Lecture 19: Network Layer
Routing in the Internet

COMP 332, Spring 2023

Victoria Manfredi

WESLEYAN
UNIVERSITY

Acknowledgements: materials adapted from Computer Networking: A Top Down Approach 7th edition: ©1996-2016, J.F Kurose and K.W. Ross, All Rights Reserved as well as from slides by Abraham Matta at Boston University, and some material from Computer Networks by Tannenbaum and Wetherall.

Today

1. Announcements
   - hw7 written due Wed. at 11:59p (~1 week), programming due next Wed. (~2 weeks)
   - what's a virtual machine?
   - run the traceroute command and look at traffic in wireshark
     - compare with pkts you're generating
   - socket.inet_aton, socket.ntoa_inet()
     - to convert string address to/from 32-bit packed address
2. Distance vector routing
3. Internet routing
   - overview
4. Internet Control Message Protocol (ICMP)

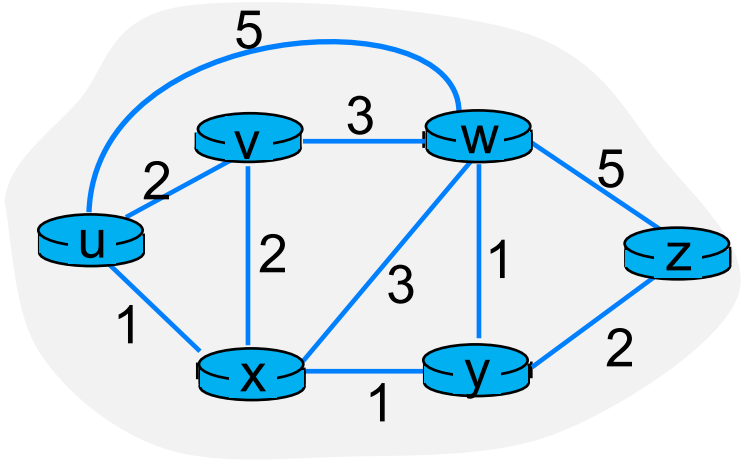
Takeaways from last time

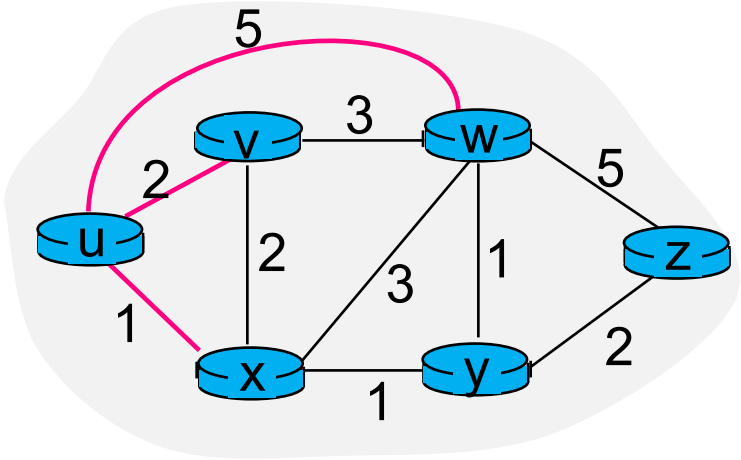
Global information

- global link state algorithms
- all routers have complete topology, link cost info
- *exchange info ony about neighbors but with all nodes*

Local/decentralized information

- decentralized distance vector algorithms
- router knows only physically-connected neighbors, link costs to neighbors
- iterative computation
- *exchange info about all nodes but only with neighbors*





Both are used on Internet. First cover abstractly and then talk about specific Internet protocols (OSPF, BGP, RIP, …)

Takeaways from last time

Link state routing

- every node exchanges with every other node in network information about its links to neighbors
- then each node runs Dijkstra's knowing complete graph

Distance vector routing

- every node exchanges with neighbors only its distance estimates to every other node in network
- then each node updates its distance estimates using new estimates from neighbors, then sends its own new estimates to neighbors

Given min cost paths

- can directly compute forwarding table
- forwarding table is used by routers to find next hops for packets
- these min cost paths will need to be periodically recomputed, which can introduce problems

Control Plane

LINK STATE ROUTING

Dijkstra's algorithm

c(i,j): **link cost** from node i to node j

D(k): **current cost** from source u to destination node k

p(k): **predecessor node** along path from source u to k

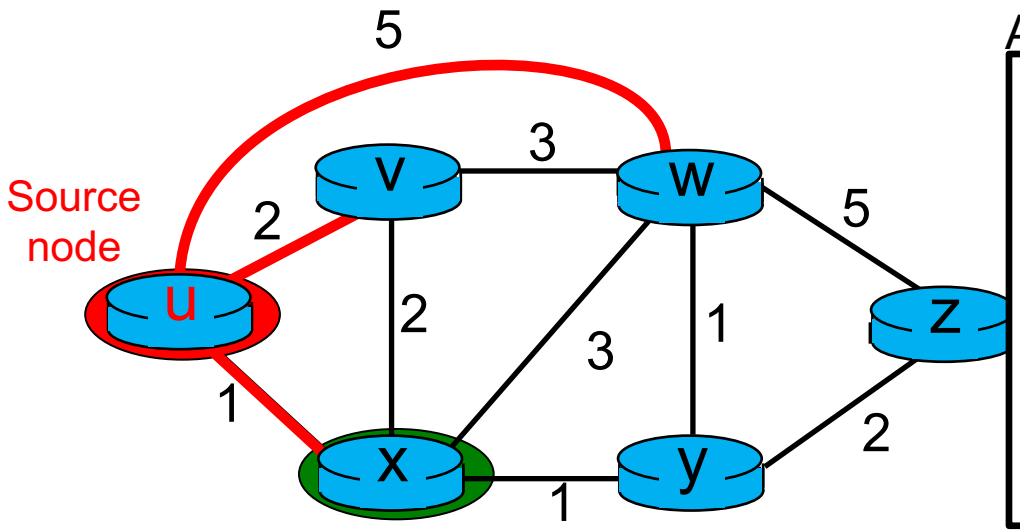
N': set of nodes whose least cost path is **definitively known**

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		x is not in N', and D(x) is lowest				
3						
4						
5						

Loop

Find j ∉ N' s.t. D(j) is min

Add j to N'



Now we know the *lowest cost path from u to x.* Why?

