CS 455/555
Intro to Networks and Communications

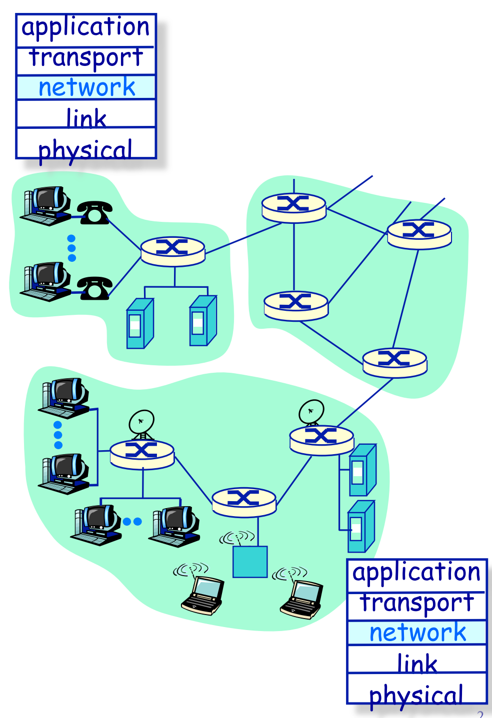
The Network Layer: Routing Algorithms

Dr. Michele Weigle
Department of Computer Science
Old Dominion University
mweigle@cs.odu.edu

http://www.cs.odu.edu/~mweigle/CS455-S13/

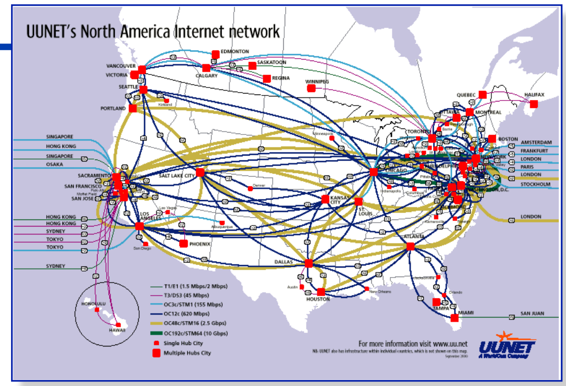
The Network Layer: Routing & Addressing
Outline

- Network layer functions
- Virtual circuits and datagram networks
- Router architecture
- IP Internet Protocol
  - » Addressing
- Routing algorithms
  - » Least cost path computation algorithms
- Hierarchical routing
  - » Connecting networks of networks
- Routing on the Internet
  - » Intra-domain routing
  - » Inter-domain routing



Routing Algorithms

Least-cost path computation

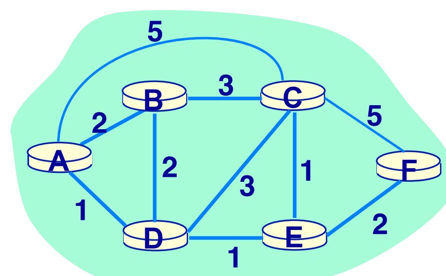


- Goal: To determine a "good" path through the network from source to destination
- Graph abstraction for routing algorithms:
  - » Nodes are routers
  - » Edges are physical links
  - » Edges have a "cost" metric
    - ❖ Cost can be delay, monetary cost, level of congestion, *etc*.
- "Good" path typically means minimum cost path
  - » Also shortest path, …
- (But often ISPs define "good" in terms of business models)

Routing Algorithms

Taxonomy

- Global or decentralized information?
- Global — all routers have complete graph (topology, costs)
  - » "Link state" algorithms
- Decentralized — router knows link costs to physically connected adjacent nodes
  - » Run iterative algorithm to exchange information with adjacent nodes
  - » "Distance vector" algorithms

