Lab 6
z5165555

Exercise 1:

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

The route the packets follow are as follows:
0->1->4->5
And 2->3->5
All nodes in a route communicate with one another. The routes do not 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?

At time 1.0, the link between nodes 1 and 4 fails. All incoming packets from node 0 to node 1 are not able to be forwarded to node 4 (where they can then be forwarded to what is presumably their destination - node 5). These packets could be potentially dropped at node 1 as these packets are not routed to an alternative route to get to their destination.

At time 1.2 the link between node 1 and 4 is restored. All incoming packets from node 0 to 1 are now able to be forwarded to node 4, where packets are then forwarded to node 5 (presumably the destination).

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?

Yes there is additional traffic compared to Step 3 above. This additional traffic occurs as part of the Distance-Vector routing protocol. After each node has initialised its routing table, they start exchanging their routing table with their neighbouring nodes until all routing tables converge to the same state. This can be seen in the form of smaller packets that are exchanged between neighbouring nodes at the start of the NAM simulation.

At time 1.0 when the link between nodes 1 and 4 fail, the distance vector/routing table of any node that has a shortest path route that involves the route between 1 to 4 changes, as this link no longer exists. These routing table changes are then sent to neighbouring nodes, which eventually determine through many iterative updates that the shortest distance from node 0 to 5 is through the route 0->1->2->3->5 (only possible route between 0 to 5). The packets with source 0 and destination 5 get redirected to this route.

At time 1.2 when the link is restored, this results in a change in the distance vector table/routing table of any nodes that can find a shorter path to their goal node through the link between 1 and 4. After many iterations, the distance vector tables converge to a state where it determined that the shortest path from 0 to 5 is through 1->4. Packets previously

going through 0->1->2->3->5 are now directed through 0->1->4->5 as this is a shorter route from 0 to 5.

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

This change affects the routing by redirecting packets previously sent from node 0 to node 5 through the path 0->1->4->5 to the path 0->1->2->3->5. This is presumably because the change has increased the cost of the link between 1 and 4 or 4 and 5 (alternatively, the change could have possibly decreased the weight of any of the edges in the path 1->2->3->5), meaning the least cost path to get from node 0 to 5 is no longer through 1->4 but instead through 1->2->3->5.

After the distance vector routing tables of each node converge to a common state, node 0 identifies that its least cost path to node 5 is using 2 as a successor of 1 instead of 4. This is why packets are sent from 0 to 5 using a different route 0->1->2->3->5.

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

Some packets with the source node as node 2 and the destination node as node 5 have their packets directed through the route 2->1->4->5. Some packets with the same source and destination nodes have their packets directed through the original route 2->3->5.

Based on the running of the simulation, the uncommenting of the line has allowed for packets to be routed from source to destination using multiple least cost paths which have identical cost according to the distance vector routing calculations. It is likely that these paths are of the same cost as the distance vector routing algorithm will only pick the least cost path to route a packet from source to destination and there are no updates to the distance vector routing tables after the initial convergence of the tables.

Exercise 2:

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

Between time span 6s to 8s, the flow of TCP1 is going through more nodes experiencing congestion in comparison to TCP2 - in particular, node 0 and node 1 are dealing with increased incoming traffic generated by node 7, which increases the congestion experienced by TCP1's flow. Because between times 6s-8s, TCP1 is dealing with these congested links/nodes in addition to dealing with the congested links/nodes that TCP2 is dealing with (nodes 2, 4 and 5), it has more places to drop packets due to congestion in comparison to TCP2. As a result, TCP1 starts to drop more ACKs/packets, which lowers its throughput compared to TCP2.

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

The throughput for flow tcp1 is fluctuating between time span 0.5s to 2s because of two reasons. The first reason is due to congestion control. When the source to sink path between node 0 and 5 is activated, TCP probes available bandwidth using slow start, so the throughput received by node 5 exponentially increases until a triple duplicate ACK/timeout occurs. This leads to fluctuating throughput received by node 5. The second reason is due to the delays on each link. While the TCP window is small, all packets in that window are received by the destination node while incoming packets are still travelling on the links due to delays or haven't been transmitted yet due to the delays it takes for ACKs to arrive at the source. This means that the destination node might not receive incoming packets for a period of time, which so the instantaneous throughput at these moments would be 0.

