CSE 322

Computer Networks Sessional

Report on ns-3 Project

Asif Ajrof

Student ID: 1705092

Department of Computer Science and Engineering,

Bangladesh University of Engineering and Technology.

Contents

Task A **1**
- Wireless high-rate (static) 1
  - Network topology under simulation 1
  - Parameters under variation 1
  - Varying Number of Nodes 2
  - Varying Number of Flows 4
  - Varying Number of Packets per Second 7
  - Varying Coverage Area 9
- Wireless low-rate (static) 13
  - Network topology under simulation 13
  - Parameters under variation 13
  - Varying Number of Nodes 13
  - Varying Number of Flows 16
  - Varying Number of Packets per Second 18
  - Varying Coverage Area 21

Task B **24**
- Network topology under simulation 24
- Overview of the proposed algorithm 25
- Modifications in simulator to implement algorithm 27
- Results with graphs 27
- Summary 29

References **30**

Task A

Varying Parameters Without Modifications

Assigned networks according to ID (1705092)

- Wireless high-rate (static)
- Wireless low-rate (static)

Wireless high-rate (static)

Network topology under simulation

The simulation file is a modified version of the *third.cc* file, along with some other example files provided in the examples of ns-3. Here, the topology is a dumbbell network as shown in Figure 1. The central two nodes are the access point nodes for the respective stationary wifi nodes. The ap nodes are in point to point connection. A node from one side sends a packet to a node on the other side.

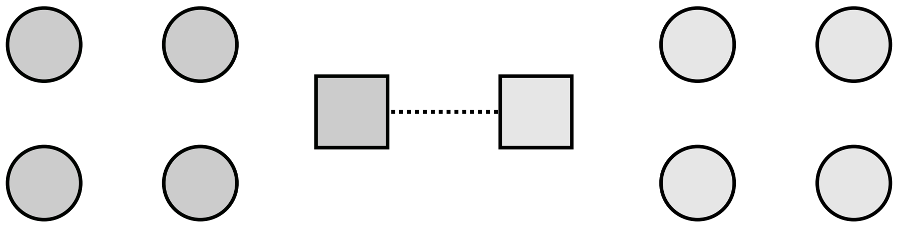

Figure 1: Task A - Wireless high-rate (static) network topology

Parameters under variation

- Number of nodes (20, 40, 60, 80, 100)
- Number of flows (10, 20, 30, 40, 50)
- Number of packets per second (64, 128, 256, 384, 512)
- Coverage area (1 * tx_range, 2 * tx_range, 3 * tx_range, 4 * tx_range, 5 * tx_range)

Varying Number of Nodes

Datarate = 5 Mbps, P2P Datarate (bottleneck) = 2 Mbps, Number of Flows = 10, Transmission range = 20 m, Simulation Time = 20 s

- **Results with graphs**
  - Network Throughput

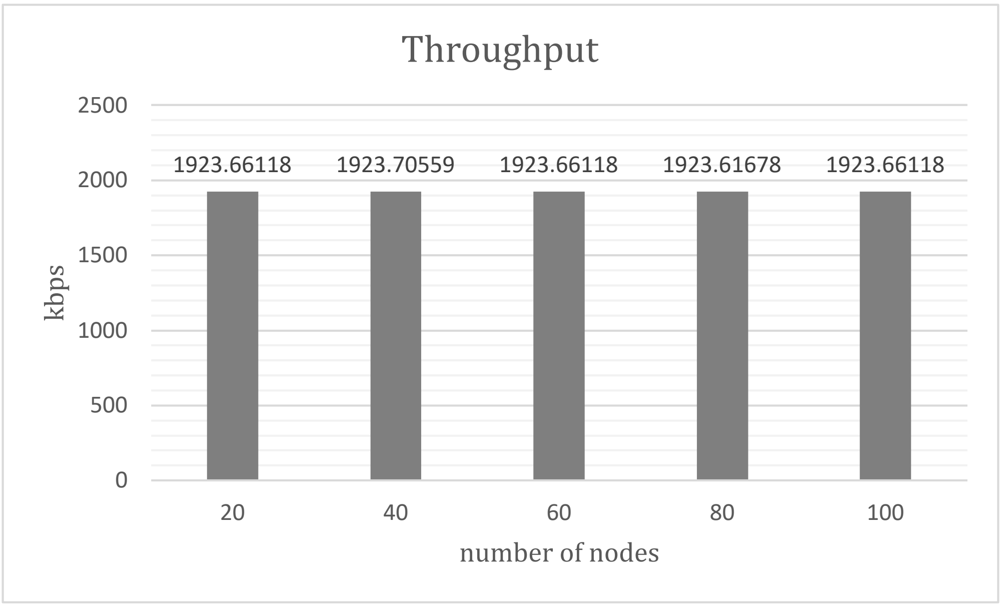

Figure 2: Throughput vs Number of nodes

  - End-to-end delay

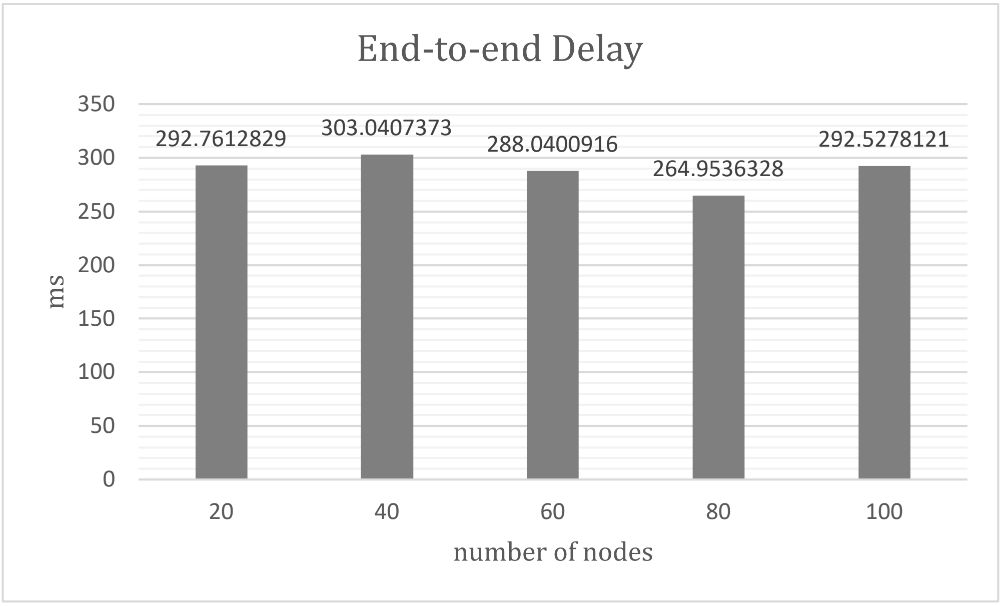

Figure 3: End-to-end delay vs Number of nodes

– Packet delivery ratio

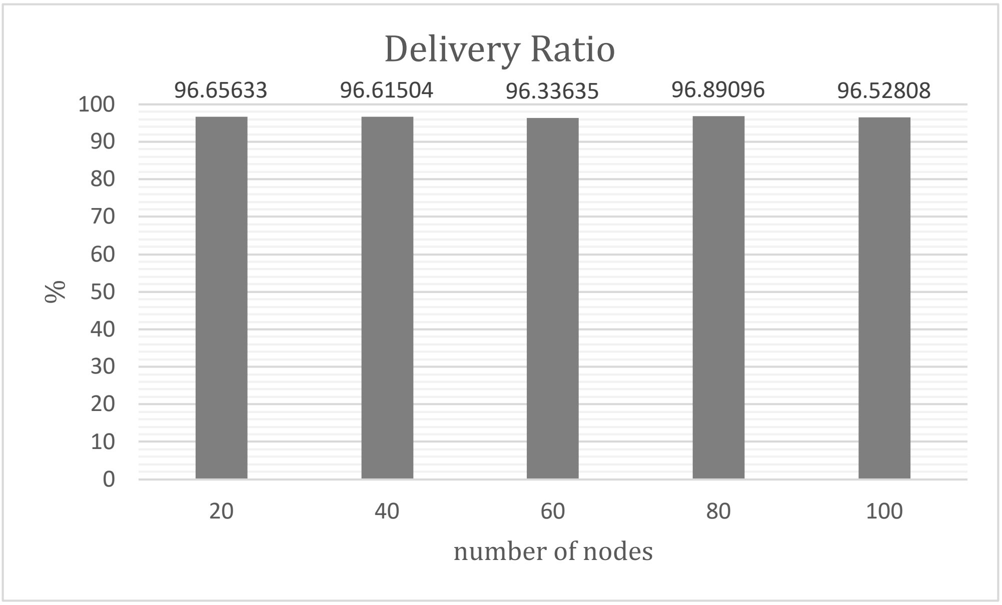

Figure 4: Delivery ratio vs Number of nodes

– Packet drop ratio

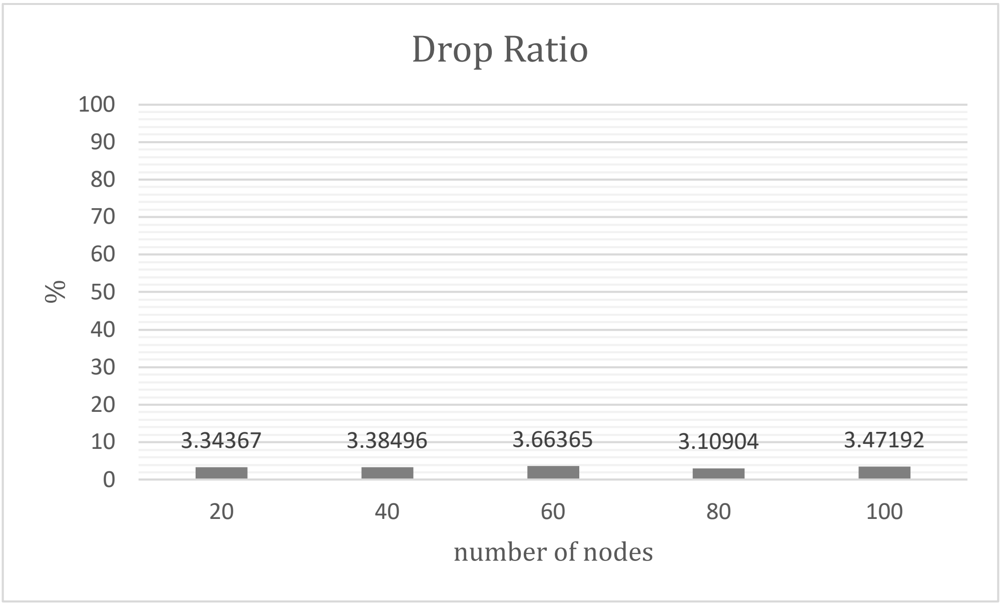

Figure 5: Drop ratio vs Number of nodes

- **Summary**
  Varying number of nodes while keeping the other parameters constant, there is no significant changes in the network statistics. As the extra nodes do not send or receive any flow they do not affect the statistics. So we get almost similar data for all the parameters varying only the number of nodes.

