

CMPE 150: Introduction to Computer Networks

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

Homework Assignments

Homework assignment #2

Due now.....

Homework Assignments

Homework assignment #3

Chapter Four

Due by May 22.....

CMPE 150: Introduction to Computer Networks

Lecture 9:

Routing, Part I

Network Layer

- The main functions at the network layer are addressing, routing, congestion control, and admission control.
- *Addressing* consists of identifying where a destination is with respect to the network topology.
- *Routing* consists of (a) computing paths from sources to destinations and (b) forwarding packets along such paths.
- *Congestion control* consists of limiting the amount of data a source can send into the network.
- *Admission control* consists of limiting the number of sources allowed to send data into the network, and in a way is part of system-wide congestion control.

Routing

- Algorithms used to compute paths from sources to destinations can be:
 - ***Static or adaptive:*** Routers compute paths off-line or dynamically in response to changes in topology or even traffic.
 - ***Hierarchical or flat (heterarchical):*** Routers and hosts are organized into clusters of nodes or all destinations and routers are treated as peers.
 - ***On-demand or table-driven:*** Routers maintain routing information for only those destinations for which need to forward data, or for all destinations.
- You will also see routing algorithms classified according to the type of information they use.

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.

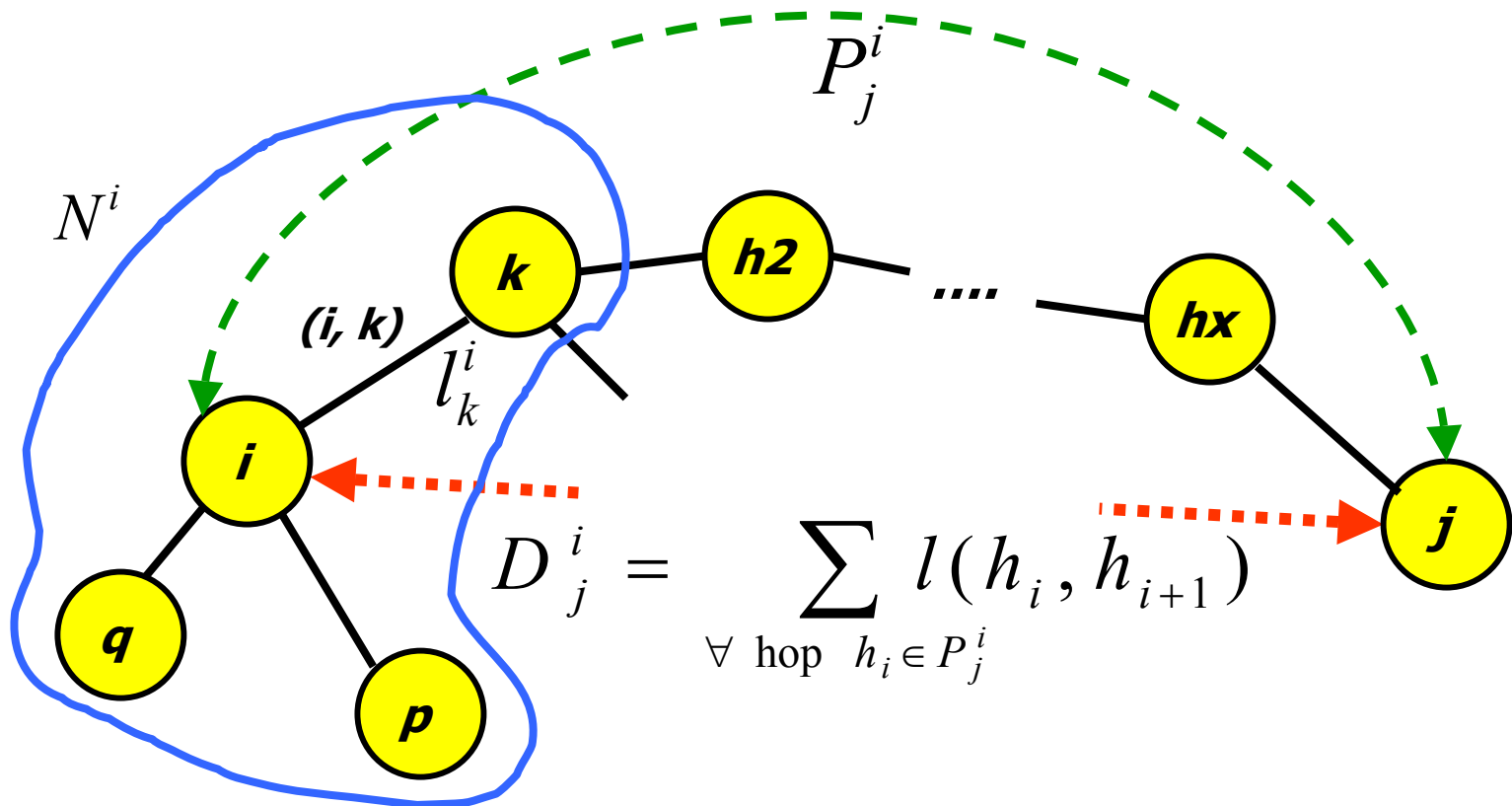
- **This is a very limiting view!**

Routing Algorithms

- A routing algorithm is carrying out a distributed computation (obtaining correct paths from each source to its desired destinations).
- Some indication that at least some aspect of this computation has ended must be available to routers.
- The end of *any* distributed computation can be detected in three known ways (and combinations of such ways):
 - Termination detection over a tree
 - Termination detection using sequence numbers
 - Termination detection over a ring
- Distances or link states can be used in each of these three types of algorithms.

Shortest-Path Routing

- **Problem:** Compute the path of minimum length from each router to each destination
- **Notation:** $G(N, E)$ is the network of $|N|$ nodes and $|E|$ links



