Distance Vector Routing

a) The least-cost route between any two nodes is the route with minimum distance.

b) Each node maintains a vector(table) of minimum distances to every node.

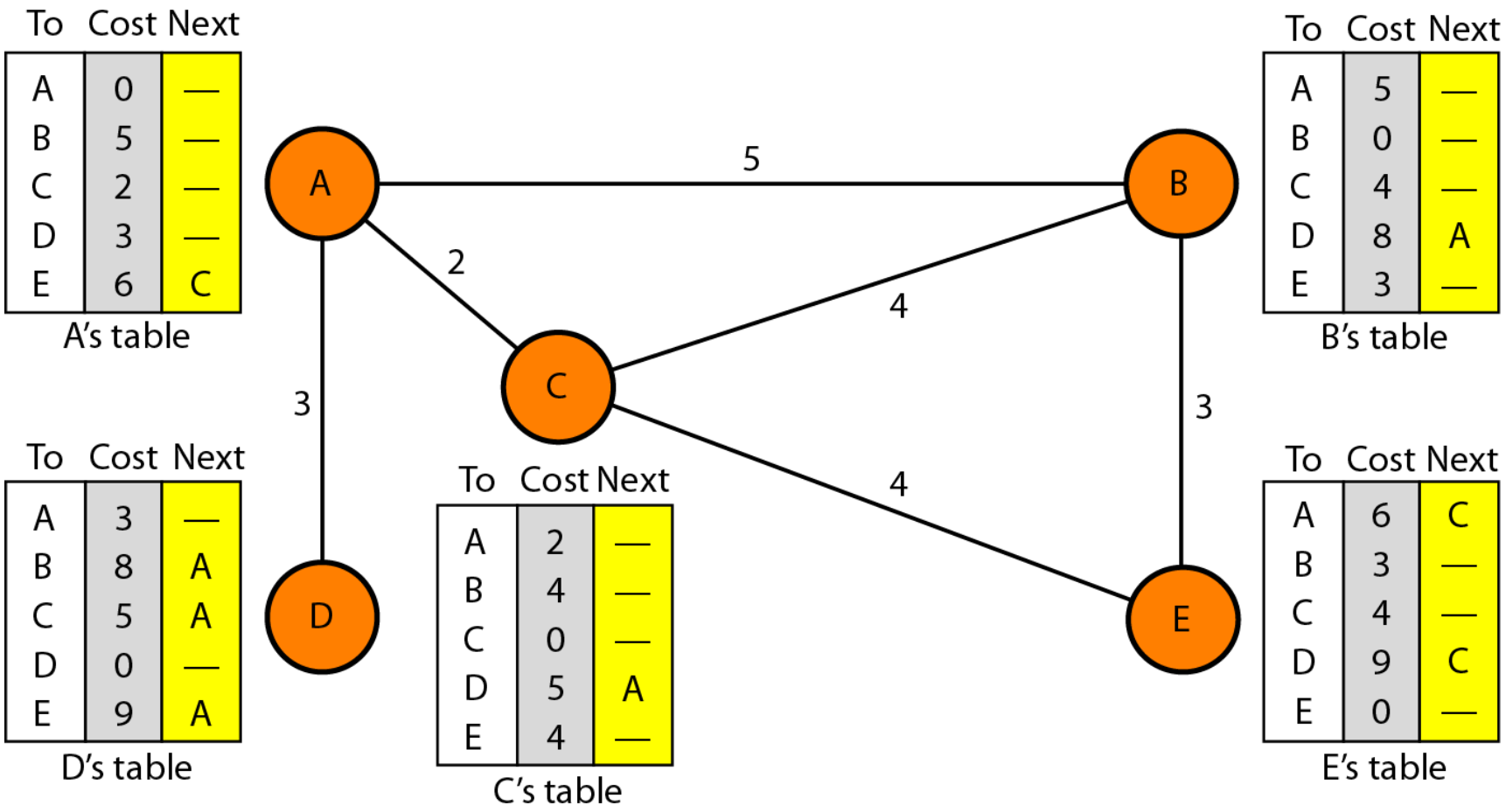
c) The table at each node also guides the packets to the desired node by showing the showing the next hop routing.

