# Lecture 19: Network Layer
# Routing in the Internet

COMP 332, Fall 2018

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 due Thurs. at 11:59p, programming due next Wed.
   - 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)

## Control Plane

# DISTANCE VECTOR ROUTING

# Distance vector algorithm run at each node x

## Initialization

For all dst $y \in N$

if y is nbr of x

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

else

$D_x(y) = \infty$

For each nbr w and dst $y \in N$

$D_w(y) = \infty$

Send x's DV to all nbrs w

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

## Loop

*x waits* for change in local link cost or DV msg from neighbor

*recompute* estimates

$D_x(y) = \min_v \{ c(x,v) + D_v(y) \}$

if x's DV to any dst has changed, *notify* neighbors

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 |

**Node y**

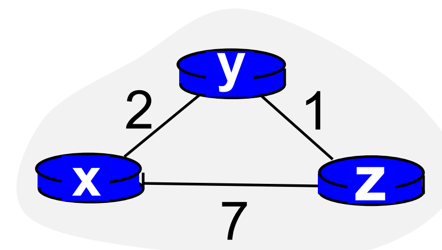
cost to

| from | x | y | z |
|---|---|---|---|
| x | ∞ | ∞ | ∞ |
| y | 2 | 0 | 1 |
| z | ∞ | ∞ | ∞ |

**Node z**

cost to

| from | 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**

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 |

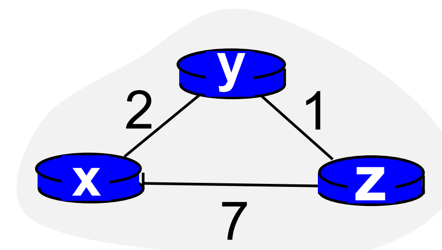
**Node y**

cost to (from)

| | x | y | z |
|---|---|---|---|
| x | ∞ | ∞ | ∞ |
| y | 2 | 0 | 1 |
| z | ∞ | ∞ | ∞ |

(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 |

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

time

## 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 |

cost to

| from | x | y | z |
|---|---|---|---|
| x | 0 | 2 | 3 |
| y | 2 | 0 | 1 |
| z | 3 | 1 | 0 |

## Node y

cost to

| from | x | y | z |
|---|---|---|---|
| x | ∞ | ∞ | ∞ |
| y | 2 | 0 | 1 |
| z | ∞ | ∞ | ∞ |

| from | x | y | z |
|---|---|---|---|

No change: don't send out DV

| from | x | y | z |
|---|---|---|---|
| x | 0 | 2 | 3 |
| y | 2 | 0 | 1 |
| z | 3 | 1 | 0 |

## Node z

cost to

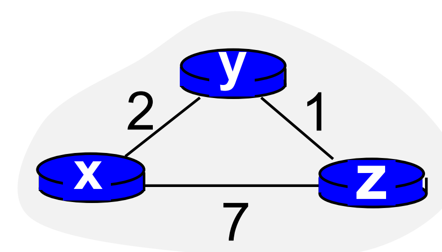
| from | x | y | z |
|---|---|---|---|
| x | ∞ | ∞ | ∞ |
| y | ∞ | ∞ | ∞ |
| z | 7 | 1 | 0 |

cost to

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

cost to

| from | x | y | z |
|---|---|---|---|
| x | 0 | 2 | 3 |
| y | 2 | 0 | 1 |
| z | 3 | 1 | 0 |




time

**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 |

| cost to (from) | x | y | z |
|---|---|---|---|

No change: don't send out DV

**Node y**

| cost to (from) | x | y | z |
|---|---|---|---|
| x | ∞ | ∞ | ∞ |
| y | 2 | 0 | 1 |
| z | ∞ | ∞ | ∞ |

