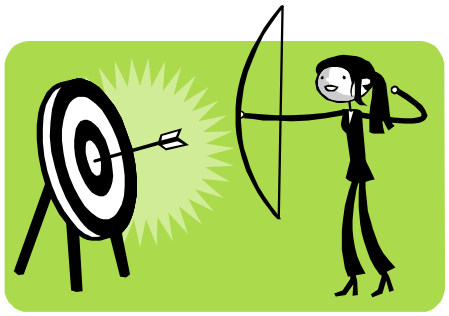
Chapter 5: Network Layer

Chapter goals:

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

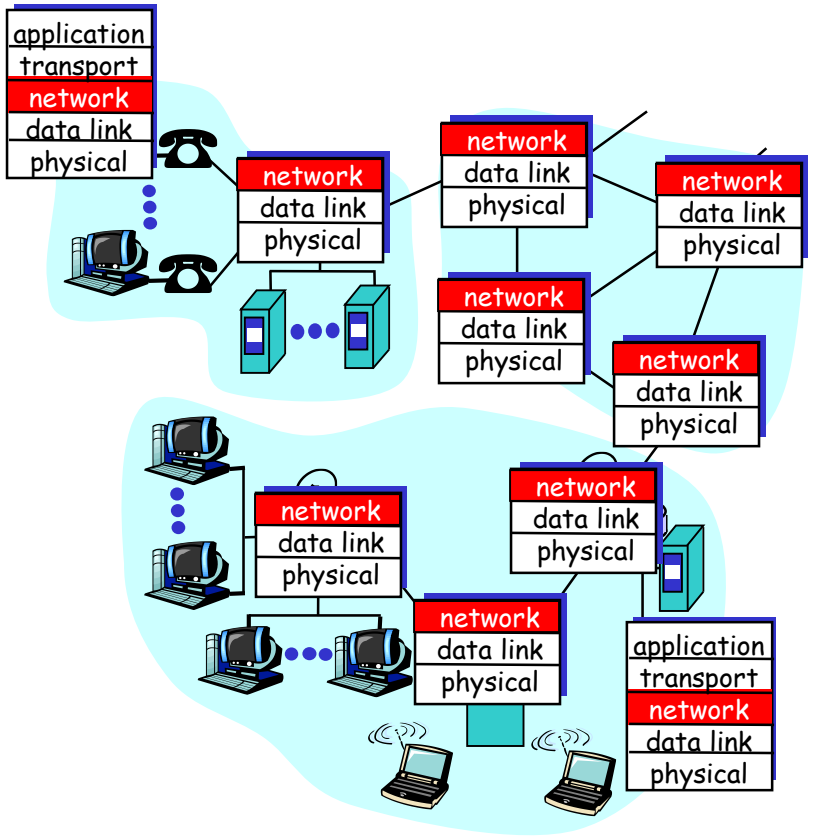


Chapter 5: Network Layer

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

Network layer

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



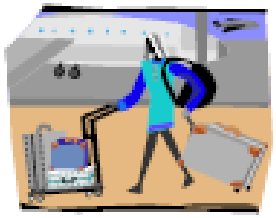
Two key Network-layer functions

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

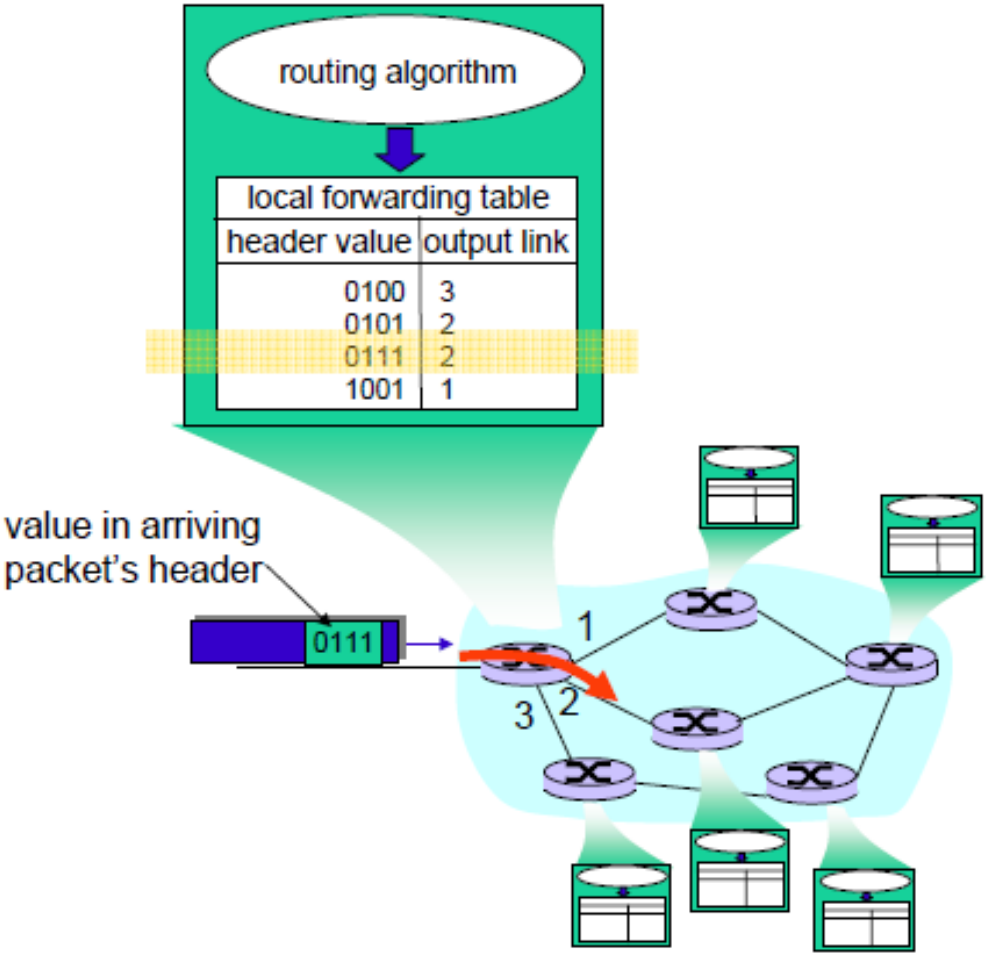
Analogy:

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

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



Interplay between routing and forwarding



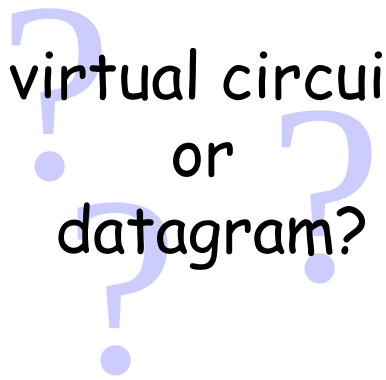
Network service model

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

service abstraction

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

The most important abstraction provided by network layer:



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
  - Implementation: in network core

Chapter 5: Network Layer

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

Virtual circuits

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

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

- call setup -> data flow -> call teardown
- each packet carries VC identifier (not destination host OD)
- *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.

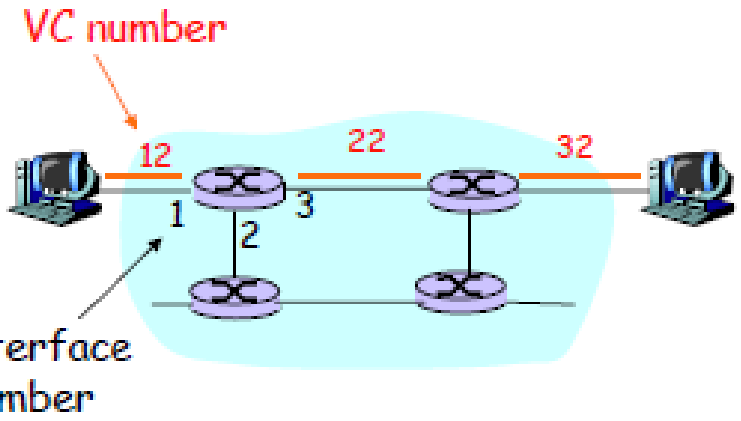
VC Implementation

A VC consists of:

1. Path from source to destination
2. VC numbers, one number for each link along path
3. Entries in forwarding tables in routers along path

- Packet belonging to VC carries VC number(rather than dest. address)
- VC number can be changed on each link
  - New VC number comes from forwarding table

Forwarding Table



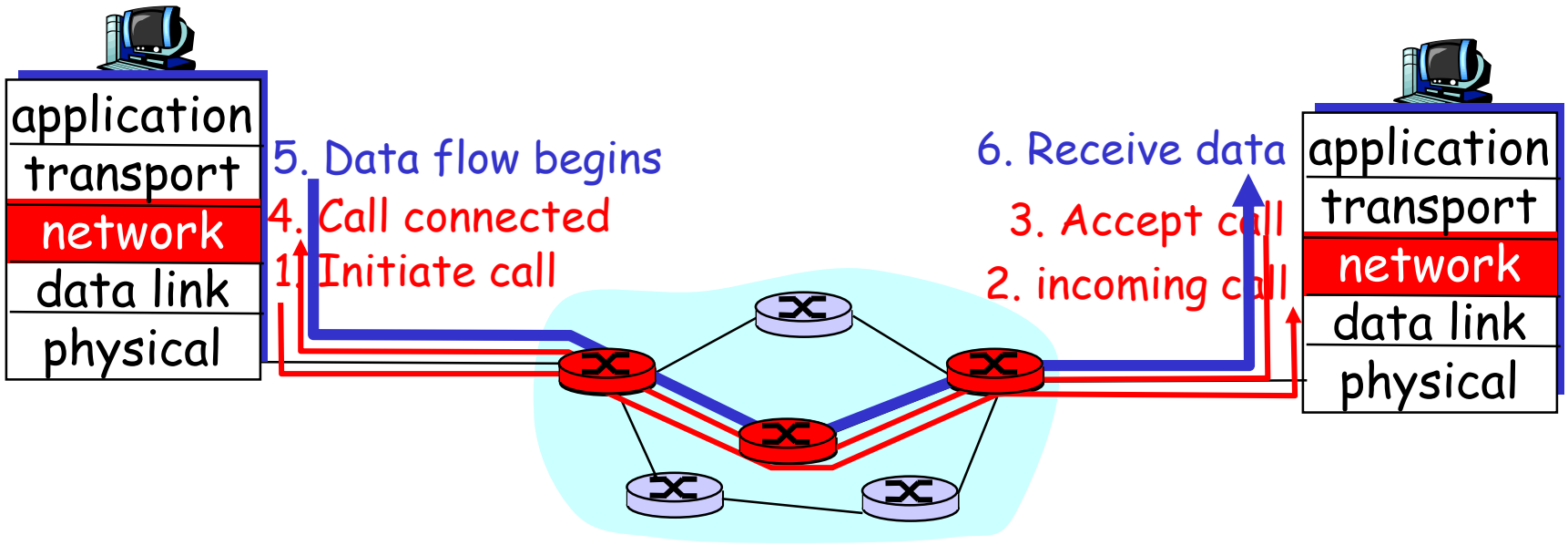
Forwarding table in northwest router:

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

Routers maintain connection state information!

