**CSE-322: NS2 Project**

**By,**

**Md Mahmudul Hasan**

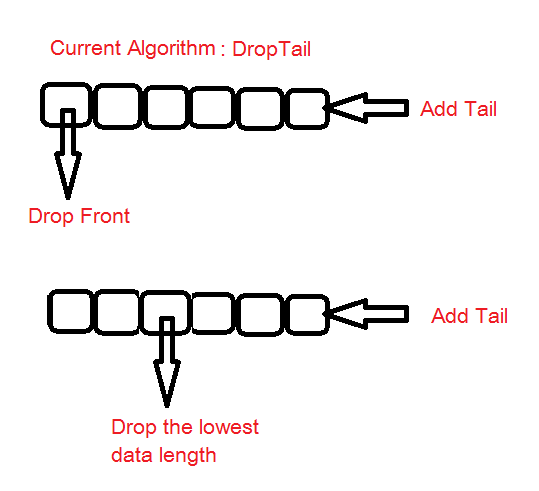
**1205006**

**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;

}

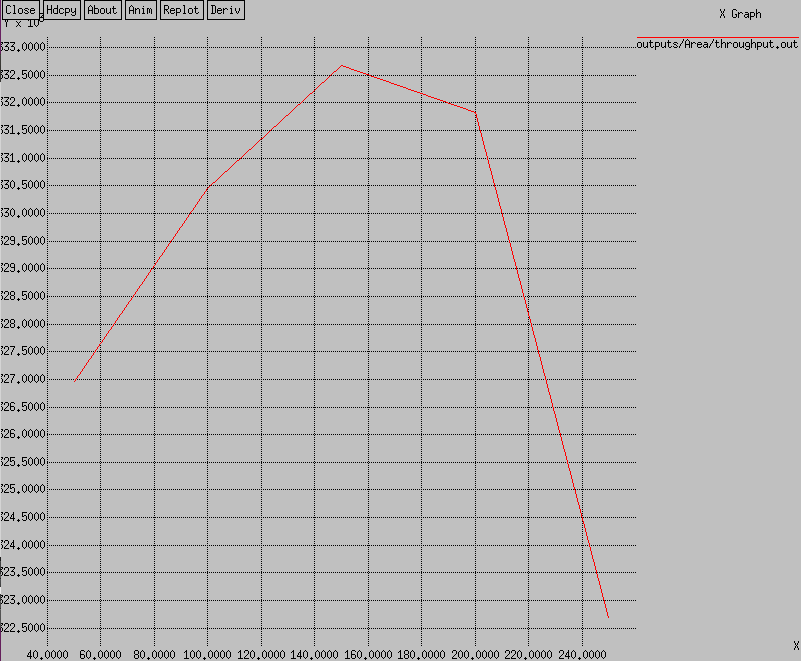
return p;

}

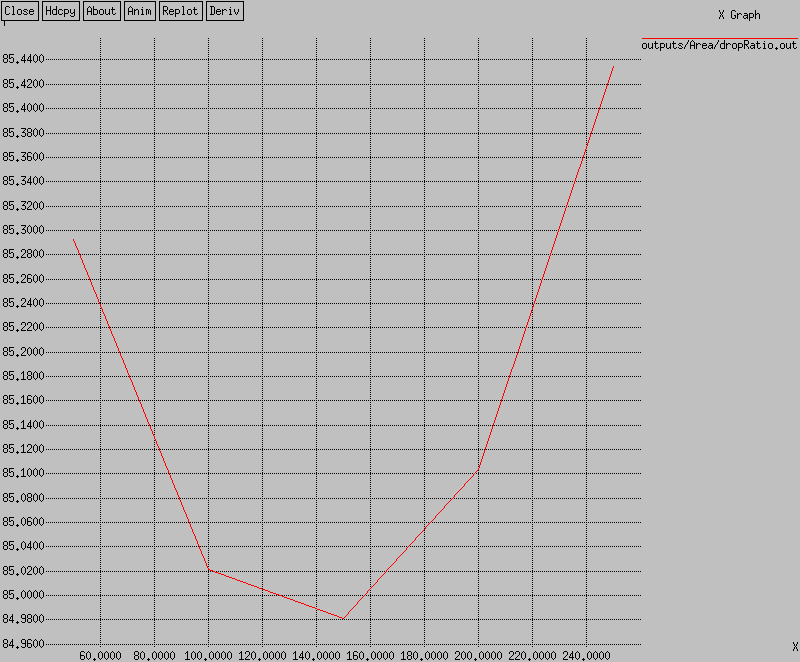
Values before applying the new technique:

**802.11 Wireless Static**

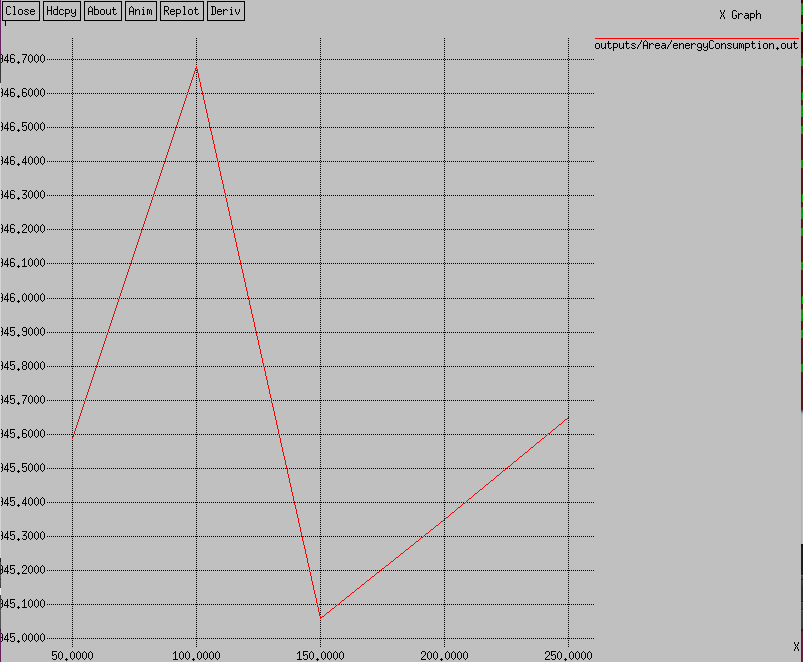
**Varying Area:**

* ms
* Kb/s

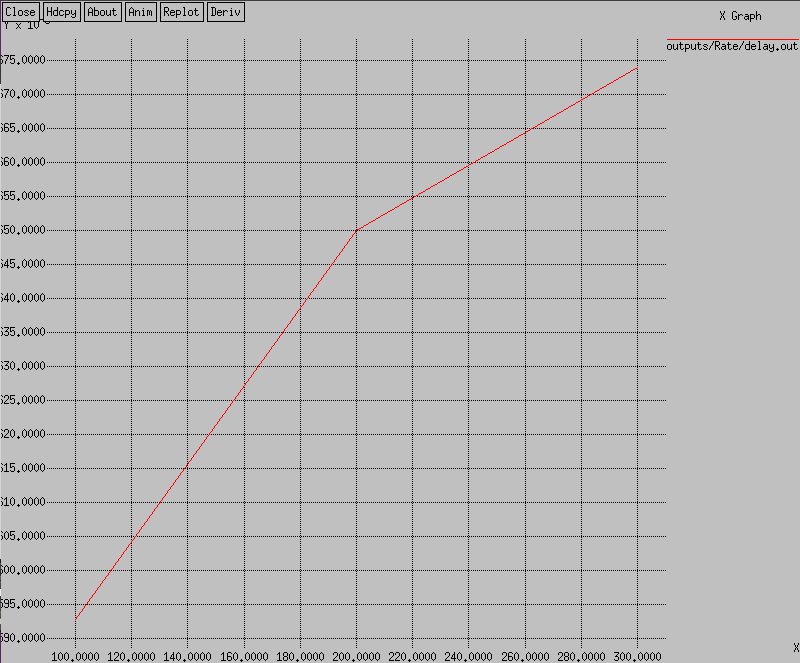
 

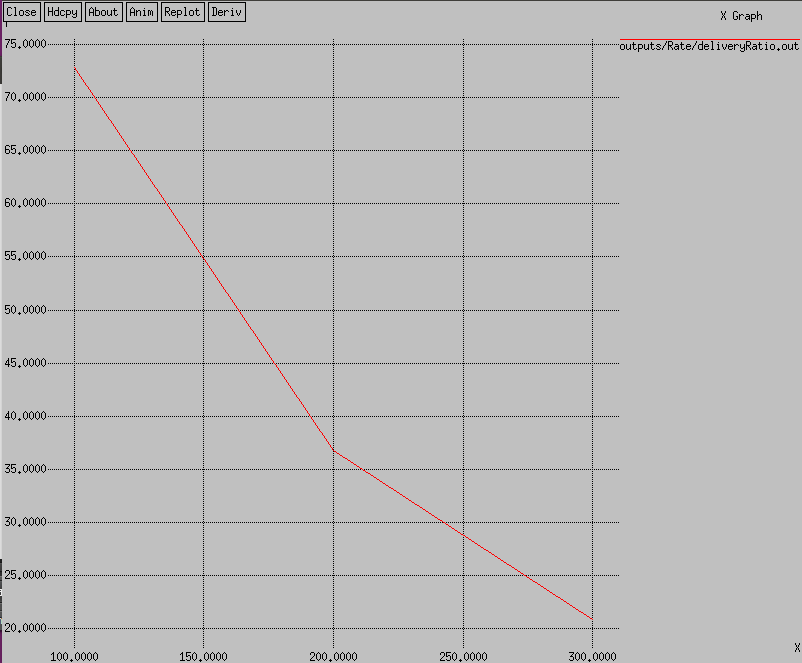
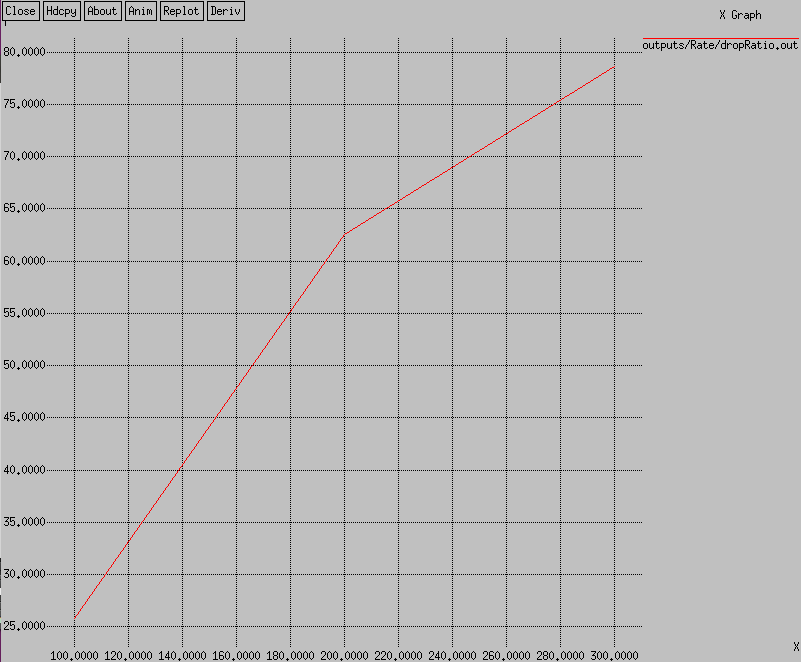
* Kb/s
* Kb/s



* mJ
* Area in meter

Varying Packet per Second:

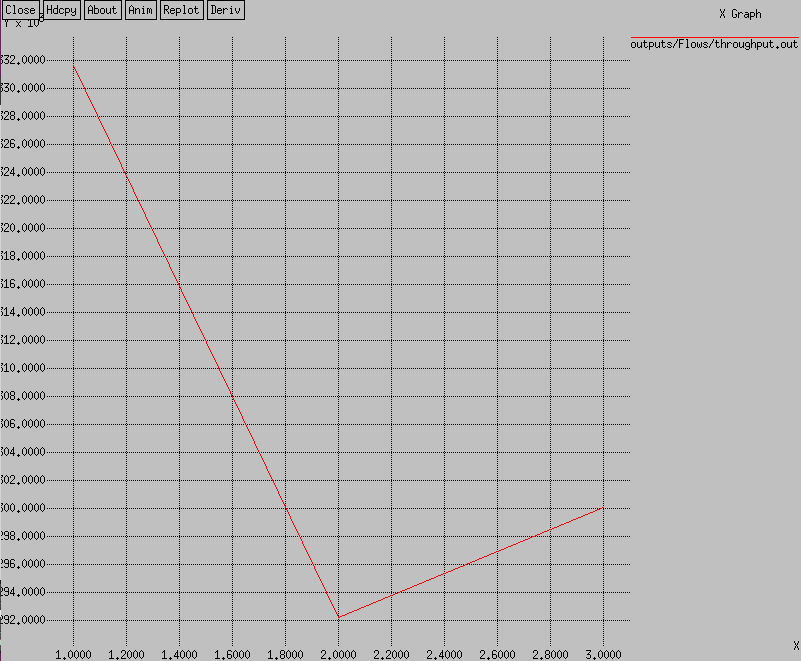
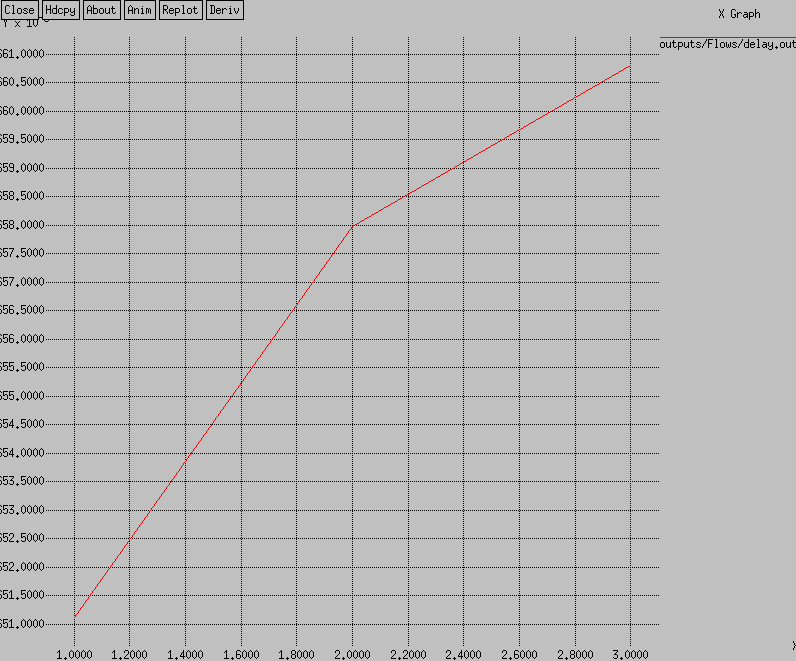
 

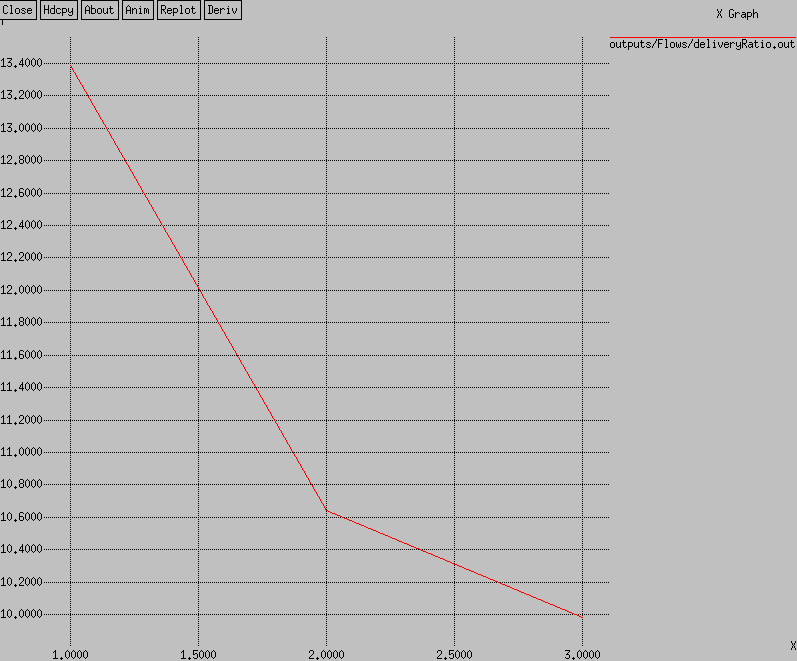
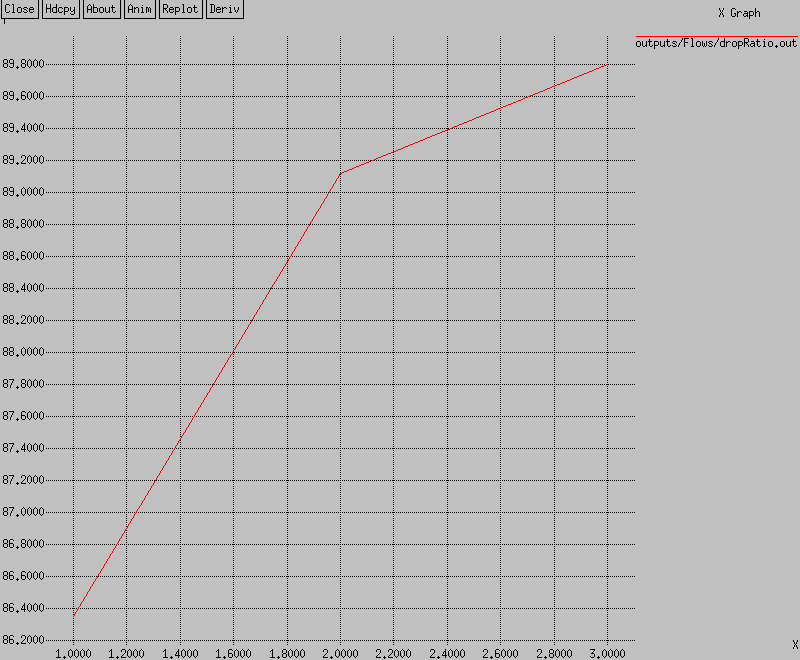
 

