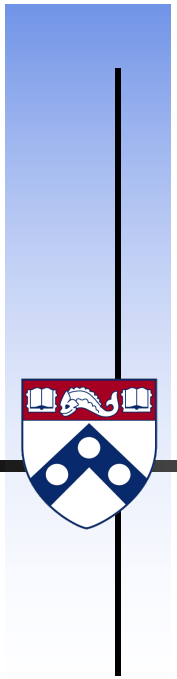


CIS 553: Networked Systems

Intradomain Routing

February 17, 2020

Agenda

- IPv6 ✓
- Discovery ✓
  - DNS NEXT
  - ARP
  - DHCP
- Intradomain Routing
  - Distance Vector
  - Link State

Goals: Are we there yet?

- Uniqueness: No naming conflicts
- Scalable
- Distributed, autonomous administration
- Highly available?
- Lookups are fast?

DNS Reliability

- Root servers are replicated
- Authoritative servers are also replicated
  - Parent can return multiple name servers
- When a request times out, retransmit
  - Exponential backoff when retrying same server
  - Can also try alternate servers
- Same identifier for all queries
  - Don't care which server responds

Goals: Are we there yet?

- Uniqueness: No naming conflicts
- Scalable
- Distributed, autonomous administration
- Highly available
- Fast lookups?

DNS caching

- Performing all these queries takes time
  - Up to 1-second latency before starting download
  - Need to query every time
- Caching can greatly reduce overhead
  - The top-level servers very rarely change
  - Popular sites (e.g., www.cnn.com) visited often
  - Local DNS server often has the information cached
- Cached data periodically times out
  - Lifetime (TTL) of data controlled by owner of data
  - TTL passed with every record

Setting the Time To Live (TTL)

- TTL trade-offs
  - Small TTL: fast response to change
  - Large TTL: higher cache hit rate
- Following the hierarchy
  - Top of the hierarchy: days or weeks
  - Bottom of the hierarchy: seconds to hours

Negative caching

- Remember things that don't work
  - Misspellings like www.cnn.comm and www.cnnn.com
  - These can take a long time to fail the first time
  - Good to remember that they don't work so the failure takes less time the next time around
- Negative caching is now required
  - ~85-90% actually do it

DNS Resource Records

DNS: distributed DB storing resource records (RR)

RR format: (name, value, type, ttl)

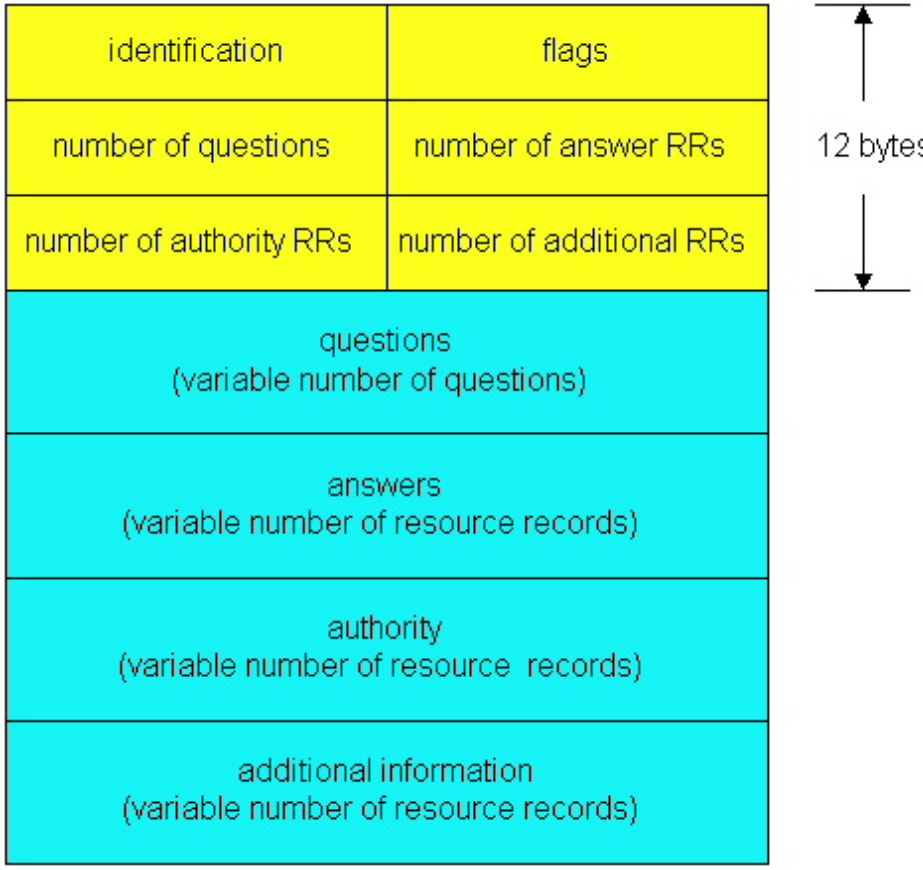
- Type: A
  - name is hostname
  - value is IP address
- Type: NS
  - name is domain (e.g. foo.com)
  - value is hostname of authoritative name server for this domain
- Type: CNAME
  - name is alias name for some "canonical" (the real) name

    www.ibm.com is really

    servereast.backup2.ibm.com
  - value is canonical name
- Type: MX
  - value is name of mailserver associated with name

DNS Protocol

- Queries and replies both have the same format
- Identification (16 bits)
  - Unique # for query
  - Reply uses same #
- Flags
  - Query or reply
  - Recursion desired
  - Recursion available
  - Reply is authoritative



DNS provides indirection

- Addresses can change underneath
  - Move www.cnn.com to 4.125.91.21
- Multiple names for the same address
  - E.g., many services (mail, www) on same machine
  - E.g., aliases like www.cnn.com and cnn.com
- Later: Name could map to multiple IP addresses!
  - Load-balancing
  - Reducing latency by picking nearby servers

