Data Communications & Networking Coursework

F29DC

Ziyad Alrasbi

H00296507

Contents

[**Question 1: Lab Reports** 3](#_Toc56776243)

[Lab 1: Link Load & Bandwidth 3](#_Toc56776244)

[Lab 2: More Communication (Congestion) 4](#_Toc56776245)

[Lab 3: Creating Large Wired Topologies 5](#_Toc56776246)

[Lab 4: Stop-N-Wait, Sliding-Window & Trace File Analysis 6](#_Toc56776247)

[Lab 5: Dynamic Networks & Creating Output Files for Xgraph 7](#_Toc56776248)

[Lab 6: Ping 8](#_Toc56776249)

[Lab 7: RED Queue Management 9](#_Toc56776250)

[**Question 2: Topologies Report** 10](#_Toc56776251)

[Star Topology 10](#_Toc56776252)

[Advantages 11](#_Toc56776253)

[Disadvantages 11](#_Toc56776254)

[Mesh Topology 12](#_Toc56776255)

[Advantages 13](#_Toc56776256)

[Disadvantages 13](#_Toc56776257)

[Subnetting 14](#_Toc56776258)

[Network Traffic 14](#_Toc56776259)

[TCP 14](#_Toc56776260)

[UDP 14](#_Toc56776261)

[Ping 14](#_Toc56776262)

[**Question 3+4: Routing Protocols Report** 15](#_Toc56776263)

[Assumptions 15](#_Toc56776264)

[Routing Protocols 16](#_Toc56776265)

[Distance Vector Protocol 16](#_Toc56776266)

[Link State Protocol 16](#_Toc56776267)

[Protocol Comparison 16](#_Toc56776268)

[Comparison 17](#_Toc56776269)

[Throughput 17](#_Toc56776270)

[X Nodes 17](#_Toc56776271)

[Y Nodes 18](#_Toc56776272)

[Z Nodes 19](#_Toc56776273)

[Dropped Packets 20](#_Toc56776274)

[X Nodes 20](#_Toc56776275)

[Y Nodes 21](#_Toc56776276)

[Z Nodes 22](#_Toc56776277)

# **Question 1: Lab Reports**

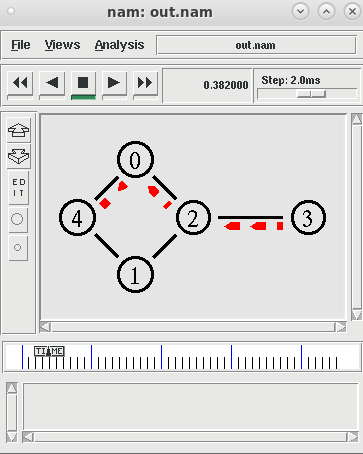
## Lab 1: Link Load & Bandwidth

The main objective of this lab was to become acquainted on how to load and use ns2, the network simulation tool that we will use for all sections of the coursework and the labs.

After setting up, we were instructed to download a tcl file which contained an example of a simple simulation, containing 4 nodes. After running the file, a simple topology was visible, and I was able to run it. After looking over the tcl file, I became familiar with the different definitions:

* **TCP:** used as a communication protocol to connect different devices that are on the internet
* **UDP:** like TCP in the sense that it allows for messages/signals to be exchanged between different devices on a network.

I added a 5th node to the topology by creating new UDP/TCP connections and I then altered the UDP connection to send data from node 3 to node 4. I witnessed that the topology chose to take the path from 3>2>0>4 instead of 3>2>1>4, which was interesting. Here is the program running:



## Lab 2: More Communication (Congestion)

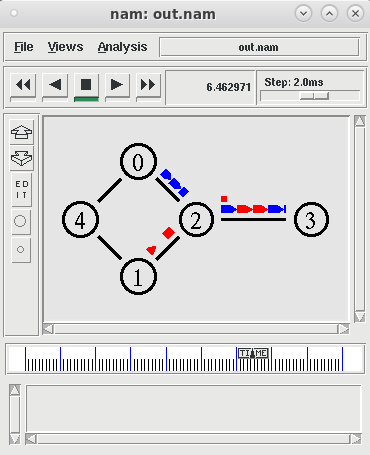
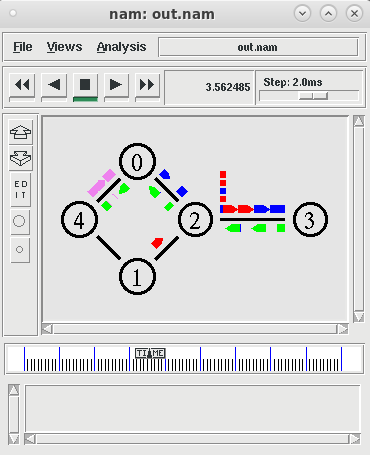
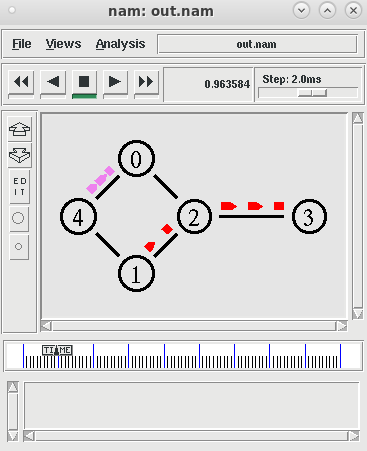
In this lab, we extended on the first lab. We started with the same network topology; however, we created more connections between nodes. It consisted of creating more UDP/TCP connections across the topology and sticking to a strict timeline on it.

From this lab, I become more familiar with:

* **Queueing theory:** the theory of queueing objects (packets in this case) and processing them in order. First in first out
* **Network Congestion:** a section of the queueing theory which happens when a node attempts to carry more information than it can hold; this results in packet loss.

I created multiple new UDP/TCP connections to allow different nodes to communicate with each other. Following the queueing theory, I also created a timeline of events that the nodes had to follow. I learnt that the topology will always follow the timeline, however congestion was seen, as the nodes were trying to handle too much data. Here are some screenshots of the topology at different times:

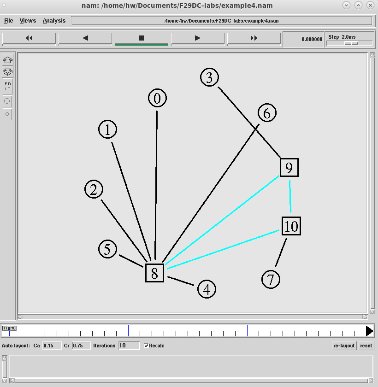
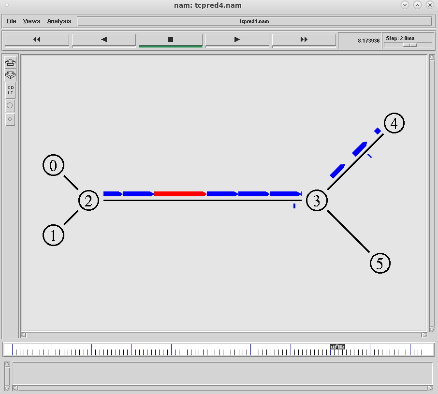
*0.9sec: 3.5sec: 6.5sec:*

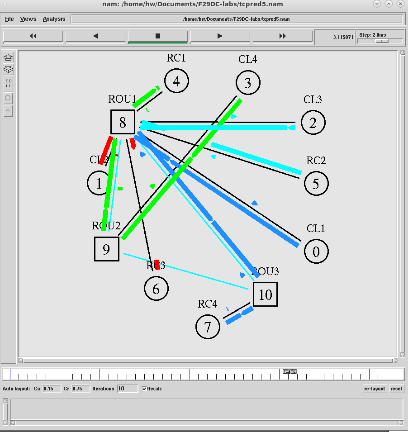
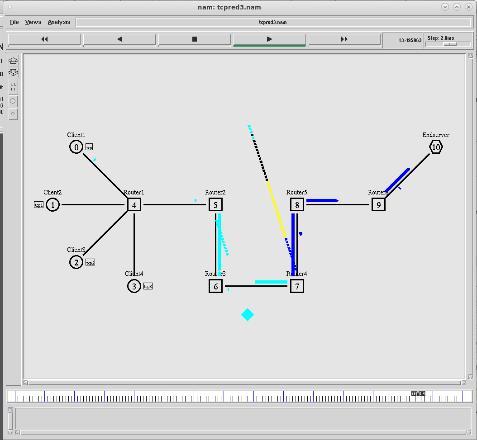


## Lab 3: Creating Large Wired Topologies

The objective of this lab was to become familiar with different network topologies and how they work. It required us to test running some example topologies and understanding how they worked. I tested on 4 different topologies, including the initial example.

* **Example\_4.tcl:** this example helped me understand colour identification for links, setting shapes and how to create a larger network topology.
* **tcpred3.tcl:** this network allowed me to understand how a network can setup a communication method between 4 clients and an endserver using TCP. It also showed how error checking is conducted.
* **tcpred4.tcl:** from this example I saw how nodes connect to each other and communicate efficiently using UDP.
* **tcpred5.tcl:** this example used a similar topology to Example\_4.tcl, however it also allowed for TCP communication between the nodes. I learnt how a network establishes connections between each node, following a timeline of events.