Any other path from u to x must go through *neighbor of u to get to x.* But we just looked at all neighbors of u

Dijkstra's algorithm

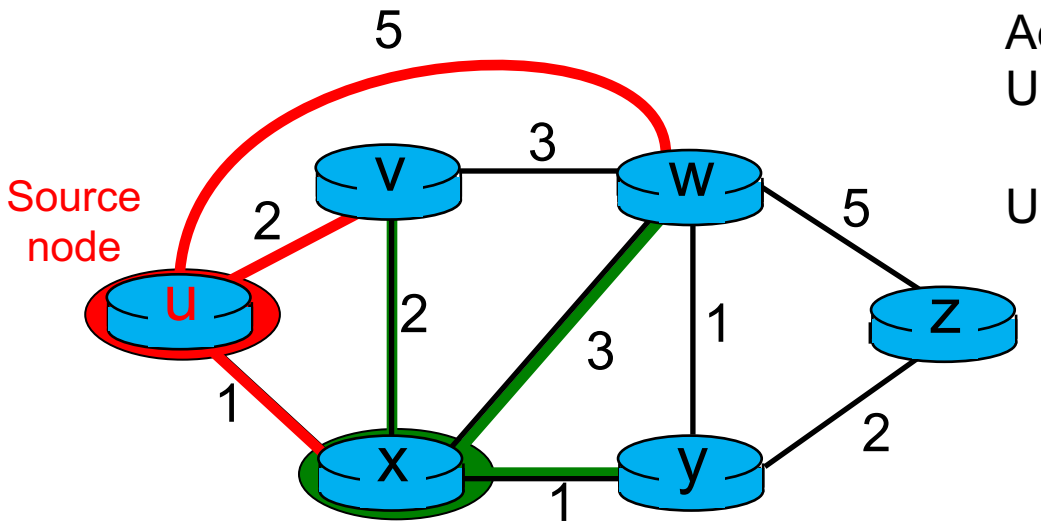
c(i,j): **link cost** from node i to node j

D(k): **current cost** from source u to destination node k

p(k): **predecessor node** along path from source u to k

N': set of nodes whose least cost path is **definitively known**

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						
3						
4						
5						



Loop

Find j ∉ N' s.t. D(j) is min

Add j to N'

Update D(k) for all neighbors k ∉ N' of j

D(k) = min(D(k), D(j)+c(j,k))

Until all nodes in N'

Now we check whether any *neighbors of x that are not in N'* can be reached with lower cost path by *first going through x*

Dijkstra's algorithm

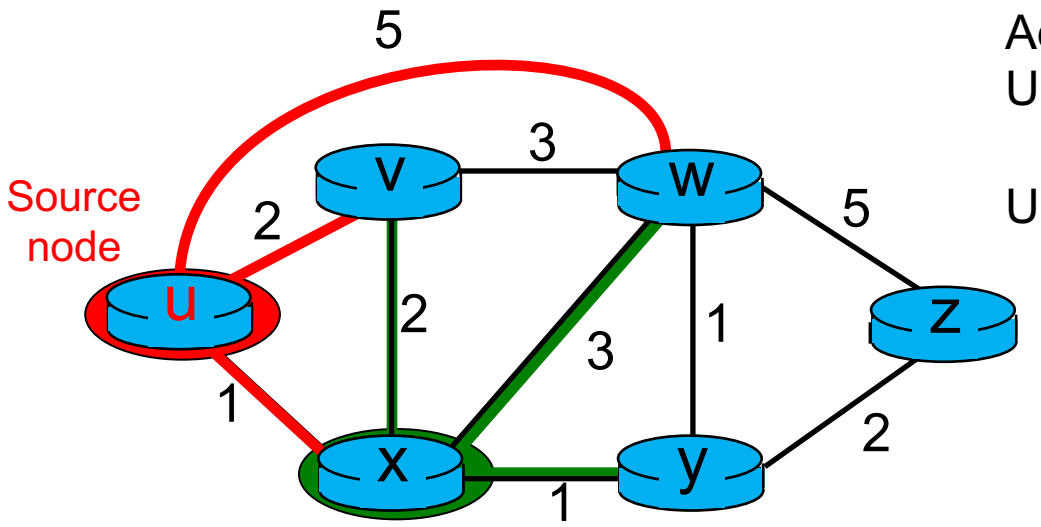
c(i,j): **link cost** from node i to node j

D(k): **current cost** from source u to destination node k

p(k): **predecessor node** along path from source u to k

N': set of nodes whose least cost path is **definitively known**

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				
2		D(v)				
3		= min(D(v), D(x)+c(x,v))				
4		= min(2, 1+2)				
5						



Loop

Find j ∉ N' s.t. D(j) is min

Add j to N'

Update D(k) for all neighbors k ∉ N' of j

D(k) = min(D(k), D(j)+c(j,k))

Until all nodes in N'

3 min: compute the updated values of D(v), D(w), D(y)

Dijkstra's algorithm

c(i,j): **link cost** from node i to node j

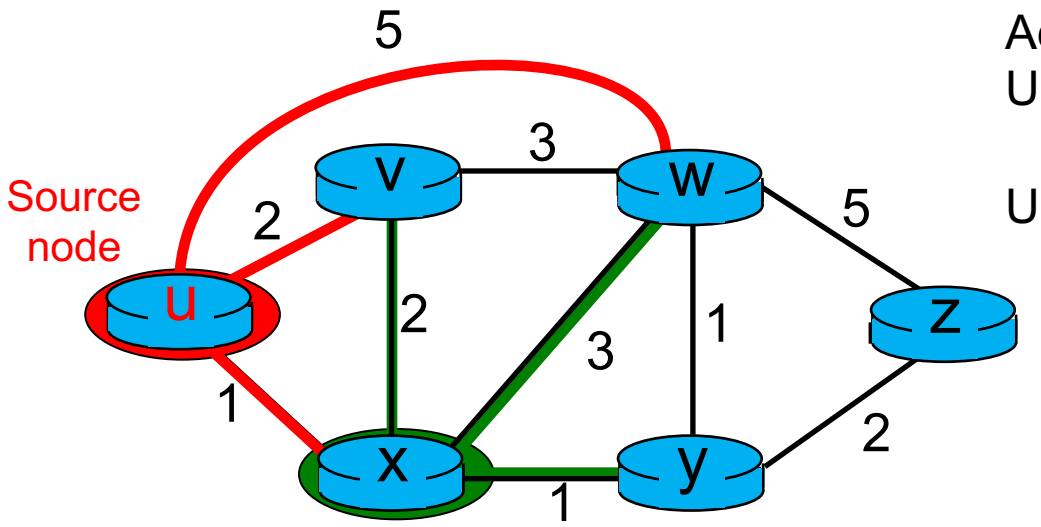
D(k): **current cost** from source u to destination node k

p(k): **predecessor node** along path from source u to k

N': set of nodes whose least cost path is **definitively known**

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						
3						
4						
5						

D(w)
= min(D(w), D(x)+c(x,w))
= min(5, 1+3)



Loop

Find j ∉ N' s.t. D(j) is min

Add j to N'

Update D(k) for all neighbors k ∉ N' of j

D(k) = min(D(k), D(j)+c(j,k))

Until all nodes in N'

3 min: compute the updated values of D(v), D(w), D(y)

