

Temporally-Ordered Routing Algorithm (TORA)

System Working Group
Fall 2001

Jeff Dobbelaere

From a paper by Park and Corson, A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks, *IEEE Infocomm*, 1997

Network Considerations for Routing Algorithms

- ▶ A collection of mobile wireless routers
- ▶ Connection status dependent on: node position, transmission power, antenna patterns, and interference levels
- ▶ Result in an unpredictable, dynamic network
- ▶ Link congestion

Desirable Attributes of a Routing Algorithm

- ▶ Executes distributedly
- ▶ Provides loop-free routes
- ▶ Provides multiple routes (to alleviate congestion)
- ▶ Establishes routes quickly (so they may be used before the topology changes)
- ▶ Minimizes communication overhead by localizing algorithmic reaction to topological changes when possible (to conserve available bandwidth and increase scalability)

TORA Attributes

- ▶ On demand, source initiated routing
- ▶ Distributed in that nodes only maintain one hop knowledge
- ▶ Provides multiple routes to alleviate congestion
- ▶ Creates loop free routes
- ▶ Handles partitions by erasing invalid routes

Note: TORA favors all of these qualities over locating shortest-path routes.

System Assumptions & Notations

- ▶ Network modeled as a graph: $G = (N, L)$
 - N is a finite set of nodes
 - L is the set of communication links
- ▶ Each node has a unique identifier (ID)
- ▶ Communication links are bidirectional, FIFO, and categorized as:
 - (1) undirected
 - (2) directed from node i to node j ; i is *upstream* of j and j is *downstream* of i
 - (3) directed from node j to node i ; j is *upstream* of i and i is *downstream* of j
- ▶ Each node i has a set of neighbors $N_i \in N$ defined as nodes j such that $(i, j) \in L$ (determined by a link-level protocol)
- ▶ Nodes make use of omnidirectional antennas

Algorithm Overview

Route Creation: Creating routes consists of establishing a sequence of directed links from the source to the destination. This is done by forming a destination oriented *DAG*; the destination node is the sink of the graph.

Route Maintenance: Reaction to topology changes in order to reestablish routes within a finite time.

Route Erasure: When a partition is detected in the network, all invalid routes must be removed from the network. This is done by making directed routes undirected.

General Reversal Algorithms

Each node i has a height $h_i = (\alpha_i, i)$ such that the nodes may be totally ordered.

Node j is considered downstream of node i if $h_i > h_j$. Link (i, j) is directed *from* i to j .

Node i is a local minimum when it has no outgoing links, i.e. when $h_i < h_j$ for all $j \in N_i$.

Full Reversal: Node i raises its height h_i so it is a local *maximum* by setting $\alpha_i = \max\{\alpha_j | j \in N_i\} + 1$

It then broadcasts this value to its neighbors which causes all links (i, j) for all $j \in N_i$ to reverse.

Partial Reversal: Node i raises its height h_i so it is greater than some of its neighbors, but not all.

The reversal class of algorithms has been shown to provide loop-free destination oriented DAGs in a finite number of iterations.

Node Height in TORA

Because of partition detection, node height is much more complex than regular partial reversal techniques.

Each node $i \in N$ has a height represented as a quintuple: $H_i = (\tau_i, oid_i, r_i, \delta_i, i)$

- τ_i : a time tag indicating the “time” of link failure represents the reference level.
- oid_i : originator-id; ID of node that defined the reference level
- r_i : 1 bit used to divide each reference level into 2 sub-levels
- δ_i : integer used to order nodes with respect to a unique reference level
- i : unique identifier of the node

The height of each node (except for the destination) is initially set to NULL: $H_i = (-, -, -, -, i)$.

Each node maintains a height array containing the heights of neighbors.

Route Creation

QRY packet: contains the destination-ID (did) for which the algorithm is running.

UPD packet: contains the did and the height of the node i that is broadcasting the packet, H_i

Each node maintains:

- *route-required* flag RR_i initially unset
- the time the last UPD packet was broadcast
- the time at which each link $(i, j) \in L$ for $j \in N_i$ came up.

When a node i with no directed links and an un-set RR_i requires a route, it broadcasts a QRY packet and sets RR_i .

When a node i receives a QRY message, it has four options:

1. if node i has no downstream links and RR_i is unset, it rebroadcasts the QRY message and sets RR_i .
2. if node i has no downstream links and RR_i is set, it discards the QRY packet
3. if node i has at least one downstream link and its height is NULL, it sets its height to $H_i = \min\{H_j | j \in N_i\} + \{0, 0, 0, 1, 0\}$ and broadcasts an UPD packet
4. if node i has at least one downstream link and its height is non-NULL, and if a UPD packet has been broadcast since the link over which the QRY packet was received became active, it discards the QRY packet. Otherwise it broadcasts an UPD packet.

Also, if RR_i is set when a link becomes active, it broadcasts a QRY packet.

Receipt of an UPD packet

When node i receives an UPD packet from a neighbor j , i updates $HN_{i,j}$ to reflect the height of node j received in the UPD message.

There are now two options:

1. if RR_i is set (implying the height of node i is NULL), node i sets $H_i = \min\{H_j | j \in N_i\} + \{0, 0, 0, 1, 0\}$, updates the links in LS_i , unsets RR_i , and broadcasts a UPD packet with the new information.
2. if RR_i is unset, node i updates the links in LS_i (possible to lose all downstream links)

The Following examples are from the paper.

Example 1 - Route Creation

Circle around a node indicates RR_i is true.

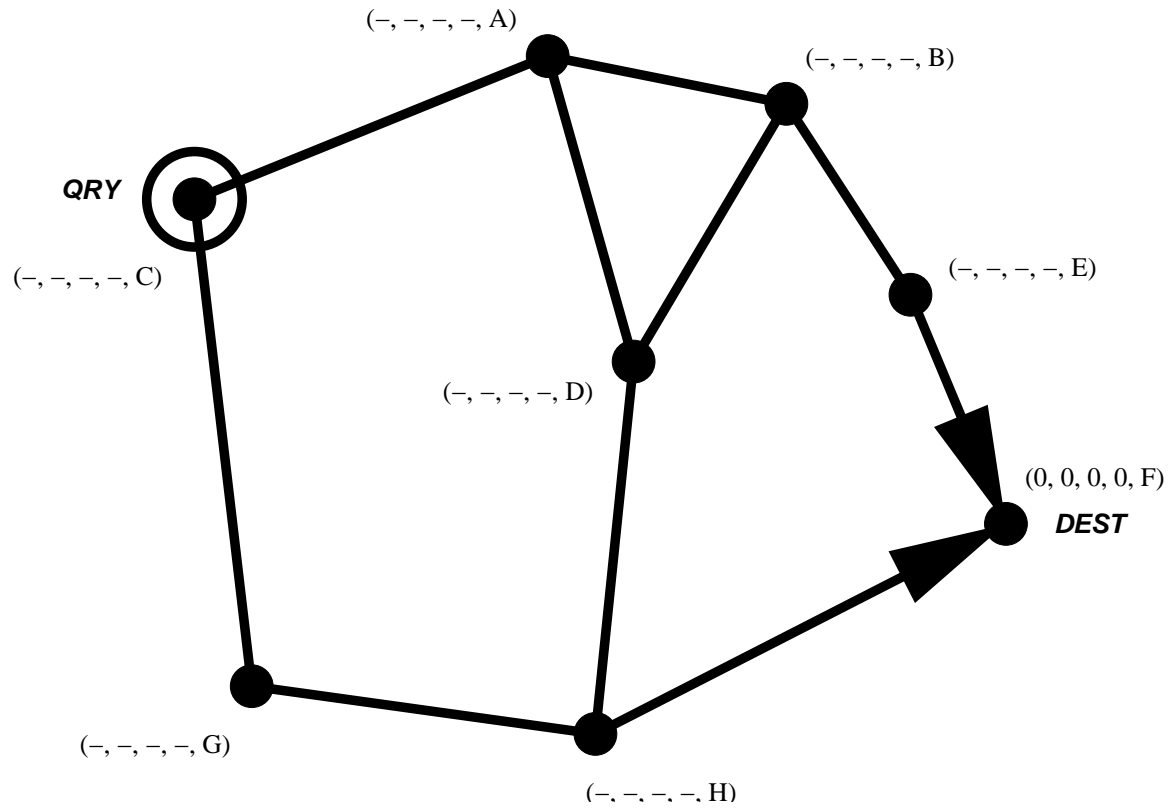


Figure 1: Node C requires a route to node F. It therefore broadcasts a QRY packet.