Bellman-Ford Algorithm

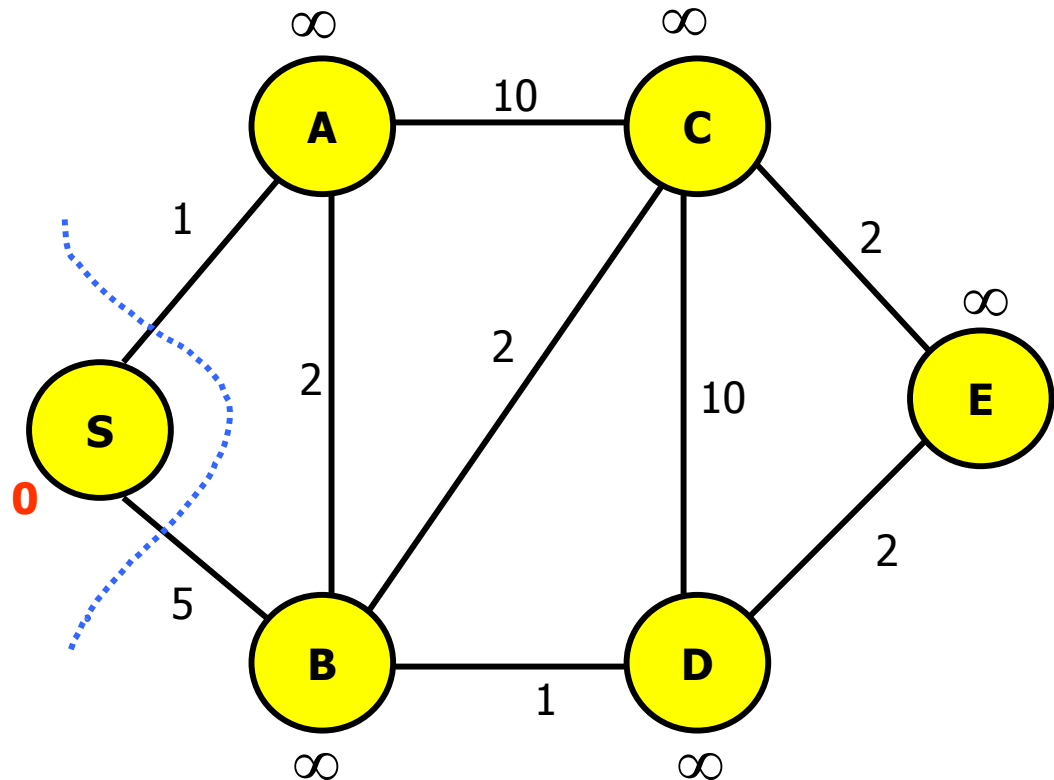
- BF iterates on the number of hops away from a node.

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.

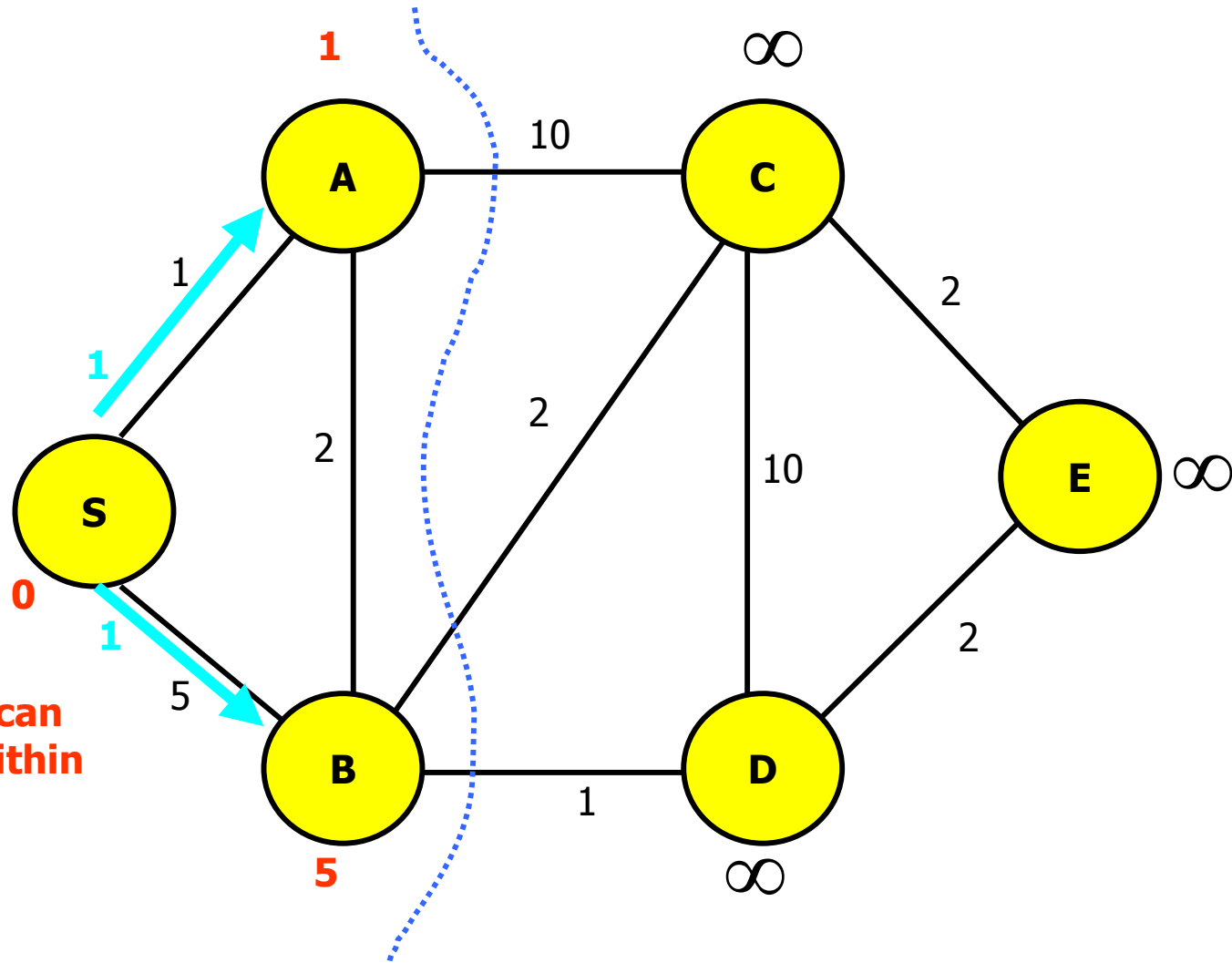
Step 4: Stop if all nodes have been covered and no label can be reduced by increasing H .
Else, set $H = H + 1$ and repeat Step 3



