# 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.

Current Algorithm : DropTail

Add Tail

Drop Front

Add Tail

Drop the lowest
data length
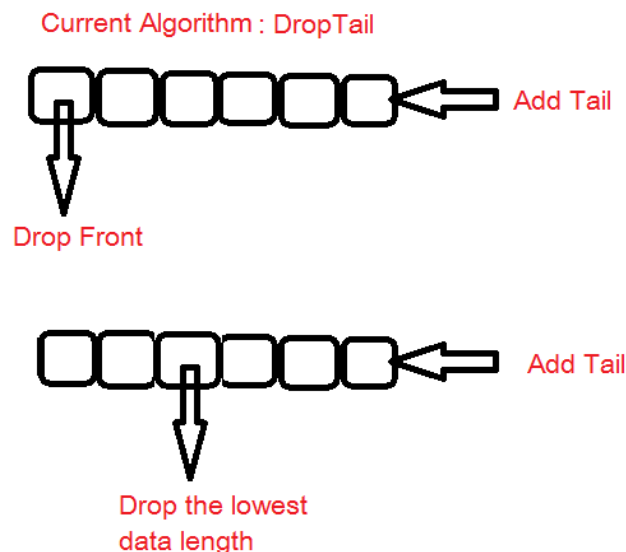
Algorithm:

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;
}
```

```cpp
