CSE-322: NS2 Project

By,

Md Mahmudul Hasan

1205006

Problem:

Both Wired Network, and Wireless 802.11 network I was assigned use DropTail queue to temporarily store the data to be sent to the network. The basic idea behind DropTail as implemented in NS2 is, after a new packet arrives, the queue discards the oldest item to make place for the newest item, which is generally called FIFO data structure (First In First Out). This blind selection causes an inefficient network, because an important packet can be dropped on arrival of a less important packet.

Attempted improvement:

I tried to solve this problem in this project by setting up a importance weight on each packet. For this project, the criteria for packet to be important is to carry maximum amount of data. As each packet has equal size, the packet having the maximum DataSize is considered the most important one. And packet with least DataSize is considered least important one, and will be replaced as soon as a more important packet arrives.

Algorithm:

1. New Packet arrives
2. Remove the least important packet from the queue
3. Insert the new packet in the queue.

Implementation:

2 functions in the file queue.h was changed to attain our improvement:

- virtual Packet* enque(Packet* p);
- virtual Packet* deque(Packet* p);

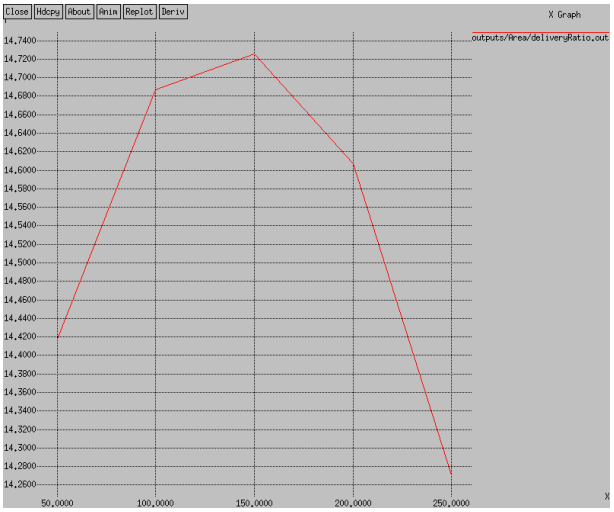
```
virtual Packet* enque(Packet* p) { // Returns previous tail
          int p_len = p->datalen();

          Packet* pt = tail_;
          if (!tail_) {
                head_= tail_= p;
                //Check the new packet data length,
                //if smaller than already found smallest, then set it smallest
                //Iftail is empty, set min_parents next to p
                if ( p_len < min_len){
                      min_parent->next_=p;
                      min_len = p_len;
                }
          }
          else {
                //If the packet found has the lowest data length
                //set parent of minimum to the tail
                if ( p_len < min_len){
                      min_parent=tail_;
                      min_len = p_len;
                }
                tail_->next_= p; //Change here //Masum
                tail_= p;
          }
          tail_->next_= 0;
          ++len_;
          bytes_ += hdr_cmn::access(p)->size();
          return pt;
     }




     virtual Packet* deque() {
          if (!head_) return 0;
          Packet* p = min_parent->next_;//head_;
          if (!p)
          {
                return 0;
          }
          else if (head_==tail_)
          {
                head_=tail_=0;
                min_parent->next_=0;

          }
          else if(head_==p)
          {
                head_=head_->next_;
          }
```

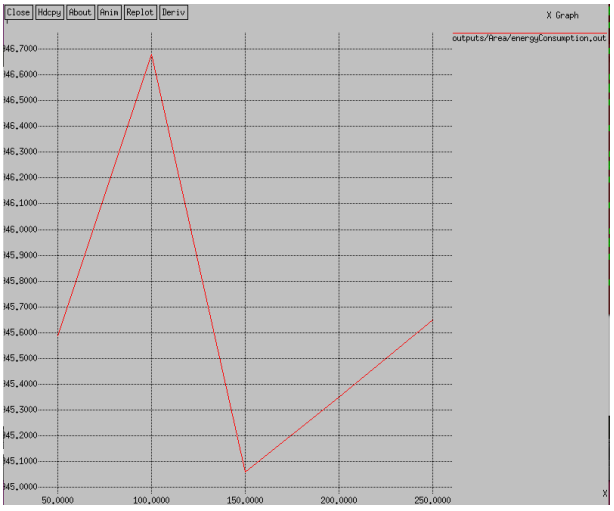
```
        else if (p==tail_)
        {
              tail_ =min_parent;
        }
        else {
              min_parent->next_ = p->next_;
        }
        //head_= p->next_; // 0 if p == tail_
        --len_;
        bytes_ -= hdr_cmn::access(p)->size();

        //Find the packet with lowest data size and remove
        //check for lowest value of datalen()
        min_len=MAX_INT;
        int p_len;
        Packet* parent;
        parent->next_=head_;
        //find the next smallest data
        for (Packet* p = head_; p != 0; p = p->next_) {
              p_len = p->datalen();
              if(p_len<min_len){
                    min_len = p_len;
                    min_parent = parent;
              }
              parent = p;
        }

        return p;
  }
```

