Intra-Domain Routing

Slides adapted from Daehyoek Kim

Based on Chapter 3.4 of the book

Logistics

- Assignment 1 has been graded. All of you did well!
- Assignment 2 will be posted soon
- We are trying a new group creation system for assignment 2
- This time, we allowed arbitrarily many individual groups:
    - Let us do a class poll on everyone's preferences

Recap

- Ethernet layer (layer 2) switches operate by:
  - If I do not know which port the destination is at, I forward on all of them
  - If I hear a packet from source address A on a port X, next time if I get a packet with the destination address A, I will forward the packet only on port X
  - I will never forward a packet to the same port that I heard it from, because that would be stupid
- Problem: If there is a loop, they will forward forever
- Solution: Disable enough links so that the graph becomes a tree (i.e. no loops). This is done by a distributed algorithm

Today

- The spanning tree protocol is designed to support extremely simple forwarding switches. It does not focus on performance *at all*
- We can do much better

Two types of routing

Routing domain: An internetwork where all the routers are under the same administrative control (e.g., a university campus or an ISP)

Intra-domain routing: routing packets within a domain

- E.g., within a UT campus network
- Simpler routing policy under one administrative domain

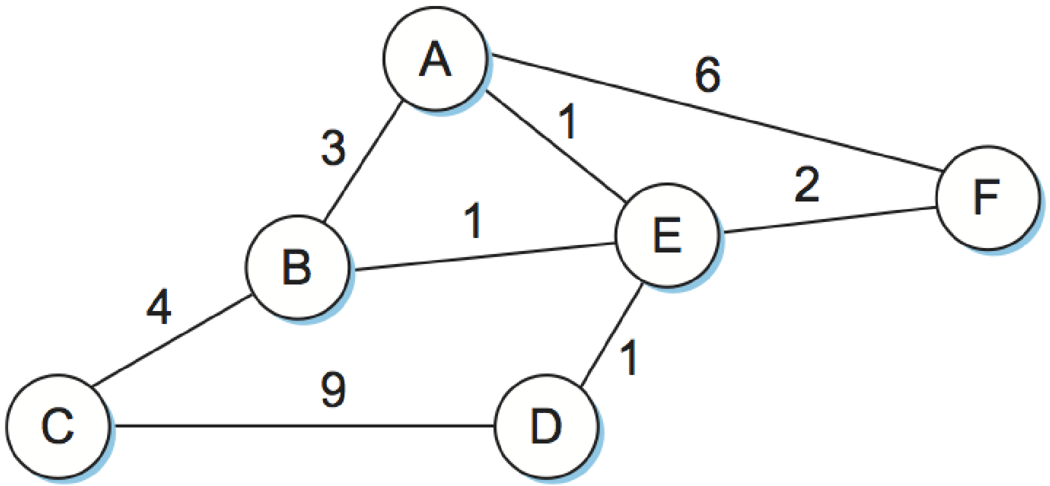
This lecture

Inter-domain routing: routing packets across multiple domains

- E.g., across AT&T and UT campus network
- Involving complex policies between multiple domains

Constructing a routing table

Recall: we can view a network as a weighted graph



Routing is to find the lowest-cost path between any two nodes

- Cost of a path == Sum of the costs of all the edges that make up the path

This objective function is not perfect, but it is not bad either

Why not using static routing?

One might calculate all shortest paths and load them into routers

Problem?

- It does not deal with node or link failures
- It does not consider the addition of new nodes or links
- It implies that edge costs cannot change

Alternatives: Dynamic and/or distributed protocol

- Distance vector
- Link state

Distance vector algorithm

Each node constructs a one-dimensional array (i.e., a vector) containing the “distances” (costs) to all other nodes

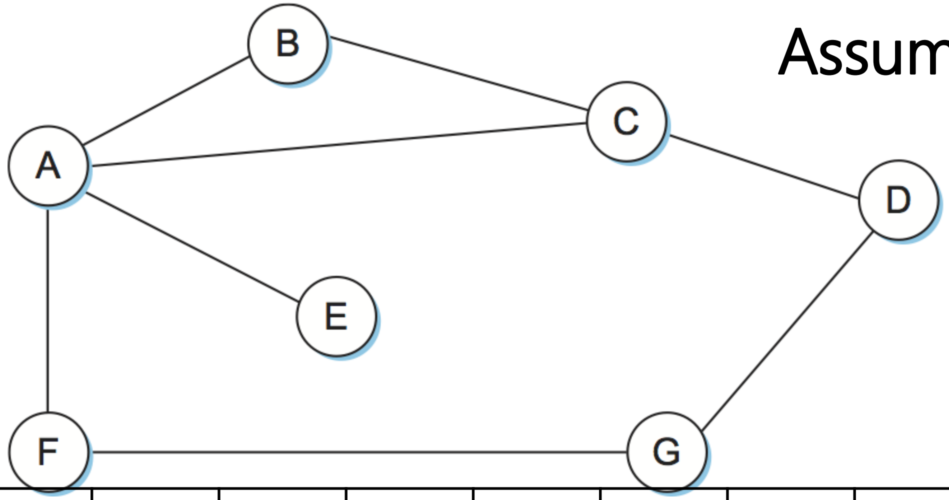
Initially, each node knows the cost of the link to each of its directly connected neighbors

Then, it distributes that vector to its immediate neighbors

It computes shortest paths using Bellman-Ford algorithm

Distance vector algorithm: Initial vectors

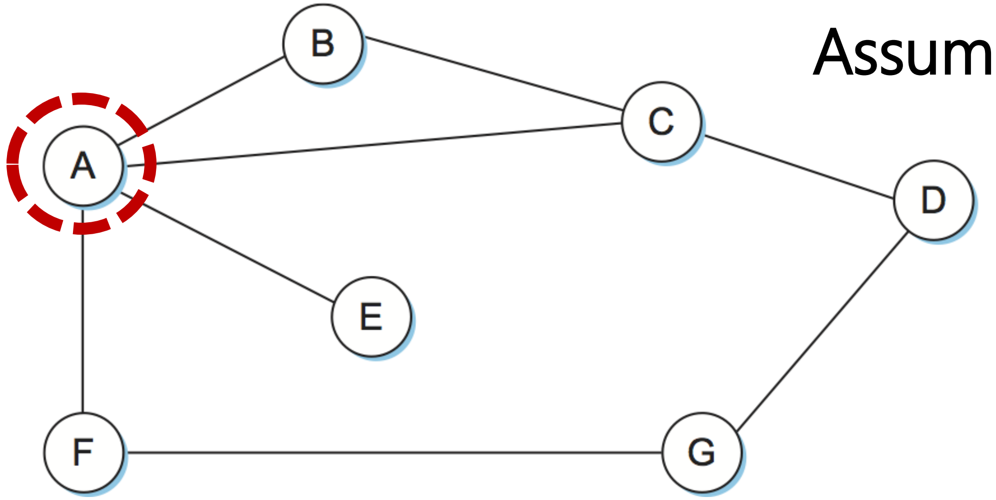
Assumption: Distance between each router is 1



		A	B	C	D	E	F	G
Routers	A	0	1	1	∞	1	1	∞
	B	1	0	1	∞	∞	∞	∞
	C	1	1	0	1	∞	∞	∞
	D	∞	∞	1	0	∞	∞	1
	E	1	∞	∞	∞	0	∞	∞
	F	1	∞	∞	∞	∞	0	1
	G	∞	∞	∞	1	∞	1	0

