Ns2 Report

Submitted by
Sanjay Malakar
Shoumik Saha

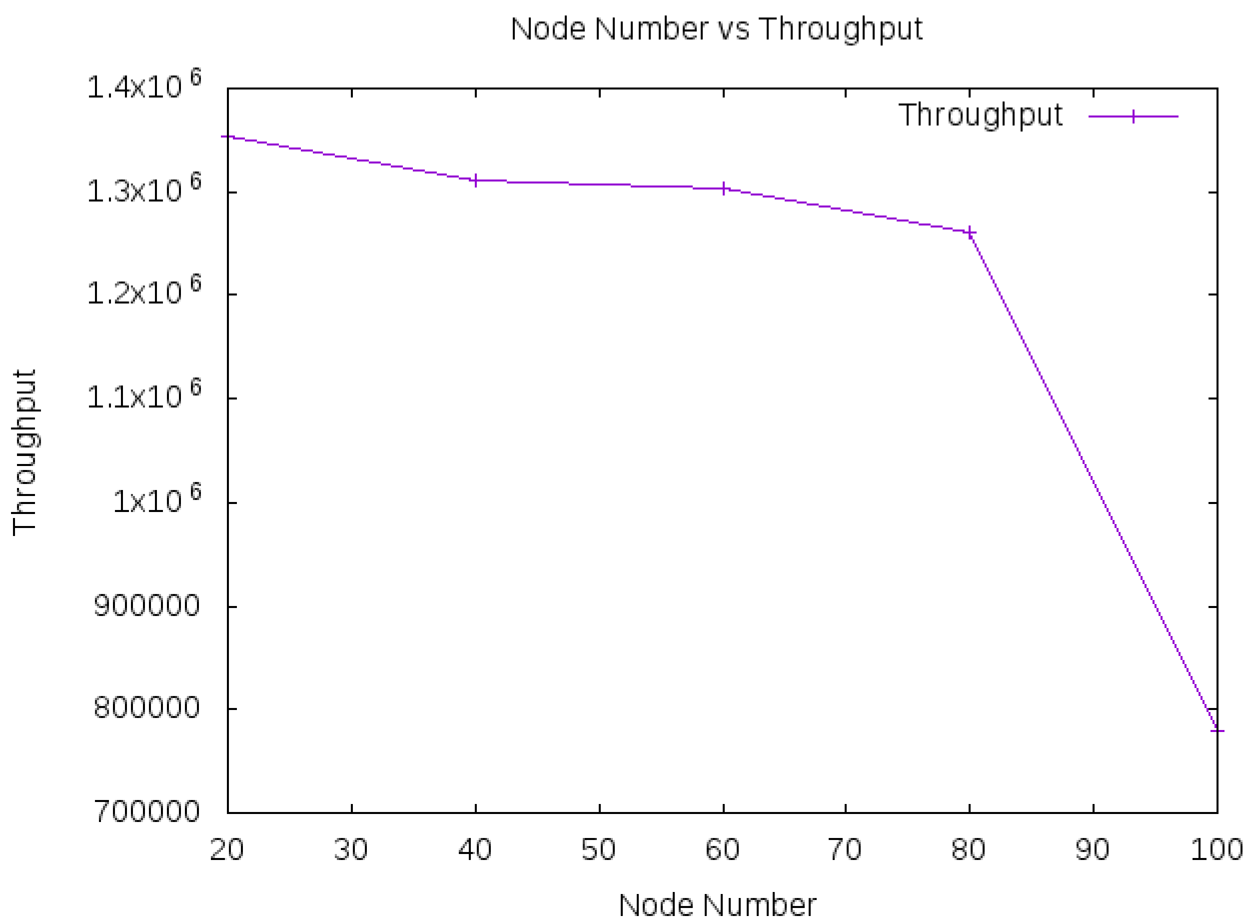
802.11

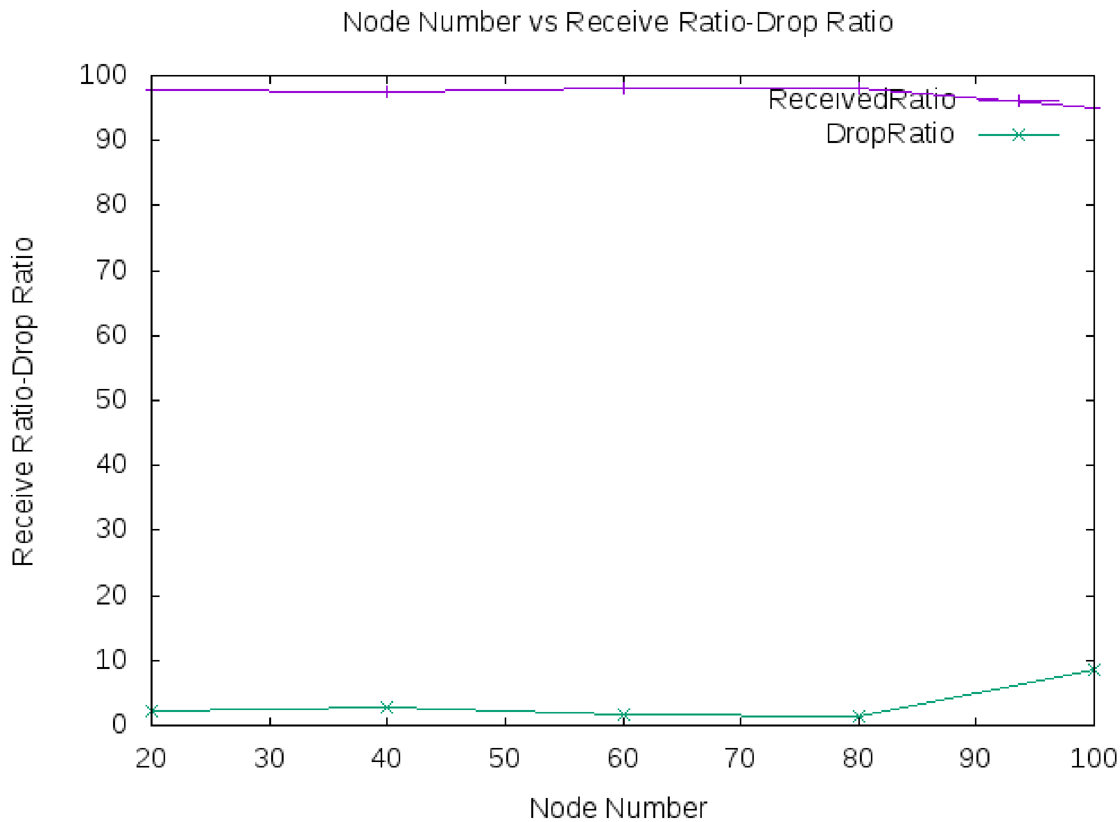
Varying nodes with constant flow,packetrate and txrange :

Flow=20, PacketPerSecond=100
and Range=400

Node 20,40,60,80,100

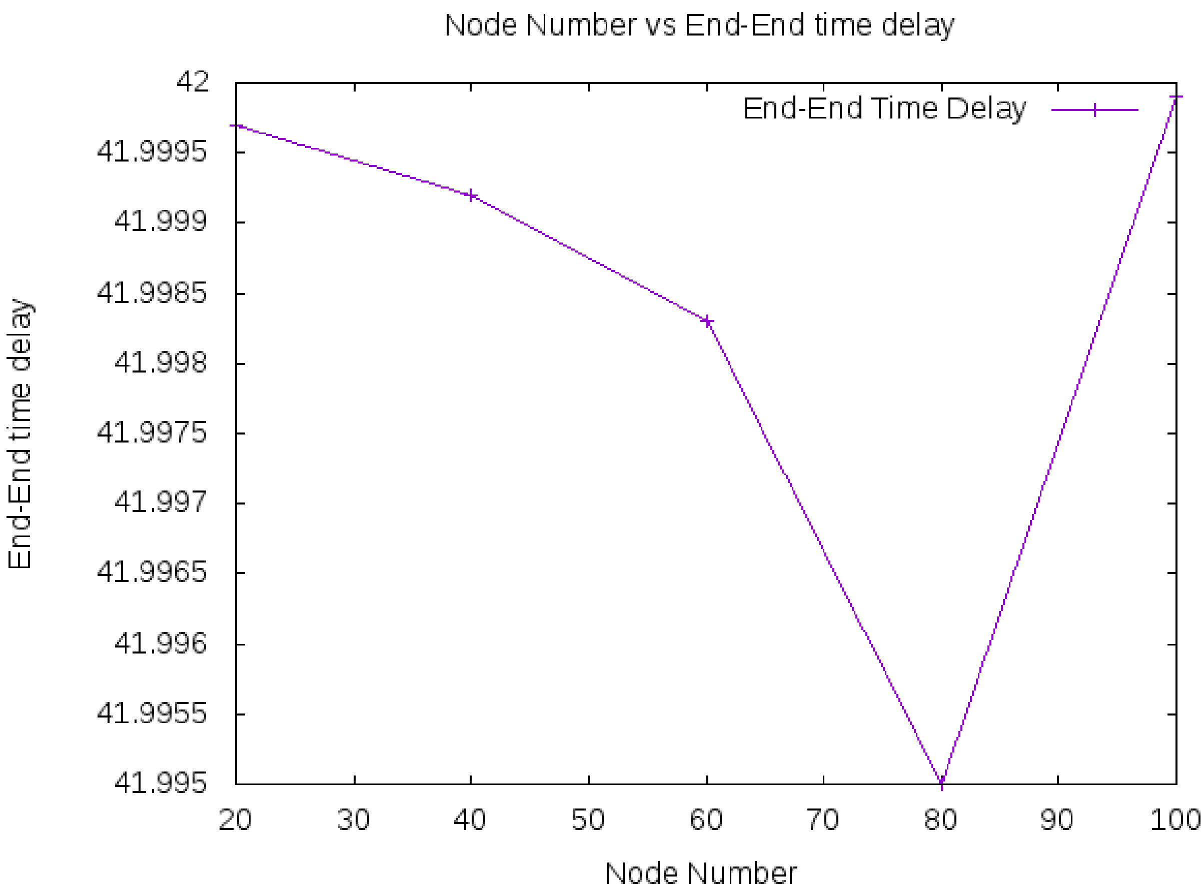
Node vs Throughput



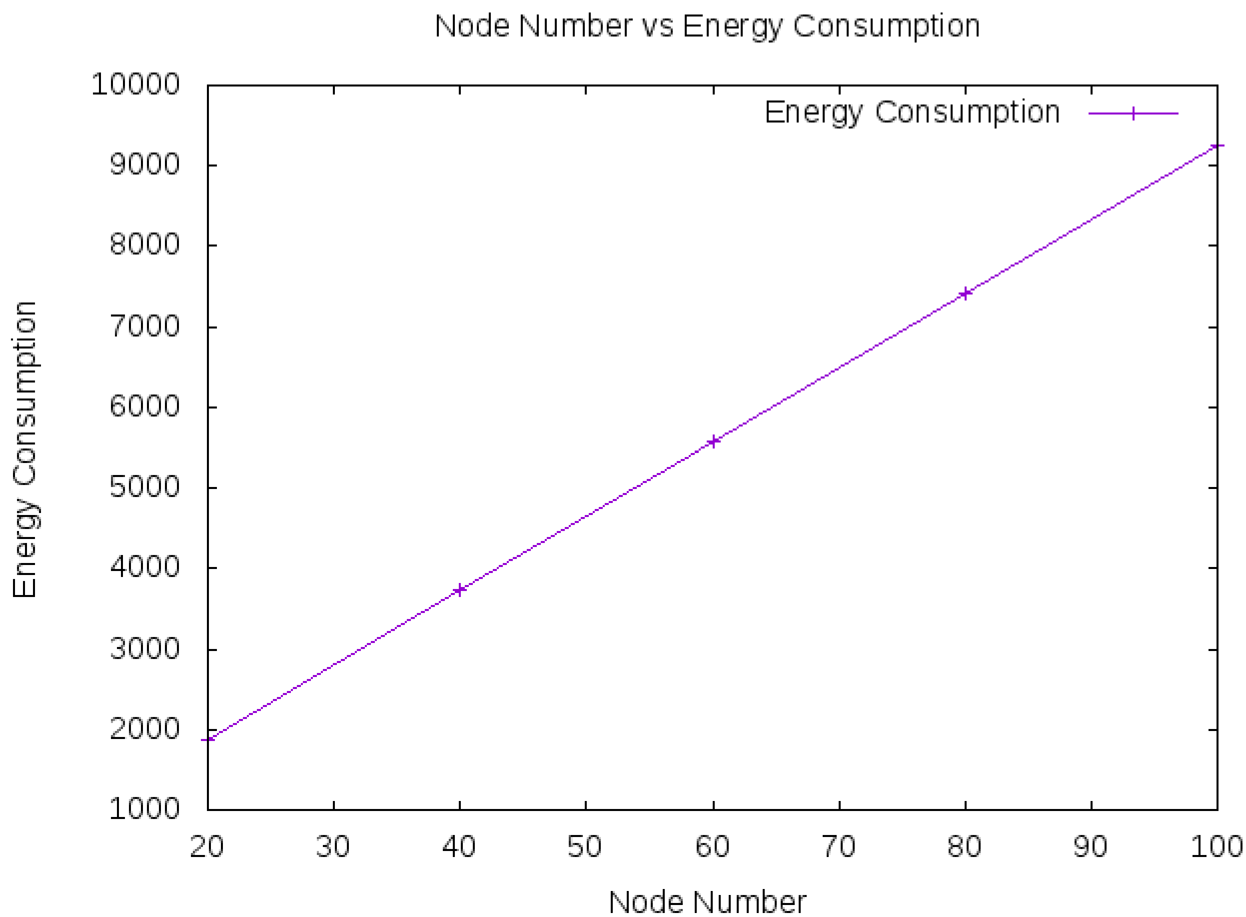


Node Number vs Receive Ratio-Drop Ratio
ReceivedRatio
DropRatio
Receive Ratio-Drop Ratio
Node Number
0
10
20
30
40
50
60
70
80
90
100
20
30
40
50
60
70
80
90
100