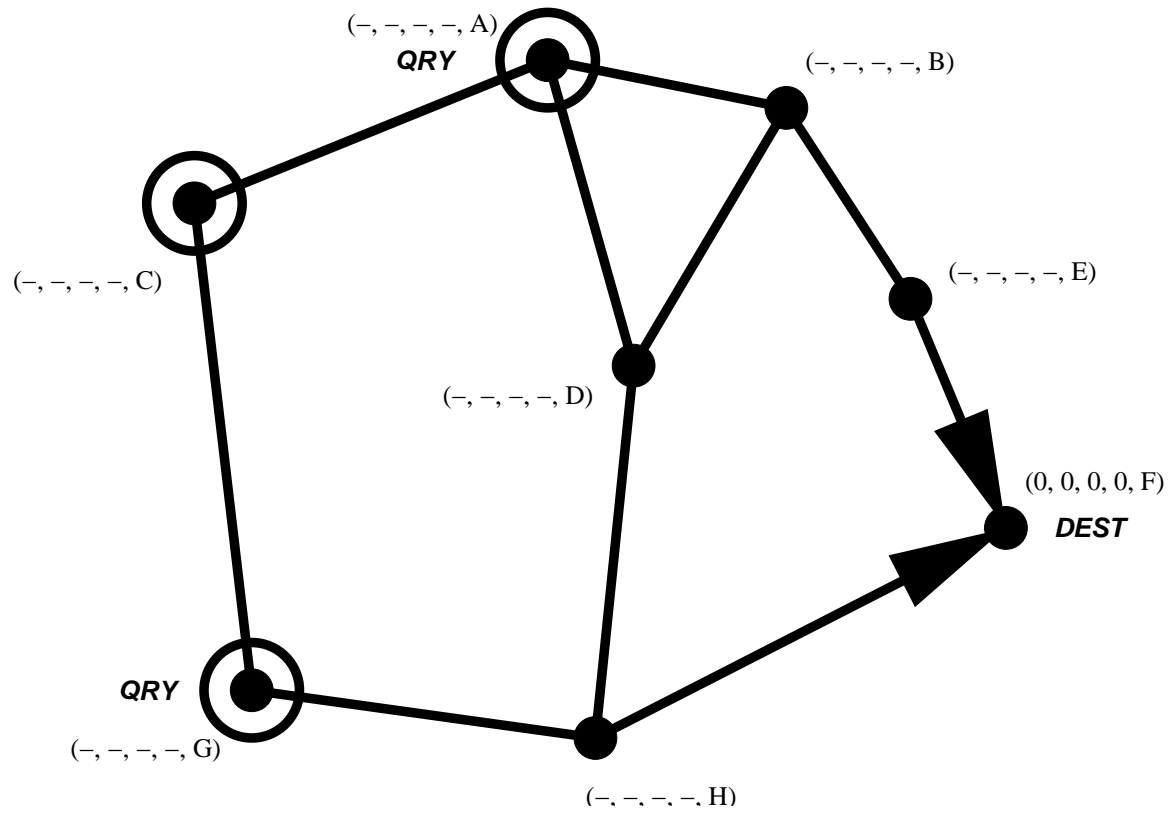


Figure 2: Node A and Node G propagate the QRY packet.

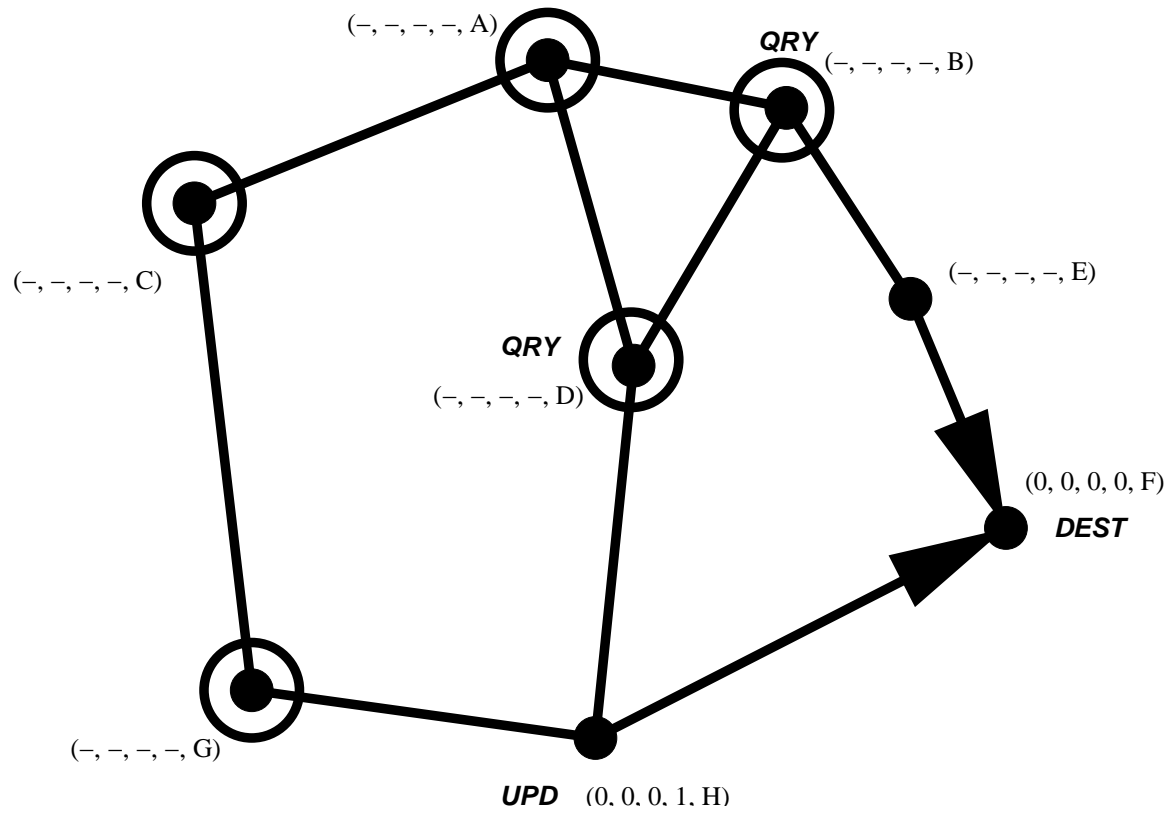


Figure 3: Nodes B and D propagate the QRY packet. Node H generates a UPD packet.

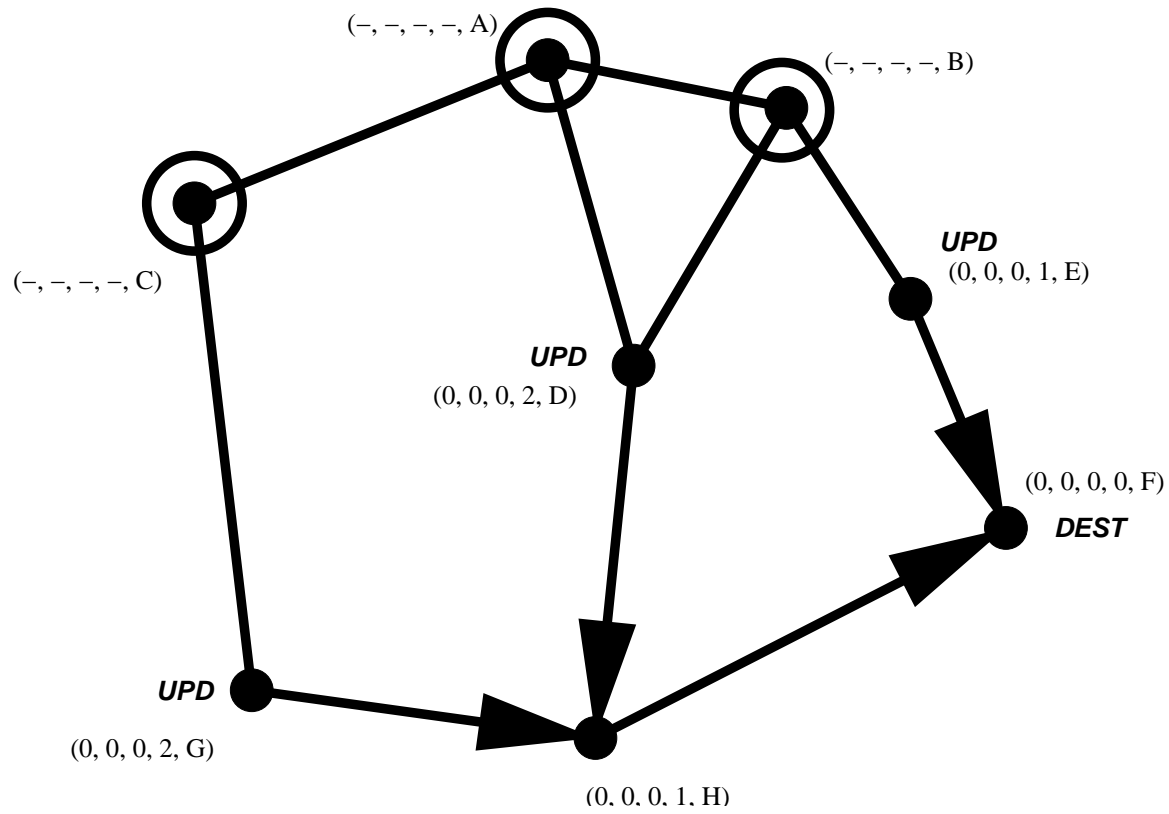


Figure 4: Nodes D and G propagate the UPD packet while node E generates a UPD packet.