| from | x | y | z |
|---|---|---|---|

No change: don't send out DV

| from | x | y | z |
|---|---|---|---|

No change: don't send out DV

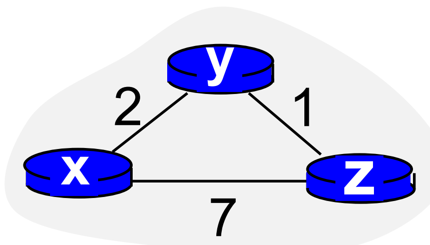
**Node z**

| 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 |

| cost to (from) | x | y | z |
|---|---|---|---|

No change: don't send out DV

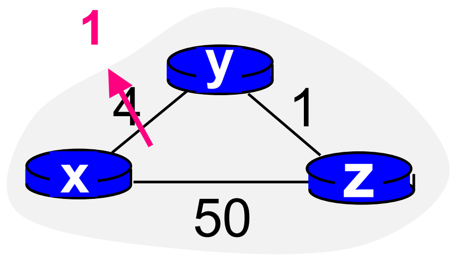



DONE

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*

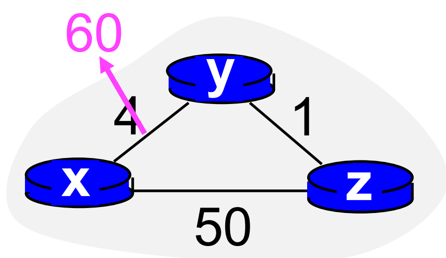
# Bad news travels slow

## 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




Y (from / cost to):

| from \ cost to | x | y | z |
|---|---|---|---|
| x | 0 | 4 | 3 |
| y | 4 | 0 | 1 |
| z | 5 | 1 | 0 |

⇒

Y (from / cost to):

| from \ cost to | x | y | z |
|---|---|---|---|
| x | 0 | 4 | 3 |
| y | 6 | 0 | 1 |
| z | 5 | 1 | 0 |