Values before applying the new technique:

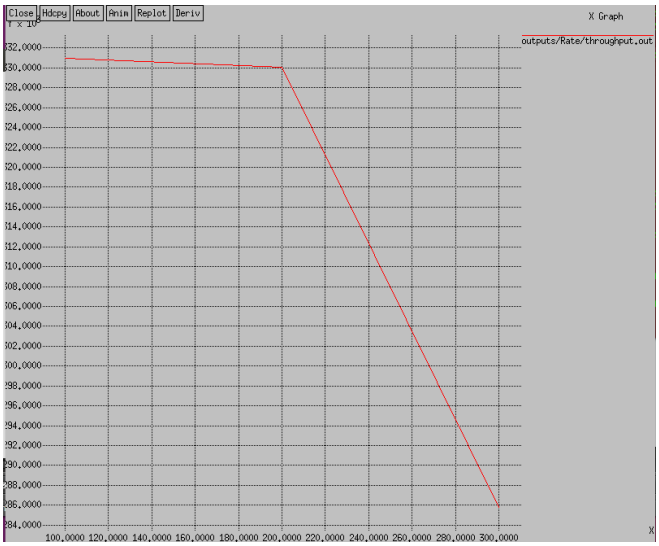
802.11 Wireless Static

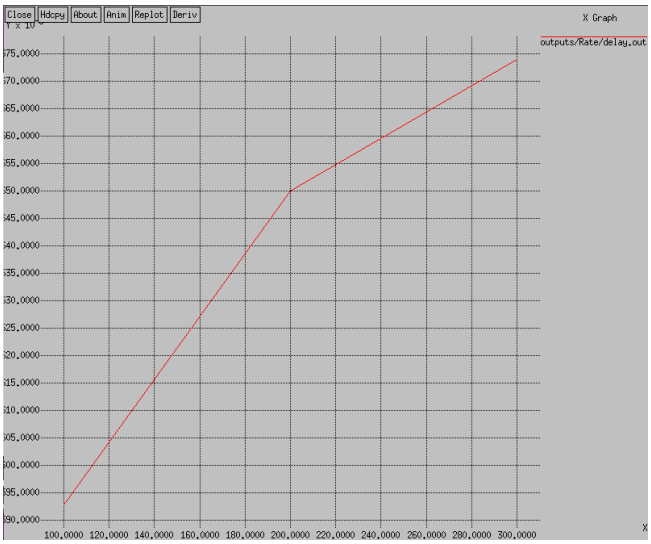
Varying Area:

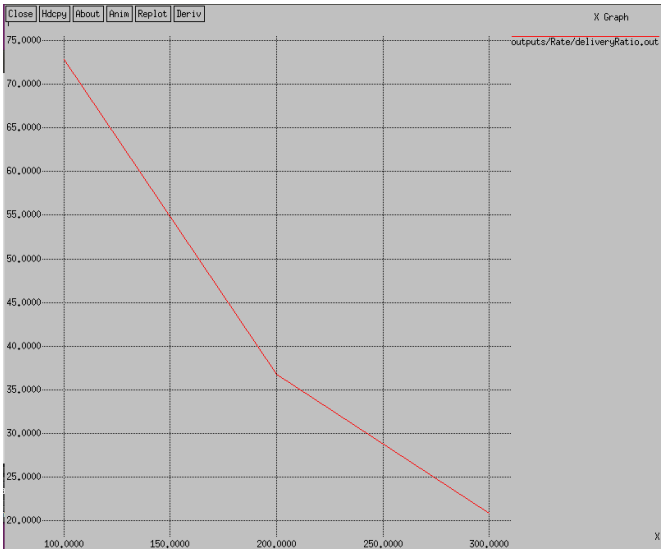


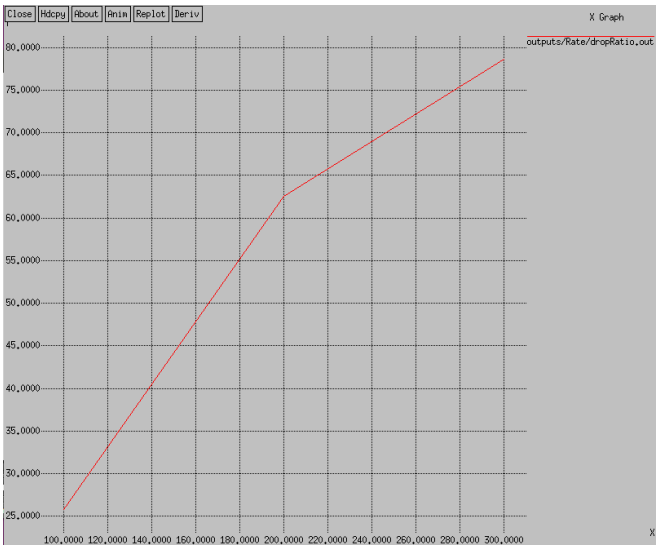


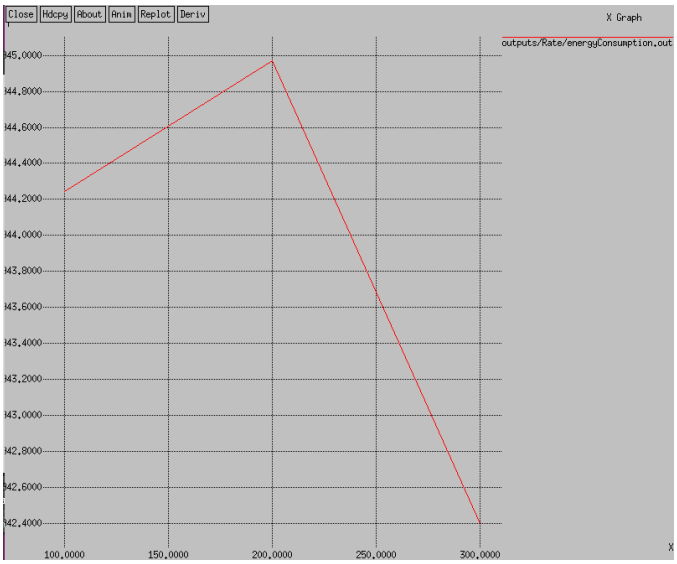
Varying Packet per Second:



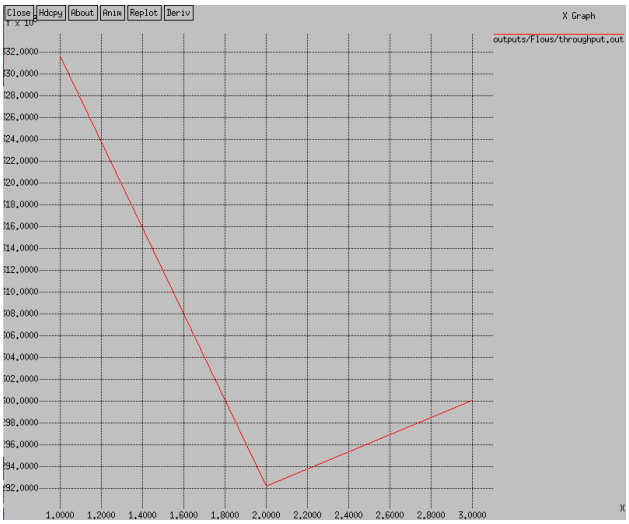


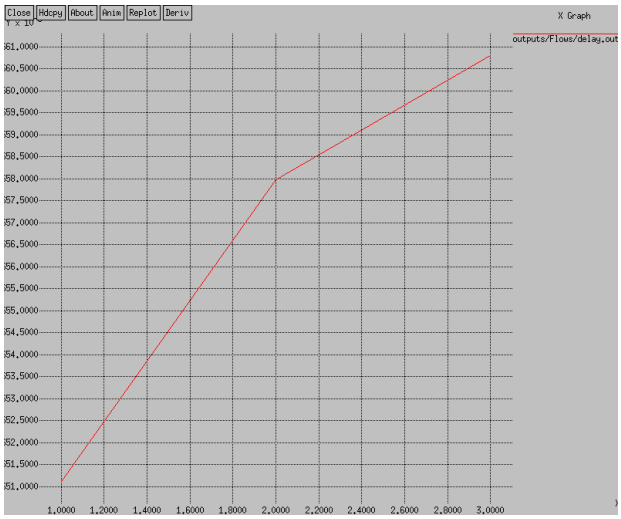


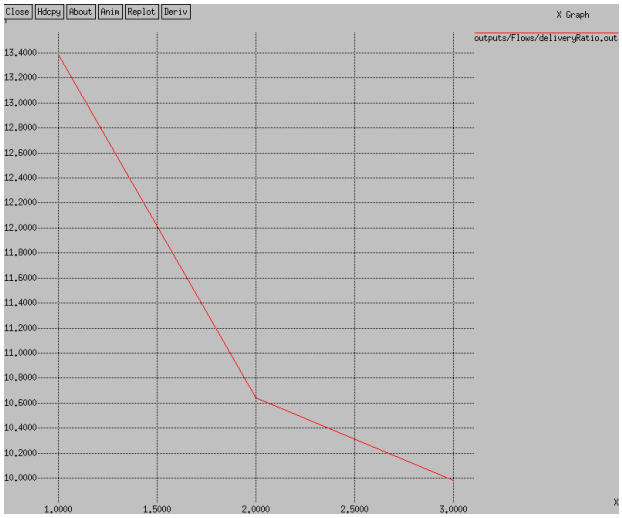


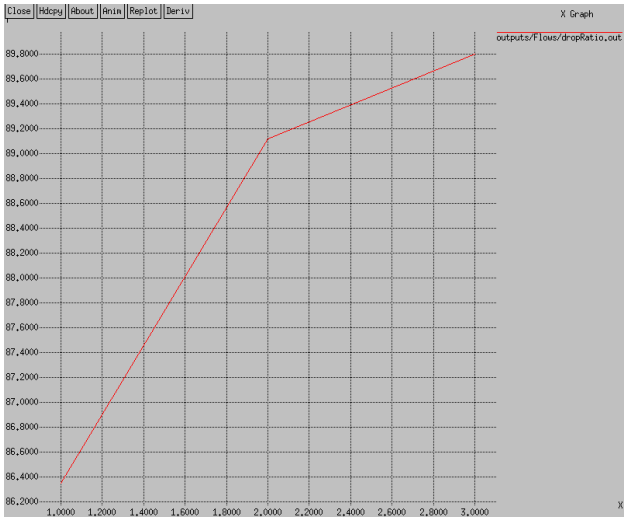


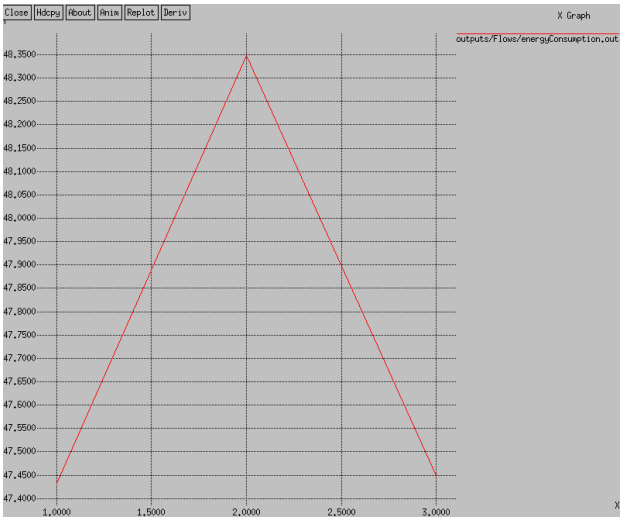
Varying flow:


Close Hdcpy About Anim Replot Deriv
X Graph
outputs/Flows/throughput.out
T x 10
32.0000
30.0000
28.0000
26.0000
24.0000
22.0000
20.0000
18.0000
16.0000
14.0000
12.0000
10.0000
08.0000
06.0000
04.0000
02.0000
00.0000
98.0000
96.0000
94.0000
92.0000
1.0000 1.2000 1.4000 1.6000 1.8000 2.0000 2.2000 2.4000 2.6000 2.8000 3.0000
X

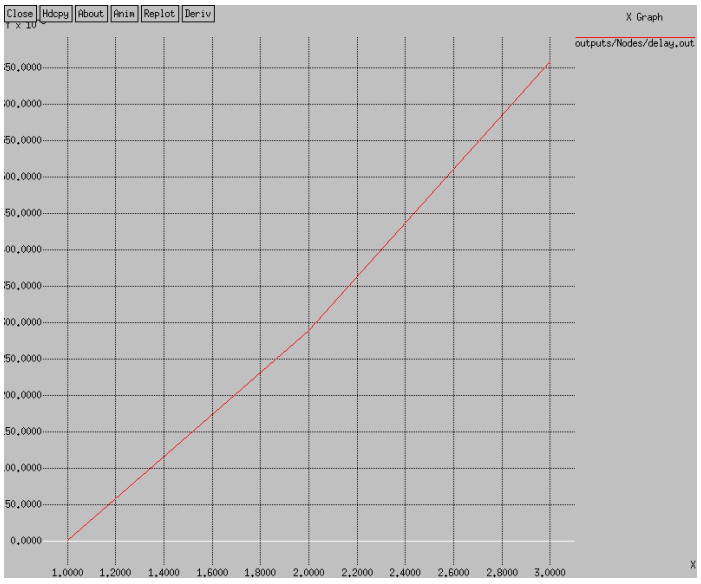

Close Hdcpy About Anim Replot Deriv
X Graph
outputs/Flows/delay.out
T x 10
61.0000
60.5000
60.0000
59.5000
59.0000
58.5000
58.0000
57.5000
57.0000
56.5000
56.0000
55.5000
55.0000
54.5000
54.0000
53.5000
53.0000
52.5000
52.0000
51.5000
51.0000
1.0000 1.2000 1.4000 1.6000 1.8000 2.0000 2.2000 2.4000 2.6000 2.8000 3.0000
X

