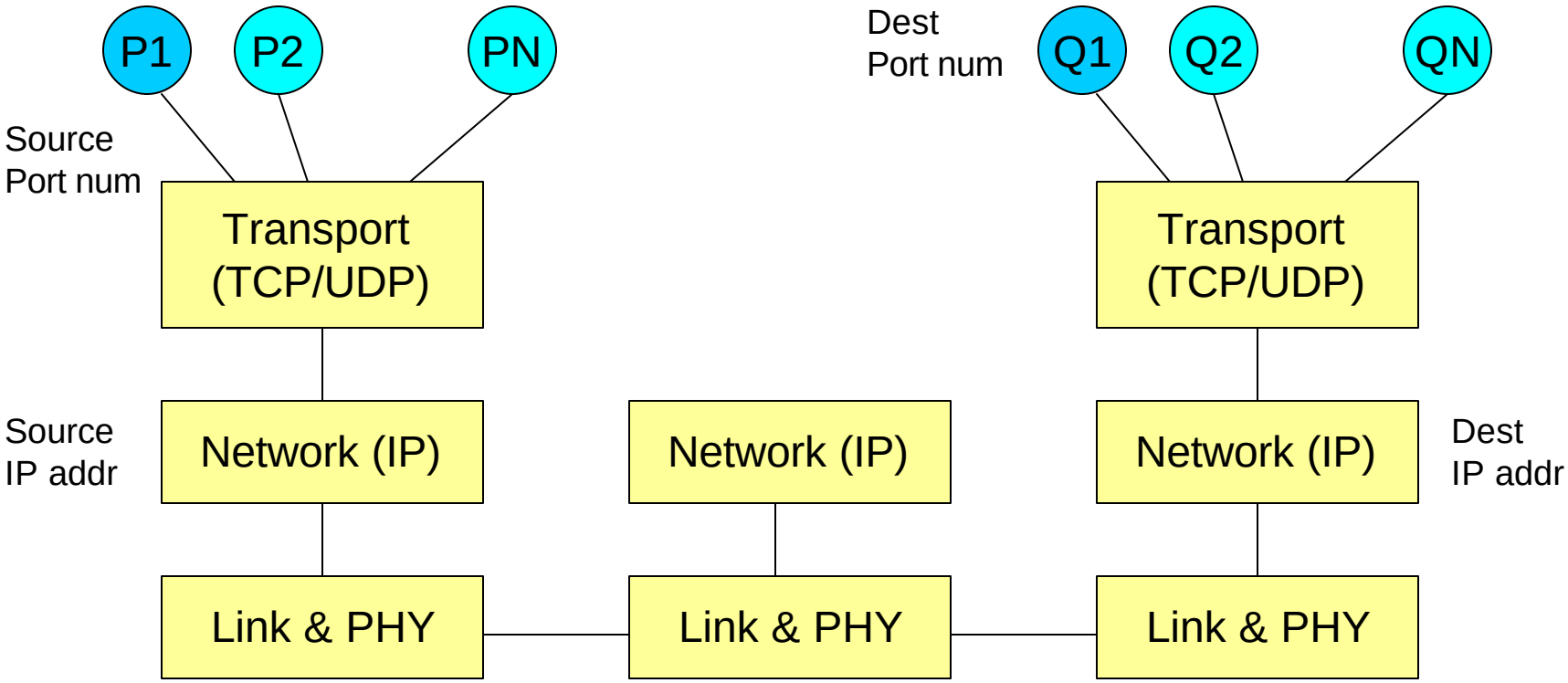
CDA 4506
Design and Implementation of Data Communication Networks

Lecture Set 3

Dr. R. Lent

Simple Communication Model



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

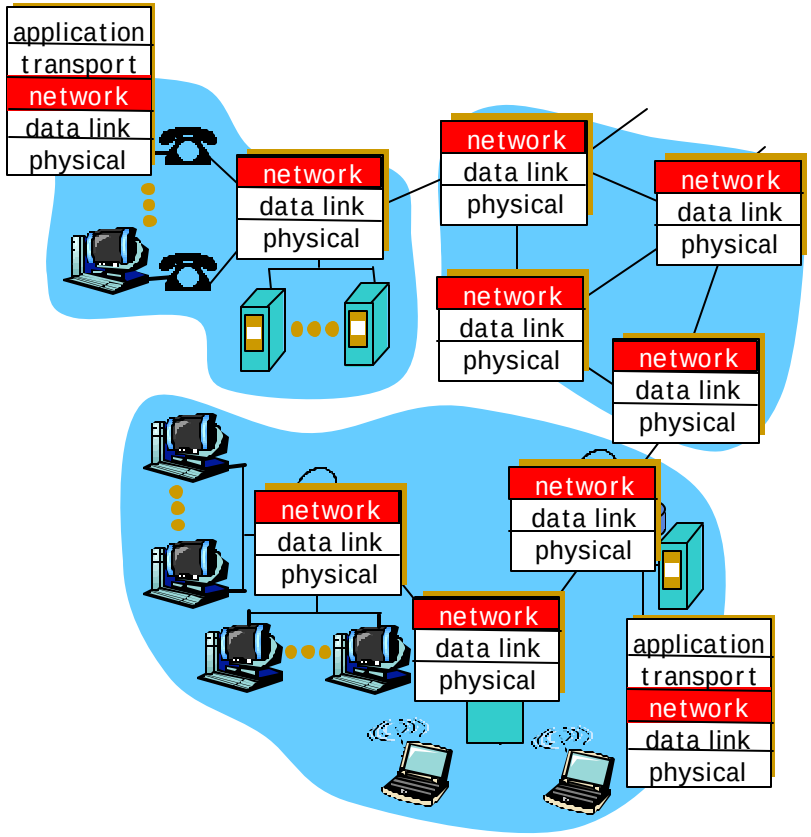
4.7 IPv6

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



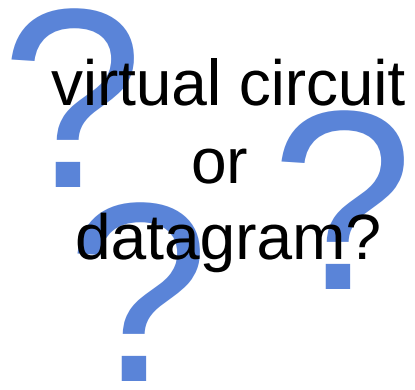
Network service model

service abstraction

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

- 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 circuits

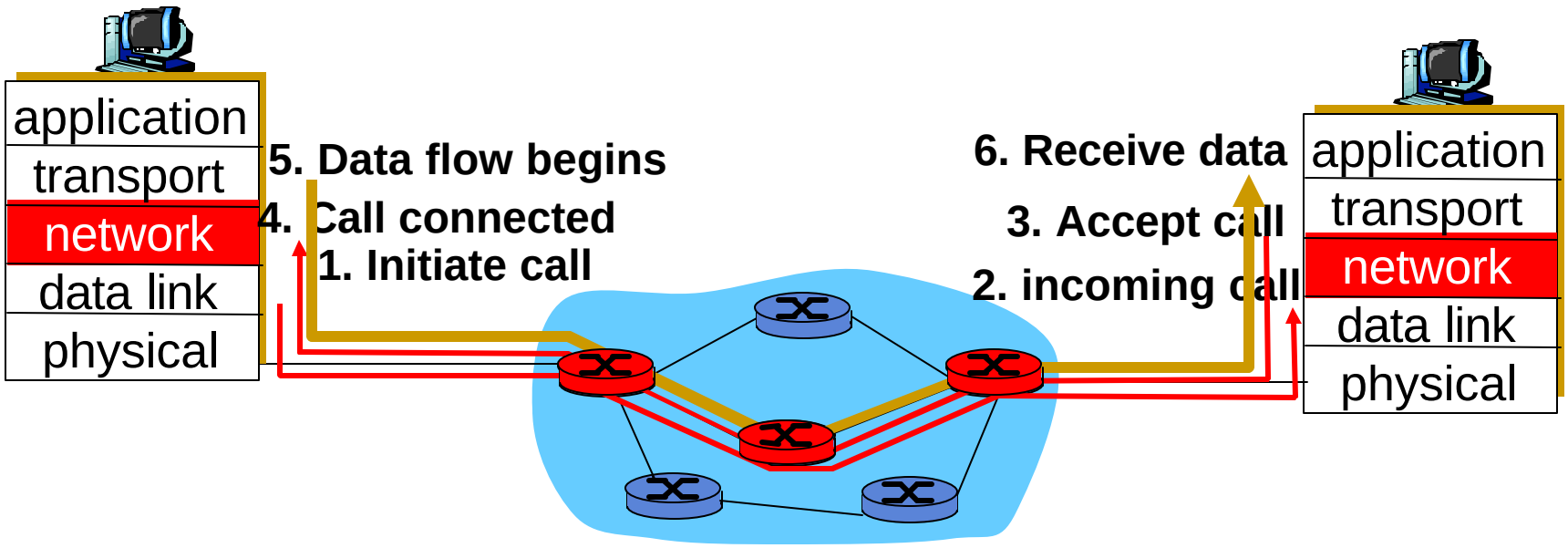
"Source-to-dest path behaves much like telephone circuit"

- 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
  - Similar to a Network layer connection-oriented service
  - transport-layer connection only involves two end systems
- Link, router resources (bandwidth, buffers) may be *allocated* to VC to get circuit-like performance

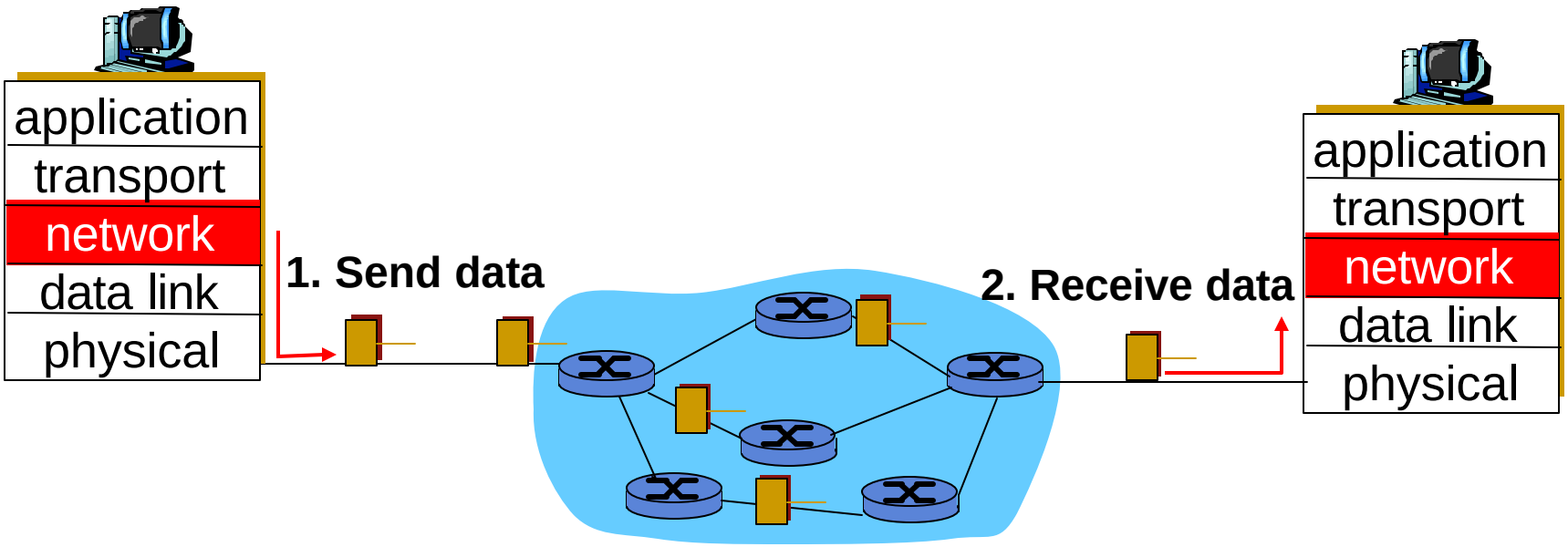
Virtual circuits: signaling protocols

- used to setup, maintain teardown VC
- used in ATM, frame-relay, X.25



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
  - Routing or forwarding table
  - Packets between same source-dest pair may take different paths



Datagram or VC network: why?

Datagram (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” (easier to interconnect networks)
- many link types
  - different characteristics
  - uniform service difficult

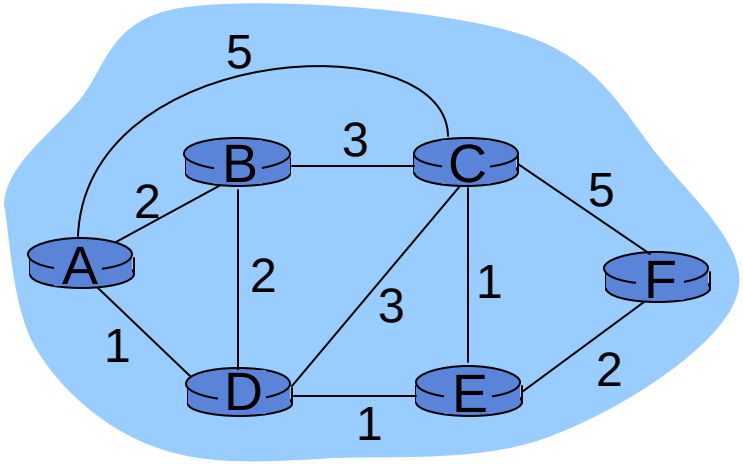
VC (e.g. ATM)

- evolved from telephony
- human conversation:
  - strict timing, reliability requirements
  - need for guaranteed service
  - Complex system!
- “dumb” end systems
  - telephones
  - complexity inside network

Routing

Routing protocol

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



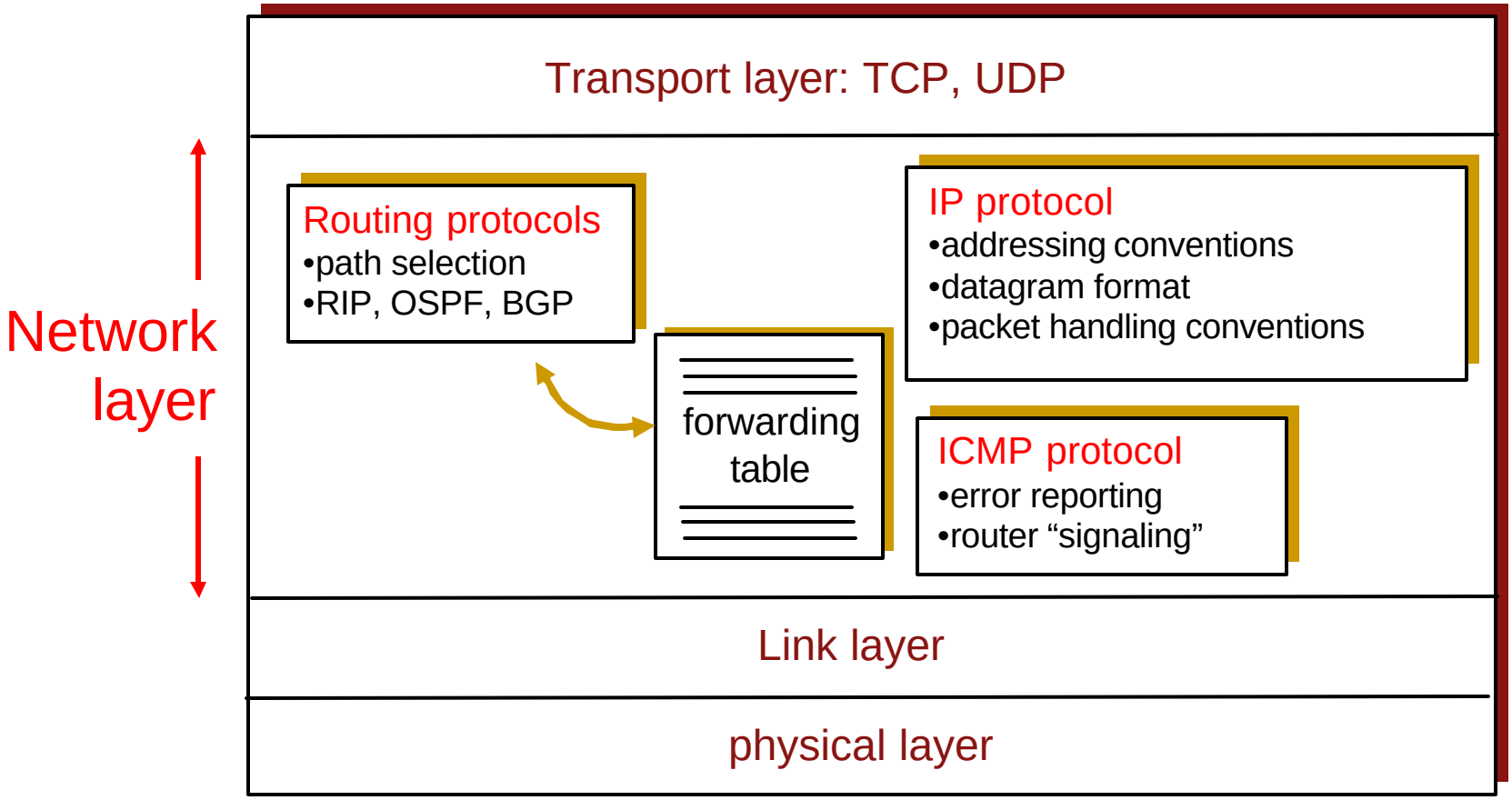
Graph abstraction for routing algorithms:

- 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

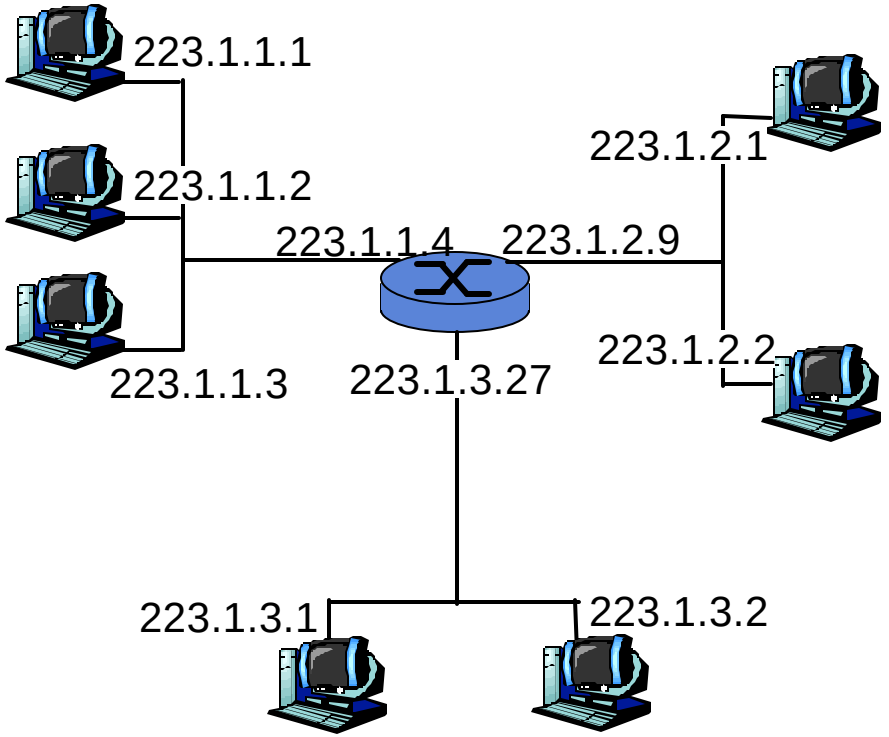
The Internet Network layer

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

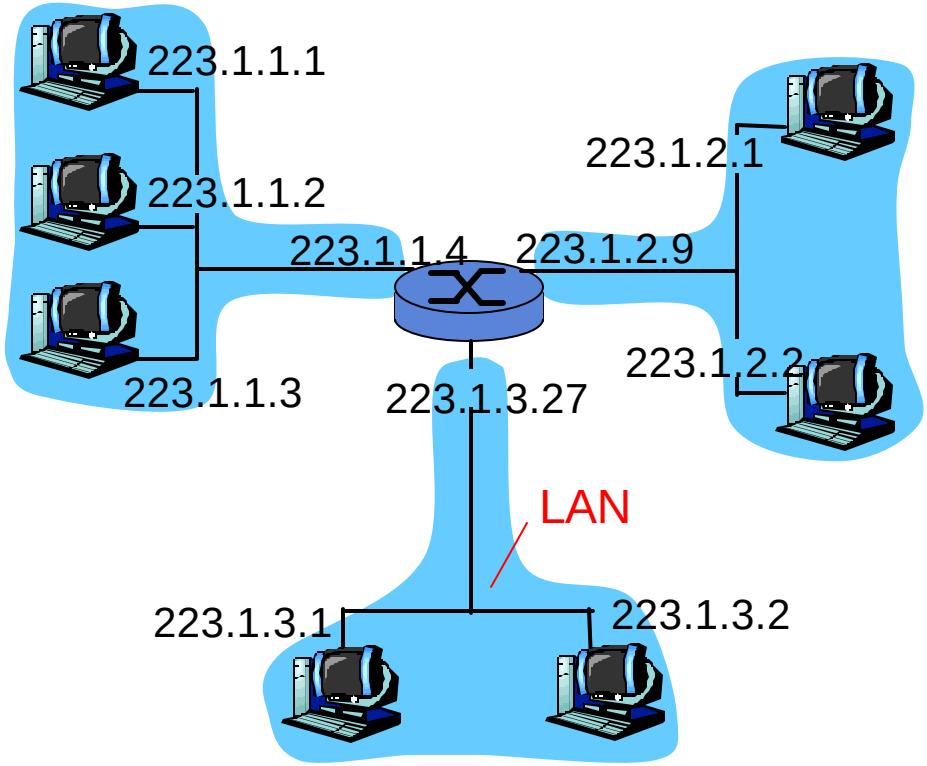


223.1.1.1 = 11011111 00000001 00000001 00000001

223 1 1 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

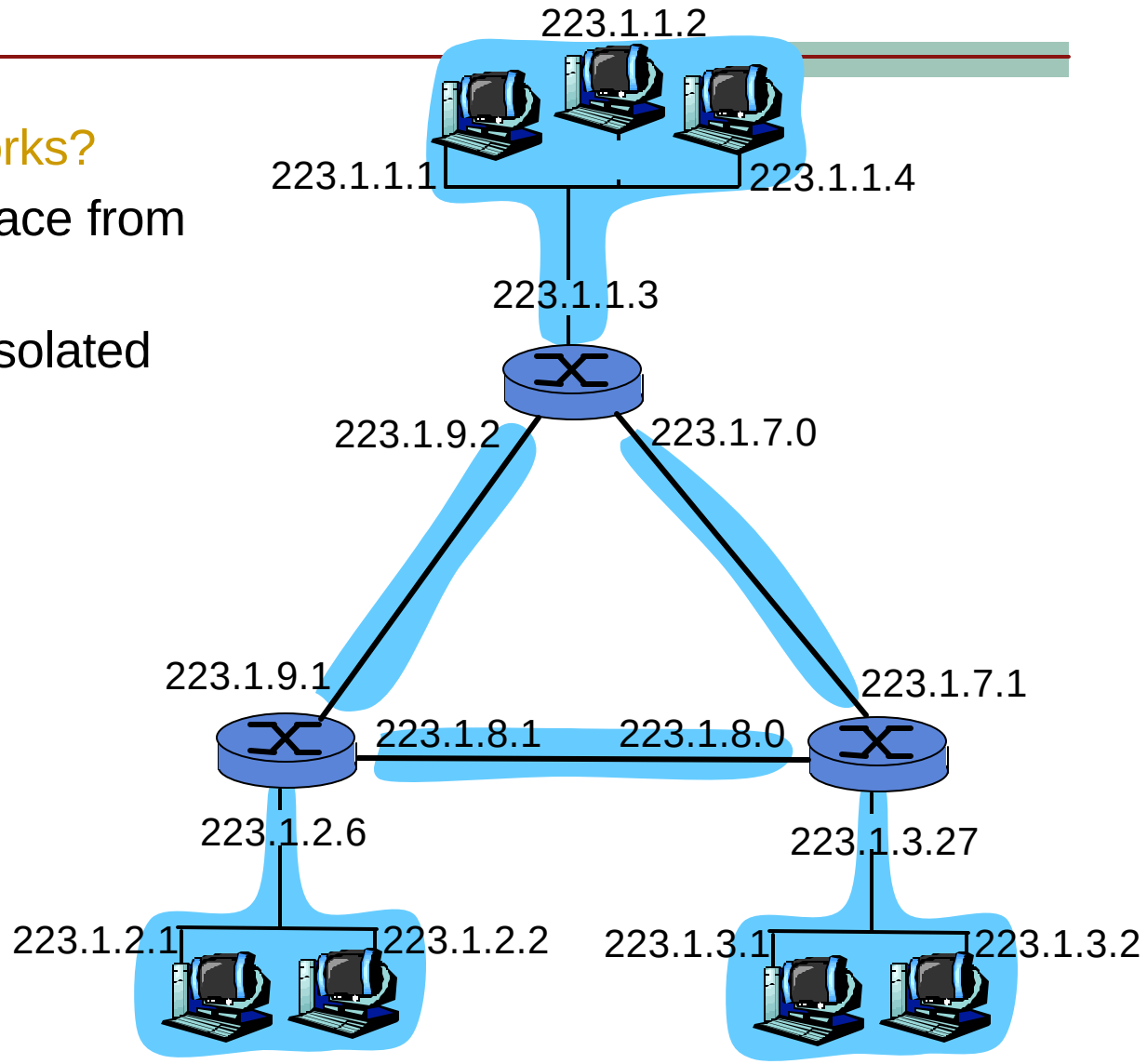


network consisting of 3 IP networks (for IP addresses starting with 223, first 24 bits are network address)

IP Addressing

How to find the networks?

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

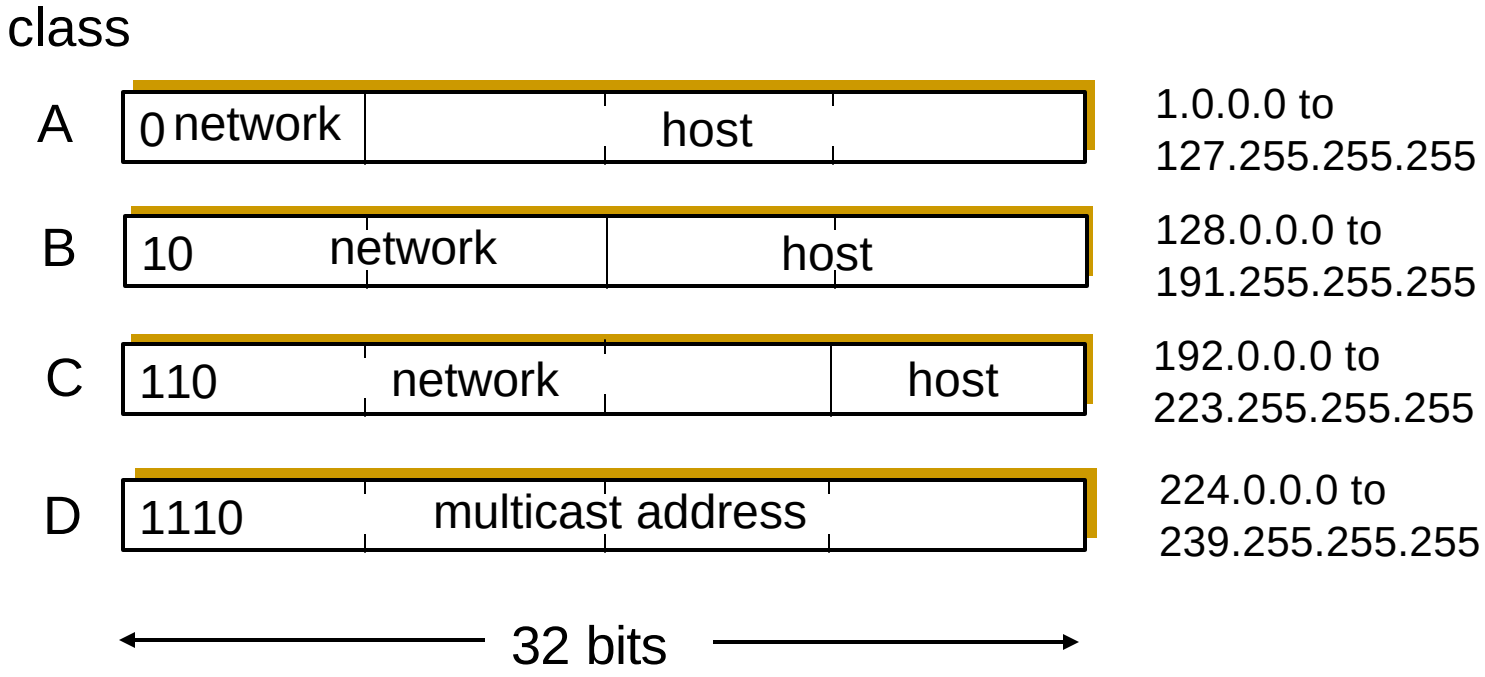


Interconnected system consisting of six networks

IP Addresses

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

“class-full” addressing:



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

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?

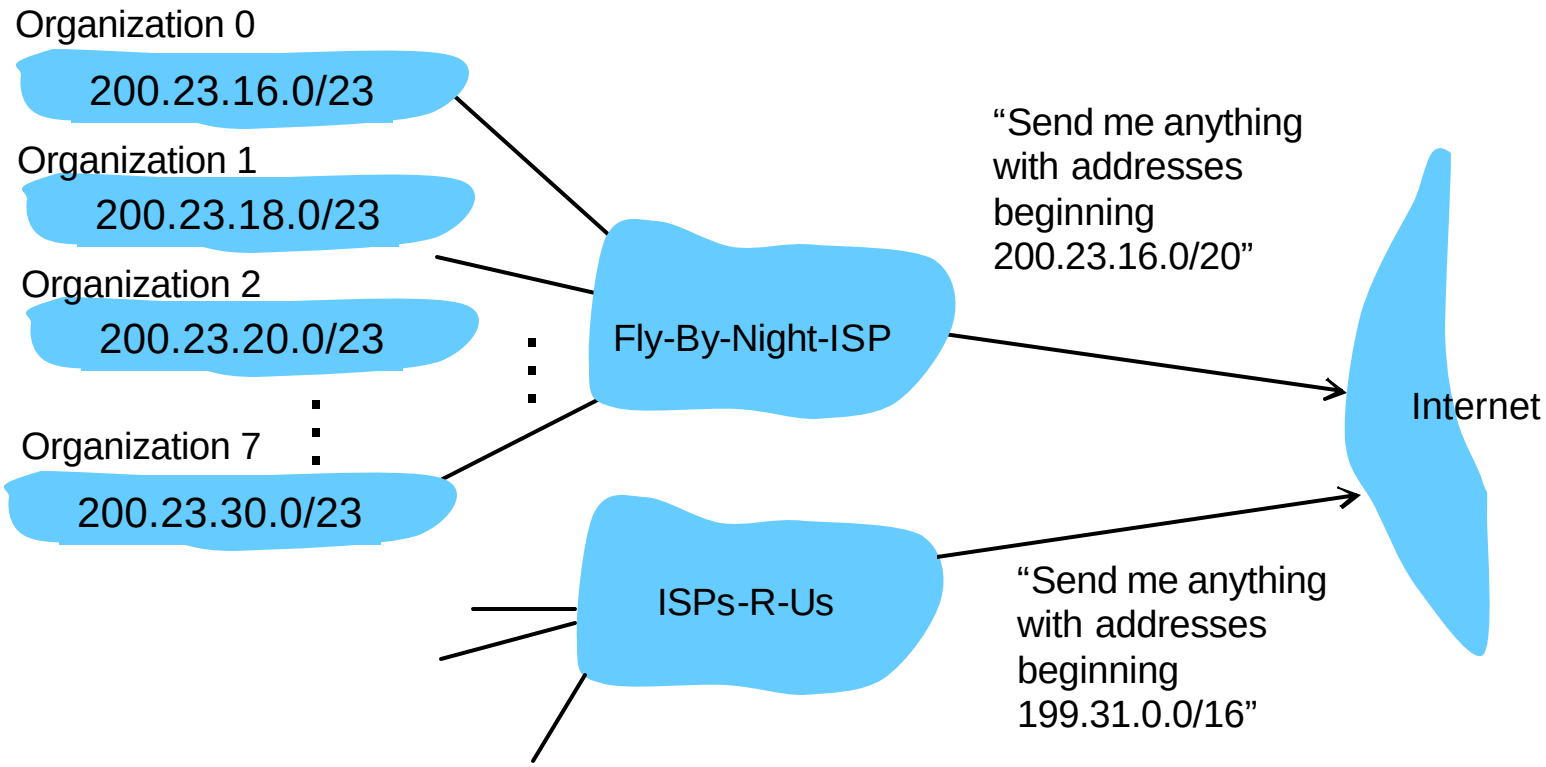
Q: How does *network* get network part of IP addr?

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

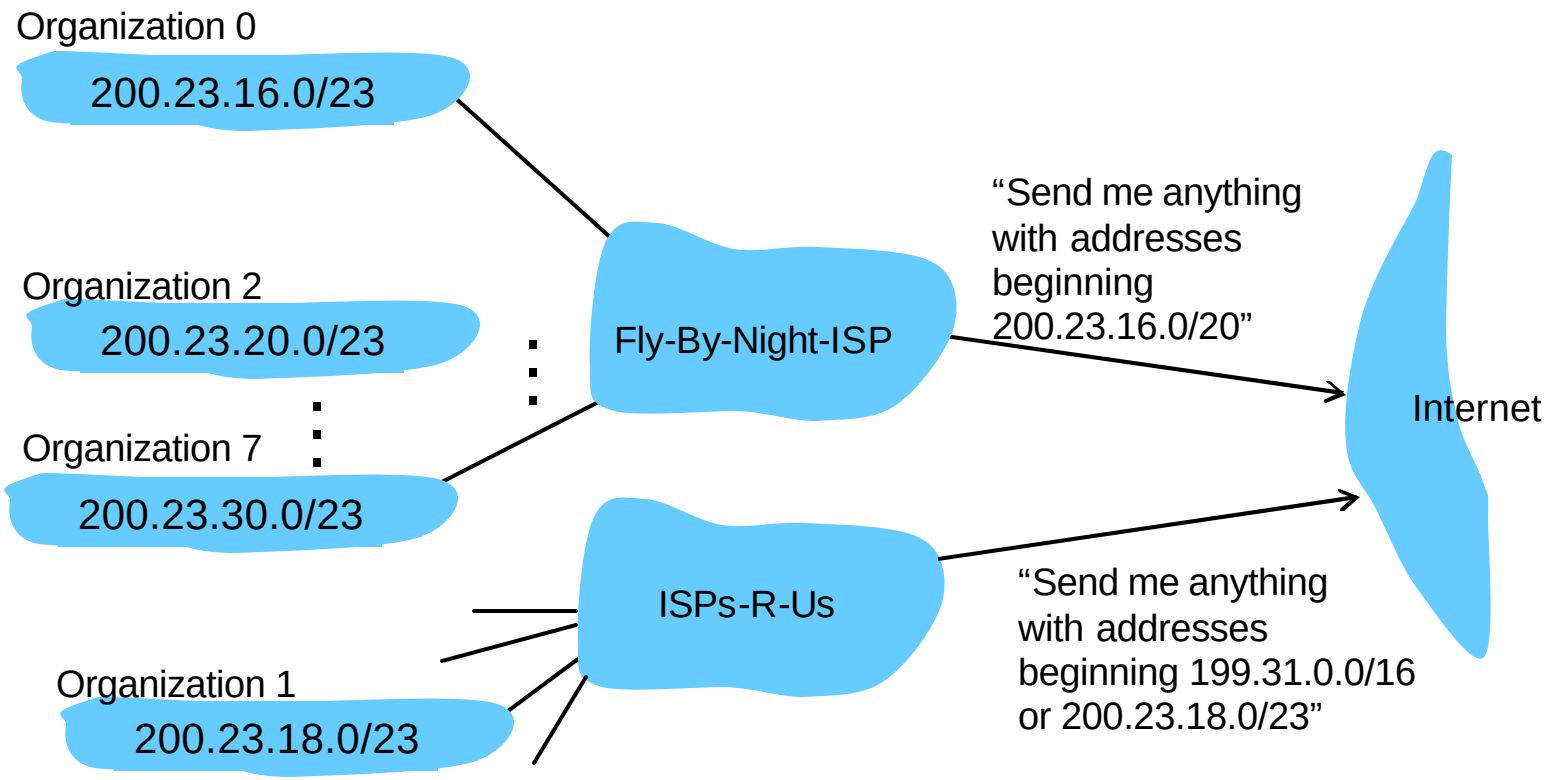
Hierarchical addressing: route aggregation

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.

