TORA : Temporally Ordered Routing Algorithm

- Invented by Vincent Park and M.Scott Corson from University of Maryland.
- TORA is an on-demand routing protocol.
- The main objective of TORA is to limit control message propagation in the highly dynamic mobile computing environment.
- Each node has to explicitly initiate a query when it needs to send data to a particular destination.

- TORA belongs to a class of algorithms called the link reversal algorithms. (We will defer a discussion of what this means to later).
- TORA essentially performs three tasks:
  - Creation of a route from a source to a destination.
  - Maintenance of the route.
  - Erasure of the route when the route is no longer valid.
- TORA attempts to build what is known as a directed acyclic graph (DAG) which is rooted at the destination.

Directed Acyclic Graph (DAG)

- A graph G(V,E) is a directed graph with V nodes and E edges if every edge in the graph has an associated direction.
- Now, if a path in a directed graph forms a cycle if it originates and terminates at the same node and has at least one link in it.
  - In other words, a path from v_0 to v_k in the graph forms a cycle if $v_0 = v_k$ and the path has at least one edge.
- A directed graph with no cycles is a DAG.
- A DAG is rooted at the destination if the destination is the only node with no downstream nodes, i.e., no links lead out of the destination. Such a DAG is often called a destination oriented DAG.
- Creation of such a DAG from a source to a destination would contain multiple routes to the destination.

- The idea is to first build a DAG from the source to the destination.
- Then as links fail, it might be necessary to re-compute a DAG in order to find a route.
  - Link Reversal algorithms are used for this.
- If network gets partitioned, erasures of routes is required.
- TORA uses three kinds of messages:
  - $ The QRY message for creating a route.
  - $ The UPD message for both creating and maintaining routes.
  - $ The CLR message for erasing a route.

Destination Disoriented DAG

- Topology of the Mobile Ad Hoc Network changes in time.
- Thus, it is possible that one of the nodes may have all inbound links (since for example, the only outbound link failed).
- This is not ok since only the destination should have this property.
- Such a graph is called “destination disoriented”.
- At this point TORA resorts to link reversals.

Link Reversals

- Let us call a process that is invoked in response to a topology change an “iteration”.
- Full Reversals
  - If this is invoked a node that is not the destination, upon finding that it has only inbound links will reverse the direction on all its links.
- Partial reversal
  - During each iteration every node (say a node Node i) keeps a list of all its neighbors j such that during the iteration, a link from node i to node j was in fact reversed to now show node j ✍ node i. It then reverses directions of links to only those nodes that do not belong to the list.

- Idea is similar to that of a fluid flow scheme.
- All the fluid is to flow from the source to the destination.
- The source is at the highest level, the destination is at the lowest level and the fluid flows from the highest level to the lowest level.
- Thus, if an intermediate node has all inbound links, it is a local minimum and it is possible that all the the fluid flows to this node rather than the destination. Now, the node has to increase its level ✍ !

TORA details and examples

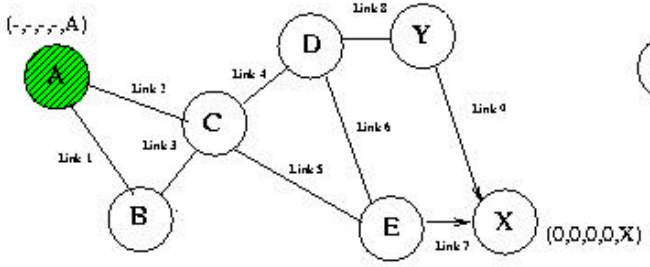
- Each node has an associated height. This height is represented as a quintuple $h_i = (?_i, oid_i, r_i, ?_i, i)$.
- This quintuple represents the height in terms of two parameters.
  - The first is called the reference level and is indicated by the first three elements.
  - The second is the offset from the reference level and is represented by the last two elements.
- When a node loses its last downstream neighbor it creates a new “reference level”.

- For this reference level, $?_i$ is the time at which the failure occurred.
- oid_i is the ID of the originating node.
- r_i indicates one of two unique sub-levels in the reference level – why ? what ? Later ✍ !

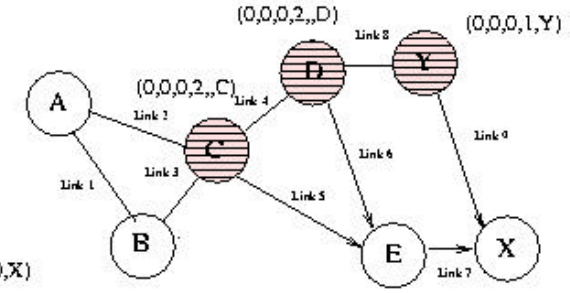
- The second part refers to an offset where:
  - ✍ $?_i$ refers to an integer which is used to order nodes with respect to the reference level.
  - ✍ i is the ID of the node itself.
- Why do I think an example is necessary now ✍ ?
- But a few more things ....

- Initially height of each node is NULL and is represented by (-,-,-,-, i).
- The height of the destination is set to zero, i.e., h_{dest} = (0,0,0,0,dest).
- Each node reports its height to its neighbors.
- If no info about a neighbor set its height to NULL.
- If neighbor is destination, set its height to zero.
- Each node also maintains the state of the links incident on it. If its height is greater than its neighbor j, the link to j is marked downstream (DOWN). Else it is marked upstream (UP).

