CSE 322 Project Report

Submitted By

Nazmul Takbir
ID: 1705103
Section: B2

Supervised By

Md. Toufikuzzaman
Lecturer
Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology

February 22, 2022

Contents

1 Task A: Wired Network

1.1 Topology

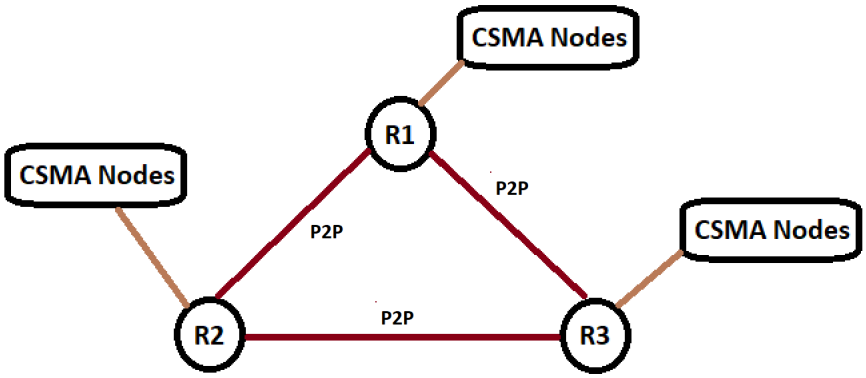
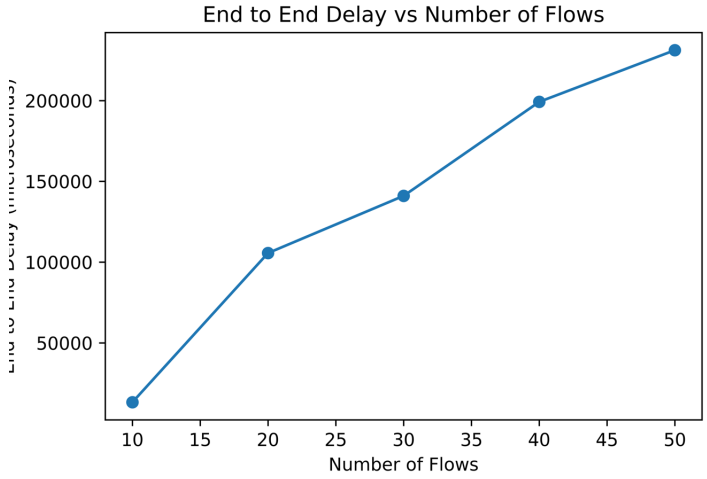


Figure 1: Wired Topology

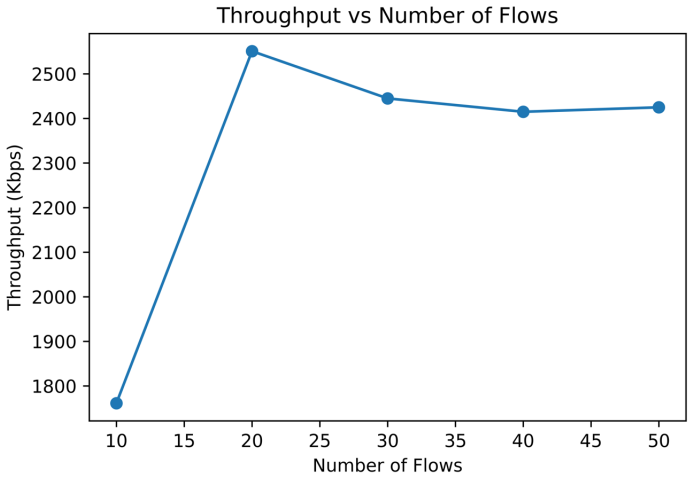
The wired topology used has 3 router nodes connected to each other using point to point connections. Also each router node is connected to its own CSMA network of nodes. When we vary the parameter "number of nodes", it is the number of CSMA nodes that we are changing. The number of nodes in each CSMA is kept more or less equal - 1 network has some extra nodes when the total number of nodes is not a multiple of 3. Each flow in the network is between 2 nodes in 2 different CSMA networks.

1.2 Results

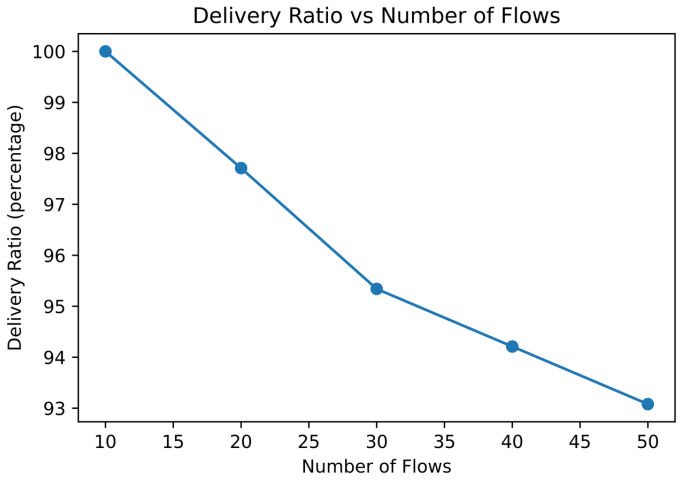
1.2.1 Varying Number of Flows



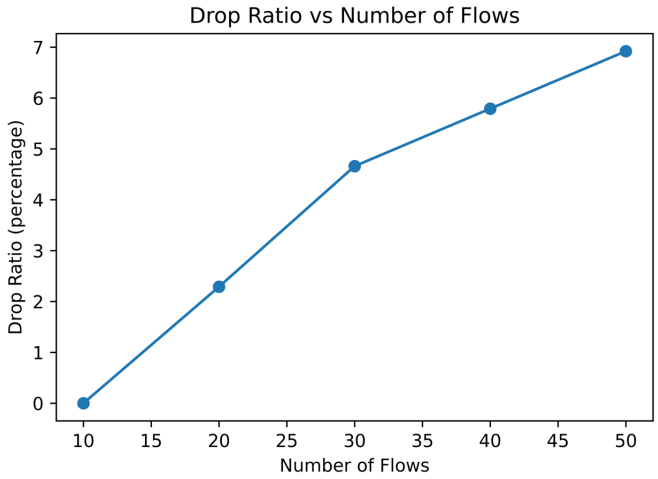
The end to end network delay increases as the number of flows is increased. This makes sense since as the number of flows increase, there is more competition for network resources and as such queuing delays increase. The network throughput increases at first as the number of flows increases. This happens as a greater number of flows makes



more use of the bandwidth available in the network. However, when the number of flows increase beyond a certain level, the throughput starts dropping since the network starts experiencing congestion. The packet delivery ratio decreases and the packet drop



