**Ns2 Simulation Report**

Team 23

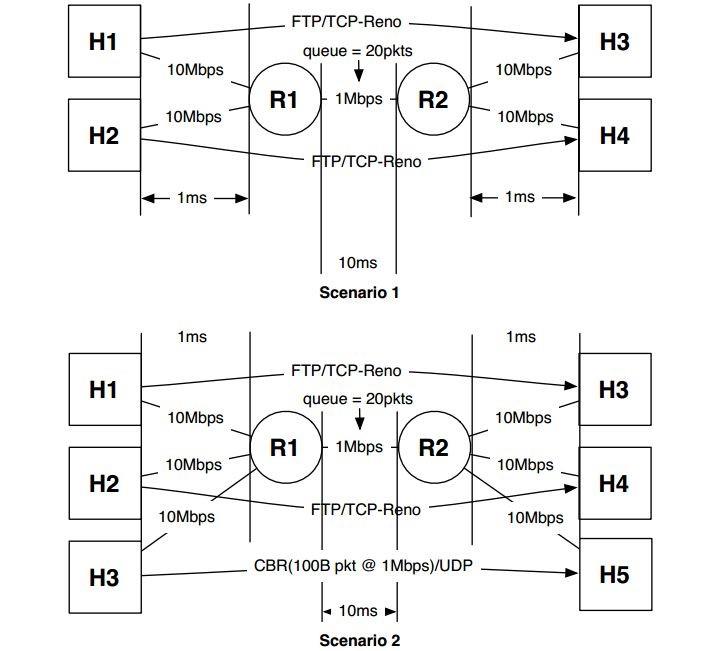
Kranthi Kumar Katam (UIN: 225009204)

Mathew Roe (UIN: 321007055)

Jifang Mu (UIN: 821005313)

**1. Description**

In this assignment, we use NS-2 simulator to configure and simulate following two topologies



We have implemented two buffer management strategies namely DropTail and RED. RED parameters used in the assignment are threshold value is 10, max threshold of the queue is 15 and linterm is 50. We have implemented TCP Reno at the source and TCP sink at the receiver end and Loss monitor for receiver with UDP connection.

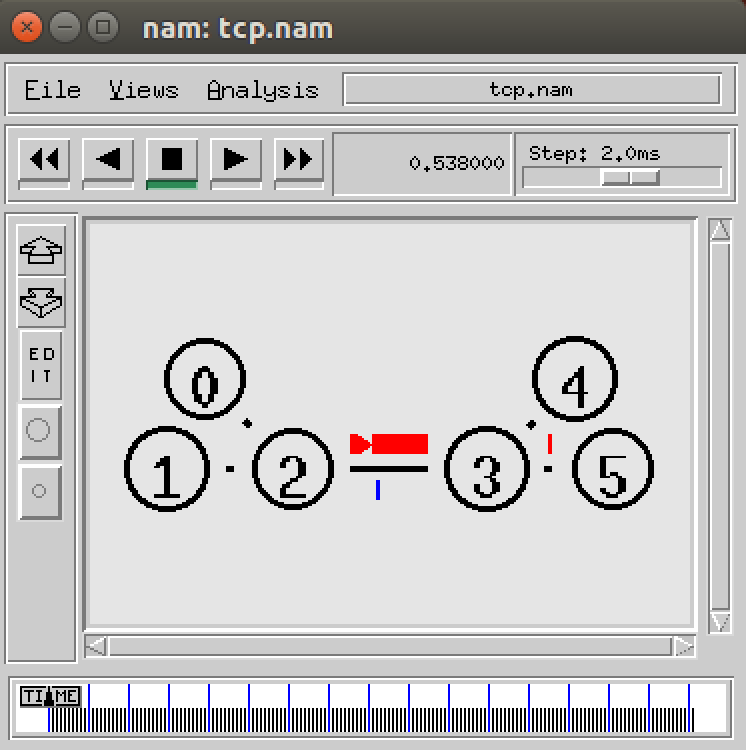
**2. Procedure**

First, create a new network simulator, and then create the network topologies as given in the description. Take input from the user command line such as Buffer management strategy and Scenario number. Set the buffer management given by the user, for RED use the additional strategies as given in the description. Setup TCP Reno and sinks for TCP connections, UDP and loss monitor for the UDP connection in scenario 2 as shown. Implement FTP over TCP connection sources and CBR over UDP source.