Varying Number of Flows

Datarate = 5 Mbps, P2P Datarate (bottleneck) = 2 Mbps, Number of Nodes = 20*2, Transmission range = 20 m, Simulation Time = 20 s

- **Results with graphs**
  - Network Throughput

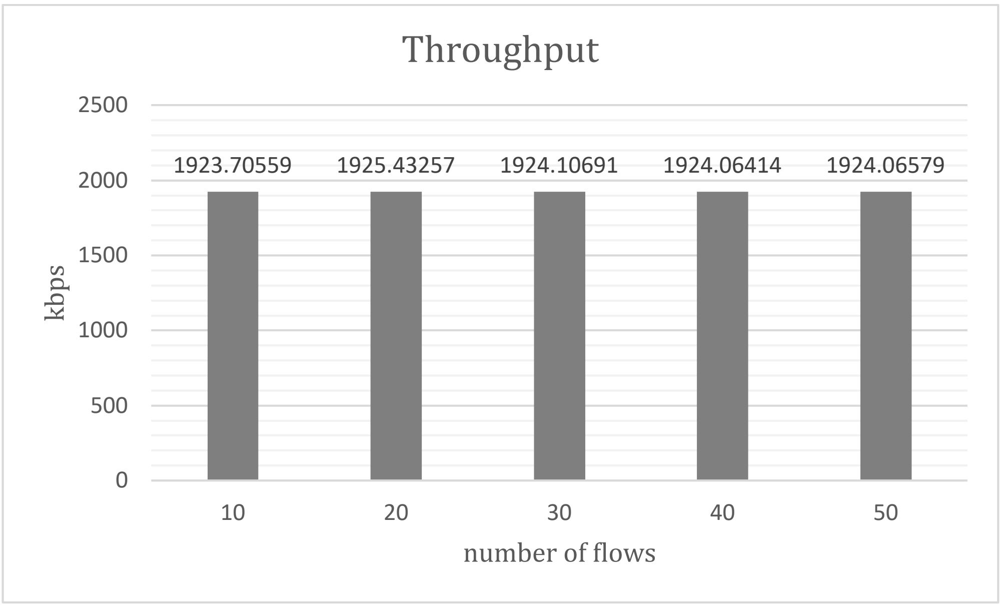

Figure 6: Throughput vs Number of flows

– End-to-end delay

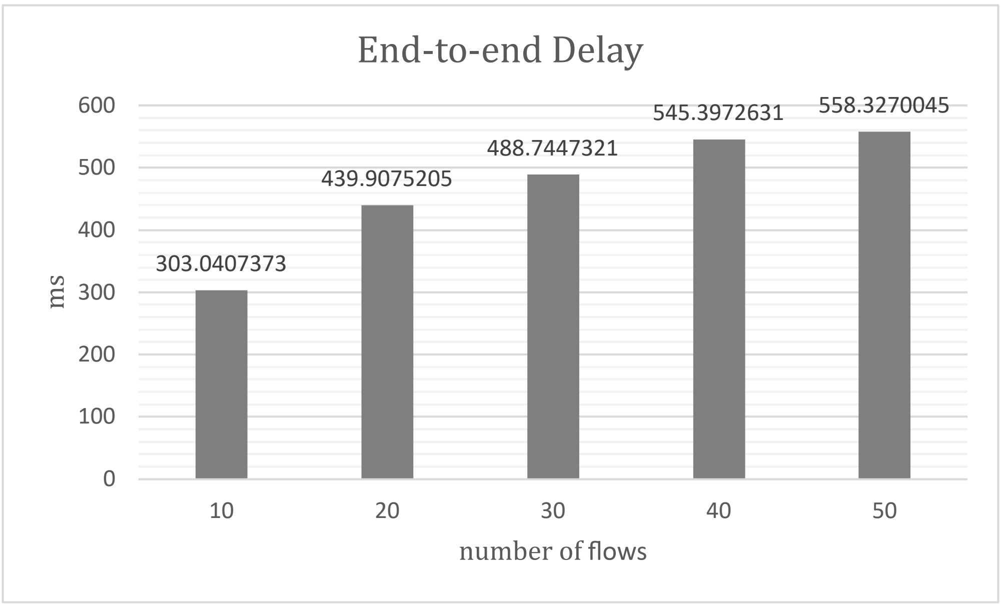

Figure 7: End-to-end delay vs Number of flows

– Packet delivery ratio

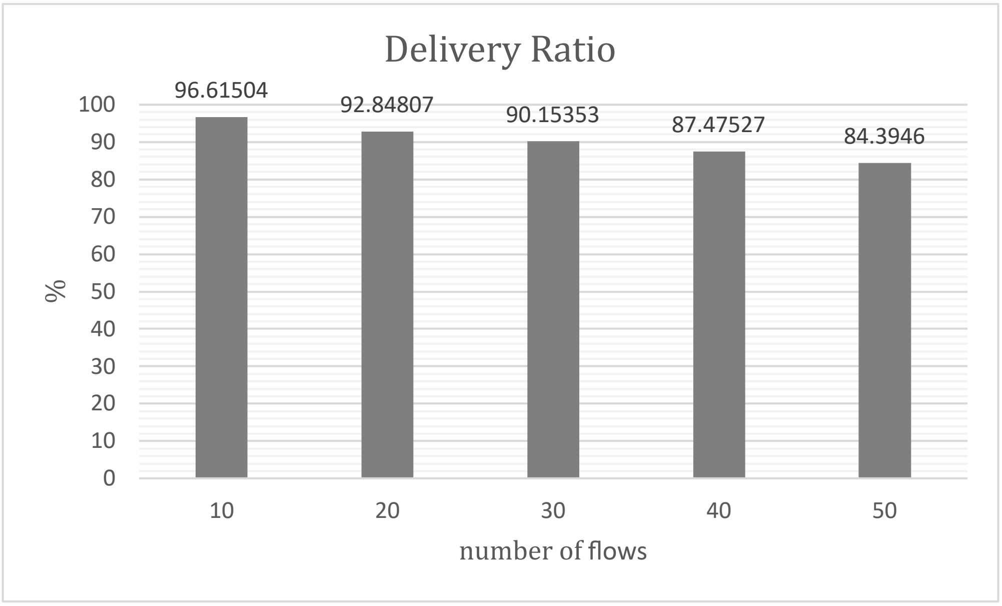

Figure 8: Delivery ratio vs Number of flows

– Packet drop ratio

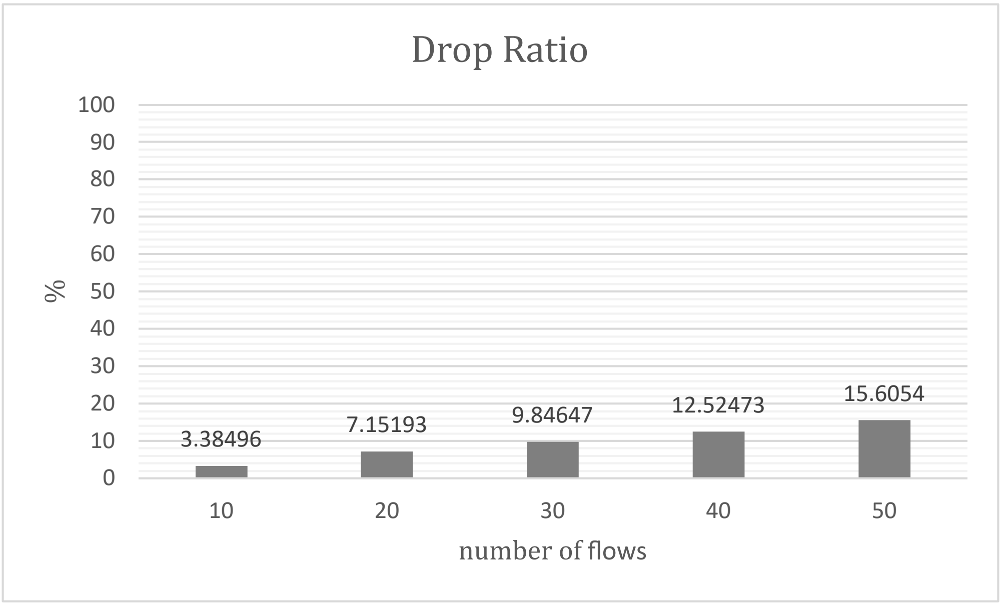

Figure 9: Drop ratio vs Number of flows

- **Summary**
  The network throughput does not increase as much with the increasing number of flows. This is because of the bottleneck p2p channel of the access point nodes. It caps the throughput of the network even if the flows are increased. And as more flows try to pass, the delay increases, delivery ratio decreases and drop

ratio increases.

Varying Number of Packets per Second

P2P Datarate (bottleneck) = 2 Mbps, Number of Nodes = 10*2, Number of Flows = 10, Transmission range = 20 m, Simulation Time = 20 s

