Report

NS2 Term Project: *TCP Vegas-A: Improving the Performance of TCP Vegas*

Date of Submission: 28th February 2023

Submitted by:

Sanjida Islam Era
1805116
L-3/T-2
CSE-B2

Introduction:

Congestion control is a crucial aspect of network performance, especially in wired and wireless networks, where resources are limited, and network traffic is high. Among the several new features implemented in TCP Vegas that was originally proposed by Brakmo et al., one of the most important differences between TCP Vegas and TCP Reno is its congestion avoidance scheme. While TCP Reno (and its variants like NewReno) rely on packet loss detection to detect network congestions, TCP Vegas uses a sophisticated bandwidth estimation scheme to proactively gauge network congestion. While it has been shown that TCP Vegas provides better performance compared to TCP Reno, studies have identified various issues associated with the protocol. In this project, we propose modifications to the congestion avoidance mechanism of the TCP Vegas to overcome these limitations which is proposed in the paper " TCP Vegas-A: Improving the Performance of TCP Vegas" by K.N. Srijith, Lillykutty Jacob, A.L. Ananda and a modification of this.

The proposed congestion control algorithm aims to improve TCP throughput and some other parameters like delivery ratio in wired and wireless networks by obtaining a fairer share of the network bandwidth in wired and satellite scenarios. The algorithm achieves this by dynamically adjusting the congestion window size of each TCP Vegas based on network conditions, and probing the network for available bandwidth. We have also modified this Vegas-A algorithm to achieve higher performance in wired networks.

To evaluate the performance change, we have compared the metrics of the modified algorithm to the existing algorithm of TCPVegas, and analysed the corresponding graphs subsequently.

Network Topologies:

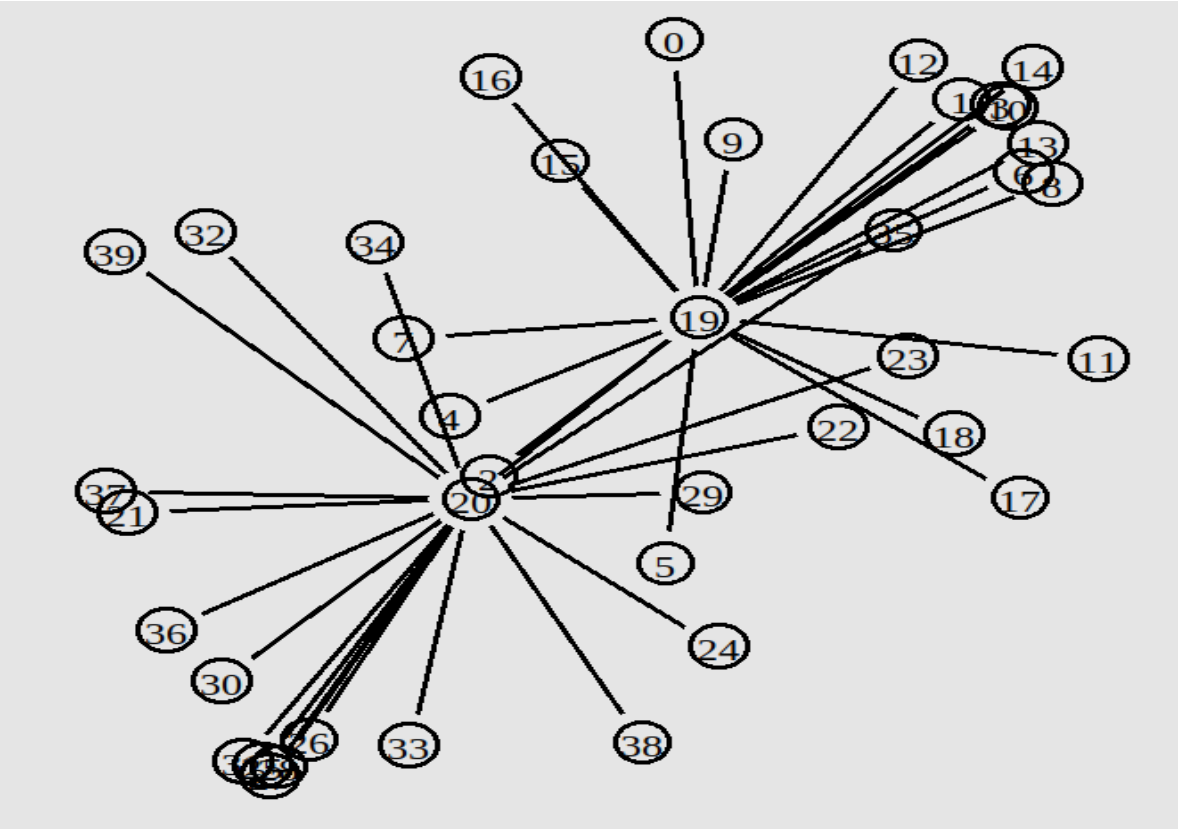
The network topology assigned to this project were:

1. Wired nodes
2. Wireless mobile nodes with MAC 802.11

Here the assigned topology was “Wireless mobile nodes with MAC 802.15.4” but it results results negative throughput. Since no possible solution to this problem was found at the time of concluding data collection, the agent type was switched to IEEE 802.11.