Dijkstra's algorithm

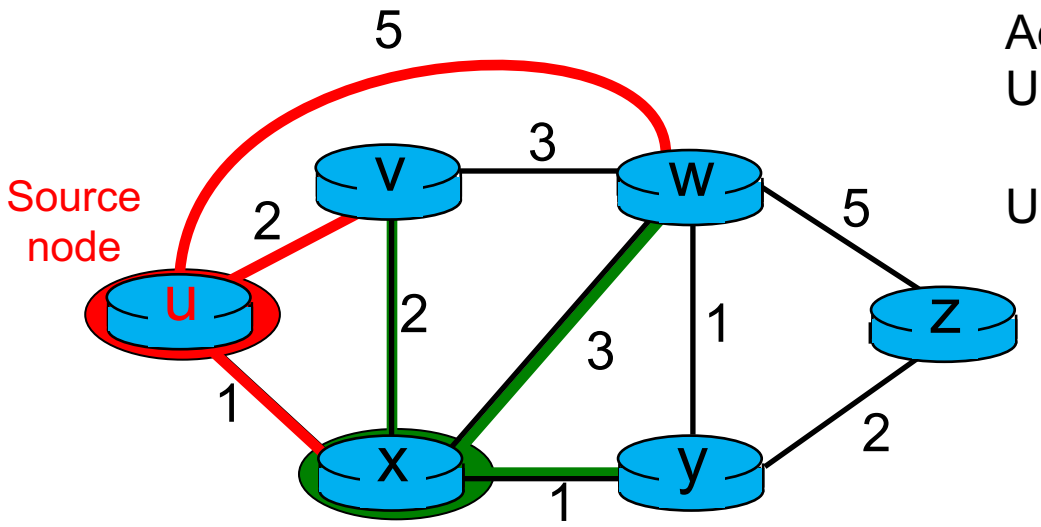
c(i,j): **link cost** from node i to node j

D(k): **current cost** from source u to destination node k

p(k): **predecessor node** along path from source u to k

N': set of nodes whose least cost path is **definitively known**

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 is in N', don't update		
3						
4						
5						



Loop

Find j ∉ N' s.t. D(j) is min

Add j to N'

Update D(k) for all neighbors k ∉ N' of j

D(k) = min(D(k), D(j)+c(j,k))

Until all nodes in N'

3 min: compute the updated values of D(v), D(w), D(y)

Dijkstra's algorithm

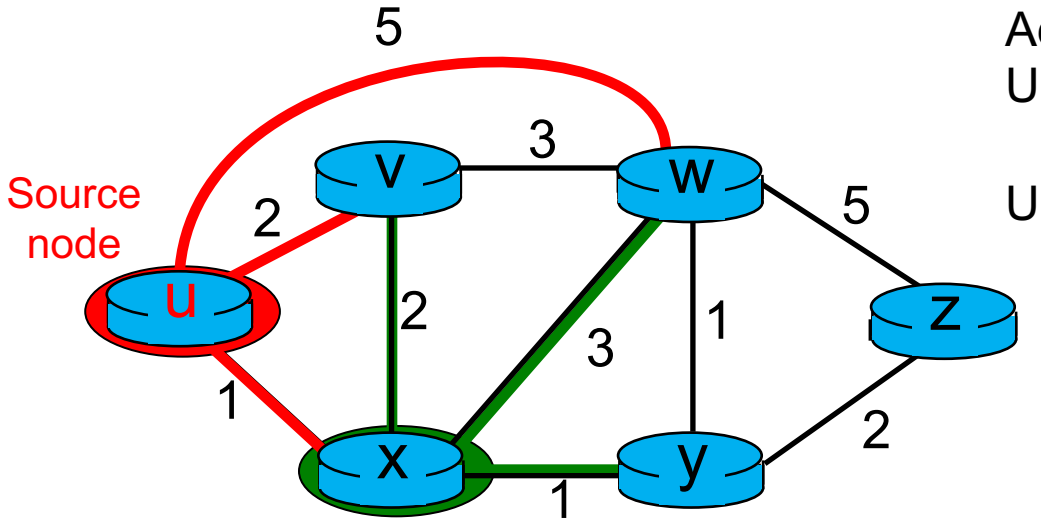
c(i,j): **link cost** from node i to node j

D(k): **current cost** from source u to destination node k

p(k): **predecessor node** along path from source u to k

N': set of nodes whose least cost path is **definitively known**

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					D(y)	
3					= min(D(y), D(x)+c(x,y))	
4					= min(∞, 1+1)	
5						



Loop

Find j ∉ N' s.t. D(j) is min

Add j to N'

Update D(k) for all neighbors k ∉ N' of j

D(k) = min(D(k), D(j)+c(j,k))

Until all nodes in N'

3 min: compute the updated values of D(v), D(w), D(y)

Dijkstra's algorithm

c(i,j): **link cost** from node i to node j

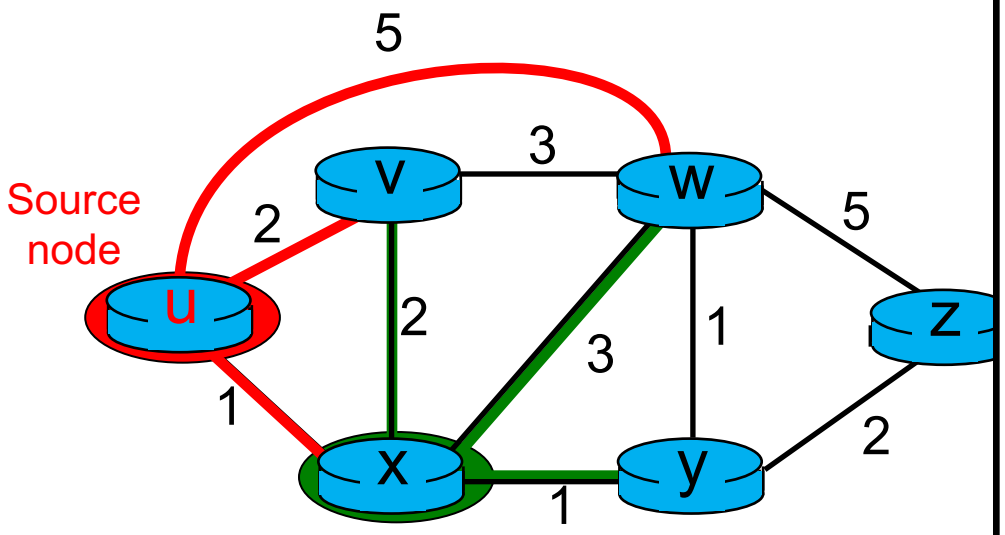
D(k): **current cost** from source u to destination node k

p(k): **predecessor node** along path from source u to k

N': set of nodes whose least cost path is **definitively known**

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						
3						
4						
5						

D(z): z is not a neighbor of x so don't update



Now we know the *lowest cost path from u to y.* Why?

