

# **CMPE 150: Introduction to Computer Networks**

**Dr. Chane L. Fullmer**  
**chane@cse.ucsc.edu**

# Homework Assignments

Homework assignment #3

Chapter Four

Due by May 22.....

# **CMPE 152: Introduction to Computer Networks**

## **SET 10:**

### ***Routing, Part II***

# Routing Algorithms

- Most books and papers classify routing algorithms into distance-vector and link-state algorithms.
- ***Distance-Vector Algorithm:*** Routers exchange their distances to known destinations; a router uses the distance vectors received from its neighbors to compute its own distances. Computation is distributed.
- ***Link-State Algorithm:*** Routers exchange information about the state of the links in the network; a router uses this information to compute its distances to destinations. Computation is local.

# DBF

- **Information maintained at each router:**
  - **Distance Table:** Distance to each destination reported by each neighbor
  - **Link-Cost Table:** Cost of link to each adjacent node
  - **Routing Table:** Distance and successor (next hop) to each destination
- **Information exchanged among routers:**
  - Vector of one or more entries, each entry stating the distance to a destination
- **Services assumed:**
  - Update messages are exchanged reliably, a node knows who its neighbors are

# Bellman-Ford Algorithm

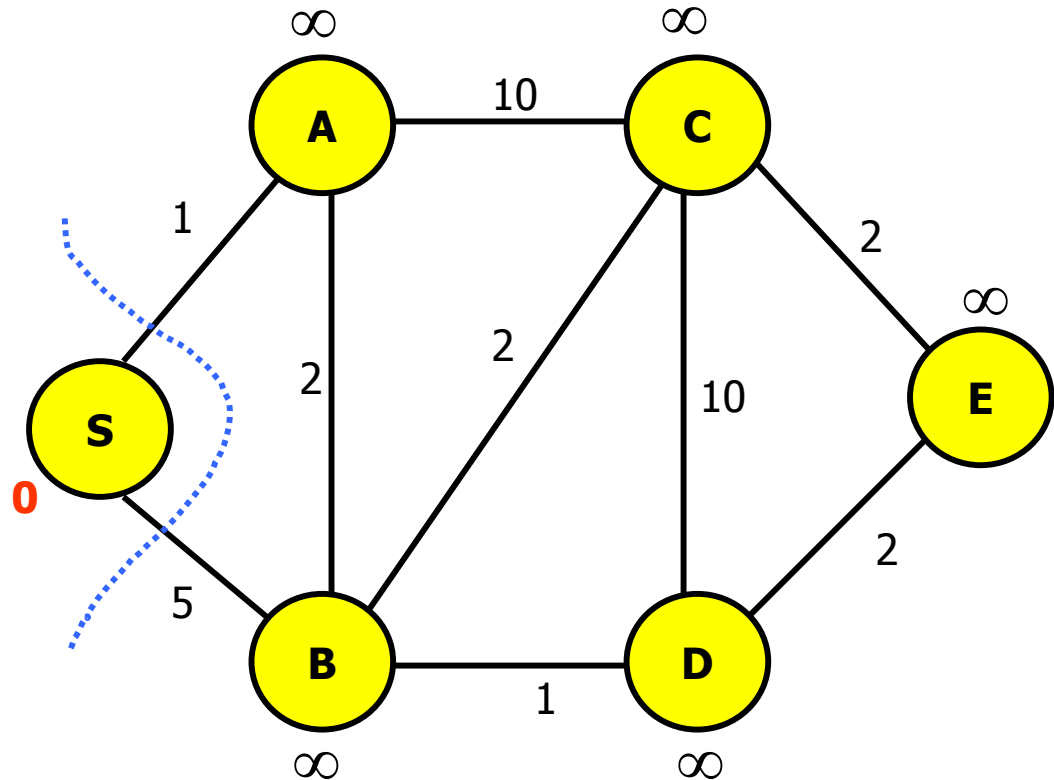
- BF iterates on the number of hops away from a node.

