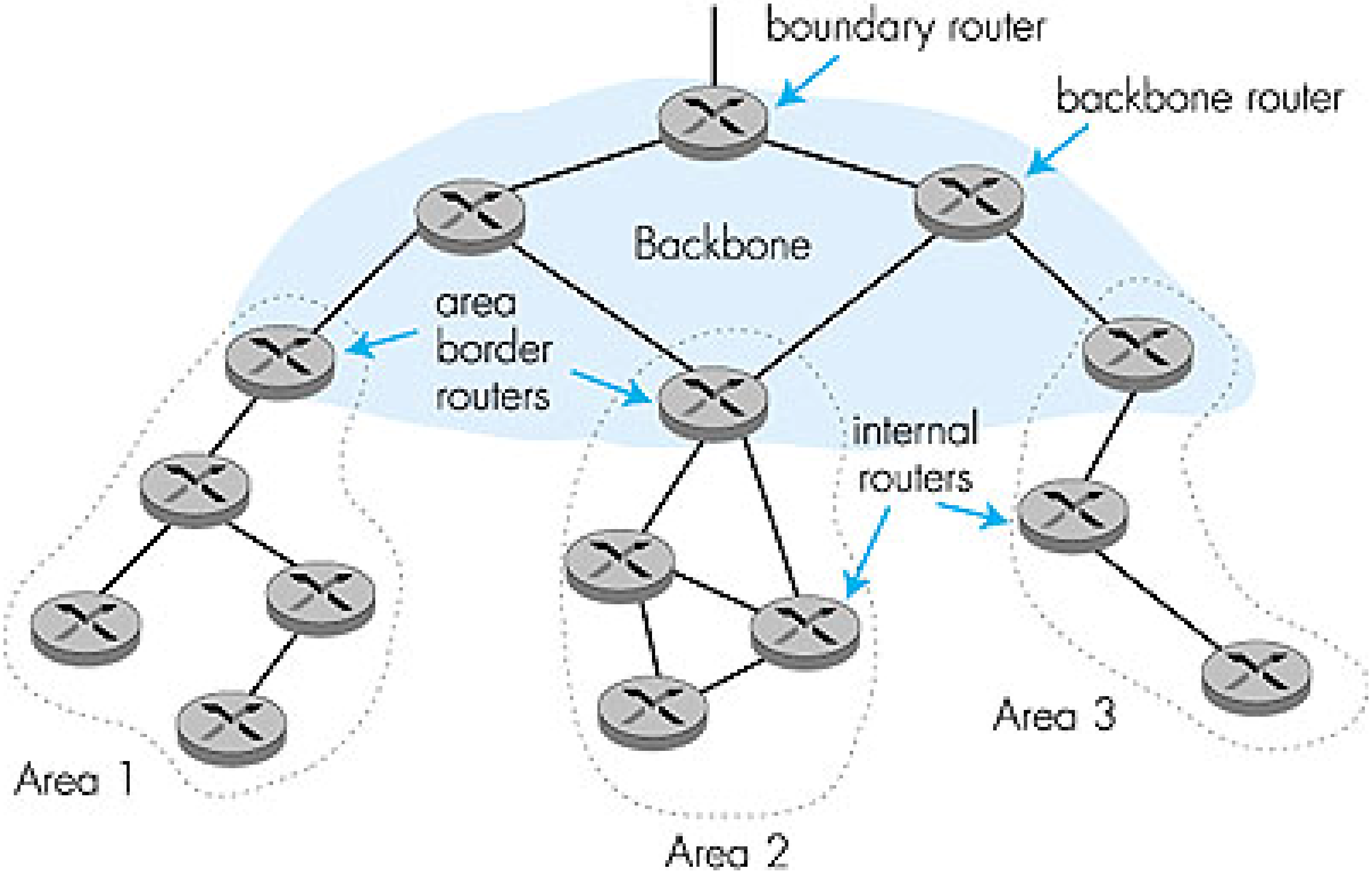
Link State Advertisement

Type 1 = link state advertisement

<table>
<tr><td colspan="3">LS Age</td><td>Options</td><td>Type=1</td></tr>
<tr><td colspan="5">Link-state ID</td></tr>
<tr><td colspan="5">Advertising router</td></tr>
<tr><td colspan="5">LS sequence number</td></tr>
<tr><td colspan="3">LS checksum</td><td colspan="2">Length</td></tr>
<tr><td>0</td><td>Flags</td><td>0</td><td colspan="2">Number of links</td></tr>
<tr><td colspan="5">Link ID</td></tr>
<tr><td colspan="5">Link data</td></tr>
<tr><td colspan="2">Link type</td><td>Num_TOS</td><td colspan="2">Metric</td></tr>
<tr><td colspan="5">Optional TOS information</td></tr>
<tr><td colspan="5">More links</td></tr>
</table>

- LS age ~= TTL
- Linkstate ID = adv. Router for type 1
- LS checksum – everything except age
- LinkID/Link Data: id of link
- Metric = cost
- Type: about link (e.g. p2p)

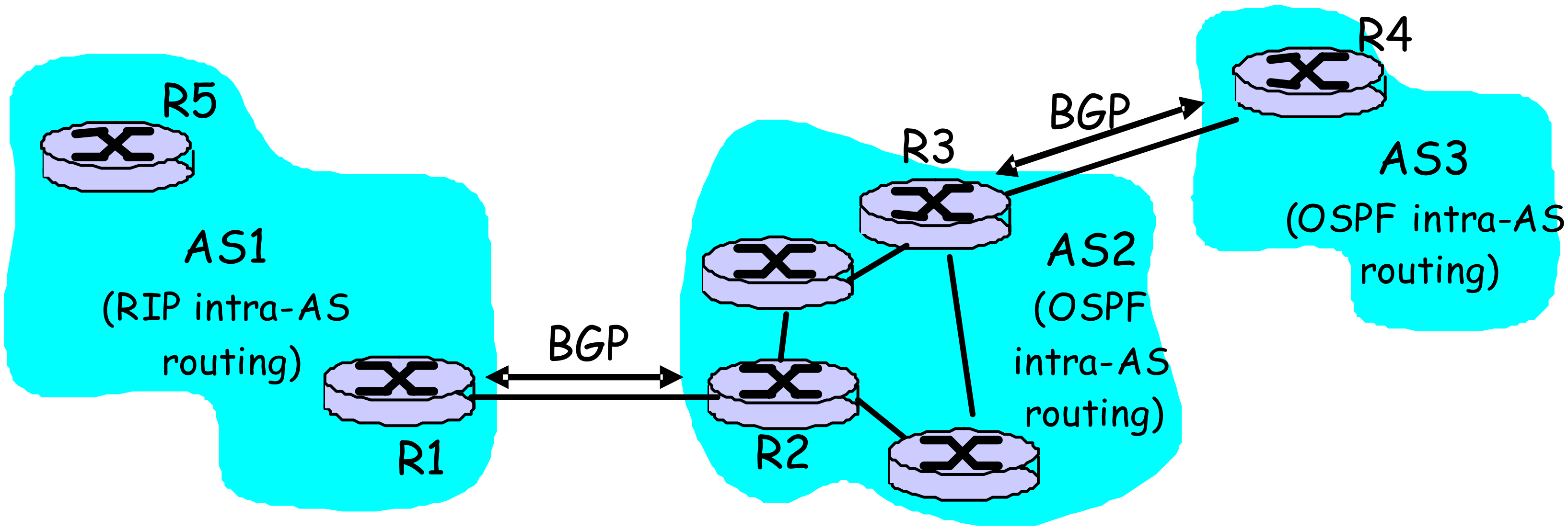
Hierarchical OSPF



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 AS's.

Inter-AS routing in the Internet: BGP



BGP: Design goals and challenges

- Goal:
  - Leave "optimality" aside
  - Just *find* a loop-free path
- Thus only bothers with *reachability*
- Why?
  - Buck stops here – backbone routers must be able to route everywhere
  - Variability of metrics used by different ASes
  - Trust!
  - Policies.

Internet inter-AS routing: BGP

- BGP (Border Gateway Protocol): *the* de facto standard
  - Requires AS numbers, assigned by ICAAN
- **Path Vector** protocol:
  - similar to Distance Vector protocol
  - each Border Gateway broadcast to neighbors (peers) *entire path* (i.e., sequence of AS's) to destination
  - BGP routes to networks (ASs), not individual hosts
  - E.g., Gateway X may send its path to dest. Z:

Path (X,Z) = X,Y1,Y2,Y3,…,Z

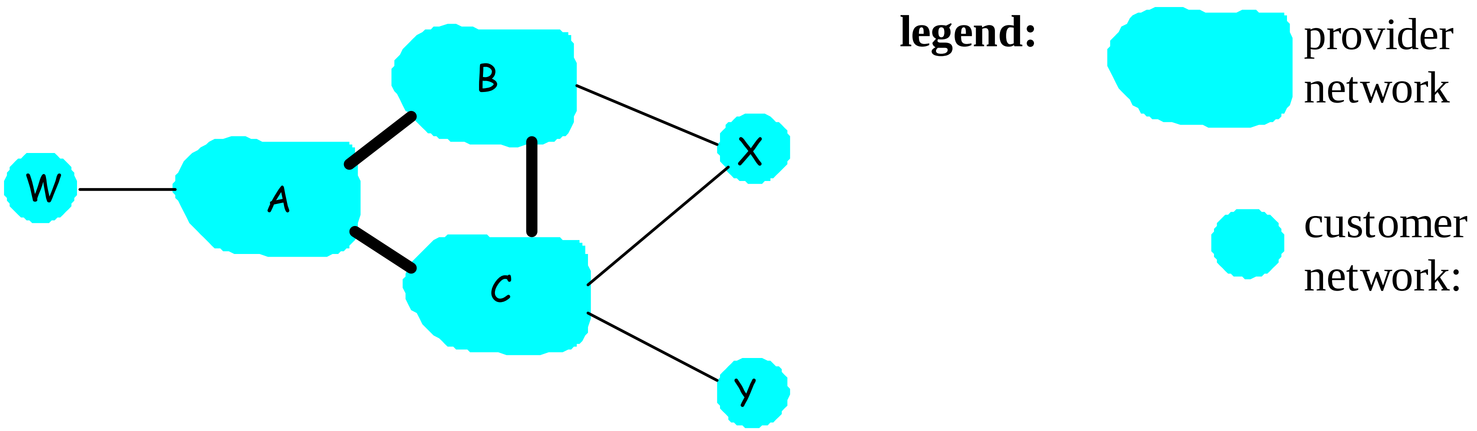
Internet inter-AS routing: BGP

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:

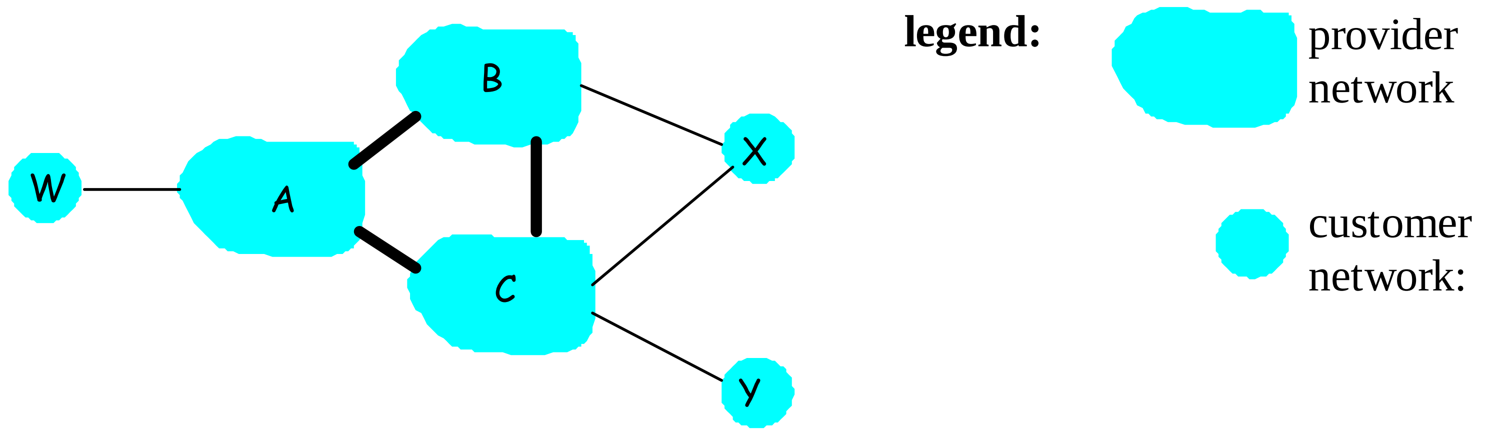
  Path (W,Z) = w, Path (X,Z)
- Note: X can control incoming traffic by controlling it's route advertisements to peers:
  - e.g., don't want to route traffic to Z -> don't advertise any routes to Z

BGP: controlling who routes to you



- 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: controlling who routes to you



- 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!

BGP operation

Q: What does a BGP router do?

- Receiving and filtering route advertisements from directly attached neighbor(s).
- Route selection.
  - To route to destination X, which path (of several advertised) will be taken?
- Sending route advertisements to neighbors.

