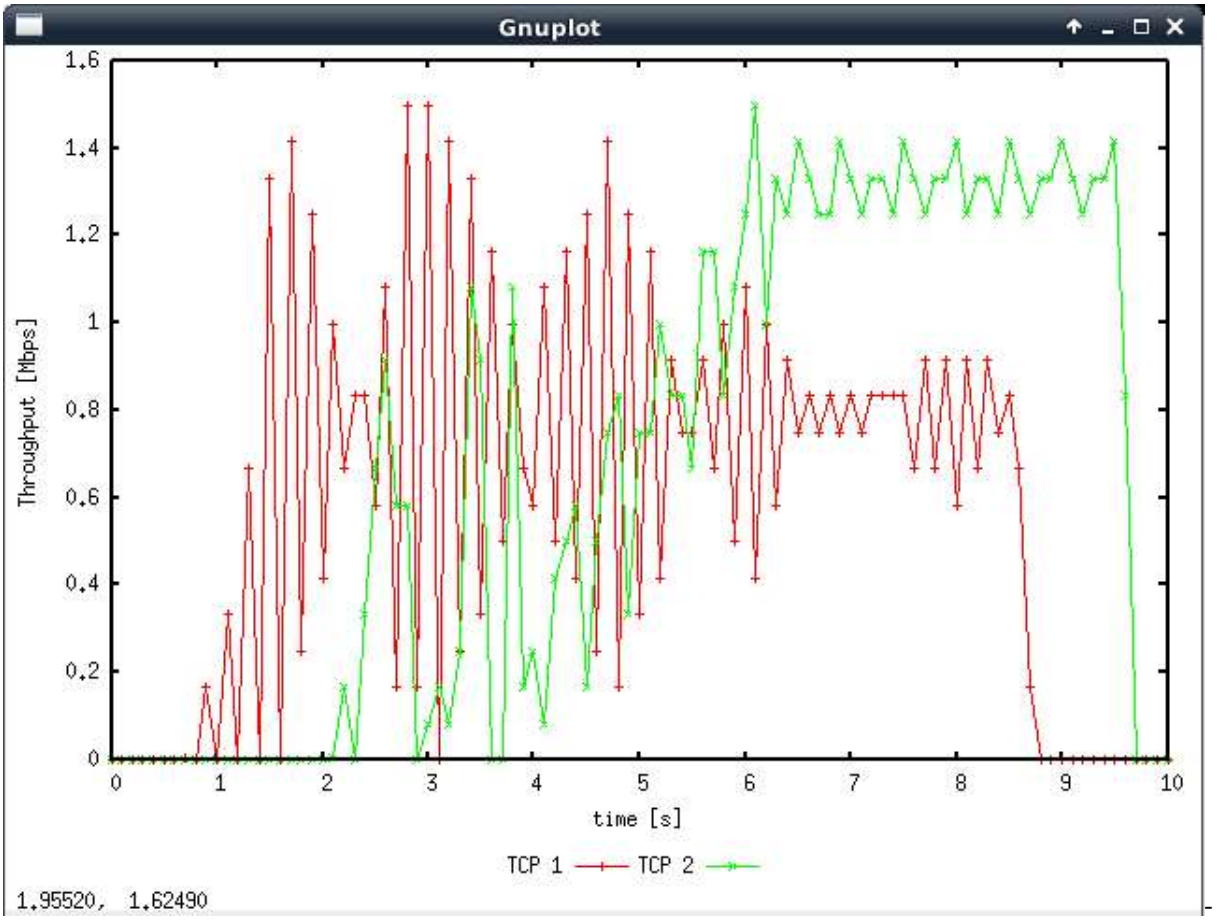
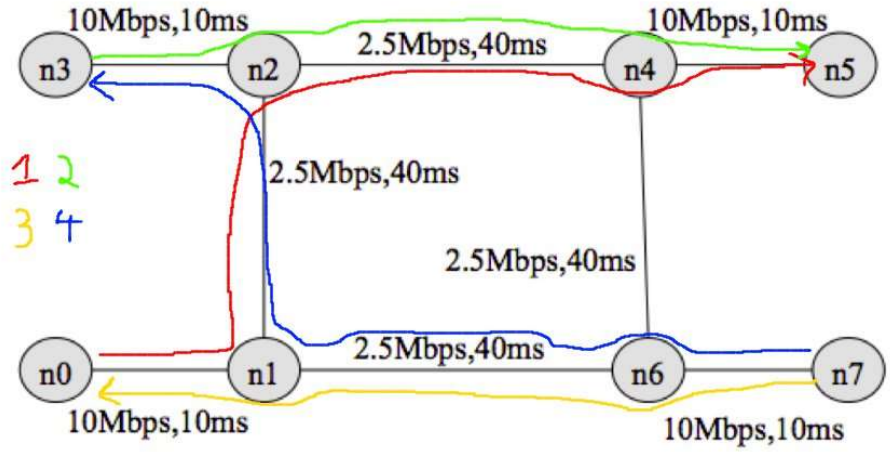
Lab06 – z5117408