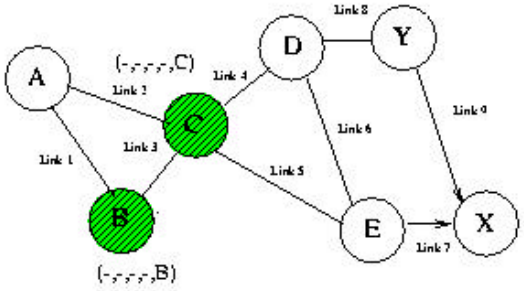
Example of TORA operations



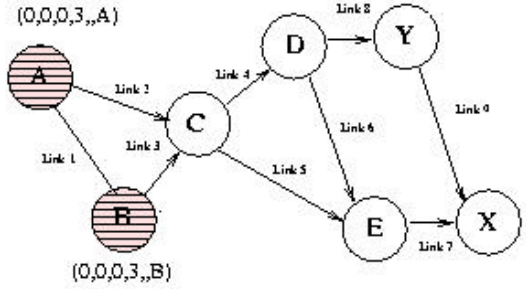
(a)



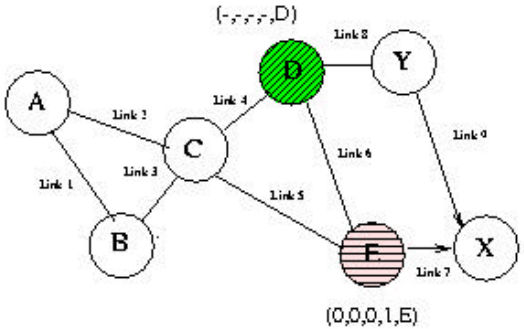
(d)



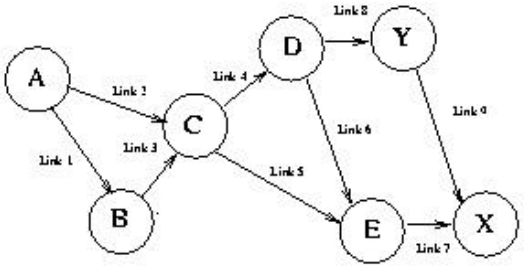
(b)



(e)

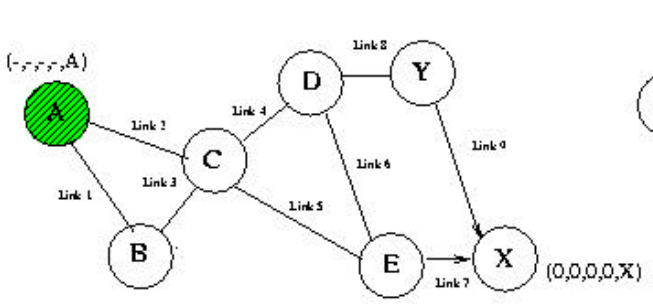


(c)

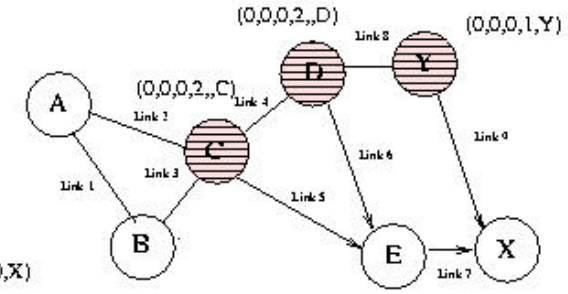


(f)

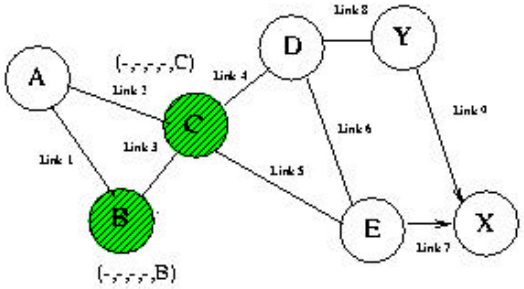
- The destination is X.
- A needs to find a route to X.
- A sends a QRY packet to C.
- C and B broadcast this message.
- E knows that X is one hop away.
- D broadcasts QRY message (contains height which is currently NULL).
- What does E do ?



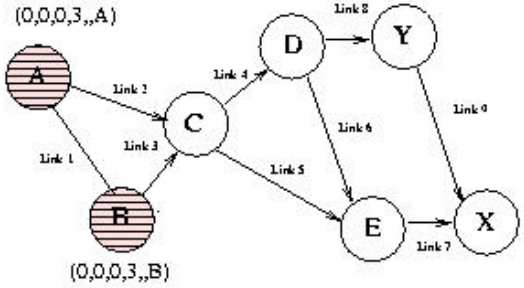
(a)