Any other path from u to y must go through *neighbor of u but x is lowest cost neighbor.*

And adding on cost from x to y still gives *lower (same) cost than even to just go to other neighbors of u.*

Dijkstra's algorithm

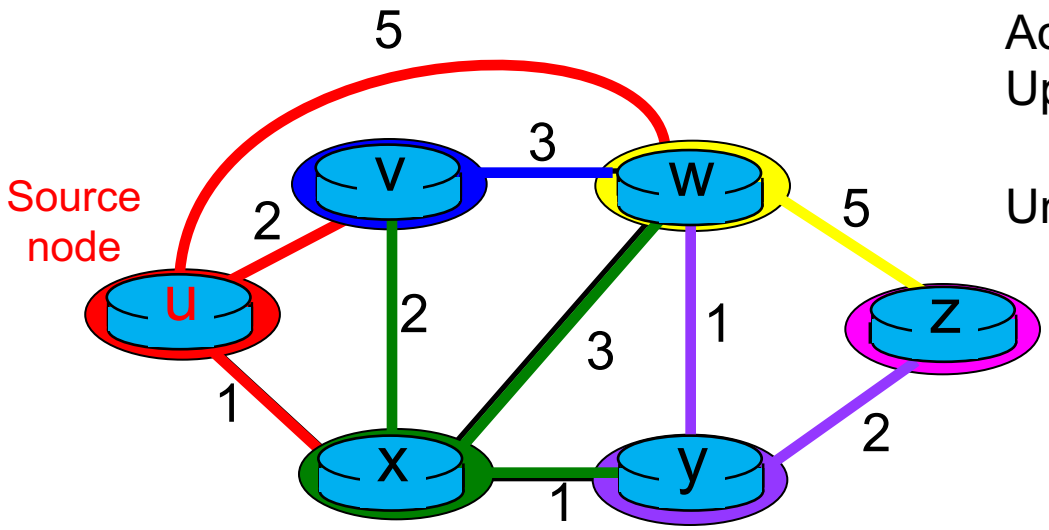
c(i,j): **link cost** from node i to node j

D(k): **current cost** from source u to destination node k

p(k): **predecessor node** along path from source u to k

N': set of nodes whose least cost path is **definitively known**

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					



Loop

Find j ∉ N' s.t. D(j) is min

Add j to N'

Update D(k) for all neighbors k ∉ N' of j

D(k) = min(D(k), D(j)+c(j,k))

Until all nodes in N'

Dijkstra's algorithm

c(i,j): **link cost** from node i to node j

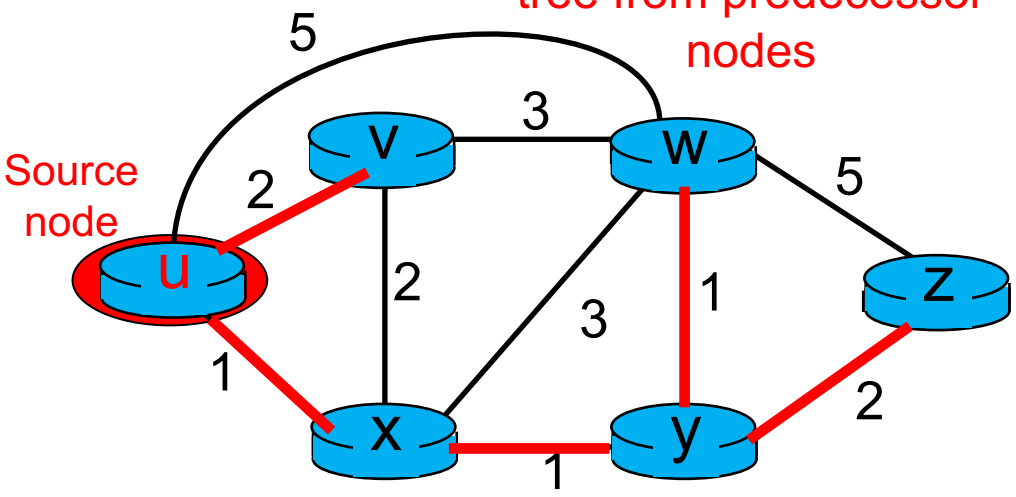
D(k): **current cost** from source u to destination node k

p(k): **predecessor node** along path from source u to k

N': set of nodes whose least cost path is **definitively known**

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					

1. Build shortest path tree from predecessor nodes



2. Build forwarding table at u

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

Algorithm complexity with n nodes

Each iteration: need to check all nodes not in N'

- n in 1st iteration, n-1 in 2nd iteration, n-2 in 3rd iteration …
- n(n+1)/2 comparisons: $O(n^2)$, more efficient implementations possible

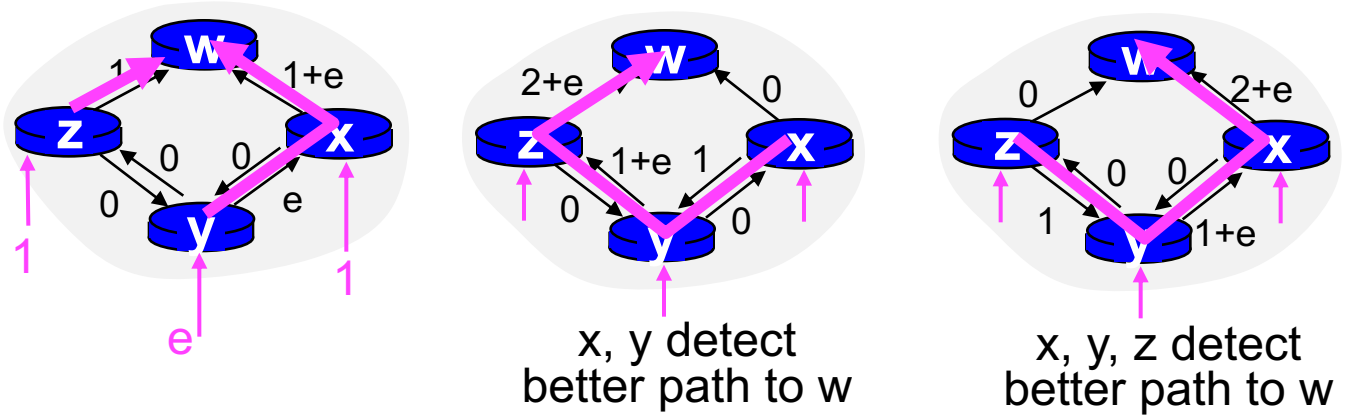
Network is dynamic

- link goes down: link state broadcast
- router goes down: remove link and all nodes recompute

Oscillations possible

- when congestion or delay-based link cost

initially … recompute routing … recompute



x, y detect better path to w

x, y, z detect better path to w

Need to prevent routers from synchronizing computations:
Have routers randomize when they send out link advertisements

Control Plane

DISTANCE VECTOR ROUTING

Distance vector algorithm run at each node x

Initialization

For all dst y ϵ N

if y is nbr of x

$D_x(y) = c(x, y)$

else

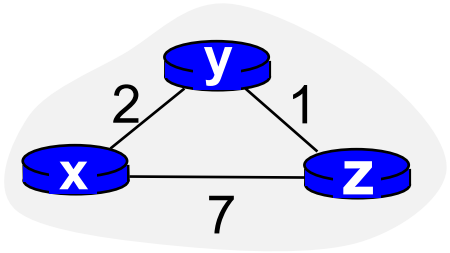
$D_x(y) = \infty$

For each nbr w and dst y ϵ N

$D_w(y) = \infty$

Send x's DV to all nbrs w

$D_x = [D_x(y) : y \epsilon N]$



# of nodes

Node x

# of neighbors

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

Node y

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

Node z

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

Node x

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

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

Node y

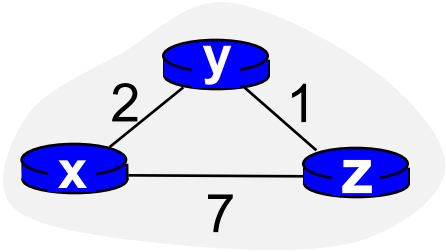
from	cost to x	cost to y	cost to z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