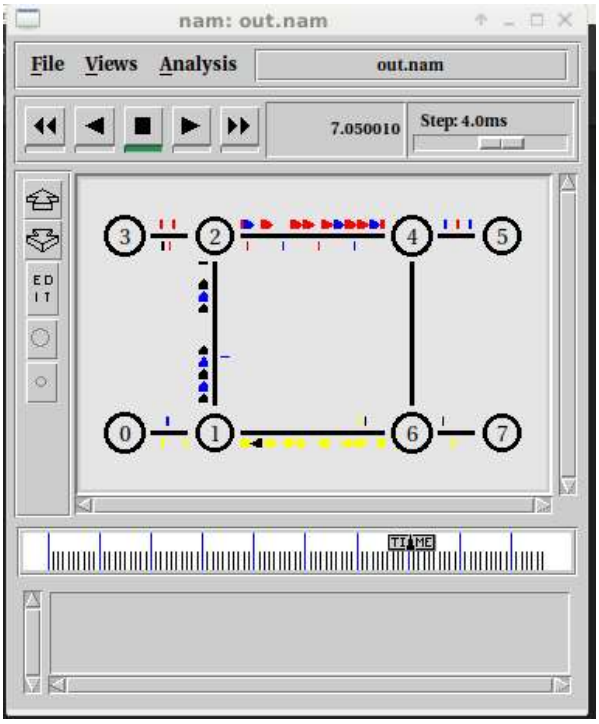
EXERCISE 1



Graph output from plot script

Question 1: Why the throughput achieved by flow tcp2 is higher than tcp1 between time span 6 sec to 8 sec?





These pictures represent the flows of traffic. As you can see, flow1 (red in top pic, blue in bottom) must share the bandwidth of the already small 2.5Mbps link between nodes 1 and 2 with traffic from flow4 (blue in top pic, black in bottom). Flow2 (green in top, red in bottom) does not have to use this shared link. Flow4 starts at 4 seconds and so it's slow start would probably lead to lots of traffic around 6-8 seconds (meaning tcp1 would back off upon noticing signs of congestion because it now doesn't have the entire bandwidth to itself – tcp is fair). Even though flow1 and flow2 travel along the same top path (n2->n4->n5), flow2 has more packets on it (as seen in second picture) because flow1's packets have likely been queued and/or dropped at the n1->n2 bottleneck.

Question 2: Why the throughput for flow tcp1 is fluctuating between time span 0.5 sec to 2 sec?

It is in the bandwidth discovery stage. The window is being exponentially (^2) increased per RTT but when it overshoots the bandwidth it must reduce the window size again (reduce to 1 on timeout, half on triple dupe ack/fast retransmit).