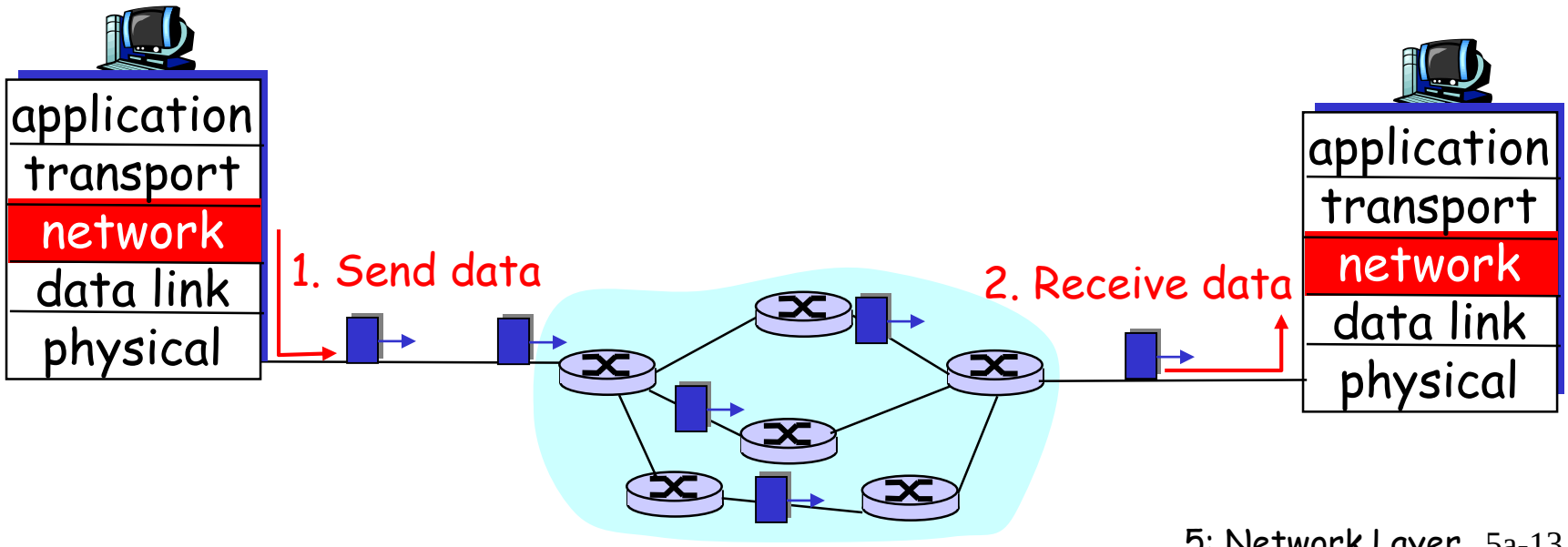
Virtual circuits: signaling protocols

- 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 typically routed using destination host ID
  - packets between same source-dest pair may take different paths



Forwarding Table

4 billion possible entries

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

Longest prefix matching

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

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

examples:

DA: 11001000 00010111 00010110 10100001 which interface?

DA: 11001000 00010111 00011000 10101010 which interface?

Datagram or VC network: why?

Internet

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

ATM

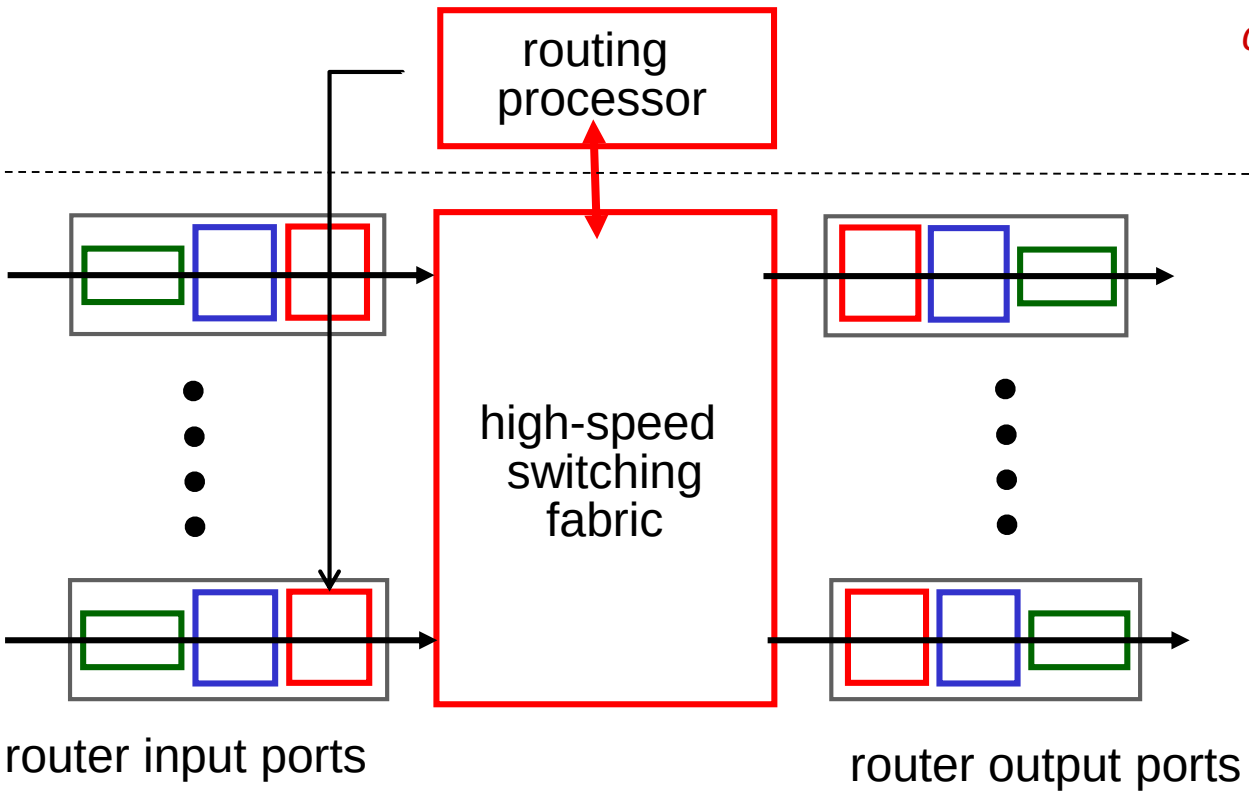
- evolved from telephony
- human conversation:
  - strict timing, reliability requirements
  - need for guaranteed service
- "dumb" end systems
  - telephones
- complexity inside network

Chapter 5: Network Layer

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

Router architecture overview

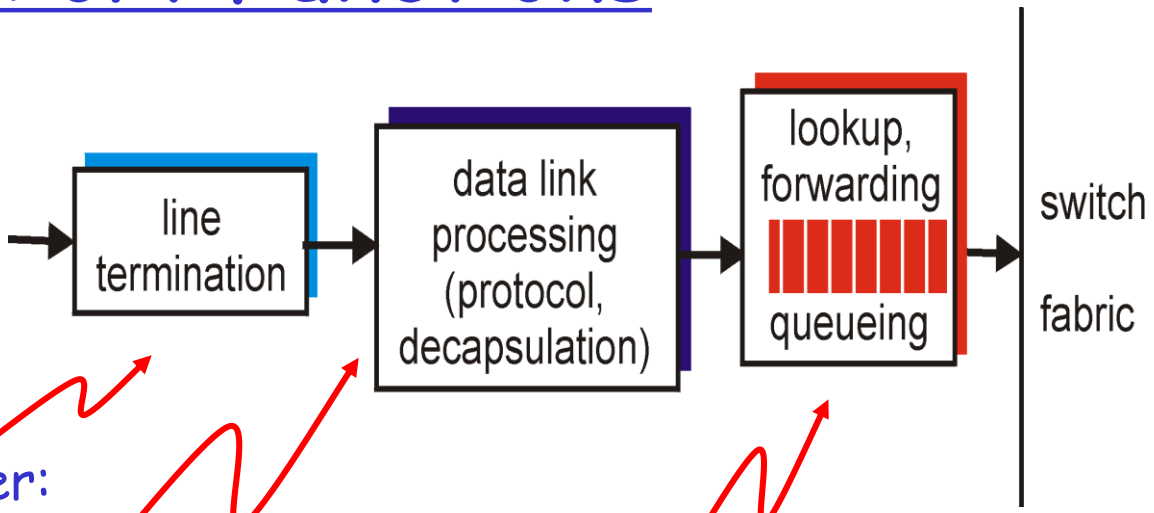
- high-level view of generic router architecture:



routing, management control plane (software) operates in millisecond time frame

forwarding data plane (hardware) operates in nanosecond timeframe

Input Port Functions



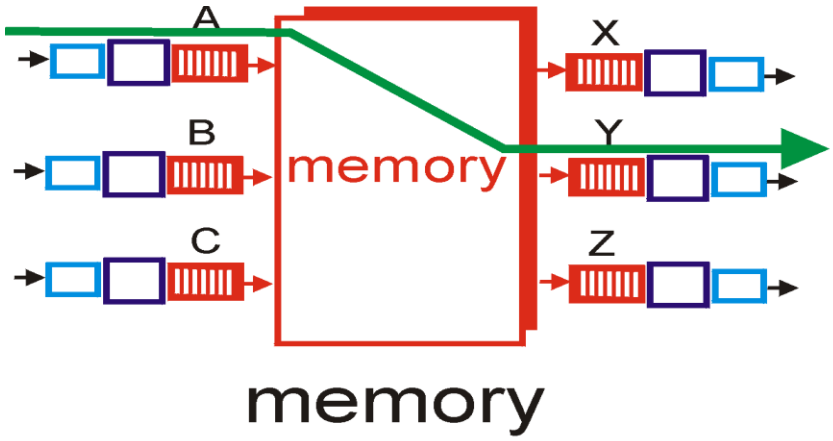
Physical layer:
bit-level reception

Data link layer:
e.g., Ethernet

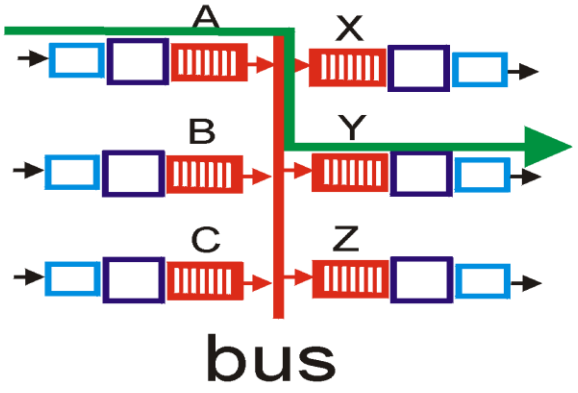
Decentralized switching:

