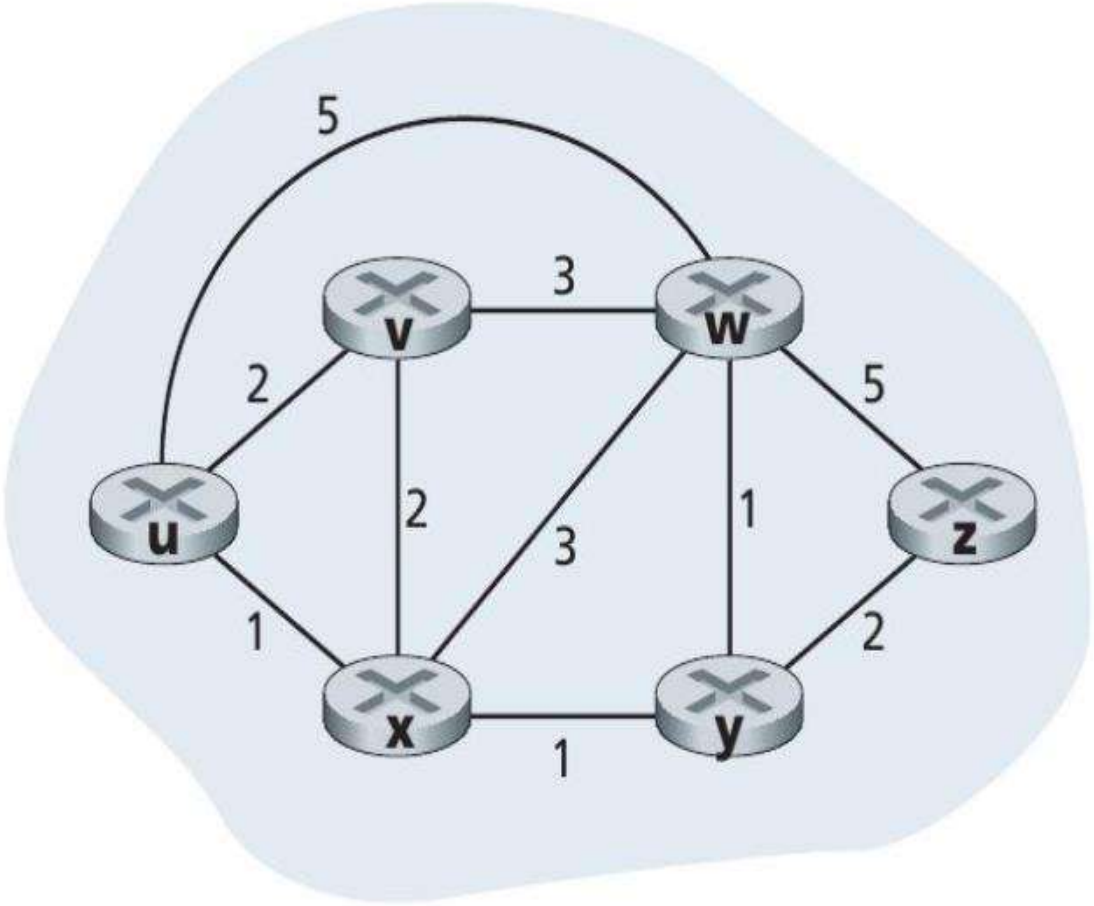
Routing

Notation



Notation

- We represent a network by an undirected graph $G = (N, E)$
- N is the set of Nodes (routers)
- E is the set of edges connecting nodes (links)
- $c(x, y)$ is the cost of the edge between x and y.
- Cost of a path $(x_1, \ldots, x_p)$ is sum of costs of edges along the path: $c(x_1, x_2) + \cdots + c(x_{p-1}, x_p)$
- We aim to find paths with least cost.