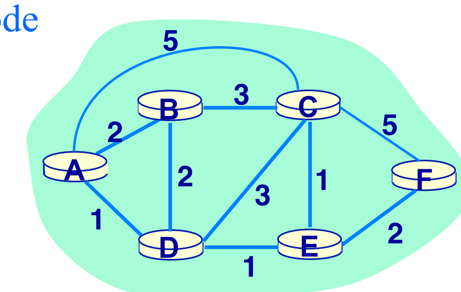


- Static or dynamic?
  - » Static — routes change slowly over time
  - » Dynamic — routes change more quickly
    - ❖ Periodic updates, or
    - ❖ Updates in response to link outages or cost changes

Global Routing Algorithms

A link-state routing algorithm

- Uses Dijkstra's shortest path graph algorithm
- Complete network topology and link costs known at *all* nodes
  - » Accomplished via *link state flooding*
  - » All nodes learn the "same" topology and cost data
- Each node computes least cost paths from itself to all other nodes
  - » Produces a *routing table* for that node
  - » All nodes compute consistent routing tables
- Algorithm complexity:
  - » *N* nodes (routers) in the network
  - » *N* x (*N*+1)/2 comparisons
  - » (More efficient implementations possible)



Link State Routing

Dijkstra's Algorithm

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

Loop
   find node w not in N such that D(w) is a minimum
   add node w to N
   update D(v) for all nodes v adjacent to w and not in N:
        D(v) = min( D(v), D(w) + c(w,v) )
     /* new cost to node 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
```

N	is the set of nodes to which we have computed the minimum cost path
D(*x*)	is the current minimum cost path to *x*
c(*n*,*m*)	is the cost of the link from *n* to *m*

Animation: http://en.wikipedia.org/wiki/File:Dijksta_Anim.gif

Link State Routing

Dijkstra's algorithm: example

Step
0
1
2
3
4
5

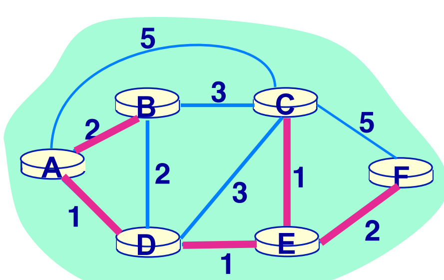
F),p(F)
nfinity

s to
omputed
t path
imum
r of x on
num cost

Link State Routing

Dijkstra's algorithm: example

Step	start N	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	A	2,A	5,A	1,A	infinity	infinity
1						
2						
3						
4						
5						



N is the set of nodes to which we have computed the minimum cost path

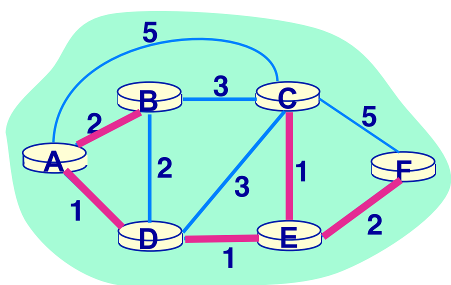
$D(x)$ is the current minimum cost path to x

$p(x)$ is the predecessor of x on the current minimum cost path to x

Link State Routing

Dijkstra's algorithm: example

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



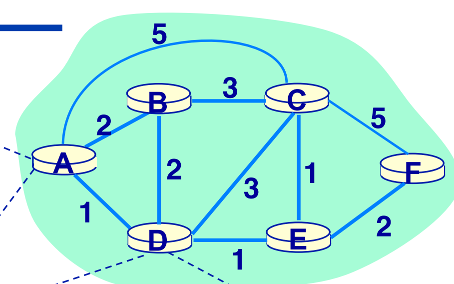
N is the set of nodes to which we have computed the minimum cost path