Link costs are the same in both link directions

Bellman-Ford Algorithm

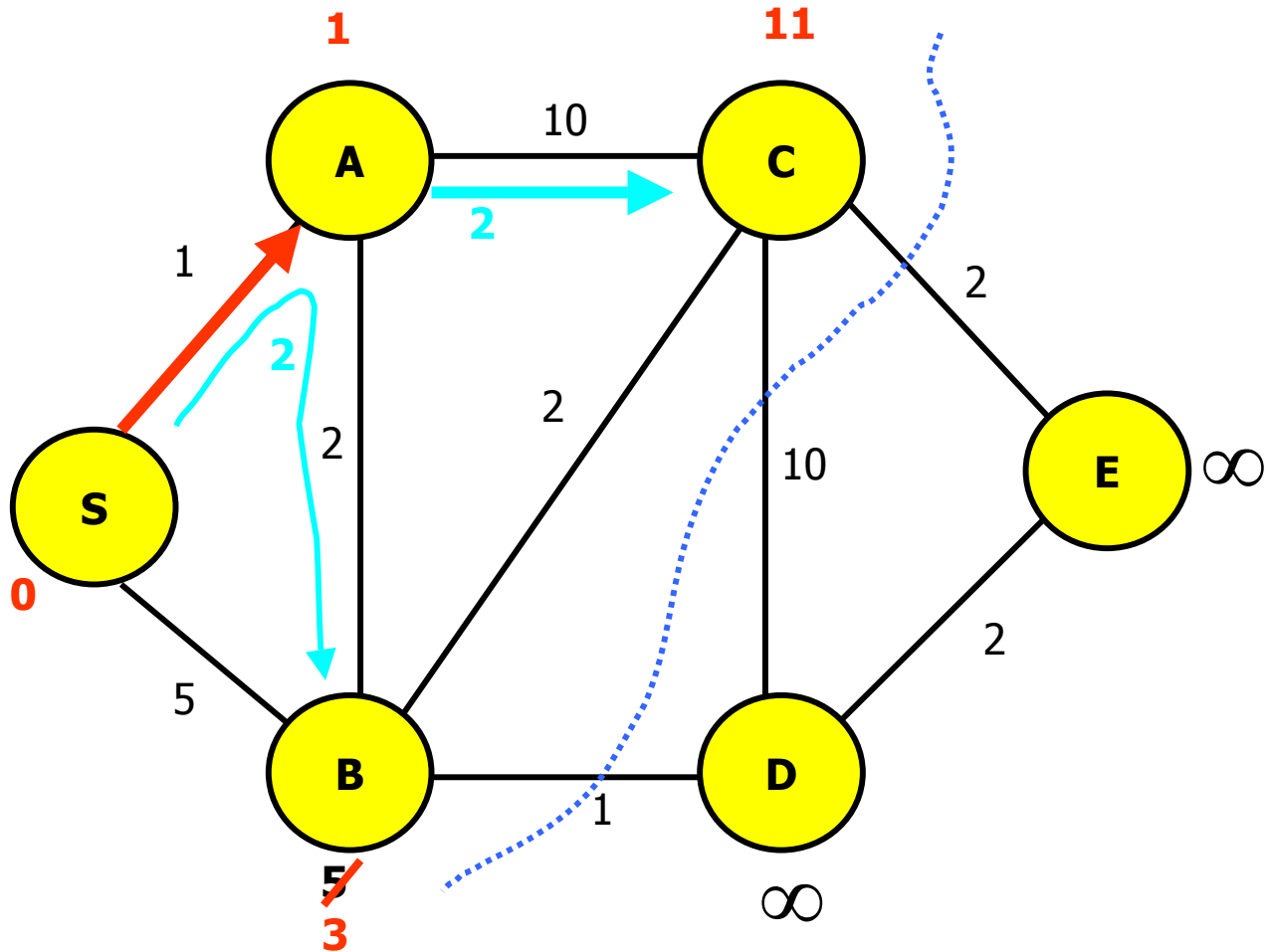
H = 1:



Only A and B can be reached within one hop

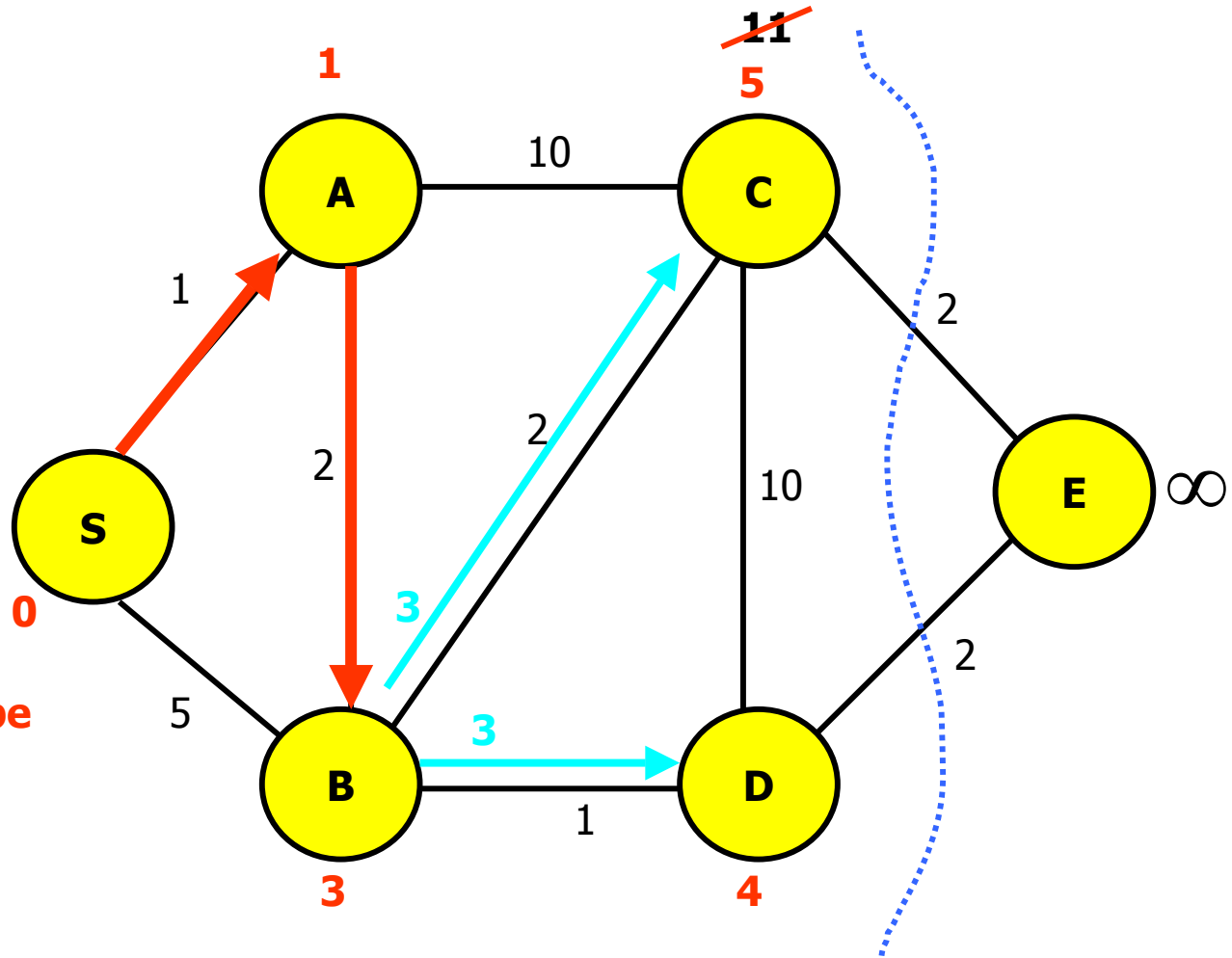
Bellman-Ford Algorithm

H = 2:



Bellman-Ford Algorithm

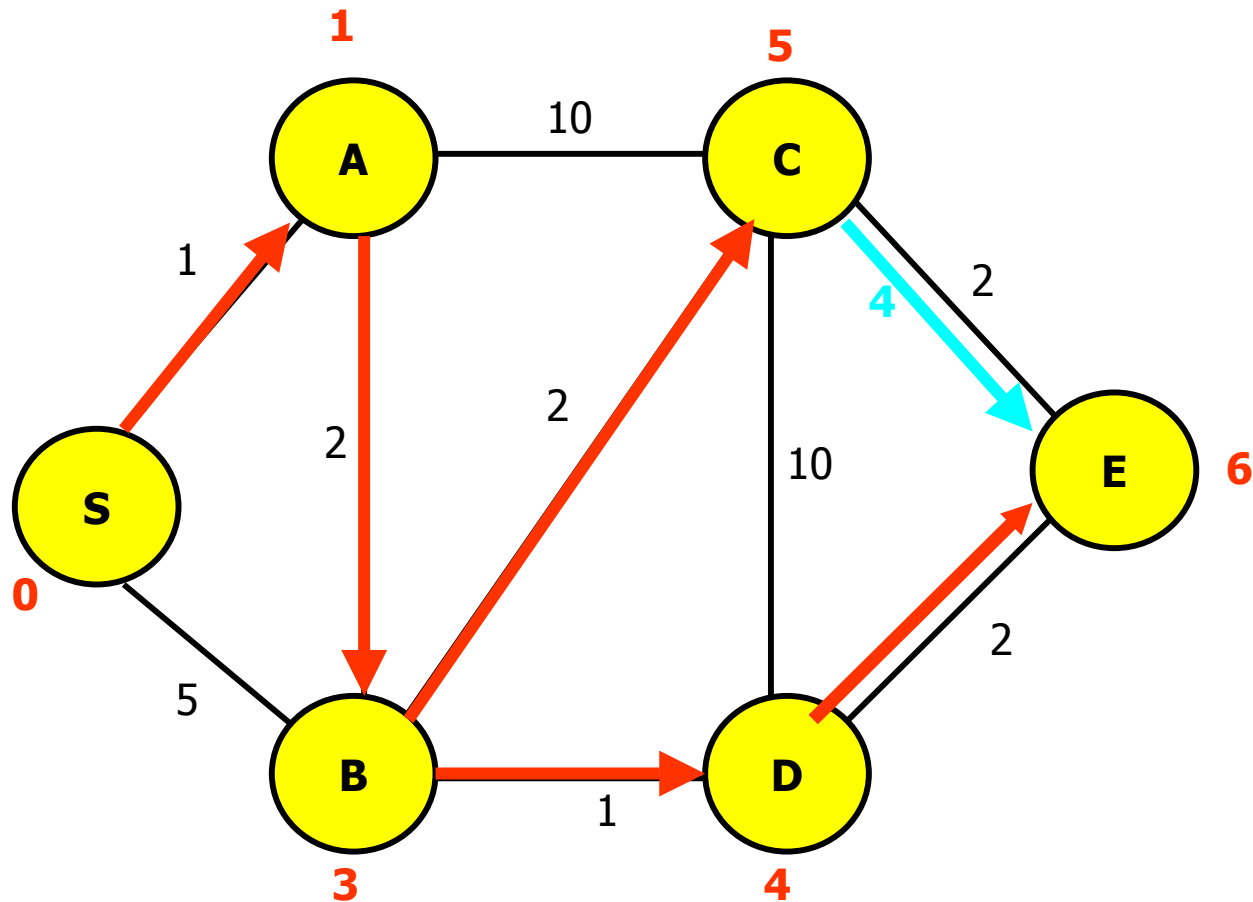
H = 3:



Only E cannot be reached within three hops

Bellman-Ford Algorithm

H = 4:

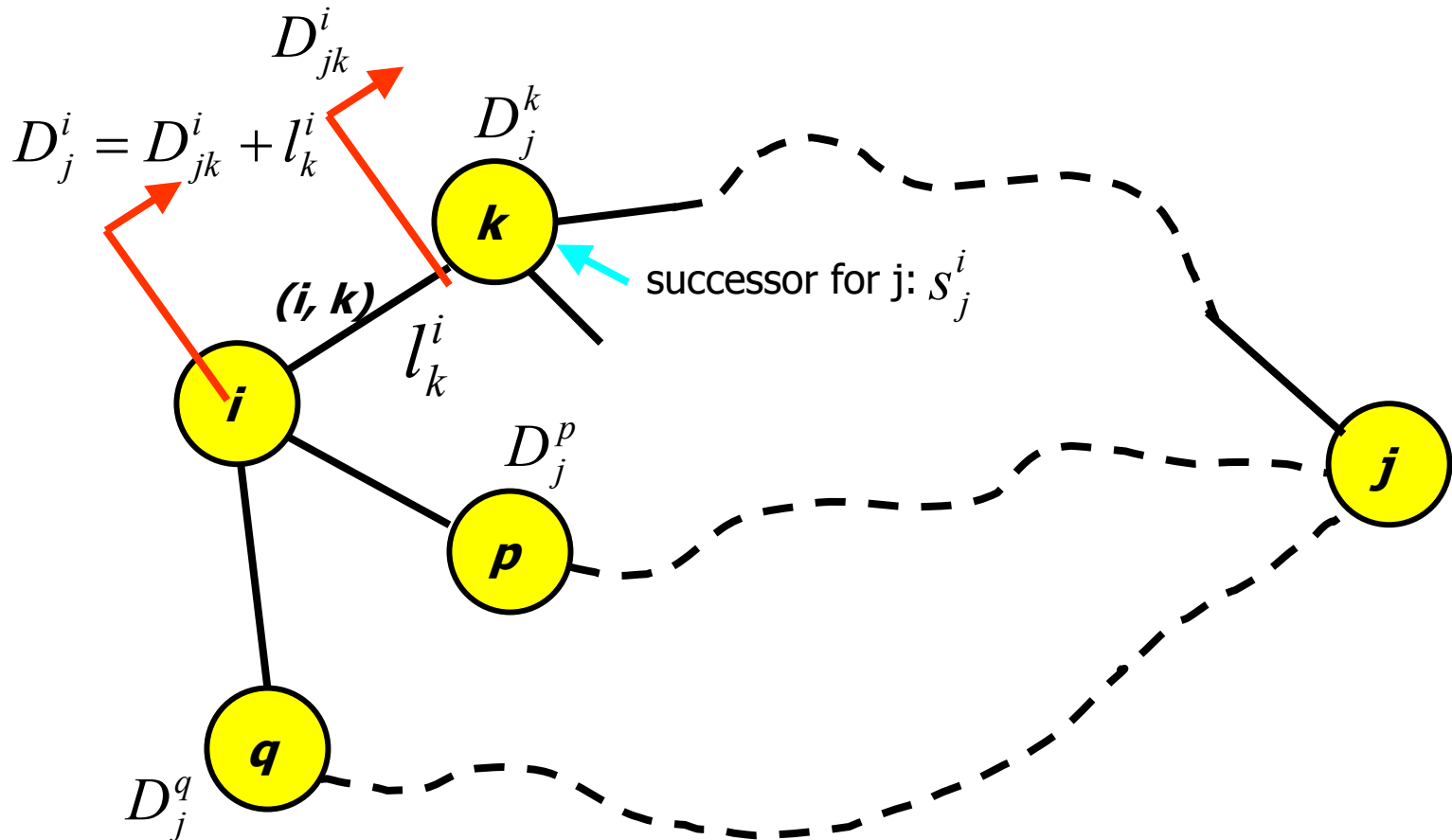


No more nodes can be reached and no label can be reduced

Distributed Bellman-Ford Algorithm (DBF)

- The objective of DBF is to have a distributed implementation of BF, so that routers can compute distances to destinations distributedly.
- To accomplish this, the computation of a distance to a destination starts at the destination itself.
- The iteration of DBF is on the number of hops away from a destination.
- DBF operates independently for each destination.
- Destination starts by stating the distance to itself is 0
- The neighbors of the destination receive this information, process it and send their own updates.
- Distances propagate throughout the network.