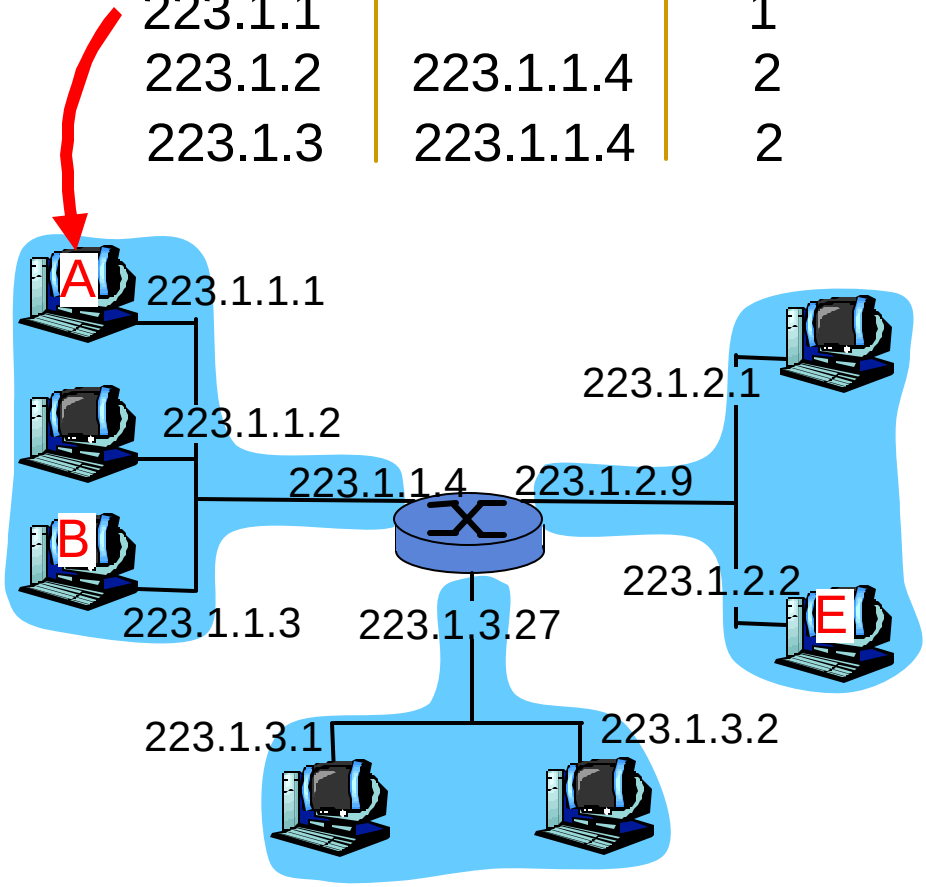
IP datagram:

misc fields	source IP addr	dest IP addr	data

- datagram remains unchanged, as it travels source to destination

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



Getting a datagram from source to dest.

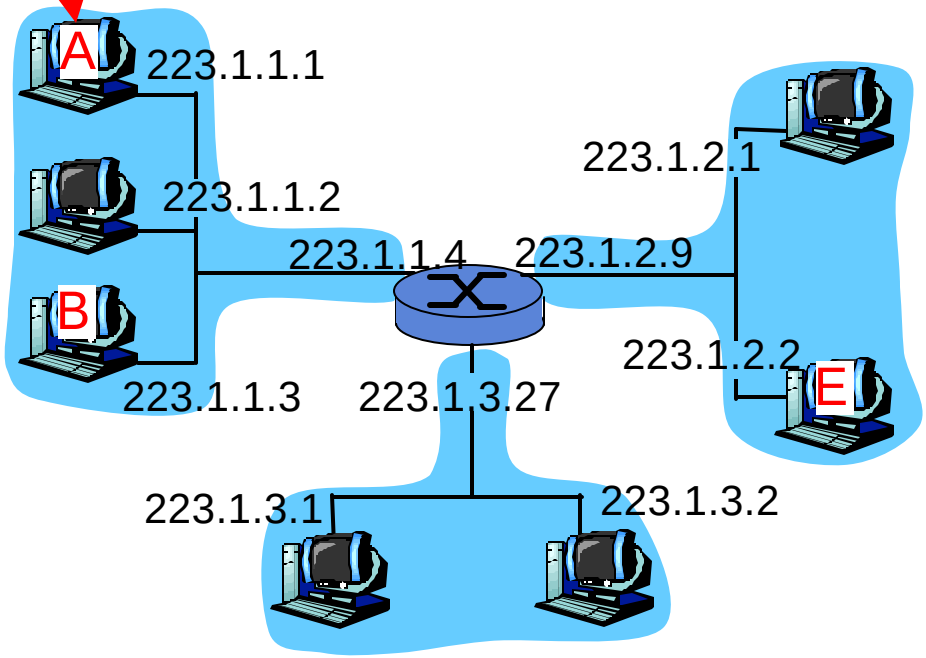
misc fields	223.1.1.1	223.1.1.3	data

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

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



Getting a datagram from source to dest.

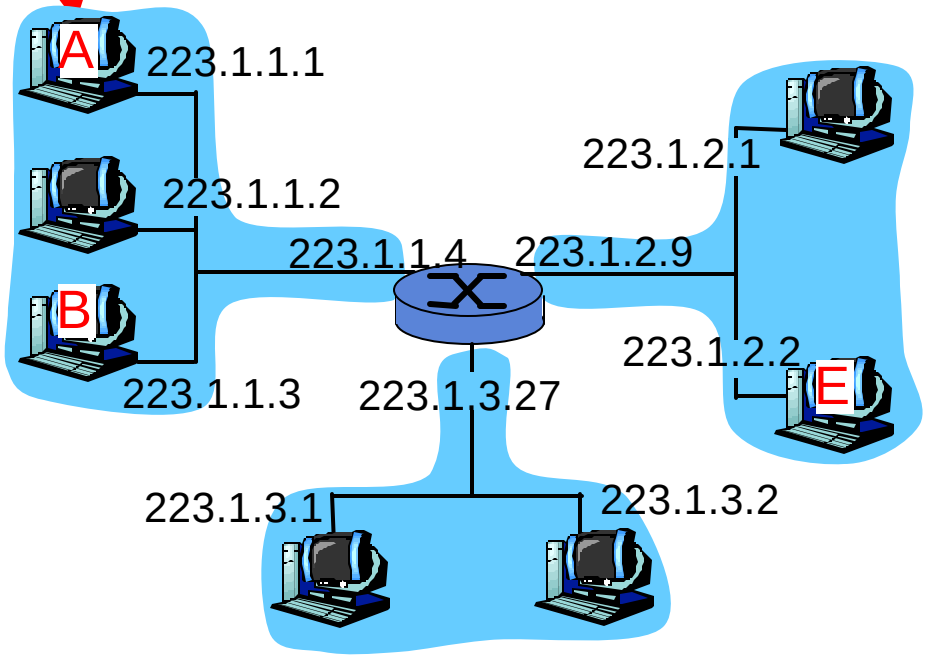
misc fields	223.1.1.1	223.1.2.3	data

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…..

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



Getting a datagram from source to dest.

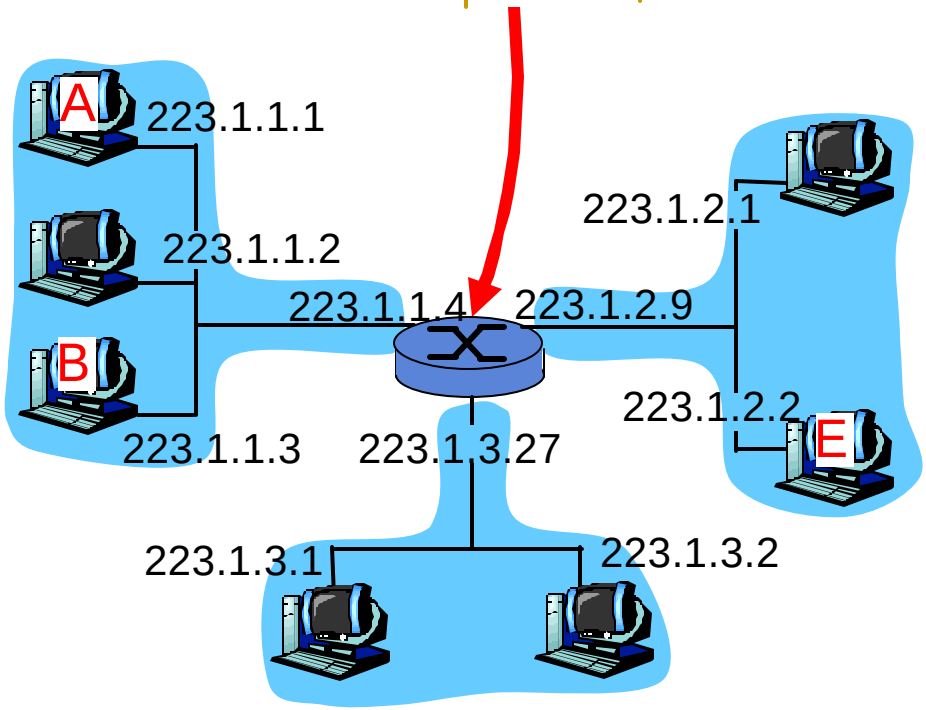
misc fields	223.1.1.1	223.1.2.3	data

Arriving at 223.1.4, destined for 223.1.2.2

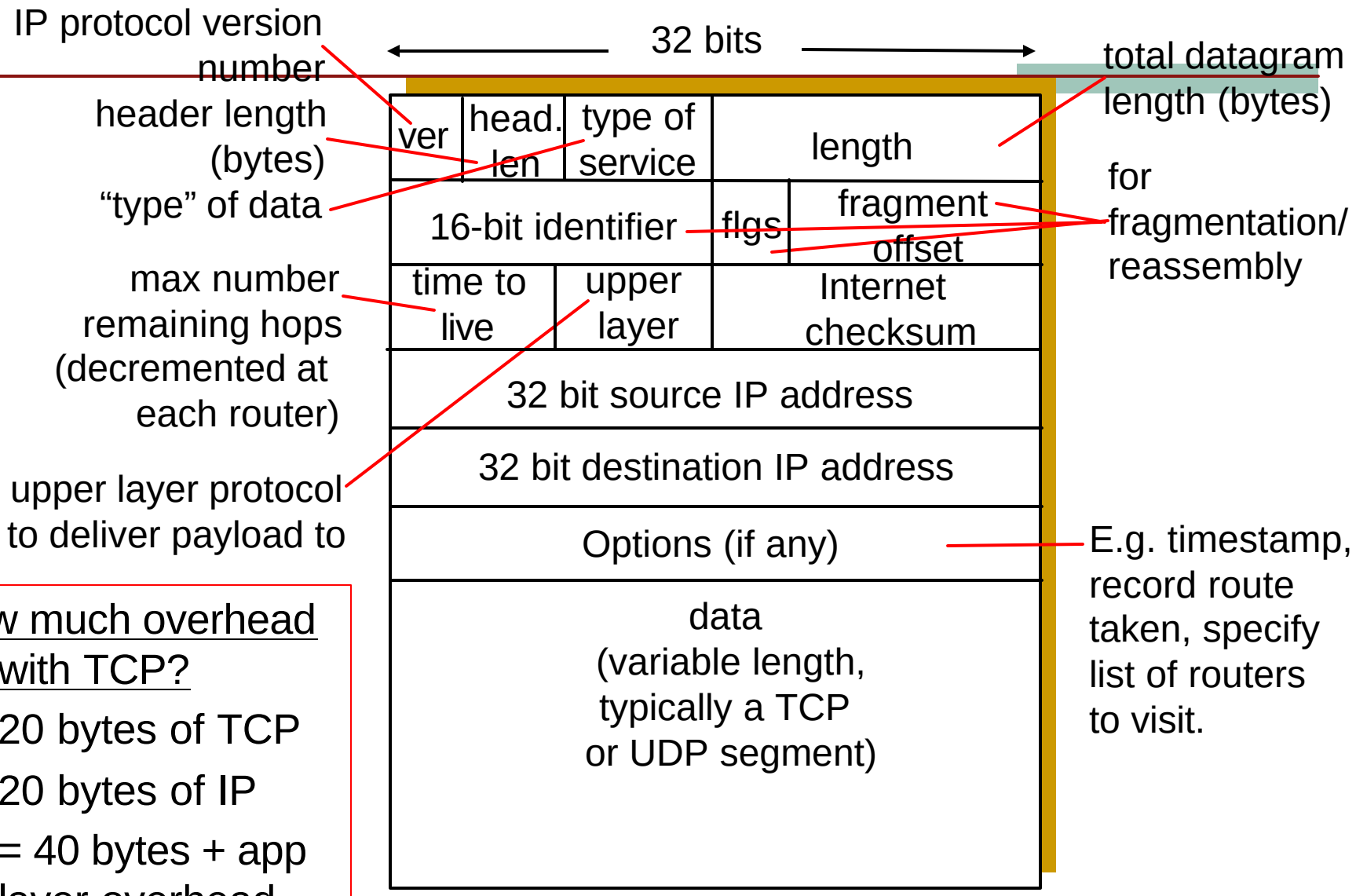
- look up network address of E in router's forwarding table
- E on *same* network as router's interface 223.1.2.9
  - router, E directly attached
- link layer sends datagram to 223.1.2.2 inside link-layer frame via interface 223.1.2.9
- datagram arrives at 223.1.2.2!!! (hooray!)

forwarding 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



IP datagram format

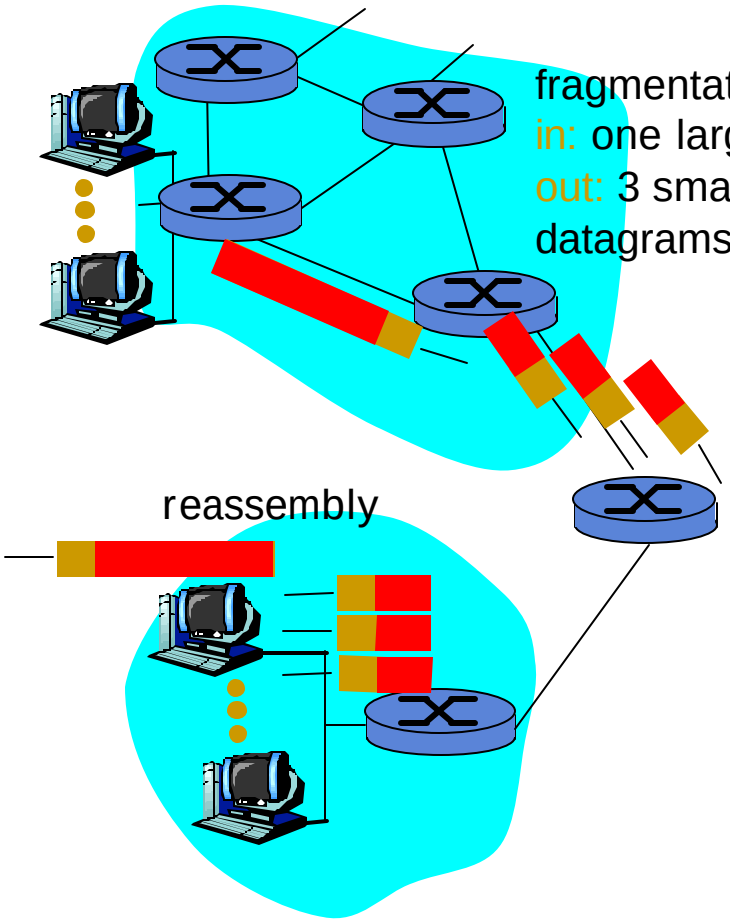


how much overhead with TCP?

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

IP Fragmentation & Reassembly

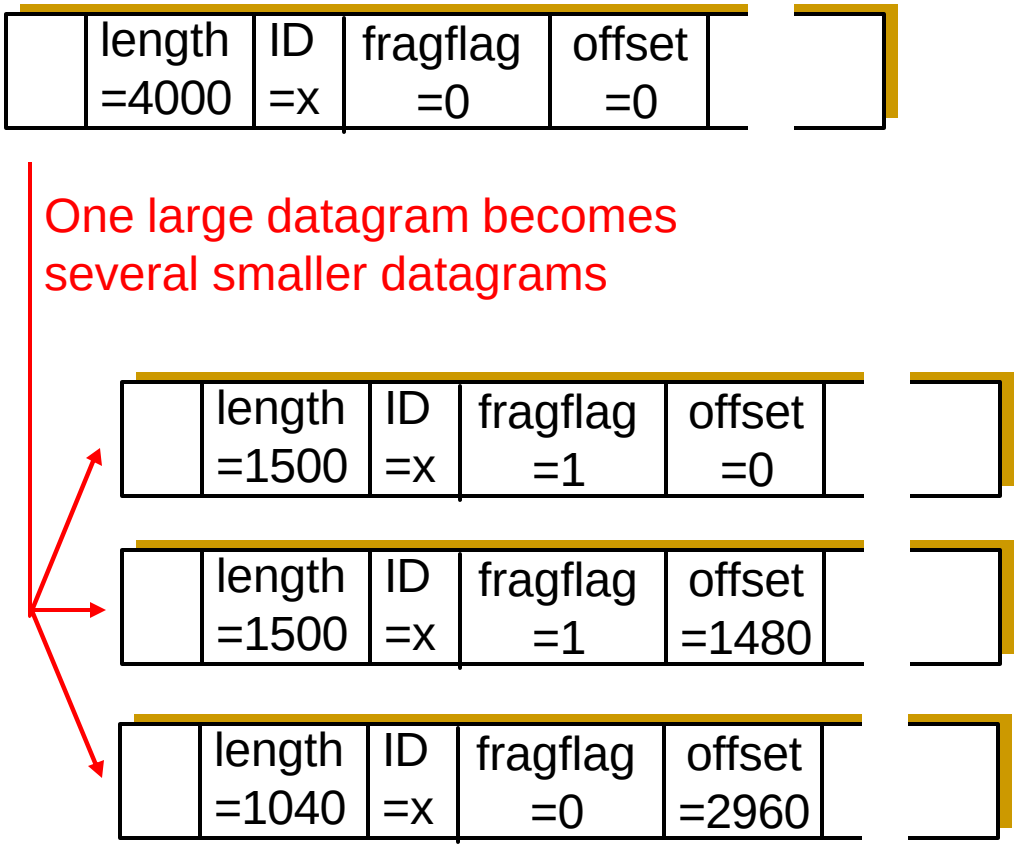
- network links have MTU (max.transfer size) - largest possible link-level frame
  - different link types, different MTUs
- large IP datagram divided ("fragmented") within net
  - one datagram becomes several datagrams
  - "reassembled" only at final destination
  - IP header bits used to identify, order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



IP Fragmentation and Reassembly

Example

- 4000 byte datagram
- MTU = 1500 bytes



DHCP: Dynamic Host Configuration Protocol

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

Can renew its lease on address in use

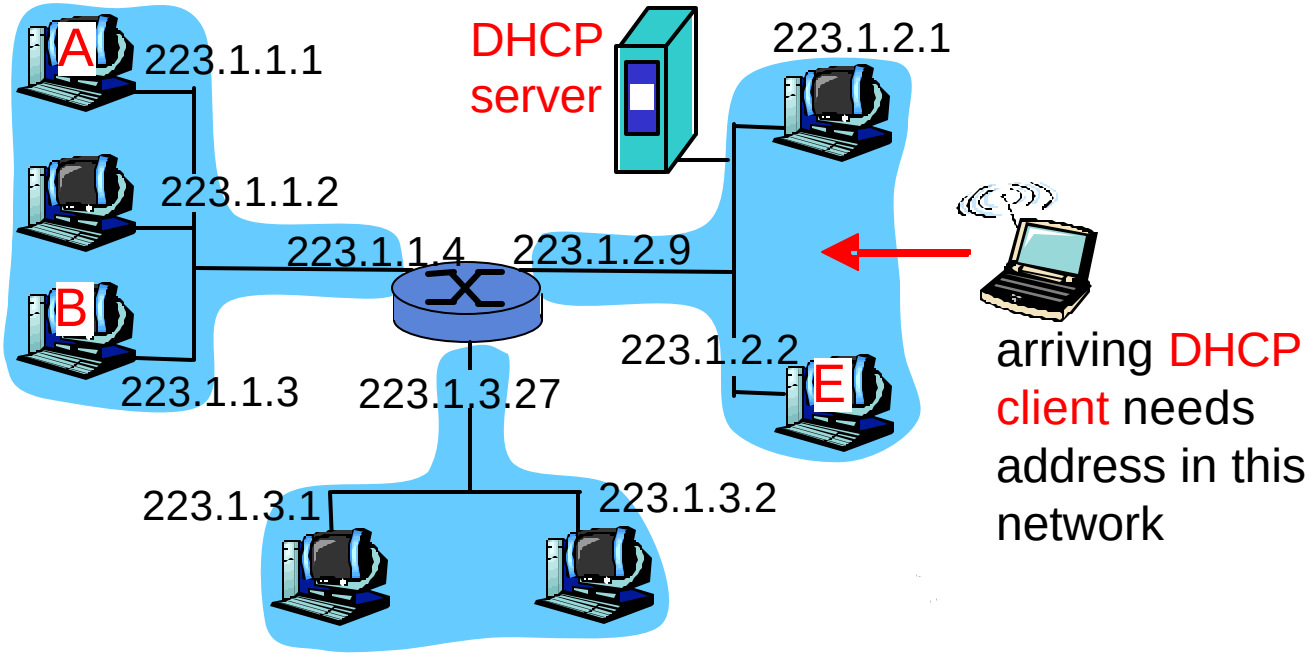
Allows reuse of addresses (only hold address while connected)