Example:

Assume each node as the cities.

Lines as the roads connecting them.

Final Distance vector routing tables



Initialization

a) The table in figure are stable.

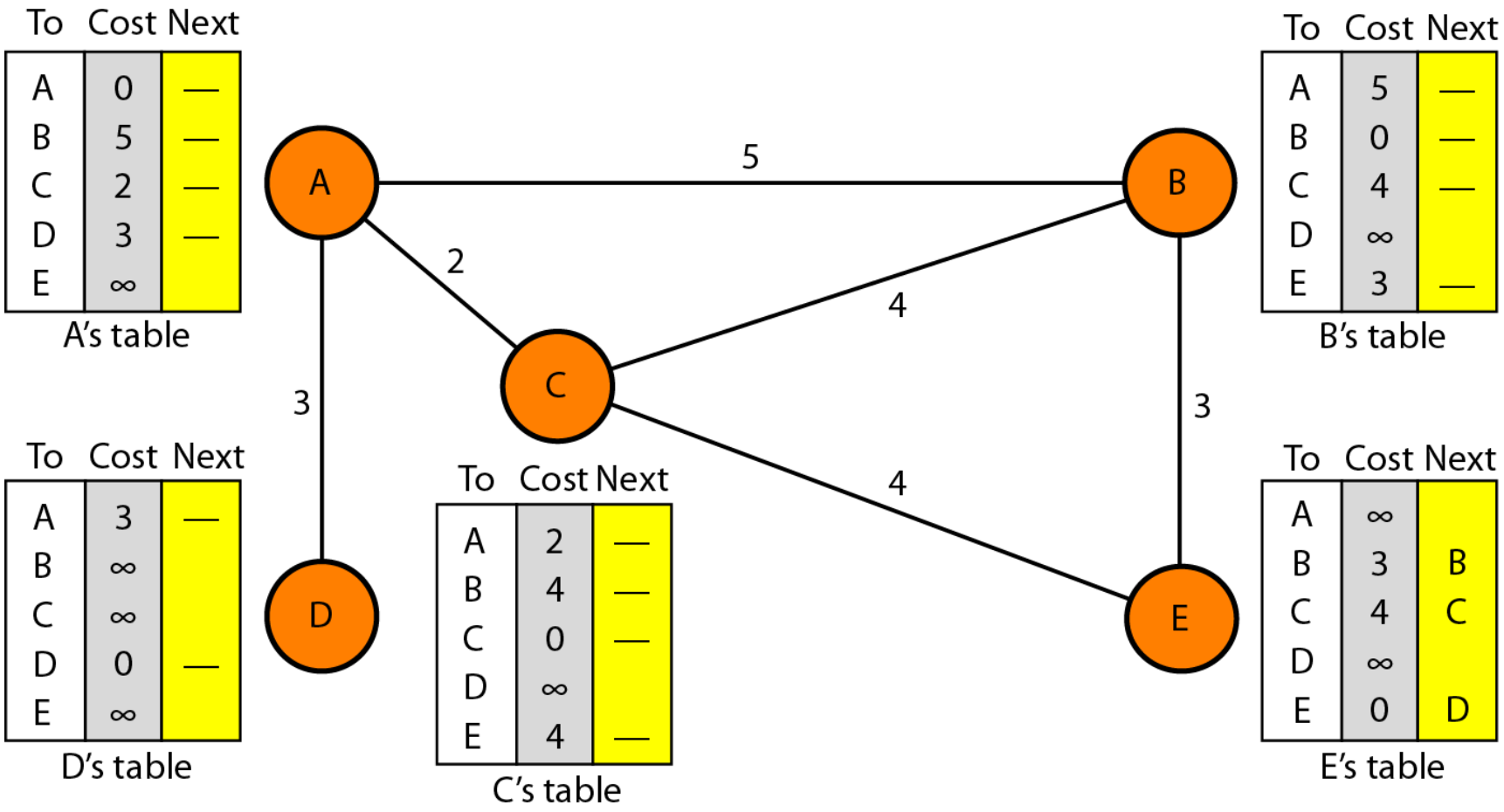
b) Each node knows how to reach any other node and their cost.

c) At the beginning, each node know the cost of itself and its immediate neighbor [those node directly connected to it].

d) Assume that each node send a message to the immediate neighbors and find the distance between itself and these neighbors.

e) The distance of any entry that is not a neighbor is marked as infinite(unreachable).

