CSE-322

(Computer Networks Sessional)

Report on NS-2 Project

Submitted by:

Tanjeem Azwad Zaman

Roll: 1805006

Dept. of CSE, BUET

Date of Submission:

27.02.2023

1. Introduction:

In this project, 2 different topologies were primarily simulated, namely a wired topology and a Wireless (802.11 - static) Topology [since 1805006 % 8 = 6], using the ns2 simulator.

Modifications made to the ns2 source code were Congestion-Control based and were inspired by this paper. Data was collected for the modified algorithm as well as the base TCP-Reno algorithm. And comparison graphs were plot for Throughput, End-to-end Delay, Packet Delivery Ratio, Packet Drop Ratio and Energy Consumption, while varying the parameters number of nodes, flows, coverage area and packets per second. Improvements were observed in most cases.

Furthermore, some bonus tasks were also completed, namely collecting the per-node-throughput (demonstrated for one instance), as well as simulating a satellite network and cross transmission of wired-wireless packets.

2. Network Topologies Under Simulation:

2.1 Wired Topology:

- A dumbbell topology with a connection from sources to the destinations was simulated.
- Equal number of nodes set as sources and destinations,
- flows randomly generated from sources to destinations

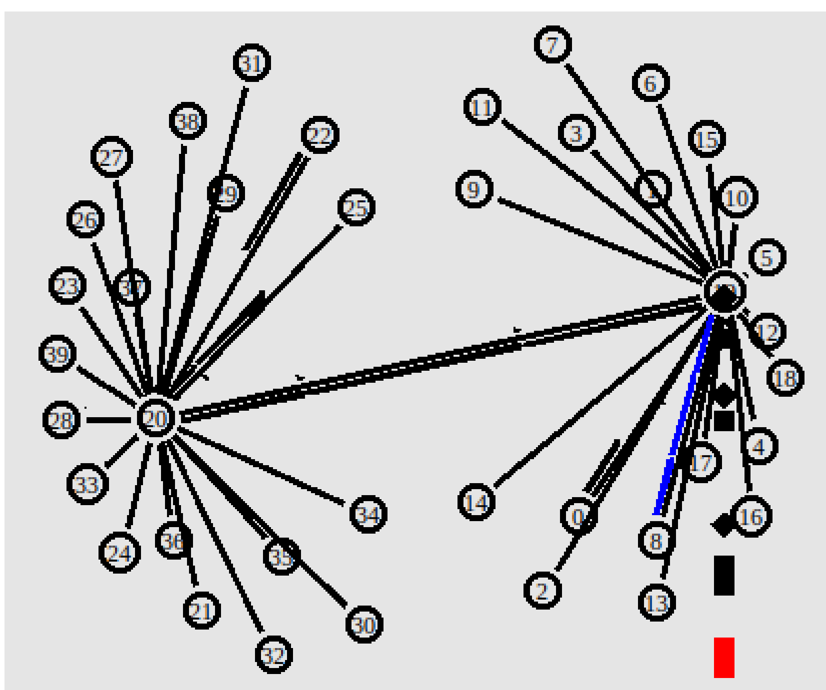


Figure 1: Wired Topology with 40 nodes

- The duplex links had 2Mb Bandwidth, 10ms delay and Droptail Queues
- FTP data transfer was simulated.
- Runtime was set to 50 seconds.

2.2 Wireless Topology (802.11- Static):

- Since the modification algorithm was mainly targeted for wireless networks, this topology reflected the most change
- A square-like grid of static nodes were placed.
- The base parameters were as follows:
    - base_nodes: 40
    - base_flows: 20
    - base_TxRange: 250m (Tranmission Range)
    - base_pckt_rate: 100
- A runtime of 150 seconds was set
- The following specifications were maintained:
    - Queue : Droptail, max size 50
    - Antenna: Omni Directional
    - Propagation Model: Two Ray Ground Propagation Model
    - Error Model: Uniform Distribution with rate = 10%
    - Routing Protocol: DSDV
    - Application: FTP
- Data was collected for the existing TCPReno algorithm and the modified algorithm

Figure 2: Wireless Topology for 40 nodes

3. Parameters under Variation:

3.1 Base Parameters:

- Base Node# = 40
- Base Flow# = 20
- Base Transmission Range (TxRange)= 250m (only for wireless)
- Base packets per second = 200

3.2 Varied parameters:

- Nodes: 20, 40, 60 ,80, 100
- Flows : 10, 20, 30, 40, 50
- TxRange = 1x, 2x, 3x, 4x, 5x of base TxRange
- Packets per second = 100, 200, 300, 400, 500

4. Modifications in the Simulator:

4.1 Basis: Adaptive TCP Transmission Adjustment for UAV Network Infrastructure

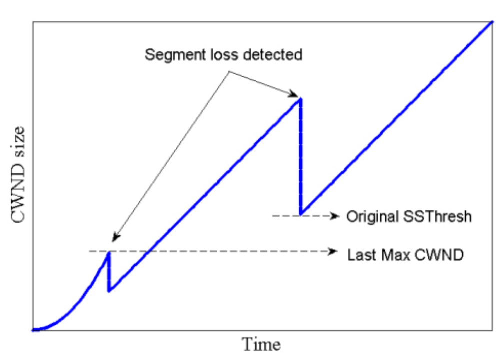
In summary, this paper implements the ASRAN (Adaptive Ssthresh Reviser for Flying Adhoc Network) to differentiate between loss due to transient link instability and congestion in flying UAV communications.

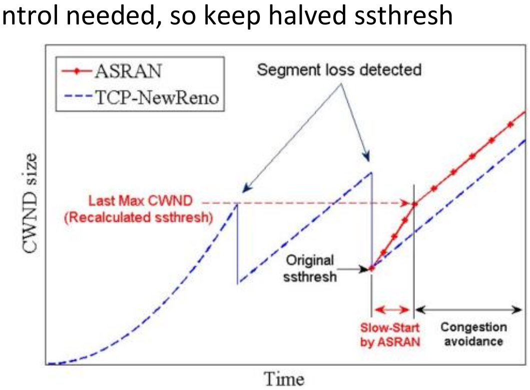
4.1.1 Motivation:

- UAVs must move in sync and transmit data over a wide area
- They suffer from a high probability of transmission failure due to interference or mobility (transient link instability)
- Thus, congestion is not the only cause of packet drops
- As such, congestion control algorithms should not reduce ssthresh as usual, since reducing slow start will lead to lower packet transmission rates for longer times.

4.1.2 Proposal:

- Keep track of previous cwnd
- If ssthresh/2 < oldcwnd, then packet loss due to transient link instability, assign ssthresh = oldcwnd
- If ssthresh/2 > oldcwnd, then congestion control needed, so keep halved ssthresh





4.2 Algorithm: Modified ssthresh calculator.

```
while segment_loss = true do
    function ssthresh_calculator(current_max_cwnd)
        original_ssthresh = current_max_cwnd/2
        recalculated_ssthresh = max(original_ssthresh, last_max_cwnd)
        last_max_cwnd = current_max_cwnd
        return recalculated_ssthresh
    end function
end while
```

4.3 Modifications:

- A new class that extends the tcp class, similar to tcpReno, named tcpTanj
- This class sets a flag ASRAN_ON, which tells the model to use the modified algorithm defined in the tcp.cc file.
- This also required us to keep track of old_cwnd in the class, which had an initial value of 0.
- Apply change in slowdown function of tcp.cc

4.4 Code Snippets:

```
Tanj_action:
    // we are now going to fast-retransmit and will trace that event
    trace_event("TANJ_FAST_RETX");
    recover_ = maxseq_;
    last_cwnd_action_ = CWND_ACTION_DUPACK;
    //ASRAN_ON indicates modification
    slowdown(CLOSE_SSTHRESH_HALF|CLOSE_CWND_HALF|ASRAN_ON);
    reset_rtx_timer(1,0);
    output(last_ack_ + 1, TCP_REASON_DUPACK);   // from top
        dupwnd_ = numdupacks_;
    return;
}
```

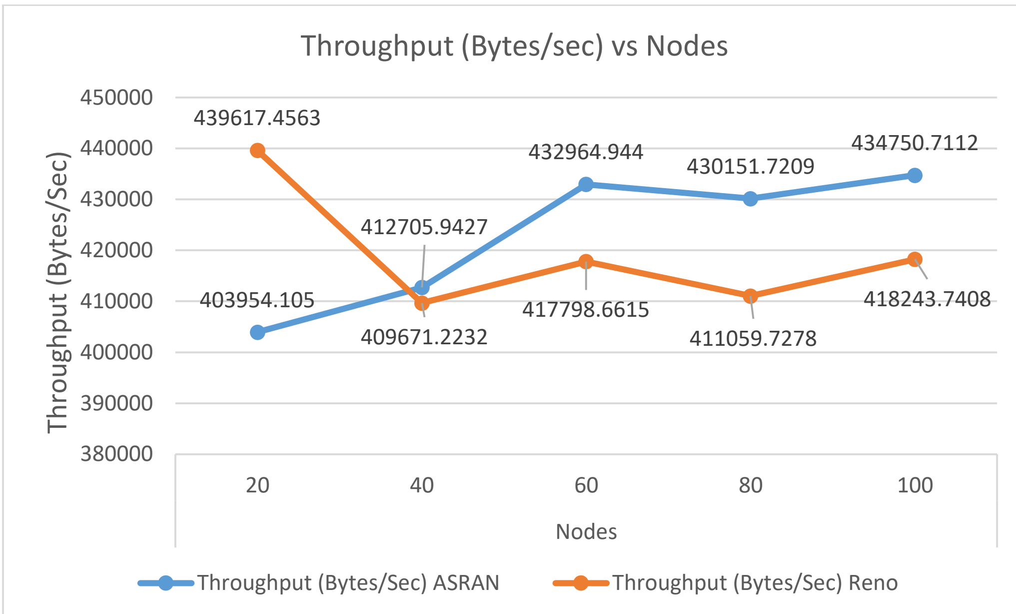
```
if (how & ASRAN_ON){

    if(old_cwnd_ > ssthresh_){
        ssthresh_ = old_cwnd_;

    }
    old_cwnd_ = cwnd_;
}
```

5 Results with Graphs:

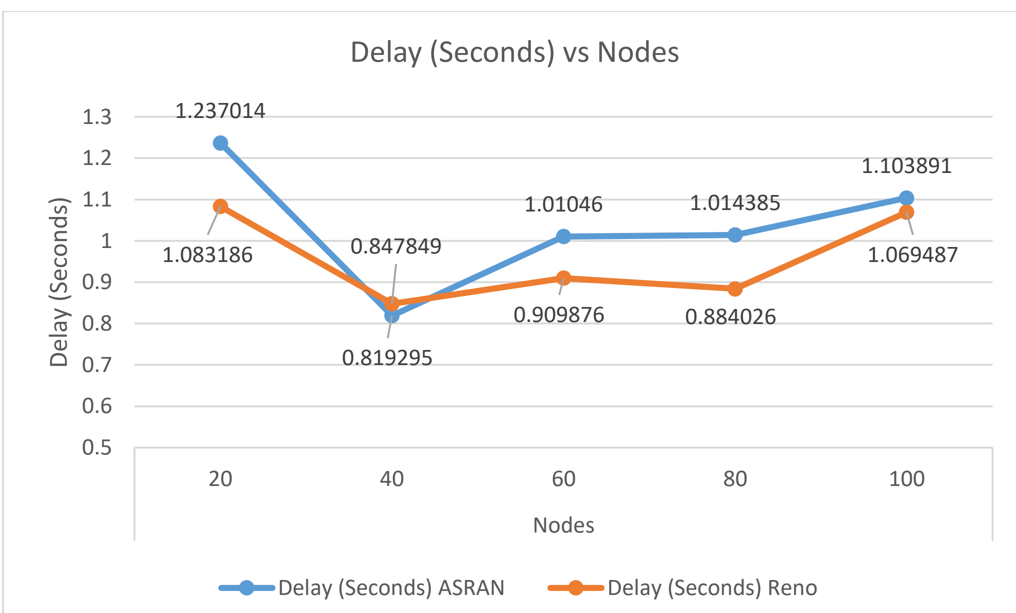
5.1 Wireless (802.11- static):

