ID : 1805113

Name:K.M Fahim Shahriyar

NS2 Project Report

MAC-Address: Wireless 802.11(static) and Wireless 802.15.4(static)

Routing Protocol: DSDV

Agent type : TCP

Application : FTP

Flow : Random source destination

Modification in C++ code:

In my modification, I modified the congestion control algorithm of "Old Reno".In my modification,I changed the slow start phase,congestion avoidance phase and fast recovery phase such that the variation of congestion window size in minimized.

So to achieve this , I changed **tcp.cc** ,**tcp.h** and **tcpreno.cc.**

In my modification,

In the slow start phase,the congestion window is allowed to increase exponentially until the first packet drop is detected.When the first packet drop is detected, I stored the value of **cwnd_** at that time as the maximum allowable congestion window size(**max_allowed_cwnd**) as well as I changed the current window size to the **three fourth** of **max_allowed_cwnd.**

```
269     bool dupacks_occurs;
270     bool timeout_occurs;
271     bool first_packet_dropped;
272     double max_allowed_cwnd;
273
```

Figure 1:adding some variables in "tcp.h"

```
68  TcpAgent::TcpAgent()
69      : Agent(PT_TCP),
70        t_seqno_(0), dupacks_(0), curseq_(0), highest_ack_(0),
71            cwnd_(0), ssthresh_(0), maxseq_(0), count_(0),
72            rtt_active_(0), rtt_seq_(-1), rtt_ts_(0.0),
73            lastreset_(0.0), closed_(0),
74            dupacks_occurs(false),timeout_occurs(false),
75            first_packet_dropped(false),max_allowed_cwnd(0), t_rtt_(0), t_srtt_(0), t_rttvar
76        t_backoff_(0), ts_peer_(0), ts_echo_(0), tss(NULL), tss_size_(100),
77        rtx_timer_(this), delsnd_timer_(this), burstsnd_timer_(this),
78        first_decrease_(1), fcnt_(0), nrexmit_(0), restart_bugfix_(1),
79            cong_action_(0), ecn_burst_(0), ecn_backoff_(0), ect_(0),
80            use_rtt_(0), qs_requested_(0), qs_approved_(0),
81        qs_window_(0), qs_cwnd_(0), frto_(0)
```

Figure 2:Initalizing those variables in "tcp.cc"

```
1120  */
1121  void TcpAgent::opencwnd()
1122  {
1123      double increment;
1124      if (!first_packet_dropped) {
1125          /* slow-start (exponential) */
1126          cwnd_ += 1;
```

Figure 3:cwnd_ is increasing exponentially until first packet loss detected

After detecting the very first packet drop "**slow start phase**" stops permanently and "**congestion avoidance phase**" starts.From this point the congestion window size is allowed to increase linearly until it reach to the **max_allowed_cwnd** or a packet drop detected.Whenever the current window size(**cwnd_**) reach the **max_allowed_cwnd or** packet drop is detected,the current window size is reduced to the **three-fourth** of **max_allowed_cwnd** again.

```
212     last_cwnd_action_ = CWND_ACTION_DUPACK;
213     //printf("Packet Drop detected due to dupack\n");
214     //slowdown(CLOSE_SSTHRESH_HALF|CLOSE_CWND_HALF);
215     //dupacks_occurs=true;
216     //timeout_occurs=false;
217     if(first_packet_dropped == false)
218         {
219             first_packet_dropped = true;
220             max_allowed_cwnd = (double)cwnd_;
221         }
222
223     slowdown();
```

Figure 4:when a packet loss is detected via "duplicate acknowledgement" ,the function "dupack()" is called in tcp.cc.Reducing the "cwnd_" at this point

```
1138         case 1:
1139             /* This is the standard algorithm. */
1140             increment = increase_num_ / cwnd_;
1141             if ((last_cwnd_action_ == 0 ||
1142               last_cwnd_action_ == CWND_ACTION_TIMEOUT)
1143               && max_ssthresh_ > 0) {
1144                 increment = limited_slow_start(cwnd_,
1145                   max_ssthresh_, increment);
1146             }
1147             cwnd_ += increment;
1148
1149             if((double)cwnd_ >= max_allowed_cwnd)
1150                 slowdown();
```

Figure 5:in When the "cwnd_" reaches "max_allowed_cwnd",cwnd_ is reduced

```
1245
1246  void
1247  TcpAgent::slowdown()
1248  {
1249      cwnd_ = (double)(max_allowed_cwnd * 3.0/4);
1250      //printf("max allowable window size: %lf\n",max_allowed_cwnd);
1251  }
1252
```

Figure 6:overloading "slowdown()" function in "tcp.cc"

```
        ++nrexmit_;
        last_cwnd_action_ = CWND_ACTION_TIMEOUT;
        //slowdown(CLOSE_SSTHRESH_HALF|CLOSE_CWND_RESTART);
       // printf("Packet drop detected due to timeout\n");
       // dupacks_occurs=false;
        //timeout_occurs=true;
        if(first_packet_dropped == false)
        {
            first_packet_dropped = true;
            max_allowed_cwnd = (double)cwnd_;
        }
        slowdown();
```

Figure 7:when a packet loss is detected via "Retransmission timeout" ,the function "timeout()" is called in tcp.cc.Reducing the "cwnd_" at this point

In this way,I limited the variation of the "**cwnd_**" in between **max_allowed_cwnd** and three-fourth of **max_allowed_cwnd.**

Impact of My Modification:

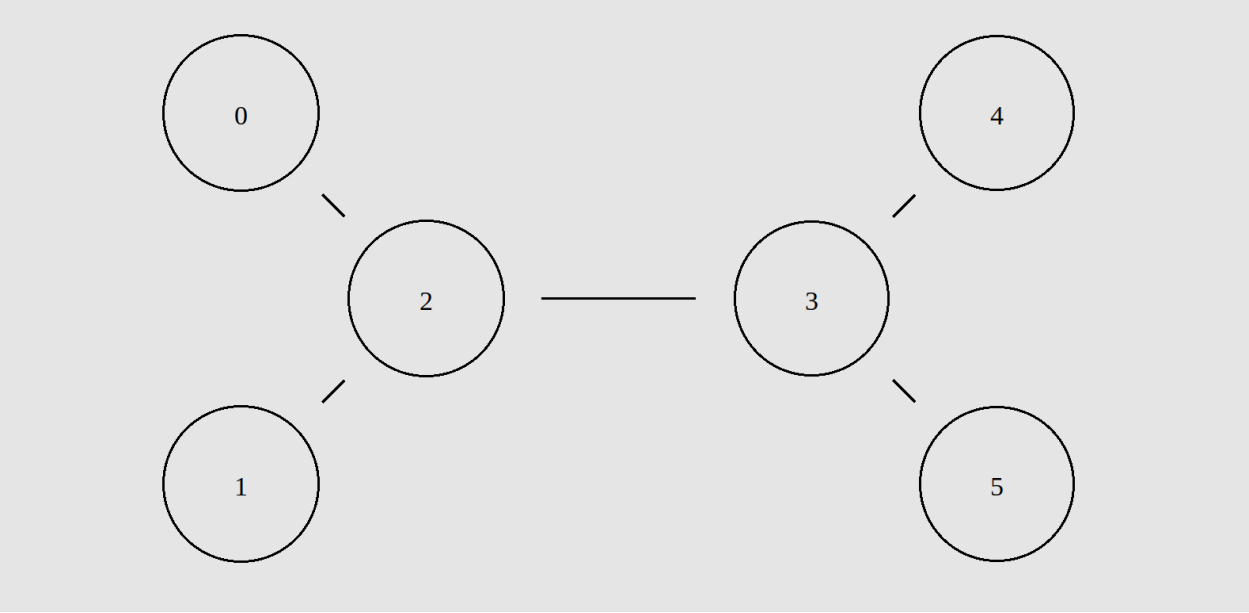
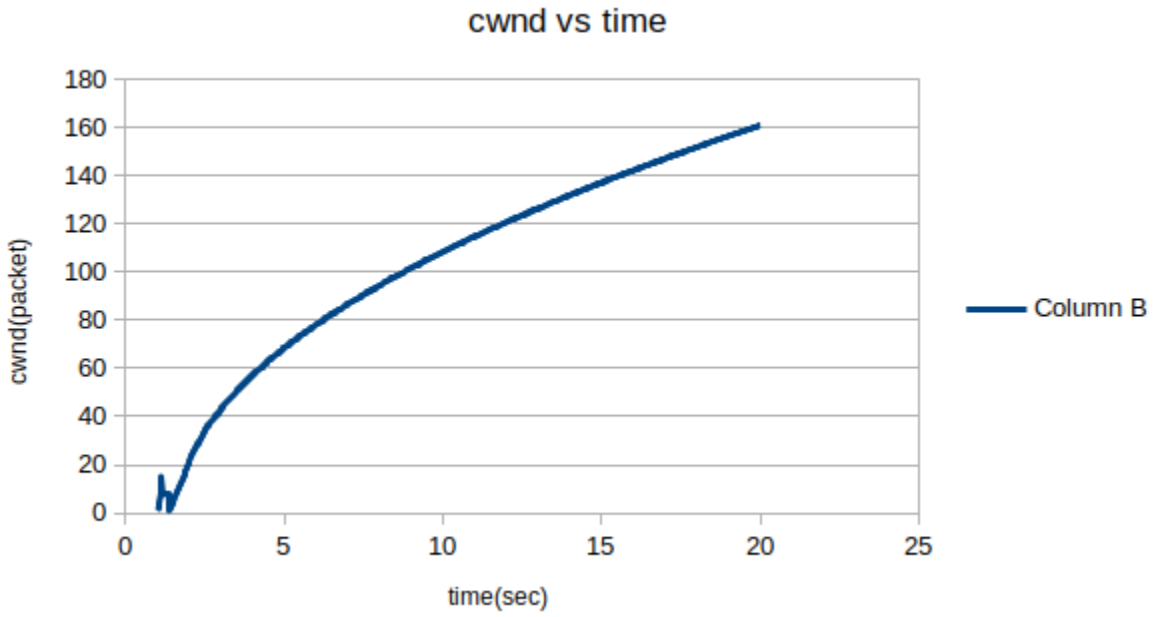
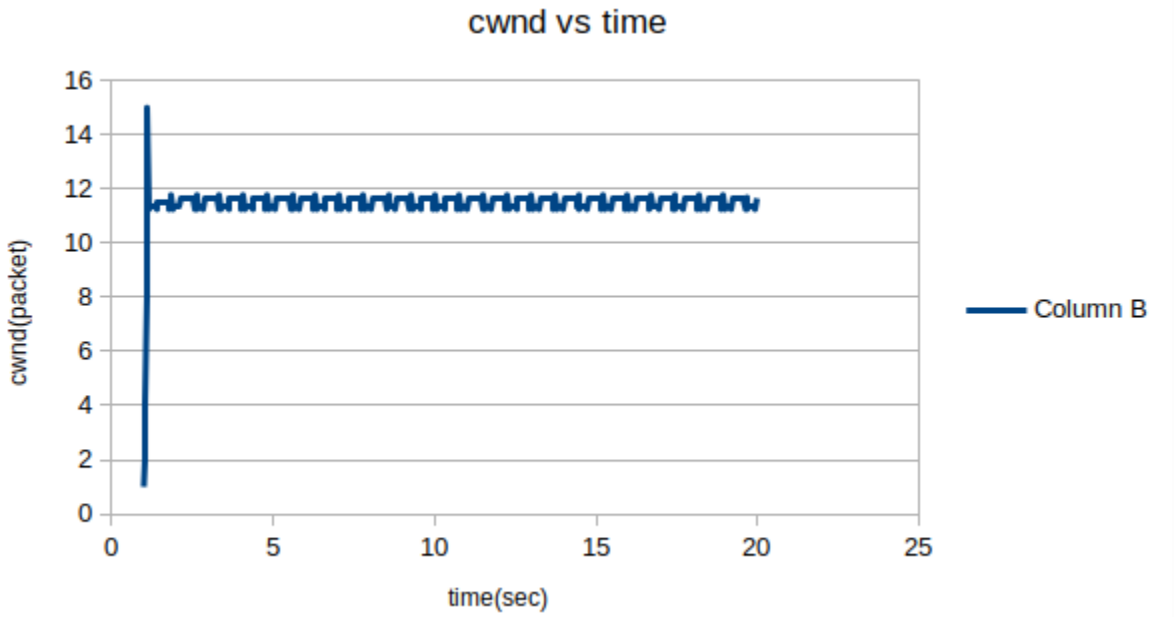


Figure:Paper Suggested topology

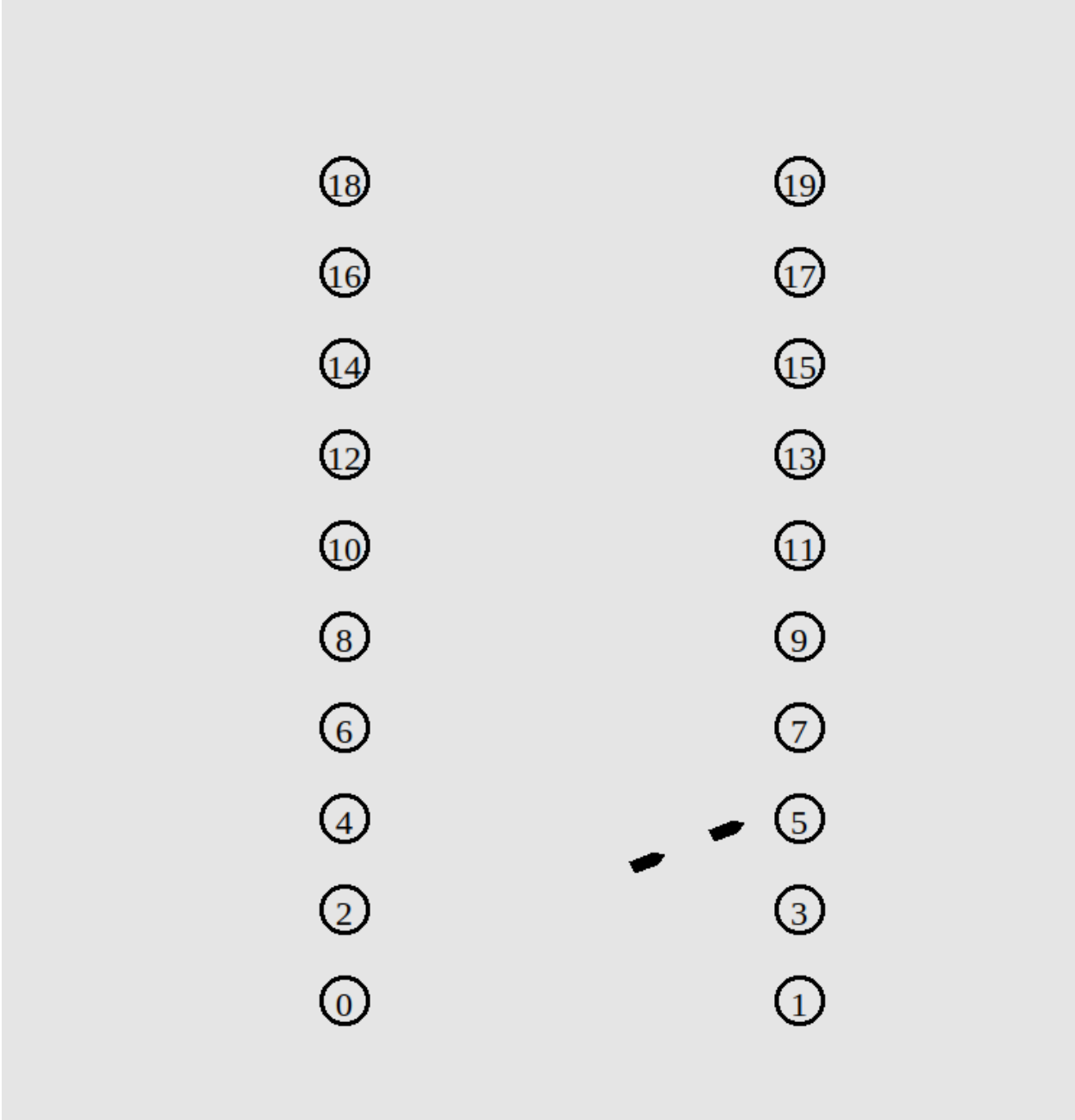
Congestion window vs time(Existing implementation):



Congestion window vs time(Modified implementation):

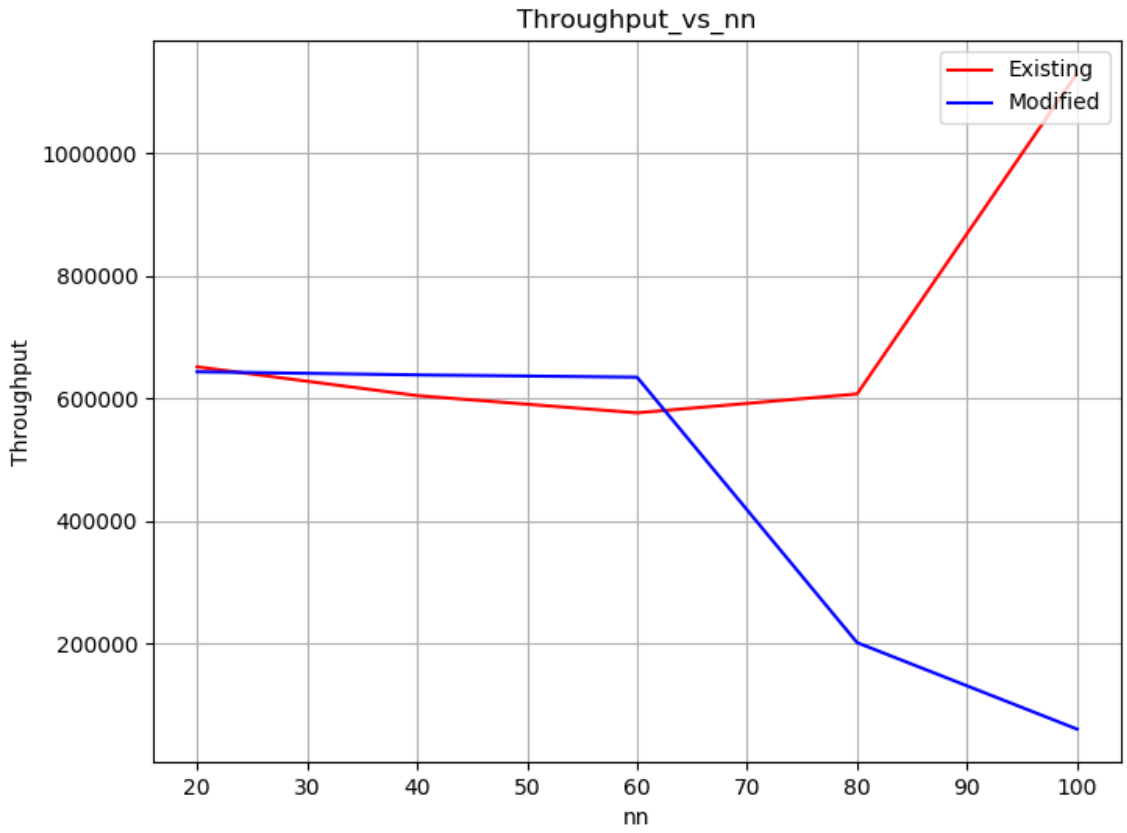


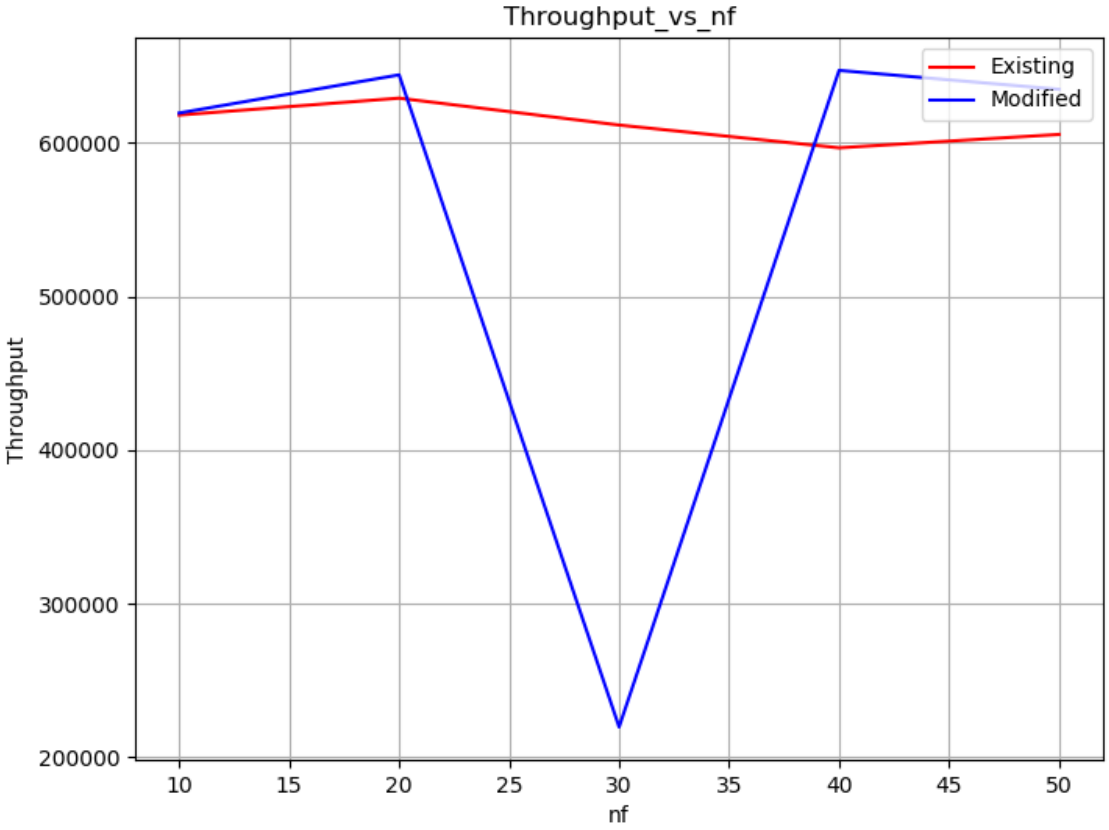
Wireless topology:

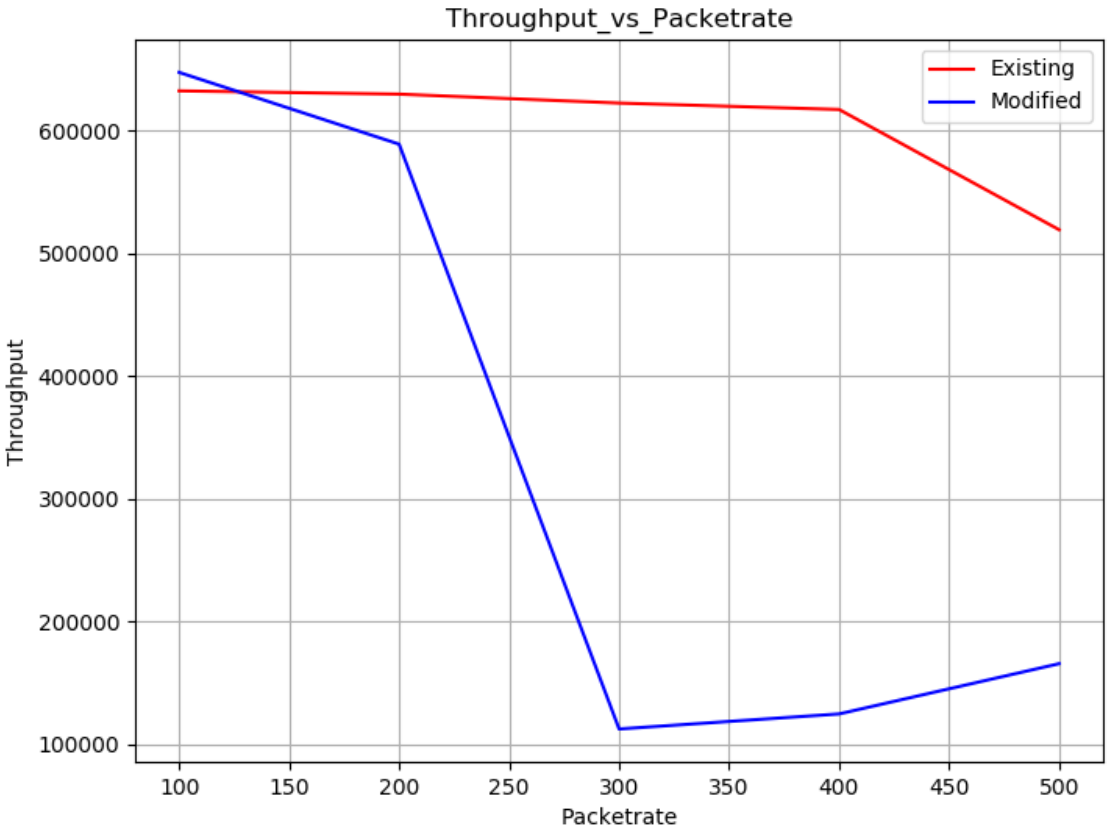


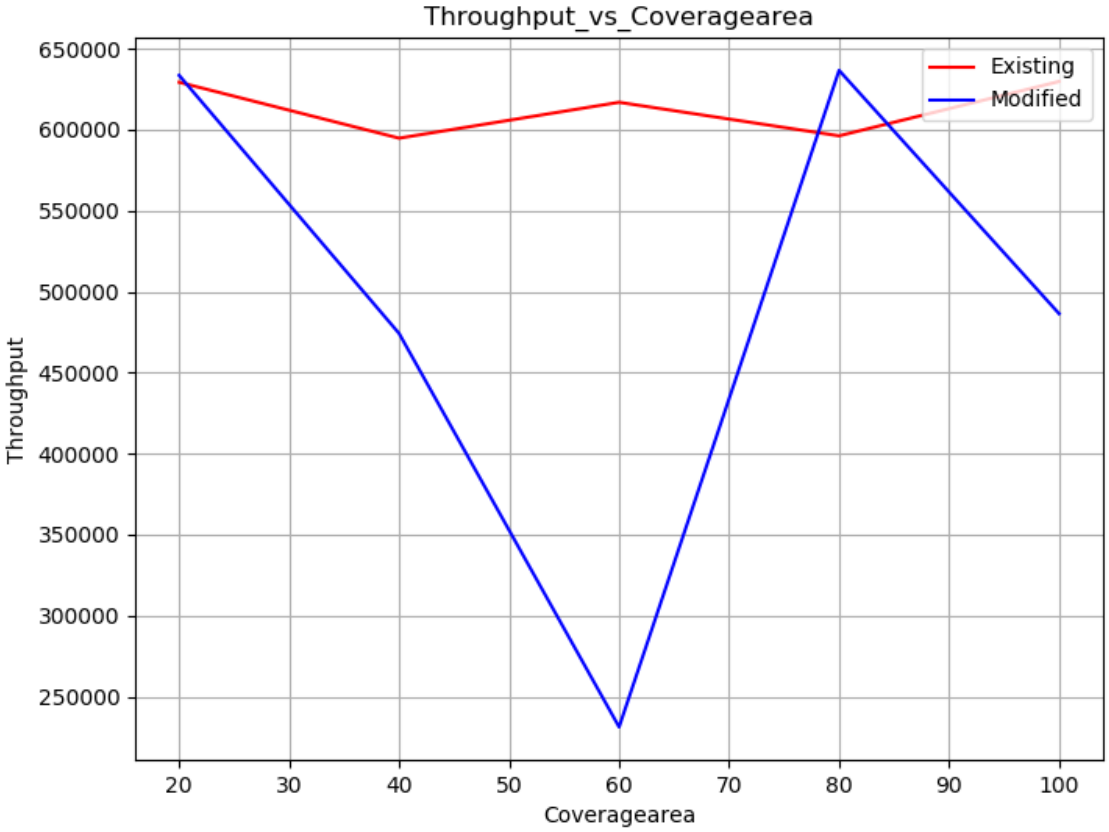
PLOTS(For 802.11 static) :

Throughput:

