Grading of the programming project

Below are some scenarios for the programming project. I will use similar scenarios for the project, I don't guarantee that I will use the same ones, but perhaps similar ones.

You will get between 0 and 40 points if your program does not run any of the following scenarios. How much between 0 and 40 is a subjective call by the TA

Each of the scenarios below is worth 15 points (there are four scenarios), for a total of 100

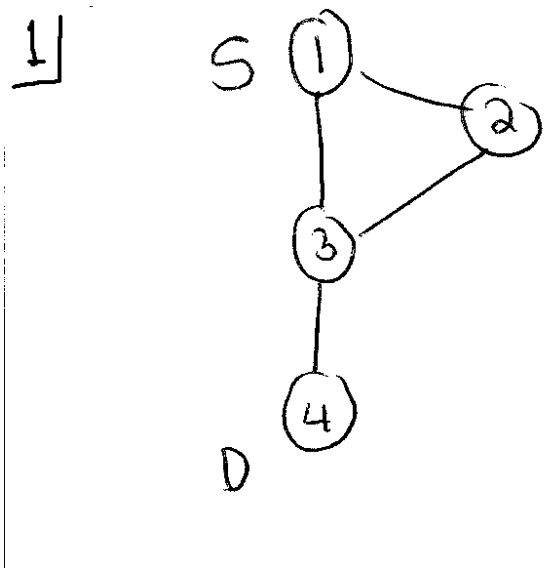
Again, we will not check that your code is nice or pretty, we just care that it runs the scenarios correctly. If it does, you get a 100.

When you submit it, don't forget to include a README.txt file with instructions on how (and where!) you compiled your code, so we can compile it the same way.

Here we go.

Scenario 1

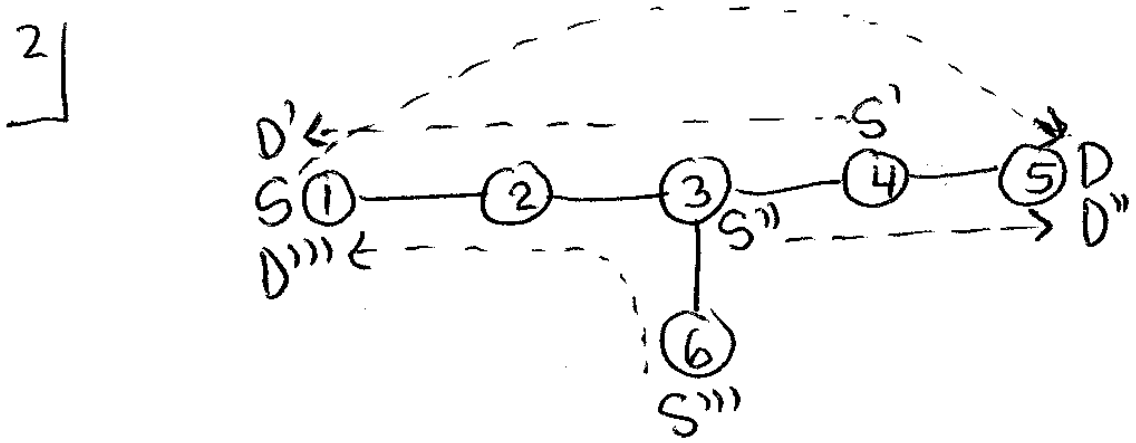
This scenario tests basic functionality, i.e., sending RREQ, receiving RREPLY, and making sure the data reaches the destination. All links are up (in both directions) and remain so. I will let you come up with the topology file (straightforward)



```
node 1 3 “this is a message from 1 to 4” 10 &
node 2 2&
node 3 3 &
node 4 4 &
```

Scenario 2

Here, we will have more nodes, more sources and destinations. We are trying to take advantage of the fact that if a node already knows how to reach a destination, it should not initiate a RREQ



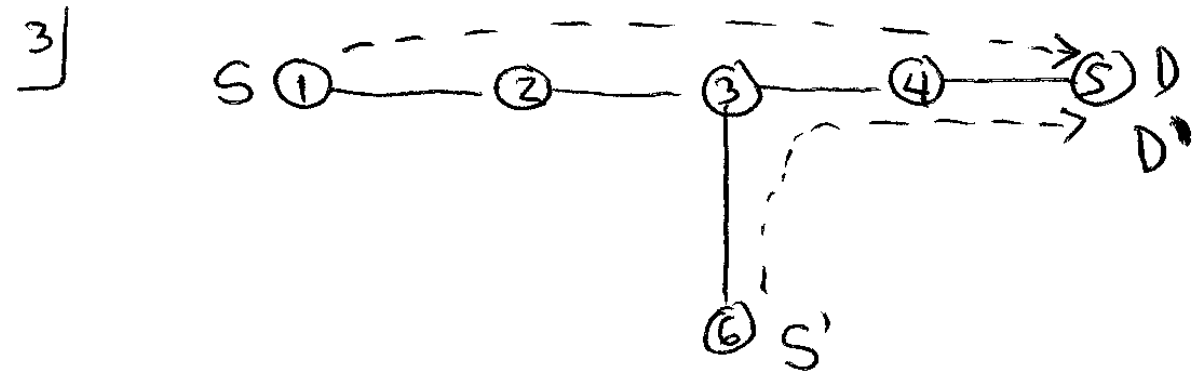
All links are up in both directions and remain up. Note that sources 3 and 6 begin early enough that the intermediate nodes should still have information about their destinations, so only node 1 should initiate a RREQ

```
node 1 5 “Hello from 1 to 5” 5 &
node 2 2 &
node 3 5 “This messages from 3 to 5” 15 &
node 4 1 “4 says hello to 1” 15 &
node 5 5 &
node 6 1 “Hello 1 from 6” 15 &
```