- **Results with graphs**
  - Network Throughput

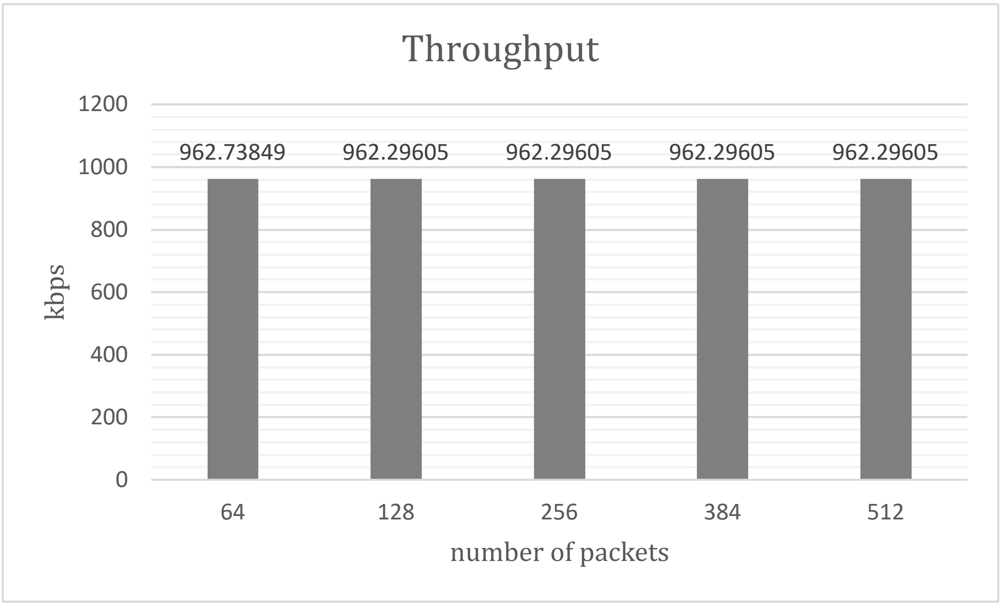

Figure 10: Throughput vs Number of packets/sec

  - End-to-end delay

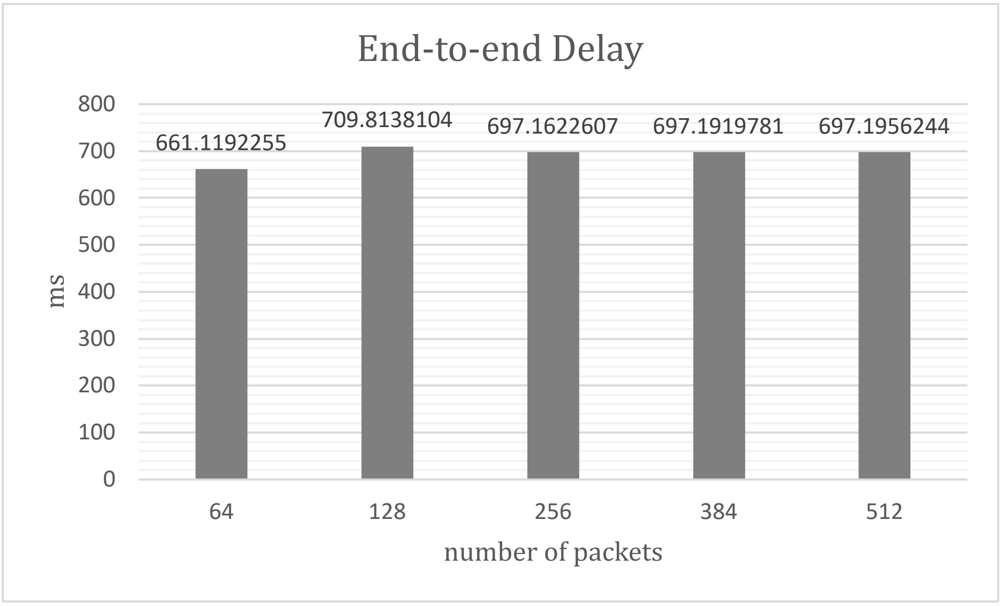

Figure 11: End-to-end delay vs Number of packets/sec

– Packet delivery ratio

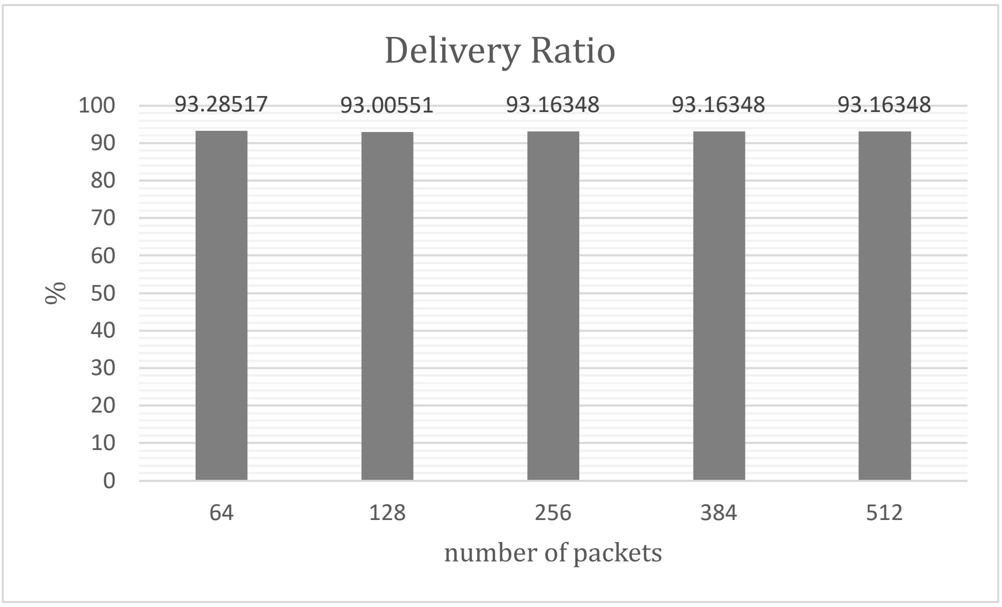

Figure 12: Delivery ratio vs Number of packets/sec

– Packet drop ratio

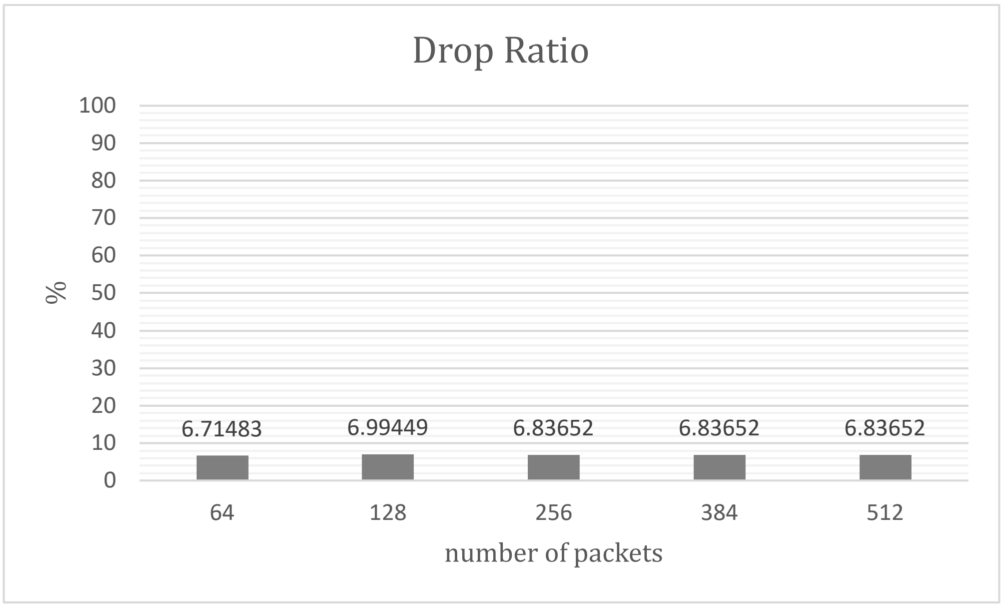

Figure 13: Drop ratio vs Number of packets/sec

- **Summary**
  Number of packets are varied as such the datarate becomes 0.5, 1, 2, 3, 4 Mbps respectively. And as the bottleneck channel caps the datarate with its 2 Mbps rate, the effect of changing packets per second does not hold much significance in the network statistics. So we get same statistics for datarate over 2 Mbps and before that a slight change.

Varying Coverage Area

Datarate = 5 Mbps, P2P Datarate (bottleneck) = 2 Mbps, Number of Nodes = 10*2, Number of Flows = 10, Transmission range = 10 m, Simulation Time = 20 s