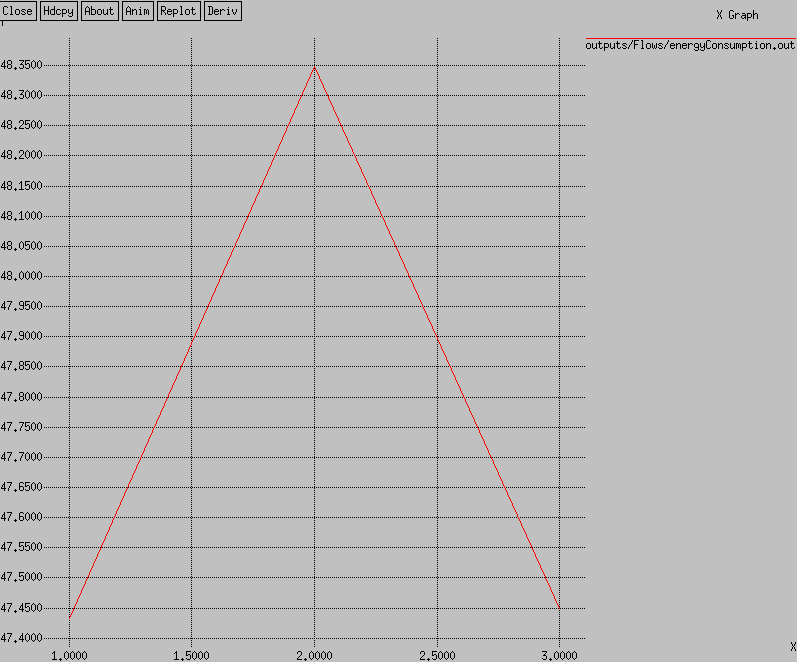


* Packet/s

Varying flow:

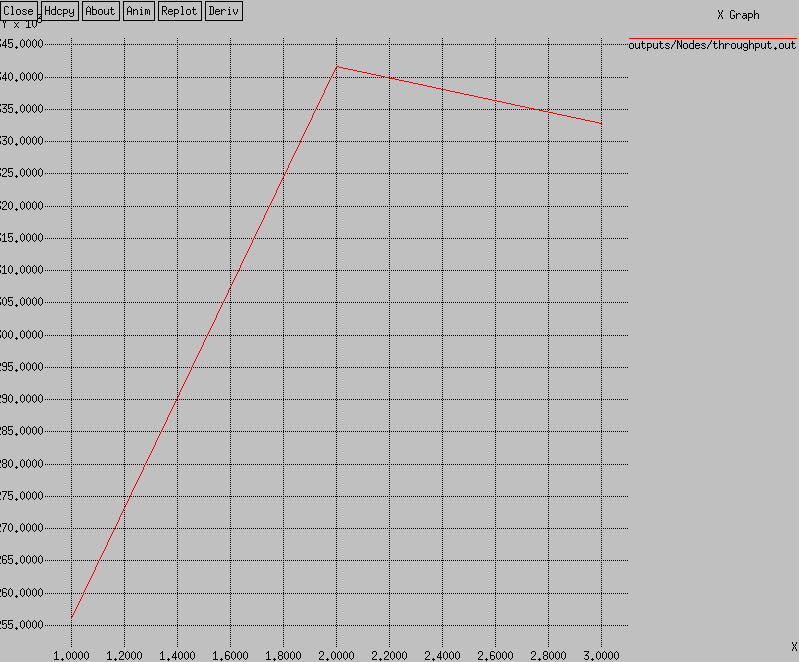
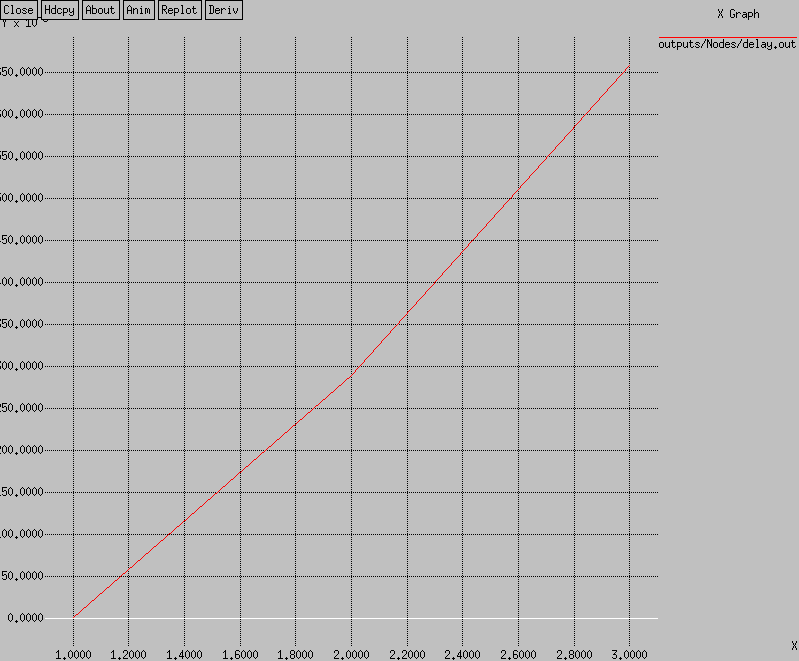
 