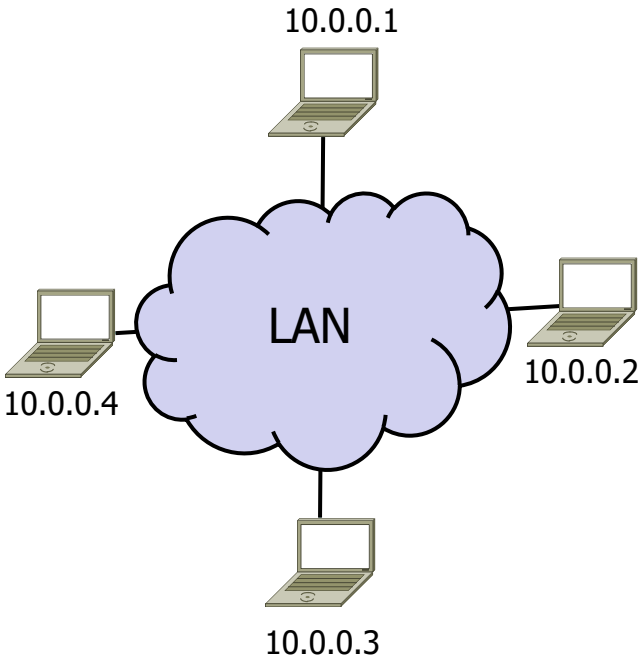
Agenda

- IPv6 ✓
- Discovery ✓
  - DNS ✓
  - ARP NEXT
  - DHCP
- Intradomain Routing
  - Distance Vector
  - Link State

Different Kinds of Names

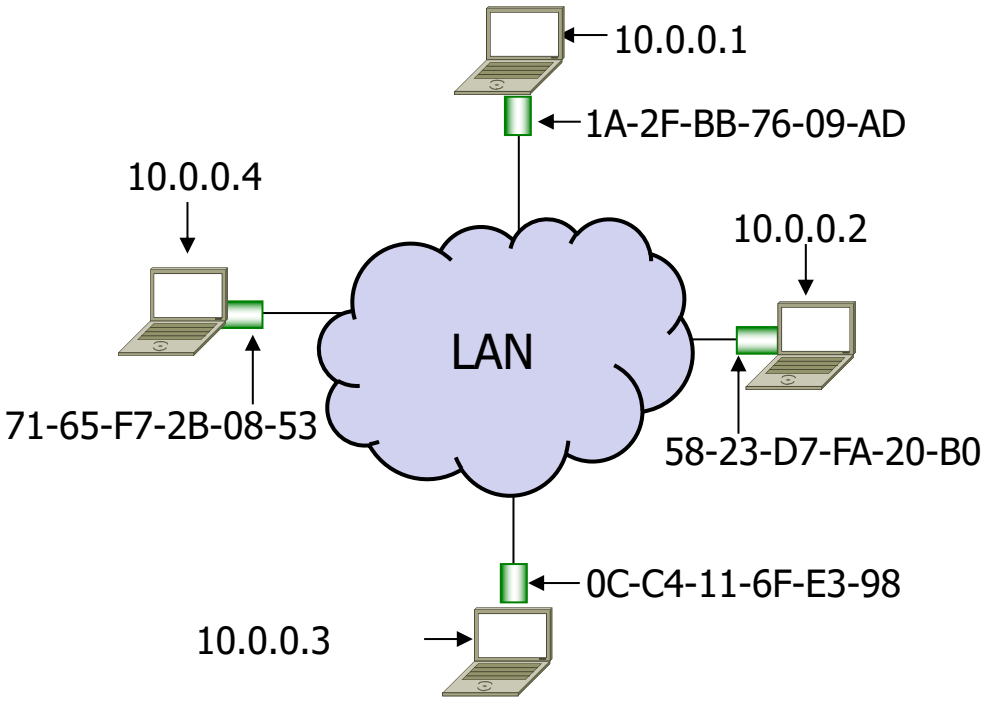
- Host name (e.g., www.cis.upenn.edu)
  - Mnemonic, variable-length, appreciated *by humans*
  - Hierarchical, based on organizations (who)
- IP address (e.g., 158.130.69.163)
  - Numerical 32-bit address appreciated *by routers*
  - Hierarchical, based on organizations and topology (where)
- MAC address (e.g., 00-15-C5-49-04-A9)
  - Numerical 48-bit address appreciated *by adapters*
  - Non-hierarchical, unrelated to network topology

Consider a single LAN



- Need to fill in the L2 header before giving the frame to Ethernet
  - Otherwise, switches won't know what to do
  - Non-promiscuous receivers will drop it
- How do I find an interface's MAC address given its IP address?

ARP: Address Resolution Protocol



ARP table: each IP node (host, router) on LAN has table

- IP/MAC address mappings for some LAN nodes:

  < IP address; MAC address; TTL>

- TTL (Time To Live): time after which address mapping will be forgotten (typically 20 min)

ARP Protocol: Same LAN

- A wants to send frame to B
  - B's MAC address not in A's ARP table.
- A broadcasts ARP query packet, containing B's IP address
  - dest MAC address = FF-FF-FF-FF-FF-FF
  - all nodes on LAN receive ARP query
- B receives ARP packet, replies to A with its (B's) MAC address
  - frame sent to A's MAC address (unicast)
- A caches (saves) IP-to-MAC address pair in its ARP table until information becomes old (times out)
  - soft state: information that times out (goes away) unless refreshed
- ARP is "plug-and-play":
  - nodes create their ARP tables *without intervention from net administrator*

Key ideas in ARP (and learning switches and DHCP)

- Broadcasting: Can use broadcast to make contact
  - Scalable because of limited size
- Caching: remember the past for a while
  - Store the information you learn to reduce overhead
- Soft state: eventually forget the past
  - Associate a time-to-live field with the information
  - … and either refresh or discard the information
  - Key for robustness in the face of unpredictable change

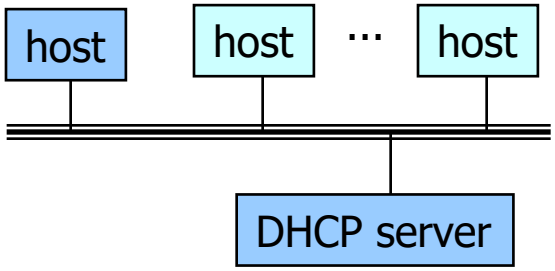
Agenda

- IPv6 ✓
- Discovery ✓
  - DNS ✓
  - ARP ✓
  - DHCP NEXT
- Intradomain Routing
  - Distance Vector
  - Link State

Bootstrapping Problem

- Host doesn't have an IP address yet
  - So, host doesn't know what source address to use
- Host doesn't know who to ask for an IP address
  - So, host doesn't know what destination address to use

- Solution: broadcast to discover a server who can help
  - Broadcast a DHCP server-discovery message



DHCP

- Dynamic Host Configuration Protocol
  - Defined in RFC 2131
- A host uses DHCP to discover
  - Its own IP address
  - IP address(es) for its local DNS server(s)
  - Basic routing information
    - IP address of gateway router
    - Prefix length of LAN