Initialization of tables in distance vector routing (DVR)



Sharing

a) Idea is to share the information between neighbors.

b) The node A does not know the distance about E, but node C does.

c) If node C share it routing table with A, node A can also know how to reach node E.

d) On the other hand, node C does not know how to reach node D, but node A does.

e) If node A share its routing table with C, then node C can also know how to reach node D.

f) Node A and C are immediate neighbors, can improve their routing tables if they help each other.

Sharing Contd.,

a) How much of the table must be shared with each neighbor?

b) The third column of the table(next hop) is not useful for the neighbor.

c) When the neighbor receives a table, this column needs to be replaced with the sender's name.

d) If any of the rows can be used, the next node column filled with sender of the table.

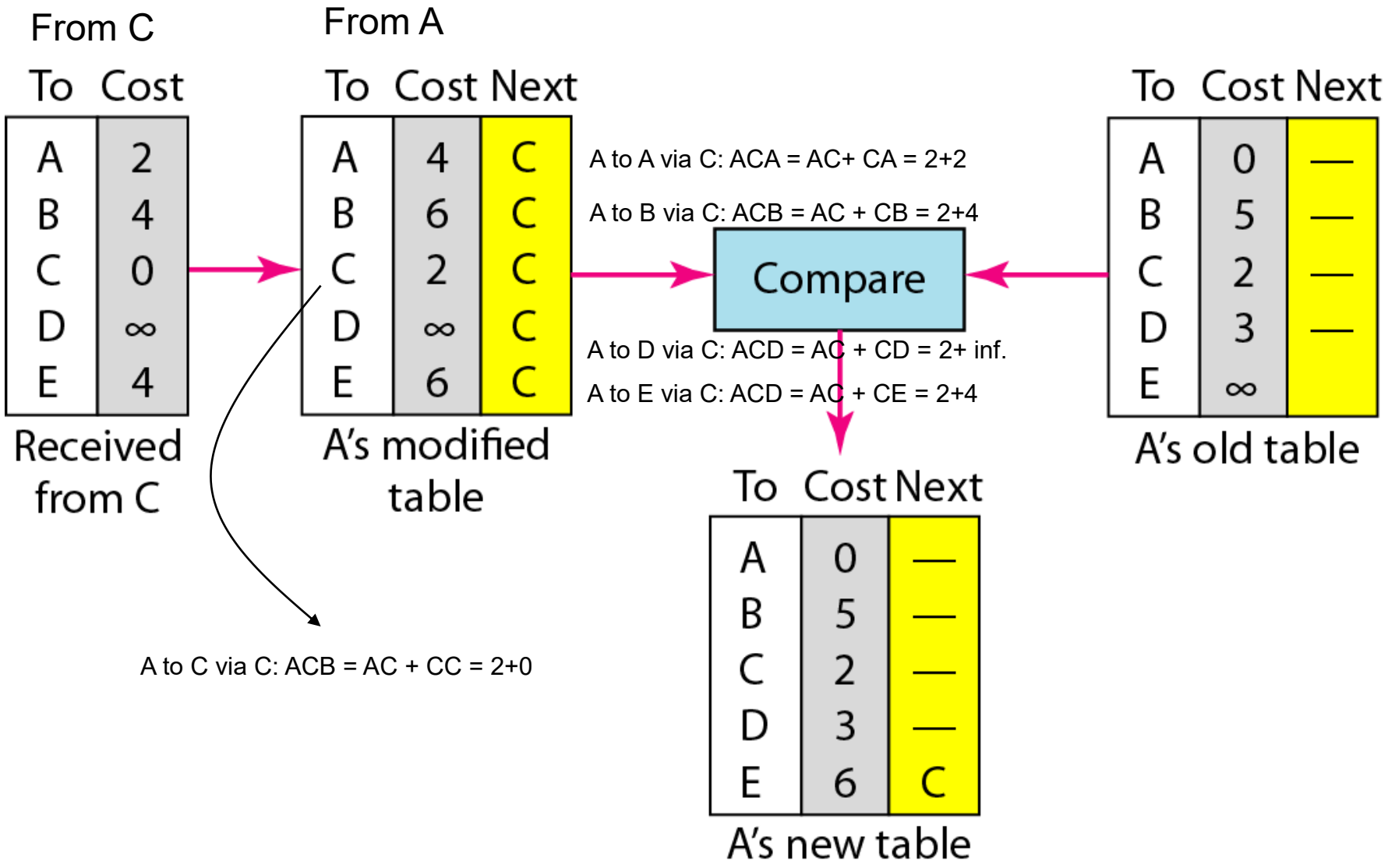
e) Therefore, a node can send only the first two column of its table to any neighbor.