$D(x)$ is the current minimum cost path to x

$p(x)$ is the predecessor of x on the current minimum cost path to x

Link State Routing

Link state routing table



Link State Routing Table for *A*

destination	***first*** node in least cost path
B	B
C	D
D	D
E	D
F	D

Link State Routing Table for *D*

destination	***first*** node in least cost path
A	A
B	B
C	E
E	E
F	E

Link State Routing

Link State Flooding Algorithm

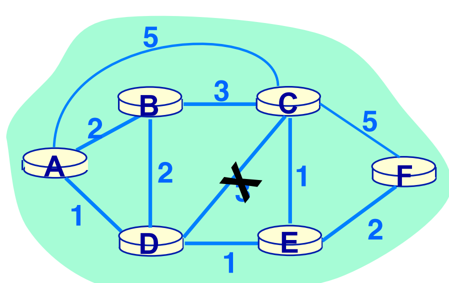
- The data stored for an edge in the graph (the link between nodes *X* and *Y*) consists of:
  - » Cost from *X* to *Y* (*X*-*Y*) and from *Y* to *X* (*Y*-*X*)
  - » A unique timestamp for the last update to each cost

- A node that discovers a change in cost for one of its attached links forwards the update to all adjacent nodes

- A node receiving an update forwards it based on a comparison of the update timestamp and the timestamp on its local data for the link:
  - » Update is later (or new): Forward to all adjacent nodes (except sender) and update local data
  - » Update is earlier: Send local data back to sender
  - » Update is equal: Do nothing

Link State Flooding Algorithm

Example



Step 1: Link C-D fails

Step 2: C sends C->D=∞ to A,B,E,F
D sends D->C=∞ to A,B,E

Step 3: A sends C->D=∞ to B,D
D->C=∞ to B,C
B sends C->D=∞ to A,D
D->C=∞ to A,C
E sends C->D=∞ to D,F
D->C=∞ to C,F
F sends C->D=∞ to E

Step 4: C sends D->C=∞ to B,E,F
(received from A)
D sends C->D=∞ to B,E
(received from A)
F sends D->C=∞ to C
(received from E)

Link State Flooding Algorithm

Example

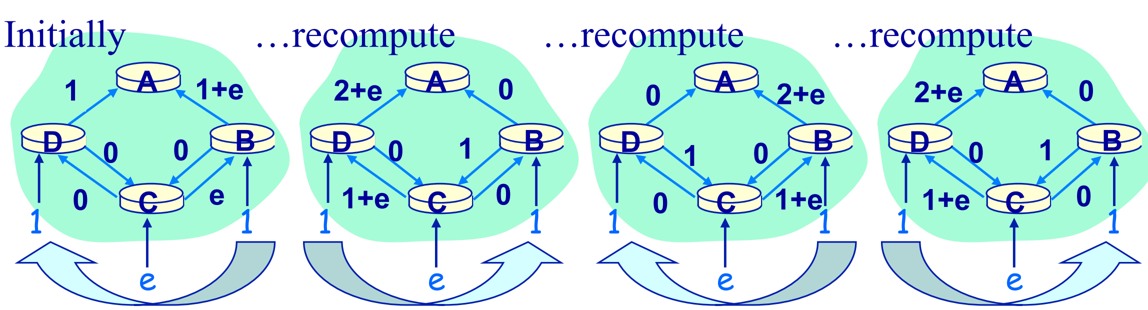
	C->D = ∞			D->C = ∞		
rcvr	**Step 2**	**Step 3**	**Step 4**	**Step 2**	**Step 3**	**Step 4**
A	C	B		D	B	
B	C	A	D	D	A	C
C	---				A, B, E	F
D		A, B, E		---		
E	C	F	D	D		C
F	C	E			E	C

Notation: Letter in cell indicates the sender of the message.
Circle around letter indicates the receipt of this message triggered a broadcast.

Link State Routing

Oscillating routes

- "Route oscillations" are possible in link state algorithms
- Let the link cost equal the amount of carried traffic
  - » Assume the link cost is updated as traffic changes



Node	Path to A
D	A
C	B
B	A

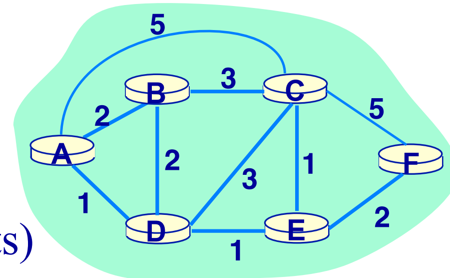
