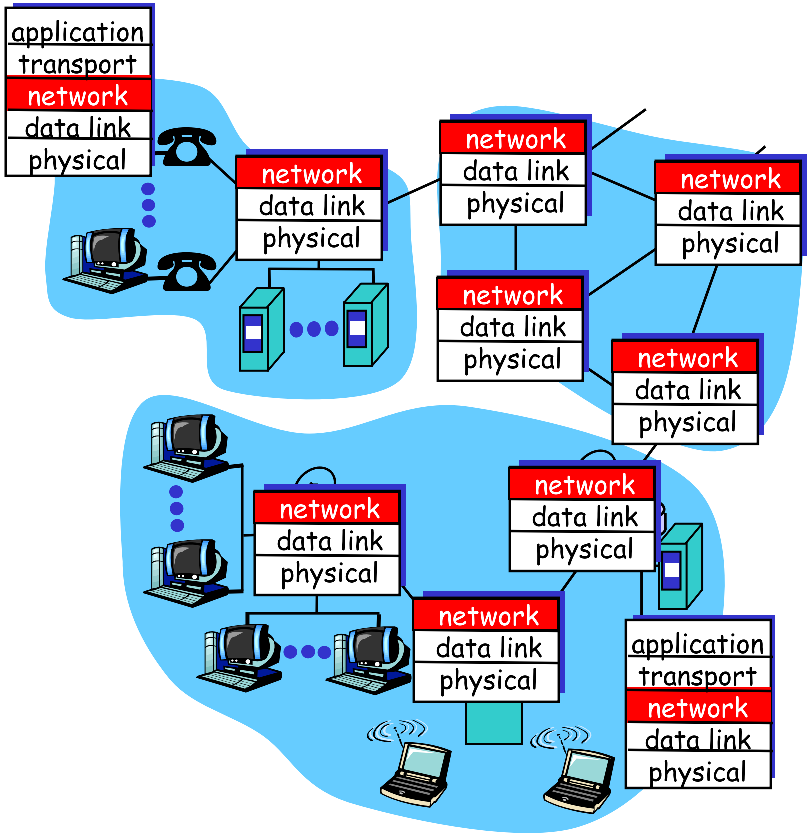
Chapter 4: Network Layer

Chapter goals:

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

Network layer

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



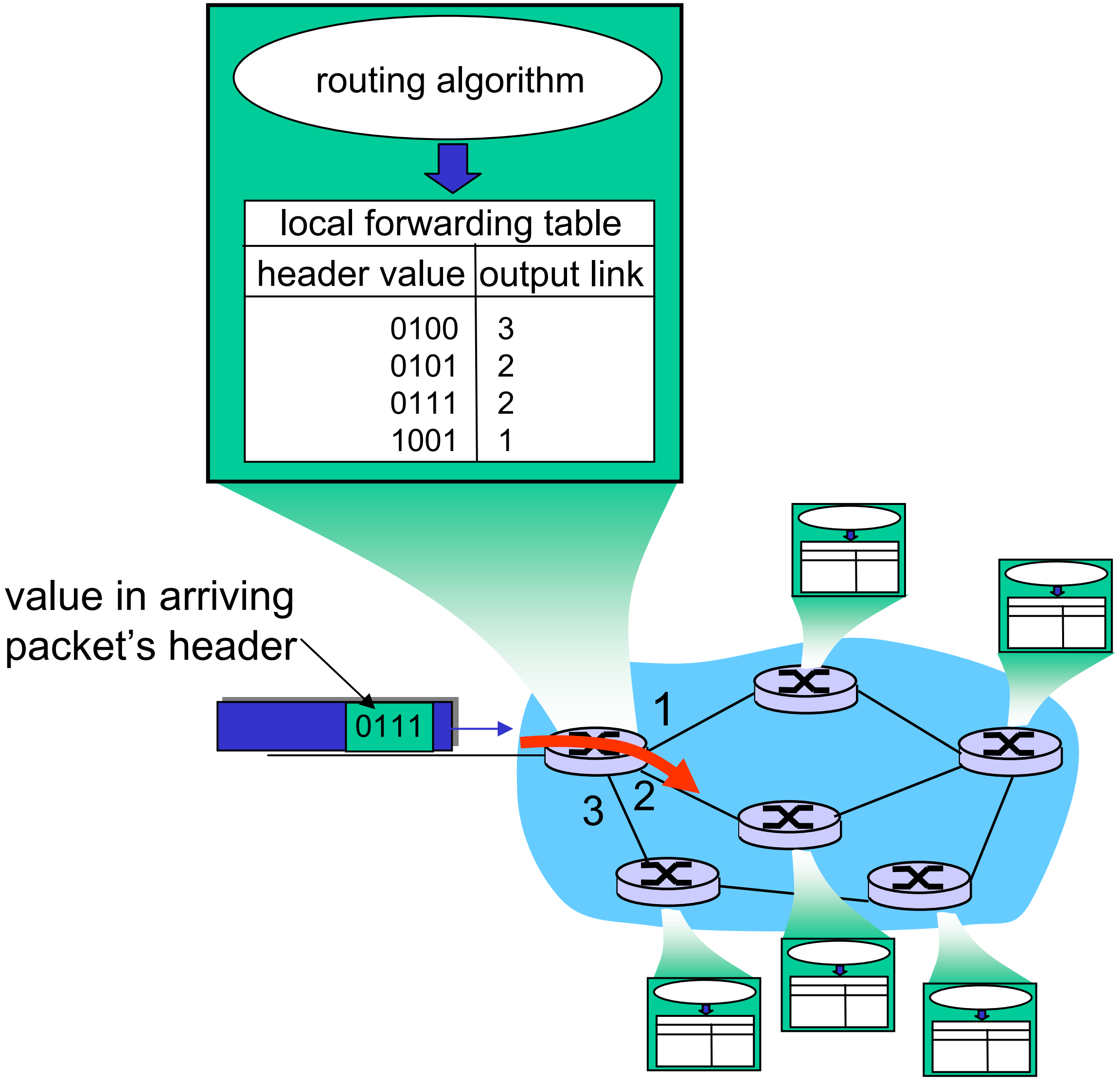
Key Network-Layer Functions

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

analogy:

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

Interplay between routing and forwarding



routing algorithm

local forwarding table

header value	output link
0100	3
0101	2
0111	2
1001	1

value in arriving packet's header

Connection setup

- Important function in *some* network architectures:
  - ATM, frame relay, X.25
- Before datagrams flow, two hosts and intervening routers establish virtual connection
  - Routers get involved
- Network and transport layer connection service:
  - Network: between two hosts
  - Transport: between two processes

Network layer connection and connection-less service

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

Virtual circuits

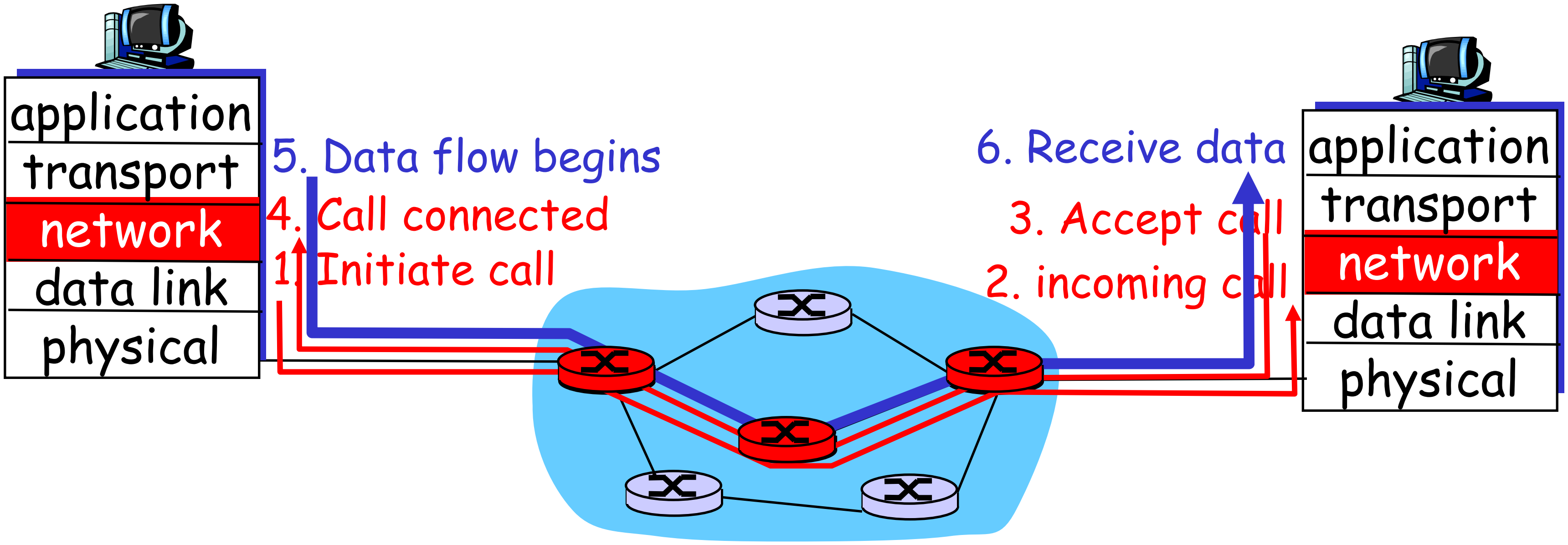
"source-to-dest path behaves much like telephone circuit"

- performance-wise
- network actions along source-to-dest path

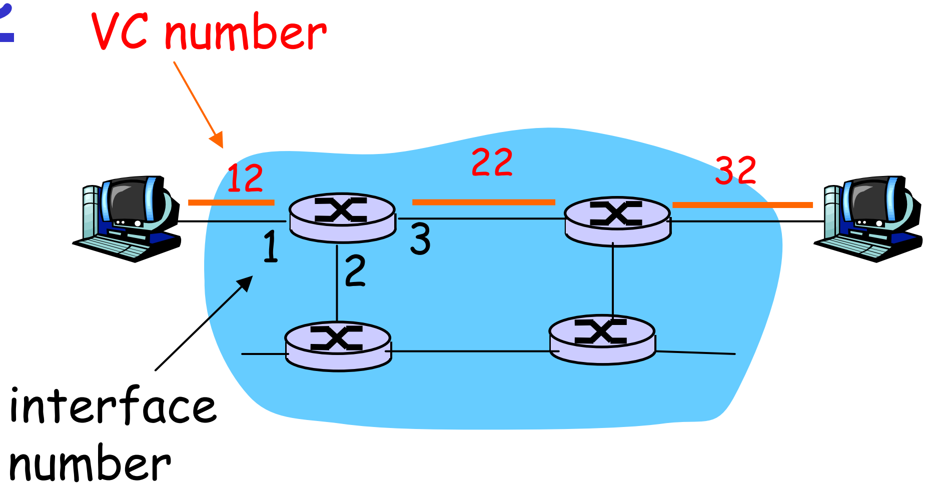
- call setup, teardown for each call *before* data can flow
- each packet carries VC identifier (not destination host address)
- *every* router on source-dest path maintains "state" for each passing connection
- link, router resources (bandwidth, buffers) may be *allocated* to VC

Virtual circuits: signaling protocols

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



Forwarding table



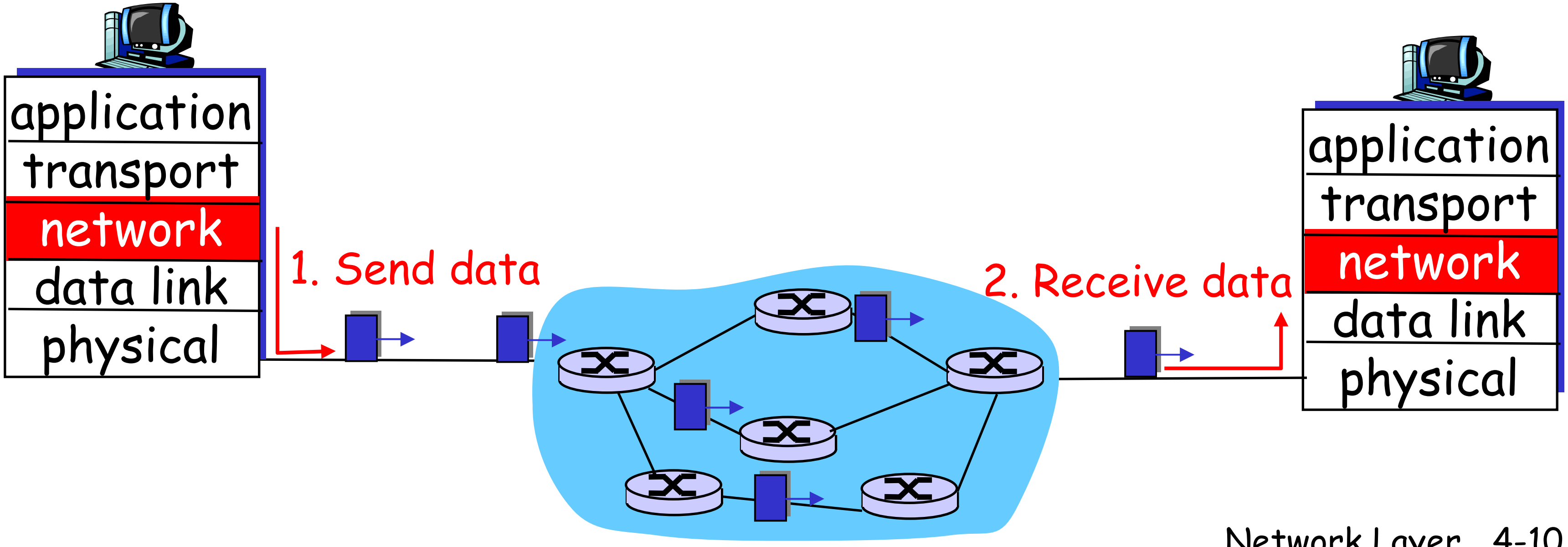
Forwarding table in northwest router:

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

Routers maintain connection state information!

Datagram networks

- 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



Forwarding table

4 billion possible entries

Destination Address Range	Link Interface
11001000 00010111 00010000 00000000 through 11001000 00010111 00010111 11111111	0
11001000 00010111 00011000 00000000 through 11001000 00010111 00011000 11111111	1
11001000 00010111 00011001 00000000 through 11001000 00010111 00011111 11111111	2
otherwise	3

Longest prefix matching

Prefix	Link Interface
11001000 00010111 00010	0
11001000 00010111 00011000	1
11001000 00010111 00011	2
otherwise	3

Examples

DA: 11001000 00010111 00010110 10100001   Which interface?

DA: 11001000 00010111 00011000 10101010   Which interface?

Datagram or VC network: why?

Internet

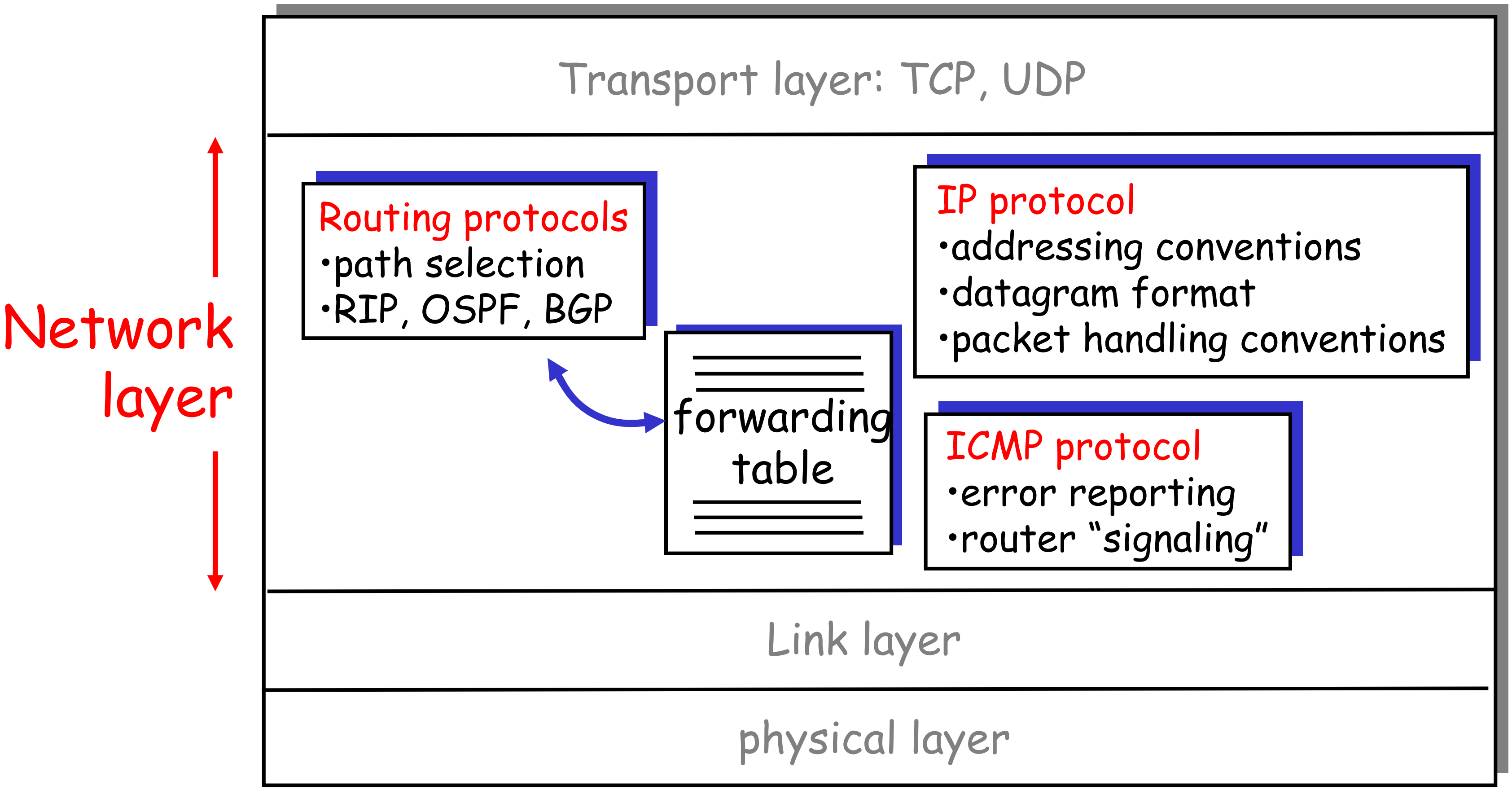
- data exchange among computers
  - "elastic" service, no strict timing req.
- "smart" end systems (computers)
  - can adapt, perform control, error recovery
  - simple inside network, complexity at "edge"
- many link types
  - different characteristics
  - uniform service difficult