Here are some screenshots of each topology I tested on:



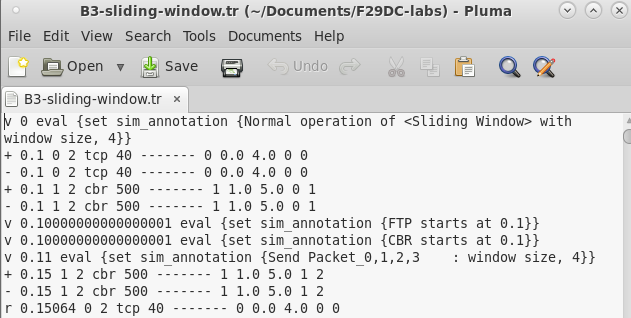
## Lab 4: Stop-N-Wait, Sliding-Window & Trace File Analysis

The purpose of this lab was to become familiar with trace files and understanding the differences between the Stop-N-Wait protocol and the Sliding Window protocol.

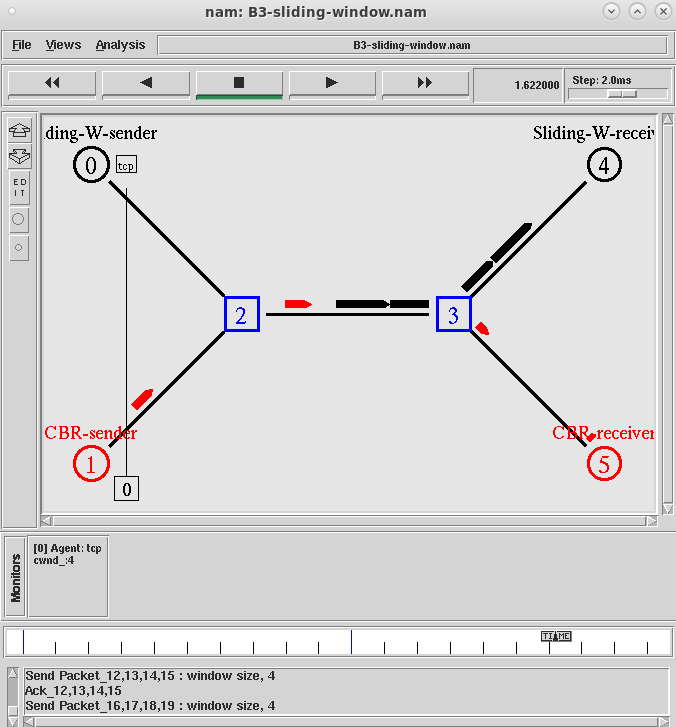
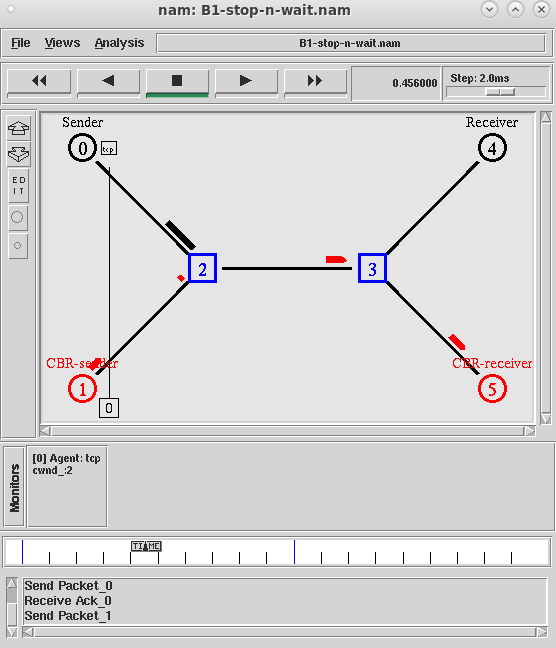
Trace files play a very important role in error checking and debugging code as it shows a timeline of events that a network takes. Inside a trace file there are different symbols that mean different things:

* **r : receive**
* **+ : enqueue**
* **- : dequeue**

Each symbol is followed by an event that happened at a certain time. For example, part of the trace file for the Sliding Window protocol was:



A single trace line shows information such as packet type, packet size, sequence number (etc.). This is extremely helpful in understanding the steps a network takes.

I also learnt about the two protocols, where I saw that the Sliding Window protocol was more efficient as it was able to handle more packets at a given time, even though congestion is seen (resulting in packet loss). Sliding Window also does not need to wait for acknowledgement after sending frames, whereas the Stop-N-Wait does. Here are the different protocols:

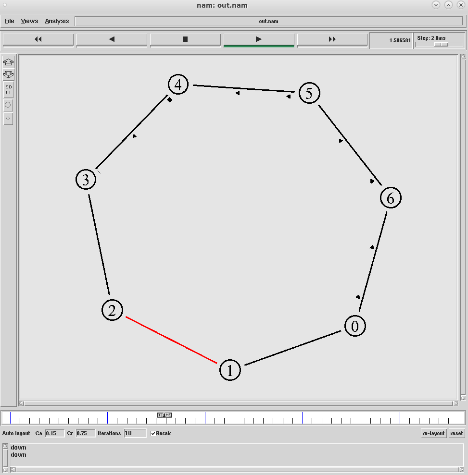
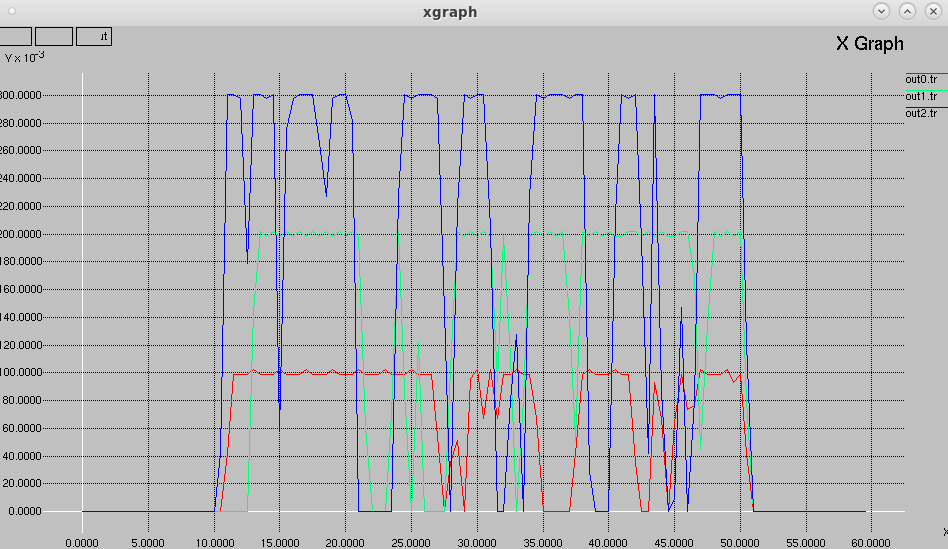
## Lab 5: Dynamic Networks & Creating Output Files for Xgraph

This lab was essential for us to become familiar with dynamic networks and analysing Xgraphs. We were given a sample file which contained a network that ran on a distance vector routing protocol, and we were then asked to change the routing protocol to link state. After testing, I learnt that:

* **Distance vector:** a routing protocol where each node calculates the distance between itself and its direct neighbours
* **Link state:** a routing protocol where instead of only sending information to its direct neighbours, each node sends information of its neighbours to all other nodes on the network
* **Traffic:** the amount of data being moved across the nodes in a network.

I also tested on Xgraph in this lab and saw that Xgraph essentially outputs a graph with different details on the different connections in a network and their statistics at different points in the scenario. I learnt that Xgraph plays a very important role in being able to analyse a network and understand what happens in it, to give space for debugging or learning how a network works.

I also tested what happens when a fault is put in place at a certain timeline. I saw that the network automatically switched to finding an alternative path to send data. After the fault was over in the timeline, the network returned to the optimal path. Here are examples:

 *Xgraph: Path with fault (node 1> node 2):*

## Lab 6: Ping

This lab was focused on understanding what ping is and how a user/computer utilises it to figure out if the connection is functioning correctly. We were given an example file which had an implementation of ping in ns2.

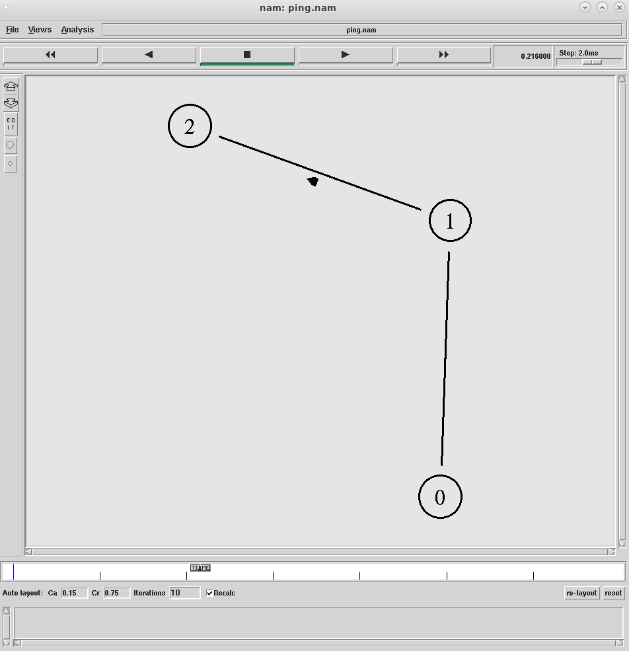
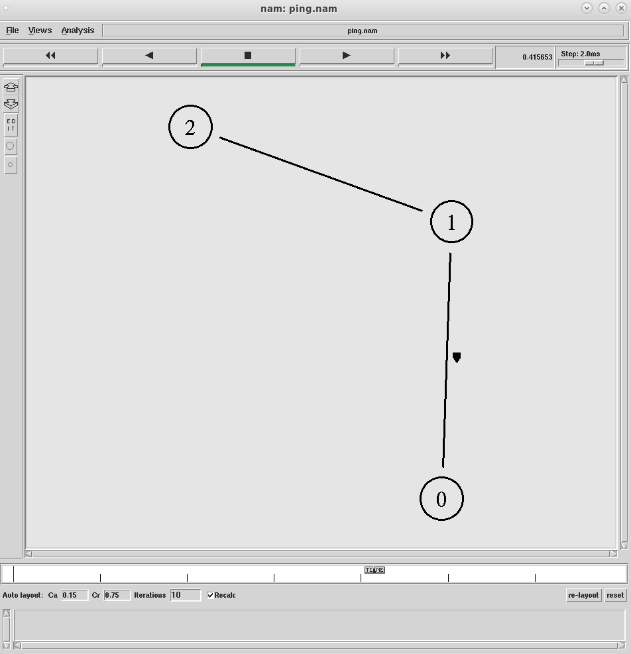
I understood that ping is a web-based algorithm that gives users the chance to check if an IP address can take requests, by checking if the IP address exists. The main sections of ping are:

* **Internet Control Message Protocol (ICMP):** a protocol that sends information to the source (user for example) about problems in the network connectivity, by sending an error message. It is a part of TCP/IP
* **Echo Request/Reply:** a query message from the ICMP which sends a message to a source and waits for a reply
* **Throughput:** the amount of data that is available to be sent from a source to the destination.