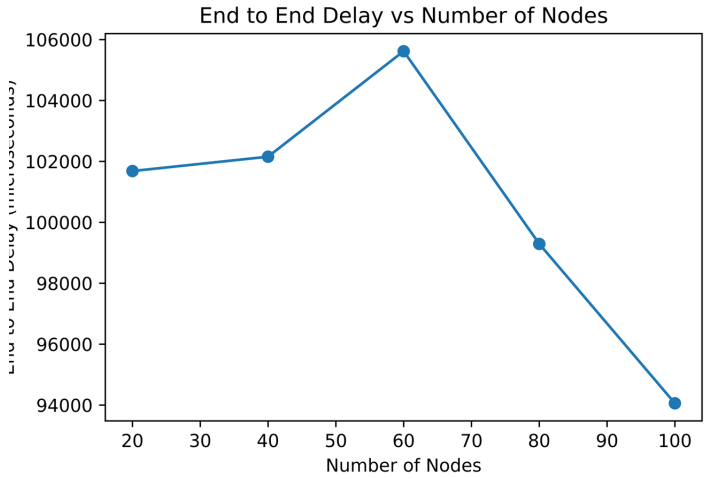
ratio increases as the number of flows increase. Both of these can be explained using congestion in the network - as the number of flows increase, the network faces more congestion and packets are dropped at the networking queues.

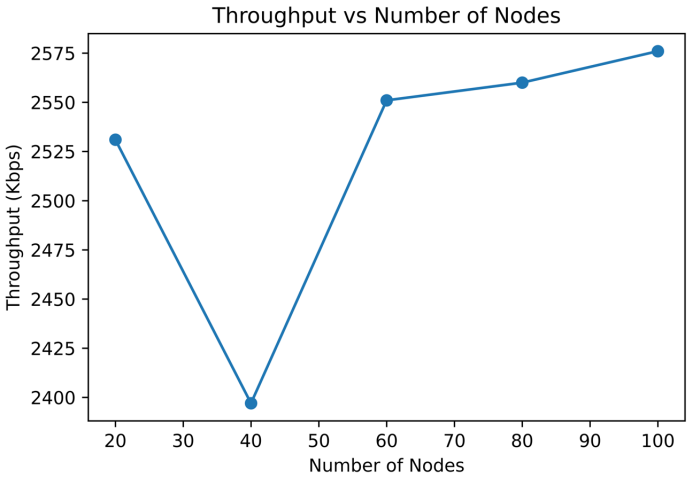


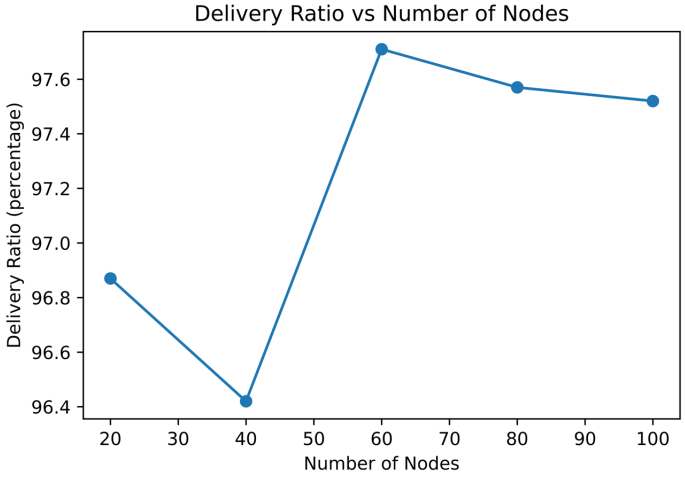
1.2.2 Varying Number of Nodes

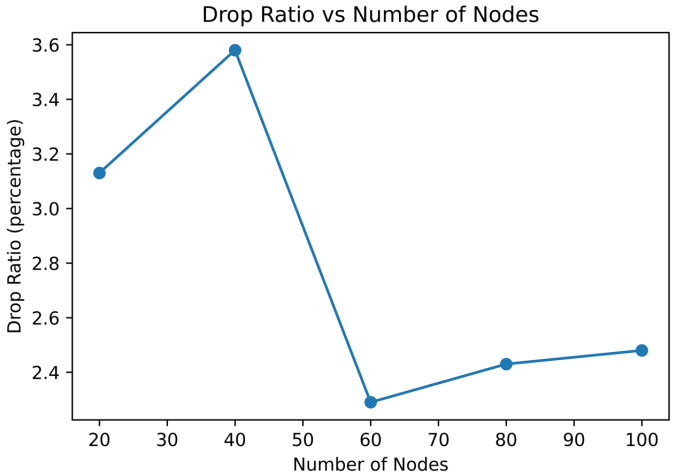
When the number of flows and the packets per second of each flow is kept constant, increasing the number of node is nothing but adding more idle nodes to the network. In a CSMA network, idle nodes shouldn't have much effect on the network metric.

A first glance at the graphs below though seems to suggest otherwise. However, on closer inspection we can see that when the number of nodes is increased 10 times the percentage change in the metric values is not actually too large.



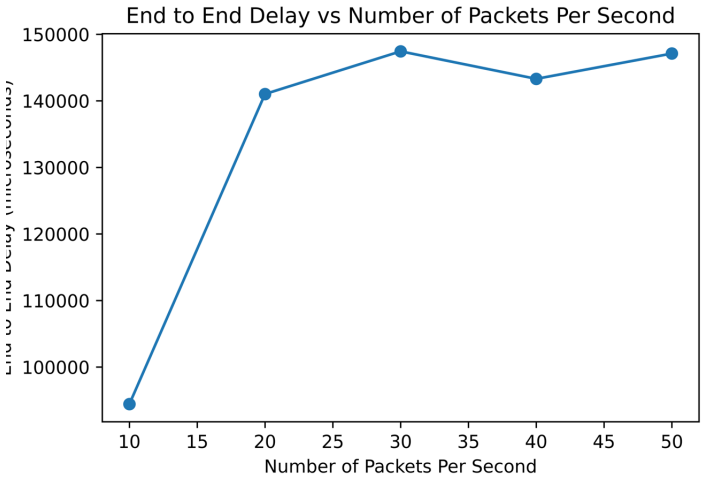

Throughput vs Number of Nodes
Throughput (Kbps)
2575
2550
2525
2500
2475
2450
2425
2400
20
30
40
50
60
70
80
90
100
Number of Nodes


Delivery Ratio vs Number of Nodes
Delivery Ratio (percentage)
97.6
97.4
97.2
97.0
96.8
96.6
96.4
20
30
40
50
60
70
80
90
100
Number of Nodes