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

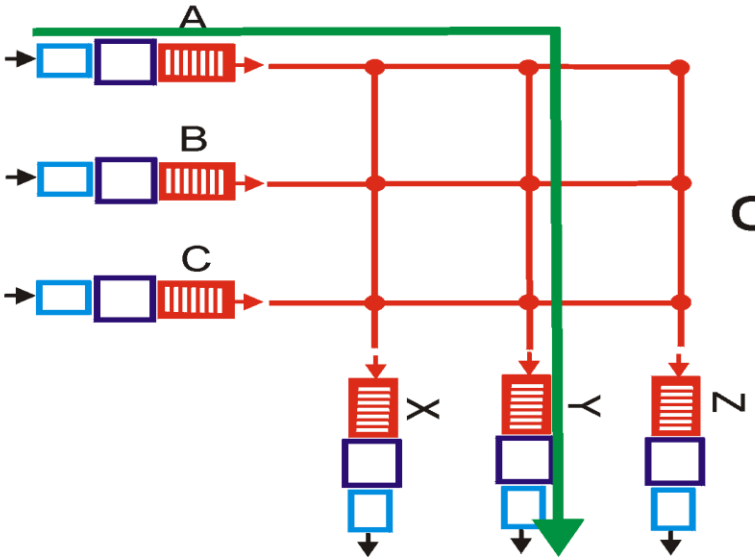
Three types of switching fabrics



memory



bus

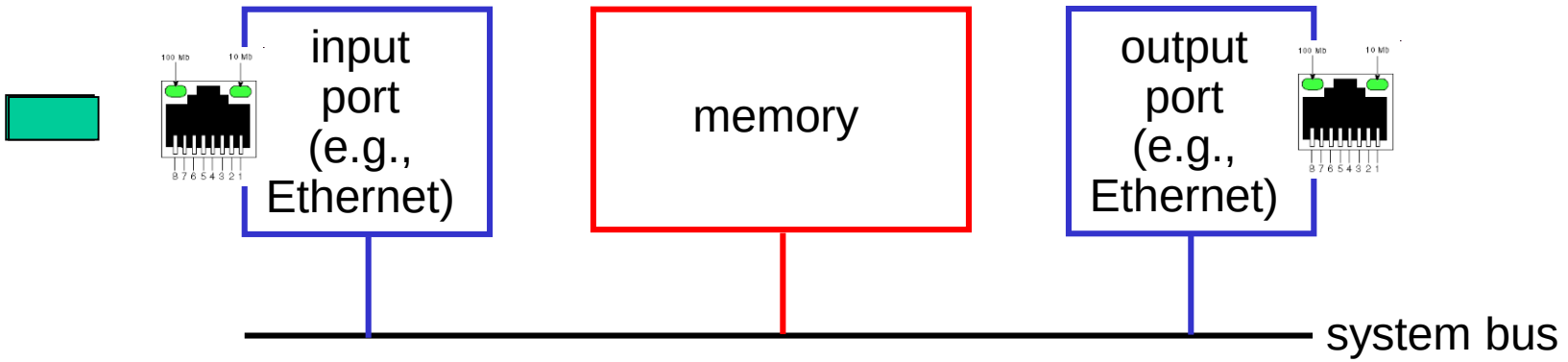


crossbar

Switching Via Memory

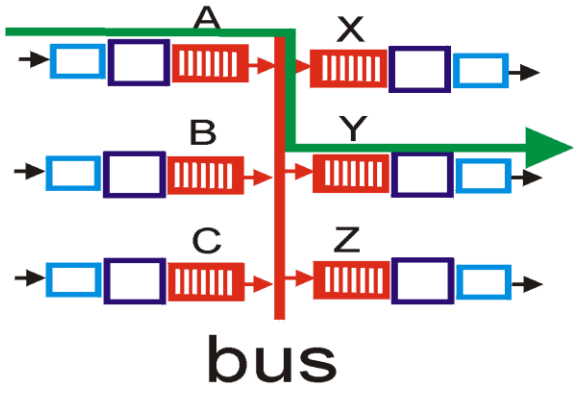
First generation routers:

- 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)



Later routers: E.g., Cisco Catalyst 8500 – Retired

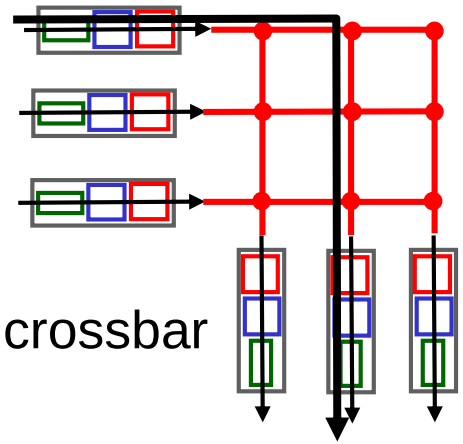
Switching Via Bus



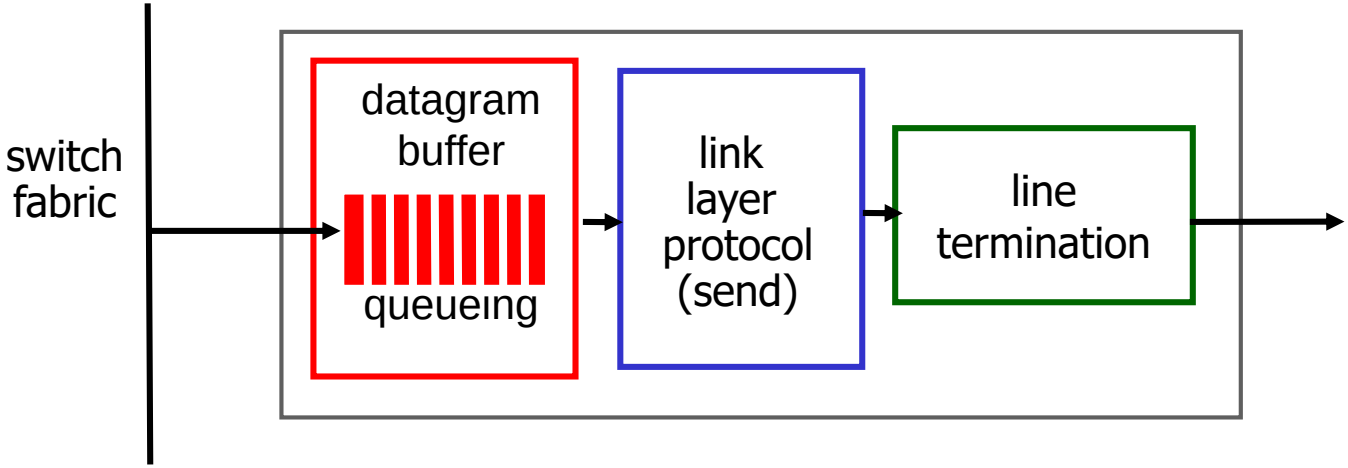
- datagram from input port to output port via a shared bus
- bus contention: switching speed limited by bus bandwidth
- 32 Gbps bus, Cisco 5600: sufficient speed for access and enterprise routers (not regional or backbone)

Switching Via Interconnection Network

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



Output Ports



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

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

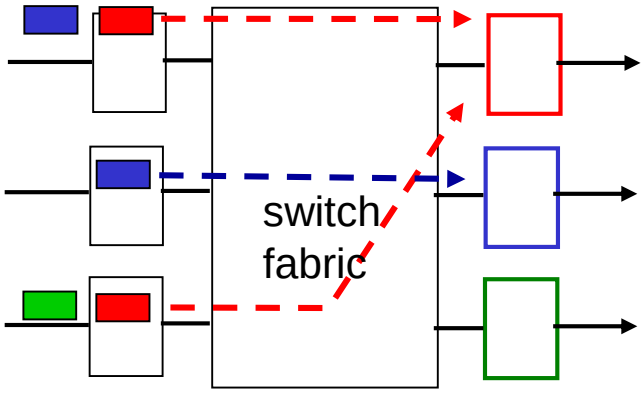
- *Scheduling discipline* chooses among queued datagrams for transmission

Priority scheduling – who gets best performance, network neutrality

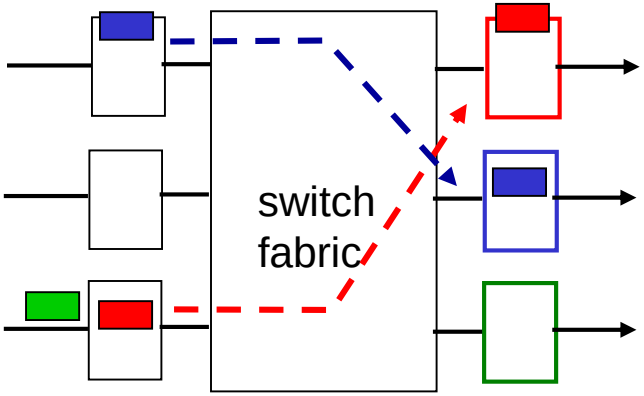
Input Port Queuing

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

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

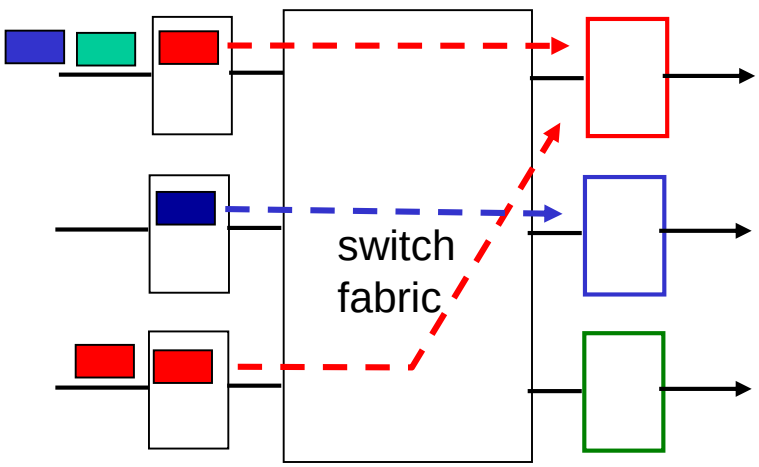


output port contention:
only one red datagram can be transferred.
lower red packet is blocked