We have performed necessary simulations and derived results accordingly of these two kinds of networks. Visualisations of both the networks are as follows:

Wired nodes:



Wireless mobiles nodes:

Parameters under variation:

The parameters varied for the topologies in the simulation are as follows:

1. Number of nodes: 20, 40, 60, 80, 100
2. Number of flows: 10, 20, 30, 40, 50
3. Number of packets sent per second: 100, 200, 300, 400, 500
4. Speed of mobile nodes: 5 m/s, 10 m/s, 15 m/s, 20 m/s, 25 m/s
   (applicable for the simulation involving mobile nodes in wireless network)

Modifications made:

The modification of this project is regarding the Congestion Avoidance mechanism of TCP Vegas. The reference paper presents modification to TCP Vegas and refers to the modified algorithm as TCP Vegas-A, where 'A' stands for adaptive. TCP Vegas uses fixed values for variables a and b, set to 1 and 3 usually. Recall that Vegas's strategy is to adjust the source's congestion window in an attempt to keep a small number of packets buffered in the routers along the path. Still subscribing to the idea that the average number of packets in the router buffer is to be kept within a and b, the main idea behind TCP Vegas-A is that rather than fixing a and b, they be made dynamically changeable, adaptive. At the start of a connection, a is set to 1 and b to 3. These values are then changed dynamically depending on the network conditions. Another way of looking at this modification is that we are trying to bring the network probing capability of TCP Reno into TCP Vegas

The congestion control in TCP Agents are divided into three phases:

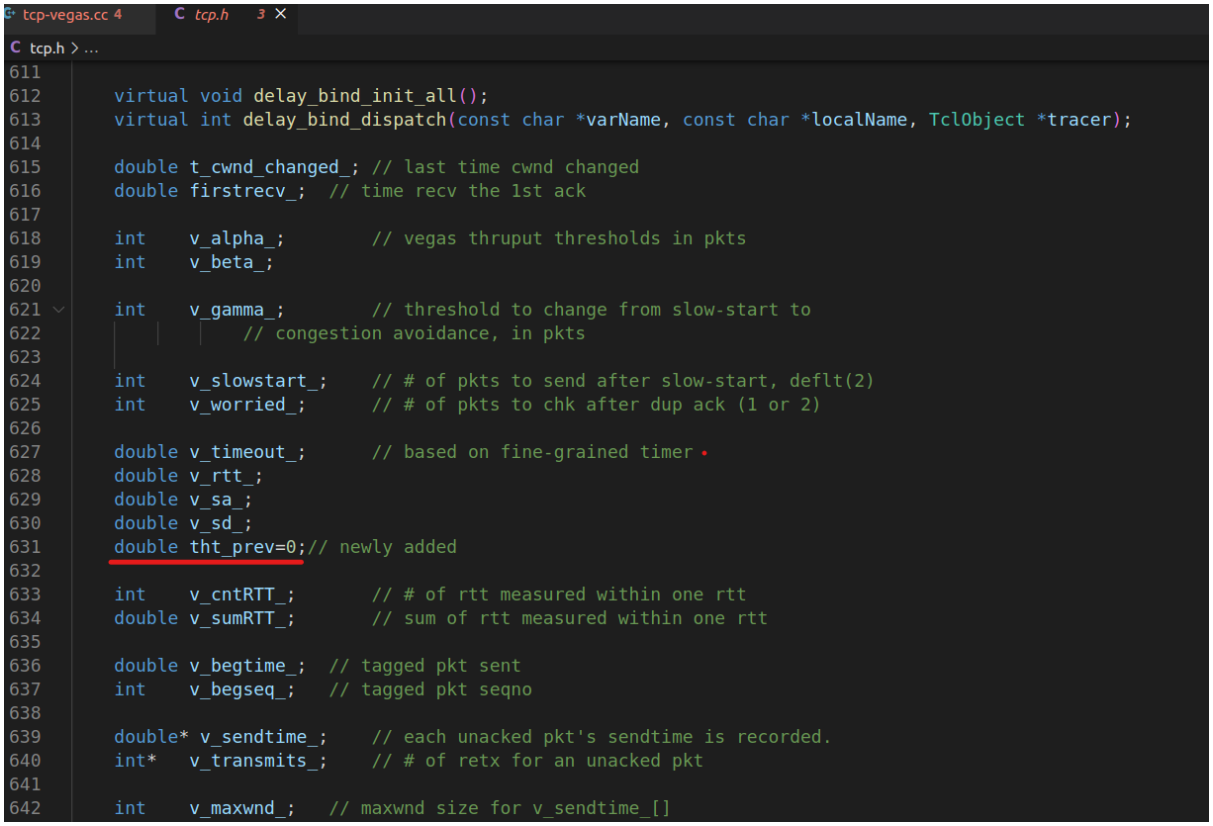
1. Slow start: The initial phase of TCP connection establishment in which the sender starts transmitting data to the receiver with a conservative transmission rate. During this phase, the sender increases its transmission rate exponentially until it detects congestion in the network.
2. Congestion Avoidance: The slow start threshold (ssthresh) is reached, and we increase the congestion window according to the protocol.
3. Congestion Control: Congestion is detected, and similar to avoidance, a procedure is followed to decrease the window to decrease the loss due to congestion.

These phases are handled by two functions in ns2, namely opencwnd() and slowdown().While slow start and congestion recovery algorithms of Vegas-A are the same as that of Vegas, we propose a modified congestion avoidance mechanism.