Initial vector stored on router A

Distance vector algorithm: Initial routing tables

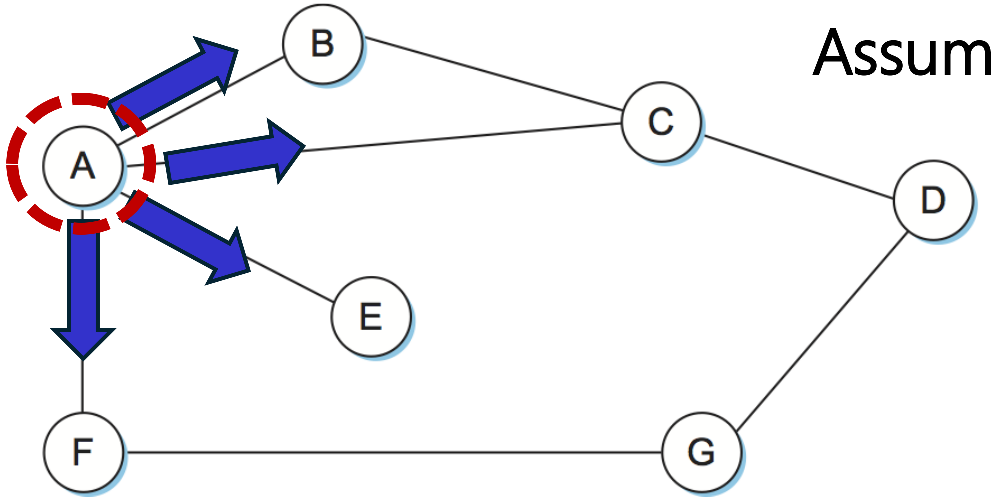


Assumption: Distance between each router is 1

Dest	Cost	NextHop
B	1	B
C	1	C
D	∞	—
E	1	E
F	1	F
G	∞	—

Initial routing table stored on router A

Distance vector algorithm: Each iteration



Assumption: Distance between each router is 1

Every T seconds each router sends its table to its neighbor each each router then updates its table based on the new information

Upon receiving an update, calculate **$D_x(y)$:** cost of least-cost path from x to y:

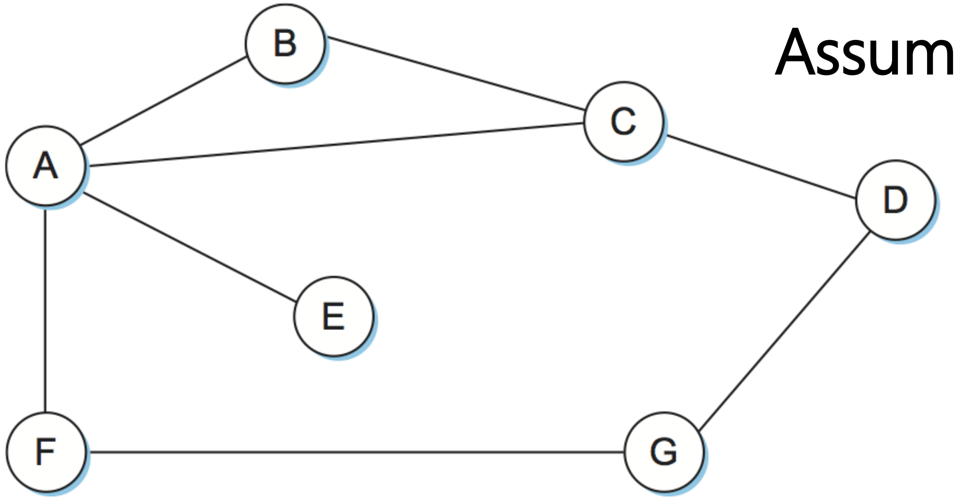
$$D_x(y) = \min_v \{ c_{x,v} + D_v(y) \}$$