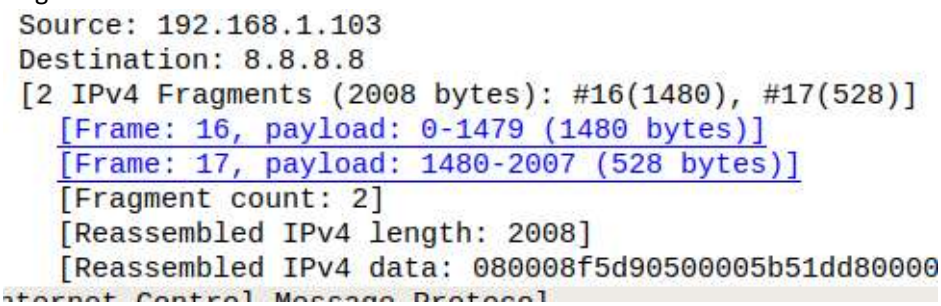
Question 3: Why is the maximum throughput achieved by any one flow capped at around 1.5Mbps?

Each of the flows use a 2.5Mbps link at some point in their path from src->dst and in all cases this link is shared with one other flow. As TCP is fair, the flow can only use about half of this bandwidth on average (slightly more in the case that the competing flow has less incoming packets at this given time e.g. flow2 can get more than half the bandwidth as a smaller amount of flow1's packets make it to their shared 2.5Mbps link, n2->n4).

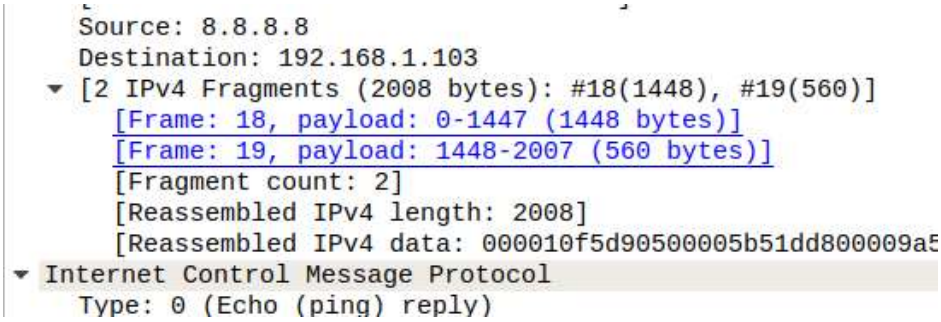
EXERCISE 2

Question 1: Which data size has caused fragmentation and why? Which host/router has fragmented the original datagram? How many fragments have been created when data size is specified as 2000?

Both data sizes 2000 and 3500 have caused fragmentation, splitting into 2 and 3 fragments respectively. This fragmentation has been caused because there is obviously an MTU along the path that is less than 2008. The packets appear to be fragmented when travelling in each direction, so the src (192.168...) was the first to fragment the packet. The source (192.168.1.103) is a private IP address though, and 8.8.8.8 is the google DNS server (and so the local DNS cache server was probably contacted first), so the actual router that is doing the fragmenting is likely the router of this private network i.e. 192.168.1.1.Note that the fragments in the forwards direction are of a different size (1480 bytes max fragment size) to the reverse direction (1448 max fragment size), meaning a different MTU is likely causing this and not the same one both ways (differing inlets/outlets of the router being used?). A data size of 2000 results in 2 fragments total.



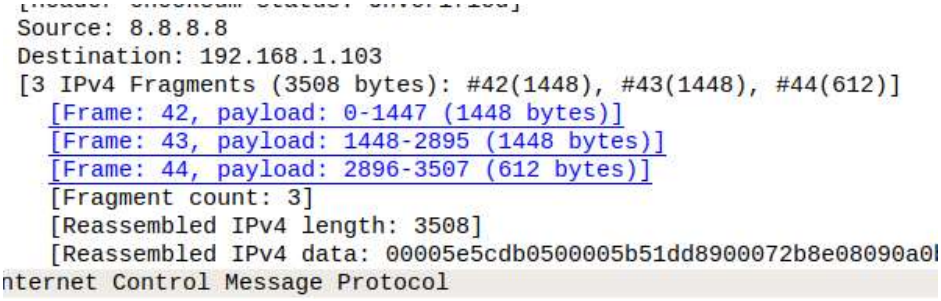
- 2 fragments from 2000 data length packet, 192... to 8.8... with 1480-byte fragments



- 2 fragments from 2000 data length packet, 8.8... to 192... with 1448-byte fragments

Question 2: Did the reply from the destination 8.8.8.8. for 3500-byte data size also get fragmented? Why and why not?