I will run the same scenario again (after removing all channel files, of course), but with 50 seconds instead of 15. By then all the nodes should have emptied the routing tables, and hence 3, 4, and 6 will issue a RREQ looking for their destinations.

Scenario 3

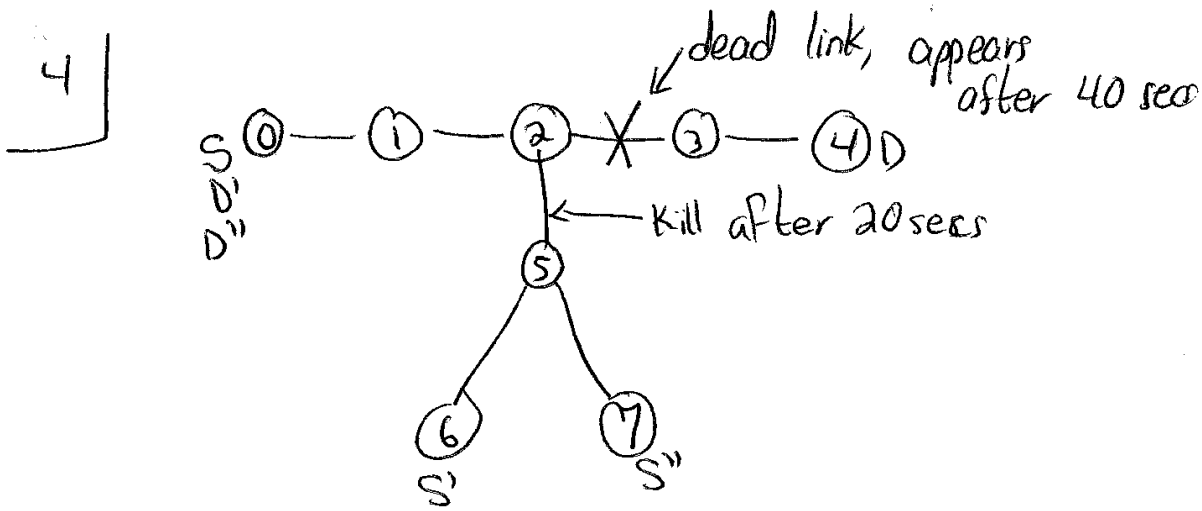
Here I am interested in the intermediate node generating an early route reply, and sending a gratuitous route reply to the destination. Again, all links are always up.




```
node 1 5 “this goes from 1 to 5” 5 &
node 2 2 &
node 3 3 &
node 4 4 &
node 5 5 &
node 6 5 “this message from 6 to 5” 15 &
```

Scenario 4.

This last scenario deals with links breaking. Thus, I will give you the topology file. Note that if a link is unidirectional it is still considered as “down” since AODV can only handle bidirectional links.



```
node 0 4 “message from 0 to 4” 4&
node 1 &
node 2 &
node 3 &
node 4 &
node 5 &
node 6 0 “message from 6 to 0” 14 &
node 7 0 “message from 7 to 0” 14 &
```

The fact that the top link is dead from the beginning will cause node 0 to retransmist the RREQ. After the link comes back, it should then be successful in its next RREQ

Nodes 6 and 7 know how to get to 0 from the first RREQ that 0 sent out, so they should not send RREQ of their own

When the link from 5 to 2 dies, node 5 should send back a gratuitous RREPLY to its active backward neighbors indicating that the destination is not reachable.

`topology.txt:`

```
0 up 0 1
0 up 1 0
0 up 1 2
0 up 2 1
0 up 2 3    /* note the absence of link 3 to 2, so in effect, the whole link 2—3 is down */
0 up 3 4
0 up 4 3
0 up 2 5
0 up 5 2
0 up 5 6
0 up 6 5
0 up 5 7
0 up 7 5
20 down 5 2
40 up 3 2
```