min taken over all neighbors v of x

direct cost of link from x to v

v's estimated least-cost-path cost to y

Distance vector algorithm: Final routing tables

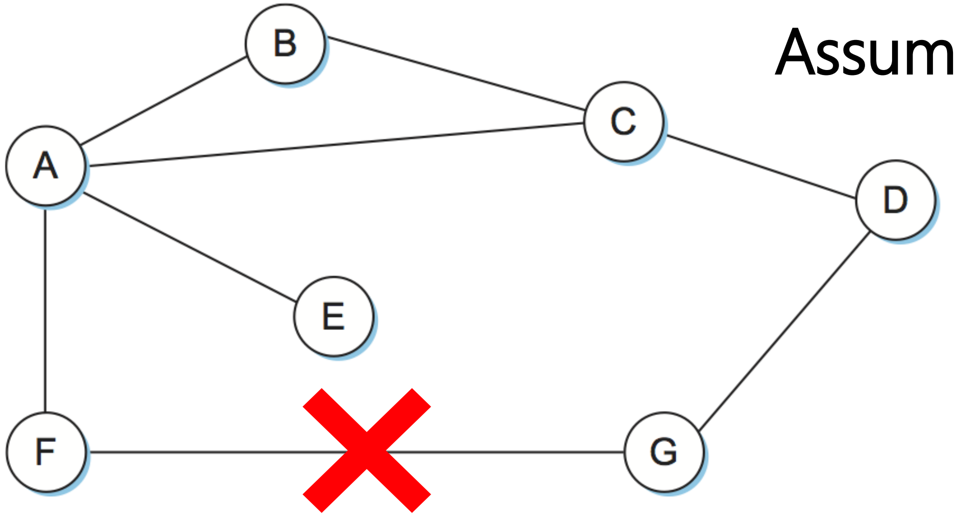


Assumption: Distance between each router is 1

Dest	Cost	NextHop
B	1	B
C	1	C
D	∞	—
E	1	E
F	1	F
G	∞	—

Dest	Cost	NextHop
B	1	B
C	1	C
D	2	C
E	1	E
F	1	F
G	2	F

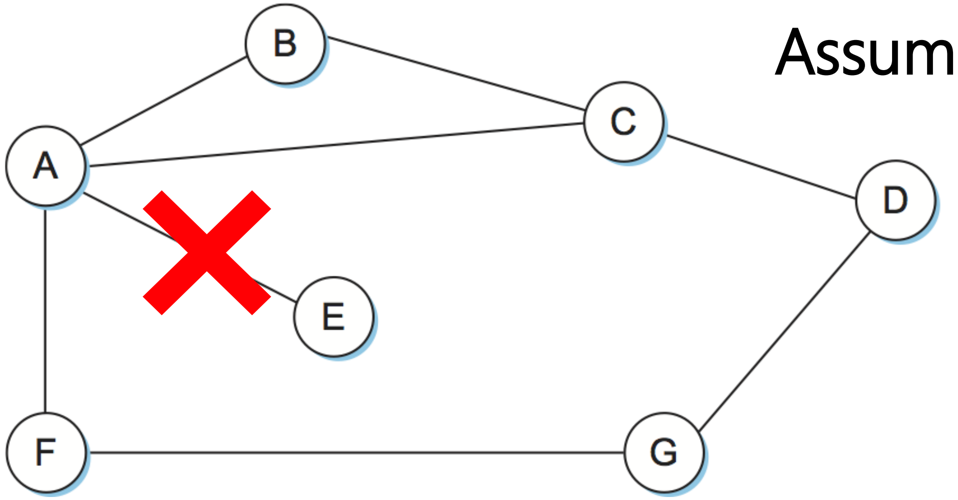
Handling link failure



Assumption: Distance between each router is 1

1. F detects that link to G has failed
2. F sets distance to G to infinity and sends update to A
3. A sets distance to G to infinity since it uses F to reach G
4. A receives periodic update from C with 2-hop path to G
5. A sets distance to G to 3 and sends update to F
6. F decides it can reach G in 4 hops via A

Count-to-infinity problem



Assumption: Distance between each router is 1

1. A advertises a distance of infinity to E
2. At the same time, B and C advertise a distance of 2 to E
3. B, upon hearing that E can be reached in 2 hops from C, concludes that it can reach E in 3 hops and advertises this to A
4. A concludes that it can reach E in 4 hops and advertises this to C
5. C concludes that it can reach E in 5 hops; and so on.