Classification

- Global vs Decentralized:
  - Global routing algorithm: requires global information about links and costs at every router. Also known as Link-State algorithm
  - Decentralized routing algorithm: no node has complete information
- Static vs Dynamic routing
- Load-sensitive vs Load-insensitive routing

Link-State Routing Algorithm

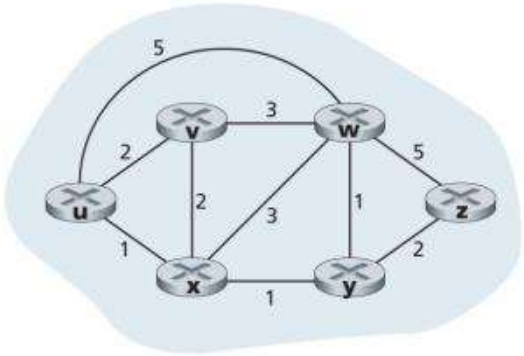
- We study Dijkstra's algorithm
- $D(v)$: cost of the least cost path from source to destination v as of this iteration
- $p(v)$: previous node along the current least cost path from the source to v
- N' : subset of N. If $v \in N$, then least cost path to v from source is definitely known.

LS Algorithm

```
1  Initialization:
2    N' = {u}
3    for all nodes v
4      if v is a neighbor of 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 each neighbor v of 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    least path cost to w plus cost from w to v */
15 until N'= N
```

LS Routing 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					



Distance-Vector (DV) Routing Algorithm

- Decentralized,asynchronous
- Iterative process
- $d_x(y)$ denotes cost of least cost path from x to y
- Bellman-Ford equation

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

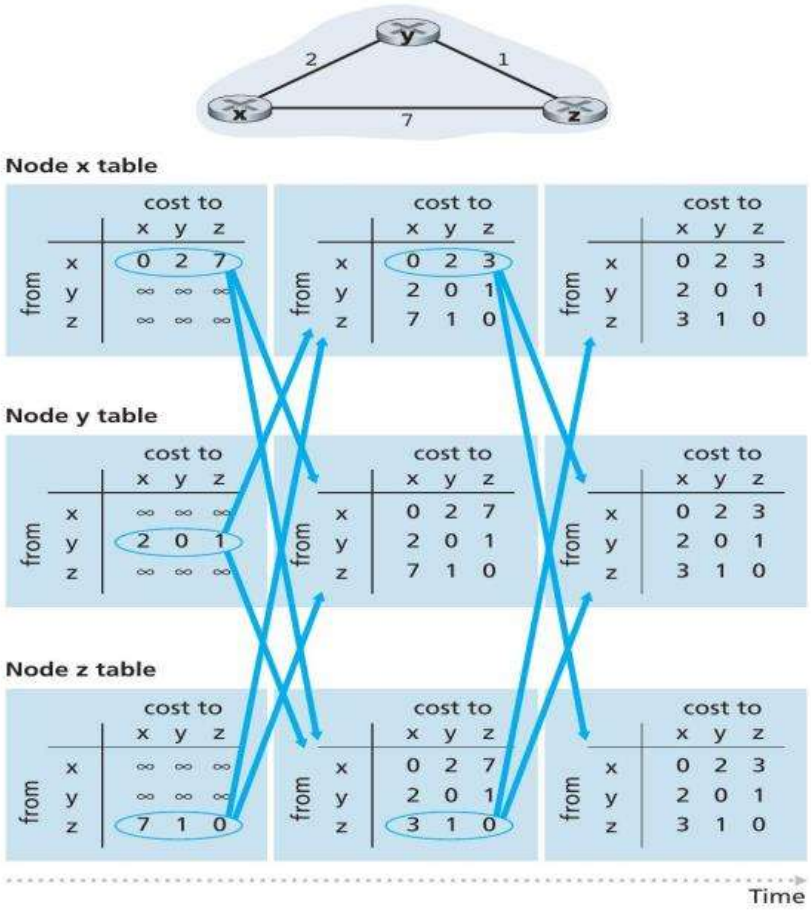
  v is a neighbor of x.
- Each node x maintains the following routing information:
  - For each neighbor v, the cost $c(x,v)$
  - Node x's distance vector, $\mathbf{D}_x = [D_x(y) : y \in N]$
  - Distance vectors of each of its neighbors $\mathbf{D}_v$