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

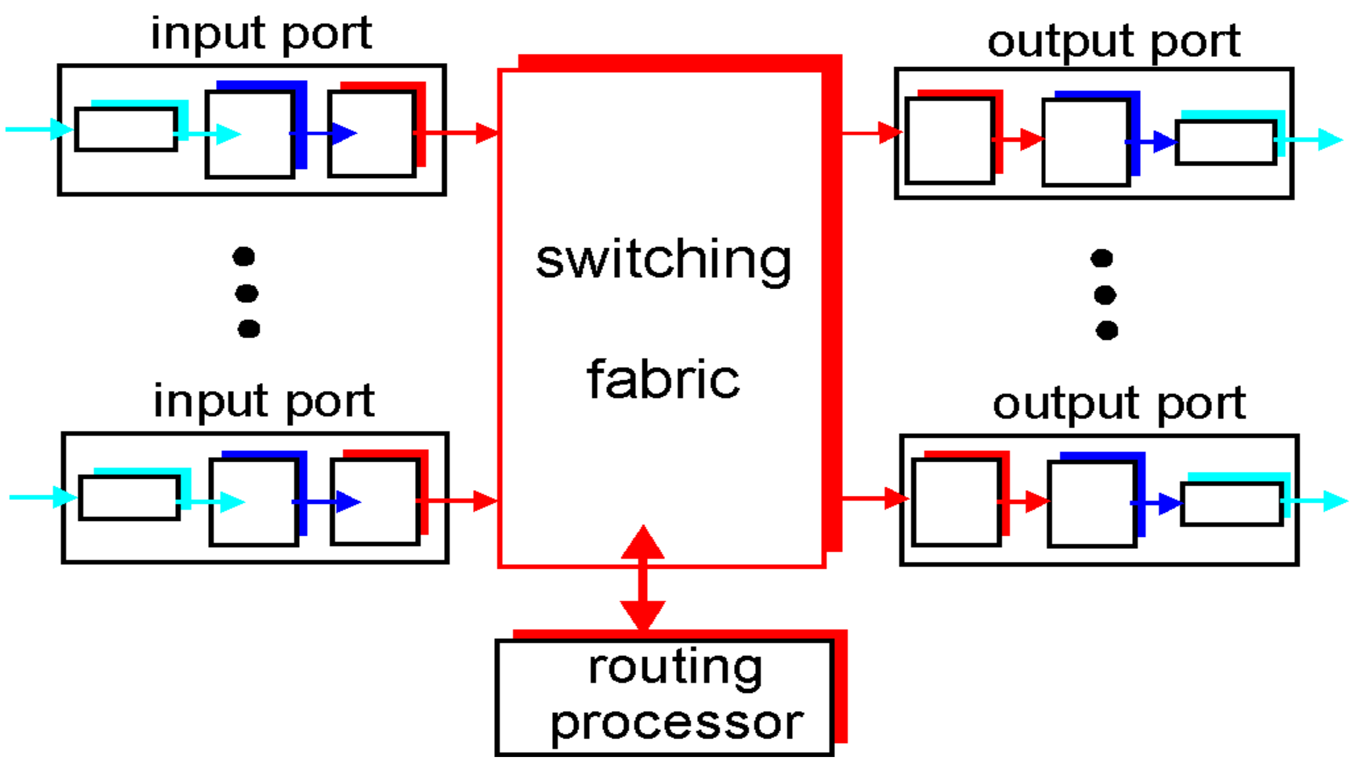
Chapter 4 roadmap

4.1 Introduction and Network Service Models
4.2 Routing Principles
4.3 Hierarchical Routing
4.4 The Internet (IP) Protocol
4.5 Routing in the Internet
4.6 What's Inside a Router?
4.7 IPv6
4.8 Multicast Routing
4.9 Mobility

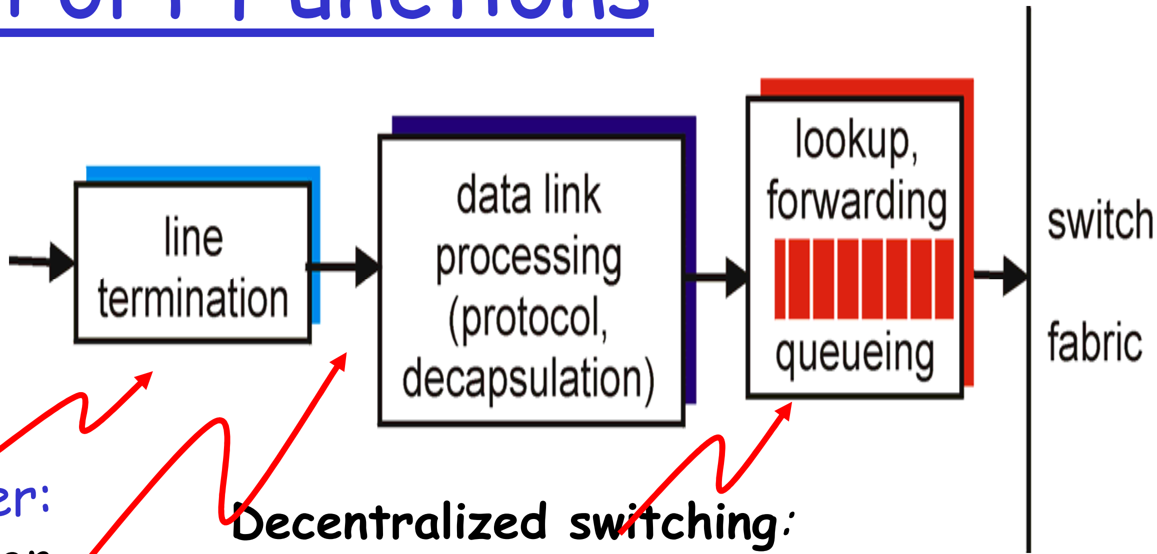
Router Architecture Overview

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

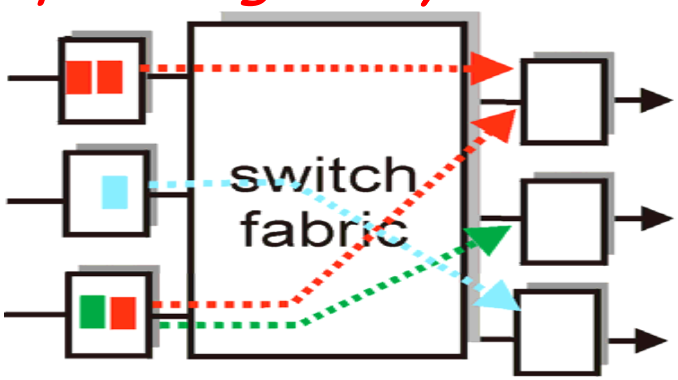
Data link layer:
e.g., Ethernet
see chapter 5

Decentralized switching:

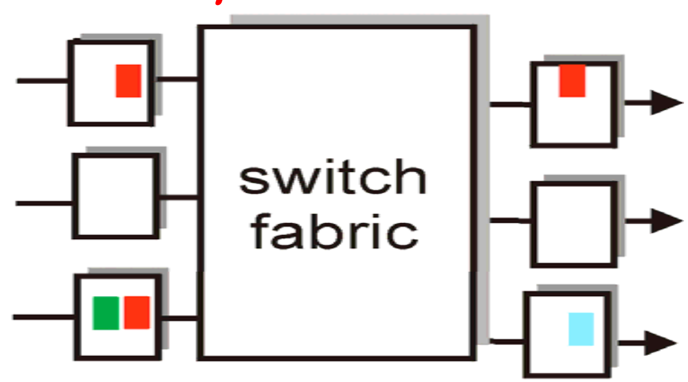
- given datagram dest., lookup output port using routing table in input port memory
  - Local copy received from routing processor
  - Can also be done centrally (thru routing processor).
- goal: complete input port processing at 'line speed'
  - Longest prefix matching etc. Need appropriate data structures.
- queuing: if datagrams arrive faster than forwarding rate into switch fabric

Input Port Queuing

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

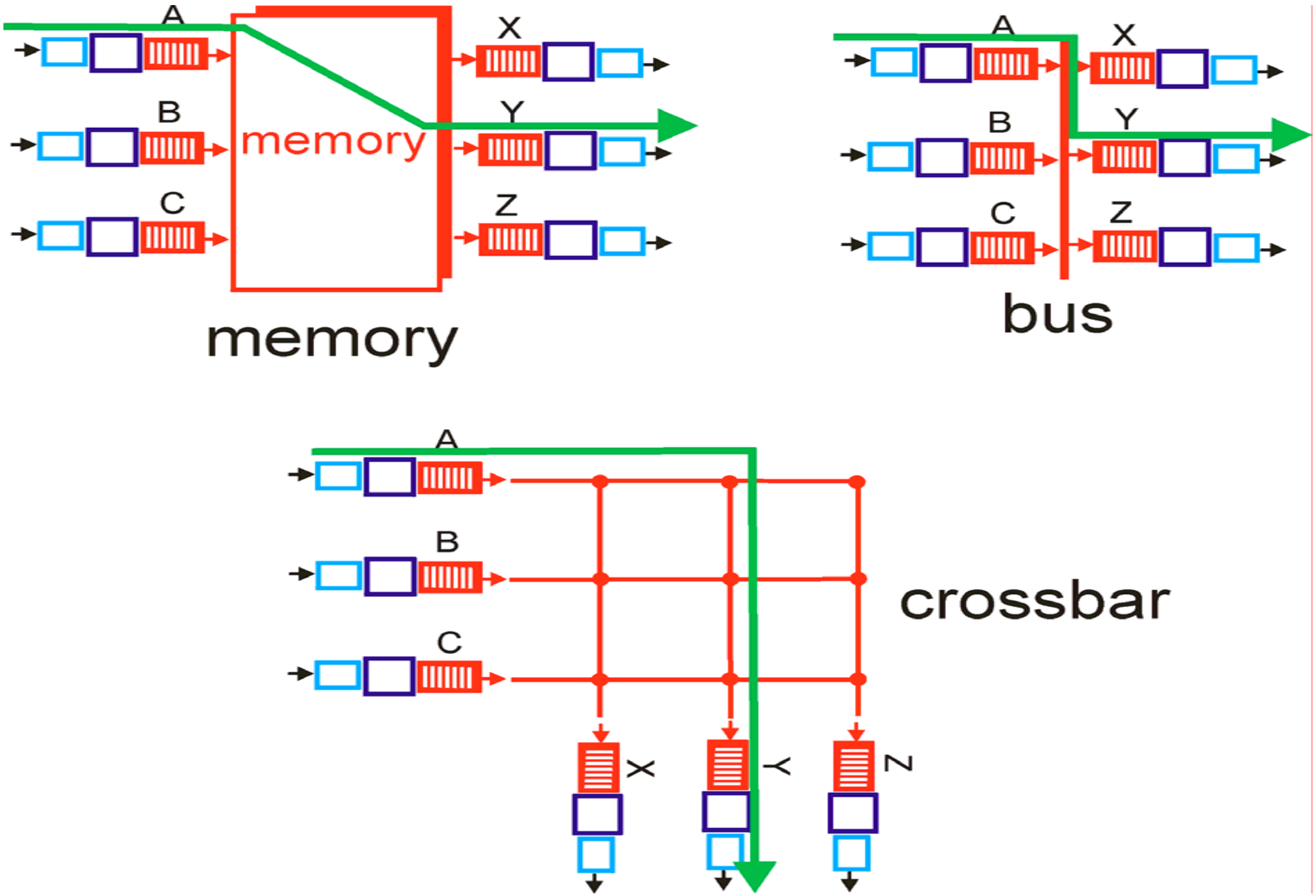


output port contention at time t - only one red packet can be transferred



green packet experiences HOL blocking

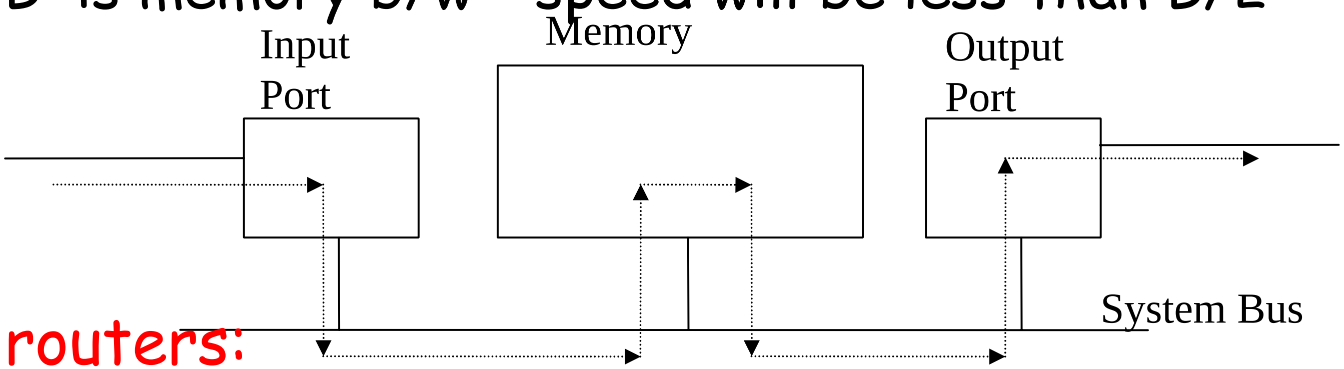
Three types of switching fabrics



Switching Via Memory

First generation routers:

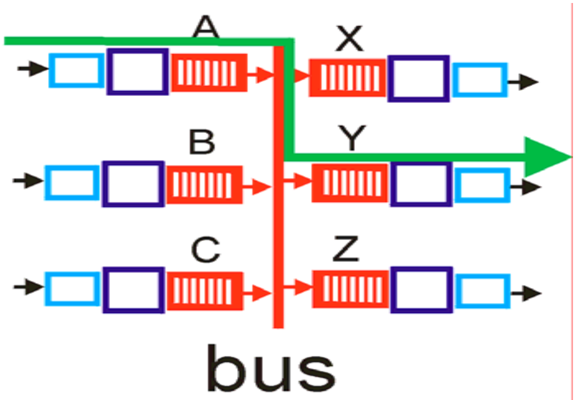
- packet copied by system's (single) CPU
  - Input port sends interrupt to CPU, packet copied to CPU, look up by CPU, copied to output port buffer.
- speed limited by memory bandwidth (2 bus crossings per datagram)
  - If B is memory b/w - speed will be less than B/2



Modern routers:

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

Switching Via a Bus

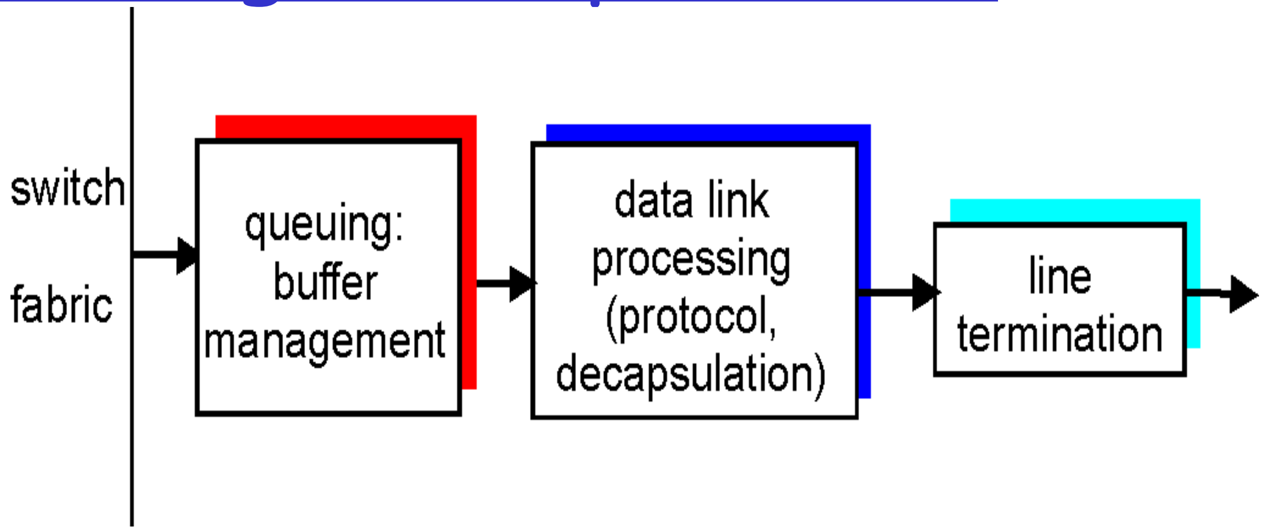


- datagram from input port memory to output port memory via a shared bus
- bus contention: switching speed limited by bus bandwidth
- 1 Gbps bus, Cisco 1900: sufficient speed for access and enterprise routers (not regional or backbone)