DHCP: Operation

- One or more local DHCP servers maintain required information
  - IP address pool, netmask, DNS servers, etc.
  - Application that listens on UDP port 67

DHCP: Operation

- One or more local DHCP servers maintain required information

Phase 1:

- Client broadcasts a DHCP discovery message
  - L2 broadcast, to MAC address FF:FF:FF:FF:FF:FF

DHCP: Operation

- One or more local DHCP servers maintain required information

Phase 1:

- Client broadcasts a DHCP discovery message
- One or more DHCP servers responds with a DHCP "offer" message
  - Proposed IP address for client, lease time
  - Other parameters

DHCP: Operation

- One or more local DHCP servers maintain required information

Phase 1:

- Client broadcasts a DHCP discovery message
- One or more DHCP servers responds with a DHCP "offer" message

Phase 2:

- Client broadcasts a DHCP request message
  - Specifies which offer it wants
  - Echoes accepted parameters
  - Other DHCP servers learn they were not chosen

DHCP: Operation

- One or more local DHCP servers maintain required information

Phase 1:

- Client broadcasts a DHCP discovery message
- One or more DHCP servers responds with a DHCP "offer" message

Phase 2:

- Client broadcasts a DHCP request message
- Selected DHCP server responds with an ACK

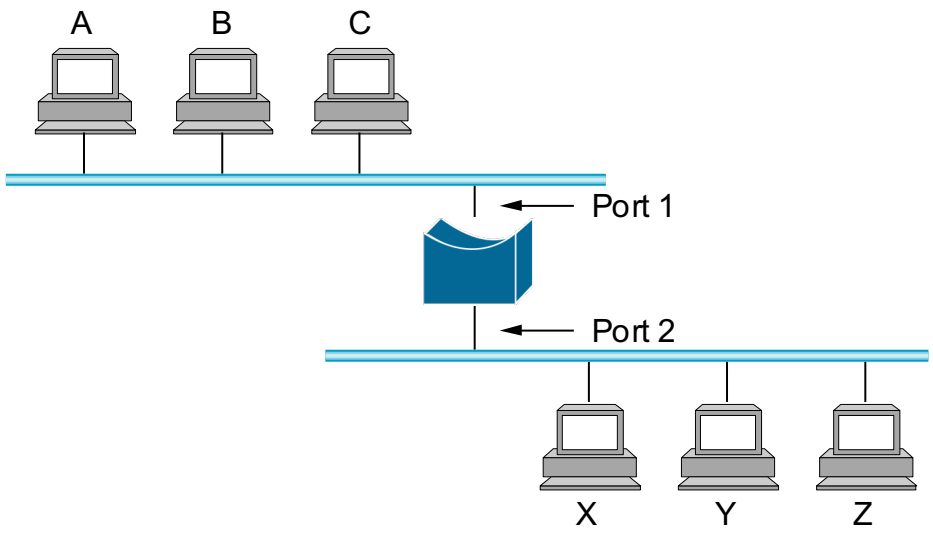
Agenda

- IPv6 ✓
- Discovery ✓
  - DNS ✓
  - ARP ✓
  - DHCP ✓
- Intradomain Routing 

  - Distance Vector
  - Link State

Recap: Forwarding

- Directing a packet to the correct interface so that it progresses to its destination
  - Read address from packet header
  - Search forwarding table



Host	Port
A	1
B	1
C	1
X	2
Y	2
Z	2

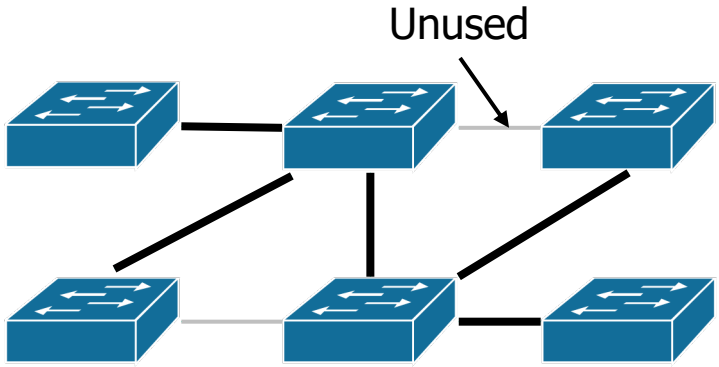
Forwarding table

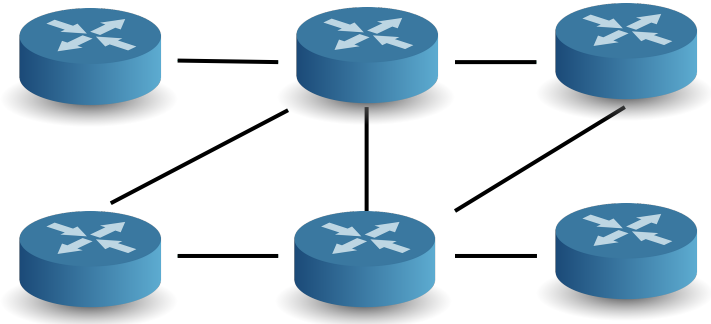
Forwarding vs. Routing

- Forwarding
  - Directing one data packet
  - Each router using local routing state
- Routing
  - Computing the forwarding tables that guide packets
  - Jointly computed by routers using a distributed algorithm

Improving on the Spanning Tree

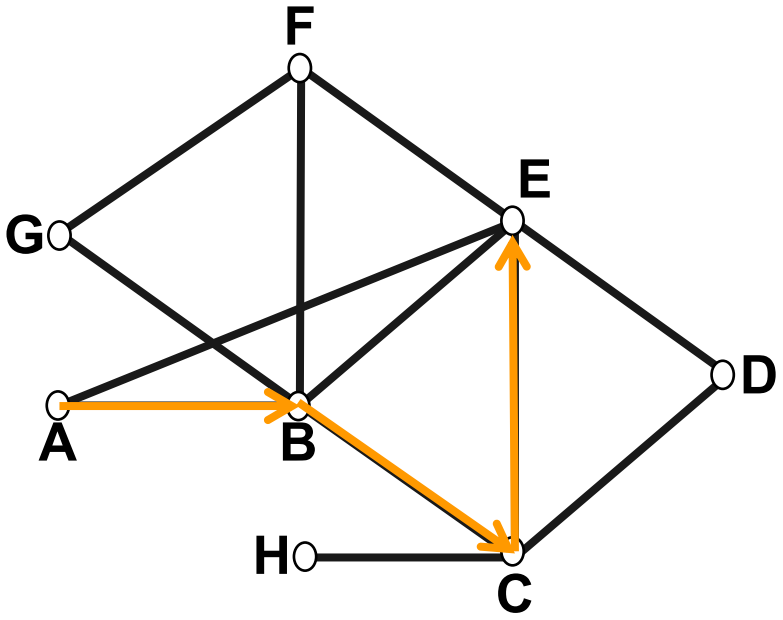
- Spanning tree provides basic connectivity
  - Wastes capacity
  - No guarantee of optimality
- Routing implies intelligence
  - Use all links to find "best" paths





