Link State Routing Project

Simulate routers by using Dijkstra algorithm and send packets between them from the manager to create a traceroute.

Manager-

* Creates the routers on a separate thread for each
* Creates the links between them
* Sends the packets from the input file (parsing necessary) to each associated router for them to relay.
* Input file:

Needs to be parsed, first line being the number of routers (Needed locally)

Second -> EOF(-1) is a list of links and associated costs. (Probably fed into a Router object)

Router-

* Communicate and build the forwarding? Table via UDP
* Communicate with the manager over TCP to receive tasks

Order of Operations

1. Parse the file
2. Create threads for each router object created during parsing
3. (Instruct each router to create a UDP socket for connecting to neighbors)
4. Router must know: it’s address (0...N), it’s neighbors and the associated UDP ports for them.
5. Create Manager/Router communications via TCP. After the above, a router opens a TCP connection to the manager. The router sends its UDP Port(?) and the manager is requested to send the router a node address and a connectivity table. (Basically giving blank routers (just on a thread with a UDP port), their Node number (N) and connections to create)?
6. Routers communicate to build table for routing between each other using Dijkstra's and LS.
7. Once the final table is completed, the manager can begin sending packets. These also need parsed from the input file and are sorted as (src, dest) pairs.
8. The manager sends these pairs to the src router (sleeps for a few seconds before sending the next) and the src router uses its table to send to the dest router. This is repeated. Manager then sleeps for a while(?) to wait for the routers to finish.
9. Manager sends a QUIT message to all routers causing the threads(routers) to exit.
10. Manager exits.

Manager.out

Output file for the manager.

* Prints information at each step as to show what is occurring.

-For example: Creating routers, sending Node values to router, sending connection information to routers, waiting for routers to build their table, sending packet X to router X destined for router X, waiting for packets to send, sending quit messages, exiting.

Router tasks:

1. Create a UDP port
2. Create a TCP port
3. Connect to manager with TCP
4. Send manager its UDP port, requesting for a number and connections associated with it.
5. Send ready message to manager so LS algorithm can begin
6. Wait for go ahead from manager
7. Send link requests to neighbors and wait for ACK
8. Once all ACKs are in, send message to manager
9. (At some point around here, create a forwarding table for the router with its neighbors as the only connections)
10. Send its Link State Packet to each neighbor (contains info about its links)
11. When a LSP is received (can be told the number of links by the manager to speed things up), send it to all other neighbors but the one it was received from.
12. Once all neighbor LSPs are received, use SPT (Dijkstra’s) to find shortest paths to all nodes.
13. Output this table to routerX.out
14. Update the forwarding table based of the SPT
15. Wait to receive packets to forward (could be from manager or another router)
16. Check the dest of the packet, if it is the router’s N - add it to router.out as completed and kill the packet. Otherwise, lookup the dest in the forwarding table and pass it on.
17. Exit when Quit message is received.

Router.out

Same idea as manager.out, log anything that happens.

README

Must contain:

* Implementation description
* Message formats for Manager->router, router->Manager, router -> router
* Documentation of functions, including: params and return values as well.
* Problems/Bugs

Other info:

Test files are assumed to have correct formatting

Output files should have timestamps and EVERYTHING going on.

How to start:

-Skeletonize threading first and make sure basic creating and exiting works correctly.

-Create functions for parent and children independently. Ie. fork() -> parent calls managerprocess() & fork() -> child calls routerprocess().

-Increase children slowly, make sure everything works/exits correctly and then increase. (up to 10?)

-Determine data structures for SPT algorithm, LS should be a function that should receive a matrix and point to a forwarding table to fill.

-Implement a lookup function that takes the forwarding table and a dest as arguments. Return a hop.

Current tasks:

-Implement threading and process creation for manager and routers.

-Implement LS algorithm

-Parse input file

-Implement lookup function

-Implement UDP/TCP messaging for r->m, m->r, r->r along with message formats.