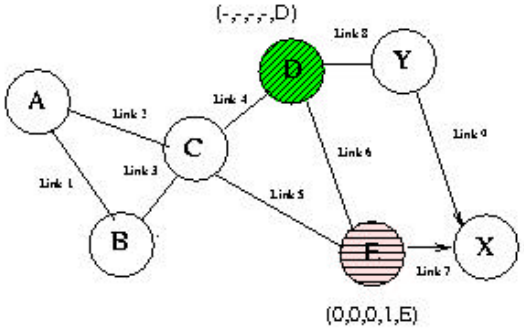
(d)



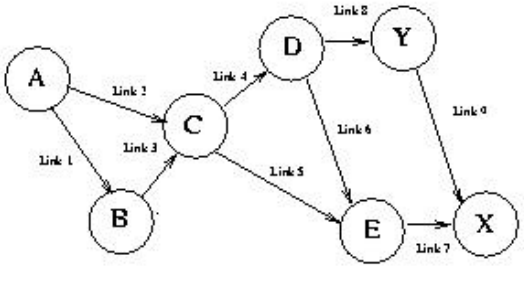
(b)



(e)

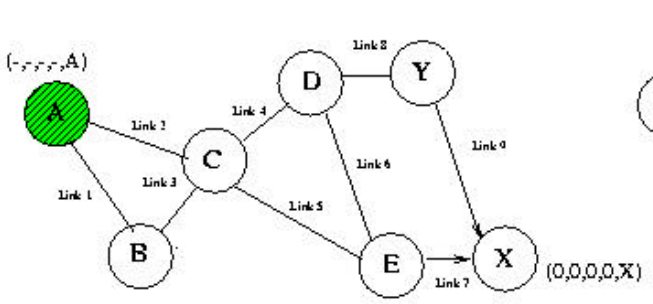


(c)

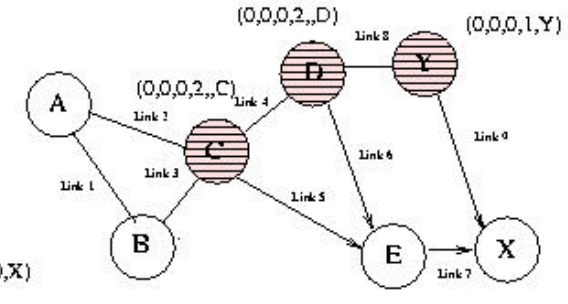


(f)

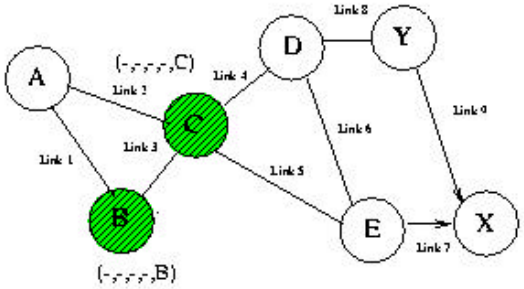
- E creates a reference level (0,0,0) and indicates that it has an offset height $?_E = 1$ from the reference level.
- It then generates an UPD message.
- It also sets a direction for its link to X i.e., from E to X. Why ?



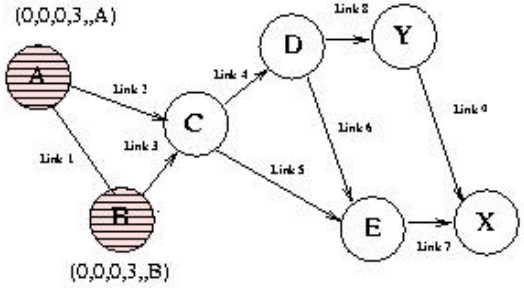
(a)



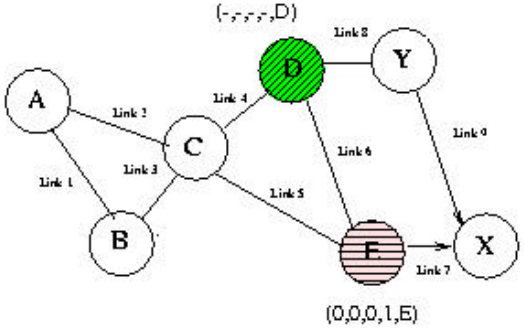
(d)



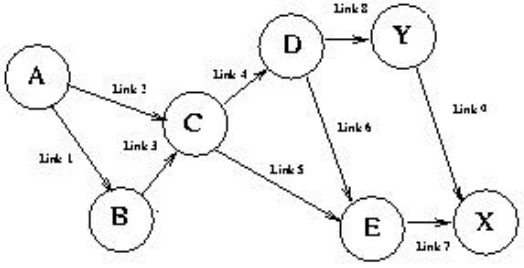
(b)



(e)



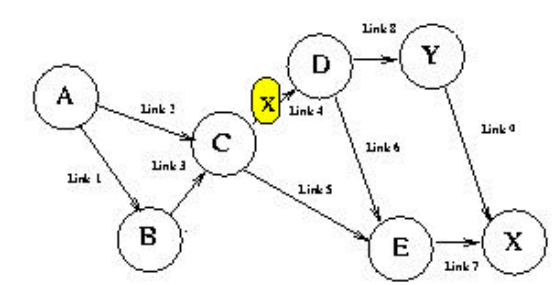
(c)



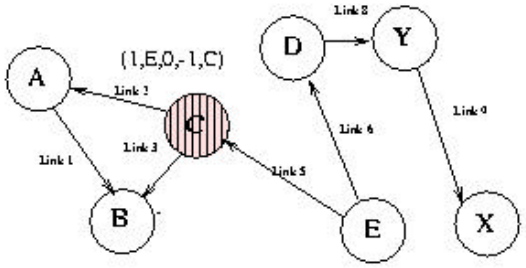
(f)

- C and D receive this message from E, and generate an update. They also set E to be a downstream node.
- Future broadcasts of the UPD message results in the generation of the DAG.
- Note in the final part of the figure, all links are directed and the destination, viz. X does not have any downstream nodes.