Through the ICMP, debugging, troubleshooting and Throughput calculation can all be conducted using ping. After research and reviewing the lectures, I found that RTT (round trip time) can also be used to calculate the Throughput by the calculation: Throughput = TCP buffer size/RTT.

Here are some examples of the ping program:

*From node 0 to node 2: From node 2 to node 0:*



## Lab 7: RED Queue Management

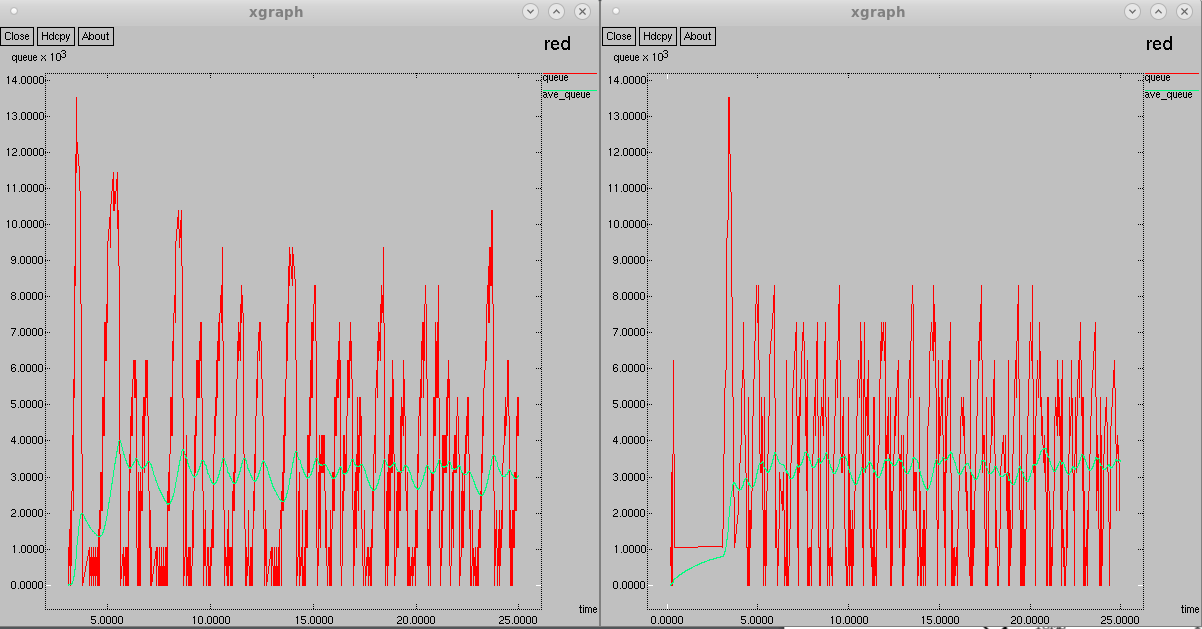
In this lab, we learnt about Random Early Detection (RED), what it does and its role in congestion control in a network.

RED is applied by many different routers for congestion avoidance. Using RED, the average queue length of a network, the max threshold and the min threshold are visible in a graph. If the average queue length exceeds the maximum threshold, the packet is discarded, and if it reaches the minimum threshold the packet will be randomly accepted or rejected.

Because of this, RED is widely used for its efficiency as congestion avoidance is something that is necessary for networks.

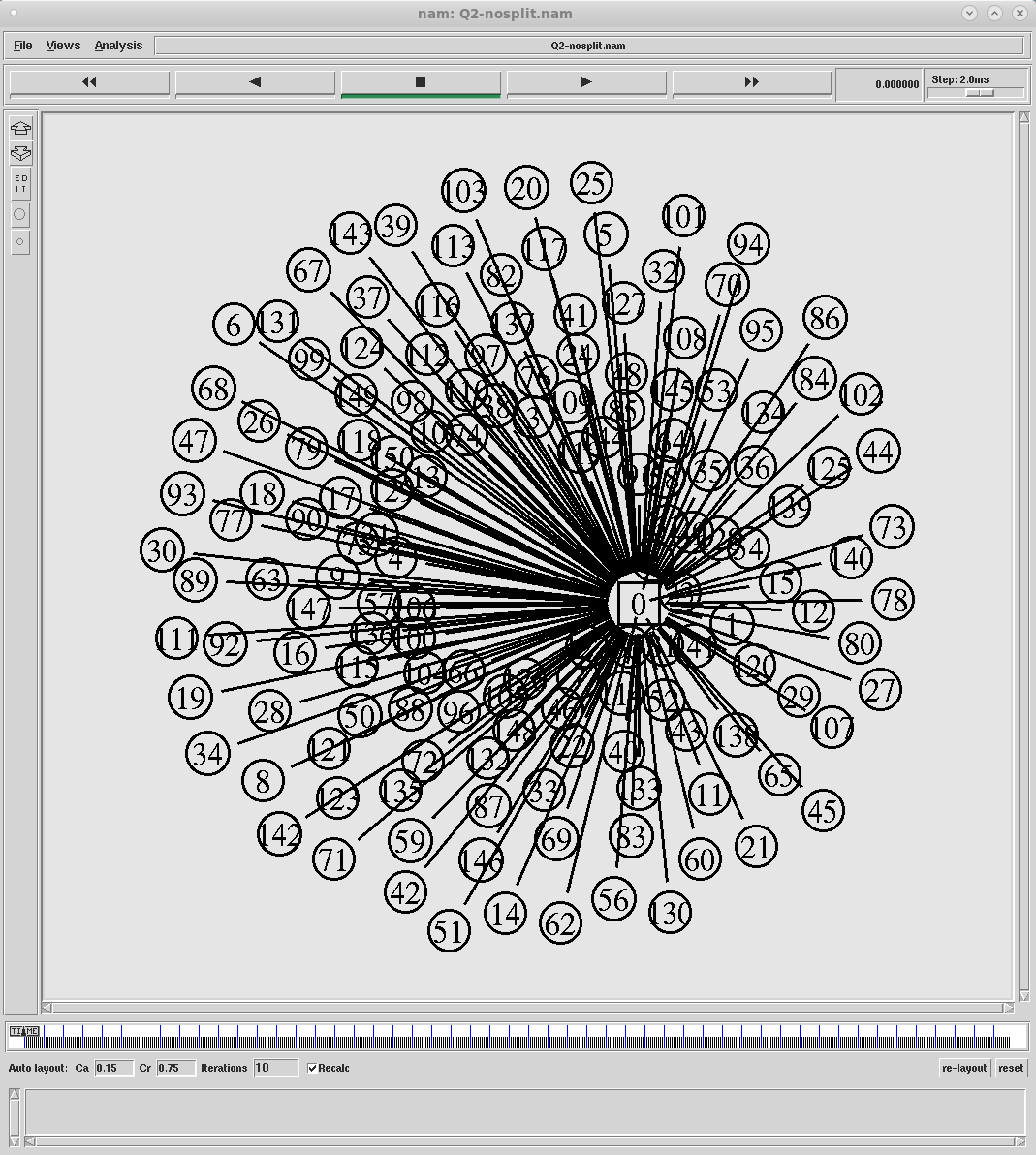
We were supplied with a program that runs RED in this lab. It outputs an Xgraph showing the average queue length on a network. I first ran the file with a low link bandwidth and analysed the output. I then altered it to a higher link bandwidth, and a visible difference was seen; on the low link bandwidth, it was peaking higher than in the one with a high bandwidth which means that it was getting closer to the maximum threshold, which would result in the packet being discarded, in comparison to the one with the high bandwidth which had a lower peak.

Increasing the scenario time also showed that the higher link bandwidth had a more stable average queue length compared to the low link bandwidth.