Switching Via An Interconnection Network

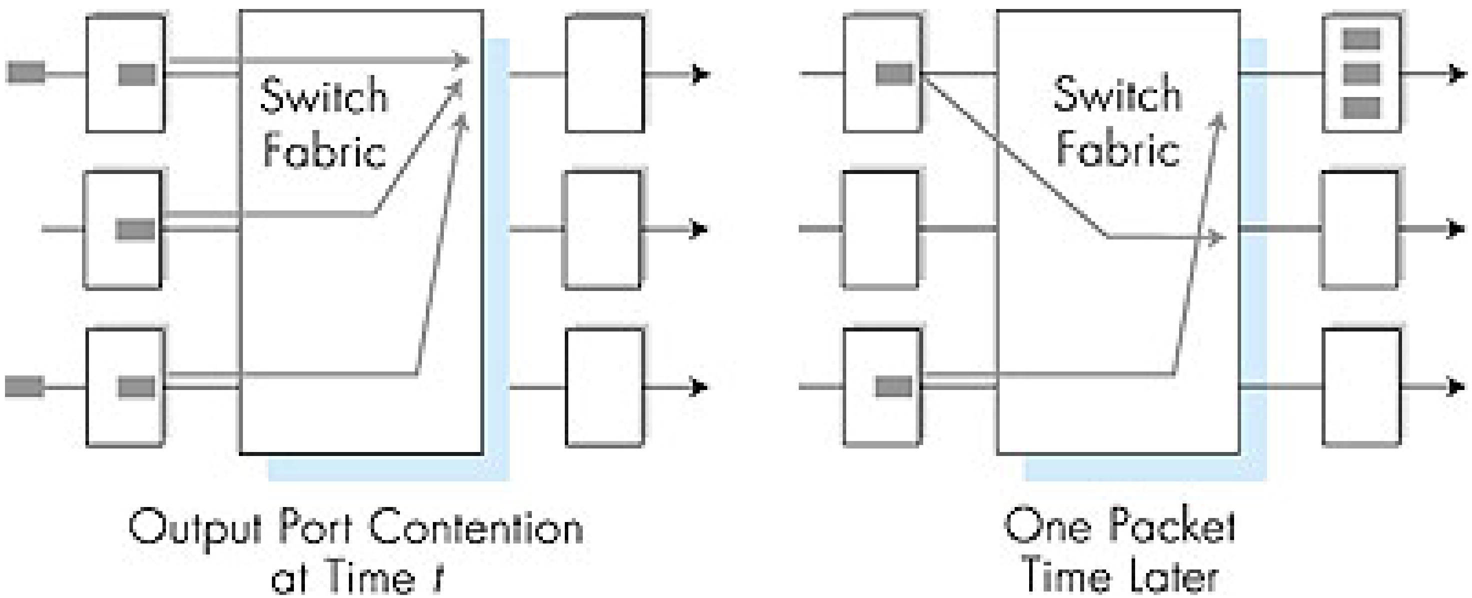
- overcome bus bandwidth limitations
- Banyan networks, other interconnection nets initially developed to connect processors in multiprocessor
- Advanced design: fragmenting datagram into fixed length cells, switch cells through the fabric.
- Cisco 12000: switches Gbps through the interconnection network

Queueing & Output Ports



- *Input port queueing: if n input ports, switching fabric must be n times faster than input speed, for no queueing*
- *Buffering* required when datagrams arrive from fabric faster than the transmission rate
  - Can happen if packets from multiple input ports are destined to the same output port.
- *Scheduling discipline* chooses among queued datagrams for transmission

Output port queueing



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

Output Port Queueing

- Need a packet scheduler at the output port
  - This is where mechanisms for QoS (quality of service) gurantees will have to be implemented
- Simplest one: FIFO
  - "Drop-tail" behavior (drop packets at the end of the buffer, when it starts overflowing)
- Active Queue Management (AQM) – do something smarter
  - e.g. RED: drop packets if average queue size is above threshold, accept if below another threshold, and drop with some probability, if in between the two thresholds.

Chapter 4 roadmap

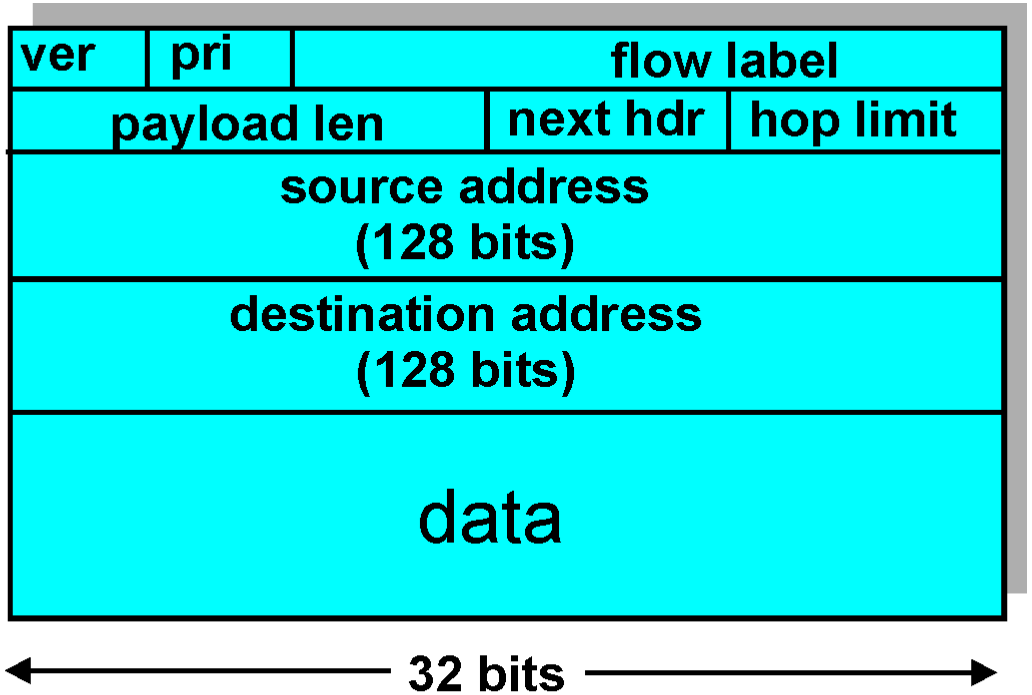
4.1 Introduction and Network Service Models
4.2 Routing Principles
4.3 Hierarchical Routing
4.4 The Internet (IP) Protocol
4.5 Routing in the Internet
4.6 What's Inside a Router?
4.7 IPv6
4.8 Multicast Routing
4.9 Mobility

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



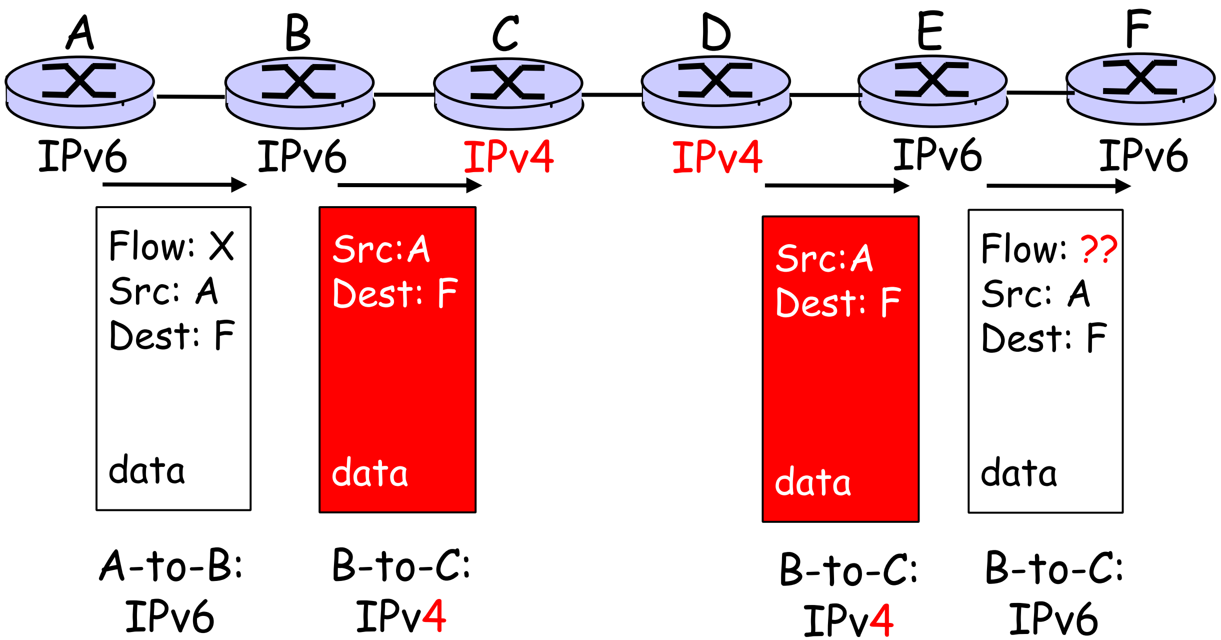
Other Changes from IPv4

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

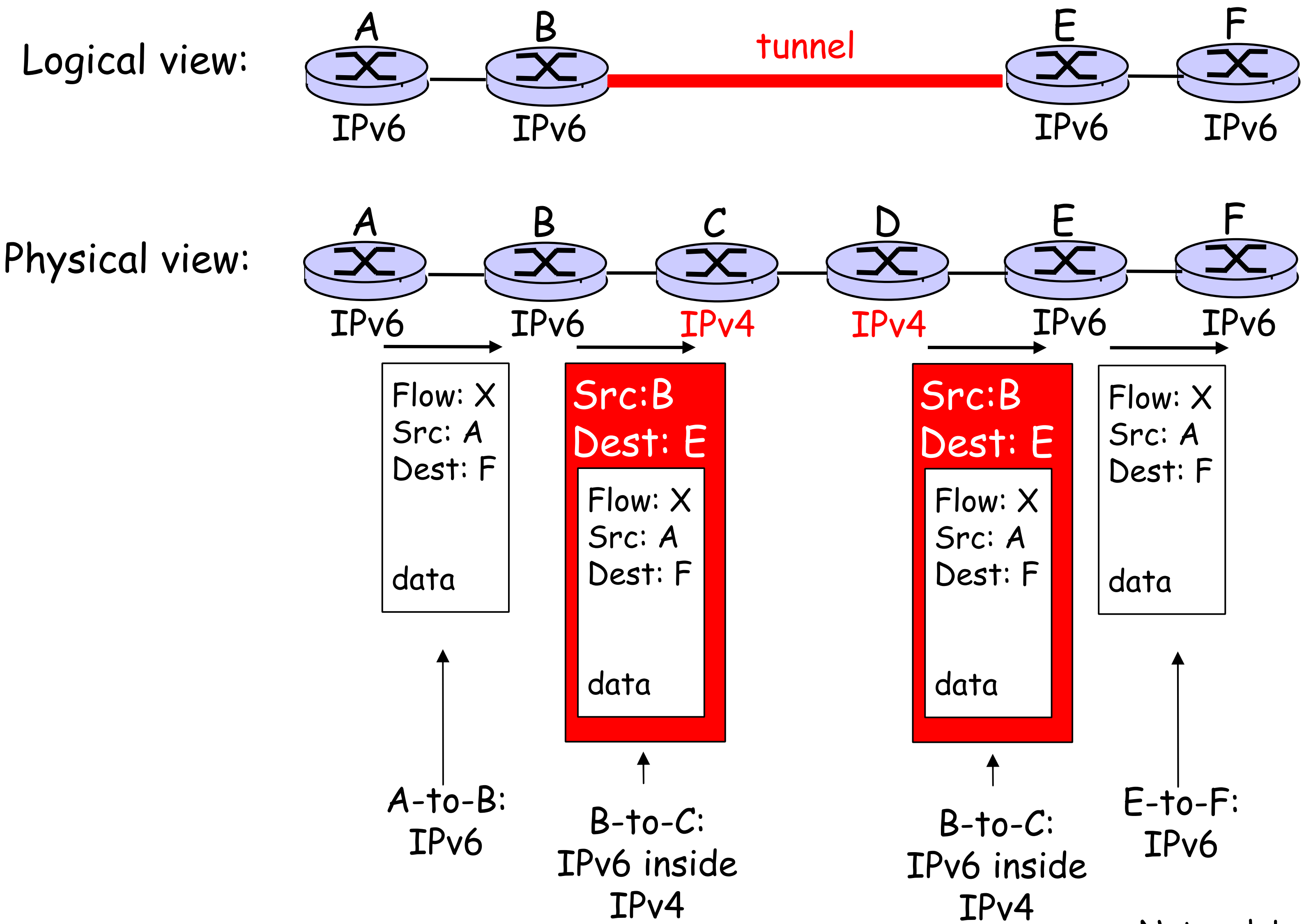
Transition From IPv4 To IPv6

- Not all routers can be upgraded simultaneous
  - no "flag days"
  - How will the network operate with 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 in IPv4 datagram among IPv4 routers

Dual Stack Approach



Tunneling

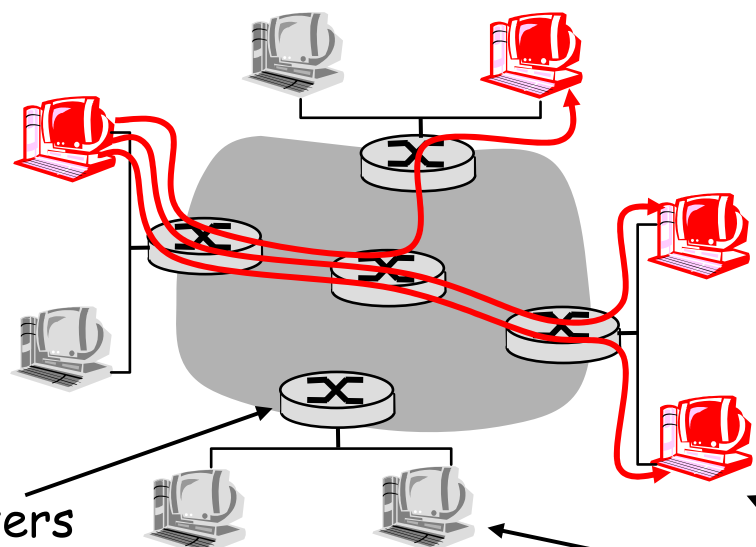

Logical view:
A
B
tunnel
E
F
IPv6
IPv6
IPv6
IPv6
Physical view:
A
B
C
D
E
F
IPv6
IPv6
IPv4
IPv4
IPv6
IPv6
Flow: X
Src: A
Dest: F
data
Src:B
Dest: E
Flow: X
Src: A
Dest: F
data
Src:B
Dest: E
Flow: X
Src: A
Dest: F
data
Flow: X
Src: A
Dest: F
data
A-to-B:
IPv6
B-to-C:
IPv6 inside
IPv4
B-to-C:
IPv6 inside
IPv4
E-to-F:
IPv6