This cycle stops only when the distances reach some number that is large enough to be considered infinite → "Count-to-infinity" problem

Count-to-infinity problem: Example

Dest	Cost	Next
A	1	A
E	2	A

Dest	Cost	Next
C	1	C
E	1	E

- **Initial state**

Dest	Cost	Next
A	1	A
E	2	A

Dest	Cost	Next
C	1	C
E	∞	E

- A to E link goes down

Dest	Cost	Next
A	1	A
E	2	A

Dest	Cost	Next
C	1	C
E	3	C

- A receives C's advertisement before A can advertise to C
- A finds shorter route to E via C

Dest	Cost	Next
A	1	A
E	4	A

Dest	Cost	Next
C	1	C
E	3	C

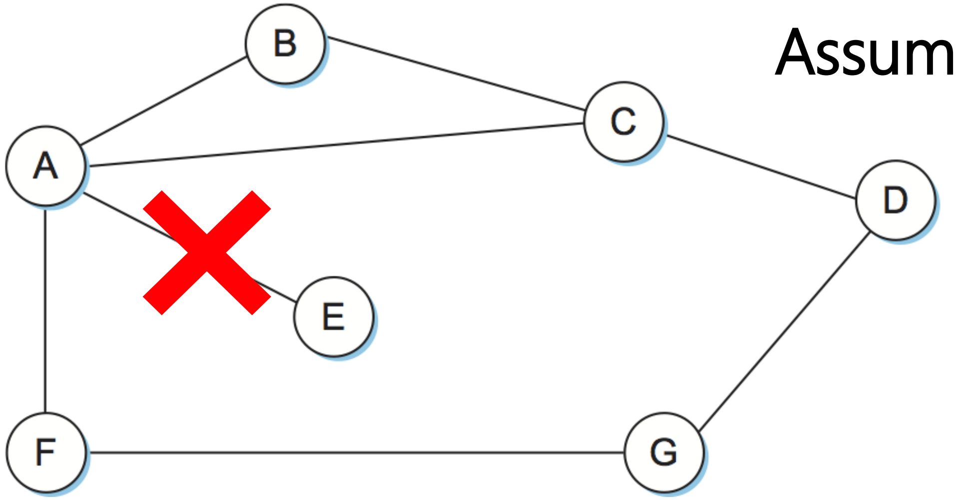
- A advertises updated table to C
- C registers change in path length to E via A

Dest	Cost	Next
A	1	A
E	4	A

Dest	Cost	Next
C	1	C
E	5	C

- C advertises updated table to A
- A registers change in path length to E via C

Solution 1: Using a small approximation of infinity



Assumption: Distance between each router is 1

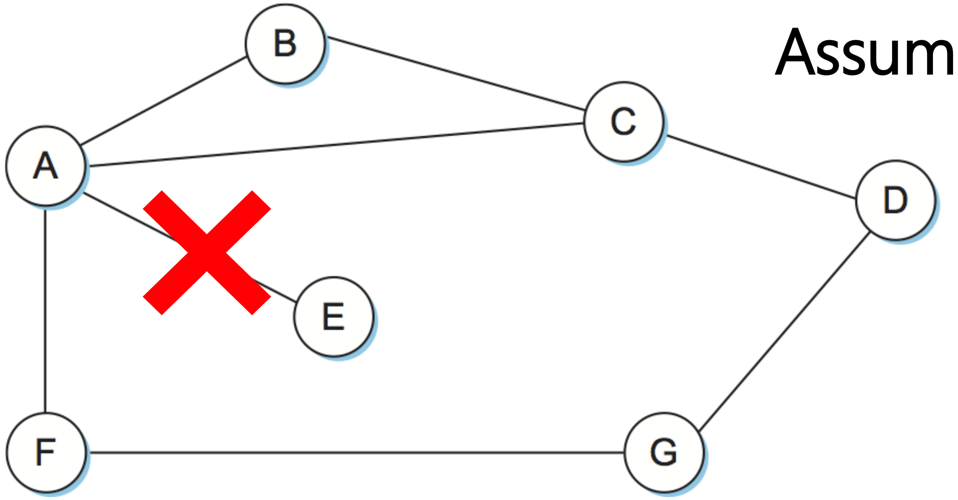
Use some relatively small number as an approximation of infinity

For example, the maximum number of hops to get across a certain network is never going to be more than 16

Problem?