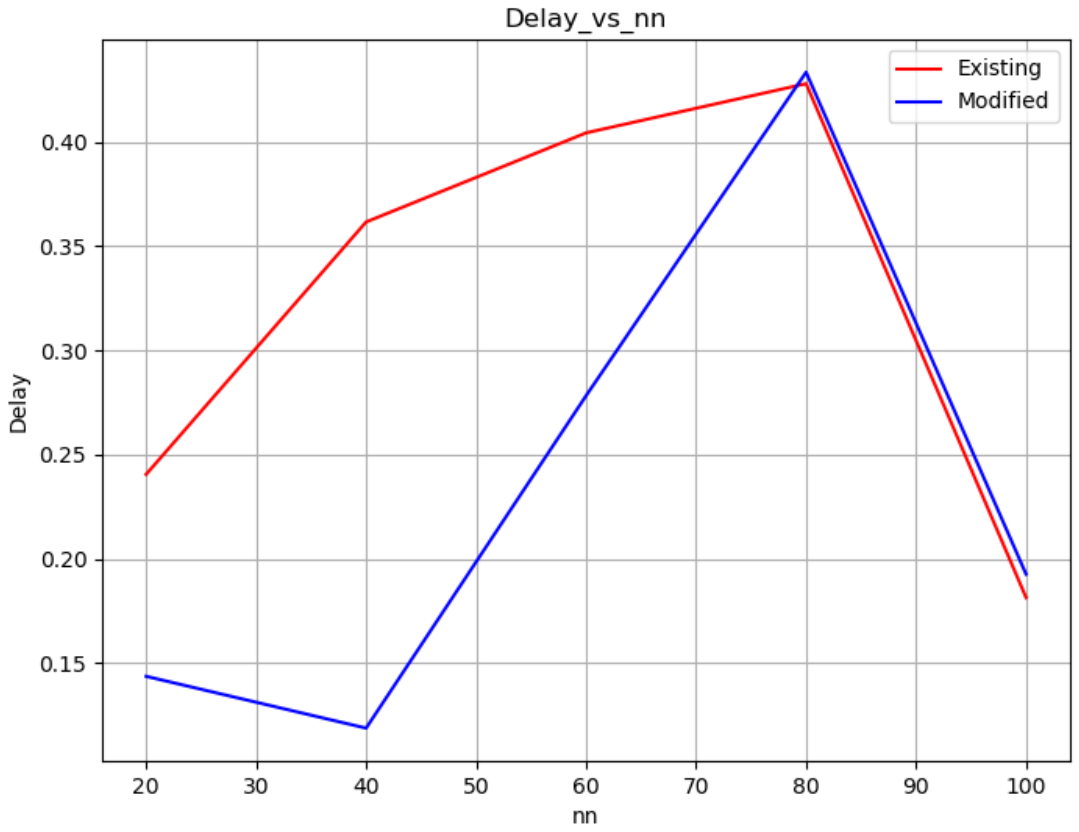


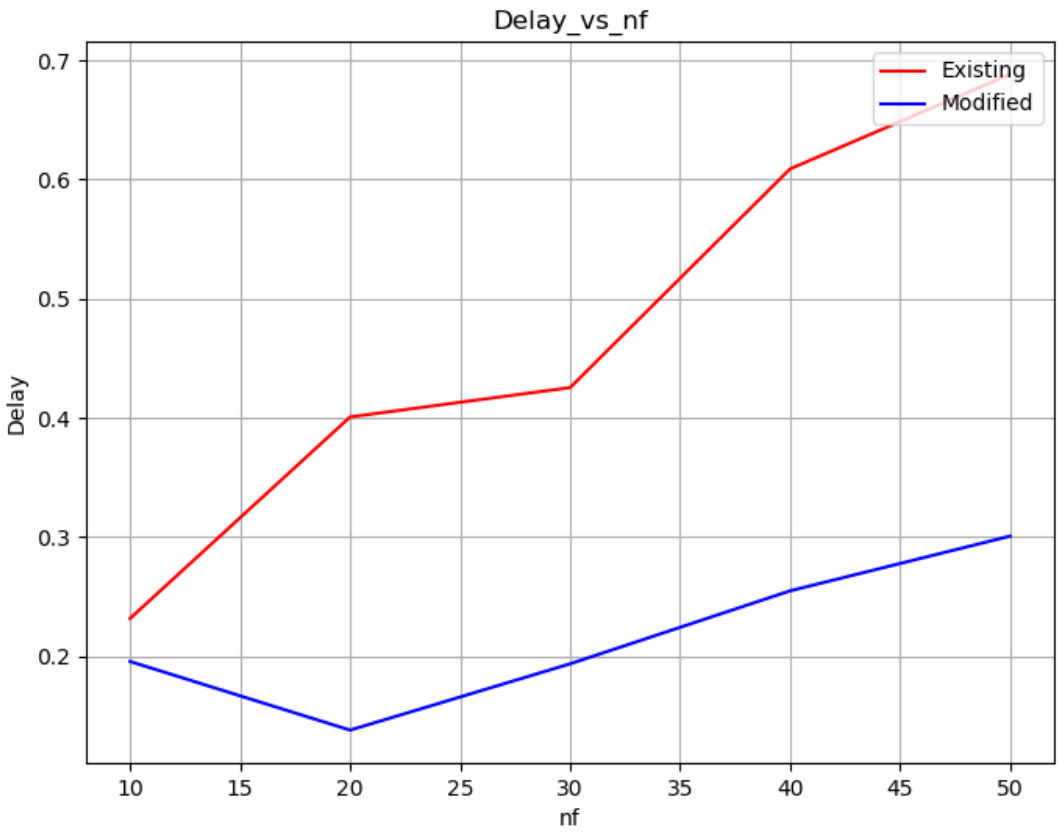

Throughput_vs_nf
Existing
Modified
600000
500000
400000
300000
200000
Throughput
10
15
20
25
30
35
40
45
50
nf

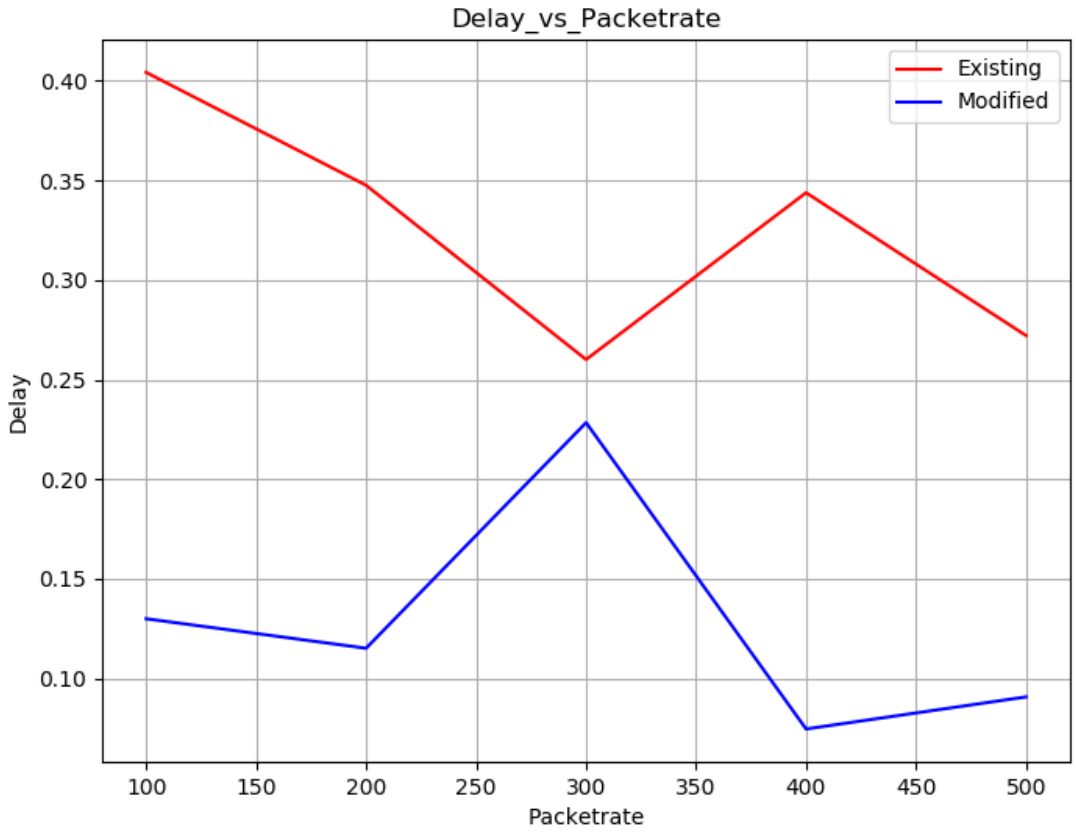

Throughput_vs_Packetrate
Existing
Modified
600000
500000
400000
300000
200000
100000
Throughput
100
150
200
250
300
350
400
450
500
Packetrate

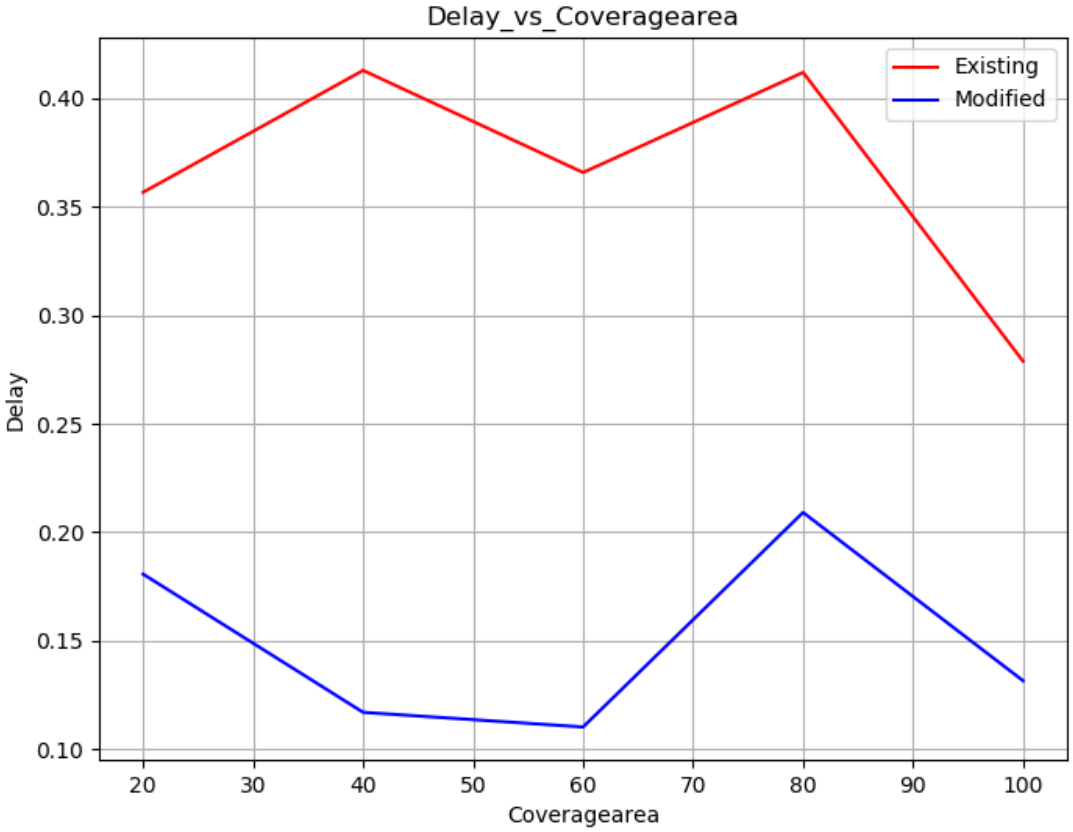

Throughput_vs_Coveragearea
Existing
Modified
Throughput
Coveragearea
650000
600000
550000
500000
450000
400000
350000
300000
250000
20
30
40
50
60
70
80
90
100

End to End delay:

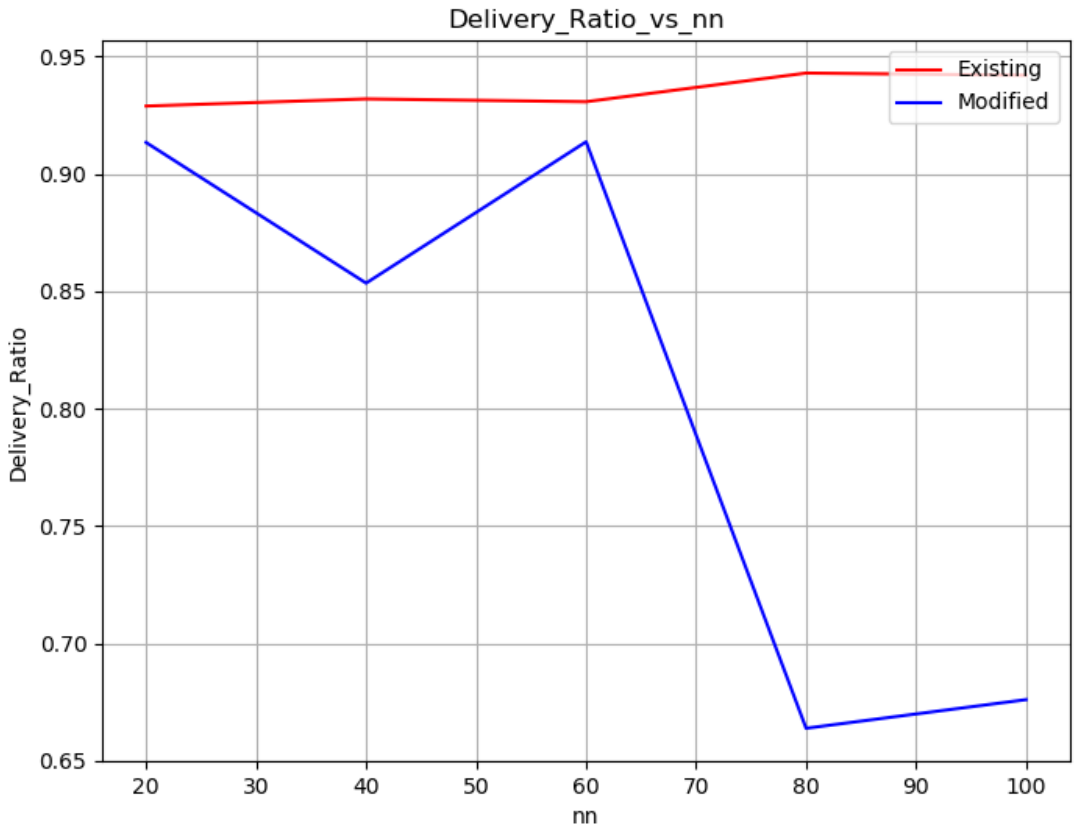


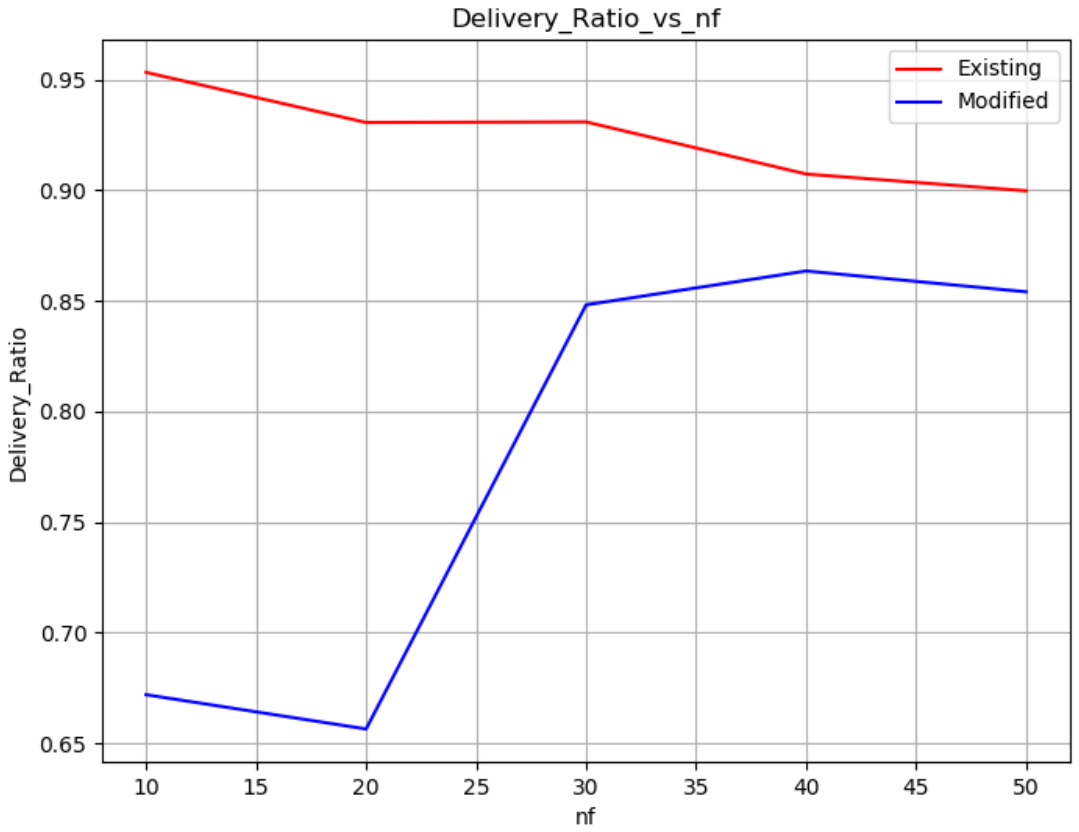

Delay_vs_nf
Existing
Modified
Delay
nf
0.7
0.6
0.5
0.4
0.3
0.2
10
15
20
25
30
35
40
45
50

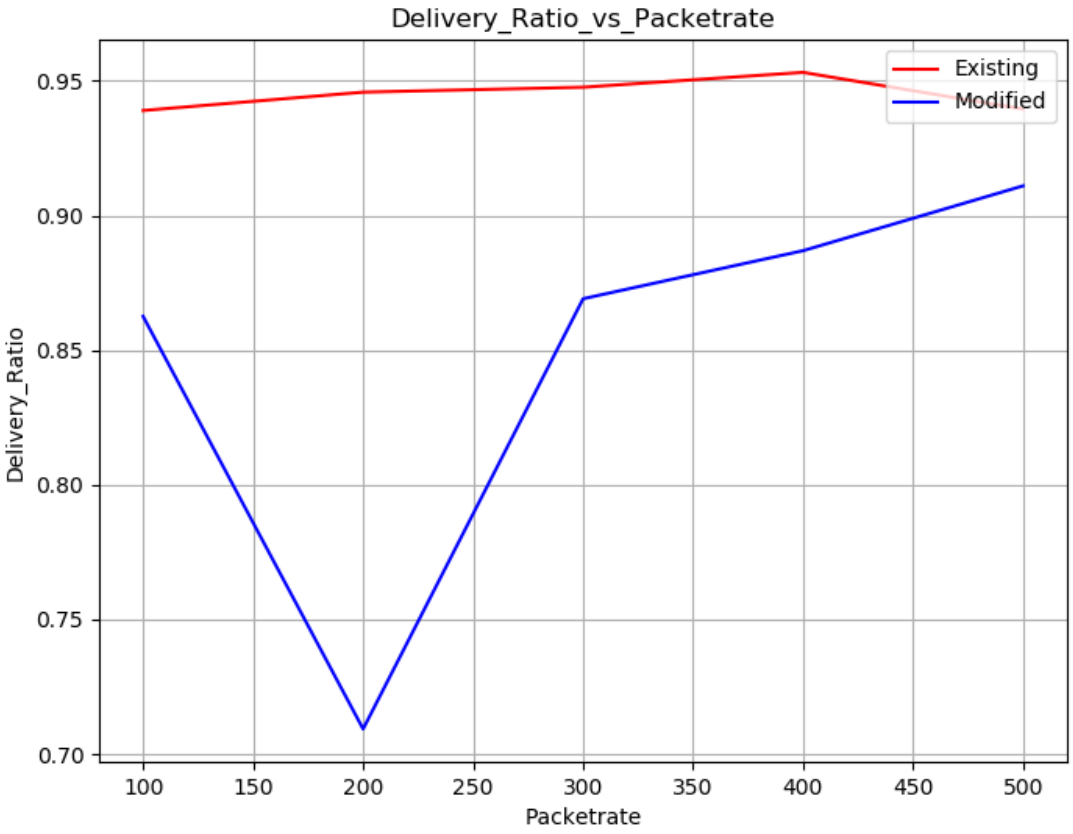

Delay_vs_Packetrate
Existing
Modified
Delay
0.40
0.35
0.30
0.25
0.20
0.15
0.10
100
150
200
250
300
350
400
450
500
Packetrate

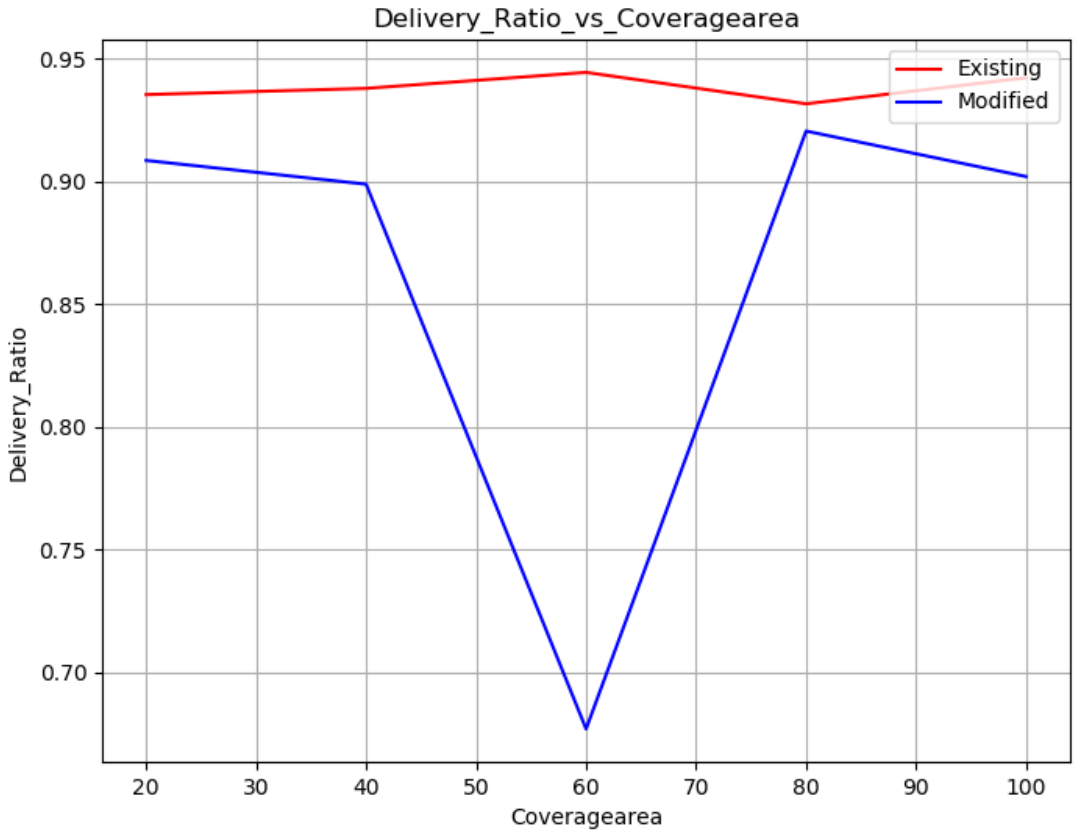

Delay_vs_Coverageare
Existing
Modified
0.40
0.35
0.30
0.25
0.20
0.15
0.10
Delay
20
30
40
50
60
70
80
90
100
Coverageare

Delivery Ratio:

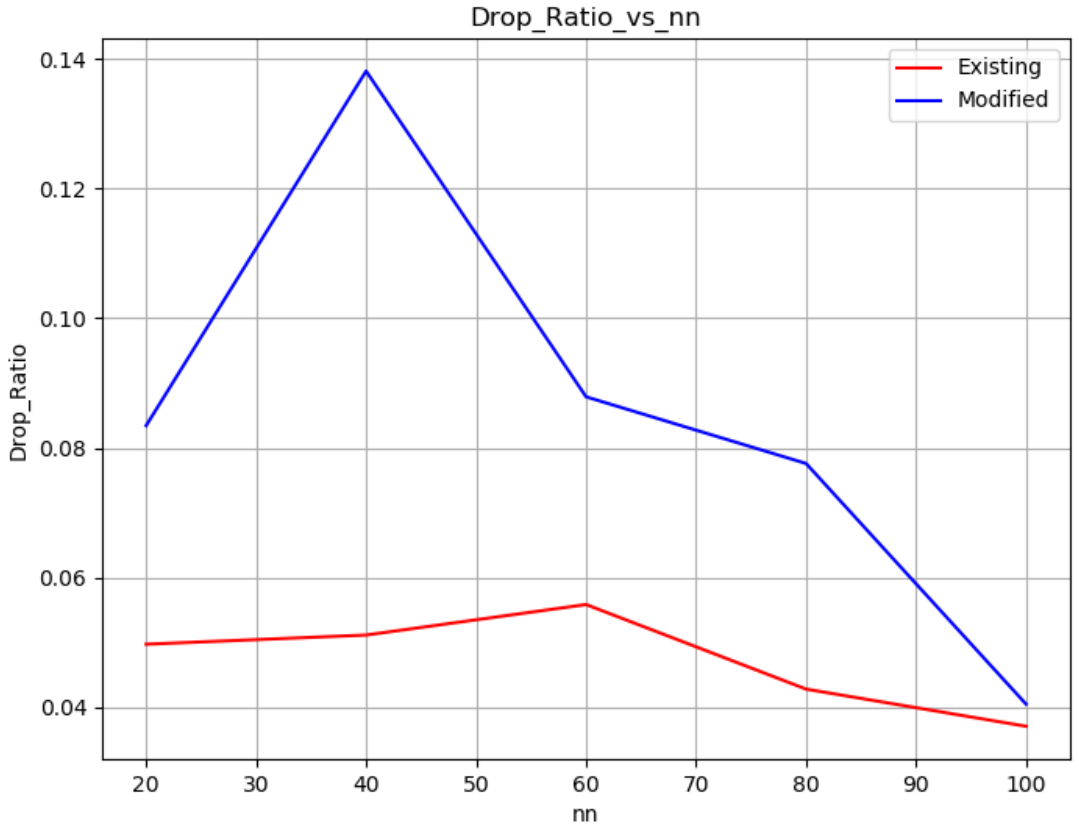


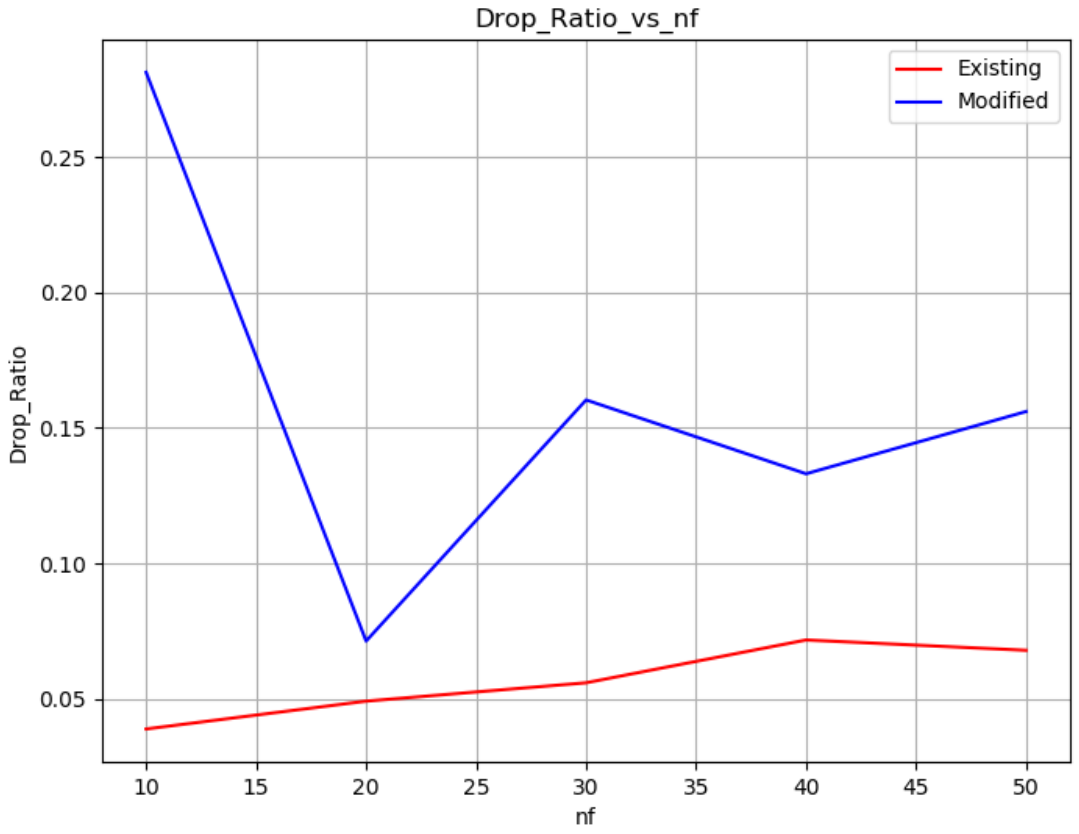

Delivery_Ratio_vs_nf
Existing
Modified
Delivery_Ratio
nf
0.95
0.90
0.85
0.80
0.75
0.70
0.65
10
15
20
25
30
35
40
45
50