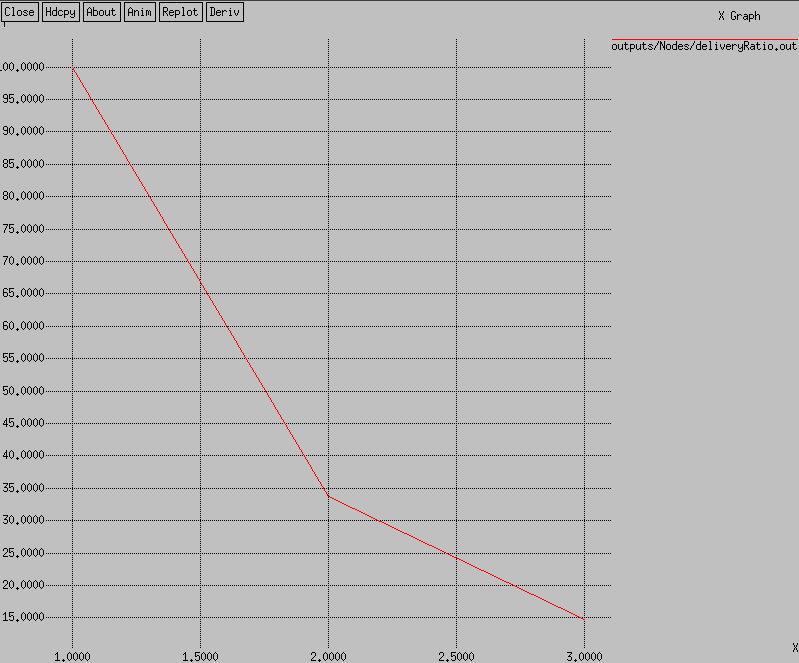
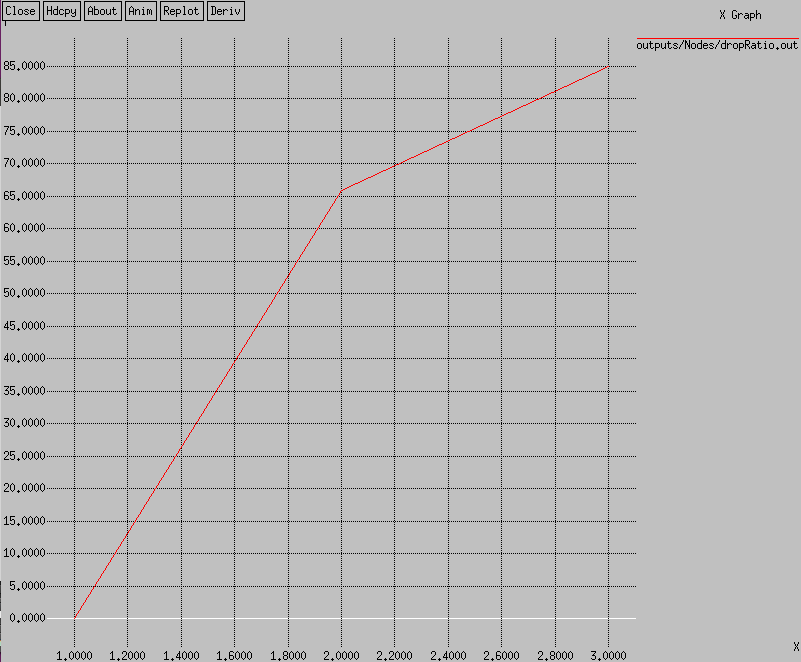
 

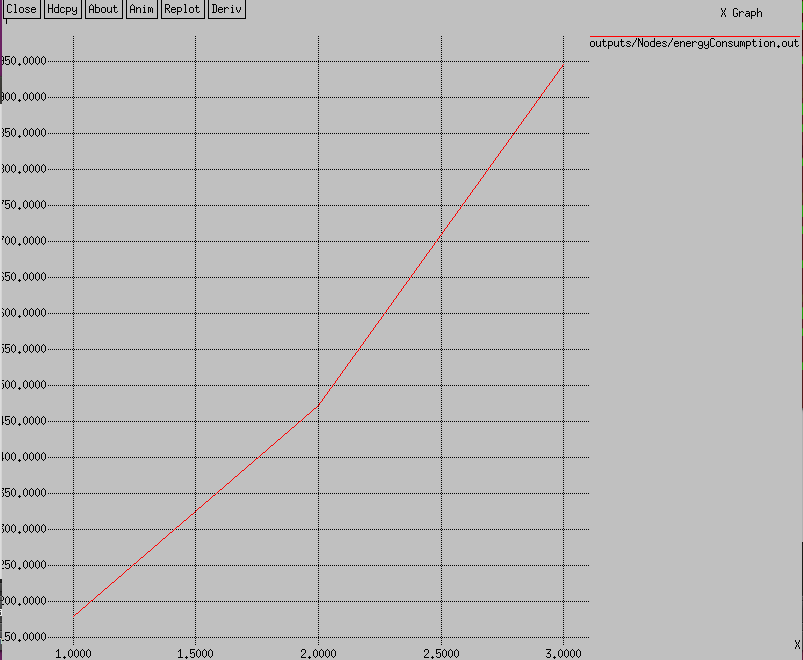


* Number of flow

Varying Nodes:

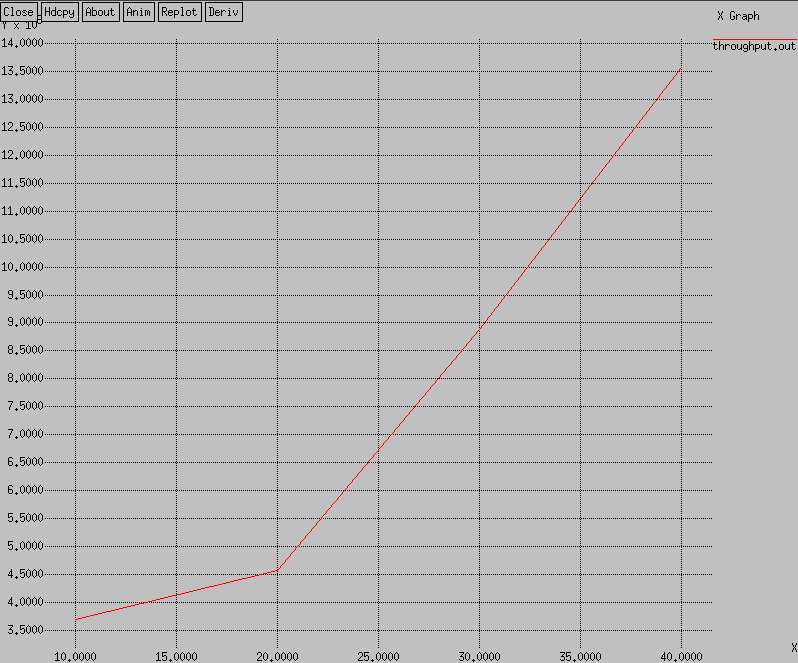
 

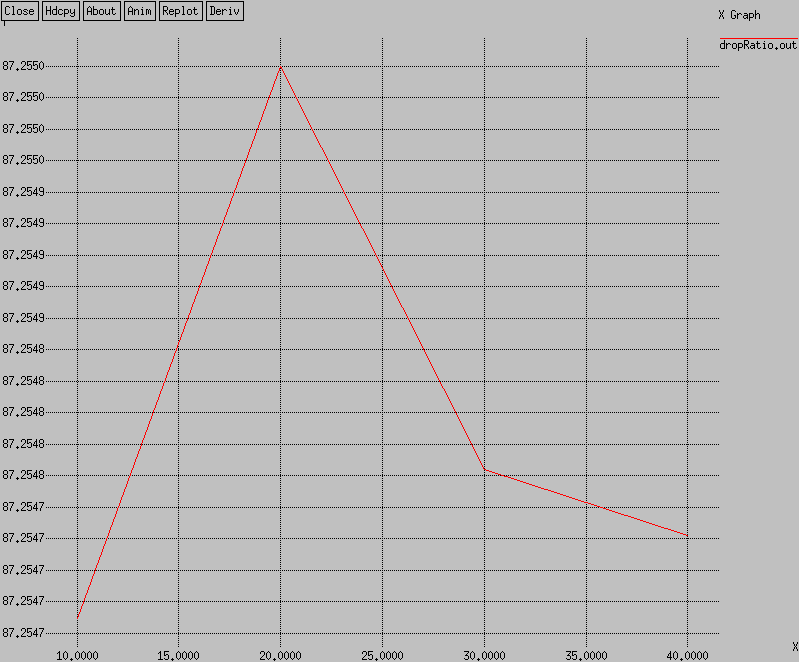


* Number of Nodes

Wired network

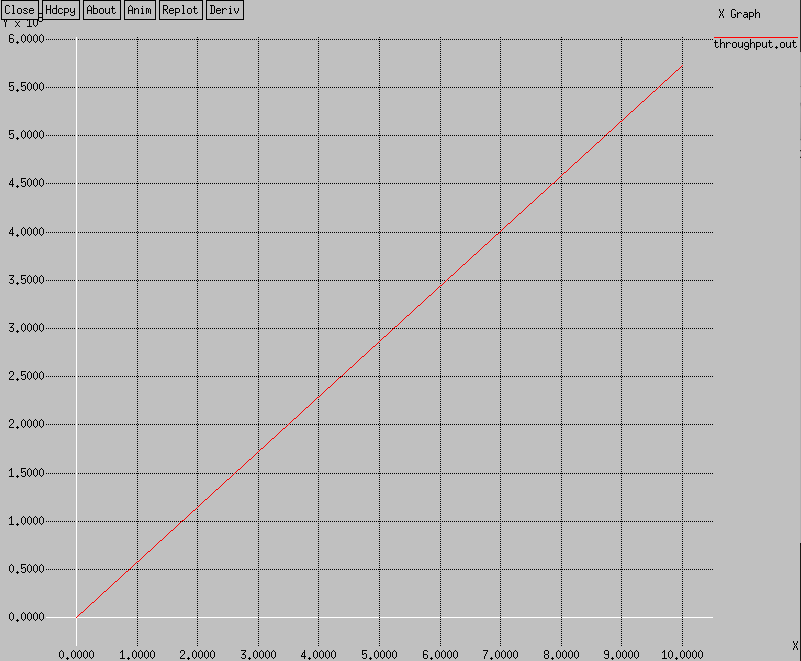
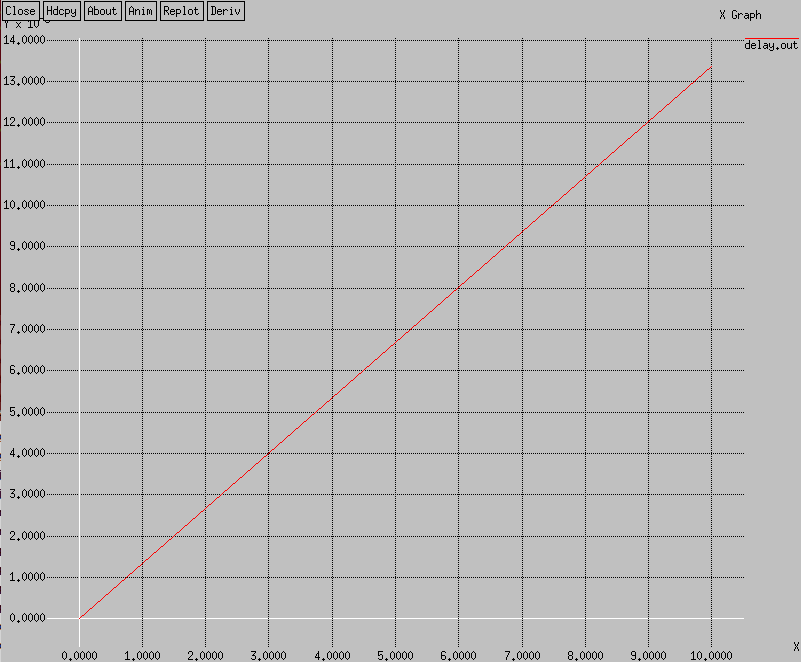
Varying nodes:

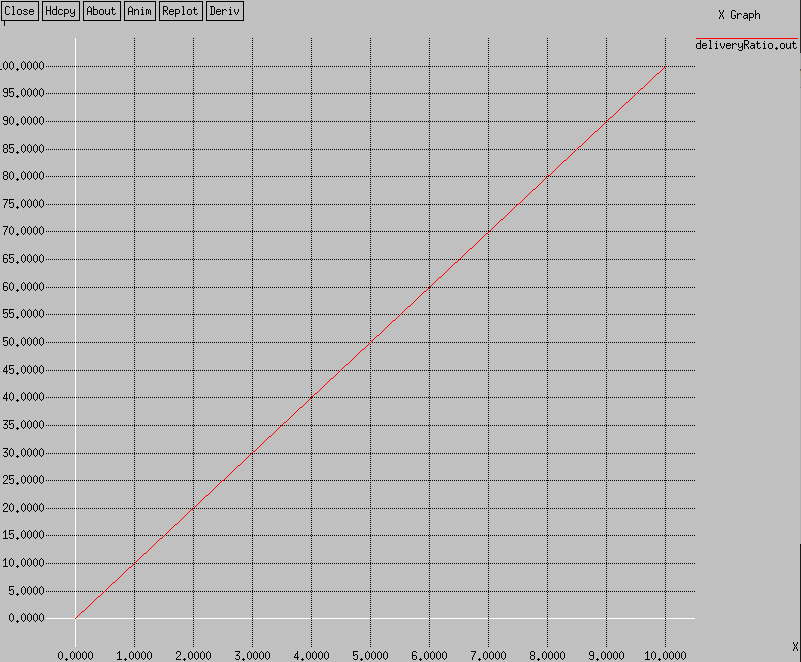
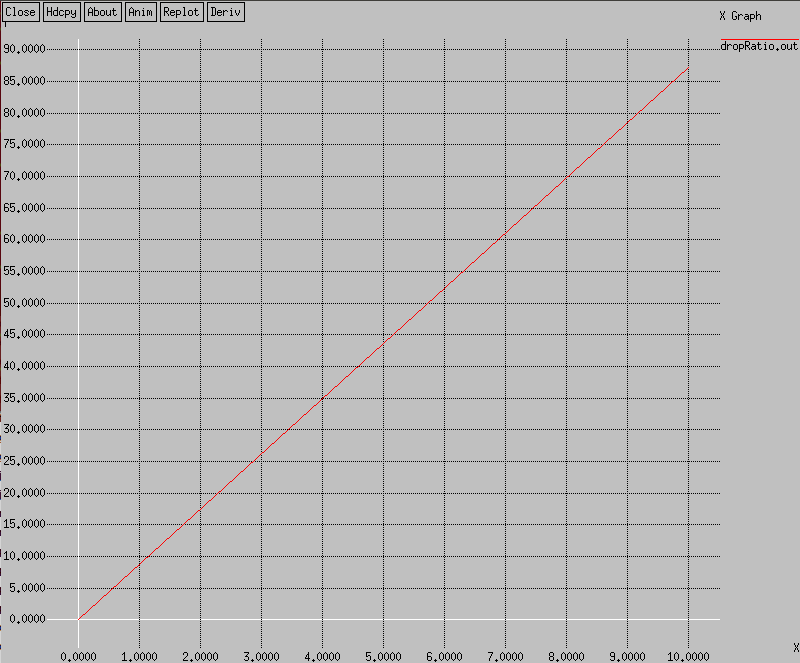
 