Yes, since a 2000 data length packet (2008 total) caused fragmentation, it is quite obvious that a 3500 byte packet will cause fragmentation (since 2000 is above one of the MTU's, so is 3500).

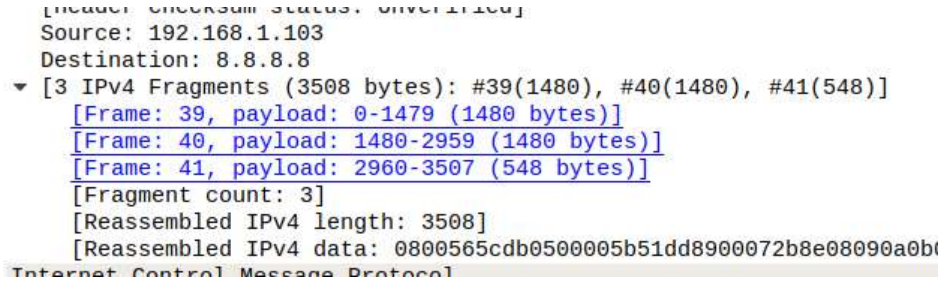


- 3 fragments from - 3500 data length packet

Question 3: Give the ID, length, flag and offset values for all the fragments of the first packet sent by 192.168.1.103 with data size of 3500 bytes?

First frame: ID = Frame 39, length = 1480 bytes, flag = 1, offset = 0
Second frame: ID = Frame 40, length = 1480 bytes, flag = 1, offset = 1480/8 = 185
Third frame: ID = Frame 41, length = 548 bytes, flag = 0, offset = 2960/8 = 370

*note: couldn't find flag field anywhere but am assuming it is referring to the more fragments flag and so theoretically it should be 1 for the first 2 fragments, and 0 for the 3rd one (as it is the last fragment).



- First 3500 data length packet fragments (sent from 192.168...)

Question 4: Has fragmentation of fragments occurred when data of size 3500 bytes has been used? Why and why not?

No. The maximum of the two fragment sizes is 1480 bytes. It must be the case that after the router that does the fragmentation of the 3500-byte length packet into these smaller fragments, there is no MTU smaller than 1480, hence no more fragmenting is required to be done (I.e. after fragmentation, all links can support transmissions of 1480 bytes and above).

Question 5: What will happen if for our example one fragment of the original datagram from 192.168.1.103 is lost?

If one fragment is lost, then the whole packet (not just that one fragment) will have to be retransmitted. The router doesn't really care if it is forwarding a packet or a fragment, it just tries to forward it (best effort – it can drop). Thus, it is the responsibility of the sender-receiver end systems (and the protocols running on them e.g. TCP) to request a retransmit (possible and required with TCP, but only at the packet level, not the fragment level).

EXERCISE 3

Question 1: Which nodes communicate with which other nodes? Which route do the packets follow? Does it change over time?

There is a UDP agent on node 0 and also one on node 2, this is why they are the source of the packets. Node 5 is the sink for both UDP agents. This results in the following flows of traffic/routes;

