READ ME

STUDENT I.D S0840844

1. How to run

To run the simulator, perform the following steps:

(1) cd compiled
(2) java RIP.Input
    This prompts a GUI interface that allows the user to choose the input file containing network description (See the folder Sample Input file)

2. Documentation

The simulator consists of 3 main classes:

(1) Input.java
    Methods:
    (a) Main method: Generates an GUI file chooser to assist the user to select the network description file through browsing directories and then creates and runs the simulator
    (b) printTable() Prints the routing tables of all nodes in the network
    (c) performLinkFailures() Executes commands to inform nodes when a given link in the network is broken
(2) InputNode.java
    Constructor:
    InputNode(String P, int[] adr): Creates a node with name 'P' and local addresses 'adr'
    Methods:
    (a) addEntries(InputNode):
        Creates the local routing table containing local addresses of addresses of adjacent nodes
    (b) getName():
        Returns the name of the node
    (c) getRoutingTable()
        Returns the local routing Table
    (d) copyRoutingTable()
        the process generates a *deep* copy of its routing table to send to its neighbours whenever it experiences a update in its local table

(e) receiveMsgFrom(String sender, Hashtable¡Integer,LinkCost¿ routingtable):
Receives the node sender's routing table
(f) receiveTable():
Uses the RIP protocol to update local routing table
(g) settableUpdate(boolean state):
sets the boolean tableUpdated to state
(h) getRoutingTable(boolean state):
returns tableUpdated . If tableUpdated is true, this implies local routing table has been updated
(i) run():
runs the thread that performs receiveTable()

(3) InputCommand
Methods:
sendProcess(String P): creates threads to simultaneously send copies of the updated routing table of process P to all the neighbours

3. Algorithm

Th current program initiates the simulator whenever it reads a *send name1* command in the network description file. The simulator proceeds as follows:

- Step1:
  To each process P in {neighbours of name1}
  A copy of name1's routing table is sent
- Step2:
  As soon as process receives a message, a new thread is created and fired to run the node's receiveTable() method that implements RIP protocol. This allows the simulator to model concurrent communication between processes.
- Step3:
  If a process P's routing table is updated, then we perform step 1 again but this time setting name1=P

4. Loops

In the current simulator, *link-fails* are only allowed when the network has reached a stable situation after the executing all *send* commands specified in the network description file. However, in reality this is definitely not the case. Failure between links can occur at any stage during the execution of the routing algorithm . This means that it is equally probable that links between any two nodes can fail during the execution,termination or initialisation of the routing algorithm.

To check the effect of link failures during the execution of the routing algorithm,I had altered the code to allow link failures between node to occur before the network has reached a stable situation. The network setup that I used to conduct the simulation is illustrated by figure 1:

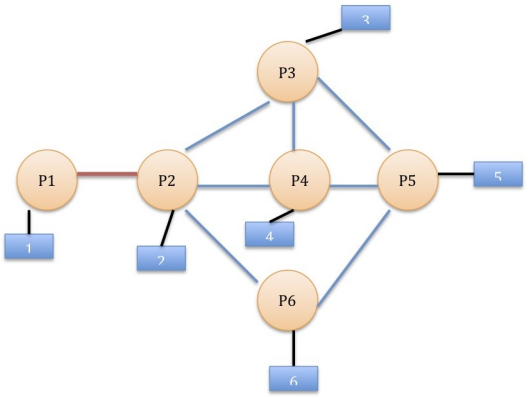

FIGURE 1. Network(The description of the network is given in "Sample input file/sample_input2.txt")

The simulation is initiated through node p4 sending its routing table to its neighbours. Since messages are been carried by threads, the order in which nodes will get updated will vary from run to run. Figure 2 shows an example of one of the runs that I have observed where the network experiences a routing loop. In this run, the table updates occur according in the following manner:

(1) P5 and P2 both receive copies of P4 routing table and undergo a routing table update
(2) P3 receives the same copy of the routing table from P4 but it's local tab;e does not get updated.
(3) After P2 has been updated, it sends copies of its routing tabLe to P1,P4,P3,P6
(4) P4, P3 and P6 receives P2's routing table and gets updated.
(5) P3 now sends copies of its routing table to P4 ,P5 and P2 and similarly P4 does the same. Let message from P3 to P4 be "M1" and let the message from P4 to P3 be "M2"
(6) The simulator now initiates the *link-fail* command between P1 and P2. This forces P2 to send copies of routing table containing link failure information to P3 P4 and P6
(7) P4,P3 and P6 receives P2's table and updates their own.