* Number of nodes

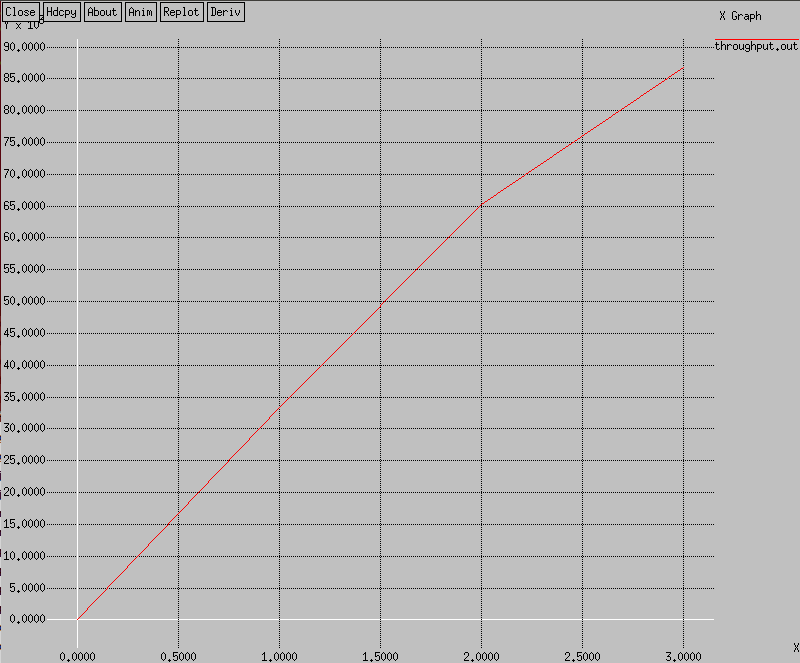
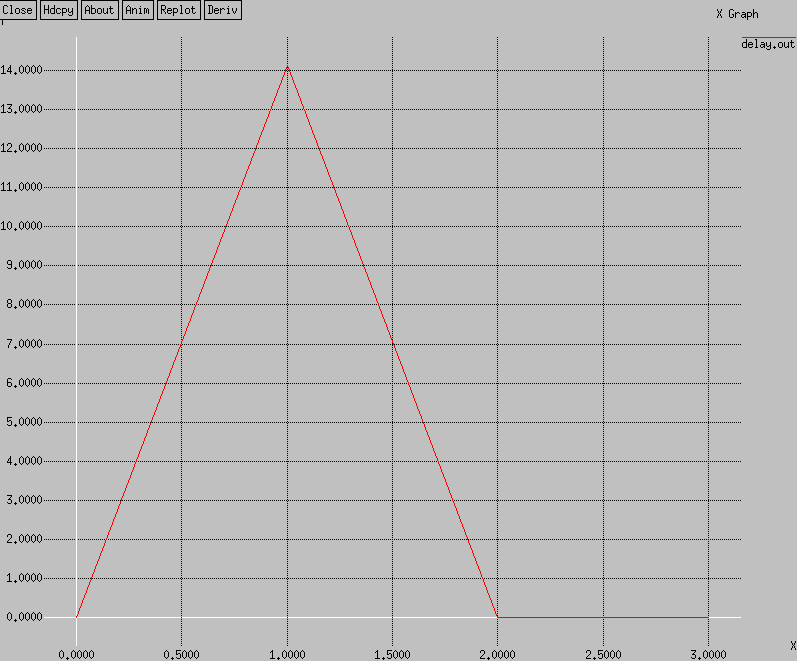
Varying Flow:

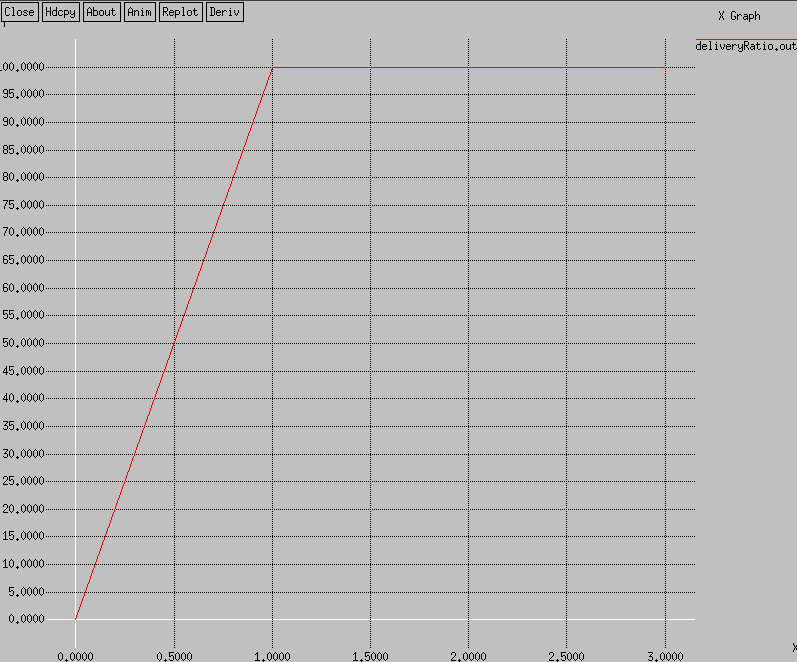
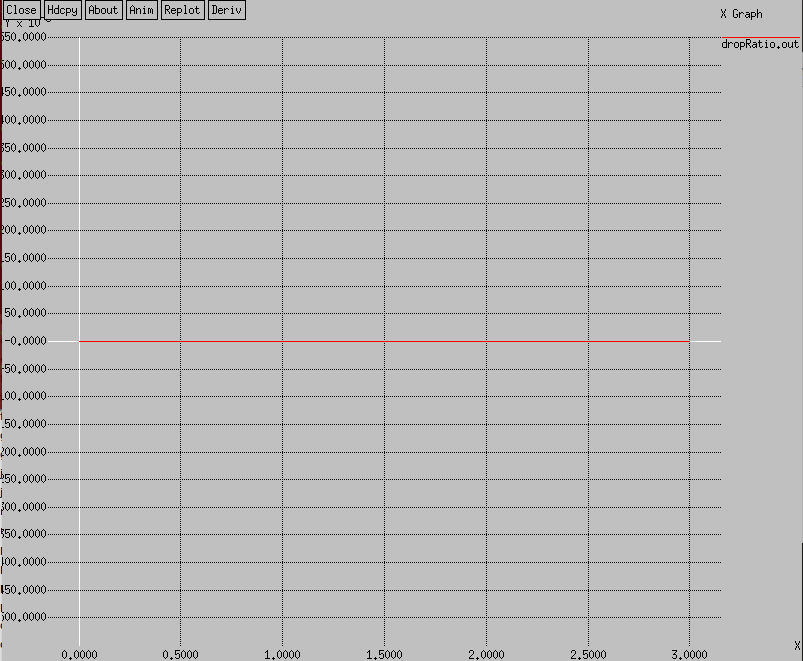
 

* Number of Flows

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.