I implemented the following changes:

In tcp.h:

1. New variables for optimal congestion window calculation:
   Specially for calculating the



In tcp-vegas.cc:

1. Modified functions to run when modifications are not enabled:
   Mainly I have done this modification in recv() function of tcp-vegas.cc.I have added an extra part in the congestion avoidance section. For this I have declared some parameters to calculate current throughput using current rtt, and previous throughput by the previous declared parameter tht_prev. Other local variables are given:

```
// expect = (current window size)/baseRTT
expect = double(t_seqno_-last_ack_)/v_baseRTT_;
double new_actual = double(t_seqno_-last_ack_)/rtt;


// calc actual and expect thruput diff, delta
int delta=int((expect-v_actual_)*v_baseRTT_+0.5);
double delta_check = (expect-new_actual);
int cwnd_prev = cwnd_;
int ssthresh_prev = ssthresh_;
```

2. The main modification for Vegas-A:

```
tcp-vegas.cc 3 ×   tcp.h 3
tcp-vegas.cc > recv(Packet *, Handler *)
817                 // printf("actual - %f\n", new_actual);
818                 // printf("expected: %f\n", expect);
819
820                 if(cwnd_prev >= ssthresh_prev){
821                     if(delta_check < v_beta_ && delta_check > v_alpha_){
822                         if(new_actual>tht_prev){
823                             v_alpha_=v_alpha_+1;
824                             v_beta_=v_beta_+1;
825                         }
826                     }
827                     else if(delta_check<v_alpha_){
828                         if(v_alpha_>1){
829                             if(new_actual < tht_prev){
830                                 v_alpha_=v_alpha_-1;
831                                 v_beta_=v_beta_-1;
832                             }
833                         }
834                     }
835                     else if(delta_check>v_beta_){
836                         if(v_alpha_>1){
837                             v_alpha_=v_alpha_-1;
838                             v_beta_=v_beta_-1;
839                         }
840                     }
841                     else{
842
843                     }
844                 }
845             } // end of if(rtt > 0)
```

3. Another approach in modification for getting better result in wired topology (by modifying the Vegas-A algorithm):

```
tcp-vegas.cc 3 ×   tcp.h 3
tcp-vegas.cc > recv(Packet *, Handler *)
817                 // printf("actual - %f\n", new_actual);
818                 // printf("expected: %f\n", expect);
819
820                 if(cwnd_prev >= ssthresh_prev){
821                     if(delta_check < v_beta_ && delta_check > v_alpha_){
822                         if(new_actual>tht_prev){
823                             v_alpha_=v_alpha_+1;
824                             v_beta_=v_beta_+1;
825                         }
826                     }
827                     else if(delta_check<v_alpha_){
828                         if(v_alpha_>1){
829                             if(new_actual < tht_prev){
830                                 v_alpha_=v_alpha_-1;
831                             }
832                         }
833                     }
834                     else if(delta_check>v_beta_){
835                         if(v_alpha_+1 < v_beta_){
836                             v_beta_=v_beta_-1;
837                         }
838                     }
839                     else{
840
841                     }
842                 }
843             } // end of if(rtt > 0)
844
```

4. Finally setting the throughput as previous throughput for next rtt:

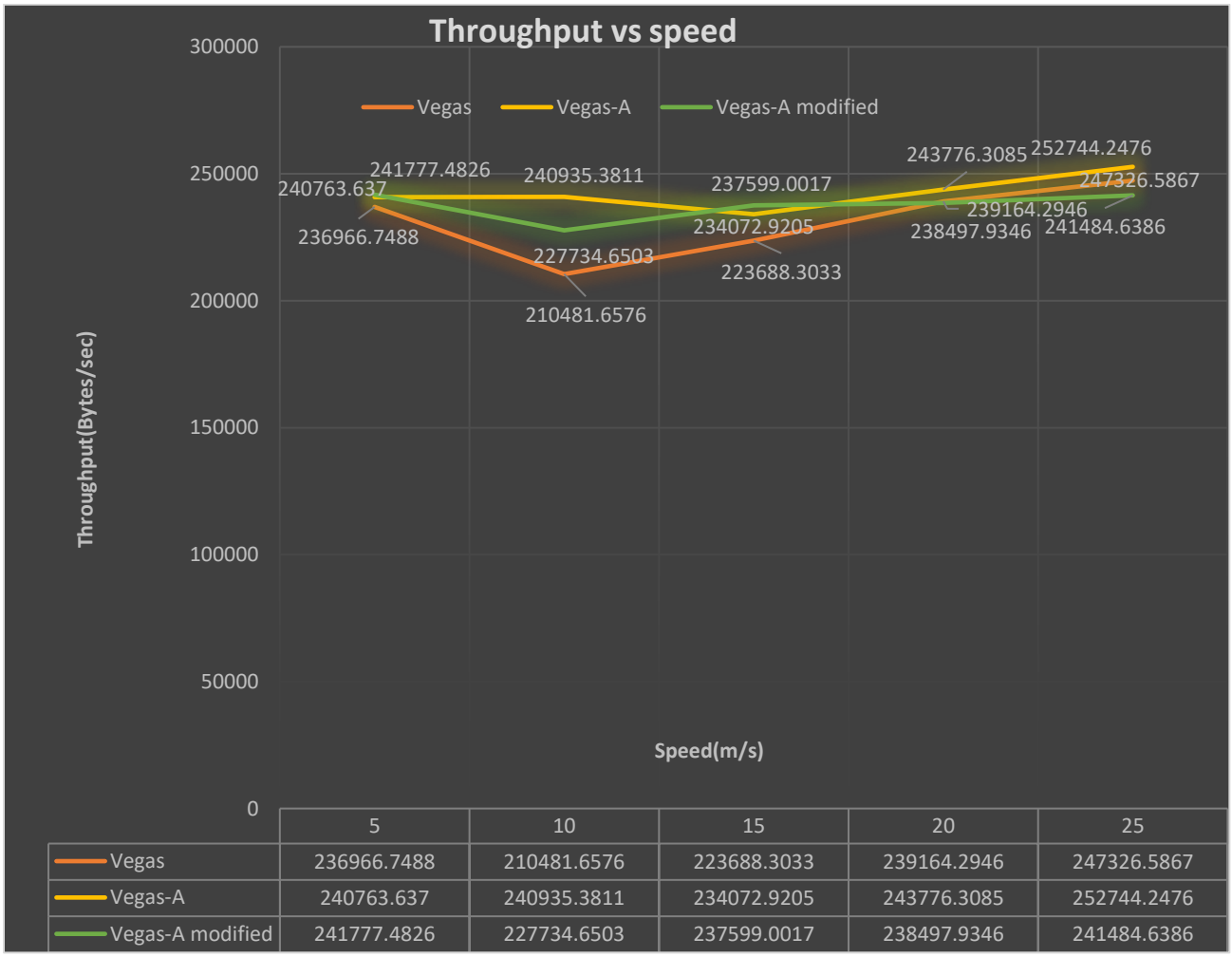
```
861                 if(rtt>0) tht_prev=v_actual_;
862                 else tht_prev=0;
```