Node vs End to End delay



Node vs Energy Consumption

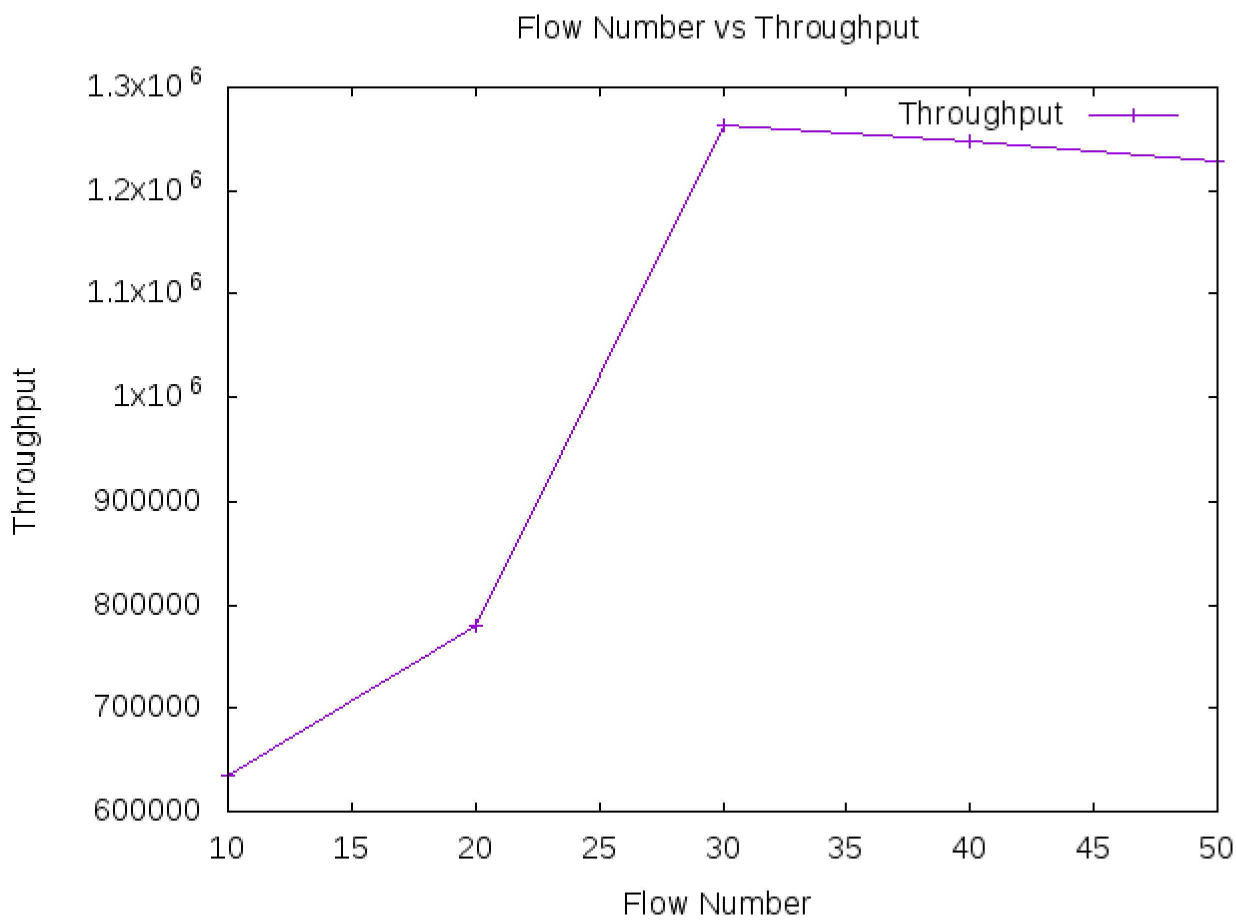


Varying Flow

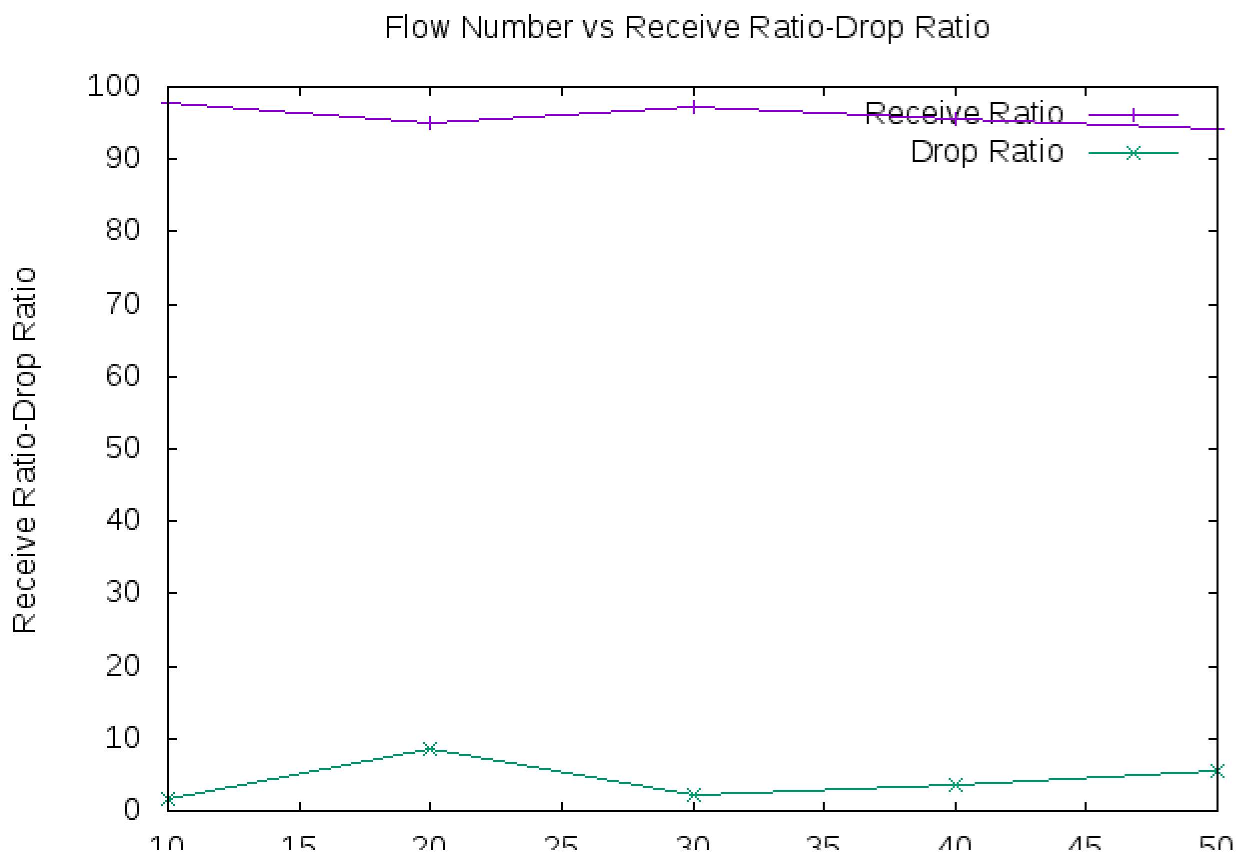
Node = 20, Speed =100, Range = 40

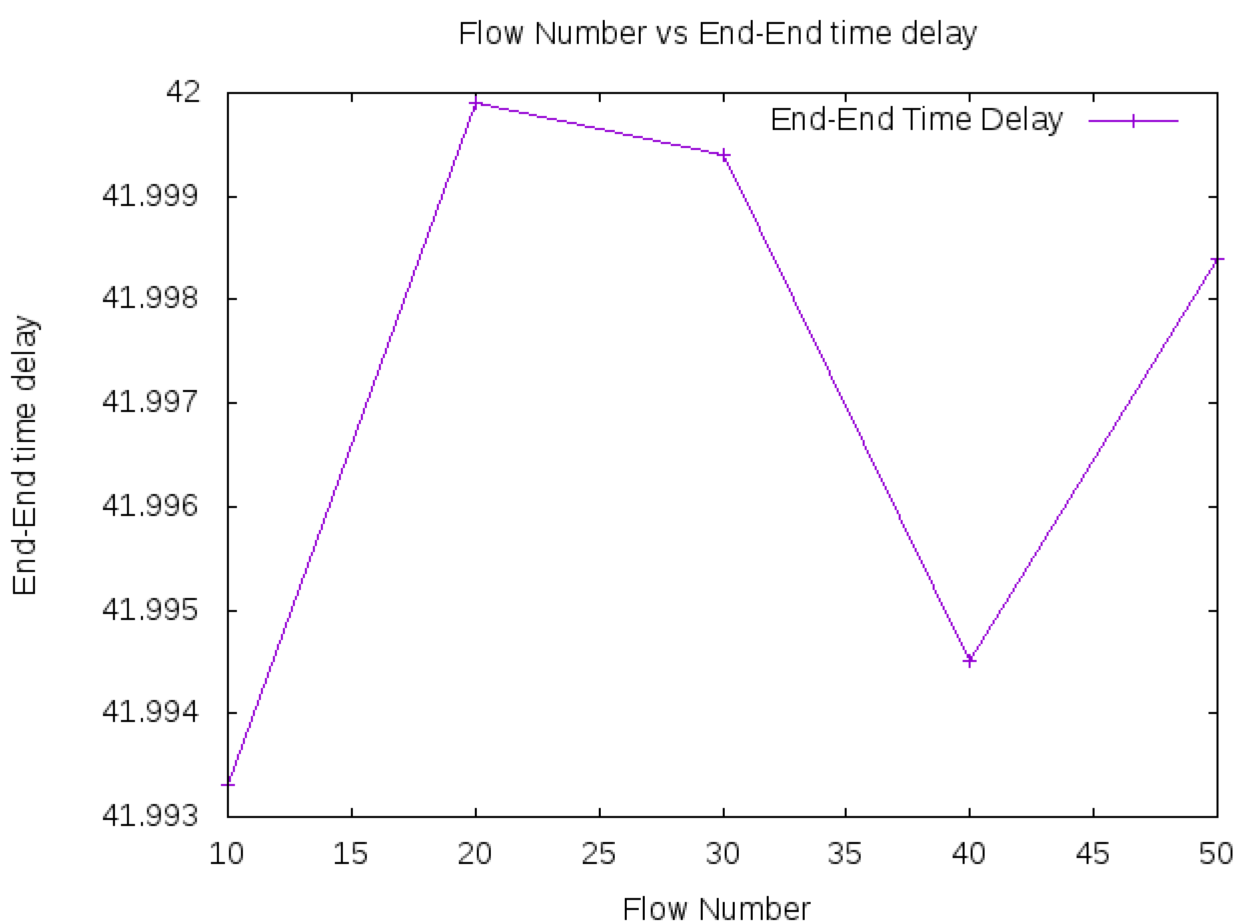
Flow 10,20,30,40,50

Flow vs Throughput

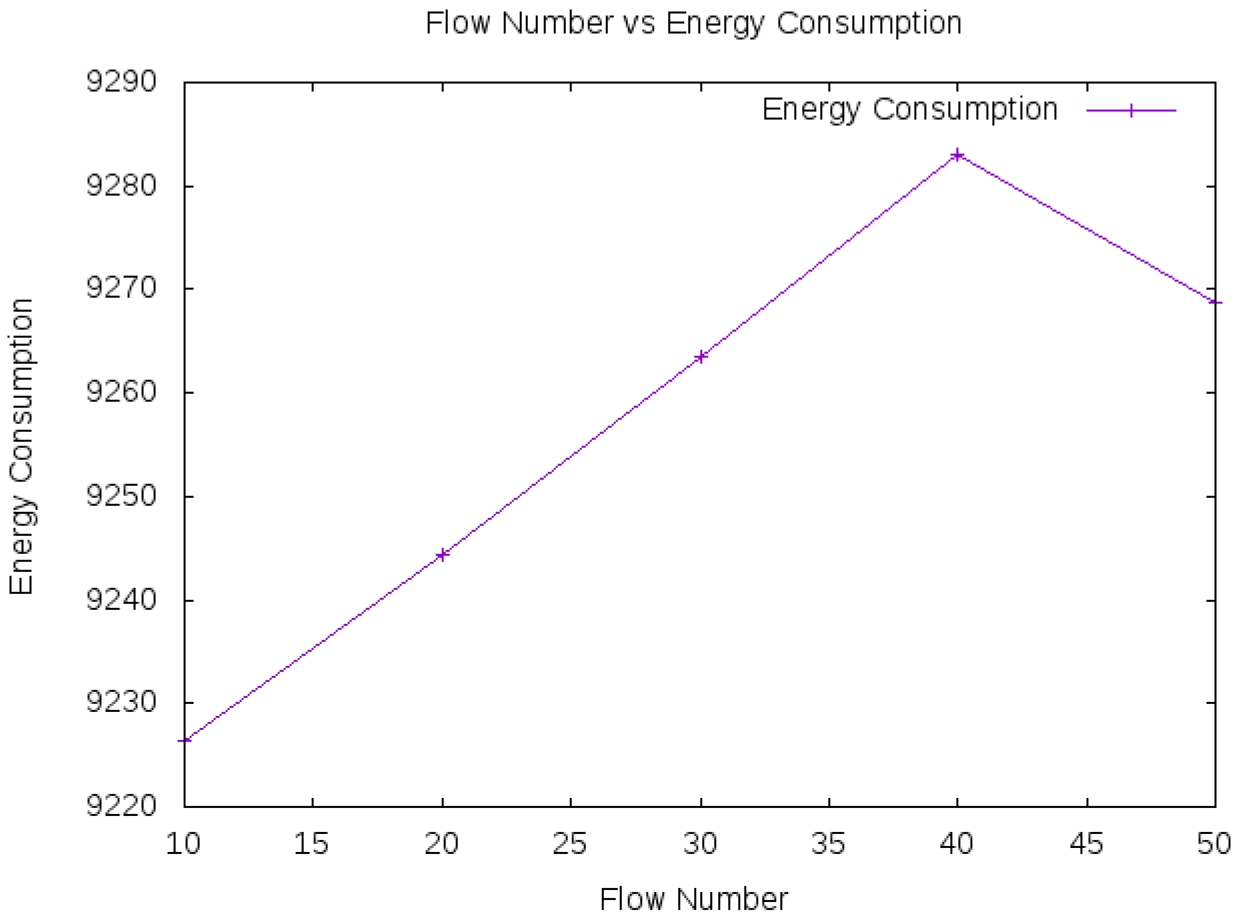


Flow vs Packet Send/Drop Ration





Flow vs Energy Consumption

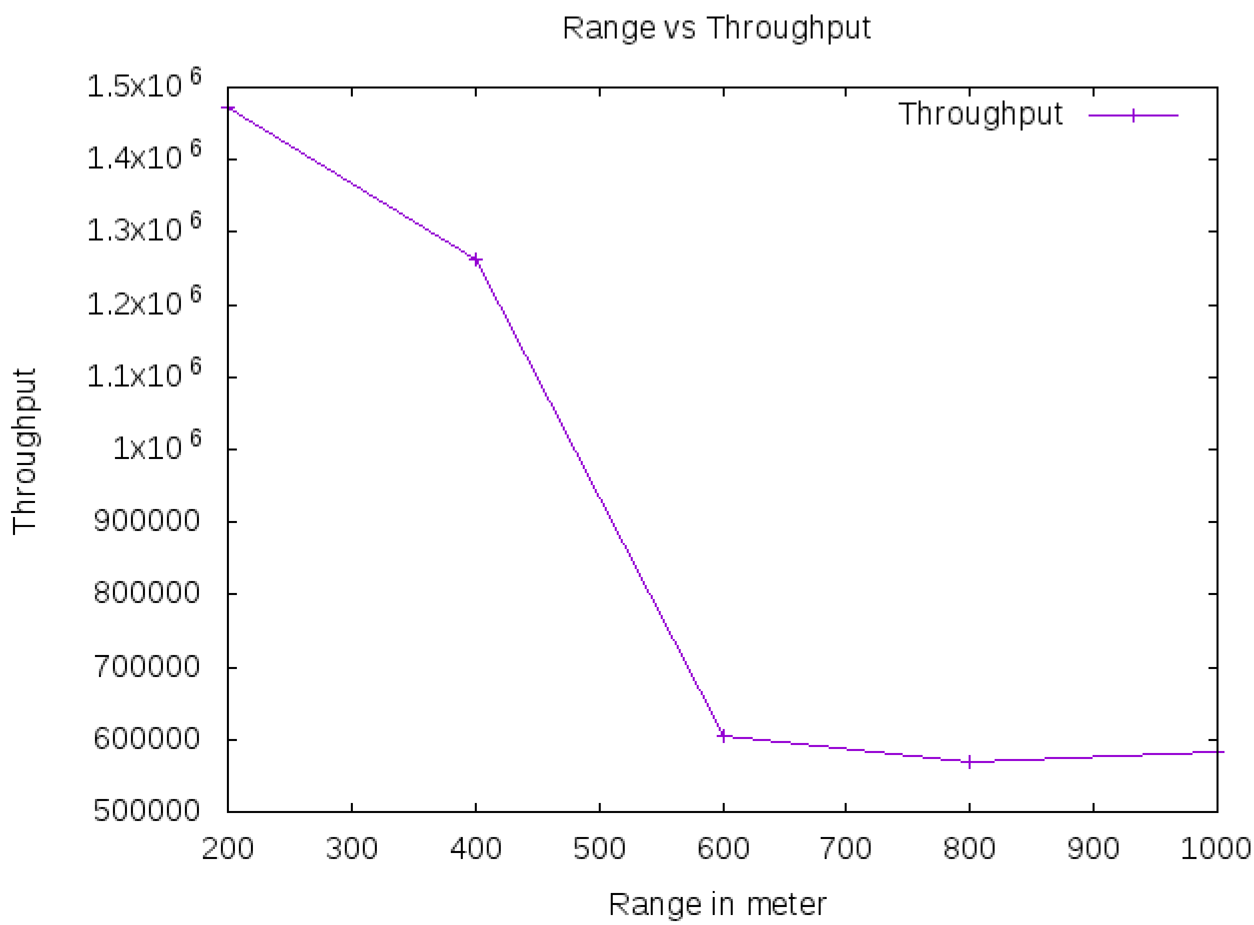