Node	Path to A
D	A
C	D
B	C

Node	Path to A
D	C
C	B
B	A

Node	Path to A
D	A
C	D
B	C

Routing Algorithms
Taxonomy

- Global or decentralized information?

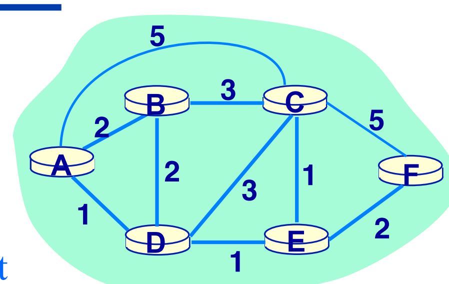


- Global — all routers have maintain the complete graph of the network (topology, costs)
  - » "Link state" algorithms

- Decentralized — router knows link costs to physically connected adjacent nodes
  - » Run iterative algorithm to exchange information with adjacent nodes
  - » "Distance vector" algorithms

Decentralized Routing Algorithms
Distance Vector Routing

- Iterative:
  - » Nodes exchange cost information until each node has the current route costs
  - » The algorithm is *self-terminating* — there's no explicit stopping point



- Asynchronous:
  - » Nodes need not exchange information and iterate in lock step
  - » Intermediate results may be inconsistent across nodes

- Distributed:
  - » Each node communicates only with directly-attached adjacent nodes
  - » (But there is no flooding of cost information)

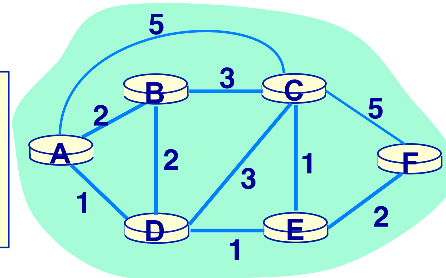
Decentralized Routing Algorithms
Distance Vector Routing

Bellman-Ford equation

$d_X(Y) =$ *cost of least-cost path from X to Y*

$= \min_v \{c(X,v) + d_v(Y)\}$

c(x,y) = cost of link x to y *v = {neighbors of X}*



◆Let's calculate this for source A and destination F.

» A has 3 neighbors (B, C, D)

» $c(A,B)=2$ » $d_B(F)=5$

» $c(A,C)=5$ » $d_C(F)=3$

» $c(A,D)=1$ » $d_D(F)=3$

◆$\mathbf{d_A(F)} = \min \{2+5, 5+3, 1+3\} = \mathbf{4}$

Distance Vector Routing
Algorithm

◆ Iterative, asynchronous

Each local iteration caused by:

»Local link cost change, or

»Message from adjacent node that its least cost path to some destination has changed

◆ Distributed:

»Each node notifies adjacent nodes *only* when its least cost path to some destination changes

»Adjacent nodes then notify their adjacent nodes if this update changes a least cost path

Each node:

wait for change in local link cost ***or*** message from adjacent node

↓

recompute distance table

↓

if least cost path to any destination has changed, ***notify*** adjacent nodes

Distance Vector Routing
Algorithm

- Initialization phase: At all nodes *X*:

```
for all destinations V in N {
     D_X(V) = "∞"          /* the cost to reach all destinations is infinite */
}

for each neighbor V {
     D_X(V) = c(X,V)      /* record the cost to reach each adjacent node
}                            (cost from X to each V) */

for each neighbor V {
     send D_x = [D_X(y): y in N] to V  /* send current minimum costs for all
}                                         destinations to all neighbors    */
```


Distance Vector Routing
Algorithm main loop (at node *X*)

loop

wait until (**receive** link cost change to adjacent node v
or **receive** distance vector from some neighbor v)

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

if $D_x(y)$ changed for any destination y
send distance vector $\mathbf{D_x}$ = [Dx(y): y in N] to all neighbors

forever

Distance Vector Algorithm
Example

node x table

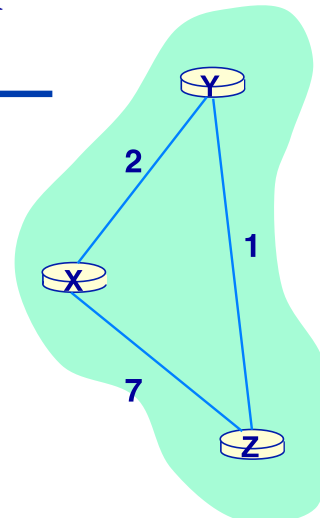
from \ cost to	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

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



Distance Vector Algorithm
Example

node x table

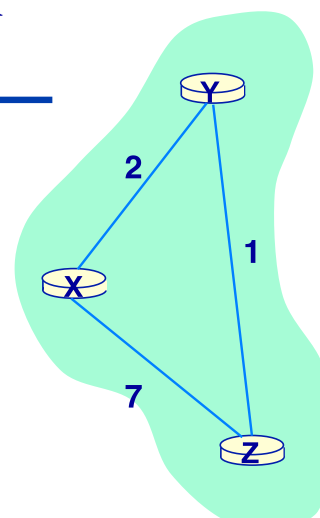
from \ cost to	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

node y table

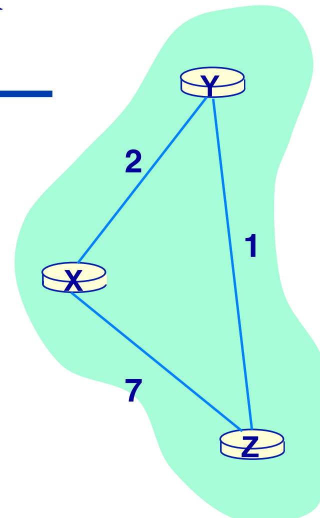
from \ cost to	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

node z table

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



Distance Vector Algorithm
Example



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	∞	∞	∞

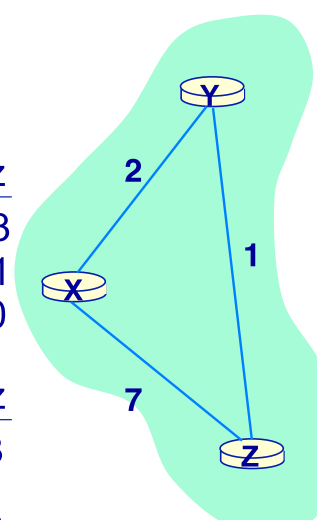
node z table

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

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

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

Distance Vector Algorithm
Example



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

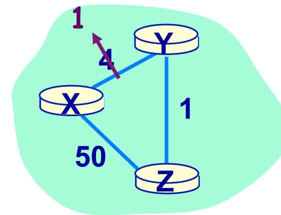
from \ cost to	x	y	z
x	0	2	7
y	2	0	1
z	**3**	1	0

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

Done sending?

Distance Vector Algorithm

Link cost changes



- When a node detects a local link cost change:
  - » The nodes updates its distance table
  - » If the least cost path (i.e., distance vector) changes, the node notifies its neighbors

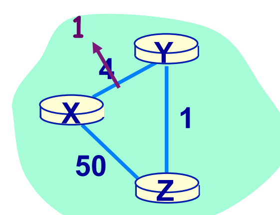
t_0 : *Y* detects link-cost change, updates its DV, informs its neighbors.

t_1 : *Z* receives update from *Y*, updates its table, computes new least cost to *X*, sends its neighbors its DV.

t_2 : *Y* receives *Z*'s update, updates its distance table. *Y*'s least costs do *not* change, so *Y* does *not* send a message to *Z*.

Distance Vector Algorithm

Link cost changes



node x

t_0 from \ cost to	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

t_1 from \ cost to	x	y	z
x	0	**1**	**2**
y	1	0	1
z	5	1	0

t_2 from \ cost to	x	y	z
x	0	1	2
y	1	0	1
z	2	1	0

node y

t_0 from \ cost to	x	y	z
x	0	4	5
y	**1**	0	1
z	5	1	0