DV Algorithm

At each node, x:

```
1  Initialization:
2     for all destinations y in N:
3        D_x(y) = c(x,y)   /* if y is not a neighbor then c(x,y) = ∞ */
4     for each neighbor w
5        D_w(y) = ? for all destinations y in N
6     for each neighbor w
7        send distance vector D_x = [D_x(y): y in N] to w
8
9  loop
10    wait (until I see a link cost change to some neighbor w or
11          until I receive a distance vector from some neighbor w)
12
13    for each y in N:
14       D_x(y) = min_v{c(x,v) + D_v(y)}
15
16    if D_x(y) changed for any destination y
17       send distance vector D_x = [D_x(y): y in N] to all neighbors
18
19 forever
```

DV Example

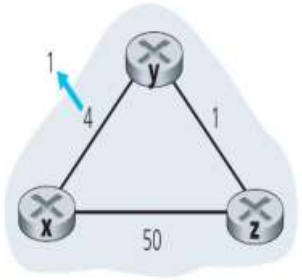


$D_x(x) = 0$

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

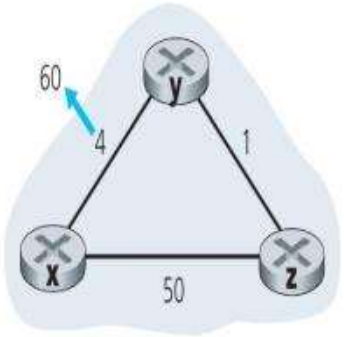
$D_x(z) = \min\{c(x,y) + D_y(z), c(x,z) + D_z(z)\} = \min\{2 + 1, 7 + 0\} = 3$

DV Algorithm: Link Cost Changes



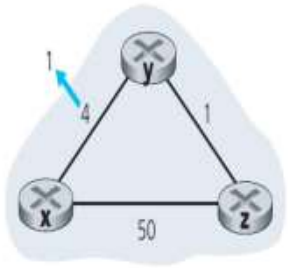
- Focus on distance tables entries of y and z to x
- At t_0, cost has changed to 1 from 4. y updates its table with $D_y(x) = 1$ and informs z
- At t_1, z receives update from y and updates its table $D_z(x) = 2$
- At t_2, y receives update from z and no changes in table.

DV Algorithm: Link Cost Changes



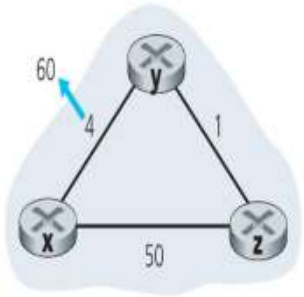
- Before link cost changes:
  $D_y(x) = 4, D_y(z) = 1, D_z(y) = 1, D_z(x) = 5$

DV Algorithm: Link Cost Changes



- Focus on distance tables entries of y and z to x
- At t_0, cost has changed to 1 from 4. y updates its table with $D_y(x) = 1$ and informs z
- At t_1, z receives update from y and updates its table $D_z(x) = 2$
- At t_2, y receives update from z and no changes in table.