- **Results with graphs**
  - Network Throughput

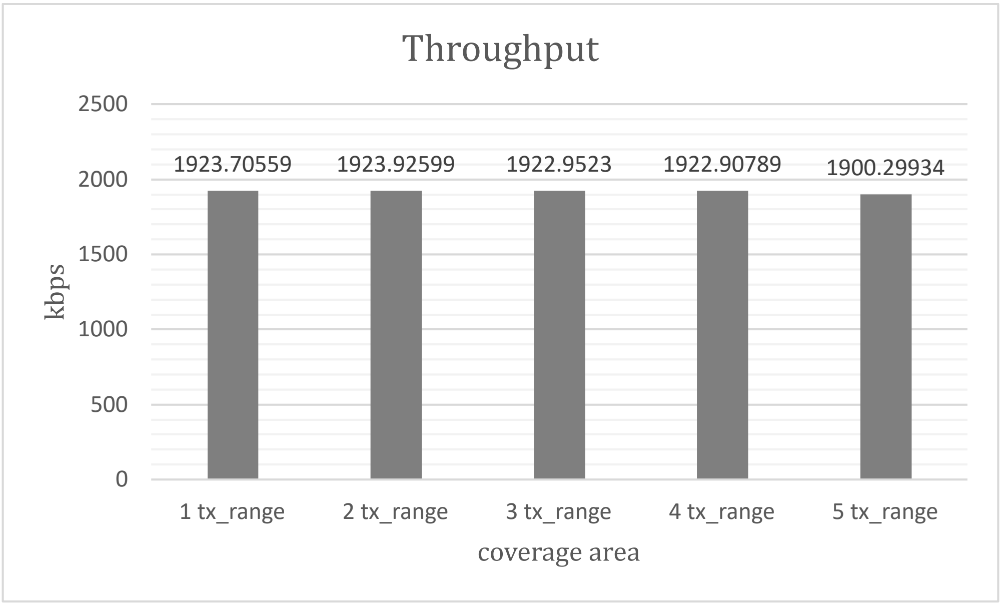

Figure 14: Throughput vs Coverage area

– End-to-end delay

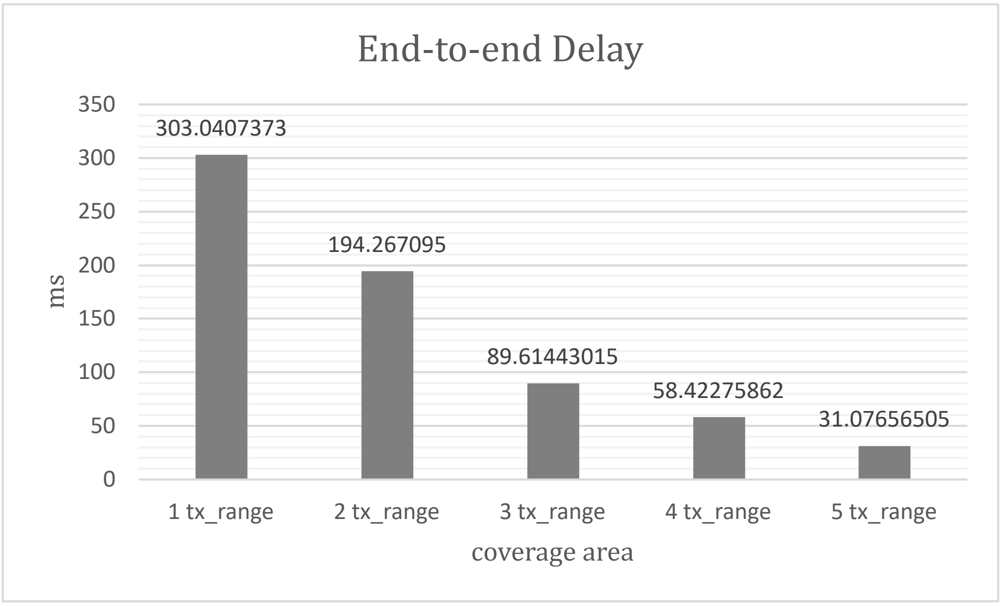

Figure 15: End-to-end delay vs Coverage area

– Packet delivery ratio

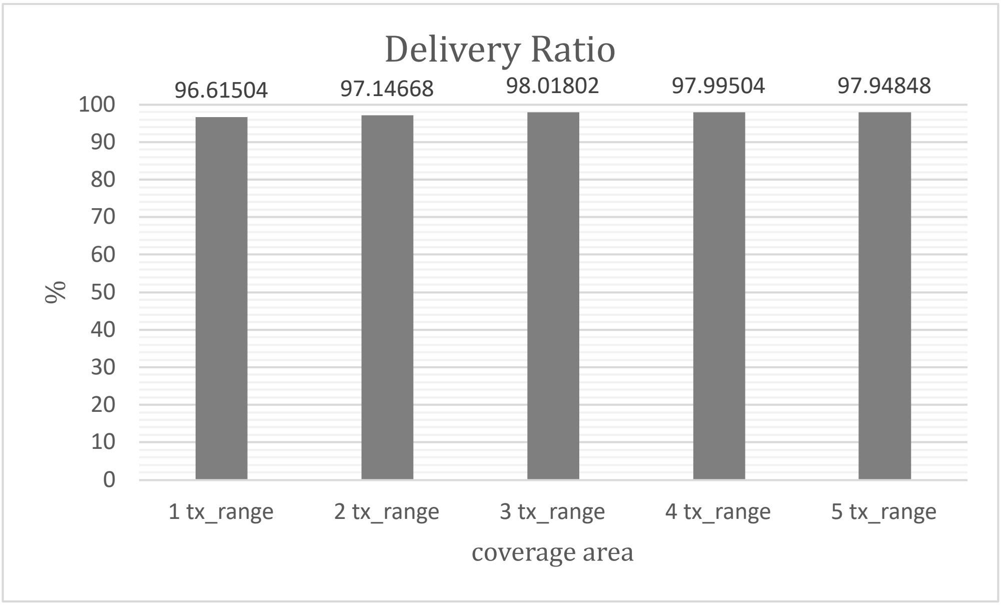

Figure 16: Delivery ratio vs Coverage area

– Packet drop ratio

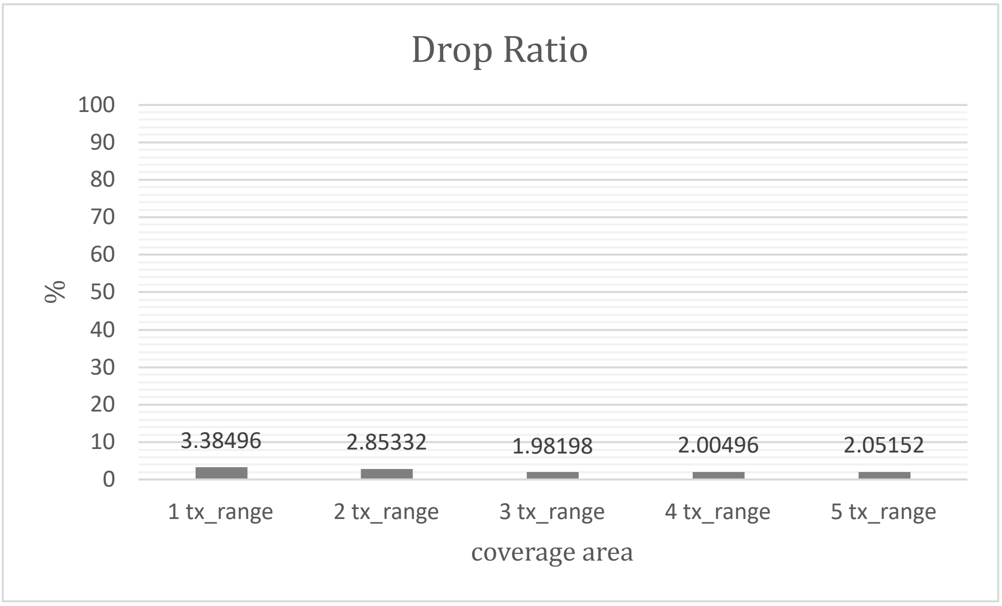

Figure 17: Drop ratio vs Coverage area

– Number of flows reached

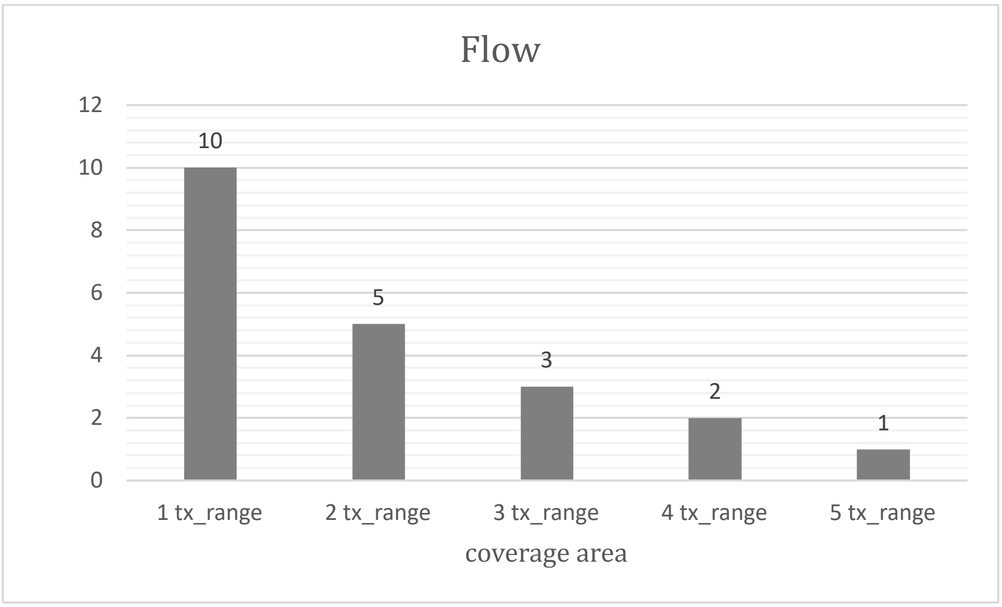

Figure 18: Flow vs Coverage area

- **Summary**
  To explain the statistics changing the coverage area, we need to look at Figure 18. Even though every time 10 flows were sent from 10 nodes, not all of those nodes were inside the transmission range and hence those flows were dropped. And the remaining flows actually affected the network statistics. That is why the throughput remains almost same. And delivery and drop ratio also remain same. But the end to end delay decreases as there are less nodes establishing connection and sending all the packets.
  In this simulated network if the coverage area is further increased, let us say 6 times of the transmission range then all of the nodes go out of the range and all the flows are dropped.

Wireless low-rate (static)

Network topology under simulation

The simulation file is a modified version of the *example-ping-lr-wpan-mesh-under .cc* file provided in the examples of ns-3. Here, the topology is a mesh wireless pan (personal area network) as shown in Figure 19. One of the nodes is the receiver and all other nodes are sender.

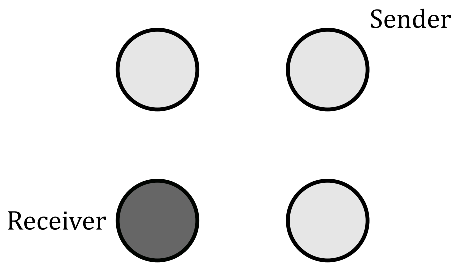

Figure 19: Task A - Wireless low-rate (static) network topology

Parameters under variation

- Number of nodes (5, 10, 15, 20, 25)
- Number of flows (5, 10, 15, 20, 25)
- Number of packets per second (100, 200, 300, 400, 500)
- Coverage area (1 * tx_range, 2 * tx_range, 3 * tx_range, 4 * tx_range, 5 * tx_range)

Varying Number of Nodes

Datarate = 500 kbps, Number of Flows = 5, Transmission range = 10 m, Simulation Time = 20 s