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

Node z

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

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



Distance vector algorithm run at each node x

Initialization

For all dst y ϵ N

if y is nbr of x

$D_x(y) = c(x, y)$

else

$D_x(y) = \infty$

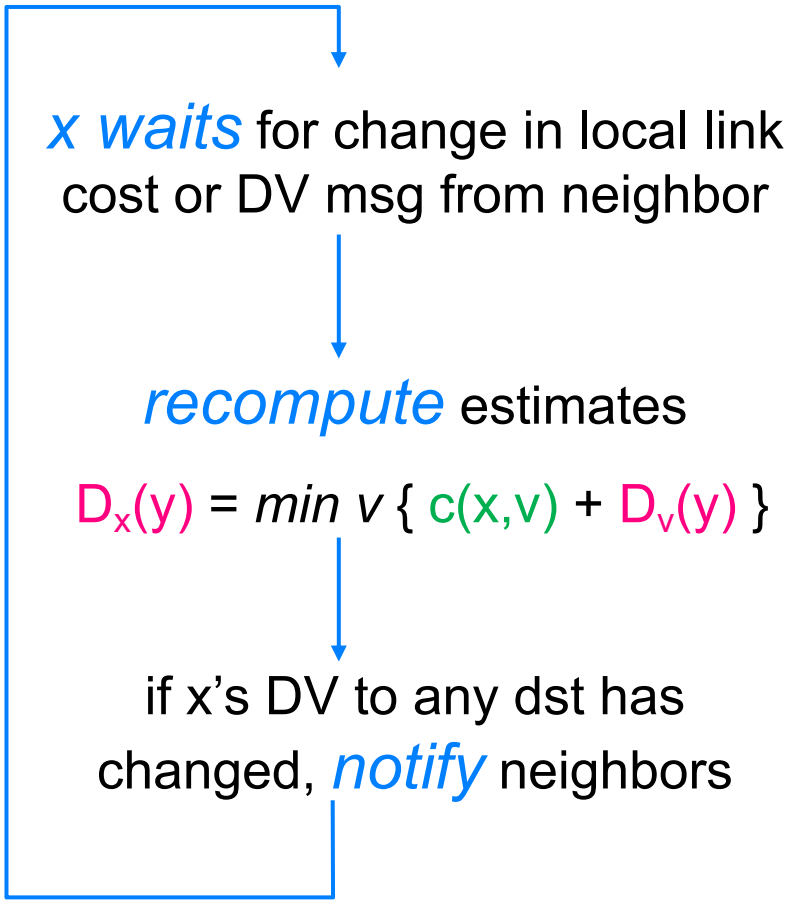
For each nbr w and dst y ϵ N

$D_w(y) = \infty$

Send x's DV to all nbrs w

$D_x = [D_x(y) : y \epsilon N]$

Loop



Q: when does loop terminate?

When no more changes

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

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

Node x

cost to

from	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

cost to

from	x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

3 min: What is $D_x(z)$ update?

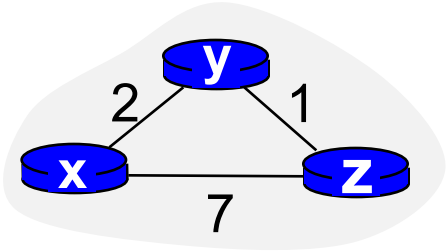
Node y

cost to

from	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

cost to

from	x	y	z
x	0	2	7
y	2	0	1
z	7	1	0



Node z

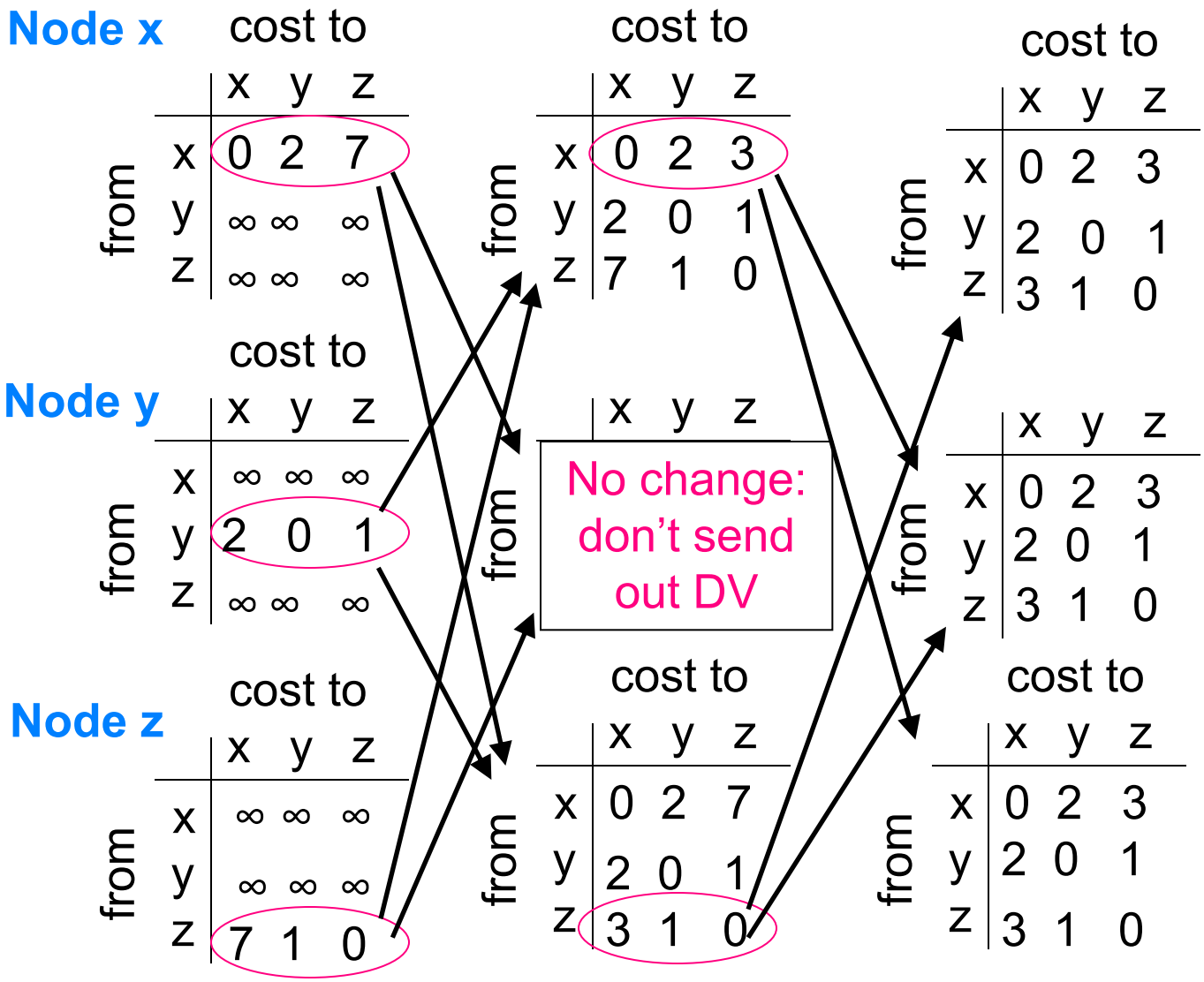
cost to

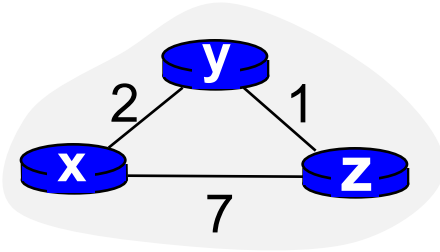
from	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