The network can grow to a point where some nodes were separated by more than 16 hops ➔ Limiting the network size

Solution 2: Split horizon

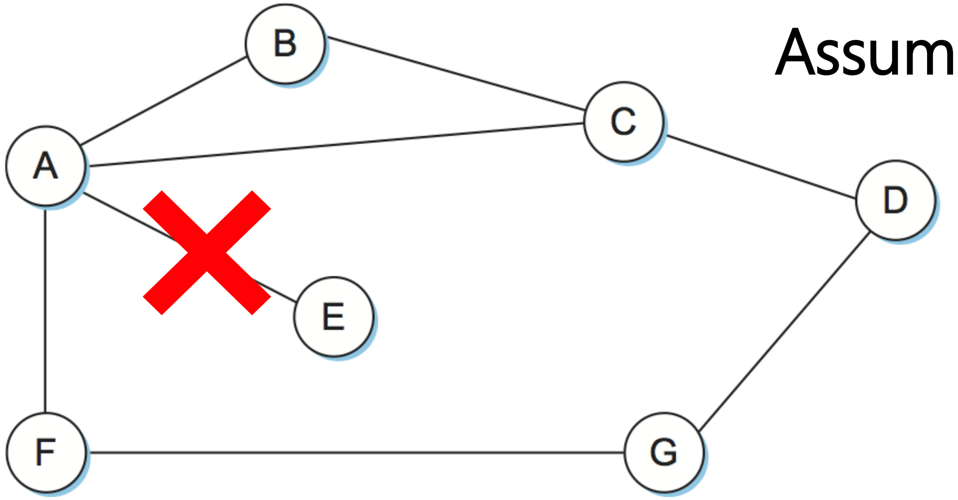


Assumption: Distance between each router is 1

When a node sends a routing update to its neighbors, it does not send those routes it learned from each neighbor back to that neighbor

For example, if B has the route (E, 2, A) in its table, then it knows it must have learned this route from A, and so whenever B sends a routing update to A, it does not include the route (E, 2) in that update

Solution 3: Split horizon with poison reverse



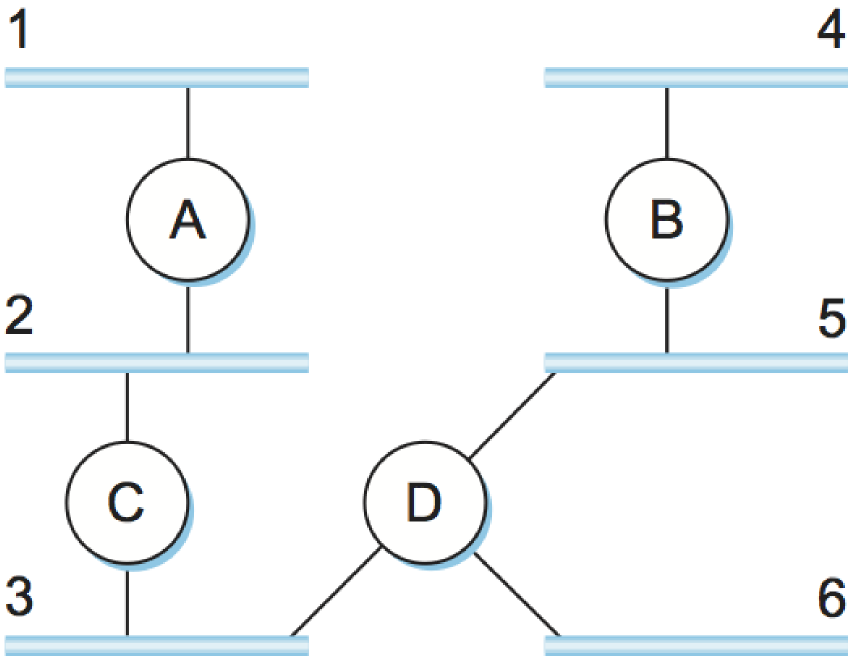
Assumption: Distance between each router is 1