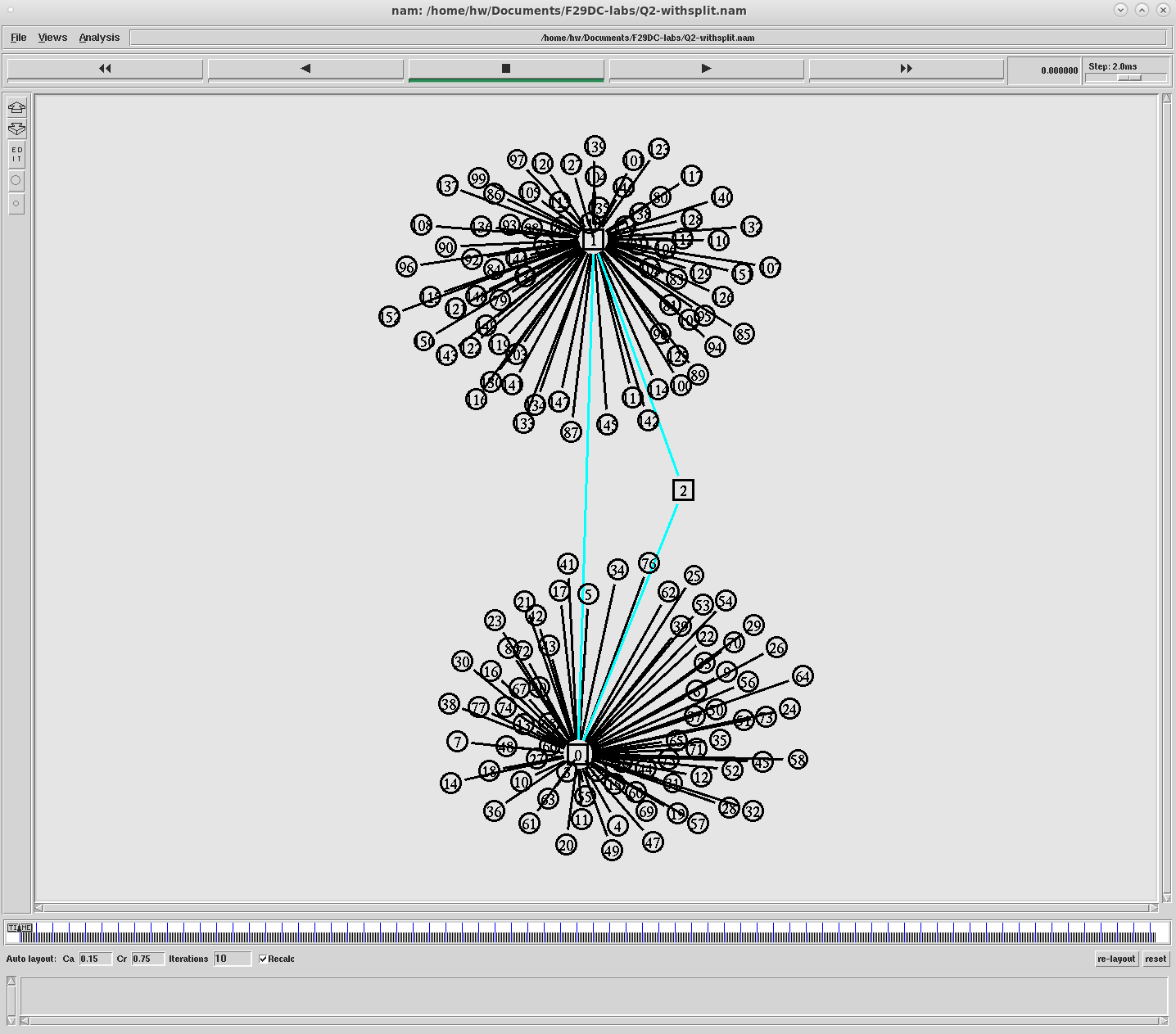
 **1MB: 100MB:**

# **Question 2: Topologies Report**

## Star Topology

**150 nodes, 3 routers**

**No Splitting:**



**With Splitting:  
(converts to tree after splitting)**

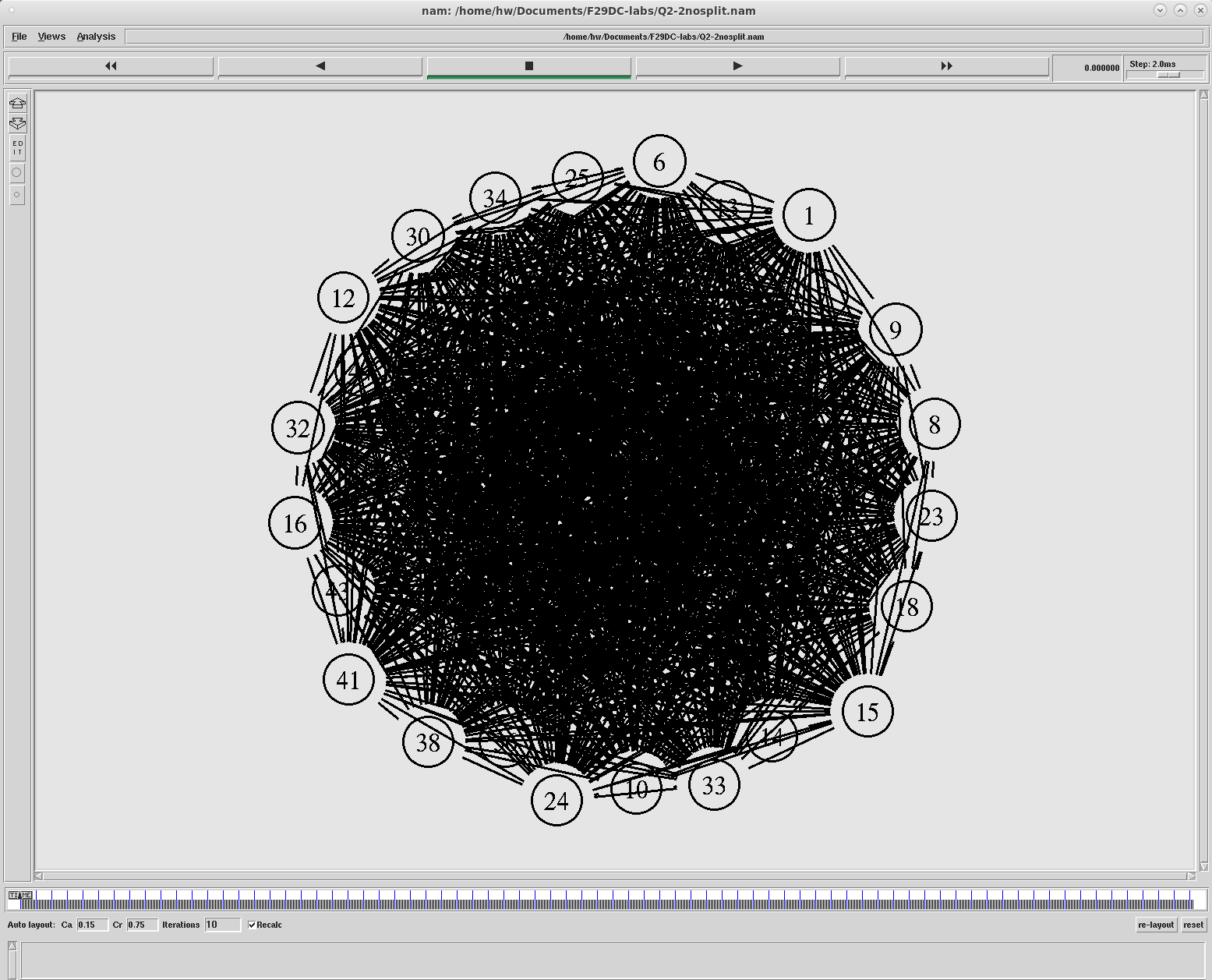
### Advantages

* Because nodes are not connected directly to each other, updating a star topology is easy as each node can be removed without affecting the other nodes in the network
* Similarly, if a node was to fail in the network, it would not affect the other nodes as they are not directly connected to each other. This also allows for easy error detection
* Because all nodes are connected to one router, maintaining and managing the network is made simpler in comparison to other topologies, as the one router can be used to manage all nodes.