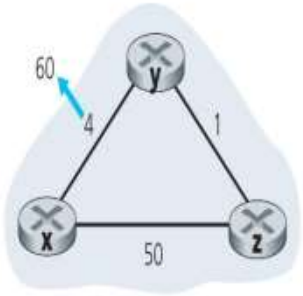
DV Algorithm: Link Cost Changes



- Before link cost changes:
  $D_y(x) = 4, D_y(z) = 1, D_z(y) = 1, D_z(x) = 5$
- At t_0, cost has changed to 60 from 4. y updates its table with

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

DV Algorithm: Link Cost Changes

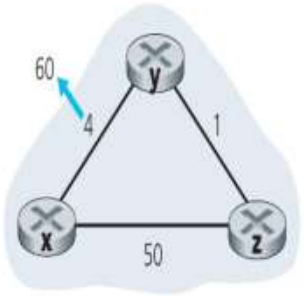


- Before link cost changes:
  $D_y(x) = 4, D_y(z) = 1, D_z(y) = 1, D_z(x) = 5$
- At t_0, cost has changed to 60 from 4. y updates its table with

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

- $D_y(x) = min\{60 + 0, 1 + 5\} = 6$

DV Algorithm: Link Cost Changes

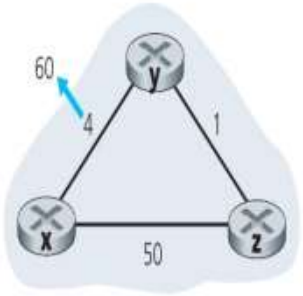


- Before link cost changes:
  $D_y(x) = 4, D_y(z) = 1, D_z(y) = 1, D_z(x) = 5$
- At t_0, cost has changed to 60 from 4. y updates its table with

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

- $D_y(x) = min\{60 + 0, 1 + 5\} = 6$
- At t_1, z receives update from y and updates its table
  $D_z(x) = min\{50 + 0, 1 + 6\} = 7$

DV Algorithm: Link Cost Changes

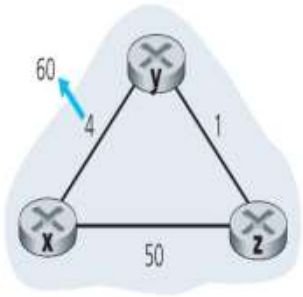


- Before link cost changes:
  $D_y(x) = 4, D_y(z) = 1, D_z(y) = 1, D_z(x) = 5$
- At t_0, cost has changed to 60 from 4. y updates its table with

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

- $D_y(x) = min\{60 + 0, 1 + 5\} = 6$
- At t_1, z receives update from y and updates its table
  $D_z(x) = min\{50 + 0, 1 + 6\} = 7$
- At t_2, y receives update from z and updates table as
  $D_y(x) = 8$. and this process repeats.

DV Algorithm: Link Cost Changes



- Before link cost changes:
  $D_y(x) = 4, D_y(z) = 1, D_z(y) = 1, D_z(x) = 5$
- At t_0, cost has changed to 60 from 4. y updates its table with

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

- $D_y(x) = min\{60 + 0, 1 + 5\} = 6$
- At t_1, z receives update from y and updates its table
  $D_z(x) = min\{50 + 0, 1 + 6\} = 7$
- At t_2, y receives update from z and updates table as
  $D_y(x) = 8$. and this process repeats.
- Count-to-infinity!

