Offline 3: Network Simulation 2 Project

Md.Shanjinur Islam, 1505066

January 2019

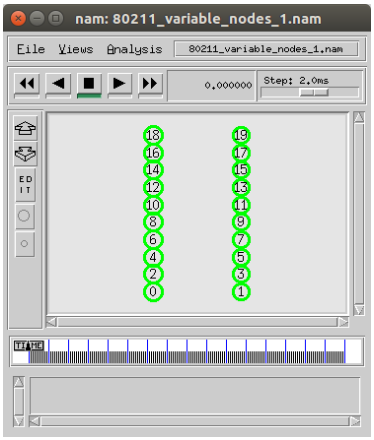
Contents

1 Implemented Protocols

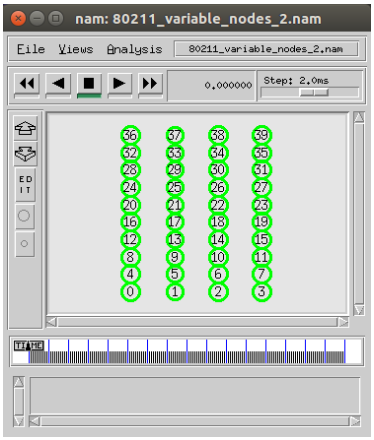
In this assignment, I am instructed to implement these protocols : 802.11 Mobile and 802.15.4 Mobile

2 Network Topologies

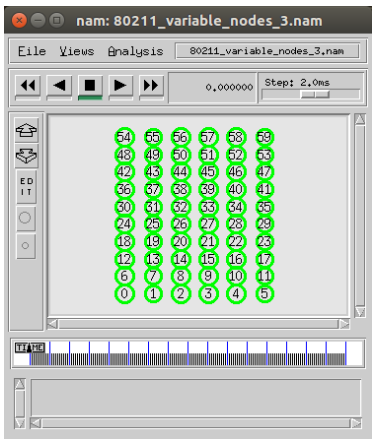
Total 5 types of topologies are used in this project. A grid based topology with nodes varying from 20,40,60,80 Nodes.



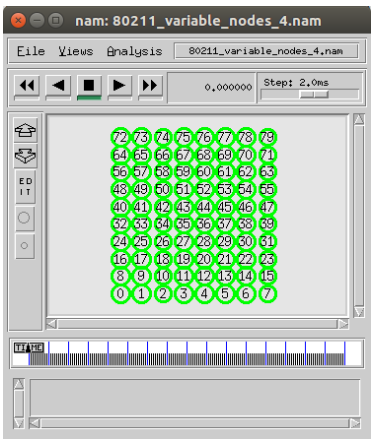
(a) Topology with 20 Nodes



(b) Topology with 40 Nodes



(c) Topology with 60 Nodes



(d) Topology with 80 Nodes

Figure 1: Topology Types

3 Parameters under variations

1. Number of nodes : 20,40,60,80
2. Number of Packet per seconds : 100, 200, 300, 400, and 500
3. Node Speed : 5 m/s, 10 m/s, 15 m/s, 20 m/s, and 25 m/s
4. Number of flows : 10, 20, 30, 40, and 50

4 Modifications

4.1 TCP Westwood

The first modification I made is in cogestion control. The paper I used for this modification is TCP Westwood: End-to-End Congestion Control for Wired/Wireless Networks

Mainly two important changes are made in these three functions of tcp.cc file.