Varying Packet per second

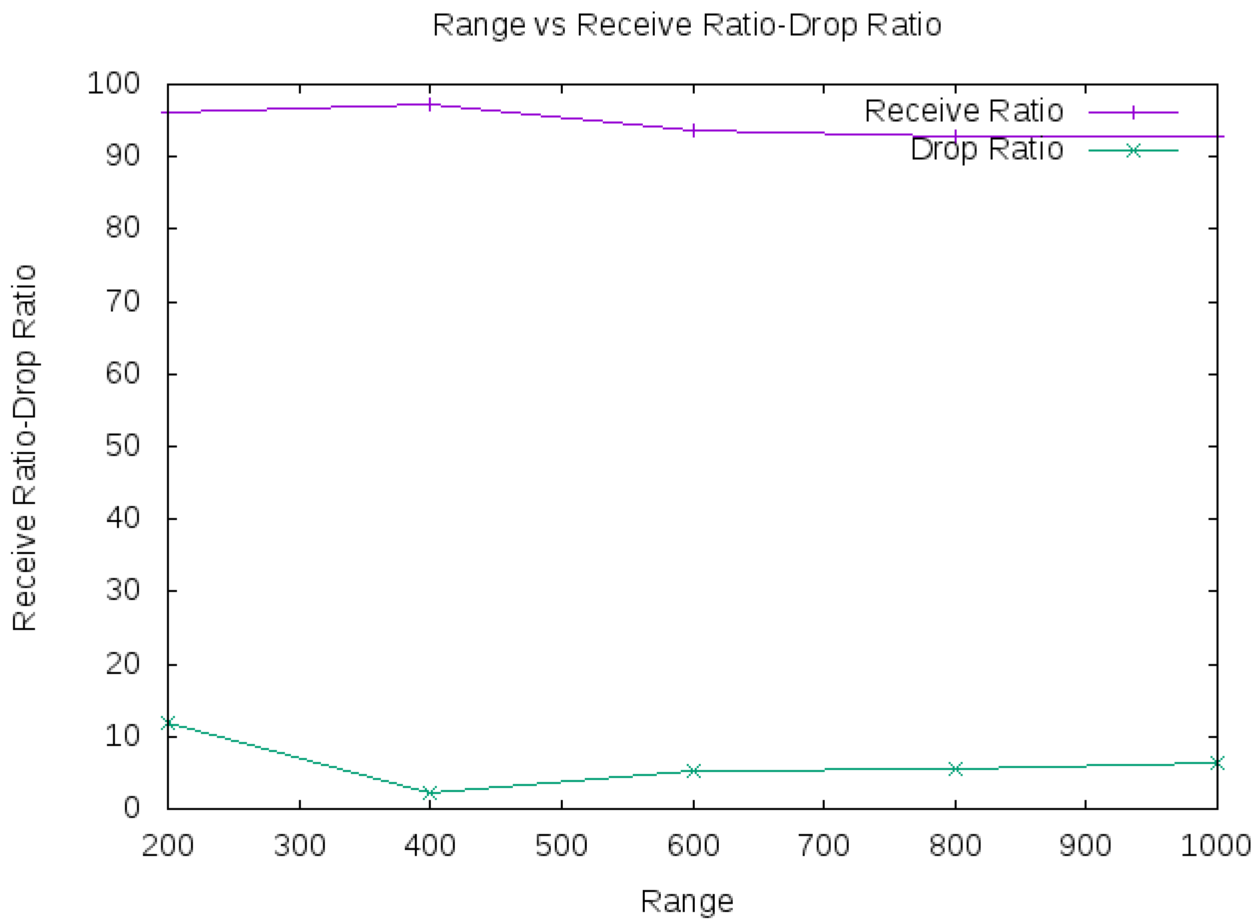
Node : 60 Flow :20 Range :400

Packet per second : 20,40,60,80,100

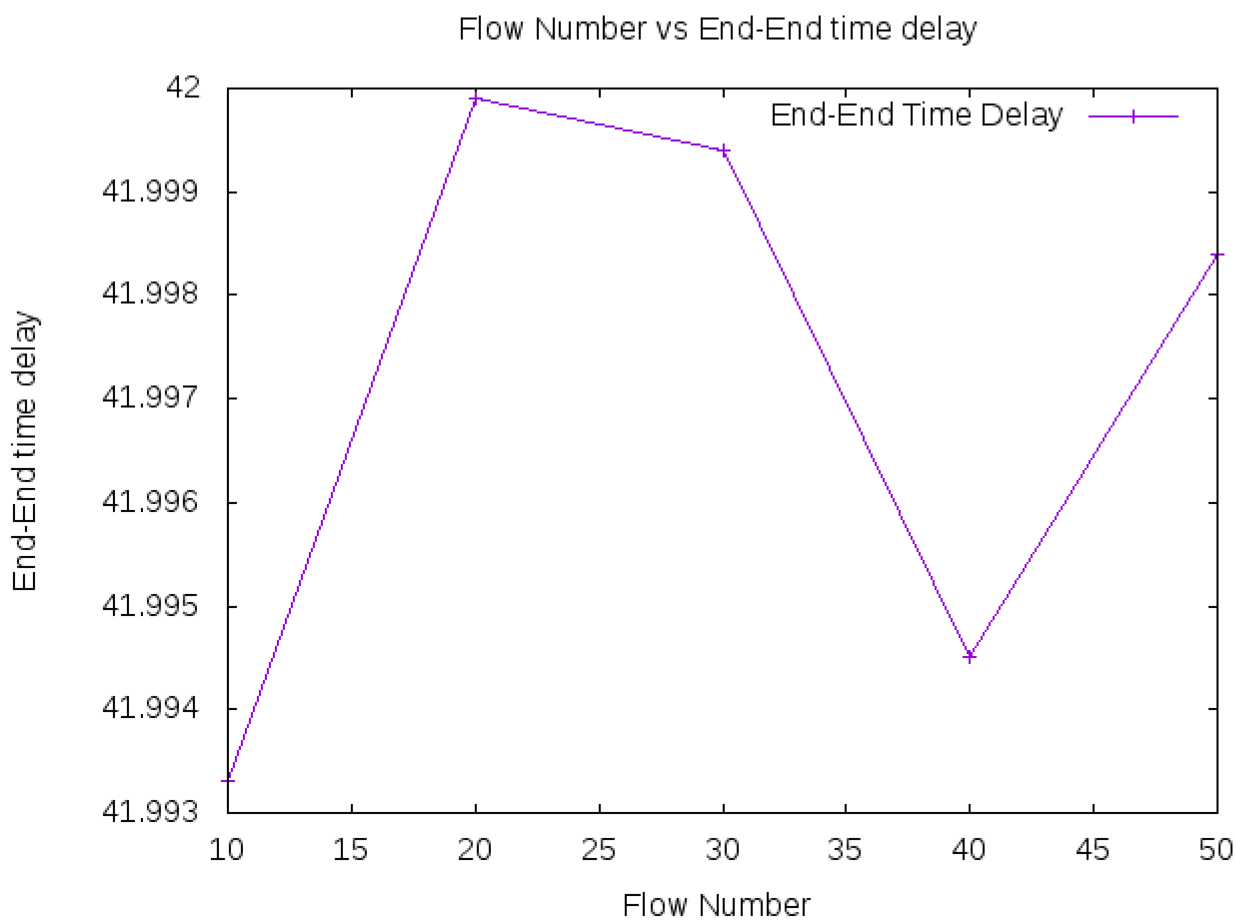
Packet per second vs Throughput



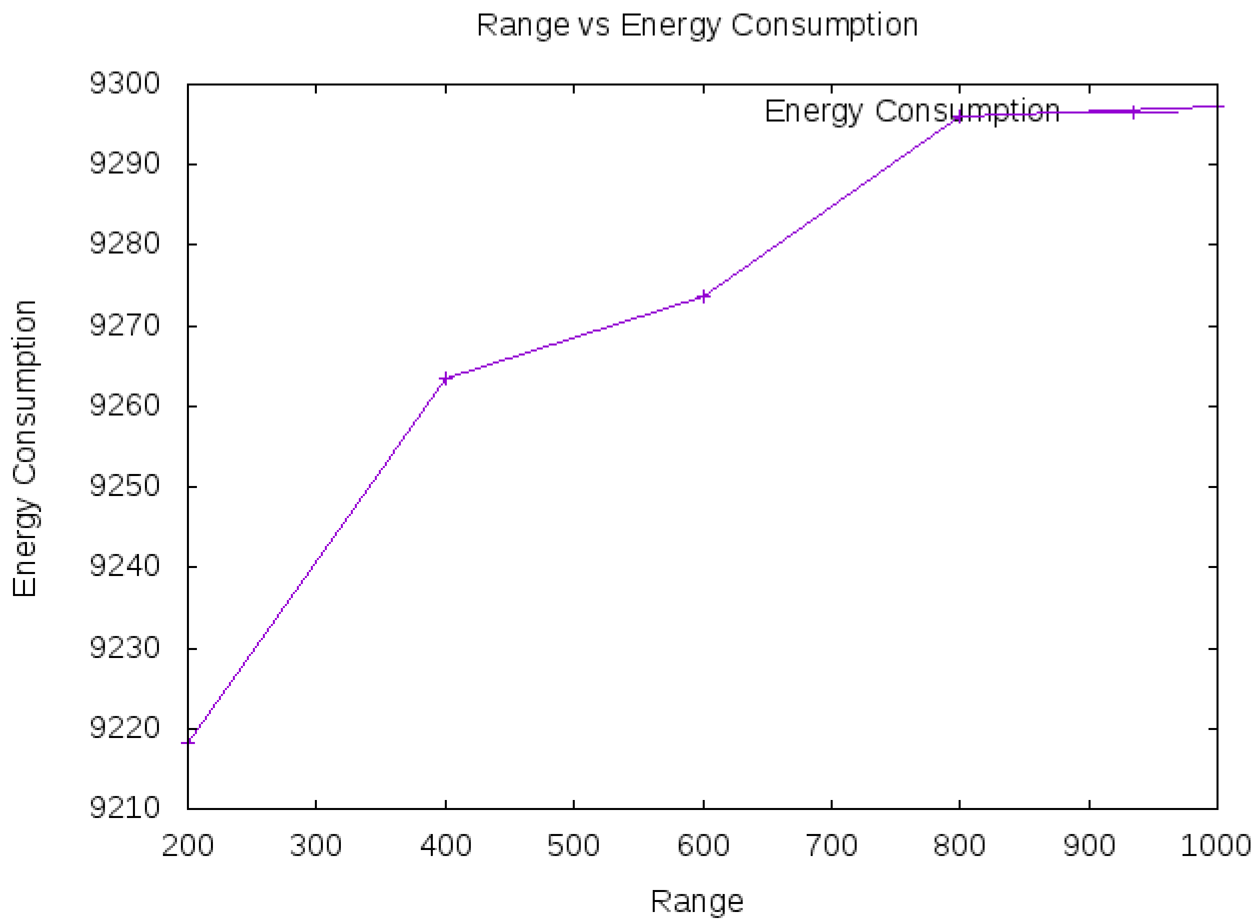
Packet per second vs Packet send/drop ration



Packets per second vs End to end time



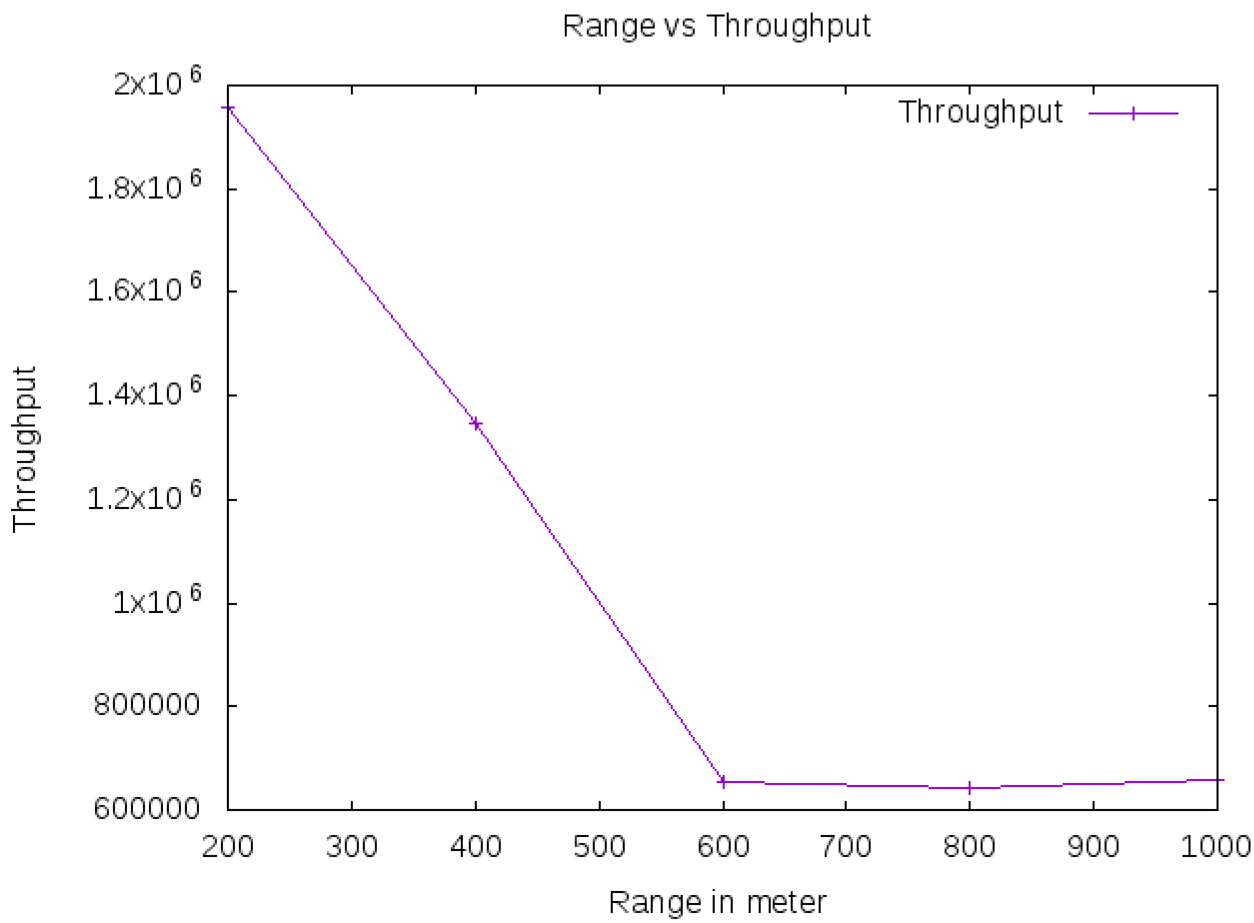
Packets per second vs Energy consumption