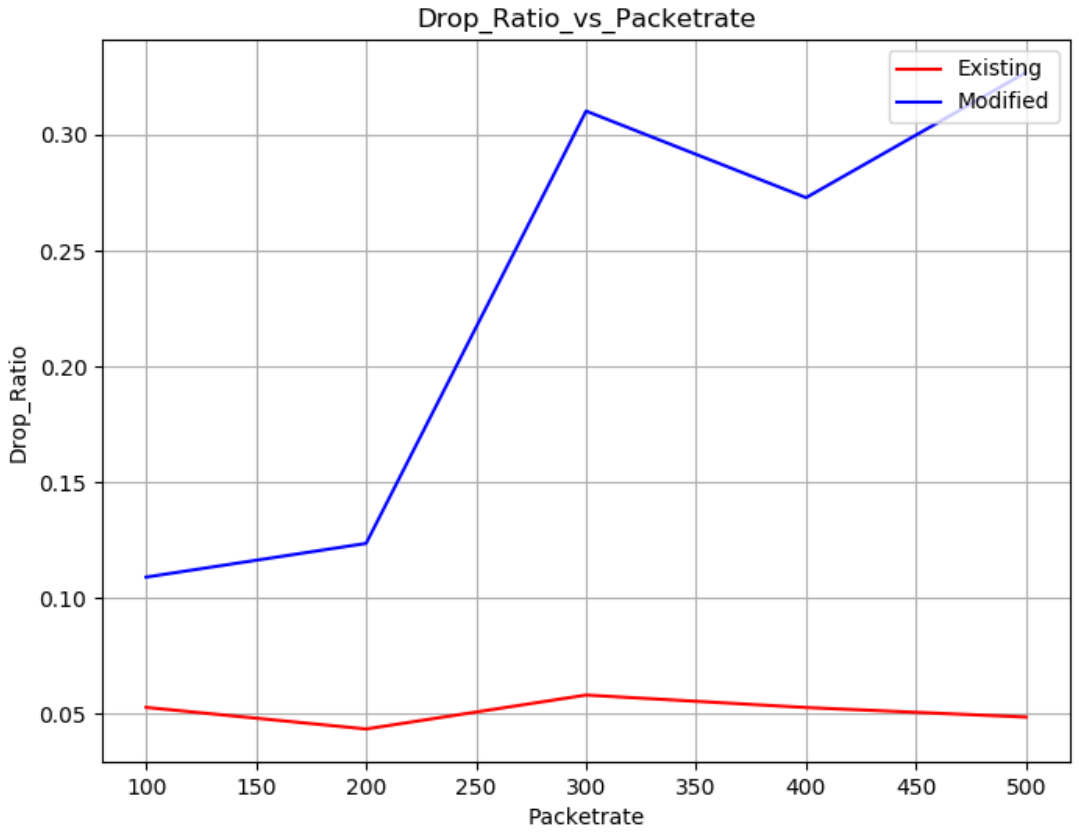

Delivery_Ratio_vs_Packetrate
Existing
Modified
Delivery_Ratio
Packetrate
0.95
0.90
0.85
0.80
0.75
0.70
100
150
200
250
300
350
400
450
500

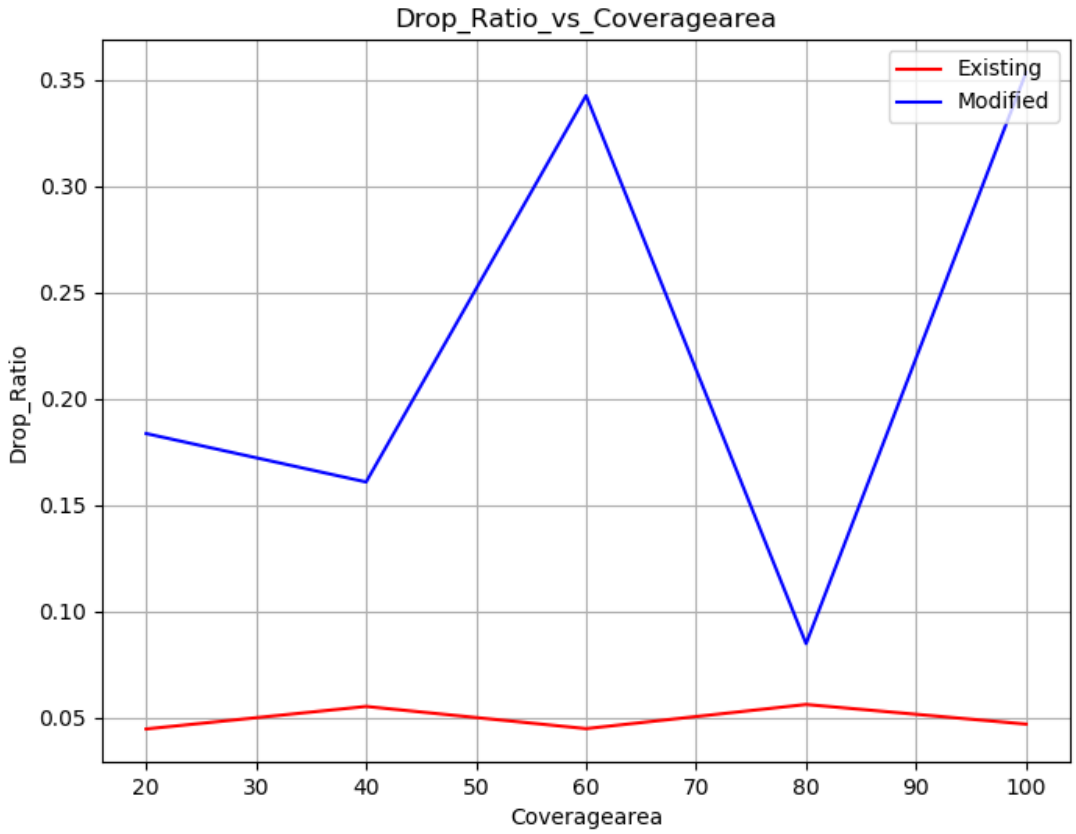

Delivery_Ratio_vs_Coveragearea
Existing
Modified
Delivery_Ratio
0.95
0.90
0.85
0.80
0.75
0.70
20
30
40
50
60
70
80
90
100
Coveragearea

Drop Ratio:


Drop_Ratio_vs_nn
Existing
Modified
Drop_Ratio
0.14
0.12
0.10
0.08
0.06
0.04
20
30
40
50
60
70
80
90
100
nn

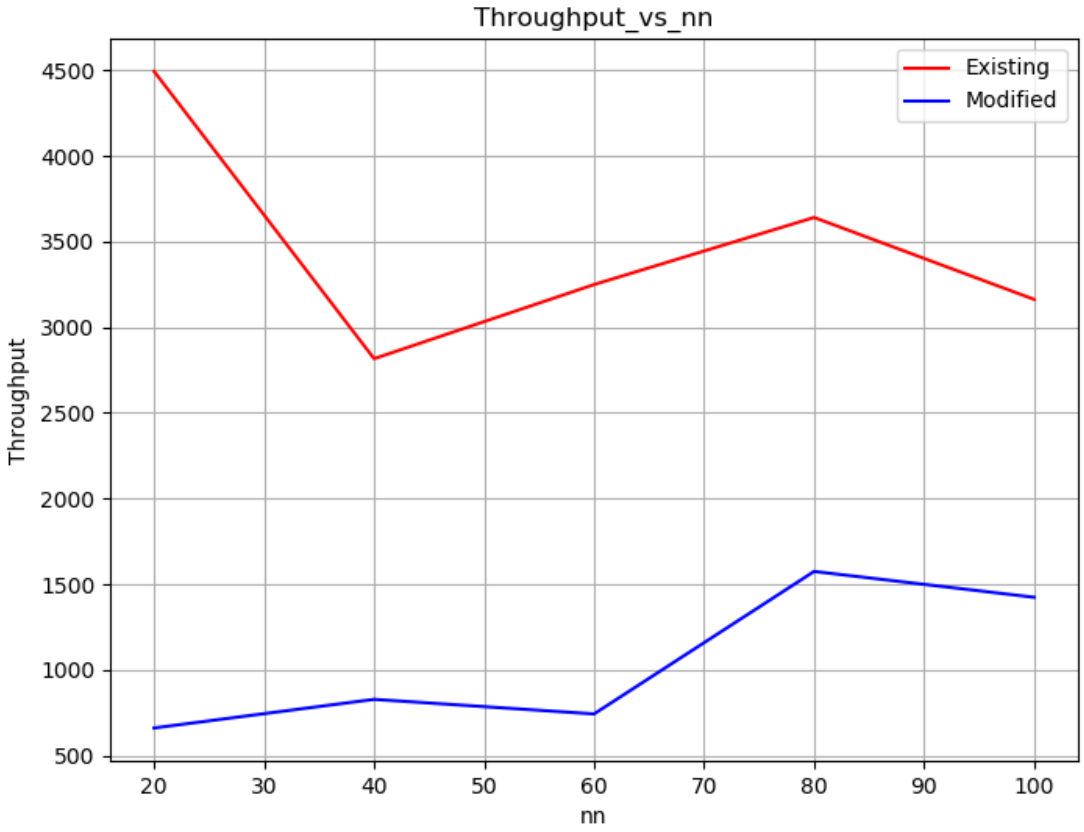

Drop_Ratio_vs_nf
Existing
Modified
Drop_Ratio
0.25
0.20
0.15
0.10
0.05
10
15
20
25
30
35
40
45
50
nf

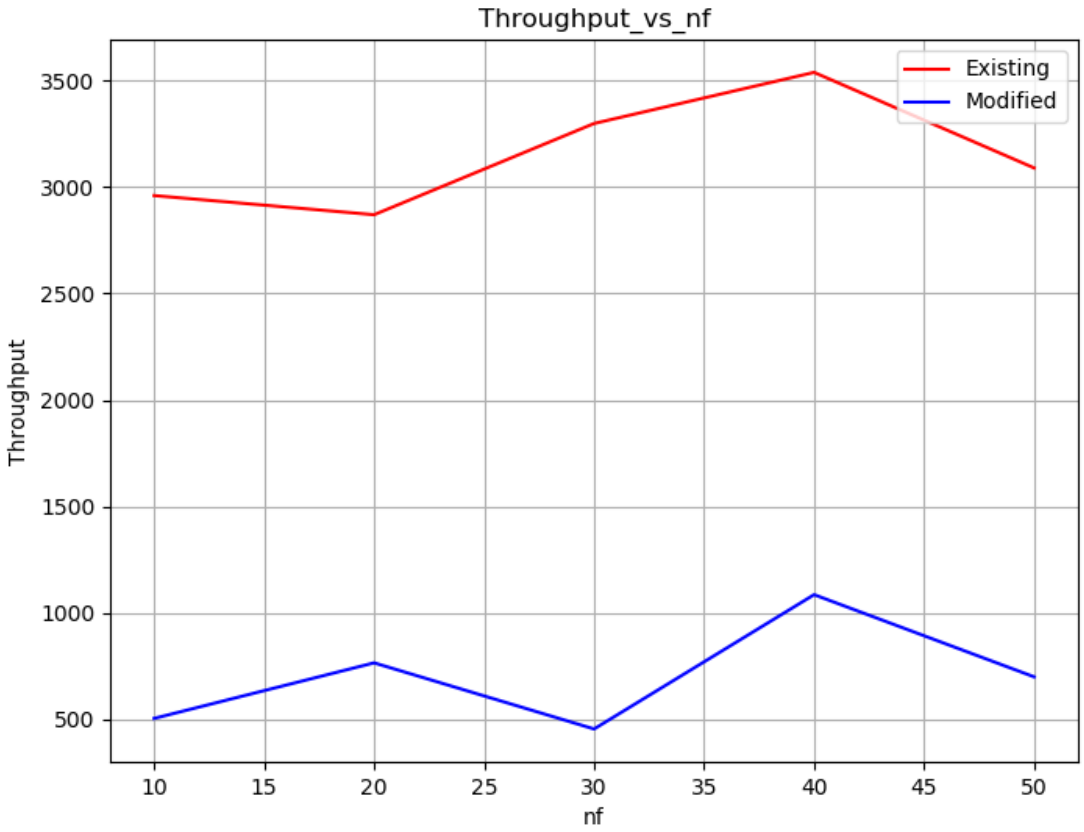

Drop_Ratio_vs_Packetrate
Existing
Modified
Drop_Ratio
0.30
0.25
0.20
0.15
0.10
0.05
100
150
200
250
300
350
400
450
500
Packetrate