B actually sends that back route to A, but it puts negative information in the route to ensure that A will not eventually use B to get to E

For example, B sends the route (E, ∞) to A https://datatracker.ietf.org/doc/html/rfc1058 - historical IETF Document describing RIP

Routing Information Protocol (RIP)

RIP is designed based on the distance-vector algorithm



Router C would advertise to router A the fact it can reach networks 2 and 3 at a cost of 0, networks 5 and 6 at cost 1, and network 4 at cost 2

0	8	16 31
Command	Version	Must be zero
Family of net 1		Route Tags
Address prefix of net 1		
Mask of net 1		
Distance to net 1		
Family of net 2		Route Tags
Address prefix of net 2		
Mask of net 2		
Distance to net 2		

RIPv2 Packet Format

Summary: Distance-vector routing

Building a routing table by distributing vector of distances to neighbors

Completely distributed and based only on knowledge of immediate neighbors

- Simpler computation, small update message size
- Slow convergence

Count to infinity problem and limited network diameter

- Used in small-sized networks (E.g., LAN or private wide-area networks)

Link state routing

Key idea: Send all nodes (not just neighbors) information about directly connected links (not entire routing table)

- Each node computes the shortest-path using a complete view of the network

Link State Packet (LSP)

- ID of the node that created the LSP
- Cost of link to each directly connected neighbor
- Sequence number (SEQNO)
- Time-to-live (TTL) for this packet

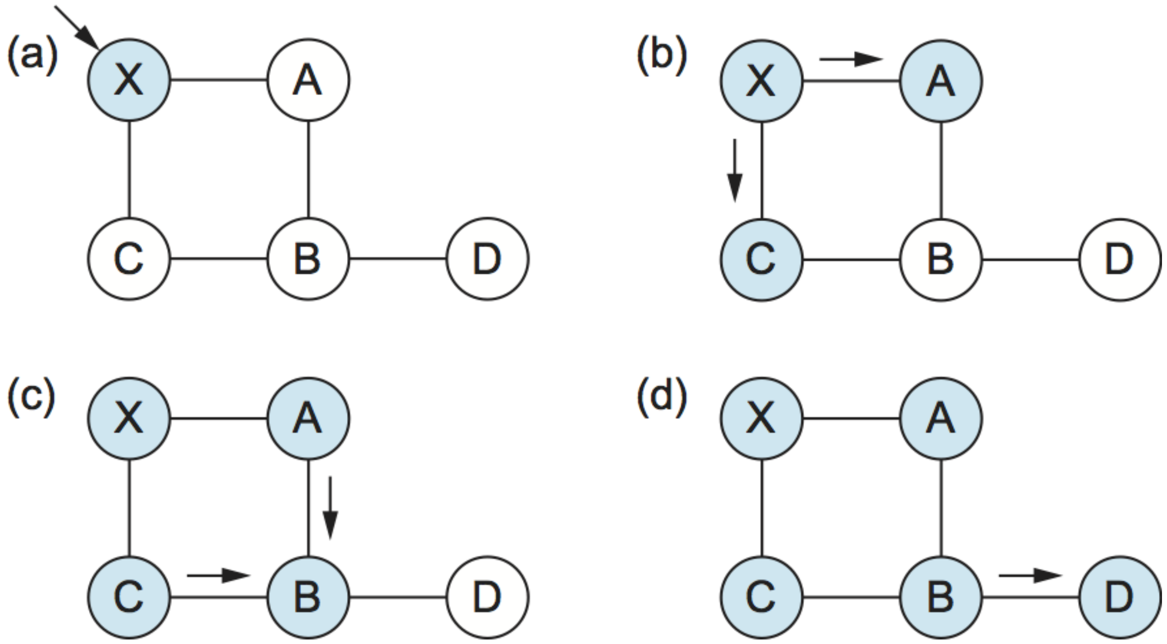
Q: What if a link state packet gets lost?