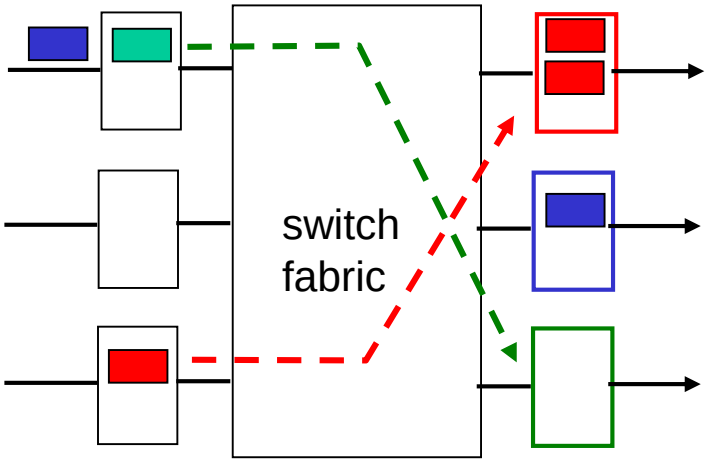


one packet time later:
green packet
experiences HOL
blocking

Output port queueing



at *t*, packets more from input to output



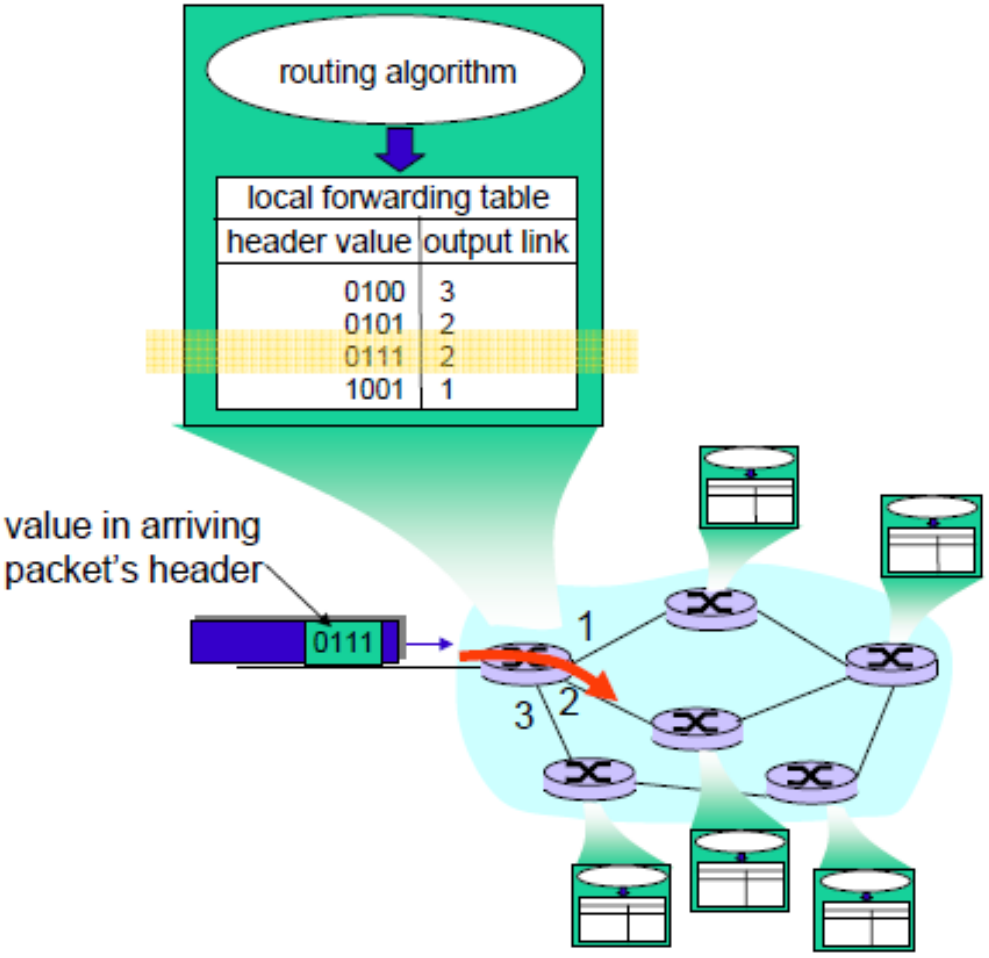
one packet time later

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

Chapter 5: Network Layer

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

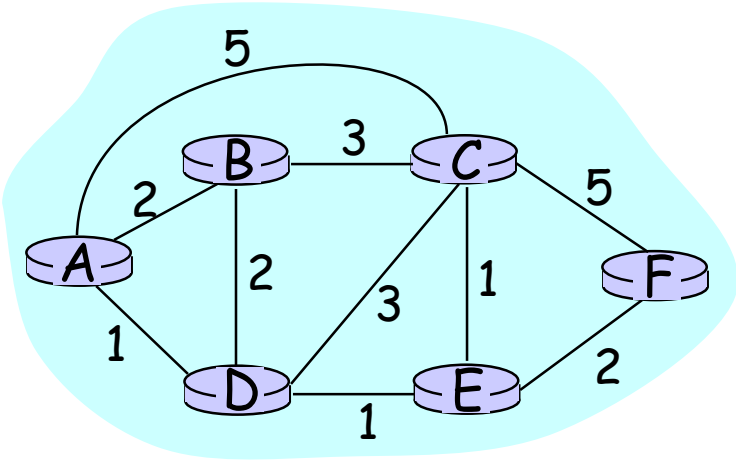
Interplay between routing and forwarding



Routing

Routing protocol

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

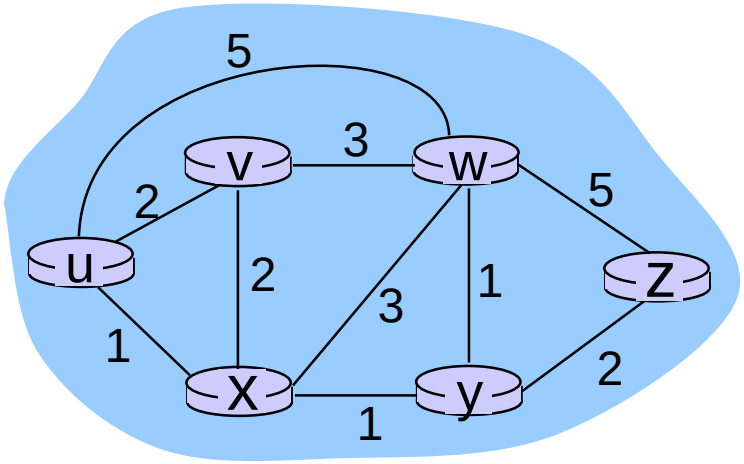


Graph abstraction for routing algorithms:

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

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

Graph abstraction of the network



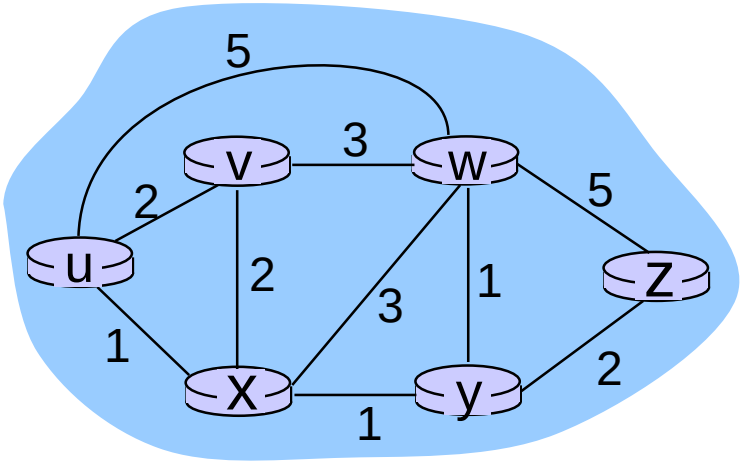
graph: G = (N,E)

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

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

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

Graph abstraction: costs



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

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

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

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

Routing Algorithm classification

Global or decentralized information?

Global:

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

Decentralized:

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

Static or dynamic?