1. void TcpAgent::newack(Packet* pkt)
2. void TcpAgent::slowdown(int how)
3. void TcpAgent::dupack_action()
4. void TcpAgent::timeout(int tno)

List of changes :

1. First I defined two Constant values in tcp.cc

```
//tcp.cc
#define TCP_WESTWOOD 0x00008000
#define TCP_WESTWOOD_TIMEOUT 0x00080000
```

2. set rtt_min and bandwidth value as global variables:

```
//tcp.cc
int bandwidth = 0 ;
float rtt_min = 9999999 ;
```

3. Calculate Bandwidth using packet size and delivery time of receiving ACK packets in void TcpAgent::newack(Packet* pkt) function:

```
//tcp.cc
          if(rtt_min>now - rtt_ts_){
             rtt_min = now - rtt_ts_ ;
```

```
            }
            if(last_ack_==0){
               bandwidth = (packet_size)/(now - rtt_ts_) ;
            }
            else{
               float cur_bandwidth = (packet_size)/(now - rtt_ts_) ;
               cur_bandwidth = (cur_bandwidth + bandwidth)*(2/42) ;
               bandwidth = (19/21)*(bandwidth) + cur_bandwidth ;
            }
```

4. On change dupack_action() function for TCP_WESTWOOD on receive 3 duplicate acknowledgements

```
//tcp.cc
if (!lossQuickStart()) {
      // we are now going to fast-retransmit and willtrace that
          event
      trace_event("FAST_RETX");
      last_cwnd_action_ = CWND_ACTION_DUPACK;
      slowdown(TCP_WESTWOOD);
   }
```

5. Add slowdown option for TCP_WESTWOOD

```
//tcp.cc
if(how & TCP_WESTWOOD){
      ssthresh_ = (bandwidth*rtt_min/1500);
      if(cwnd_ > ssthresh_){
          cwnd_ = ssthresh_ ;
      }
      return ;
   }
```

6. Add TCP_WESTWOOD_TIMEOUT option for timeout() function

```
//tcp.cc
              ++nrexmit_;
            last_cwnd_action_ = CWND_ACTION_TIMEOUT;
            slowdown(TCP_WESTWOOD_TIMEOUT);
            if (frto_enabled_ || sfrto_enabled_) {
               frto_ = 1;
            }
```

7. Add slowdown option for TCP_WESTWOOD_TIMEOUT

```
//tcp.cc
if(how & TCP_WESTWOOD_TIMEOUT){
```

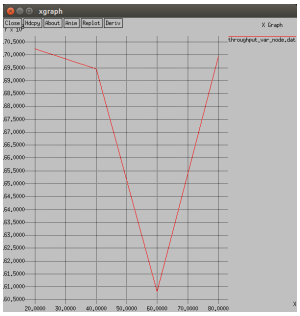
```
    ssthresh_ = (bandwidth*rtt_min/1500);
    if(ssthresh_<2){
        ssthresh_=2 ;
    }
    cwnd_ = 1;
    return ;
}
```

5 Results

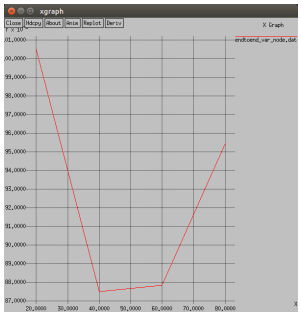
5.1 802.11 Mobile

5.1.1 Variable Nodes

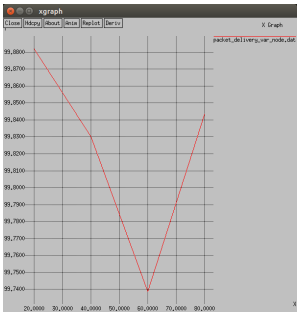
Packet interval : 1 Sec , Number of flow : 20 Queue Length: 50



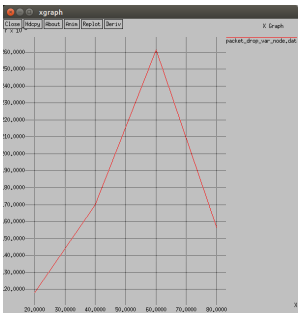
(a) Throughput



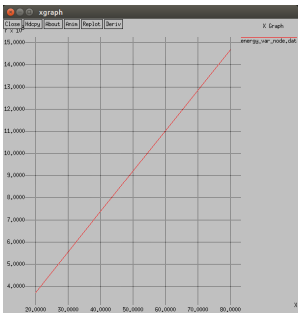
(b) End to end delay



(c) Packet Delivery Ratio



(d) Packet Drop Ratio

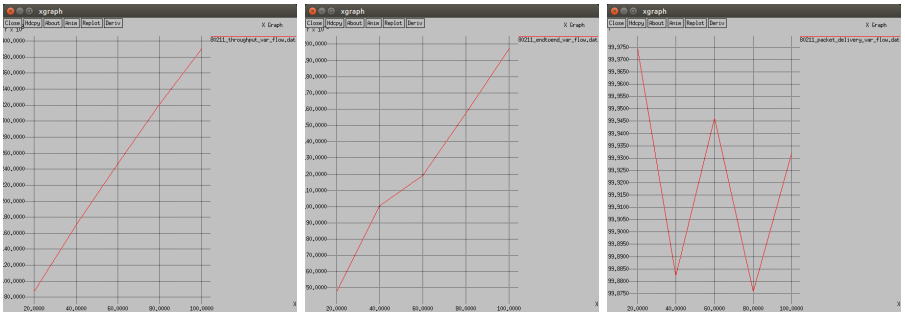


(e) Per Node Energy

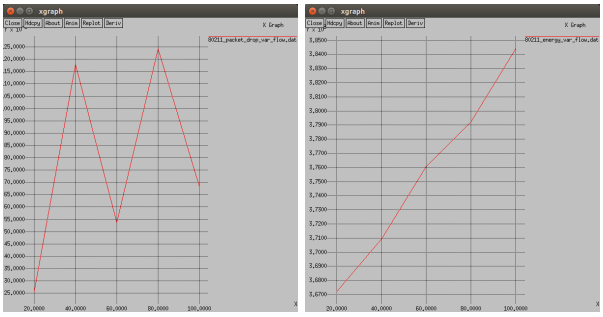
Figure 2: Result graph based on various type of metrics

5.1.2 Variable Flow

Nodes : 20 , Node Speed : 15 Queue Length: 50



(a) Throughput (b) End to end delay (c) Packet Delivery Ratio

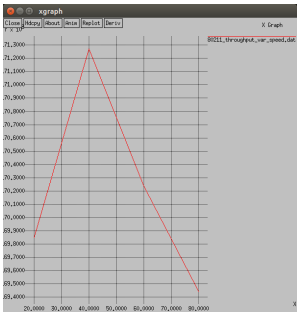


(d) Packet Drop Ratio (e) Per Node Energy

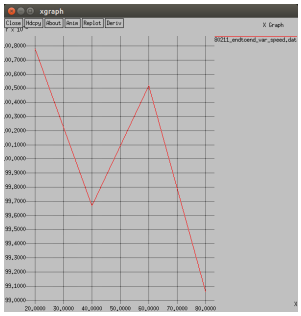
Figure 3: Result graph based on various type of metrics

5.1.3 Variable Node Speed

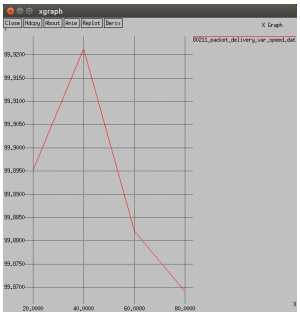
Nodes : 20 , Node Speed : 15 Queue Length: 50



