GRAPH ALOGARTHIM

Jayesh J

Table of Content

|  |  |  |
| --- | --- | --- |
| 1. | Problem statement | 3 |
| 2 | Design | 4 |
| 3. | Algorthim | 5 |

1. Problem Statement (Graph Algorithm)

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

1. Each process reads neighbour information and cost from a file
2. Each process builds a network graph from all nodes using this neighbour 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 start up and die at any point of time
2. No central manager for the set of nodes (process)

**Architecture Design**

Approach to problem statement is done using three sub-systems.

1. Neighbour Adjacency :

Store neighbour adjacency related data ( Node , cost ) read from its own file and sent from adjacent node .

Following are the things to be carried out

* 1. Forming of neighbour ship (duplex socket connection) with all neighbours mentioned in file.
  2. Exchanging of all neighbour information to all other neighbours.
  3. Trigging shortest path calculation for every in every 40 second. (Assumption spf will be triggered every 30 sec )
  4. Triggering update to neighbours we get new updates from neighbour

1. Shortest Path calculation :

Calculate shortest path to each node currently their adjacency database using Dijkstra shortest path algorithm.

1. Using prims algorithm to generate minimum spanning tree algorithm .

Detailed data structure and algorithm is mentioned below section

**Assumption**

SPF and MSPT will be triggered after each 40 sec . So changes done in between the result will be calculated in next run

**Data structure Design**

1. Each individual node will be taking adjacency related data from

*adjacency .txt .*

Each entry in file will be in format of <IP Address > < cost >

*1.1.1.1 1*

*2.2.2.2 5*

*The data structure is to implement 2-D array using adjacency linklist .*

1. Structure for holding adjacent link node related data

*Struct Adjnode*

*{*

*int cost ; /\* To cost metric to reach neighbour \*/*

*char addr[16] ; /\* To hold ip address of neighbour \*/*

*struct Adjnode \* next;*

*struct Adjnode \* prev;*

*}*

1. Structure for holding all neighbour node list

*Struct AdjNeghList*

*{*

*char addr[16];*

*struct Adjnode \* node;*

*struct Adjlist \* next;*

*struct Adjlist \*prev*

*}*

**Algorithm Design**

1. Read all the user configured data from ajcencency.txt.

Using file pointer and reading line by line input from

File which consist of ( nexthop address with cost associated with it )

1. Establish duplex connection with all the adjacent neighbour using socket connection .
2. Save all adjacency related (Link) data into our data structure

Into adjacency link list ( mentioned in Datatsrtucture).

1. Trigger a shortest path calculate at every 40 sec .

Even though u got new update from neighbour

1. Using *Dijkstra algorithm* find the shortest path and save

In a different link list

1. Using *Prims minimum spanning tree* and greedy algorithm to calculate SPT .
2. Display both result after each calculation in a tabular form .