**Step 1:** Initialize source node S with a 0 distance to itself and all other nodes with an infinite distance.

**Step 2:** Set  $H = 1$

**Step 3:** Label all nodes  $H$  hops away from S with the smallest distance from S to the nodes.

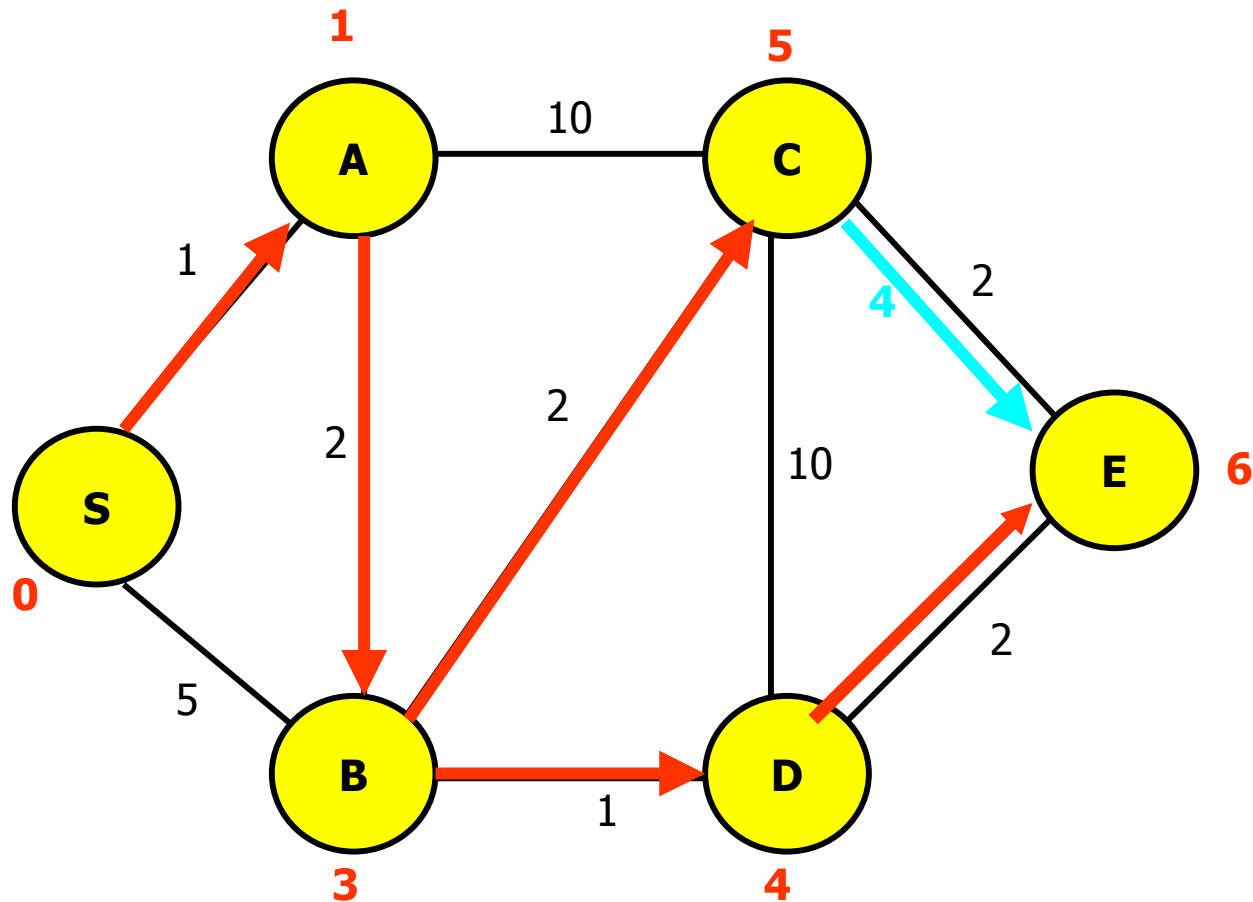
**Step 4:** Stop if all nodes have been covered and no label can be reduced by increasing  $H$ .  
Else, set  $H = H+1$  and repeat Step 3