Results with graphs:

1. Wireless 802.11 Mobile nodes:

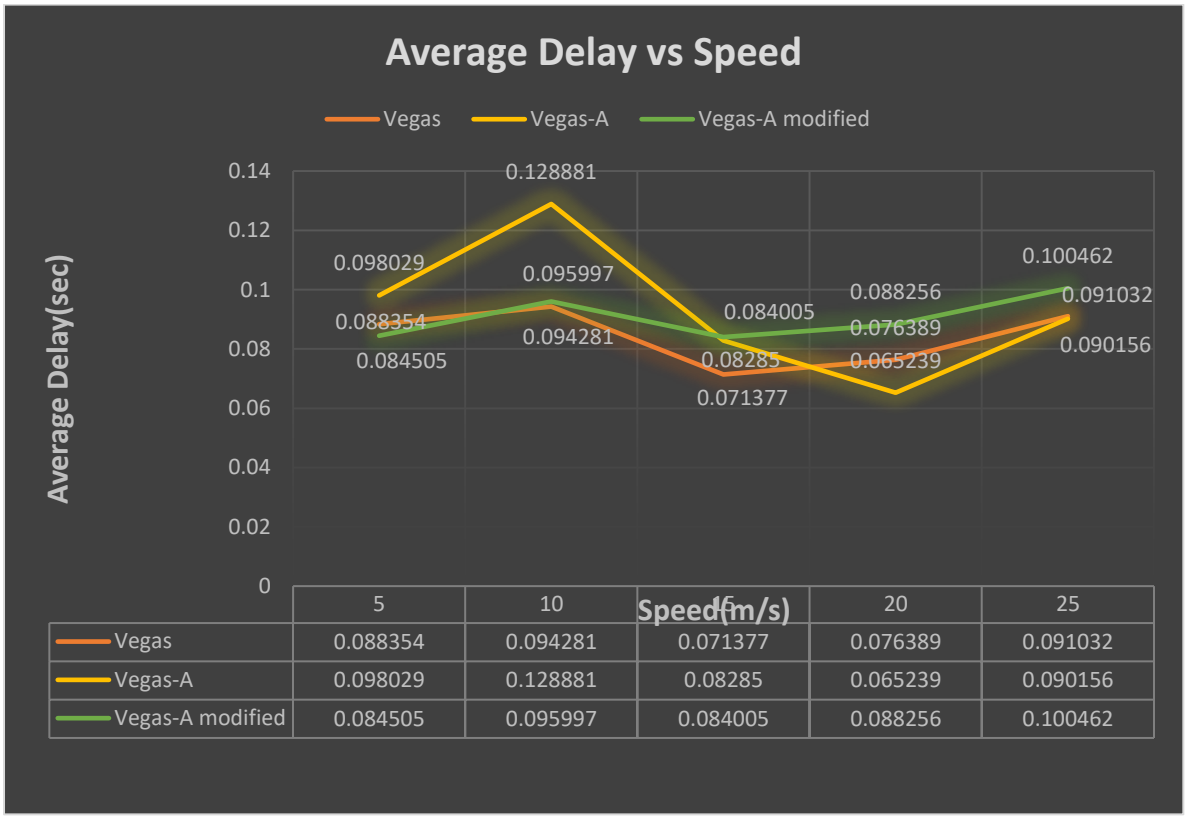
With respect to Node Speed:

Network Throughput:



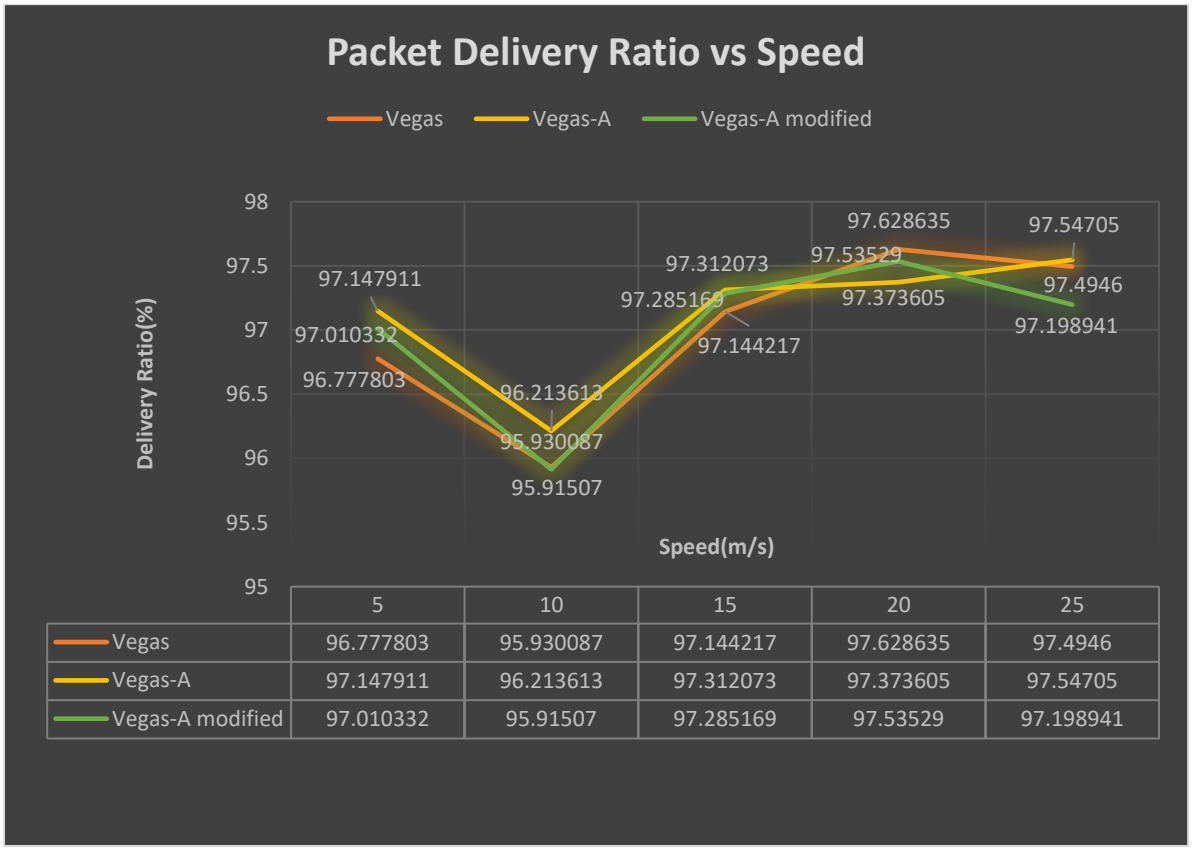
	5	10	15	20	25
Vegas	236966.7488	210481.6576	223688.3033	239164.2946	247326.5867
Vegas-A	240763.637	240935.3811	234072.9205	243776.3085	252744.2476
Vegas-A modified	241777.4826	227734.6503	237599.0017	238497.9346	241484.6386