Chapter 4 roadmap

4.1 Introduction and Network Service Models
4.2 Routing Principles
4.3 Hierarchical Routing
4.4 The Internet (IP) Protocol
4.5 Routing in the Internet
4.6 What's Inside a Router?
4.7 IPv6
4.8 Multicast Routing
4.9 Mobility

Multicast: one sender to many receivers

- Multicast: act of sending datagram to multiple receivers with single "transmit" operation
  - analogy: one teacher to many students
- Question: how to achieve multicast



Multicast via unicast

- source sends N unicast datagrams, one addressed to each of N receivers

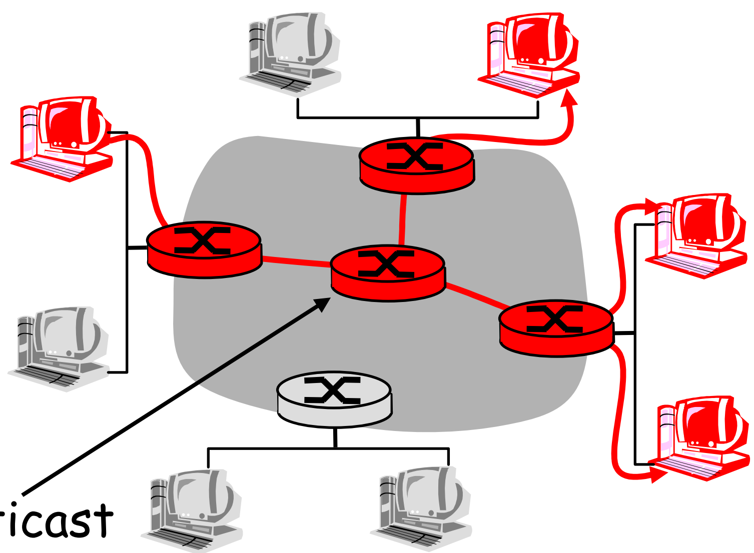
routers forward unicast datagrams

multicast receiver (red)

not a multicast receiver (red)

Multicast: one sender to many receivers

- Multicast: act of sending datagram to multiple receivers with single "transmit" operation
  - analogy: one teacher to many students
- Question: how to achieve multicast



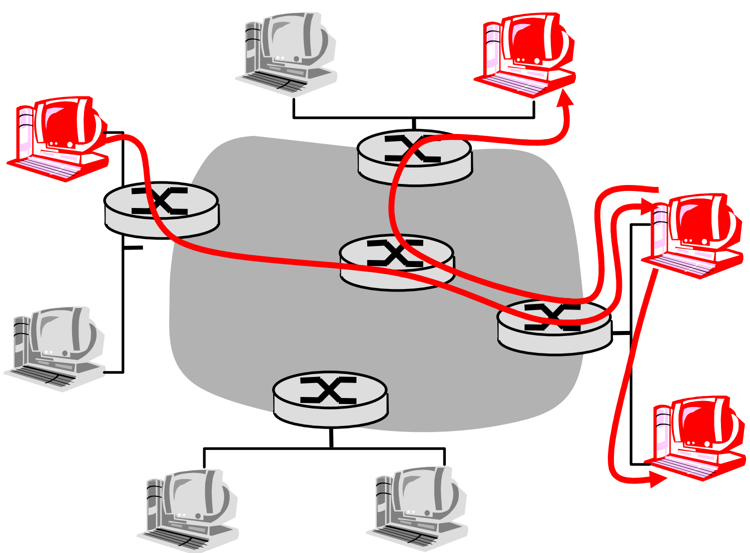
Multicast routers (red) duplicate and forward multicast datagrams

Network multicast

- Router actively participate in multicast, making copies of packets as needed and forwarding towards multicast receivers

Multicast: one sender to many receivers

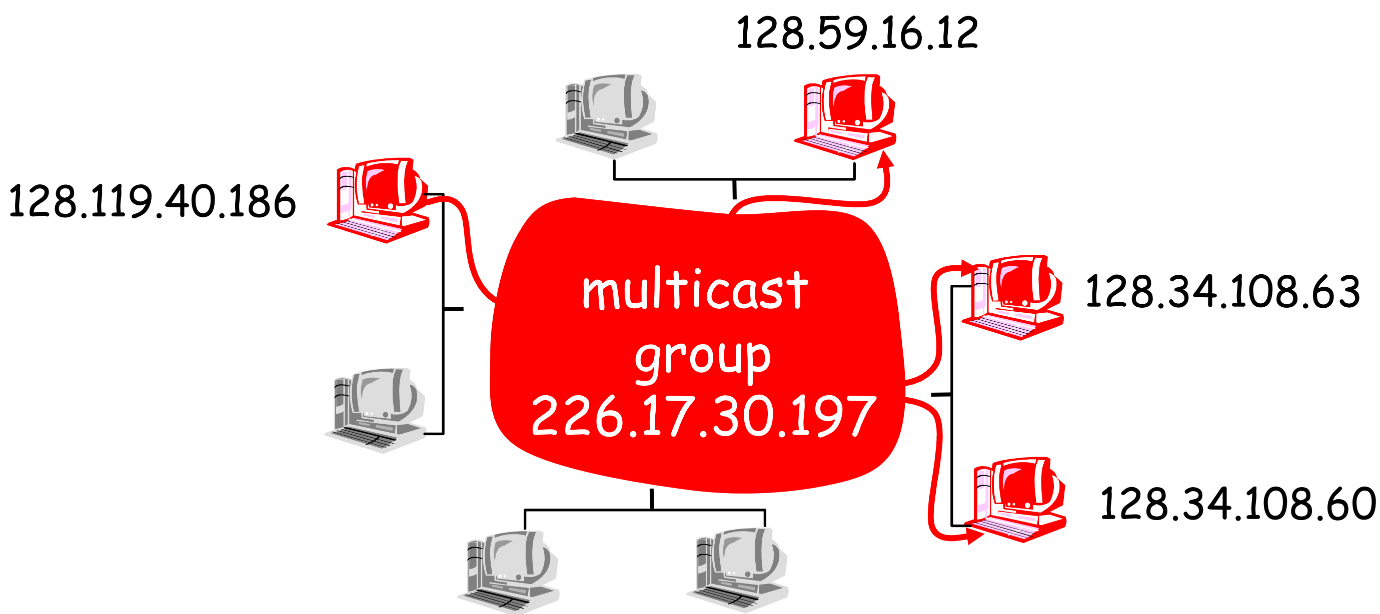
- Multicast: act of sending datagram to multiple receivers with single "transmit" operation
  - analogy: one teacher to many students
- Question: how to achieve multicast



Application-layer multicast

- end systems involved in multicast copy and forward unicast datagrams among themselves

Internet Multicast Service Model

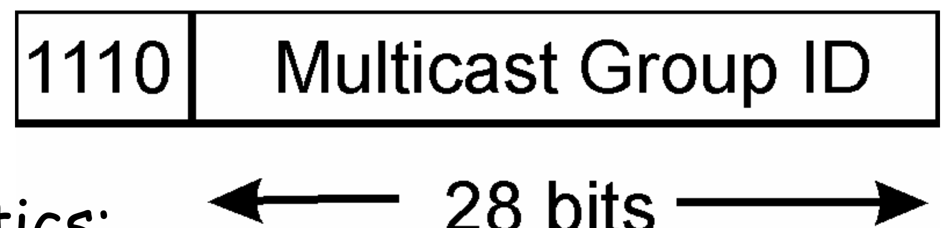


multicast group concept: use of indirection

- hosts addresses IP datagram to multicast group
- routers forward multicast datagrams to hosts that have "joined" that multicast group

Multicast groups

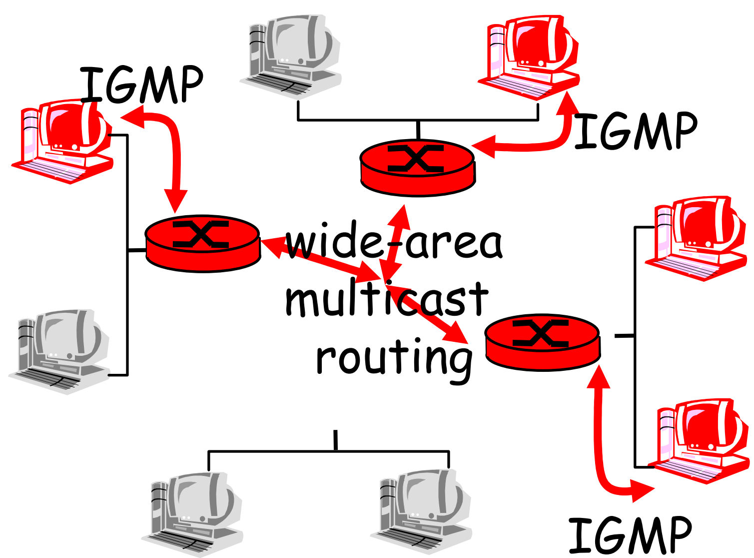
- class D Internet addresses reserved for multicast:



- host group semantics:
  - anyone can “join” (receive) multicast group
  - anyone can send to multicast group
  - no network-layer identification to hosts of members
- *needed:* infrastructure to deliver mcast-addressed datagrams to all hosts that have joined that multicast group

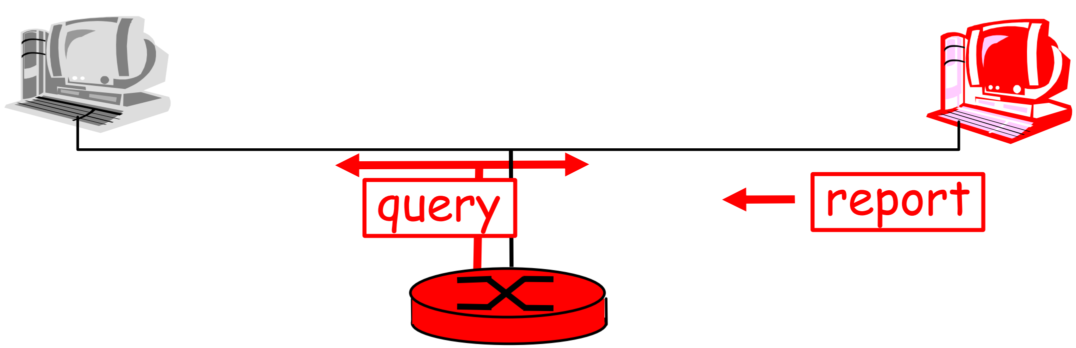
Joining a mcast group: two-step process

- *local:* host informs local mcast router of desire to join group: IGMP (Internet Group Management Protocol)
- *wide area:* local router interacts with other routers to receive mcast datagram flow
  - many protocols (e.g., DVMRP, MOSPF, PIM)



IGMP: Internet Group Management Protocol

- *host:* sends IGMP report when application joins mcast group
  - IP_ADD_MEMBERSHIP socket option
  - host need not explicitly "unjoin" group when leaving
- *router:* sends IGMP query at regular intervals
  - host belonging to a mcast group must reply to query



IGMP

IGMP version 1

- *router:* Host Membership Query msg broadcast on LAN to all hosts
- *host:* Host Membership Report msg to indicate group membership
  - randomized delay before responding
  - implicit leave via no reply to Query
- RFC 1112

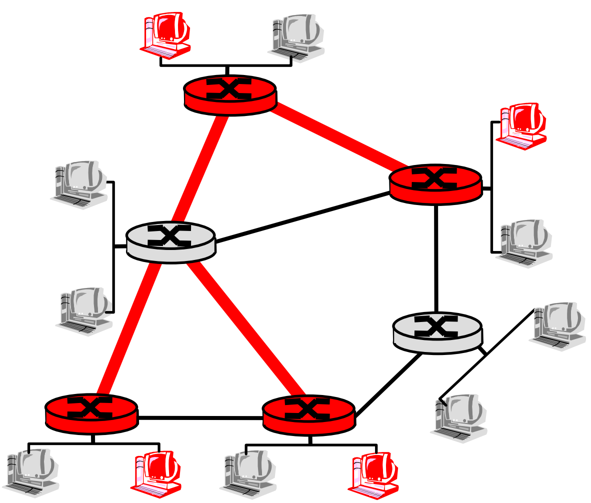
IGMP v2: additions include

- group-specific Query
- Leave Group msg
  - last host replying to Query can send explicit Leave Group msg
  - router performs group-specific query to see if any hosts left in group
  - RFC 2236

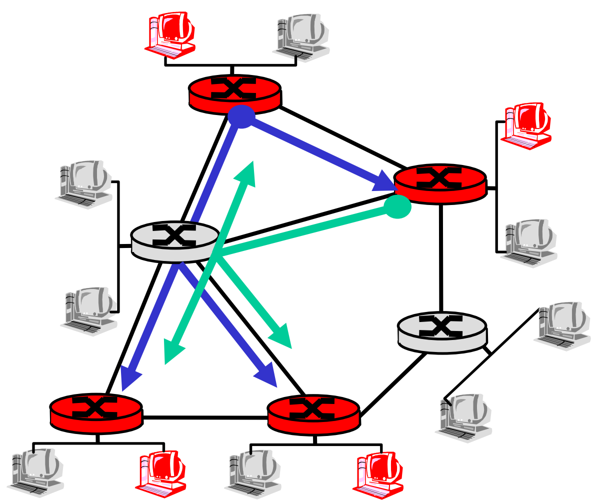
IGMP v3: under development as Internet draft

Multicast Routing: Problem Statement

- ***Goal:*** find a tree (or trees) connecting routers having local mcast group members
  - *tree:* not all paths between routers used
  - *source-based:* different tree from each sender to rcvrs
  - *shared-tree:* same tree used by all group members