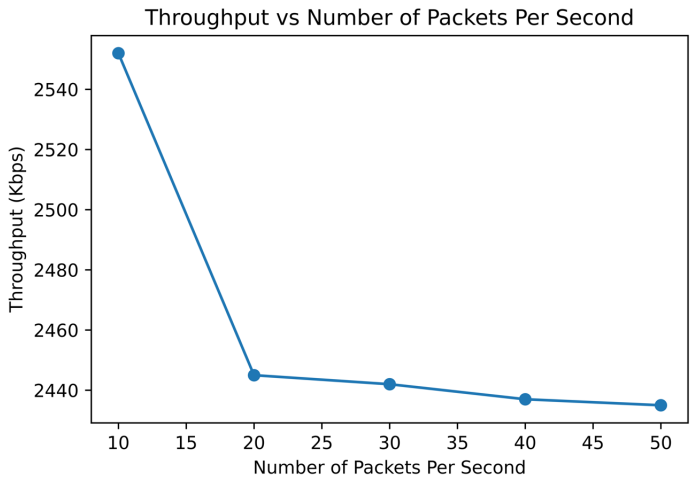

Drop Ratio vs Number of Nodes
Drop Ratio (percentage)
3.6
3.4
3.2
3.0
2.8
2.6
2.4
20
30
40
50
60
70
80
90
100
Number of Nodes

1.2.3 Varying Number of Packets Per Second

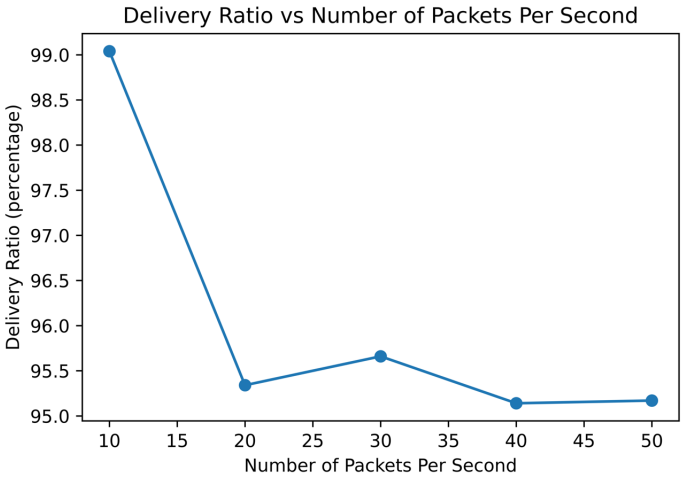
End to end delay increases as the number of packets are second is increased. This is because a larger rate of sending packets causes more congestion and queuing delays in the network.

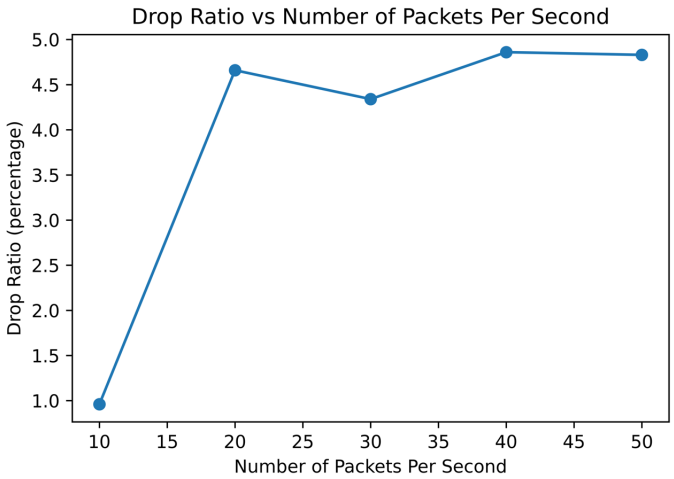


The throughput decreases as the number of packets per second is increased due to increased congestion leading to larger delays and more packet drops in the network.



The delivery ratio decreases and the drop rate increases as the number of packets per second increases since a larger rate of sending packets is more likely to cause the network queues to become full leading to packet drop.


Delivery Ratio vs Number of Packets Per Second
Delivery Ratio (percentage)
99.0
98.5
98.0
97.5
97.0
96.5
96.0
95.5
95.0
10
15
20
25
30
35
40
45
50
Number of Packets Per Second


Drop Ratio vs Number of Packets Per Second
Drop Ratio (percentage)
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
10
15
20
25
30
35
40
45
50
Number of Packets Per Second

2 Task A: WPAN Network

2.1 Topology

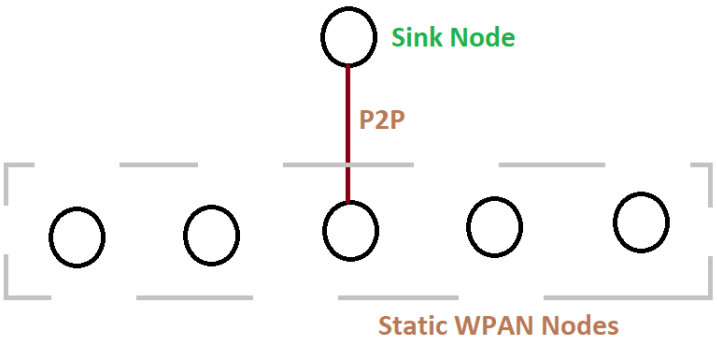


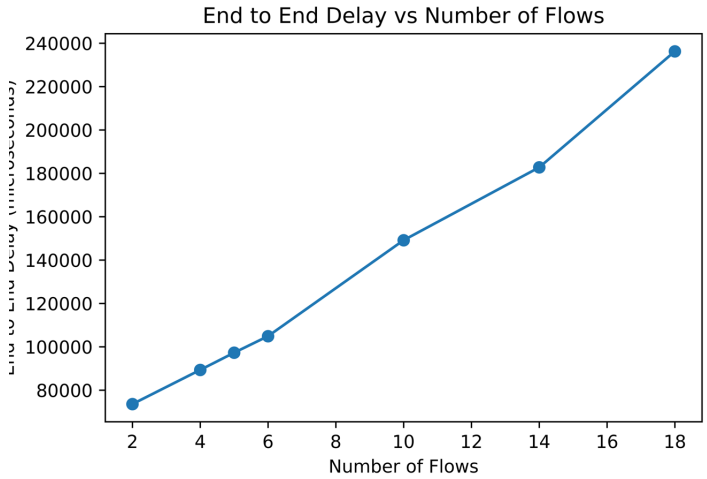
Figure 2: WPAN Topology

The WPAN topology has a variable number of wireless static WPAN nodes placed along a line. One of these nodes is connected via point to point connection to a sink node. The sink node receives the packets sent by the WPAN nodes.