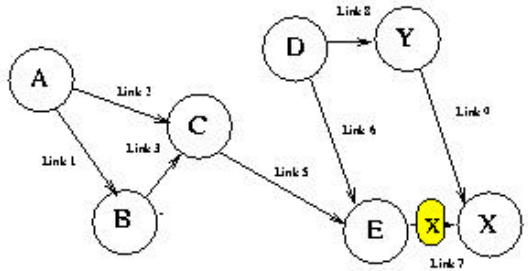
Reacting to failures: Route Maintenance



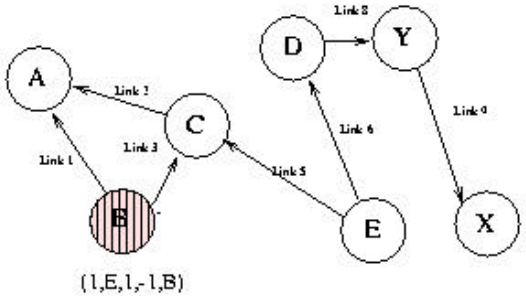
(a)



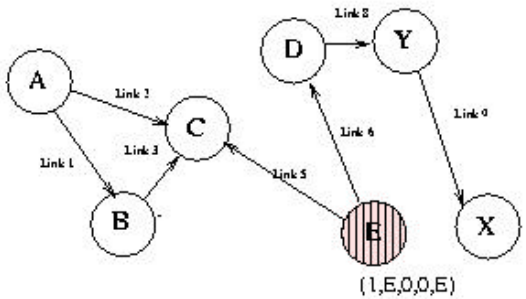
(d)



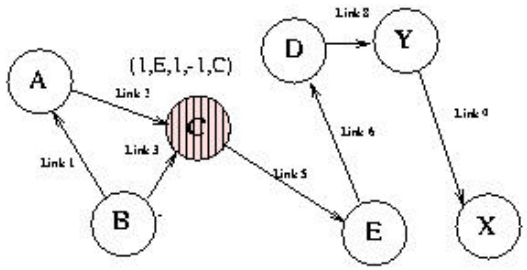
(b)



(e)

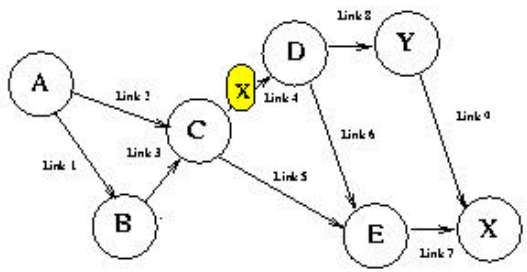


(c)

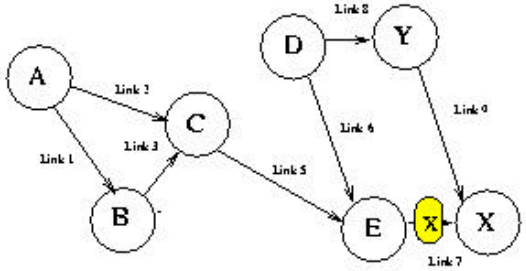


(f)

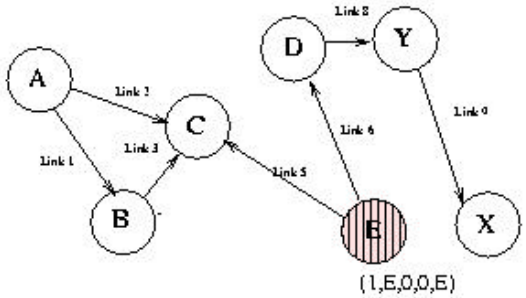
- Let Link 4 fail.
- At this time notice that other than the destination all nodes still have an outbound link.
- Thus, none of the nodes generate an UPD message.
- The DAG is still OK ✍ !
- This is especially attractive when the network is dense – most nodes have many outbound links.