ATM

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

The Internet Network layer

Host, router network layer functions:



Transport layer: TCP, UDP

Network layer

Routing protocols
- path selection
- RIP, OSPF, BGP

forwarding table

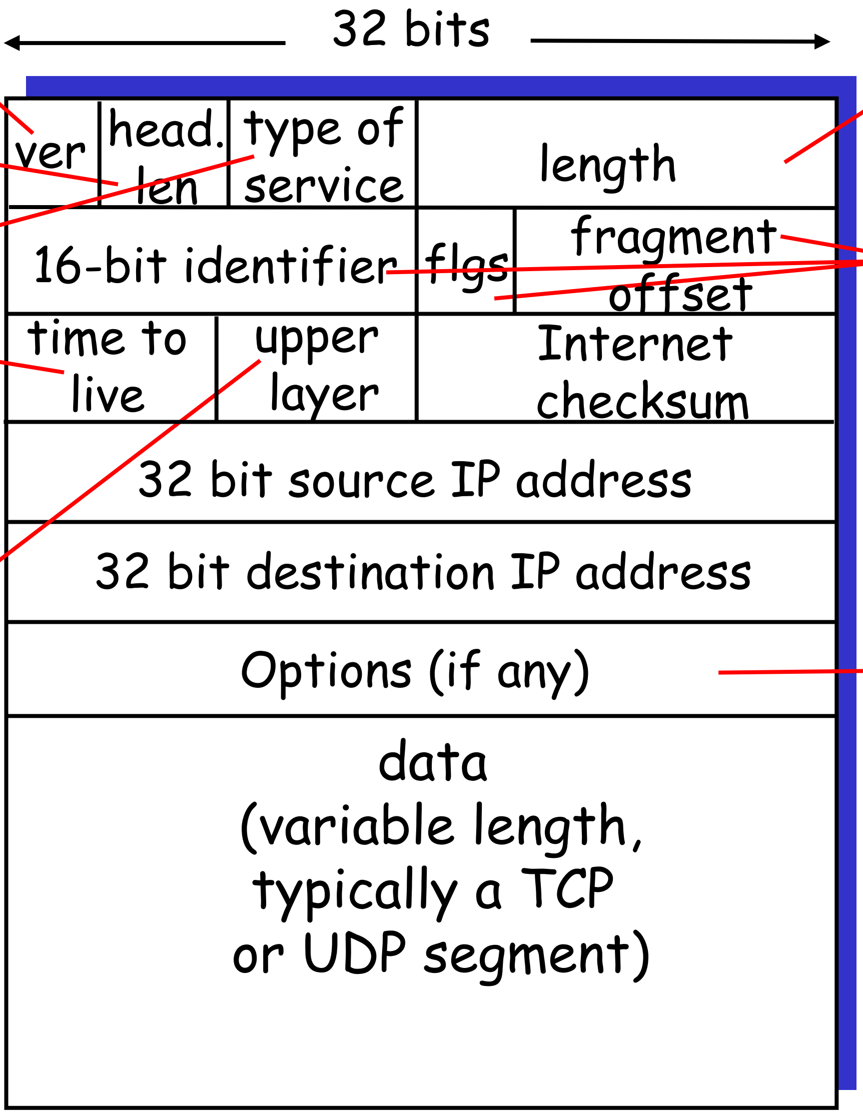
IP protocol
- addressing conventions
- datagram format
- packet handling conventions

ICMP protocol
- error reporting
- router "signaling"

Link layer

physical layer

IP datagram format



32 bits

ver | head. len | type of service | length

16-bit identifier | flgs | fragment offset

time to live | upper layer | Internet checksum

32 bit source IP address

32 bit destination IP address

Options (if any)

data (variable length, typically a TCP or UDP segment)

IP protocol version number

header length (bytes)

“type” of data

max number remaining hops (decremented at each router)

upper layer protocol to deliver payload to

total datagram length (bytes)

for fragmentation/ reassembly

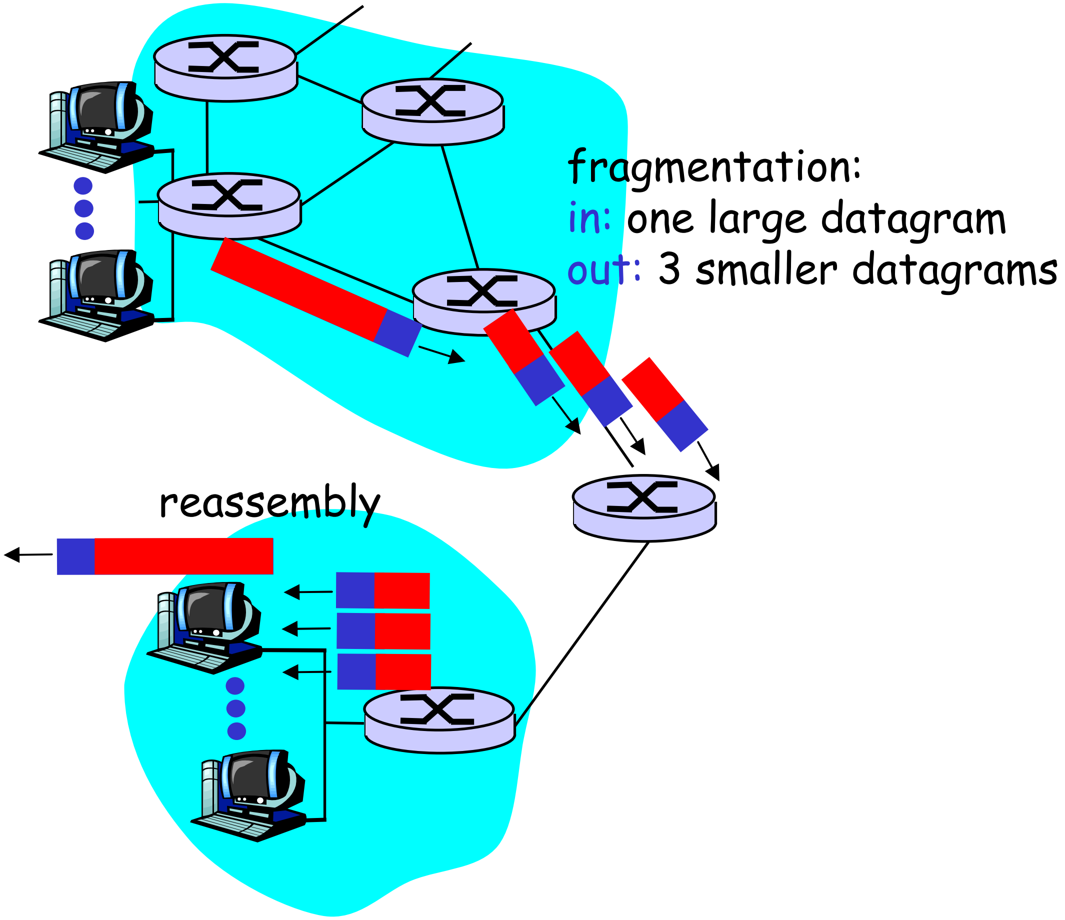
E.g. timestamp, record route taken, specify list of routers to visit.

how much overhead with TCP?

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

IP Fragmentation & Reassembly

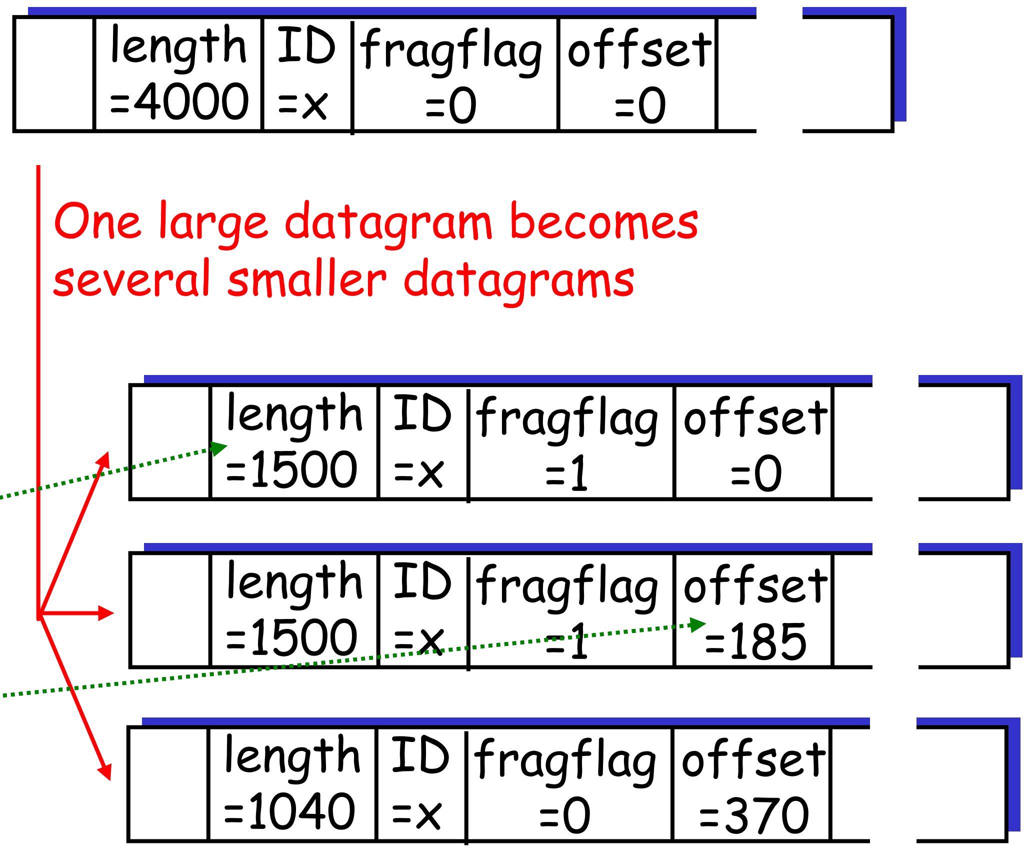
- network links have MTU (max.transfer size) - largest possible link-level frame.
  - different link types, different MTUs
- large IP datagram divided ("fragmented") within net
  - one datagram becomes several datagrams
  - "reassembled" only at final destination
  - IP header bits used to identify,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



IP Fragmentation and Reassembly

Example

- 4000 byte datagram
- MTU = 1500 bytes



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

One large datagram becomes several smaller datagrams

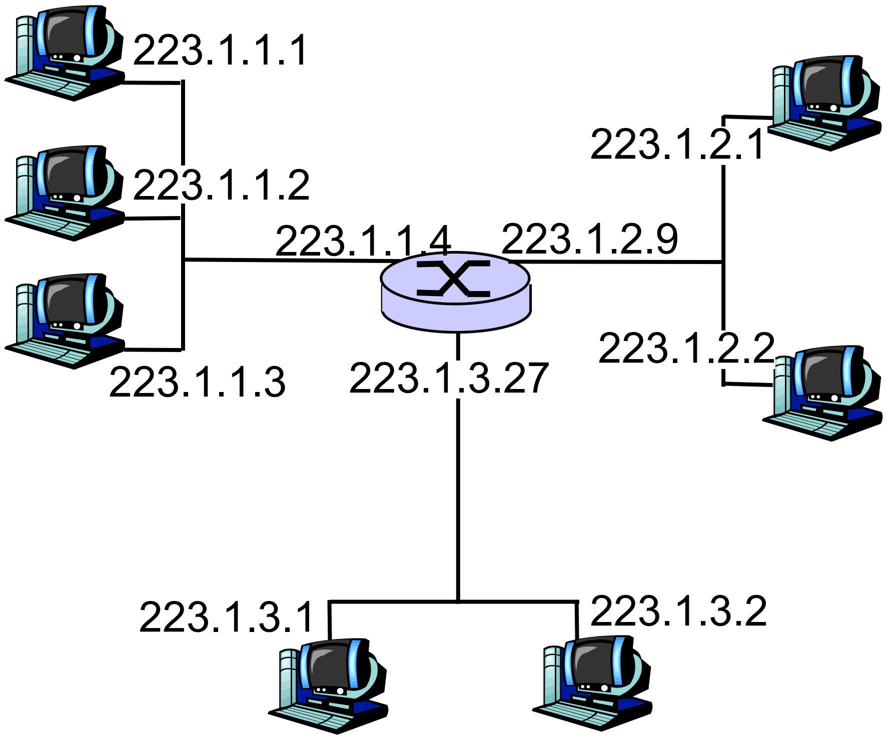
length	ID	fragflag	offset
=1500	=x	=1	=0
=1500	=x	=1	=185
=1040	=x	=0	=370

1480 bytes in data field

offset = 1480/8

IP Addressing: introduction

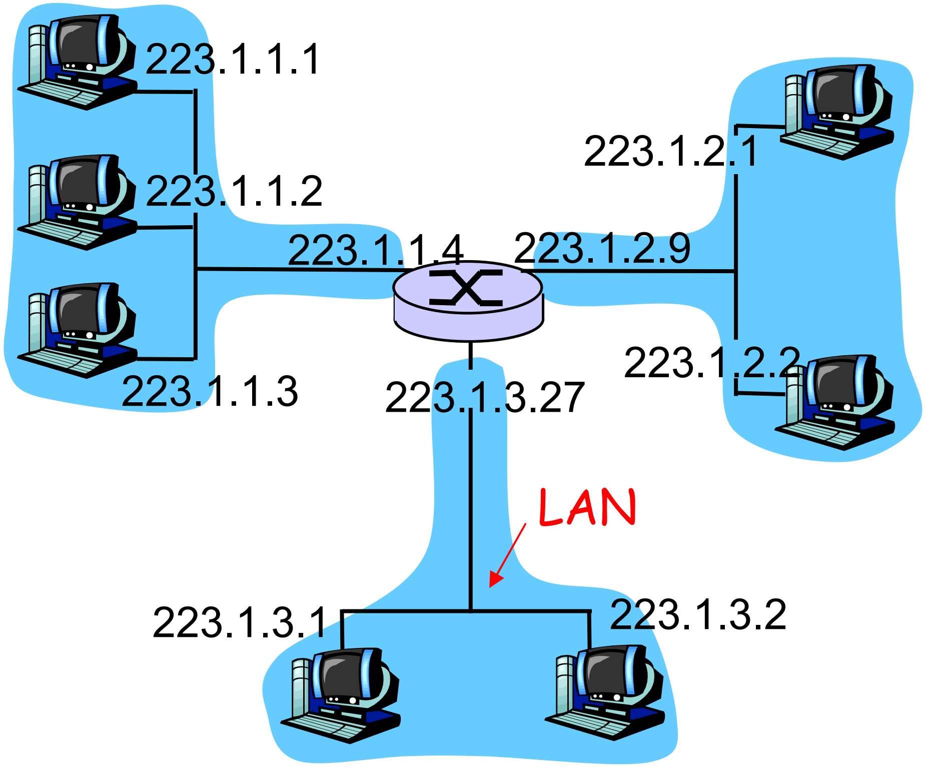
- IP address: 32-bit identifier for host, router *interface*
- *interface:* connection between host/router and physical link
    - router's typically have multiple interfaces
    - host may have multiple interfaces
    - IP addresses associated with each interface