Routing

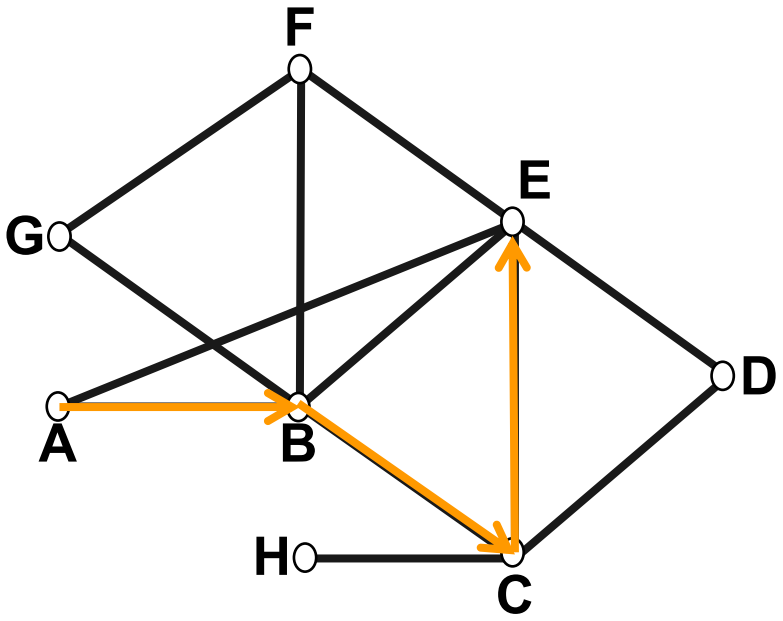
- Network modeled as a graph
  - Routers → nodes
  - Link →edges
- Goal: determine the "best" path through the network from source to destination



What does "best" mean?

What are "best" paths anyhow?

- Many possibilities:
  - Latency, avoid circuitous paths
  - Bandwidth, avoid slow links
  - Money, avoid expensive links
  - Hops, to reduce switching
- We'll only consider topology
  - Ignore workload, e.g., hotspots



Shortest Paths

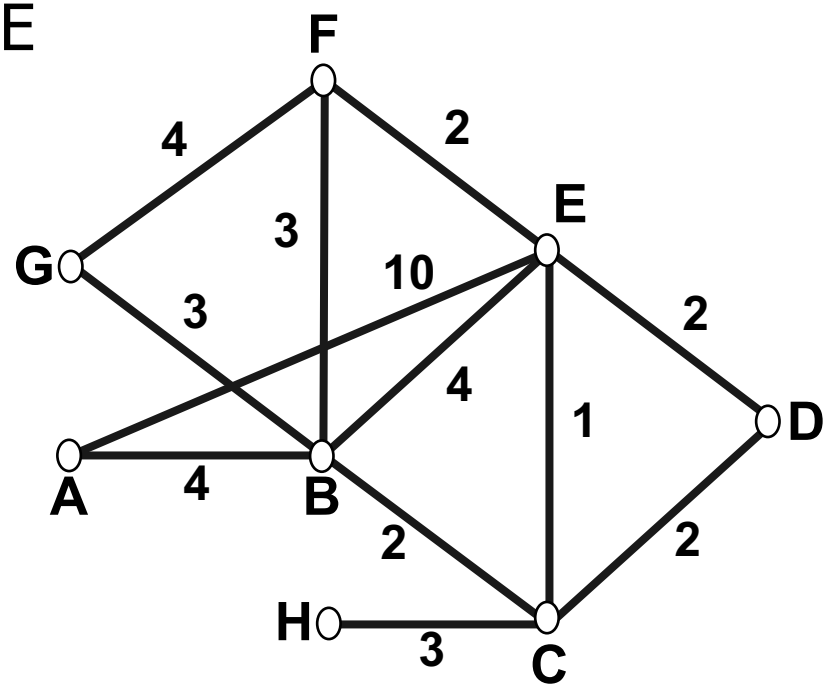
We'll approximate "best" by a cost function that captures the factors

- Often call lowest "shortest"

1. Assign each link a cost (distance)
2. Define best path between each pair of nodes as the path that has the lowest total cost (or is shortest)
3. Pick randomly to any break ties

Shortest Paths (2)

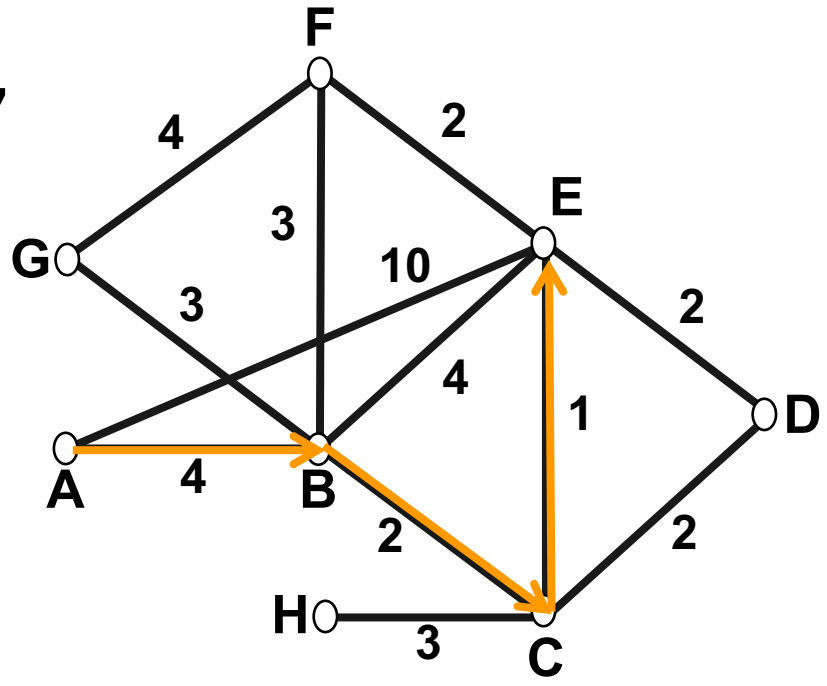
- Find the shortest path A → E
- All links are bidirectional, with equal costs in each direction
  - Can extend model to unequal costs if needed