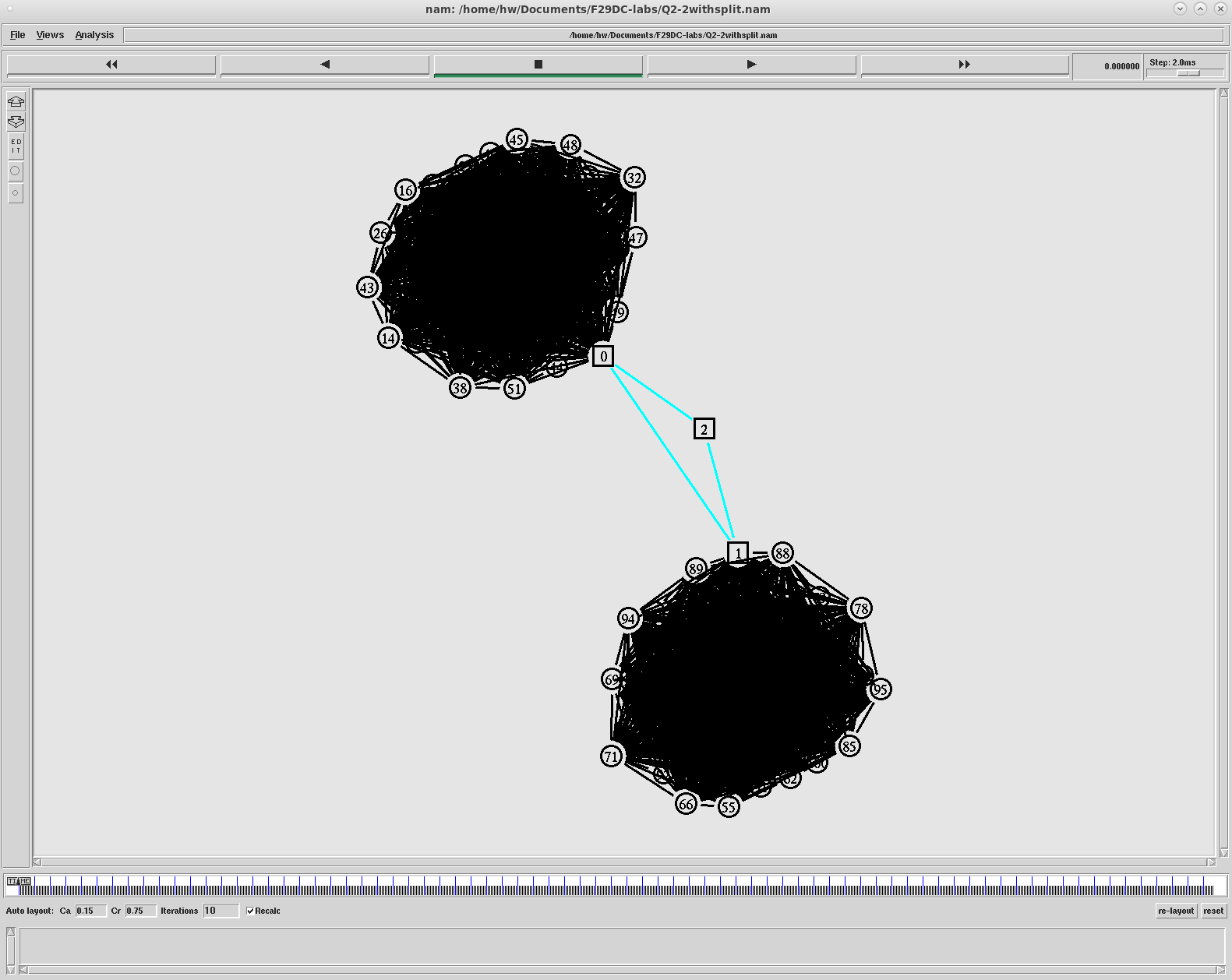
### Disadvantages

* Since each node is independently connected to the router, each node will require its own cable to connect to the router which leads in increased costs in comparison to other topologies such as a bus which is linear
* Because all nodes are connected to one router, if the router was to fail then all nodes will not have a means of communication
* In addition to the router failing, nodes are dependent on the router being capable to handle all their transmissions as they are all connected to the same router.

## Mesh Topology

**100 nodes, 3 routers**

**No splitting:**



**With splitting:**

### Advantages

* Like the star topology, if a node was to fail in the network then there will be no disruptions to the other nodes
* Updating a mesh topology is simple as the other nodes in the network will not be affected if a node was to be added/removed
* Allows for large amount of traffic to be distributed as each node is interconnected, so data can be transferred concurrently. This makes it a very efficient topology
* In addition to the amount of data, the speed of data transfer is high because of the interconnections.

### Disadvantages

* Because of the amount of wiring, it is a very costly topology as the many interconnections must mean that each node must be connected to all other nodes (in a full mesh topology)
* The amount of connections also means that maintaining a mesh topology is more difficult in comparison to other topologies
* There is a higher chance of a failure occurring due to the amount of connections in the one topology.

## Subnetting

In an IP network, subnetting is the process of subdividing a network logically, which creates multiple networks that originate from a single address block. This same process can also be done using physical splitting.

In a large network, network strain can occur as nodes that are not relevant to a certain action may become involved as they are all connected. Subnetting allows for splitting networks so that even though they are from the same address block, they are split accordingly which reduces the strain on a network.

Managing a network that uses subnetting is much simpler than managing a whole network as it allows for setting host limits in a subnetwork. This also means that subnetting increases the security of a network as it easy to manage incoming traffic, using means such as Quality of Service (QoS).

Network congestion is also decreased as there is less traffic flow in a particular network, which increases its efficiency. Unnecessary information is also not spread to other nodes since they are split, which reduces the chance of congestion occurring.

## Network Traffic

### TCP

TCP is an example of a connection-oriented protocol, which checks if a message has been received after a connection has been made. TCP also performs both error checking and error recovery when an error occurs, making it very reliable. This also means that TCP is heavier/slower in comparison to some other protocols such as UDP. TCP is used in many large applications such as email, world wide web and FTP.

### UDP

UDP is an example of a connectionless protocol, which sends a message but does not check if it has been received. This makes it less reliable than TCP and other connection-oriented protocols as it does not perform error recovery; it simply discards a packet. UDP, however, is faster than TCP as it has less actions associated to it.

### Ping

Ping is an important aspect of network traffic, and I discussed it further in lab 6 (refer to page 8).

# **Question 3+4: Routing Protocols Report**

## Assumptions

Assume same bandwidth size and same TCP/UDP/CBR connections for all, except different connection timings for Z Nodes due to memory problems caused by a large network.

I also attempted testing for End to End Delay, however because of the same memory problems I was not able to conduct a study on it.

**X Nodes:**

|  |  |
| --- | --- |
| **Scenario Time** | 100 seconds |
| **Node Amount** | 50 |
| **Link Loss 1** | 1.0 – 5.0 |
| **Link Loss 2** | 10.0 – 15.0 |
| **Link Loss 3** | 50.0 – 60.0 |

**Y Nodes:**

|  |  |
| --- | --- |
| **Scenario Time** | 100 seconds |
| **Node Amount** | 80 |
| **Link Loss 1** | 1.0 – 5.0 |
| **Link Loss 2** | 10.0 – 15.0 |
| **Link Loss 3** | 50.0 – 60.0 |

**Z Nodes:**

|  |  |
| --- | --- |
| **Scenario Time** | 70 seconds |
| **Node Amount** | 150 |
| **Link Loss 1** | 5.0 – 7.0 |
| **Link Loss 2** | 10.0 – 15.0 |
| **Link Loss 3** | 50.0 – 60.0 |

## Routing Protocols

### Distance Vector Protocol

The Distance Vector (DV) Protocol is a dynamic routing protocol which makes use of the Bellman Ford Equations to create routing tables. In DV, each specific node calculates the distance from itself to each of its neighbouring nodes. The routing table is then updated with information about the network that it receives from the nodes.

In DV, there is usually less traffic in comparison with link state. This means that the bandwidth requirement is less because packets are smaller in size. However, spreading bad information in DV usually takes longer than sharing good information, which could cause problems in networks of a large size.

### Link State Protocol

The Link State (LS) Protocol is another example of a dynamic routing protocol that uses the Dijkstra Algorithm to create routing tables. Unlike DV, in LS each node shares all its information about all its neighbours with all other nodes in a network. After that, the routing table is updated with this information.

In comparison with DV, LS has a higher bandwidth requirement as LS is more often associated with sending larger sized packets, which also means that that it has higher bandwidth requirement. Traffic in LS also tends to be heavier because of this. However, the spread of news in a network is much faster than in DV.

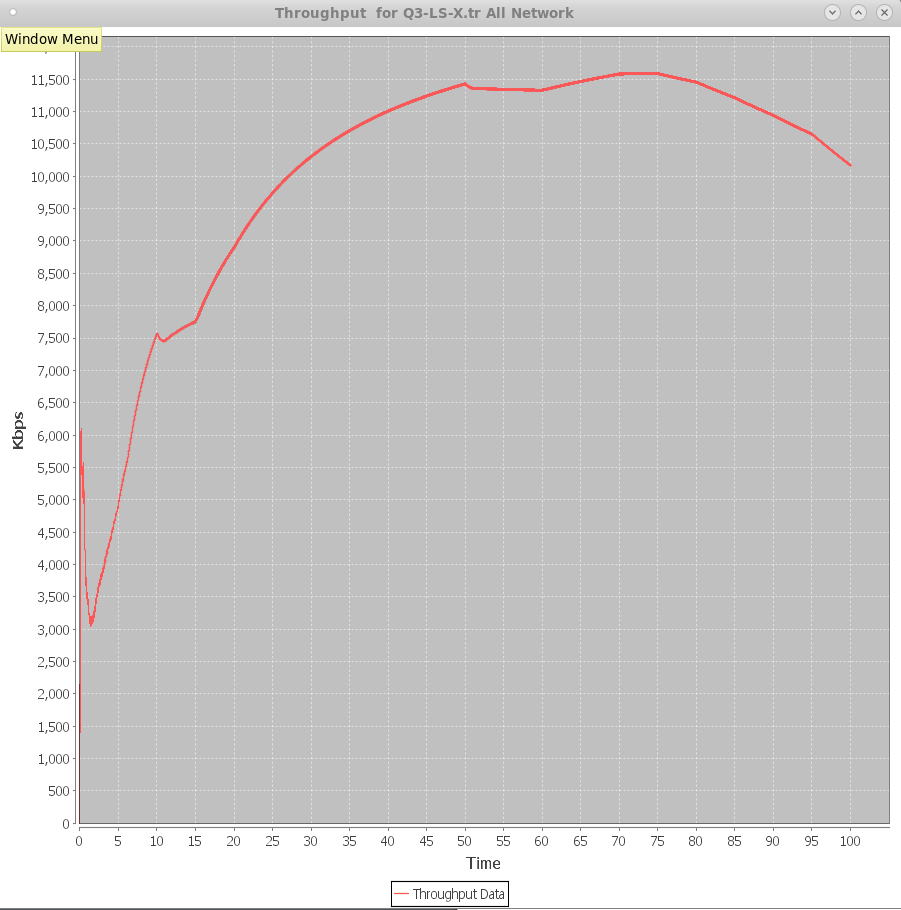
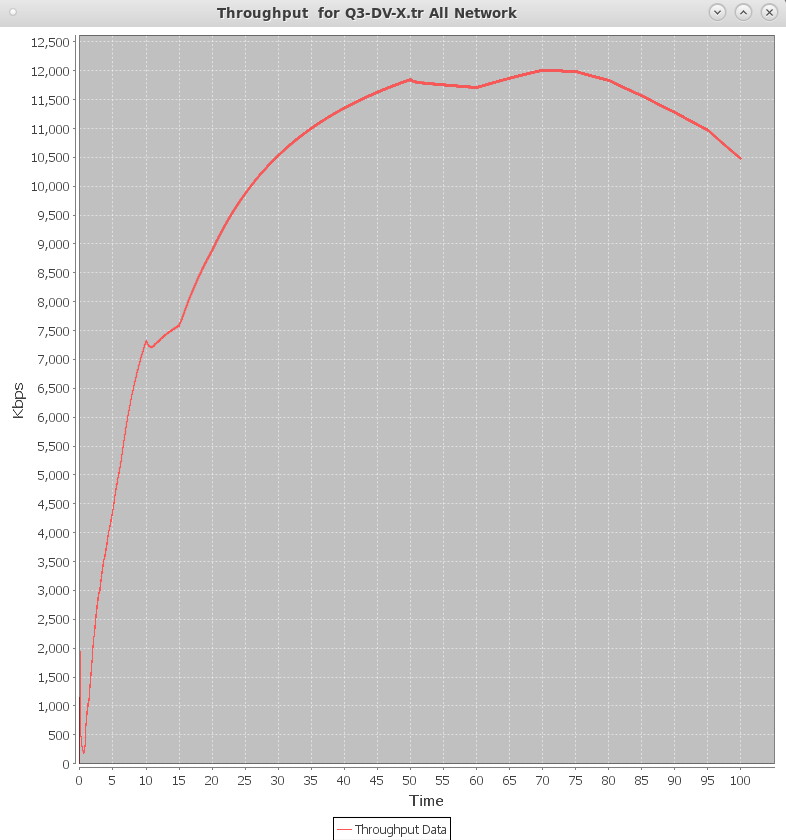
### Protocol Comparison