Updating

When a node receives a two-column table from a neighbor, it needs to update its routing table. Updating takes three steps:

1. The receiving node needs to add the cost between itself and the sending node to each value in the second column. The logic is clear. If node C claims that its distance to a destination is x mi, and the distance between A and C is y mi, then the distance between A and that destination, via C, is $x + y$ mi.
2. The receiving node needs to add the name of the sending node to each row as the third column if the receiving node uses information from any row. The sending node is the next node in the route.
3. The receiving node needs to compare each row of its old table with the corresponding row of the modified version of the received table.
   a. If the next-node entry is different, the receiving node chooses the row with the smaller cost. If there is a tie, the old one is kept.
   b. If the next-node entry is the same, the receiving node chooses the new row. For example, suppose node C has previously advertised a route to node X with distance 3. Suppose that now there is no path between C and X; node C now advertises this route with a distance of infinity. Node A must not ignore this value even though its old entry is smaller. The old route does not exist any more. The new route has a distance of infinity.

Updating in distance vector routing example: C to A



When to share

a) Periodic Update: A node sends its table, normally every 30s, in a periodic update, it depends on the protocol that is using DVR.

b) Triggered Update: A node sends its two-column routing table to its neighbors anytime there is a change in its routing table.

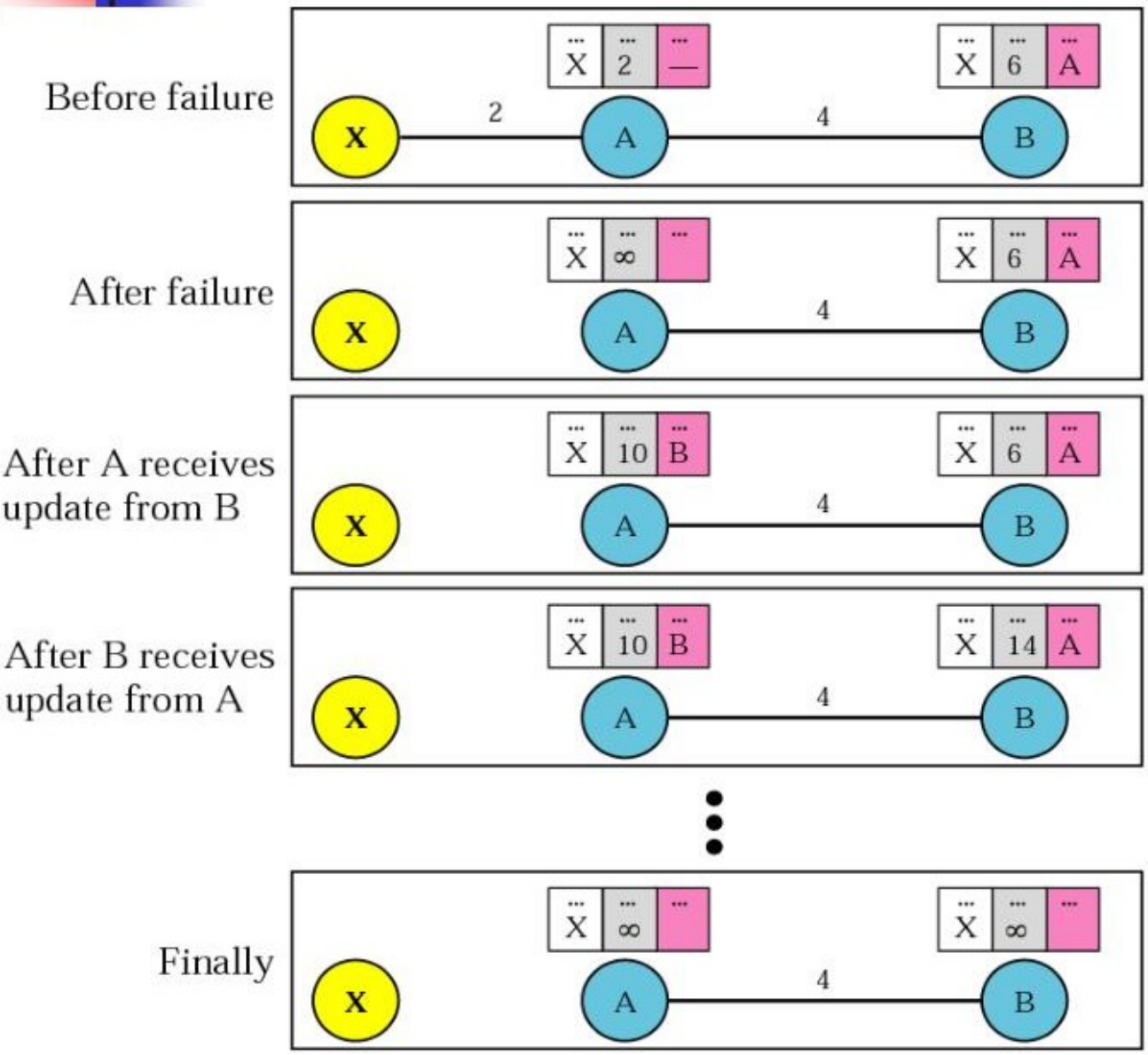
c) This is called triggered update the change can result from the following:

- ✓ A node receives a table from a neighbor, resulting in changes in its own table after updating.