t_1 from \ cost to	x	y	z
x	0	4	5
y	1	0	1
z	5	1	0

t_2 from \ cost to	x	y	z
x	0	1	2
y	1	0	1
z	2	1	0

node z

t_0 from \ cost to	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

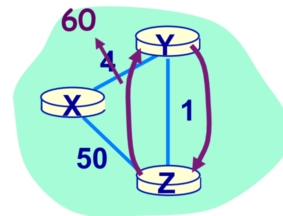
t_1 from \ cost to	x	y	z
x	0	4	5
y	4	0	1
z	**2**	1	0

t_2 from \ cost to	x	y	z
x	0	1	2
y	1	0	1
z	2	1	0

Distance Vector Algorithm

Link cost changes

Routing Loop!
Does it Terminate?

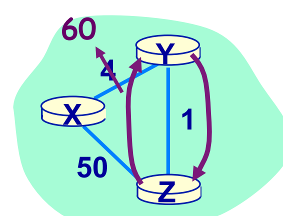


- Good news travels fast, but…
- "Bad news" travels slow!
  - » The "count to infinity" problem

Distance Vector Algorithm

Link cost changes

Routing Loop!
Does it Terminate?



$\mathbf{t_0}$

node x

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

node y

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

node z

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

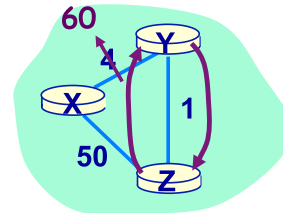
Why does Y to X change from 4 to 6 (and not 60?)

$$D_y(x) = \min\{c(y,x) + D_x(x), c(y,z) + D_z(x)\}$$
$$= \min\{60+0, 1+5\} = \mathbf{6}$$

Distance Vector Algorithm

Link cost changes

Routing Loop!
Does it Terminate?



t_0

node x

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

node y

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

node z

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

t_1

node x:

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

node y:

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

node z:

from \ cost to	x	y	z
x	0	4	5
y	6	0	1
z	**7**	1	0

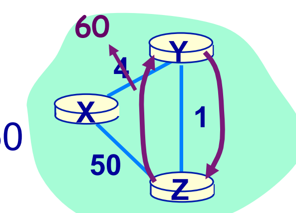
$D_x(y) = \min\{c(x,y) + D_y(y), c(x,z) + D_z(y)\}$
$= \min\{60+0, 50+1\} = \mathbf{51}$
$D_x(z) = \min\{c(x,z) + D_z(z), c(x,y) + D_y(z)\}$
$= \min\{50+0, 60+1\} = \mathbf{50}$

$D_z(x) = \min\{c(z,x) + D_x(x), c(z,y) + D_y(x)\}$
$= \min\{50+0, 1+6\} = \mathbf{7}$
$D_z(y) = \min\{c(z,y) + D_y(y), c(z,x) + D_x(y)\}$
$= \min\{1+0, 50+4\} = \mathbf{1}$

Distance Vector Algorithm

Link cost changes

$D_y(x) = \min\{c(y,x) + D_x(x), c(y,z) + D_z(x)\}$
$= \min\{60+0, 1+7\} = \mathbf{8}$



t_0

node x

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

node y

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

node z

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

t_1

node x:

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

node y:

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

node z:

from \ cost to	x	y	z
x	0	4	5
y	6	0	1
z	**7**	1	0

t_2

node x:

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

node y:

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

node z:

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

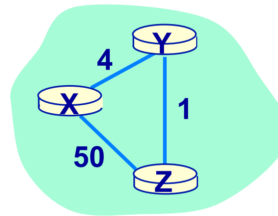
…

What happens next?

How long to stabilize?

The Count to Infinity Problem

The "poisoned reverse" technique



- If Z routes through Y to get to X:
  - » Then Z tells Y that Z's distance to X is infinite

node x

t_{-3} (cost to)