Static:

- routes change slowly over time

Dynamic:

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

Dijkstra's algorithm

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

Notation:

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

Dijkstra's algorithm

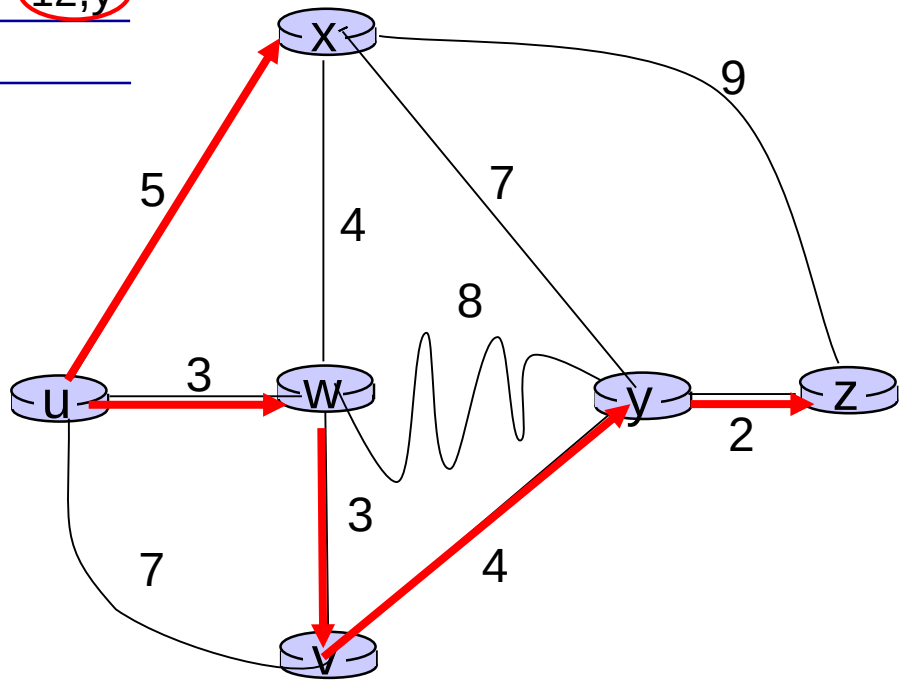
```
1  Initialization:
2    N' = {u}
3    for all nodes v
4      if v adjacent to u
5          then D(v) = c(u,v)
6      else D(v) = ∞
7
8  Loop
9     find w not in N' such that D(w) is a minimum
10    add w to N'
11    update D(v) for all v adjacent to w and not in N' :
12       D(v) = min( D(v), D(w) + 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	N'	D(v) p(v)	D(w) p(w)	D(x) p(x)	D(y) p(y)	D(z) p(z)
0	u	7,u	3,u	5,u	∞	∞
1	uw	6,w		5,u	11,w	∞
2	uwx	6,w			11,w	14,x
3	uwxv				10,v	14,x
4	uwxvy					12,y
5	uwxvyz					

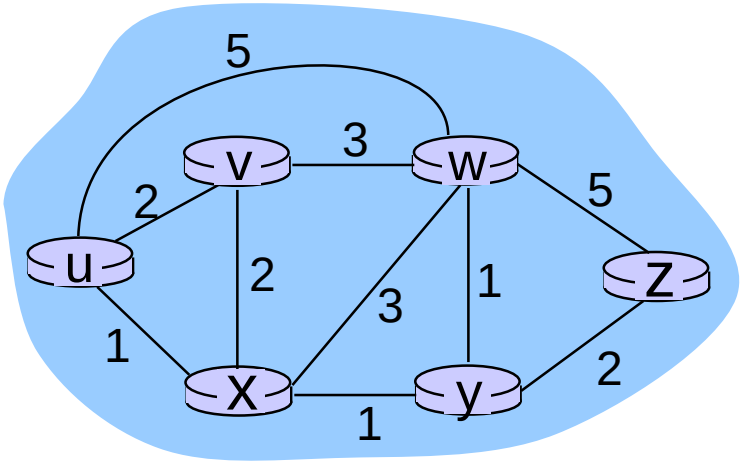
notes:

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



Dijkstra's algorithm: another example

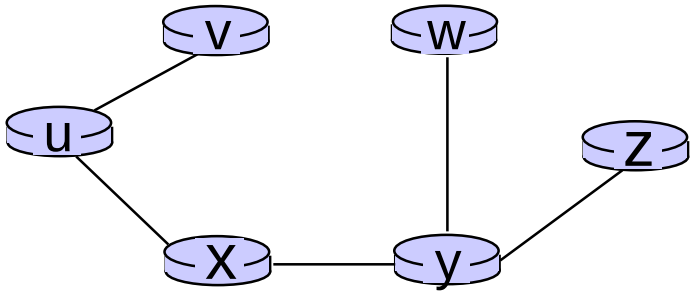
Step	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					



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

Dijkstra's algorithm: example (2)

resulting shortest-path tree from u:



resulting forwarding table in u:

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

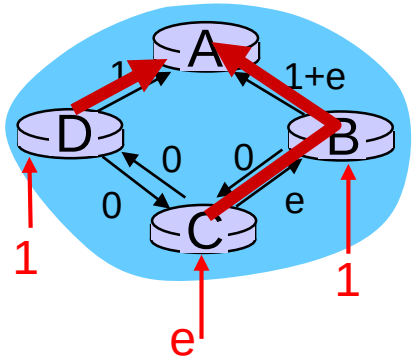
Dijkstra's algorithm,discussion

Algorithm complexity: n nodes

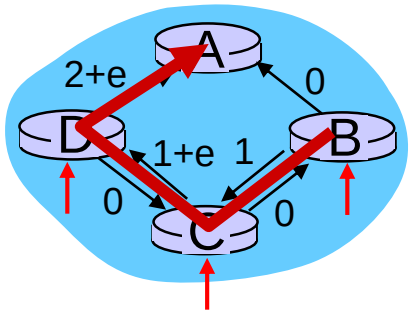
- Each iteration: need to check all nodes, w, not in N'
- n(n+1)/2 comparisons: $O(n^2)$
- More efficient implementations possible: O(nlogn)

Oscillations possible:

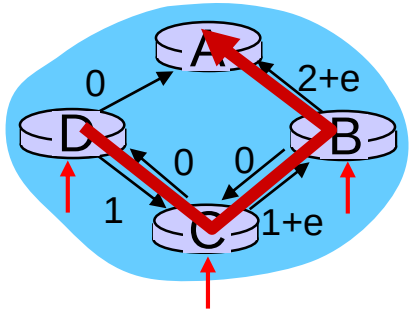
- E.g., link cost = amount of carried traffic



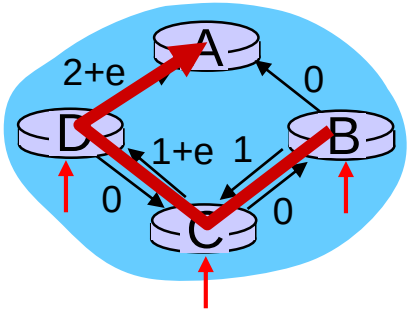
initially



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



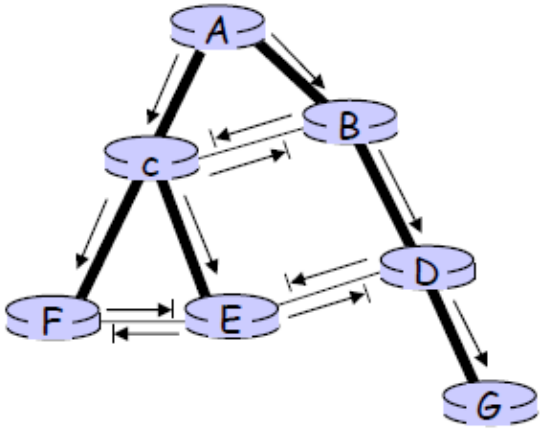
given these costs, find new routing…. resulting in new costs



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

Broadcast routing

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



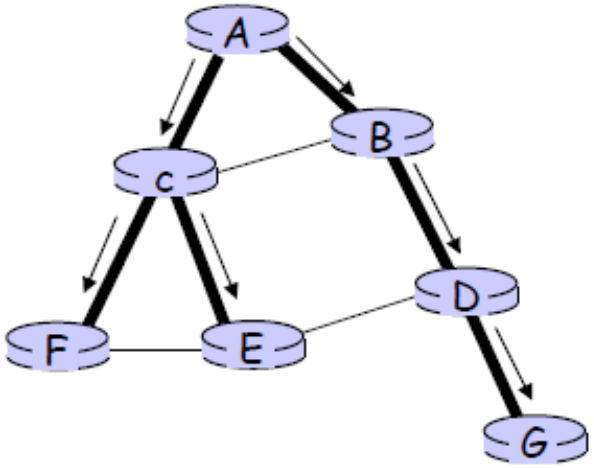
RFP: Reverse path forwarding

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

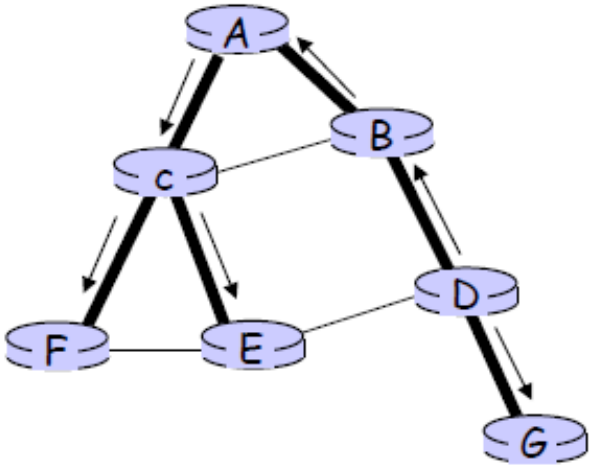
Broadcast routing

- Spanning-tree broadcast
  - Suppose you have a connected undirected graph
  - …then a spanning tree of the graph is a connected subgraph in which there are no cycles

✓ Be able to avoid the transmission of redundancy broadcast packets



(a) Broadcast initiated at A



(b) Broadcast initiated at D

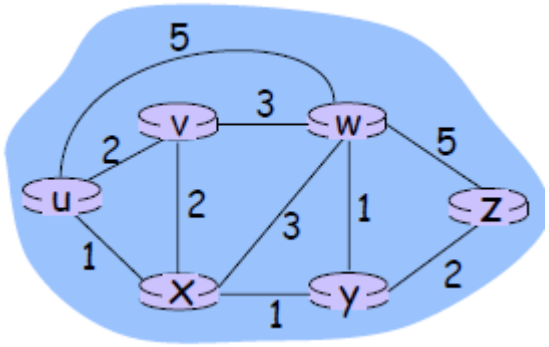
Link state algorithm

Broadcast routing + Dijkstra's algorithm

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

Distance vector algorithm

Bellman-Ford equation (dynamic programming)



Define

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

Then

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

where min is taken over all neighbors v of x

For example:

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

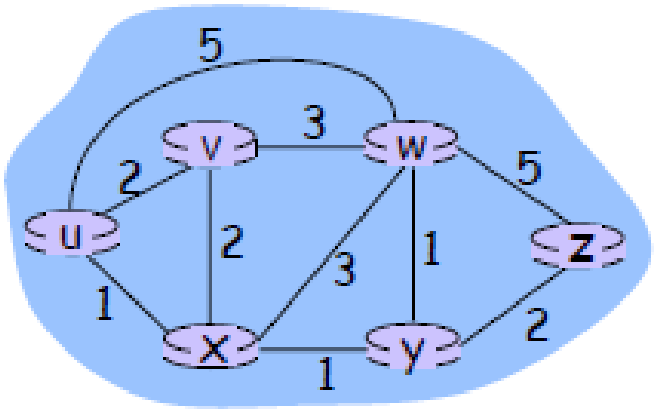
B-F equation says:

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

Distance vector algorithm

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

Bellman-Ford Example



Distance vectors stored at node x

Routing table at node x

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

Cost to

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

Distance vector algorithm

Basic idea:

- Each node periodically sends its own distance vector estimate to neighbors

- When a node x receives new DV estimate from neighbor, it updates its own DV using B-F equation

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

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

Distance vector algorithm

Iterative, asynchronous:

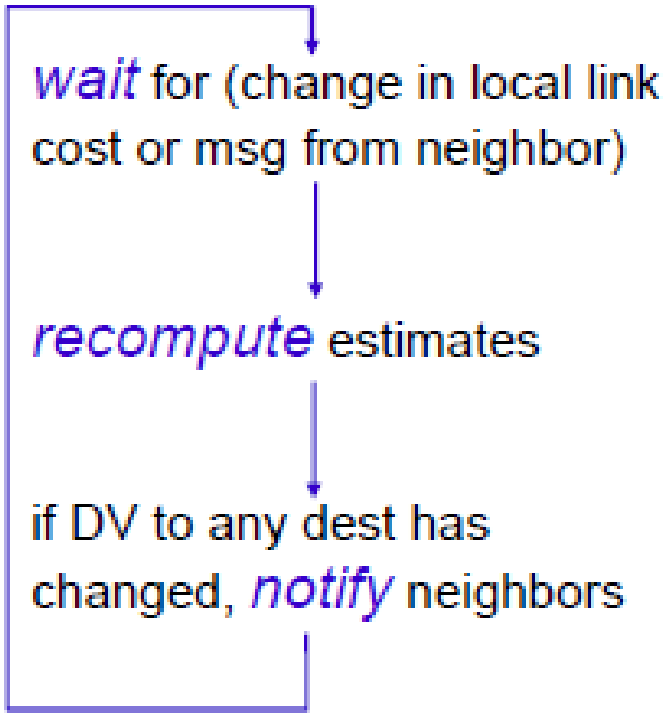
Each local iteration caused by:

- Local link cost change
- DV update message from neighbor

Distributed:

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

Each node:



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

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

node x table

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

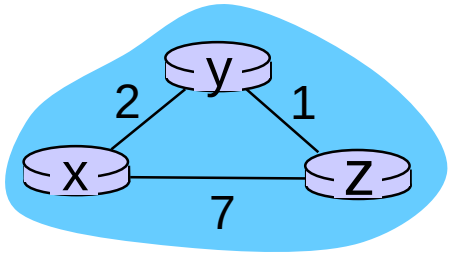
from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

node y table

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

node z table

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



time

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

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

node x table

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

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

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

node y table

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

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

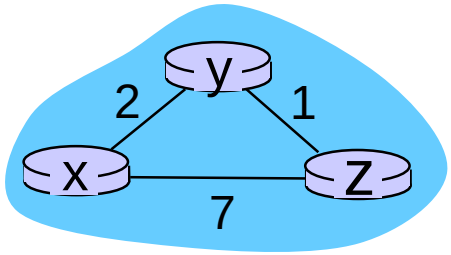
from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

node z table

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

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

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



time

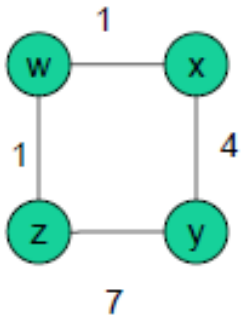
Distance vector: link cost changes

Link cost changes:

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

"good news travels fast, bad news travels slow"

"good news travels fast"



node w table

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

node x table

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

node y table

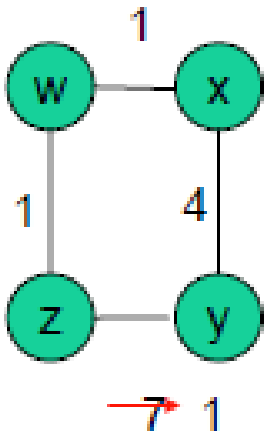
from \ cost to	x	y	z	w
y	4	0	6	5
x	0	4	2	1
z	2	6	0	1

node z table

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

"good news travels fast"

Algorithm converges in 3 steps.



node w table

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

node x table

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

node y table

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

node z table

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

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

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

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

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

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

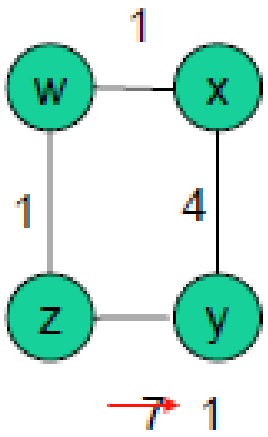
from	cost to x	y	z	w
x	0	3	2	1
w	1	2	1	0
y	3	0	1	2

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

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

"good news travels fast"

Algorithm converges in 3 steps.



node w table

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

node x table

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

node y table

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

node z table

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

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

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

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

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

Count to infinity problem

Link cost changes:

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

- for example:

  44 iterations before algorithm stabilizes

"bad news travels slow"

"count to infinity" problem

node x table

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

node y table

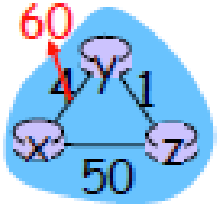
from \ cost to	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