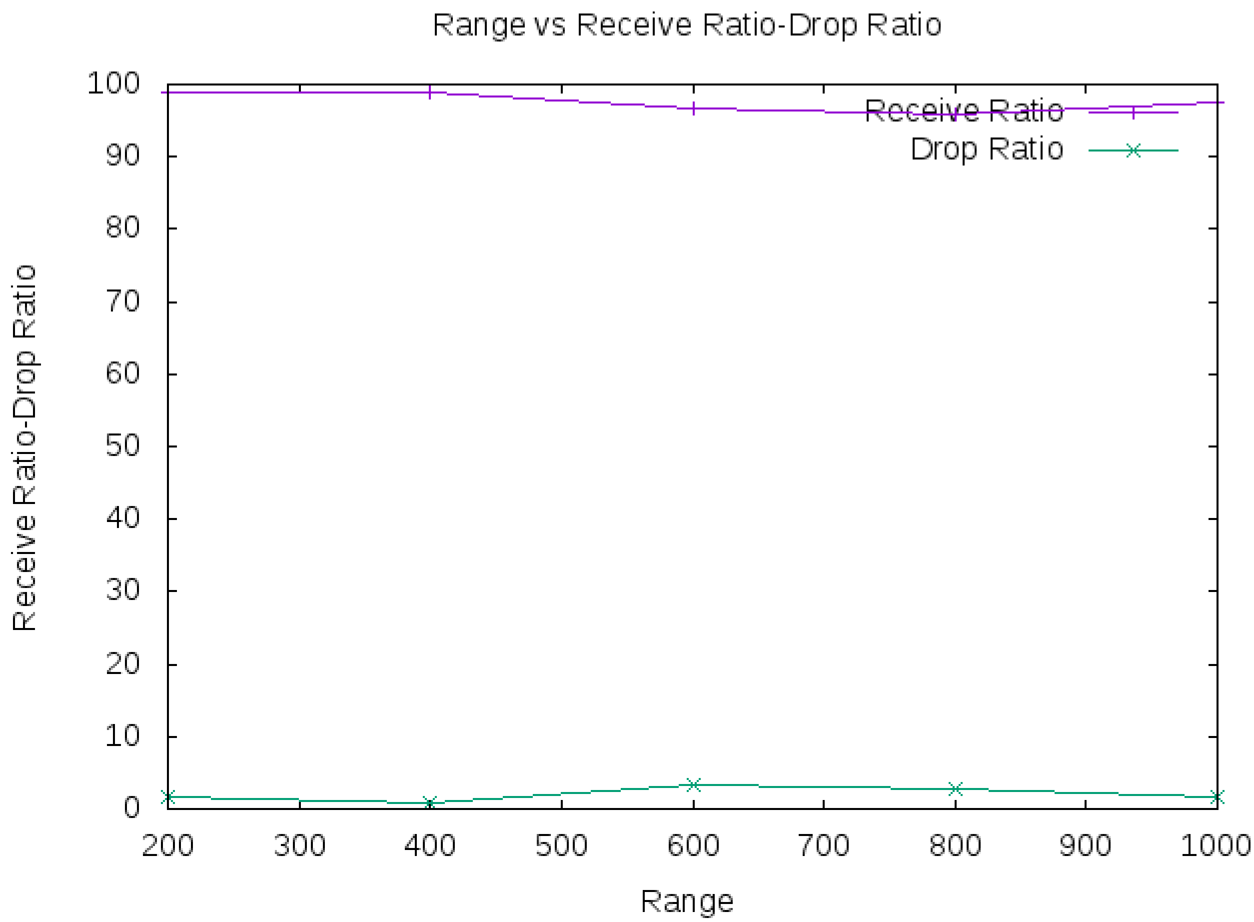
Varying Transmission range

Node 40, Flow 10, Packet per second 100

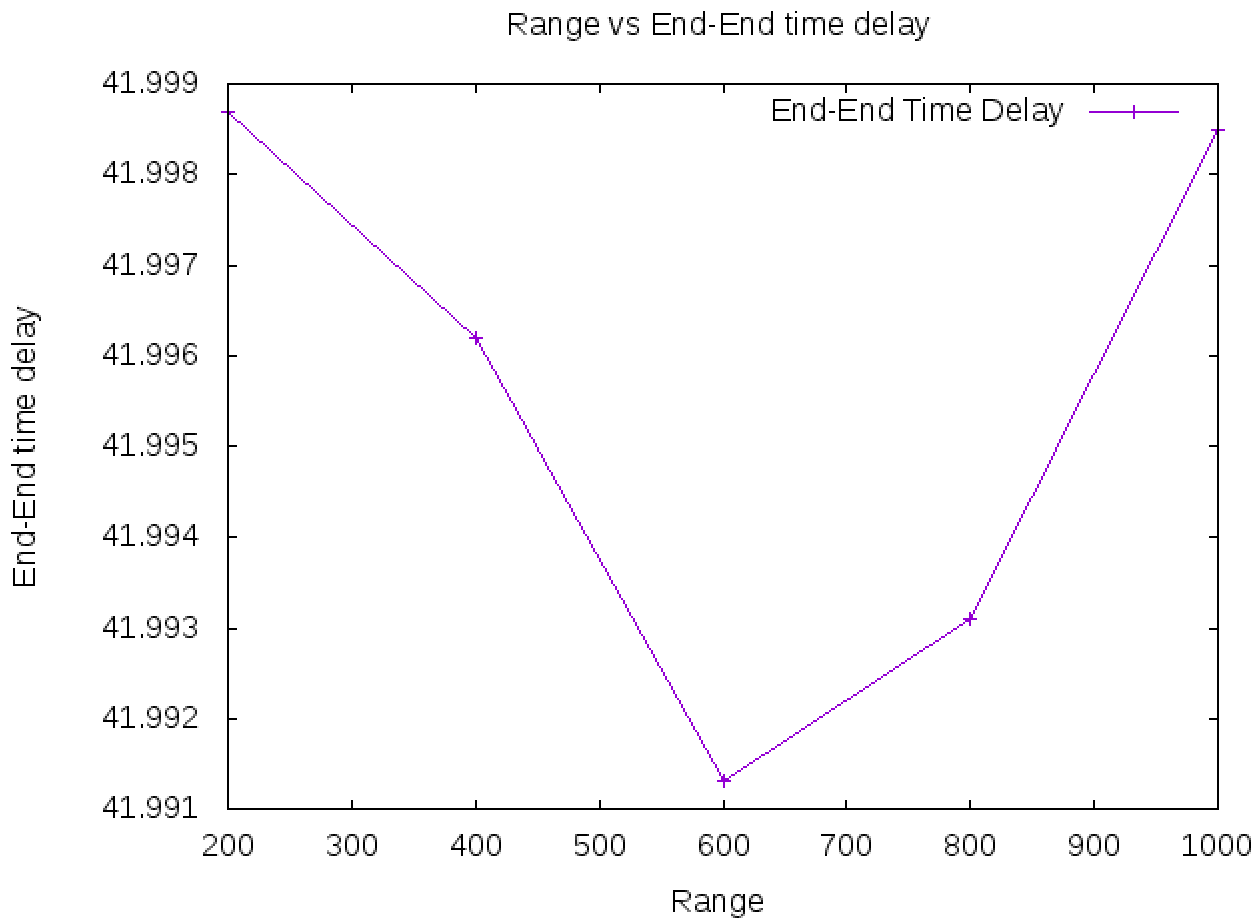
Transmission range 200,400,600,800,1000



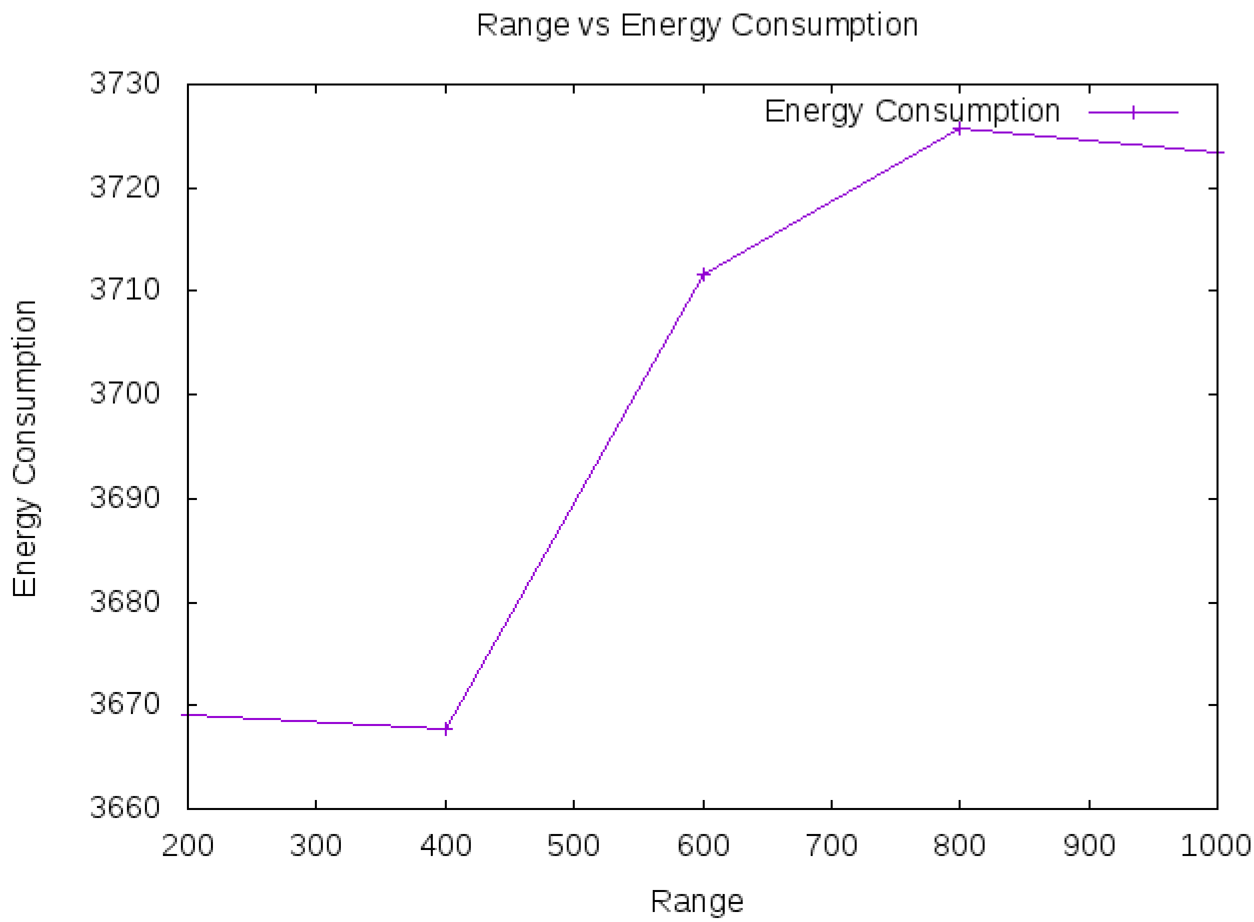
Transmission range vs Packet send/drop ratio



Transmission Range vs End to end delay



Tx Range vs Energy Consumption



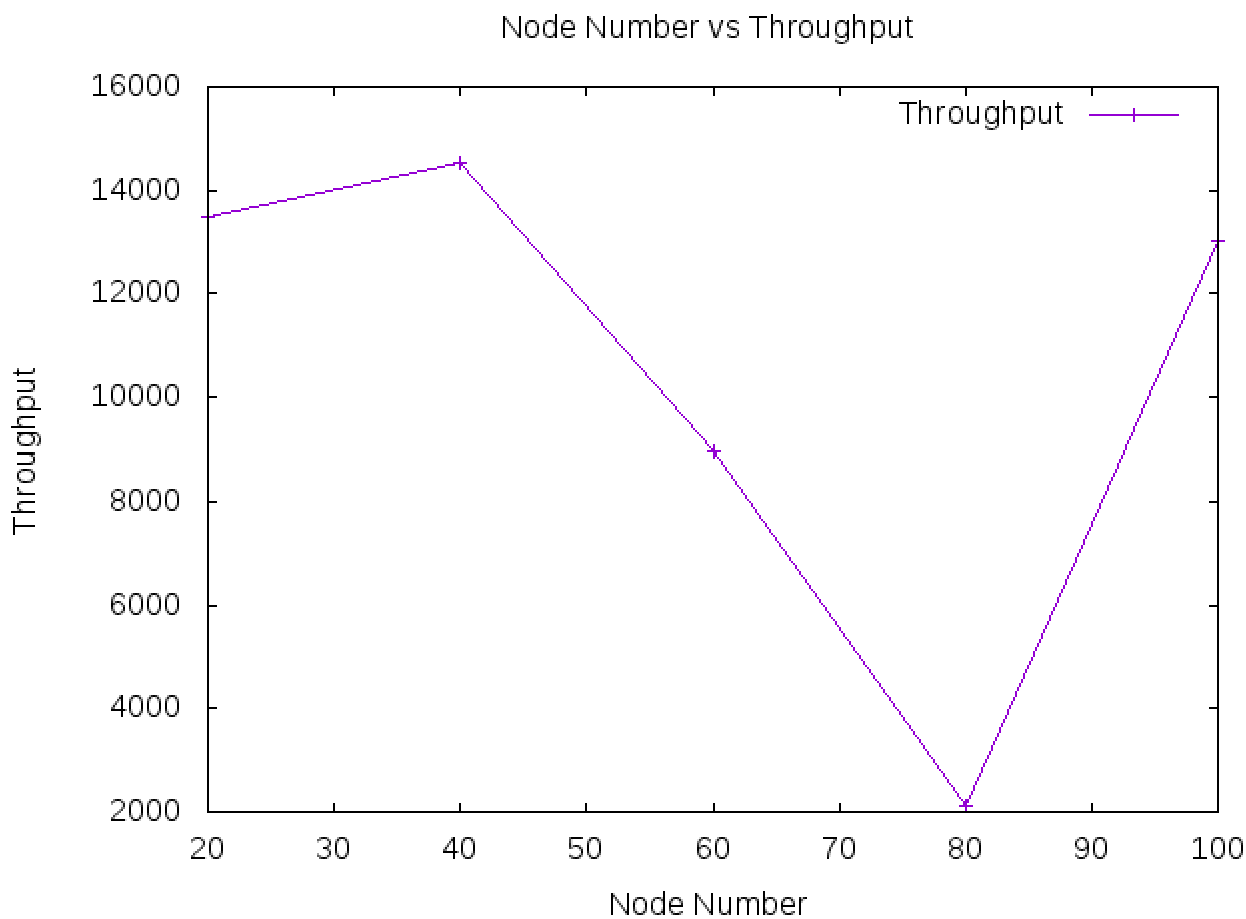
802.15.4

Varying node

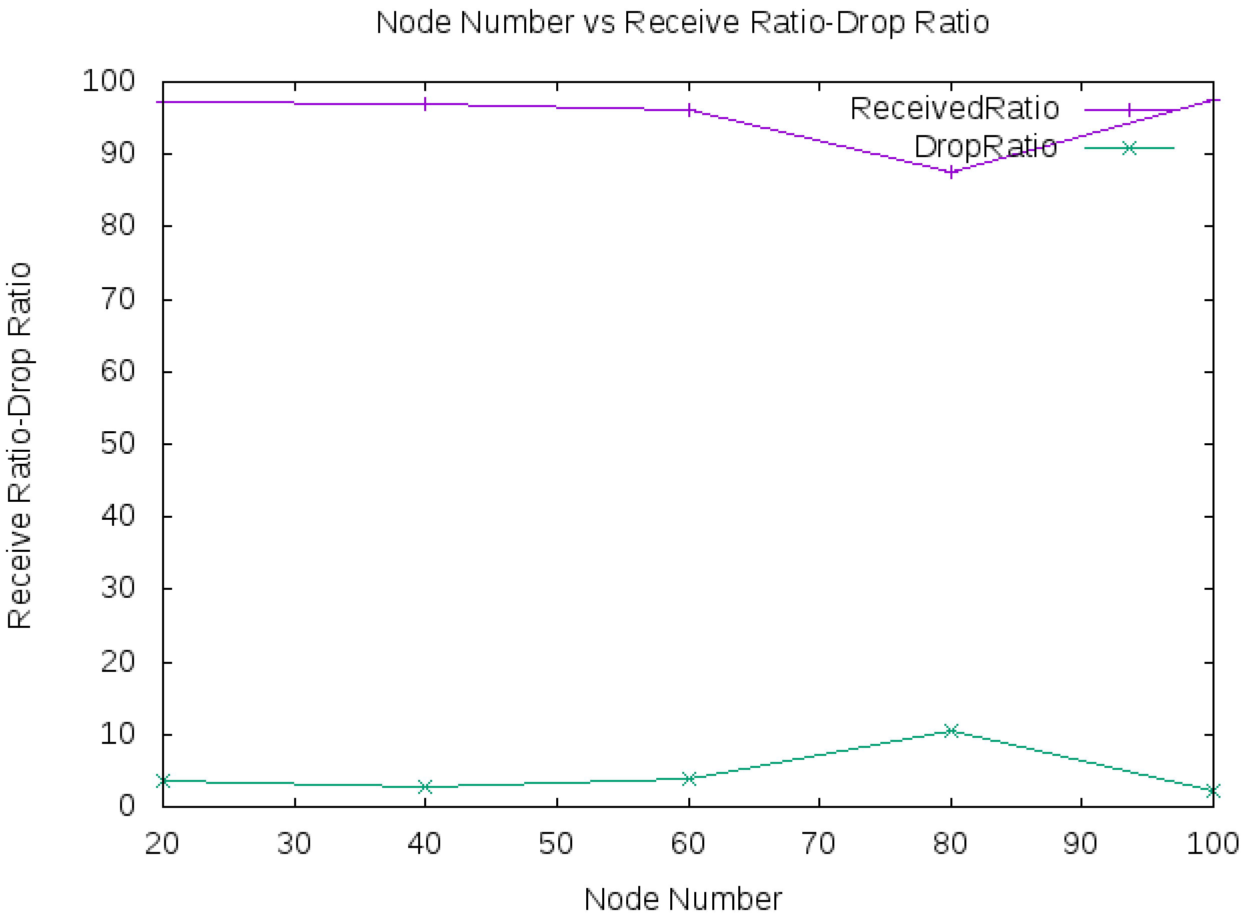
Flow 20, Packets per second 100 and Range 400

Nodes 20,40,60,80,100

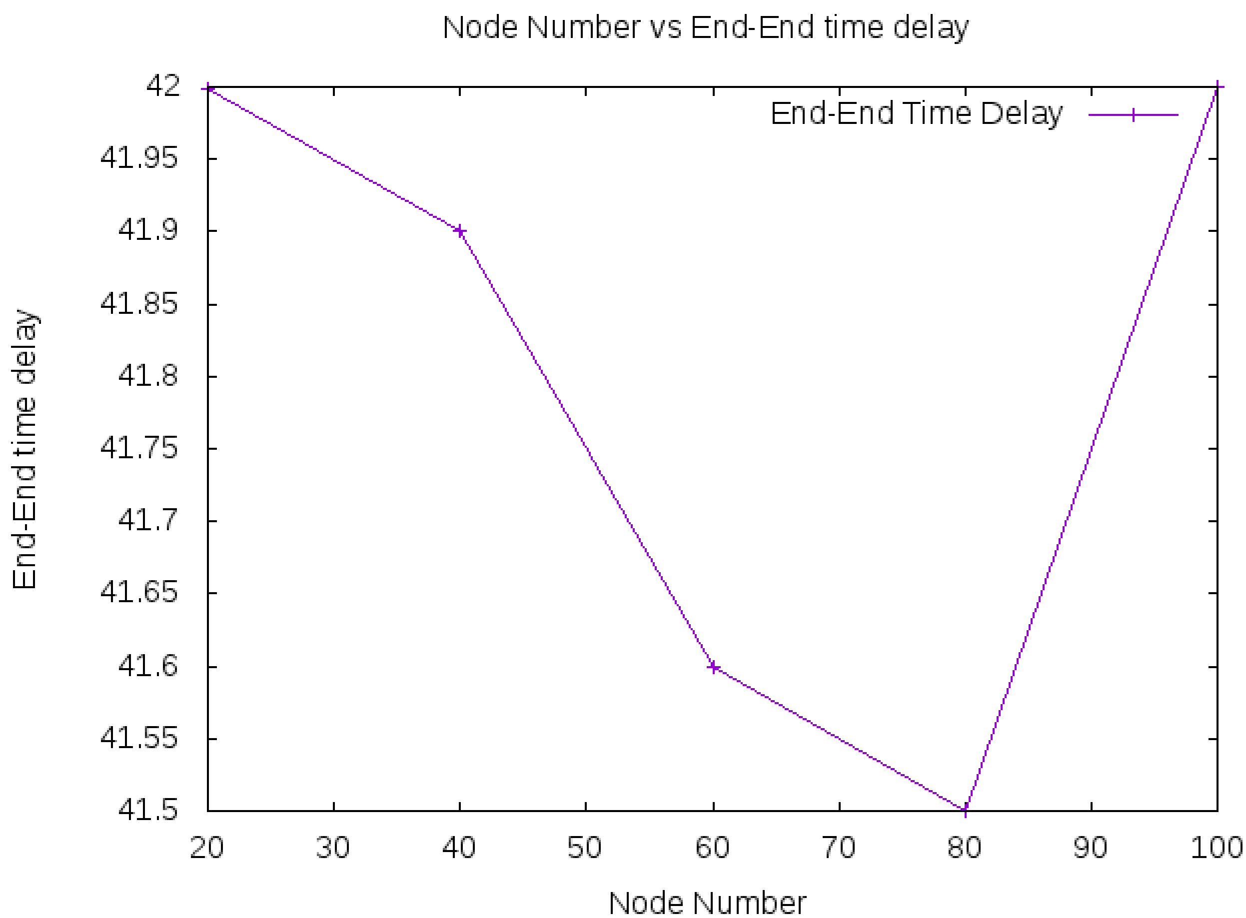
Node vs Throughput



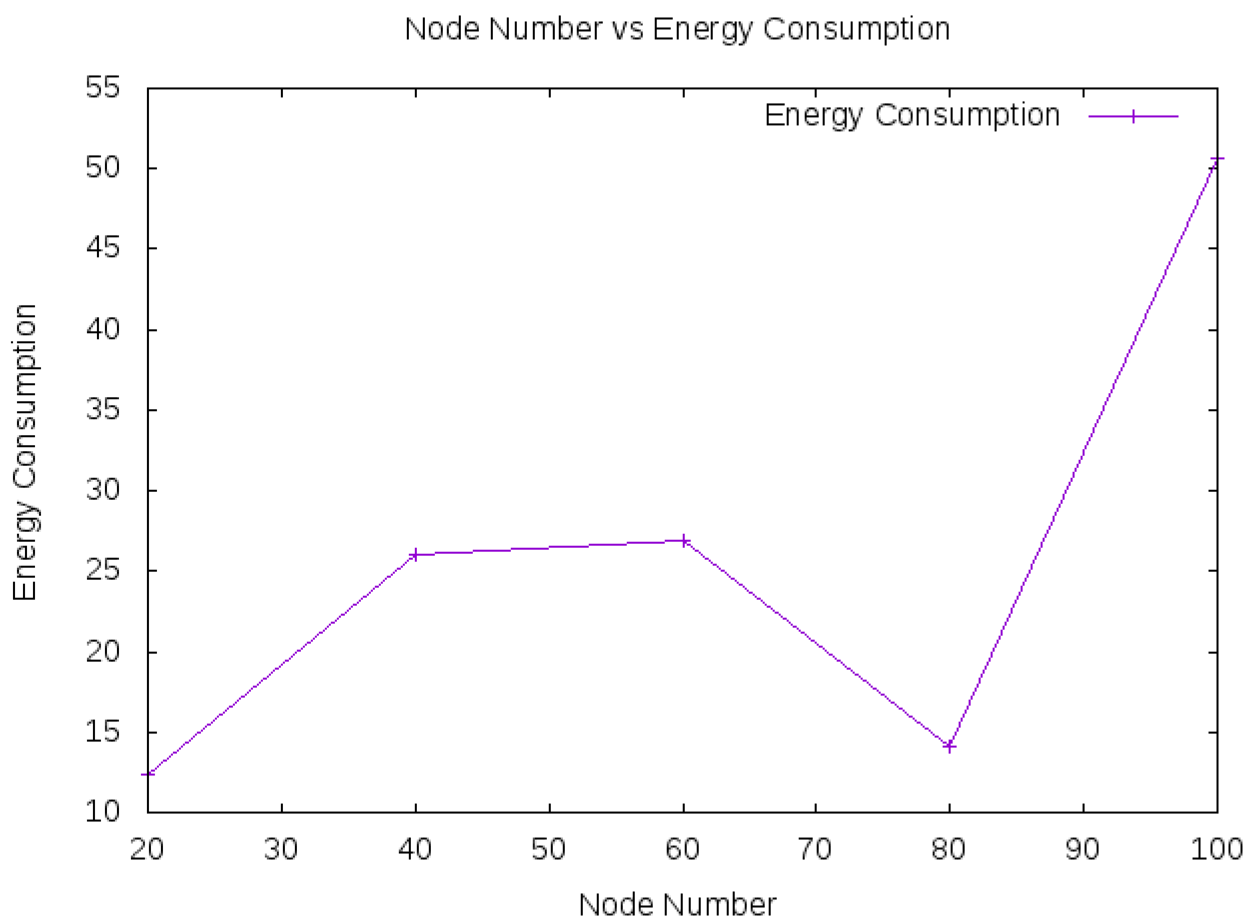
Node vs Packet Send/Drop Ratio



Node vs End to End Delay



Node vs Energy

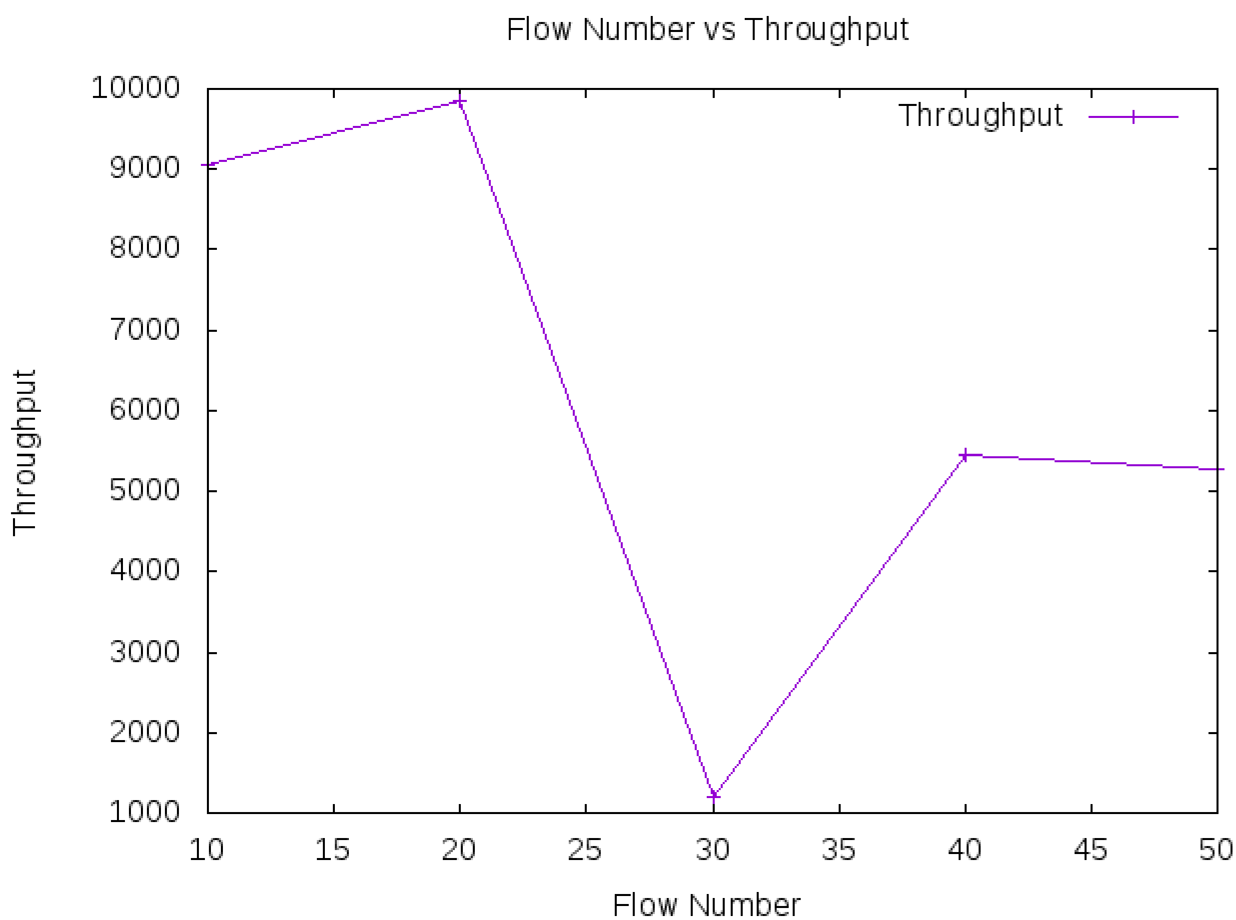


Varying Flow

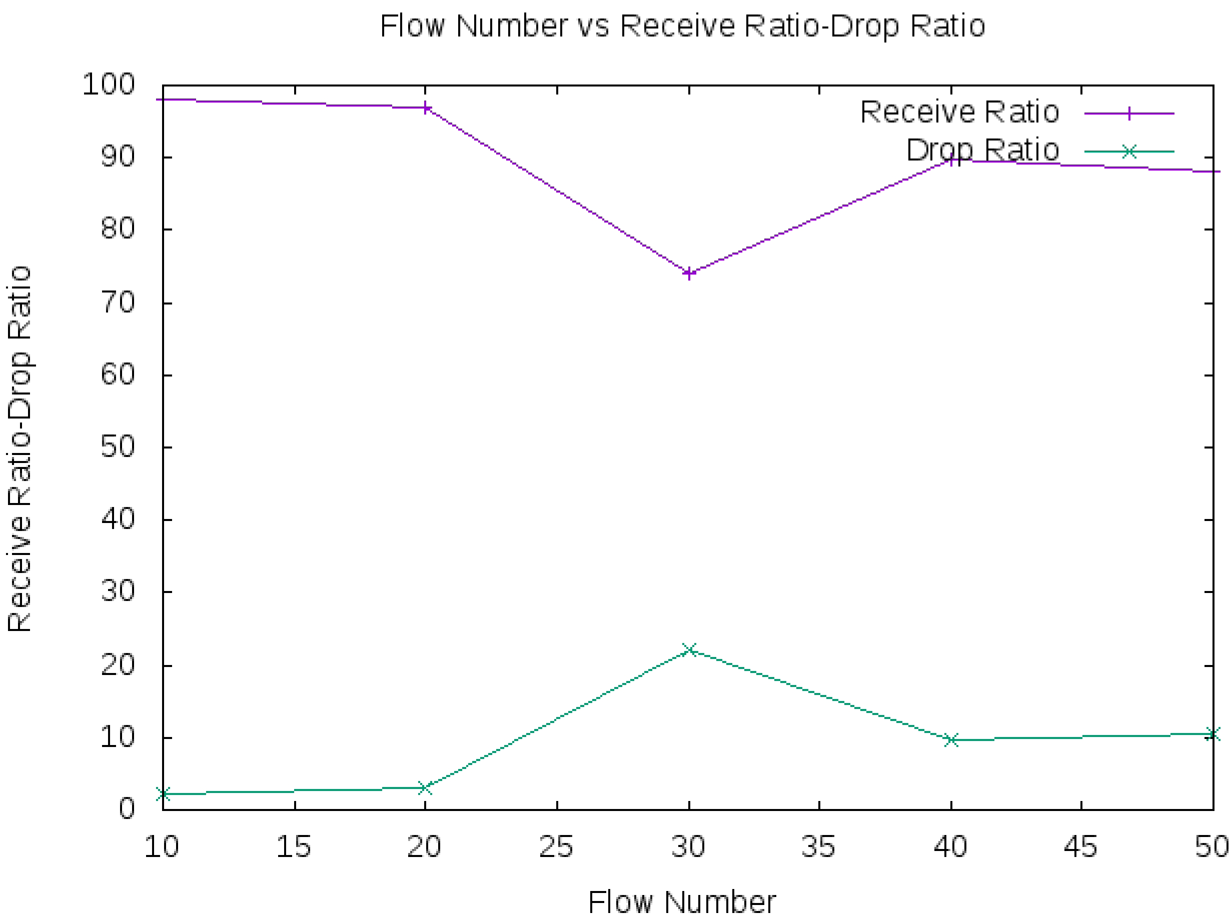
Node 100, Packets per second 100, Range 450