Shared tree



Source-based trees

Approaches for building mcast trees

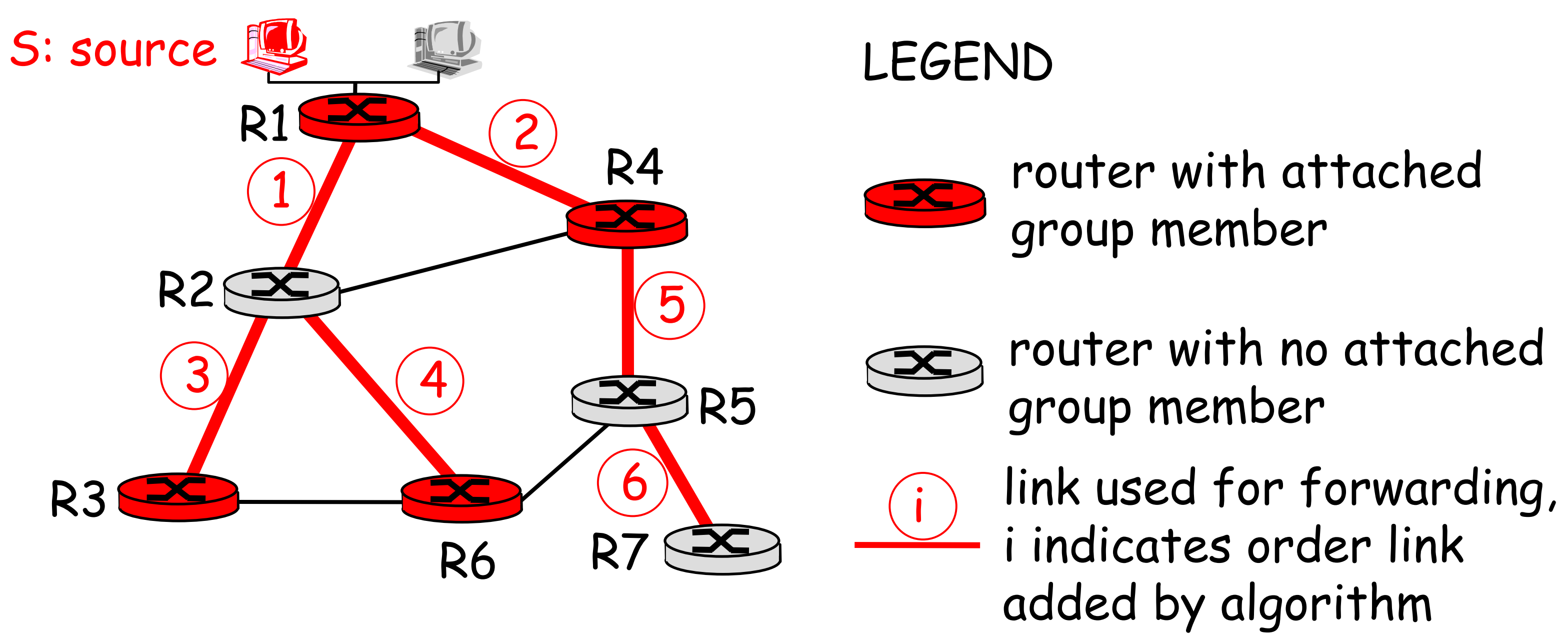
Approaches:

- source-based tree: one tree per source
  - shortest path trees
  - reverse path forwarding
- group-shared tree: group uses one tree
  - minimal spanning (Steiner)
  - center-based trees

…we first look at basic approaches, then specific protocols adopting these approaches

Shortest Path Tree

- mcast forwarding tree: tree of shortest path routes from source to all receivers
  - Dijkstra's algorithm



Reverse Path Forwarding

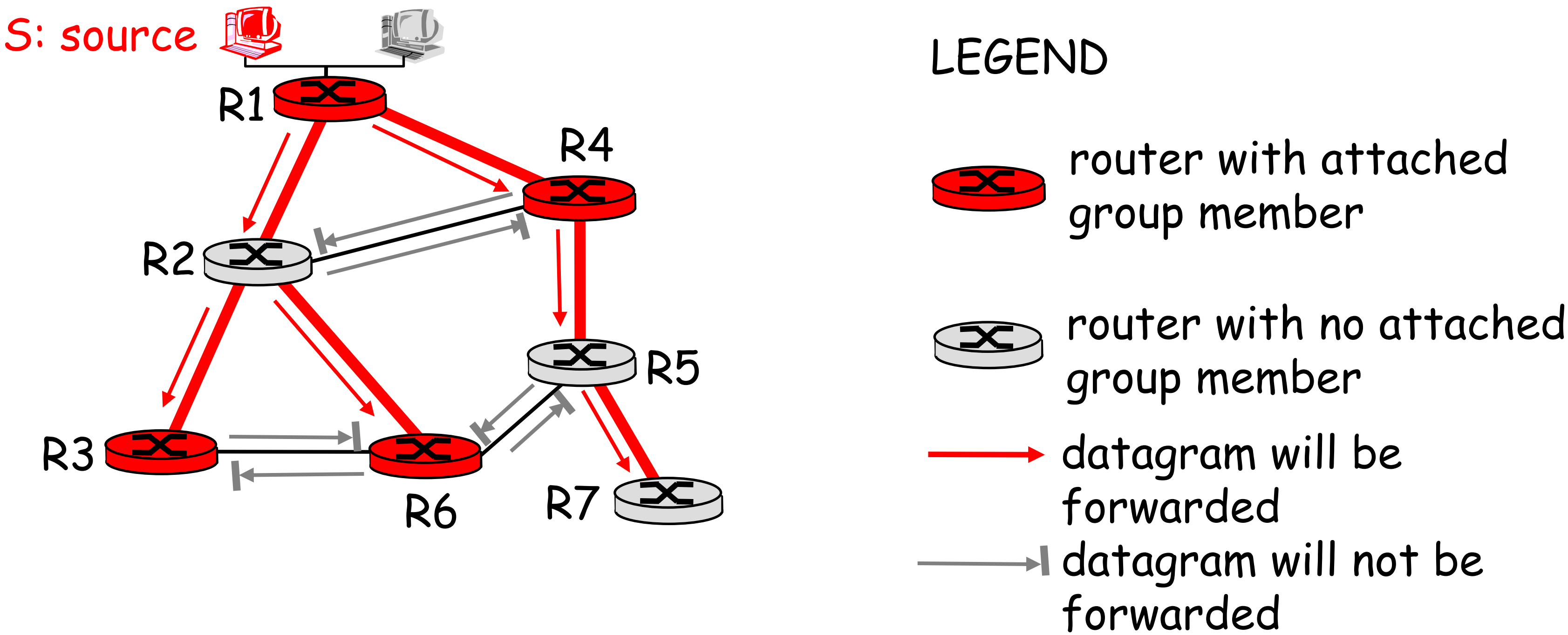
- rely on router's knowledge of unicast shortest path from it to sender
- each router has simple forwarding behavior:

if (mcast datagram received on incoming link on shortest path back to center)

then flood datagram onto all outgoing links

else ignore datagram

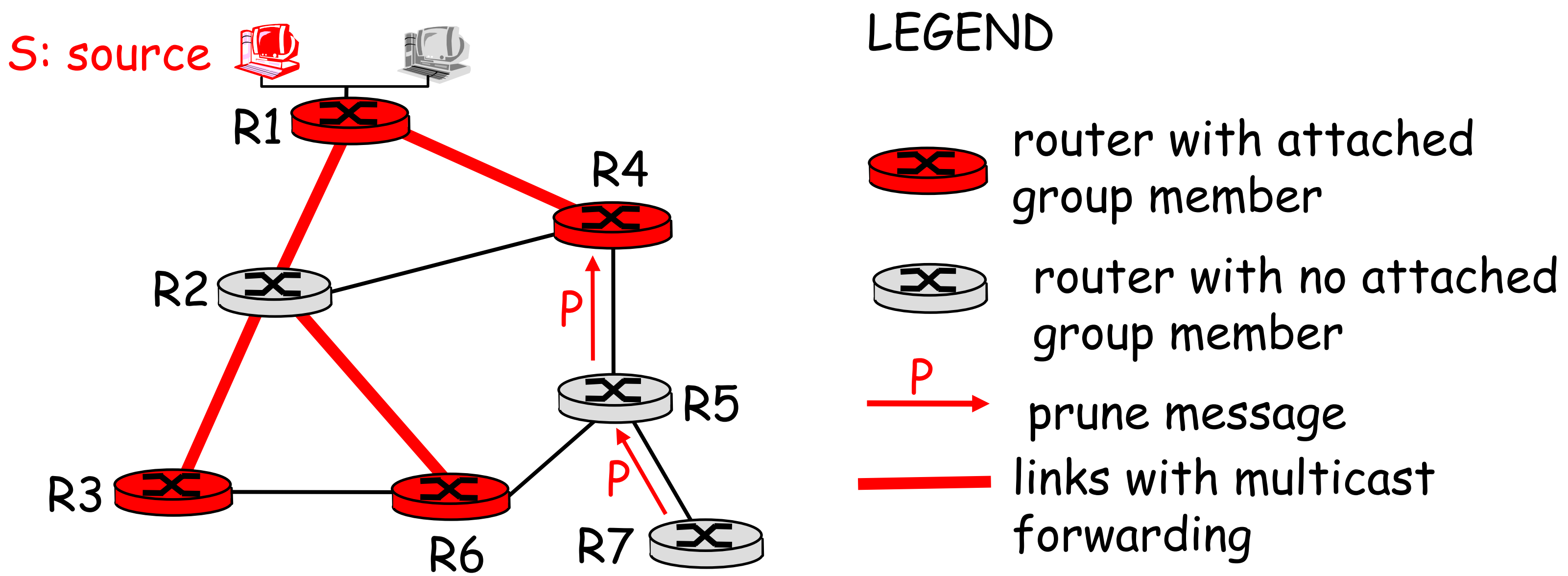
Reverse Path Forwarding: example



- result is a source-specific *reverse* SPT
  - may be a bad choice with asymmetric links

Reverse Path Forwarding: pruning

- forwarding tree contains subtrees with no mcast group members
  - no need to forward datagrams down subtree
  - "prune" msgs sent upstream by router with no downstream group members



Shared-Tree: Steiner Tree

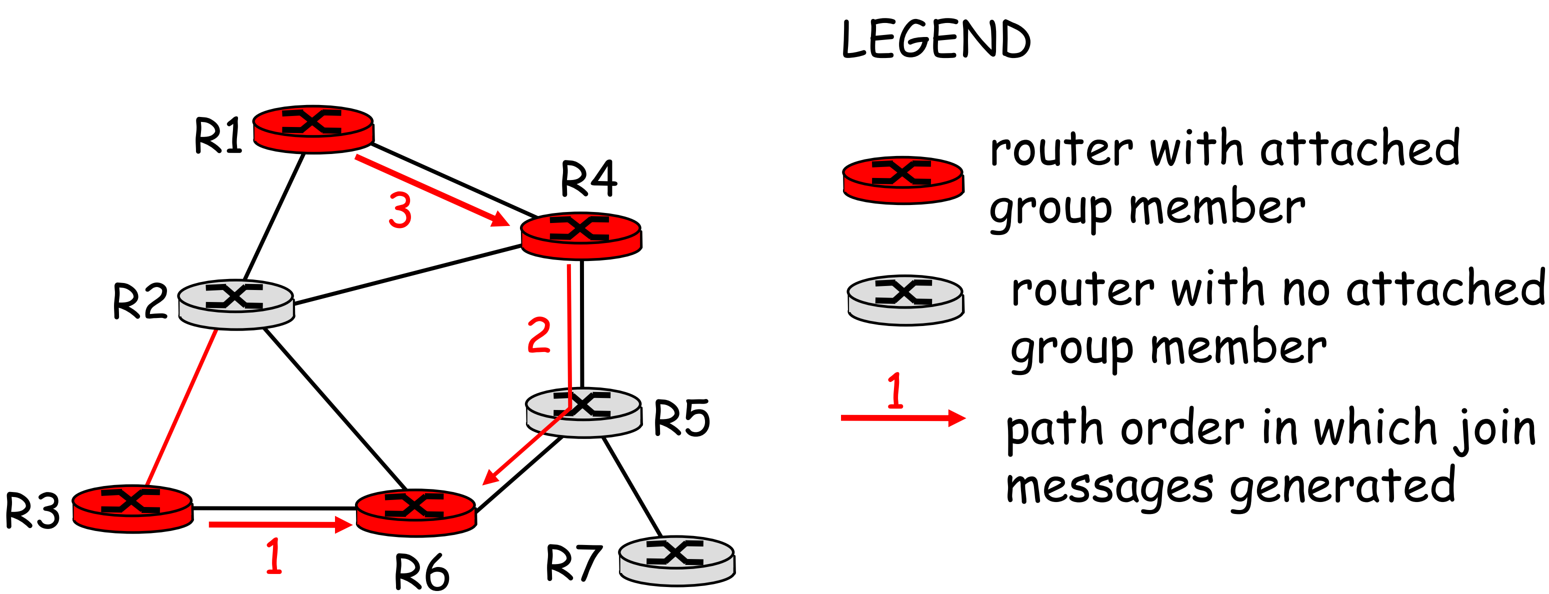
- Steiner Tree: minimum cost tree connecting all routers with attached group members
- problem is NP-complete
- excellent heuristics exists
- not used in practice:
  - computational complexity
  - information about entire network needed
  - monolithic: rerun whenever a router needs to join/leave

Center-based trees

- single delivery tree shared by all
- one router identified as *"center"* of tree
- to join:
  - edge router sends unicast *join-msg* addressed to center router
  - *join-msg* "processed" by intermediate routers and forwarded towards center
  - *join-msg* either hits existing tree branch for this center, or arrives at center
  - path taken by *join-msg* becomes new branch of tree for this router

Center-based trees: an example

Suppose R6 chosen as center:



Internet Multicasting Routing: DVMRP

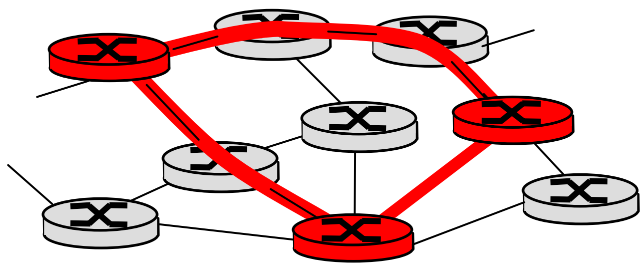
- DVMRP: distance vector multicast routing protocol, RFC1075
- *flood and prune:* reverse path forwarding, source-based tree
  - RPF tree based on DVMRP's own routing tables constructed by communicating DVMRP routers
  - no assumptions about underlying unicast
  - initial datagram to mcast group flooded everywhere via RPF
  - routers not wanting group: send upstream prune msgs

DVMRP: continued…

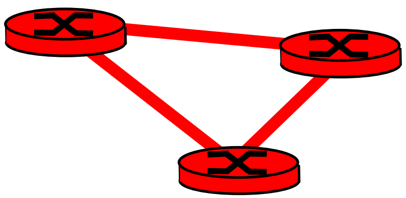
- *soft state:* DVMRP router periodically (1 min.) "forgets" branches are pruned:
  - mcast data again flows down unpruned branch
  - downstream router: reprune or else continue to receive data
- routers can quickly regraft to tree
  - following IGMP join at leaf
- odds and ends
  - commonly implemented in commercial routers
  - Mbone routing done using DVMRP

Tunneling

Q: How to connect "islands" of multicast routers in a "sea" of unicast routers?