from	x	y	z
x	0	4	50
y	4	0	1
z	50	1	0

t_{-2} (cost to)

from	x	y	z
x	0	4	**5**
y	4	0	1
z	50	1	0

t_{-1} (cost to)

from	x	y	z
x	0	4	5
y	4	0	1
z	**5**	1	0

node y

cost to

from	x	y	z
x	0	4	50
y	4	0	1
z	50	1	0

cost to

from	x	y	z
x	0	4	50
y	4	0	1
z	50	1	0

cost to

from	x	y	z
x	0	4	∞
y	4	0	1
z	∞	1	0

node z

cost to

from	x	y	z
x	0	4	50
y	4	0	1
z	50	1	0

cost to

from	x	y	z
x	0	4	50
y	4	0	1
z	**5**	1	0

cost to

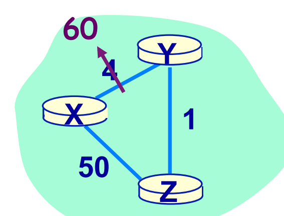
from	x	y	z
x	0	4	**5**
y	4	0	1
z	5	1	0

∞

∞

The Count to Infinity Problem

The "poisoned reverse" technique



- Will this completely solve the problem?

t_0

x (cost to)

from	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

y (cost to)

from	x	y	z
x	0	4	∞
y	**60**	0	1
z	∞	1	0

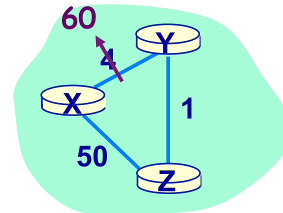
z (cost to)

from	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

$$D_y(x) = \min\{c(y,x) + D_x(x),\, c(y,z) + D_z(x)\}$$
$$= \min\{60+0,\, 1+\infty\} = \mathbf{60}$$

The Count to Infinity Problem

The "poisoned reverse" technique



◆ Will this completely solve the problem?

x, t_0 (cost to)

from	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

x, t_1 (cost to)

from	x	y	z
x	0	**51**	**50**
y	60	0	1
z	5	1	0

x, t_2 (cost to)

from	x	y	z
x	0	51	50
y	60	0	1
z	50	1	0

y, t_0 (cost to)

from	x	y	z
x	0	4	∞
y	**60**	0	1
z	∞	1	0

y, t_1 (cost to)

from	x	y	z
x	0	4	∞
y	60	0	1
z	∞	1	0

y, t_2 (cost to)

from	x	y	z
x	0	51	**50**
y	**51**	0	1
z	50	1	0

y, t_3 (cost to)

from	x	y	z
x	0	51	50
y	**51**	0	1
z	50	1	0

z, t_0 (cost to)

from	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

z, t_1 (cost to)

from	x	y	z
x	0	4	5
y	60	0	1
z	**50**	1	0

z, t_2 (cost to)

from	x	y	z
x	0	∞	50
y	60	0	1
z	50	1	0

z, t_3 (cost to)

from	x	y	z
x	0	∞	50
y	∞	0	1
z	50	1	0

Least Cost Path Computations

Comparison of the link-state & distance vector algorithms

◆ Message complexity:

» LS: With *N* nodes, *E* links, *O*(*N*x*E*) messages sent for flooding

» DV: Exchange between neighbors only (may trigger further exchanges)

◆ Speed of Convergence:

» LS: O(*N*²) algorithm and *O*(*N*x*E*) messages

❖ May have oscillations

» DV: Convergence time varies

❖ Routing loops possible

❖ Count-to-infinity problem

◆ Robustness: what happens if there are failures?

» LS: Node can advertise incorrect *link* cost

Each node computes only its *own* table

» DV: Node can advertise incorrect *path* cost

Each node's table used by others

❖ *Errors propagate through network*

The Network Layer: Routing & Addressing

Outline

- Network layer functions
- Virtual circuits and datagram networks
- Router architecture
- IP Internet Protocol
  - » Addressing
- Routing algorithms
  - » Least cost path computation algorithms
- Hierarchical routing
  - » Connecting networks of networks
- Routing on the Internet
  - » Intra-domain routing
  - » Inter-domain routing