- **Results with graphs**
  - Network Throughput

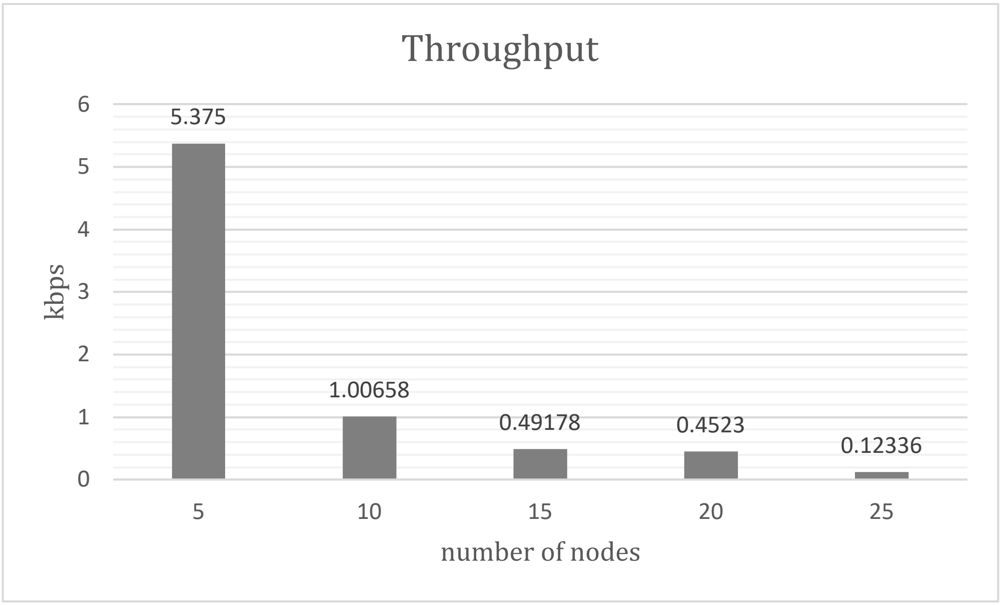

Figure 20: Throughput vs Number of nodes

– End-to-end delay

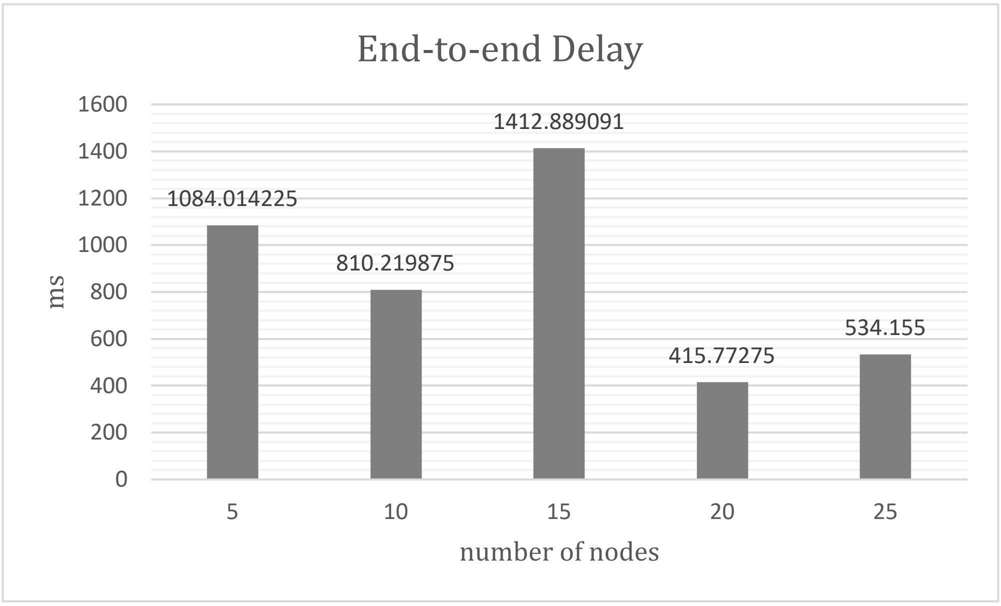

Figure 21: End-to-end delay vs Number of nodes

– Packet delivery ratio

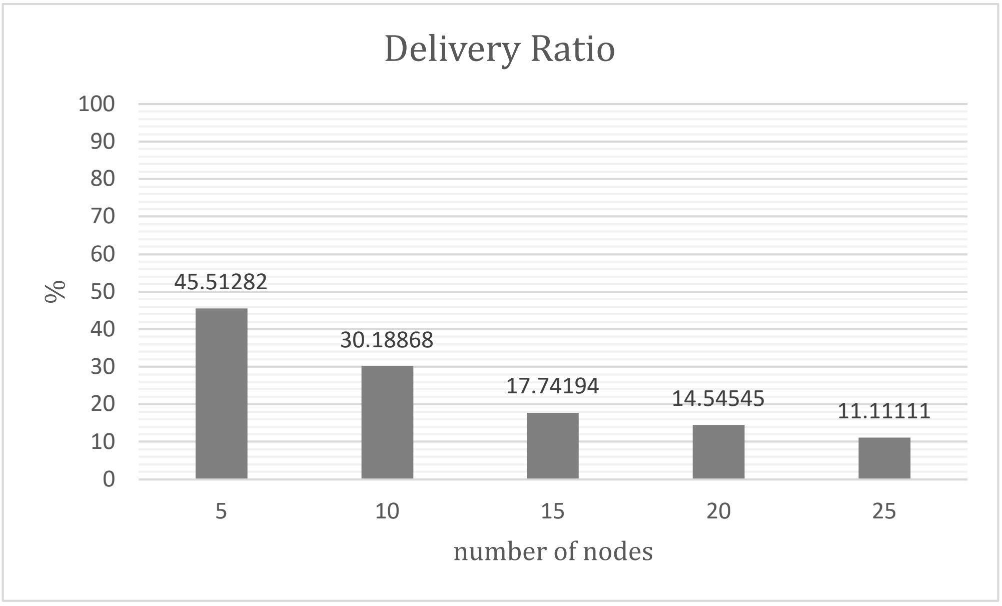

Figure 22: Delivery ratio vs Number of nodes

– Packet drop ratio

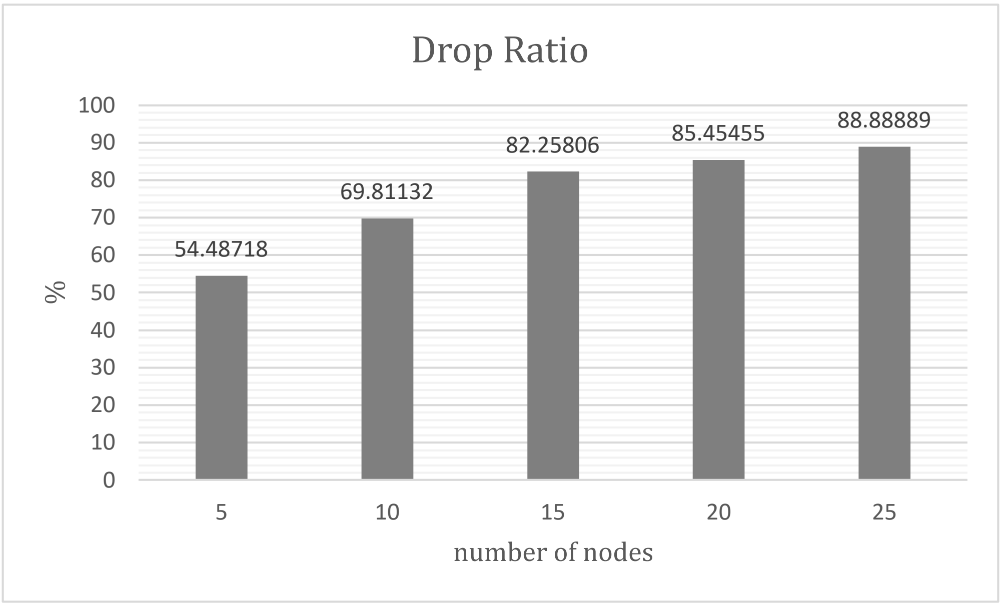

Figure 23: Drop ratio vs Number of nodes

- **Summary**
  The low rate wireless personal area network works better with small number of nodes situated in an optimal distance. As the number of nodes increases the low rate wpan throttles. The delivery ratio drops by huge margin and the throughput also decreases significantly.

Varying Number of Flows

Datarate = 500 kbps, Number of Nodes = 5, Transmission range = 10 m, Simulation Time = 20 s

- **Results with graphs**
  - Network Throughput

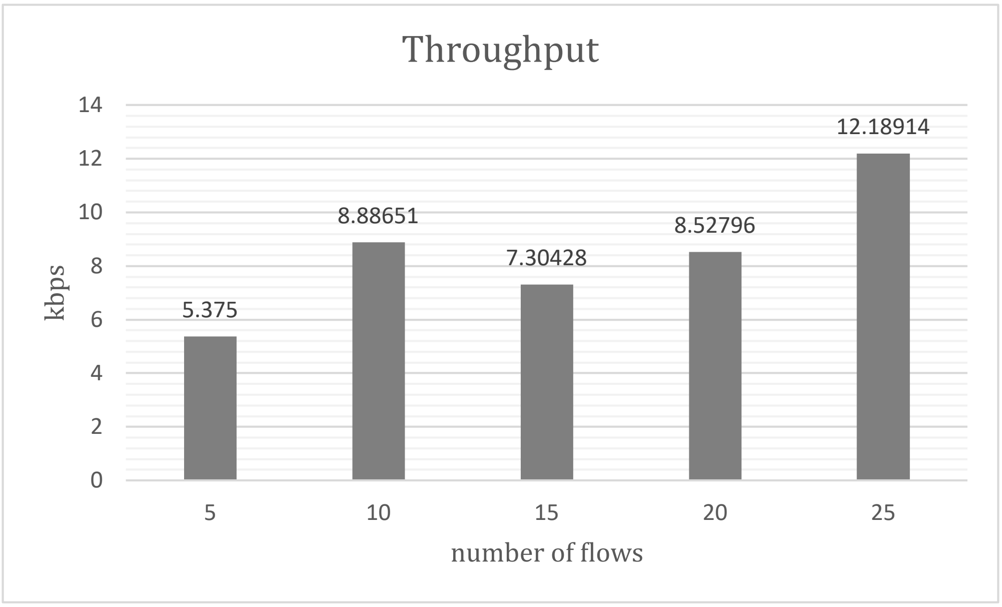

Figure 24: Throughput vs Number of flows

  - End-to-end delay

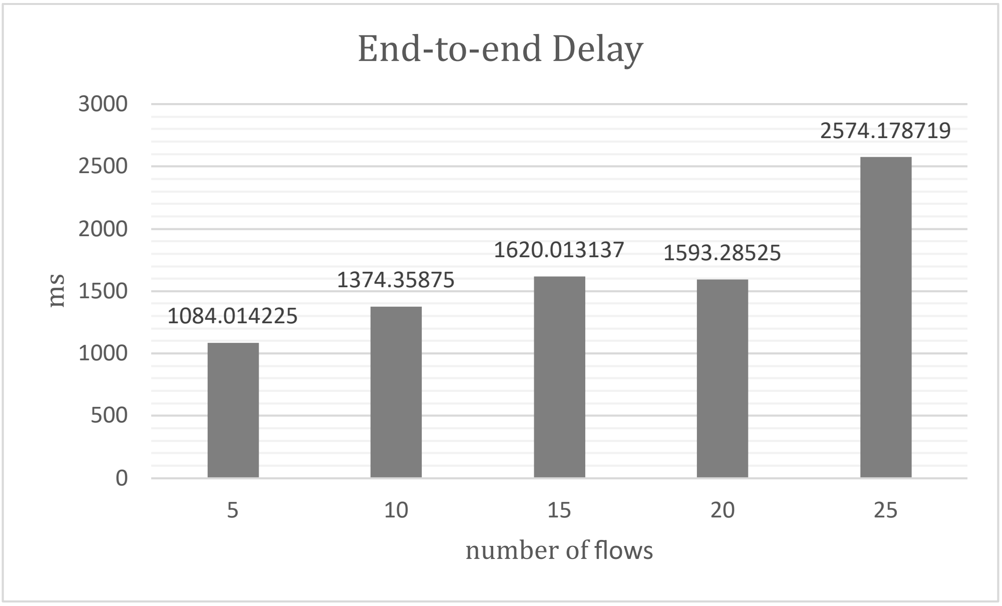

Figure 25: End-to-end delay vs Number of flows

– Packet delivery ratio

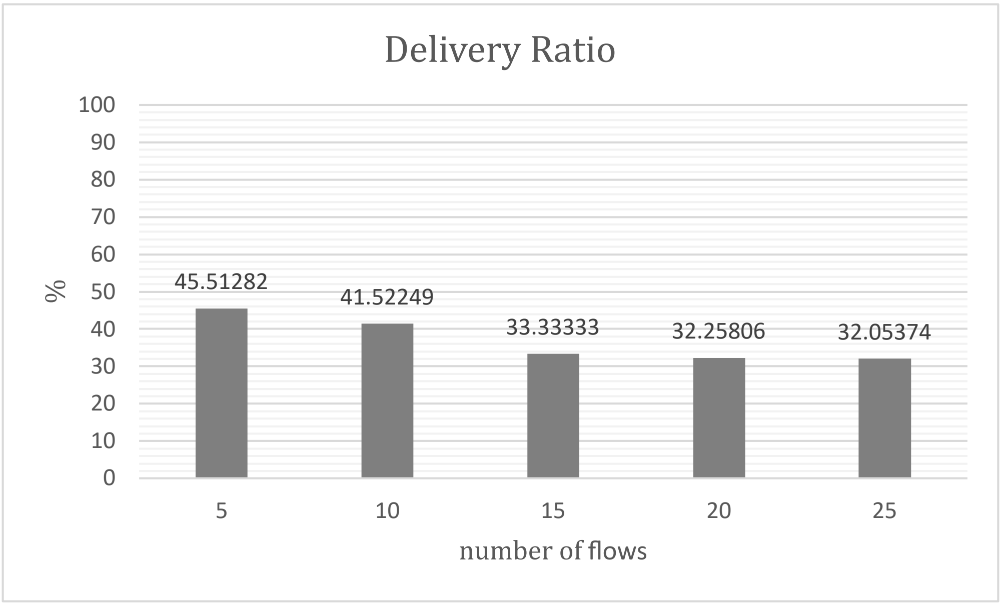

Figure 26: Delivery ratio vs Number of flows

– Packet drop ratio

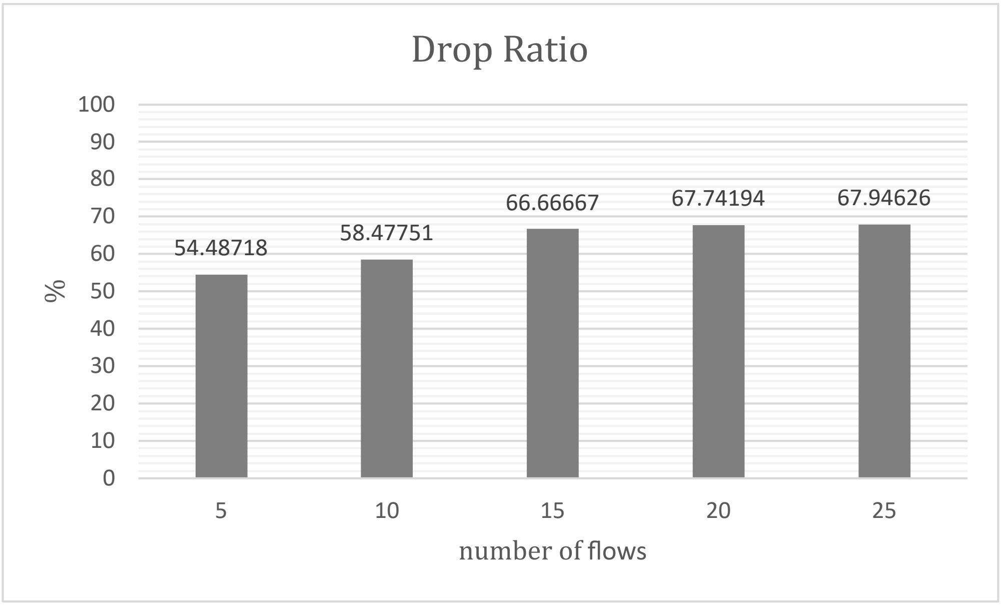

Figure 27: Drop ratio vs Number of flows

- **Summary**
  Increasing number of flows increases the throughput. As the number of flows increases the end-to-end delay also increases for every new connection establishment. And the delivery rate decreases, and drop ratio increases for the dropped packets in the new flows.