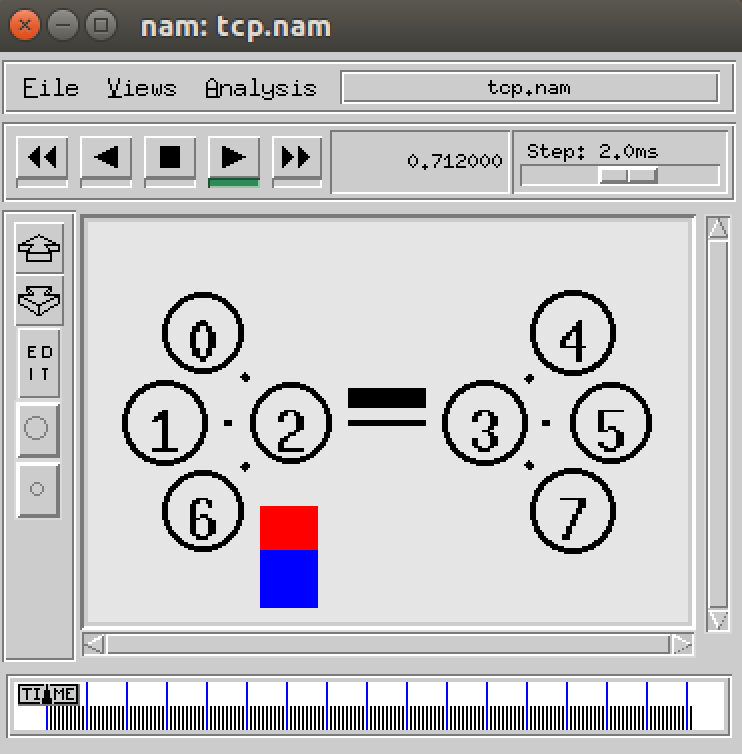
While simulating start at 0 and run up to initial value 30ms and then record process will start to create a file to write the instantaneous throughput of each source. At 180ms the process will end by calling a finish function. Then execute NAM file pops out to analyze the process and a graph will be generated using xgraph, which will give us the output of instantaneous throughputs of the sources.

The two scenarios generated by the NAM function are

* Scenario 1

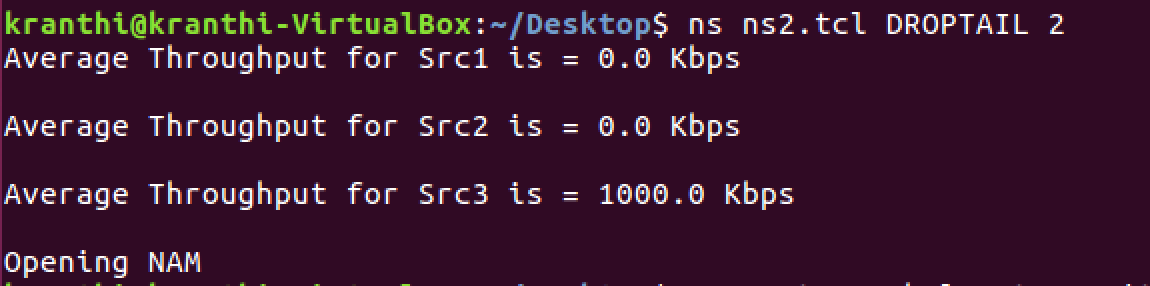
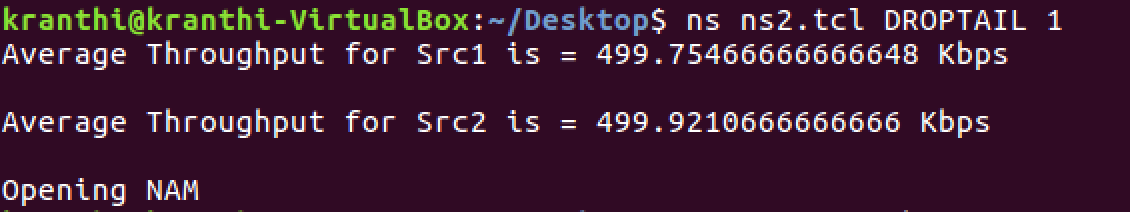