Shortest Paths (3)

- ABCE is a shortest path
- dist(ABCE) = 4 + 2 + 1 = 7

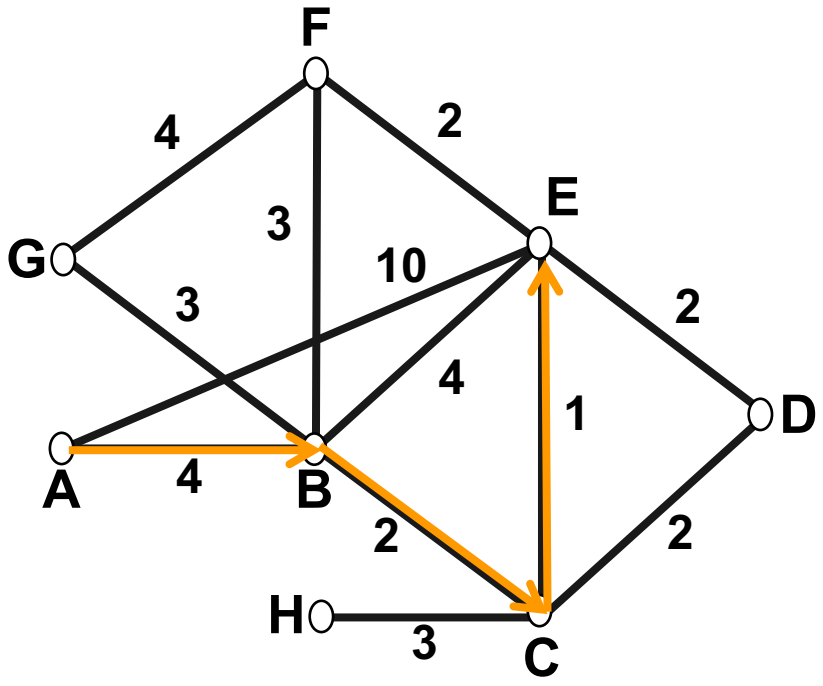
- This is less than:
  - dist(ABE) = 8
  - dist(ABFE) = 9
  - dist(AE) = 10
  - dist(ABCDE) = 10



Shortest Paths (4)

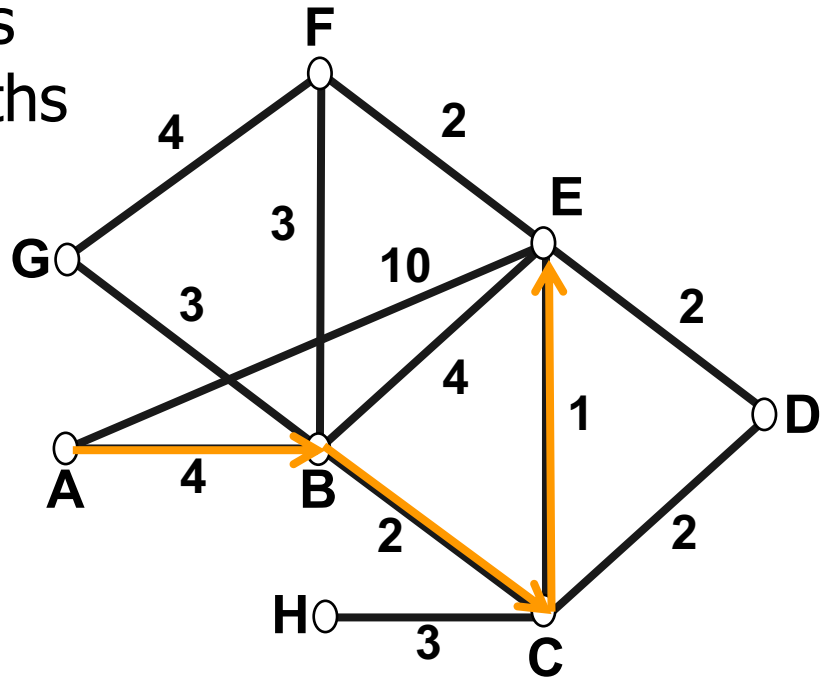
- Optimality property:
  - Subpaths of shortest paths are also shortest paths
- ABCE is a shortest path

  →So are ABC, AB, BCE, BC, CE



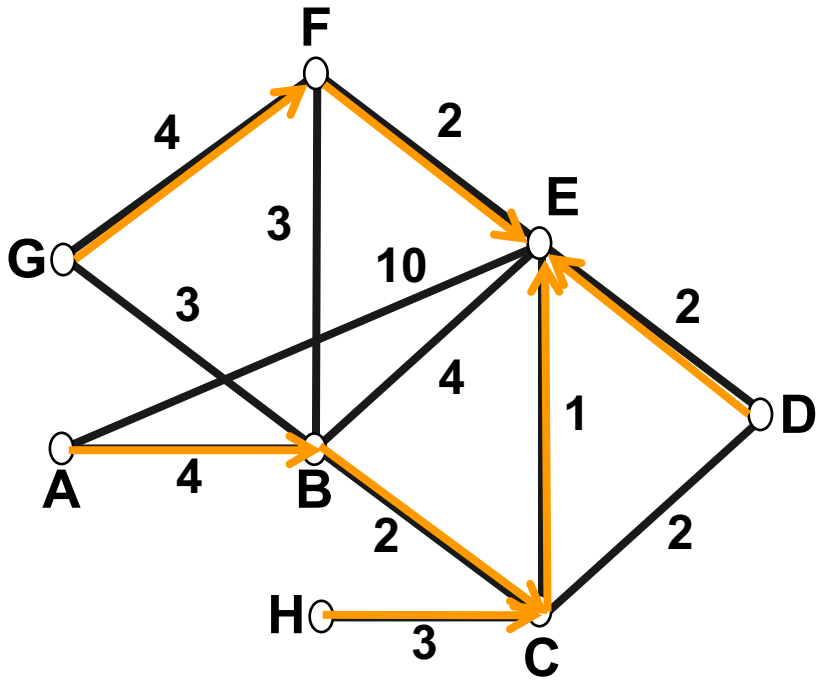
Sink Trees

- Sink tree for a destination is the union of all shortest paths towards the destination
  - Similarly source tree
- Find the sink tree for E



Sink Trees (2)

- Implications:
  - Only need to use destination to follow shortest paths
  - Each node only need to send to the next hop
- Forwarding table at a node
  - Lists next hop for each destination
  - Routing table may know more