Varying Number of Packets per Second

Number of Nodes = 5, Number of Flows = 5, Transmission range = 10 m, Simulation Time = 20 s

- **Results with graphs**
  - Network Throughput

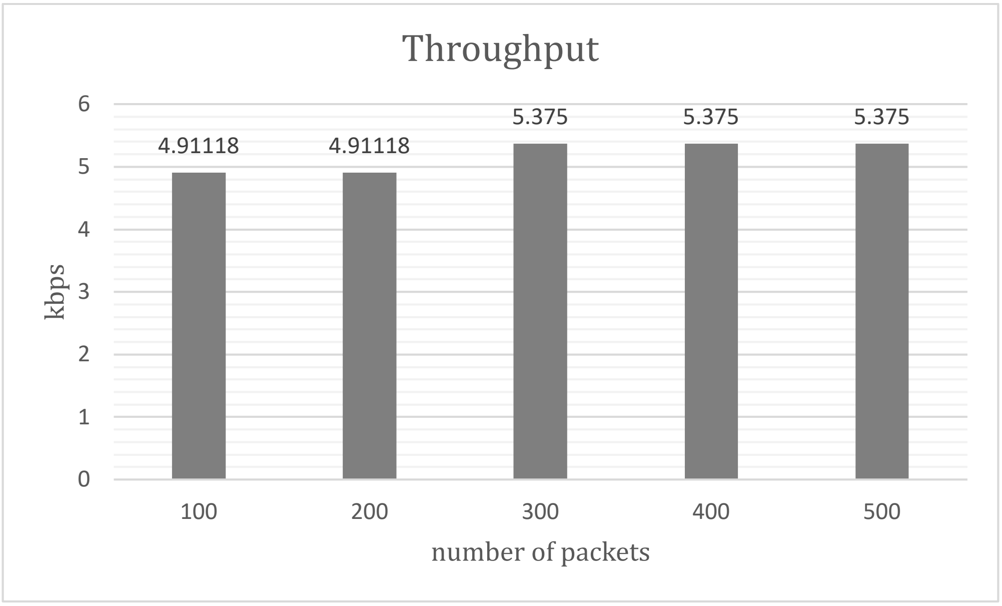

Figure 28: Throughput vs Number of packets/sec

– End-to-end delay

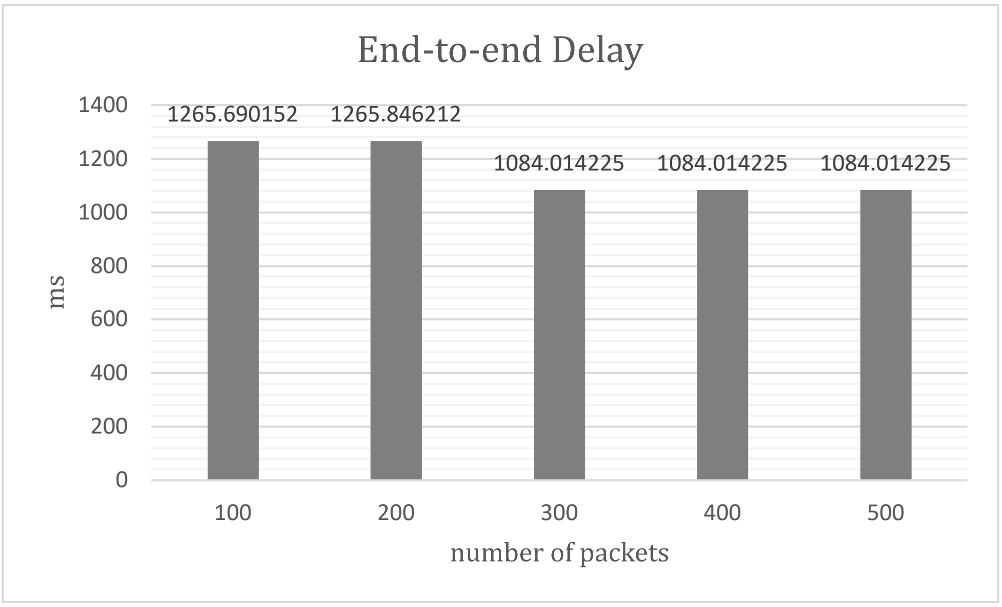

Figure 29: End-to-end delay vs Number of packets/sec

– Packet delivery ratio

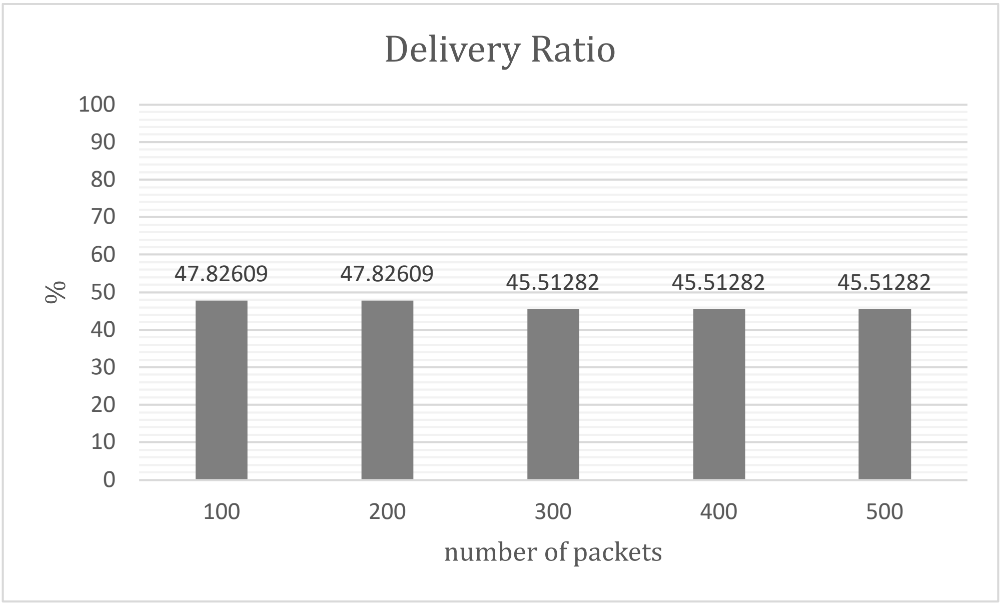

Figure 30: Delivery ratio vs Number of packets/sec

– Packet drop ratio

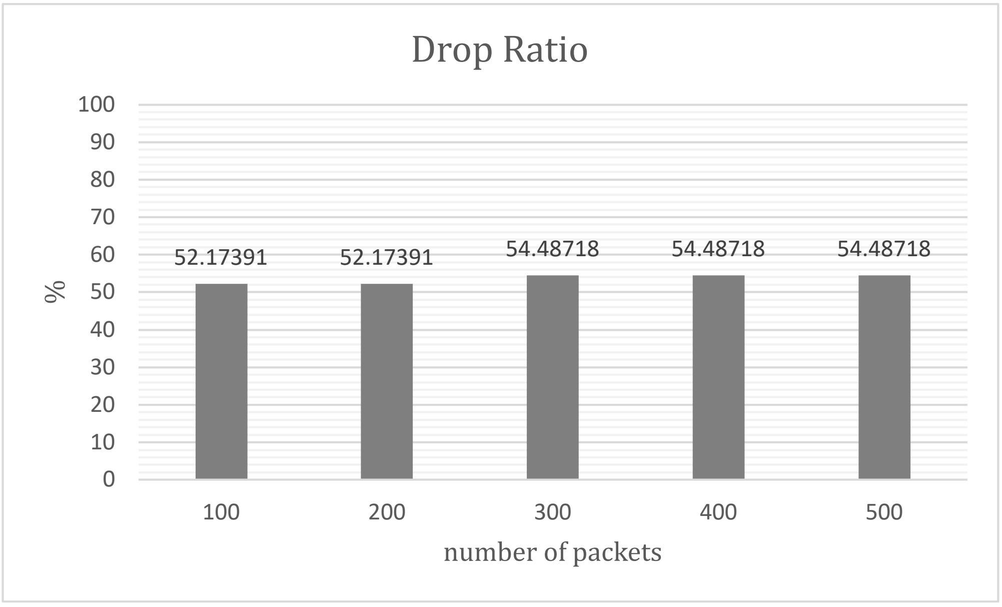

Figure 31: Drop ratio vs Number of packets/sec

- **Summary**

  As the datarate increases with increasing number of packets per second, the throughput increases. And end-to-end delay decreases. The delivery and drop ratio remains almost the same as the flow and node number and position remain the same.

Varying Coverage Area

Datarate = 500 kbps, Number of Nodes = 5, Number of Flows = 5, Transmission range = 10 m, Simulation Time = 20 s