Support for mobile users who want to join network

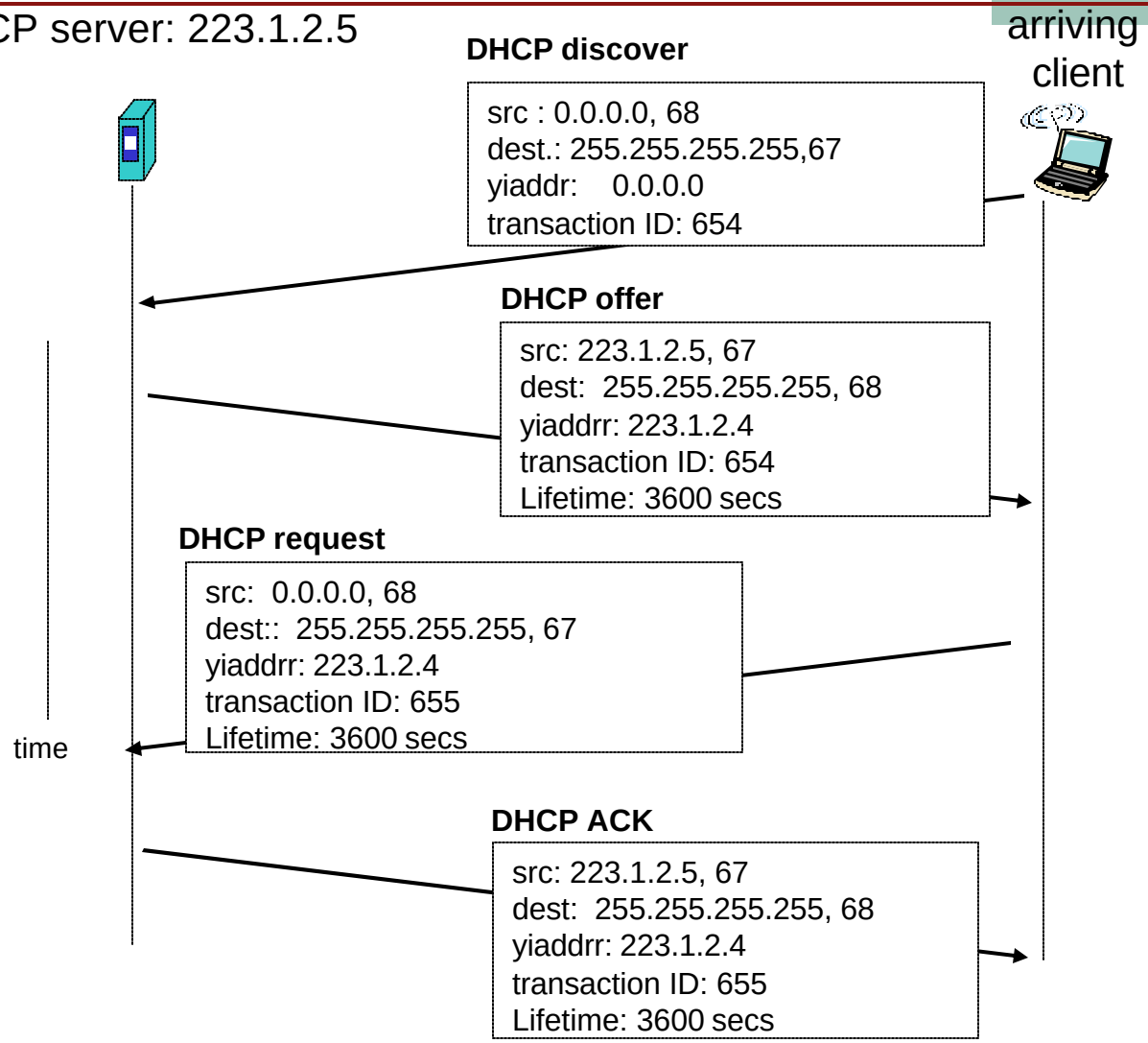
DHCP overview:

- 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

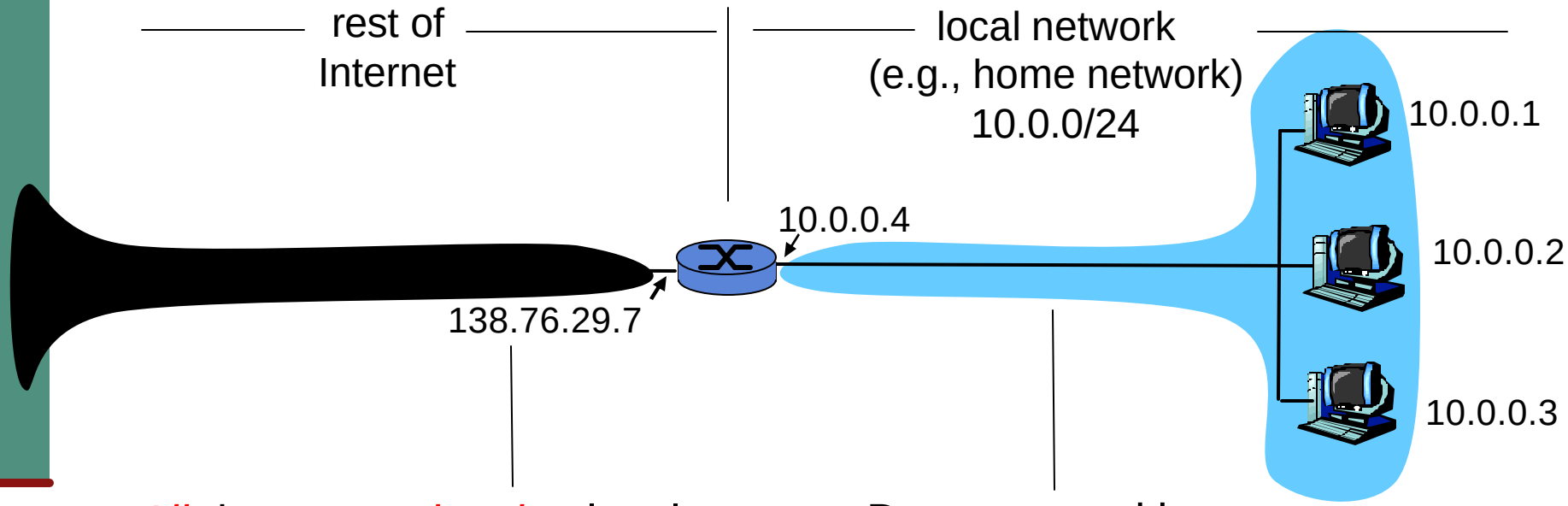


ICMP: Internet Control Message Protocol

- 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

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 word 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:

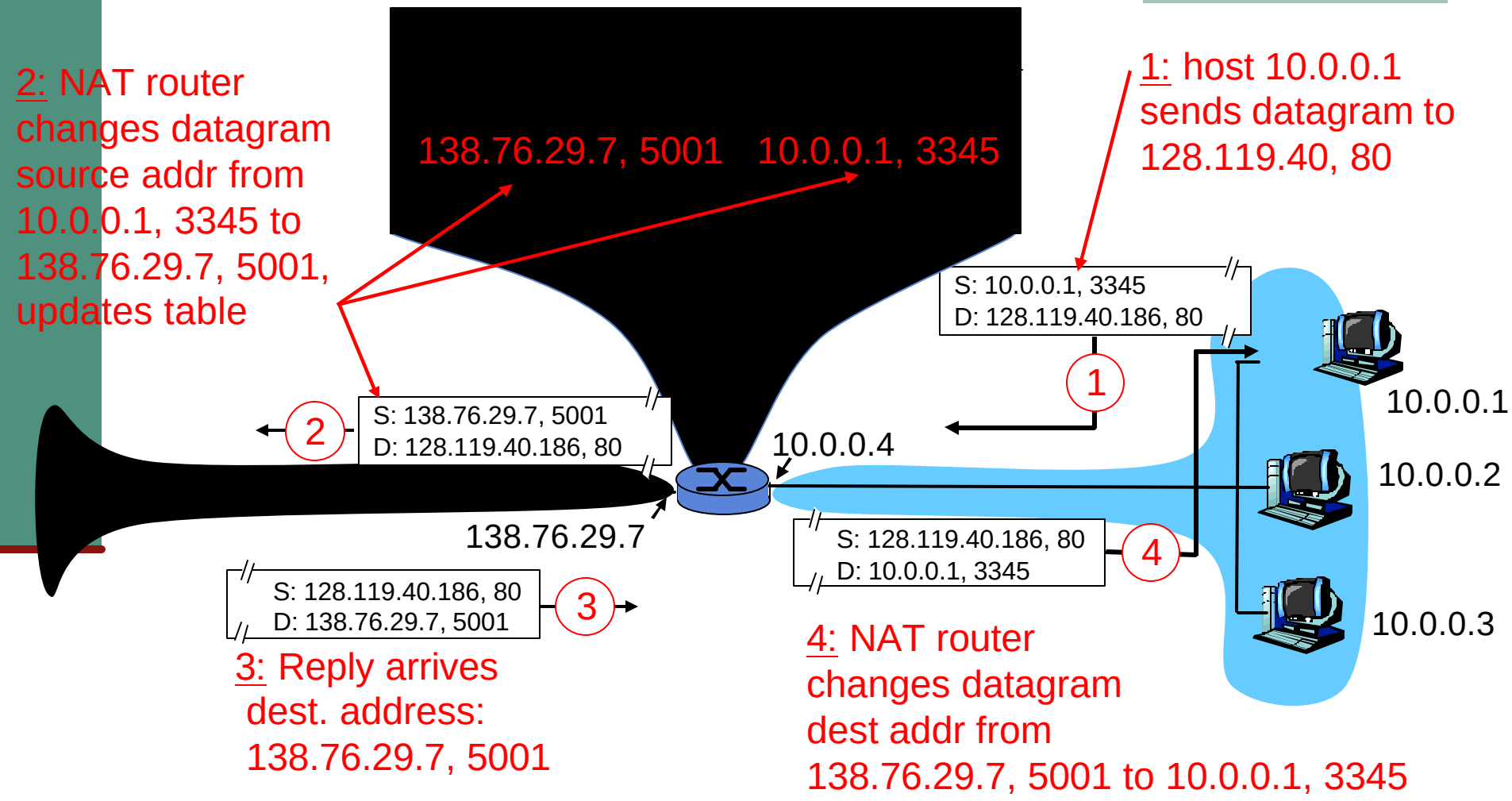
- *outgoing datagrams: replace* (source IP address, port #) of every outgoing datagram to (NAT IP address, new port #)

  . . . 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



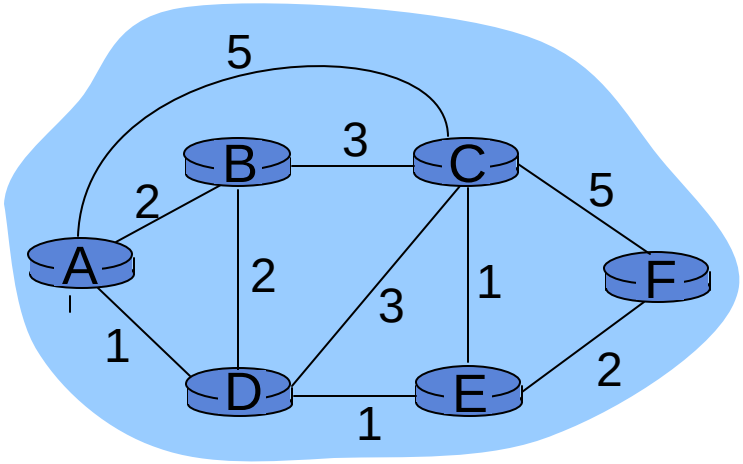
NAT: Network Address Translation

- 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, eg, P2P applications
  - address shortage should instead be solved by IPv6

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 or networks
- 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

Deterministic or Random?

- Non-det: Random, Flooding, hot potato
- Deterministic: DV, LS

(1) A Link-State Routing Algorithm

Dijkstra's algorithm

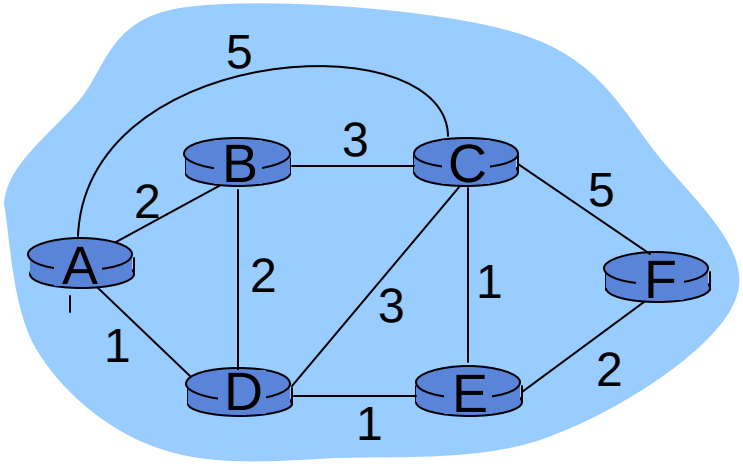
- Net topology, link costs known to all nodes
  - accomplished via "link state broadcast"
  - all nodes have same info
- Computes least cost paths from one node ('source") to all other nodes
  - gives routing table for that node
- Iterative: after k iterations, knows least cost path to k destinations