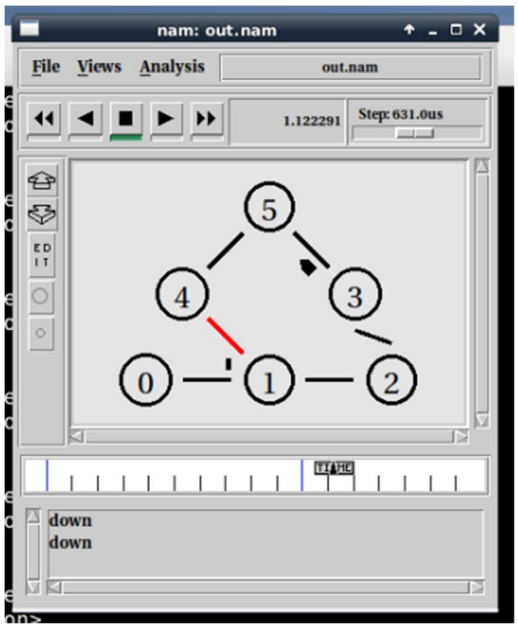
0 -> 1 -> 4 -> 5 and

2 -> 3 -> 5

(where -> means the node on the left is communicating/sending packets to the node on the right)

These routes do not appear to change over time.

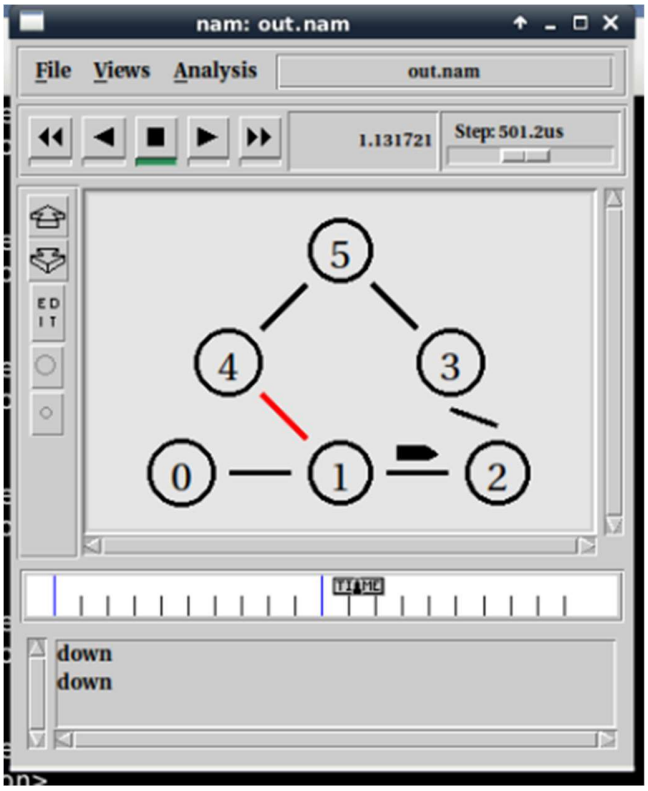
Question 2: What happens at time 1.0 and at time 1.2? Does the route between the communicating nodes change as a result of that?



- 1-1.2 seconds, link 1->4 goes down/dies

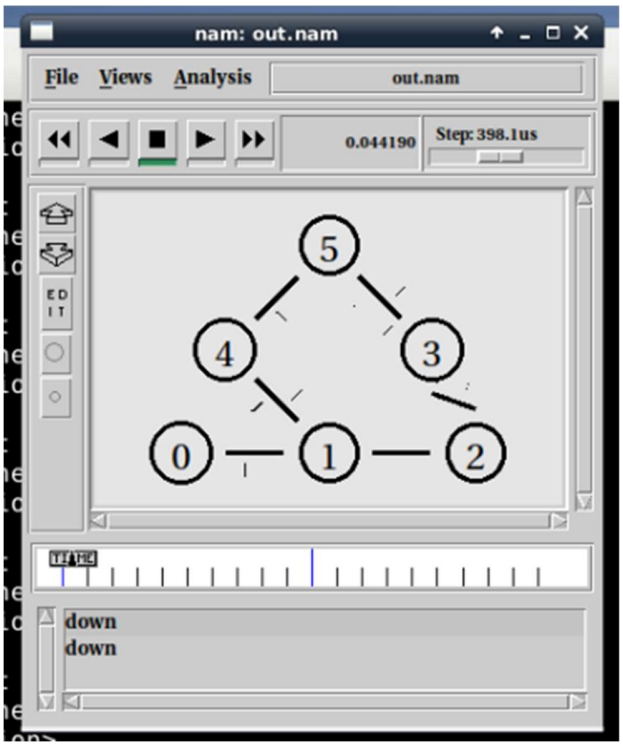
At time 1.0, the link between n1 and n4 goes down/dies. At time 1.2 it comes back up/is restored. The route does not change (because the route is statically routed, meaning it cannot adapt to changes in the network e.g. links doing down).

Question 3: Did you observe any additional traffic as compared to Step 3 above? How does the network react to the changes that take place at time 1.0 and time 1.2 now?



- w/ dynamic routing, routing adapts to changes

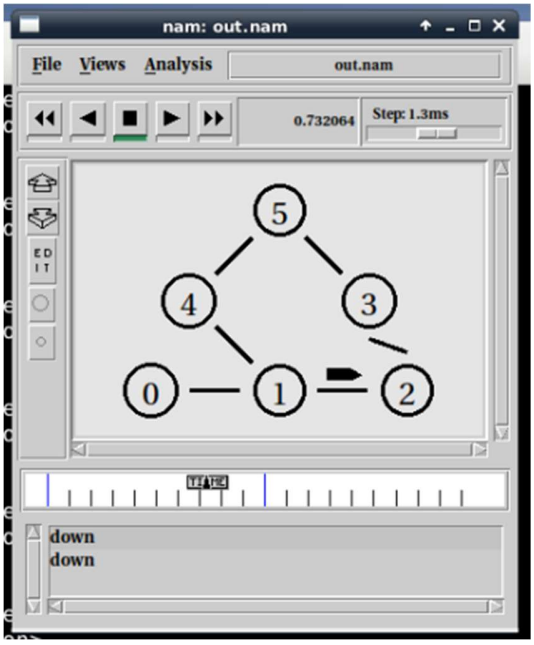
Yes, the distance vector information is distributed at the start before any of the packets are sent out.



Now the network can dynamically adapt by re-routing packets along the n1->n2 link whilst the n1->n4 link is down. It then switches back to using the n1->n4 link when it is back up. This would suggest that the n1->n4 link has a lower cost (because it is preferring to use it when it has a choice of both).

I.e. the packets travel along the paths 0 -> 1 -> 4 -> 5 and 2 -> 3 -> 5 originally, but then when the n1->n4 link goes down at 1 second, the first path is no longer used an is instead replaced with "0 -> 1 -> 2 -> 3 -> 5" (with the latter path being unaffected). This then reverts back to the original path after 1.2 seconds when the n2->n4 link is restored.

Question 4: How does this change affect the routing? Explain why.



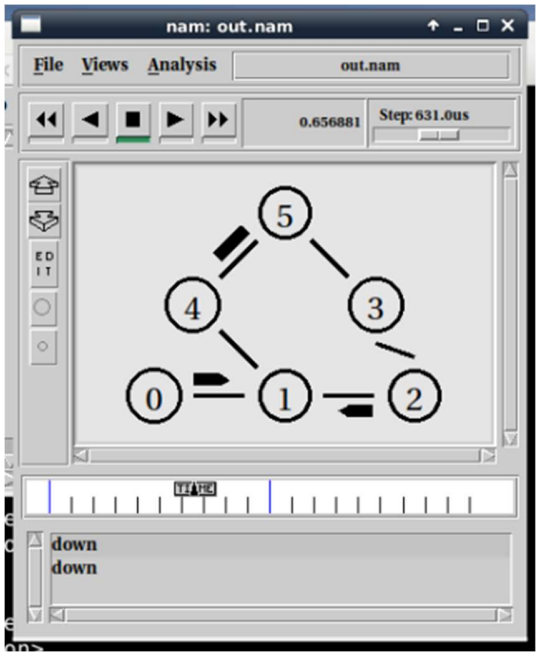
- link n1->n2 now preferred over link n1->n4

The packets sent from the UDP agent at n0 no longer use the path "0 -> 1 -> 4 -> 5" but instead "0 -> 1 -> 2 -> 3 -> 5". This is because the cost of the link n1->n4 has changed (gotten more expensive), and so the distance vector cost for routing in the latter way is less.