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

Subnets

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

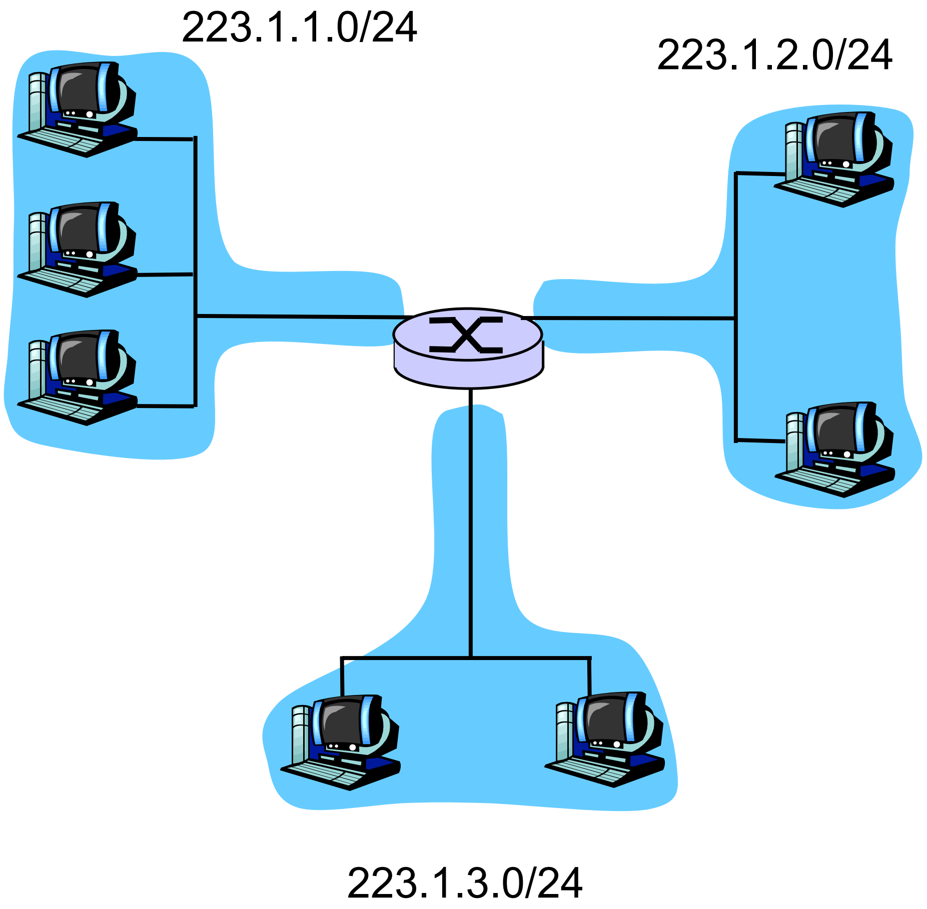


network consisting of 3 subnets

Subnets

Recipe

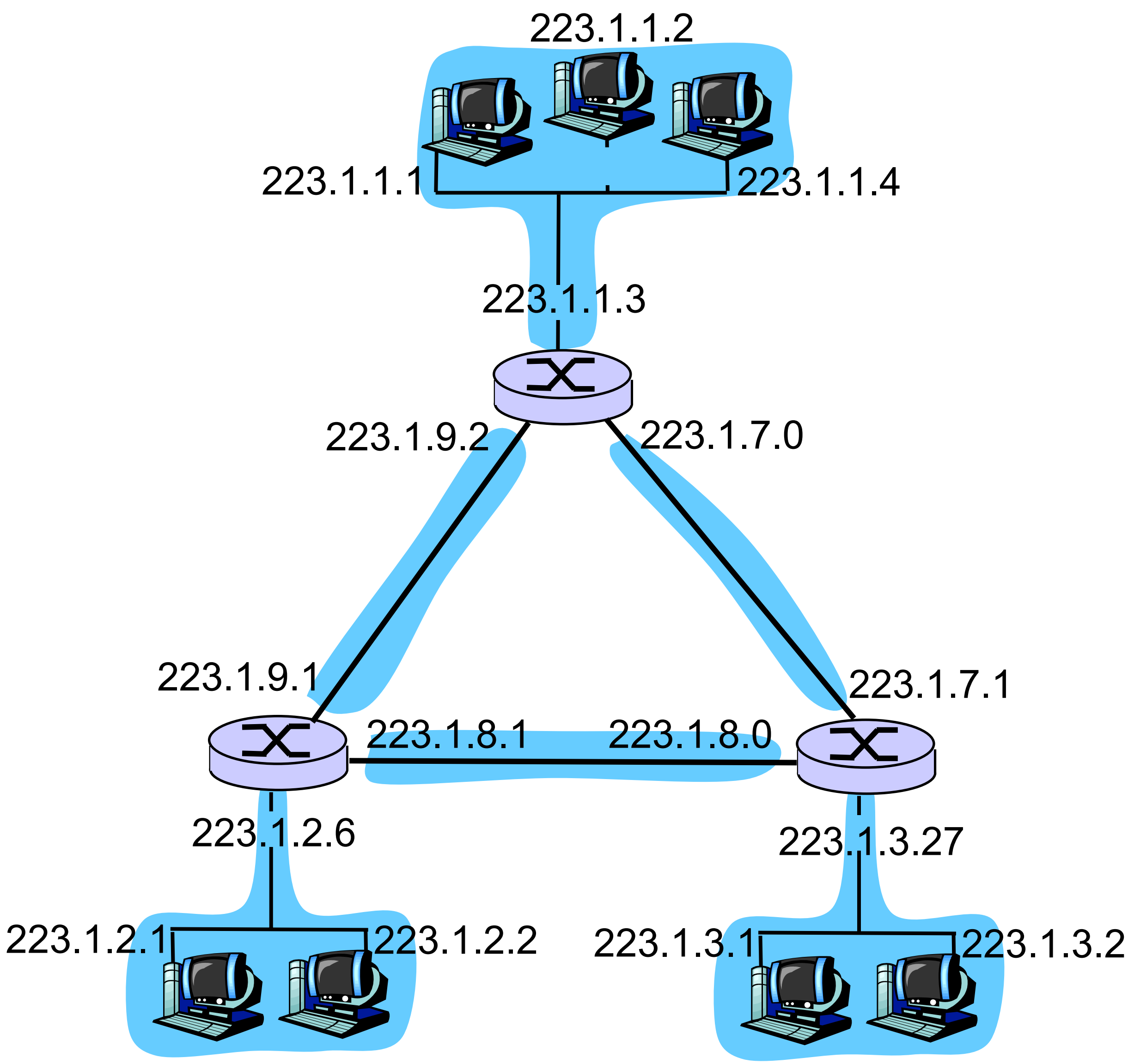
- To determine the subnets, detach each interface from its host or router, creating islands of isolated networks. Each isolated network is called a subnet.



Subnet mask: /24

Subnets

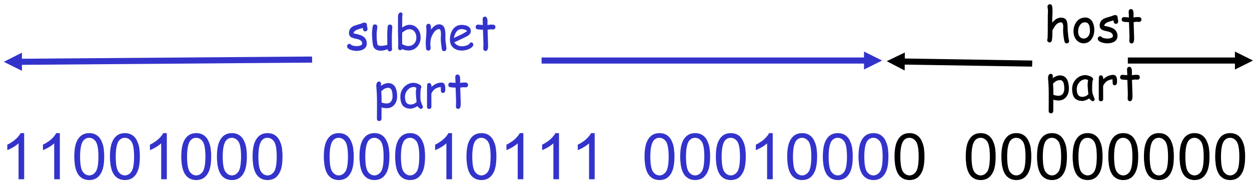
How many?



IP addressing: CIDR

CIDR: Classless InterDomain Routing

- subnet portion of address of arbitrary length
- address format: a.b.c.d/x, where x is # bits in subnet portion of address



subnet part

host part

11001000 00010111 00010000 00000000

200.23.16.0/23

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 in next chapter)

IP addresses: how to get one?

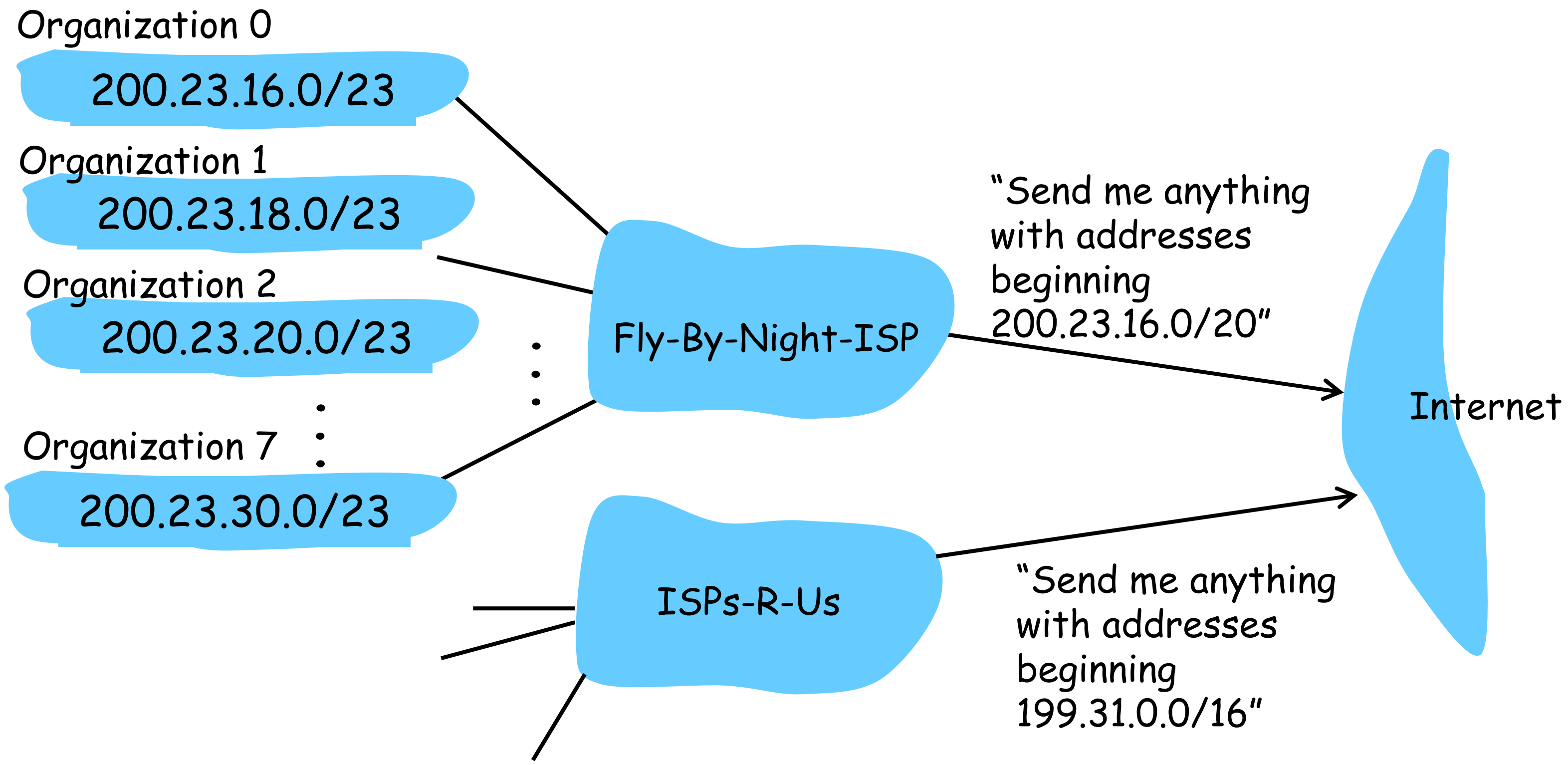
Q: How does *network* get subnet part of IP addr?

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

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:

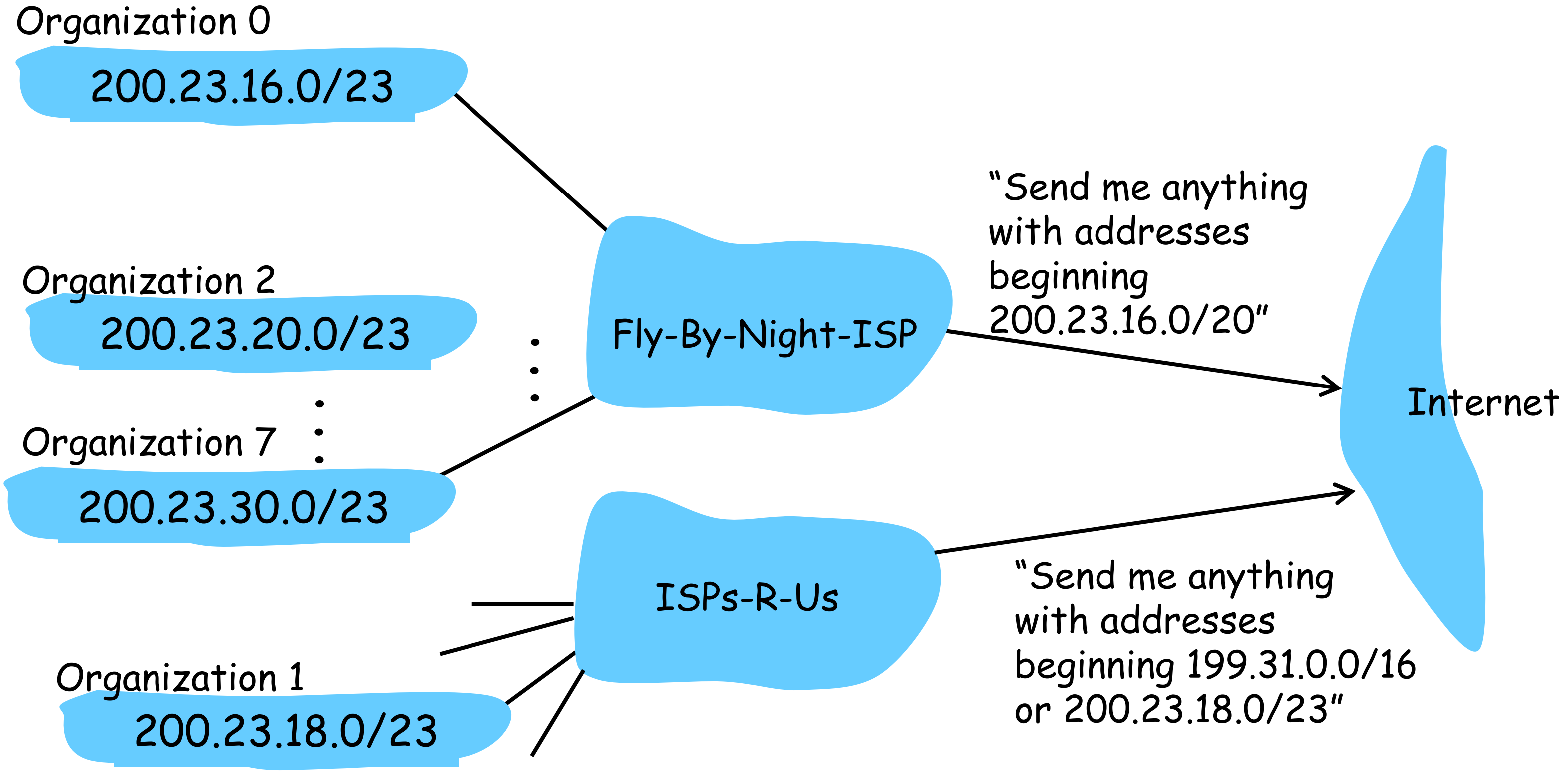


"Send me anything with addresses beginning 200.23.16.0/20"

"Send me anything with addresses beginning 199.31.0.0/16"

Hierarchical addressing: more specific routes

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



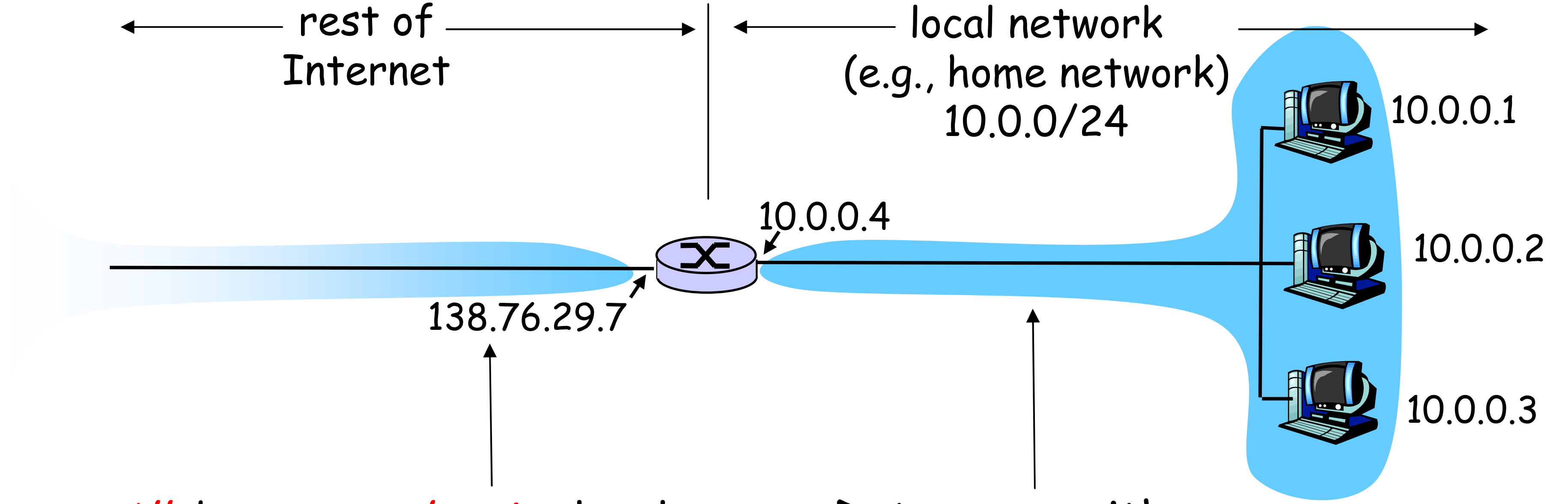
IP addressing: the last word...

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

NAT: Network Address Translation



rest of Internet

local network (e.g., home network) 10.0.0/24

10.0.0.1

10.0.0.4

10.0.0.2

138.76.29.7

10.0.0.3

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:

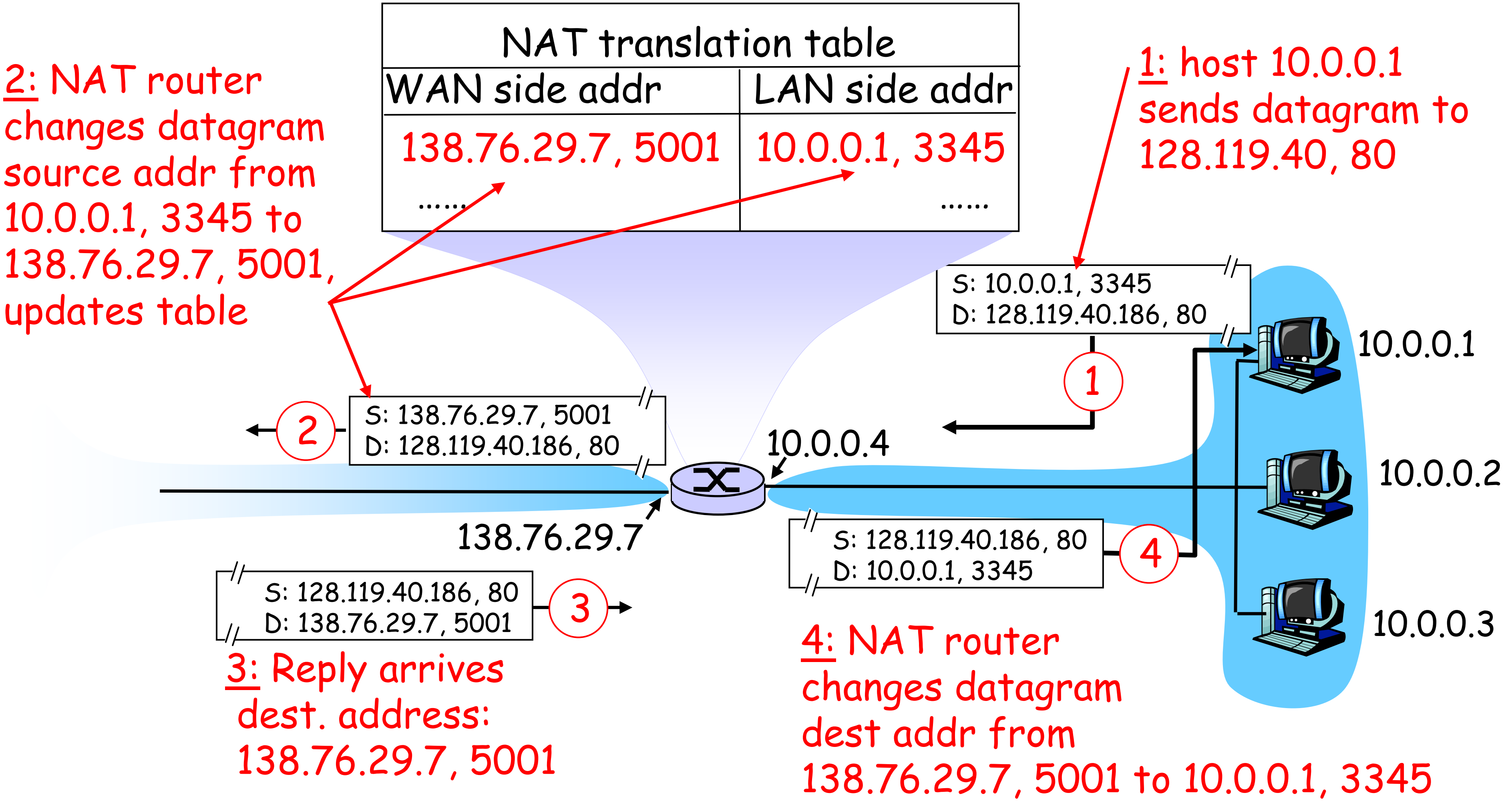
- *outgoing datagrams: replace* (source IP address, port #) of every outgoing datagram to (NAT IP address, new port #)

  . . . 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 translation table

WAN side addr	LAN side addr
138.76.29.7, 5001	10.0.0.1, 3345
……	……

1: host 10.0.0.1 sends datagram to 128.119.40, 80

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

3: Reply arrives dest. address: 138.76.29.7, 5001

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

NAT: Network Address Translation

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

ICMP: Internet Control Message Protocol

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

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

Traceroute and ICMP

- Source sends series of UDP segments to dest
  - First has TTL =1
  - Second has TTL=2, etc.
  - Unlikely port number
- When nth datagram arrives to nth router:
  - Router discards datagram
  - And sends to source an ICMP message (type 11, code 0)
  - Message includes name of router& IP address
- When ICMP message arrives, source calculates RTT
- Traceroute does this 3 times

Stopping criterion

- UDP segment eventually arrives at destination host
- Destination returns ICMP "host unreachable" packet (type 3, code 3)
- When source gets this ICMP, stops.