- **Results with graphs**
  - Network Throughput

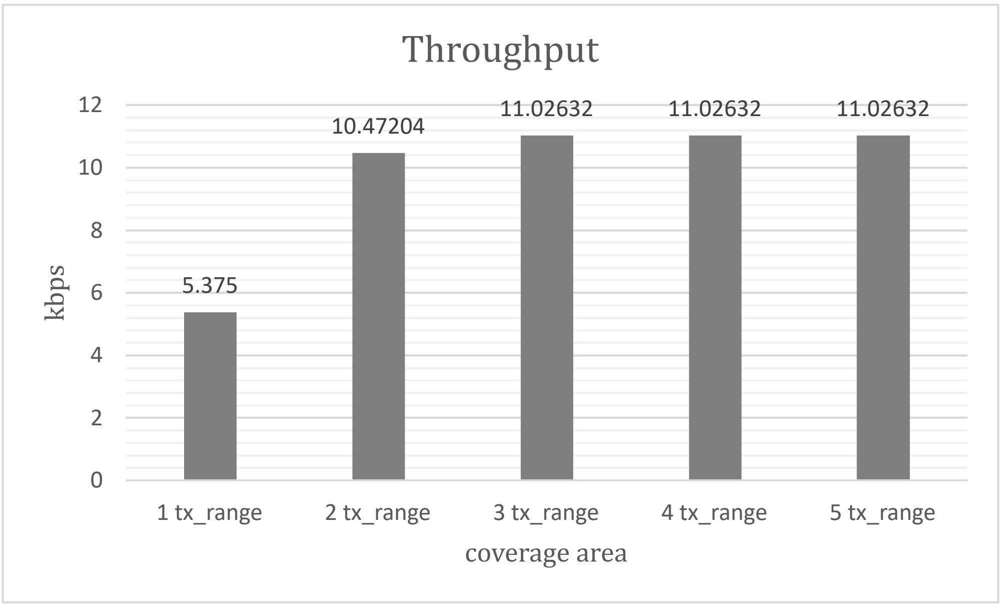

Figure 32: Throughput vs Coverage area

  - End-to-end delay

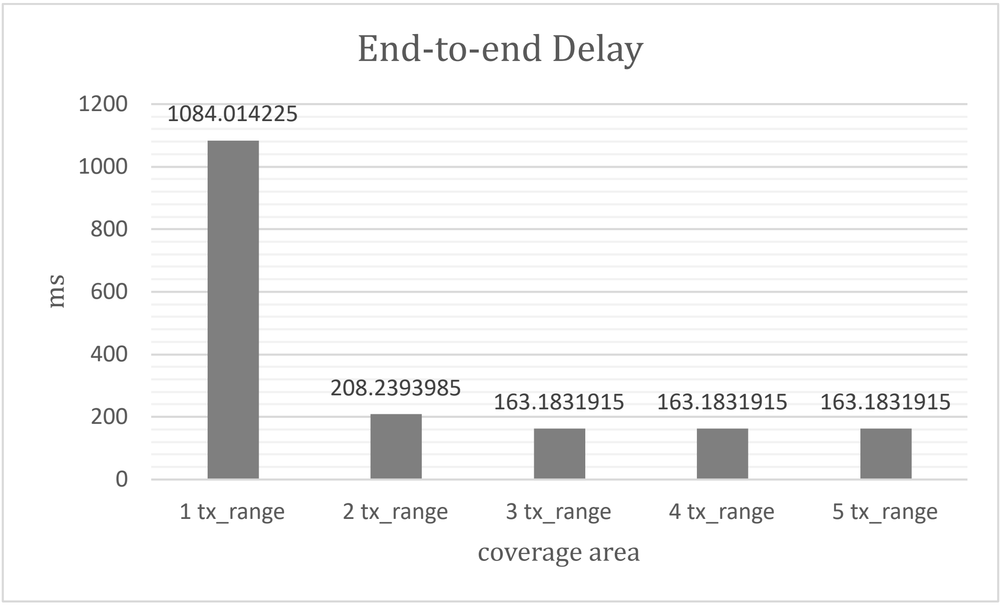

Figure 33: End-to-end delay vs Coverage area

– Packet delivery ratio

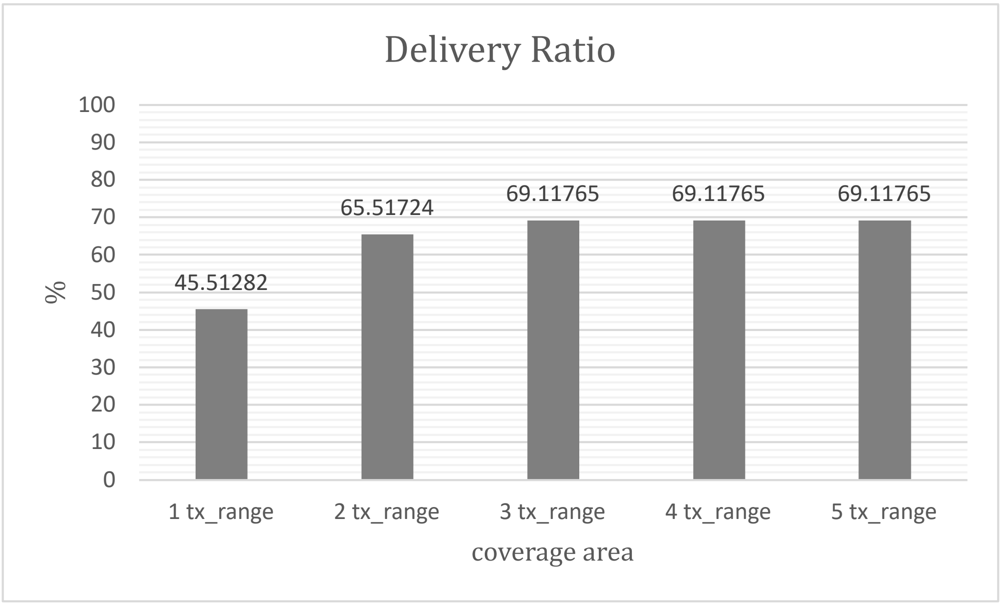

Figure 34: Delivery ratio vs Coverage area

– Packet drop ratio

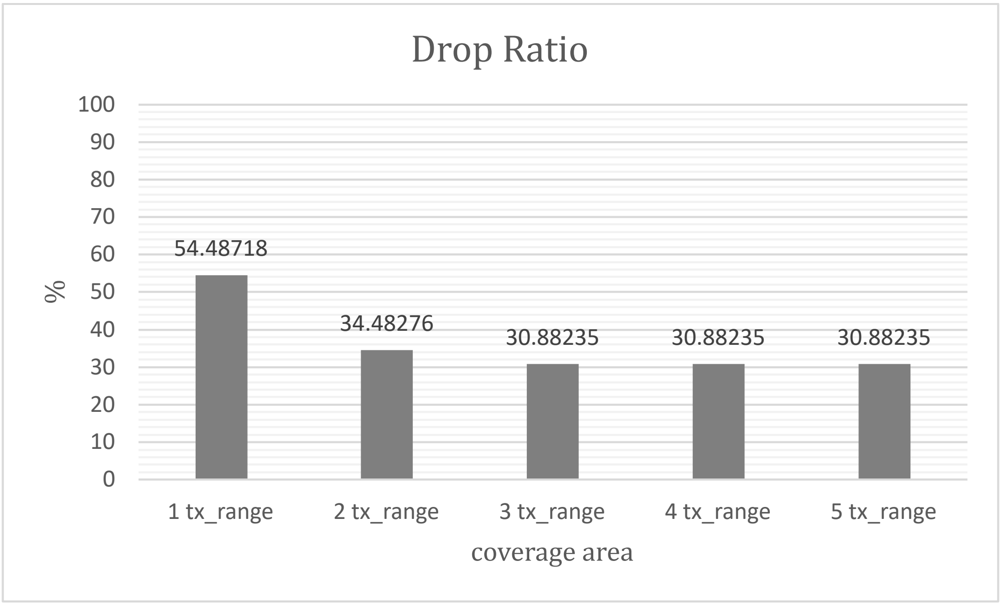

Figure 35: Drop ratio vs Coverage area

- **Summary**
  The nodes in this network are in mesh topology. So every node is connected with each other. That is why increasing the coverage area when a node goes out of the range of the receiver their sent packets go through other nodes.
  And as the nodes go distant from each other in the wpan can function better. Thus it increases the throughput of the network and the end-to-end delay decreases. And the delivery ratio also gets better.
  But if the nodes are further moved away from each other and no nodes are in the transmission range no flow can reach the receiver and every statistic of the network becomes zero.

Task B

Simulation with Proposed Modification

The paper chosen for implementing the modifications is TCP-Peach: a new congestion control scheme for satellite IP networks [1].

Network topology under simulation

The simulation scenario described in paper [1] is a satellite network as seen in Figure 36.

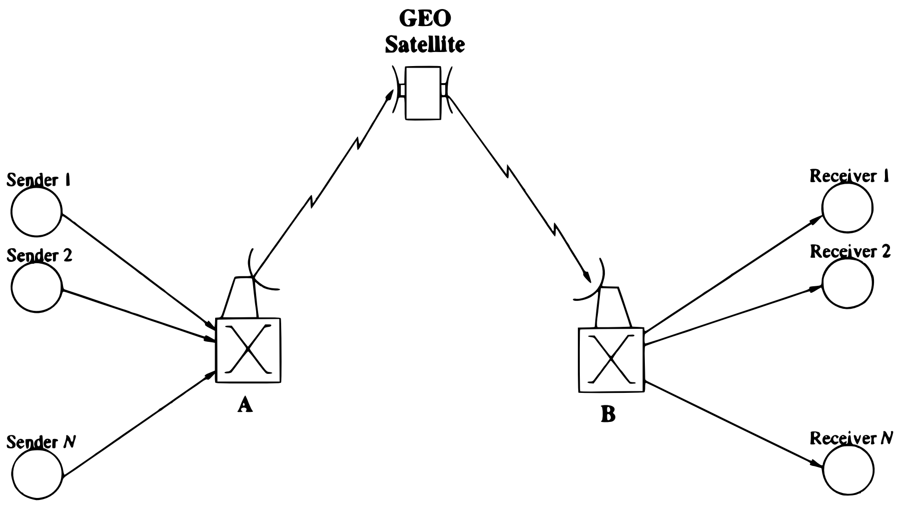

Figure 36: Network topology in paper [1]

But due to complicacy in implementing the satellite network we implemented a dumbbell network as shown in Figure 37. The central two nodes are the access point nodes for the respective stationary wifi nodes. The ap nodes are in point to point connection. A node from one side sends a packet to a node on the other side. This beats the purpose of having satellites in the network to get link errors but as this was a simulation in ns-3 to learn how to modify ns-3 modules, the network was simplified.

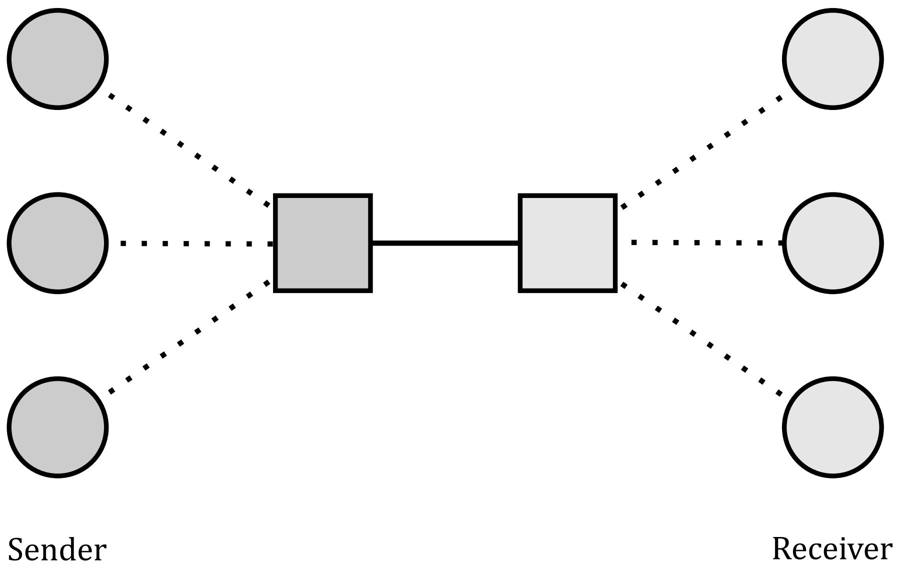

Figure 37: Implemented network topology

Overview of the proposed algorithm

TCP-Peach contains the following algorithms: *Sudden Start*, *Congestion Avoidance*, *Fast Retransmit*, and *Rapid Recovery*, as highlighted in Figure 38.

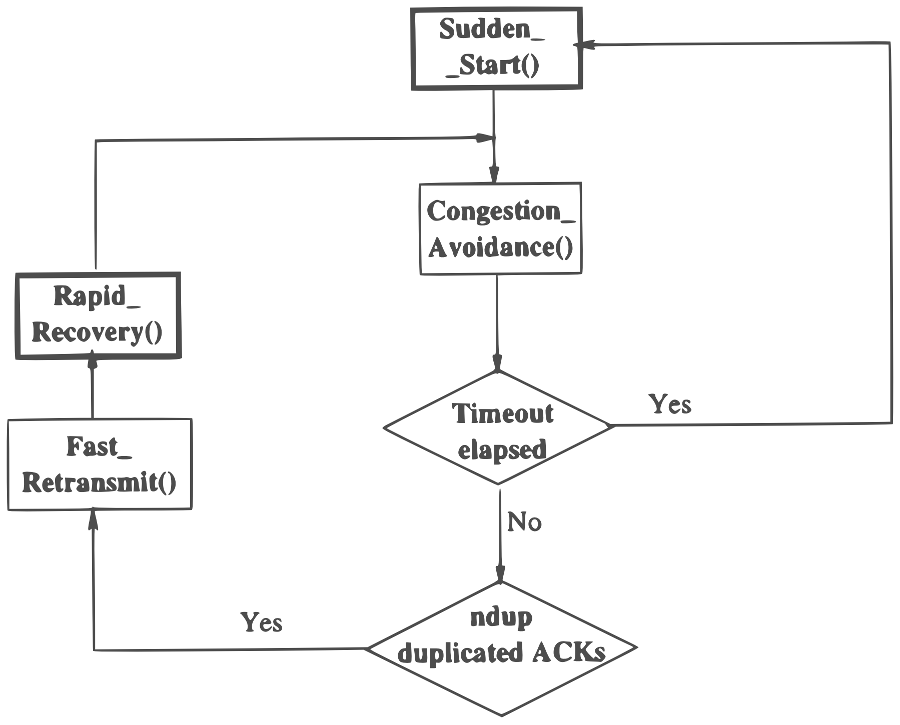

Figure 38: TCP-Peach scheme from [1]

The new, modified algorithms here are Sudden Start and Rapid Recovery. They are based on the use of *dummy segments*.

- Dummy Segment
  Low-priority segments which are copy of the last transmitted data segment. When the path is congested these are discarded first. The sender interprets ACKs for dummy segments as the evidence that there are unused resources in the network and accordingly, can increase its transmission rate.