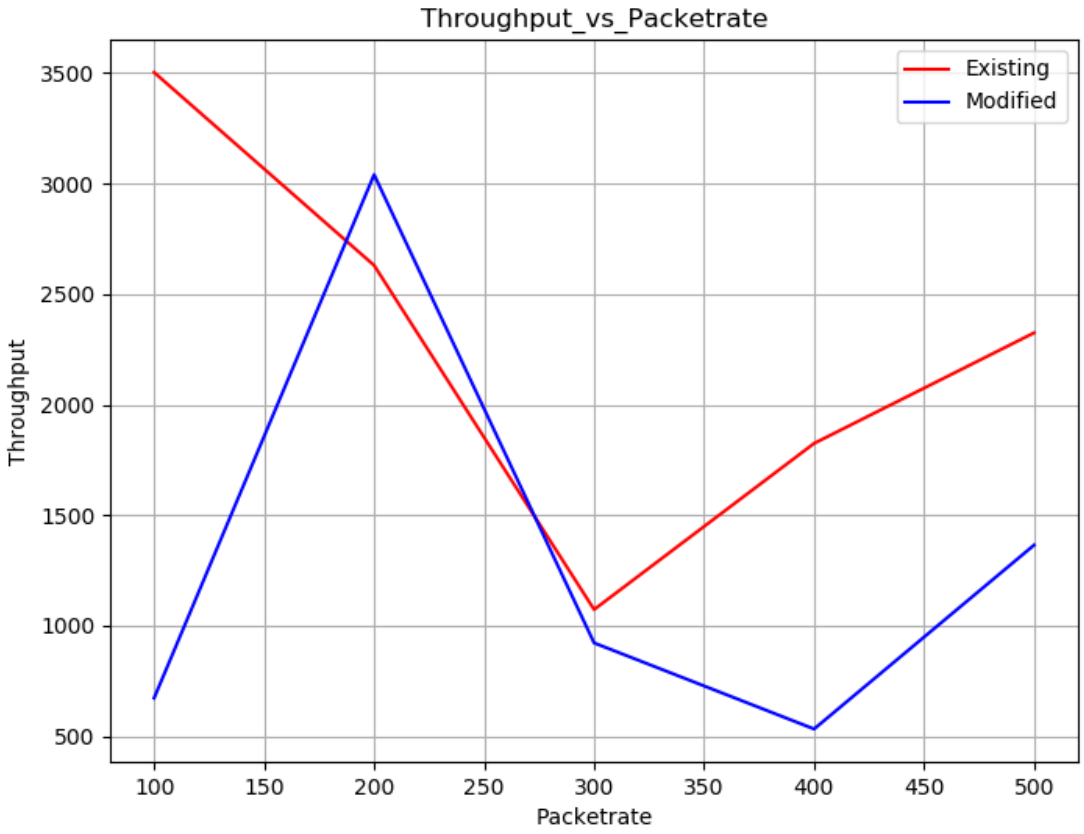

Drop_Ratio_vs_Coveragearea
Existing
Modified
Drop_Ratio
0.35
0.30
0.25
0.20
0.15
0.10
0.05
20
30
40
50
60
70
80
90
100
Coveragearea

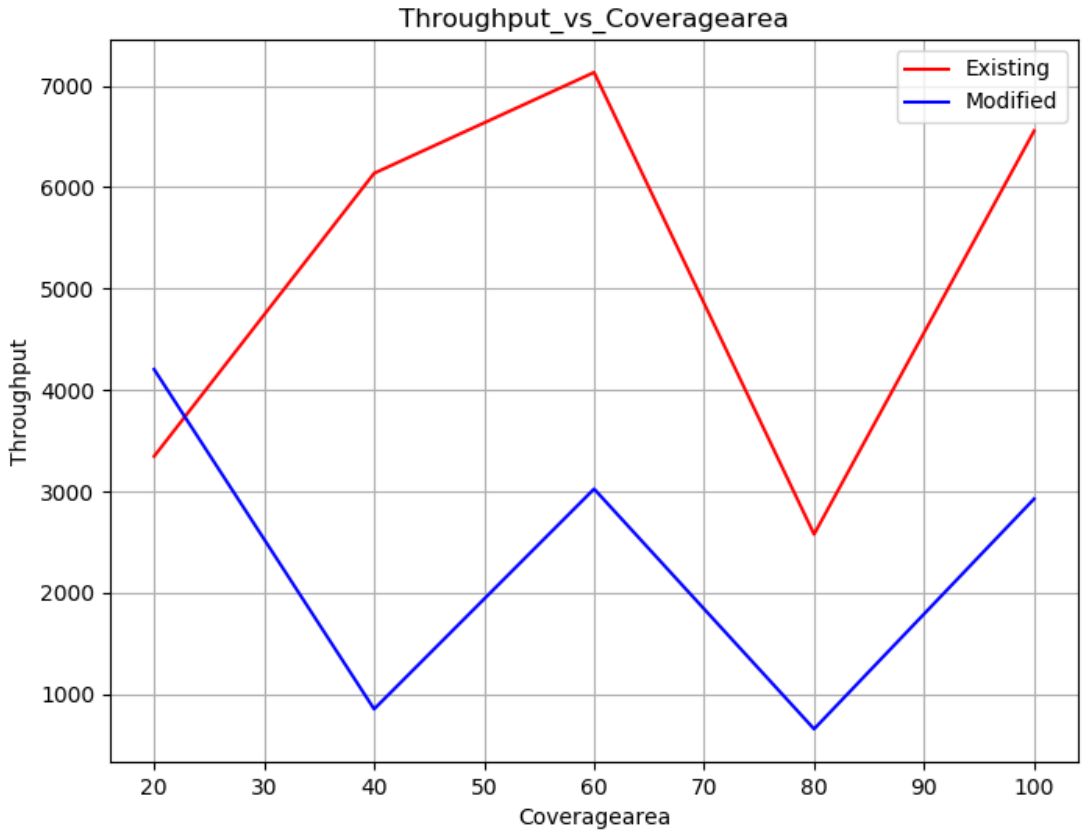
PLOTS(For 802_15_4 static):

Throughput:

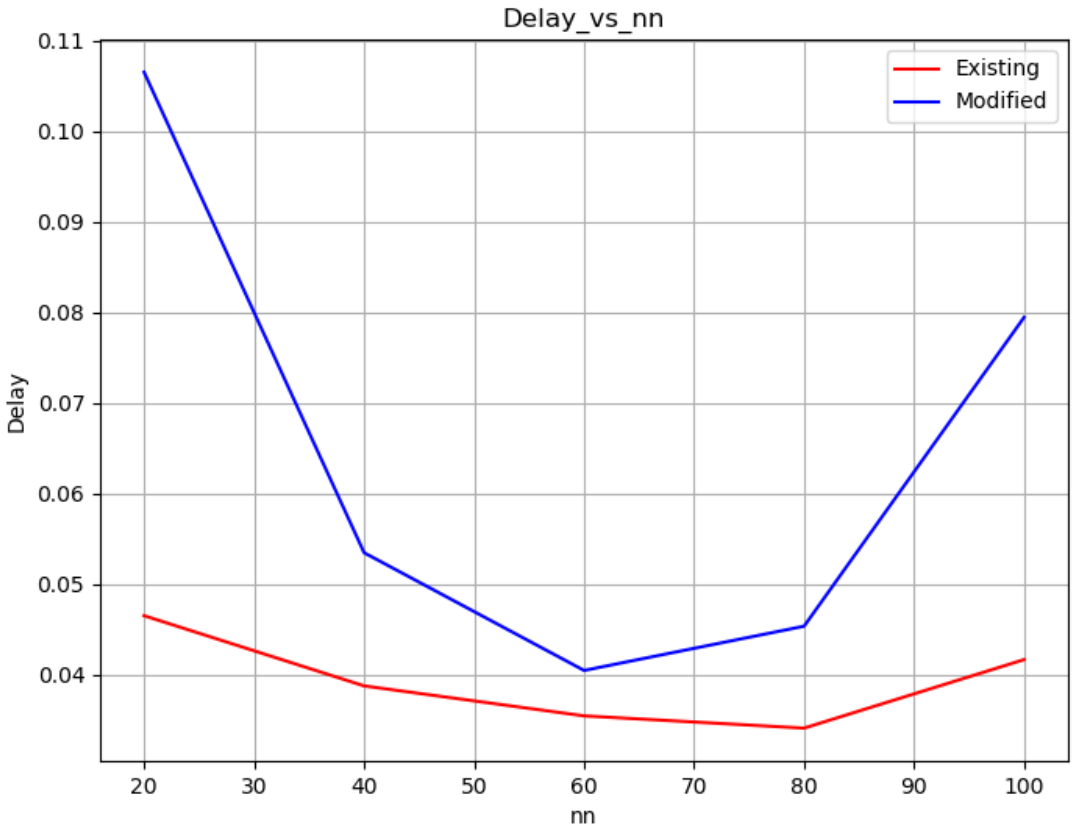


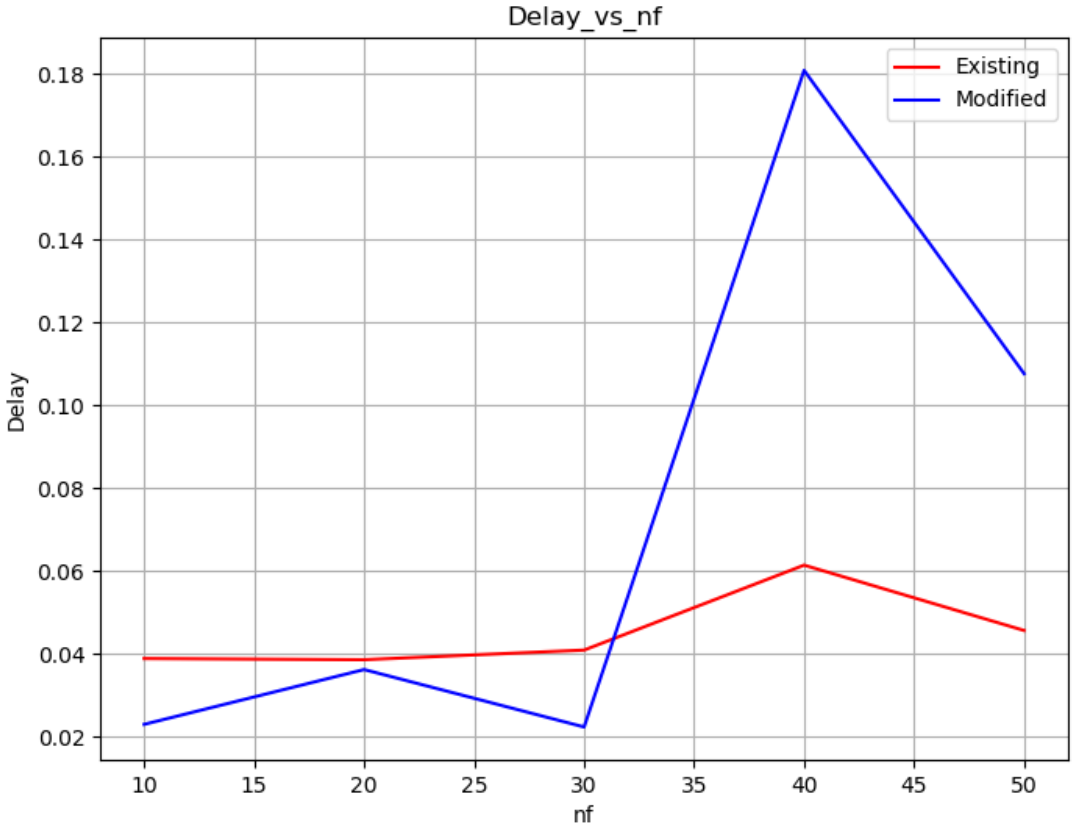

Throughput_vs_nf
Existing
Modified
Throughput
nf
3500
3000
2500
2000
1500
1000
500
10
15
20
25
30
35
40
45
50

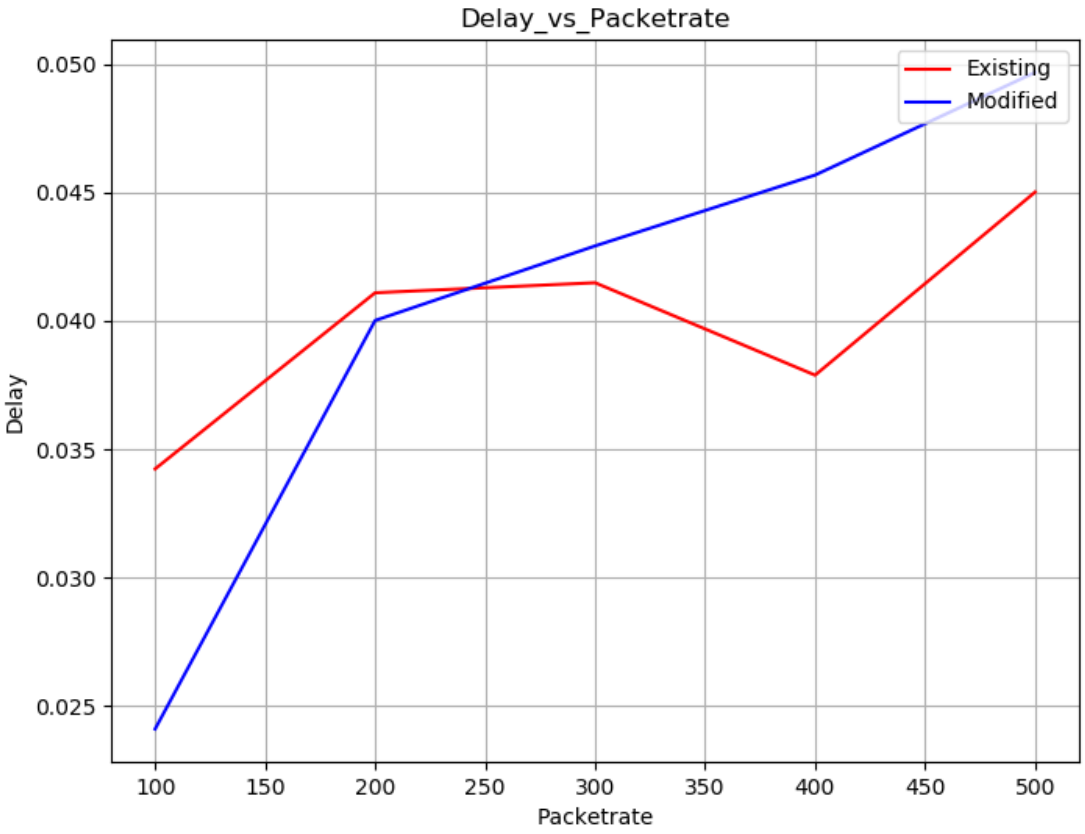

Throughput_vs_Packetrate
Existing
Modified
Throughput
Packetrate
3500
3000
2500
2000
1500
1000
500
100
150
200
250
300
350
400
450
500