Notation

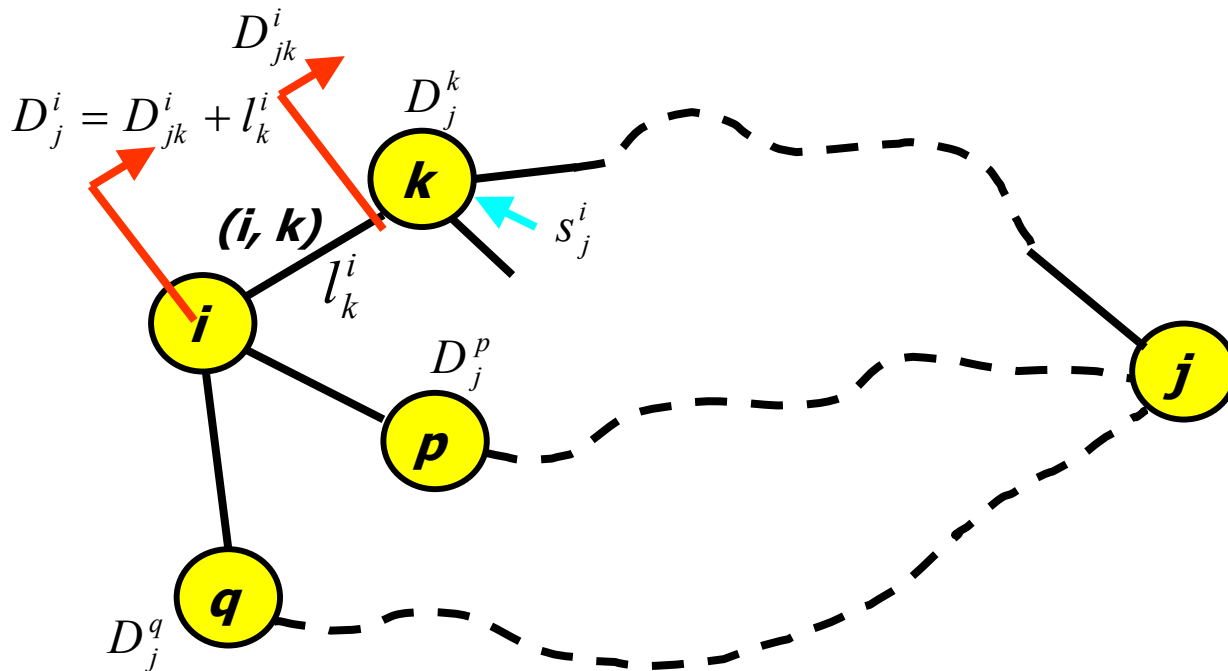


Node i must compute the shortest distance to j , which we show to be through neighbor k

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

DBF



The next hop for j is a neighbor that provides the minimum distance according to the Bellman-Ford equation.

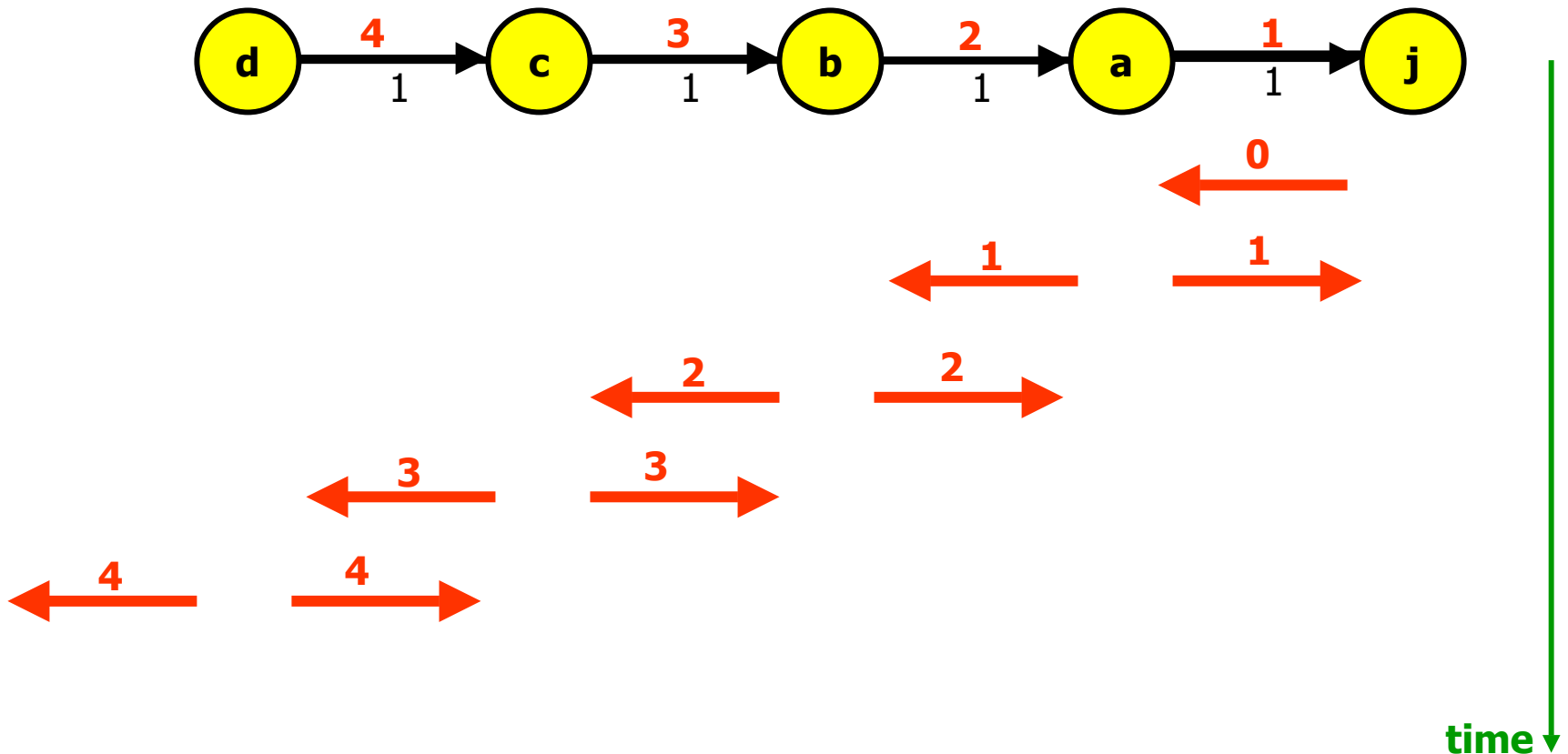
Router i computes D_j^i every time it receives an input event (link change or update message) and sends an update message to its neighbors.

D_j^i is computed with the Bellman-Ford Equation:

$$D_j^i = \text{Min} \left\{ D_{jp}^i + l_p^i \mid p \in N^i \right\}$$

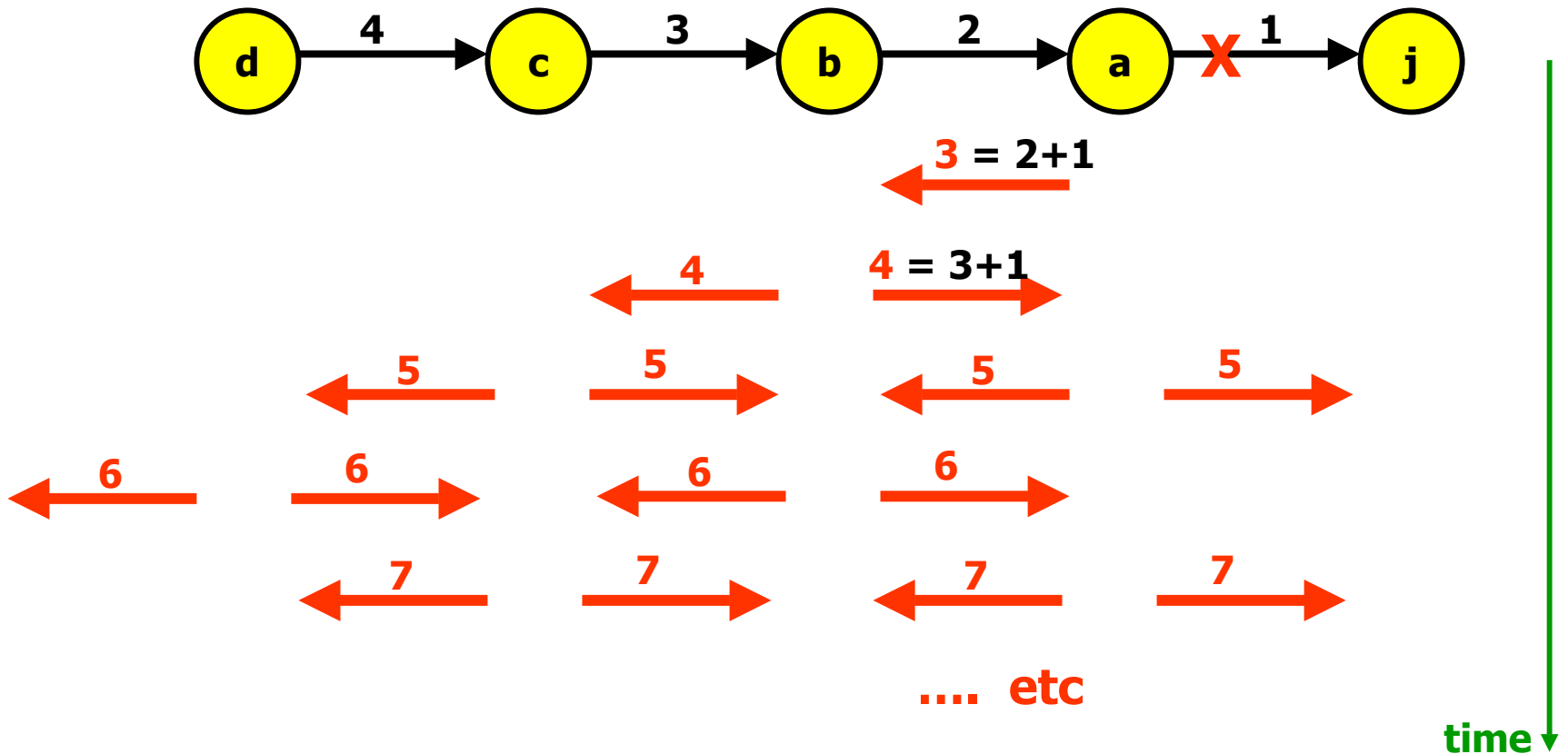
Example of DBF Operation

- For simplicity, we will assume “synchronous operation” in all cases!



Counting to Infinity in DBF

- The problem with DBF is that it does not have a termination detection mechanism!



Correctness of DBF

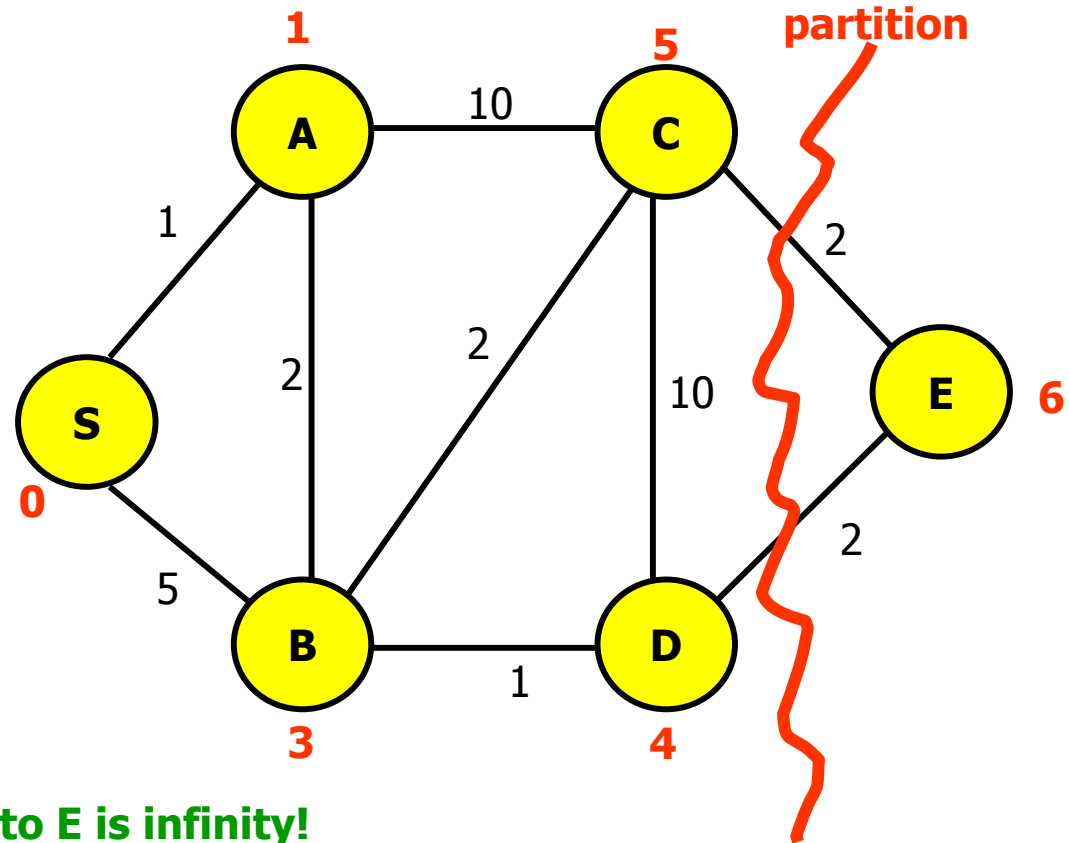
- Note that BF converges because H is finite!

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



After partition, the max H from S to E is infinity!

S can keep max number of nodes in the network (N), and the distance and hop count for each destination.

S can stop when hop count = N , because longest path must be $N-1$ hops.

Ad Hoc Solutions (do not work)

Counting to N takes too long! Alternatives include:

- **Split horizon:** Does not report routes through a successor to the successor itself.
- **Hold-down timer:** After distance to destination increases, send update stating new distance through current successor, wait for a long period of time before computing new successor and shortest distance and then act as in DBF.
- **Poisoned reverse:** After distance increase, report an infinite distance and then correct the distance. (Or in general, reporting infinity to the successor of destinations routed through it).
- **Next-hop information:** Communicate the distance and next hop to each destination (used in RIP v2)

DBF Correctness

■ Liveness:

- Each node updates its routing table independently of others.
- The only possibility of deadlock occurs in the reliable exchange of update messages (the equivalent of a point-to-multipoint ARQ)

■ Safety:

- We must show that, in the absence of deadlocks and after an arbitrary sequence of topology changes, the protocol produces correct routing tables and stops sending updates.

DBF Correctness

■ Assumptions:

- We focus on the min-hop routing case.
- Assume that G is connected (max hop count is finite) and that no more topology changes occur after time t ; proof is by simple induction on the number of hops away from a destination \mathbf{j} .
- Assume that no deadlocks occur in the exchange of correct update messages.

■ Proving that DBF Obtains Correct Routing Tables:

- Because of the underlying service, a neighbor of \mathbf{j} knows that \mathbf{j} is a neighbor within a finite time, and sets its distance to \mathbf{j} equal to 1 by some time $T_0 \geq t$
- Assume that DBF is correct for $h-1$ hops away from \mathbf{j} .

DBF Correctness

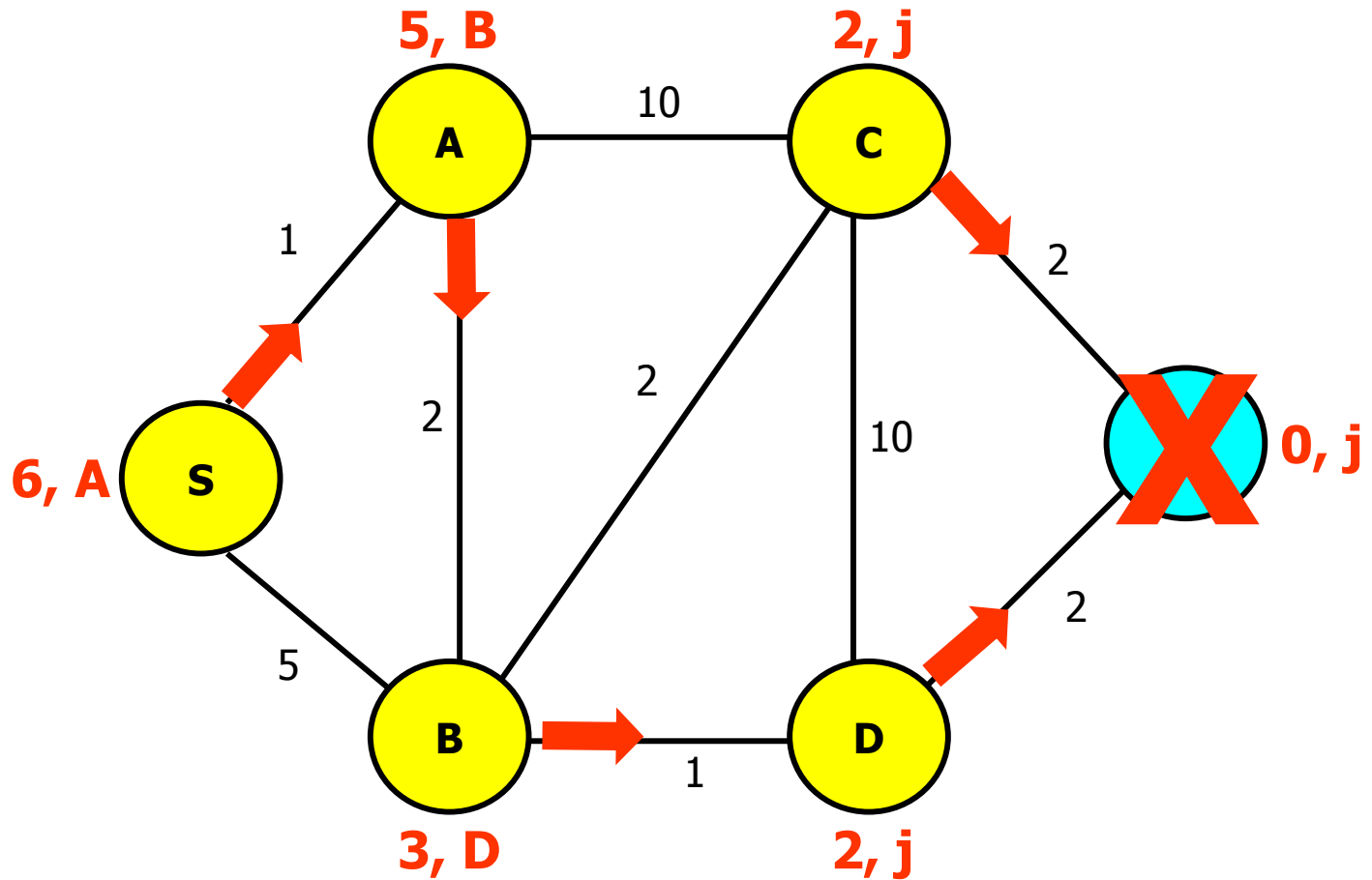
- **Proving that DBF Obtains Correct Routing Tables:**
 - By time $T_1 > T_0$, each router $h-1$ hops away from \mathbf{j} must send an update message reliably to each neighbor stating that its distance to \mathbf{j} is $h-1$.
 - Consider router \mathbf{x} , which is h hops away from \mathbf{j} ; this router must know all its neighbors within a finite time, and must receive the update from all its neighbors at $h-1$ hops from \mathbf{j} by time T_1 .
 - Whatever the distance to \mathbf{j} for node \mathbf{x} (at h hops from \mathbf{j}) is, using the BF equation, node \mathbf{x} must choose a neighbor $h-1$ hops from \mathbf{j} as its next hop to \mathbf{j} by some finite time $T_2 > T_1$.