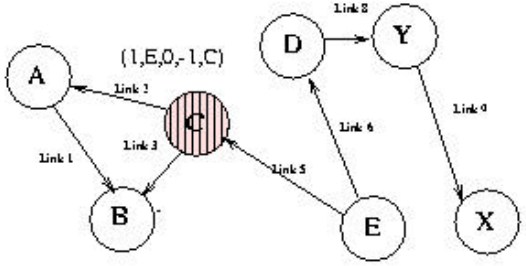
(a)



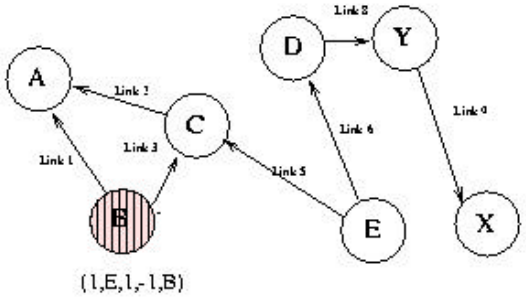
(b)



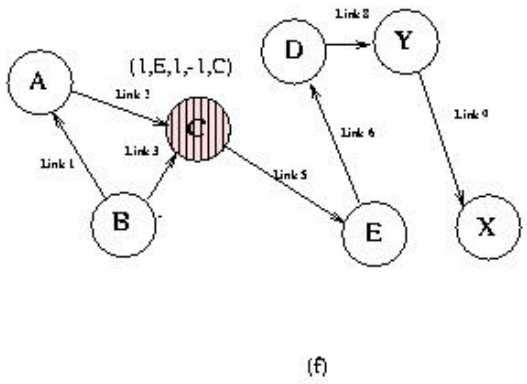
(c)



(d)

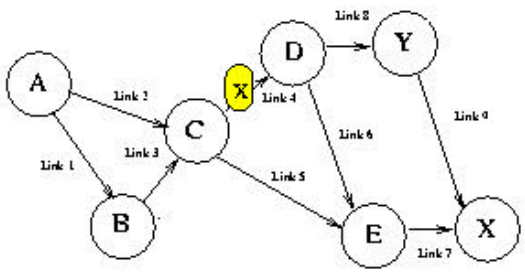


(e)

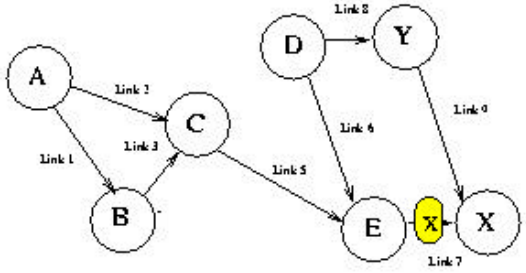


(f)

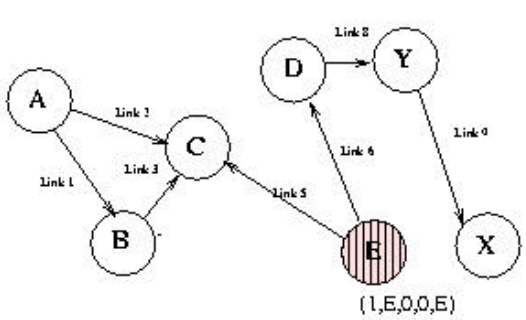
- Let Link 7 fail.
- Now, Node E does not have any outbound links !
- Thus, we resort to full link reversal at E.
- E generates a new reference level which is 1, sets the oid to E and transmits an UPD message.
- It also reverses the direction of all its inbound links.



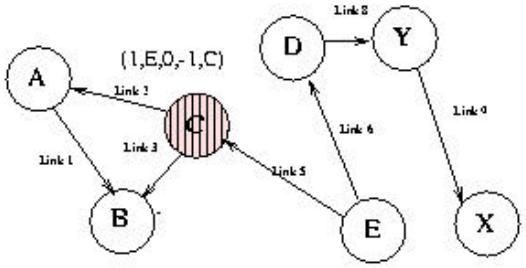
(a)



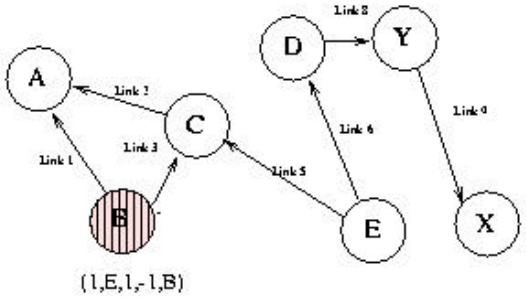
(b)



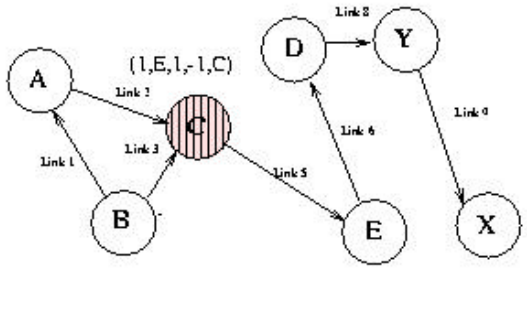
(c)



(d)

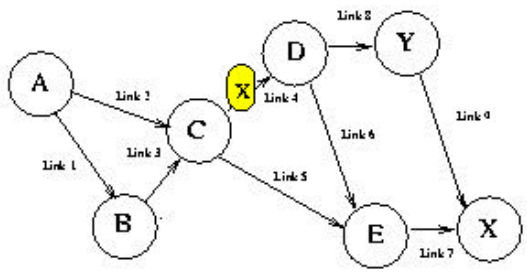


(e)

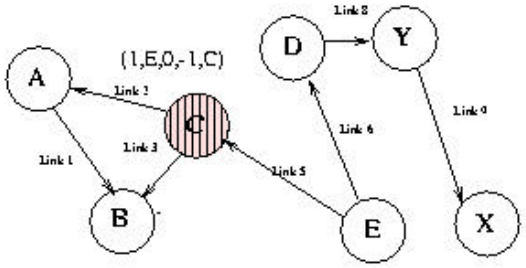


(f)

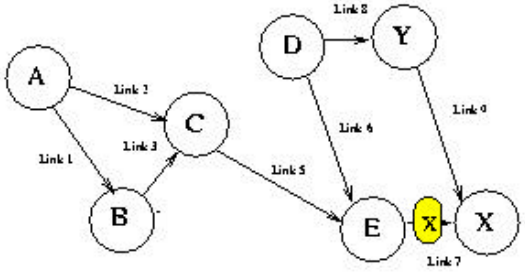
- At this, Node C no longer has outbound links.
- It resorts to partial link reversal – reverses the direction of its links to A and B and transmits an UPD.
- It also sets its own offset to –1 to ensure that it is at a lower level compared to E.



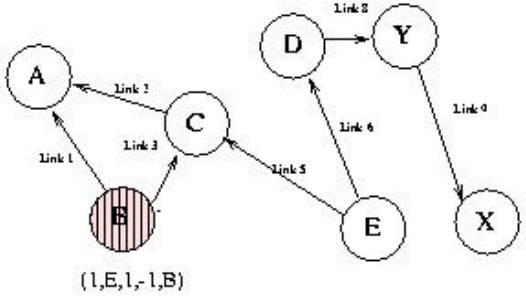
(a)



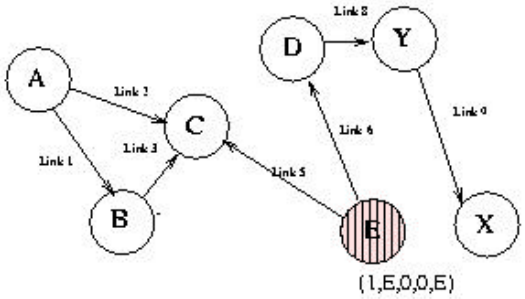
(d)



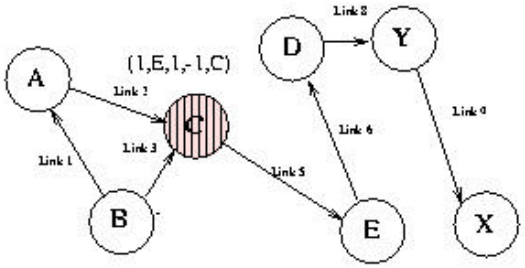
(b)



(e)