physical topology



logical topology

- mcast datagram encapsulated inside "normal" (non-multicast-addressed) datagram
- normal IP datagram sent thru "tunnel" via regular IP unicast to receiving mcast router
- receiving mcast router unencapsulates to get mcast datagram

PIM: Protocol Independent Multicast

- not dependent on any specific underlying unicast routing algorithm (works with all)
- two different multicast distribution scenarios :

Dense:

- group members densely packed, in "close" proximity.
- bandwidth more plentiful

Sparse:

- # networks with group members small wrt # interconnected networks
- group members "widely dispersed"
- bandwidth not plentiful

Consequences of Sparse-Dense Dichotomy:

Dense

- group membership by routers *assumed* until routers explicitly prune
- *data-driven* construction on mcast tree (e.g., RPF)
- bandwidth and non-group-router processing *profligate*

Sparse:

- no membership until routers explicitly join
- *receiver- driven* construction of mcast tree (e.g., center-based)
- bandwidth and non-group-router processing *conservative*

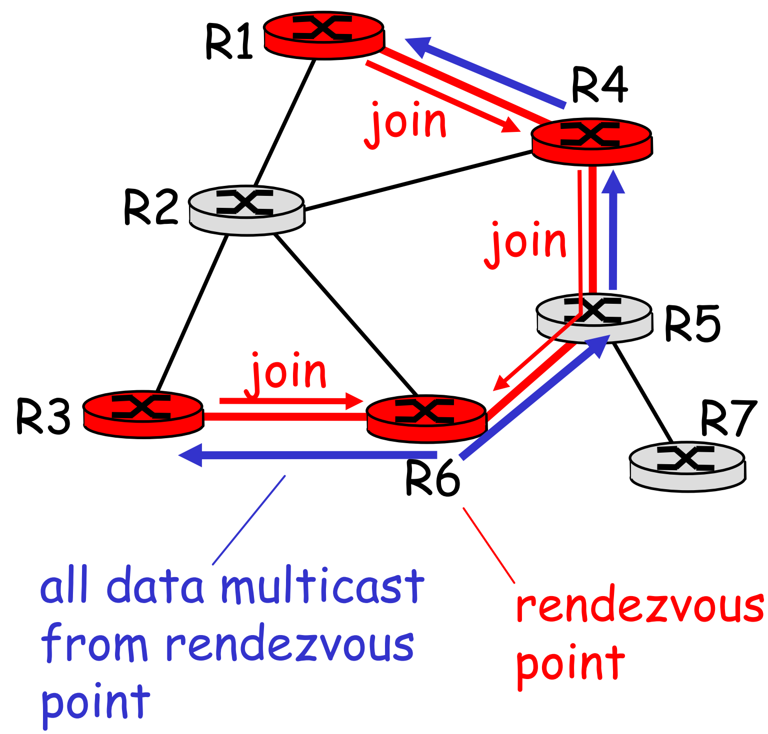
PIM- Dense Mode

flood-and-prune RPF, similar to DVMRP but

- underlying unicast protocol provides RPF info for incoming datagram
- less complicated (less efficient) downstream flood than DVMRP reduces reliance on underlying routing algorithm
- has protocol mechanism for router to detect it is a leaf-node router

PIM - Sparse Mode

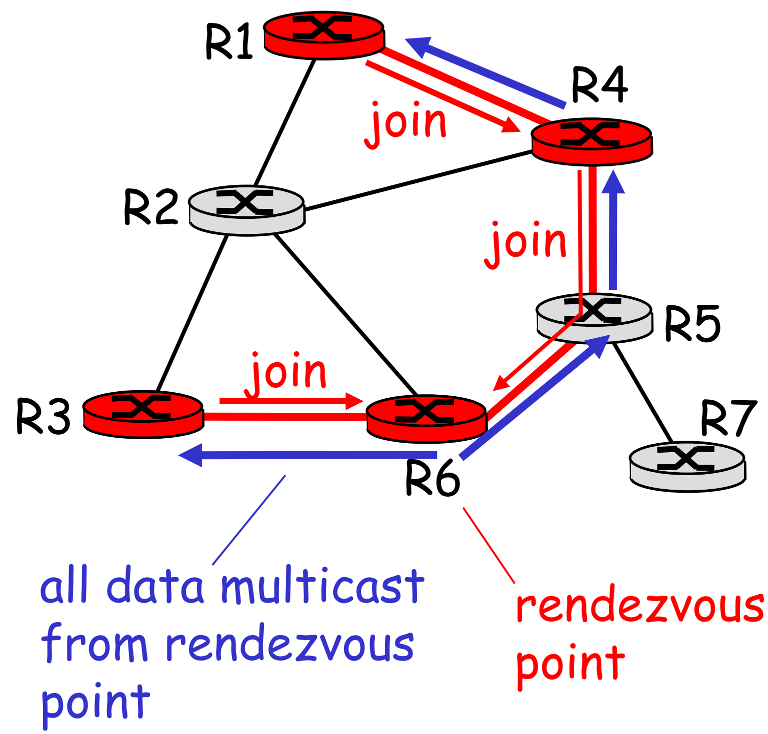
- center-based approach
- router sends *join* msg to rendezvous point (RP)
  - intermediate routers update state and forward *join*
- after joining via RP, router can switch to source-specific tree
  - increased performance: less concentration, shorter paths



PIM - Sparse Mode

sender(s):

- unicast data to RP, which distributes down RP-rooted tree
- RP can extend mcast tree upstream to source
- RP can send *stop* msg if no attached receivers
  - "no one is listening!"

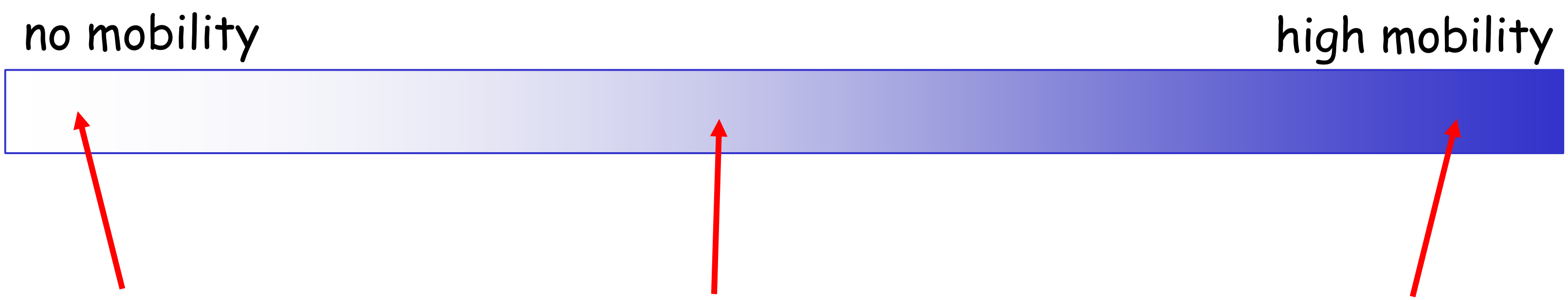


Chapter 4 roadmap

4.1 Introduction and Network Service Models
4.2 Routing Principles
4.3 Hierarchical Routing
4.4 The Internet (IP) Protocol
4.5 Routing in the Internet
4.6 What's Inside a Router?
4.7 IPv6
4.8 Multicast Routing
4.9 Mobility

What is mobility?

- spectrum of mobility, from the *network* perspective:



mobile user, using same access point

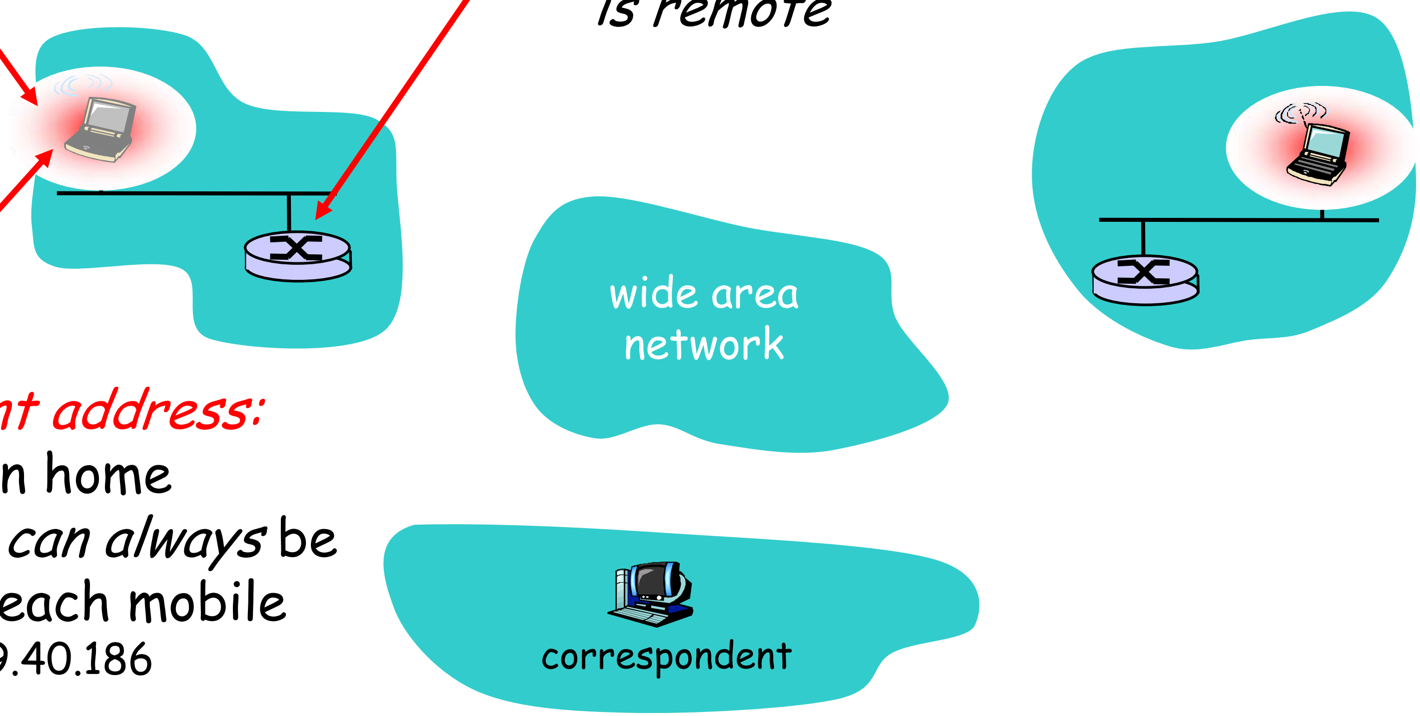
mobile user, connecting/ disconnecting from network using DHCP.

mobile user, passing through multiple access point while maintaining ongoing connections (like cell phone)

Mobility: Vocabulary

home network: permanent "home" of mobile (e.g., 128.119.40/24)

home agent: entity that will perform mobility functions on behalf of mobile, when mobile is remote



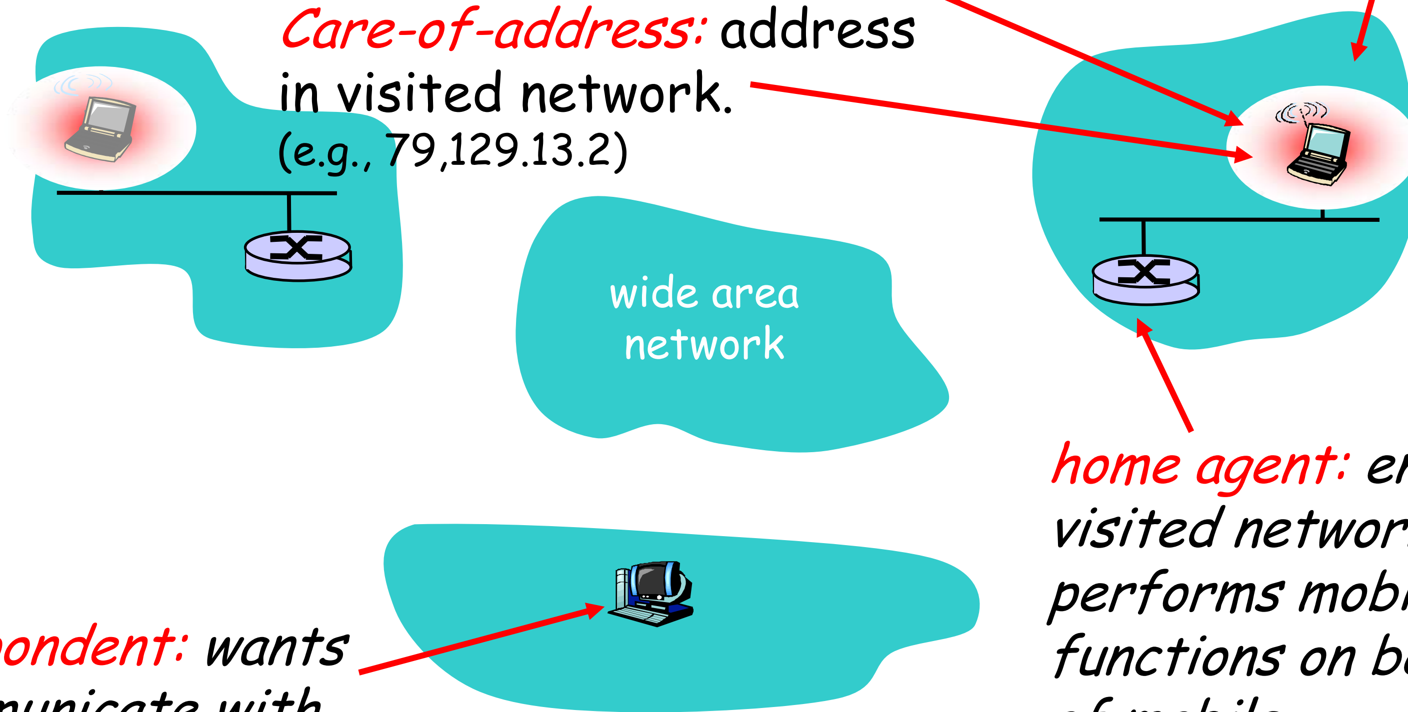
Permanent address: address in home network, *can always* be used to reach mobile e.g., 128.119.40.186

Mobility: more vocabulary

Permanent address: remains constant (e.g., 128.119.40.186)

visited network: network in which mobile currently resides (e.g., 79.129.13/24)

Care-of-address: address in visited network. (e.g., 79,129.13.2)