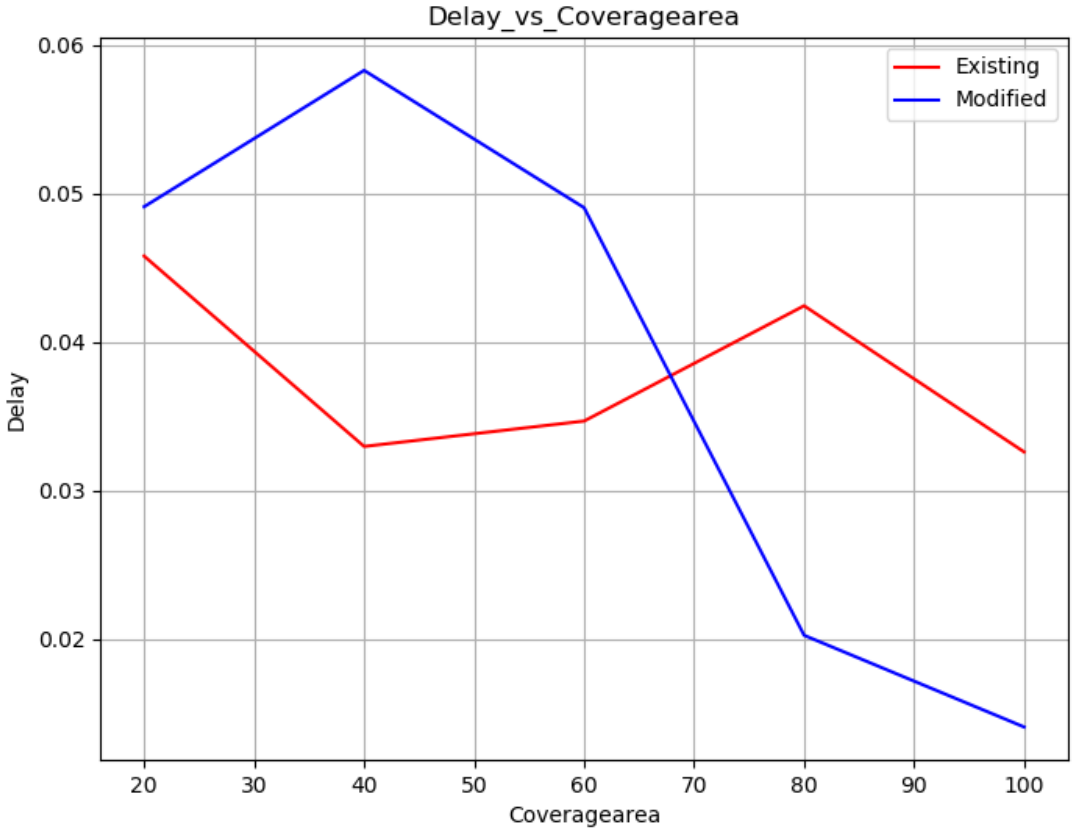

Throughput_vs_Coverageareа
Existing
Modified
Throughput
Coverageareа
7000
6000
5000
4000
3000
2000
1000
20
30
40
50
60
70
80
90
100

End to End Delay:

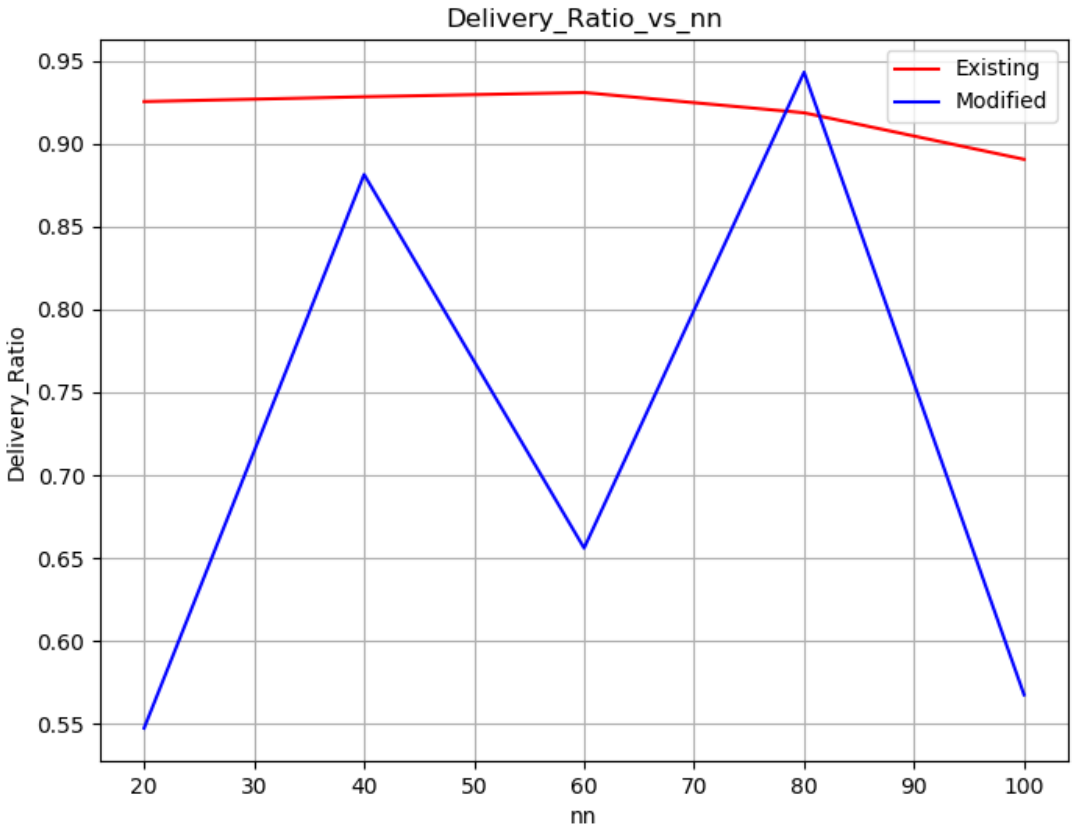


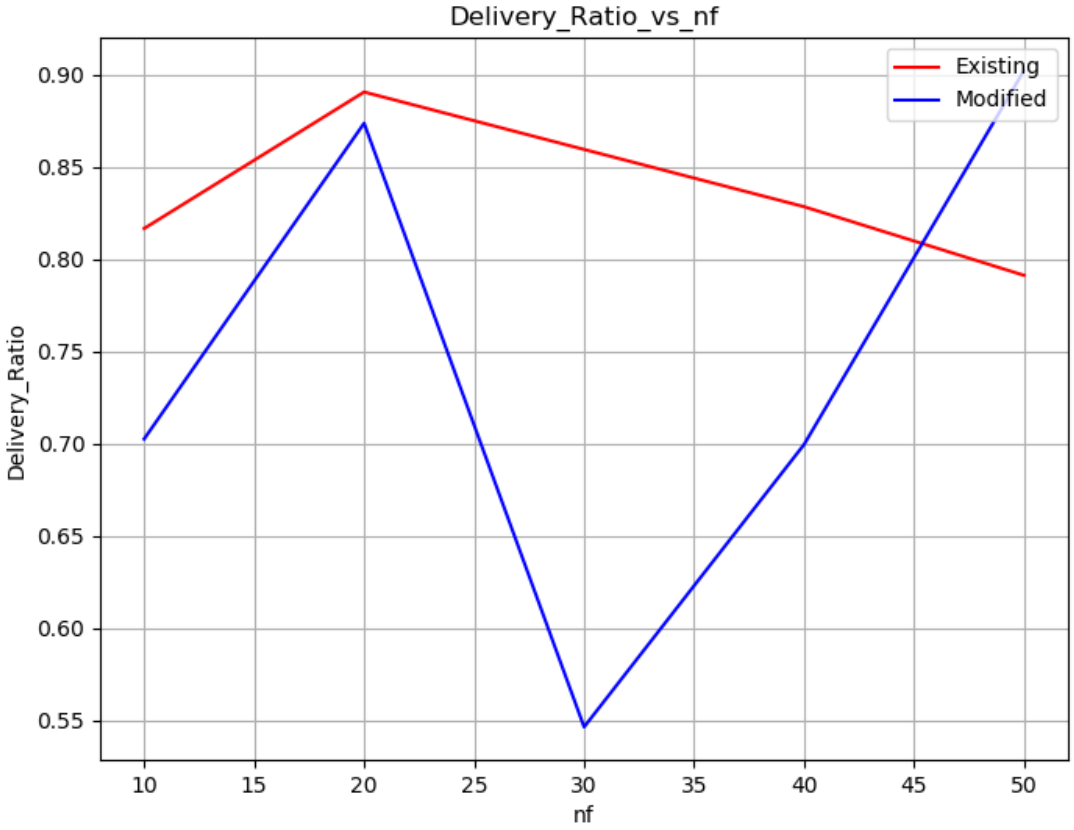

Delay_vs_nf
Existing
Modified
Delay
nf
0.18
0.16
0.14
0.12
0.10
0.08
0.06
0.04
0.02
10
15
20
25
30
35
40
45
50

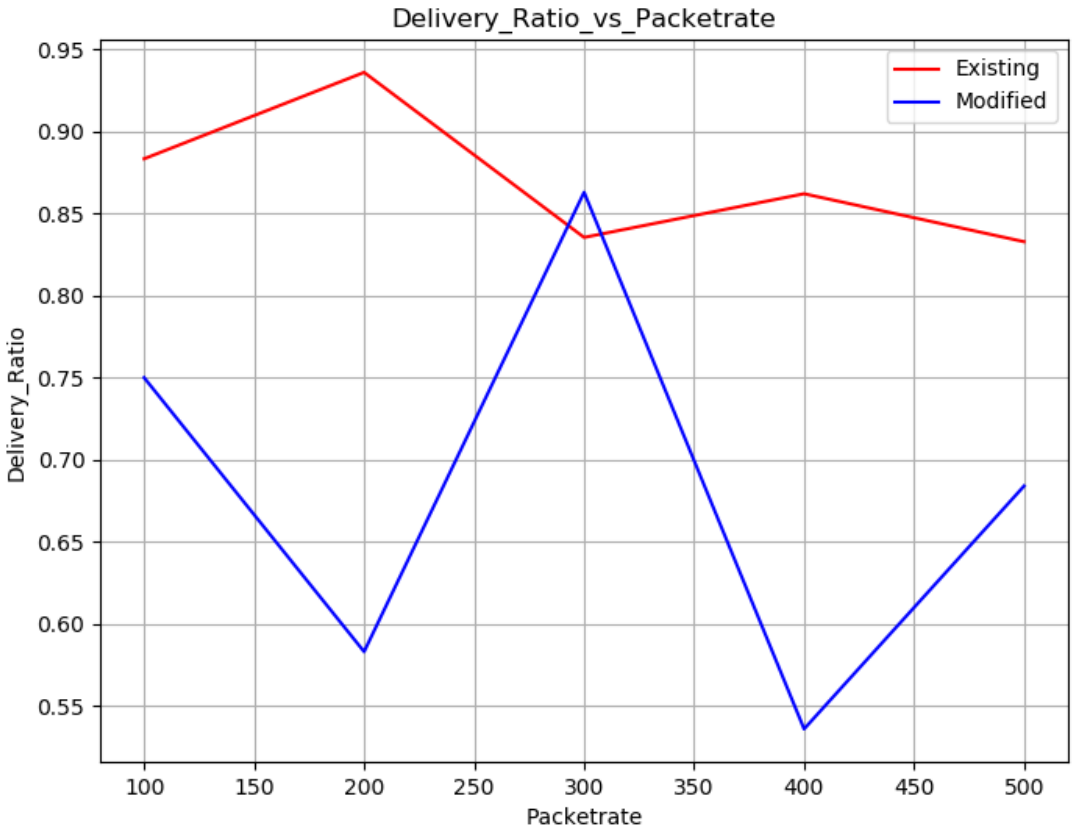

Delay_vs_Packetrate
Existing
Modified
Delay
Packetrate
0.050
0.045
0.040
0.035
0.030
0.025
100
150
200
250
300
350
400
450
500

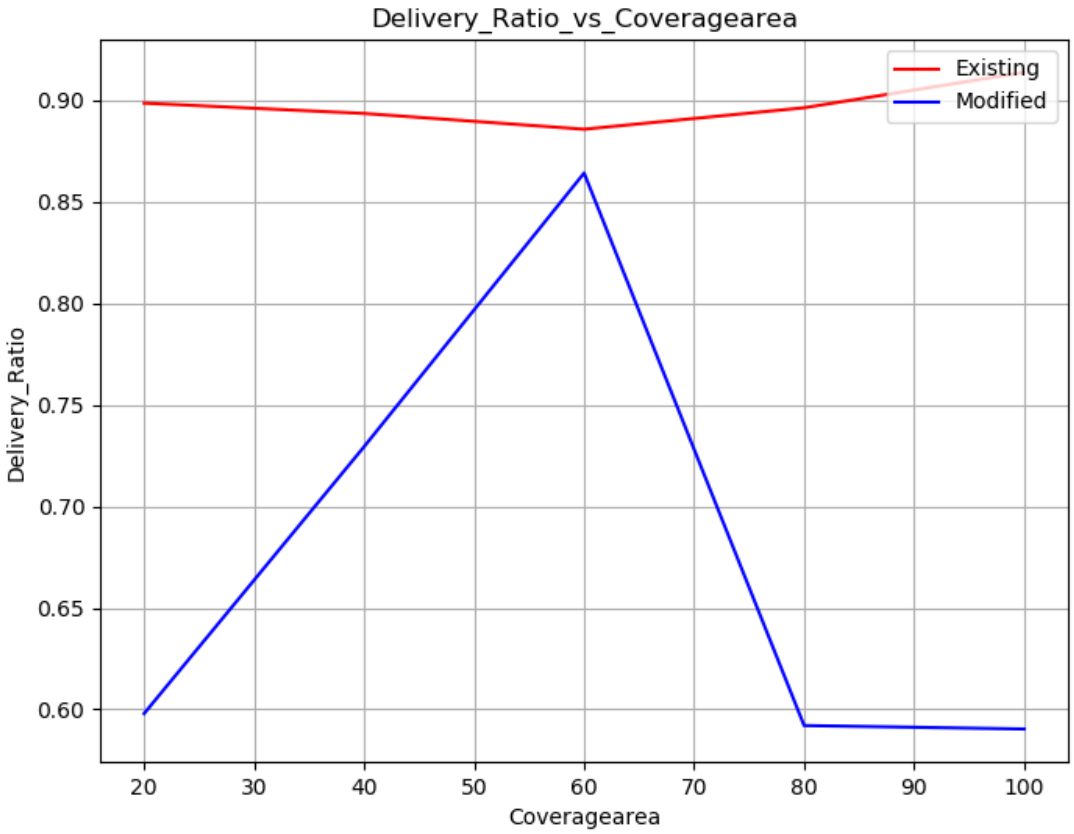

Delay_vs_Coveragearea
Existing
Modified
Delay
Coveragearea
0.06
0.05
0.04
0.03
0.02
20
30
40
50
60
70
80
90
100