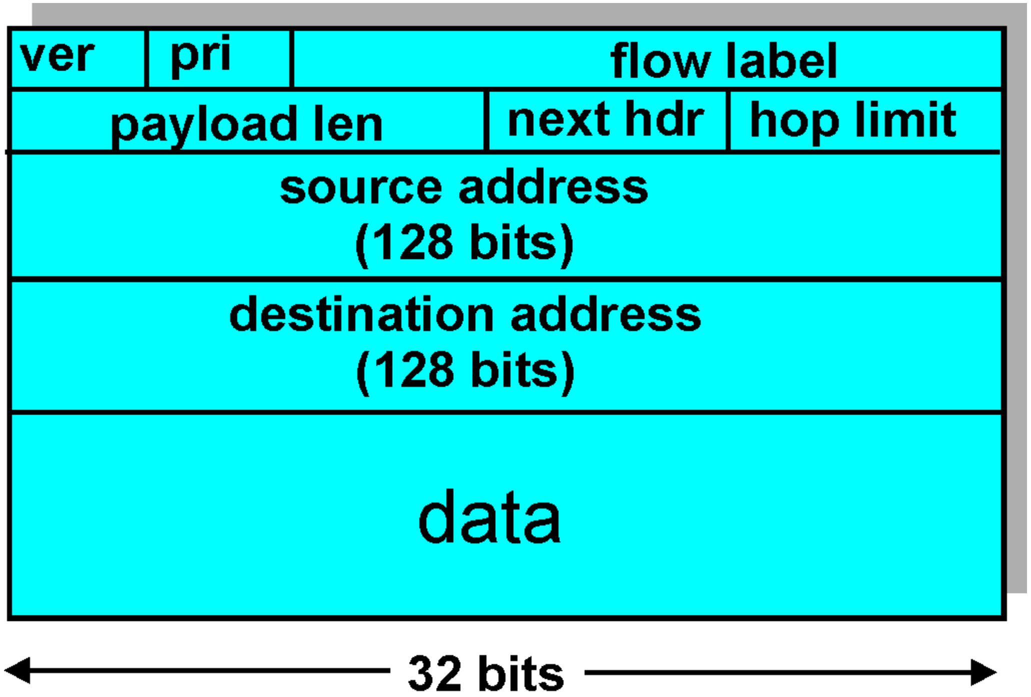
IPv6

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

  IPv6 datagram format:
  - fixed-length 40 byte header
  - no fragmentation allowed

IPv6 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



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

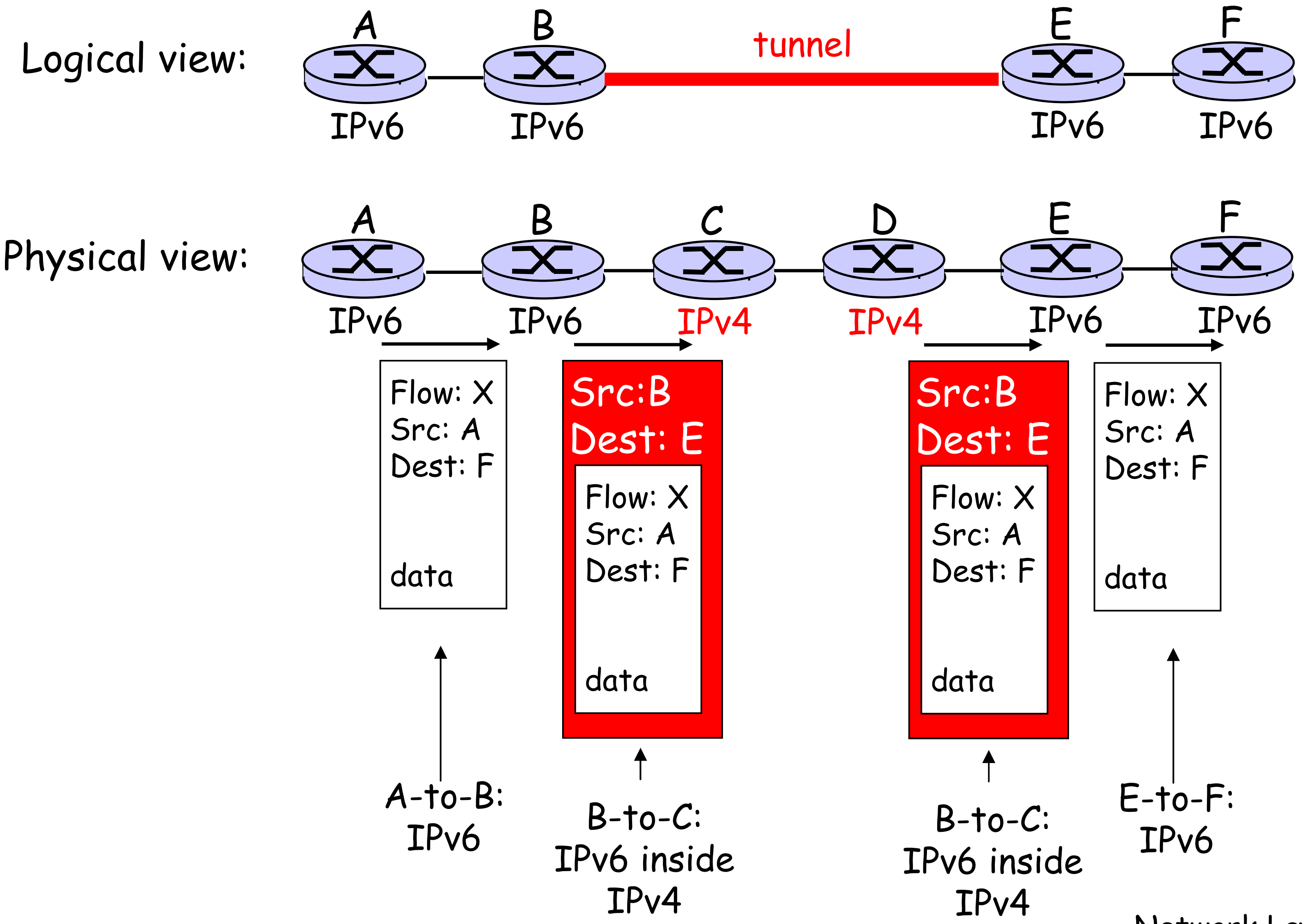
Other Changes from IPv4

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

Transition From IPv4 To IPv6

- Not all routers can be upgraded simultaneous
  - no "flag days"
  - How will the network operate with mixed IPv4 and IPv6 routers?
- *Tunneling:* IPv6 carried as payload in IPv4 datagram among IPv4 routers

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

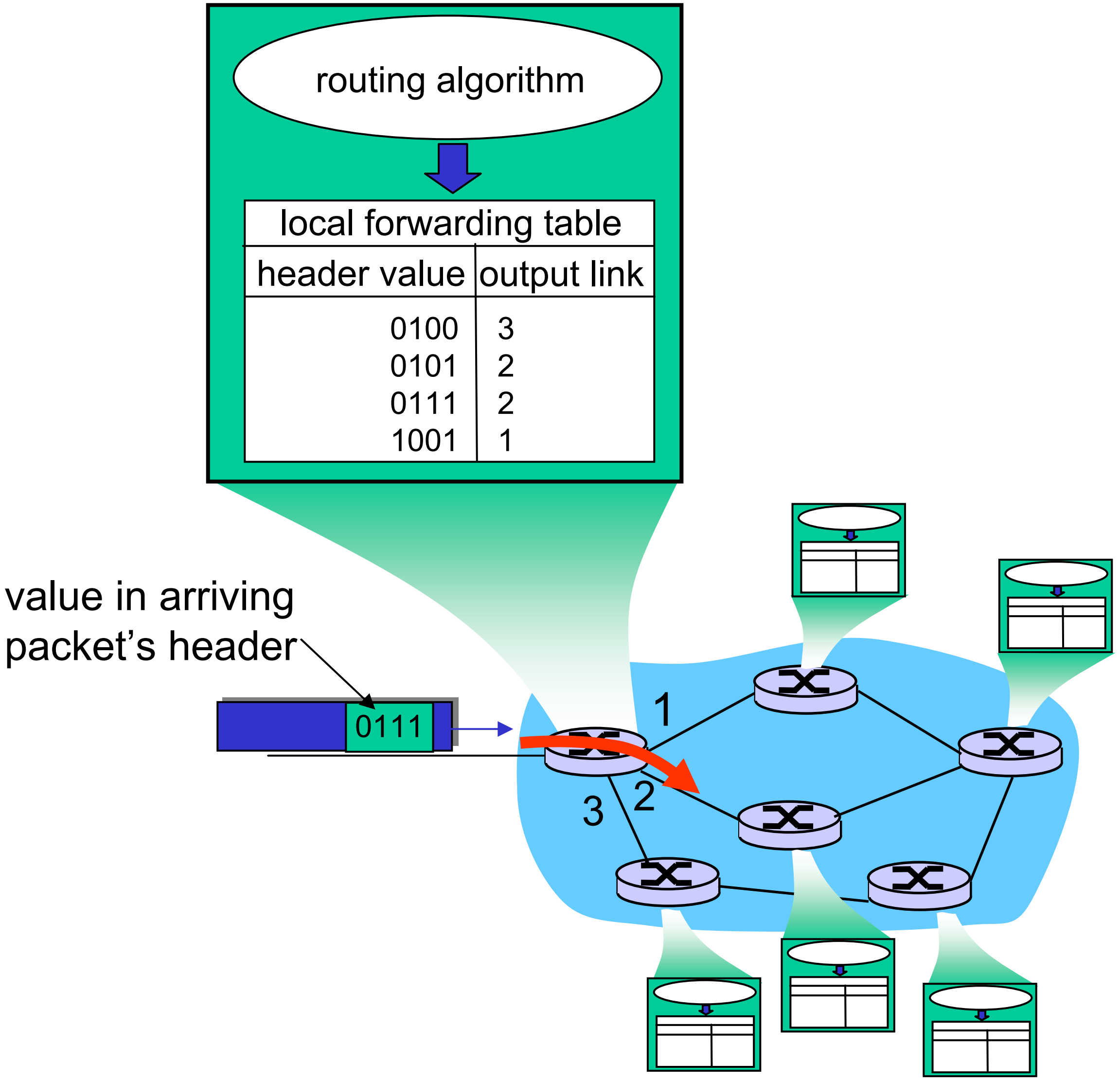
A-to-B:
IPv6

B-to-C:
IPv6 inside
IPv4

B-to-C:
IPv6 inside
IPv4

E-to-F:
IPv6

Interplay between routing and forwarding



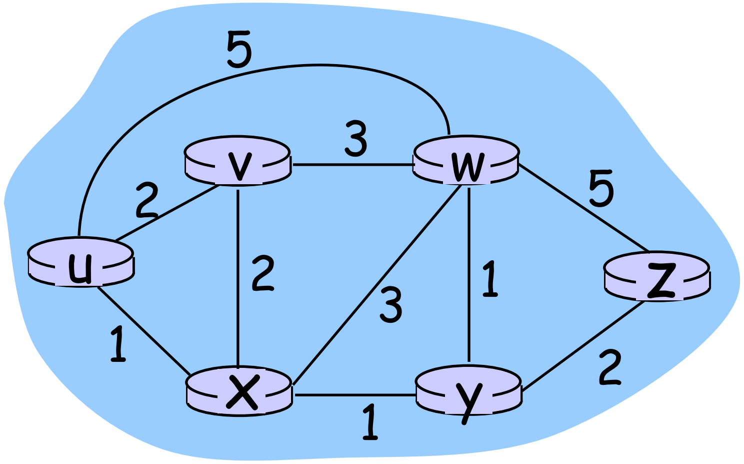
routing algorithm

local forwarding table

header value	output link
0100	3
0101	2
0111	2
1001	1

value in arriving packet's header

Graph abstraction



Graph: G = (N,E)

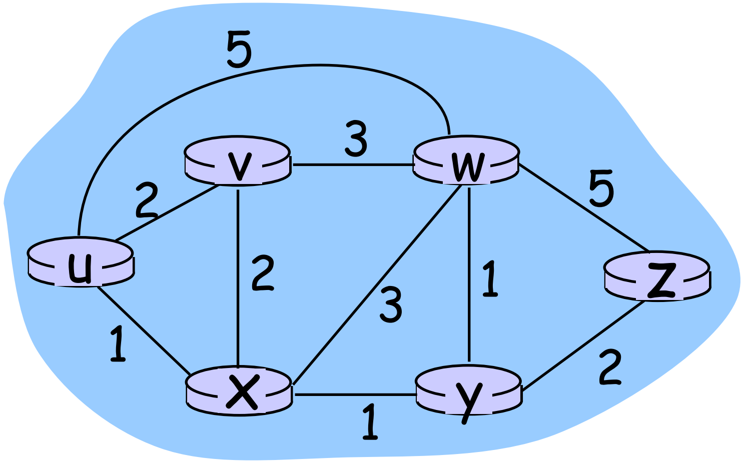
N = set of routers = { u, v, w, x, y, z }

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

Remark: Graph abstraction is useful in other network contexts

Example: P2P, where N is set of peers and E is set of TCP connections

Graph abstraction: costs



- c(x,x') = cost of link (x,x')

  - e.g., c(w,z) = 5

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

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

Question: What's the least-cost path between u and z ?

Routing algorithm: algorithm that finds least-cost path

Routing Algorithm classification

Global or decentralized information?

Global:

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

Decentralized:

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

Static or dynamic?

Static:

- routes change slowly over time

Dynamic:

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

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 forwarding table for that node
- iterative: after k iterations, know least cost path to k dest.'s

Notation:

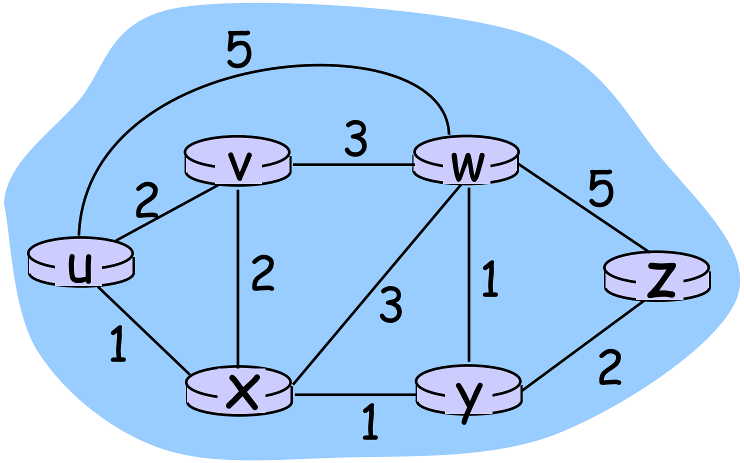
- c(x,y): link cost from node x to y; = ∞ if not direct neighbors
- D(v): current value of cost of path from source to dest. v
- p(v): predecessor node along path from source to v
- N': set of nodes whose least cost path definitively known

Dijsktra's Algorithm

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

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

Dijkstra's algorithm: example

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



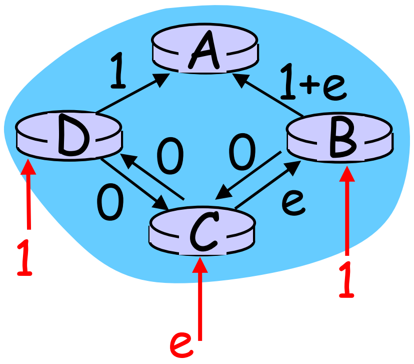
Dijkstra's algorithm, discussion

Algorithm complexity: n nodes

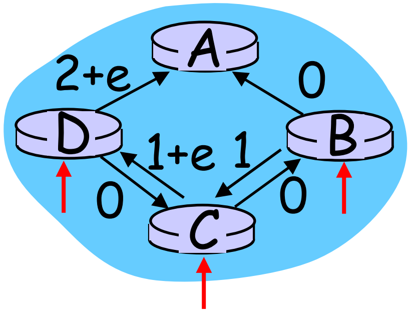
- each iteration: need to check all nodes, w, not in N
- n(n+1)/2 comparisons: O(n^2)
- more efficient implementations possible: O(nlogn)

Oscillations possible:

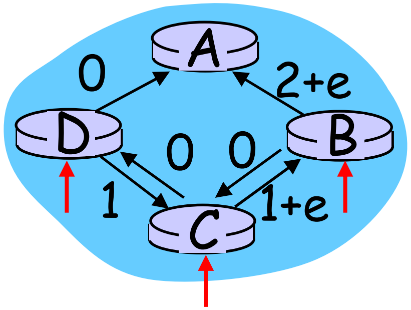
- e.g., link cost = amount of carried traffic



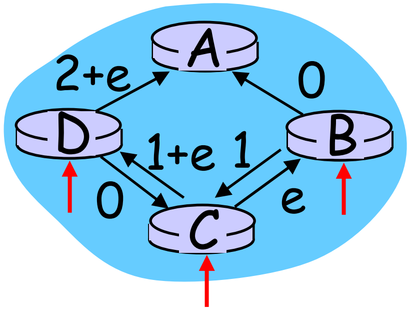
initially



... recompute routing



... recompute



... recompute

Distance Vector Algorithm (1)

Bellman-Ford Equation (dynamic programming)

Define

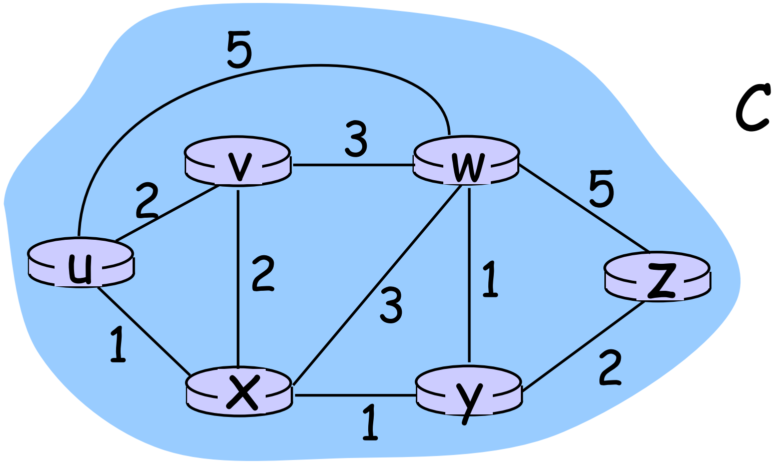
$d_x(y)$:= cost of least-cost path from x to y

Then

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

where min is taken over all neighbors of x

Bellman-Ford example (2)



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

B-F equation says:

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

Node that achieves minimum is next hop in the shortest path forwarding table

Distance Vector Algorithm (3)

- $D_x(y)$ = estimate of least cost from x to y
- Distance vector: $\mathbf{D}_x = [D_x(y): y \in N]$
- Node x knows cost to each neighbor v: $c(x,v)$
- Node x maintains $\mathbf{D}_x = [D_x(y): y \in N]$
- Node x also maintains its neighbors' distance vectors
  - For each neighbor v, x maintains $\mathbf{D}_v = [D_v(y): y \in N]$

Distance vector algorithm (4)

Basic idea:

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

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

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

Distance Vector Algorithm (5)

Iterative, asynchronous:

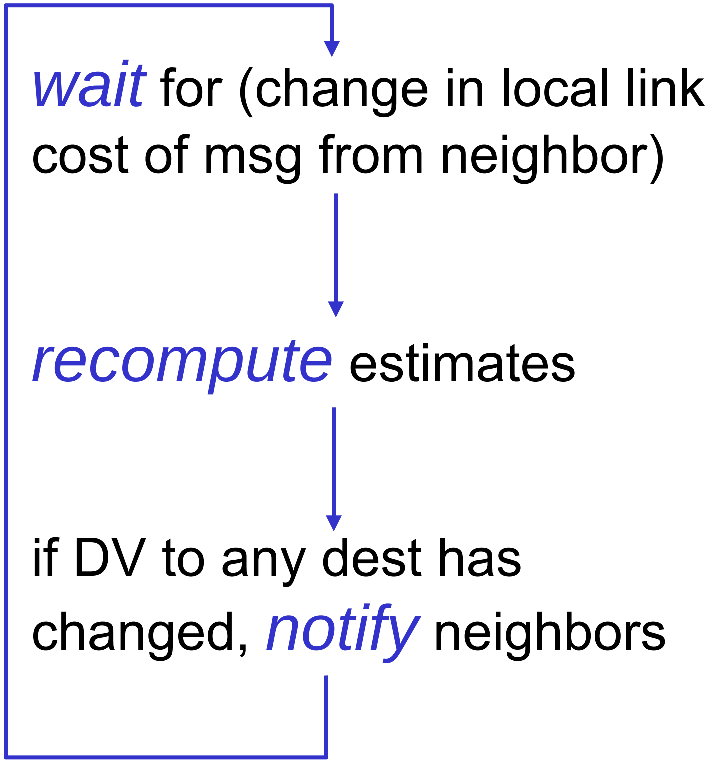
each local iteration caused by:

- local link cost change
- DV update message from neighbor

Distributed:

- each node notifies neighbors *only* when its DV changes
  - neighbors then notify their neighbors if necessary

Each node:



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

recompute estimates

if DV to any dest has changed, *notify* neighbors

Distance Vector Algorithm:

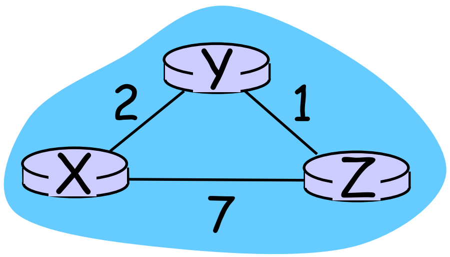
At all nodes, X:

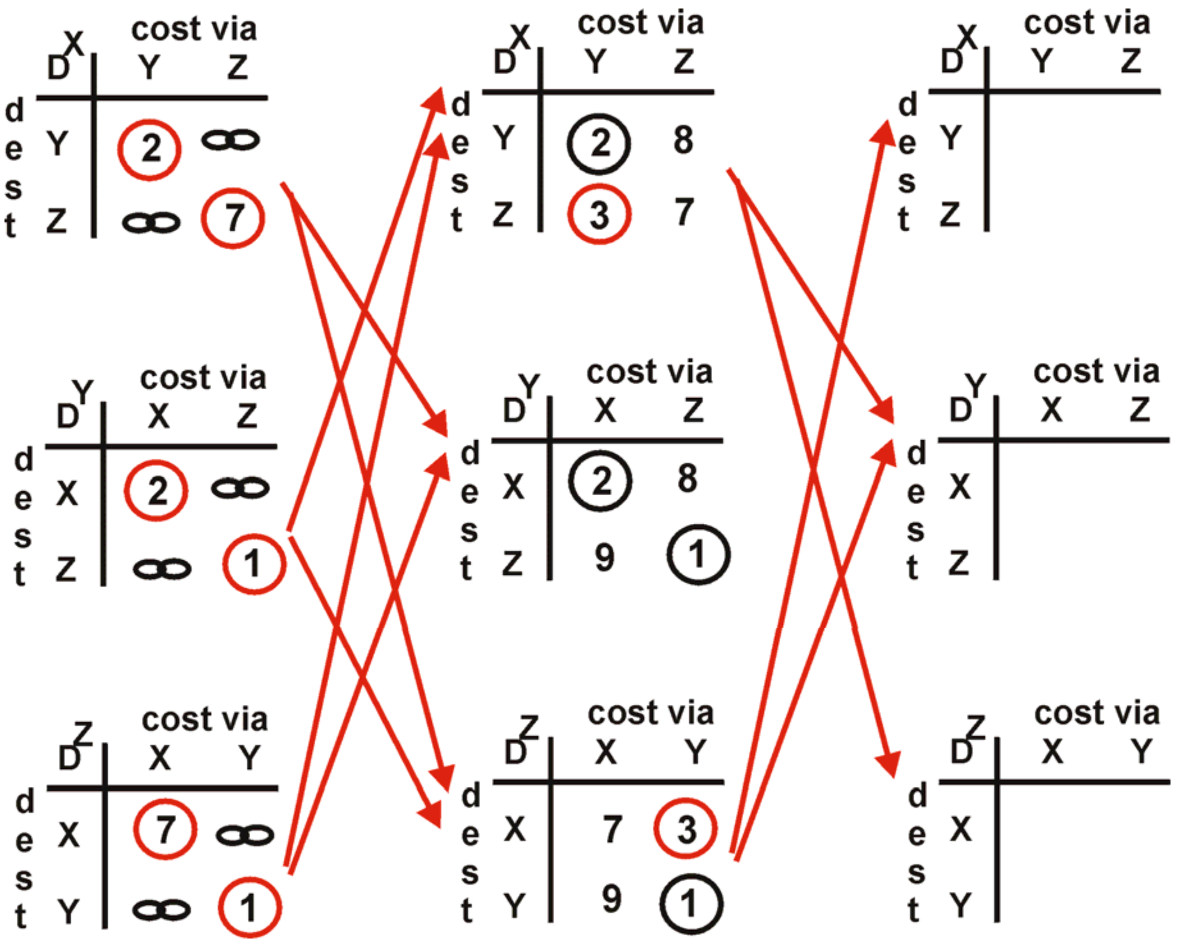
```
Initialization:
 for all adjacent nodes v:
    $D^X(*,v)$ = infinity        /* the * operator means "for all rows" */
    $D^X(v,v) = c(X,v)$
 for all destinations, y
    send $\min_w D^X(y,w)$ to each neighbor  /* w over all X's neighbors */
```

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	
		Y	Z
dest	Y	2	∞
	Z	∞	7

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

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

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

	D^Y	cost via	
		X	Z
dest	X	2	8
	Z	9	1

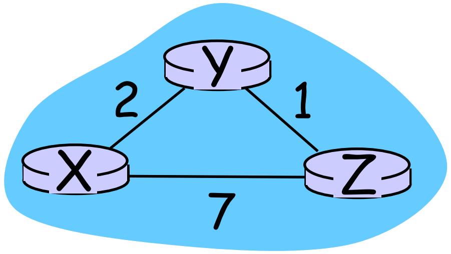
	D^Z	cost via	
		X	Y
dest	X	7	3
	Y	9	1

	D^X	cost via	
		Y	Z
dest	Y		
	Z		

	D^Y	cost via	
		X	Z
dest	X		
	Z		

	D^Z	cost via	
		X	Y
dest	X		
	Y		

Distance Vector Algorithm: example



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

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

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

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

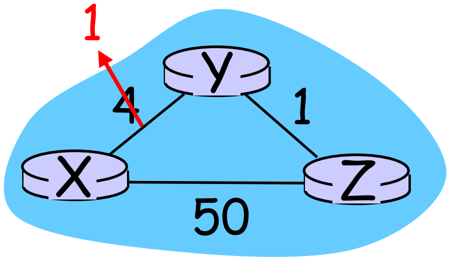
$$D^X(Y,Z) = c(X,Z) + \min_w\{D^Z(Y,w)\}$$
$$= 7+1 = 8$$

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

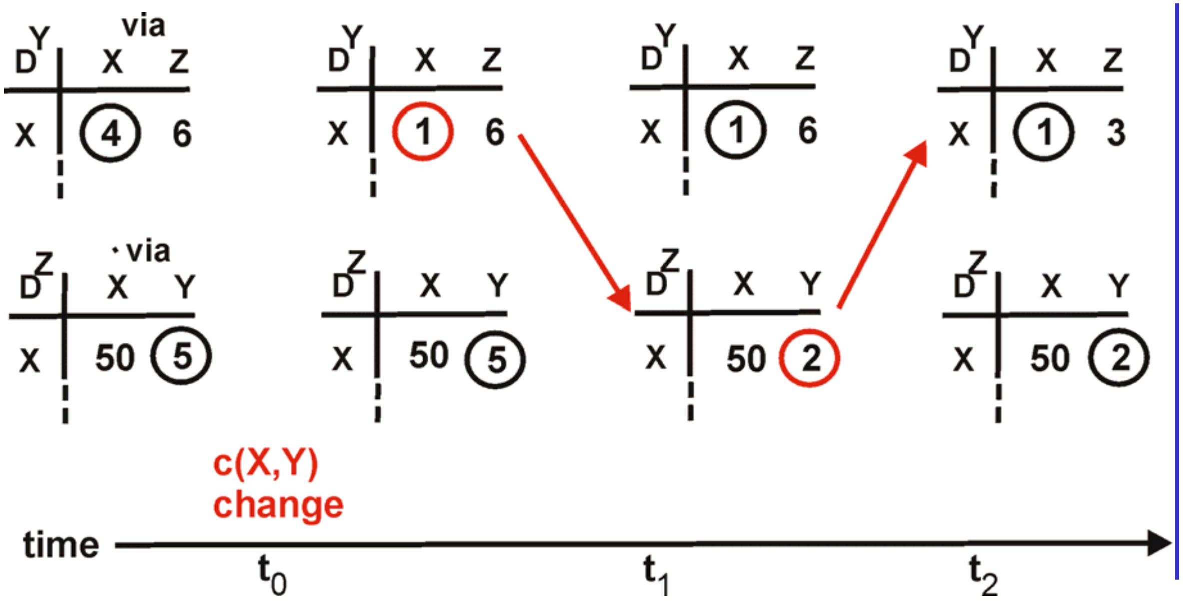
Distance Vector: link cost changes

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"



D^Y	via X	Z
X	4	6

D^Z	via X	Y
X	50	5

D^Y	X	Z
X	1	6

D^Z	X	Y
X	50	5

D^Y	X	Z
X	1	6

D^Z	X	Y
X	50	2

D^Y	X	Z
X	1	3

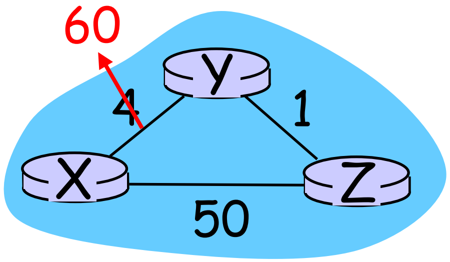
D^Z	X	Y
X	50	2

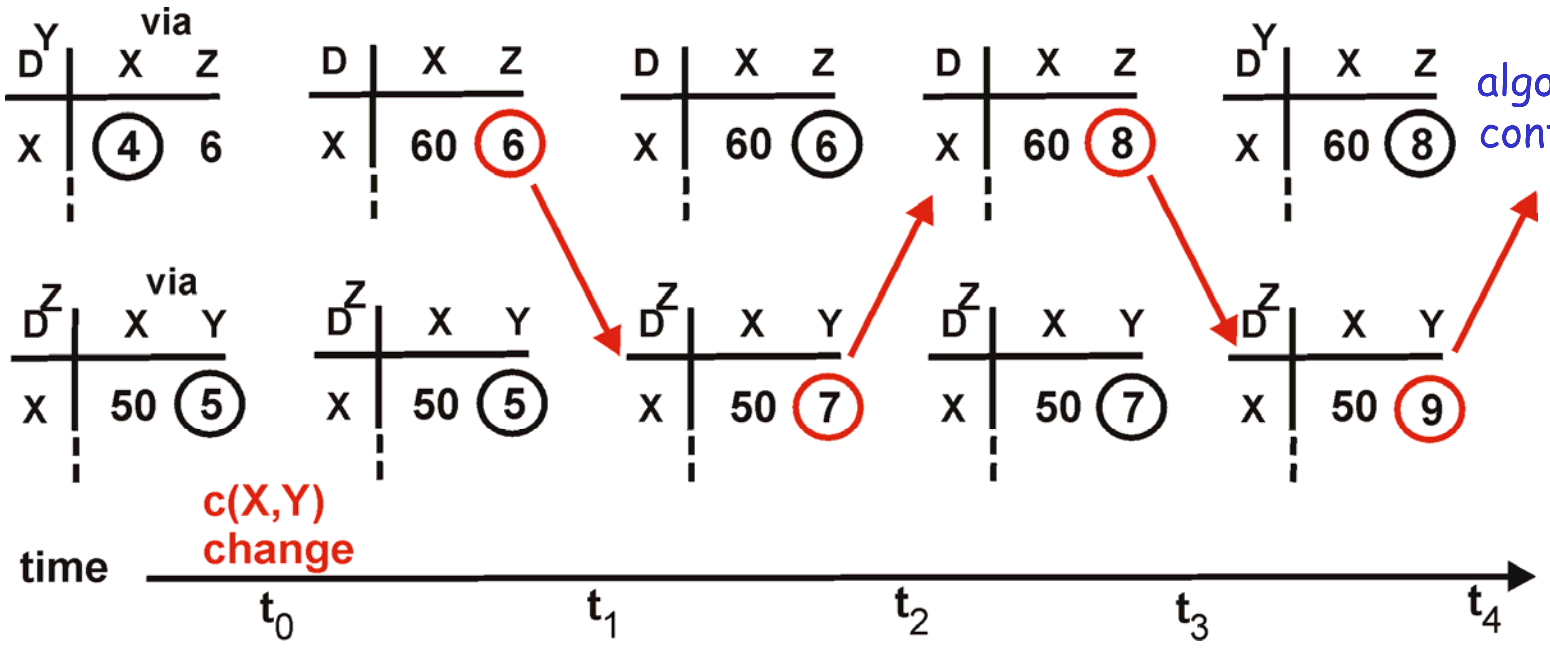
algorithm terminates

Distance Vector: link cost changes

Link cost changes:

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





via

D^Y	X	Z
X	4	6

D	X	Z
X	60	6

D	X	Z
X	60	6

D	X	Z
X	60	8

D^Y	X	Z
X	60	8

via

D^Z	X	Y
X	50	5

D^Z	X	Y
X	50	5

D^Z	X	Y
X	50	7

D^Z	X	Y
X	50	7

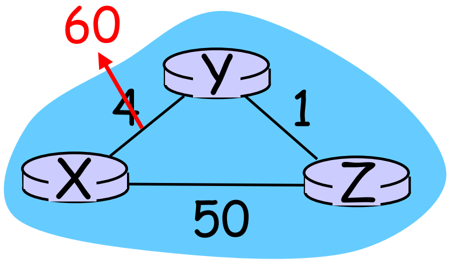
D^Z	X	Y
X	50	9

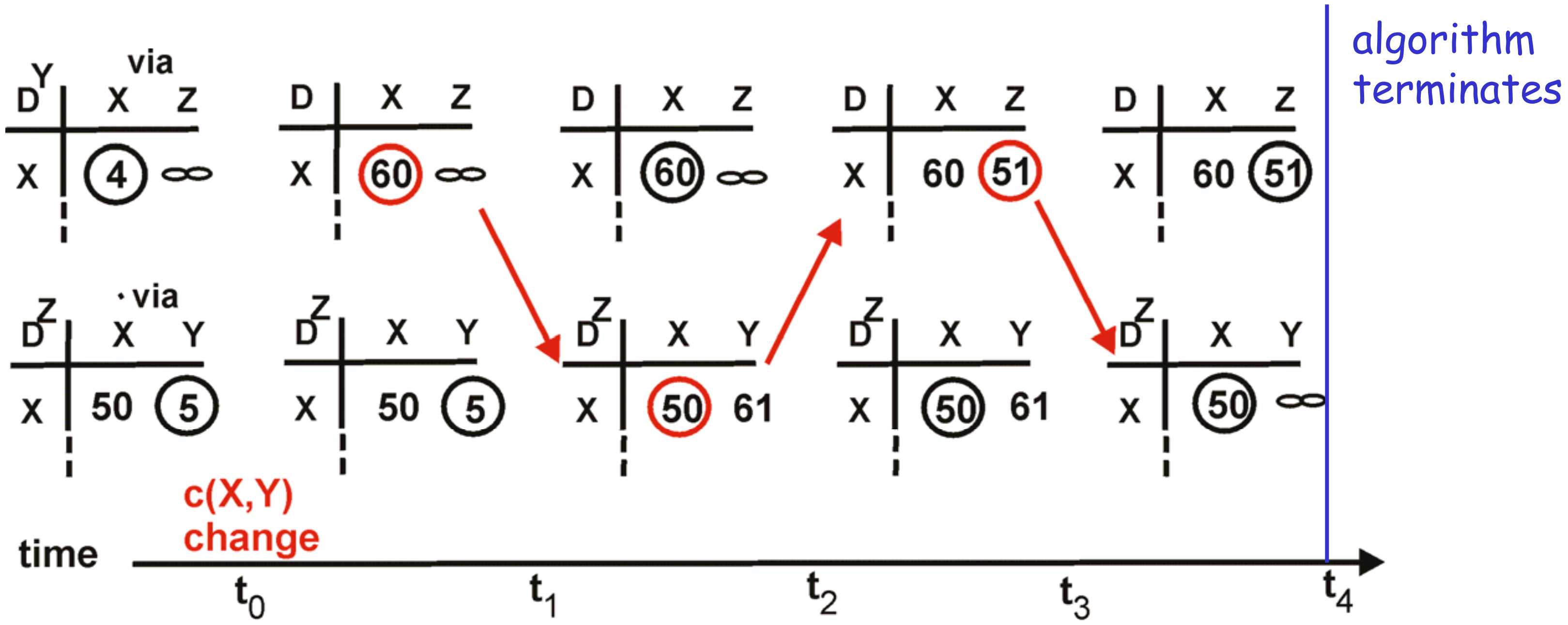
algorithm continues on!

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?





D^Y	X (via)	Z
X	4	∞

D^Y	X	Z
X	60	∞

D^Y	X	Z
X	60	∞

D^Y	X	Z
X	60	51

D^Y	X	Z
X	60	51

D^Z	X (via)	Y
X	50	5

D^Z	X	Y
X	50	5

D^Z	X	Y
X	50	61

D^Z	X	Y
X	50	61

D^Z	X	Y
X	50	∞

Comparison of LS and DV algorithms

Message complexity

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

Speed of Convergence

- LS: O(n²) algorithm requires O(nE) msgs
  - may have oscillations
- DV: convergence time varies
  - may be routing loops
  - count-to-infinity problem

Robustness: what happens if router malfunctions?

LS:

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

DV:

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

Hierarchical Routing

Our routing study thus far - idealization

- all routers identical
- network "flat"

… *not* true in practice

scale: with 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

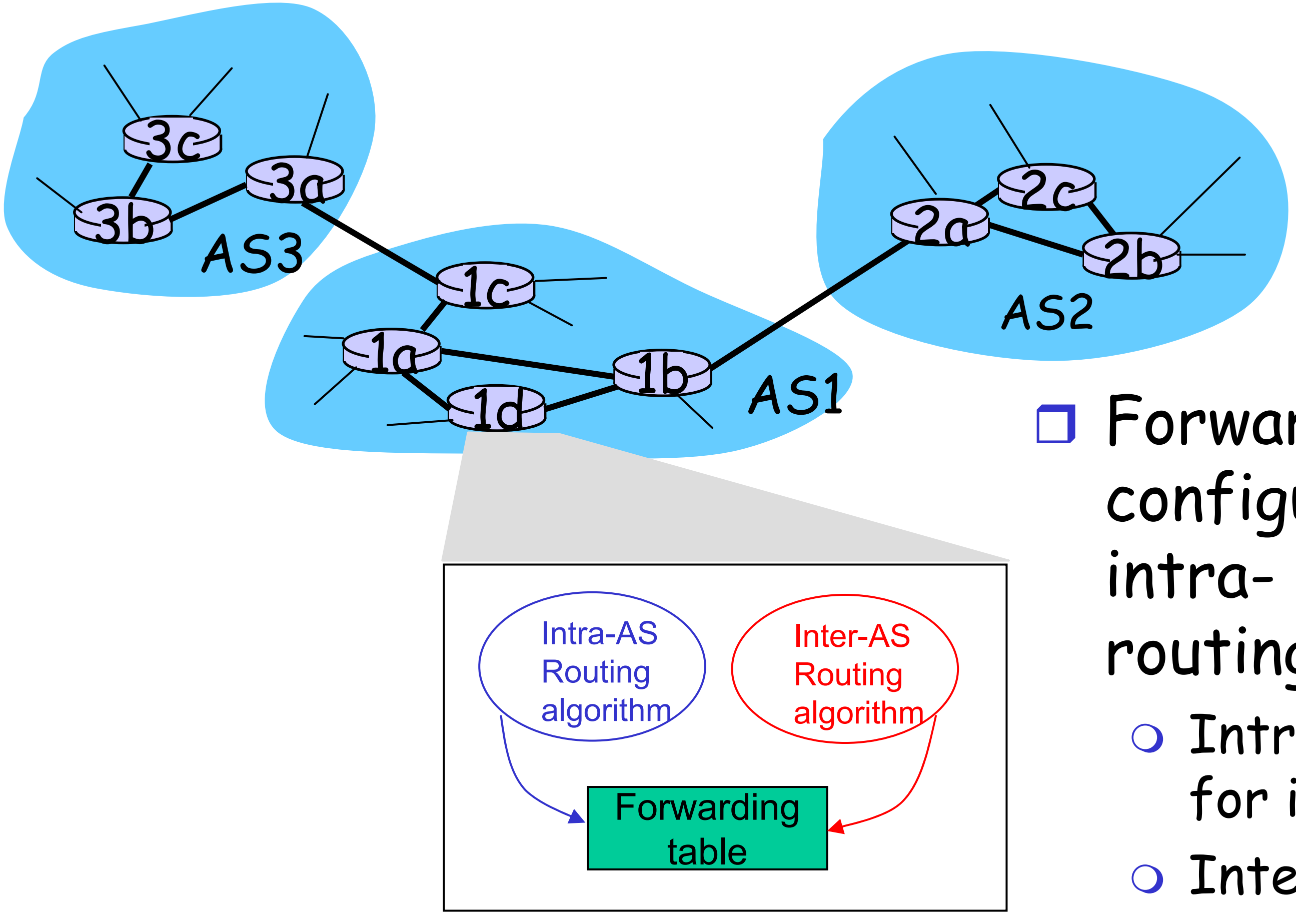
Hierarchical Routing

- 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 router

- Direct link to router in another AS

Interconnected ASes



- Forwarding table is configured by both intra- and inter-AS routing algorithm
  - Intra-AS sets entries for internal dests
  - Inter-AS & Intra-As sets entries for external dests

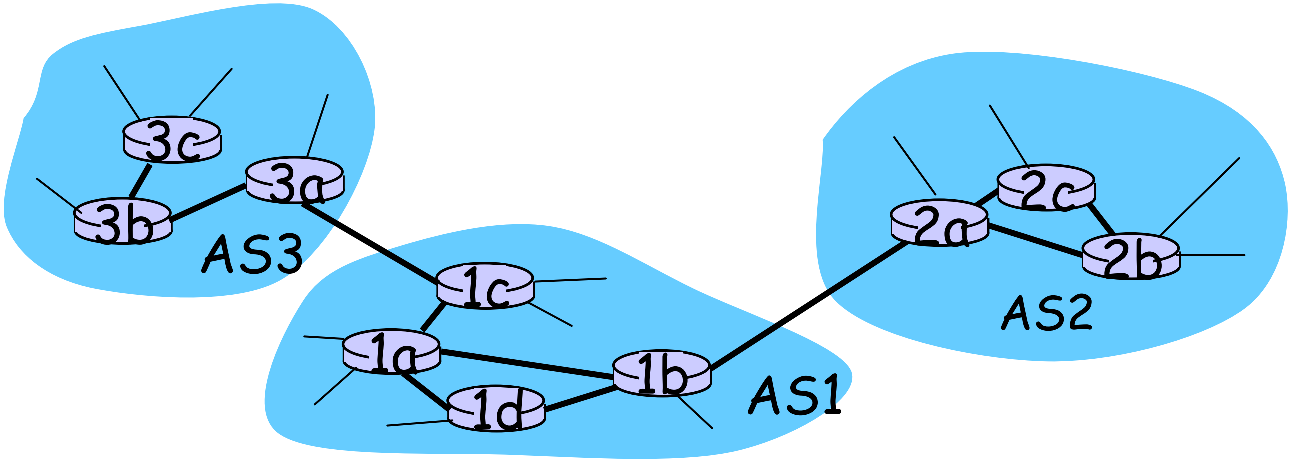
Inter-AS tasks

- Suppose router in AS1 receives datagram for which dest is outside of AS1
  - Router should forward packet towards on of the gateway routers, but which one?

AS1 needs:

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

Job of inter-AS routing!

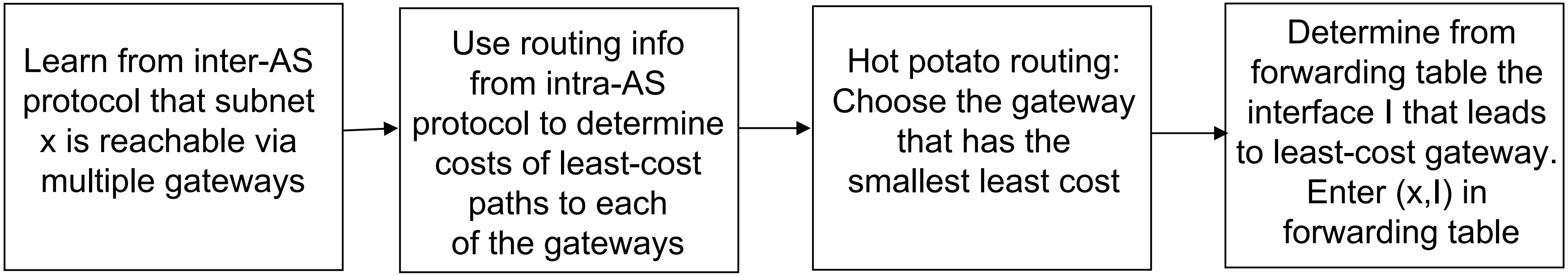


Example: Setting forwarding table in router 1d

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

Example: Choosing among multiple ASes

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



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

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

Hot potato routing: Choose the gateway that has the smallest least cost

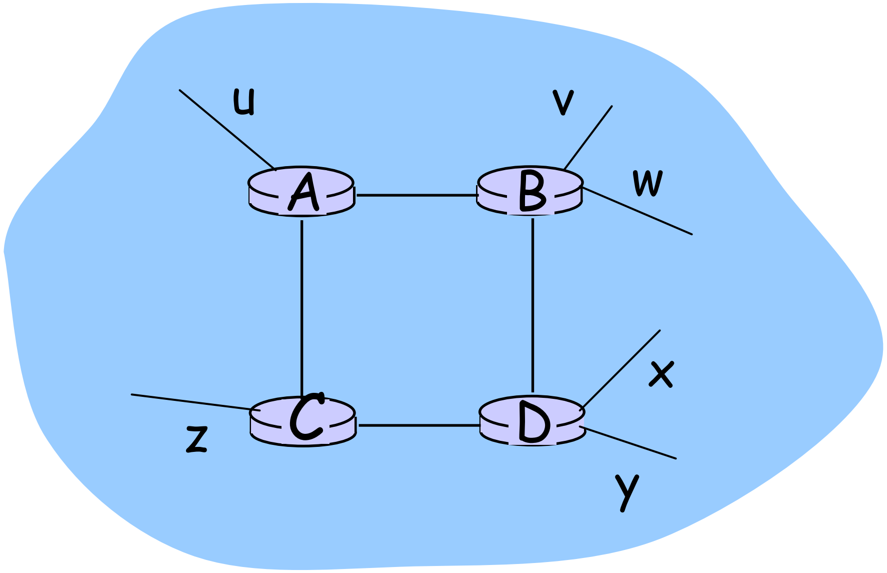
Determine from forwarding table the interface I that leads to least-cost gateway. Enter (x,I) in forwarding table

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)

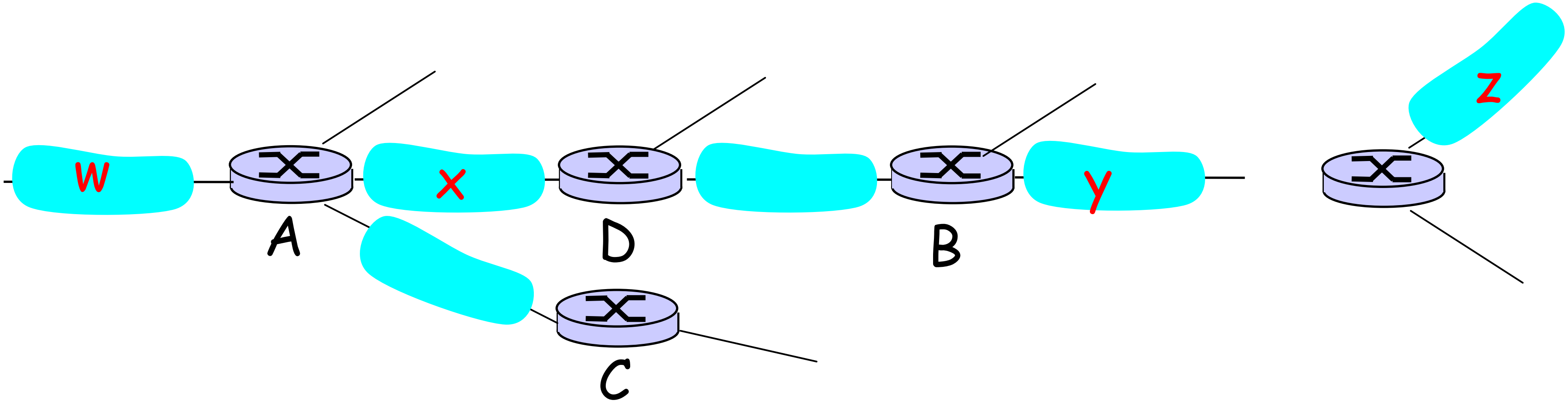


destination	hops
u	1
v	2
w	2
x	3
y	3
z	2

RIP advertisements

- 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



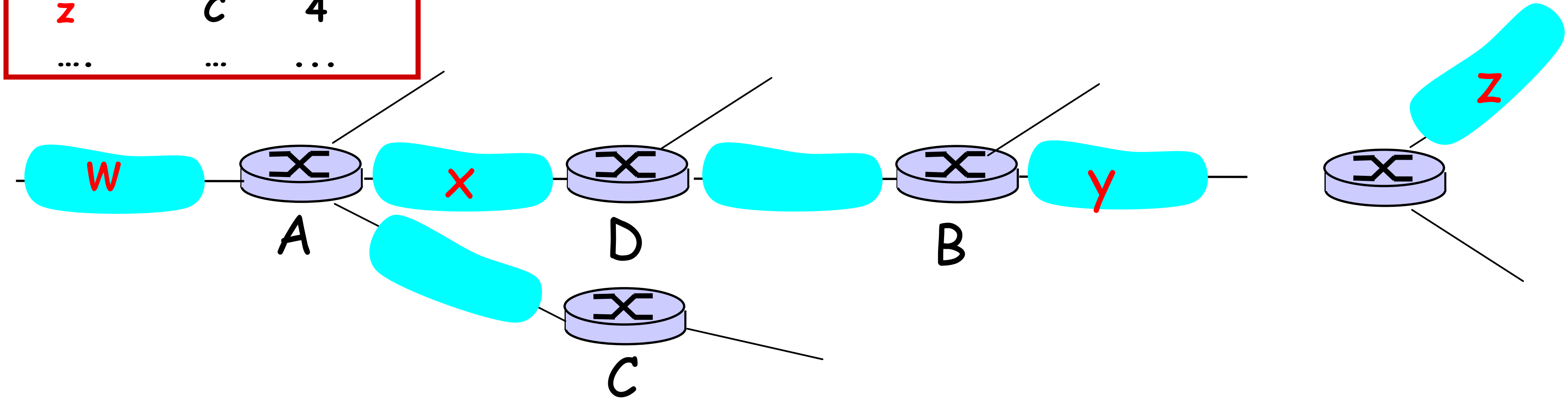
Destination Network	Next Router	Num. of hops to dest.
w	A	2
y	B	2
z	B	7
x	--	1
….	….	….