node z table

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

"bad news travels slow"

Algorithm converges in 44 steps.



Cost of link xy changes

node x table

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

node x table

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

node x table

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

node x table

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

node y table

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

node y table

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

node y table

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

node y table

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

. . . . . .

node z table

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

node z table

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

node z table

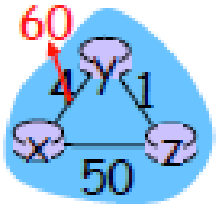
from \ cost to	x	y	z
x	0	51	50
y	6	0	1
z	7	1	0

node z table

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

"bad news travels slow"

Algorithm converges in 44 steps.



Cost of link xy changes

node x table

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

node x table

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

node x table

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

node x table

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

node y table

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

node y table

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

node y table

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

node y table

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

node z table

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

node z table

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

node z table

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

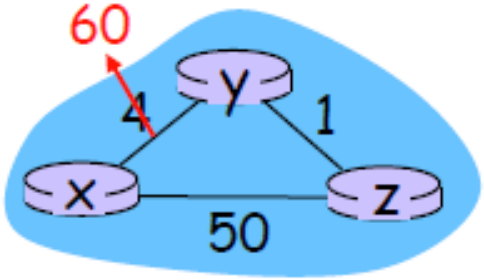
node z table

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

How to solve count to infinity problem?

Poisoned reverse:

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



node x table

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

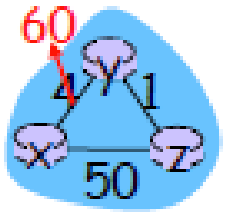
node y table

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

node z table

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

Algorithm converges in 3 steps.



node x table

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

node x table

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

node x table

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

node x table

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

node y table

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

node y table

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

node y table

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

node y table

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

node z table

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

node z table

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

node z table

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

node z table

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

Cost of link xy changes to 60

Comparison of LS and DV algorithms

Message complexity

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

Speed of convergence

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

Robustness: what happens if router malfunctions?

LS:

- Node can advertise incorrect link cost
- Each node computes only its own table

DV:

- DV node can advertise incorrect path cost
- Each node's table used by others
- Error propagate thru network

Hierarchical Routing