5.1.1 Varying Node#:

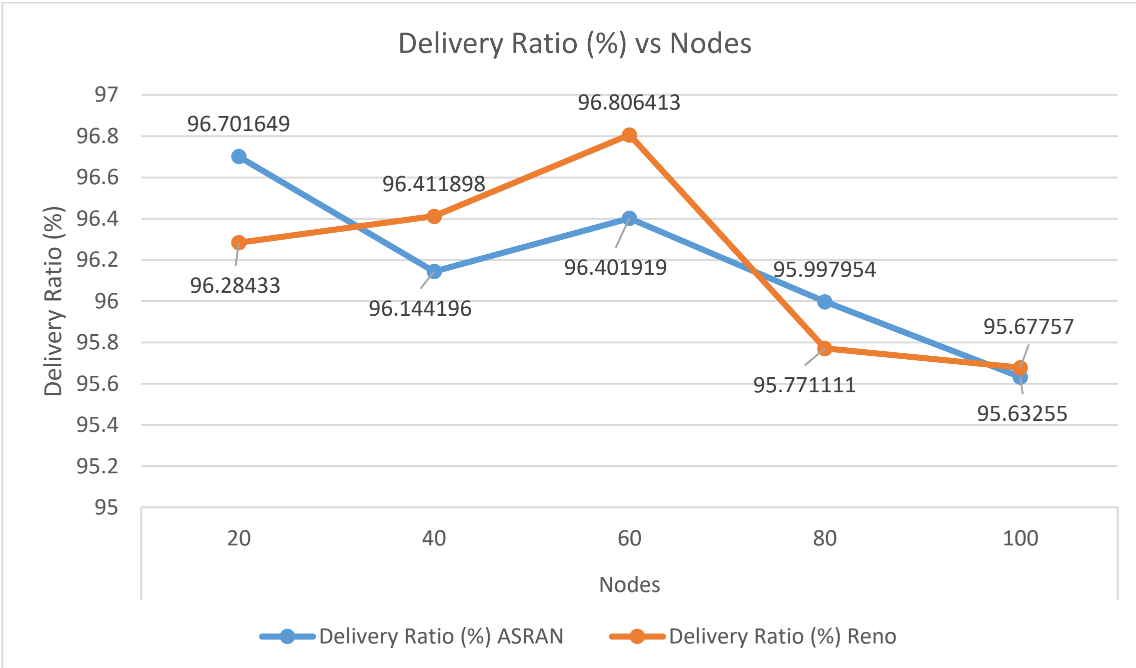
5.1.1.1 Change in Throughput:



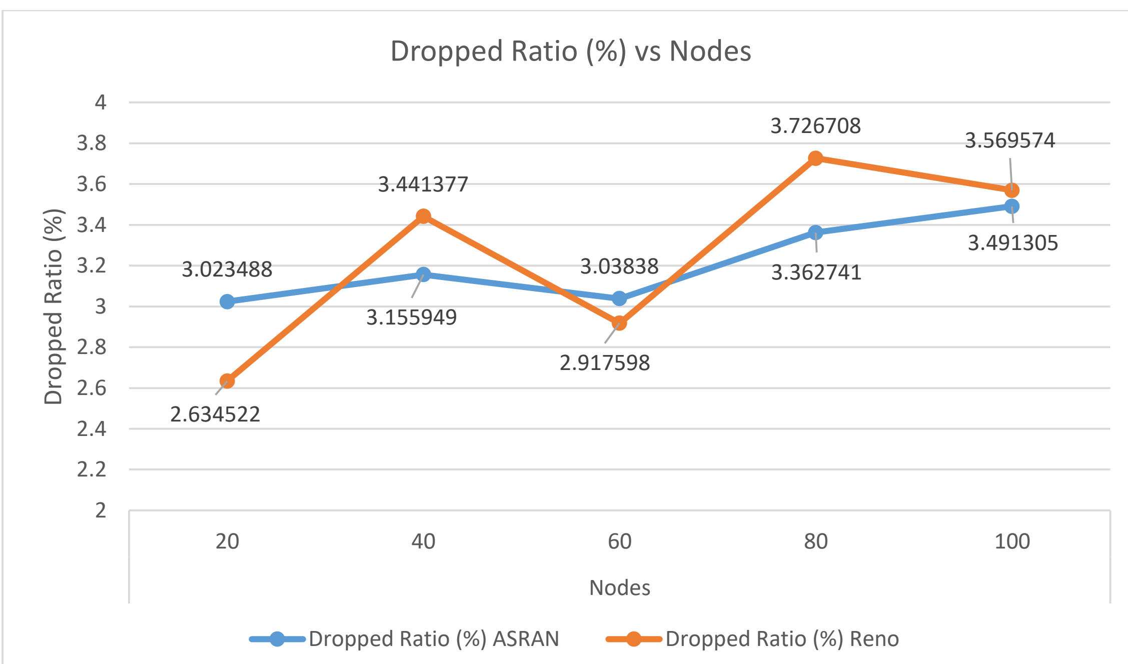
5.1.1.2 Change in End-to-end Delay:



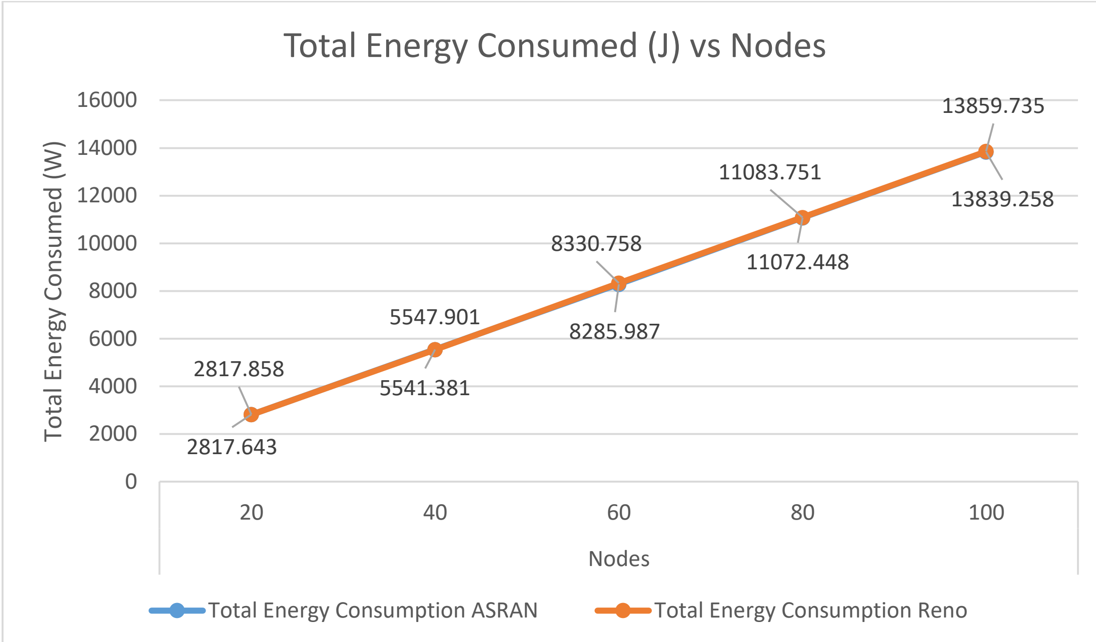
5.1.1.3 Change in Packet Delivery ratio:



5.1.1.4 Change in Packet Drop ratio:

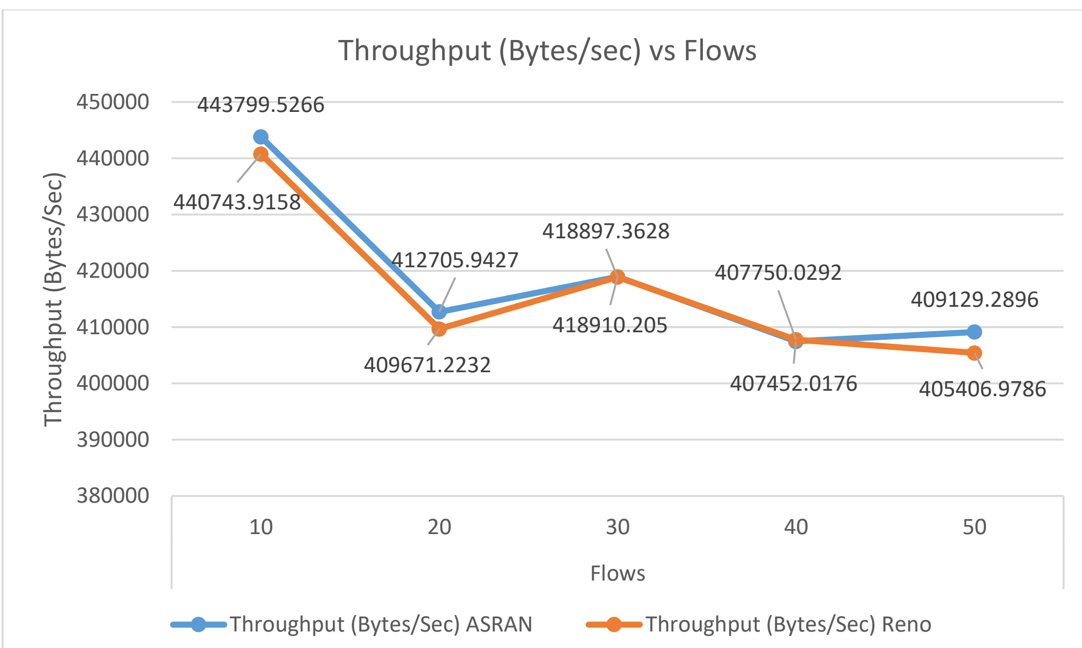


5.1.1.5 Change in Energy Consumption:

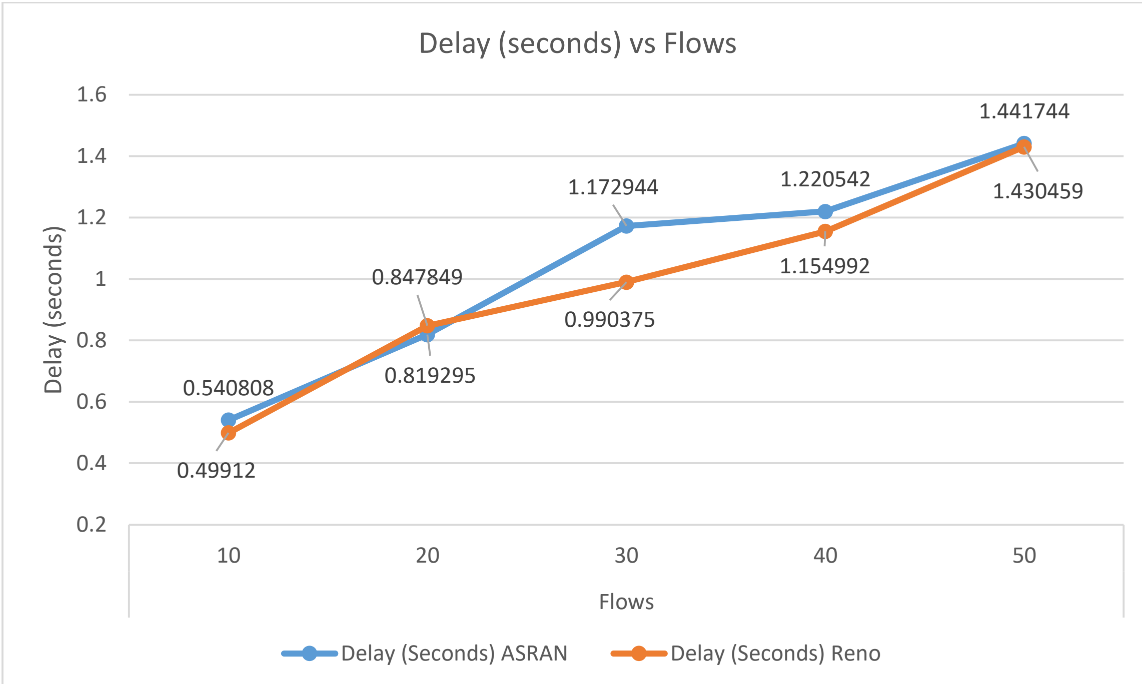


5.1.2 Varying Flow#:

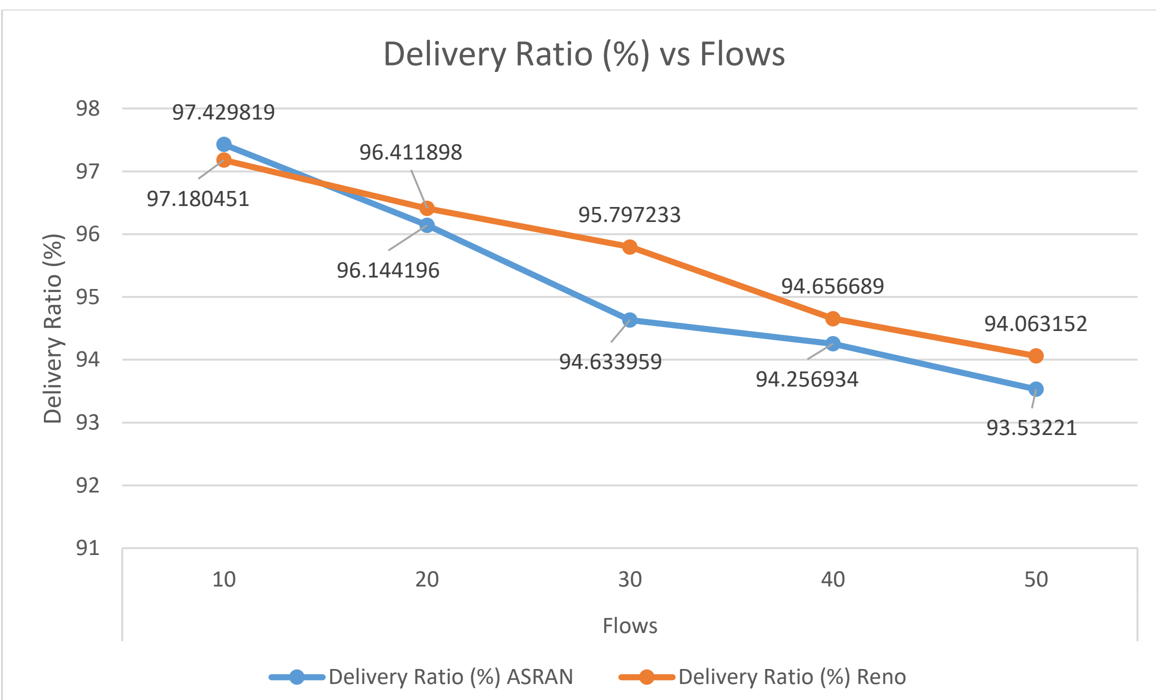
5.1.2.1 Change in Throughput:



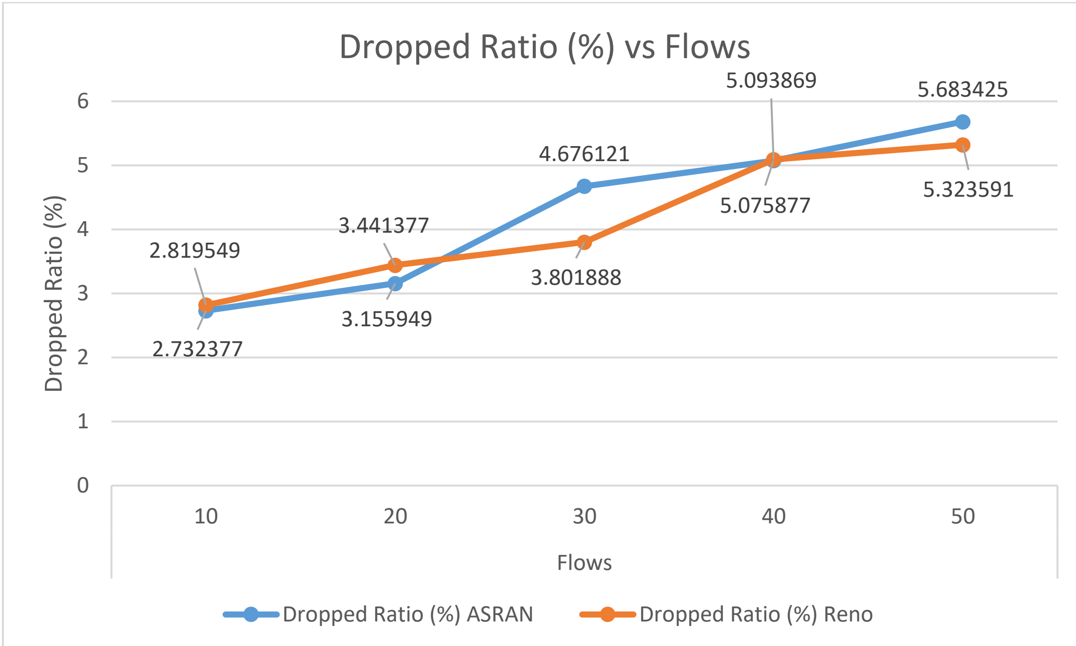
5.1.2.2 Change in End-to-end Delay:



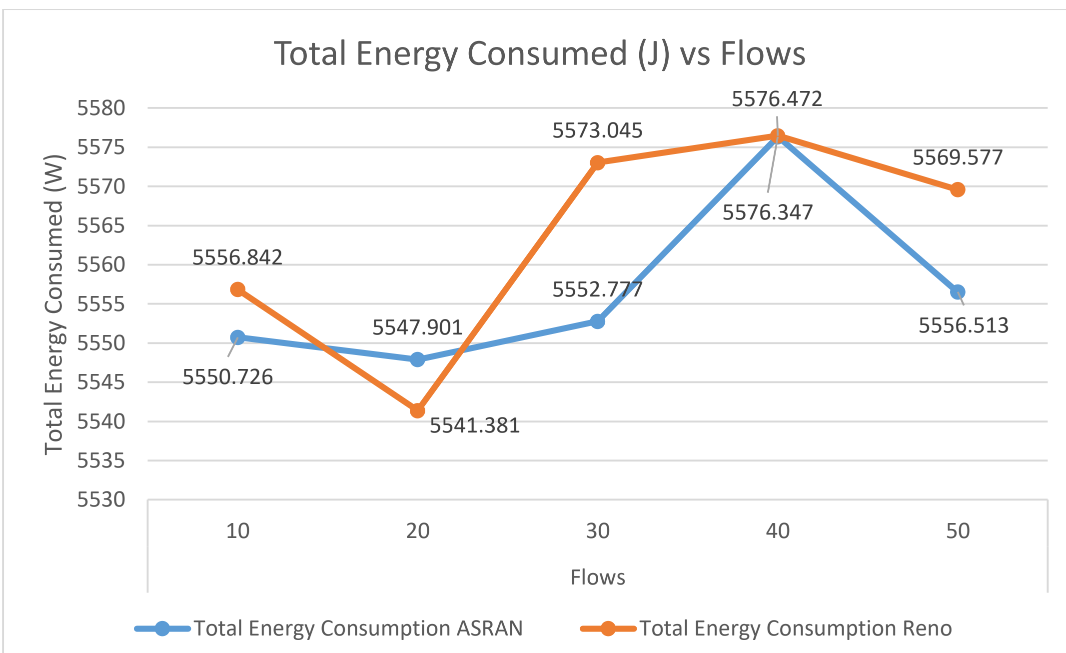
5.1.2.3 Change in Packet Delivery ratio:



5.1.2.4 Change in Packet Drop ratio:

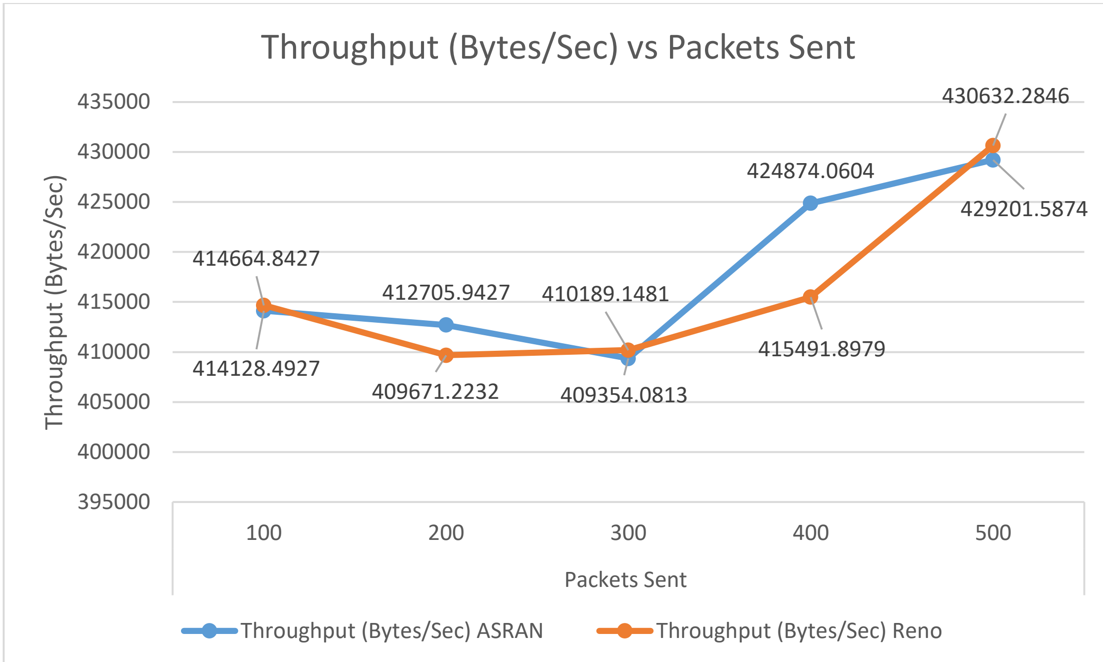


5.1.2.5 Change in Energy Consumption:

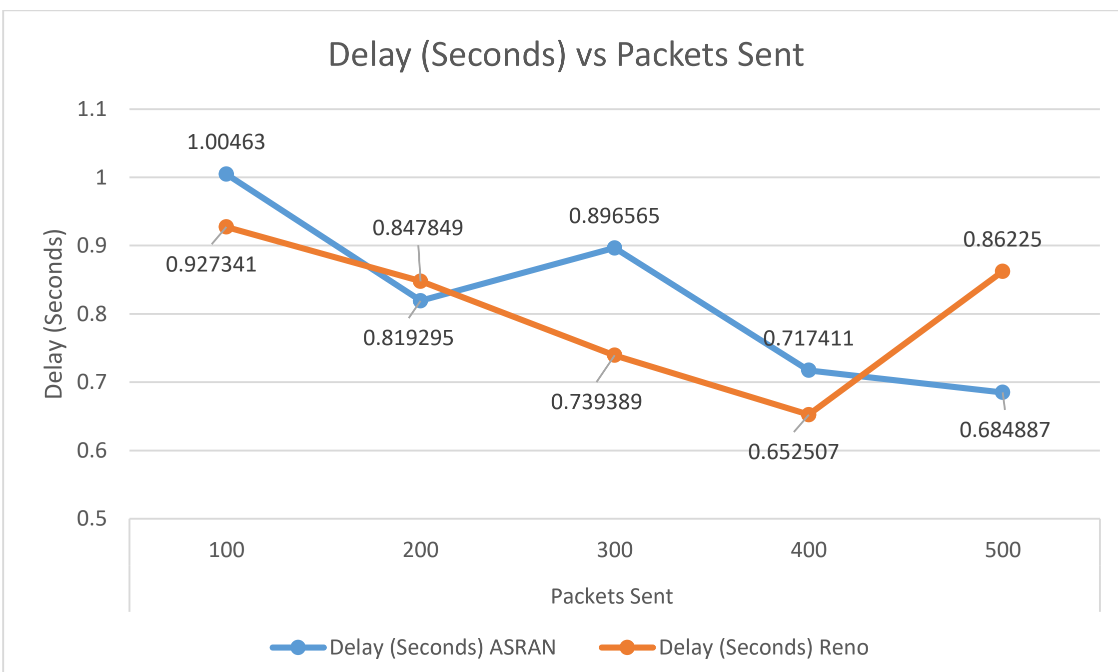


5.1.3 Varying Packets Sent:

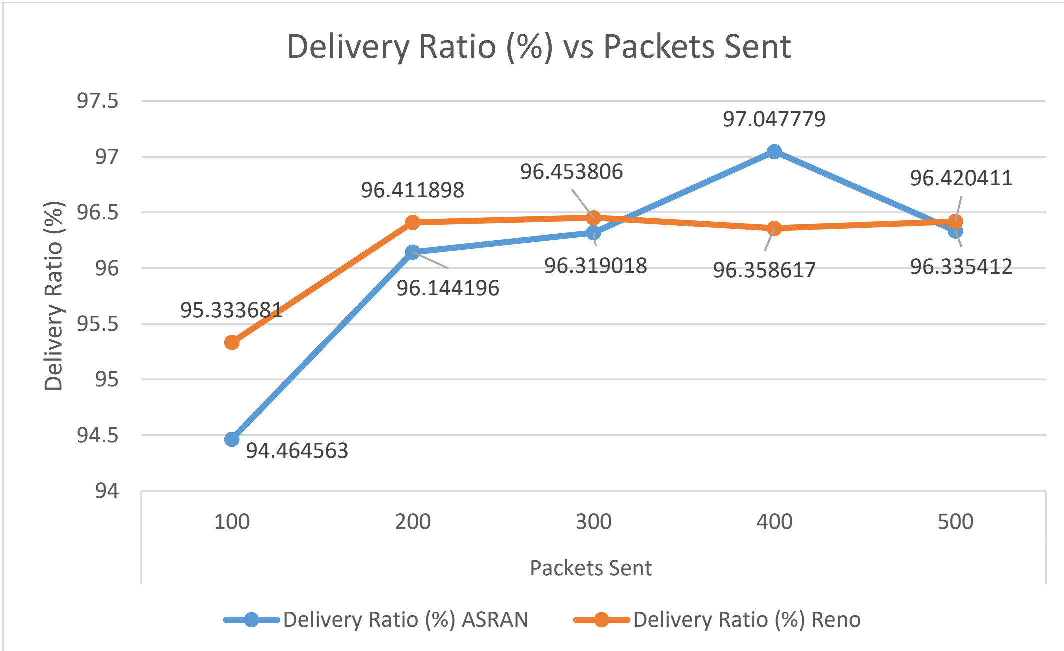
5.1.3.1 Change in Throughput:



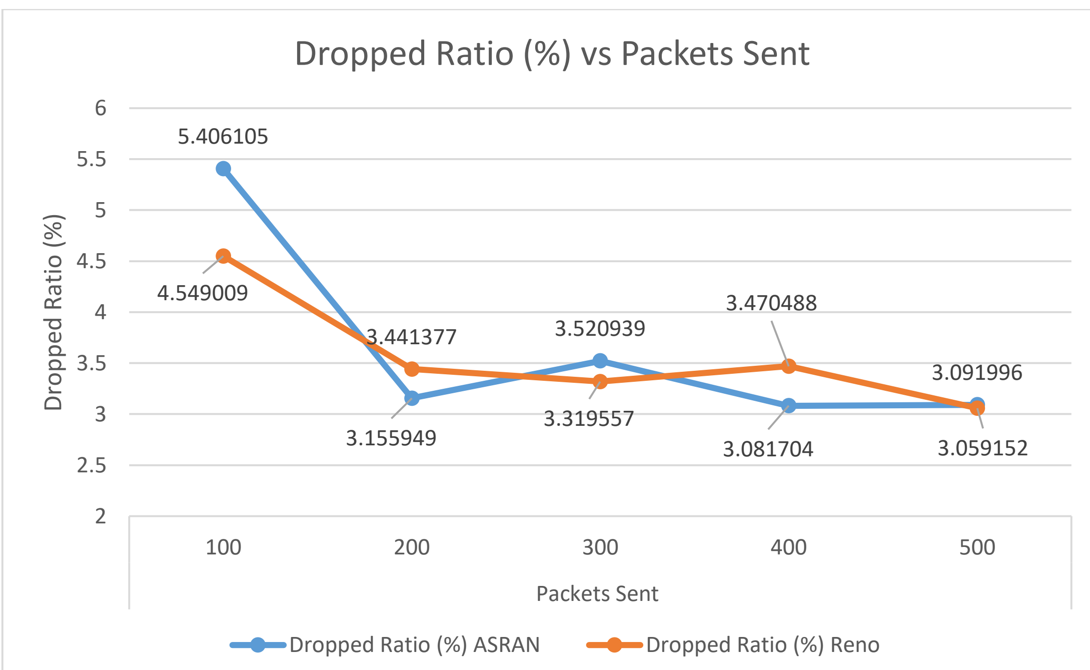
5.1.3.2 Change in End-to-end Delay:



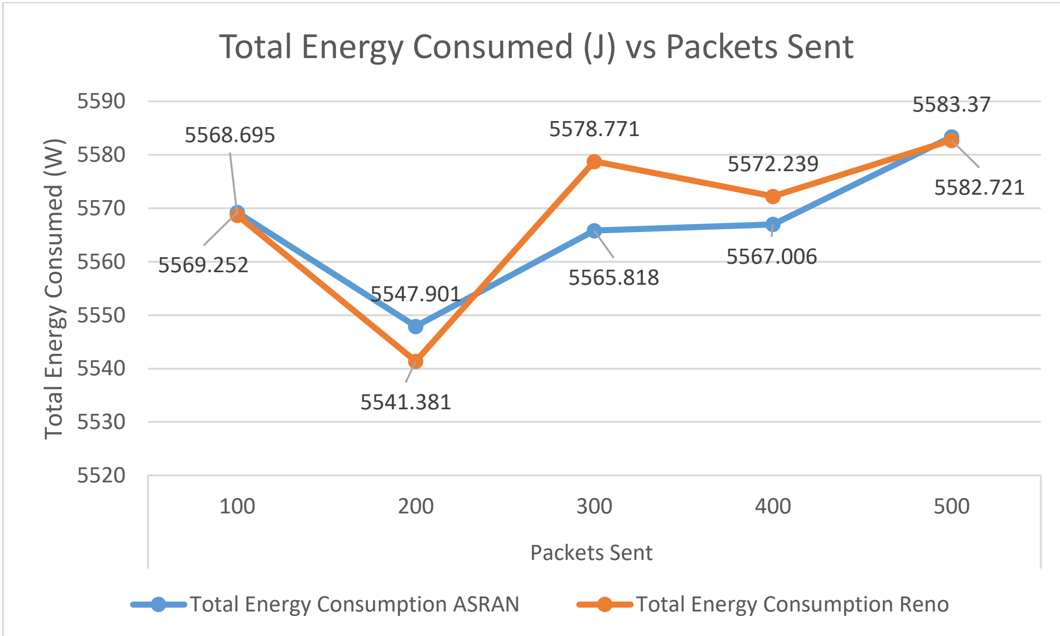
5.1.3.3 Change in Packet Delivery ratio:



5.1.3.4 Change in Packet Drop ratio:

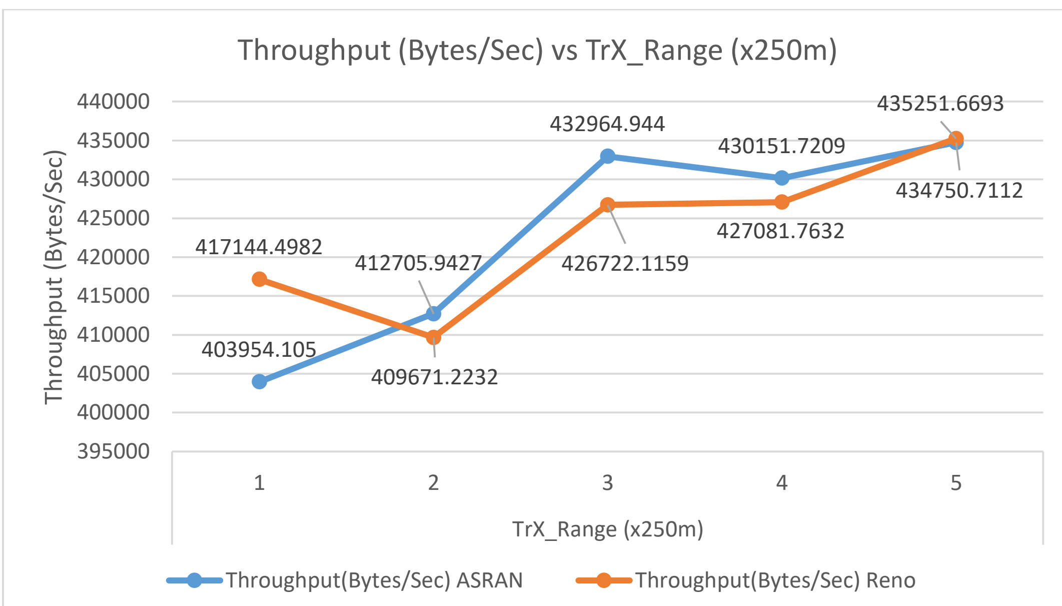


5.1.3.5 Change in Energy Consumption:

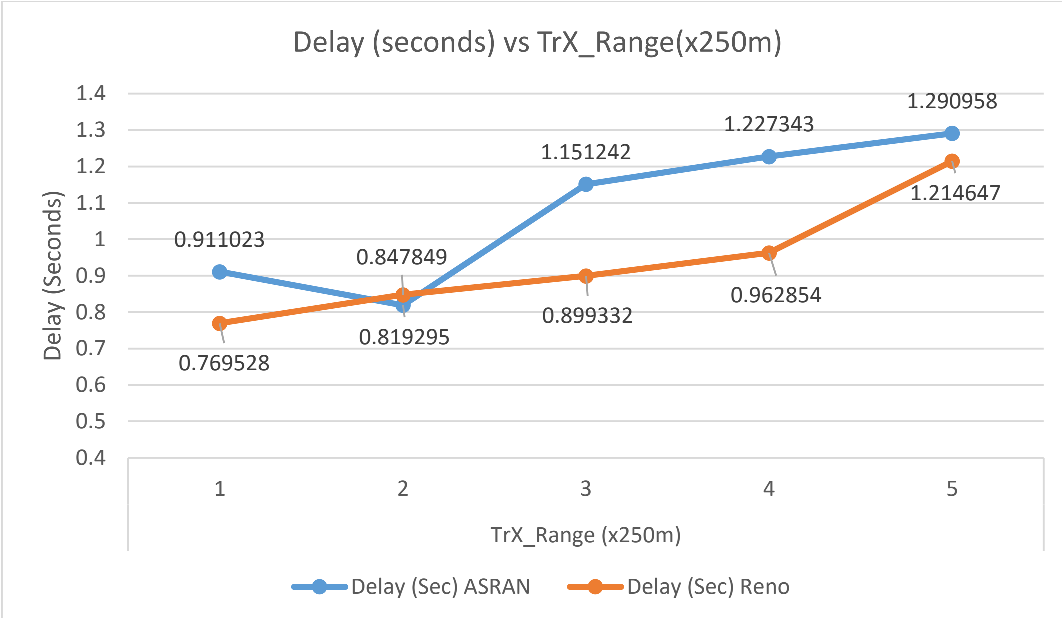


5.1.4 Varying TrX_Range:

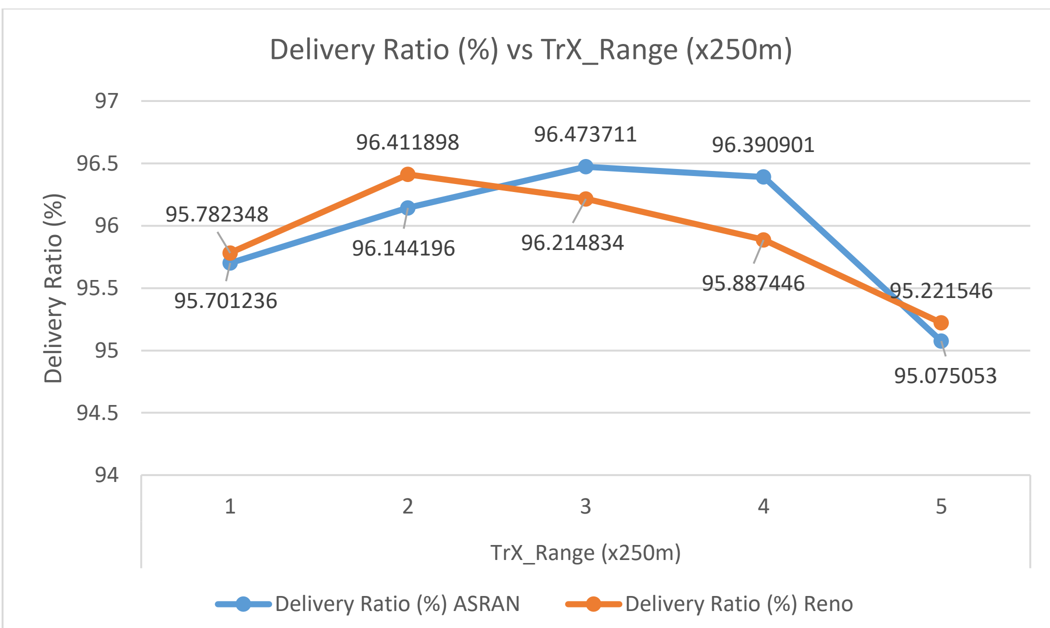
5.1.4.1 Change in Throughput:



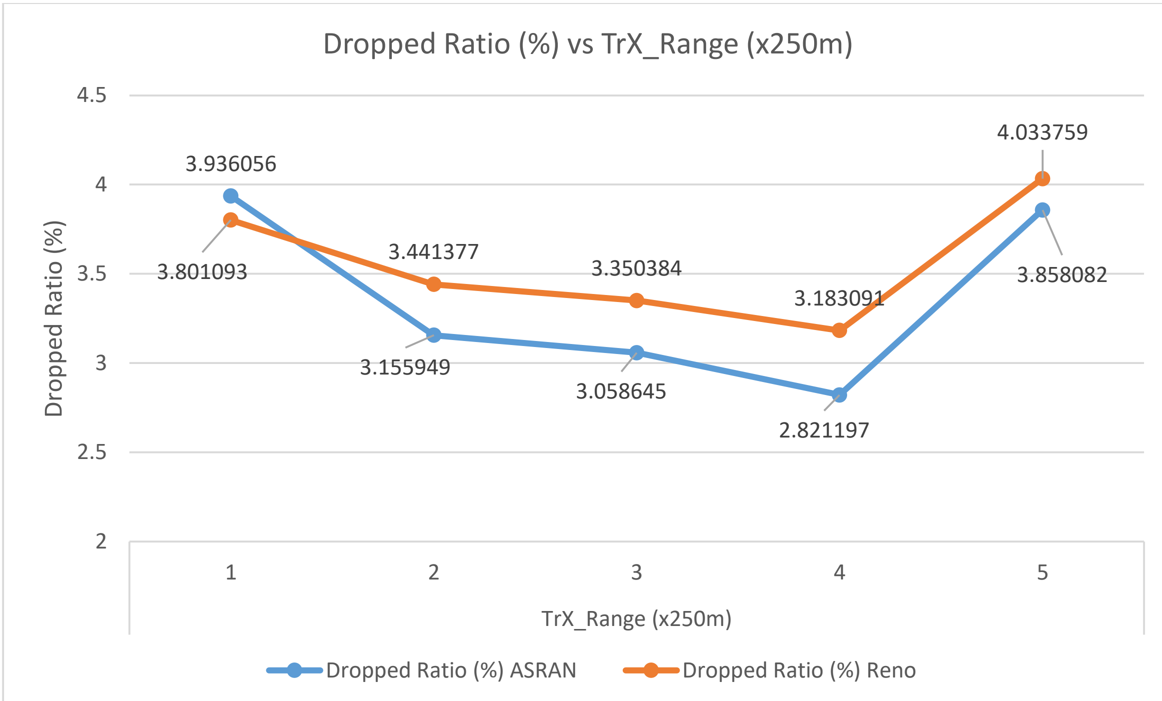
5.1.4.2 Change in End-to-end Delay:



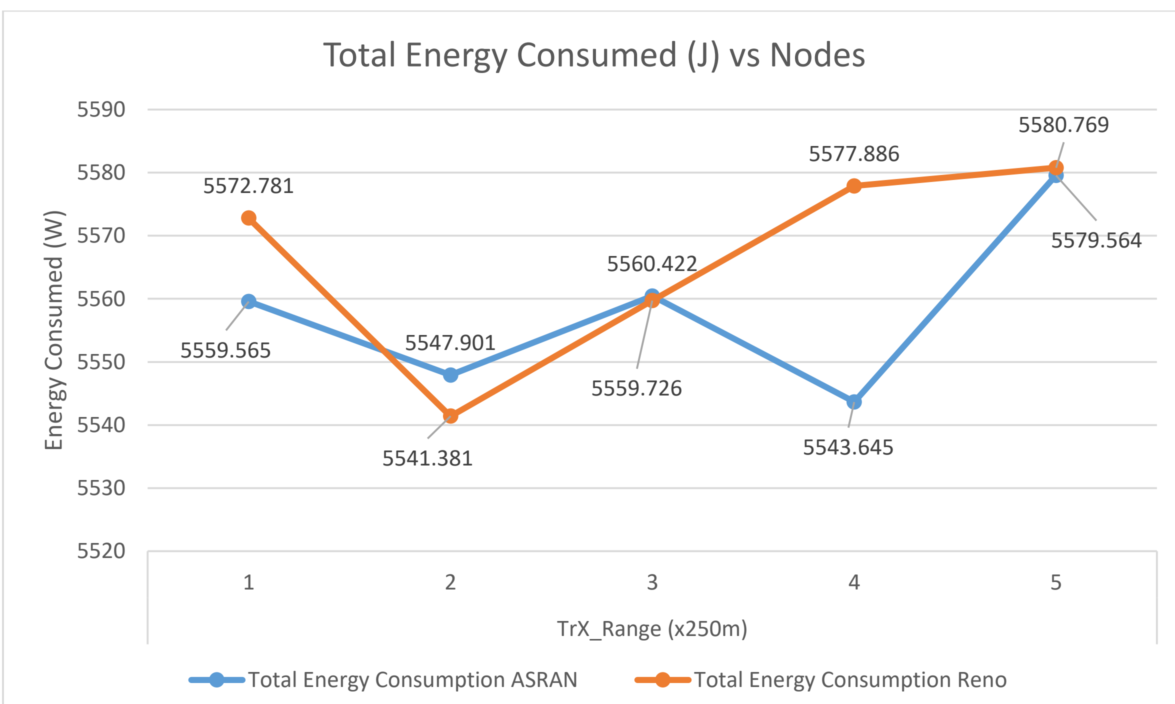
5.1.4.3 Change in Packet Delivery ratio:



5.1.4.4 Change in Packet Drop ratio:



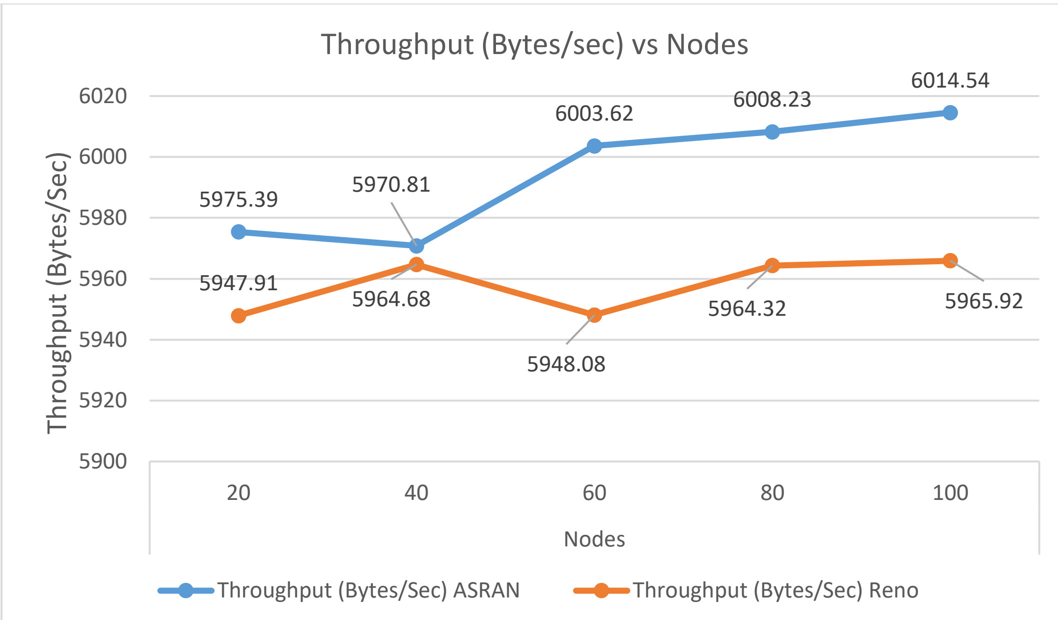
5.1.4.5 Change in Energy Consumption:



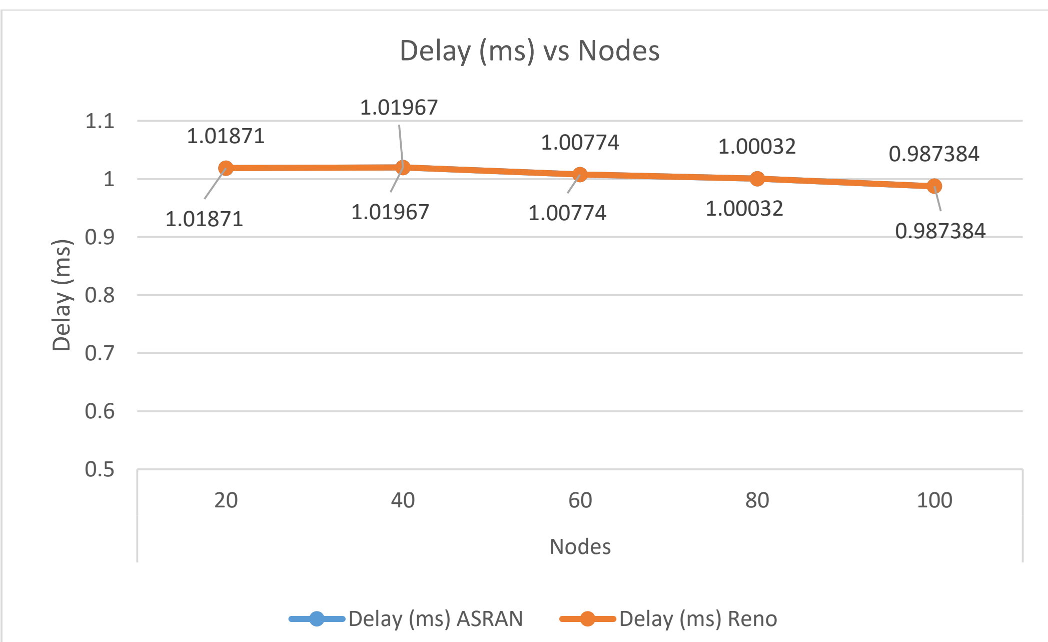
5.1 Wired:

5.2.1 Varying Node#:

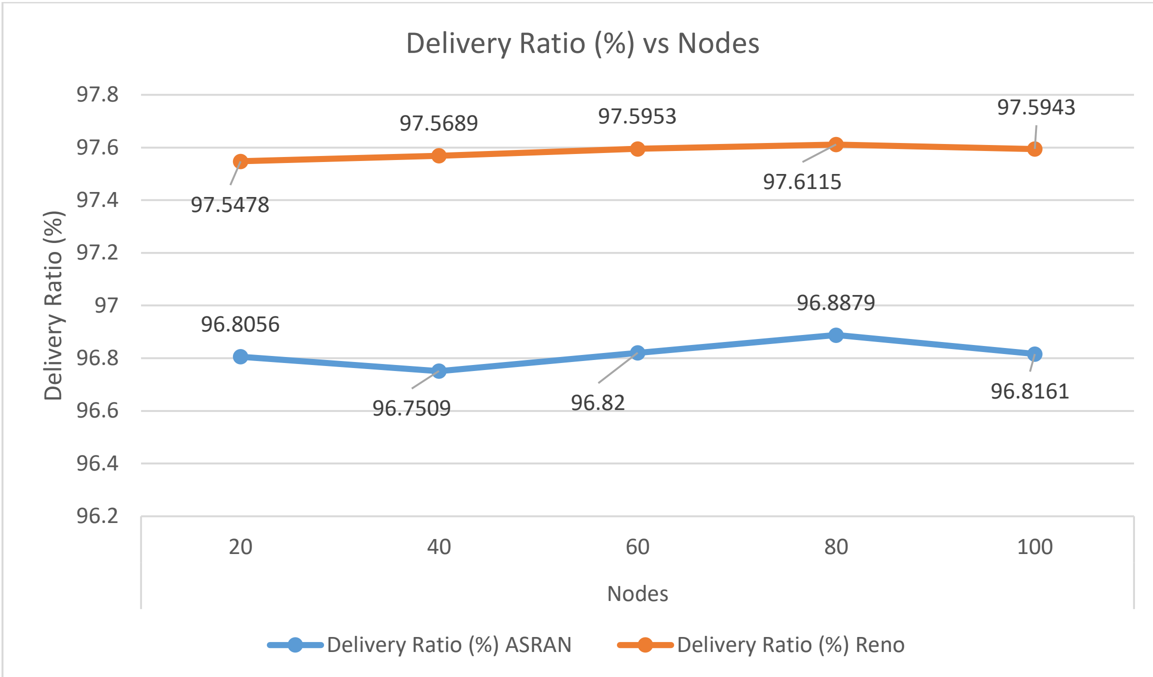
5.2.1.1 Change in Throughput:



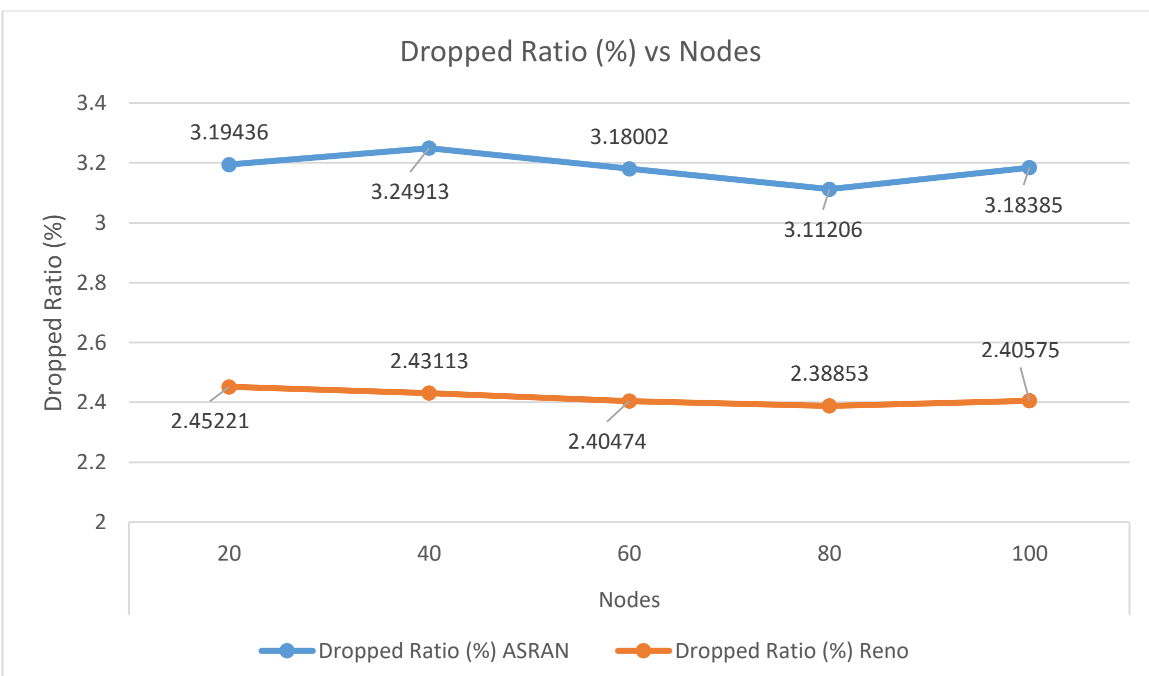
5.2.1.2 Change in End-to-end Delay:



5.2.1.3 Change in Packet Delivery ratio:

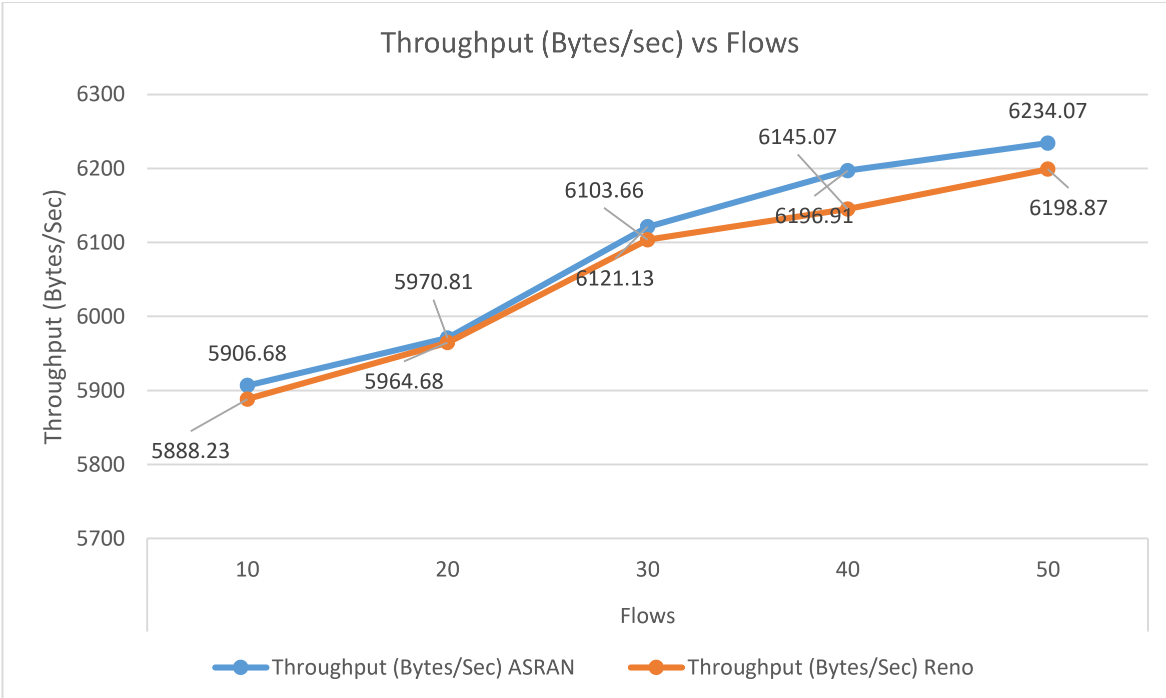


5.2.1.4 Change in Packet Drop ratio:

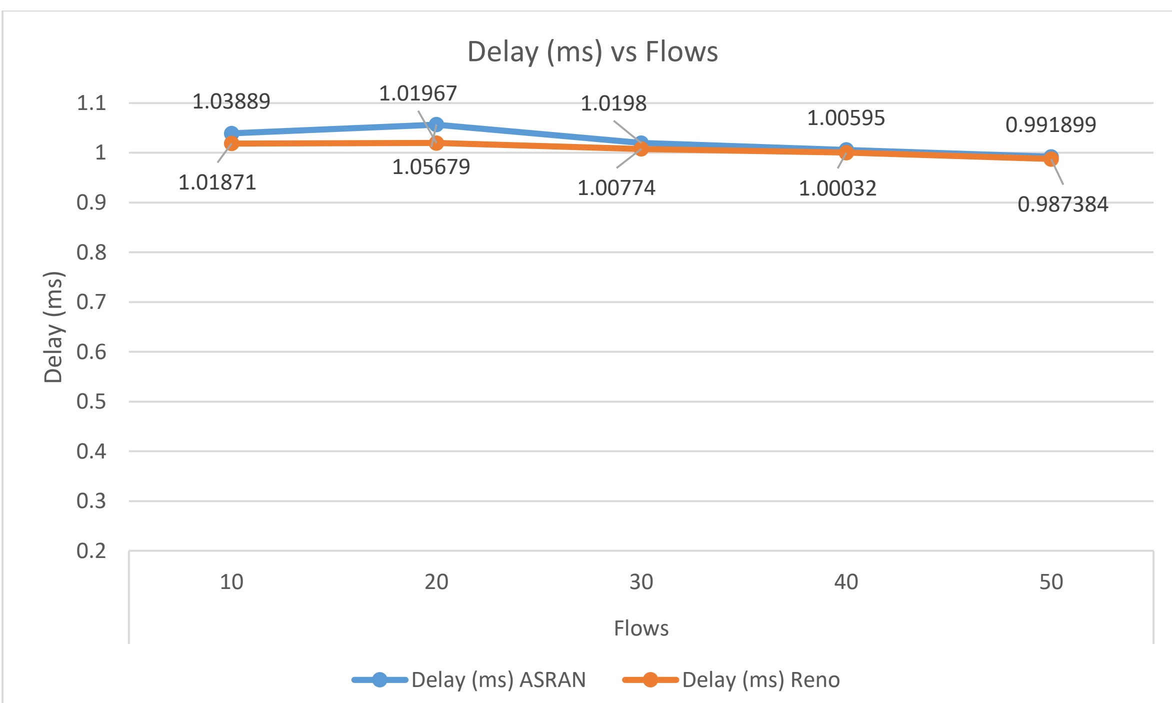


5.2.2 Varying Flow#:

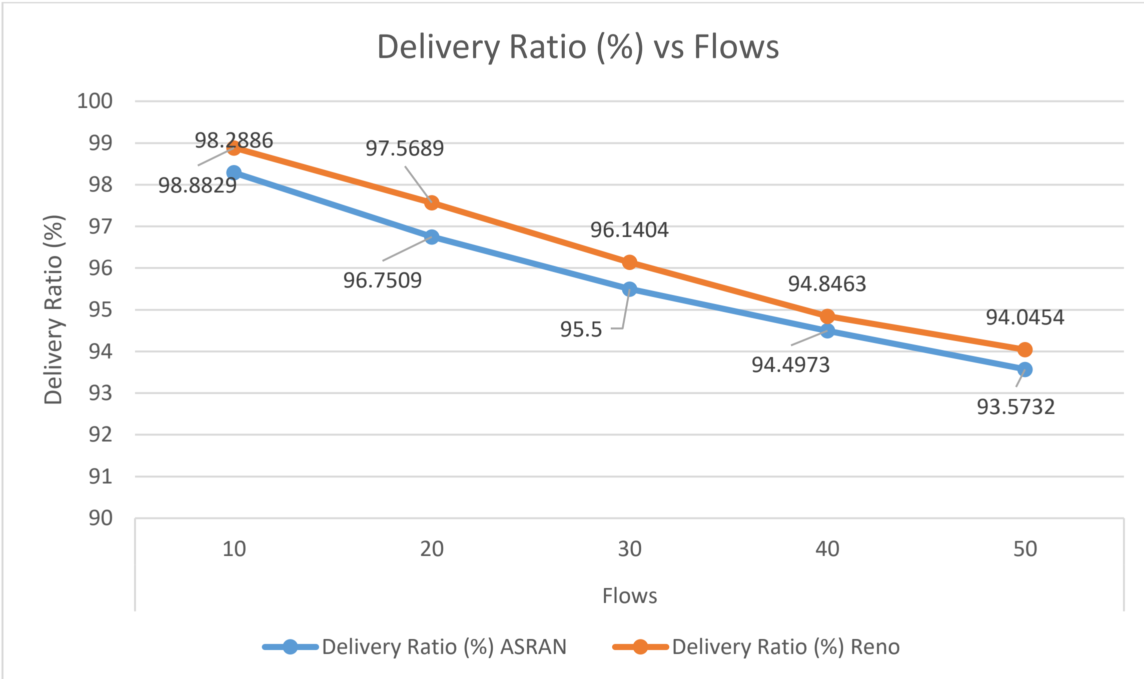
5.2.2.1 Change in Throughput:



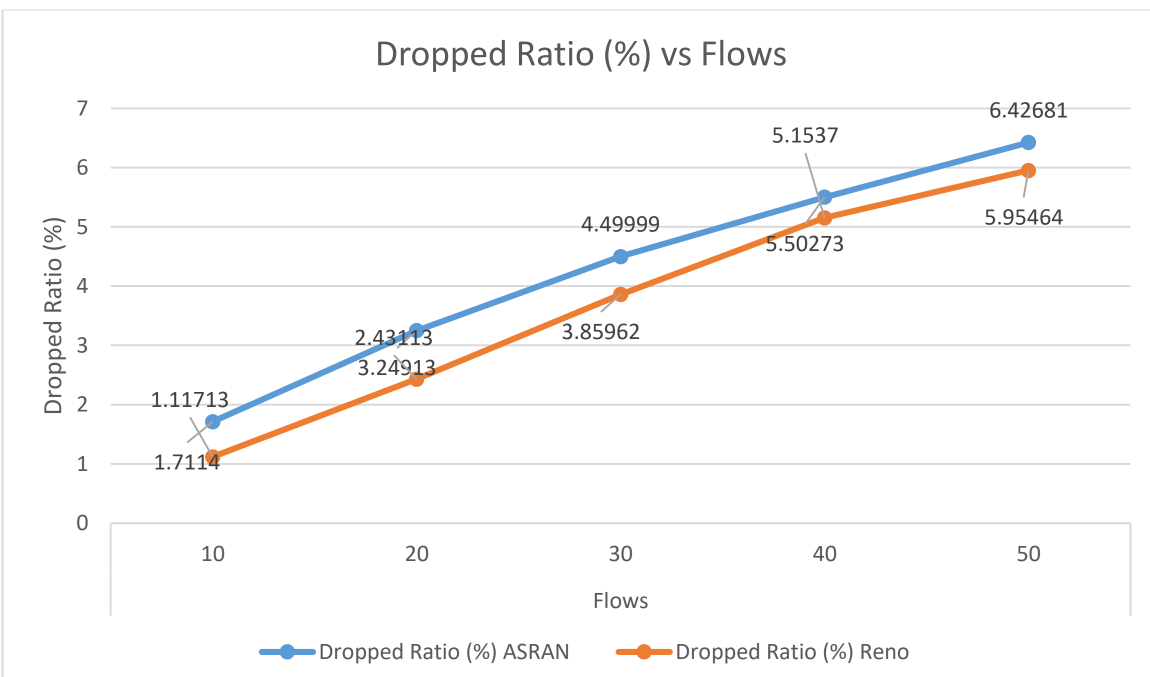
5.2.2.2 Change in End-to-end Delay:



5.2.2.3 Change in Packet Delivery ratio:

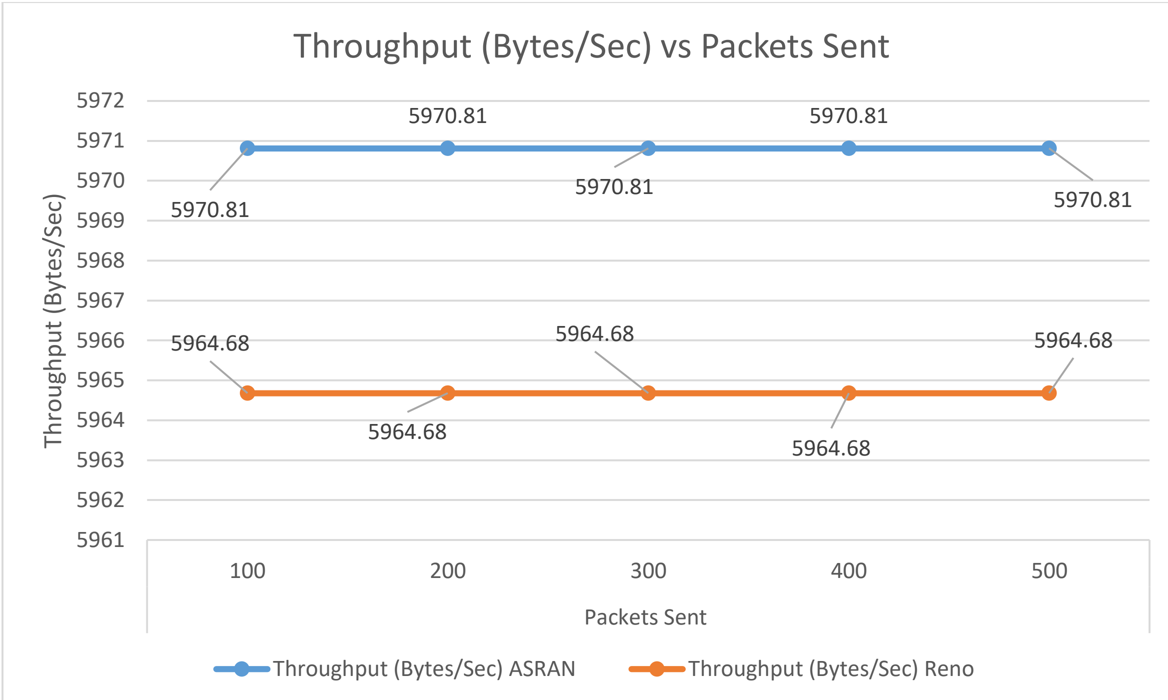


5.2.2.4 Change in Packet Drop ratio:

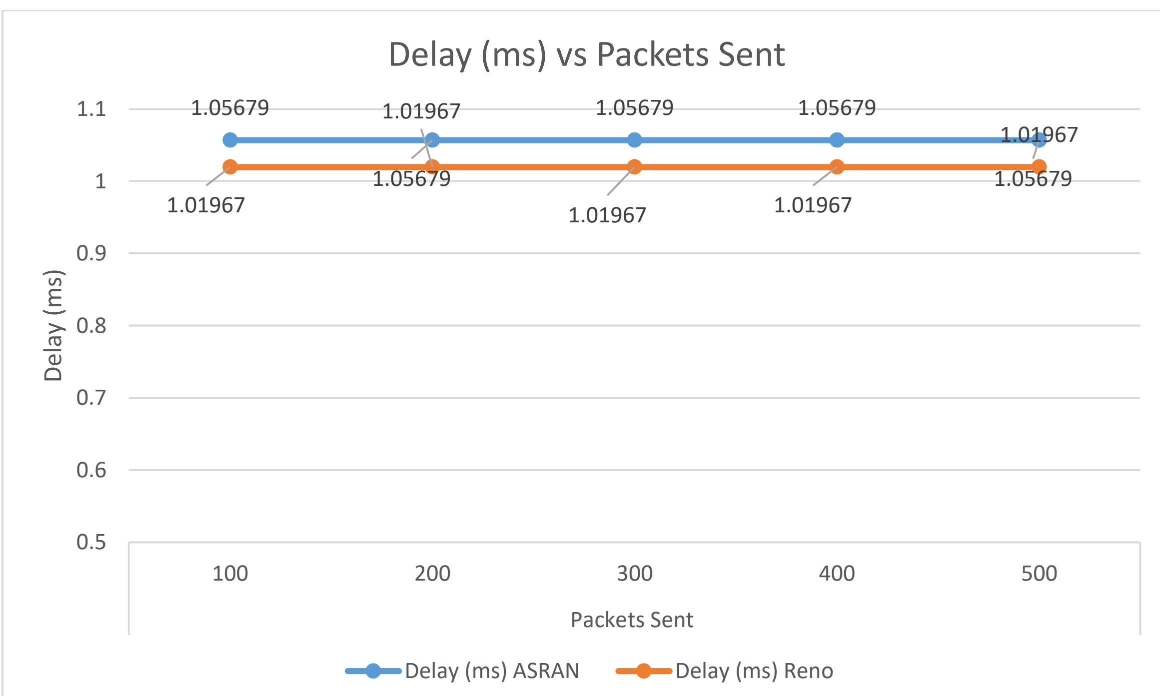


5.2.3 Varying Packets sent:

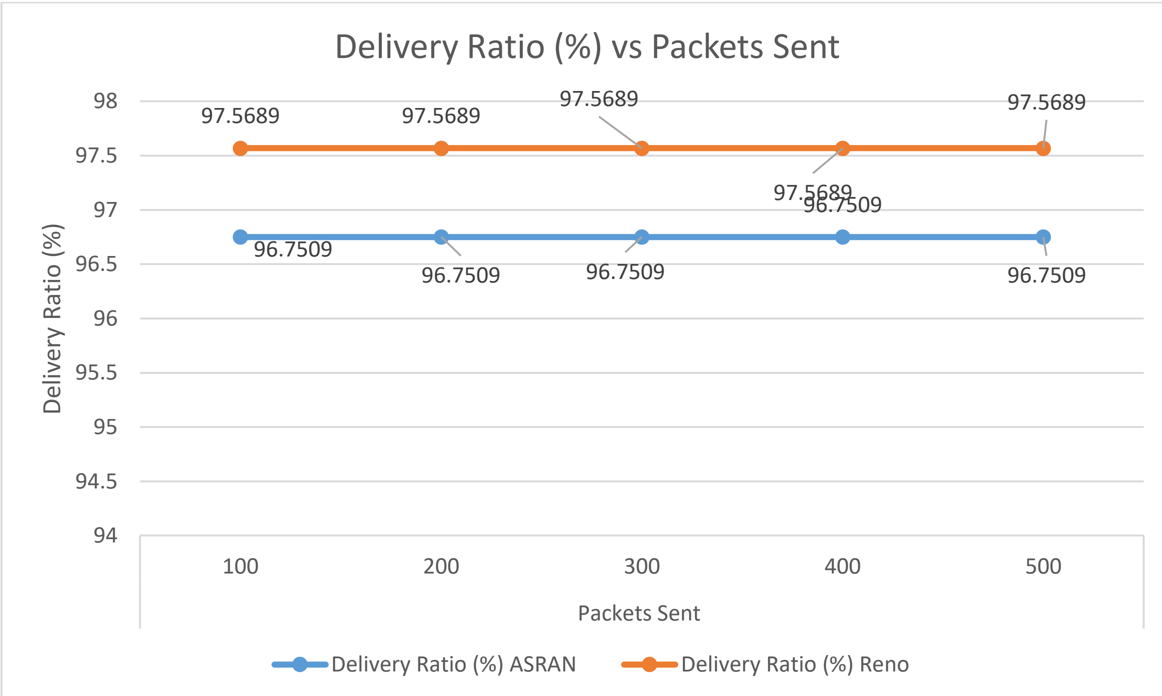
5.2.3.1 Change in Throughput:



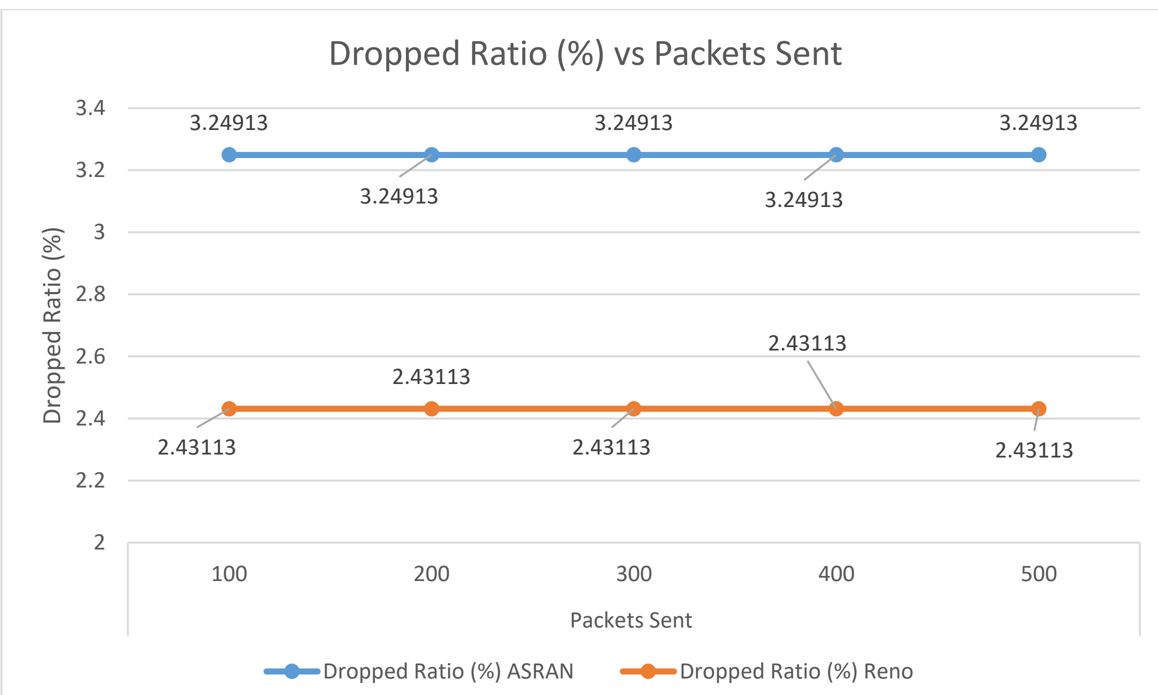
5.2.3.2 Change in End-to-end Delay:



5.2.3.3 Change in Packet Delivery ratio:



5.2.3.4 Change in Packet Drop ratio:



6. Summary Findings:

6.1 Wireless: (Averages)

	Throughput (Bytes/Sec)	Delay (Sec)	Delivery Ratio (%)	Dropped Ratio (%)	Total Energy Consumption
Reno	418769	0.922038	95.97786	3.620967	6256.358
Asran	419678.5	0.995151	95.84857	3.624112	6248.597

6.2 Wired: (Averages)

	Throughput (Bytes/Sec)	Delay (ms)	Delivery Ratio (%)	Dropped Ratio (%)
Reno	6009.142	1.015307	96.94017	3.059858
ASRAN	6040.219	1.036754	96.26905	3.730948

6.3 Summary of ASRAN vs Reno:

6.3.1 ASRAN algorithm shows improvement over the tradition TCPReno in cases of:

1. Throughput for both wired and wireless
2. Total Energy Consumption for wireless

6.3.2 ASRAN lacks behind in cases of:

1. Delay (both wired and wireless)
2. Delivery Ratio (both cases)

Overall, ASRAN is effective in increasing throughput, but is prone to more packet drops and higher end-to-end delay. It showed some fluctuations too, but overall improvement in throughput was achieved, as per data collected

6.4 General Trends:

6.4.1 Throughput:

- Clearly increases with transmission range increase
- Increases with Packets sent increase
- Decreases with flow increase (may be due to packet drop by congestion in this experiment)
- Seemingly increases with node increase

6.4.2 Delay:

- Increases with both TrX_Range increase and Flow increase
- Decreases with packet sent increase

6.4.3 Delivery Ratio (opposite of Drop Ratio):

- Decreases with flow increase (possibly due to congestion loss in the experiment)
- Other trends are not as obvious

6.4.4 Energy Consumption:

- Linearly increases with node#
- Other trends are not as obvious

6.5 Inferences and Disclaimers:

- Although the throughput gain was not as high as the value of 1.6 as proposed in the original paper, clear improvements could be observed.
- maxseq_ was the only option to control pkts/sec in FTP. This may not have had desired results.

7. Bonus:

7.1 Measuring new metric (Throughput (bytes/sec) per node)

The metric of throughput per node was measured in parsePerNode.awk file for our wired simulation for base parameters.

7.1.1 Basic Pseudocode (in parsePerNode.awk file)

- In begin, Initialize each element of receivedbytes_arr to 0
- For each received byte, do receivedbytes _arr[to_node] ++;
- In end, calculate throughputPerNode[i] = receivedbytes_arr[i] * 8 / simulation_time

7.1.2 Summary of findings is as follows:

Node#	Throughput	Node#	Throughput	Node#	Throughput	Node#	Throughput
0	0	10	0	20	1938.61	30	0
1	0	11	0	21	112.519	31	87.7357
2	0	12	0	22	206.698	32	95.8316
3	0	13	0	23	0	33	311.617
4	0	14	0	24	0	34	112.519
5	0	15	0	25	167.376	35	91.701
6	0	16	0	26	0	36	96.162
7	0	17	0	27	0	37	103.267
8	0	18	0	28	351.108	38	92.5271
9	0	19	2087.47	29	0	39	109.545

This sums to 5970.81 which is the total throughput of our simulation for that parameter set.

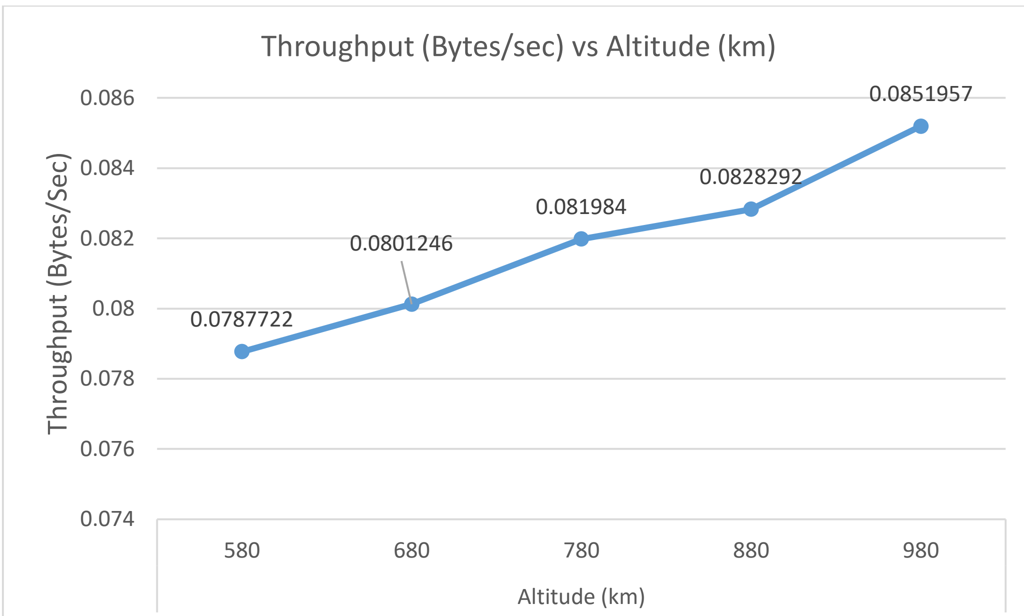
7.2 Simulating any network not mentioned above (Satellite network)

We simulated the iridium satellite and collected data by varying the altitude and inclination. We keep the base altitude as 780km, inclination as 90 degrees w.r.t. the equator