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 is definitively known

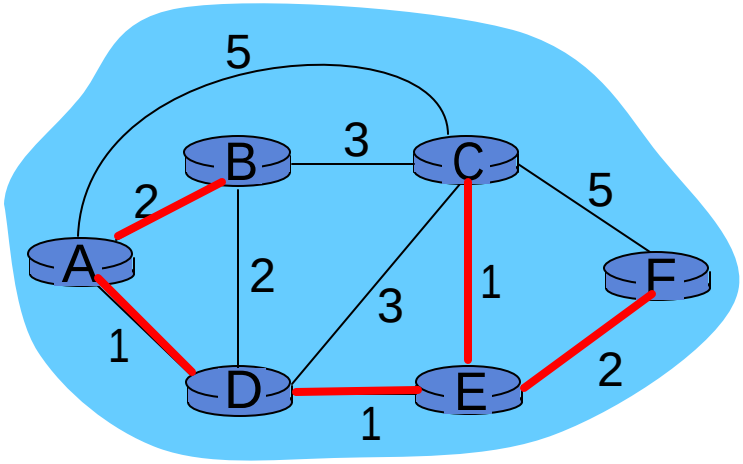
Dijsktra's Algorithm

```
1  Initialization:
2    N = {A}
3    for all nodes v
4      if v adjacent to A
5        then D(v) = c(A,v)
6        else D(v) = infinity
7
8  Loop
9     find w not in N such that D(w) is a minimum
10    add w to N
11    update D(v) for all v adjacent to w and not in N:
12       D(v) = min( D(v), D(w) + c(w,v) )
13    /* new cost to v is either old cost to v or known
14     shortest path cost to w plus cost from w to v */
15 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					



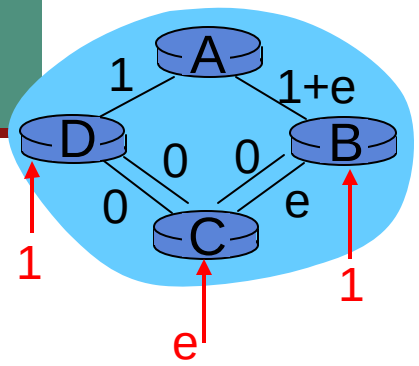
Dijkstra's algorithm, discussion

Algorithm complexity: n nodes

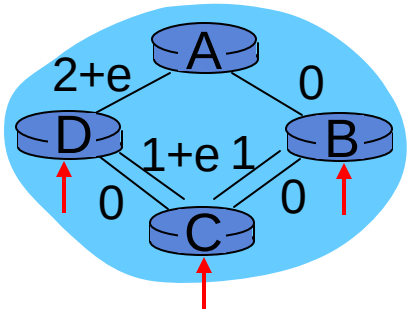
- each iteration: need to check all nodes, w, not in N
- n*(n+1)/2 comparisons: $O(n^2)$
- more efficient implementations possible: O(nlogn)

Oscillations possible:

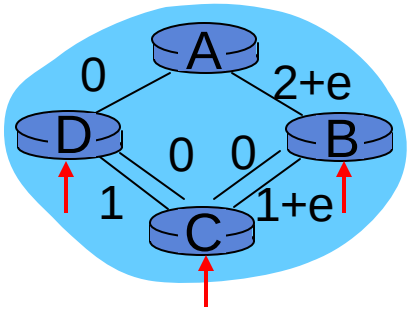
- e.g., link cost = amount of carried traffic



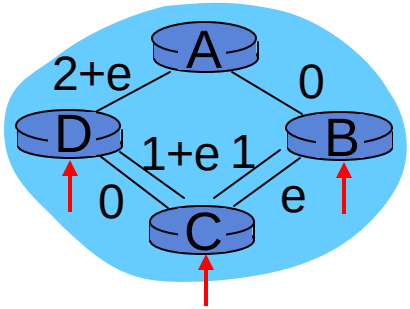
initially



… recompute routing



… recompute



… recompute

(2) 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

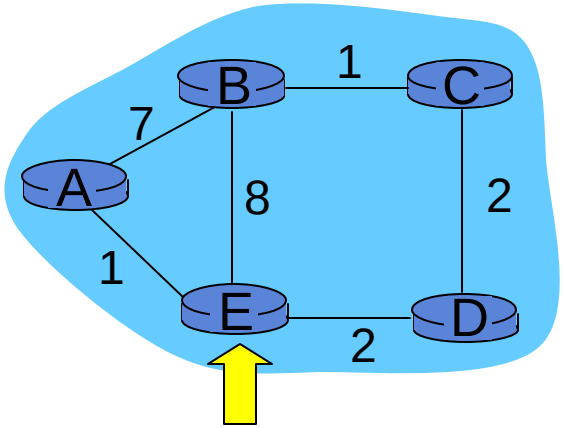
Distance Table data structure

- 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:

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

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

Distance Table: example



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

$= 2+2 = 4$

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

$= 2+3 = 5$ loop!

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

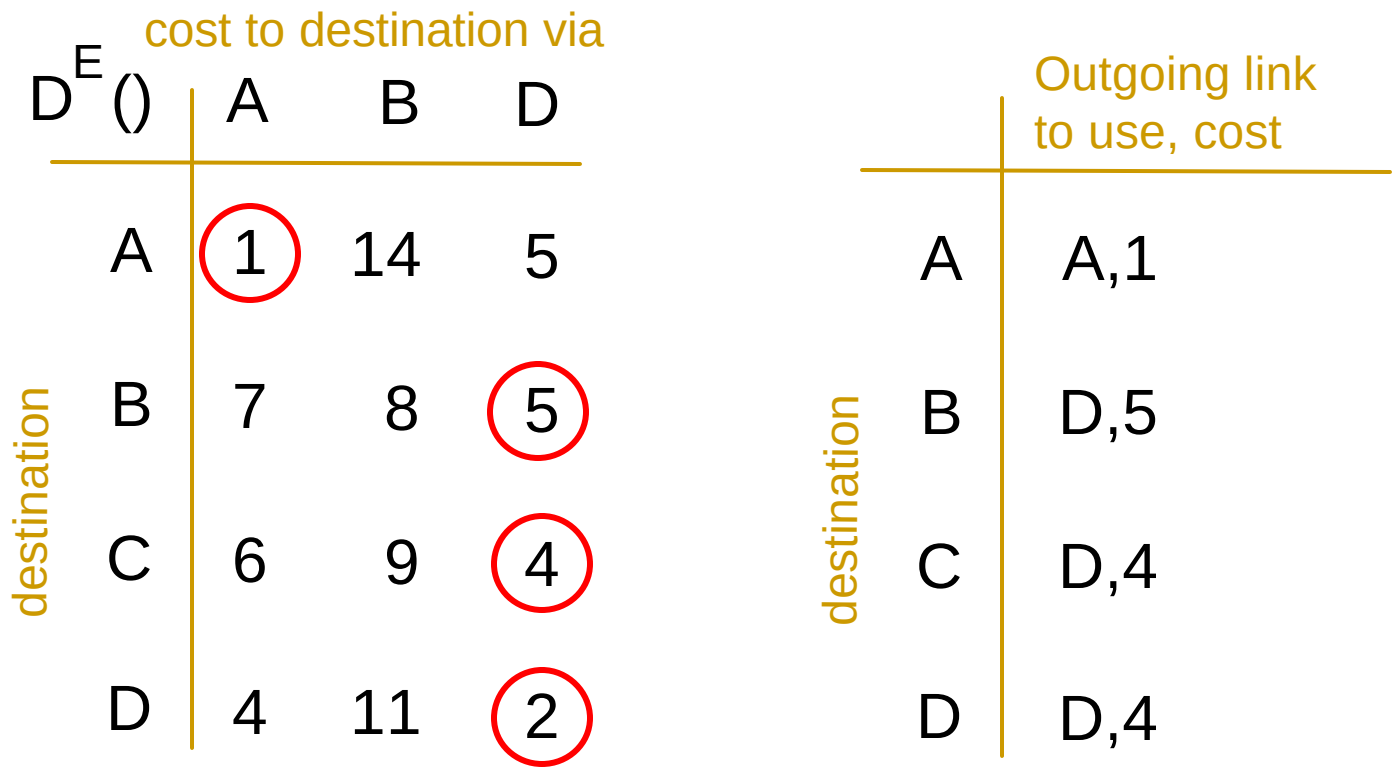
$= 8+6 = 14$ loop!

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



Distance table → Routing table

Distance Vector Routing: overview

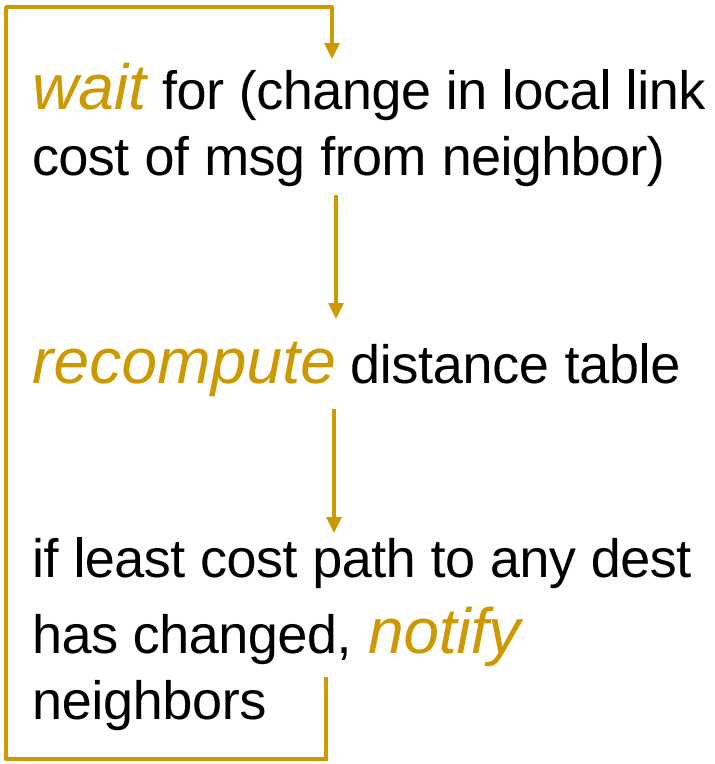
Iterative, asynchronous: each local iteration caused by:

- 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:



Distance Vector Algorithm:

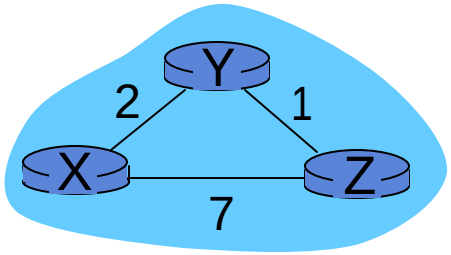
At all nodes, X:

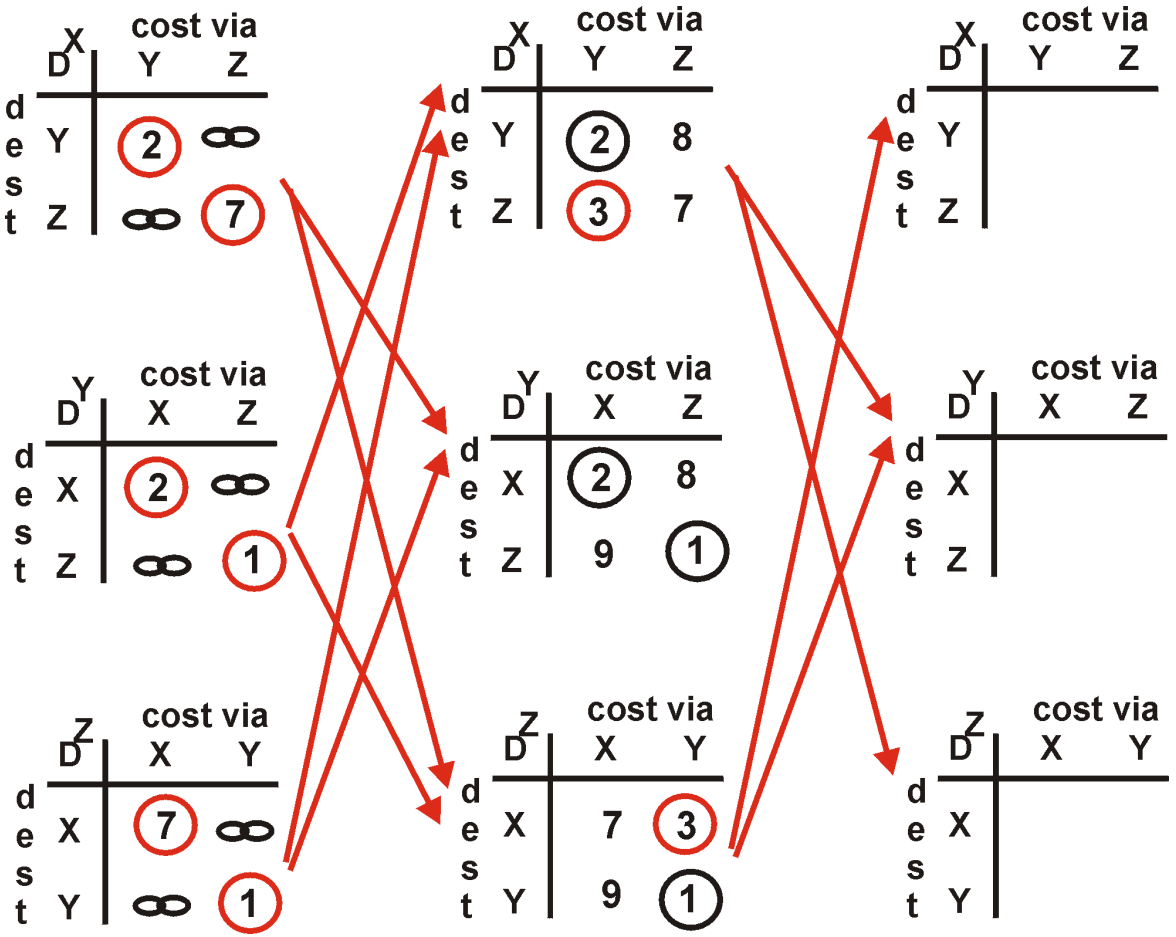
1 Initialization:
2 for all adjacent nodes v:
3 $D^X(*,v) = \text{infinity}$ /* the * operator means "for all rows" */
4 $D^X(v,v) = c(X,v)$
5 for all destinations, y
6 send $\min_w D^X(y,w)$ to each neighbor /* w over all X's neighbors */

Distance Vector Algorithm (cont'd):

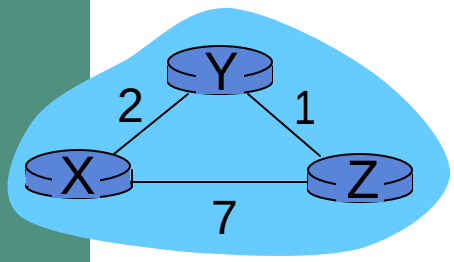
```
8  loop
9    wait (until I see a link cost change to neighbor V