Our routing study thus far - idealization

- all routers identical
- network "flat"

… *not* true in practice

scale: with billions of destinations:

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

administrative autonomy

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

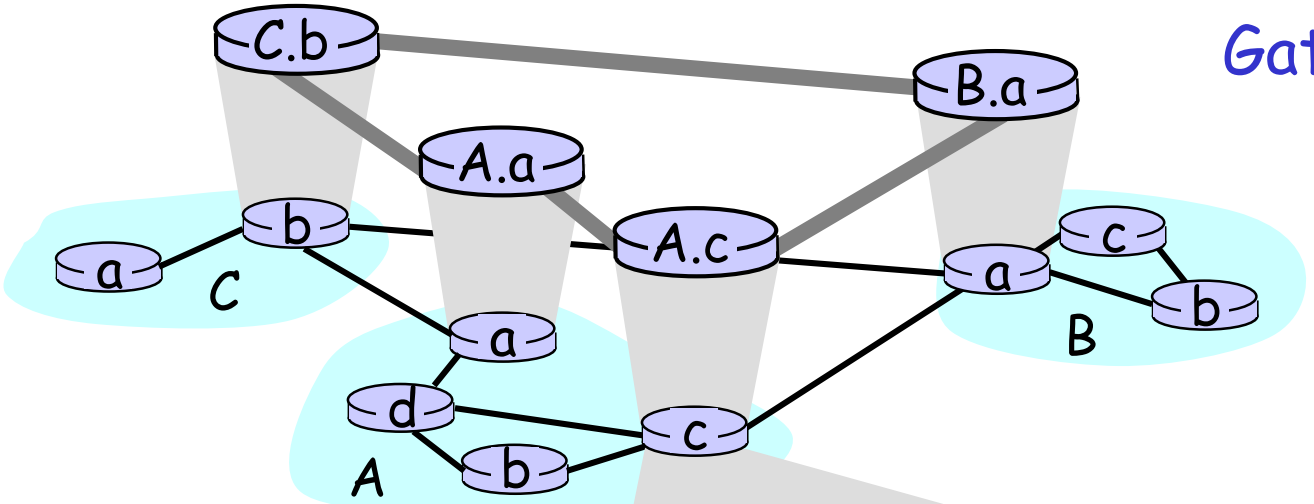
Hierarchical Routing

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

gateway routers

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

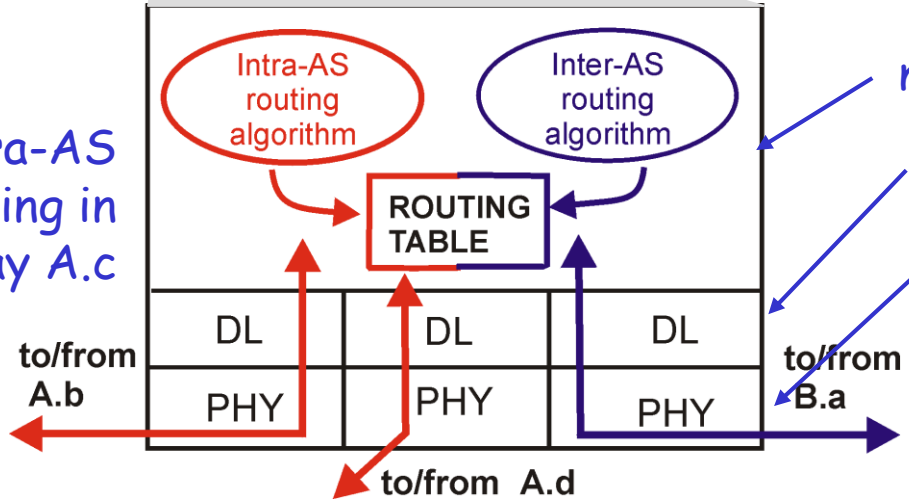
Intra-AS and Inter-AS routing



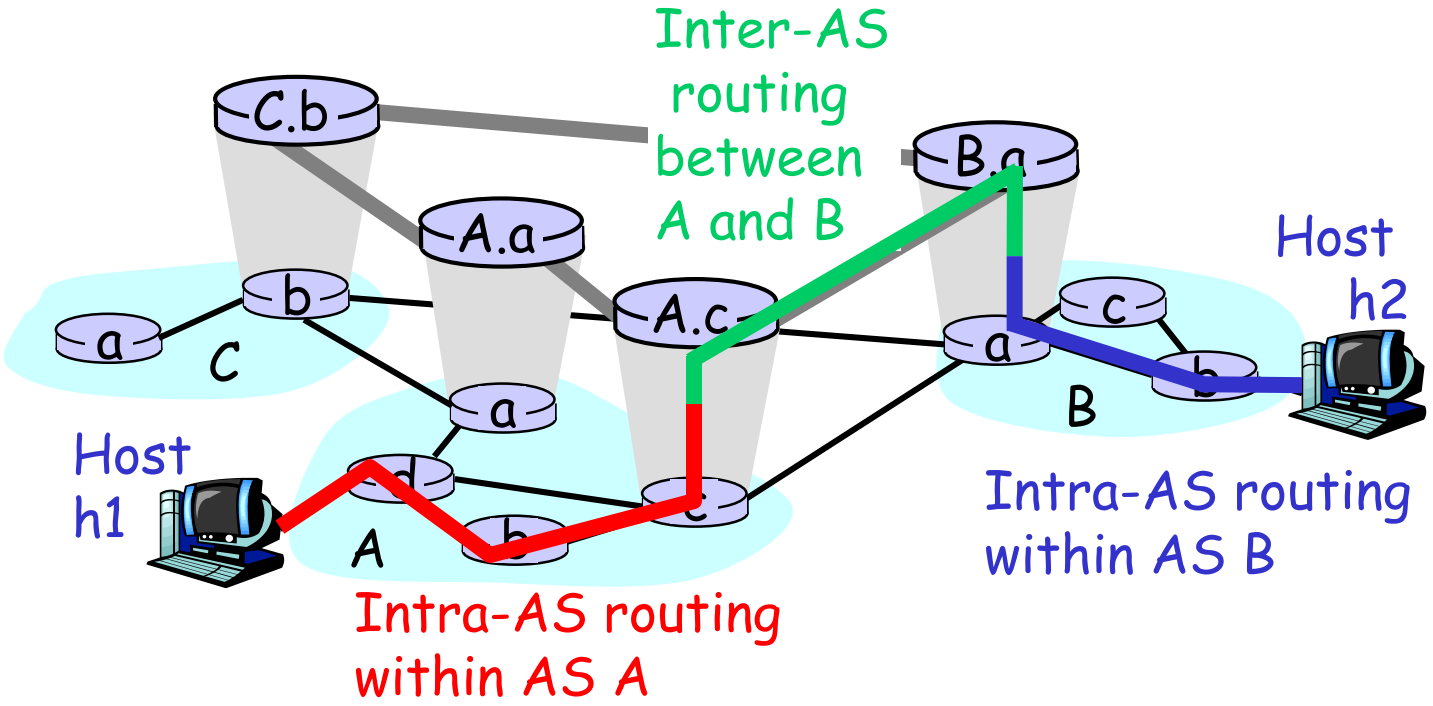
Gateways:

- 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



Intra-AS and Inter-AS routing



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

Chapter 5: Network Layer

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

Intra-AS Routing

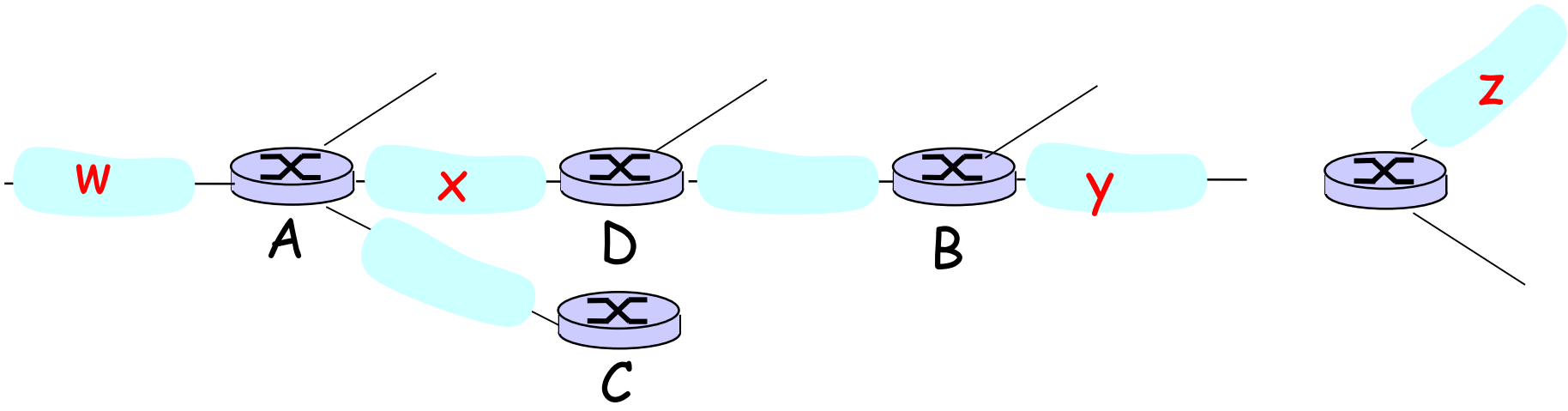
- Also known as Interior Gateway Protocols (IGP)
- Most common IGPs:
  - RIP: Routing Information Protocol
  - OSPF: Open Shortest Path First
  - IGRP: Interior Gateway Routing Protocol (Cisco propr.)

RIP (Routing Information Protocol)

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

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

RIP (Routing Information Protocol)



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

Routing table in D

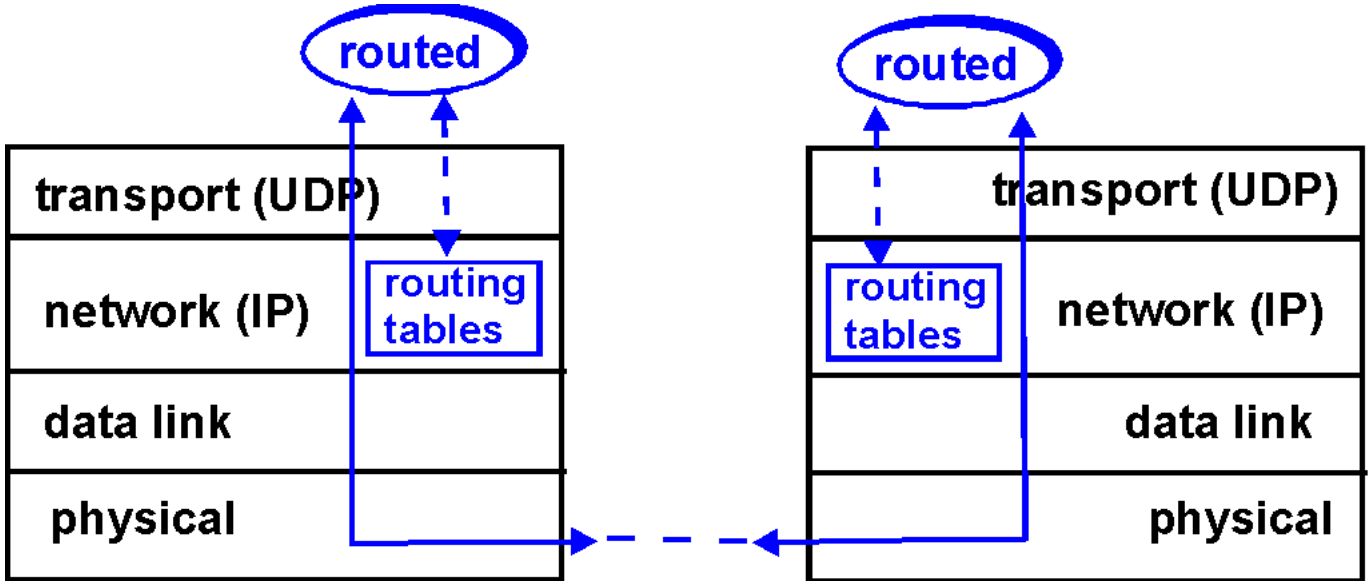
RIP: Link Failure and Recovery

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

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

RIP Table processing

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



RIP Table example (continued)

Router: *giroflee.eurocom.fr*

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

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

OSPF (Open Shortest Path First)

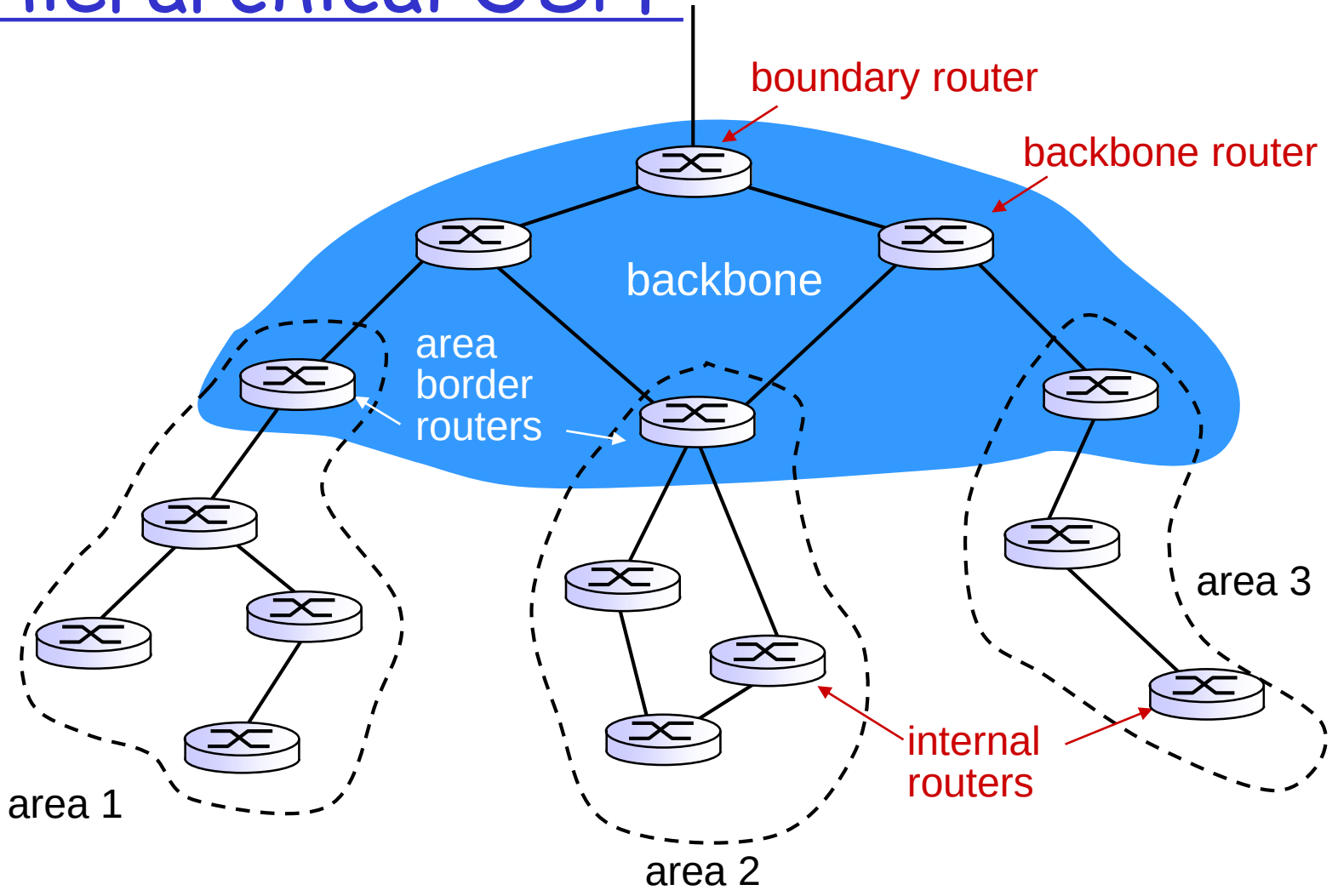
- "open": publicly available
- Uses Link State algorithm
  - LS packet dissemination
  - Topology map at each node
  - Route computation using Dijkstra's algorithm

- OSPF advertisement carries one entry per neighbor router
- Advertisements disseminated to entire AS (via flooding)

OSPF "advanced" features (not in RIP)

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

Hierarchical OSPF



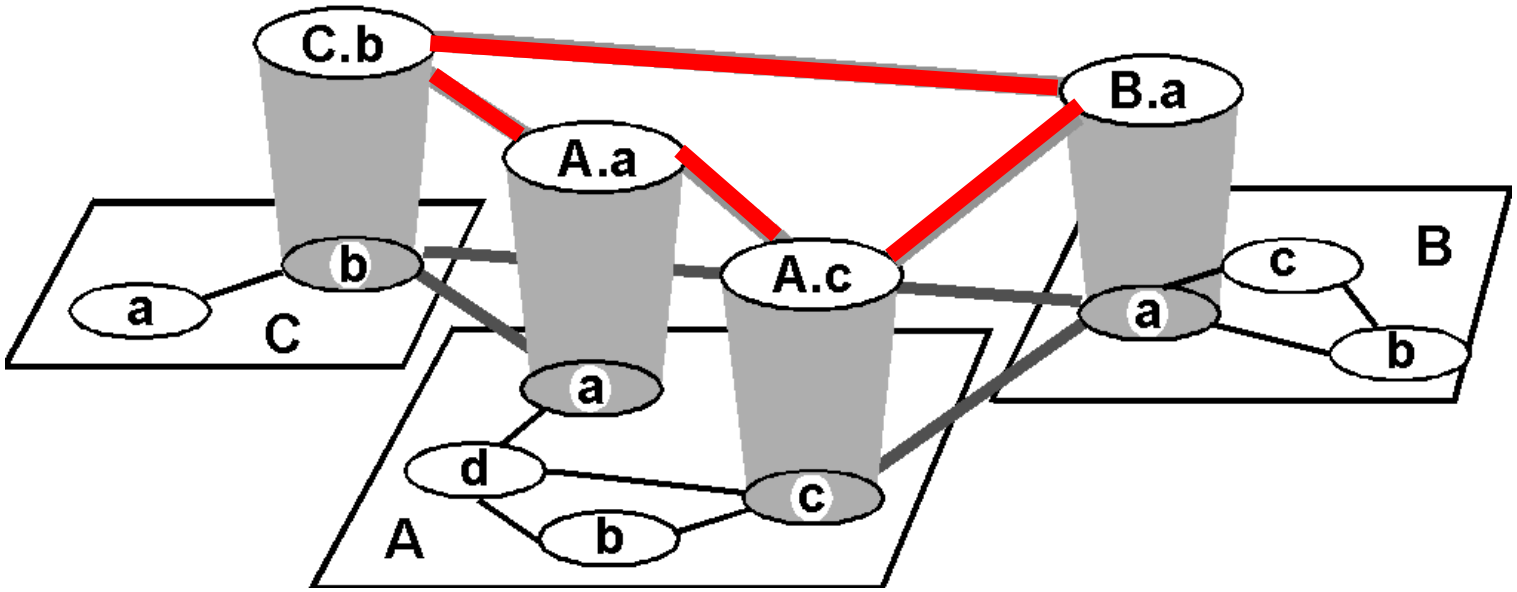
Hierarchical OSPF

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

IGRP (Interior Gateway Routing Protocol)

- CISCO proprietary; successor of RIP (mid 80s)
- Distance Vector, like RIP
- several cost metrics (delay, bandwidth, reliability, load etc)
- uses TCP to exchange routing updates
- Loop-free routing via Diffusing Update Algorithm (DUAL)

Inter-AS routing



Internet inter-AS routing: BGP

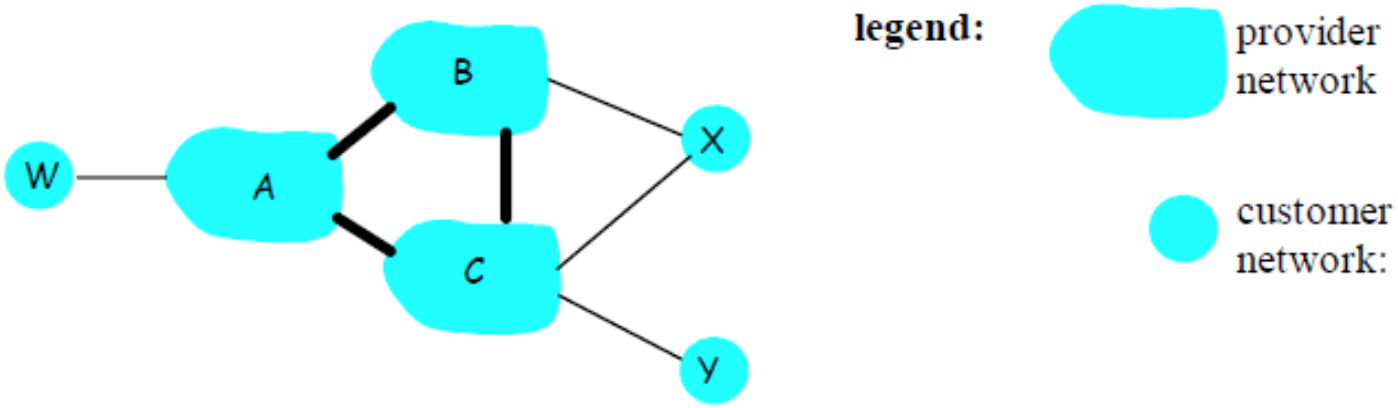
- BGP (Border Gateway Protocol): *the* de facto standard
- BGP provides each AS a means to:
  - Obtain subnet reachability information from neighboring ASs.
  - Propagate reachability information to all AS-internal routers.
  - Determine good routes to subnets based on reachability information and policy.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

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 (peers) *entire path* (i.e, sequence of ASs) to destination
  - E.g., Gateway X may send its path to dest. Z:

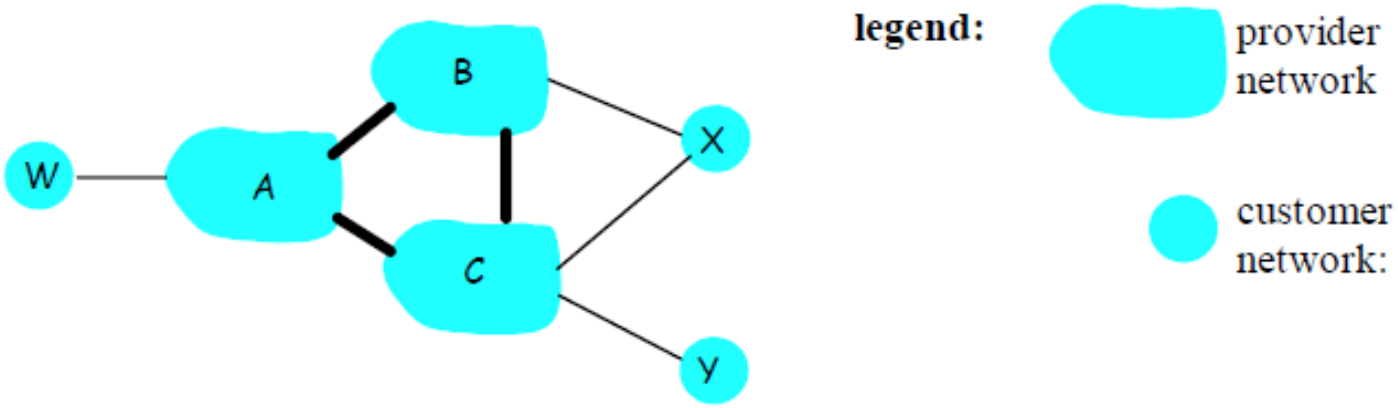
Path (X,Z) = X,Y1,Y2,Y3,…,Z

BGP Routing Policy



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

BGP Routing Policy



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

Why different Intra- and Inter-AS routing ?

Policy:

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

Scale:

- hierarchical routing saves table size, reduced update traffic

Performance:

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