After my own testing, I saw that the distance vector routing protocol had less traffic in comparison to the link state; however, the link state nodes had information of the entire network, which makes it more reliable than distance vector routing when it comes to networks of a big size. I understood that the link state routing protocol also uses the Dijkstra Algorithm which increases its efficiency in finding the best path for sending data between nodes.

## Comparison

### Throughput

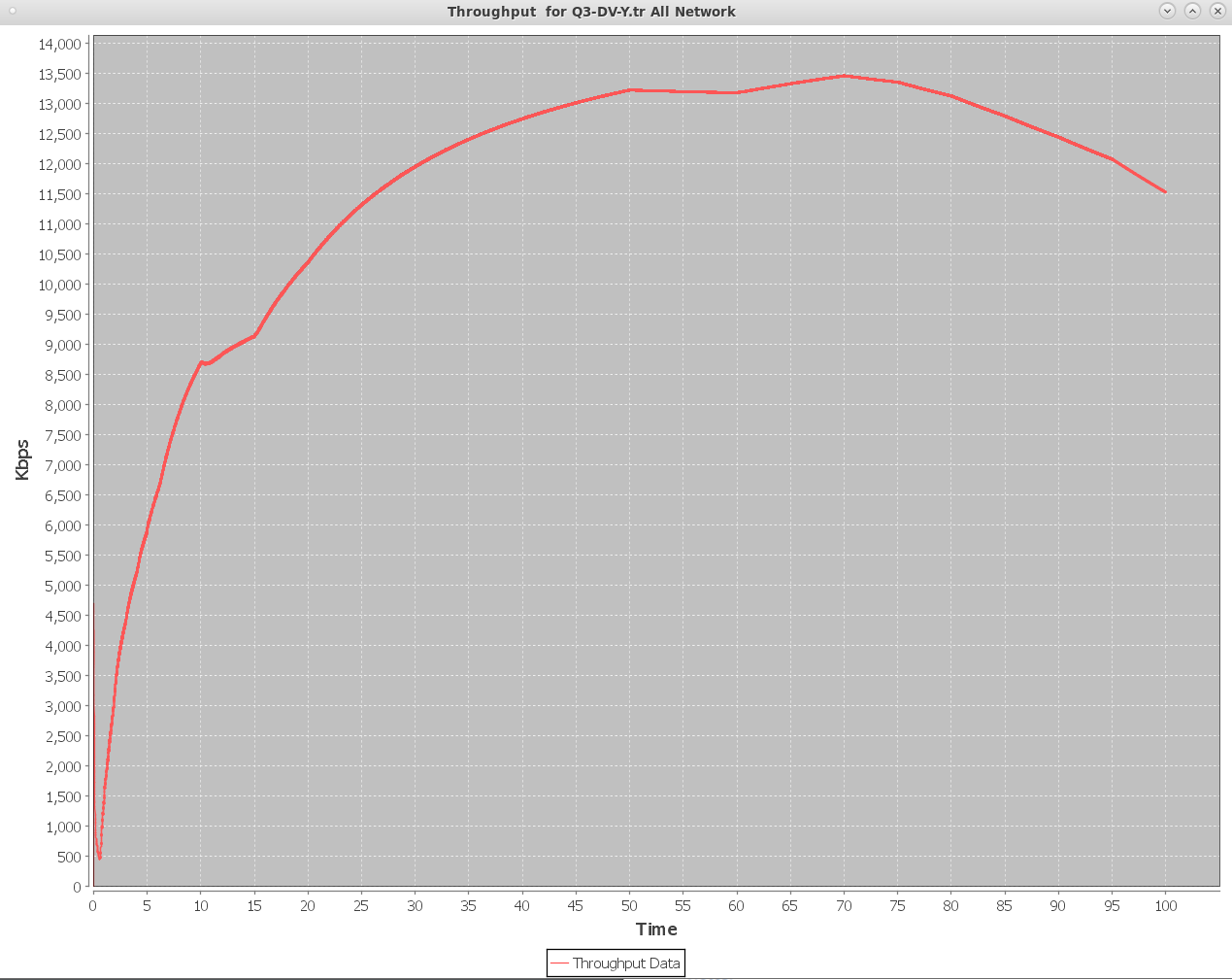
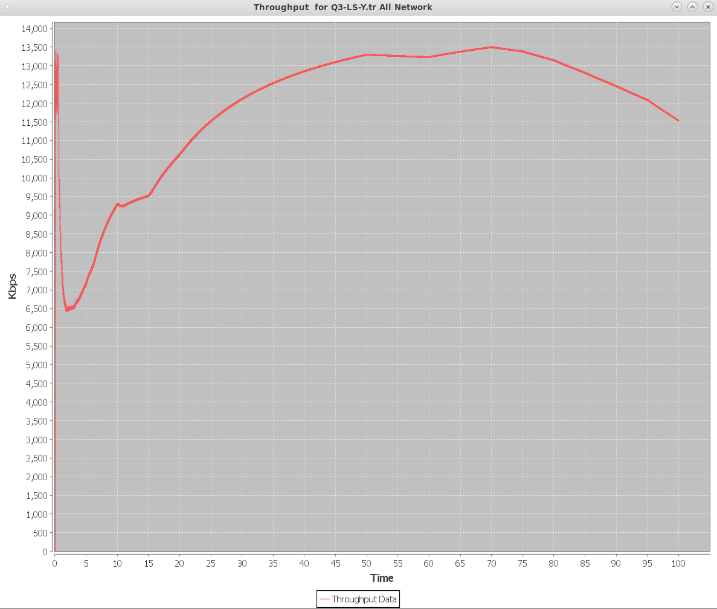
#### X Nodes

 **Link State: Distance Vector:**

Looking at the graphs above, it is seen that the LS had a slightly lower throughput peak than the DV, however the throughput was consistent throughout both networks. The LS however had a larger spike at the start of the scenario than the DV because of the distributing of knowledge for each node throughout the network.

Overall, this means that in a smaller network size, DV was more reliable at successfully sending more packets than LS.

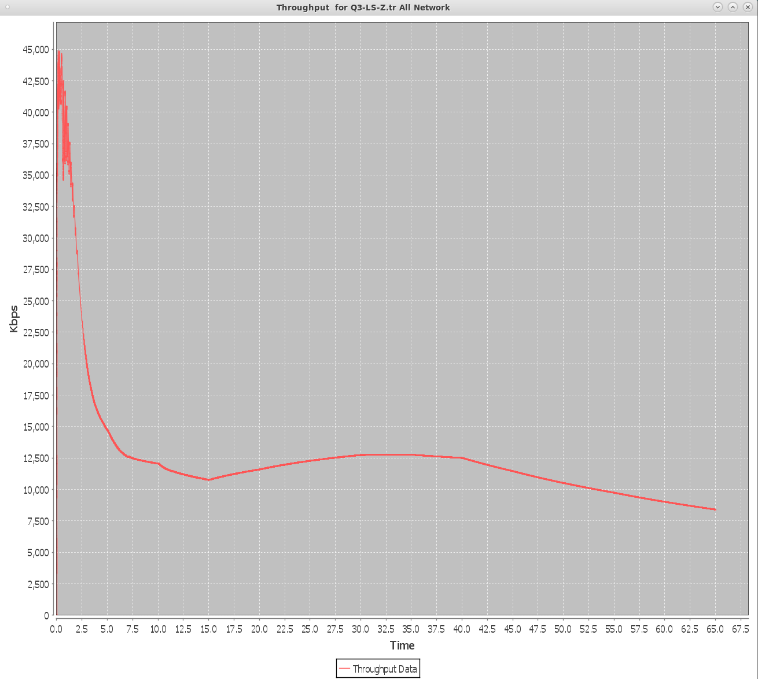
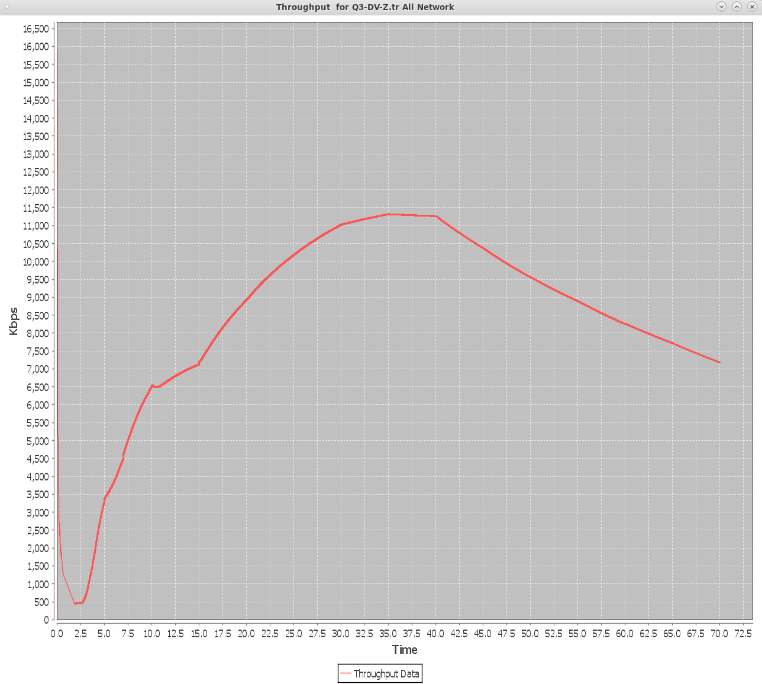
#### Y Nodes