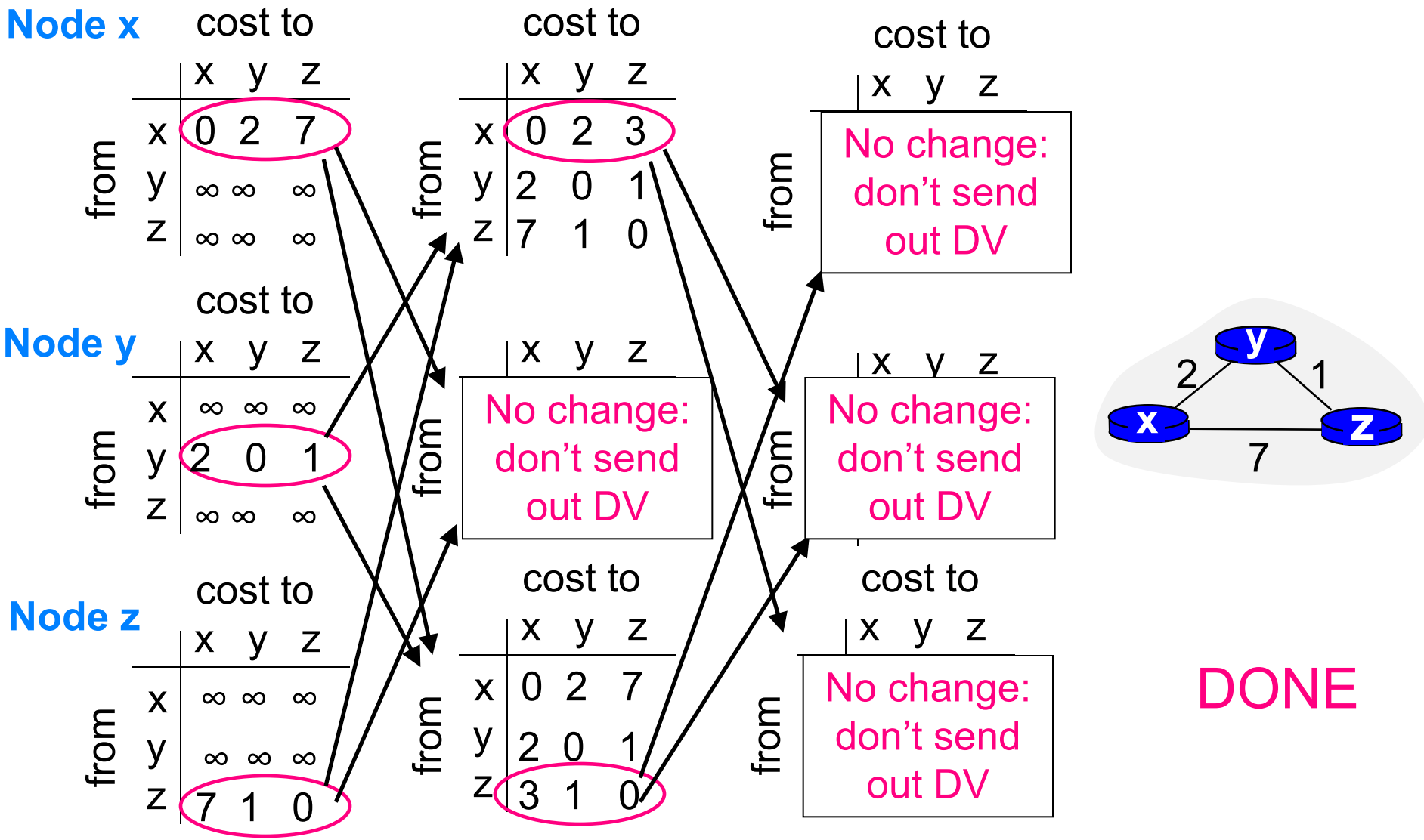
cost to

from	x	y	z
x	0	2	7
y	2	0	1
z	3	1	0

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

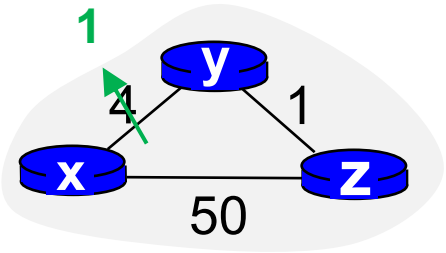

Node x
cost to
x y z
from
x 0 2 7
y ∞ ∞ ∞
z ∞ ∞ ∞
Node y
cost to
x y z
from
x ∞ ∞ ∞
y 2 0 1
z ∞ ∞ ∞
Node z
cost to
x y z
from
x ∞ ∞ ∞
y ∞ ∞ ∞
z 7 1 0
cost to
x y z
from
x 0 2 3
y 2 0 1
z 7 1 0
x y z
No change: don't send out DV
cost to
x y z
from
x 0 2 7
y 2 0 1
z 3 1 0
cost to
x y z
from
x 0 2 3
y 2 0 1
z 3 1 0
x y z
from
x 0 2 3
y 2 0 1
z 3 1 0
cost to
x y z
from
x 0 2 3
y 2 0 1
z 3 1 0


y
2
1
x
z
7


Node x
cost to
x y z
from
x 0 2 7
y ∞ ∞ ∞
z ∞ ∞ ∞
Node y
cost to
x y z
from
x ∞ ∞ ∞
y 2 0 1
z ∞ ∞ ∞
Node z
cost to
x y z
from
x ∞ ∞ ∞
y ∞ ∞ ∞
z 7 1 0
cost to
x y z
from
x 0 2 3
y 2 0 1
z 7 1 0
No change: don't send out DV
cost to
x y z
from
x 0 2 7
y 2 0 1
z 3 1 0
No change: don't send out DV
No change: don't send out DV
No change: don't send out DV
y
x
z
2
1
7
DONE
time