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

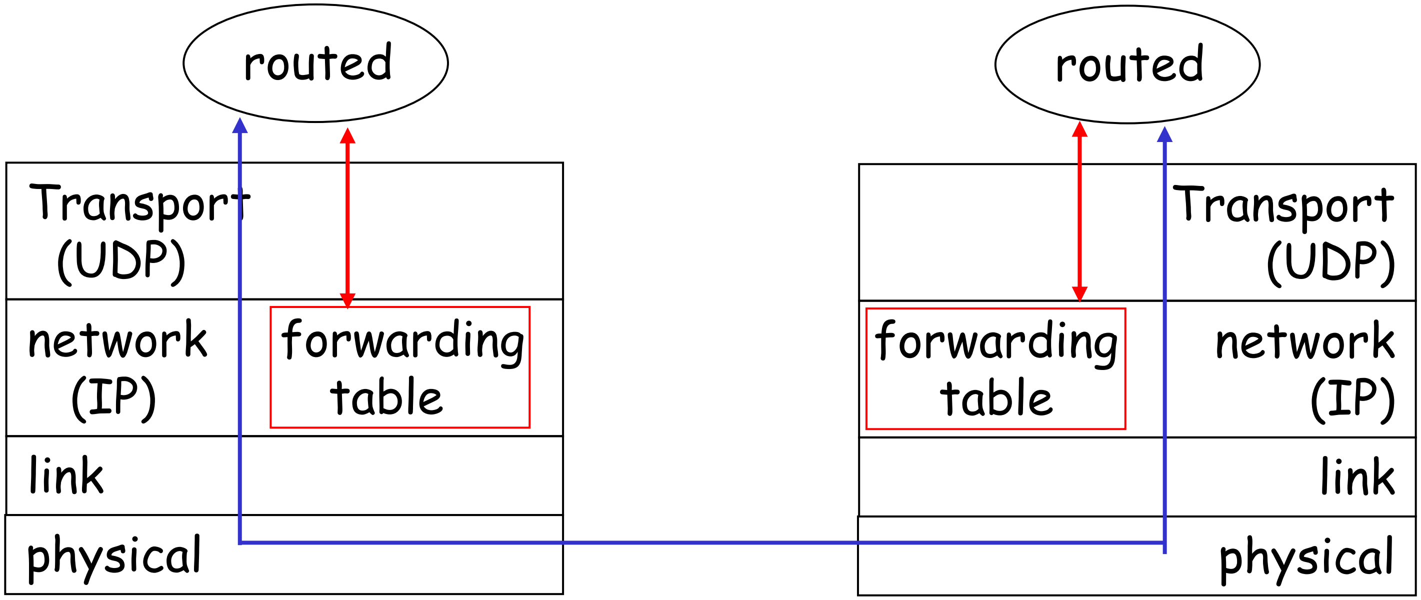
RIP: Link Failure and Recovery

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



OSPF (Open Shortest Path First)

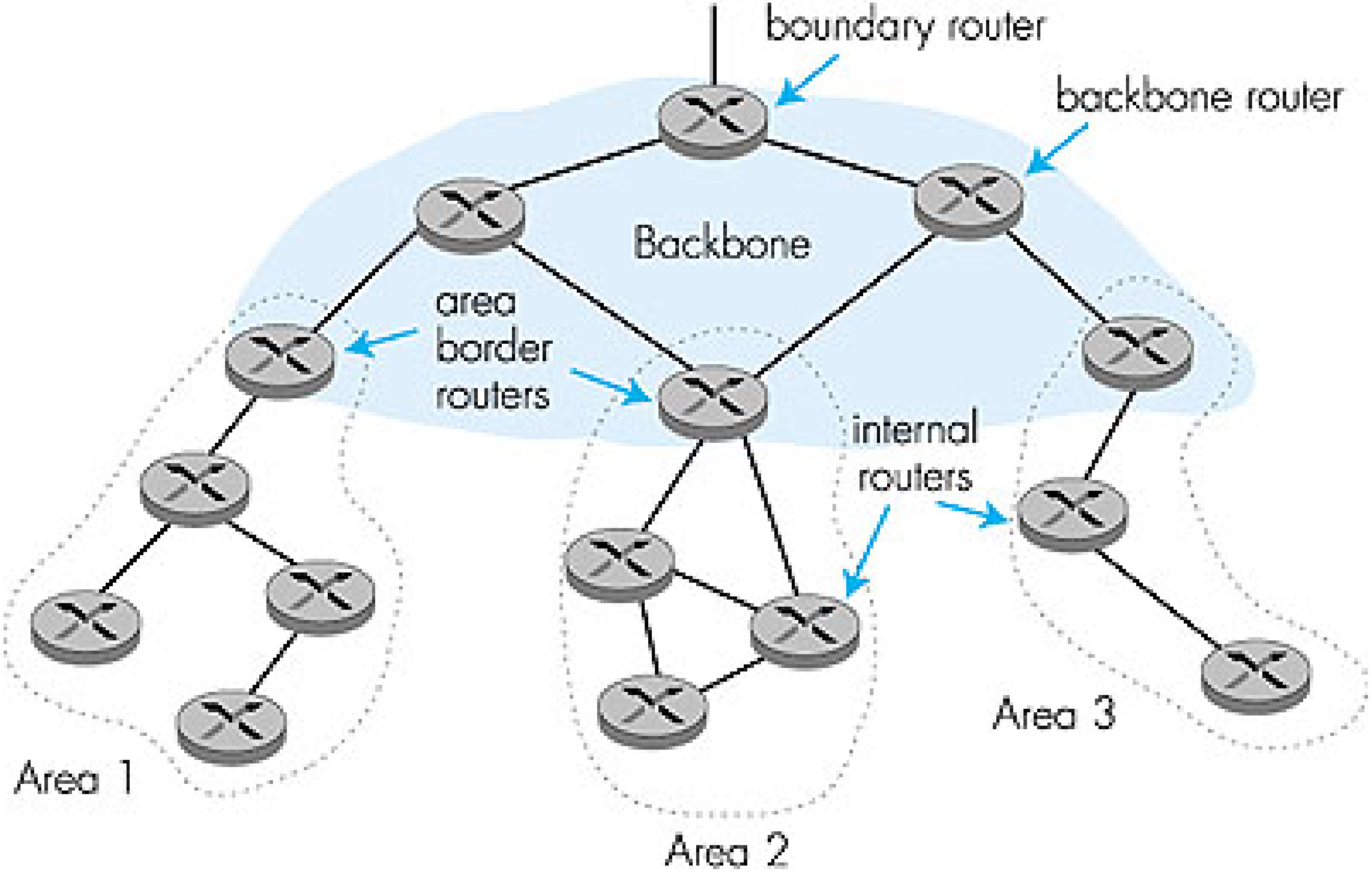
- "open": publicly available
- Uses Link State algorithm
  - LS packet dissemination
  - Topology map at each node
  - Route computation using Dijkstra's algorithm

- 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

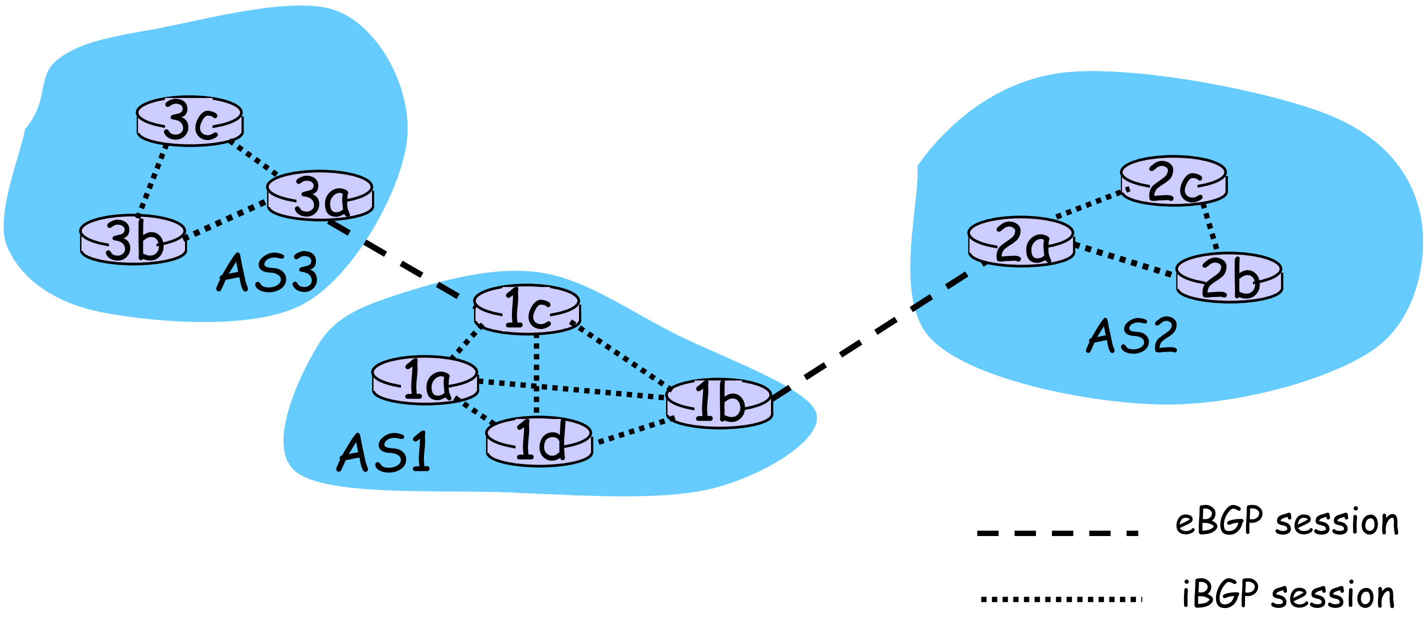
- Two-level hierarchy: local area, backbone.
  - Link-state advertisements only in area
  - each nodes has detailed area topology; only know direction (shortest path) to nets in other areas.
- **Area border routers:** "summarize" distances to nets in own area, advertise to other Area Border routers.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

Internet inter-AS routing: BGP

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

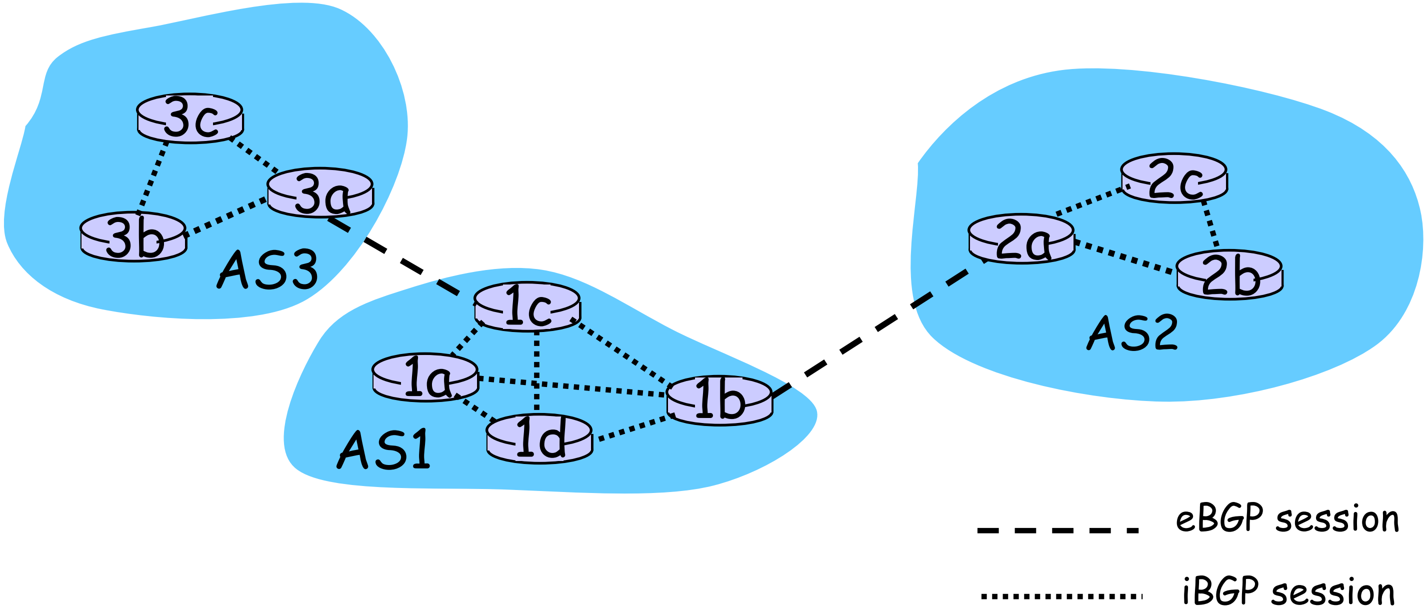
BGP basics

- Pairs of routers (BGP peers) exchange routing info over semi-permanent TCP connections: BGP sessions
- Note that BGP sessions do not correspond to physical links.
- When AS2 advertises a prefix to AS1, AS2 is *promising* it will forward any datagrams destined to that prefix towards the prefix.
  - AS2 can aggregate prefixes in its advertisement



Distributing reachability info

- With eBGP session between 3a and 1c, AS3 sends prefix reachability info to AS1.
- 1c can then use iBGP do distribute this new prefix reach info to all routers in AS1
- 1b can then re-advertise the new reach info to AS2 over the 1b-to-2a eBGP session
- When router learns about a new prefix, it creates an entry for the prefix in its forwarding table.



Path attributes & BGP routes

- When advertising a prefix, advert includes BGP attributes.
  - prefix + attributes = "route"
- Two important attributes:
  - AS-PATH: contains the ASs through which the advert for the prefix passed: AS 67 AS 17
  - NEXT-HOP: Indicates the specific internal-AS router to next-hop AS. (There may be multiple links from current AS to next-hop-AS.)
- When gateway router receives route advert, uses import policy to accept/decline.

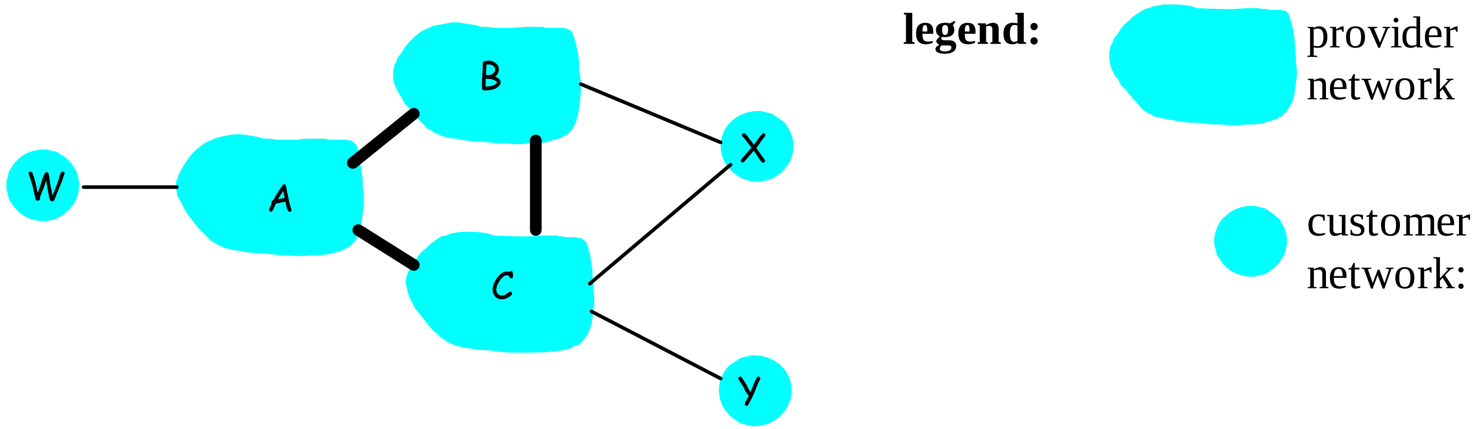
BGP route selection

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

BGP messages

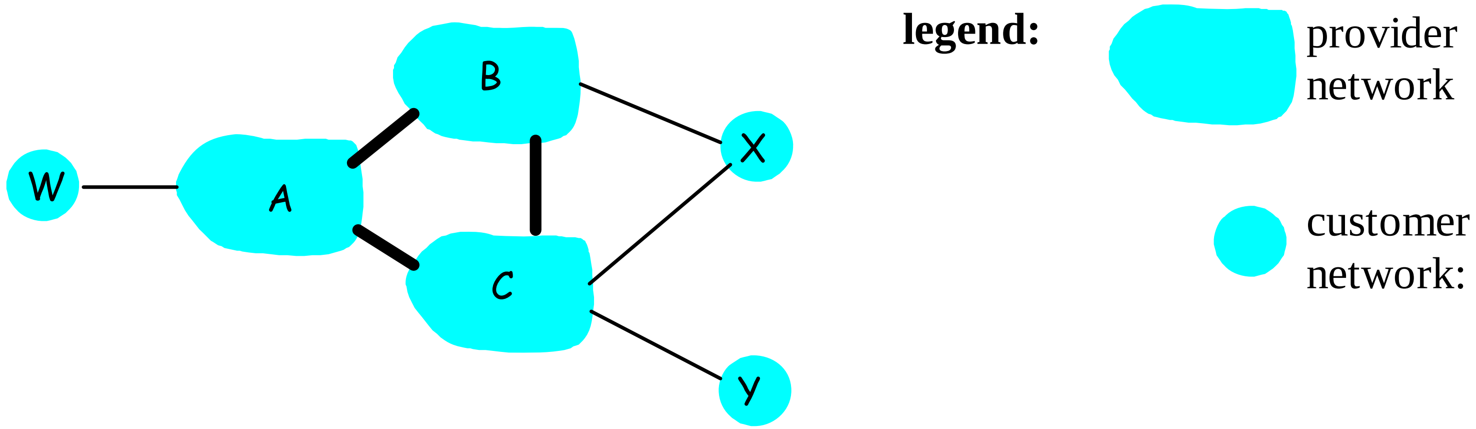
- BGP messages exchanged using TCP.
- BGP messages:
  - OPEN: opens TCP connection to peer and authenticates sender
  - UPDATE: advertises new path (or withdraws old)
  - KEEPALIVE keeps connection alive in absence of UPDATES; also ACKs OPEN request
  - NOTIFICATION: reports errors in previous msg; also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

BGP routing policy



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

BGP routing policy (2)



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

Why different Intra- and Inter-AS routing ?