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;
}
```
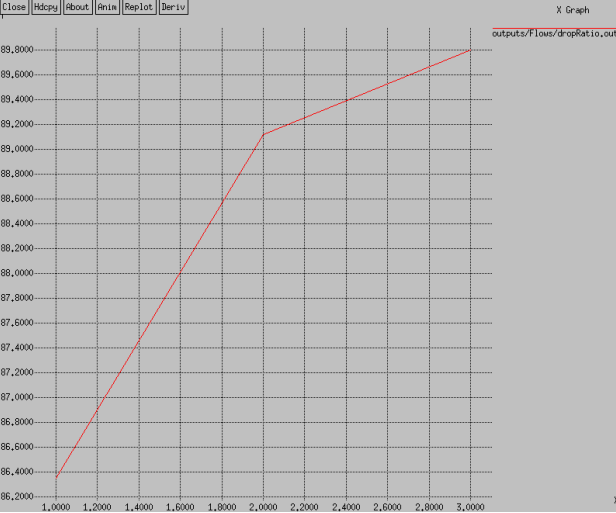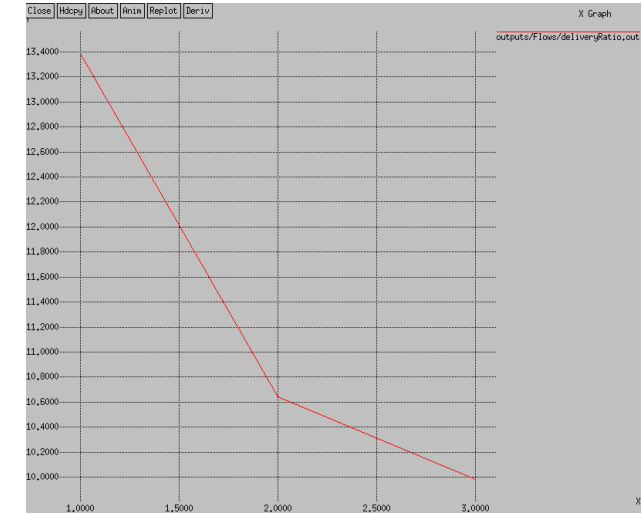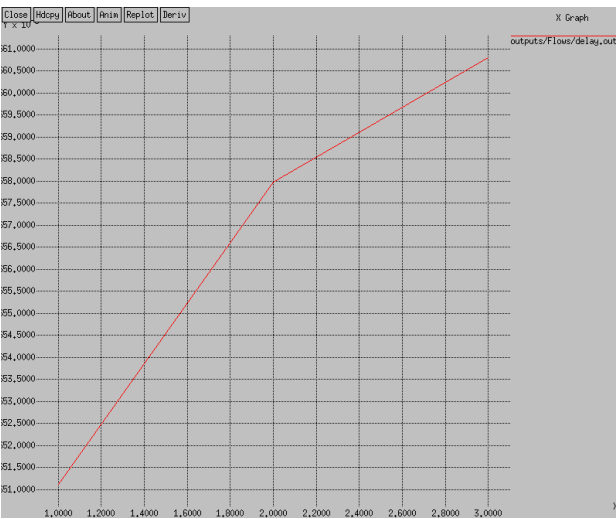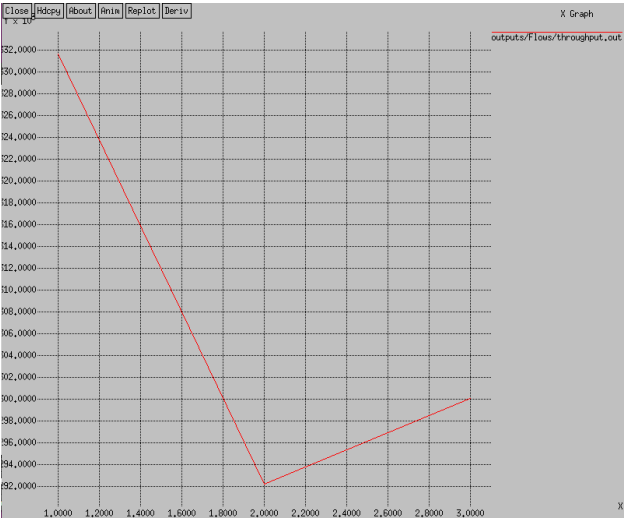
Values before applying the new technique:

802.11 Wireless Static

Varying Area:
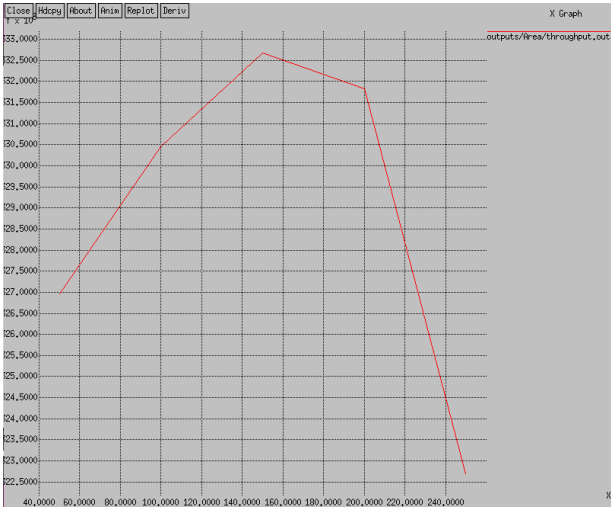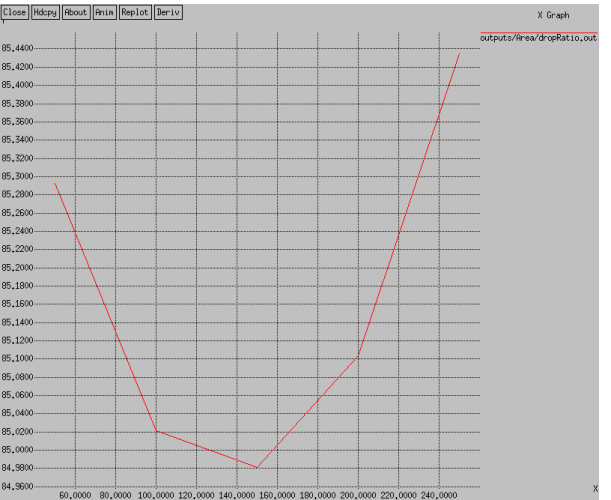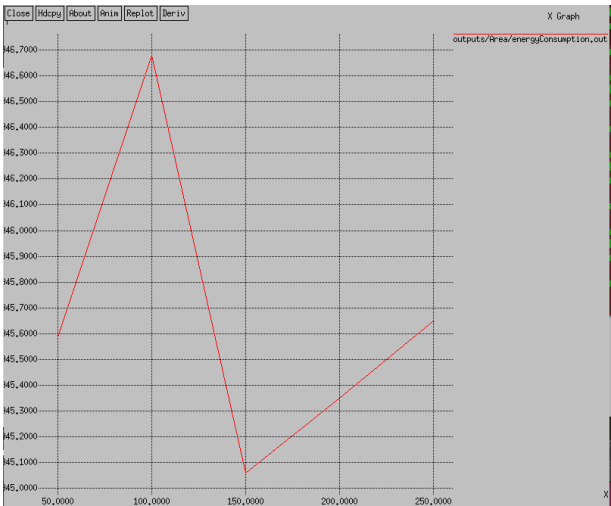
➔ Area in meter

Varying Packet per Second:

➔ Packet/s

Varying flow:

Close | Hdcpy | About | Anim | Replot | Deriv   X Graph

outputs/Flows/throughput.out

Close | Hdcpy | About | Anim | Replot | Deriv   X Graph