Reliable flooding

Goal: Ensure every node have most recent LSP from each node

- Forward LSP to all nodes but one that sent it
- Send an acknowledgment back to the sending node
- Retransmit the LSP if an acknowledgment is not received within a certain time frame
- Generate new LSP periodically; increment SEQNO
- Start SEQNO at 0 when reboot
- Decrement TTL of each stored LSP; discard when TTL=0

Reliable flooding: Example



(a) LSP arrives at node X
(b) X floods LSP to A and C
(c) A and C flood LSP to B (but not X)
(d) Flooding is complete

Compute the shorting path using Dijkstra's algorithm

Assumption: non-negative link weights

N: set of nodes in the graph

l((i, j): the non-negative cost associated with the edge between nodes i, j $\in$N and l(i, j) = $\propto$ if no edge connects i and j

Let s $\in$N be the starting node which executes the algorithm to find shortest paths to all other nodes in N

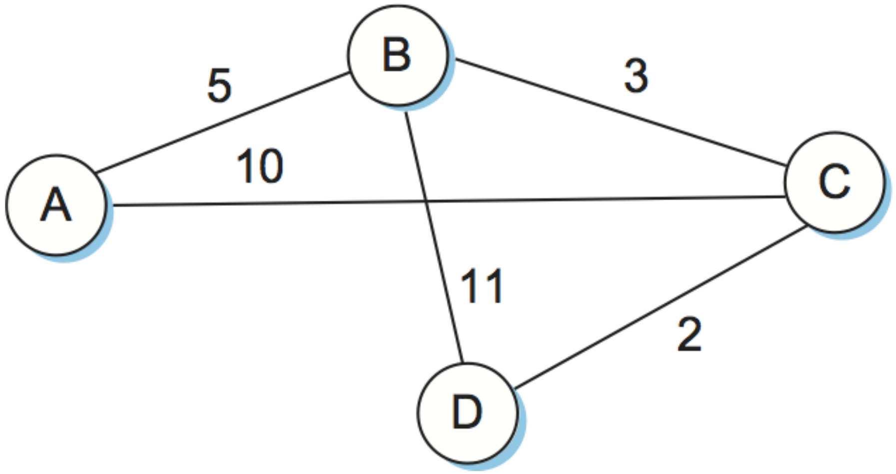
Algorithm maintains two lists: Confirmed and tentative

Each of these lists contains a set of entries (Destination, Cost, NextHop)

Realization of Dijkstra's algorithm: Forward search algorithm

1. Initialize the Confirmed list with an entry for myself; this entry has a cost of 0
2. For the node just added to the Confirmed list in the previous step, call it node Next, select its LSP
3. For each Neighbor of Next, calculate the Cost to reach this Neighbor as Cost(myself→Next) + Cost (Next→Neighbor)
   - If Neighbor is currently on neither the Confirmed nor the Tentative list, then add (Neighbor, Cost, Nexthop) to the Tentative list, where Nexthop is an intermediate node to reach Next
   - If Neighbor is currently on the Tentative list, and the Cost is less than the currently listed cost for the Neighbor, then replace the current entry with (Neighbor, Cost, Nexthop)
4. If the Tentative list is empty, stop. Otherwise, pick the entry from the Tentative list with the lowest cost, move it to the Confirmed list, and return to Step 2

Forward search algorithm: Example



Steps for building a routing table at D

Confirmed	Tentative
(D,0,–)	

Summary: Link state routing

Finding the shortest paths using a complete information about the network

- High computational complexity
- Faster convergence even under dynamic conditions (e.g., link failures)

Preventing routing loops using reliable flooding

- Sequence numbers and retransmissions

Example: OSPF (Open Shortest Path First) protocol

Q: Link state routing is the most common routing technique today? Does this mean we have forever moved past distance vector routing?