Flow 10,20,30,40,50

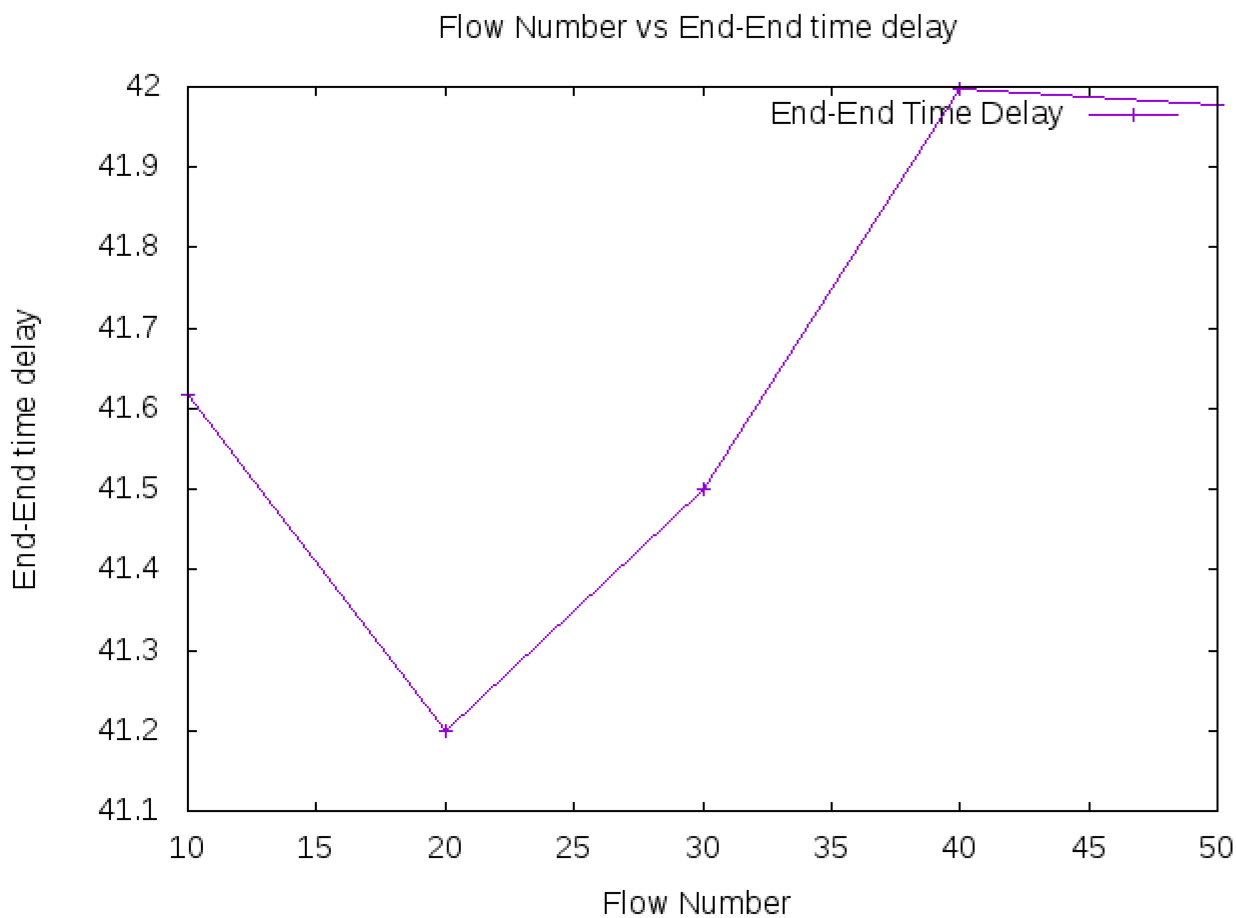
Flow vs Throughput



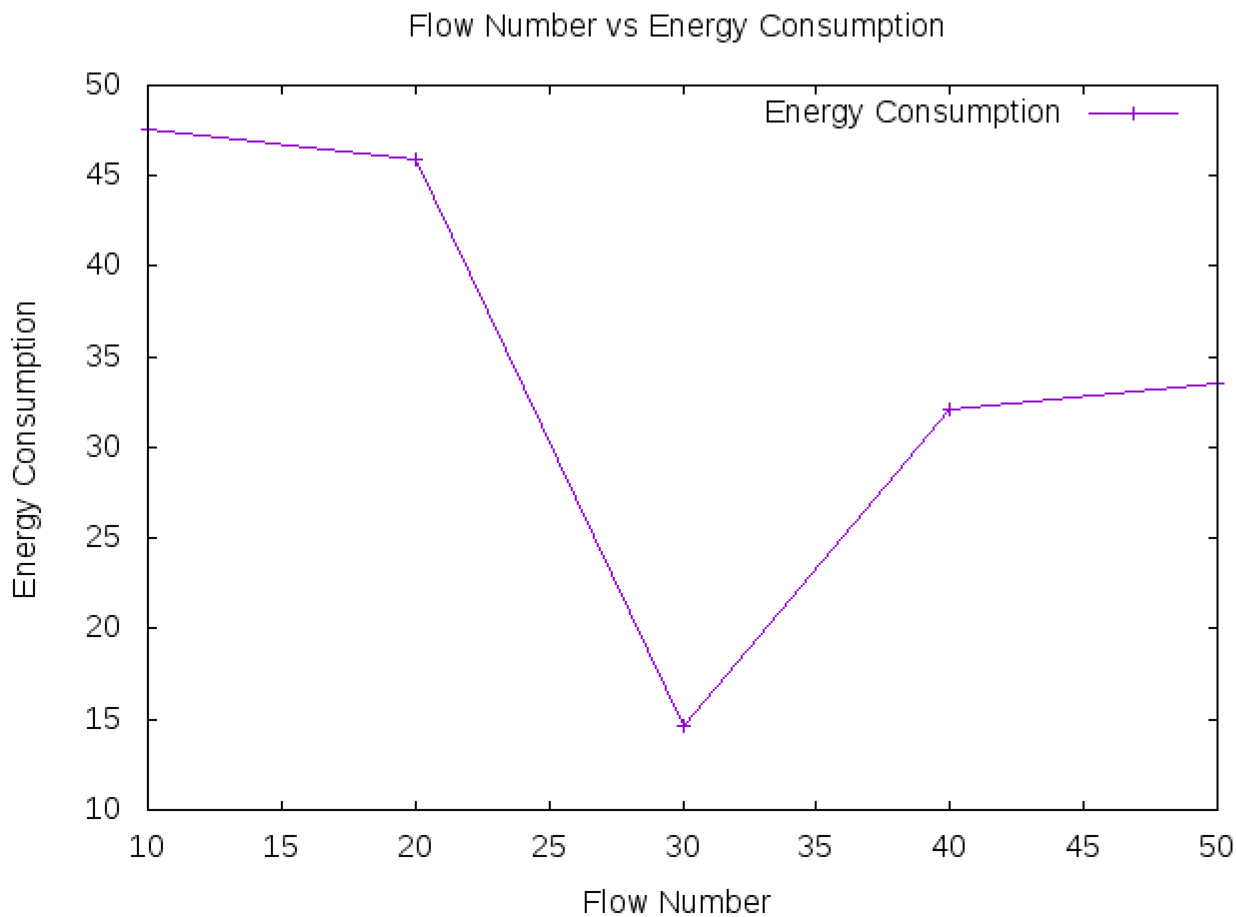
Flow vs Packet Send/Drop Ratio



Flow vs End to End Delay



Flow vs Energy Consumption

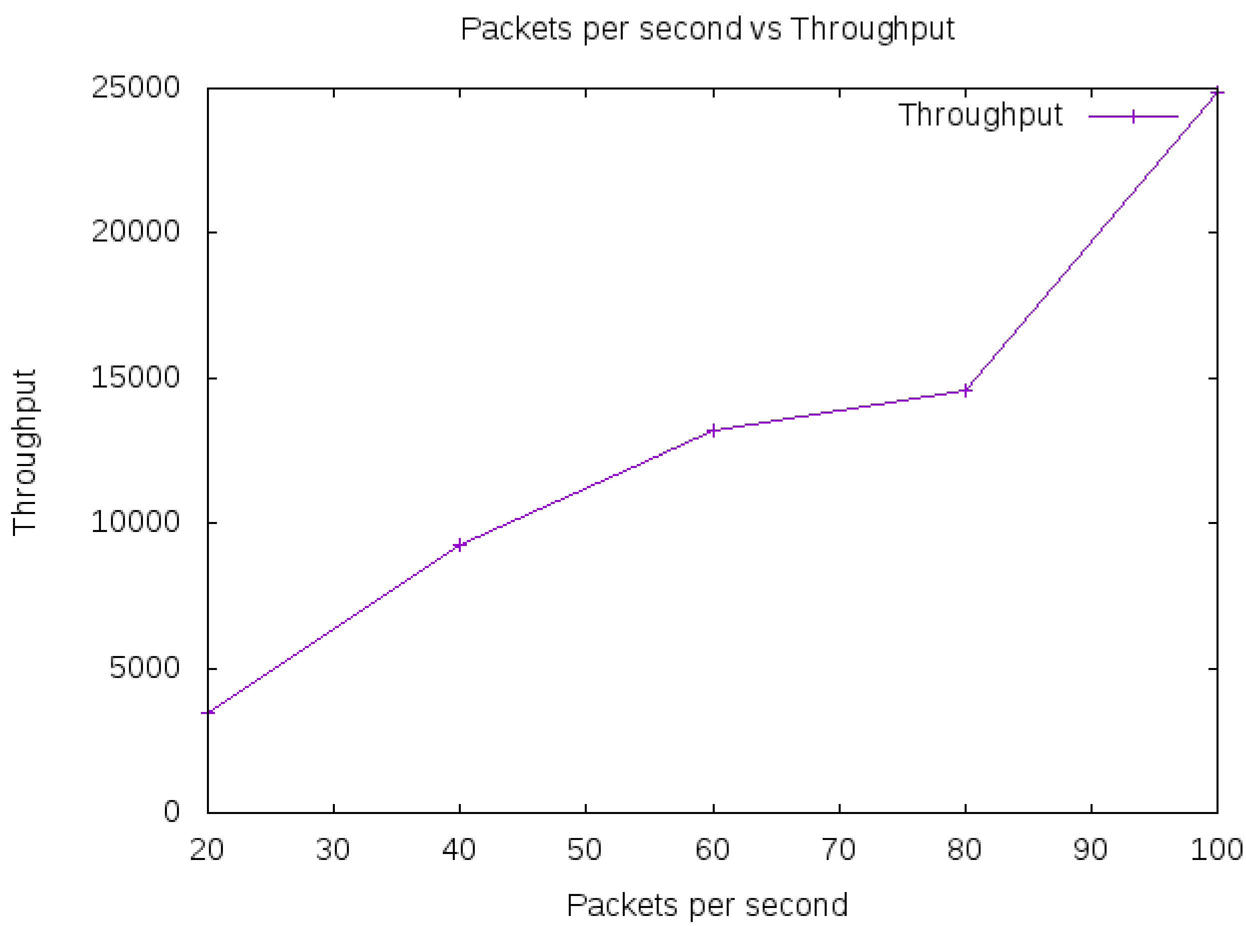


Varying packets per second

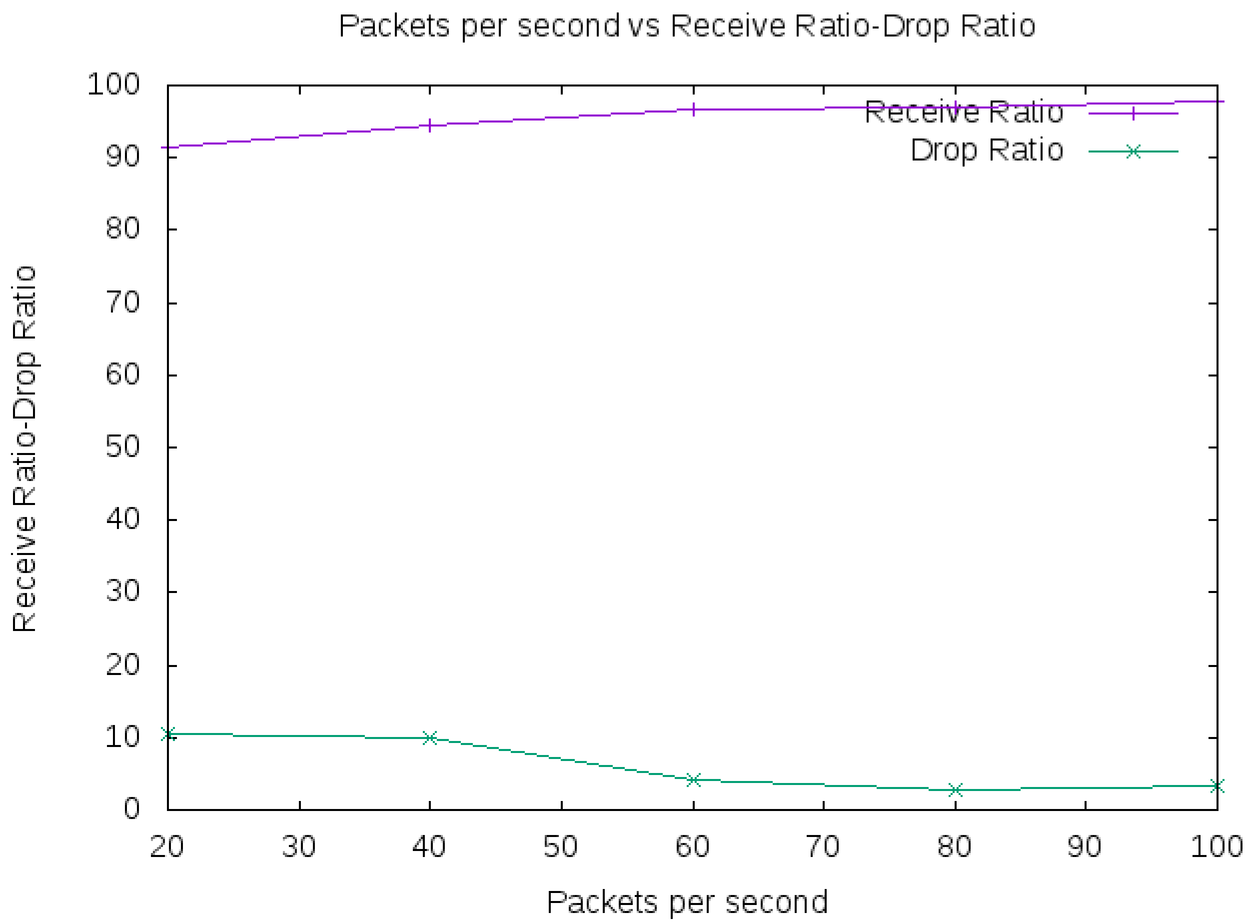
Node 60, Flow 20, Range 450

Packets per second 20,40,60,80,100

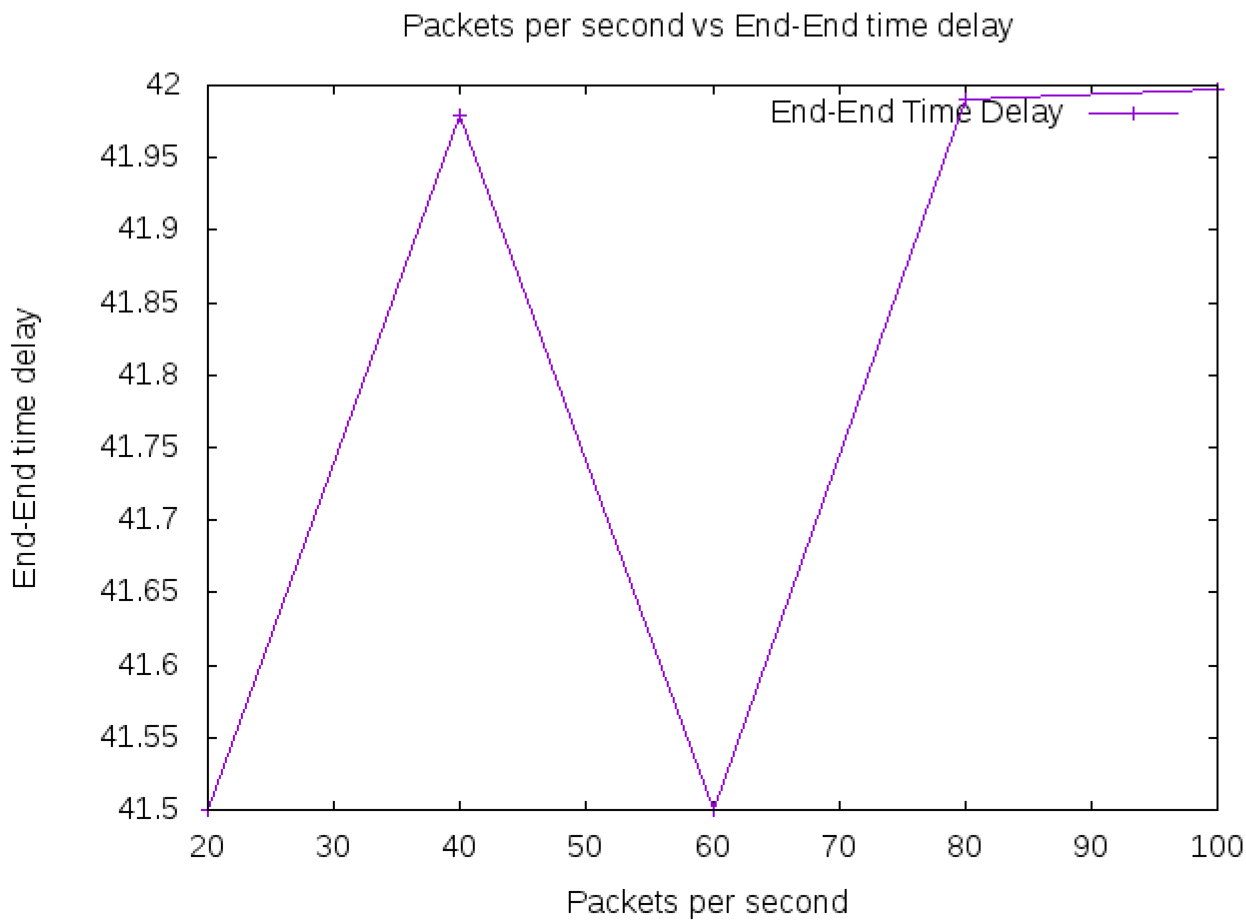
Packets per second vs Throughput



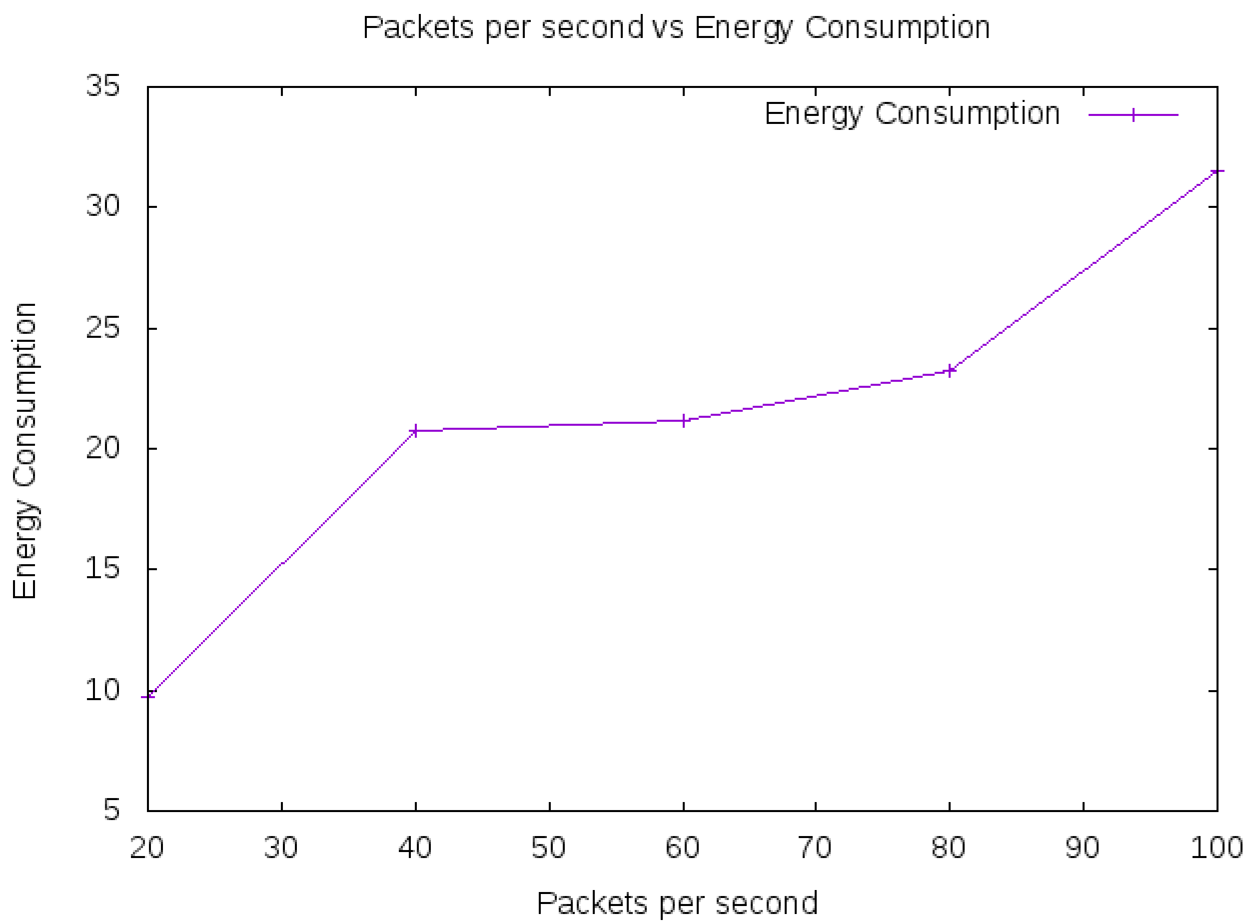
Packets per second vs Send/Drop Ratio



Packets per second vs End to End delay



Packets per second vs Energy Consumption

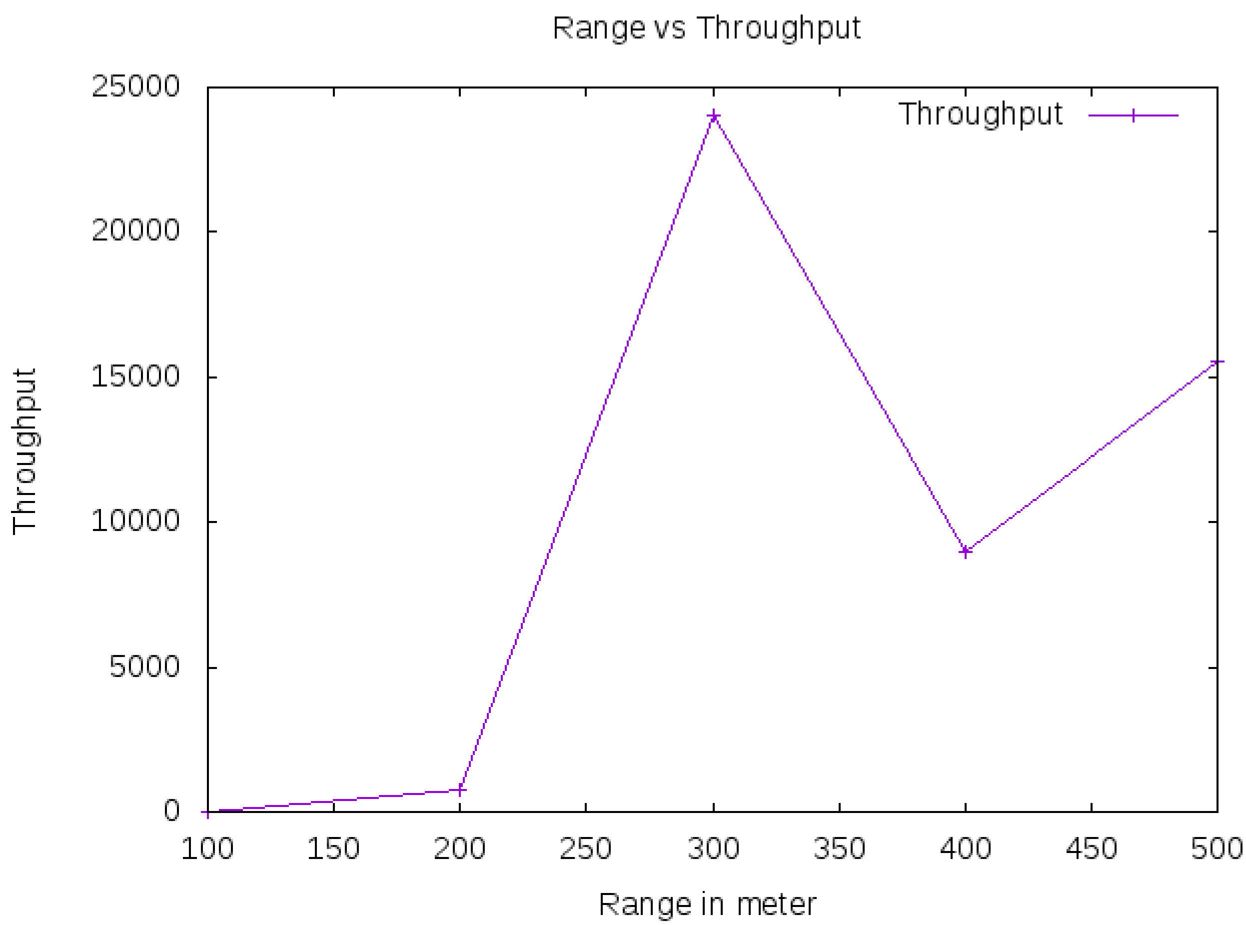


Varying Transmission range

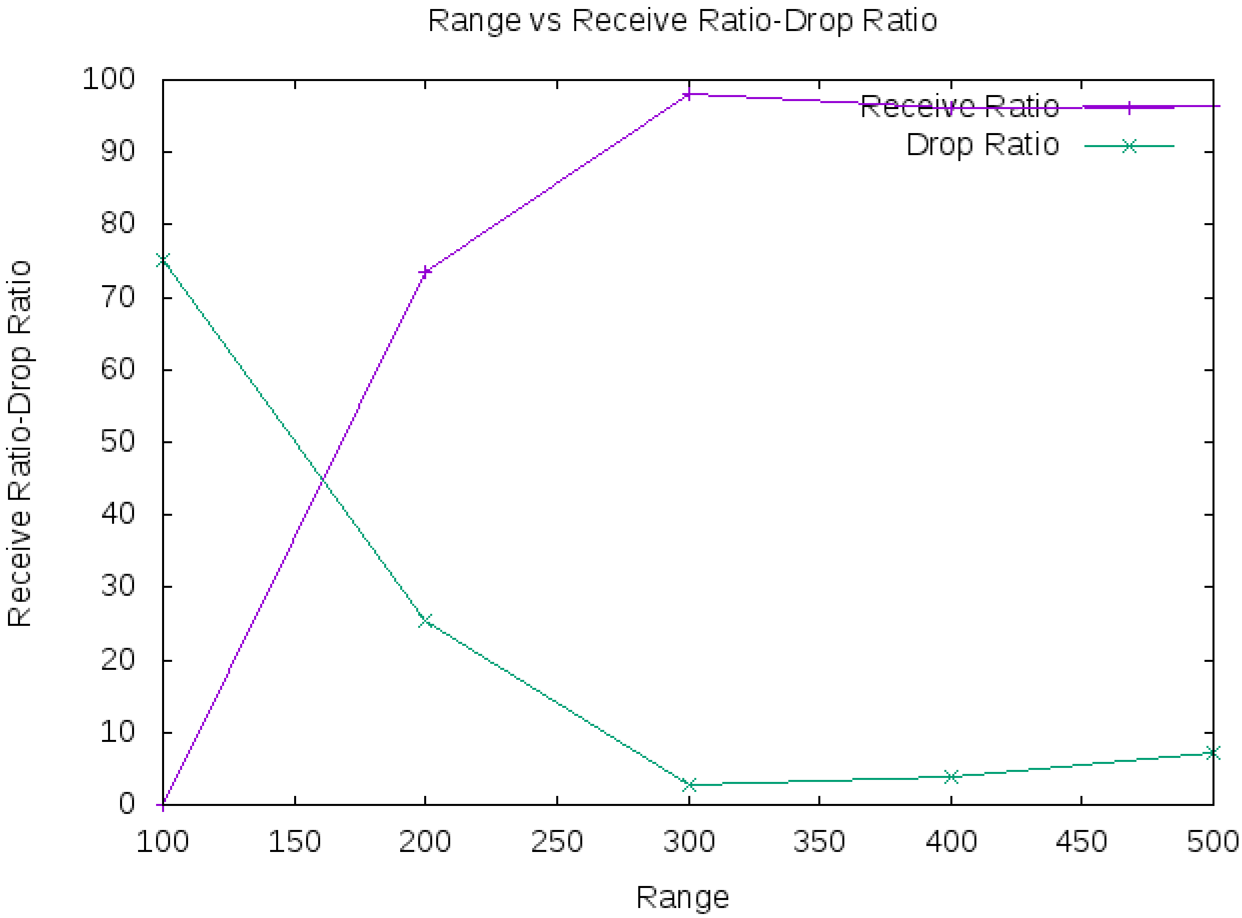
Node 60, Flow 20, Packets per second 100

TxRange 100,200,300,400,500

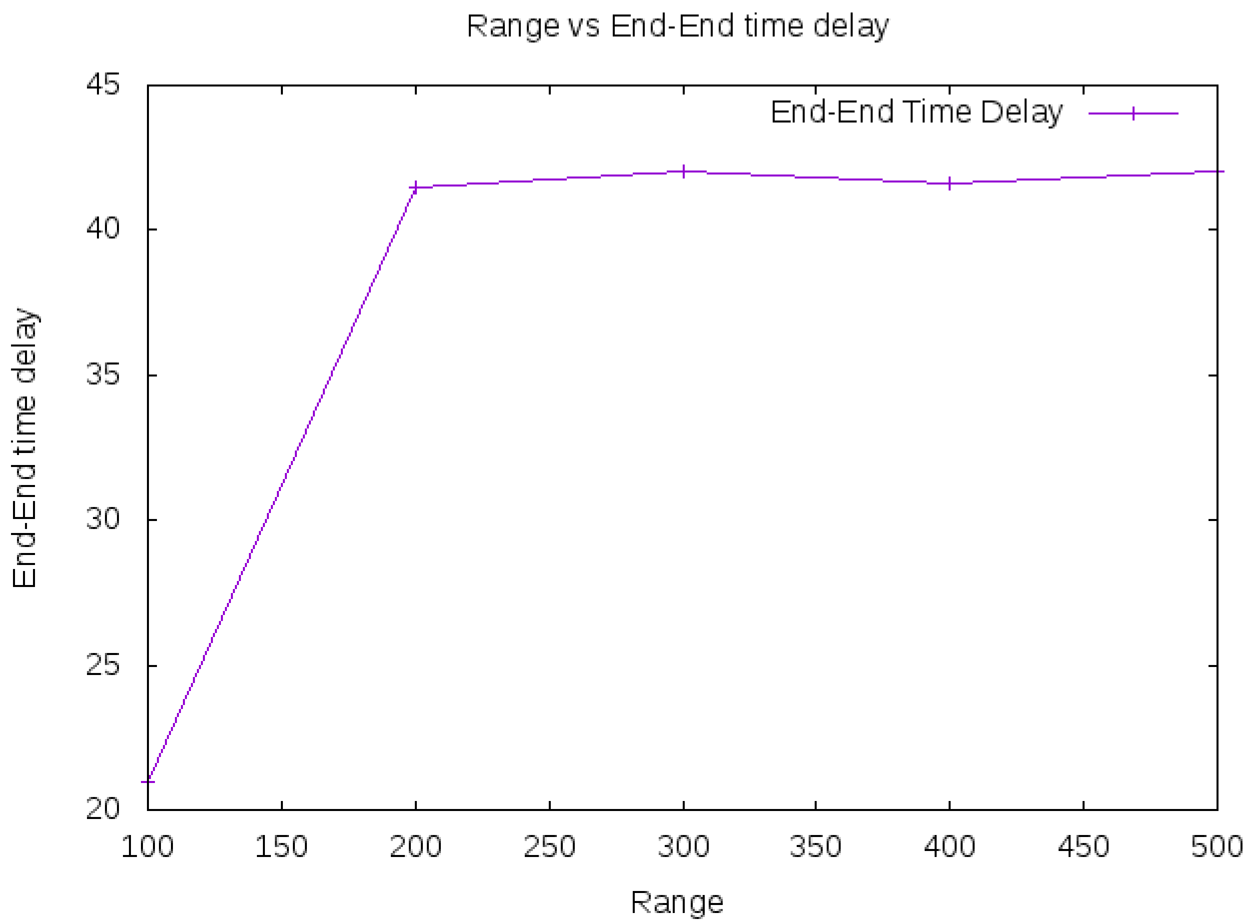
TxRange vs Throughput



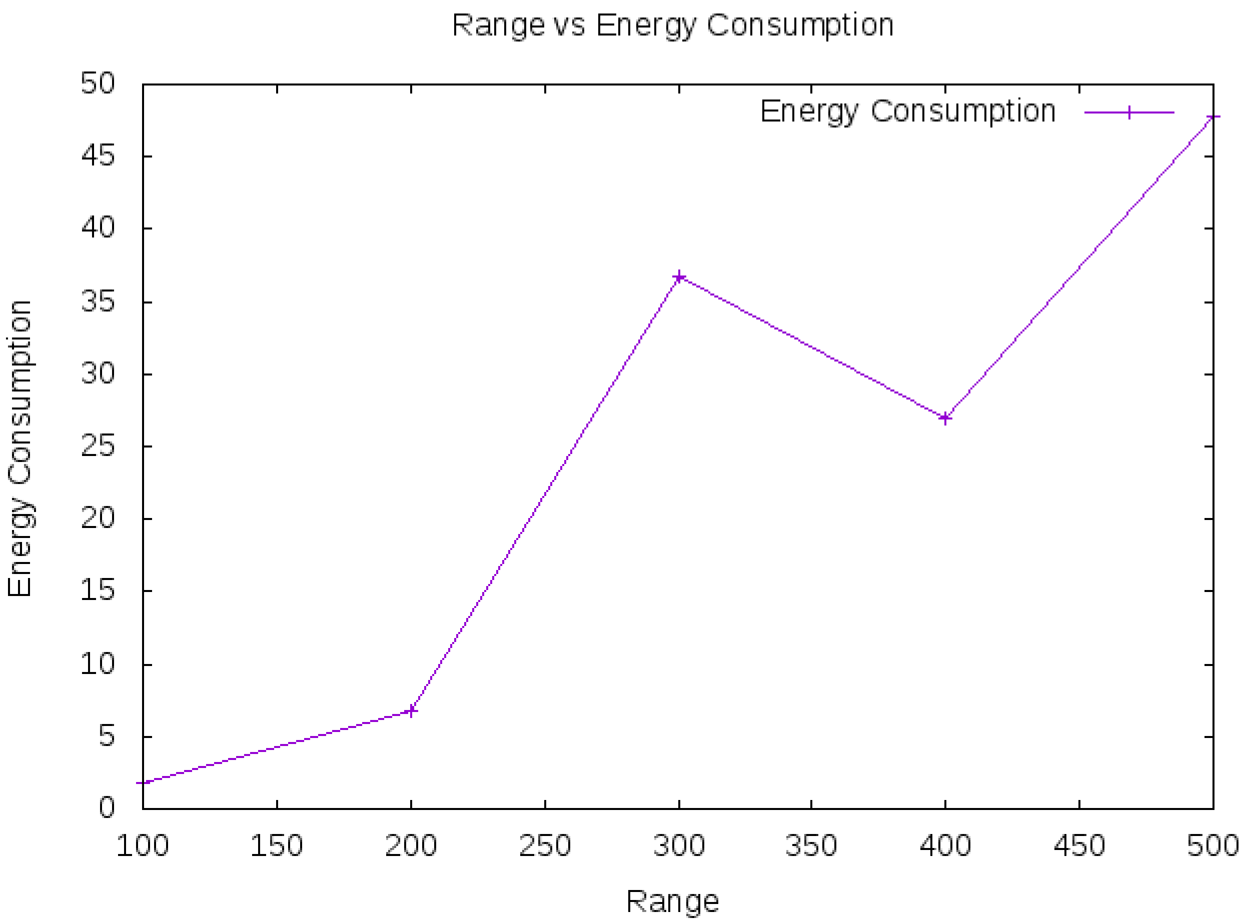
TxRange vs Packet Send/Drop Ratio



TxRange vs End to End Delay



TxRange vs Energy Consumption



Modifications:
Congestion Window Update Modification:

In proposed mechanism, we used classic exponential increment in Slow-
Start phase. When congestion window less than Slow-Start threshold (ssthresh),

the window size increases by one, as explained in following equation :
if (cwnd < ssthresh) then cwnd = cwnd + 1;
The sender TCP updates its congestion window size in the congestion avoidance
phase according to the following equation when it receives an ACK packet from
receiver TCP:
cwnd = cwnd + (f / cwnd);
In proposed mechanism, we need to update f for the equation every time that TCP
sender receives a new ACK packet. The proposed mechanism can accommodate
the fluctuation of RTTs of the network path. In fact we depend on four main
parameters to updates f values every RTT. These parameters illustrated below:

- ssthresh: slow-start threshold of network path.
- cwnd: the last value of cwnd.
- wnd_const: packets per RTT.
- k_parameter: k parameter in binomial controls.

One of big challenges to implementing the proposed mechanism is huge
increment in f value. This state happened when the throughput of the network
link is far less the next or expected throughput. This problem can cause lost in
packets and flit in performance because of the extra periods in retransmission.
When the proposed formula in the following equation has obtained a high throughput, we
need to minimize the value of f to avoiding packets losses. The control of this