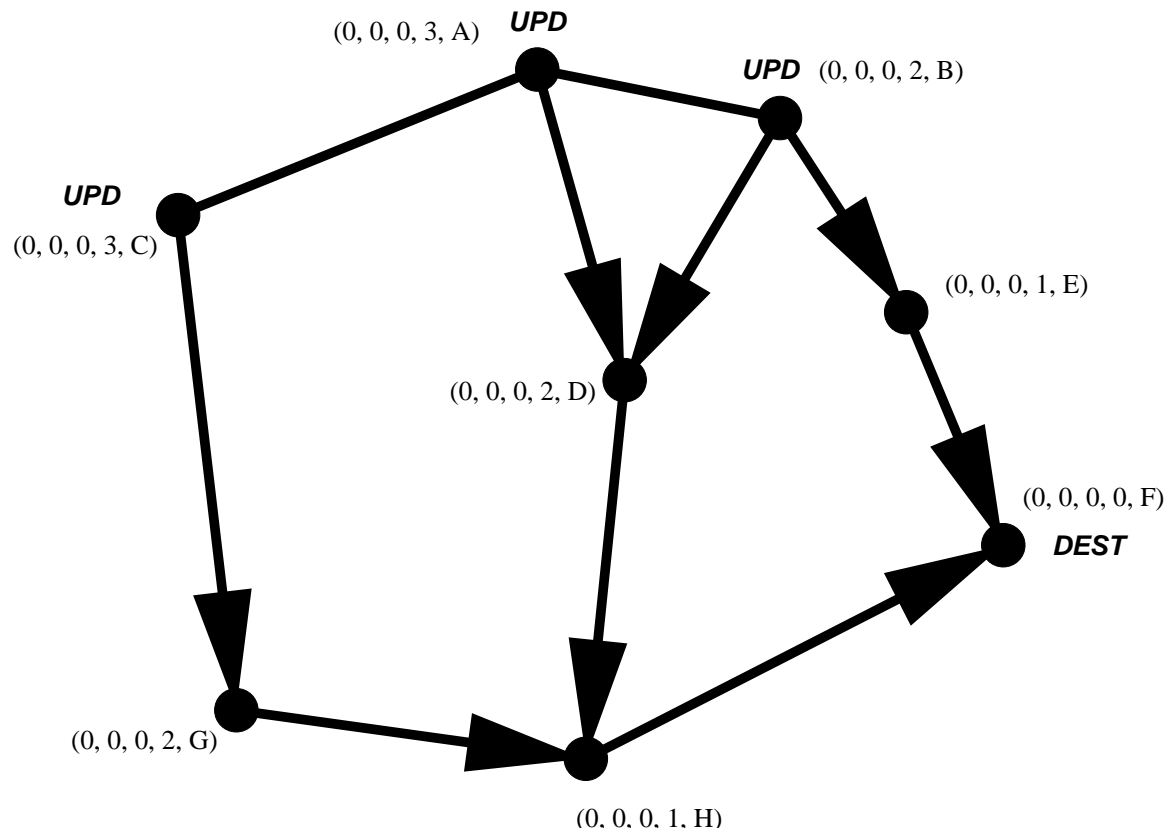


Figure 5: Nodes A, B, and C propagate the UPD packet.

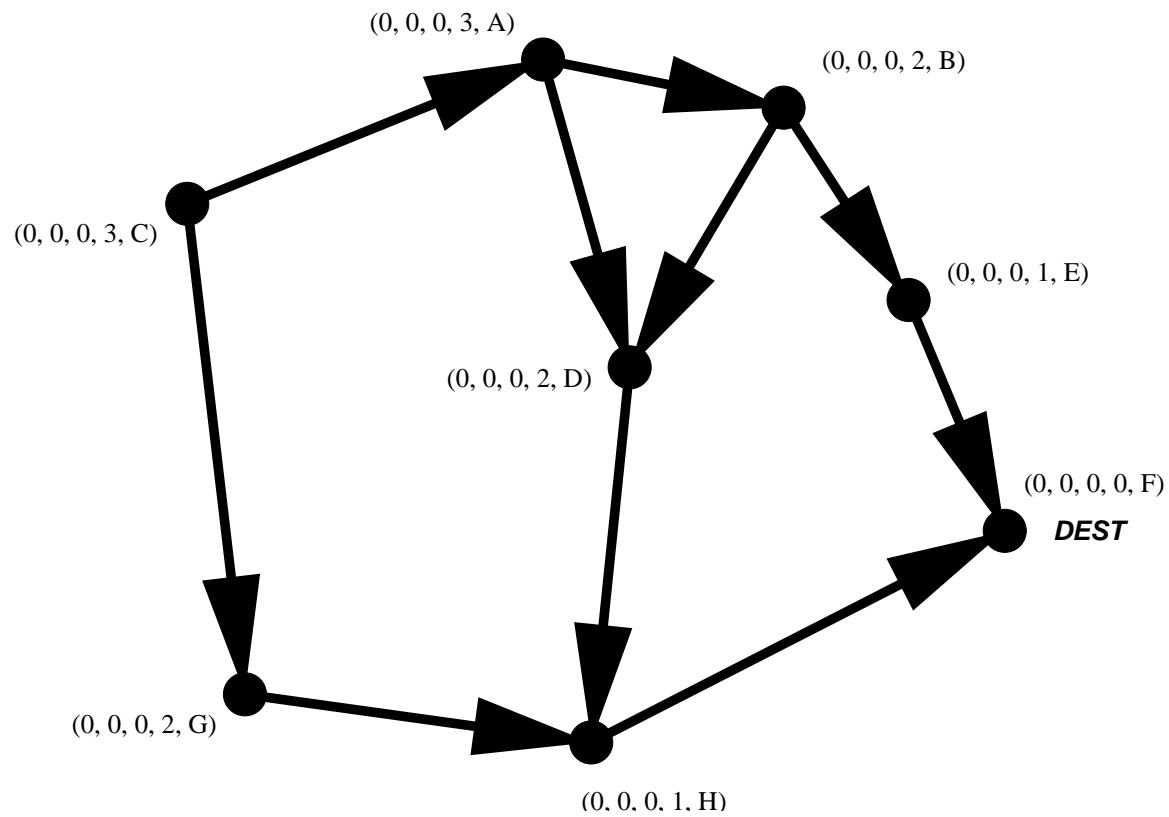


Figure 6: Route creation has completed.

Route Maintenance

A node i is said to have no downstream links if $H_i < HN_{i,j}$ for all non-NULL neighbors $j \in N_i$. This causes one of five possible reactions which modifies H_i .

1. **Generate:** Node i has no downstream links due to link failure:

$$(\tau_i, oid_i, r_i) = (t, i, 0), \text{ where } t \text{ is the time of the failure}$$

$$(\delta_i, i) = (0, i)$$

Node i defines a new *reference level*. If i has no upstream neighbors, it sets $H_i = NULL$.

2. **Propagate:** Node i has no downstream links due to link reversal following receipt of an UPD packet and the ordered sets (τ_j, oid_j, r_j) are not equal for all $j \in N_i$:

$$(\tau_i, oid_i, r_i) = \max \left\{ \left(\tau_j, oid_j, r_j \right) \mid j \in N_i \right\}$$

$$(\delta_i, i) = \left(\min \left\{ \delta_j \mid j \in N_i \text{ with } (\tau_j, oid_j, r_j) = \max \left\{ \left(\tau_j, oid_j, r_j \right) \right\} \right\} - 1, i \right)$$

Node i propagates the reference level of its highest neighbor and chooses a reference level lower than all neighbors of that reference level.

3. **Reflect:** Node i has no downstream links due to link reversal following receipt of an UPD packet and the sets (τ_j, oid_j, r_j) are equal with $r_j = 0$ for all $j \in N_i$:

$$(\tau_i, oid_i, r_i) = (\tau_j, oid_j, 1)$$

$$(\delta_i, i) = (0, i)$$

Node i reflects back the reference level by setting the r bit.

4. **Detect:** Node i has no downstream links due to link reversal following receipt of an UPD packet and the sets (τ_j, oid_j, r_j) are equal with $r_j = 1$ for all $j \in N_i$:

$$(\tau_i, oid_i, r_i) = (-, -, -)$$

$$(\delta_i, i) = (-, i)$$

Node i has detected a partition and begins route erasure.

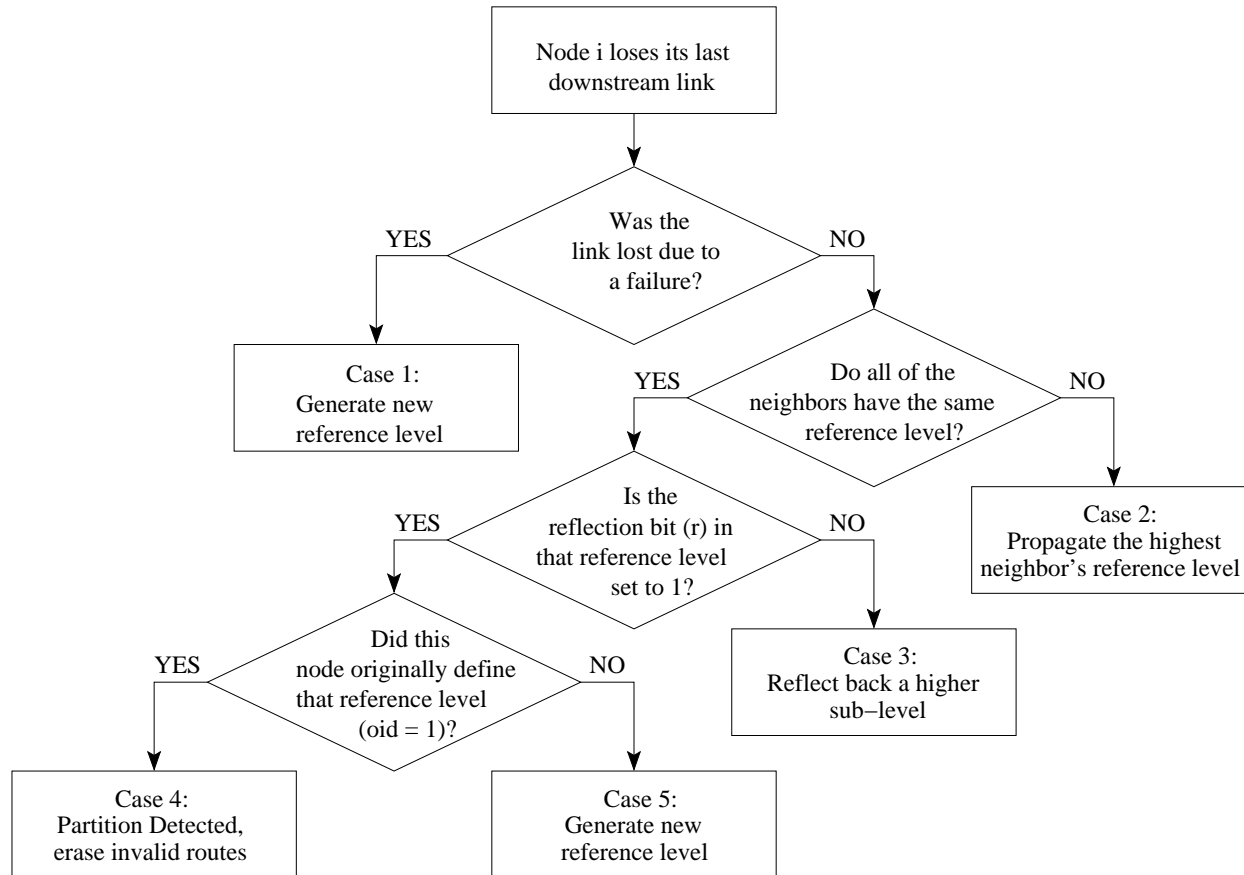
5. **Generate:** Node i has no downstream links due to link reversal following receipt of an UPD packet and the sets (τ_j, oid_j, r_j) are equal with $r_j = 1$ for all $j \in N_i$ and $oid_j \neq i$:

$$(\tau_i, oid_i, r_i) = (t, i, 0), \text{ where } t \text{ is the time of link failure}$$

$$(\delta_i, i) = (0, i)$$

Node i experienced a link failure between the time it propagated a reference level and the reflected higher sub-level returned from all neighbors. This link failure required no reaction.