Hence, DBF obtains correct routing tables!

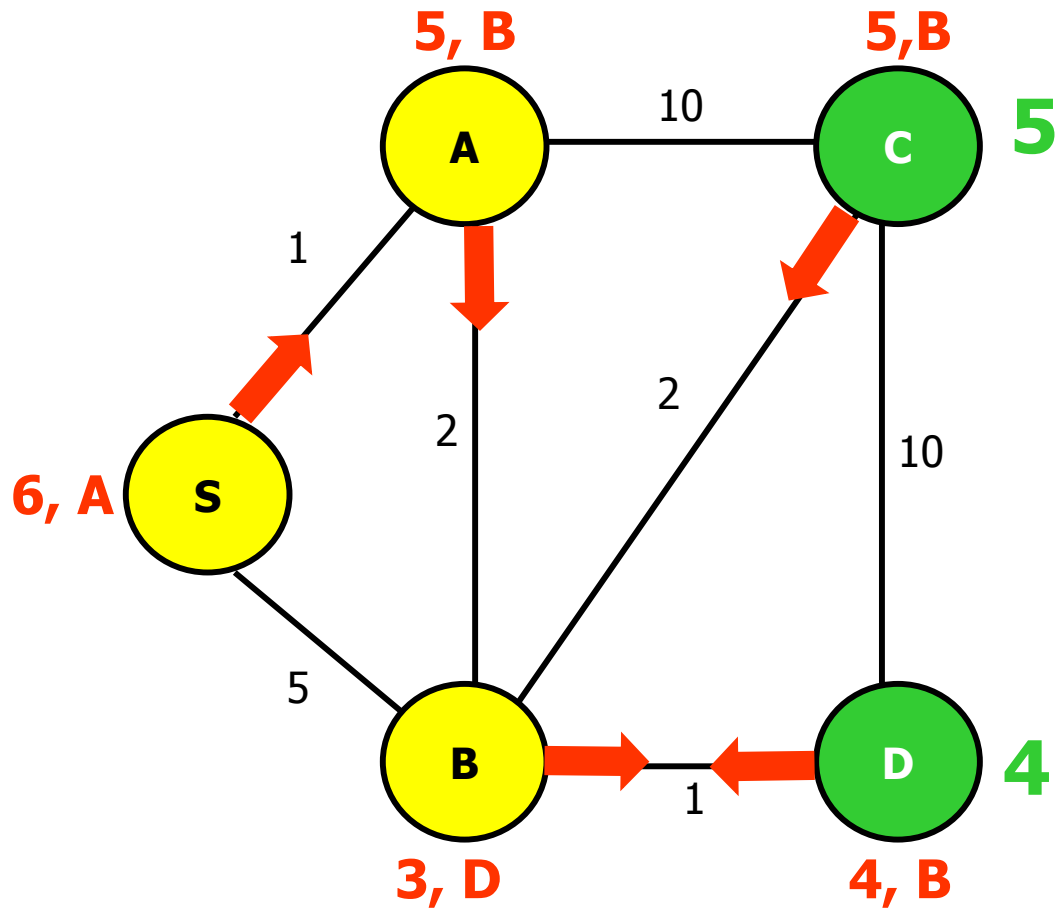
DBF Correctness

- **Proving that DBF Stops:**
 - A router updates the distance to a destination only after topology changes or update messages are received, and no more topology changes occur after time t .
 - A router sends an update message only if its distance or successor change.
 - After node i , h hops away from j , sends its update stating that its distance to j is h , it does not send more updates (**this requires that a router must continue using the same next hop unless its distance changes!**)
 - **Therefore, DBF stops.**

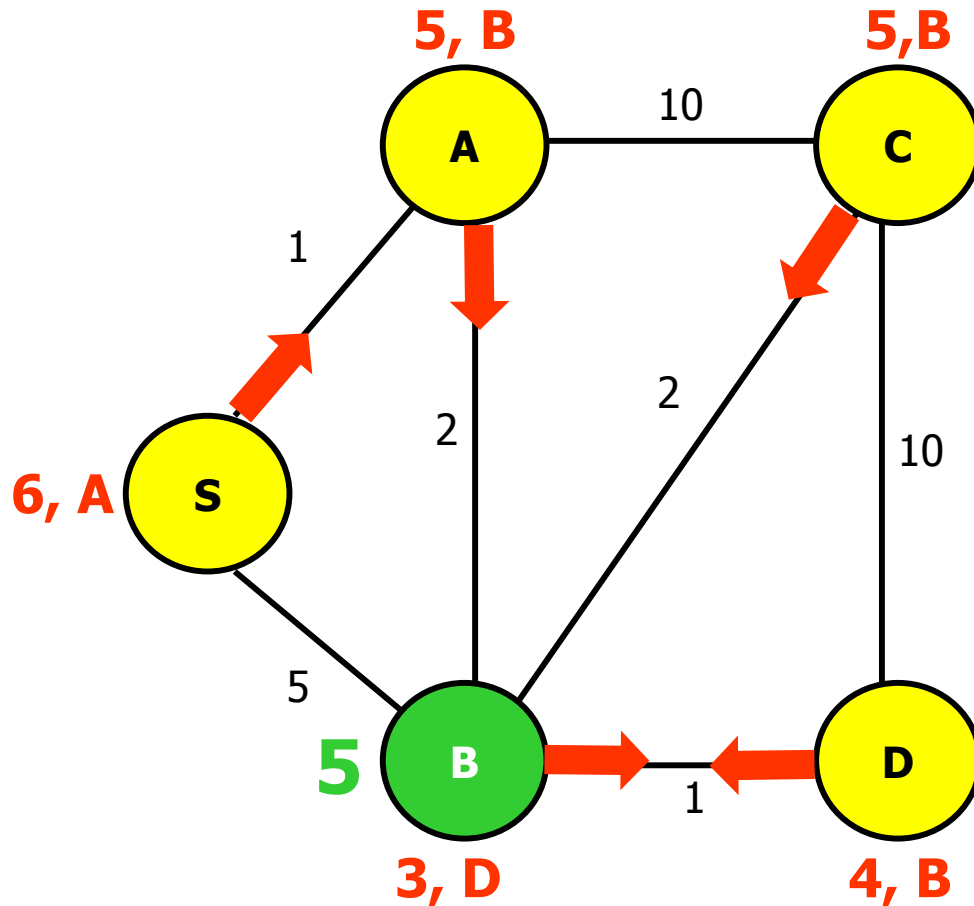
Looping in DBF



Looping in DBF



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

Examples of DBF Use

- First routing protocol of the ARPANET.
- Routing protocol of MERIT network.
- DARPA packet radio network.
- Routing Information Protocol (RIP)
- RIPv2
- Cisco's IGRP (DBF with split horizon, poison reverse, hold downs, and a complex link-cost metric).