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

The maximum throughput achieved by any one flow is capped at around 1.5Mbps for two reasons. The first reason is that both if packets from both flows arrive simultaneously at node 2, these packets must jointly share the link between nodes 2 and 4, which has a maximum bandwidth of 2.5Mbps. If the combined throughput of both flows across this link approaches this bottleneck limit, node 2 will experience packet loss/timeouts. Due to TCP's congestion control mechanism, this will result in the congestion control window being set to 0 or cwnd/2, decreasing the throughput of each flow, so there is clearly an upper bound on any one flow that is approximately the bottleneck bandwidth of 2.5Mpbs/2 (number of flows sharing this link) due to this reason. The second reason is to do with the delays of the link. The maximum throughput of a single TCP flow (assuming the bottleneck link is only utilised by one flow) is also capped by the max TCP window size as well as the RTT between endpoints, as the TCP flow cannot send any more packets until it has received an ACK.

Exercise 3:

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?

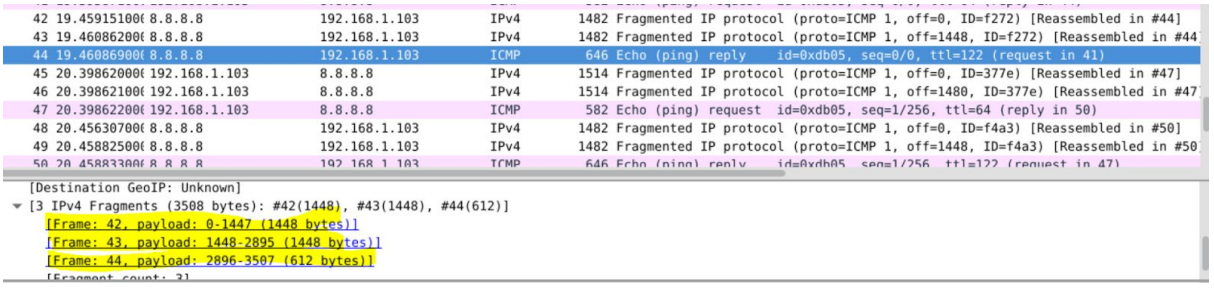
When we are pinging ICMP packets of with 2000 bytes of data this results in datagram fragmentation. The ICMP packet created has 2008 bytes totally (2000 bytes payload + 8 bytes ICMP header). When we add the IP header of 20 bytes (resulting in 2028 bytes), this exceeds Ethernet's maximum transmission unit (MTU) of 1500 bytes, so IPv4 fragments this packet into smaller fragments so it can be successfully transmitted on links from source to destination. 1472 bytes of ICMP data is the most that can be sent without causing fragmentation (1500 (MTU) - 20 (IP header) - 8 (ICMP header)).

The source that is sending a ping request (192.168.1.103) with the original datagram (the ping request) has fragmented it. The original datagram is then reassembled when it reaches the destination (8.8.8.8).

Two fragments are created when ICMP packet has 2008 bytes - one of with payload of size 1448 bytes and the second with payload of size 560 bytes.

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 the reply from the destination to the source for got fragmented because the size of the ICMP datagram being sent for the reply also exceeds the MTU of Ethernet when we add the IP header onto it. This can be seen if we observe what happens before the ping response is sent by 8.8.8.8, which shows the fragmentation of datagrams. It is also clear by observing the ping response field that the datagram being sent in the response has been fragmented at the host sending the ping response (8.8.8.8).



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?

	ID	Length (bytes)	Flag	Offset values (bytes)
Fragment 1	0x7a7b	1514	0x01 (indicates more fragments)	0
Fragment 2	0x7a7b	1514	0x01 (indicates more fragments)	1480
Fragment 3	0x7a7b	582	0x00 (last fragment)	2960

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

No, fragmentation of fragments have not occured in this case. This is because all fragments have a size that do not exceed the MTU of Ethernet, so they do not need to be further fragmented to be successfully sent on the links from source to destination.

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

As we are transmitting over UDP and UDP is not a reliable transport mechanism, unless reliability is built at the application layer, this may result in many fragmented datagrams getting lost as UDP does not guarantee that they will be successfully transmitted from source to destination. If any fragment is lost, the entire datagram will not be able to be reassembled. This means that the datagram is effectively lost as the receiver will not be able to receive the whole datagram.