Link costs are the same in both link directions

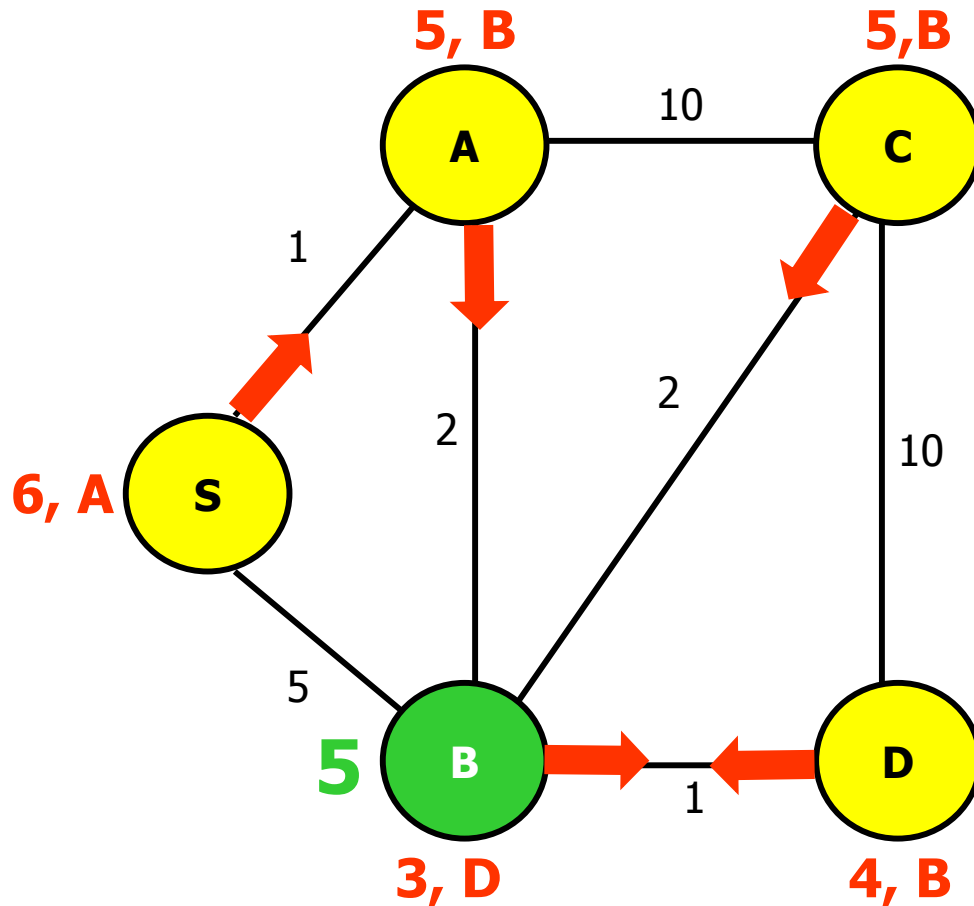
# Bellman-Ford Algorithm

**H = 4:**



**No more nodes can be reached and no label can be reduced**

# Looping in DBF



Erroneous paths persist as long as they appear to be the shortest paths.

Similar looping could occur if the cost of the links to  $j$  increased drastically (e.g., to 20).

DBF cannot be used with link costs that have a large variance!

... etc

# Traditional Link-State Algorithm (LSA)

- Developed as a result of DBF's looping and non-termination problems.
- Two components:
  - Topology map distribution
  - Local shortest path computation
- Each router runs a local shortest-path algorithm (Dijkstra's) using the topology stored locally.
- Flooding is used to replicate the topology map at every router.
- Each router is responsible for reporting the state of outgoing links to the rest of the network.
- Two link-state updates per link reach every router.

# Shortest-Path First (SPF) Algorithm

## Step 1: Initialize

Set  $SPF = \{ \text{root} \}$ , where root is router running SPF  
Distance to root = 0 and distance to other nodes = cost of link or infinity

## Step 2: Find next node for SPF set:

Find a node  $x$  not in SPF set such that:  
distance to/from root =

$\text{Min}\{\text{distance to node outside of SPF set}\}$

Augment SPF set with  $x$

**Stop if SPF set contains all nodes**

## Step 3: Change minimum distance:

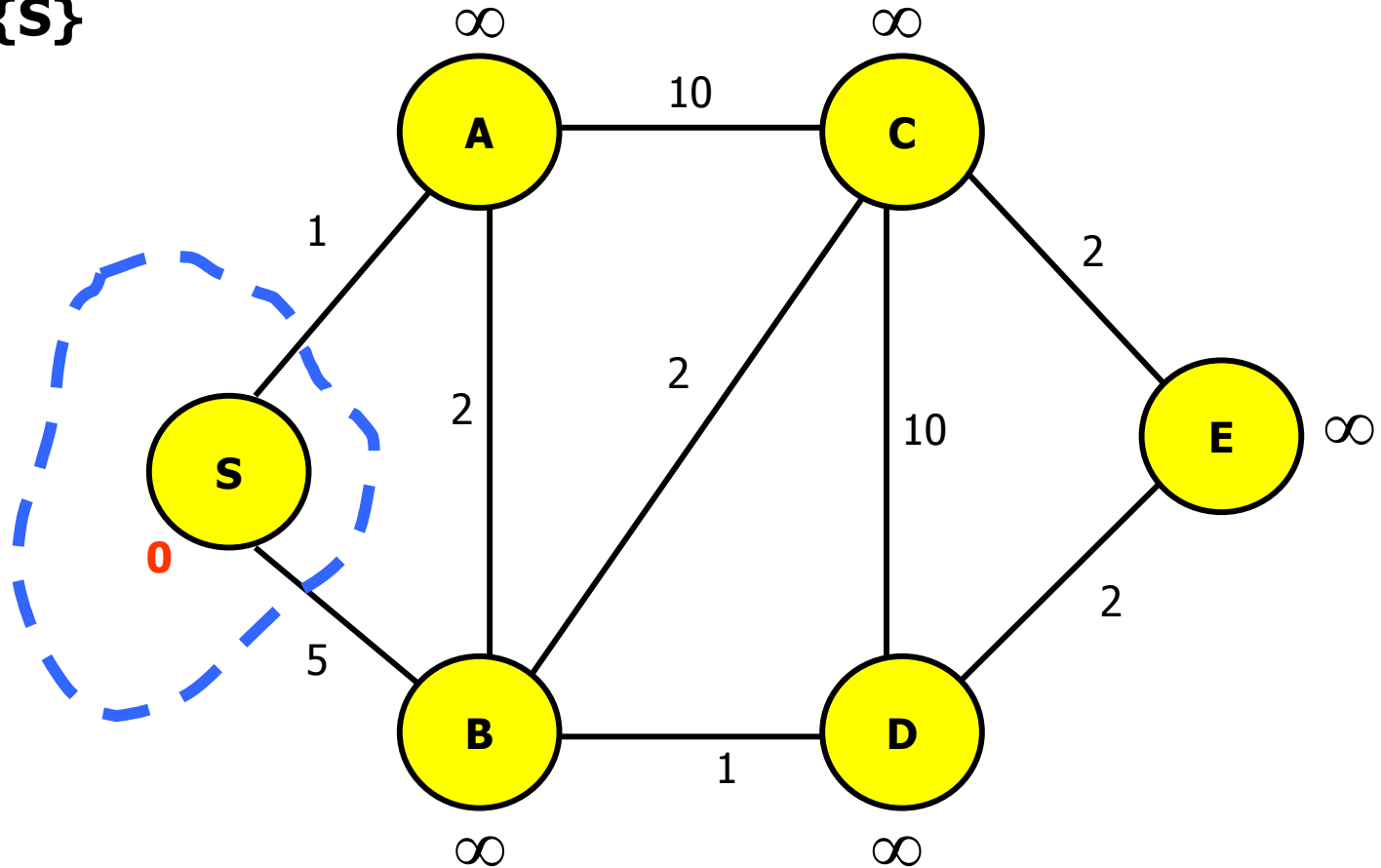
For each node  $y$  outside SPF set do:

$\text{dist. to } y = \text{Min}\{ \text{dist. to } y, \text{dist. to } z \text{ in SPF} + \text{cost of } (z, y) \}$

Repeat Step 2

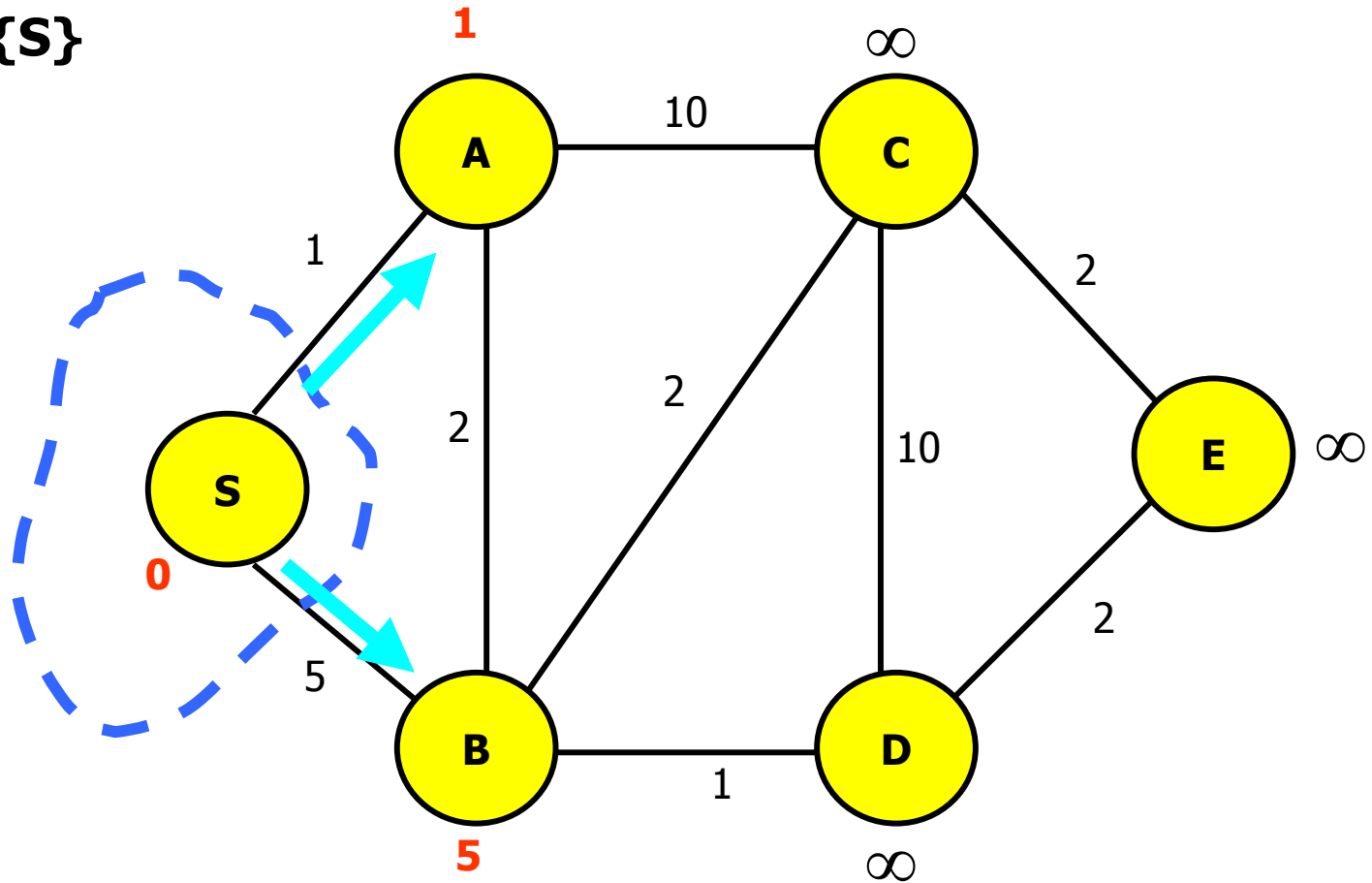
# SPF Example

SPF = {S}



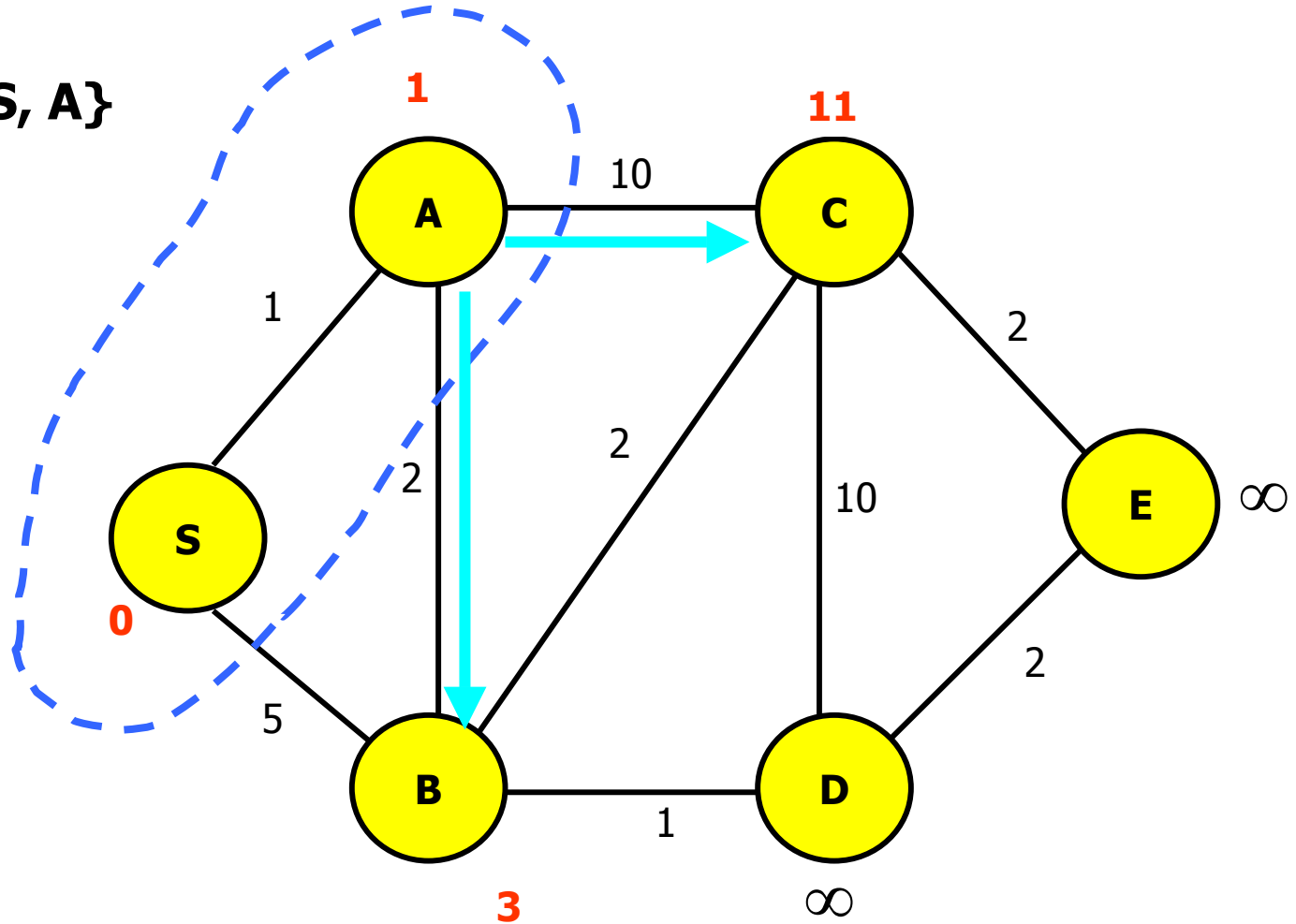
# SPF Example

SPF = {S}



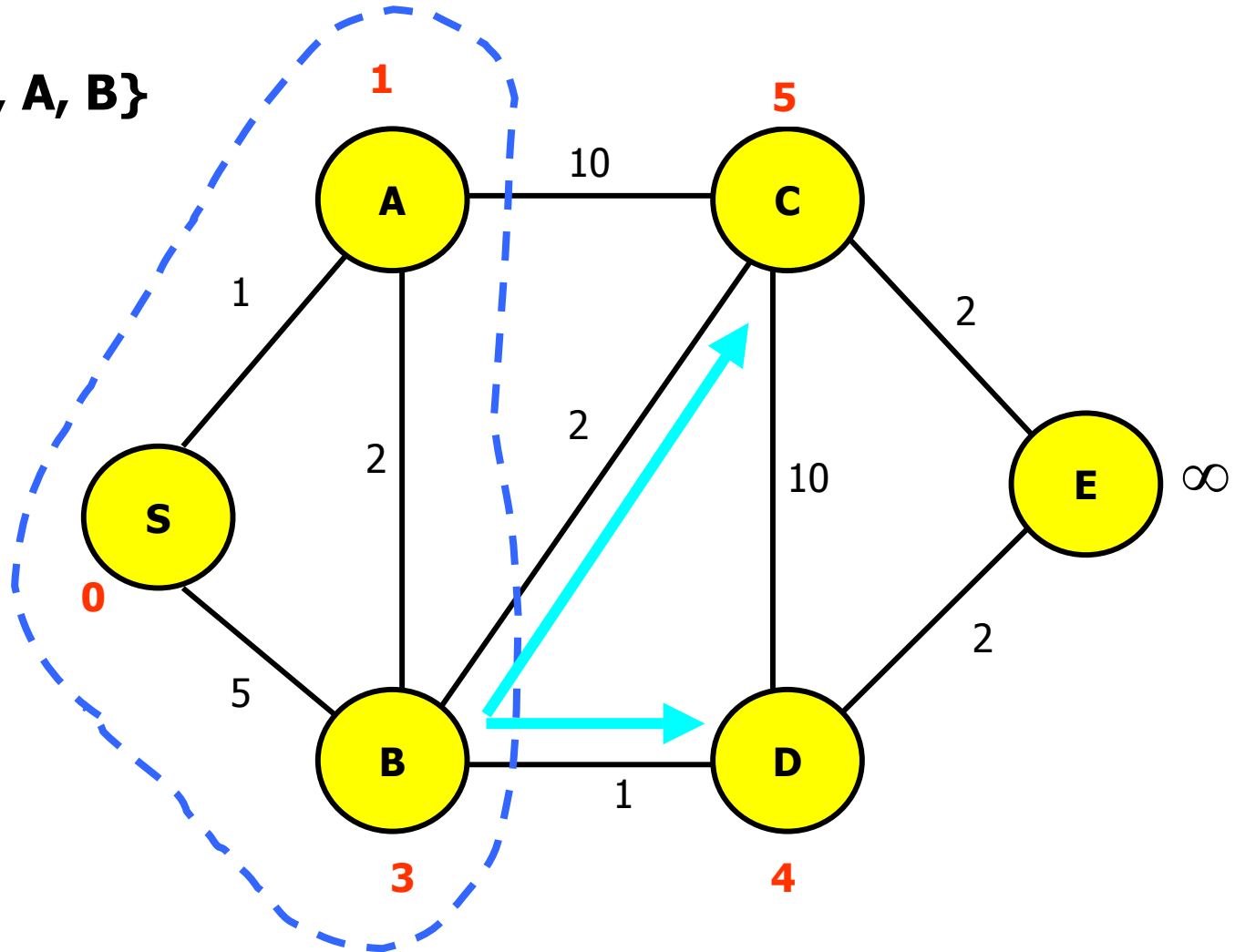
# SPF Example

SPF = {S, A}



# SPF Example

SPF = {S, A, B}



# SPF Example

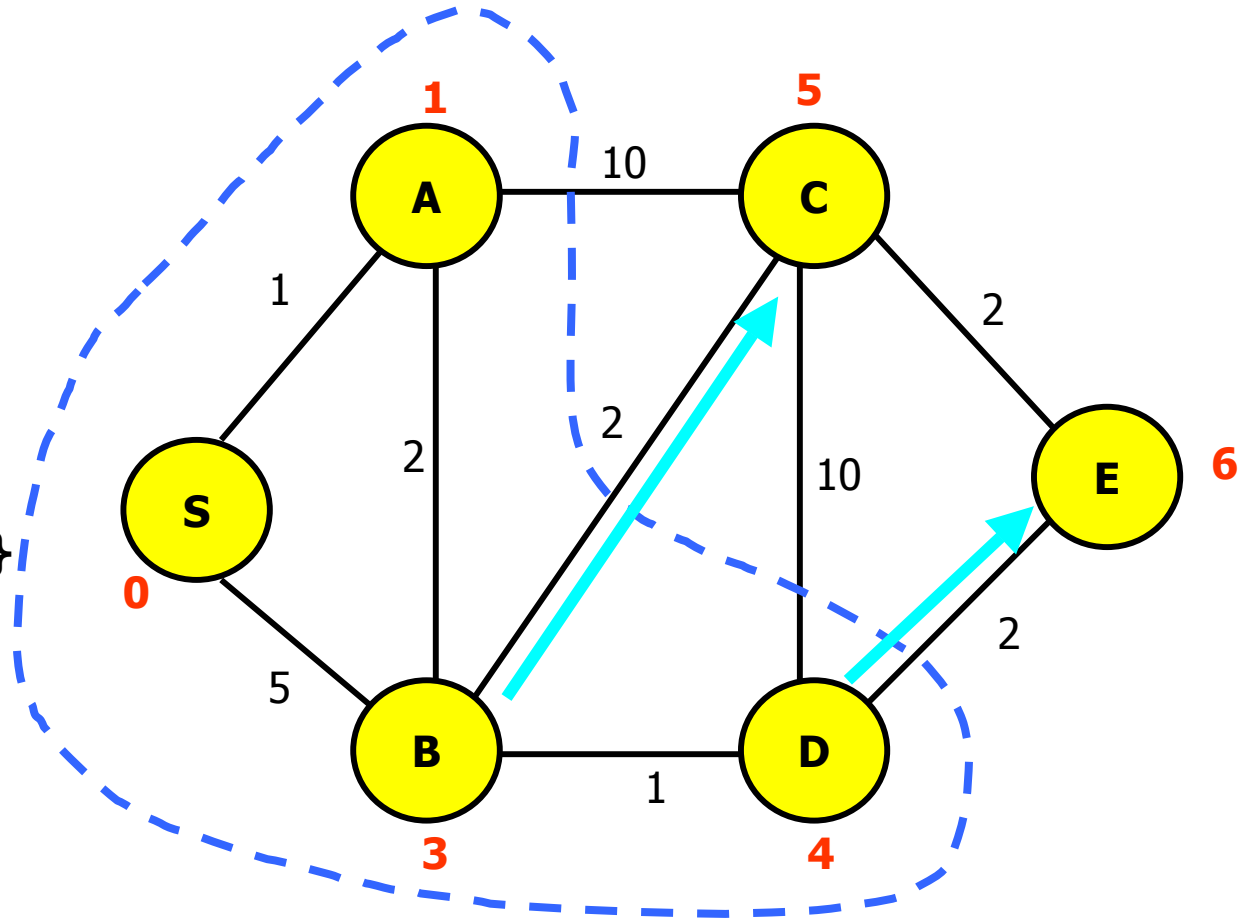
**SPF = {S, A, B, D}**

Labels do not change as we continue to expand SPF set

**SPF = {S, A, B, D, C}**

**SPF = {S, A, B, D, C, E}**

Stop after covering E since all nodes are covered by SPF set.



Note that iteration is on the **next node that can be covered with the next shortest path**; hence **complete topology must be known by router**.

# Flooding of Link States

## Information Stored at Routers:

- Each router maintains all the nodes and all the links in the network in a topology graph.
- Each link in the graph has a cost, a sequence number, and an age.

# Flooding of Link States

## Information Exchanged:

- Each router is responsible for communicating the latest state of each adjacent outgoing link.
- The router sends a link state update (LSU) to report changes on an adjacent outgoing link.
- A sequence number is used to identify the latest LSU.
- An LSU also specifies the age of the LSU, and the age of an LSU is decremented each time it is forwarded and while it is in storage.

**We assume that LSUs are exchanged reliably between any two routers and that a router knows who its neighbors are!**

# Flooding of LSUs

**Flooding Mechanism consists of three rules:**

- **Rule 1: Deleting old LSUs**

- **Discard old LSUs locally:**

- A router discards an LSU in its topology graph when it reaches a maximum age.

- **Refresh own LSUs:**

- A router transmits periodically an LSU for each of its outgoing links, and assigns a 0 age and the highest sequence number to the LSU. (Allows recycling of sequence numbers).

- **Hands off sequence numbers of LSUs:**

- The router originating an LSU is the only one that can change the sequence number of the LSU.

# Flooding of LSUs

## ■ Rule 2: Propagating LSUs

### □ Forward valid LSUs:

A router that receives a more recent LSU with a valid age from a given neighbor propagates it to all its *other* neighbors.

### □ Correct neighbor that reported old data:

A router that receives an outdated LSU from a neighbor *discards* the LSU received and sends its more recent LSU to the neighbor.

### □ Propagate "resets"

A router that receives a *more recent LSU with a zero age* propagates the LSU to all its other neighbors if the link is in its topology graph and deletes the link from its topology graph; else, it ignores the LSU.

# Flooding of LSUs

- **Rule 3: Handling Topology Changes**

- **Make sure that neighbors have the same topology map:**

A router that detects a new neighbor sends its topology graph to that neighbor.

- **Cope with partitions and reboots:**

A router that hears a more recent LSU from a neighbor for one of its outgoing links creates a new LSU with a higher sequence number and sends it to all its neighbors.

# Examples of LSA

- New routing algorithm of ARPANET
- OSPF (open shortest path first)
- IS-IS (intermediate system to intermediate system)
  
- The limitation with LSA is that it incurs substantial communication and processing overhead (to flood link states and compute shortest paths after that).

# Correctness of LSA

- The fact that LSA works correctly is intuitive.
- The correctness of LSA is easily shown by induction on the number of hops away from the origin of an LSU.
- **Process:**
  - Assume that local shortest-path algorithm is correct; assume that neighbors exchange LSUs reliably without deadlocks; assume that topology remains static after an arbitrary sequence of changes.
  - Argue that each LSU propagates independently of others; focus on one LSU and show that it propagates to the farthest node in the network.
  - Show that nodes that fail and come back up, or components that reconnect, eliminate old LSUs by means of Rule 3.