10         or until I receive update from neighbor V)
11
12   if (c(X,V) changes by d)
13     /* change cost to all dest's via neighbor v by d */
14     /* note: d could be positive or negative */
15     for all destinations y:  D^X(y,V) =  D^X(y,V) + d
16
17   else if (update received from V about destination Y)
18     /* shortest path from V to some Y has changed  */
19     /* V has sent a new value for its  min_w DV(Y,w) */
20     /* call this received new value is "newval"     */
21     for the single destination y: D^X(Y,V) = c(X,V) + newval
22
23   if we have a new min_w D^X(Y,w)for any destination Y
24      send new value of min_w D^X(Y,w) to all neighbors
25
26 forever
```

Distance Vector Algorithm: example





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

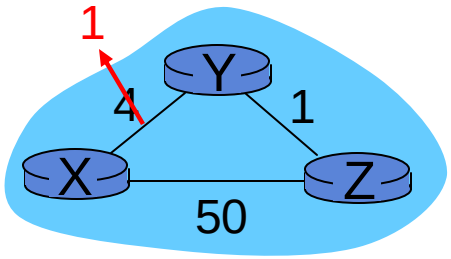
$$D^X(Y,Z) = c(X,Z) + \min_w\{D^Z(Y,w)\}$$
$$= 7+1 = 8$$

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

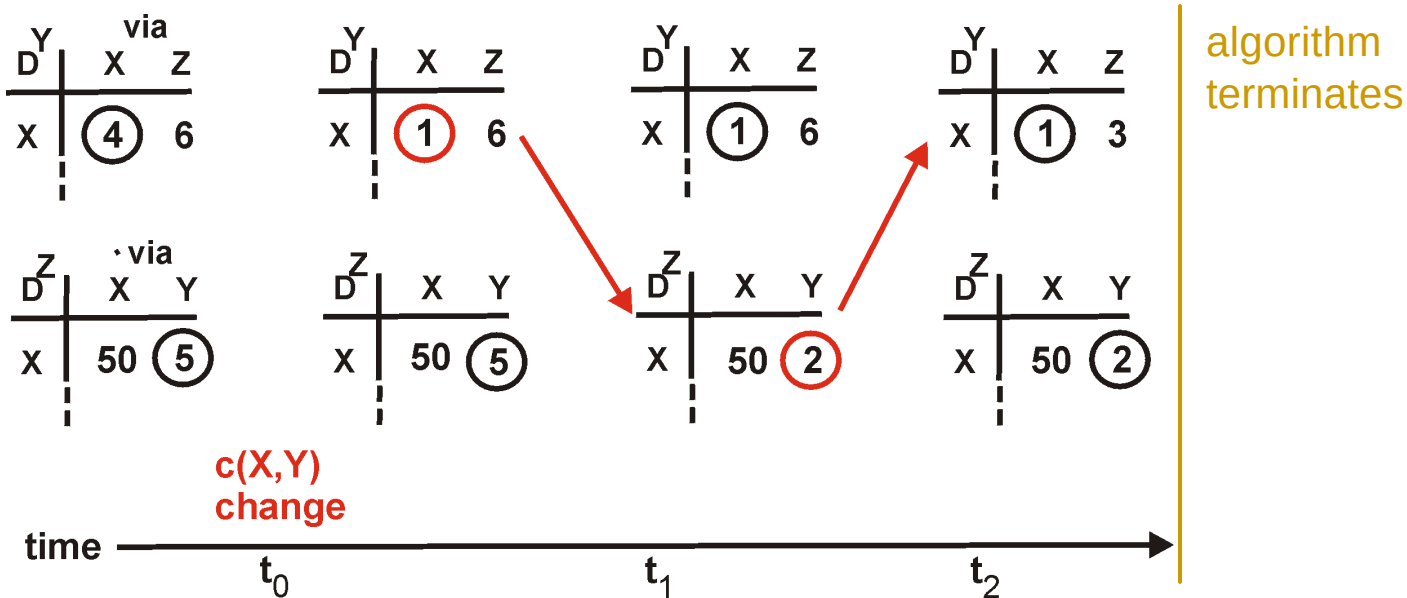
Distance Vector: link cost changes

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)



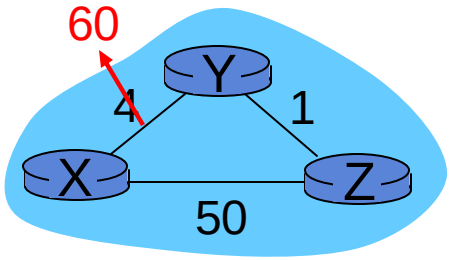
"good news travels fast"

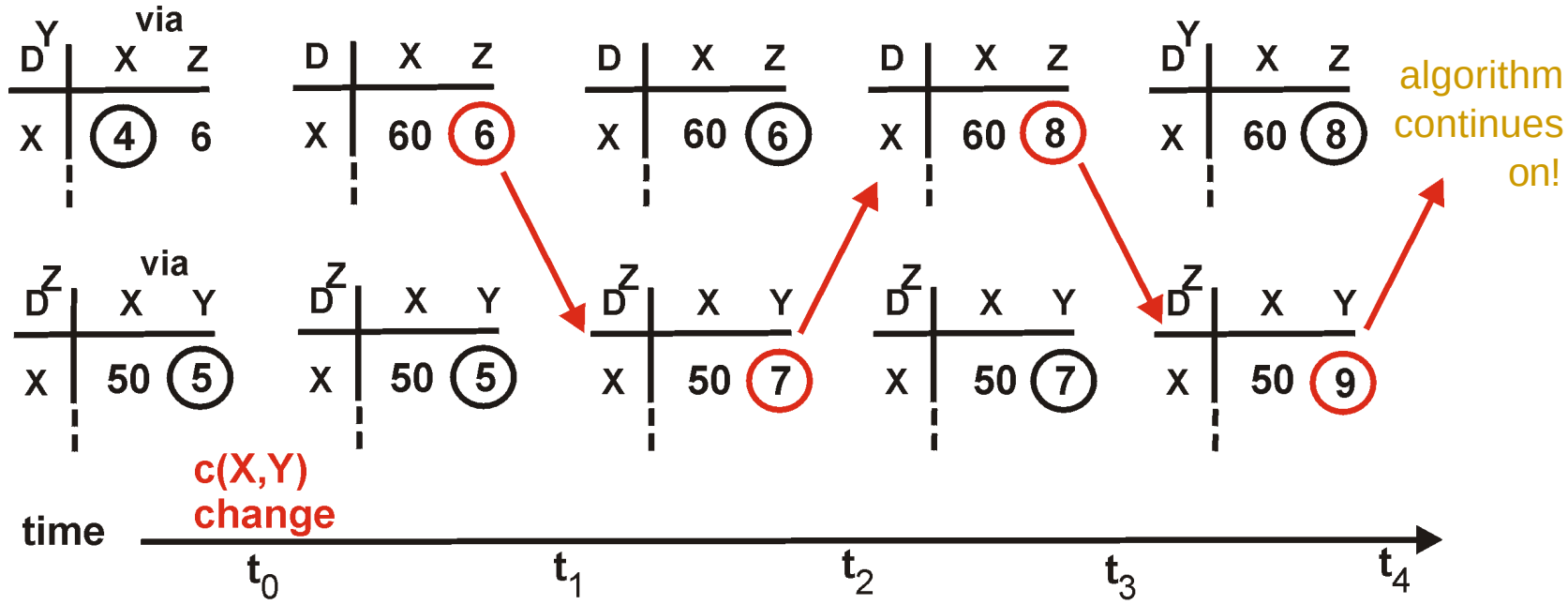


Distance Vector: link cost changes

Link cost changes:

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

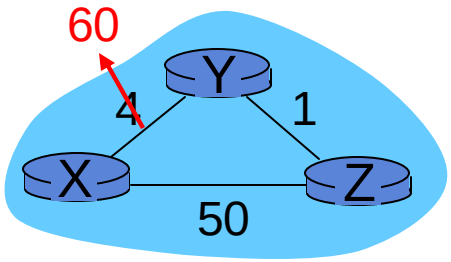


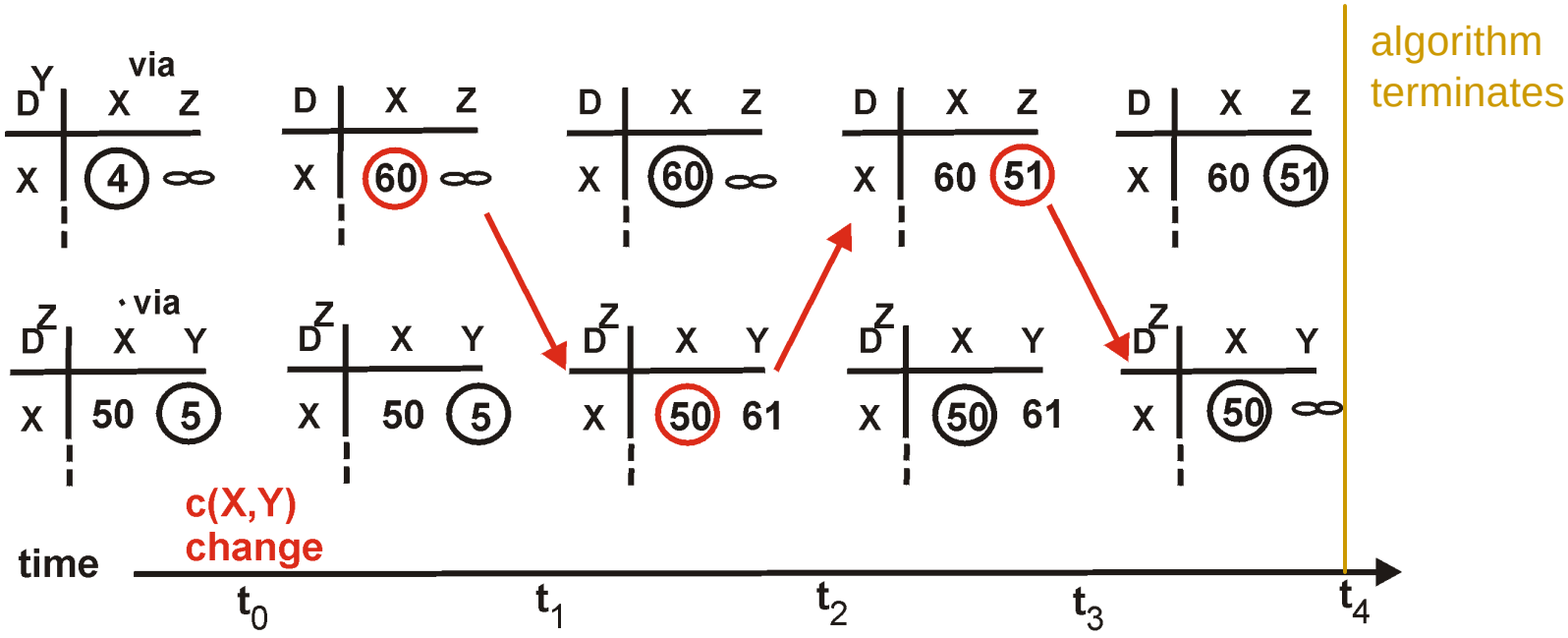


Distance Vector: poisoned reverse

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

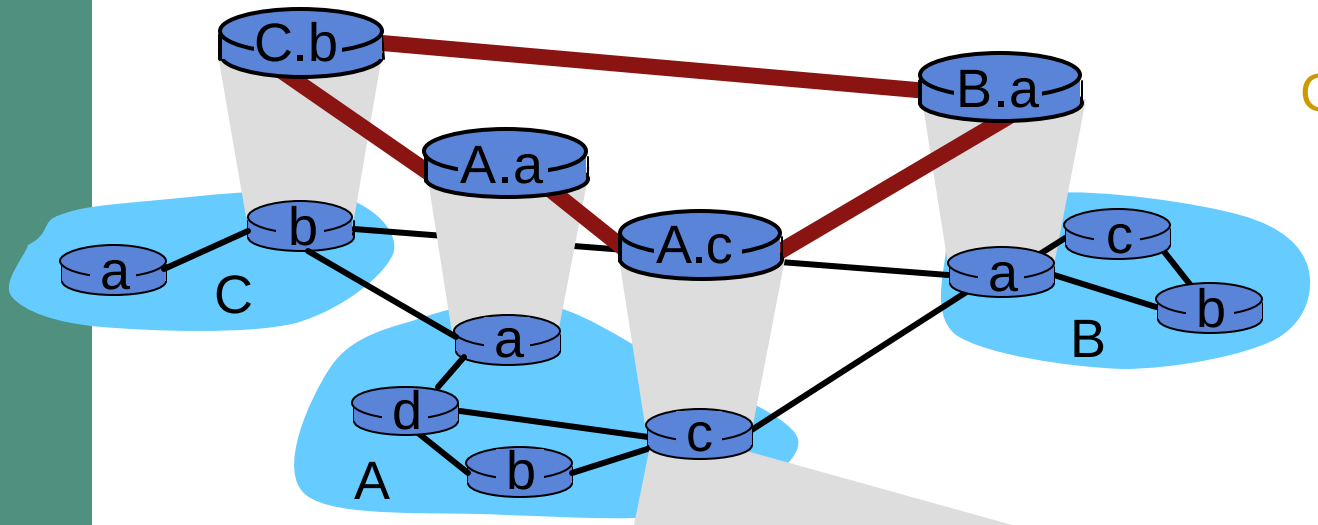
Hierarchical Routing

- 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

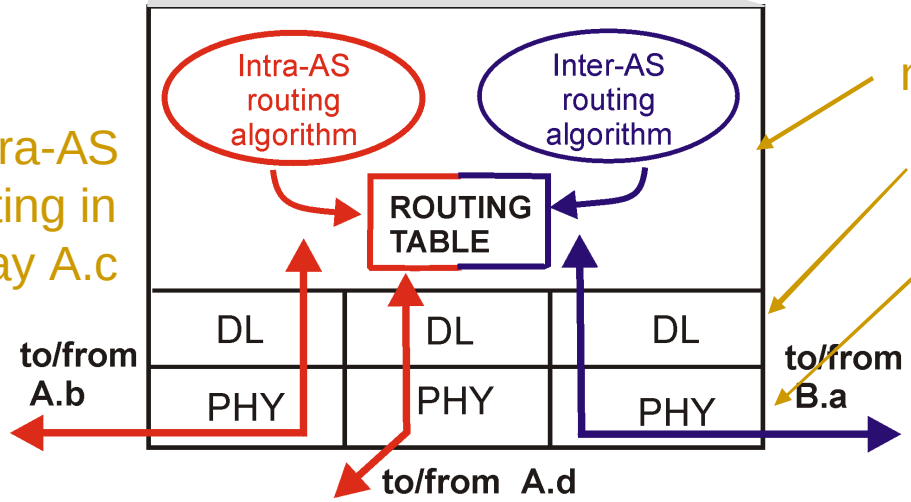
Intra-AS and Inter-AS routing



Gateways:

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

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

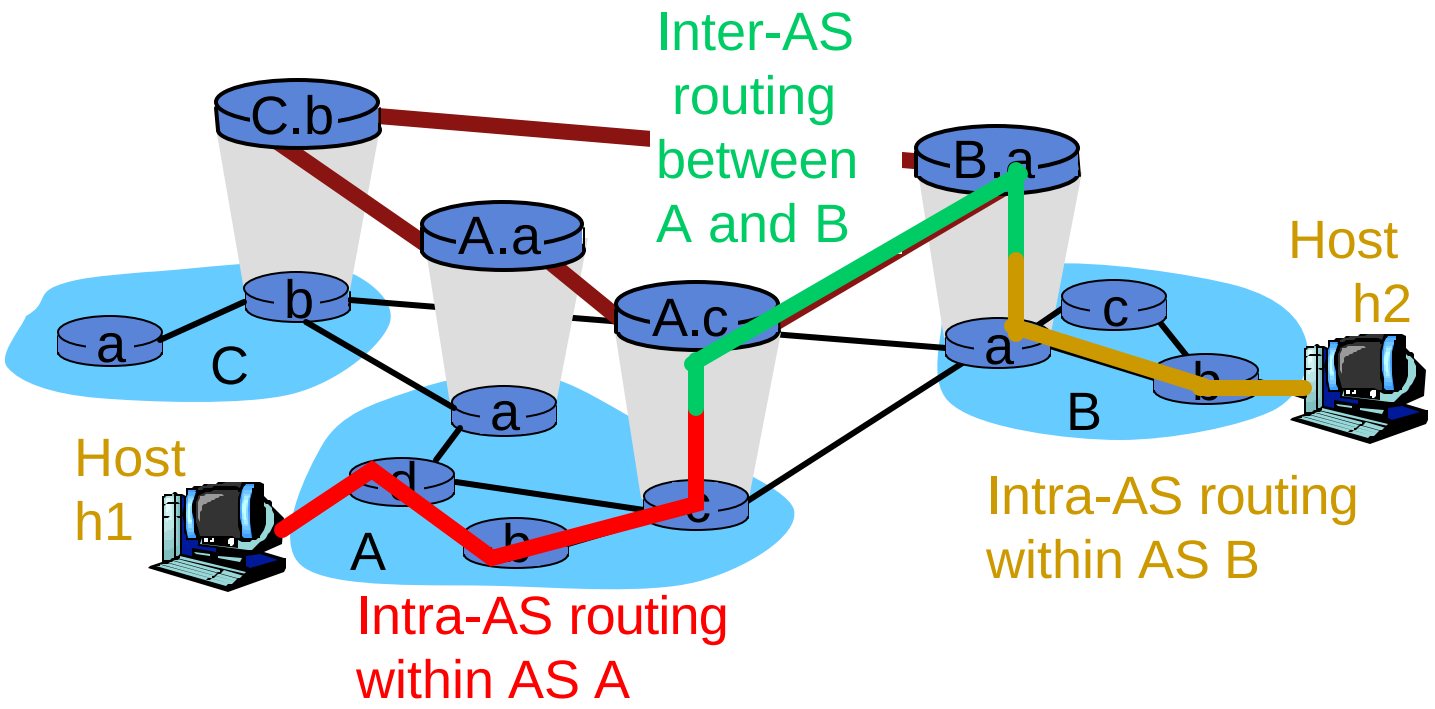


network layer

link layer

physical layer

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

4.5 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

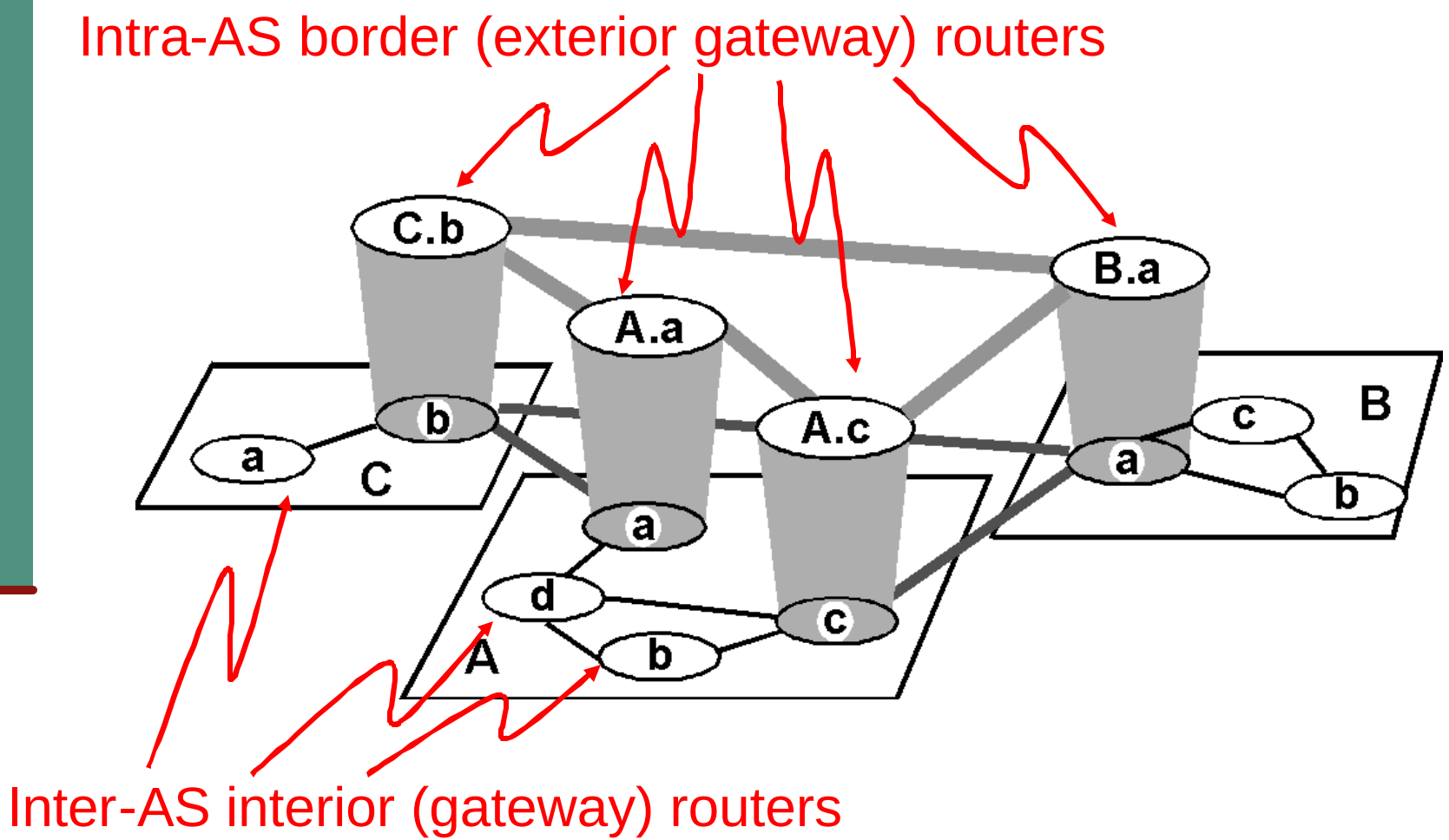
4.9 Mobility

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
  - **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



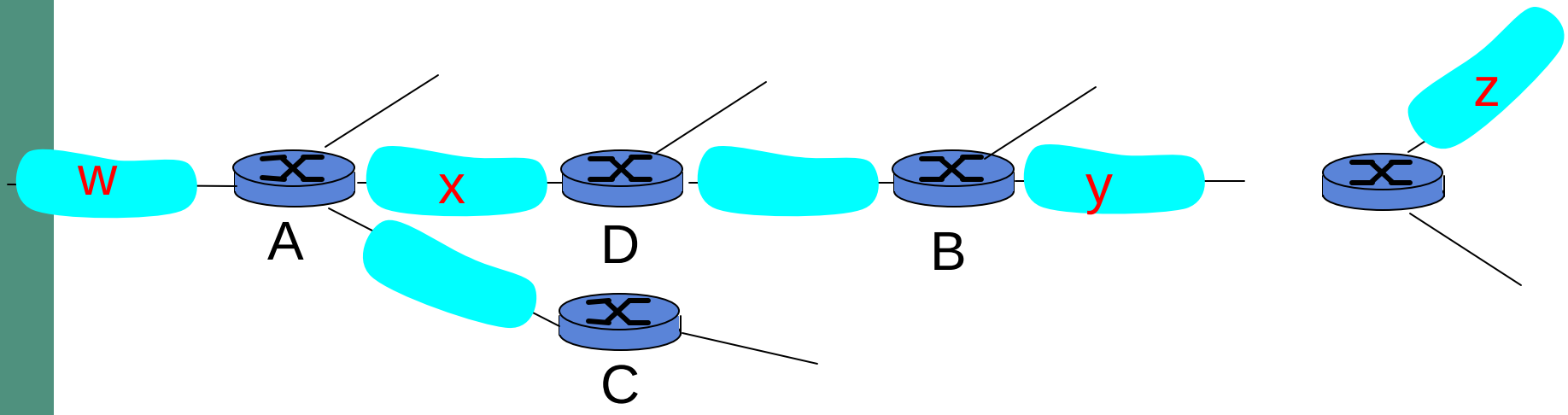
Intra-AS Routing

- 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)
  - *Can you guess why?*
- 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



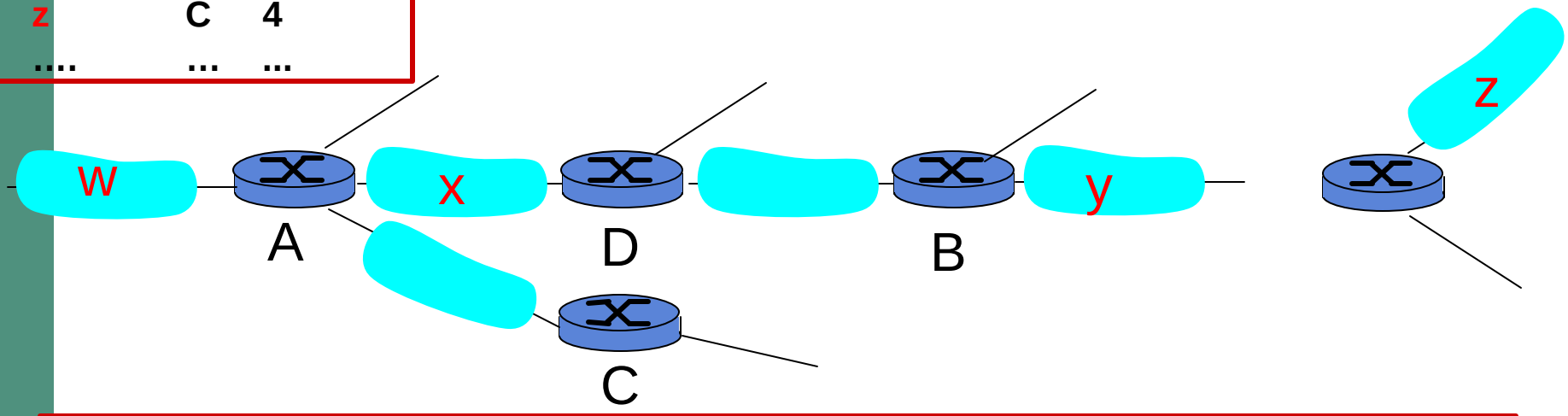
Destination Network	Next Router	Num. of hops to dest.
w	A	2
y	B	2
z	B	7
x	--	1
....	....	....