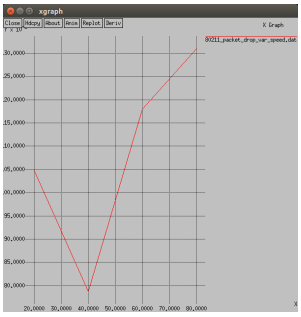
(a) Throughput



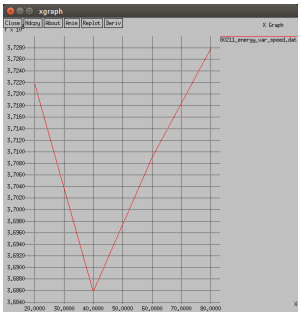
(b) End to end delay



(c) Packet Delivery Ratio



(d) Packet Drop Ratio

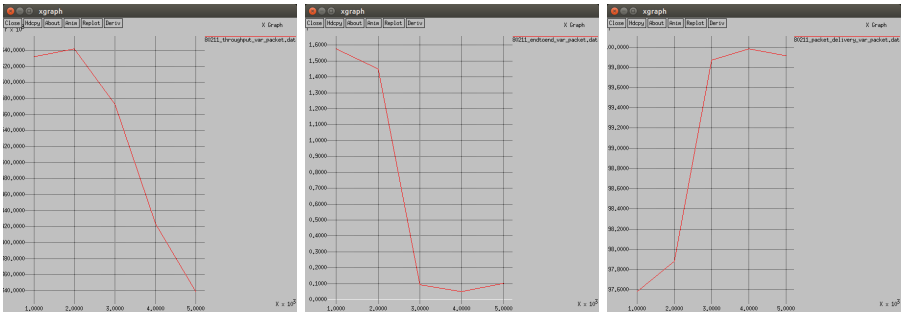


(e) Per Node Energy

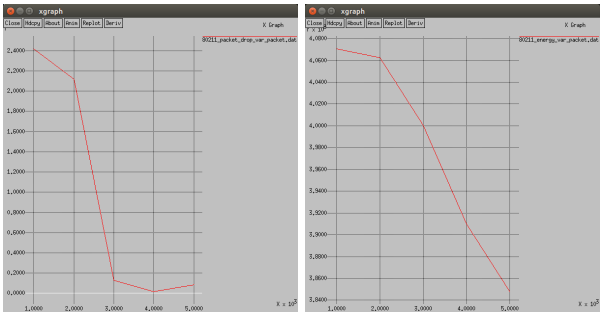
Figure 4: Result graph based on various type of metrics

5.1.4 Variable Packet Rate

Nodes : 20 , Node Speed : 15 Queue Length: 50



(a) Throughput (b) End to end delay (c) Packet Delivery Ratio



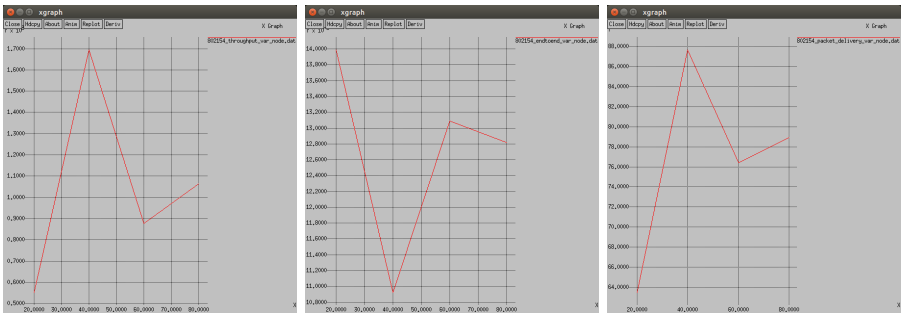
(d) Packet Drop Ratio (e) Per Node Energy