# Termination Detection over Trees

- **Motivation:**

DBF is simple but performs poorly when distances increase, and LSA incurs substantial overhead.

- **Basic idea:**

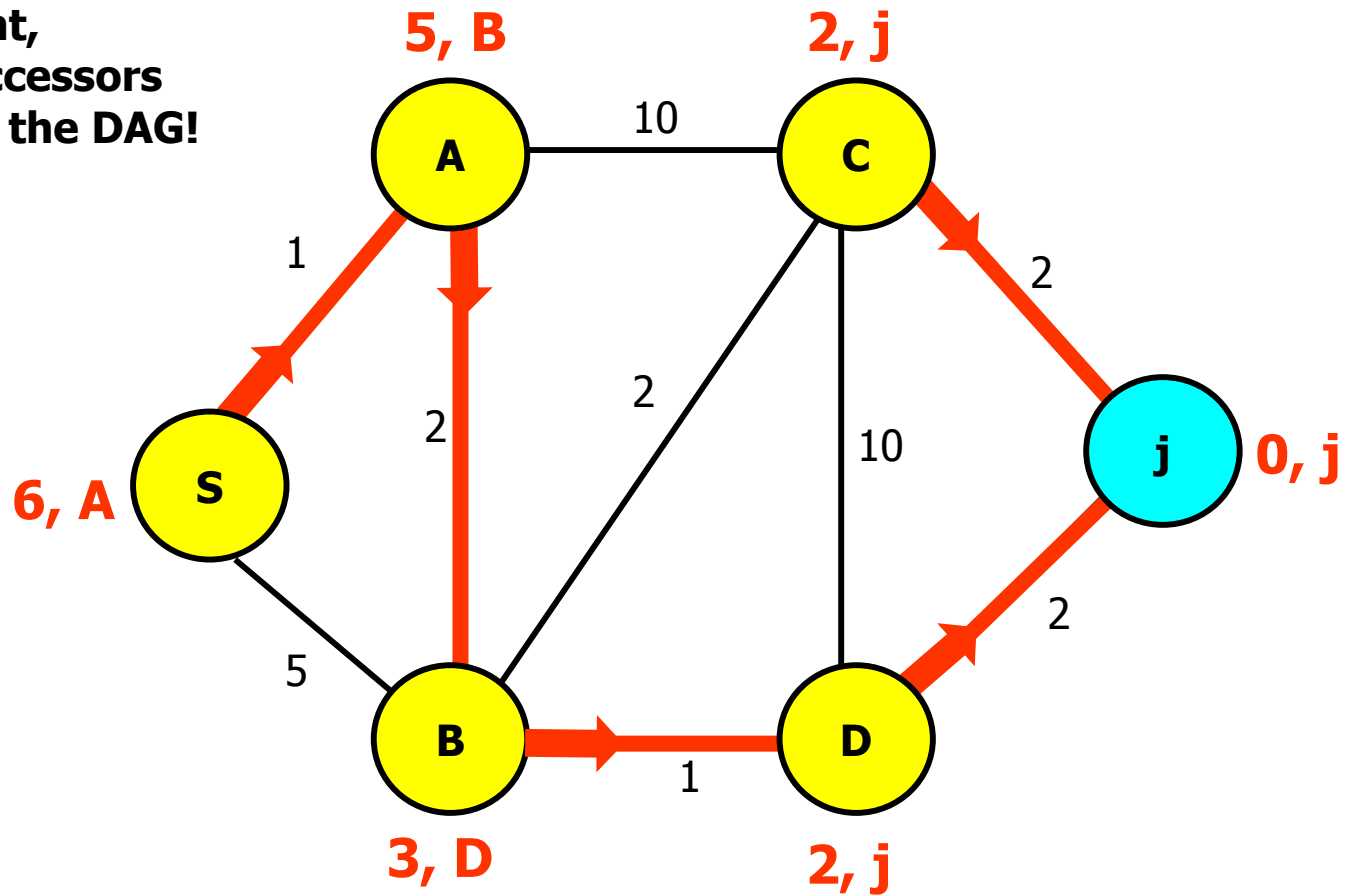
Force a router that must change its successor to a destination to apply a ***dynamic hold-down time*** that lasts as long as it takes for the router to know that its action will not create any loops, and no longer.

- **Mechanism:**

A query-response scheme. A router queries the other routers over a tree defined dynamically and receives feedback over the same tree.

# The “DAG” of a Destination

To ensure loop freedom at every instant, changes in successors must preserve the DAG!



# Diffusing Update Algorithm (DUAL)

- Used in Cisco's EIGRP.
- Paper on DUAL by Albrightson et al.
  - *available on web page (with lecture notes).*
- DUAL provides loop-free paths at every instant in arbitrary topologies, and uses update messages with the same information used in DBF.

# Termination Detection over Cycles

- Consider the DBF algorithm extended as follows:
  - A node reports its distance and path to each destination.
  - The path to a destination is specified in terms of the nodes traversed in the path.
  - A node cannot choose as next hop to a destination any neighbor that has reported the node in its path to the destination.
- This is the basis for BGP's operation.

**That's all  
for today**