 **Link State: Distance Vector:**

In these graphs, it can be seen that both routing protocols had a very close peak; however, both routing protocols were, again, consistent with throughput and no sudden changes were seen. Because it is now a larger network, more nodes require more information so there is a much larger spike in bandwidth at the start of the scenario in LS compared to DV.

Overall, in a medium sized network both routing protocols are equally reliable at successfully sending packets.

#### Z Nodes

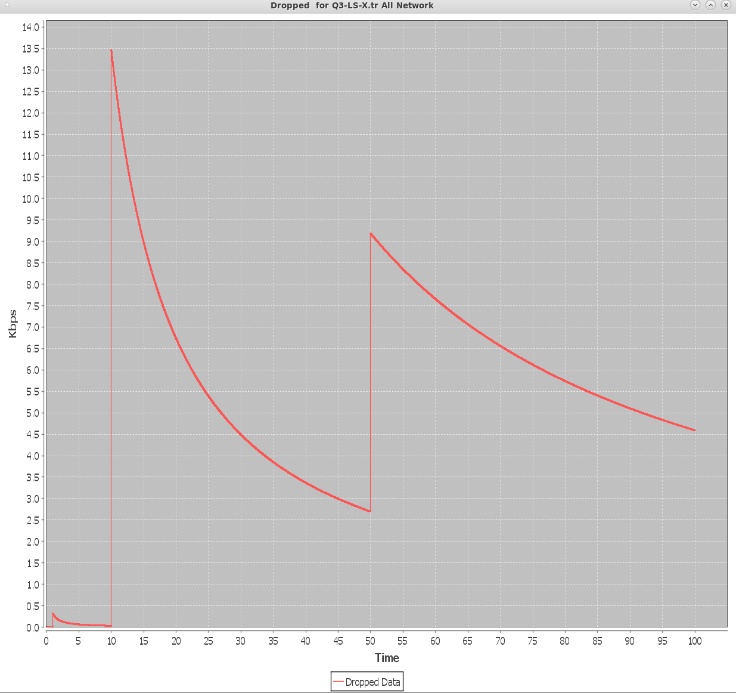
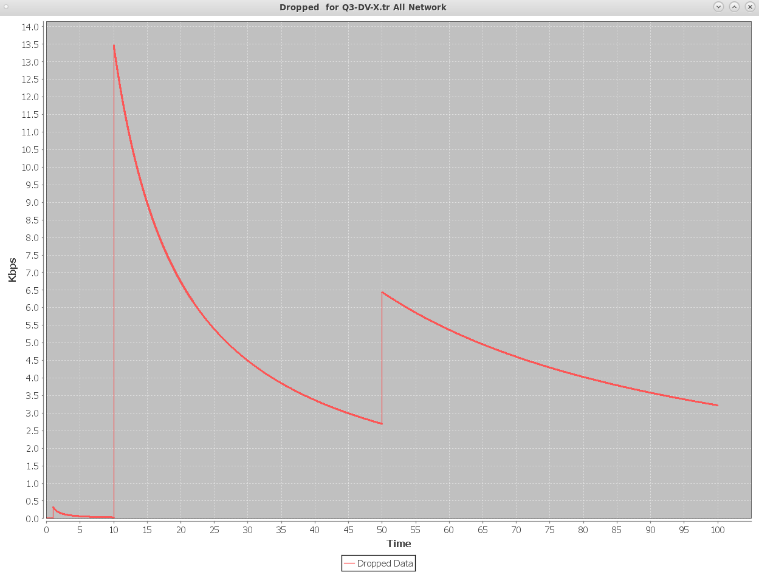
  
 **Link State: Distance Vector:**

In the large network, a bigger difference is seen between the protocols. LS had a considerably higher peak than DV. However, due to the network size an extremely large spike is seen at the start of the LS scenario because of the amount of nodes that require information about other nodes. This shows that in LS, more bandwidth is required as the size of the network expands.

To conclude, in a larger network LS is much more reliable than DV at being able to transfer packets because it is a routing protocol that is more suited for networks that are large in size.

### Dropped Packets

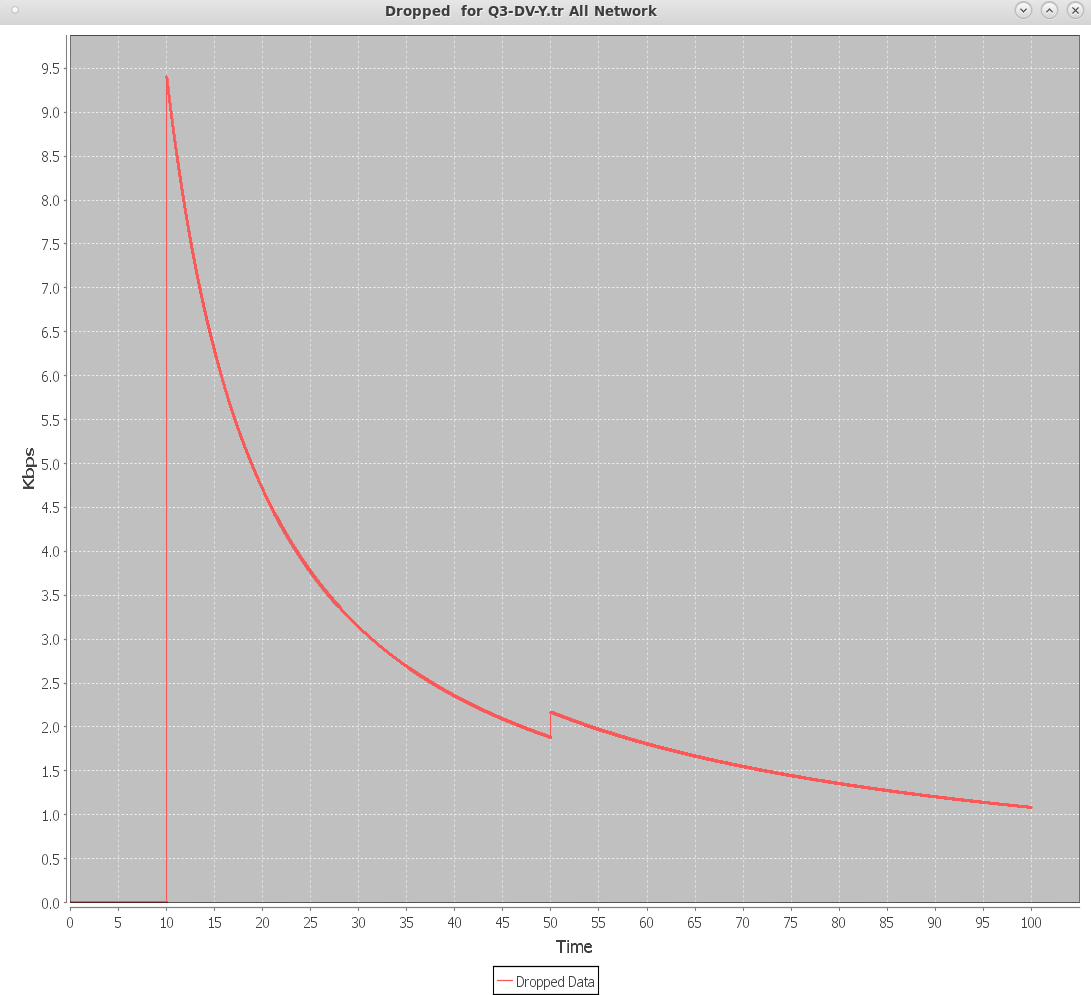
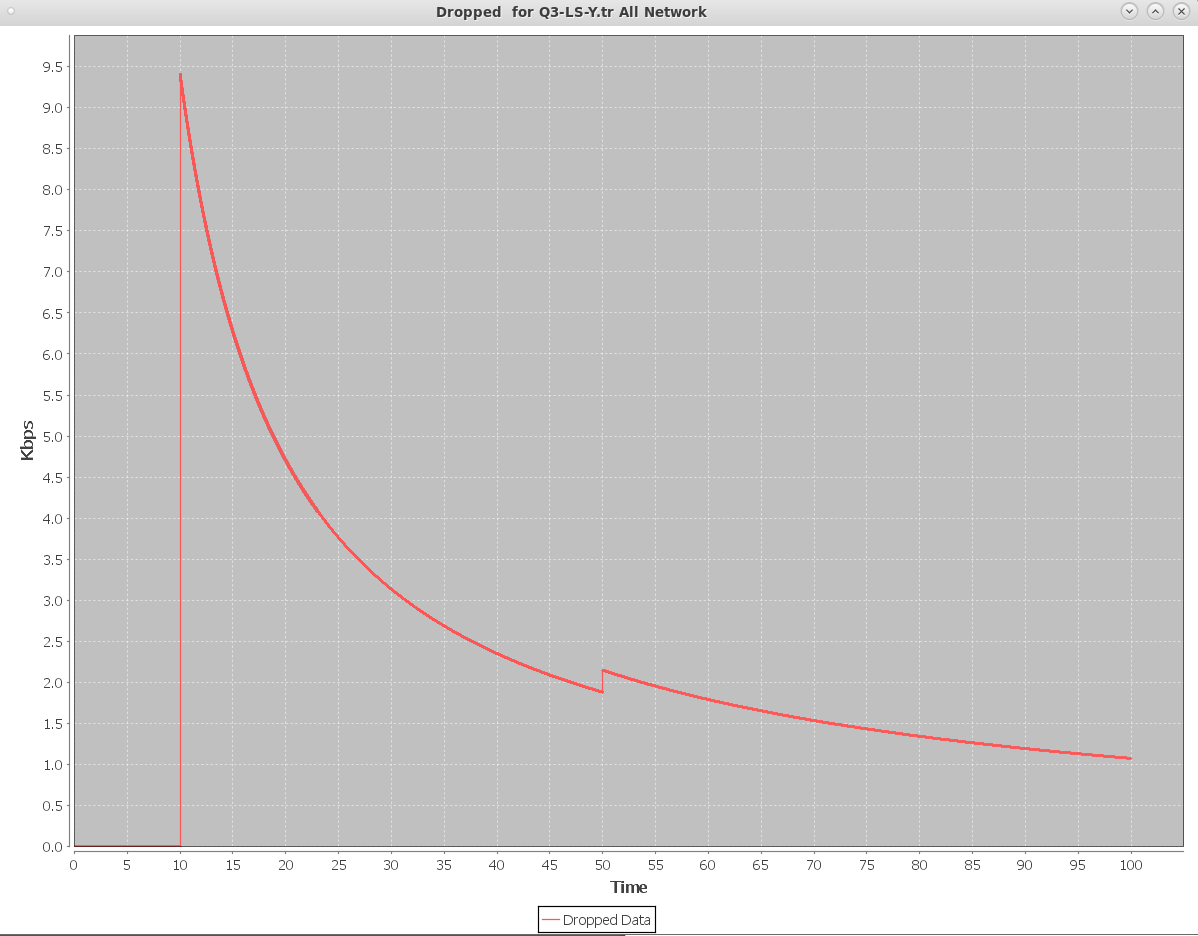
#### X Nodes