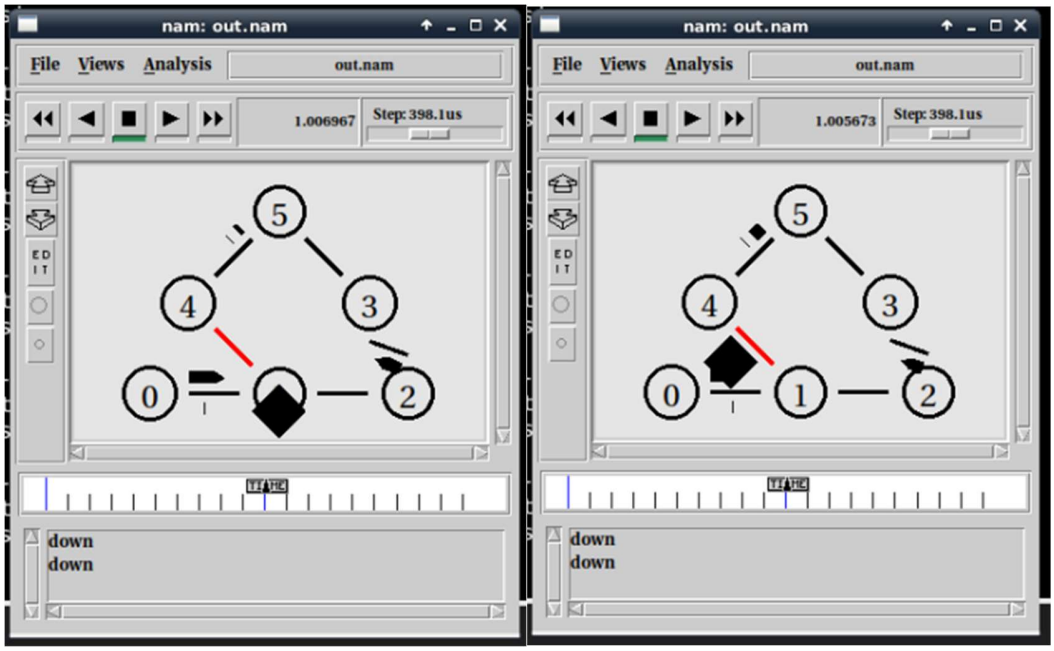
Question 5: Describe what happens and deduce the effect of the line you just uncommented.

By leaving line 29 (Node set multiPath_ 1) commented, I observe the exact same traffic flows as in question 3. This means that the new weights given to connection n1->n4 and n3->n5 still result in the same least cost paths as in q3 (evident because the same paths are taken).

However, once line 29 is uncommented, the packets sent from the UDP agent at n2 alternate between using the path 2->3->5 and 2->1->4->5 between start and 1.0 seconds (and UDP agent 1 uses its normal 0->1->4->5 path). This is because the line presumably (especially given its name) enables multi path routing, and so the packets are divided up between the two paths with the goal of spreading the load more (I don't know if this is right but I'd assume the two paths would have to be of equal cost or very close in cost otherwise it could be detrimental to split the traffic - I.e. no point splitting traffic over a super low-cost and a super high-cost link if the super low-cost link can easily handle all the traffic).



This changes when the n1->n4 link goes down at 1.0 seconds, at which point I observed some packet loss (drop), which I had not seen/noticed in the previous questions.



Sometimes it appears that the packet is dropped by n1 and other times by n4, but I'd assume this is just a timing issue w/ regards to the DV/routing/forwarding algorithms and where the packets are at the point of the link going down.

Ultimately, path 0->1->2->3->5 is used instead of path 0 -> 1 -> 4 -> 5 whilst link n1->n4 is down. It then switches back to the above alternating multi path behavior.