Route Maintenance - Decision Tree



Example 2 - Route Maintenance

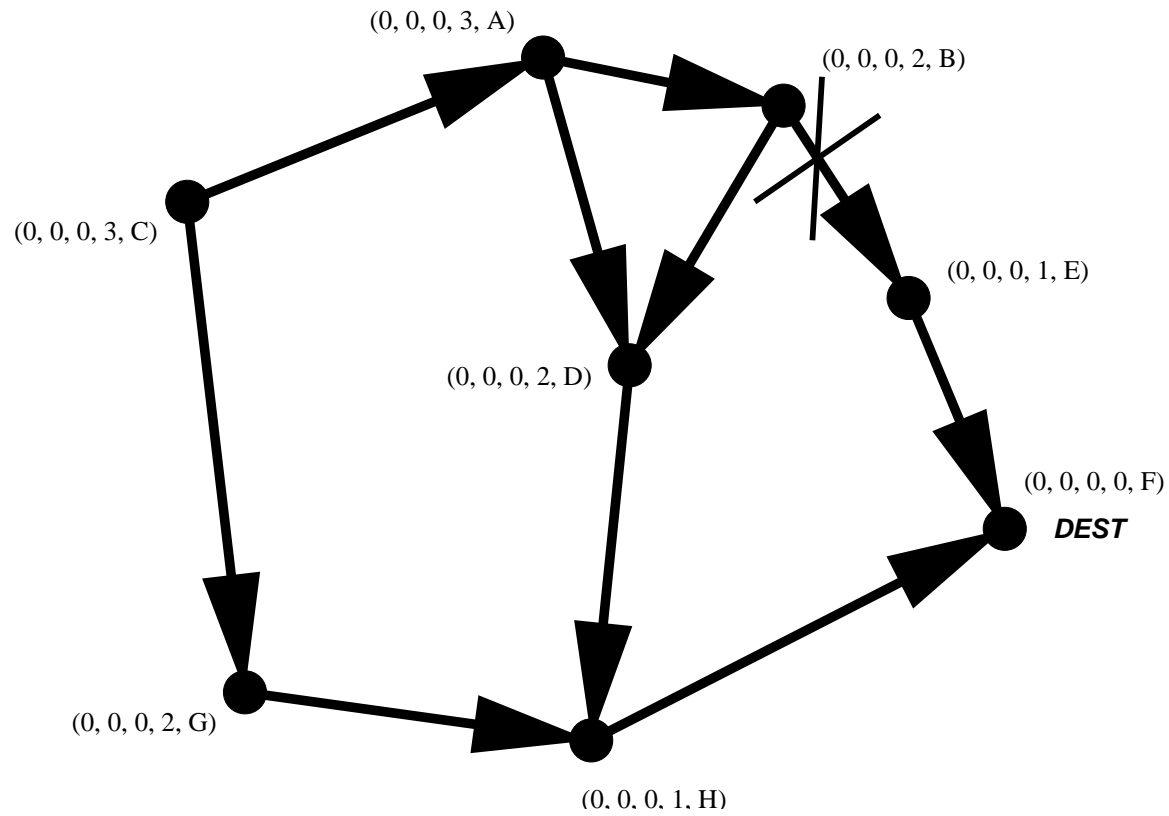


Figure 7: The link between nodes B and E fails.

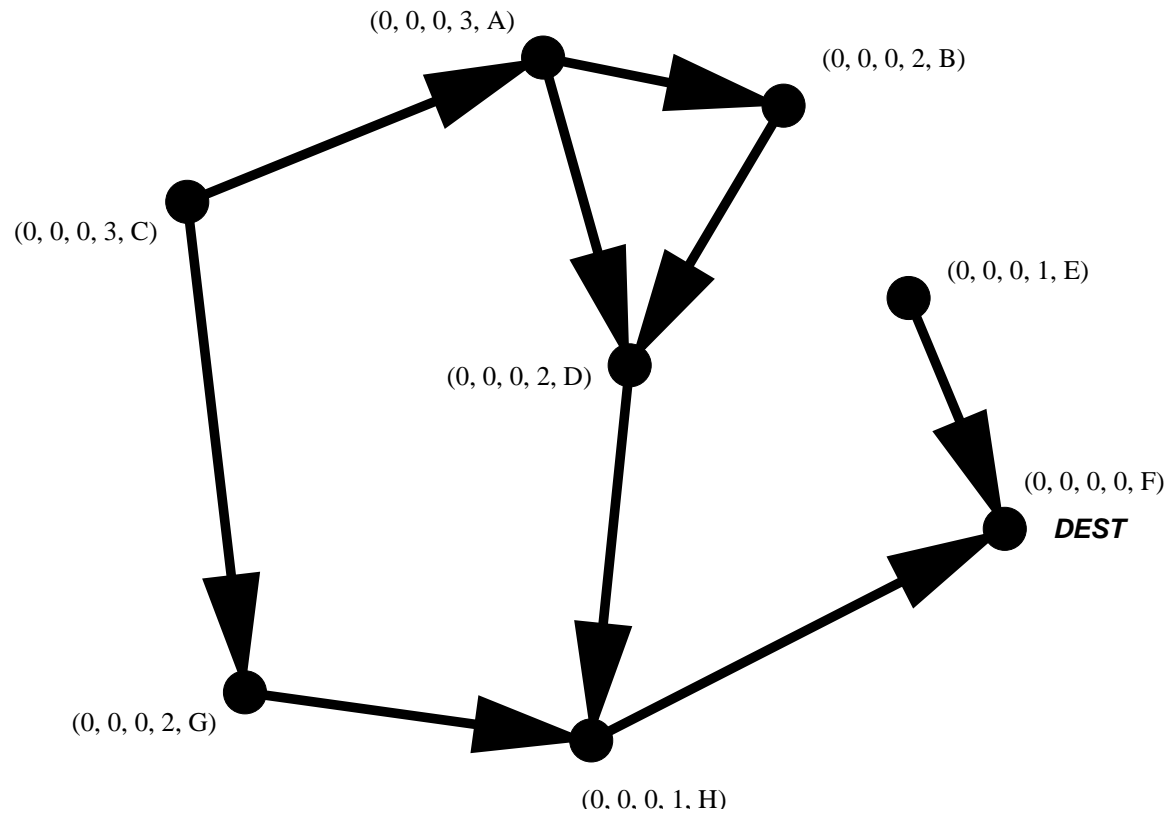


Figure 8: No route maintenance is necessary because all nodes have at least one outgoing link (except for DEST) which means each node has a route to DEST.

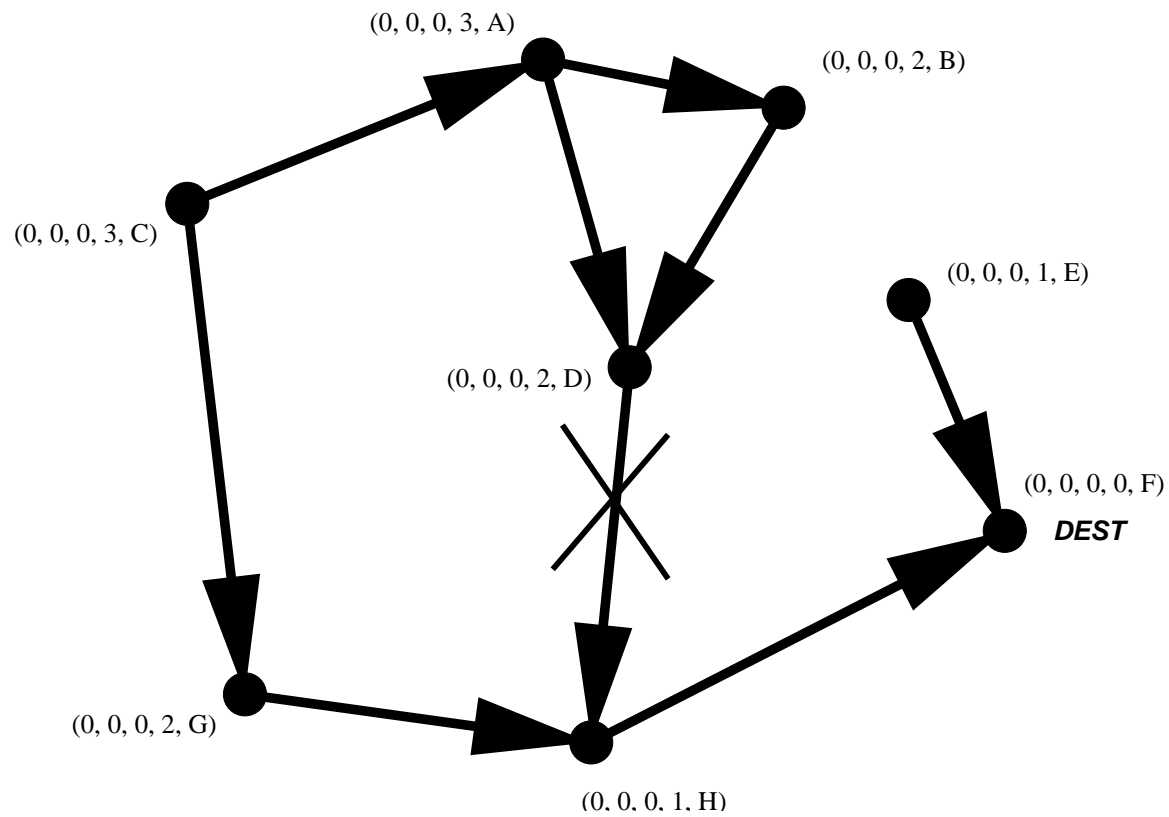


Figure 9: Link (D, H) fails.

