Graph Algorithms

You are given a set of interacting processes that exchange neighbor adjacency data. The process works like this:

1. Each process reads neighbor information and cost from a file
2. Each process builds a network graph from all nodes using this neighbor information

Given this information answer the following questions:

1. Find the shortest path from any to any node given.
2. Find the minimum spanning tree.

Also, reconfigure the network connectivity graph on a node when a node (process) dies or an edge is lost

Issues to be handled:

1. Nodes (process) can startup and die at any point of time
2. No central manager for the set of nodes (process)

Some aspects of evaluation:

1. Write-up on approach, design, unit testing, debugging hooks
2. Choice of data structures, threads, processes, IPC
3. Simplicity of implementation
4. Efficiency of implementation
5. Bugs/corner cases handling

Language to be used:

1. C/C++

**Solution**

**Approach & Design**

Approach is simple: **Making each node capable of change**.

* Create a server client model of Socket IPC to communicate with each nodes in every nodes
* Create a graph at each node and keep a timer to check the nearby nodes periodically. Keep checking for nearby nodes in periods of time to see whether any new nodes are created or any nodes died or link lost
* Initiate a client and server session at every node.
* Maintain a structure for storing the information and cost
* Use **Dijkstra's** algorithm, for finding the shortest distance.
* Use **Kruskal’s** Minimum Spanning Tree Algorithm for finding the minimum spanning tree.
* Dynamically handle edge lost and node dies situations.(ie whenever a lost is identified recreate the graph.)

**Unit testing**

Test cases for various scenarios like node deletion, insertion and every time minimum spanning tree and shortest distance are calculated. User must have provision for deleting/inserting specific nodes.

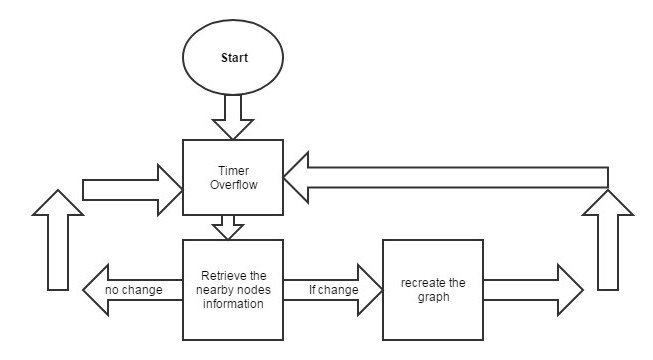
**Data structure**

Create a Graph data structure and populate the information.

**IPC**

Socket based client server model, each node can act as both node and server. And in every interval nodes will keep checking for the nearby nodes and update the information.

**Node’s workflow**

****