* Scenario 2

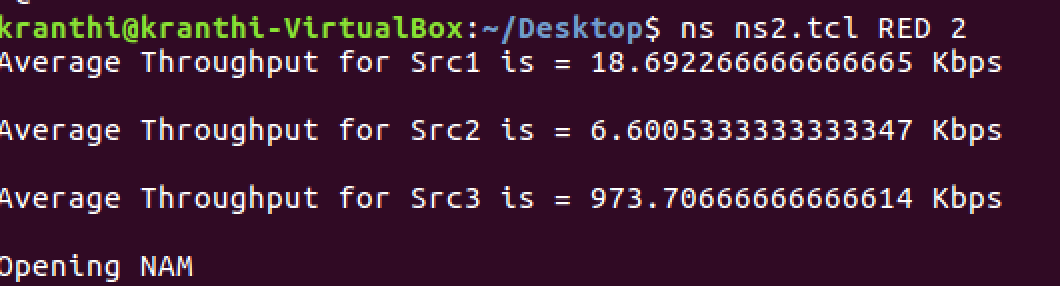
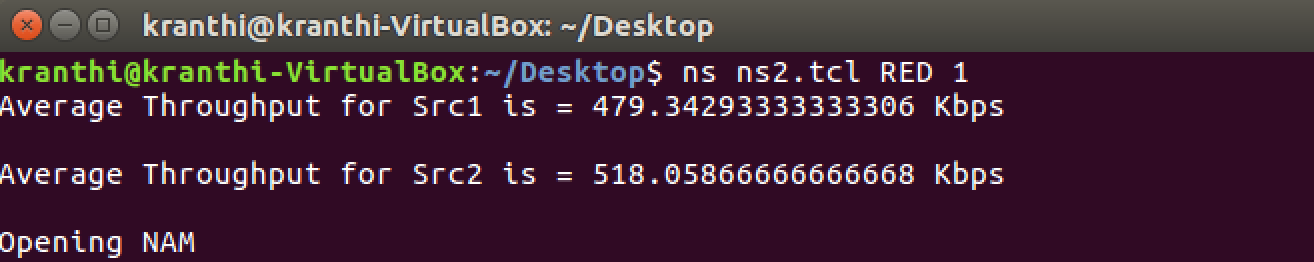


The average throughput for different sources can be seen in the command window.