Figure 5: Result graph based on various type of metrics

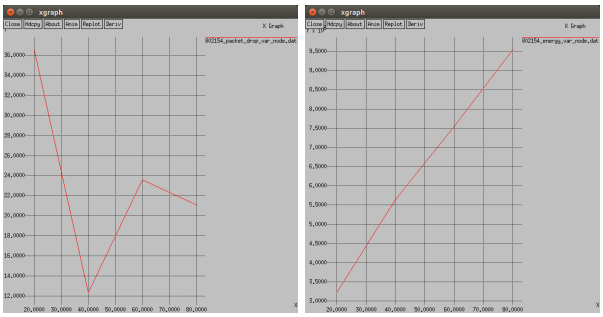
5.2 802.15.4 Mobile

5.2.1 Variable Nodes

Packet interval : 1 Sec , Number of flow : 20 Queue Length: 50



(a) Throughput (b) End to end delay (c) Packet Delivery Ratio

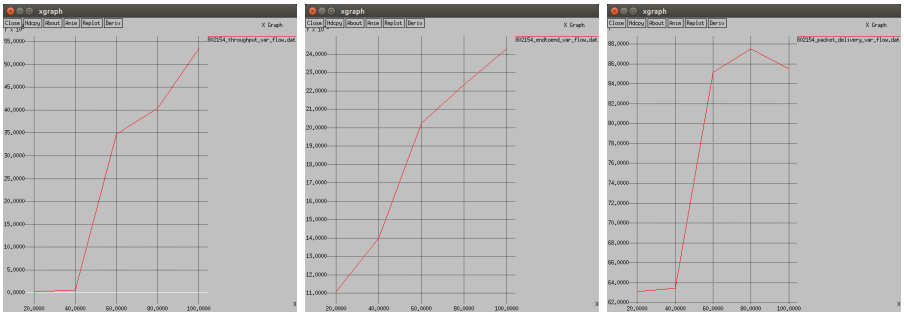


(d) Packet Drop Ratio (e) Per Node Energy

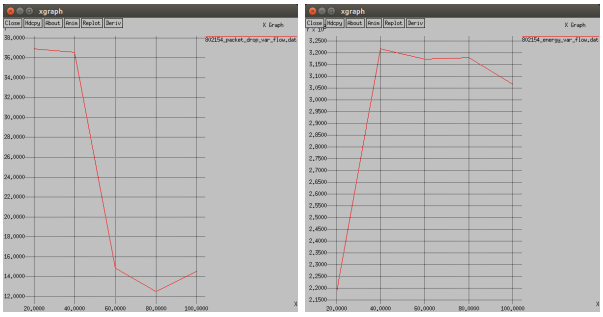
Figure 6: Result graph based on various type of metrics

5.2.2 Variable Flow

Nodes : 20 , Node Speed : 15 Queue Length: 50



(a) Throughput (b) End to end delay (c) Packet Delivery Ratio

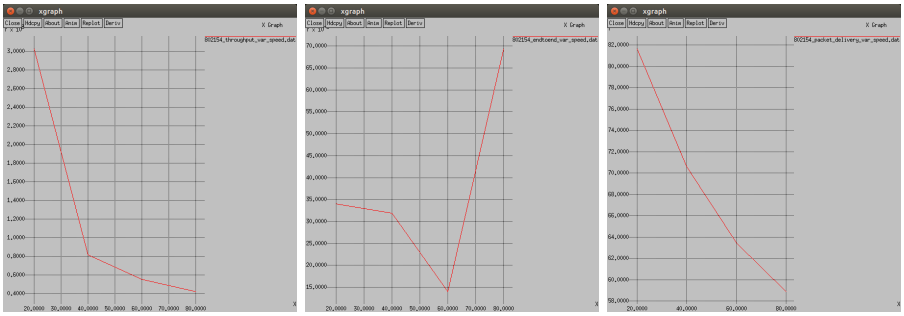


(d) Packet Drop Ratio (e) Per Node Energy

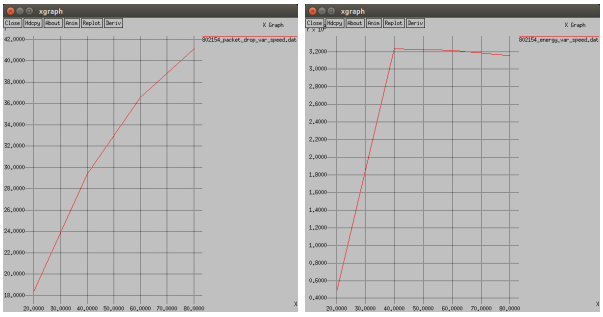
Figure 7: Result graph based on various type of metrics

