Design Document for

Sortest Path Protocol

*Submitted By: Suprit Paul*

1. Introduction
2. Prerequisite
3. Overview
4. Database for information
5. Designing of the model
6. Communication between process
7. Finding the shortest path
8. Finding the minimum spanning tree
9. Remodelling the topology
10. Future Change

# Introduction

This document focuses on creating an algorithm which generates the network topology and finds the shortest path between the nodes in a network. Each node in the network will be a process which creates network topology locally.

# Prerequisite

The following points shall be taken in to consideration while designing the algorithm.

1. Each process will have its neighbour information and cost in a file. The format which shall be present in the file is given below assuming file\_p1, file\_p2,file\_3 are files for processes P1,P2 and P3 respectively.

P1 10 P2 5 P3

Node1 , node2, cost

File\_p1.txt >> 1 , 2, 10

File\_p2.txt >> 2 , 1, 10

2, 3, 5

File\_p3.txt >> 3 , 2, 5

1. Each process shall build a network graph locally
2. No Central Manager for forming the network topology

# Overview

This document focuses on running all the processes on same device as well as on the different machines. Each process shall have two modules to handle all the cases. One module(**RouteMe**) shall deal with creating the network topology and other modul(**CommuteMe**)e shall talk to the different neighbouring process to gather information. Both the module shall interact with the data base to insert, delete or modify information regarding the nodes/process.

**RouteMe** Module shall create the network topology and shall find the shortest path between the nodes and minimum spanning tree.

**CommuteMe** Module shall broadcasts its Socket information and status information using UDP broadcast mechanism as well as it shall keep listening on a TCP socket for any information coming from the other process.

# Database for information

Each process shall maintain two databases.

1. Topology Database : contains shall contain the graph of the network. When the correct process receives the information of the neighbouring process it keeps adding to this topology.
2. Neighbouring Node database: containing the linkied list with the nearest node ,the cost of the edge, status of the node (dead/alive).

# Designing of the model

The model shall parse the file and shall create a linked list <Adjacent\_list>, where it can add the adjacent node information <nodes>.

Struct Node {

Int node2;

Int cost;

Bool isAlive;

Int tcpSocketId;

};

Struct adjacent\_list {

Struct Node node;

}LIST;

Void addEdges(int node1, int node2, int cost);

Void findShortestPath(int s); // print shortest path from s

Void minimumSpanningTree(Struct List s); // find the minimum spanning tree

# Communication between different processes

Communication between processes shall happen in two ways.

1. Create a TCP socket in a particular port number for communication with different process and have a client and server running in it to handle any incoming messages. Once other process receives the information from neighbour it can be shared across nearest neighbour, so that all will have the information about the process connected in the network.
2. Broad casting the own information using UDP.

As soon as a process starts it shall broadcast a hello message with its tcp socket number. The Broadcast has to be in frequent interval (once in every 3 minutes to handle if any process is existed from the network). Now this information has to store in neighbour node list inside the status variable.

# Finding the Shortest Path

The shortest path can be found using Dijkstra’s algorithm, two sets are maintained, one set contains list of vertices already included in SPT (Shortest Path Tree), other set contains vertices not yet included. With adjacency list representation, all vertices of a graph can be traversed in O(V+E) time using BFS. The idea is to traverse all vertices of graph using BFS and use a Min Heap to store the vertices not yet included in SPT (or the vertices for which shortest distance is not finalized yet).  Min Heap is used as a priority queue to get the minimum distance vertex from set of not yet included vertices

# Finding Minimum Spanning tree