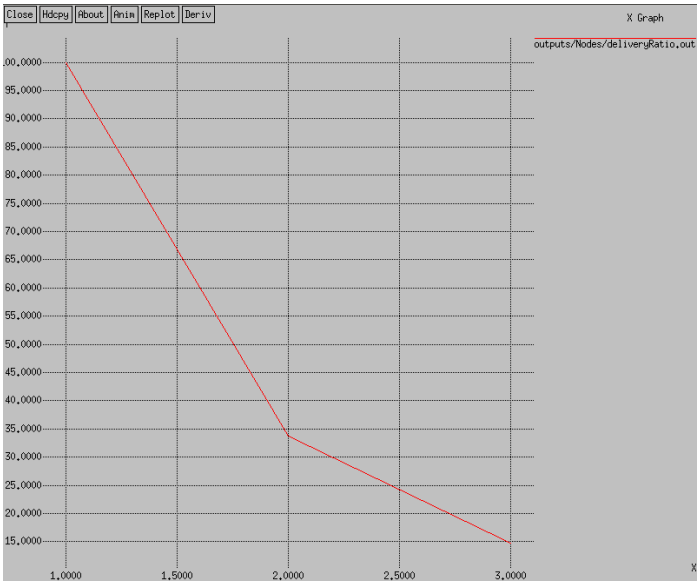

Close Hdcpy About Anim Replot Deriv
X Graph
outputs/Flows/deliveryRatio.out
13.4000
13.2000
13.0000
12.8000
12.6000
12.4000
12.2000
12.0000
11.8000
11.6000
11.4000
11.2000
11.0000
10.8000
10.6000
10.4000
10.2000
10.0000
1.0000 1.5000 2.0000 2.5000 3.0000
X

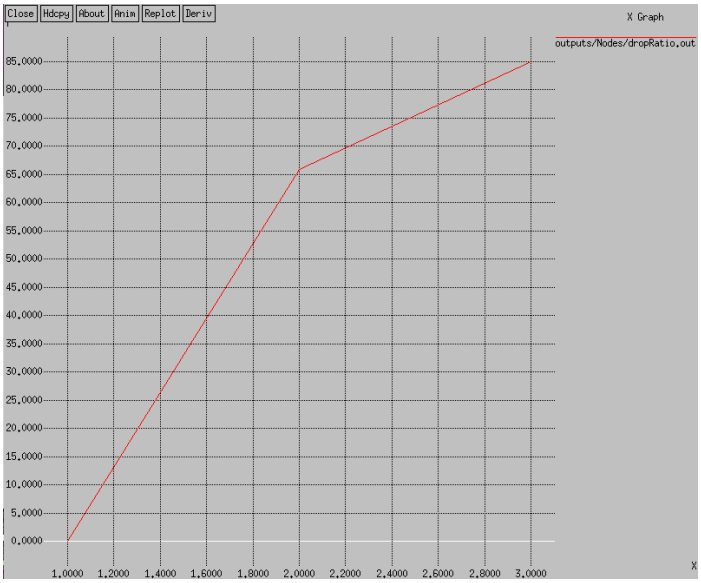

Close Hdcpy About Anim Replot Deriv
X Graph
outputs/Flows/dropRatio.out
89.8000
89.6000
89.4000
89.2000
89.0000
88.8000
88.6000
88.4000
88.2000
88.0000
87.8000
87.6000
87.4000
87.2000
87.0000
86.8000
86.6000
86.4000
86.2000
1.0000 1.2000 1.4000 1.6000 1.8000 2.0000 2.2000 2.4000 2.6000 2.8000 3.0000
X

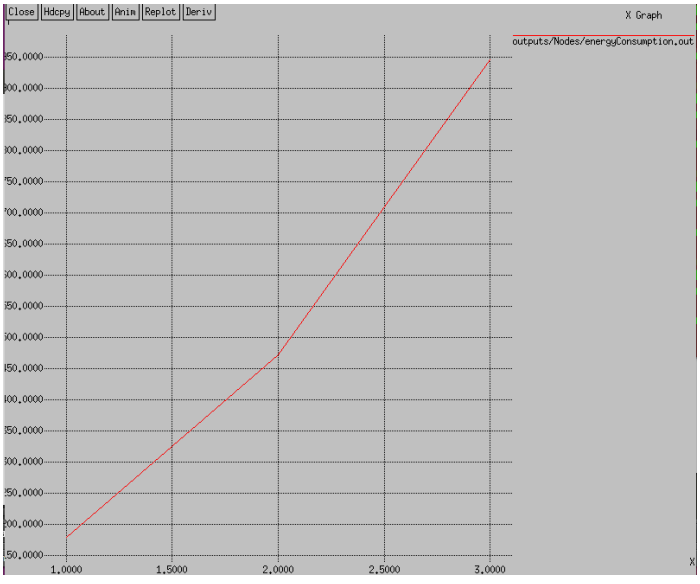

Close Hdcpy About Anim Replot Deriv
X Graph
outputs/Flows/energyConsumption.out
48.3500
48.3000
48.2500
48.2000
48.1500
48.1000
48.0500
48.0000
47.9500
47.9000
47.8500
47.8000
47.7500
47.7000
47.6500
47.6000
47.5500
47.5000
47.4500
47.4000
1.0000 1.5000 2.0000 2.5000 3.0000
X