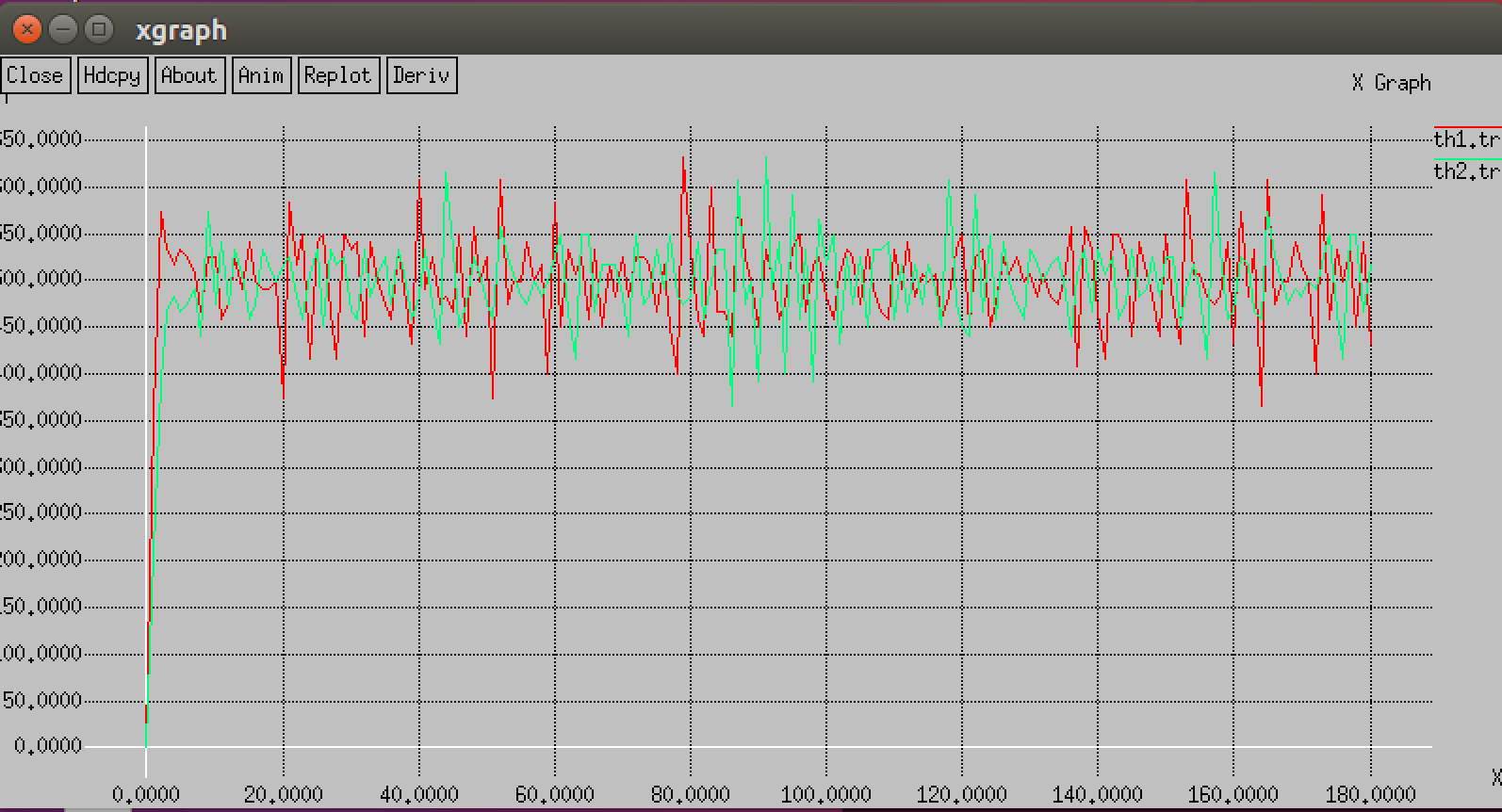
* DropTail management for the two scenarios

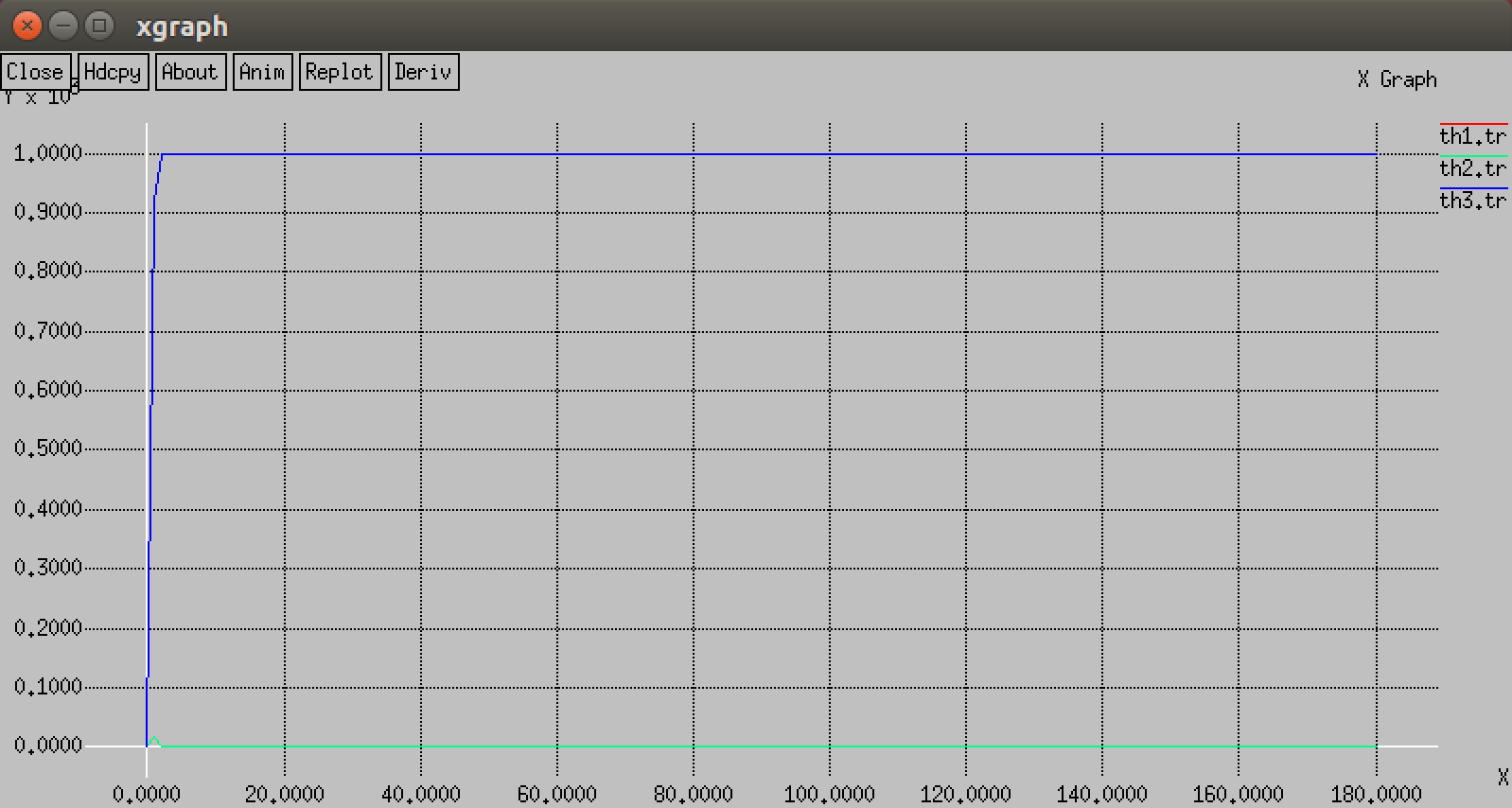


* RED queue management strategy for two scenarios



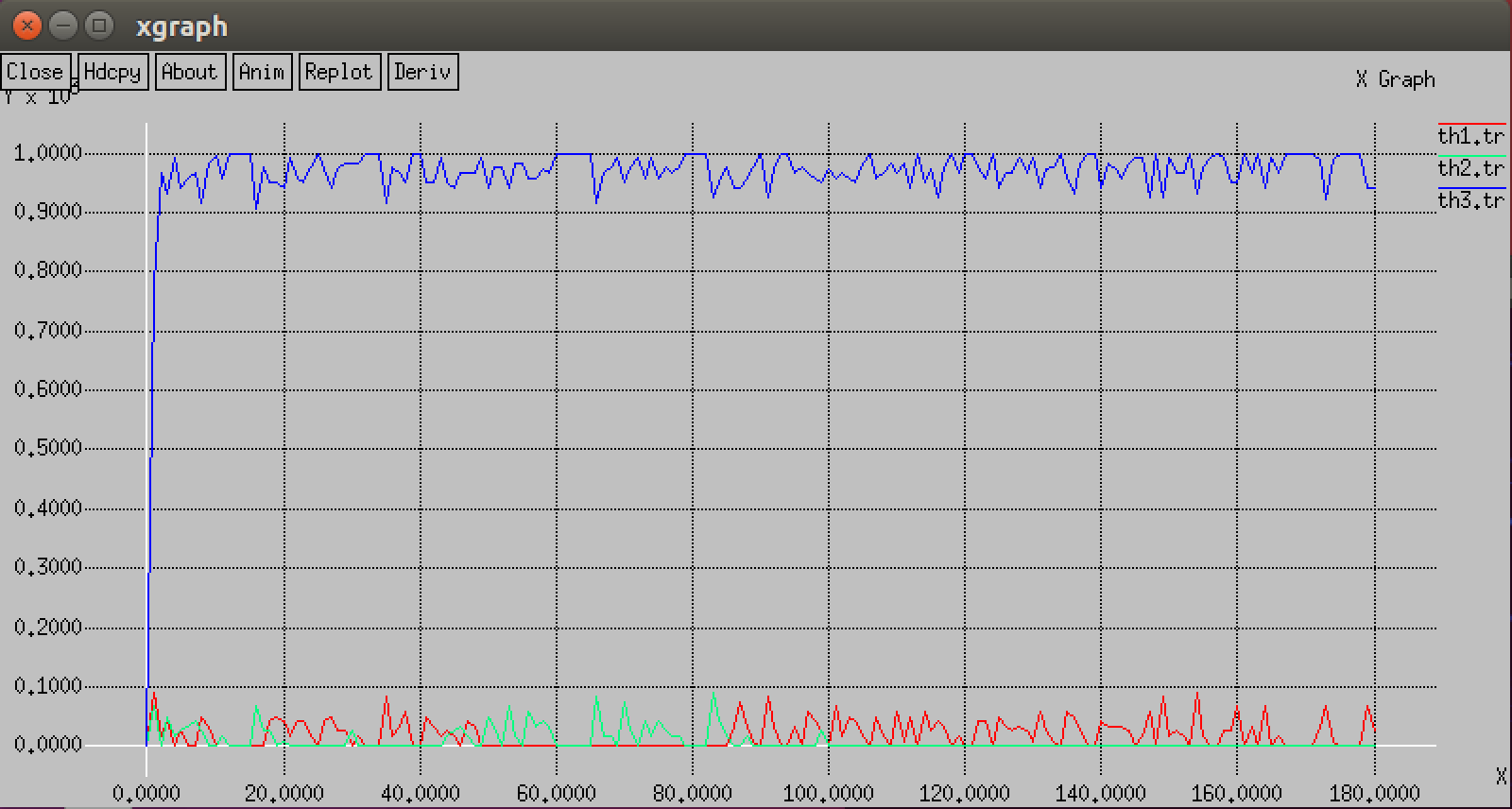
DropTail buffer management: instantaneous throughput for both scenarios





RED buffer management: instantaneous throughputs





From the obtained above results we can tabulate the average throughputs for the sources in each of the strategy

* Scenario 1: average throughput for sources in Kbps

|  |  |  |
| --- | --- | --- |
| Management Strategy | Source 1 | Source 2 |
| DropTail | 499.7546 | 499.9211 |
| RED | 479.3429 | 518.0566 |

* Scenario 2: average throughput for sources in Kbps

|  |  |  |  |
| --- | --- | --- | --- |
| Management Strategy | Source 1 | Source 2 | Source 3 |
| DropTail | 0.00 | 0.00 | 1000 |
| RED | 18.6923 | 6.6005 | 973.7066 |

**3. Analysis**

There are two queuing mechanisms being examined in this experiment, DROPTAIL and RED. The first mechanism being tested was DROPTAIL. This mechanism does not differentiate packets being received by the buffer. This was seen in the graphic of dropped packets as the program was running and more clearly in the instantaneous throughput for scenario 1. This graph shows how the buffer is filled by whichever packets arrive and after filling will drop packets regardless of percentage of packets currently in buffer. A major problem with this strategy is shown in scenario 2; in this scenario the UDP stream is sending significantly more traffic than the TCP streams. This results in the UDP stream filling the buffer entirely and maintaining 100% of the bandwidth after the initial round of queuing. This result is best shown in the overall throughput for scenario 1 with a 499.7546/499.9211 Kbps and scenario 2 with 0/0/1000Kbps. These results show the dangers of DROPTAIL queuing; DROPTAIL does not differentiate service based upon previous traffic and this allows a heavy traffic stream to use all of the available bandwidth.

The second mechanism is RED queuing, which is able to detect the amount of packets currently in the buffer for each stream, and uses this knowledge to drop packets accordingly. This means that the more packets a stream has in the buffer will result in a higher chance of dropped packets. This mechanism allows for more fair service by not favoring bursty data and is better at handling the problem shown in scenario 2. The overall throughput is slightly less even than in scenario 1, this is most likely due to a combination of the early detection algorithm and the sample size. This result is not too concerning because the throughput is still relatively even for scenario 1. The need for the added complexity of the early detection is shown in scenario 2; DROPTAIL resulted in 100% bandwidth consumption for the UDP stream, while RED split the throughput as follows 18.6923/6.6005/973.7066 Kbps respectively. While this is not perfect it is an improvement from DROPTAIL.

Overall this lab shows that DROPTAIL is sufficient if all traffic is relatively uniform but performance will suffer if there is heavy or bursty traffic from a stream. RED uses its early detection algorithm to provide more equal service to all streams. The drawback is added complexity as the router must maintain the amount of packets received by each stream and use this knowledge to drop packets accordingly. RED is better capable of handling heavy and bursty traffic on the network by controlling which packets are dropped from the queue and this results in more fair service.