outputs/Flows/delay.out

Close | Hdcpy | About | Anim | Replot | Deriv   X Graph

outputs/Flows/deliveryRatio.out

Close | Hdcpy | About | Anim | Replot | Deriv   X Graph

outputs/Flows/dropRatio.out

Close | Hdcpy | About | Anim | Replot | Deriv   X Graph

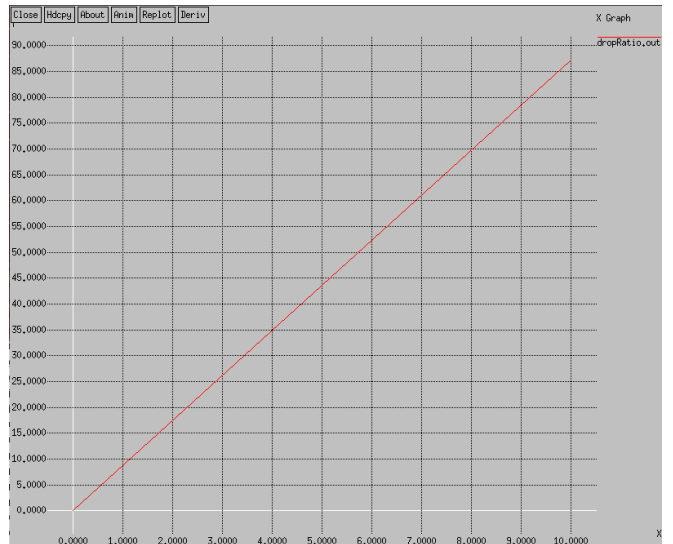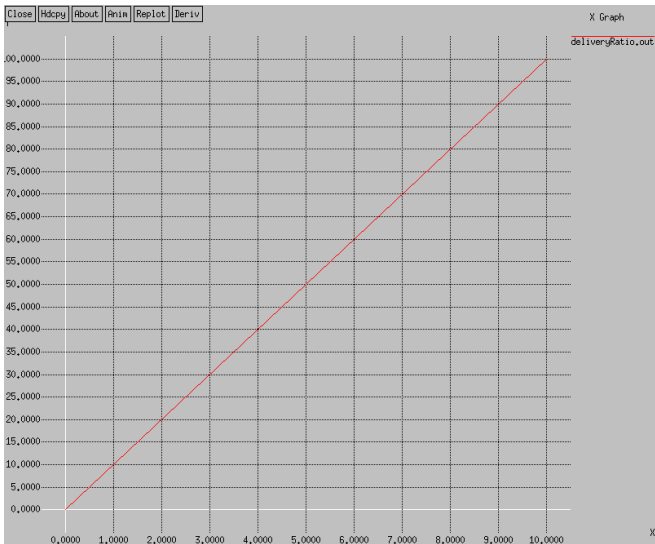outputs/Flows/energyConsumption.out
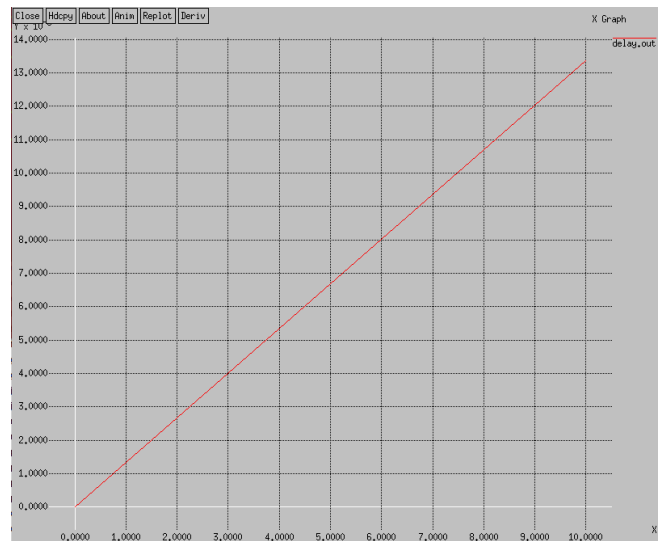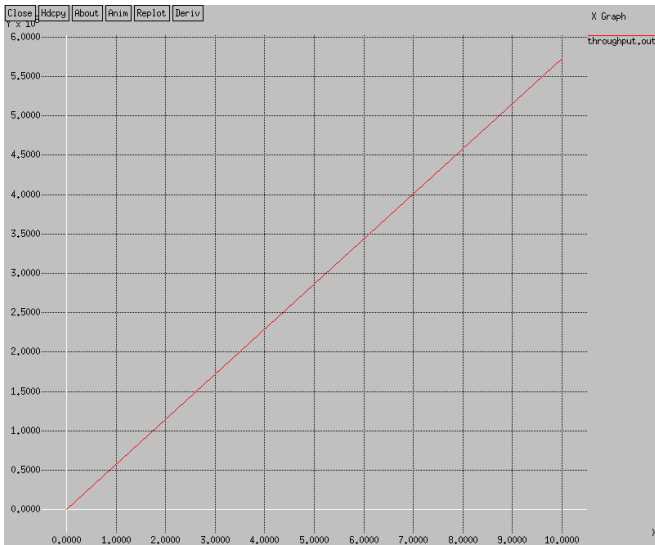
➔ Number of flow

Varying Nodes:
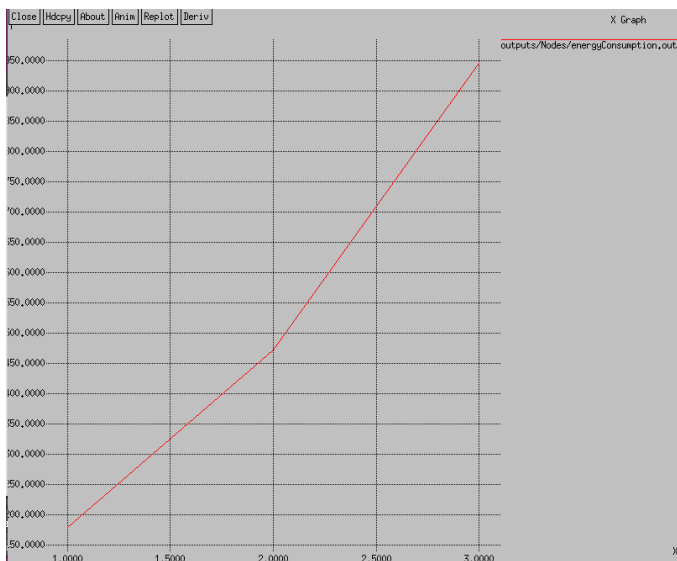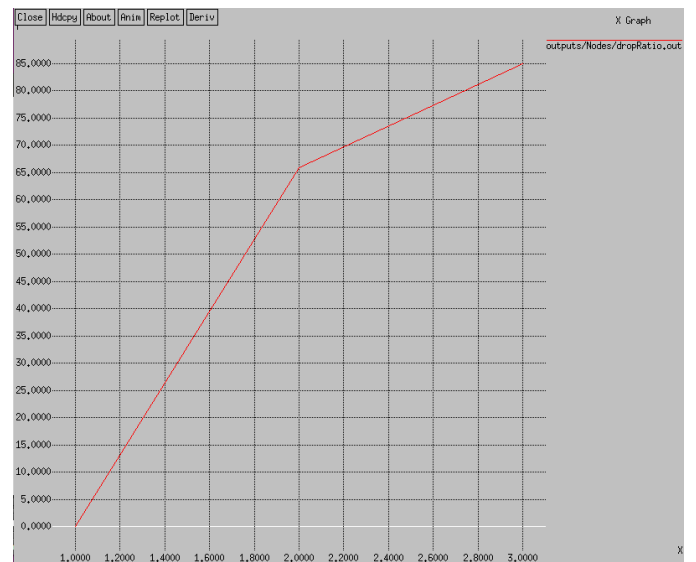