- Sudden Start
  In the beginning of a connection, the sender sets the congestion window to 1 and after the first data segment transmission, it transmits 1 less than maximum value of the congestion window dummy segments in one RTT.

```
Sudden_Start()
    $cwnd = 1$;
    $\tau = RTT/rwnd$;
    $send$ (Data_Segment) ;
    $for\ (i = 1\ to\ rwnd - 1)$
        $wait\ (\tau)$ ;
        $send$ (Dummy_Segment) ;
    $end$;
$end$;
```

Figure 39: Sudden Start algorithm from [1]

- Rapid Recovery

```
Rapid_Recovery()
    $cwnd = cwnd/2$;
    $adsn = 2 * cwnd$;
    $wdsn = cwnd$;
    $infl\_seg = 0$;
    $t_{Retr} = t$;
    $END = 0$;
    $while\ (END = 0)$
        $if$ (ACK_ARRIVAL)
            $if$ (DATA_ACK_ARRIVAL)
                $cwnd = cwnd + 1$;
                $infl\_seg = infl\_seg + 1$;
            $else\ if$ (DUMMY_ACK_ARRIVAL)
                $if\ (wdsn = 0)$
                    $cwnd = cwnd + 1$;
                    $infl\_seg = infl\_seg + 1$;
                $else$
                    $wdsn = wdsn - 1$;
                $end$;
            $end$;
            $if\ (cwnd > nackseg)$
                $while\ (cwnd > nackseg)$
                    $send$ (Data_Segment) ;
                    $nackseg = nackseg + 1$;
                $end$;
            $else\ if\ (adsn > 0)$
                $send$ (Dummy_Segment) ;
                $send$ (Dummy_Segment) ;
                $adsn = adsn - 2$;
            $end$;
            $if$ (LOST_SEGMENT_ACKED)
                $END = 1$;
                $cwnd = cwnd - infl\_seg$;
            $end$;
        $end$;
        $if\ (t > t_{Retr} + RTO)$
            Slow_Start();
        $end$;
    $end$;
$end$;
```

Figure 40: Rapid Recovery algorithm from [1]

The Rapid Recovery algorithm substitutes the classical Fast Recovery algorithm with the objective of improving throughput while there is link errors in satellite networks. As we ignored the satellites in our proposed implementation we will ignore explaining the algorithm in details here.

Modifications in simulator to implement algorithm

Unfortunately I was unsuccessful to implement the algorithm in our ns-3 simulation. The following is what I planned and what I succeeded to do.

- For sending dummy segments, the header of the TCP packets is to be modified before sending. The headers are added to a packet before sending in the *TcpSocketBase* class.
- While receiving, the packet header is to be checked for dummy segment and the ACK for it is to be sent accordingly. The responsible methods for this operation are also in the *TcpSocketBase* class.
- With the help of these dummy segments and their ACKs, the sudden start algorithm in a separate TCP Congestion Control Algorithm can increase their congestion window accordingly.
- For a new TCP congestion control algorithm I wrote a header (*.h*) and a source code (*.cc*) file, named *tcp-peach.h* and *tcp-peach.cc* respectively. And added their signatures in the *wscript* file for the ns-3 build to include them. This class extends the ns-3 *TcpCongestionOps* class to implement the TCP Peach algorithms.
- Where I failed to implement this *TcpPeach* class properly is I was unable to modify the *TcpSocketBase* class. Editing the file or extending the class and modifying one of its methods was resulting in compilation error.

Results with graphs

Let us consider the results from the paper [1] to understand our expected outputs.

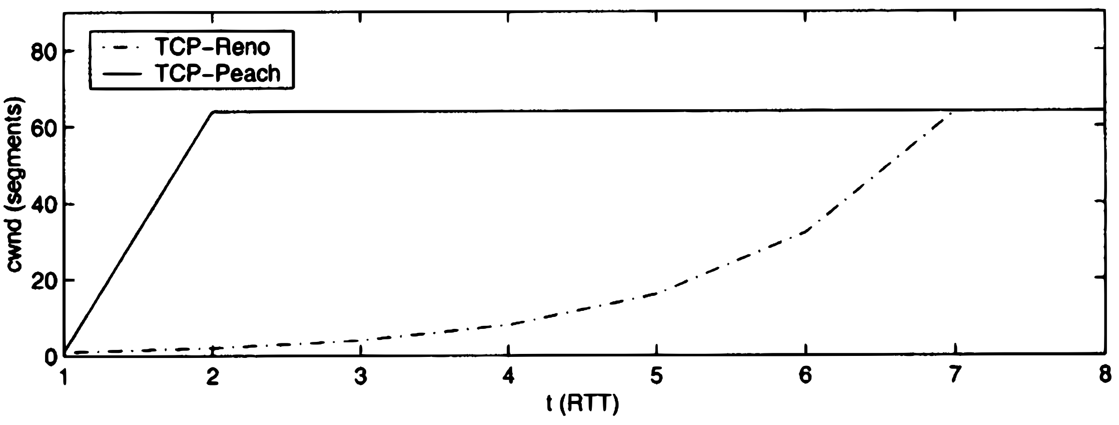

Figure 41: Comparison between TCP Peach and TCP Reno in the beginning of a new connection [1]

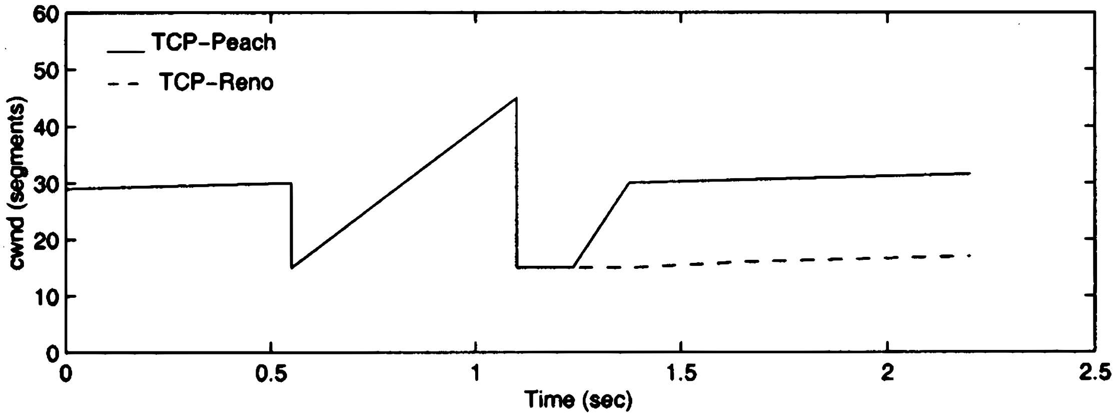

Figure 42: TCP Peach and TCP Reno behaviour when segment losses occur due to link errors [1]

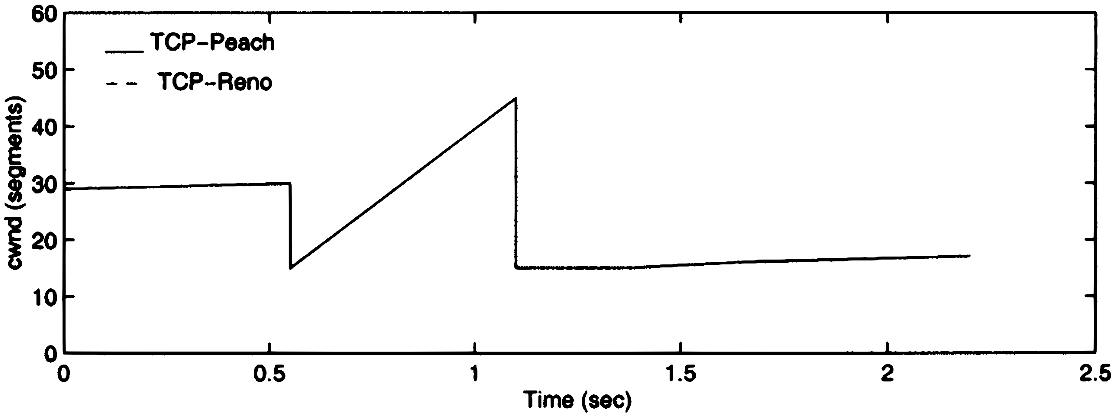

Figure 43: TCP Peach and TCP Reno behaviour when segment losses occur due to network congestion [1]

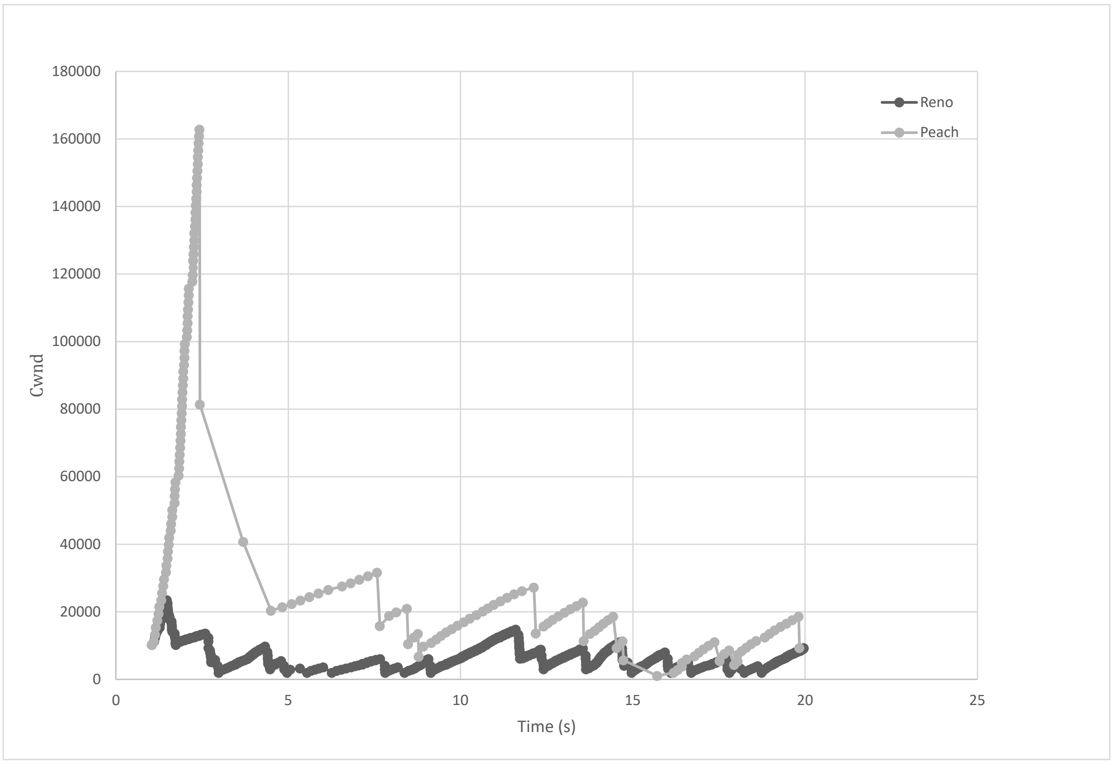

Figure 44: Comparison between congestion window of Reno and my unfinished modified file from ns-3 simulation

Summary

The Figure 41 shows the *cwnd* of TCP Reno and TCP Peach. As TCP Peach uses dummy segment to check availability of the network and increases the congestion window accordingly it reaches *max cwnd* in one *RTT* while TCP Reno takes up a long time.
The Figure 42 shows the comparative graph in the recovery stage if there is link error in the network and the segment loss happens due to that link error. TCP Peach performs better because of its Rapid Recovery. But in a network with no link error when the segment losses are due to congestion both the algorithms work same as shown in Figure 43.
The Figure 44 is the plotting of *cwnd* of the unfinished modified file for TCP Peach against the *cwnd* of existing Reno algorithm in ns-3.

References

[1] I.F. Akyildiz, G. Morabito, and S. Palazzo. Tcp-peach: a new congestion control scheme for satellite ip networks. *IEEE/ACM Transactions on Networking*, 9(3):307–321, 2001. doi: 10.1109/90.929853.