Lecture : The Temporally Ordered Routing Algorithm

Lecture 1 : Basic ideas behind the TORA algorithm and initialization of a network

Lecture 2 : Route maintenance and detailed description

TORA

- TORA is a merger of the ideas from the GB algorithm and the LMR algorithm.
- It uses the QRY-RPY mechanism of the LMR algorithm as well as the partial link reversal mechanism of the GB algorithm.
- However, both of these mechanisms are modified in TORA.

The Properties of TORA

- The key feature of TORA is its reaction to link failures. It erases invalid routes, searches for new routes and builds new routes in a single-pass of the distributed algorithm.
- LMR uses a two-pass mechanism through FQ/RPY messages.
- DSR and AODV use a three-pass mechanism through route error/ route request / route reply messages.

The Properties of TORA

- This aggresive single-pass route search and rebuilding capability results in high localization of rebuilding and low control overhead in highly connected networks.
- This behaviour is achieved through the use of a physical or logical clock to establish a temporal order of topological changes.
- We will assume initially that each node maintains a synchronized clock, however we will show later that this assumption is not necessary.

Description of the Protocol

- TORA is a distributed protocol. A logically separate version of TORA is run for each destination to which routing is required.
- The protocol has three basic functions :
  - Route creation
  - Route maintenance
  - Route erasure
- Like other link reversal protocols, a Directed Acyclic Graph (DAG) is maintained rooted at the destination.

Height of a Node

- A link in the DAG is directed according to the relative heights of the two nodes that define the link.
- A downstream link is from a higher to a lower height and an upstream link is from a lower to a higher height.
- Links to neighbouring nodes with an unknown or null height are considered undirected and cannot be used for routing.

Route Creation, Maintenance and Erasure

- The route creation process converts an undirected network into a DAG rooted at the destination by assigning directions to the links.
- Some nodes may lose all paths to destinations due to link failures. The purpose of route maintenance is to reverse some of the links so that network reorients itself in a state where each node has a path to the destination.
- If a network is partitioned, the route erasure mechanism erases all paths in partitions which do not contain the destination.

Things to Remember

- TORA uses both full reversal and partial reversal according to the terminology of the GB algorithm.
- The basic strategy each node follows is to find a route through any neighbour that has a valid route to the destination.
- A node defines a new height only if no neighbour has a valid route.

Definition of Height

- At any given time and for each destination, an ordered quintuple is associated with each node :

$$H_i = (\tau_i, oid_i, r_i, \delta_i, i)$$

(τ_i, oid_i, r_i) is called the reference level and

(δ_i, i) is called the offset with respect to the reference level.

Definition of Height

- A new reference level is defined each time a node loses its last downstream link due to a link failure.
- τ_i is a time tag set to the time of the link failure
- oid_i is the originator ID, i.e., the ID of the node that defined this reference level.
- r_i is a single bit where 0 indicates an original level and 1 indicates a reflected level
- δ_i is an integer used to order nodes with respect to a common reference level

Definition of Height

- i is the ID of the node itself.
- This quintuple associated with each node ensures that all nodes can be totally ordered lexicographically at all time.
- We need the quintuple since in several cases the height of two nodes may be same according to their reference level. The node ID alone may not be sufficient to order the nodes lexicographically.

Initial Heights

- Each node i other than the destination maintains its height H_i with respect to the destination.
- Initially, the height of each node is set to null, i.e., $H_i = (-,-,-,-,i)$
- The height of the destination is always 0, i.e.,

$$H_{dest} = (0,0,0,0,dest)$$

- Each node i maintains a height array $HN_{i,j}$ for each neighbour $j \varepsilon N_i$

Link Status Array

- Initially the height of each neighbour is set to null, i.e., $HN_{i,j} = (-,-,-,-,j)$. If destination is a neighbour of node i , then

$$HN_{i,dest} = (0,0,0,0,dest)$$

- The status of a link between i and a neighbour j is detemined by the heights H_i and $HN_{i,j}$ and directed from the higher to the lower node.
- If the height of j is higher than the height of i then the link from i to j is upstream (UP), otherwise it is downstream (DN).

Link Status Array

- If the neighbour´s height entry, $HN_{i,j}$ is null, then the link is undirected (UN).
- If the height of node i is null, then any neighbour´s height that is not null is considered lower and the coresponding link is downstream.
- When node i has a new neighbour j, node i adds entries for j in its height and link status arrays.
- Nodes need not exchange any height information during link activation.

Creating Routes

- Like the Lightweight Mobile Routing (LMR) algorithm, creating routes requires the use of two packets QRY and UPD.
- A QRY packet consists of a DEST field which identifies the destination.
- An UPD packet consists of a DEST field and the height of the node i , i.e., H_i . Here i is broadcasting the UPD packet.
- Each node i maintains a route required flag RR_i (initially 0) and the time when it broadcast the last UPD packet.

Creating Routes

- When a node i with no directed links and with the value $RR_i = 0$ requires a route to DEST, it broadcasts a QRY packet and sets $RR_i = 1$
- There are four different cases how the node i reacts when it receives a QRY packet.
- Case 1 :
- If $RR_i = 1$, i discards the QRY packet, as this means that i has already broadcast a QRY packet for DEST.

Actions Taken When a QRY is Received (Case 2)

- Case 2 :
- If $RR_i = 0$ and H_i is non-null with $r_i = 0$ node i should send an UPD packet if it has not done so before.
- It first compares the time when it broadcast the last UPD packet to the time when the link (over which the QRY was received) was activated.
- If the last UPD packet was sent after the link was activated, no action is needed. Otherwise, i broadcasts a UPD packet which includes its current height H_i .

Actions Taken When a QRY is Received (Case 3)

- Case 3 : If $RR_i = 0$ and H_i is either null or non-null with $r_i = 1$ but has a neighbour j whose height is non-null with $r_j = 0$, i
  - Sets its height to $H_i = (\tau_j, oid_j, r_j, \delta_j + 1, i)$

    where $HN_{i,j} = (\tau_j, oid_j, r_j, \delta_j, j)$ is the minimum height of its non-null neighbours with $r_j = 0$
  - Updates all the entries in its link status array LS
  - Broadcasts a UPD packet that contains its new height H_i

Purpose of Case 3

- If a node has null height or if its $r_i = 1$, the node currently holds an invalid height.
- Hence, the node´s invalid height is updated by taking the valid height from a neighbour and broadcasting it in a UPD packet.
- The main idea is the following. If a neighbour has a valid height, the node borrows this height and increases its δ value, in effect creating a higher offset and a downstream link from itself to the neighbour.

Actions Taken When a QRY is Received (Case 4)

- Case 4 : If none of the above conditions are true, i.e., $RR_i = 0$ and there is no neighbour with a valid height, the node i broadcasts the QRY and sets $RR_i = 1$
- In this case, no neighbour has a valid path to DEST and hence the QRY should be propagated further to find a path.

Actions When a New Link is Established

- Whenever, node i has a new neighbour establishing a new link, i broadcasts a QRY packet over this new link if $RR_i = 1$
- This ensures that the search for a route to the destination continues to propagate as the network topology changes.
- The destination may have been unreachable earlier from the source that propagated the QRY. But now the destination may be reachable due to some new link.

Actions When a UPD Packet is Received

- When a node i receives a UPD packet from a neighbour j, it first updates the entry $HN_{i,j}$ with H_j contained in the received UPD packet
- Then it does the following if $RR_i = 1, HN_{i,j}$ is non-null , $r_j = 0$ (the action is given in next slide)
- $r_j = 0$ means the neighbour has a valid route.

Actions When a UPD Packet is Received

1. Sets $H_i = (\tau_j, oid_j, r_j, \delta_j + 1, i)$ where the height $H_j = (\tau_j, oid_j, r_j, \delta_j, j)$ this is a partial reversal
1. Updates all the entries in its link status array LS.
2. Sets $RR_i = 0$
3. Broadcasts a UPD packet that contains its new height H_i

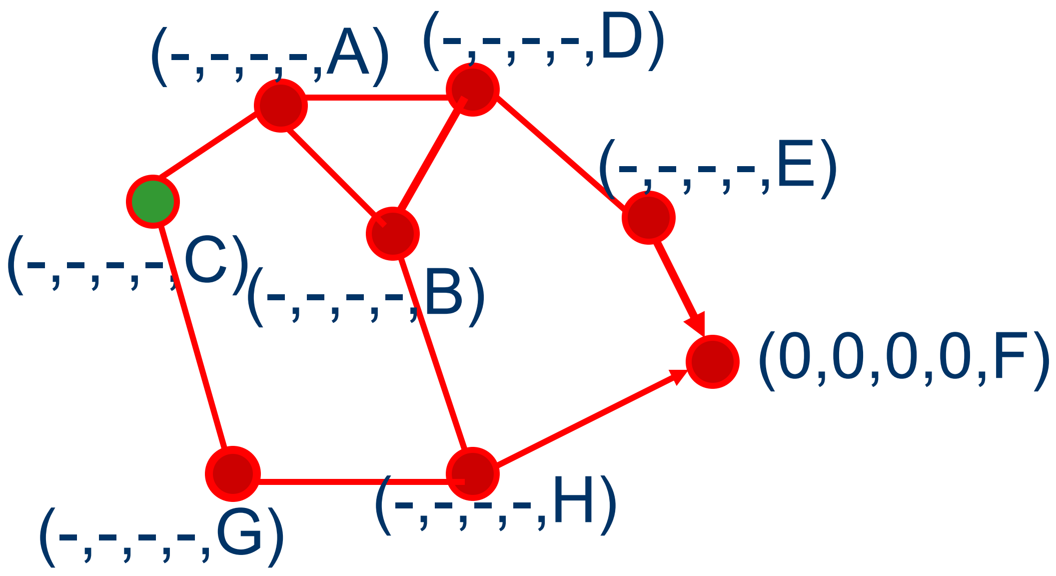
The Purpose of the action

- The purpose of these updates is the following.
- Previously i did not have a route.
- Now it has received an UPD packet from a neighbour that has a valid route ($r = 0$ indicates a valid route).
- So, i now establishes a downstream link to this neighbour since the δ value is $\delta_j + 1$
- Also i broadcasts a UPD packet in case its other neighbours want to route through it.

Actions When a UPD Packet is Received

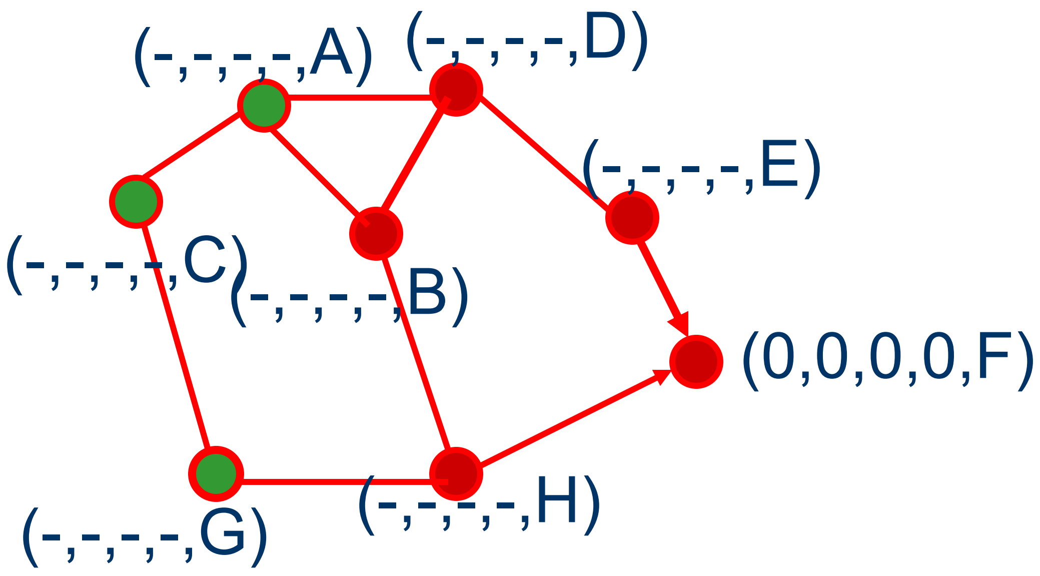
- If the preceding condition is not true, i.e., when either a route is not required or a neighbour does not offer a valid path, then
- Node i simply updates the entry $LS_{i,j}$ in its link status array.
- This second condition may result in the loss of the last downstream link for node i
- We will discuss route maintenance later.

An Example



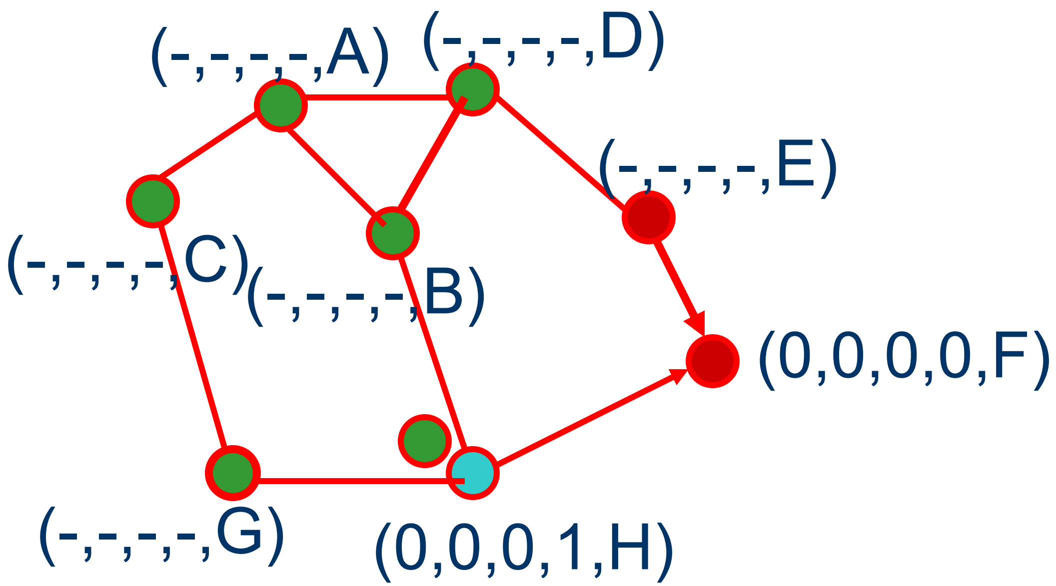
- Nodes that have QRY ($RR_i = 0$)
- Nodes that have UPD

An Example



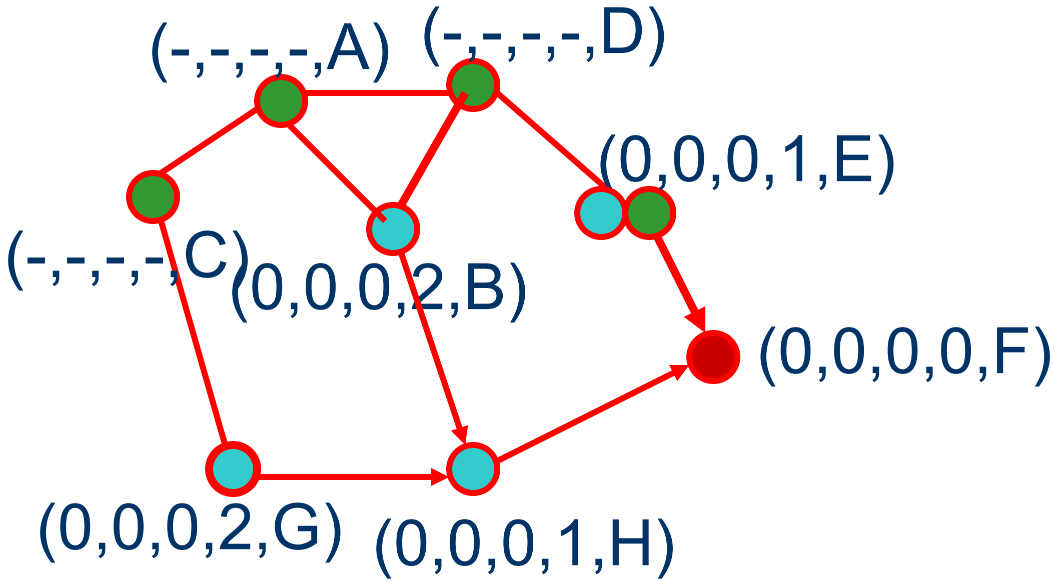
- Nodes that have QRY ($RR_i = 0$)
- Nodes that have UPD

An Example



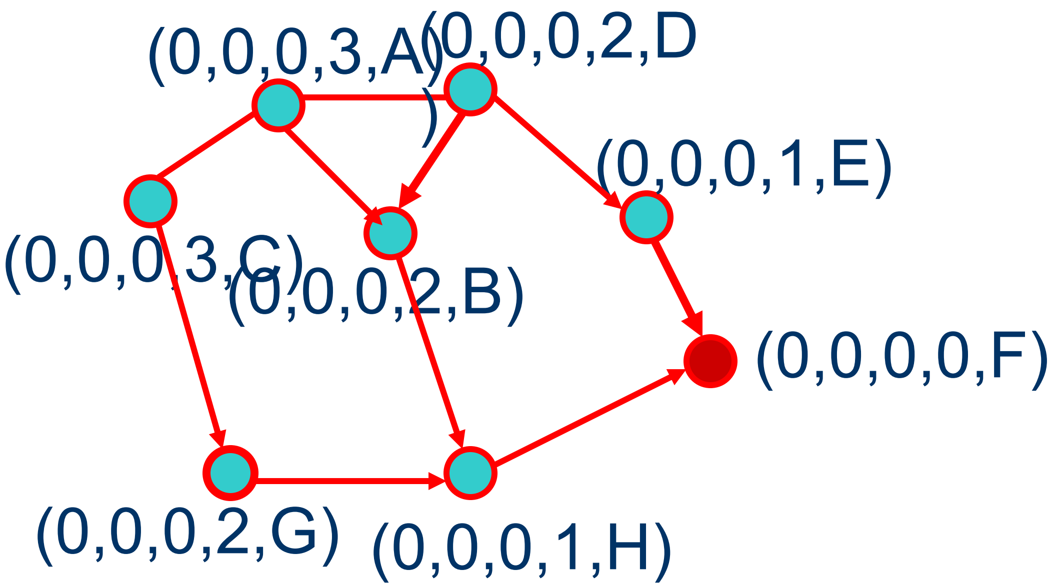
- Nodes that have QRY ($RR_i = 0$)
- Nodes that have UPD

An Example



- Nodes that have QRY ($RR_i = 0$)
- Nodes that have UPD

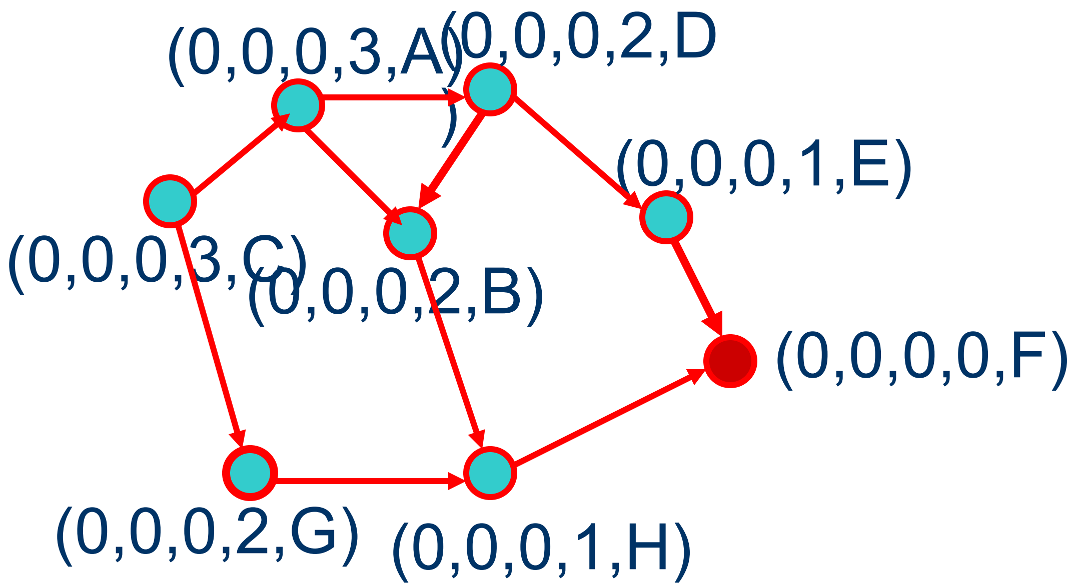
An Example



Nodes that have QRY ($RR_i = 0$)

Nodes that have UPD

An Example



- Nodes that have QRY ($RR_i = 0$)
- Nodes that have UPD