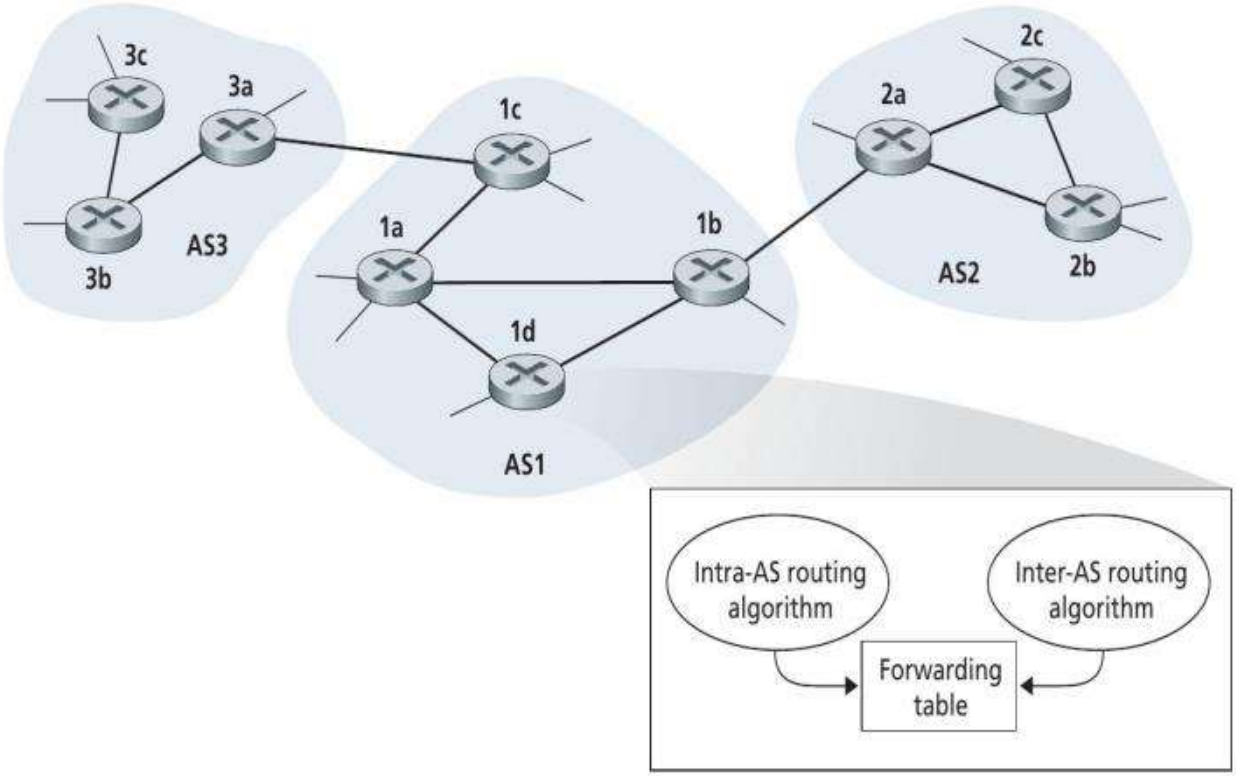
Poisoned Reverse

- If z routes through y, z will inform y that $D_z(x) = \infty$
- y cannot route to x via z as there is no path!
- When $c(x, y) = 60$, y updates its table with $D_y(x) = 60$!
- After receiving an update z routes to x via direct path and updates its table with $D_z(x) = 50$
- After receiving update from z, y recomputes route to x via z and informs z with $D_y(x) = \infty$ (infact it is 51!)

Hierarchical Routing

- Scale: number of routers in internet is very large. Which algorithm to use?
- Administrative autonomy: an organization should be able to run and administer its network as it wishes.
- These problems can be solved by organizing routers into autonomous systems (AS)
- Each AS will have a gateway router

Hierarchical routing



- Hot-potato routing

Routing in Internet

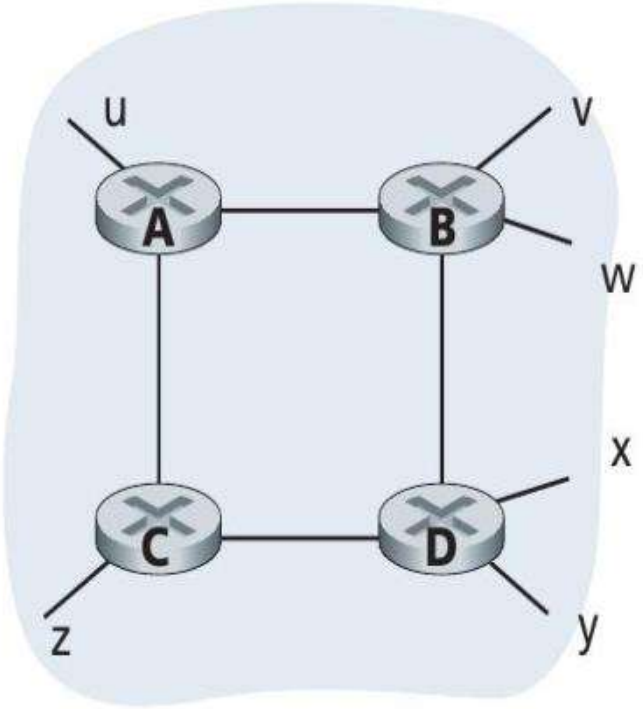
- Intra-AS routing
  - Routing information protocol (RIP): based on DV algorithm
  - Open shortest path first (OSPF): based on LS algorithm
- Inter-AS routing
  - Border Gateway Protocol (BGP)