Delivery Ratio:

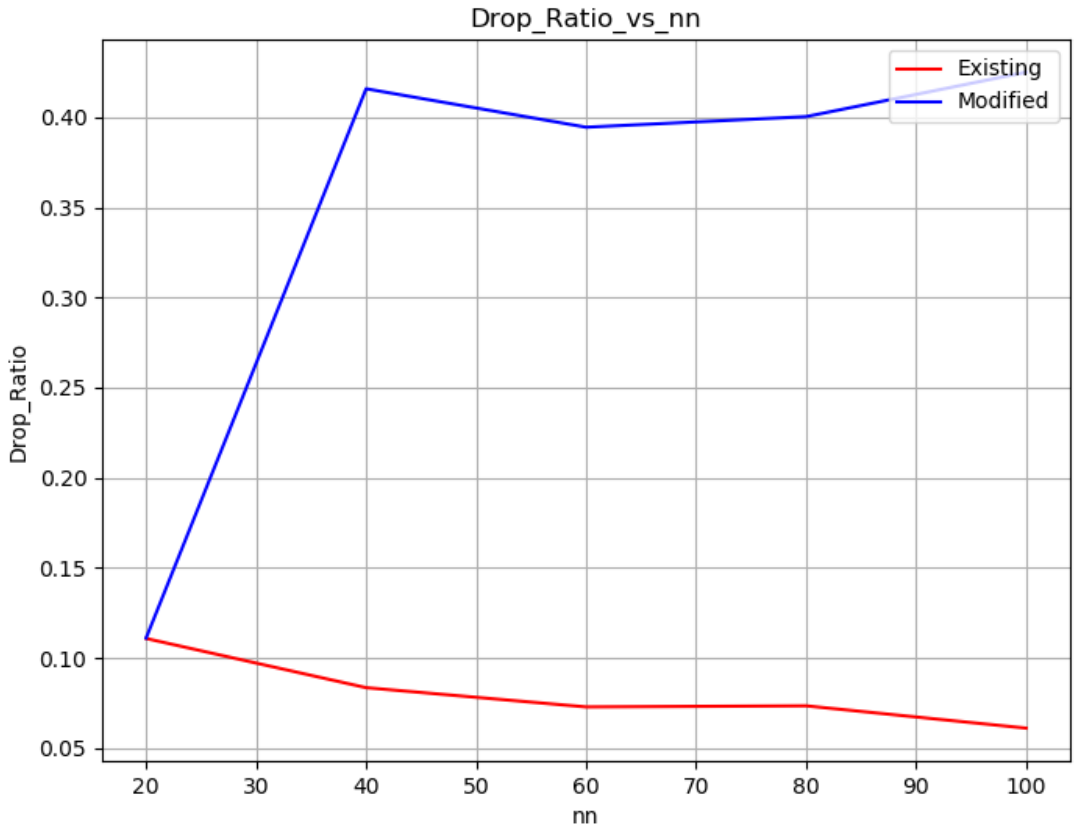


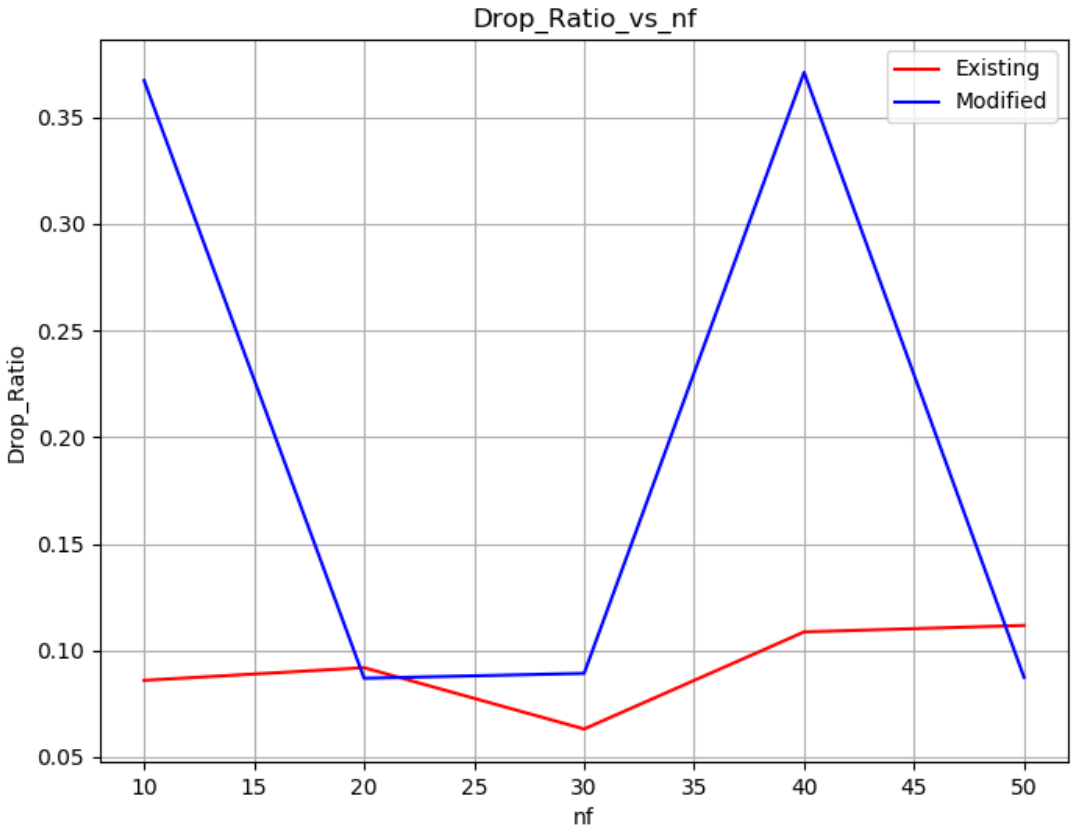

Delivery_Ratio_vs_nf
Existing
Modified
Delivery_Ratio
nf
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0.55
10
15
20
25
30
35
40
45
50

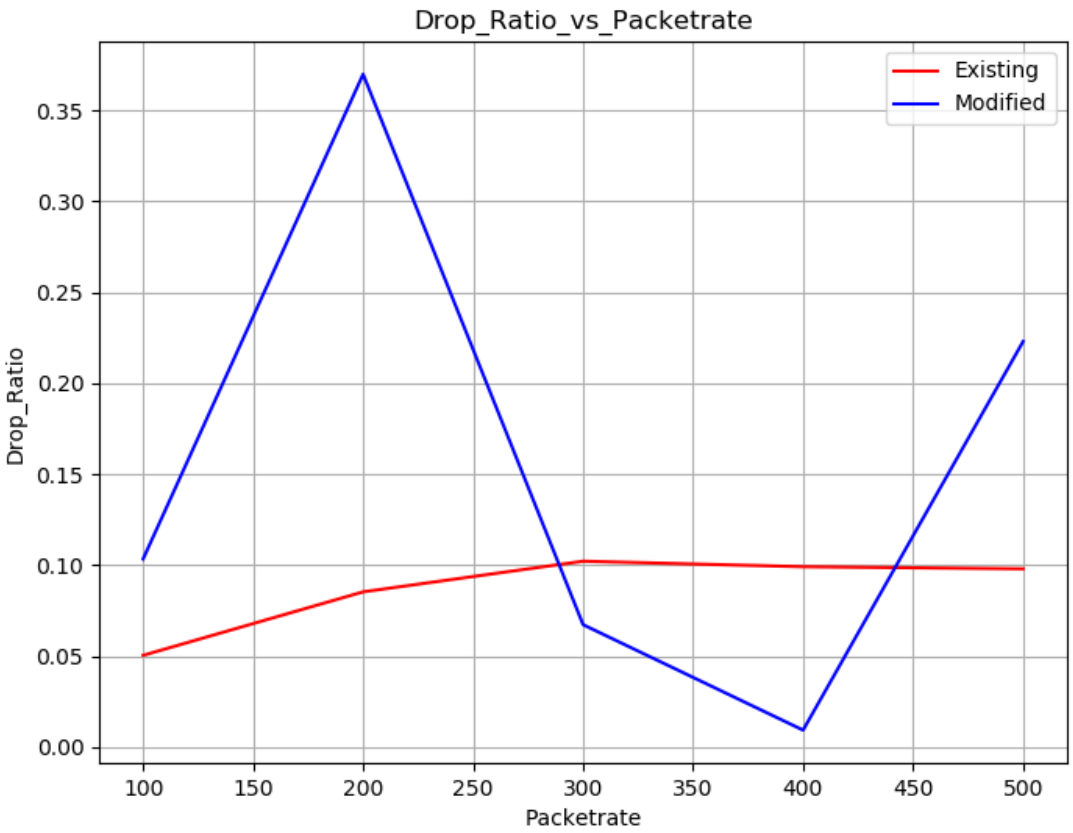

Delivery_Ratio_vs_Packetrate
Existing
Modified
Delivery_Ratio
Packetrate
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0.55
100
150
200
250
300
350
400
450
500

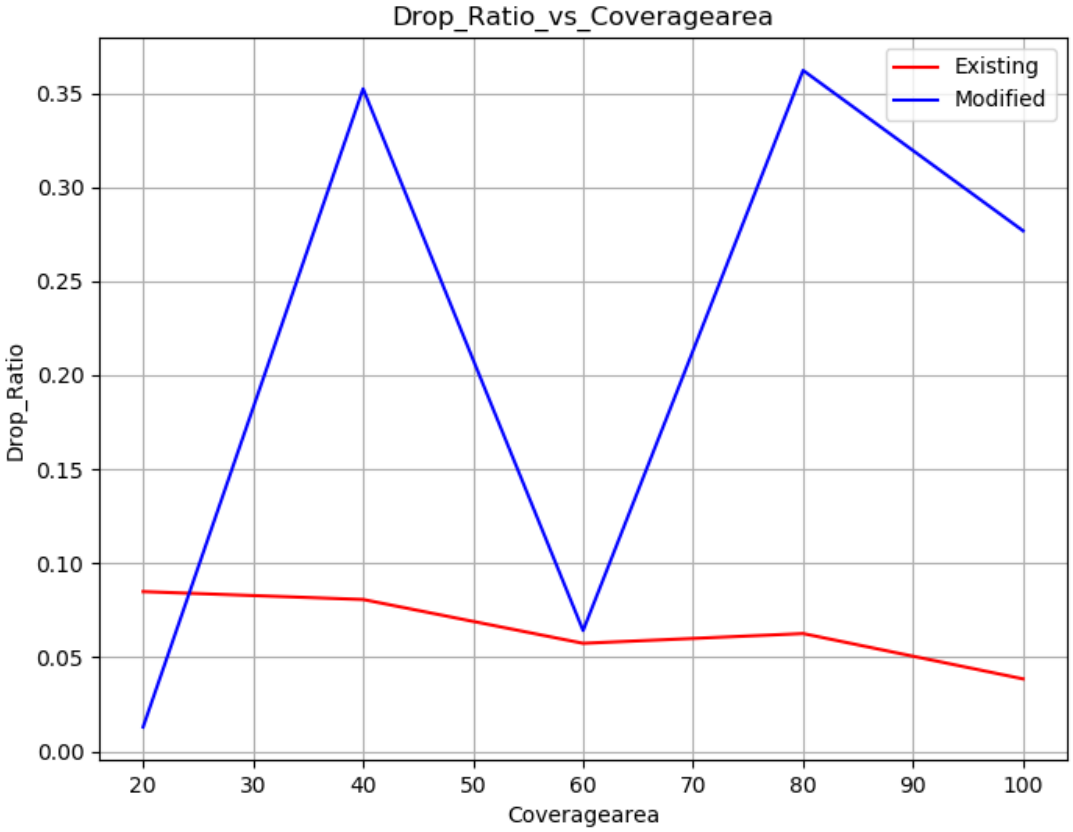

Delivery_Ratio_vs_Coveragearea
Existing
Modified
Delivery_Ratio
0.90
0.85
0.80
0.75
0.70
0.65
0.60
20
30
40
50
60
70
80
90
100
Coveragearea

Drop Ratio:


Drop_Ratio_vs_nn
Existing
Modified
Drop_Ratio
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
20
30
40
50
60
70
80
90
100
nn


Drop_Ratio_vs_nf
Existing
Modified
Drop_Ratio
0.35
0.30
0.25
0.20
0.15
0.10
0.05
10
15
20
25
30
35
40
45
50
nf


Drop_Ratio_vs_Packetrate
Existing
Modified
Drop_Ratio
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
100
150
200
250
300
350
400
450
500
Packetrate


Drop_Ratio_vs_Coveragearea
Existing
Modified
Drop_Ratio
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
20
30
40
50
60
70
80
90
100
Coveragearea

Summary:

In my modification,I wanted to stabilize the variation of congestion window size(cwnd) around a certain value.Because I thought stabilizing the congestion window size will improve the average throughput.But unfortunately this did not work.To stabilize the variation of congestion window I limited the increment of congestion window size to a certain value(the value of cwnd when very first packet drop was detected).

That's why on average more packet drops are occurred due to limited window size.That's why from the above figures we can observe that modified implementation has lower throughput, delivery ratio and higher drop ratio on average.