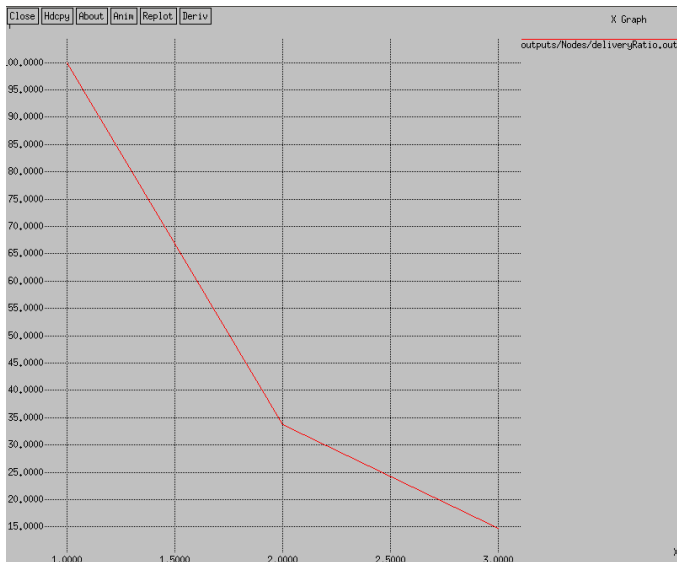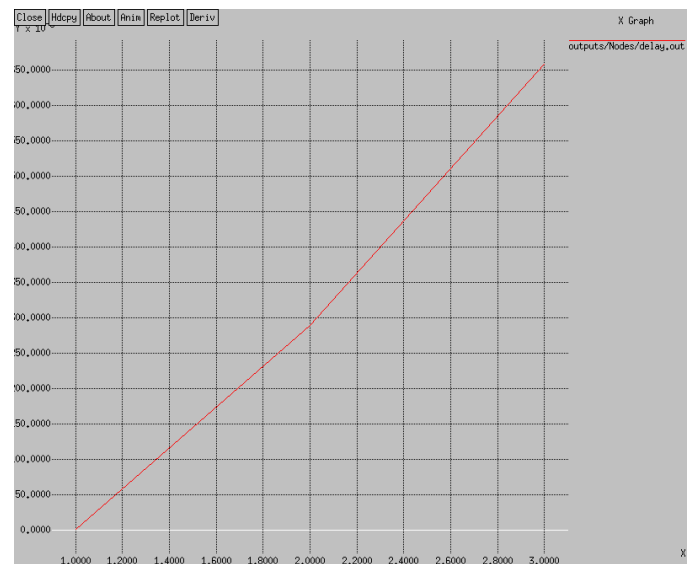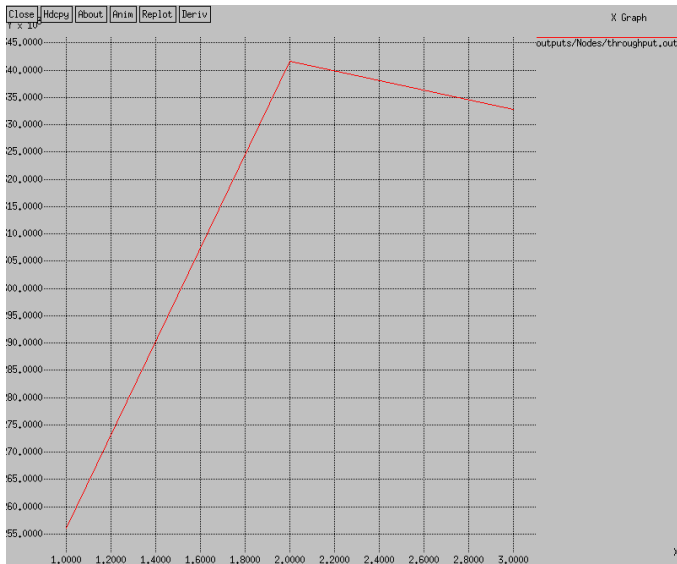
➔ Number of Nodes
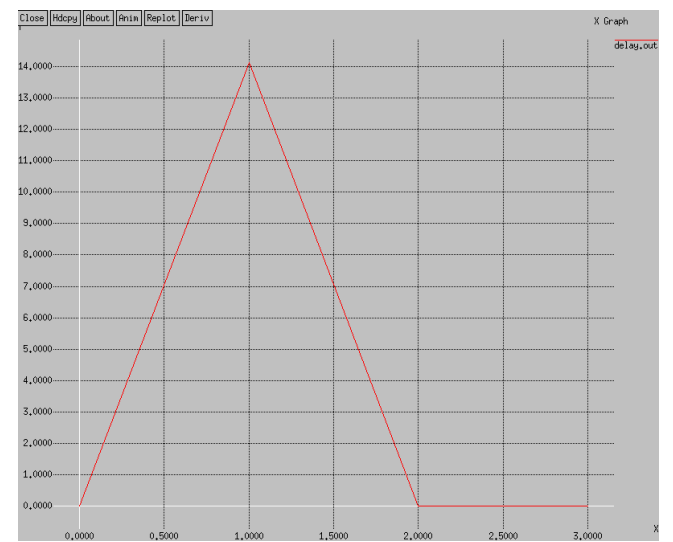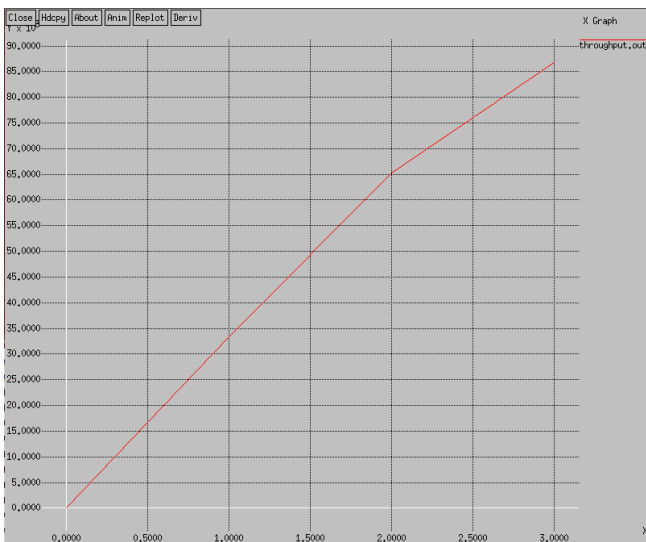
Wired network

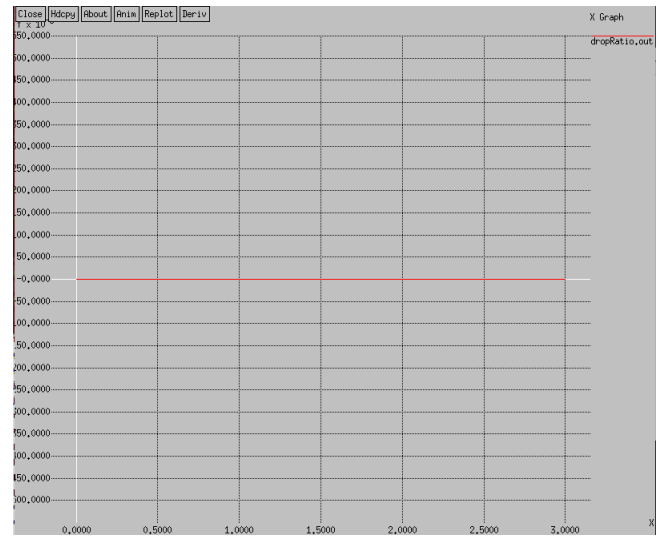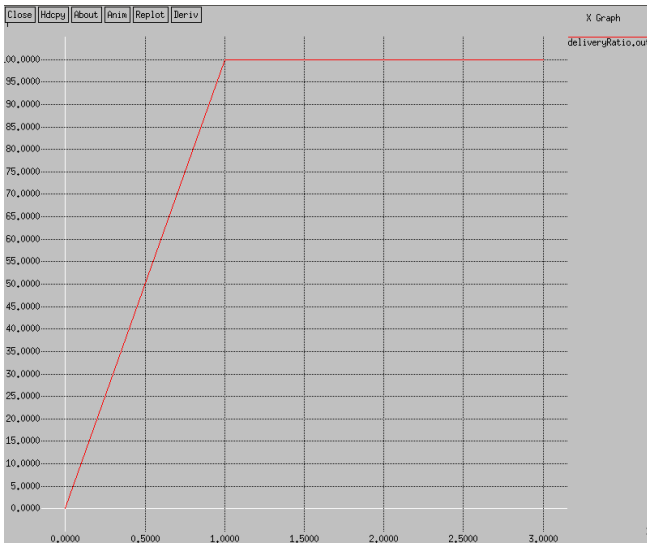Varying nodes:
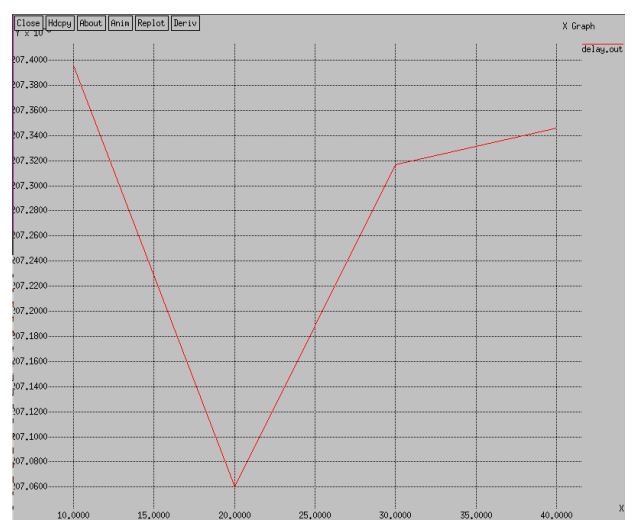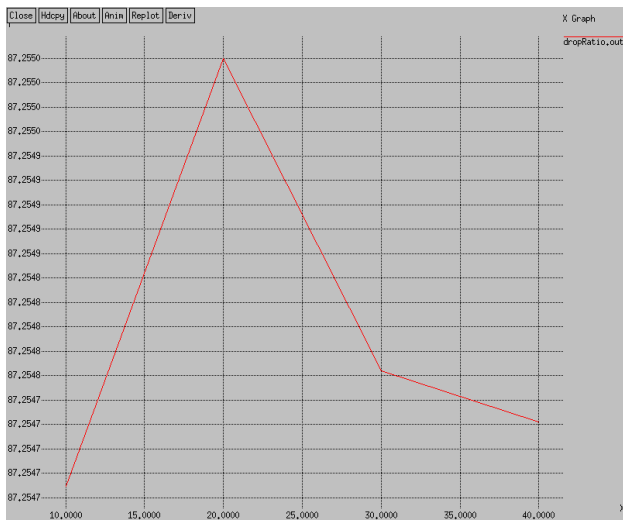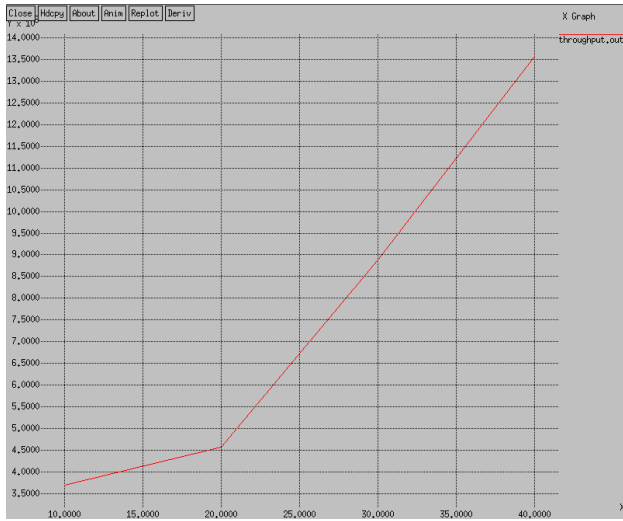
→ Number of nodes

Varying Flow:

➔ Number of Flows

Varying Packet per Second:

→ Packet/s

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.