Node detects local link cost change

1. Updates routing info
2. Recalculates DV
3. If DV changes, notify neighbors



Good news travels fast

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*

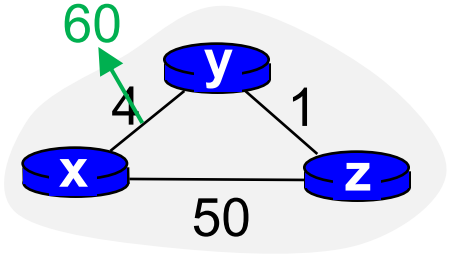
Bad news travels slow

Count to infinity problem

– 44 iterations before algorithm stabilizes

Intuitively

– when z tells y it has a path to x, y has no way of knowing that z is using y on its path



cost to

from Y	x	y	z
x	0	4	3
y	4	0	1
z	5	1	0

cost to

from Y	x	y	z
x	0	4	3
y	6	0	1
z	5	1	0

3 min: Compute new $D_y(x)$ and $D_z(x)$ after change

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

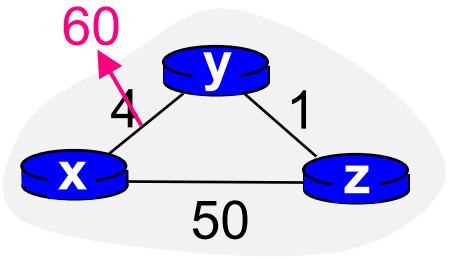
$D_z(x) = \min\{c(z,x) + D_x(x),\ c(z,y) + D_y(x)\}$
$= \min\{50+0\ ,\ 1+6\} = 7$
→ Count-to-infinity

Problem arises because y still expects z can get to x with cost of 5

A proposed solution: 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)



Y (from) \ cost to	x	y	z
x	0	4	5
y	4	0	1
z	∞	1	0

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

Q: Will this completely solve count to infinity problem?

– no, only for 2 node loops

Another proposed solution: hold time

– don't process route updates for period of time after route retraction

– ameliorates problem but does not solve

Distance vector routing summary

Easy to implement

- you will implement for hw9 :-)

Distributed

- x doesn't compute paths in isolation
- requires route info (path costs) computed by neighbors

Iterative

- x updates its DV whenever
  - local link costs change
  - DV update received from nbr

Asynchronous

- updates, exchanges happen asynchronously

Self-terminating

- x stops updating DV when no more changes received

Control Plane

LINK STATE VS. DISTANCE VECTOR ROUTING

Message complexity

n nodes
E links

Link state

- O(nE) messages sent
  - every node floods its link state message out over every link in network to reach every node
- smaller messages
  - message size depends on the number of neighbors a node has
  - any link change requires a broadcast

Distance vector

- # of messages depends on convergence time which varies
  - nodes only exchange messages between neighbors
- larger routing update messages
  - message size is proportional to the number of nodes in the network
  - if link changes don't affect shortest path, no message exchange

Speed of convergence

n nodes
E links

Link state

- $\sum_{i=1}^{n-1} i = n(n+1)/2 = O(n^2)$
  - search through n-1 nodes to find min, recompute routes
  - search through n-2 nodes to find min, recompute routes
  - …
- converges quickly but may have oscillations
  - route computation is centralized
  - a node stores a complete view of the network

Distance vector

- slow to converge and convergence time varies
  - route computation is distributed
- may be routing loops, count-to-infinity problem

What happens if router malfunctions?

n nodes
E links

Link state

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

Distance vector

- DV node can advertise incorrect path cost
- each node's DV used by others: errors propagate through network

Both have strengths and weaknesses.
One or the other is used in almost every network

Internet Routing

OVERVIEW

From graph algorithms to routing protocols

Need to address Internet reality

1. Internet is network of networks
   - hierarchical structure
   - routers not all identical
     - some routers connect different networks together
   - each network admin may want to control routing in its own network

2. Scalability with billions of destinations
   - don't all fit in one routing table
   - can't exchange routing tables this big
     - would use all link capacity

Scalable routing on the Internet

Aggregate routers into regions called Autonomous Systems

Autonomous Systems (AS)

- aka domain
- network under single administrative control
  - company, university, ISP, …
- 30,000+ ASes: AT&T, IBM, Wesleyan …
- each AS has a unique 16-bit AS #
  - Wesleyan: AS167
  - BBN: used to be AS1: was first org to get AS # then L3 later acquired

```
AS160   U-CHICAGO-AS - University of Chicago, US
AS161   TI-AS - Texas Instruments, Inc., US
AS162   DNIC-AS-00162 - Navy Network Information Center (NNIC), US
AS163   IBM-RESEARCH-AS - International Business Machines Corporation
AS164   DNIC-AS-00164 - DoD Network Information Center, US
AS165   DNIC-AS-00165 - DoD Network Information Center, US
AS166   IDA-AS - Institute for Defense Analyses, US
AS167   WESLEYAN-AS - Wesleyan University, US
AS168   UMASS-AMHERST - University of Massachusetts, US
AS169   HANSCOM-NET-AS - Air Force Systems Networking, US
```

Hierarchical routing

2-level route propagation hierarchy

1. intra AS routing protocol between routers in same AS
   - aka intra domain routing protocol
   - aka interior gateway protocol
   - each AS selects its own