Varying Nodes:



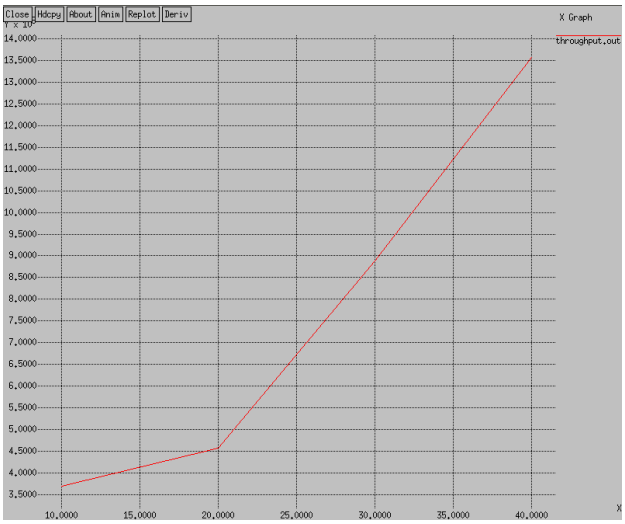


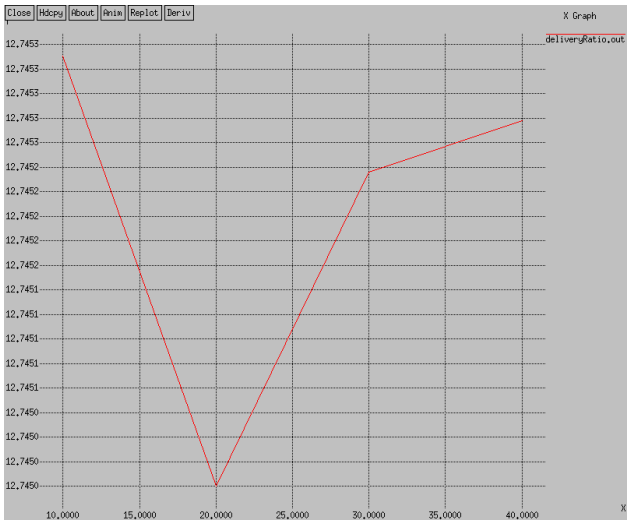


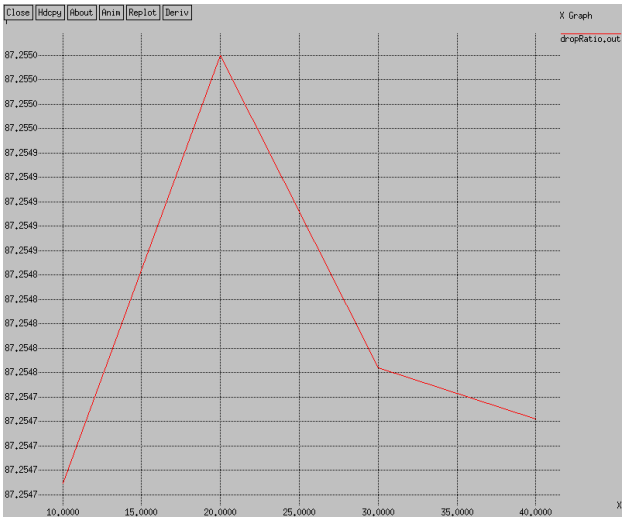


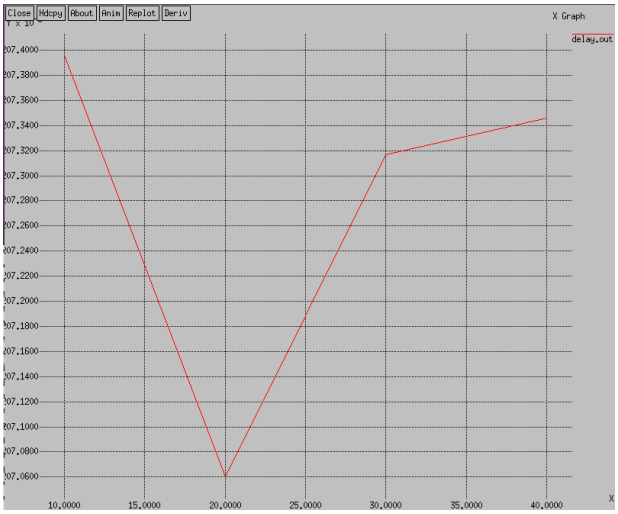
Wired network

Varying nodes:

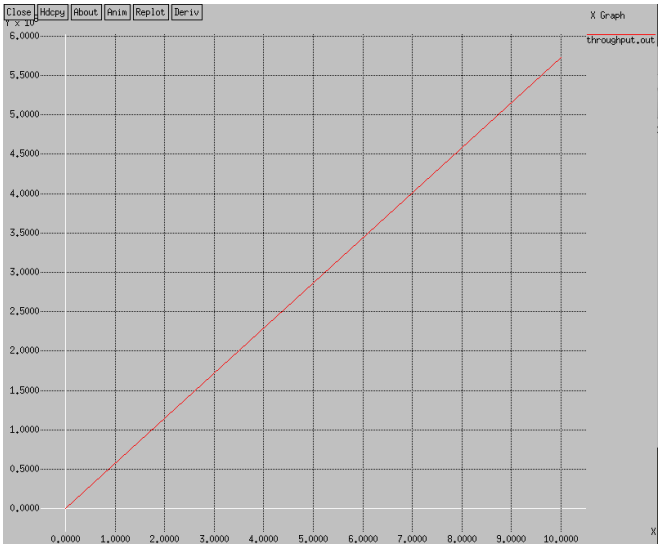


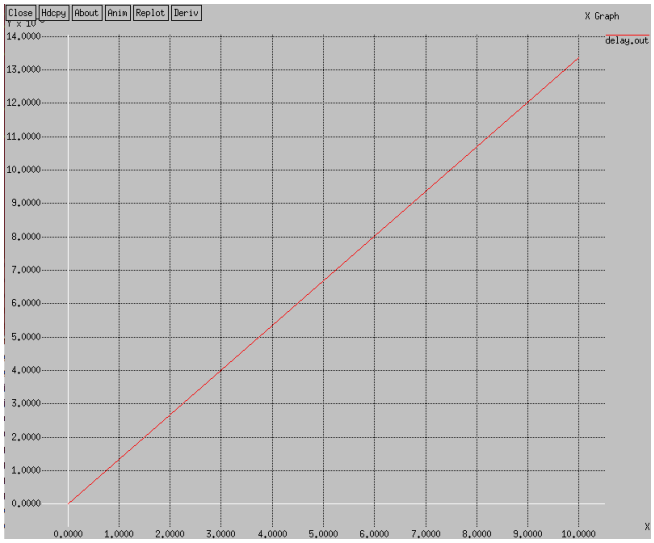


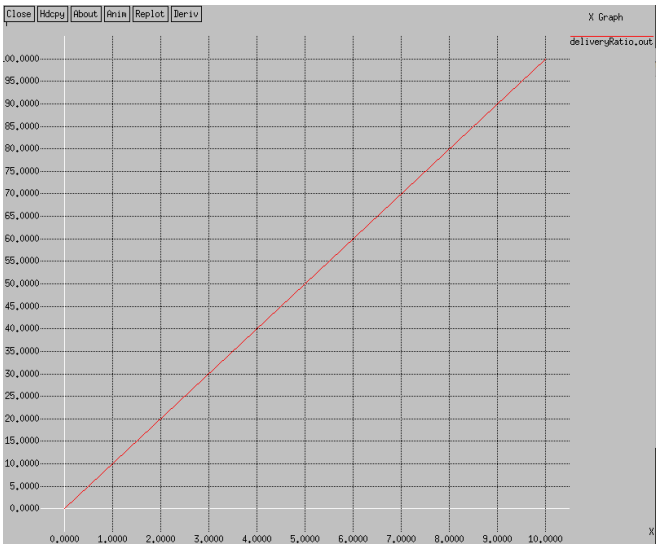


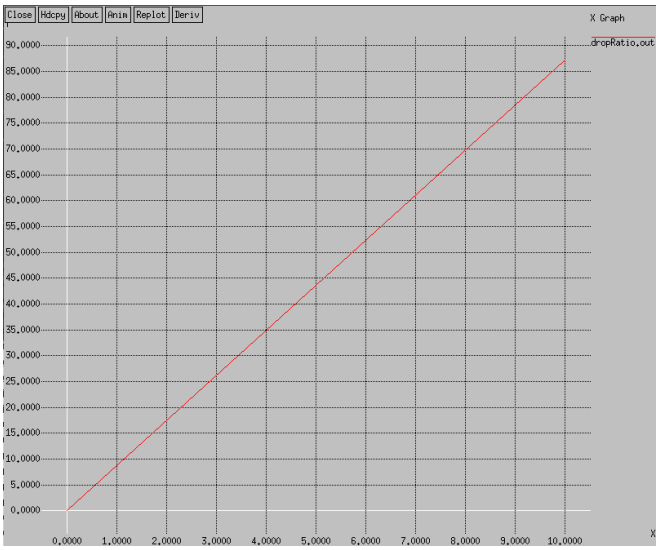


Varying Flow:

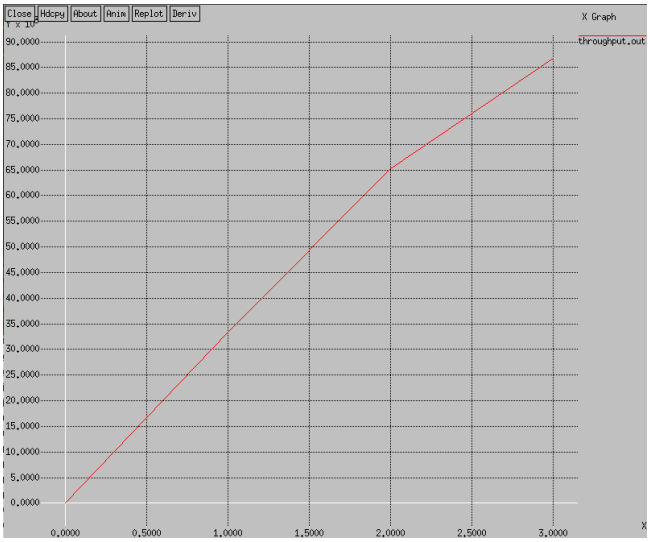


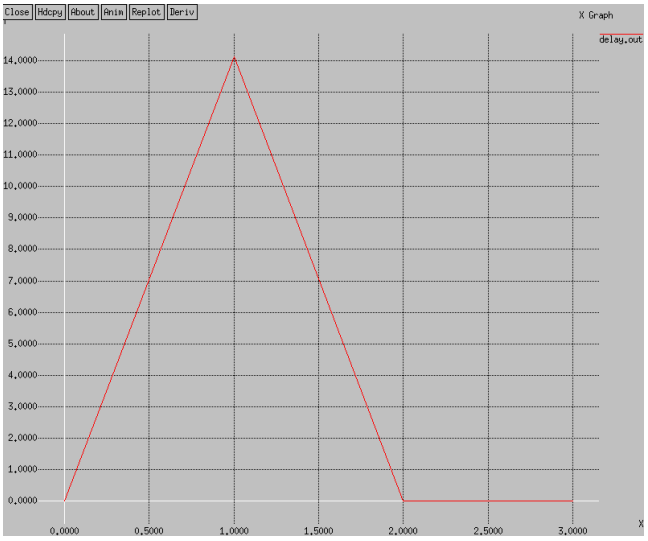


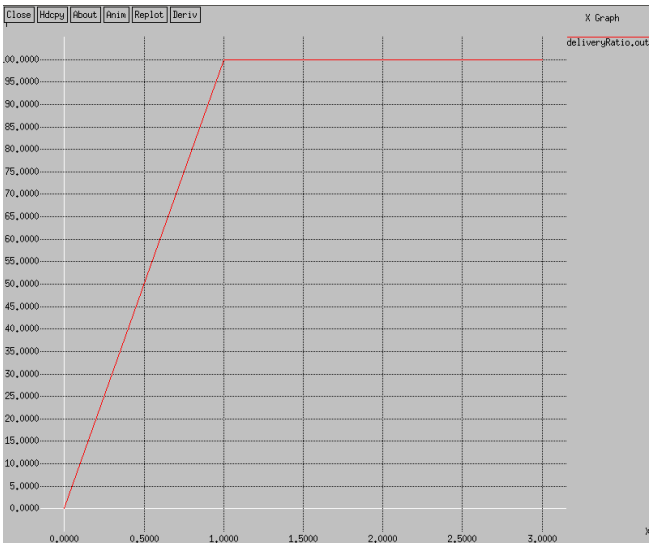


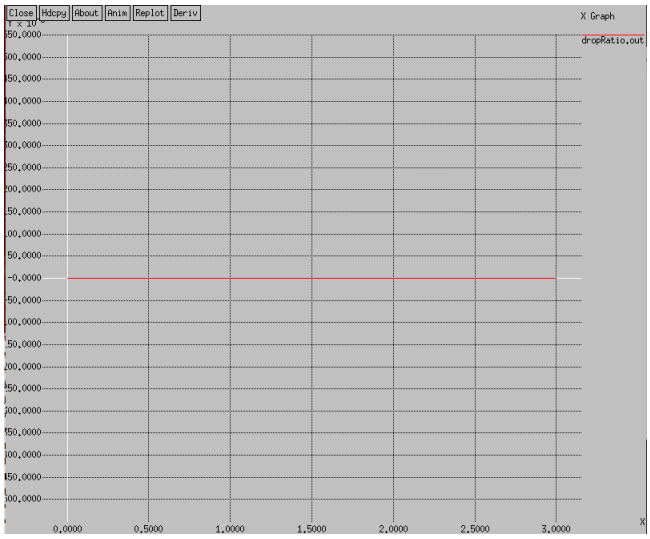


Varying Packet per Second:









Result:

Unfortunately my attempt was an error. And it resulted in Segmentation Fault. So, no graph for after applying the proposed technique can be demonstrated.