Routing table in D

RIP: 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

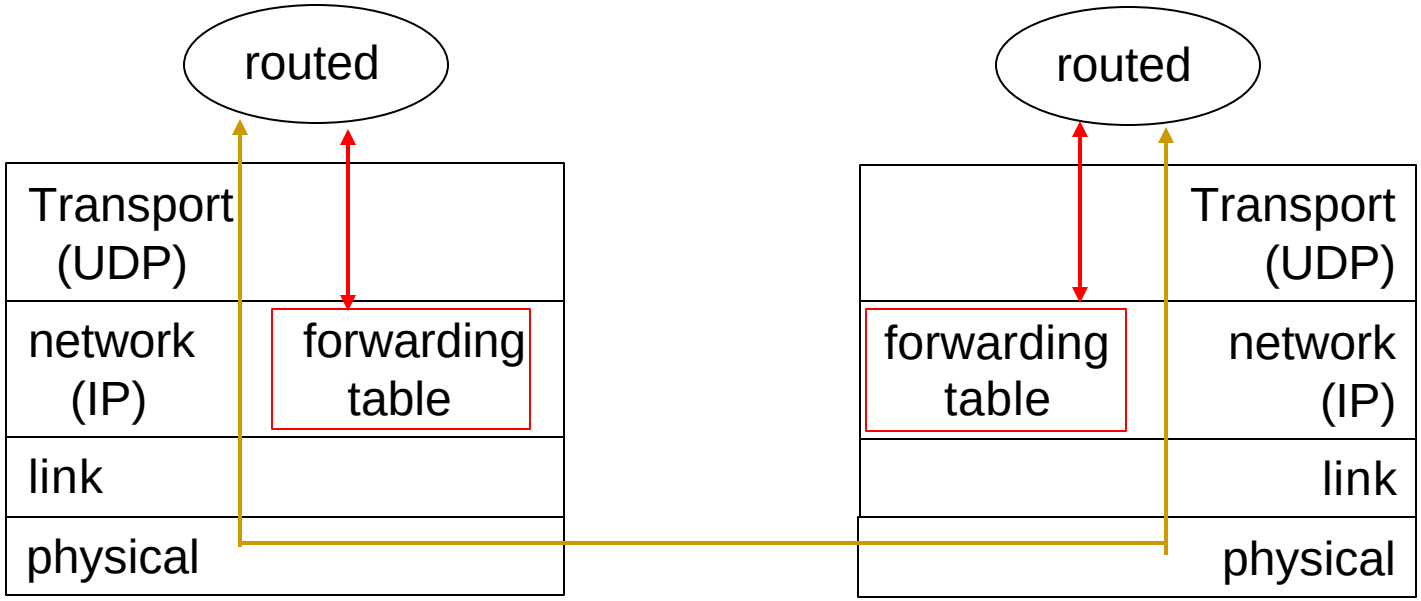
RIP: Link Failure and Recovery

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)

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: *router.cs.ucf.edu*

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)

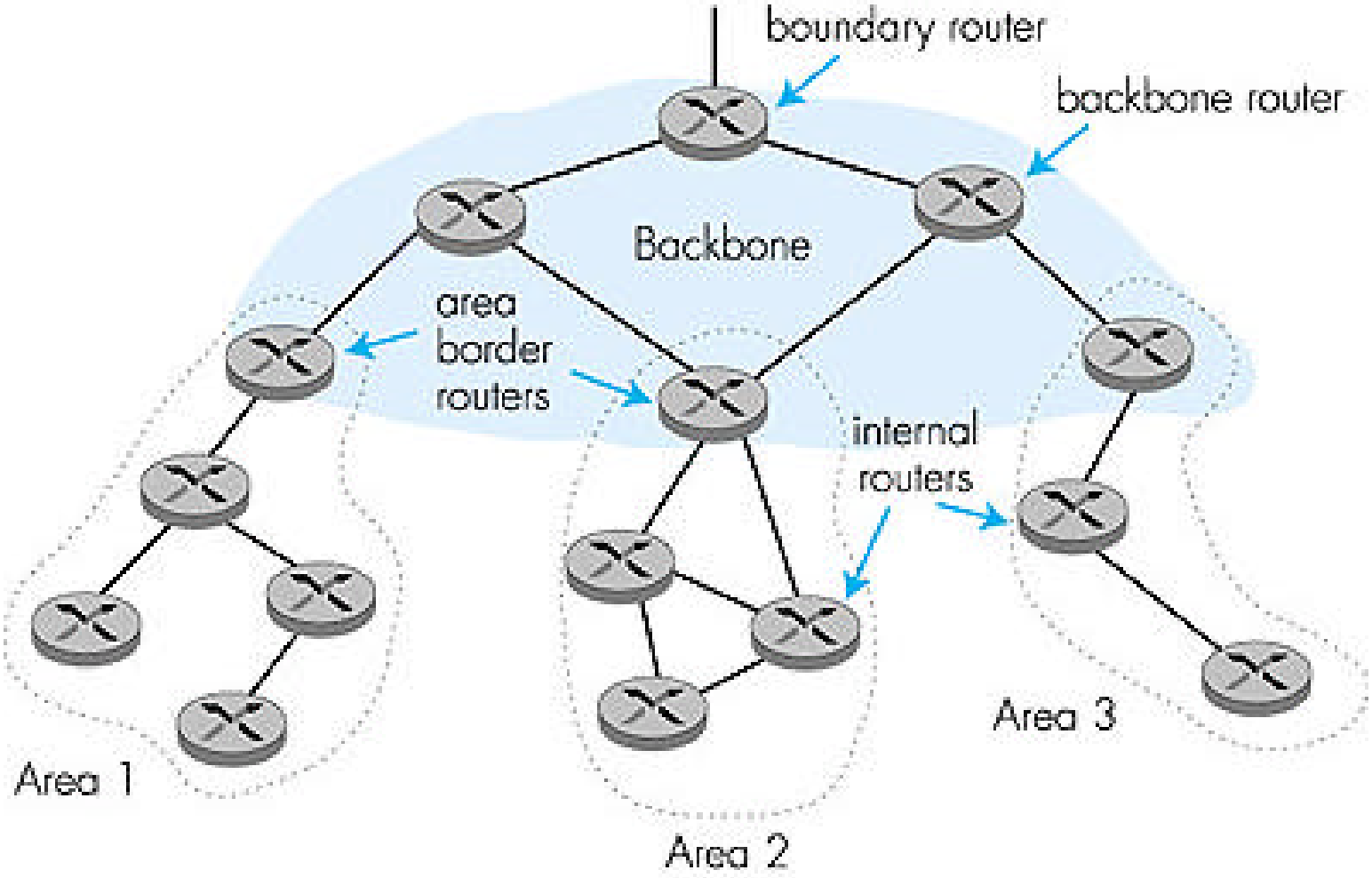
OSPF (Open Shortest Path First)

- “open”: publicly available
- Uses Link State algorithm
  - LS packet dissemination
  - Topology map at each node
  - Route computation using Dijkstra’s algorithm
- OSPF 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 “advanced” features (not in RIP)

- Security: all OSPF messages authenticated (to prevent malicious intrusion)
- Multiple same-cost paths allowed (only one path in RIP)
- 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.

Hierarchical OSPF



Hierarchical OSPF

- Two-level hierarchy: local area and backbone
  - Link-state advertisements only within an area
  - each node has detailed area topology
  - Each node only knows direction (shortest path) to nets in other areas
- Components:
  - **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.

Internet inter-AS routing: BGP

- BGP (Border Gateway Protocol): *the* de facto standard
- **Path Vector** protocol:
  - similar to Distance Vector protocol
  - each Border Gateway broadcast to neighbors (BGP peers) *entire path* (i.e., sequence of AS's) to destination
  - E.g., Gateway X may send its path to dest. Z:

    Path (X,Z) = X,Y1,Y2,Y3,…,Z

  - An AS is identified by a unique AS number
  - BGP routes to networks (ASs), not individual hosts

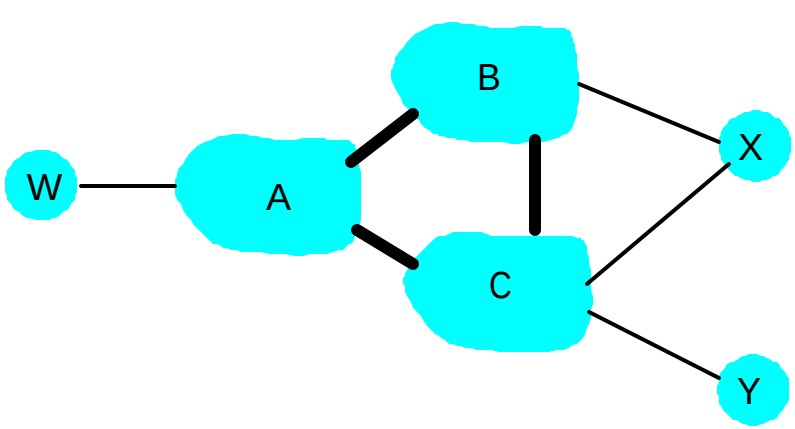
Internet inter-AS routing: BGP

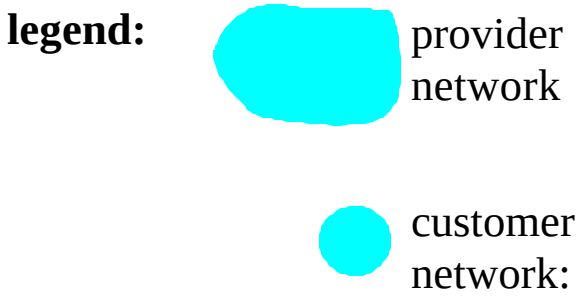
Suppose: gateway X send its path to peer gateway W

- 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 route advertisements to peers:
  - e.g., don't want to route traffic to Z -> don't advertise any routes to Z

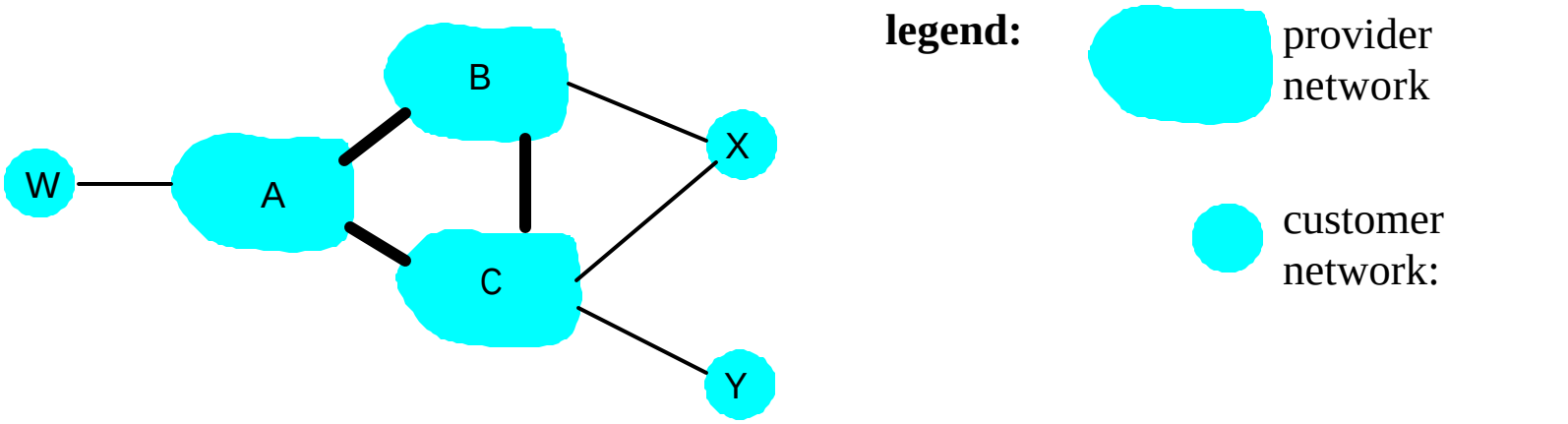
BGP: controlling who routes to you





- 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 because 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

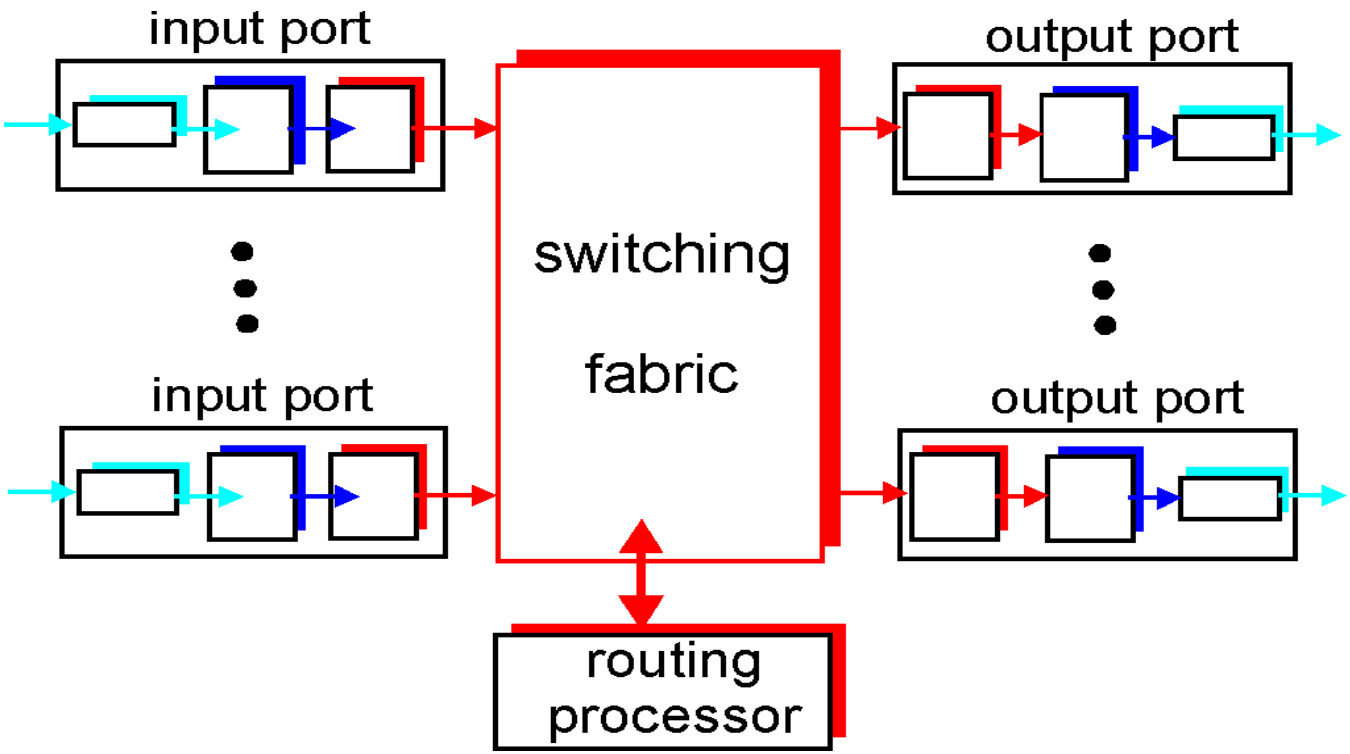
Performance:

- Intra-AS: can focus on performance
- Inter-AS: policy may dominate over performance

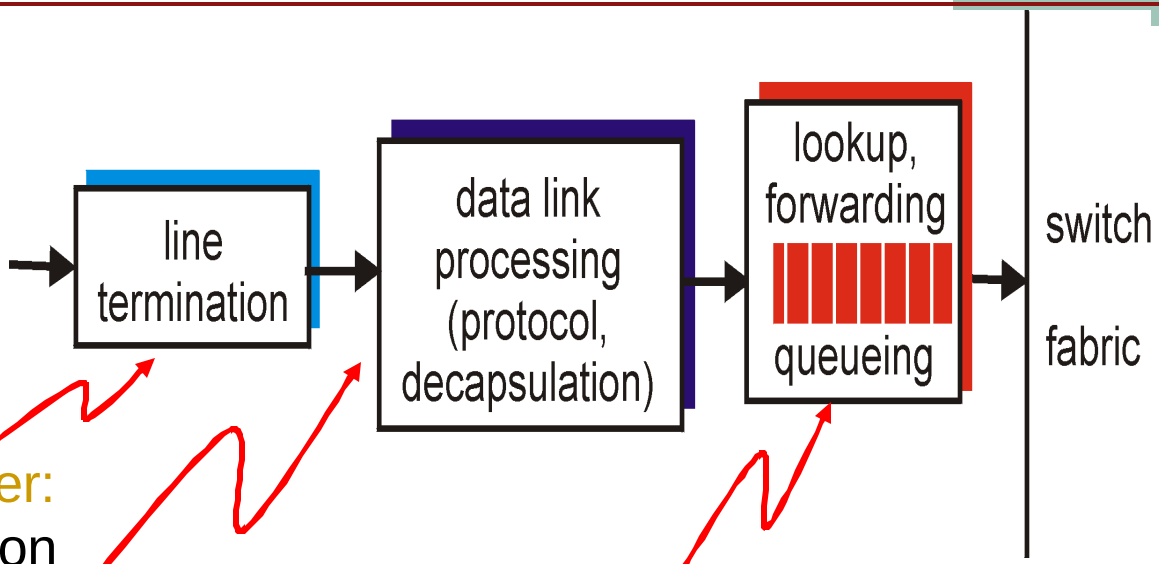
Router Architecture Overview

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

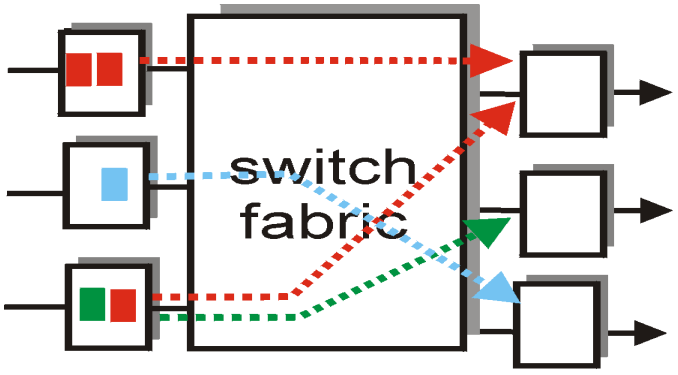
Data link layer:
e.g., Ethernet
see chapter 5

Decentralized switching:

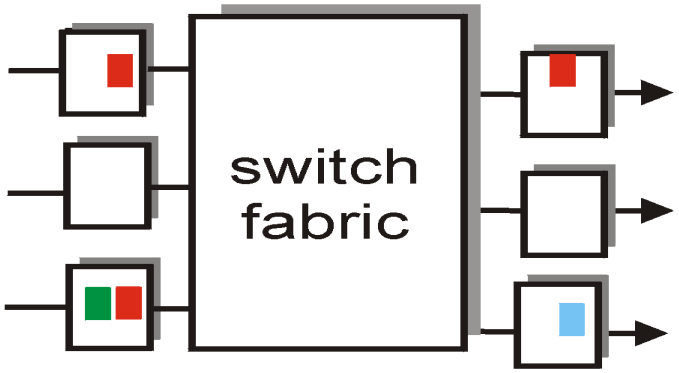
- given datagram dest., lookup output port using routing table in input port memory
- goal: complete input port processing at 'line speed'
- queuing: if datagrams arrive faster than forwarding rate into switch fabric

Input Port Queuing

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

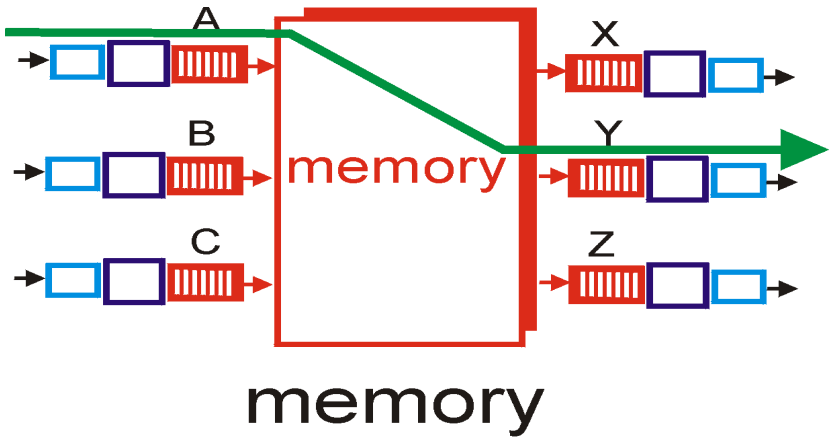


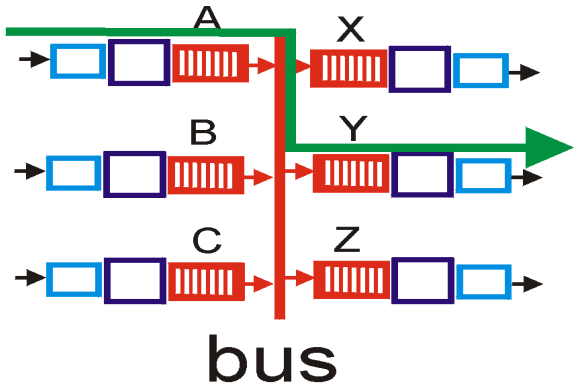
output port contention at time t - only one red packet can be transferred

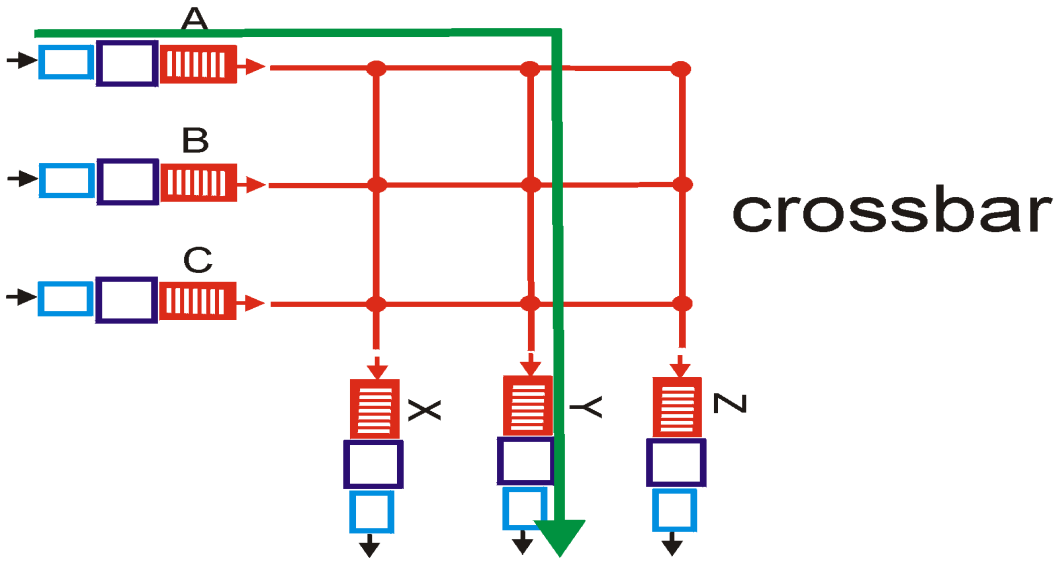


green packet experiences HOL blocking

Three types of switching fabrics



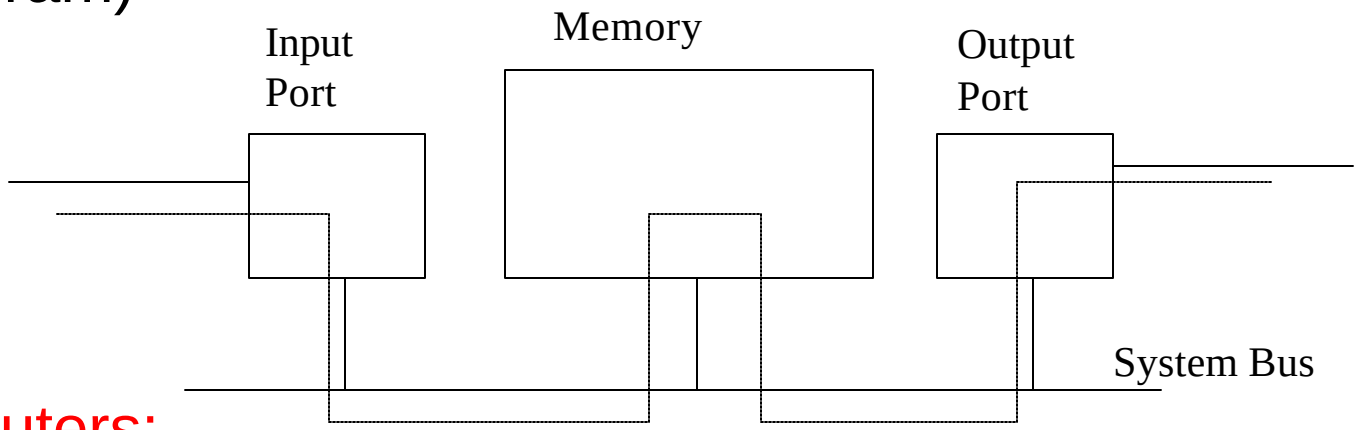




Switching Via Memory

First generation routers:

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

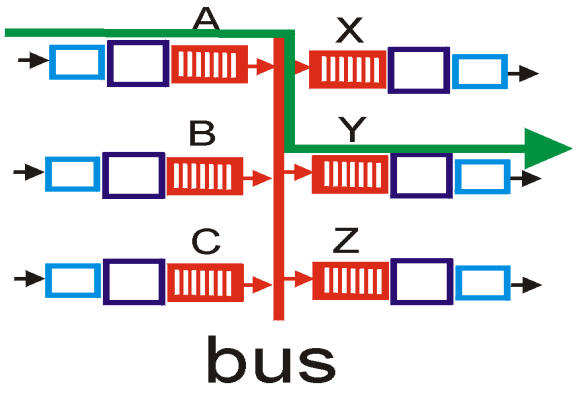


Modern routers:

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

Switching Via a Bus

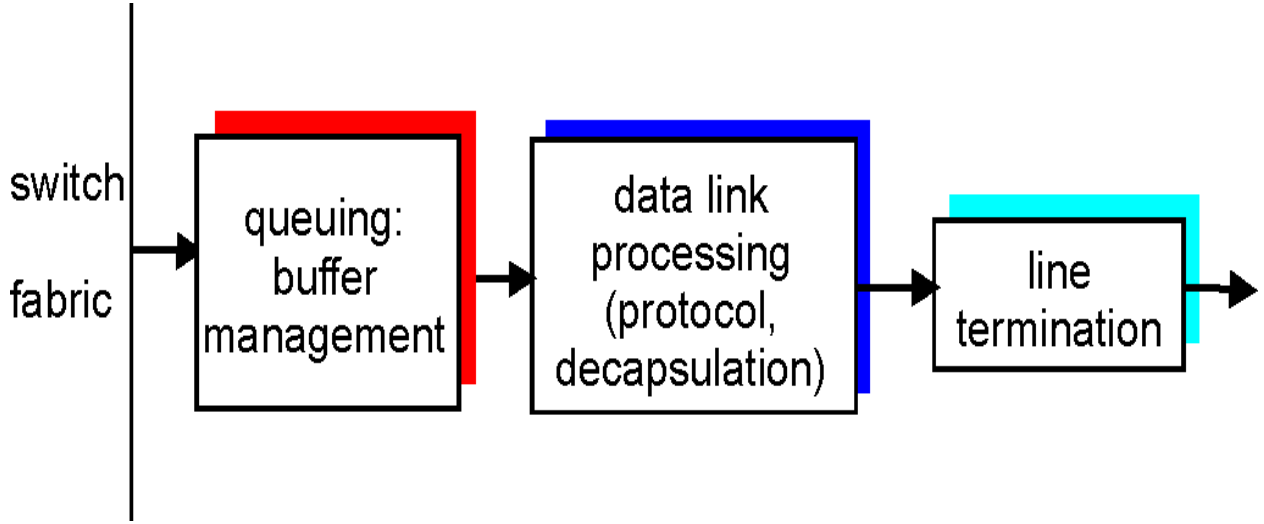
- datagram from input port memory to output port memory via a shared bus
- bus contention: switching speed limited by bus bandwidth
- 1 Gbps bus, Cisco 1900: sufficient speed for access and enterprise routers (not regional or backbone)



Switching Via An Interconnection Network

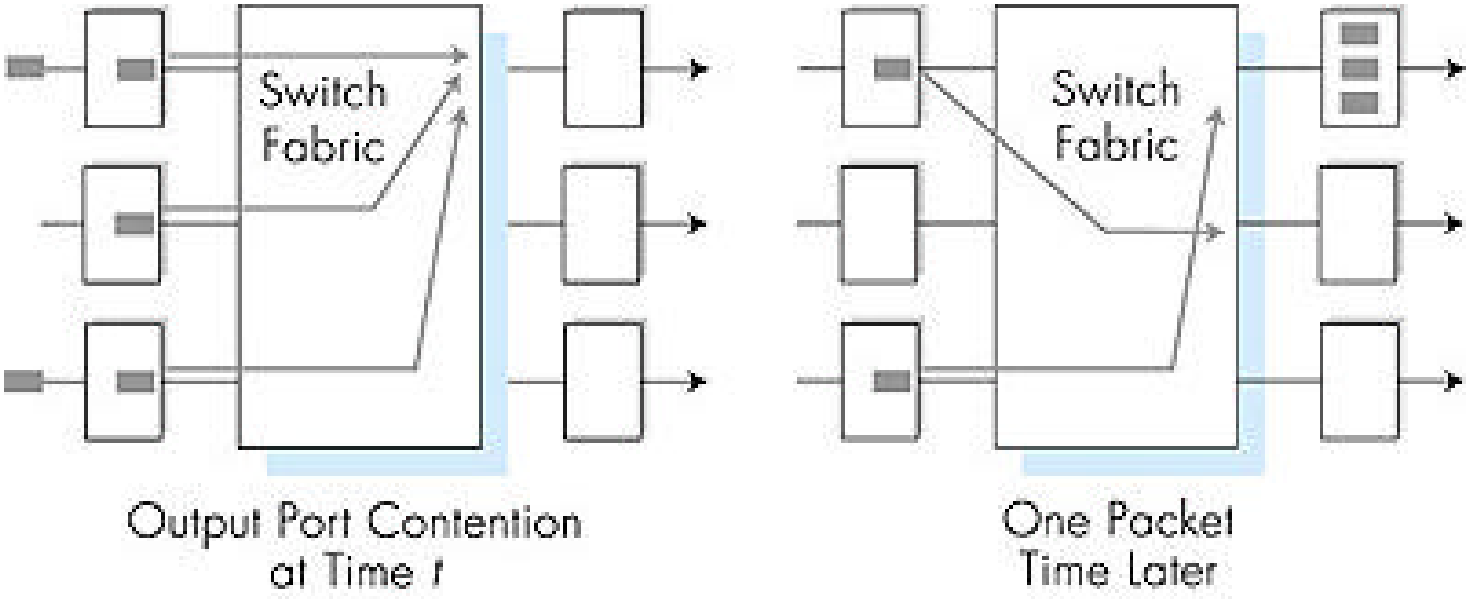
- overcome bus bandwidth limitations
- Banyan networks, other interconnection nets initially developed to connect processors in multiprocessor
- Advanced design: fragmenting datagram into fixed length cells, switch cells through the fabric.
- Cisco 12000: switches Gbps through the interconnection network

Output Ports



- *Buffering* required when datagrams arrive from fabric faster than the transmission rate
- *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!*

IPv6

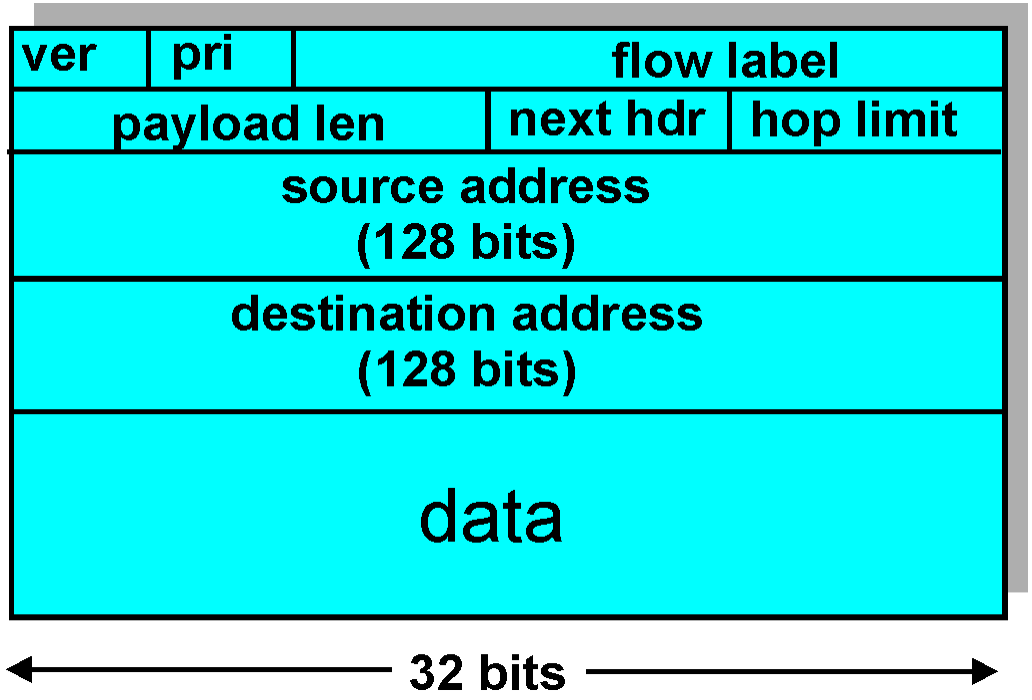
- Initial motivation: 32-bit address space completely allocated by 2008.
- Additional motivation:
  - header format helps speed processing/forwarding
  - header changes to facilitate QoS
  - new “anycast” address: route to “best” of several replicated servers
- IPv6 datagram format:
  - fixed-length 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



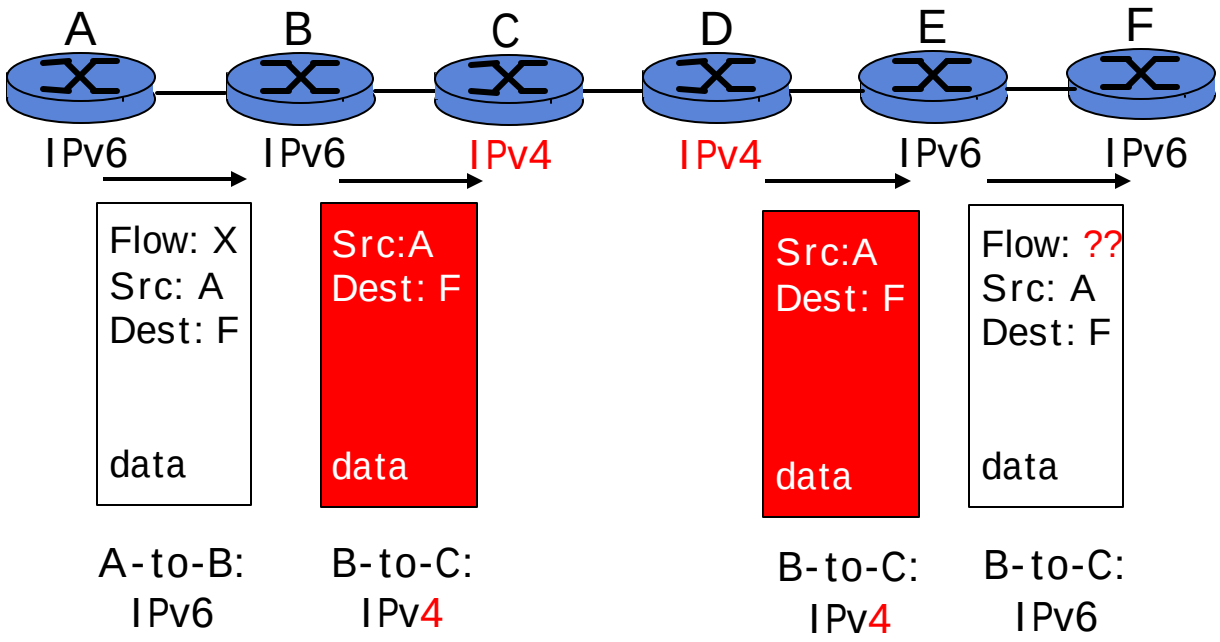
Other Changes from IPv4

- *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

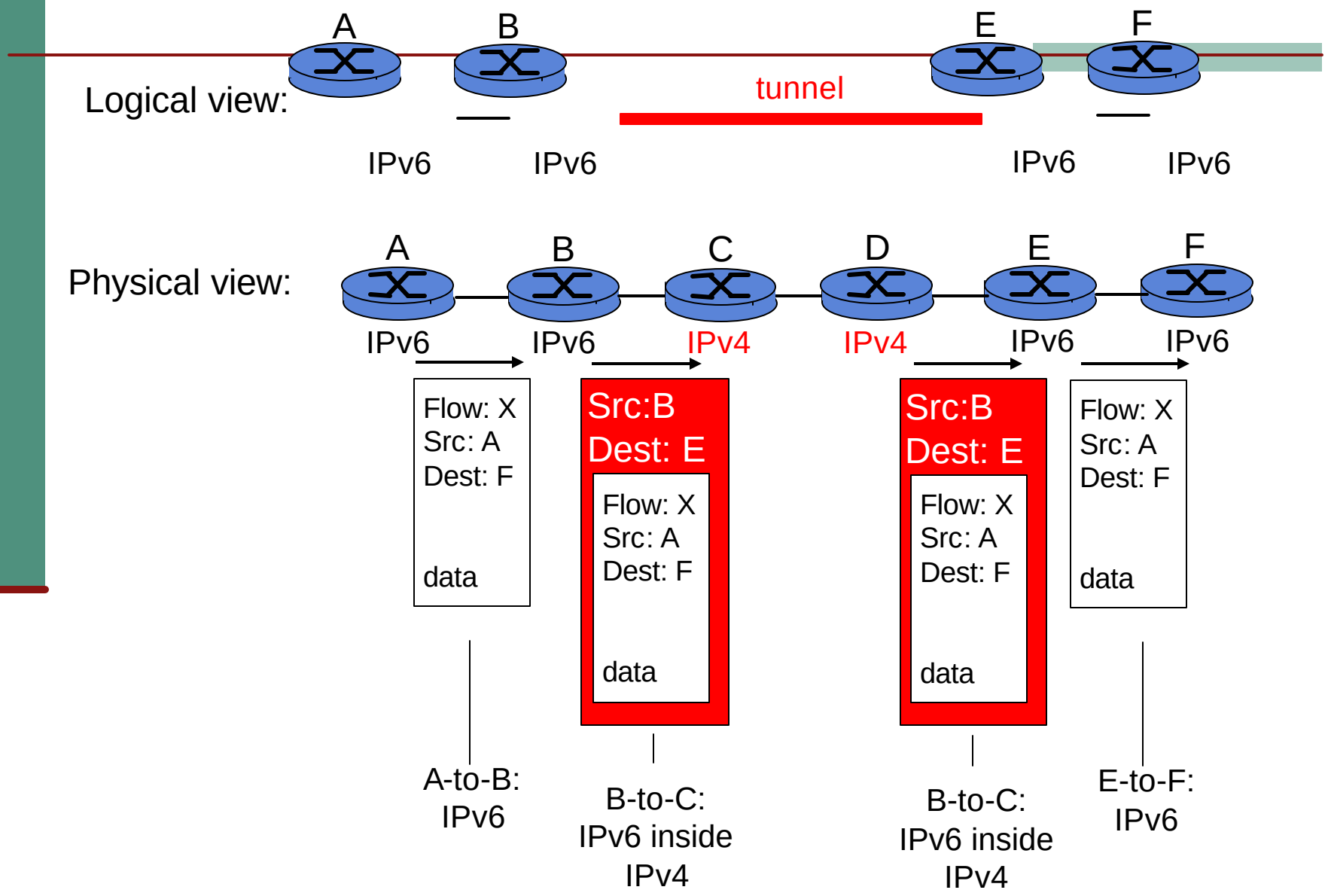
Transition From IPv4 To IPv6

- 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
E
F
tunnel
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