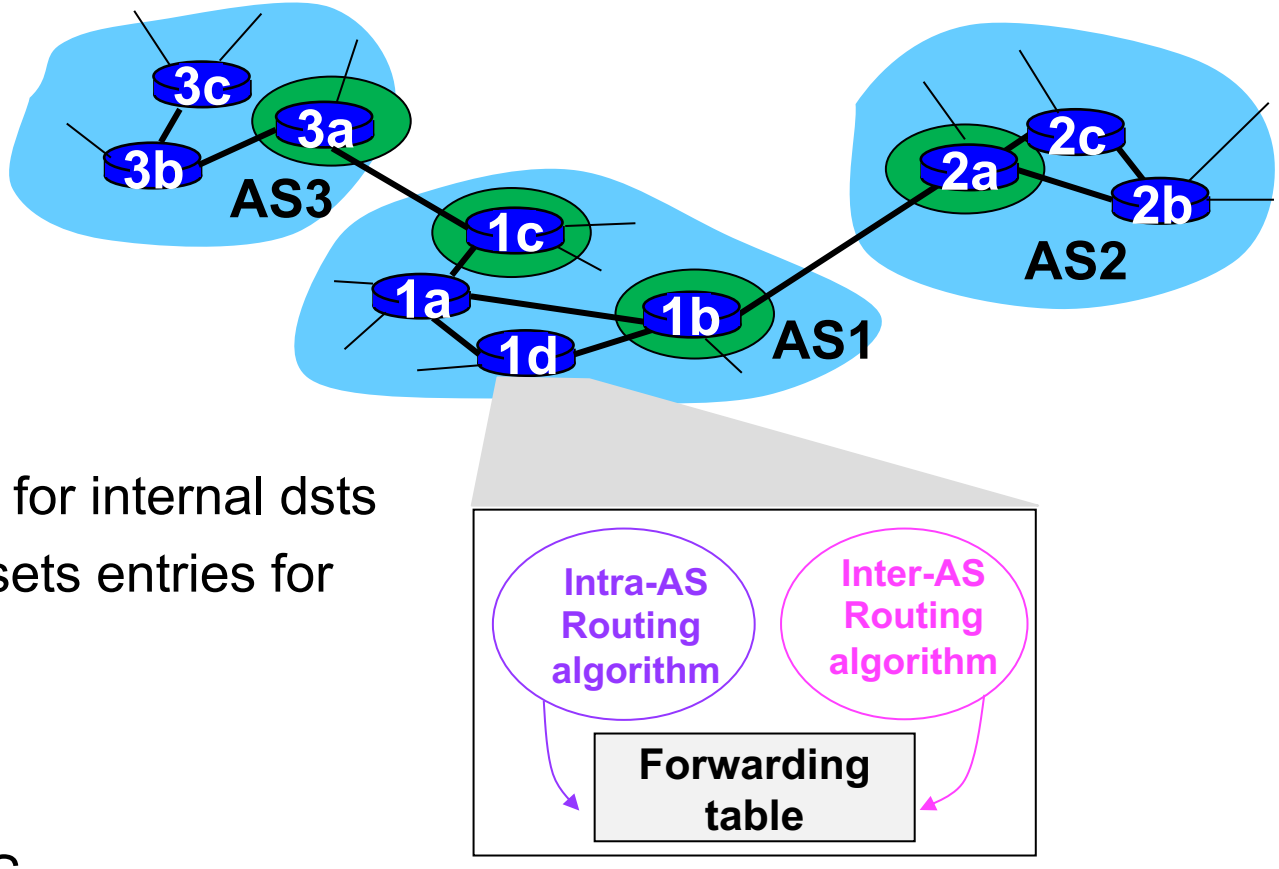
   Focus is performance

2. inter AS routing protocol between gateway routers in different ASes
   - aka inter domain routing protocol
   - aka exterior gateway protocol
   - Internet-wide standard

   Policy may dominate performance

Q: Can routers in different ASes run different intra AS routing protocol?

Hierarchical routing



Forwarding table

- intra-AS sets entries for internal dsts
- inter-AS & intra-AS sets entries for external dsts

Gateway router

- at edge of its own AS
- direct link to router in another AS
- perform inter-AS as well as intra-AS routing
- distributes results of inter-AS routing to other routers in AS

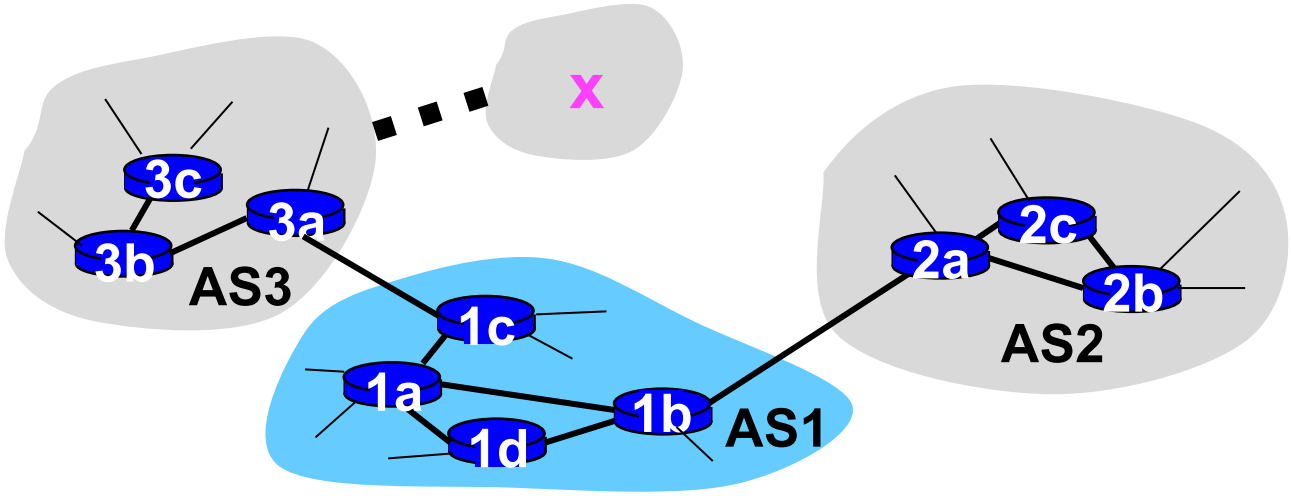
Example: set forwarding table in router 1d

AS1 learns (from inter-AS protocol)

- subnet **x** is reachable via AS3 (gateway 1c) but not via AS2

Router 1d learns (from intra-AS protocol)

- that its interface **y** is on least cost path to 1c.
- installs forwarding table entry **(x,y)**



Q: What if multiple ASes can be used to reach x?

INTERNET CONTROL MESSAGE PROTOCOL

OVERVIEW

Internet Control Message Protocol (ICMP)

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 pkts

ICMP message

- type, code plus first 8 bytes of IP pkt 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 segments or packets to destination

- first set has TTL =1
- second set has TTL=2, etc.
- unlikely port number

When *n*th set arrives to nth router

- router discards and sends source ICMP message (type 11, code 0)
- ICMP message includes name of router & IP address

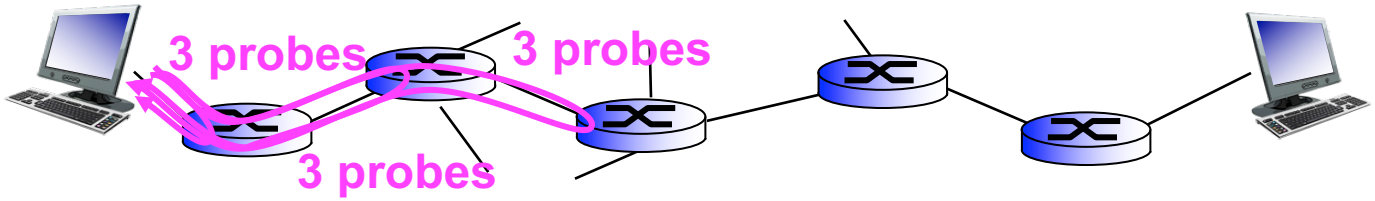
When ICMP msg arrives

- source records RTTs

Stopping criteria

TCP segment or UDP datagram eventually arrives at dst host

- dst returns ICMP "port unreachable" message
- source stops



Q: why can traceroute work with segments, datagrams, or packets?

ICMP traceroute

We're generating an ICMP echo request

Intermediate routers

- respond with ICMP ttl expired

Final destination

- responds with ICMP echo reply

NETWORK PROGRAMMING

BIT-WISE OPERATIONS IN PYTHON

Bit-wise operations on variables

x << y

- returns x with bits shifted to left by y places
  - new bits on right-hand-side are zeros
  - same as multiplying x by 2^y

x >> y

- returns x with bits shifted to right by y places
  - same as dividing x by 2^y

x & y

- does a bitwise and
  - each bit of output is 1 if corresponding bit of x AND of y is 1, otherwise 0

~ x

- returns complement of x
  - number you get by switching each 1 for 0 and each 0 for 1

E.g.,

- use to pack ip_version and ip header length into 8 bits

https://wiki.python.org/moin/BitwiseOperators
https://www.tutorialspoint.com/python3/bitwise_operators_example.htm