  
 **Link State: Distance Vector:**

When it comes to dropped packets, both the LS and the DV suffered from the same amount of packet drop at the first and second lost link occurrence. However, what is interesting to see is that at the third link loss occurrence the DV suffered from less packet drop in comparison to the LS.

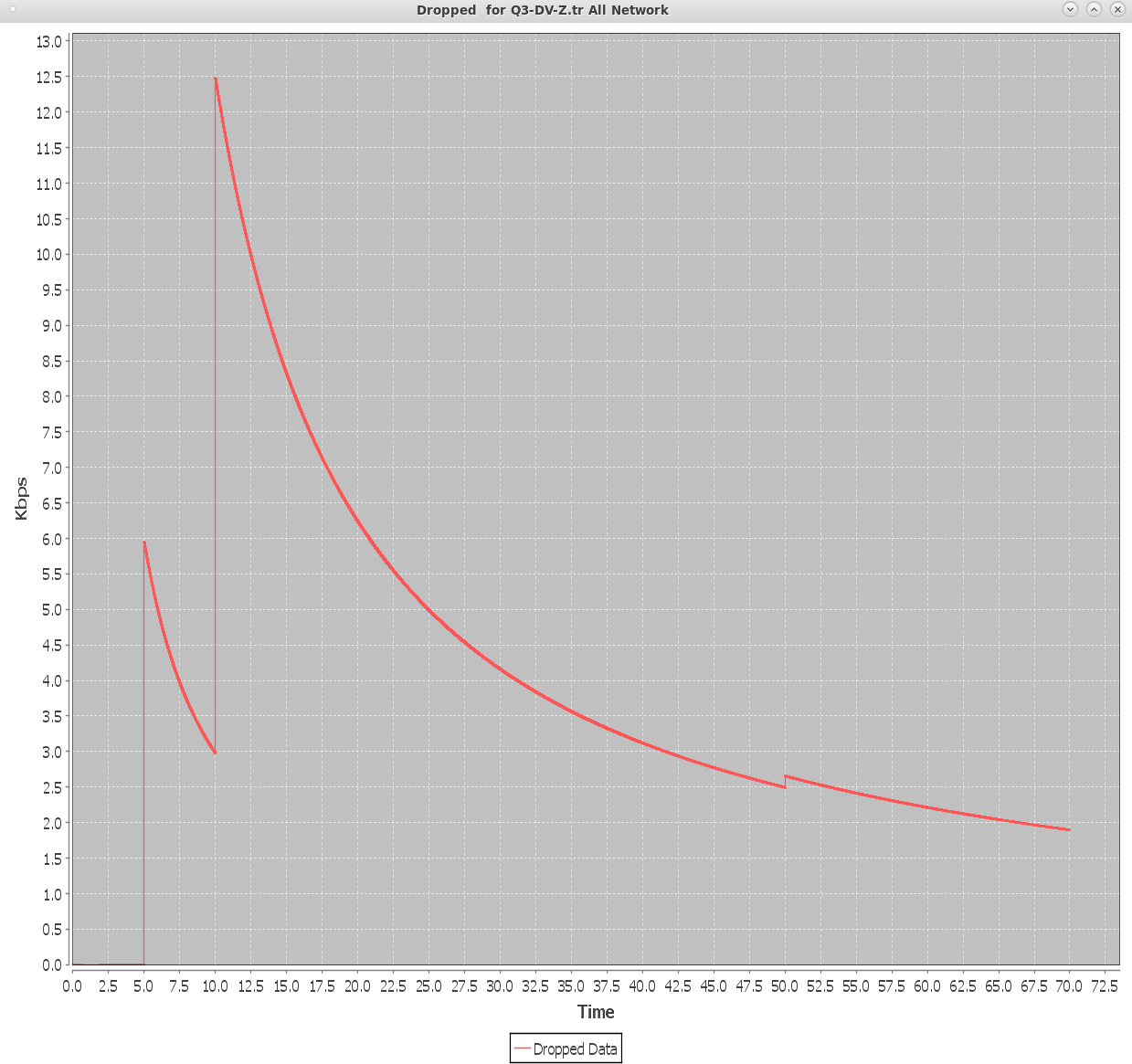
This shows that DV is more reliable at handling smaller networks.

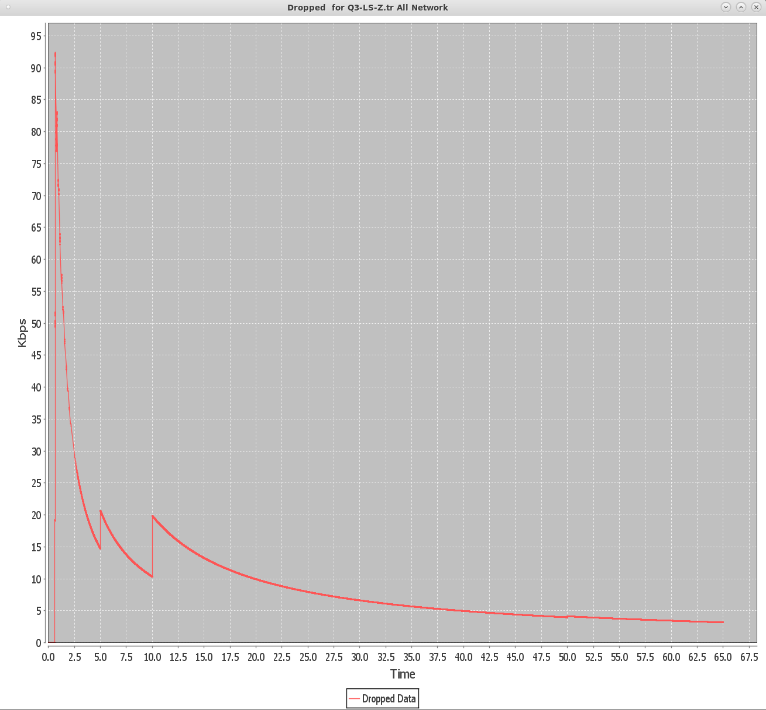
#### Y Nodes

  
 **Link State: Distance Vector:**

In a medium network size, both routing protocols had an equal amount of packet drop at each of the link loss occurrences. This is interesting to see as it means that when it comes to a medium network size, both routing protocols are equally as efficient in handling data.

#### Z Nodes

  
 **Link State: Distance Vector:**



Because both networks are running on the exact same bandwidth size (for accurate comparison), it is seen that at the very start of the scenario in the LS a huge amount of packet drop happens, due to the fact that LS requires a much bigger bandwidth as the size of the network expands. Throughout the scenario, the LS network still suffered from a much larger packet drop in each of the link loss occurrences because the bandwidth size is not enough for it to handle this amount of information.

Although this may make it seem like DV is more efficient, LS is much more reliable than DV in large networks, however it requires a much higher bandwidth to handle all the information.