Policy:

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

Scale:

- hierarchical routing saves table size, reduced update traffic

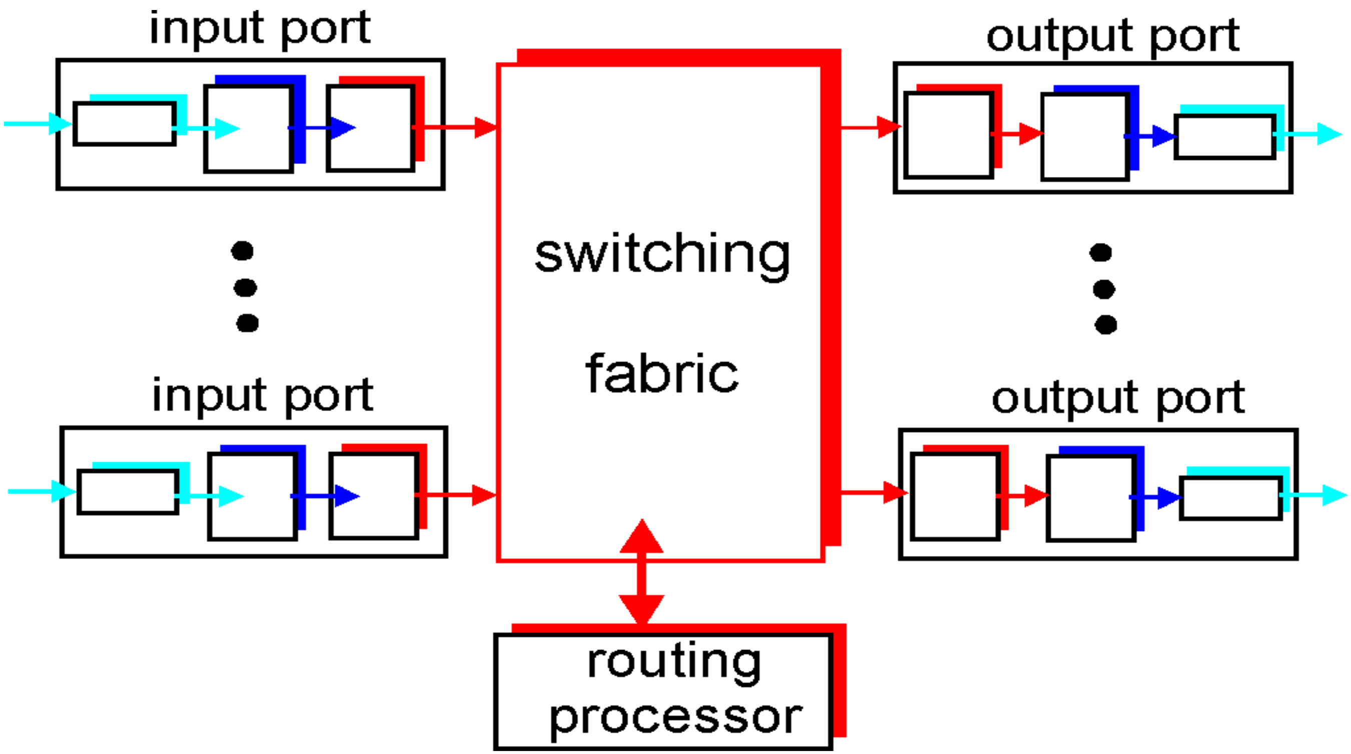
Performance:

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

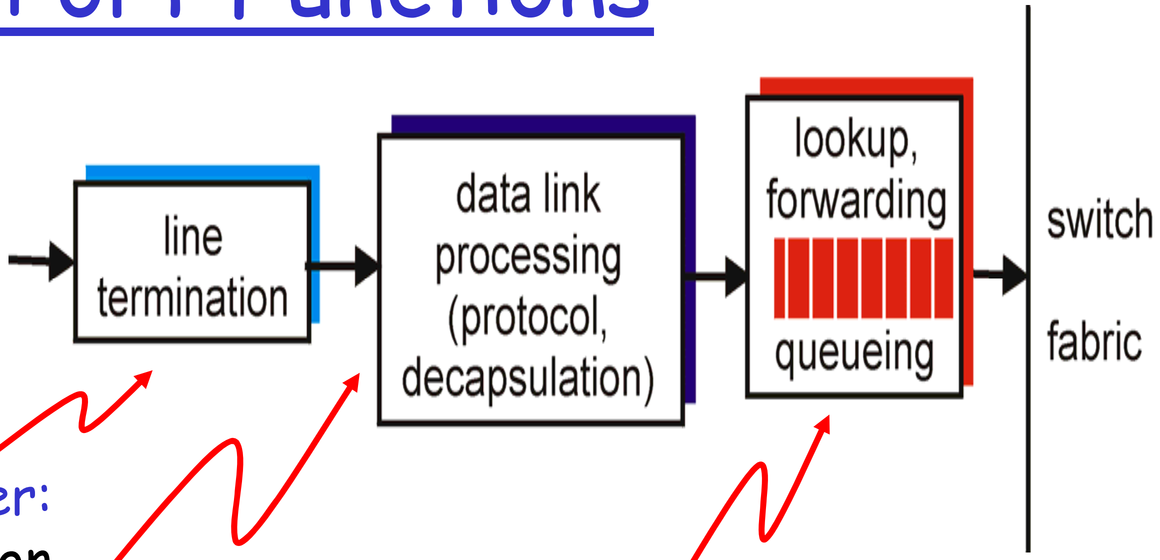
Router Architecture Overview

Two key router functions:

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



Input Port Functions



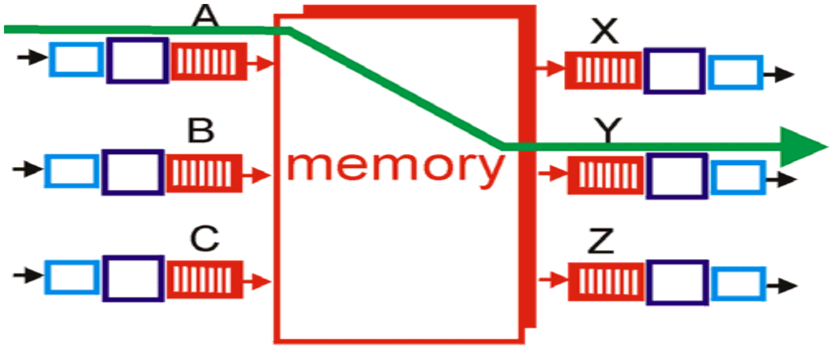
Physical layer:
bit-level reception

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

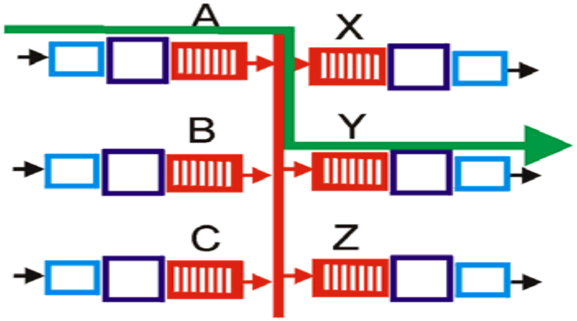
Decentralized switching:

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

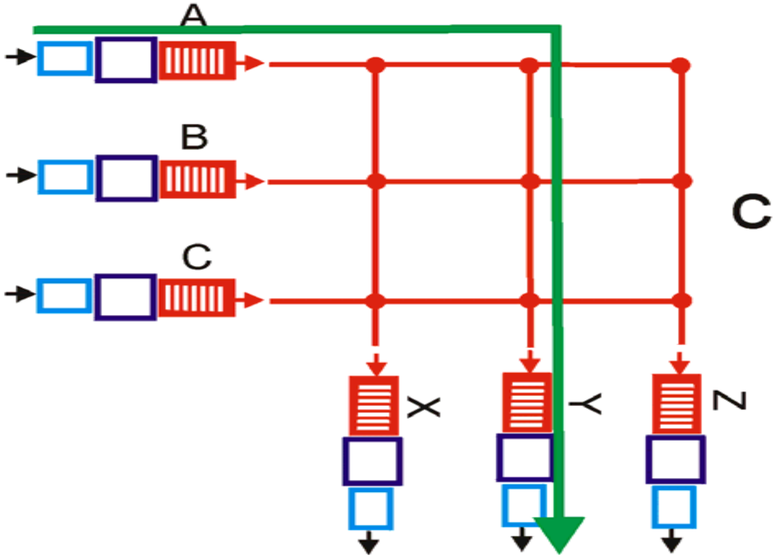
Three types of switching fabrics



memory



bus

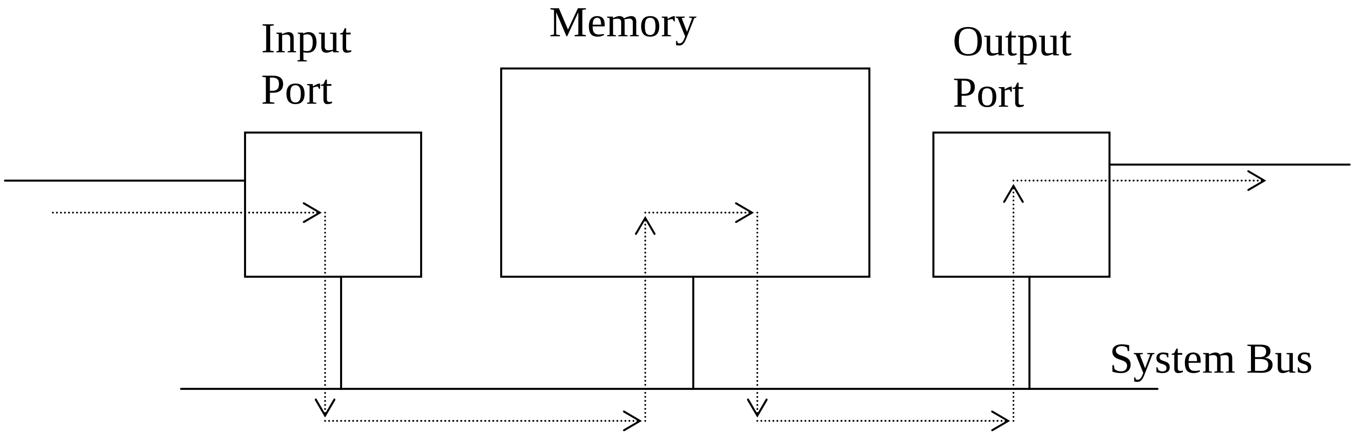


crossbar

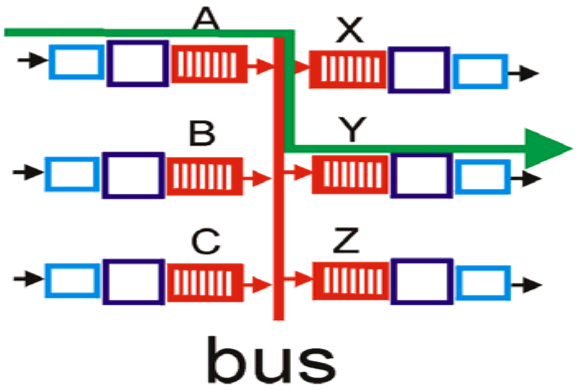
Switching Via Memory

First generation routers:

- traditional computers with switching under direct control of CPU
- packet copied to system's memory
- speed limited by memory bandwidth (2 bus crossings per datagram)



Switching Via a Bus

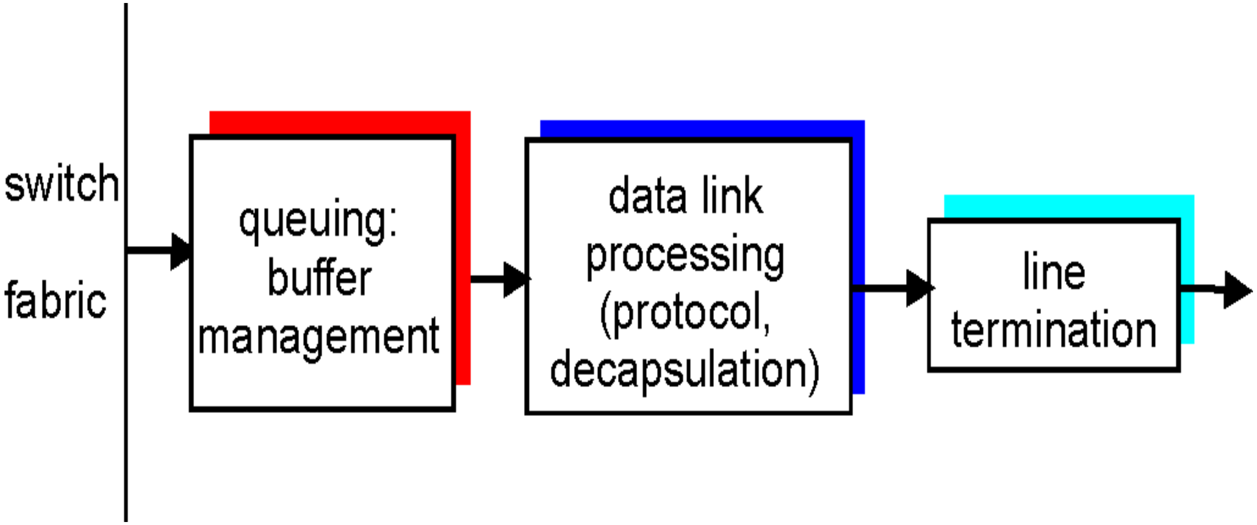


- datagram from input port memory to output port memory via a shared bus
- bus contention: switching speed limited by bus bandwidth
- 1 Gbps bus, Cisco 1900: sufficient speed for access and enterprise routers (not regional or backbone)

Switching Via An Interconnection Network

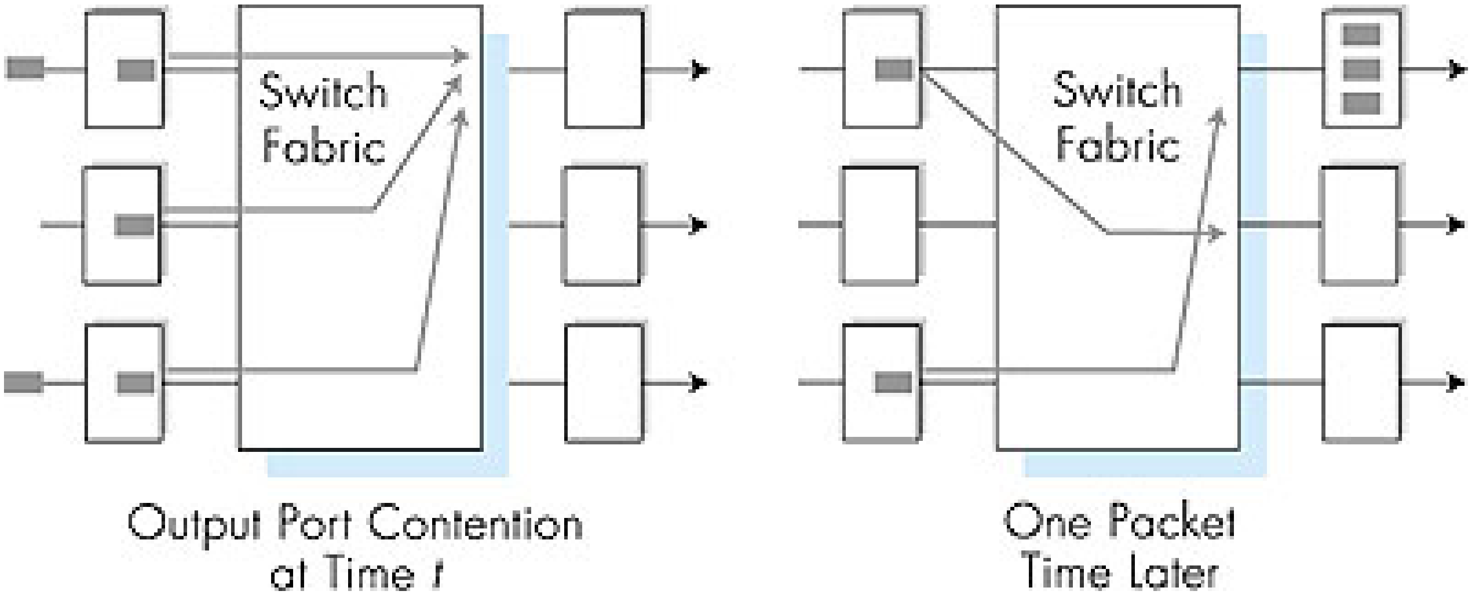
- overcome bus bandwidth limitations
- Banyan networks, other interconnection nets initially developed to connect processors in multiprocessor
- Advanced design: fragmenting datagram into fixed length cells, switch cells through the fabric.
- Cisco 12000: switches Gbps through the interconnection network

Output Ports



- *Buffering* required when datagrams arrive from fabric faster than the transmission rate
- *Scheduling discipline* chooses among queued datagrams for transmission

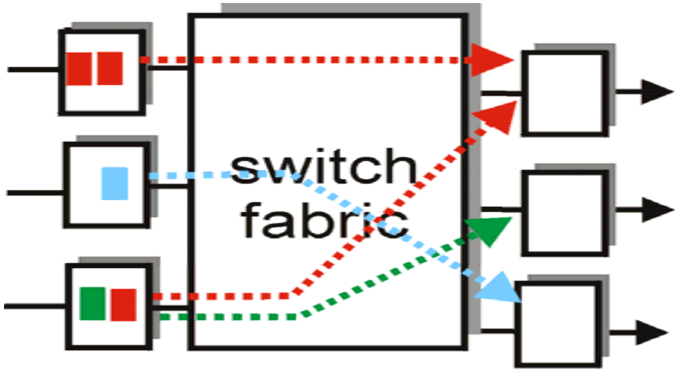
Output port queueing



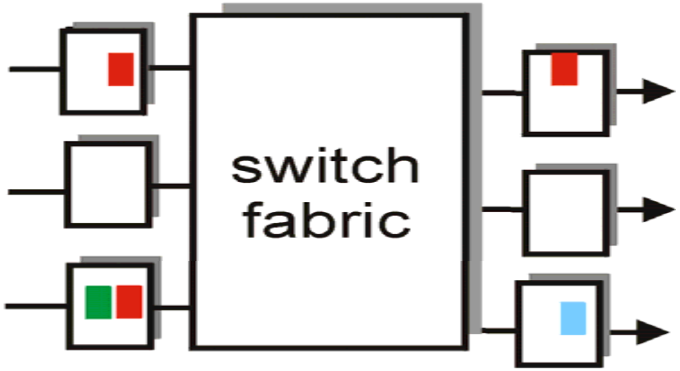
- buffering when arrival rate via switch exceeds output line speed
- *queueing (delay) and loss due to output port buffer overflow!*

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
- *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