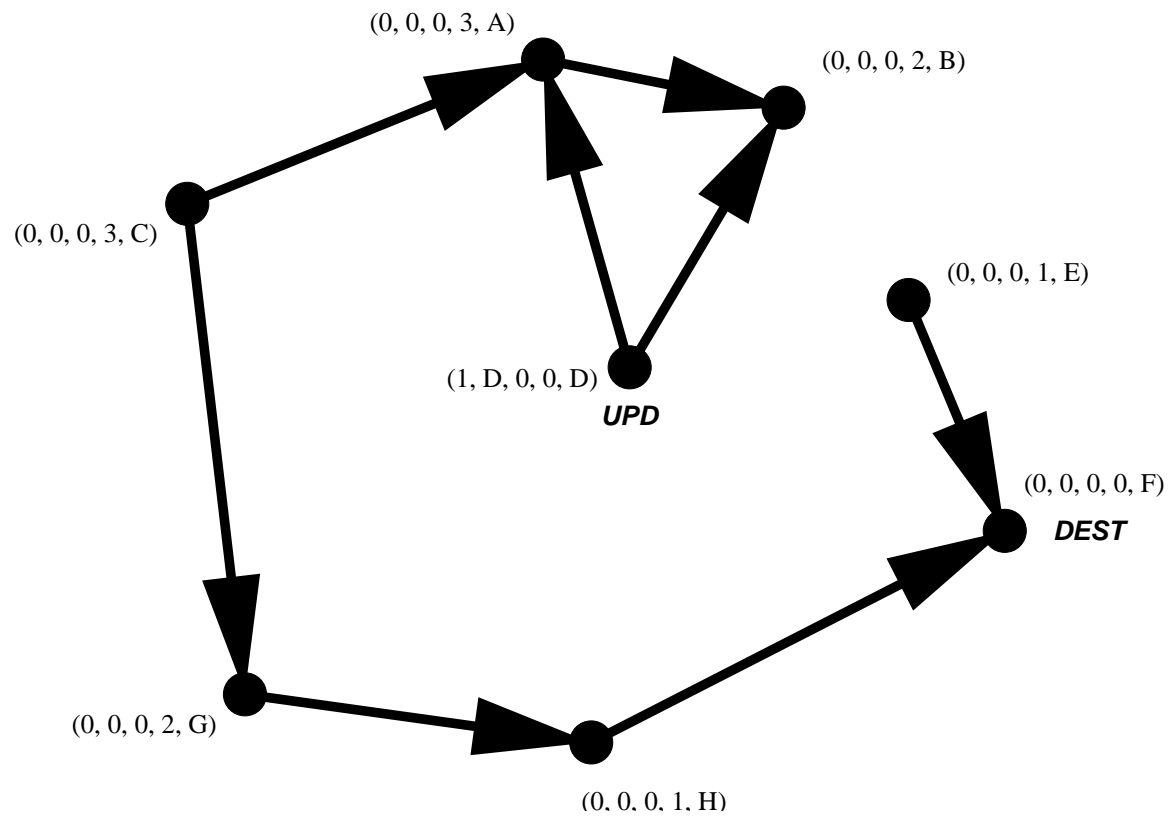


Figure 10: Node D defines a new reference level.

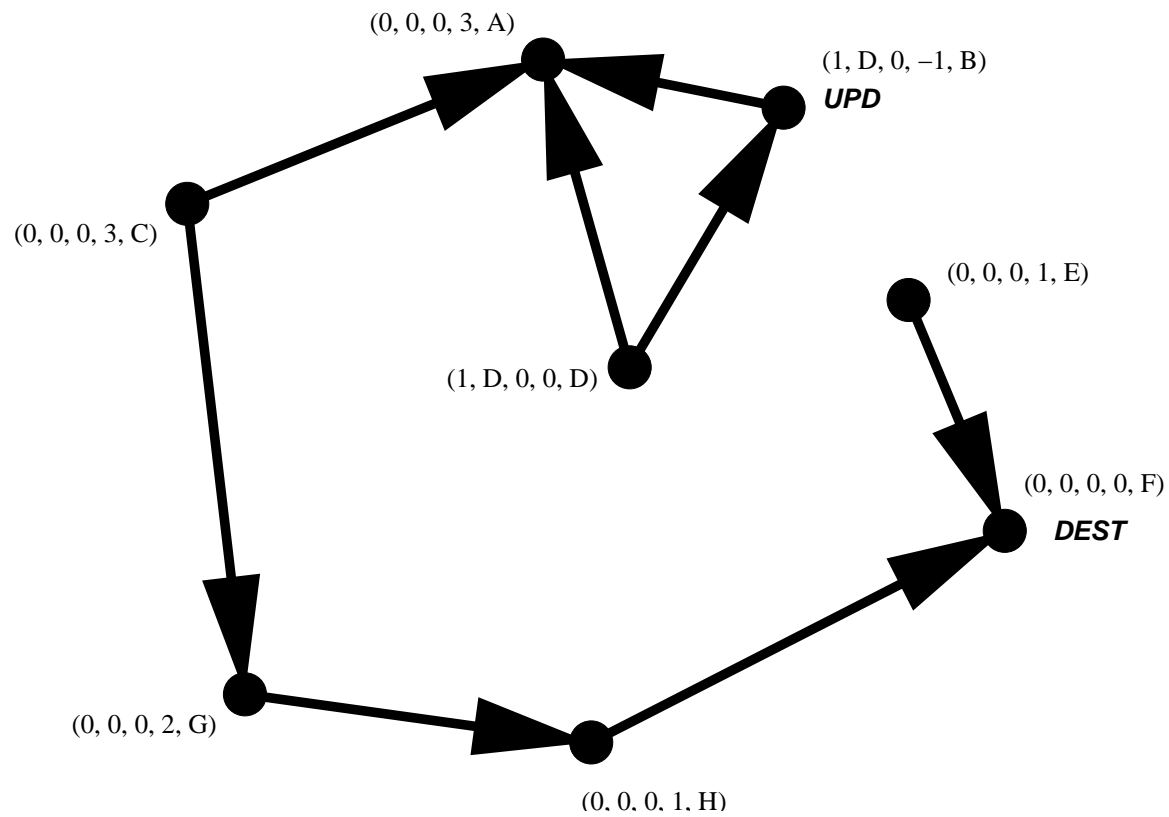


Figure 11: Node B propagates the reference level defined by D.

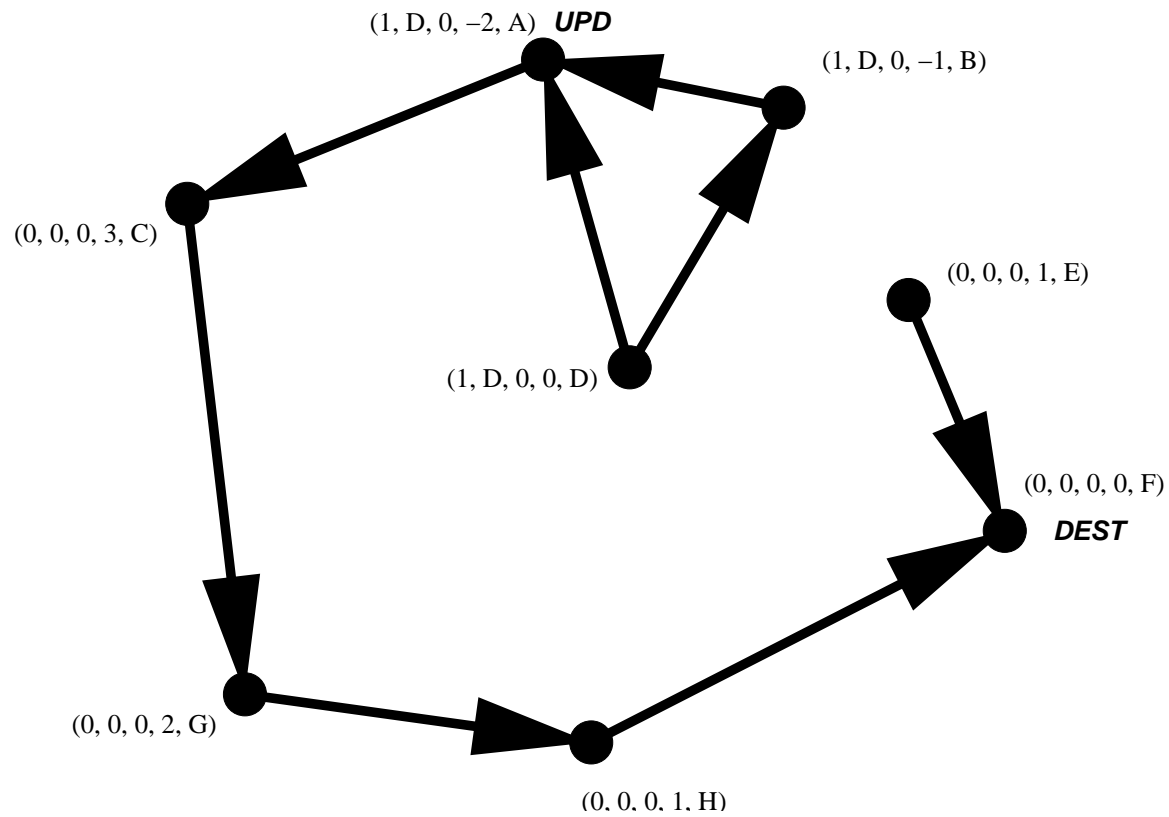


Figure 12: Node A propagates the reference level defined by D.

Example 2 - Route Maintenance

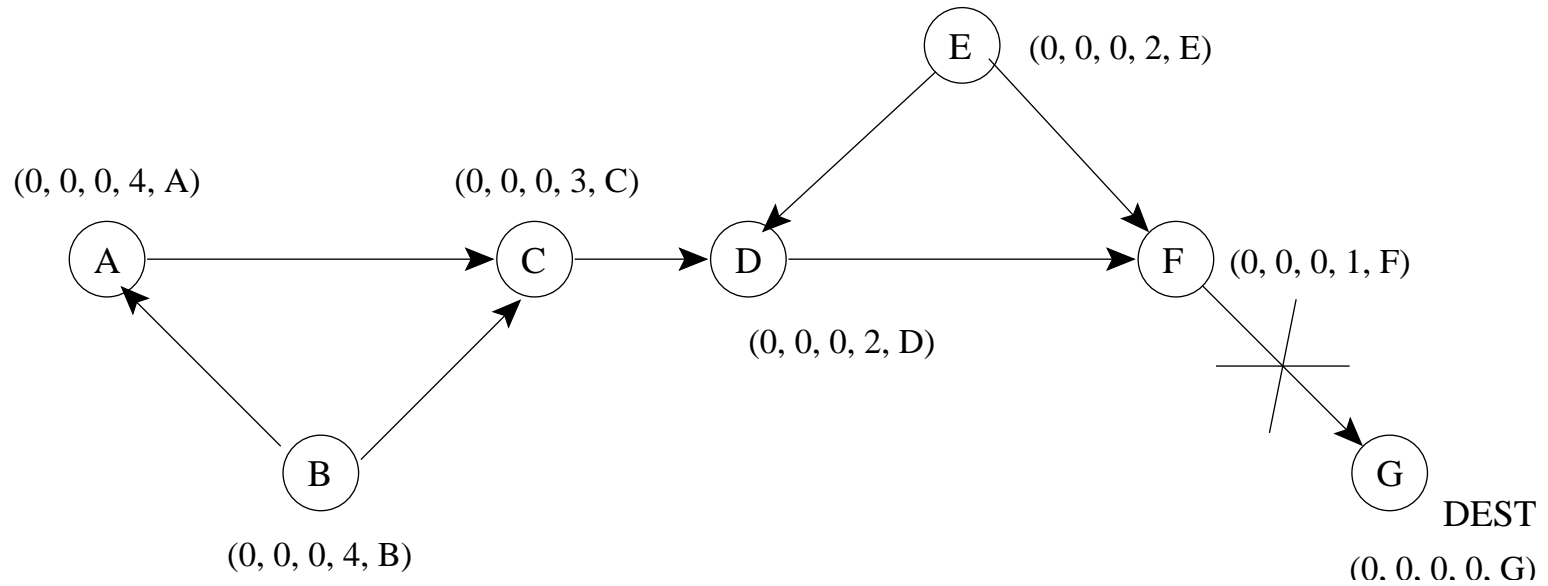


Figure 13: The link between F and G fails, causing a partition.

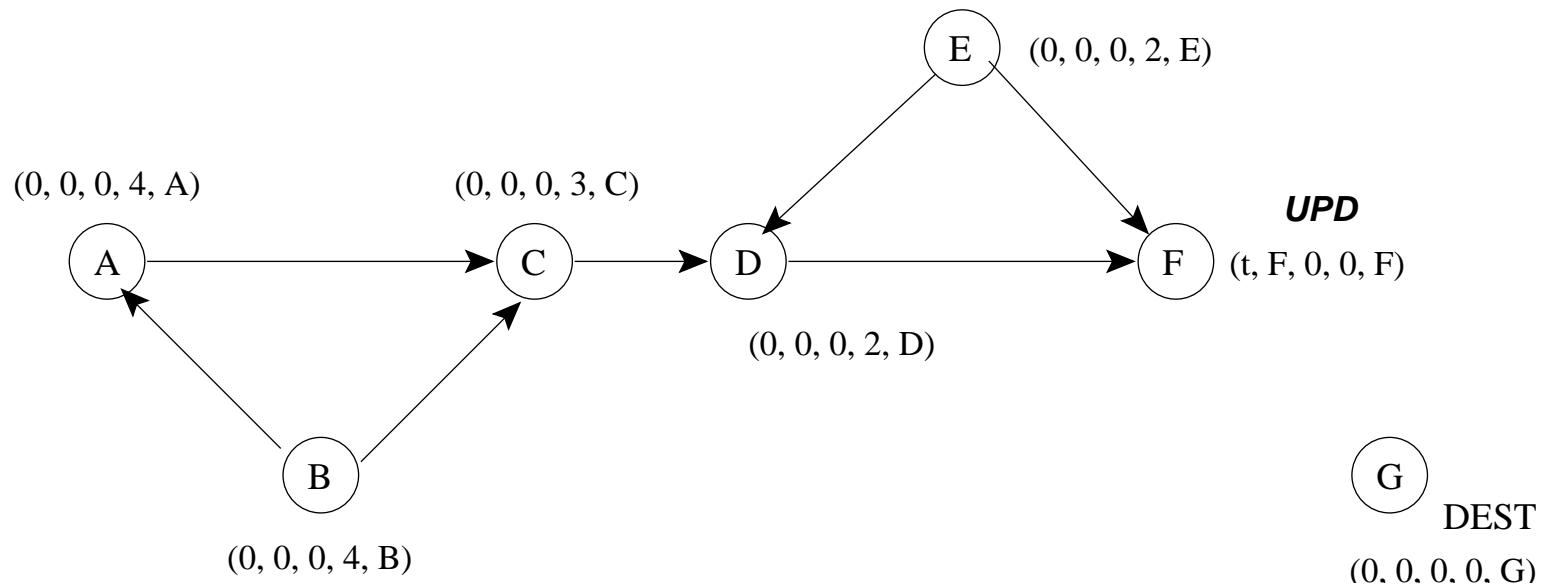


Figure 14: Node F defines a new reference level and sends an UPD message.

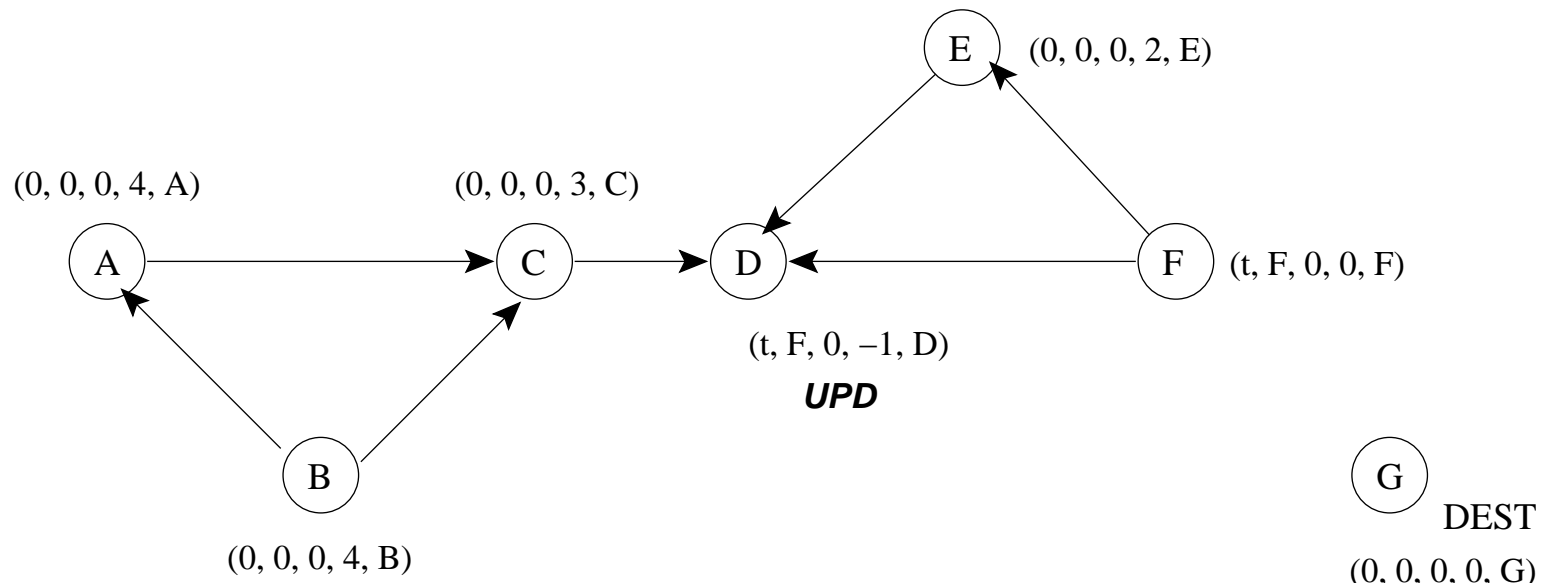


Figure 15: Both links to node F reverse. Node D updates height and propagates the reference level with an UPD message.

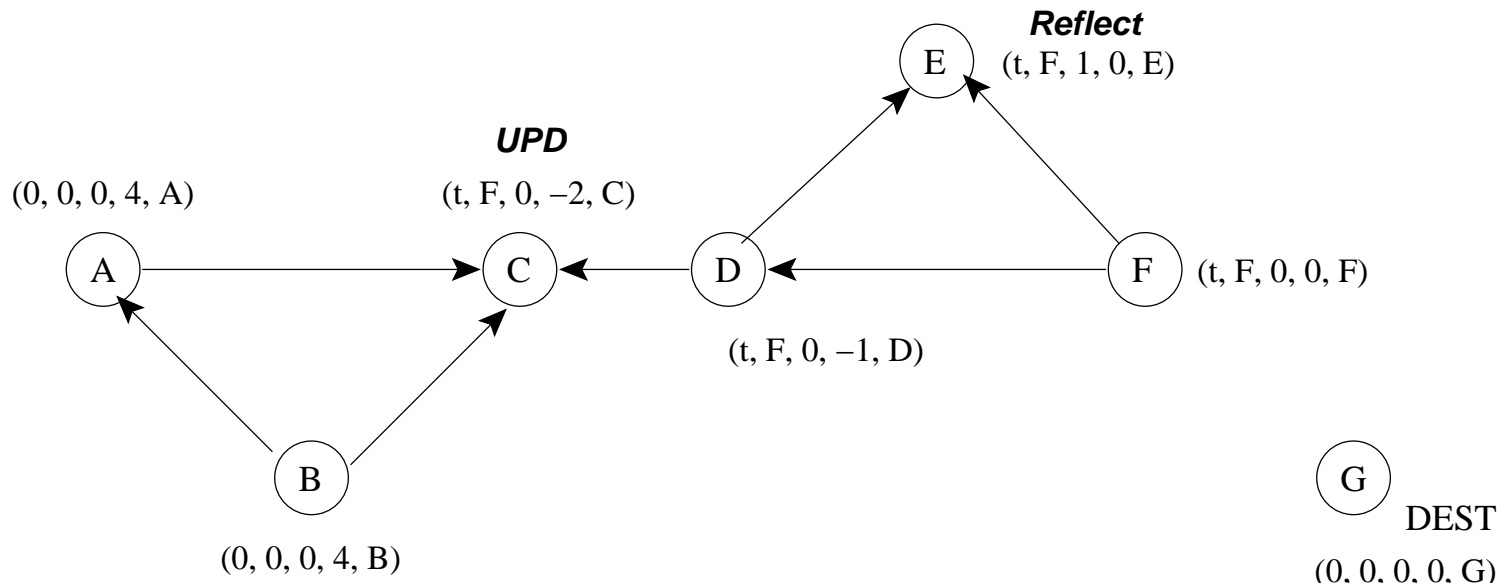


Figure 16: Node E reflects the reference level. Node C updates its height and propagates the reference level.

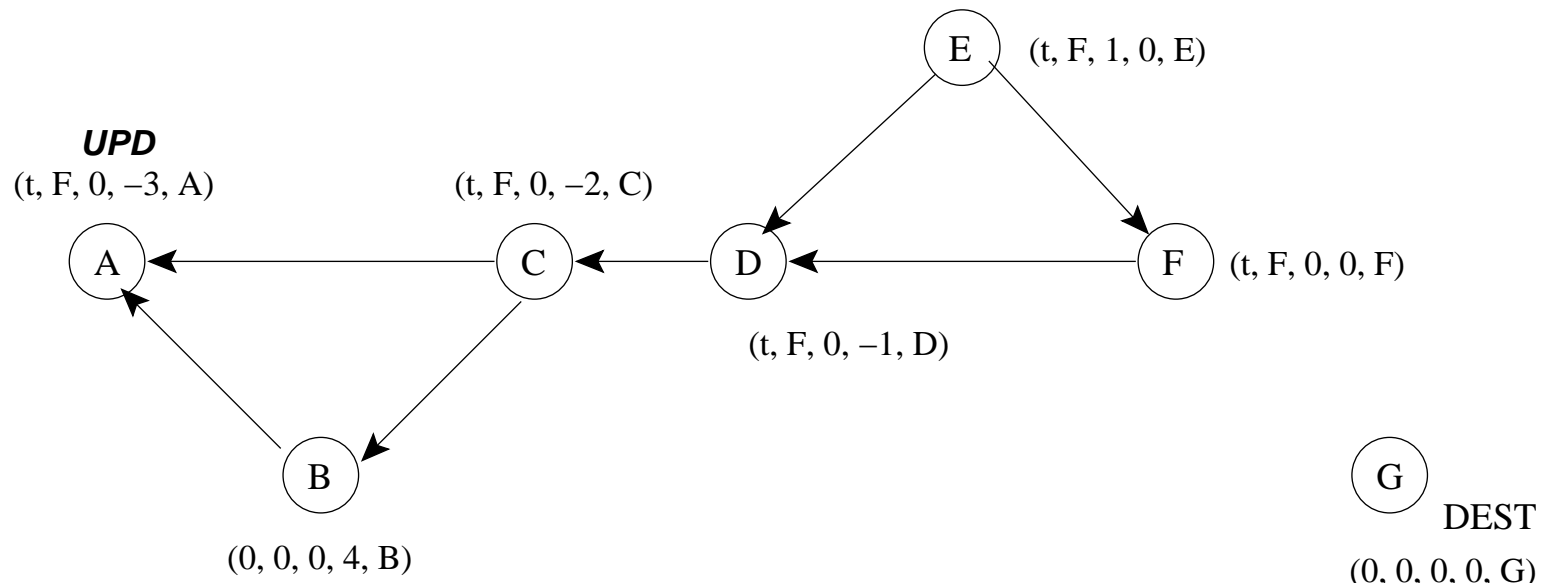


Figure 17: Node A changes its height and propagates the reference level.

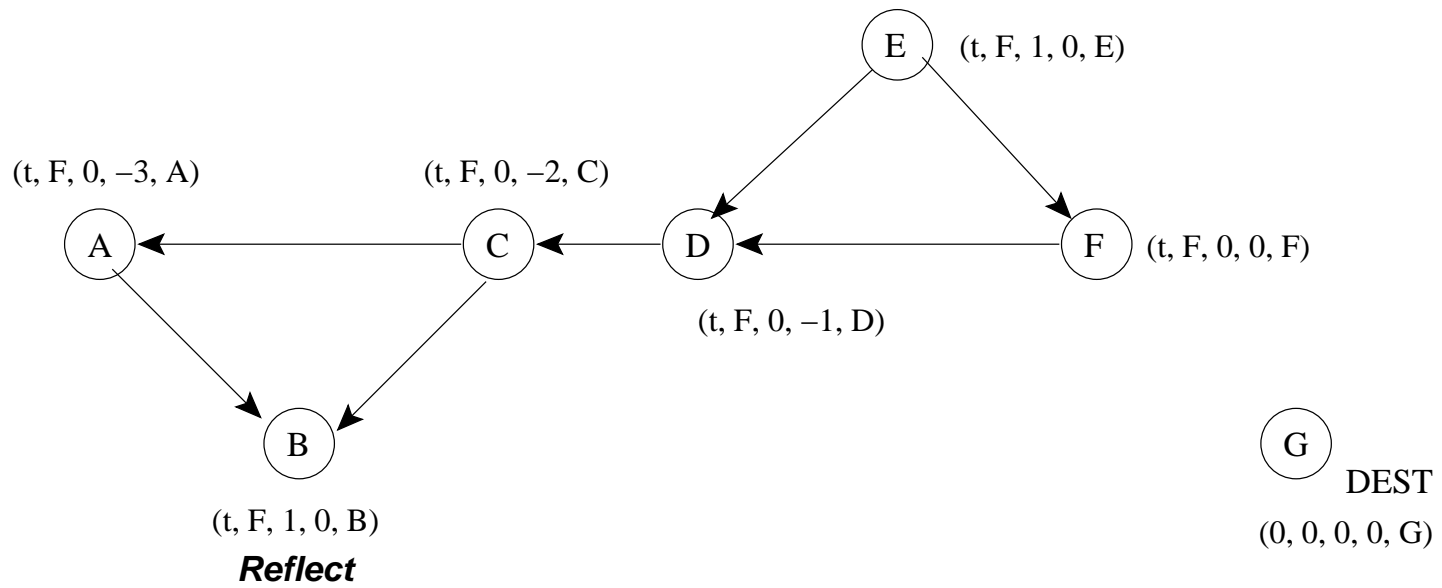


Figure 18: Node B reflects the reference level.

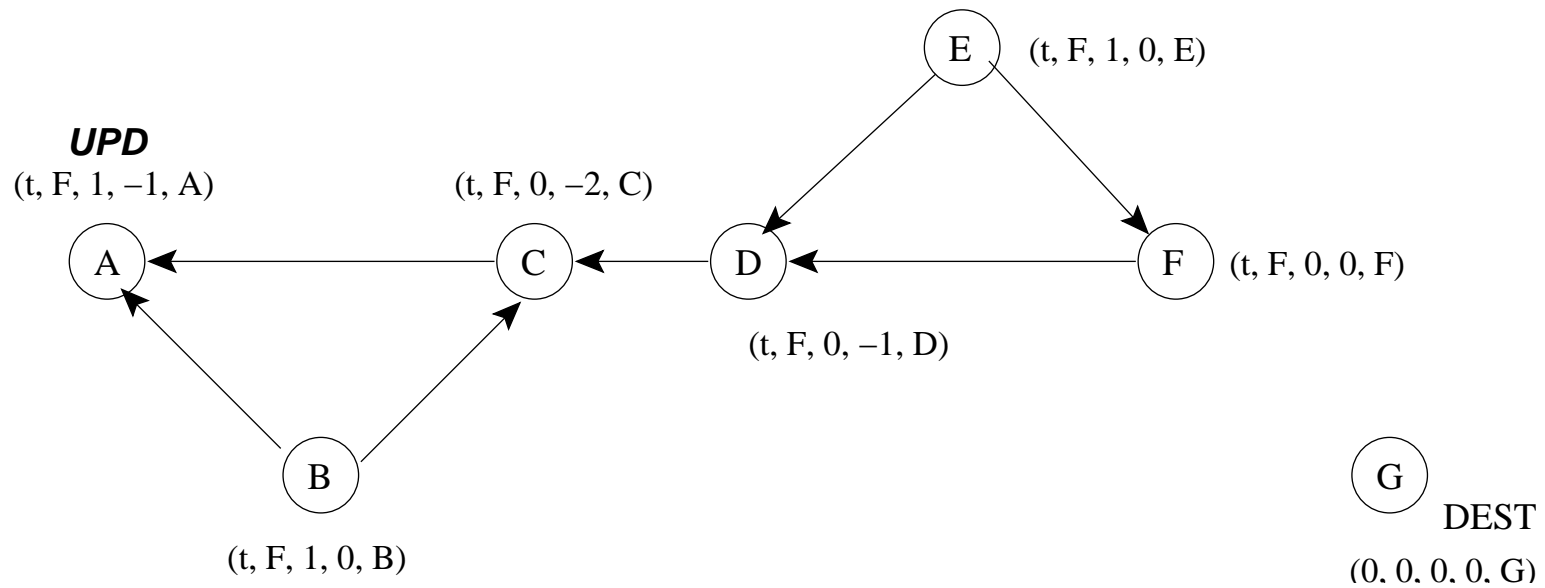


Figure 19: Node A updates its height and propagates the reference level.