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

$= \min\{60+0, 1+5\} = 6$

→ Routing Loop

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

$= \min\{50+0 , 1+6\} = 7$

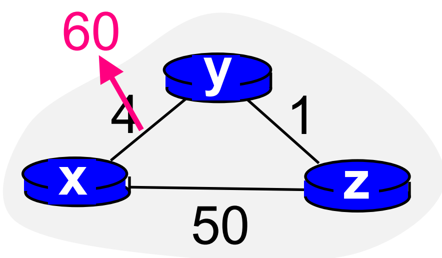
→ Count-to-infinity

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

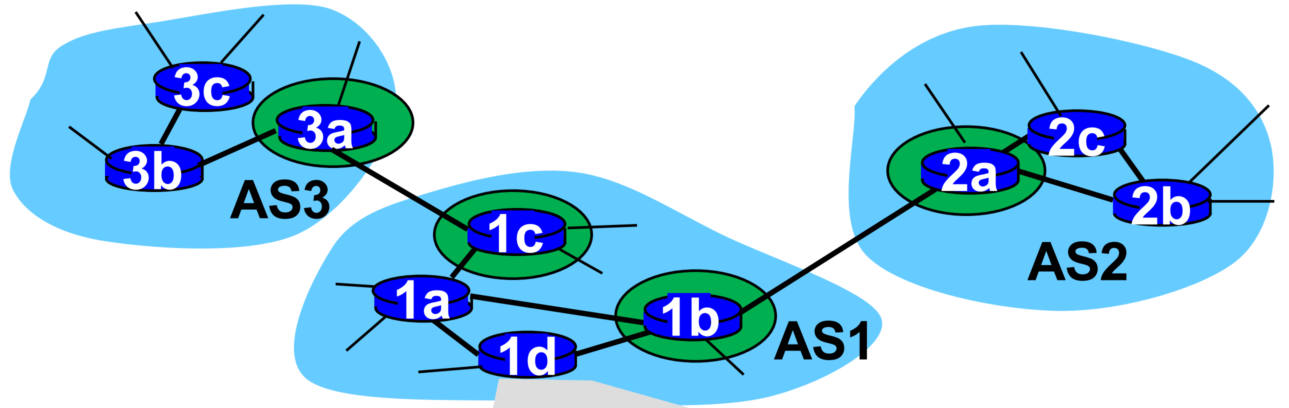
Focus is performance

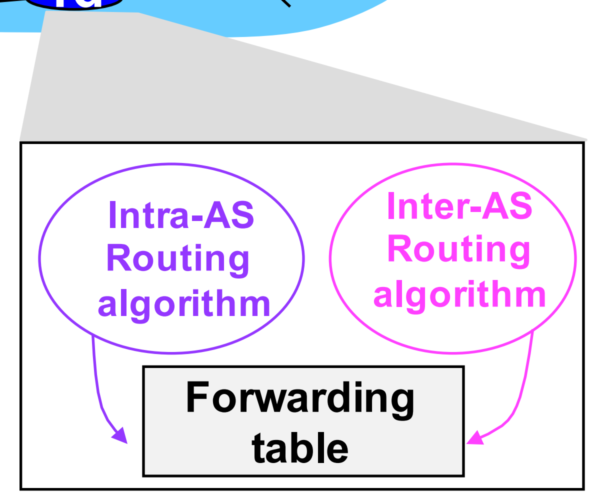
2. inter AS routing protocol between gateway routers in different ASes
   - aka inter domain routing protocol
   - aka exterior gateway protocol
   - Internet-wide standard

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

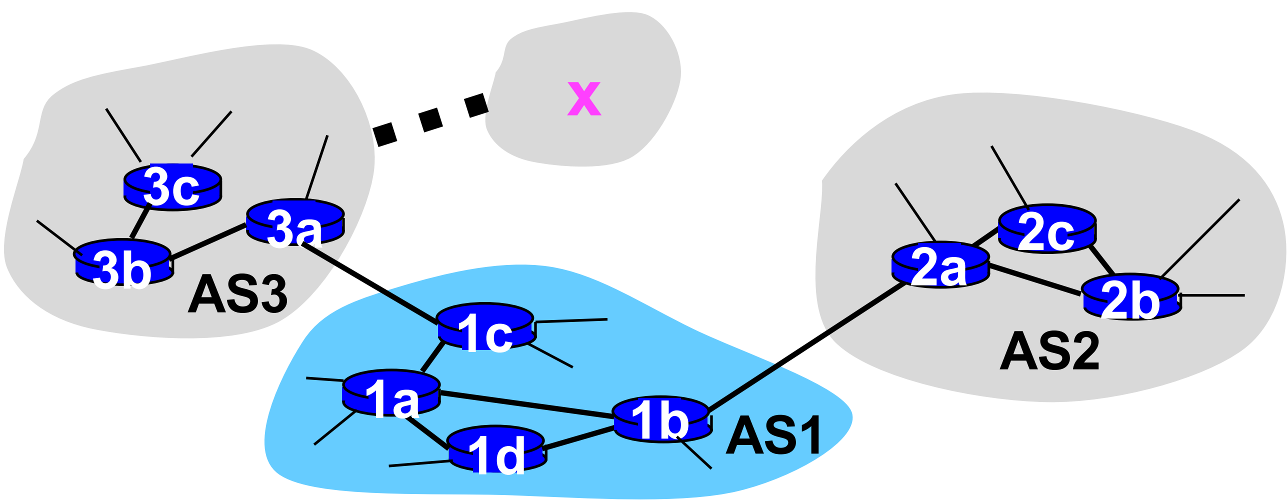
## AS1 learns (from inter-AS protocol)

– 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

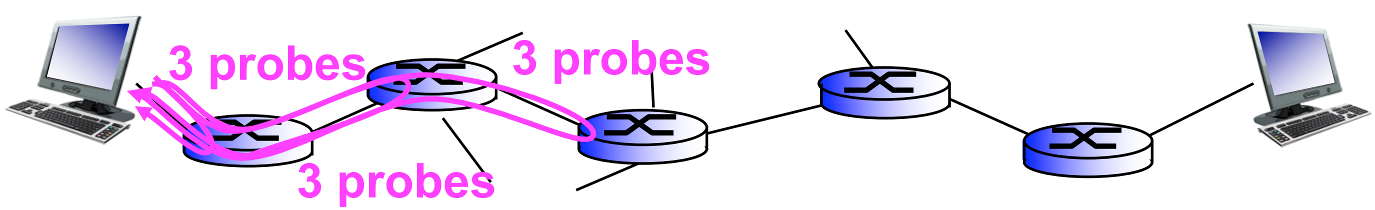
When ICMP msg arrives

- 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