However, as soon as the updated node P4 receives "M1". it updates its link for the address 1' to P3. This is because the received table is an outdated table of P3. The table was sent in the time when P3 believed the link between P1 and P2 was active. Similarly the same situation occurs when the now updated P3 node receives "M2" . Hence we get a routing loop.

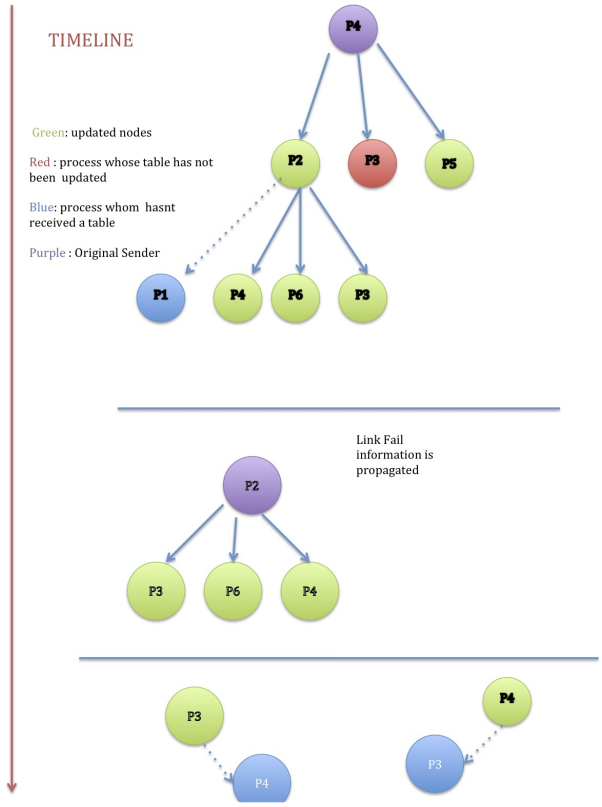

FIGURE 2

4.1. **Second Reason.** The simulator also enters a routing loop when multiple threads act on the same data. This is in fact the fundamental problem that I faced when I was implementing the code for the simulator. Consider the situation where the node P4 receives messages containing copies of routing tables from nodes P3 and P5 in the network described in figure 1. Each message is carried by a separate thread. Now when both these threads are allowed to update P4 's local routing table concurrently, the interleaving of the threads' operations lead to inconsistency in the state of P4 routing table and this causes the system to enter a routing loop.

5. DRAWBACK WITH THE CURRENT SETUP

Lets consider the network described in Figure 3:

After conducting multiple runs in the simulator, I have observed that this network stabilises and enters into one of two possible states after every run. In the

first stable state, the links to the addresses at P4 and P6 in node P2's routing table are updated to P1while in the second state, these link becomes P3. Let consider the latter case. Now if a link failure occurs between nodes P1 and P4, then P1 will become disconnected to all addresses which it accessed through the link P4. This is huge drawback. Although there exists a path between P1 and every other node in the network, the resultant routing table of P1 will infinite costs associated with some nodes hence indicating there exits no path between P1 and those respective nodes. The reason why this situation occurs is because P2 apart form P1's local address doesn't use P1 as a link to access any other address. Hence P2 does not update its local table when P1 sends it a copy of its updated routing table after the link failure occurs.

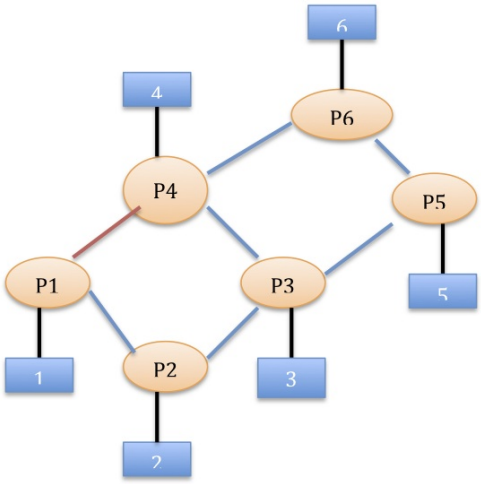

FIGURE 3. Simulated network 2