Agenda

- IPv6 ✓
- Discovery ✓
  - DNS ✓
  - ARP ✓
  - DHCP ✓
- Intradomain Routing ✓
  - Distance Vector NEXT
  - Link State

Distance Vector

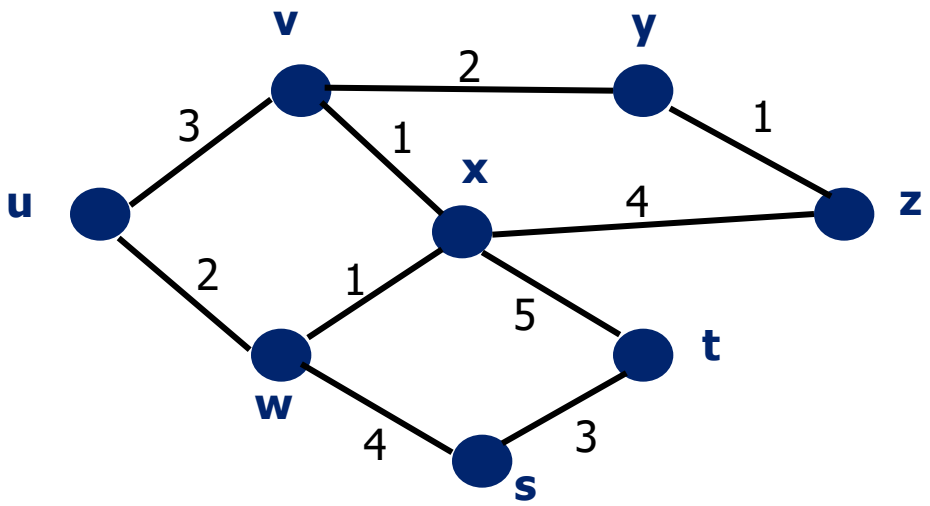
key idea:

- from time-to-time, each node sends its own distance vector estimate to neighbors
- when x receives new DV estimate from neighbor, it updates its own DV

- under minor, natural conditions, routing will eventually converge

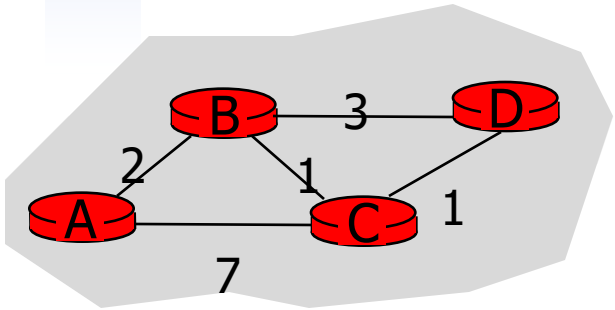
Distance Vector: Bellman-Ford

- Define distances at each node x
  - $d_x(y)$ = cost of least-cost path from x to y
- Update distances based on neighbors
  - $d_x(y) = \min \{c(x,v) + d_v(y)\}$ over all neighbors v



$$d_u(z) = \min\{\, c(u,v) + d_v(z),\; c(u,w) + d_w(z)\}$$

Example: Distance Vector Algorithm



Node A

Dest.	Cost	NextHop
B	∞	-
C	∞	-
D	∞	-

Node B

Dest.	Cost	NextHop
A	∞	-
C	∞	-
D	∞	-

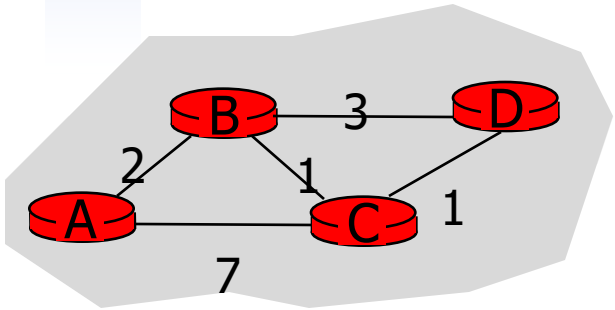
Node C

Dest.	Cost	NextHop
A	∞	-
B	∞	-
D	∞	-

Node D

Dest.	Cost	NextHop
A	∞	-
B	∞	-
C	∞	-

Example: 0th Iteration



Node A

Dest.	Cost	NextHop
B	2	B
C	7	C
D	∞	-

Node B

Dest.	Cost	NextHop
A	2	A
C	1	C
D	3	D

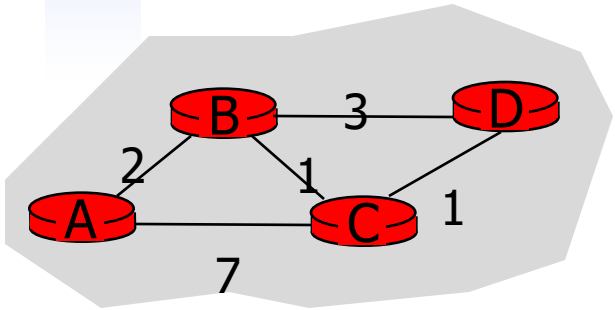
Node C

Dest.	Cost	NextHop
A	7	A
B	1	B
D	1	D

Node D

Dest.	Cost	NextHop
A	∞	-
B	3	B
C	1	C

Example: 1st Iteration (A's Updates)