- ✓ A node detects some failure in the neighboring links which results in a distance change to infinity.

Distance Vector Routing (DVR)

a) 3 keys to understand how this algorithm works:
   - **Sharing knowledge about the entire AS**. Each router shares its knowledge about the entire AS with neighbours. It sends whatever it has.
   - **Sharing only with immediate neighbours**. Each router sends whatever knowledge it has thru **all** its interface.
   - **Sharing at regular intervals**. sends at fixed intervals, e.g. every 30 sec.

b) Problems: Tedious comparing/updating process, slow response to infinite loop problem, huge list to be maintained!!



Both A and B know where X is.

Link between A and X fails. A updates its table immediately.

But before A can tell B, B sends its info to A!

A, using B's info, updates its table (error!). Then A send its table to B and B updates its table (more error).

Both routers keep updating tables, eventually hitting infinity. In the meantime, chaos!

Possible Solutions to two-node instability:

1. Define infinity to be a much smaller value, such as 100. Then it doesn't take too long to become stable. But now you can't use distance vector routing in large networks.

2. Split Horizon – instead of flooding entire table to each node, only part of its table is sent. More precisely, if node B thinks that the optimum router to reach X is via A, then B does not need to advertise this piece of info to A – the info has already come from A.

3. Split Horizon and Poison Reverse – Normally, the distance vector protocol uses a timer. If there is no news about a route, the node deletes the route from its table. So when A never hears from B about the route to X, it deletes it. Instead, Node B still advertises the value for X, but if the source of info is A, it replaces the distance with infinity, saying "Do not use this value; what I know about this route comes from you."