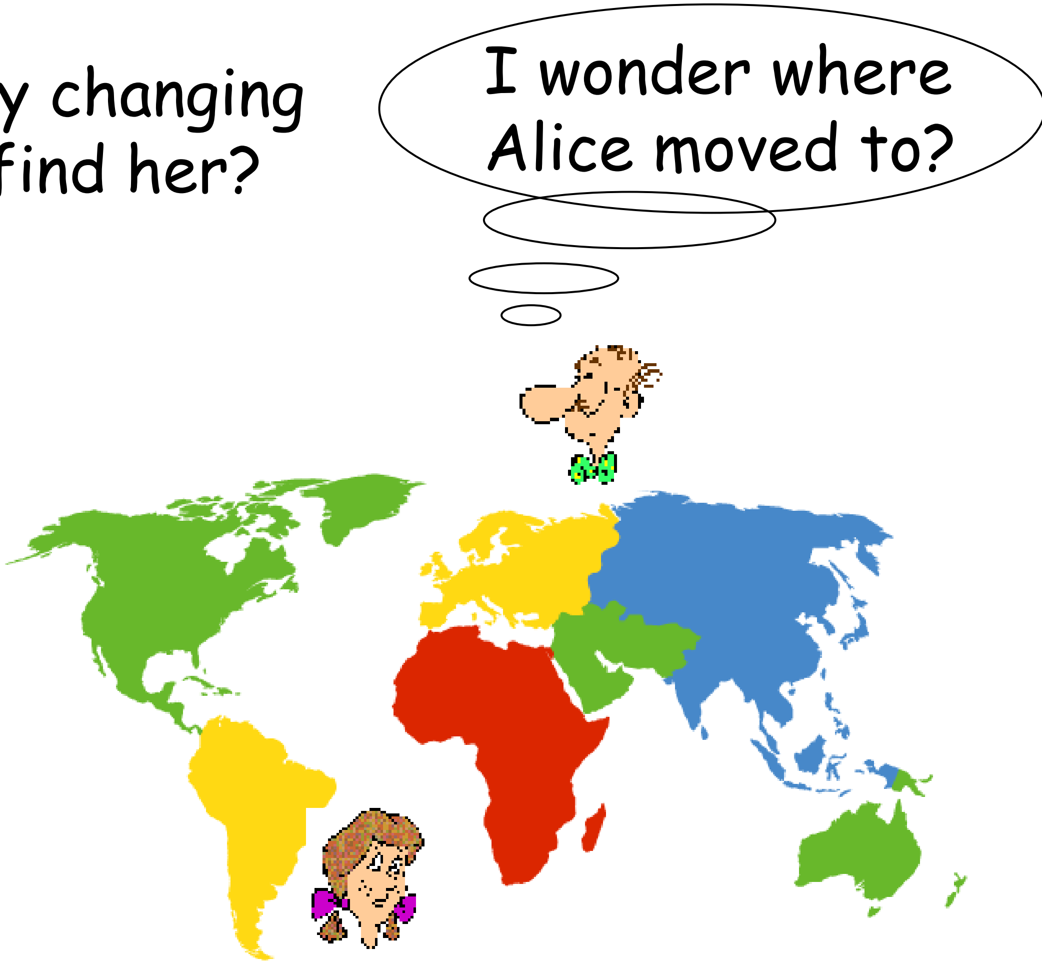
home agent: entity in visited network that performs mobility functions on behalf of mobile.

correspondent: wants to communicate with mobile

How do *you* contact a mobile friend:

Consider friend frequently changing addresses, how do you find her?

- search all phone books?
- call her parents?
- expect her to let you know where he/she is?



Mobility: approaches

- *Let routing handle it:* routers advertise permanent address of mobile-nodes-in-residence via usual routing table exchange.
  - routing tables indicate where each mobile located
  - no changes to end-systems
- *Let end-systems handle it:*
  - *indirect routing:* communication from correspondent to mobile goes through home agent, then forwarded to remote
  - *direct routing:* correspondent gets foreign address of mobile, sends directly to mobile

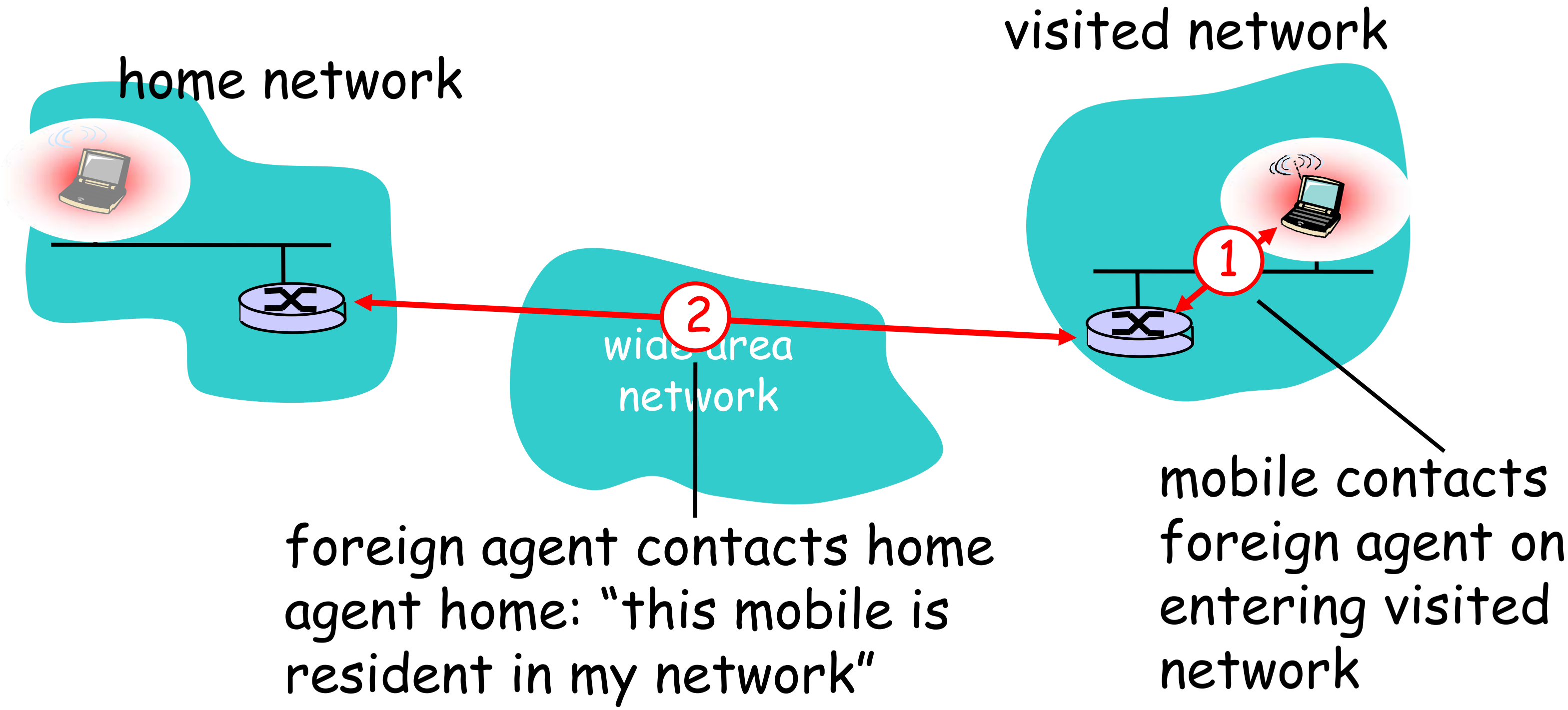
Mobility: approaches

- *Let routing handle it:* ...ters advertise permanent address of mobi... residence via usual routing table ex...
  - routing table... ...ere each mobile located
  - no changes to en... ...ems

not
scalable
to millions of
mobiles

- *let end-systems handle it:*
  - *indirect routing:* communication from correspondent to mobile goes through home agent, then forwarded to remote
  - *direct routing:* correspondent gets foreign address of mobile, sends directly to mobile

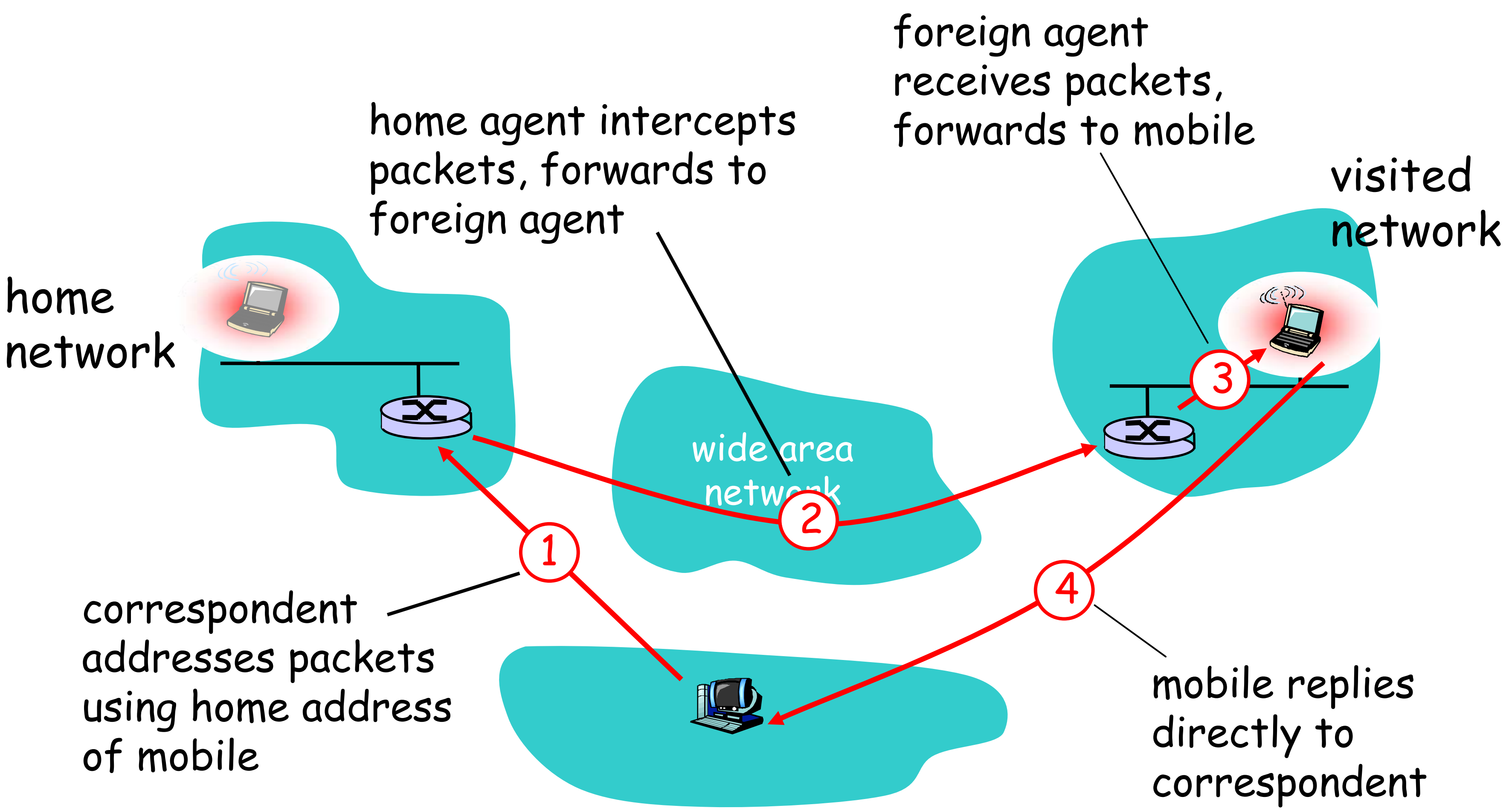
Mobility: registration



End result:

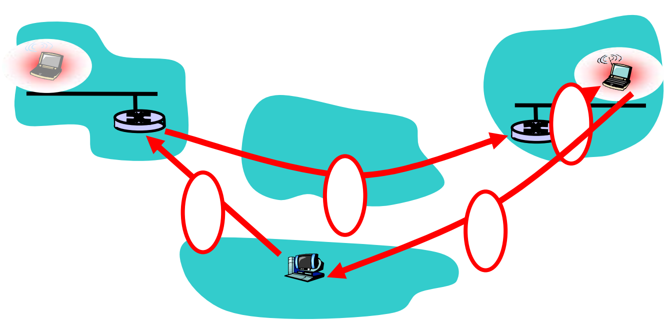
- Foreign agent knows about mobile
- Home agent knows location of mobile

Mobility via Indirect Routing

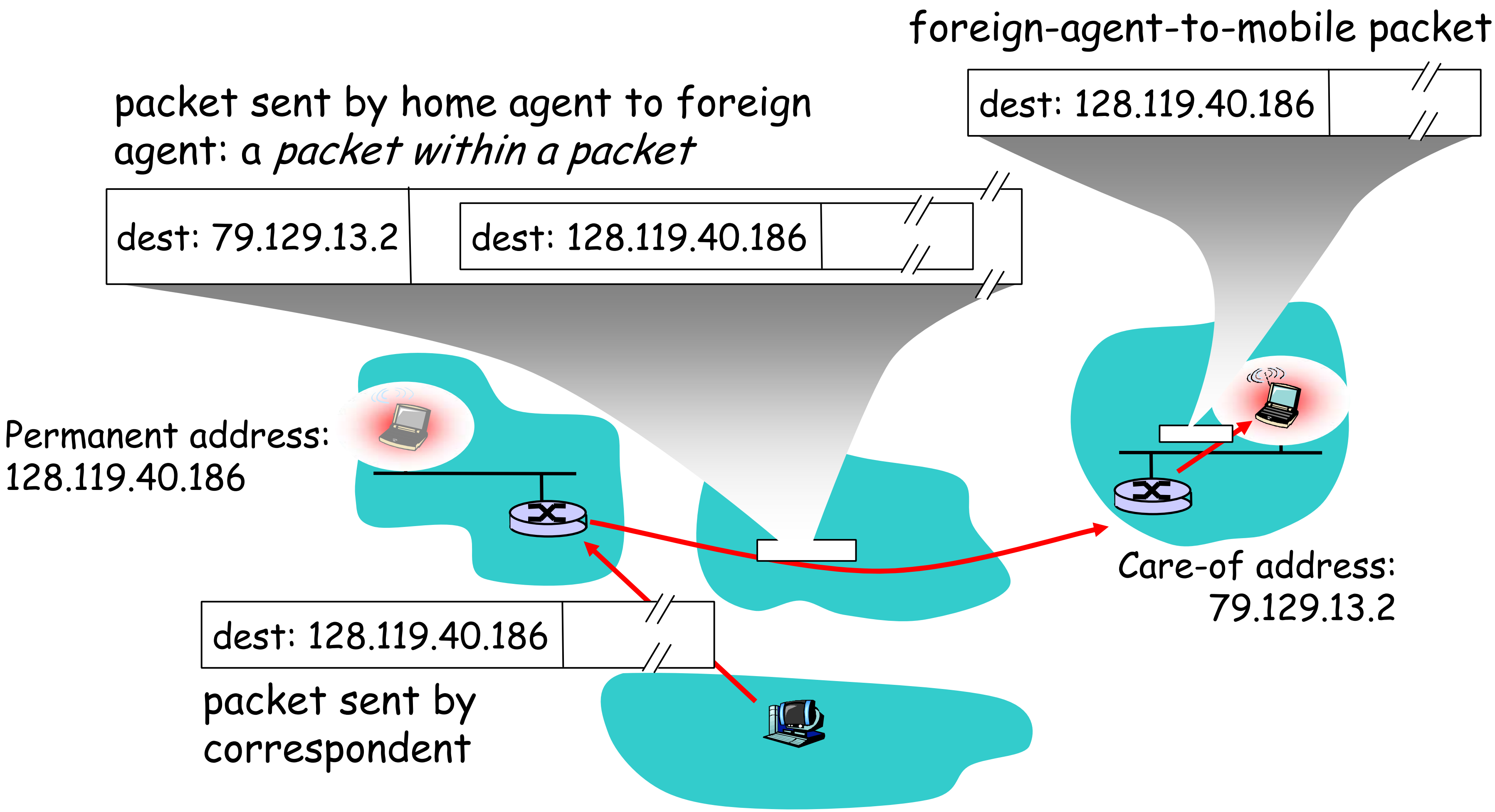


Indirect Routing: comments

- Mobile uses two addresses:
  - permanent address: used by correspondent (hence mobile location is *transparent* to correspondent)
  - care-of-address: used by home agent to forward datagrams to mobile
- foreign agent functions may be done by mobile itself
- triangle routing: correspondent-home-network-mobile
  - inefficient when correspondent, mobile are in same network



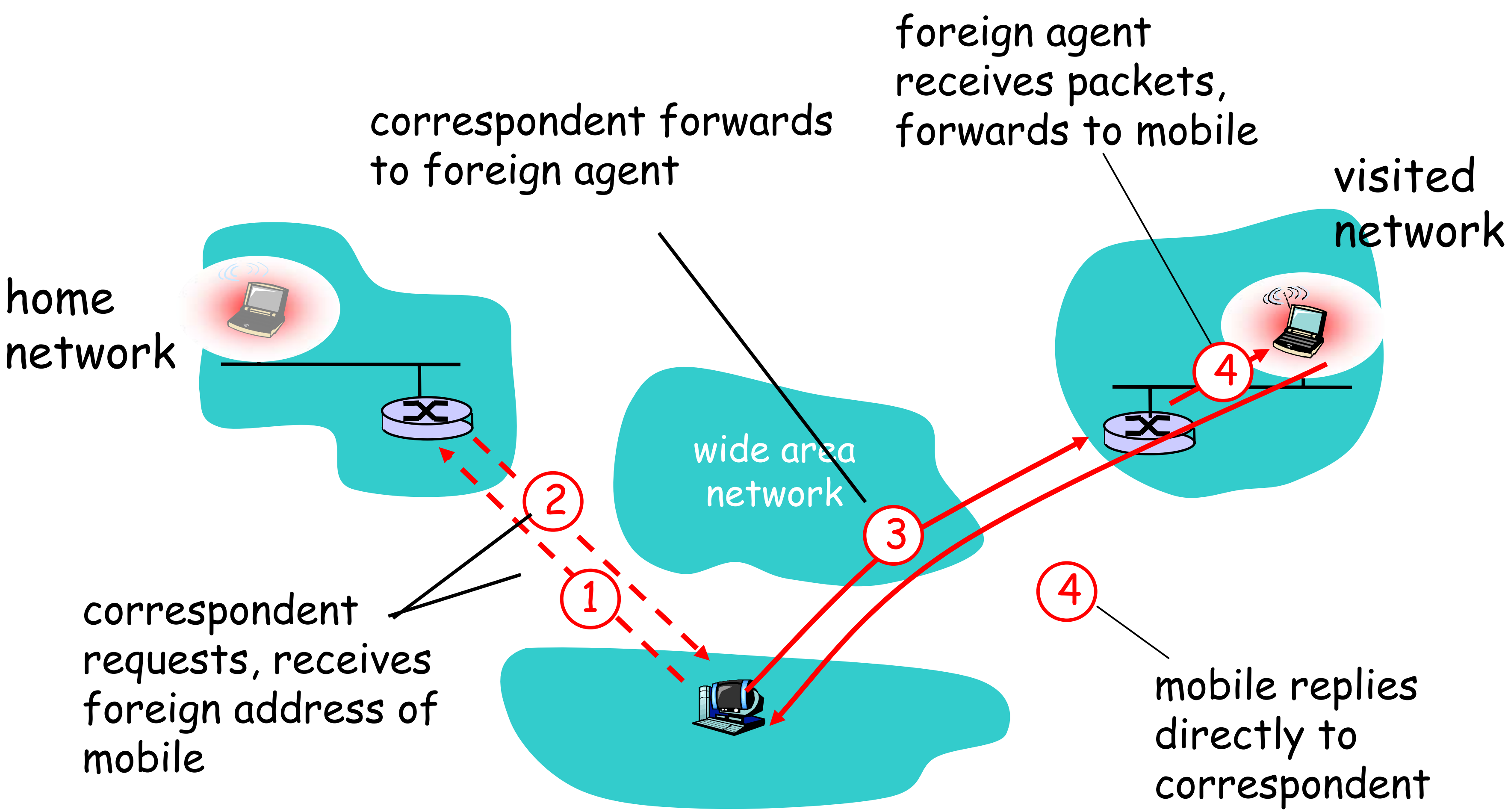
Forwarding datagrams to remote mobile



Indirect Routing: moving between networks

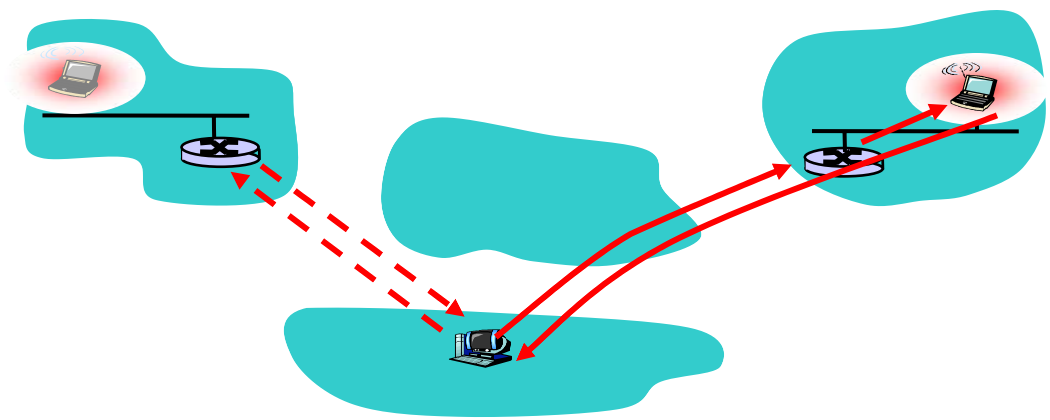
- suppose mobile user moves to another network
  - registers with new foreign agent
  - new foreign agent registers with home agent
  - home agent update care-of-address for mobile
  - packets continue to be forwarded to mobile (but with new care-of-address)
- Mobility, changing foreign networks transparent: *on going connections can be maintained!*

Mobility via Direct Routing



Mobility via Direct Routing: comments

- overcome triangle routing problem
- non-transparent to correspondent: correspondent must get care-of-address from home agent
  - What happens if mobile changes networks?



Mobile IP

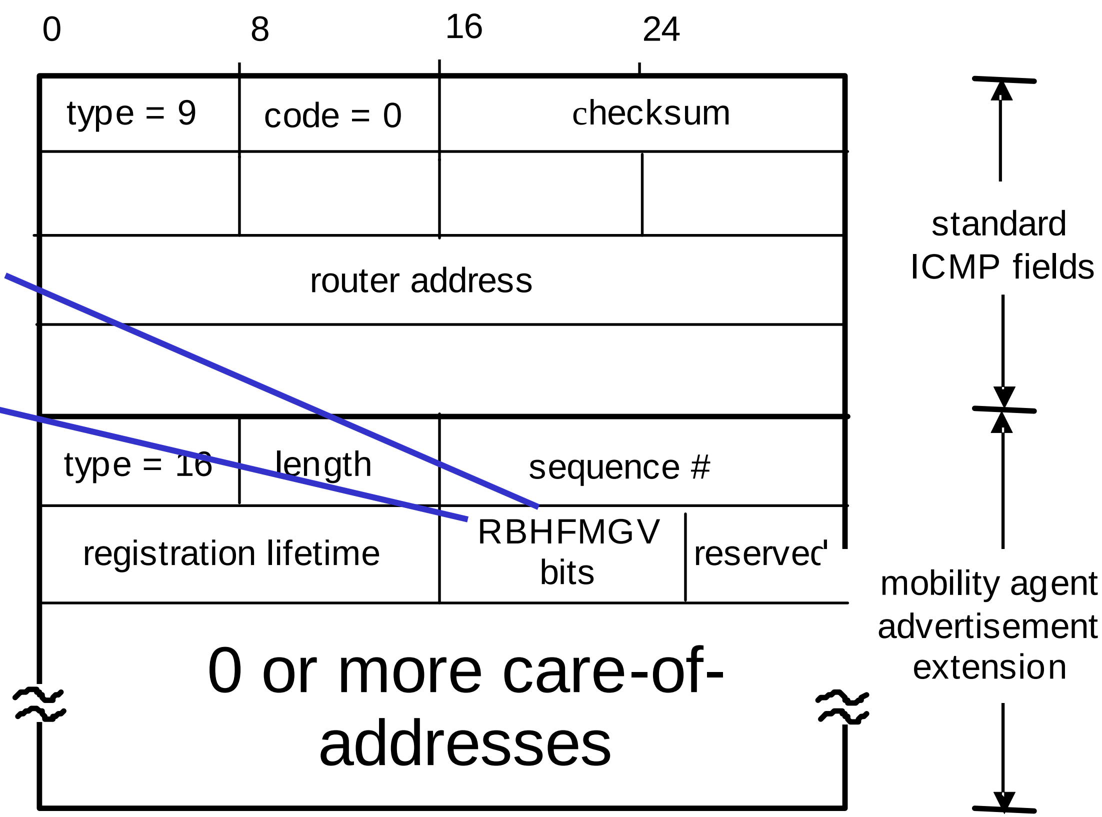
- RFC 3220
- has many features we've seen:
  - home agents, foreign agents, foreign-agent registration, care-of-addresses, encapsulation (packet-within-a-packet)
- three components to standard:
  - agent discovery
  - registration with home agent
  - indirect routing of datagrams

Mobile IP: agent discovery

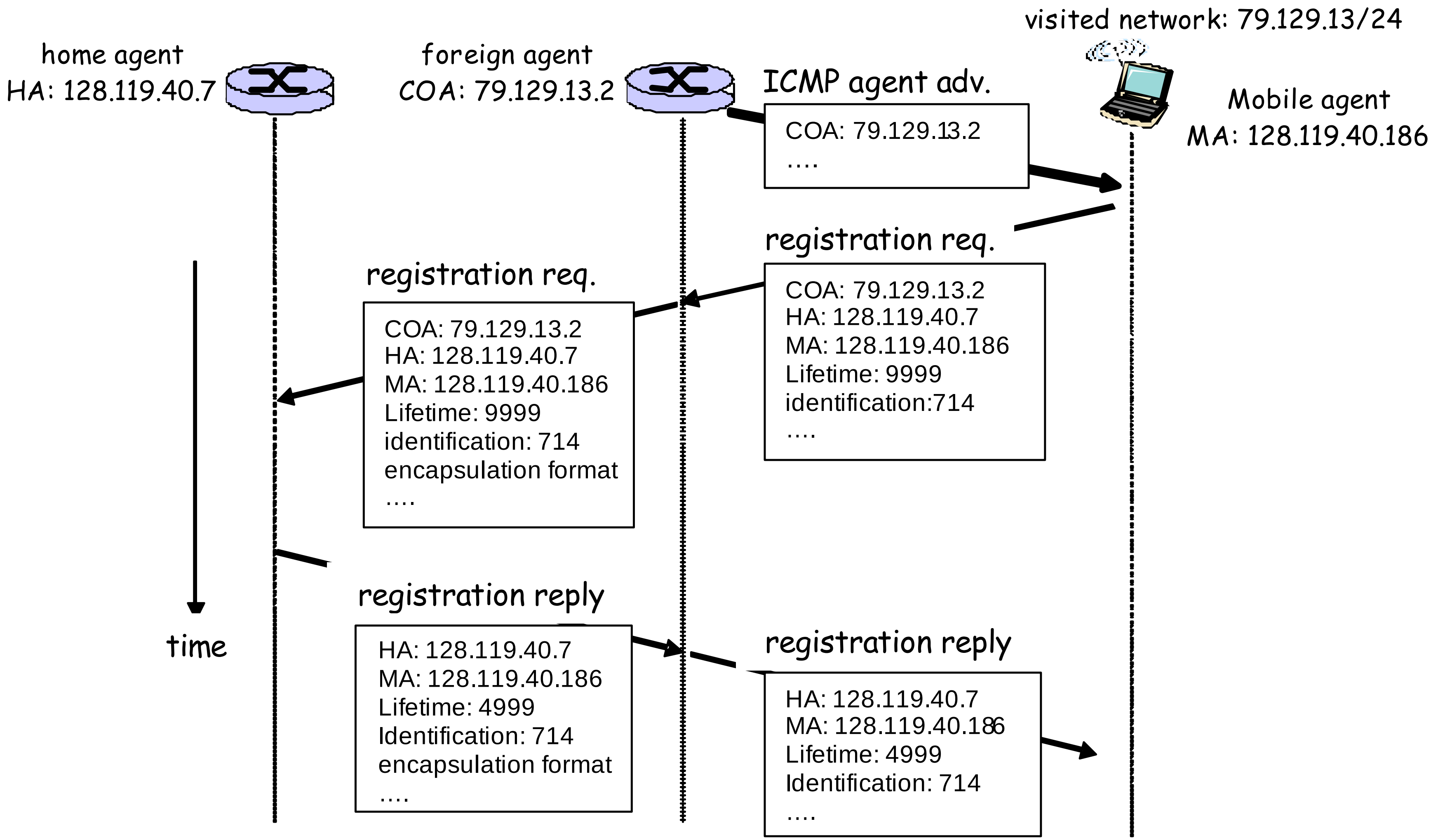
- agent advertisement: foreign/home agents advertise service by broadcasting ICMP messages (typefield = 9)

H,F bits: home and/or foreign agent

R bit: registration required



Mobile IP: registration example



Network Layer: summary

What we've covered:

- network layer services
- routing principles: link state and distance vector
- hierarchical routing
- IP
- Internet routing protocols RIP, OSPF, BGP
- what's inside a router?
- IPv6
- mobility

Next stop:
the Data
link layer!

BGP messages

- 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