5.2.3 Variable Node Speed

Nodes : 20 , Node Speed : 15 Queue Length: 50



(a) Throughput (b) End to end delay (c) Packet Delivery Ratio

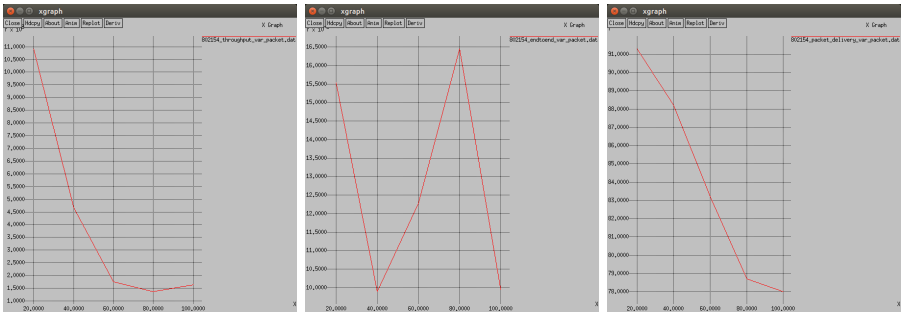


(d) Packet Drop Ratio (e) Per Node Energy

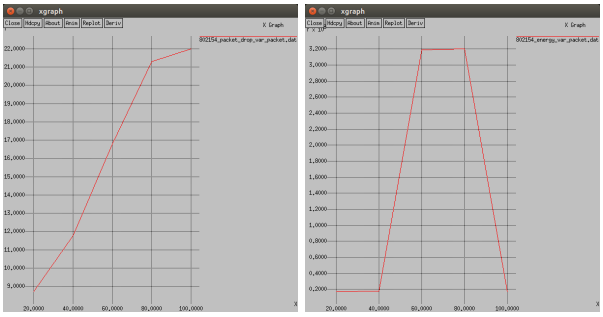
Figure 8: Result graph based on various type of metrics