Node A

Dest.	Cost	NextHop
B	2	B
C	3	B
D	5	B

Node B

Dest.	Cost	NextHop
A	2	A
C	1	C
D	3	D

D(A,D) = D(A,B) + D(B,D) = 2 + 3 = 5

D(A,C) = D(A,B) + D(B,C) = 2 + 1 = 3

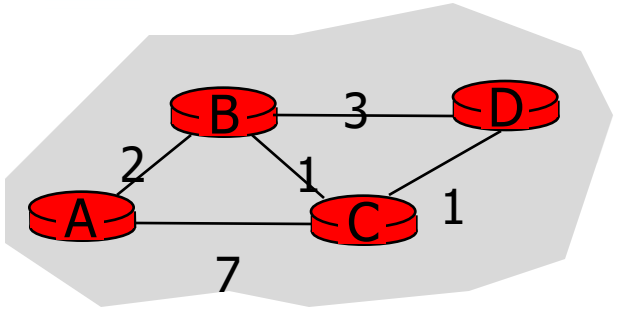
Node C

Dest.	Cost	NextHop
A	7	A
B	1	B
D	1	D

Node D

Dest.	Cost	NextHop
A	∞	-
B	3	B
C	1	C

Example: End of 1st Iteration



Node A

Dest.	Cost	NextHop
B	2	B
C	3	B
D	5	B

Node B

Dest.	Cost	NextHop
A	2	A
C	1	C
D	2	C

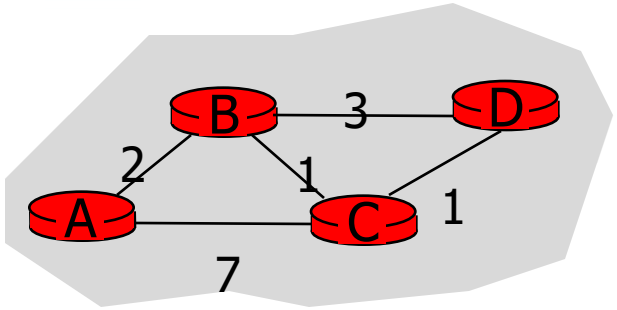
Node C

Dest.	Cost	NextHop
A	3	B
B	1	B
D	1	D

Node D

Dest.	Cost	NextHop
A	5	B
B	2	C
C	1	C

Example: End of 2nd Iteration



Node A

Dest.	Cost	NextHop
B	2	B
C	3	B
D	4	B

Node B

Dest.	Cost	NextHop
A	2	A
C	1	C
D	2	C

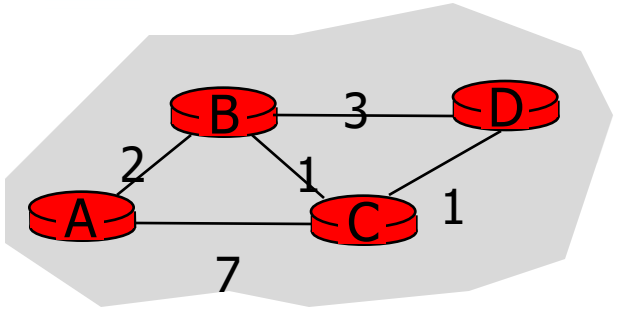
Node C

Dest.	Cost	NextHop
A	3	B
B	1	B
D	1	D

Node D

Dest.	Cost	NextHop
A	4	C
B	2	C
C	1	C

Example: End of 3rd Iteration



Node A

Dest.	Cost	NextHop
B	2	B
C	3	B
D	4	B

Node B

Dest.	Cost	NextHop
A	2	A
C	1	C
D	2	C

Node C

Dest.	Cost	NextHop
A	3	B
B	1	B
D	1	D

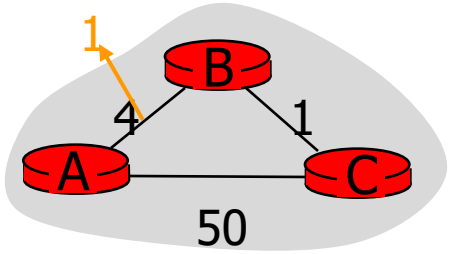
Node D

Dest.	Cost	NextHop
A	4	C
B	2	C
C	1	C

Nothing changes → algorithm has converged

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



Node B

D	C	N
A	4	A
C	1	B

Node C

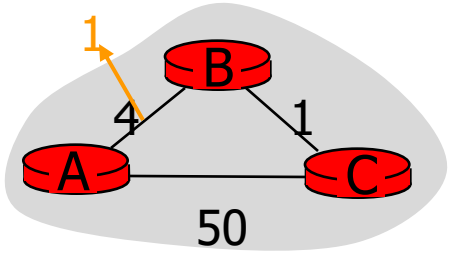
D	C	N
A	5	B
B	1	B

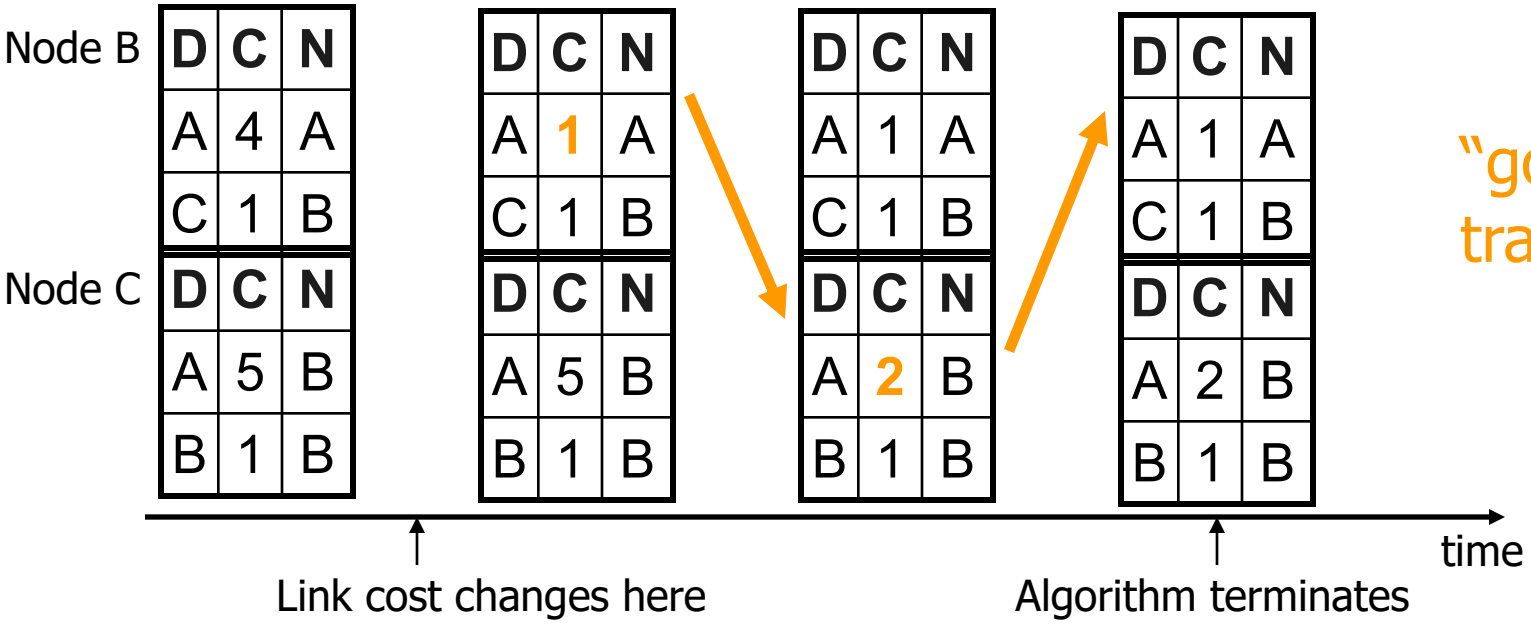
time

Link cost changes here

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

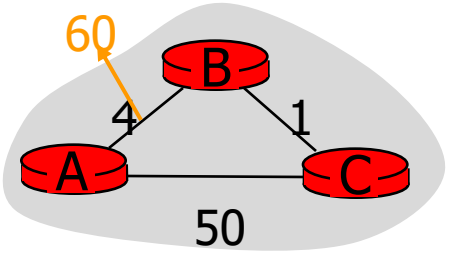




"good news travels fast"