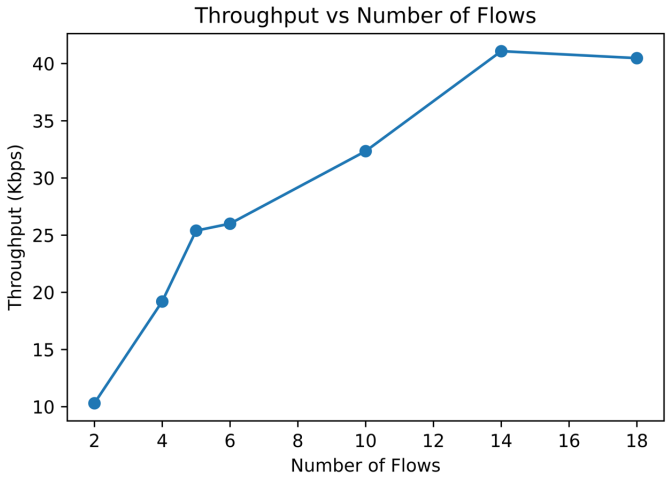
2.2 Results

2.2.1 Varying Number of Flows

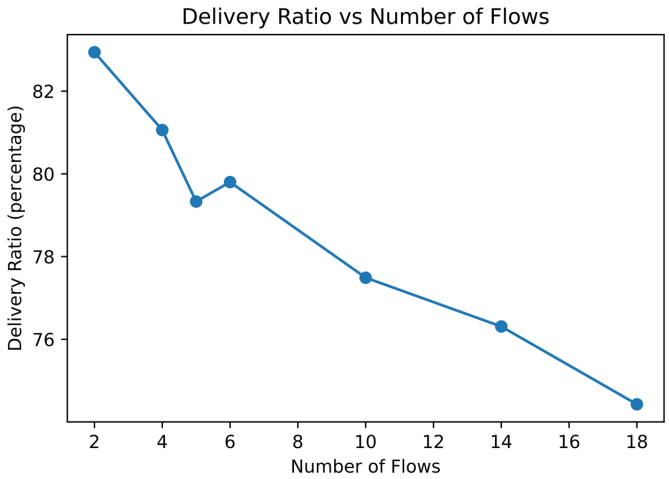
Increasing the number of flows increases the end to end delay in the network due to increased congestion.

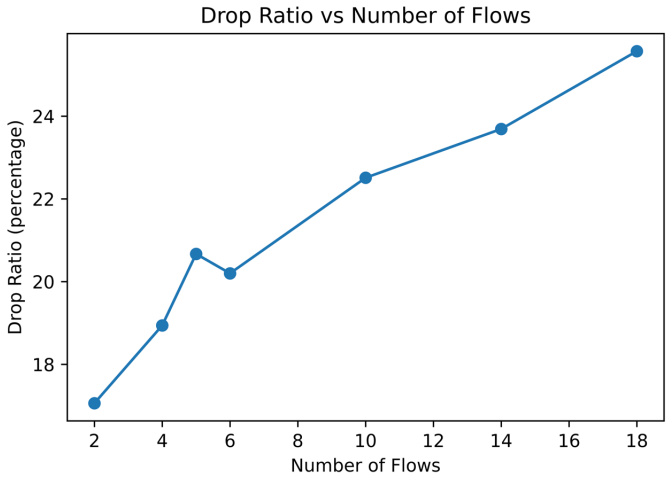


When the number of flows increases, the network bandwidth can be better utilized, but also it increases the chances of network congestion. It seems in this case the former factor is more prevalent. Hence, as the number of flows is increased the throughput increases.



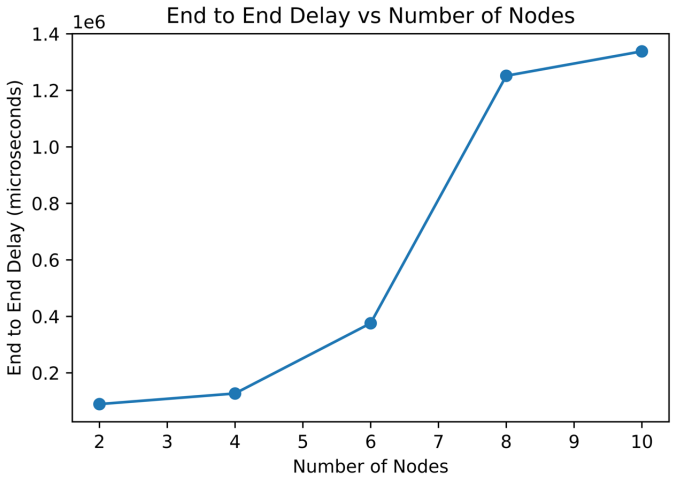
The delivery ratio decreases and the drop rate increases as the number of flows increases since the greater the number of flows the greater the chances of the network queues becoming full.

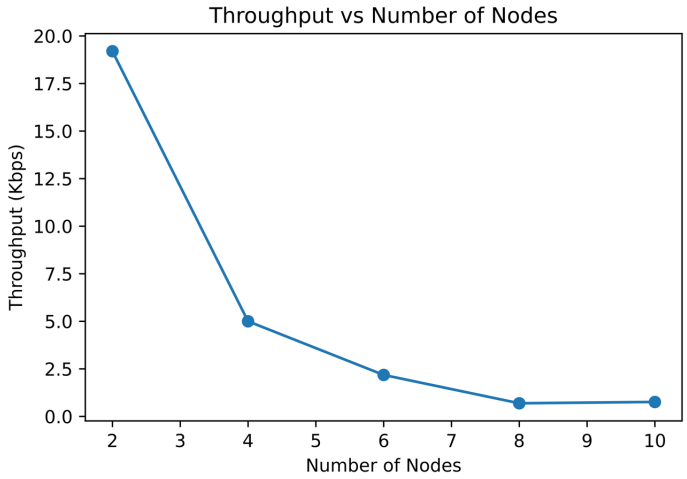


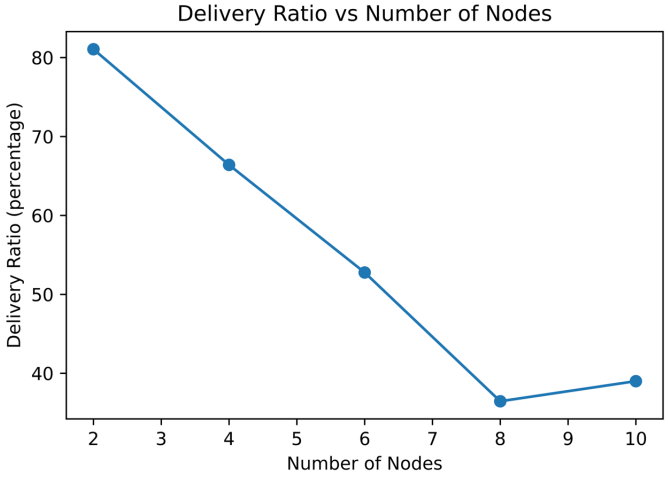


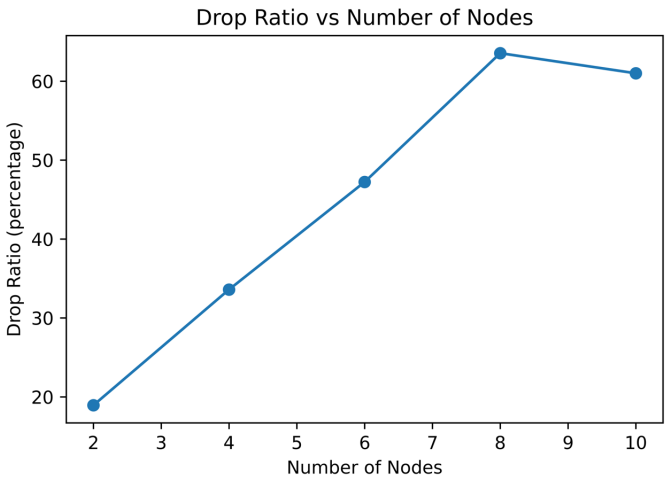
2.2.2 Varying Number of Nodes

In WPAN increasing the number of nodes increases the end to end delay since a greater number of nodes means that a greater proportion of the wireless channel is being used for routing related traffic. This same factor can be used to explain the decrease in throughput, increase in drop ratio and the decrease in delivery ratio when the number of nodes increase.



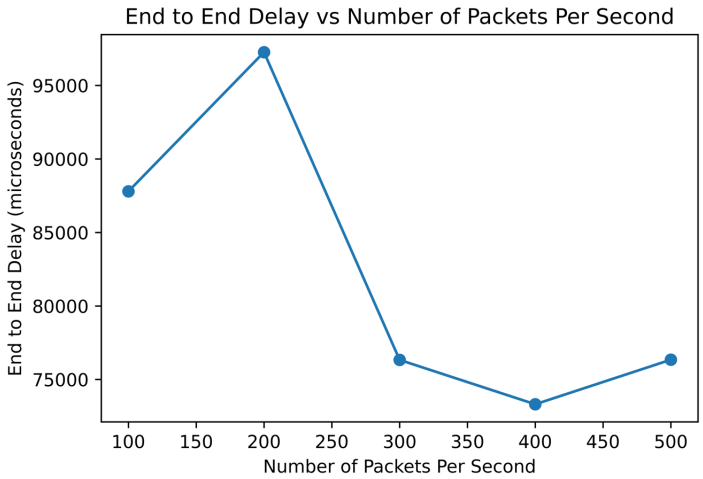

Throughput vs Number of Nodes
Throughput (Kbps)
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
2
3
4
5
6
7
8
9
10
Number of Nodes

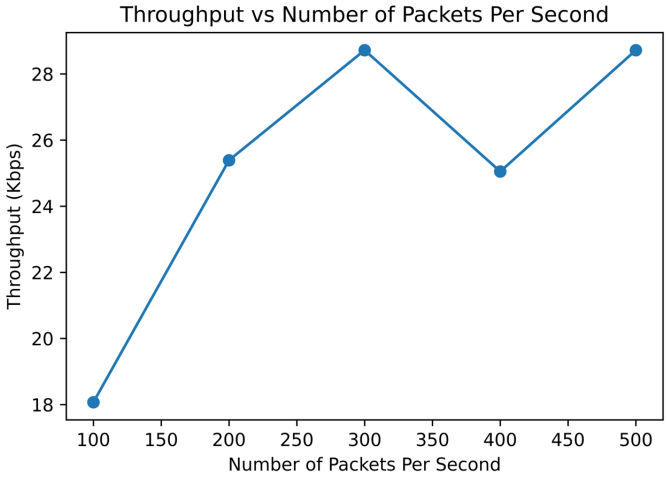

Delivery Ratio vs Number of Nodes
Delivery Ratio (percentage)
80
70
60
50
40
2
3
4
5
6
7
8
9
10
Number of Nodes

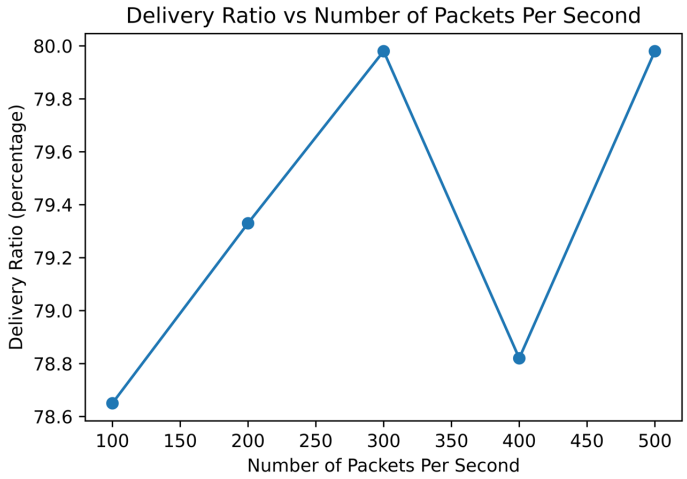

Drop Ratio vs Number of Nodes
Drop Ratio (percentage)
60
50
40
30
20
2
3
4
5
6
7
8
9
10
Number of Nodes

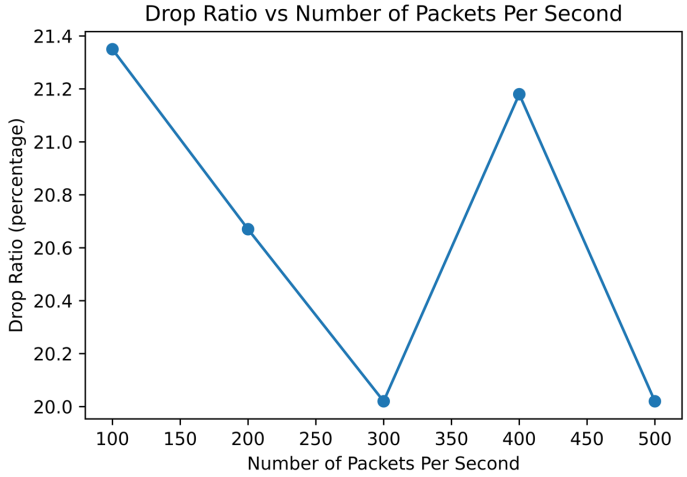
2.2.3 Varying Number of Packets Per Second

Increasing the number of packets per second causes the throughput to increase significantly, and the end to end delay to decrease significantly. The changes in delivery and drop ratio are insignificant in terms of the size of the percentage change in value.









2.2.4 Varying Maximum Transmission Range

To test the effect of transmission range on the network metrics I have used a topology where the 4 WPAN static nodes are placed a four different corners of a rectangle. Nodes N2, N3 and N4 are used the sources of 3 flows.

When the maximum transmission range is 1, none of the nodes N2, N3 and N4 can communicate with node N1, so none of the flows can successfully transmit any packet to the sink node. When the maximum transmission range is 2, node N2 can successfully send packets to the sink node. When the maximum transmission range is 3, nodes N2 and N3 can send packets to the sink node. When the maximum transmission range is 4 or more, nodes all of nodes N2, N3 and N4 can send packets to the sink node. These ideas can partly explain the graphs obtained below. However, another key idea is required. As the increase in maximum transmission range increases the number of flows that can send packets to the sink, it also increases the congestion in the network. Hence, we observe that the values of throughput and delivery ratio are highest when there is 1 flow within the transmission range.

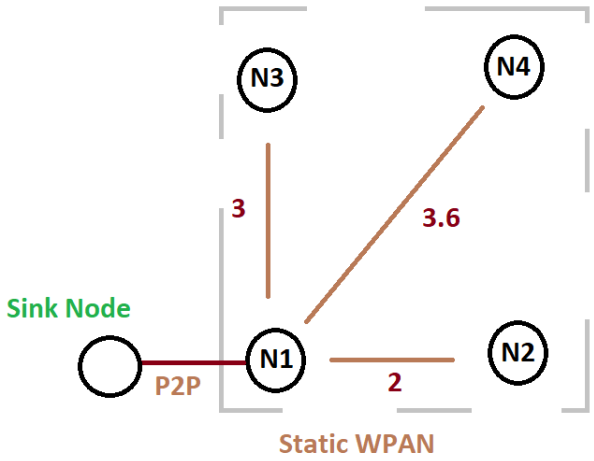
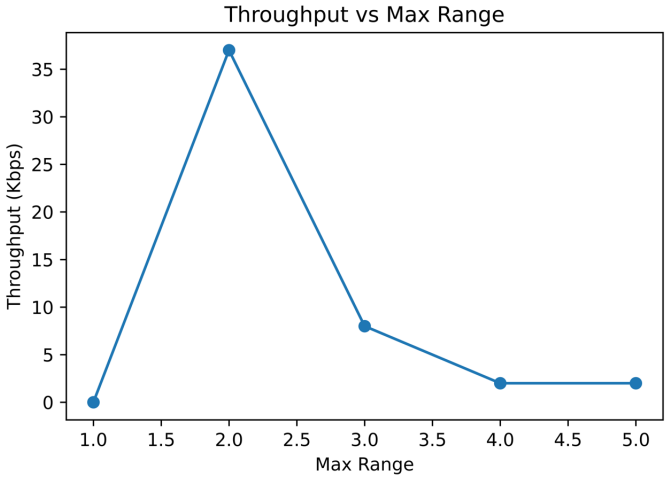
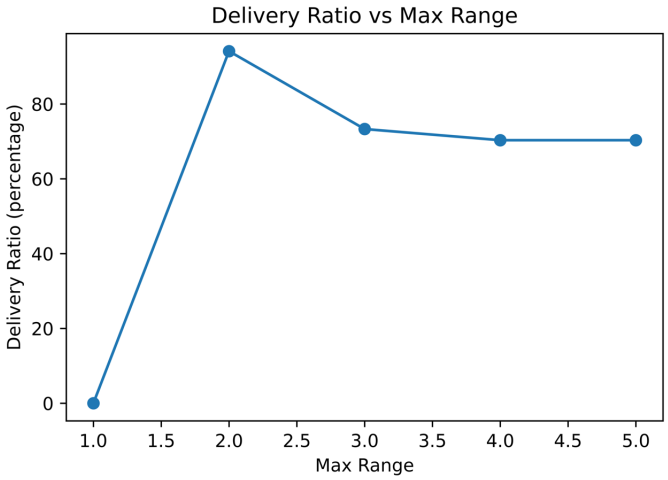
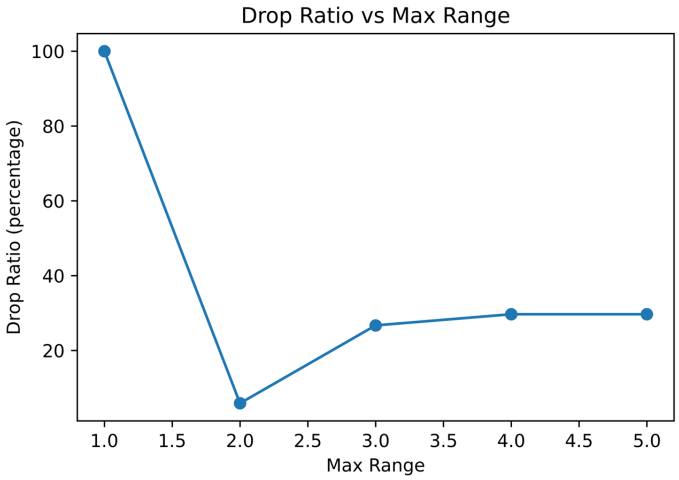


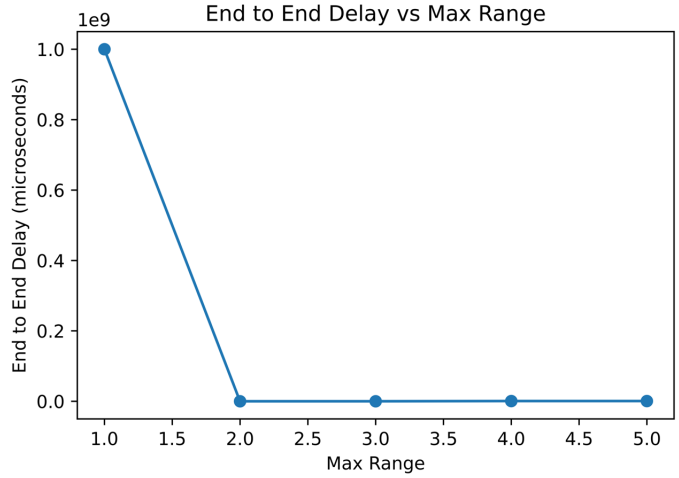
Figure 3: Topolgy Used to Test Transmission Range







When the maximum transmission range is 1, none of the flows can send packets to the sink. Since no packets can reach the sink, the value of end to end delay is infinite. Here, I am representing the infinity using a value which is much larger than the other values.