5.2.4 Variable Packet Rate

Nodes : 20 , Node Speed : 15 Queue Length: 50



(a) Throughput (b) End to end delay (c) Packet Delivery Ratio



(d) Packet Drop Ratio (e) Per Node Energy

Figure 9: Result graph based on various type of metrics

6 Comparison With Modifications

6.1 802.11 Mobile

6.1.1 Throughput Variation on Variable Packet Rate

Figure 10: Throughput Variation on Variable Packet Rate

6.1.2 End to end delay Variation on variable packet rate

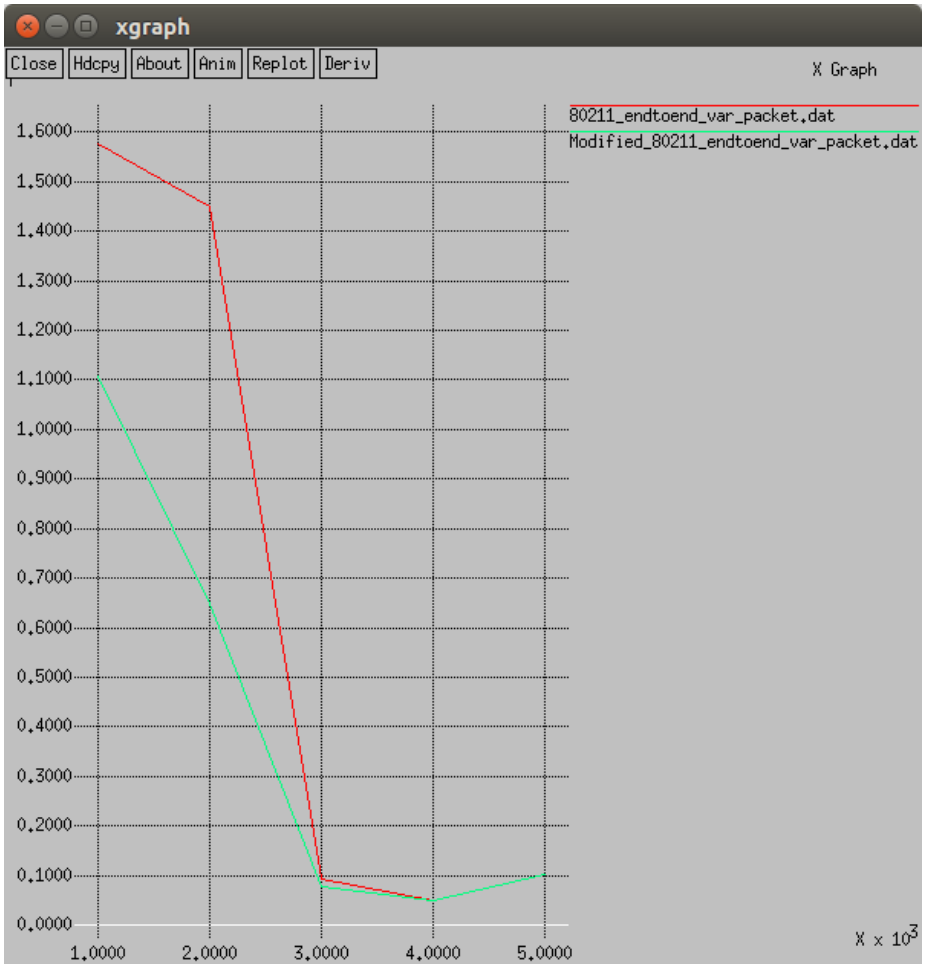


Figure 11: End to End Delay Variation on Variable Packet Rate