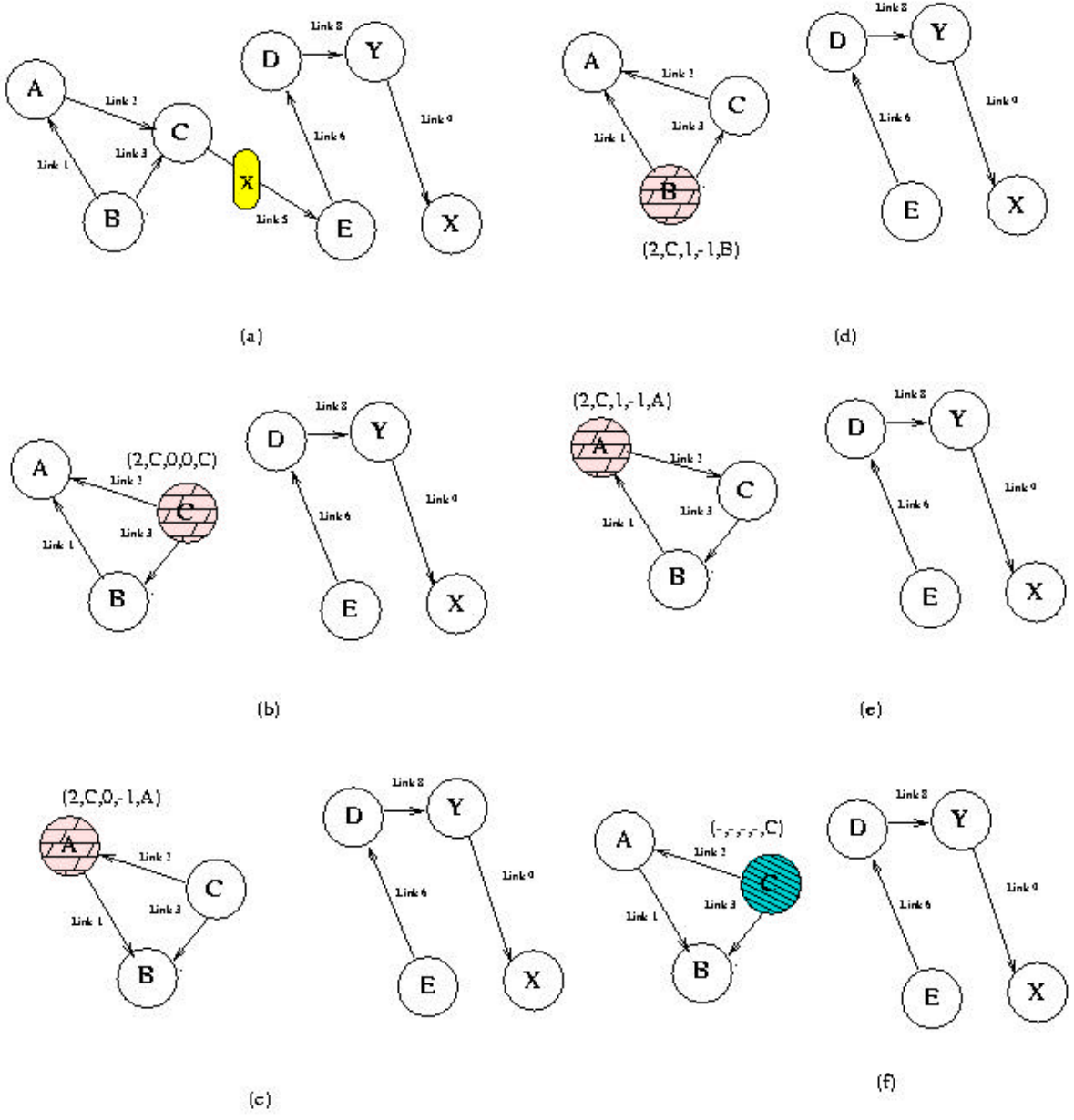


(c)

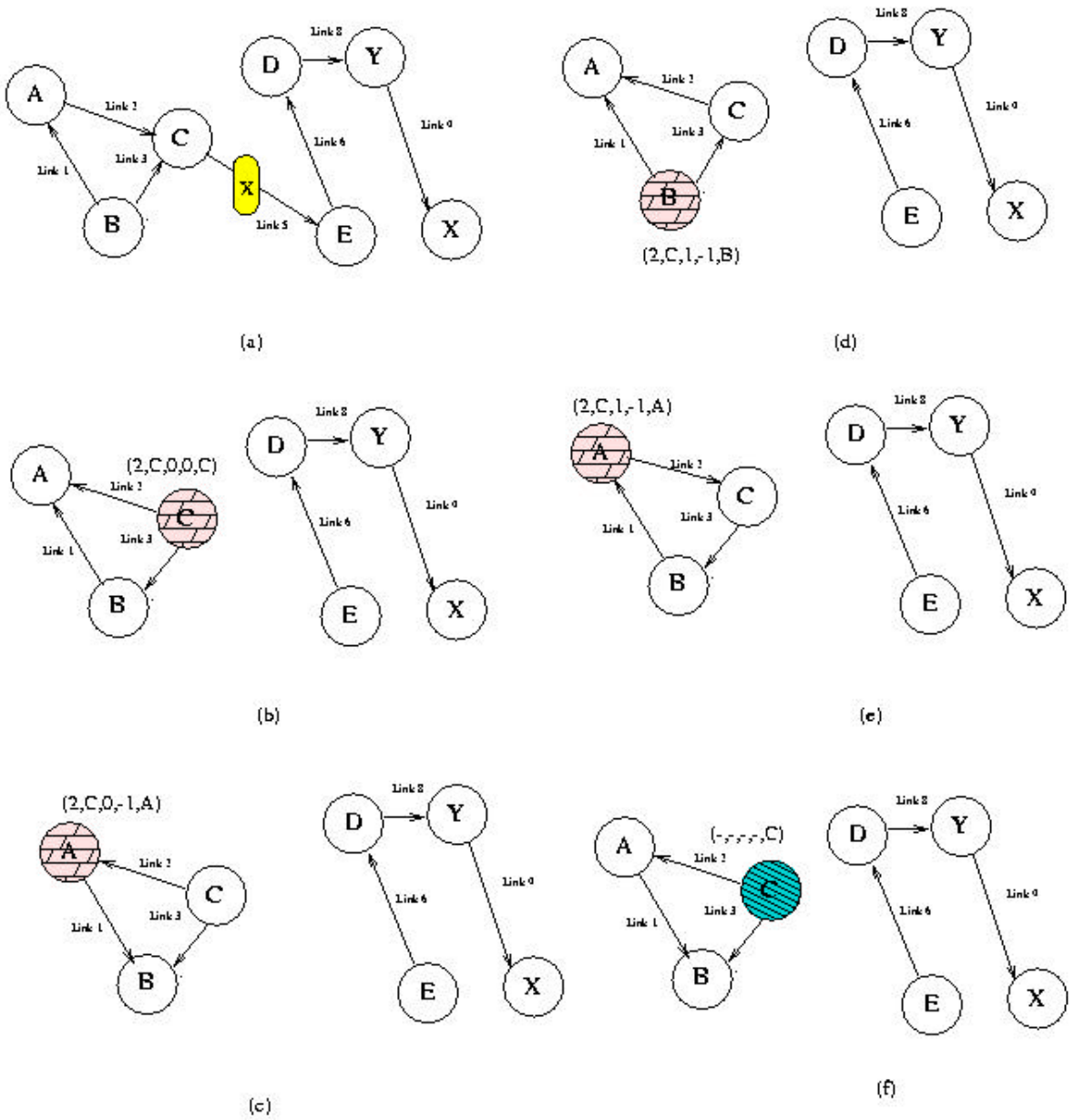


(f)

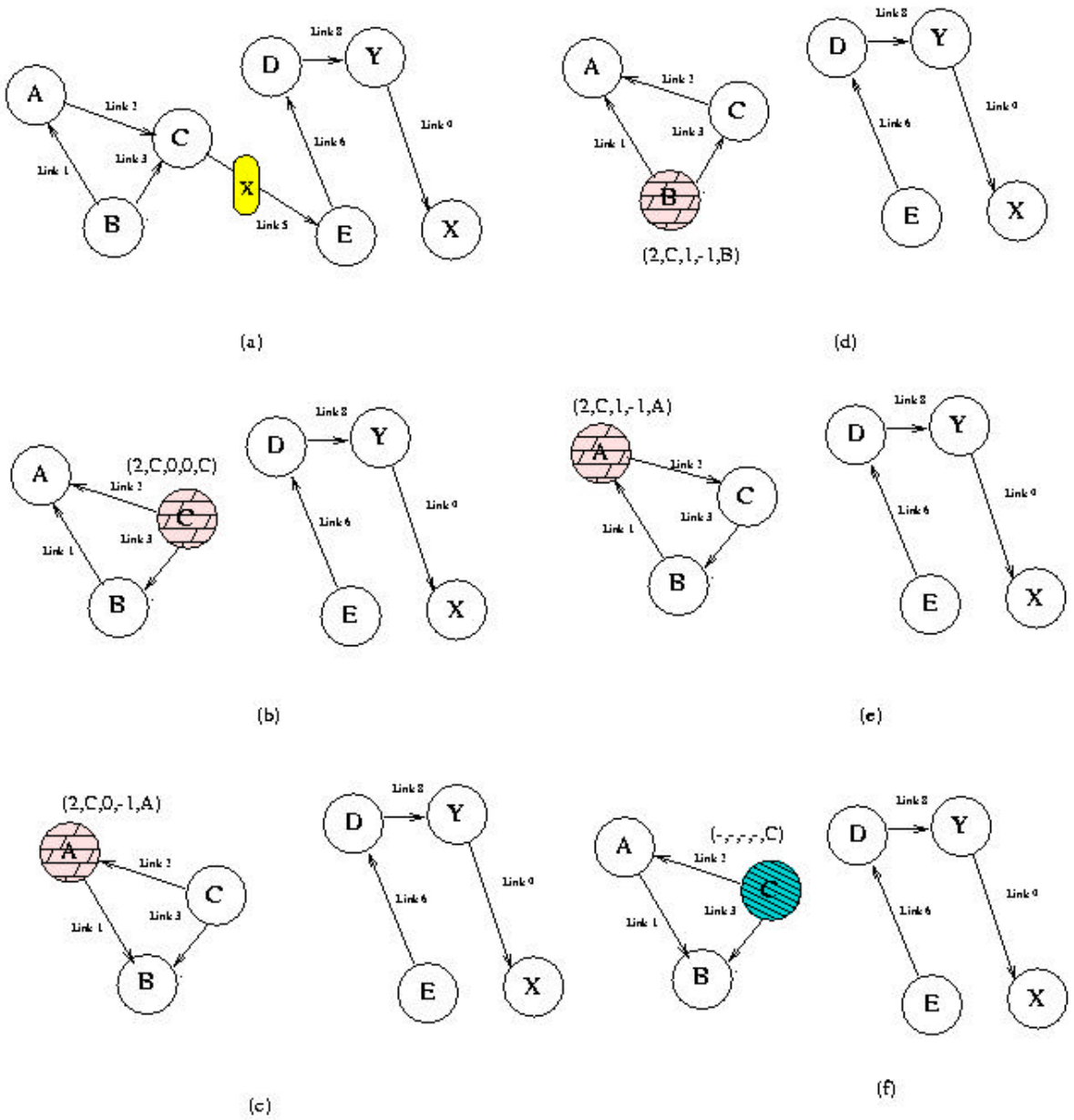
- Now the situation repeats at B.
- After B reverses its links and transmits an UPDATE. This is now a full reversal. Thus, it stays at the same level as C, but indicates the full reversal by flipping r_i.
- This causes a partial reversal at A. (Not shown)
- Finally an update is generated at C.
- DAG is restored ✍!



- Now let Link 5 fail. This causes a network partition.
- E,D,Y and X are ok.
- C has no outbound links.
- It creates a new reference level which is 2, and sets the oid to C and sends a UPD.
- This causes A to have no outbound links.



- A resorts to partial reversal.
- It sets its height to –1, reverses its link to B and broadcasts an update.
- Now B does not have outbound links.
- It resorts to a full reversal. At full reversal r_i is flipped.
- This causes a partial reversal at A.



- A's UPD message after the partial reversal creates the same situation at C.
- This would cause C to realize that there is no path to X. It sets its height to NULL and sends an UPD to A and B.
- Now the nodes realize that there is no path to X.

Advantages:

- That of an on-demand routing protocol – create a DAG only when necessary.
- Multiple paths created.
- Good in dense networks.

Disadvantages

- Same as on-demand routing protocols.
- Not much used since DSR and AODV outperform TORA.
- Not scalable by any means.

References

- Chapter 8 of book.
- V.D.Park and Scott.M.Corson, “A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks”, Proceedings of INFOCOM 1997.