3 Task B: TCP-LR-NewReno

3.1 Reference

- **Paper Title:** TCP LR-Newreno congestion control for IEEE 802.15.4-based network
- **Authors:** Andri Sembiring, Maman Abdurohman, Fazmah Arif Yulianto
- **Date:** October 2017
- **Venue:** International Journal of Intelligent Engineering and Systems

3.2 Description of Algorithm

TCP LR-NewReno modifies the Congestion Avoidance phase and Fast Retransmit Phase of the well known TCP NewReno Algorithm.

The Congestion Avoidance phase is divided into parts. If the current congestion window is less than the value of congestion window during the last congestion event, then the congestion window is increasing non-linearly at a decreasing rate. But if the current congestion window is greater than the value congestion window during the last congestion event, the congestion window is increased linearly. The diagram below shows the pseudocode for this congestion avoidance algorithm.

Phase 3: CA (Additive Increase)

```
If (rec ACKs) {
   If (CW < CW_max) {  /* First sub phase */
        CW = CW + (CW_max-CW)/(α.CW_max)
} else { /* Second sub phase */
   CW = 1/(α.CW)
   Ideal_CW = CW_max
}}
```

ENHANCEMENT CODE

Figure 4: Congestion Avoidance Algorithm of TCP LR NewReno

For the fast retransmit phase TCP LR NewReno identifies a value called **Ideal-CW**. It is defined as follows: if the current congestion window crosses the value of congestion window during the last congestion event by more than a certain amount (a hyperparameter value), this value of congestion window during the last congestion event is the **Ideal-CW**. In the next congestion event, the value of congestion window falls to this Ideal-CW value instead of being halved.

3.3 Adding Algorithm to NS3 Source Files

- create: src/internet/models/tcp-lr-newreno.h
- create: src/internet/models/tcp-lr-newreno.cc
- add name of newfiles to src/internet/wscript

The diagrams below show the most important aspects of the code of the TCP LR-NewReno algorithm.

tcp-lr-newreno.h (Required Private Variables Declared)

```
class TcpLrNewReno : public TcpCongestionOps {
public:
  static TypeId GetTypeId (void);

  TcpLrNewReno ();
  TcpLrNewReno (const TcpLrNewReno& sock);
  ~TcpLrNewReno ();
  std::string GetName () const;
  virtual void IncreaseWindow (Ptr<TcpSocketState> tcb, uint32_t segmentsAcked);
  virtual uint32_t GetSsThresh (Ptr<const TcpSocketState> tcb, uint32_t bytesInFlight);
  virtual Ptr<TcpCongestionOps> Fork ();

protected:
  virtual uint32_t SlowStart (Ptr<TcpSocketState> tcb, uint32_t segmentsAcked);
  virtual void CongestionAvoidance (Ptr<TcpSocketState> tcb, uint32_t segmentsAcked);

private:
  uint32_t CW_MAX {0};
  uint32_t Ideal_CW {0};
  uint32_t Ideal_CW_SET {0};
  uint32_t  alpha {4};
  double  beta {0.5};
};
```

tcp-lr-newreno.cc (Congestion Avoidance Changed)

```
void TcpLrNewReno::CongestionAvoidance (Ptr<TcpSocketState> tcb, uint32_t segmentsAcked) {
  NS_LOG_FUNCTION (this << tcb << segmentsAcked);
  if (segmentsAcked > 0){
    if( tcb->m_cWnd < CW_MAX ) {
      double num = static_cast<double> ( CW_MAX-tcb->m_cWnd.Get() );
      double den = static_cast<double> ( alpha*tcb->m_cWnd.Get() );
      double f = static_cast<double> (tcb->m_segmentSize);
      double adder = f*(num/den);
      adder = std::max (1.0, adder);
      tcb->m_cWnd += static_cast<uint32_t> (adder);
    }
    else {
      double adder = static_cast<double> (tcb->m_segmentSize*tcb->m_segmentSize)/(alpha*tcb->m_cWnd.Get());
      adder = std::max (1.0, adder);
      tcb->m_cWnd += static_cast<uint32_t> (adder);

      if( (tcb->m_cWnd - CW_MAX) > (beta*CW_MAX-tcb->m_ssThresh) ) {
        Ideal_CW = CW_MAX;
        Ideal_CW_SET = 1;
      }
    }
  }
  NS_LOG_DEBUG ("At end of CongestionAvoidance(), m_cWnd: " << tcb->m_cWnd);
}
```

tcp-lr-newreno.cc (Fast Retransmit Changed)

```
uint32_t
TcpLrNewReno::GetSsThresh (Ptr<const TcpSocketState> state,
                           uint32_t bytesInFlight)
{
  NS_LOG_FUNCTION (this << state << bytesInFlight);

  CW_MAX = state->m_cWnd;
  uint32_t ssThresh = std::max<uint32_t> (2 * state->m_segmentSize, state->m_cWnd / 2

  if( Ideal_CW_SET ) {
    ssThresh = std::max<uint32_t> (ssThresh, Ideal_CW);
    Ideal_CW_SET = 0;
  }

  return ssThresh;
}
```

3.4 Simulation Parameters

I have tried to replicate the experimental scenario as described in the paper as much as possible. Concretely, I have ensured the follow parameters are the same as mentioned in the paper:

- Queue type
- Queue size
- Packet size
- Topology type
- Range in which number of nodes are varied
- Range in which error rate is varied

However, I failed to ensure similarity with the paper in the follow issues:

- Used NS3 instead of NS2 - constraint of this project
- Used Mesh routing instead of DSDV routing (DSDV routing not available for IPv6 in NS3
- The exact spatial orientation in which the nodes are placed - not specified in the paper.

3.5 Topology

The WPAN topology has a variable number of wireless static WPAN nodes placed along a line. One of these nodes is connected via point to point connection to a sink node. The sink node receives the packets sent by the WPAN nodes. And as mentioned in the paper, the number of WPAN nodes sending packets to the sink is either 1 or 4.