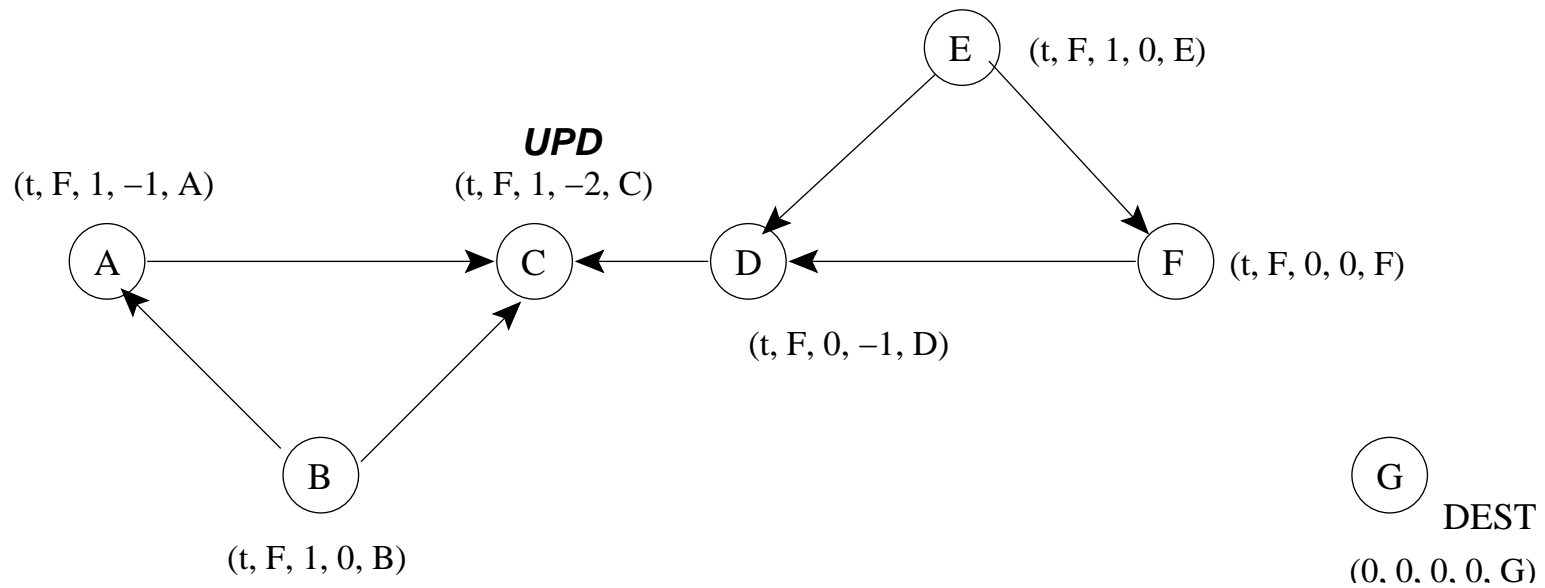


Figure 20: Node C updates its height and propagates the reference level.

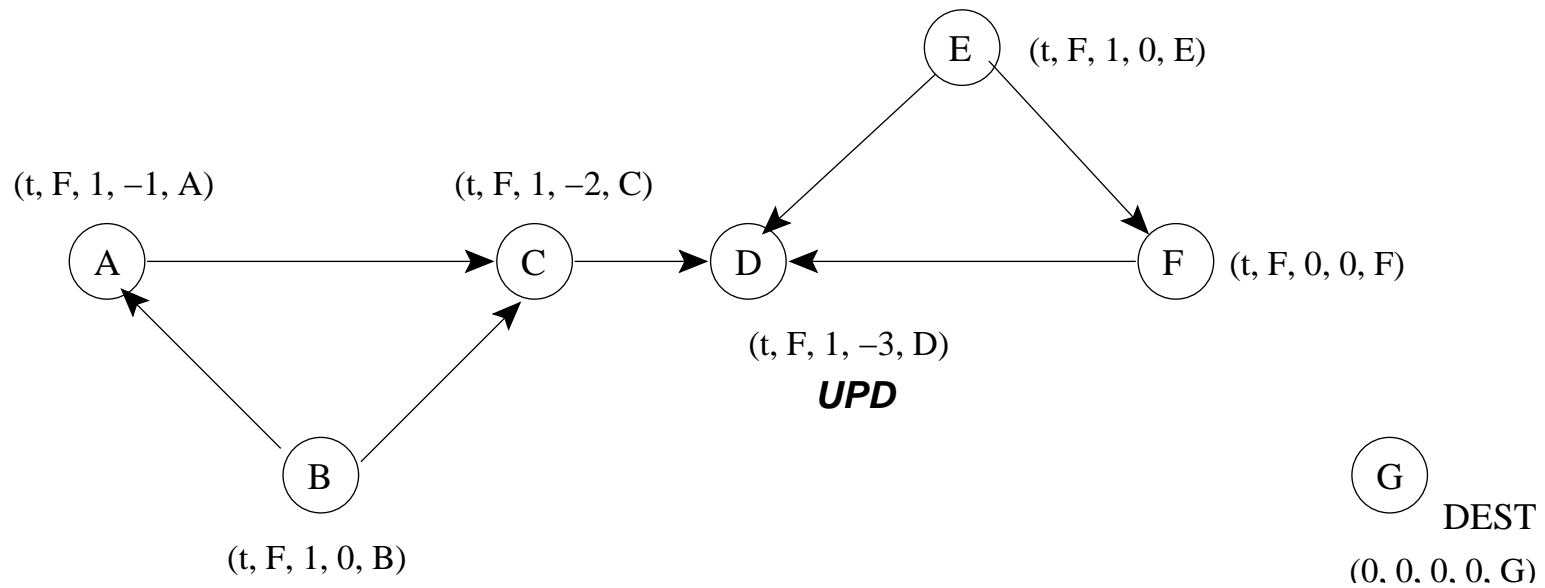


Figure 21: Node D updates its height and propagates the reference level.

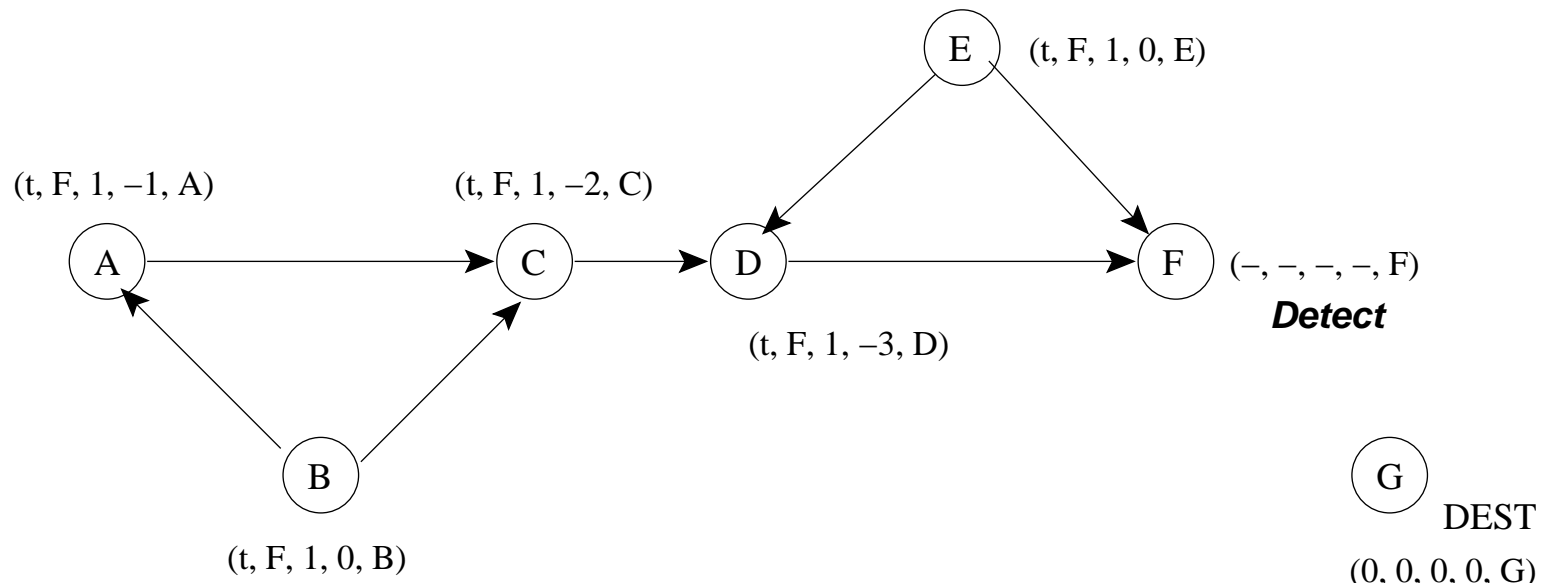


Figure 22: Node F detects the partition and begins the route erasing protocol.