6.2 802.15.4 Mobile

6.2.1 Variable Packet: Throughput Variation

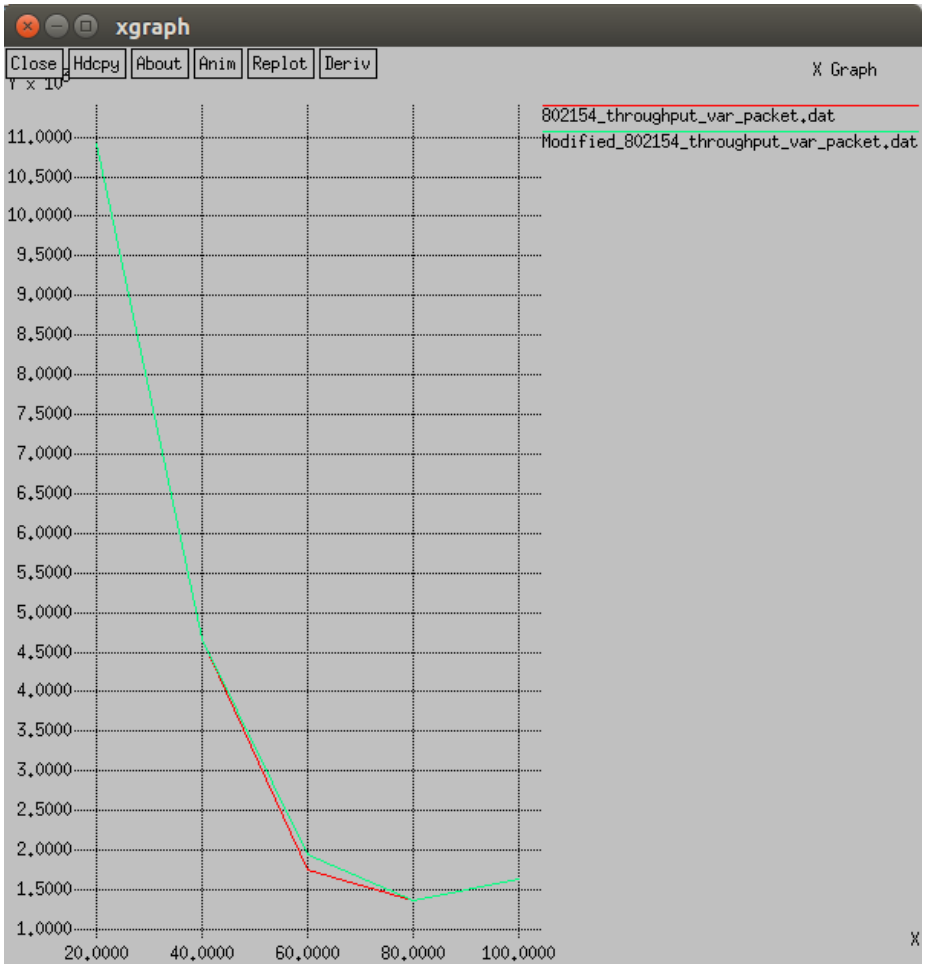


Figure 12: Throughput Variation on Variable Packet Rate

6.2.2 Variable Packet: End to End Delay Variation

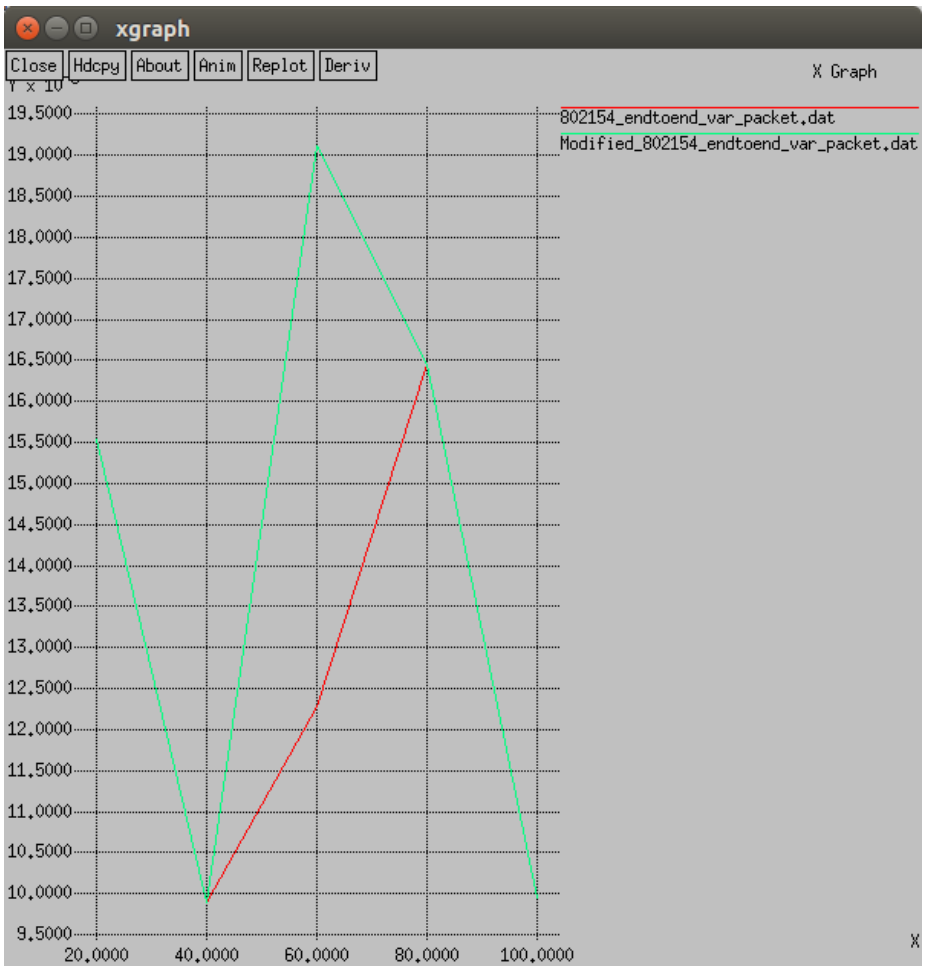


Figure 13: Throughput Variation on Variable Packet Rate

7 Summary

TcpTahoe is much more efficient than TcpWestwood but for variable packet rate but as the paper suggested, TcpWestwood has slightly lower end to end delay in 802.11 and 802.15.4 Protocols.