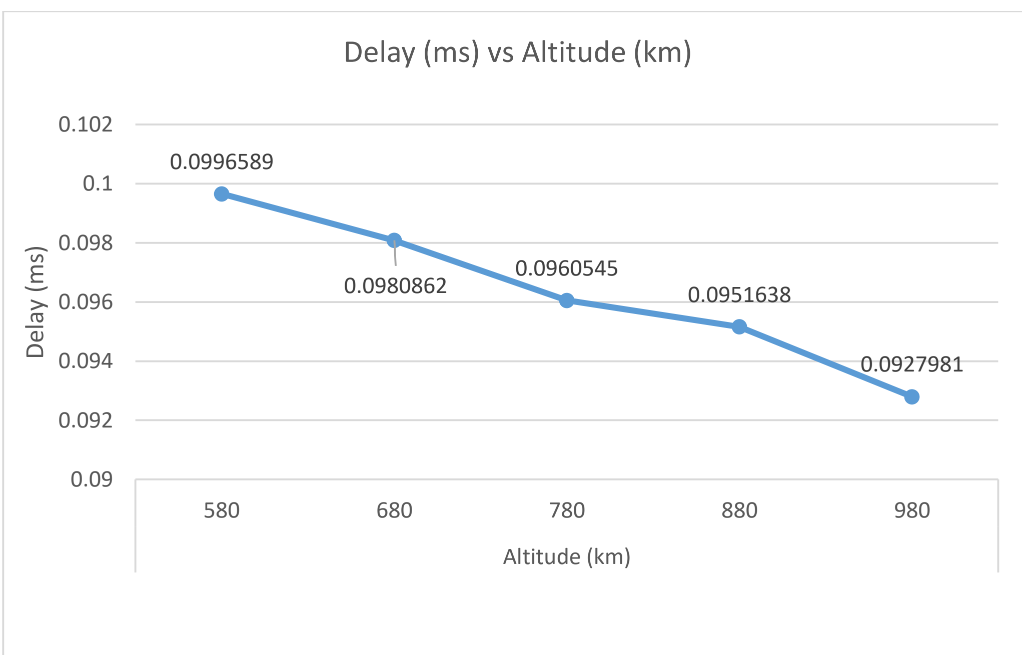
Graphs

7.2.1 Varying Altitude

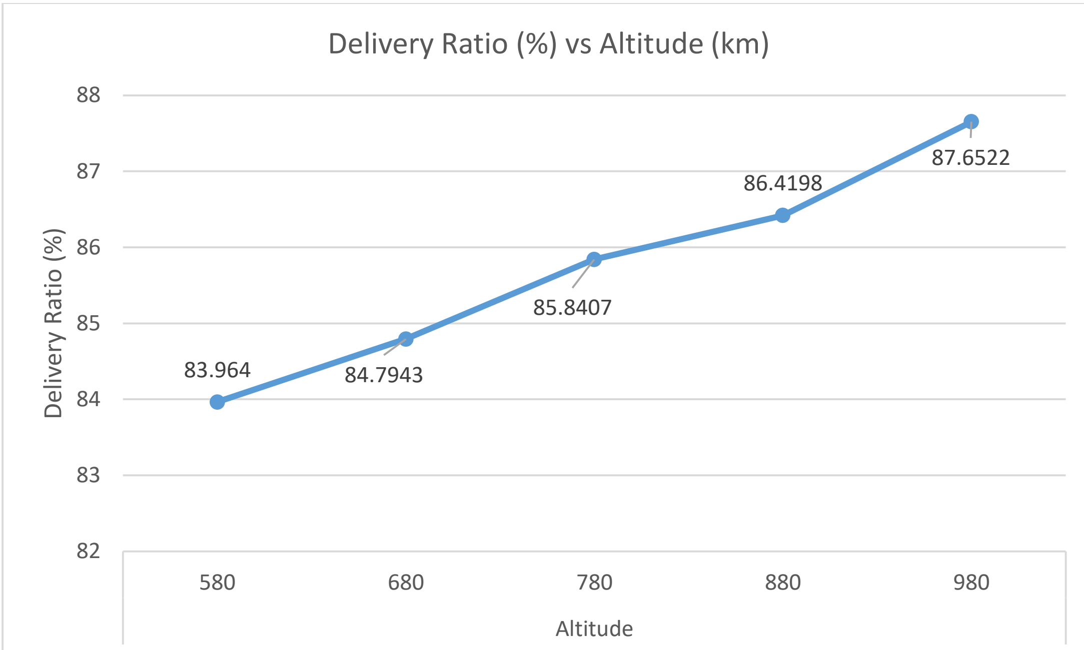
7.2.1.1 Change in Throughput:



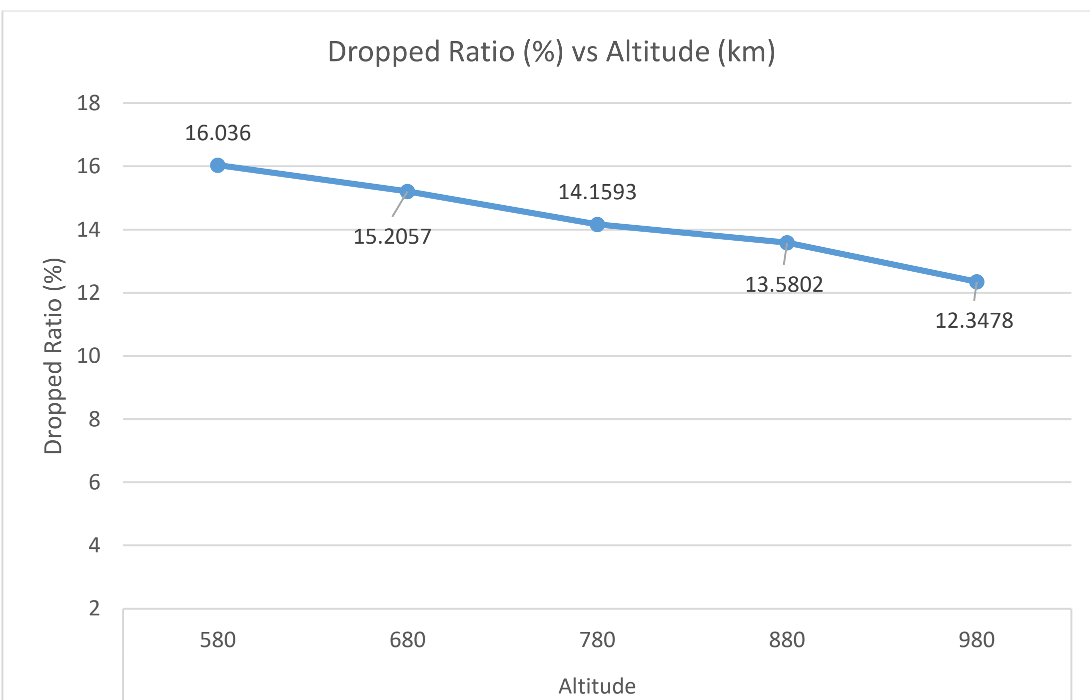
7.2.1.2 Change in End-to-end Delay:



7.2.1.3 Change in Packet Delivery ratio:

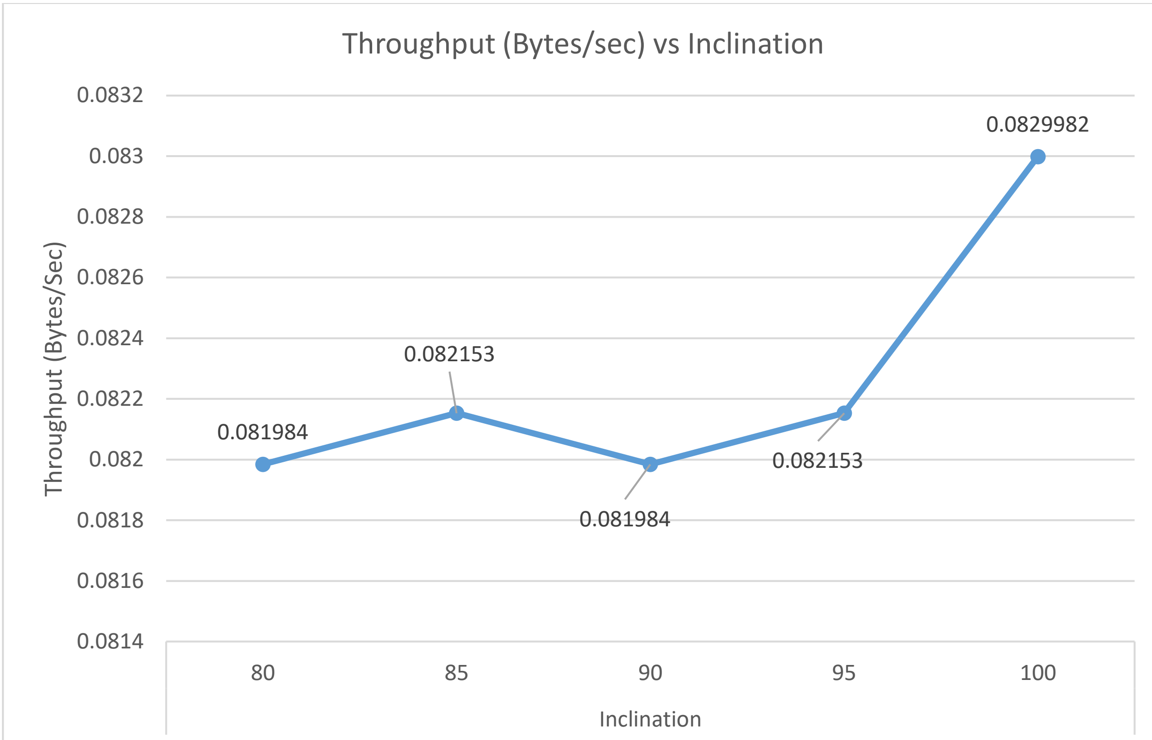


7.2.1.4 Change in Packet Drop ratio:

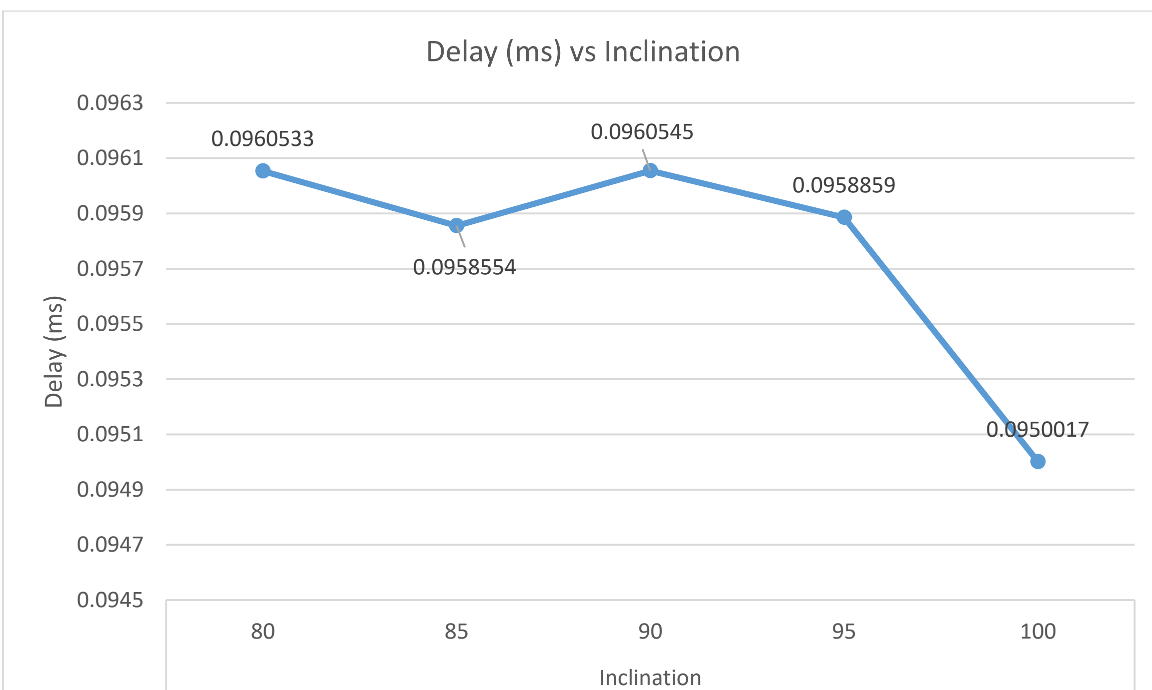


7.2.1 Varying Inclination

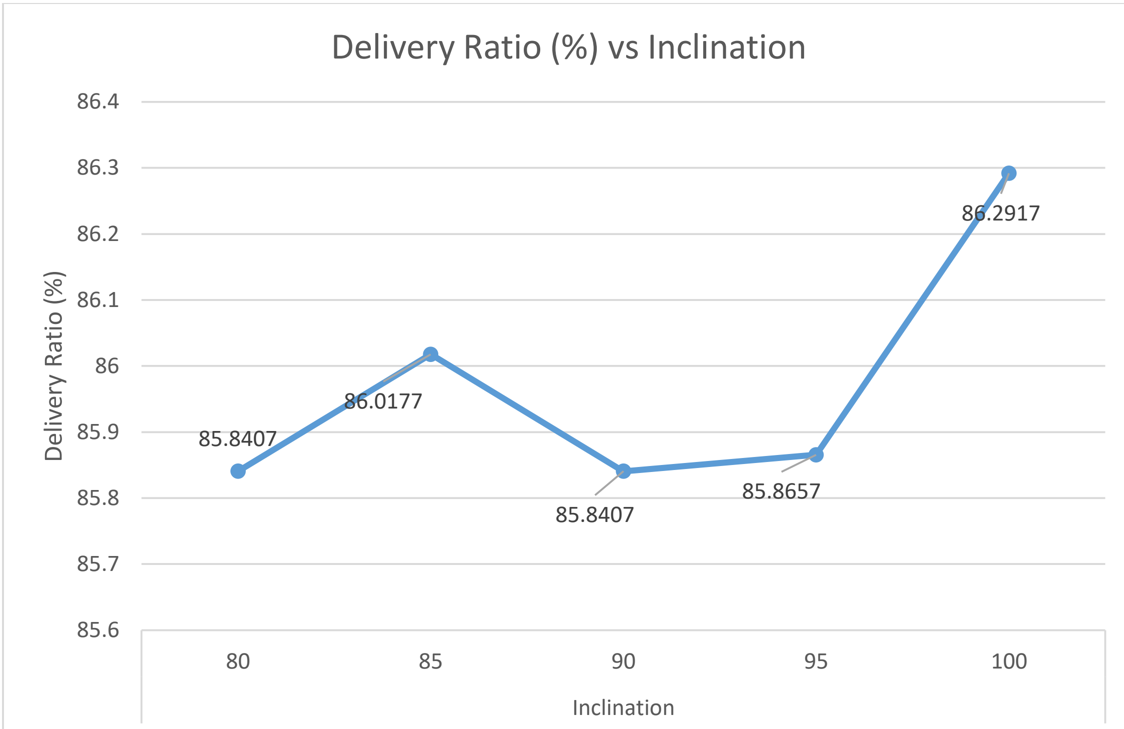
7.2.1.1 Change in Throughput:



7.2.1.2 Change in End-to-end Delay:



7.2.1.3 Change in Packet Delivery ratio:



7.2.1.4 Change in Packet Drop ratio: