Low Level Design Document

Introduction

- The application will request for a network topology represented in the form of matrix in a text file which will be uploaded and can be processed to find out the shortest path between the source and the destination.
- The processing involves building a connection table for each router, that is the path that router will take to reach other routers.
- Also the application can modify the topology matrix to represent if any router is shut down and that it cannot participate in the paths to find the shortest path between the source and the destination

Software Interfaces

- The application is built upon Python 2.7 programming language and the GUI implementation is done using the Python library called TKinter.
- Operating system : Windows 10

Design Description

- The application is designed using the djkstraws algorithm to find the shortest path between the source and the destination.
- Multiple shortest paths use depth first search tree algorithm.
- Below mentioned are the methods created for each functionality that can be found in our source code.

i. def process_file(fname):
   - This function will check if the user uploaded file containing the topology matrix is a text file. If correct file format is found then the function will check for the number of nodes for the topology matrix which has minimum limit set to 8.

ii. def connection_table():
   - This function will take the processed file uploaded and generate interfaces for each router that will represent the path that each router will take to reach every other router if any path exists between those routers.

iii. def set_distances(router_matrix):
   - This function sets the distances of the routers based on the topology entered by the user.
   - A list of list is created with the distances attached to the routers

iv. def dijkstra(start):
   - This function calculates the shortest distances based on Djiktra’s algorithm.

- The minimum distance within the routers in computed and selected.
- The previous of the router is then added to the previous list. In case multiple paths exists the previous router is a list of lists with multiple routers

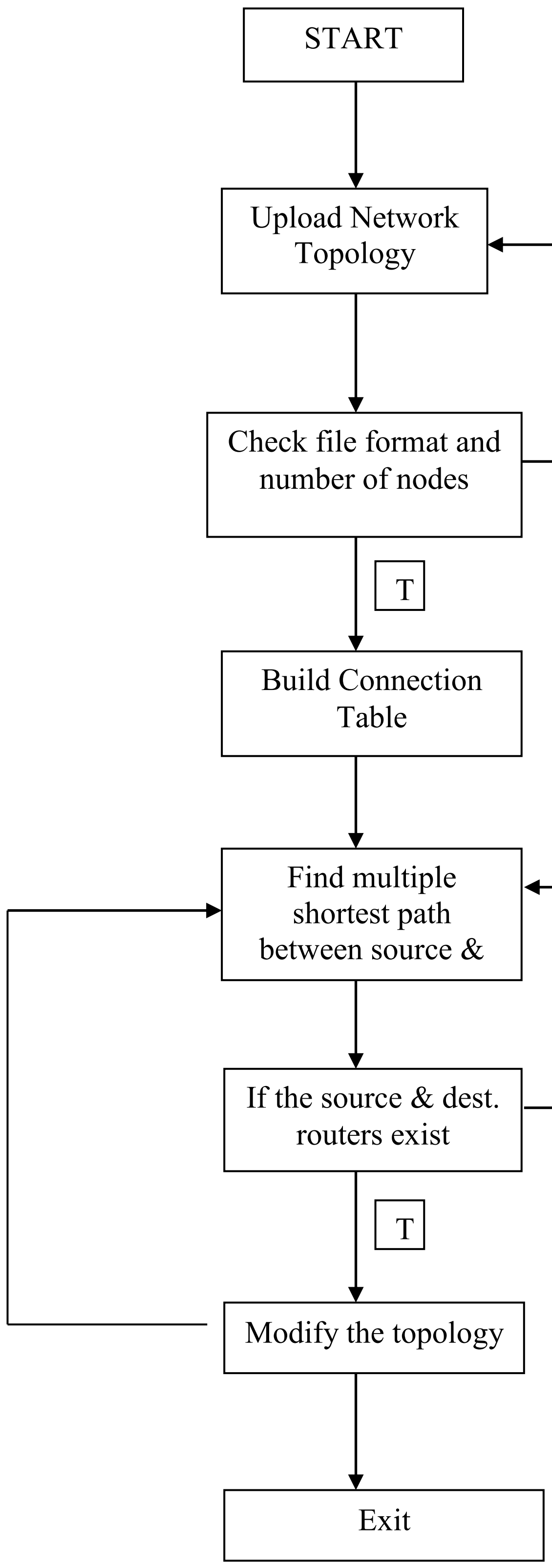
v. def dfs_paths(start, dest):
- This function returns all the paths that occur with respect to the shortest path.
- The Depth First Search algorithm is used to navigate across the previous nodes in order to compute all the distances from source to destination

vi. def modify_topology():
- This function asks the user for the router to be deleted.
- In case the router is a source or the destination router selected in "destination_router()" then the intelligent approach is source/ destination router is asked again to the user and the shortest path is then computed from the new source or destination router
- In case the router deleted in not the source / destination router then the connection table for all routers are displayed on the screen and the shortest path is computed from source to destination router.

vii. def network_topology():
- This function calls the process_file() and converts the txt file into a network topology which is visible on the screen
- It also sets the distances of the routers using the set_distances()

viii. def destination_router():
- This function calls the djiktras() and dfs_paths() to calculate the shortest path and display all the multiple paths between the source and the destination

Flowchart:


START
Upload Network Topology
Check file format and number of nodes
F
Invalid file format
T
Build Connection Table
Find multiple shortest path between source &
If the source & dest. routers exist
F
Routers not found
T
Modify the topology
Exit

References :

- Docs.python.org,. '24.1. Tkinter â€” Python Interface To Tcl/Tk â€” Python 2.7.11Rc1 Documentation'. N.p., 2015. Web. 22 Nov. 2015.
- Docs.python.org,. 'The Python Tutorial â€” Python 2.7.10 Documentation'. N.p., 2015. Web. 22 Nov. 2015.
- Wiki.python.org,. 'Tkinter - Python Wiki'. N.p., 2015. Web. 22 Nov. 2015.