BGP (Section 4.6.3) is left for self study! Its part of our CCN course

Routing Information Protocol

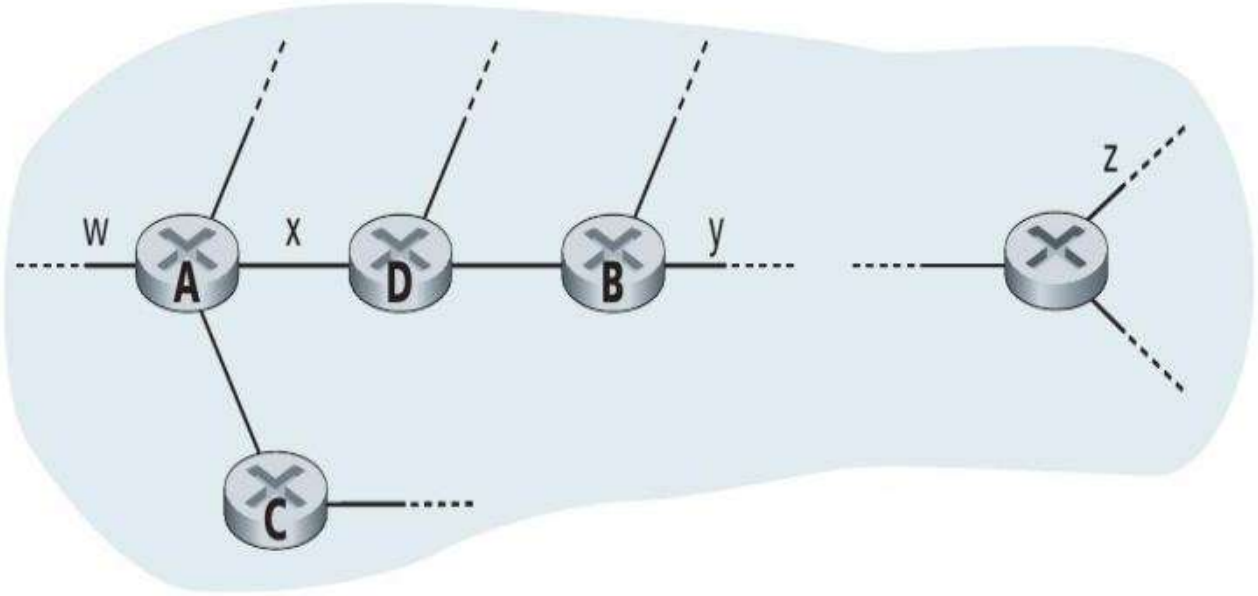
- In RIP, a hop means a subnet
- Cost of a path from source router to destination subnet is the number of hops (subnets) along the path including the destination subnet
- The maximum cost of a path is limited to 15
- RIP uses distance vector algorithm: the routers need to exchange distance vectors or routing updates every 30 seconds
- The RIP response messages or RIP advertisements can contain a list up to 25 destination subnets within the AS.

Example at A



Destination	Hops
u	1
v	2
w	2
x	3
y	3
z	2

A portion of AS



Routing Table at D

Destination Subnet	Next Router	Number of Hops to Destination
w	A	2
y	B	2
z	B	7
x	—	1
....	....	....