Distance Vector: Count to Infinity Problem

- Link cost changes:
  - node detects local link cost change
  - updates routing info, recalculates distance vector
  - if DV changes, notify neighbors



Node B

D	C	N
A	4	A
C	1	B

Node C

D	C	N
A	5	B
B	1	B

D	C	N
A	6	C
C	1	B

D	C	N
A	5	B
B	1	B

D	C	N
A	6	C
C	1	B

D	C	N
A	7	B
B	1	B

D	C	N
A	8	C
C	1	B

D	C	N
A	7	B
B	1	B

"bad news travels slowly"

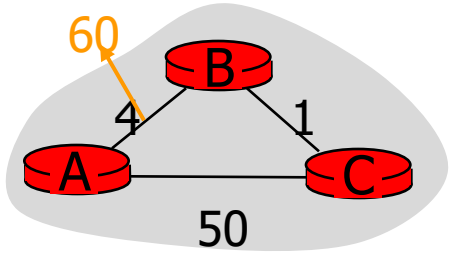
...

time

Link cost changes here; (A,4,A) at node B is invalidated.
Node C advertises D(C,A)=5 to node B. Thus D(B, A) becomes 6!

Distance Vector: Poisoned Reverse

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



Node B

D	C	N
A	4	A
C	1	B

D	C	N
A	60	A
C	1	B

D	C	N
A	60	A
C	1	B

D	C	N
A	51	C
C	1	B

D	C	N
A	51	C
C	1	B

Node C

D	C	N
A	5	B
B	1	B

D	C	N
A	5	B
B	1	B

D	C	N
A	50	A
B	1	B

D	C	N
A	50	A
B	1	B

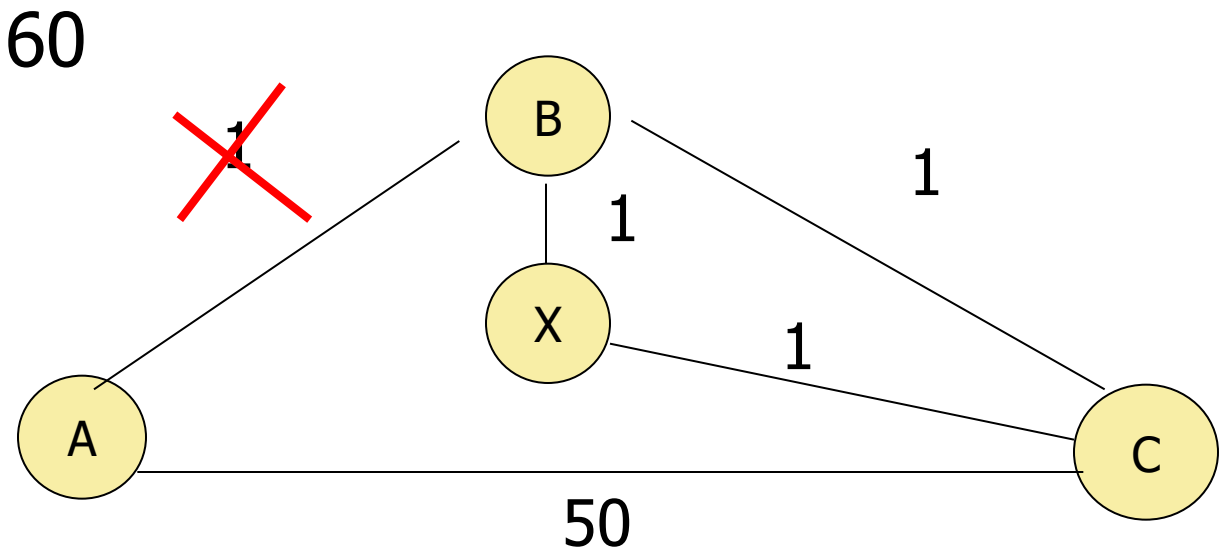
D	C	N
A	50	A
B	1	B

time

Link cost changes here;B updates D(B, A) = 60 as C has advertised D(C, A) = ∞ instead of D(C,A)=5.
At node C, (A,5,B) is replaced by (A,60,B). Node C switches to use (A,50,A).

Algorithm terminates

Does Poison Reverse Always Work?



Node B

D	C	N
A	4	A
C	1	B

Node C

D	C	N
A	5	B
B	1	B

Example: Routing Information Protocol

- Earliest IP routing protocol (1982 BSD)
  - Version 1: RFC 1058
  - Version 2: RFC 2453
- Features
  - Edges have unit cost
  - "Infinity" = 16
- Sending Updates
  - Router listens for updates on UDP port 520
  - Message can contain up to 25 table entries

Agenda

- IPv6 ✓
- Discovery ✓
  - DNS ✓
  - ARP ✓
  - DHCP ✓
- Intradomain Routing ✓
  - Distance Vector ✓
  - Link State NEXT

Link-State Routing

Key idea: distribute a network map

- Each node performs shortest path (SPF) computation between itself and all other nodes
- Initialization step
  - Add costs of immediate neighbors, D(v), else infinite
  - Flood costs c(u,v) to neighbors, N
- For some D(w) that is not in N
  - D(v) = min(c(u,w) + D(w), D(v))

Dijsktra's Algorithm

```
1  Initialization:
2    S = {A};
3    for all nodes v
4      if v adjacent to A
5        then D(v) = c(A,v);
6        else D(v) =     ;
7
8  Loop
9      find w not in S such that D(w) is a minimum;
10     add w to S;
11     update D(v) for all v adjacent to w and not in S:
12       D(v) = min( D(v), D(w) + c(w,v) );
           // new cost to v is either old cost to v or known
           // shortest path cost to w plus cost from w to v
13 until all nodes in S;
```

Notations

- c(i,j): link cost from node *i* to *j*; cost infinite if not direct neighbors
- D(v): current value of cost of path from source to destination *v*
- p(v): predecessor node along path from source to *v*, that is next to *v*
- S: set of nodes whose least cost path definitively known