problem by divide the last value of f by the previous cwnd value multiplied by the
exponent cwnd to the k parameter in binominal control as shown below:

```
f = f / (cwnd * pow (cwnd, k_parameter));
```

RTO Calculation Modification:
The situation of RTT rapid growth:
Assume a burst in network, tend to congestion. During a TCP connection, the round-trip time
RTT will increase rapidly. The most current RTT accounted for only 12.5% of the weight, while
the historical SRTT has accounted for 87.5% of the weight, this shows that, the rapid change of
RTT are not fully taken into account, leading to SRTT response lag.
The situation of RTT rapid reduction
Consider another situation, assume that a TCP connection is just established, the network is busy,
the initial RTT of the measured relatively large. However, because of its sudden application of
Internet, congestion may soon become better; making the follow-up of RTT rapidly becomes
smaller. In this case, the latest RTT accounted for only 12.5% of the weight, while the historical
SRTT has accounted for 87.5% of the weight. Therefore, the rapid change of RTT has not been
fully considered, leading to SRTT response lag.
According to the dynamic change trend of RTT
to adjust weight α, β. Whether the change trend of RTT is a rapid increase or decrease quickly,
increase the weight of RTT, which is to increase α, making the RTT’s changes are reflected in

RTO rapidly. In the Jacobson algorithm, the change of RTT is mainly reflected by the variable RTTVAR, the variable in the downward phase of RTT leads to RTO significantly lagged behind RTT. Thus reducing β, it can reduce the influence of the variables on RTO. The objective of adjusting the weight of α, β is to make the RTO more quickly follow the RTT reduction, can be faster send data to the network, making full use of network bandwidth, improving the efficiency of transmission. A variable k --- RTT's rate of change is required by the following formula.

$k(n+1) = |RTT(n+1) – RTT(n)| / RTT(n)$;

According to the results calculated by the formula, if $k>1$, then $k=1$.
Calculated the values of α, β, follow the following formula.
$\alpha(n+1) = \alpha 0 (1+ k(n+1))$;
$\beta(n+1) = \beta 0 (1+ k(n+1))$;

The above $\alpha 0$, $\beta 0$ according to Jacobson, the recommended values is $\alpha=1/8$ and $\beta=1/4$.

$SRTT(n+1) = (1- \alpha(n+1)) * SRTT(n) + \alpha(n+1) * RTT(n+1)$;
$RTTVAR(n+1) = (1 - \beta(n+1)) * RTTVAR(n) + \beta(n+1) * |SRTT(n+1) – RTT(n+1)|$;
$RTO(n+1) = SRTT(n+1) + 4 * RTT(n+1)$;