Advertisement from A

Destination Subnet	Next Router	Number of Hops to Destination
z	C	4
w	—	1
x	—	1
....	....	....

Updated Routing Table at D

Destination Subnet	Next Router	Number of Hops to Destination
w	A	2
y	B	2
z	A	5
....	....	....

If a router does not hear from its neighbor once every 180 seconds, that neighbor is considered dead. The router propagates about this information to its neighboring routers that are alive!

Open Shortest Path First

- OSPF uses Dijkstra's shortest-path algorihtm
- Choice of link cost is left to the administrator.
- A router broadcasts routing information to all other routers in the AS.
- A router broadcasts link's state whenever there is a change and periodically every 30 seconds.
- OSPF provides features such as security, multiple same-cost paths
- RIP and OSPF are in wide use: regional ISPs use RIP, top-tier ISPs use OSPF.

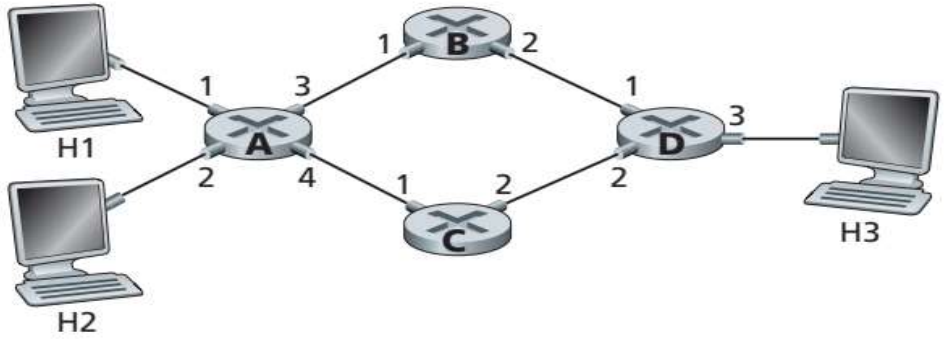
Tutorial_Network_Layer

a. Suppose that this network is a datagram network. Show the forwarding table in router A, such that all traffic destined to host H3 is forwarded through interface 3.

b. Suppose that this network is a datagram network. Can you write down a forwarding table in router A, such that all traffic from H1 destined to host H3 is forwarded through interface 3, while all traffic from H2 destined to host H3 is forwarded through interface 4? (Hint: this is a trick question.)

c. Now suppose that this network is a virtual circuit network and that there is one ongoing call between H1 and H3, and another ongoing call between H2 and H3. Write down a forwarding table in router A, such that all traffic from H1 destined to host H3 is forwarded through interface 3, while all traffic from H2 destined to host H3 is forwarded through interface 4.

d. Assuming the same scenario as (c), write down the forwarding tables in nodes B, C, and D.



P8. In Section 4.3, we noted that the maximum queuing delay is $(n-1)D$ if the switching fabric is n times faster than the input line rates. Suppose that all packets are of the same length, n packets arrive at the same time to the n input ports, and all n packets want to be forwarded to *different* output ports. What is the maximum delay for a packet for the (a) memory, (b) bus, and (c) crossbar switching fabrics?

Tutorial_Network_Layer

P9. Consider the switch shown below. Suppose that all datagrams have the same fixed length, that the switch operates in a slotted, synchronous manner, and that in one time slot a datagram can be transferred from an input port to an output port. The switch fabric is a crossbar so that at most one datagram can be transferred to a given output port in a time slot, but different output ports can receive datagrams from different input ports in a single time slot. What is the minimal number of time slots needed to transfer the packets shown from input ports to their output ports, assuming any input queue scheduling order you want (i.e., it need not have HOL blocking)? What is the largest number of slots needed, assuming the worst-case scheduling order you can devise, assuming that a non-empty input queue is never idle?