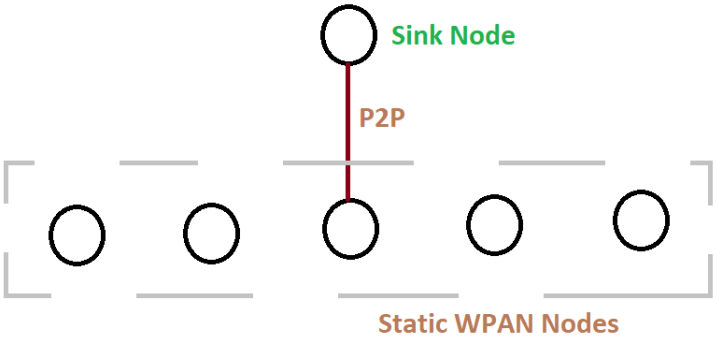
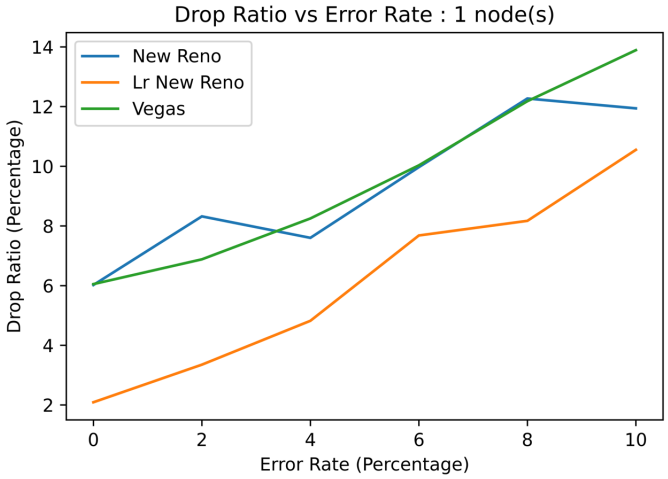


Figure 5: Topology of the WPAN network

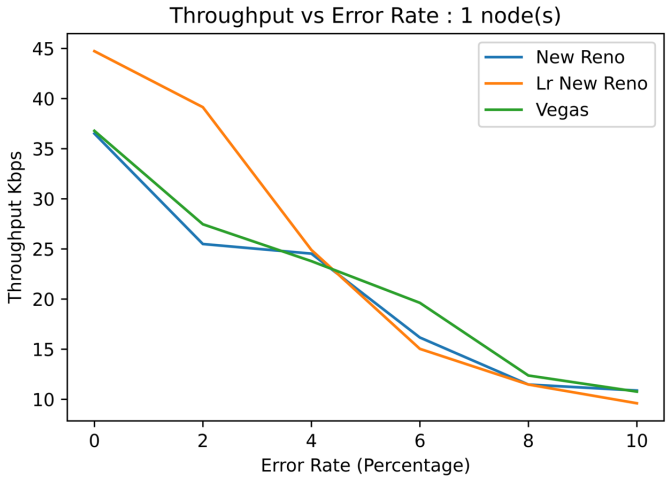
3.6 Results

The paper claims that TCP-LR-NewReno outperforms TCP-NewReno and TCP-Vegas for different values of error rate in terms of throughput, drop ratio and energy consumption. Meanwhile, the requirement of this project was to demonstrate how our chosen algorithm performs for **2 particular metrics**. Hence, in my experiments I have compared TCP Lr-NewReno against TCP-NewReno and TCP-Vegas for the metrics of throughput and drop ratio.

3.6.1 First Scenario: 1 flow

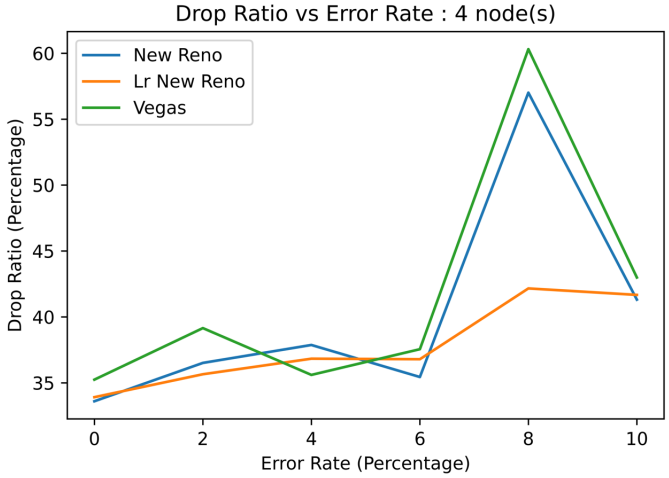


From the graph above we can see that my experiment result fully supports the claim of the paper in this particular scenario with 1 flow - TCP LR-NewReno has lower drop ratio than TCP NewReno and TCP Vegas for any error-rate.

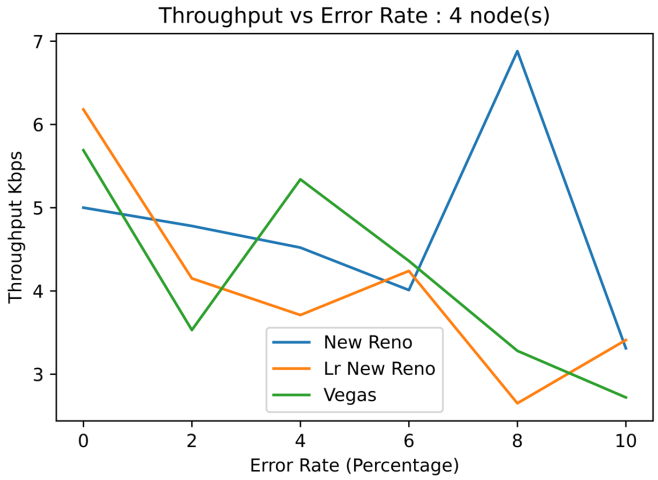


From the graph above we can see that my experiment result almost fully supports the claim of the paper in this particular scenario with 1 flow - the throughput for TCP LR-NewReno is higher than TCP NewReno and TCP Vegas upto an error rate of around 5%. Above 5% error-rate TCP LR-NewReno has similar throughput to TCP NewReno and TCP Vegas.

3.6.2 First Scenario: 4 flows



From the graph above we can see that my experiment result somewhat supports the claim of the paper in this particular scenario with 4 flows - TCP LR-NewReno has lower or similar drop ratio compared to TCP NewReno and TCP Vegas for the different error-rates.



Unfortunately, in this scenario with 4 flows my experiment does not seem to support the claim of the paper - the throughput of TCP LR-NewReno is not better than that of TCP NewReno and TCP Vegas for the different error-rates.

3.7 Remarks

So my experiment results support the claims of the paper in the 3 of the 4 different simulation scenarios described in the paper. The reason why my experimental result does not support the claim in the 4th scenario could be because I had to use a different routing algorithm than the one described in the paper (since NS3 does not support DSDV routing for IPv6). It might also be because I had not placed the static nodes in the exact spatial orientation as the authors of the paper (the exact spatial orientation is not mentioned in the paper). These two issues of routing algorithm and spatial orientation become more significant when the number of nodes is greater. Thus, it makes sense why my experimental results fully support the claims of the paper when the number of WPAN nodes sending packets to sink node is 1, but does not fully support when this number is 4.