EDA387: Computer Networks - Lab 2.4
Self-stabilizing Algorithm for finding the centers with the Tree Topology

Group 5: Haitham Babbili and Olalekan Peter Adare

8th October 2020

Given Pseudo-code

Algorithm 1: Self-stabilizing algorithm for center-finding; code for p_i

1 **Constants:**
2 $N(i) = \{p_j : (p_i, p_j) \in E\}$ p_i's neighbors;
3 **Variables:**
4 h_i stores p_i's h-value.
5 **Macros:**
6 $N_h(i) = \{h_j : (p_i, p_j) \in E\}$ p_i's neighbors h-values (multiset);
7 $N_h^-(i) = N_h(i) \setminus \{\max(N_h(i))\}$ contains all elements of $N_h(i)$ without one maximum h-value. E.g., $N_h(i) = \{2, 3, 3\} \Rightarrow N_h^-(i) = \{2, 3\}$;
8 $leaf(i) = (|N(i)| = 1)$;
9 **do forever**
10 **begin**
11 **if** $(leaf(i) = true) \wedge (h_i \neq 0)$ **then** $h_i \leftarrow 0$;
12 **if** $(leaf(i) \neq true) \wedge (h_i \neq 1 + max(N_h^-(i)))$ **then** $h_i \leftarrow 1 +$ _____;

Question 1: Missing Detail on Line 12

By comparing lines 11 and 12, the missing detail in line 12 is: $max(N_h^-(i))$.
This is now written as $h_i \leftarrow 1 + max(N_h^-(i))$

Question 2: Set of Legal Executions

Specifically, for us to have legal executions, we must consider two (2) scenarios. This is based on deduction from Lines 11 and 12 from the given algorithm. It is either you have a leaf node or a center node. Therefore:

(1) If p_i is a leaf node in the tree topology or graph, which means p_i is a degree 1, then, $h_i \leftarrow 0$, i.e set the value to zero.
(2) If p_i is not a leaf node in the tree topology or graph, which means p_i has degree greater than 1, then $h_i \leftarrow 1 + max(N_h^-(i))$. This node may eventually be the center node p_i.

Question 3: Convergence of Algorithm 1

From the pseudo-code of the algorithm, it is either a node is a leaf or it ends up being the center node. The algorithm converges when if we start from an arbitrary state and we still reach a safe configuration, that belongs to the legal execution.

Assumptions

- **The determination of a center node starts with knowing the degree, or the number of edges, of each node p_i in the graph.**
- **Every node p_i can read the degree value of all its directly connected neighbours. Continuous reading of the memory of the neighbouring nodes and updating the local register, will eventually make every node to know the degree values of every other node in the graph.**
- **The leaf of the graph always has only one degree, or one edge connection.**
- **The algorithm starts with the leaf nodes, or nodes with the lowest degree value. It logically deletes this node and by continuous reading of the neighbour's memory, then, the new degree values are updated on each node. This means that the degree values will continue to decrease until we are left with one node or two nodes remaining, which will finally be the center of the graph. We infer therefore, that a tree graph with end with either 1 or 2 center nodes.**
- **For the given algorithm 1, the height for us, h_i, is the number of hops, or edges to be crossed, from the node p_i to the leaf of the graph.**

Proof

We can proof this by induction.
If we start with p_i that has h_i = 1, this means that p_i is a leaf node, we will set h_i to 0 based on line 11. Then, it is possible with one asynchronous cycle, to reach a self-stabilizing state that belongs to the legal execution, and the center node will have the maximum h-value.
If we start with p_i that has h_i = k, and it has neighbour nodes p_{i+1} and p_{i-1}, p_i will count the value of its neighbour and will compare their values with its own value. If p_{i-1} has a h_{i-1} value that is lower than p_i, it will set h_i to 1+$max(N_h^-(i))$, and h_{i-1} is set to 0. Then, if the h_{i+1} value of p_{i+1} is greater than the h_i value of p_i, then we set h_{i+1} to 1+$max(N_h^-(i))$ and set h_i to 0. So within k+1 asynchronous cycles, the system configuration will reach a self-stabilization state following a legal execution. This process due to line 11 and 12 of the pseudo-code. As long as h_i value of node p_i is the highest in the graph, p_i will remain in the set of the centres of the graph.

Question 4: How can node p_i know whether it is part of the center set

By deduction, in a safe configuration, if the h_i value of node p_i is greater than, or equal to the h_j value of its neighbour node p_j, then p_i belongs to the set of the center nodes. This means it has to read the h-values of its neighbours, the degree values, and then compare the values with its own, h_i value. Specifically, this does not depend on the number of the centers of the graph. Although, it has been established that in a graph with leaf nodes that have only one degree each, there can be only one (1) or two (2) centers in that graph.

Question 5: Knowing the center of the graph

We think there is no need to modify the algorithm. It is already self-stabilizing.

Based on the assumption that the graph in question does not have a loop. This means that the leaf nodes have only one degree each. There is a possibility of having one or two centers in a tree graph. The center node will always be the node at the middle of the graph and will have the highest number of degrees when we have reached a safe configuration. Since we are working with undirected graph, we are not constrained to a particular computational direction from one leaf node to the center node, and vice-versa.
Consequently, if we have arrived at a safe configuration, p_i may not be part of the center set, if and only if, there is an h-value of its neighbour, p_j, that is greater than its own h_i value. This means if $h_j > h_i$. Invariably, it means that p_i is a directly connected neighbour of p_j, i.e p_i can determine the edge that connects to the nearest node that belongs to the center set. Node p_i will always be in the center set if its h_i value is the highest in the graph.