1 Introduction

Today, Dr. Soma started the class by talking about the new topic, which is "Routing in a Distributed Systems with Dynamic Links". With this topic, we will focus on a graph system where each node has specific properties and is able to communicate with others by using message passing protocol.

2 Routing in a Distributed Systems with Dynamic Links

We consider network where communication topology may change dynamically, for instance, mobile ad hoc network, etc.

2.1 Leading Idea

These are leading questions/ideas for routing with dynamic links:

1. How do neighbor vertices know the direction of the link between them?

2. The nodes need to communicate through messages, which can possibly be delayed or lost.

3. If the network changes dynamically, so, topology may no longer be connect, how can we handle this issue?

4. Mobile Ad Hoc network, which links can be created and lost.

“Neighborhood Discovery” protocol is assumed to figure out which nodes are within communication range and signals this to endpoints.

There are two properties that we are considering for this approach (given finite number of topology change):

1. After the last topology change, every node that is in same connected component as $D$, the sink node, eventually has a path to $D$

2. After the last topology change, every node that is not in the same connected component as $D$ stops taking step in the routing algorithm.
2.2 Overview of Temporally Ordered Routing Algorithm (TORA)

This was proposed by V. Park and S. Carson in 1997. They used Increasing Vertex Label (IVL) from Gafni and Bertsekas. They stated that each node keeps track of what it thinks is the current height of each of its neighbors. Since this exchanges through messages which is delayed, thus, it may be out of date.

2.3 The Initialization Procedure of TORA

Below are the initialization procedure of TORA:

1. Each node sends a query message to find route to \( D \), which is broadcasting and propagated, until it reaches a node with a height.

2. In response, the node sends its height. All recipients use this to compute own height.

We assume that topology does not change during initialization. At the end of initialization, all nodes have a height of its own and neighbors.

2.4 Node Properties

This is the properties that belong to each node in the network:

\[ < \mathcal{T}, oid, rbit, \delta, id > \]

The description of those properties are:

1. \( \mathcal{T} \) - \( \{0 \mid \text{timestamp of reference level} \} \)
2. \( oid \) - \( \{0 \mid \text{originating processor id} \} \)
3. \( rbit \) - \( \{0 \mid 1\} \)
4. \( \delta \) - This is for the same purpose as \( a, b \) in Triple Algorithm.
5. \( id \) - This represents id of node.
2.5 Route Maintenance

We suppose that a link goes down at node $i$.

If $i$ becomes a singleton, then, $i$ is partitioned from $D$ and starts erasing routs to $D$.

First, $i$ broadcasts a clear message containing the destination id and the reference level that caused the detection, $i$ is also empty out its set of neighbors. When a node receives a clear message with the same reference level as its own, it clears its height, empties neighbor set, and rebroadcasts the clear message.

When a node receives a clear message with different reference level, it removes sender from the neighbor set but does not rebroadcast. Otherwise, if $i$ is now a sink, according to what is known of its neighbor height, it begins a search for a new path to the destination.

A new reference level is created with:

1. $T \leftarrow$ Current Time
2. $oid \leftarrow$ id of node $i$
3. $rbit \leftarrow 0$

Then, $i$ broadcasts its height to all neighbors. This causes the Full Reversal (FR).

The reflection bit detects whether search is spreading outward from originater, which has $rbit = 0$, or it is being reflected back, which has $rbit = 1$.

The $\delta$ value keeps the links pointing away from originater while search is spreading out, point towards originater when reflected back.

If node $i$ becomes a sink because it receives a new height message, then:

1. If neighbors have different reference level, $i$ takes largest reference level and set $\delta$ to one less than the minimum of all neighbors with largest reference level, this causes Partial Reversal/ Triple Algorithm.

2. If all neighbors have same unreflected reference level ($rbit = 0$), then, $i$ is dead-end. $i$ takes on same $\delta$, $oid$ but $rbit = 1$.

3. If all neighbors have same reflected reference level and $i$ was the originater ($oid = 1$), then, $i$’s attempt at finding a path failed, so, $i$ is partitioned and it broadcasts clearing procedure.
4. If all neighbors have same reflected reference level \((rbit = 1)\) but \(i\) was not originater, then there is a sign that something bad happens with a link due to node mobility. Then, \(i\) starts a new reference level.

When a link comes up, node exchanges height information (This will not cause any node to lose existing path).

### 3 Conclusion

For this topic, we are interested in Routing Algorithm with the graph system that has a specific sink node \(D\). Each node in the system has a set of properties, which are timestamp \((T)\), originating processor \((oid)\), \(rbit\), and two others \((\delta, \text{ and } id)\) that take responsibility as same as in Triple Algorithm. Each one communicates with each other on the same network by using message. The node \(i\) will broadcast message to each others if there is an incident such as \(i\) becomes a singleton or \(i\) becomes a sink. So, other nodes can receive this message and update the information.