End-to-End Delay:



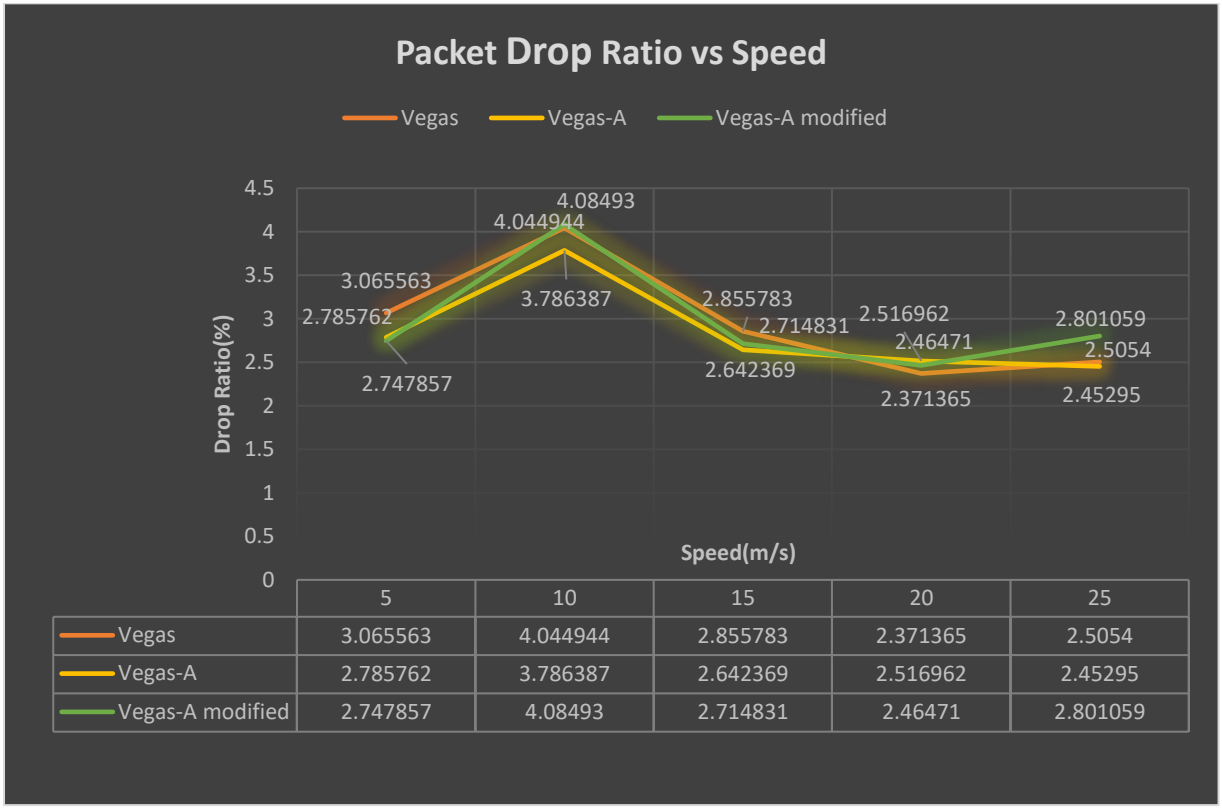
	5	10	15	20	25
Vegas	0.088354	0.094281	0.071377	0.076389	0.091032
Vegas-A	0.098029	0.128881	0.08285	0.065239	0.090156
Vegas-A modified	0.084505	0.095997	0.084005	0.088256	0.100462

Packet Delivery Ratio:



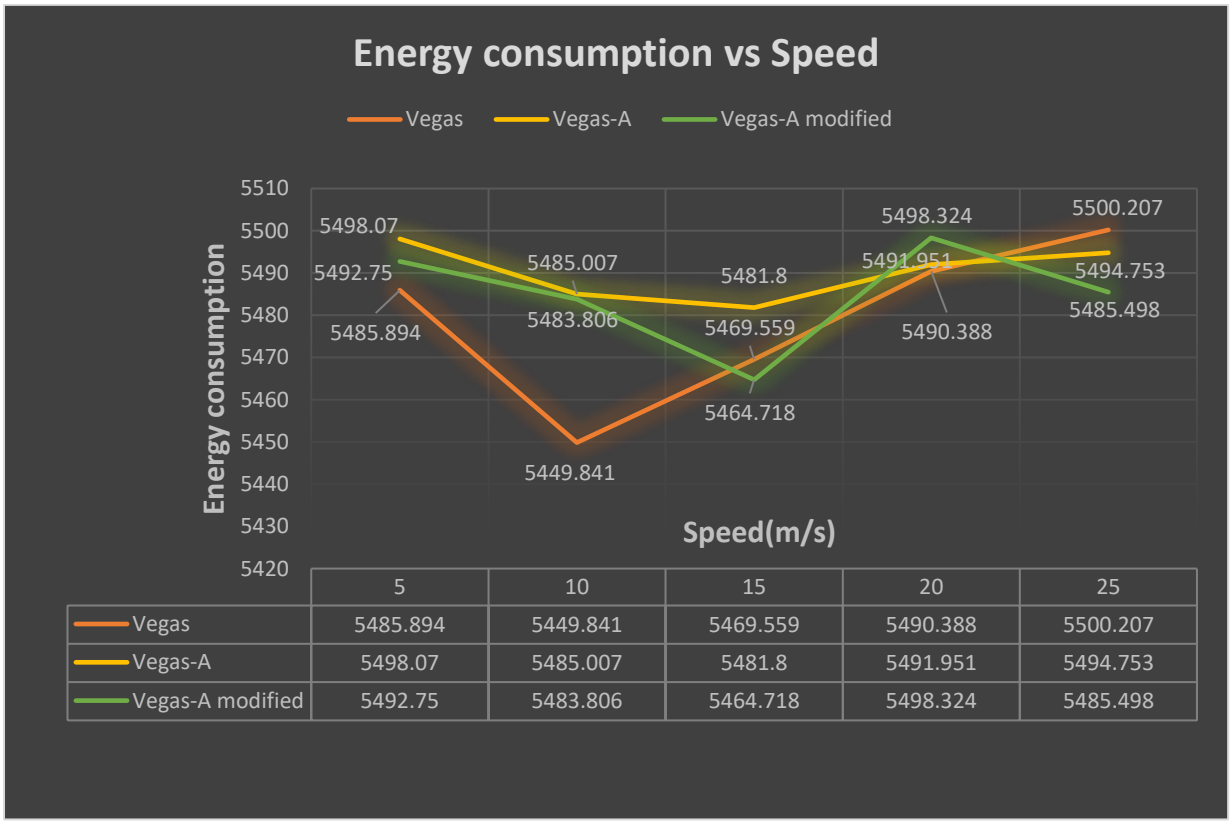
	5	10	15	20	25
Vegas	96.777803	95.930087	97.144217	97.628635	97.4946
Vegas-A	97.147911	96.213613	97.312073	97.373605	97.54705
Vegas-A modified	97.010332	95.91507	97.285169	97.53529	97.198941

Packet Drop Ratio:



	5	10	15	20	25
Vegas	3.065563	4.044944	2.855783	2.371365	2.5054
Vegas-A	2.785762	3.786387	2.642369	2.516962	2.45295
Vegas-A modified	2.747857	4.08493	2.714831	2.46471	2.801059

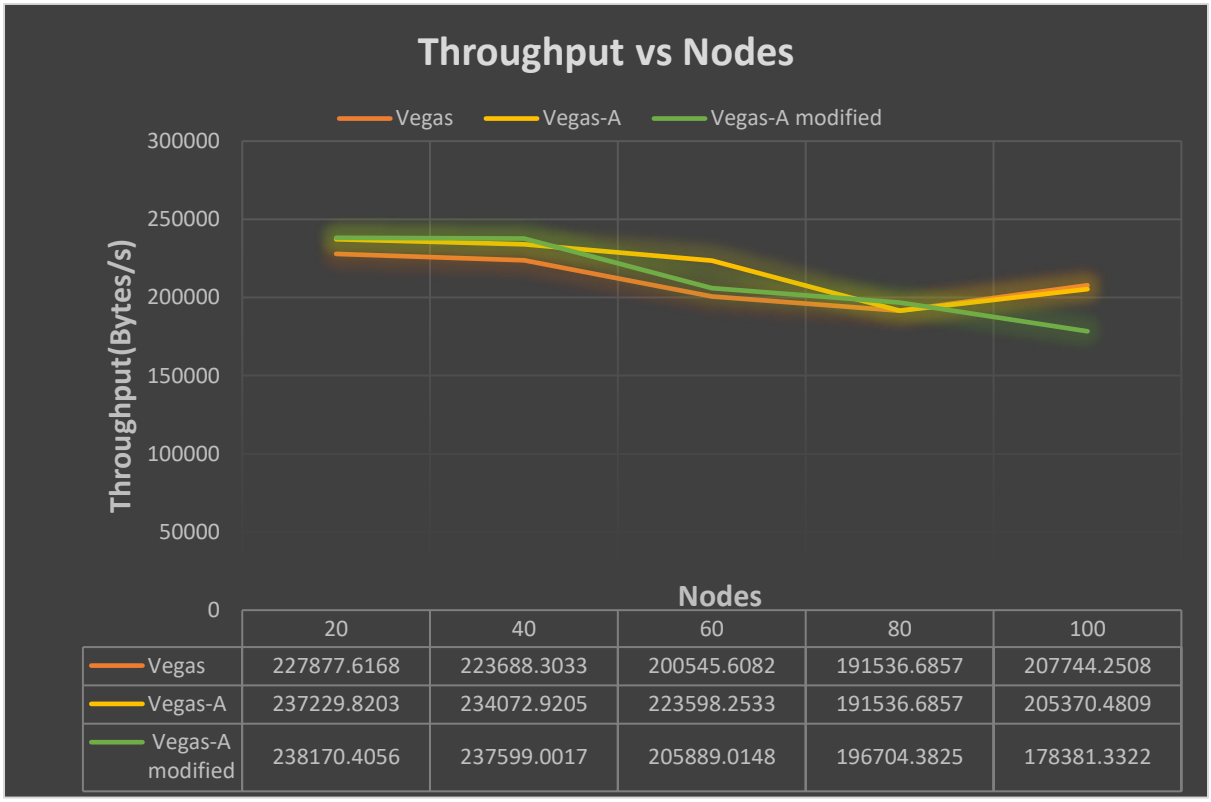
Energy Consumption:



	5	10	15	20	25
Vegas	5485.894	5449.841	5469.559	5490.388	5500.207
Vegas-A	5498.07	5485.007	5481.8	5491.951	5494.753
Vegas-A modified	5492.75	5483.806	5464.718	5498.324	5485.498

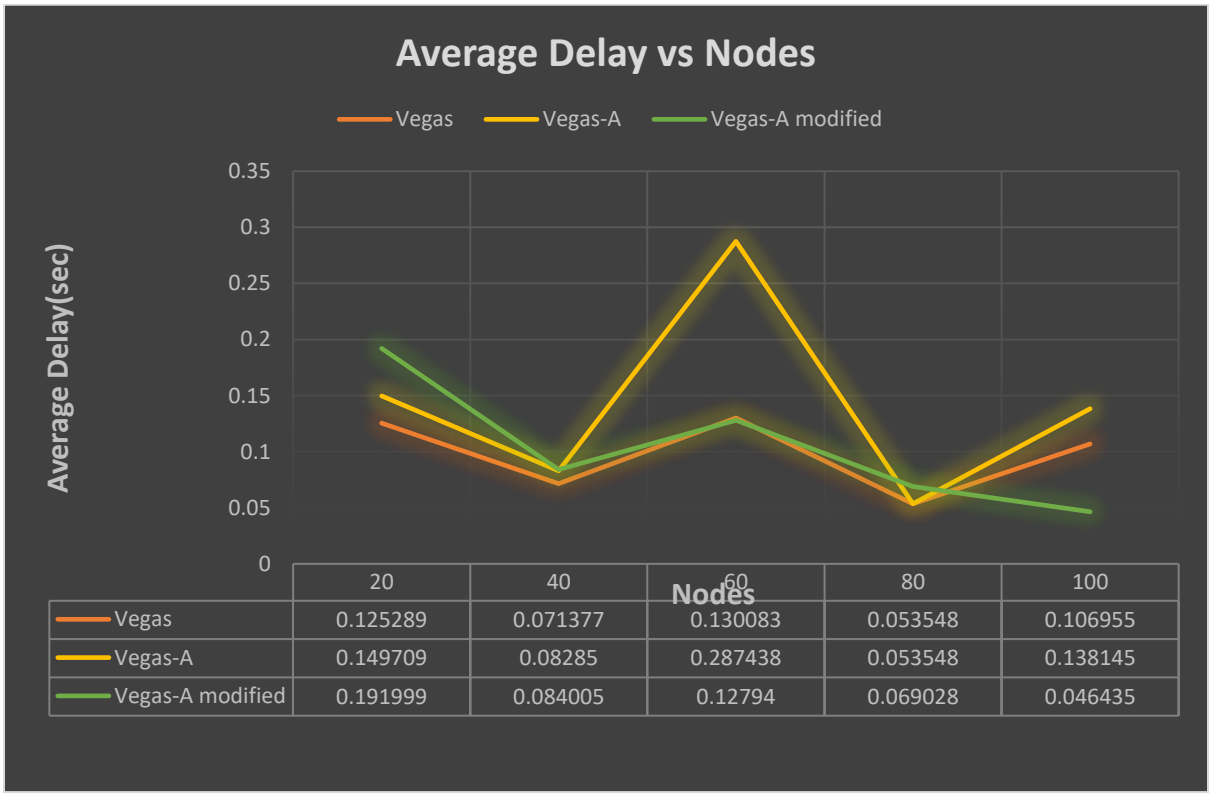
With respect to Number of Nodes:

Network Throughput:



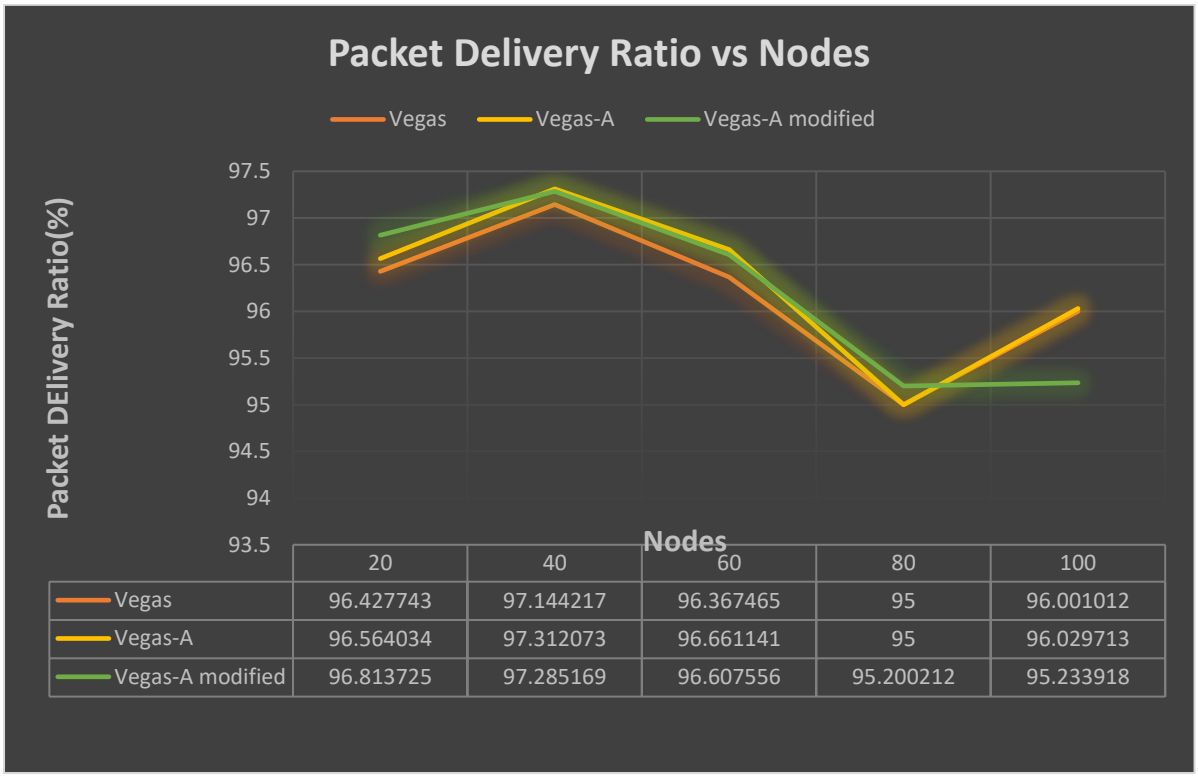
	20	40	60	80	100
Vegas	227877.6168	223688.3033	200545.6082	191536.6857	207744.2508
Vegas-A	237229.8203	234072.9205	223598.2533	191536.6857	205370.4809
Vegas-A modified	238170.4056	237599.0017	205889.0148	196704.3825	178381.3322

End-to-End Delay:



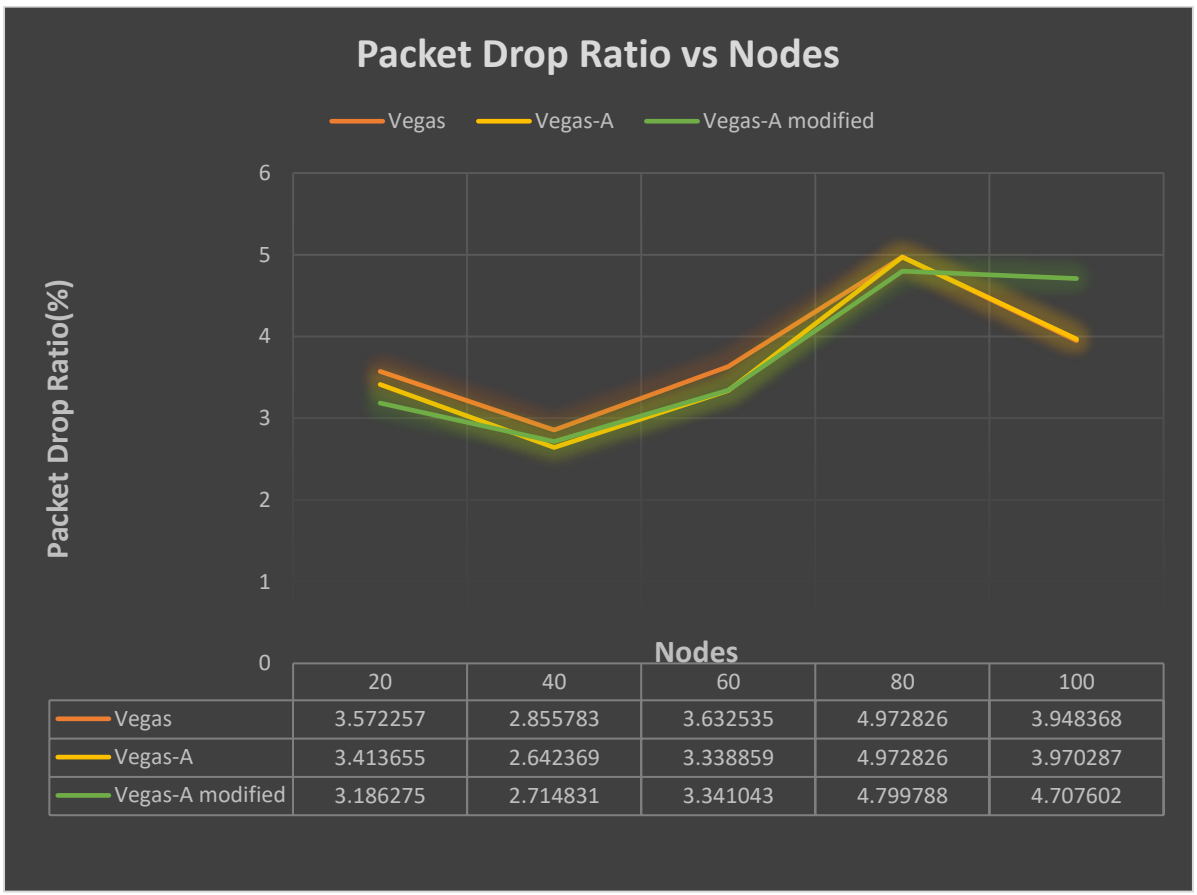
	20	40	60	80	100
Vegas	0.125289	0.071377	0.130083	0.053548	0.106955
Vegas-A	0.149709	0.08285	0.287438	0.053548	0.138145
Vegas-A modified	0.191999	0.084005	0.12794	0.069028	0.046435

Packet Delivery Ratio:



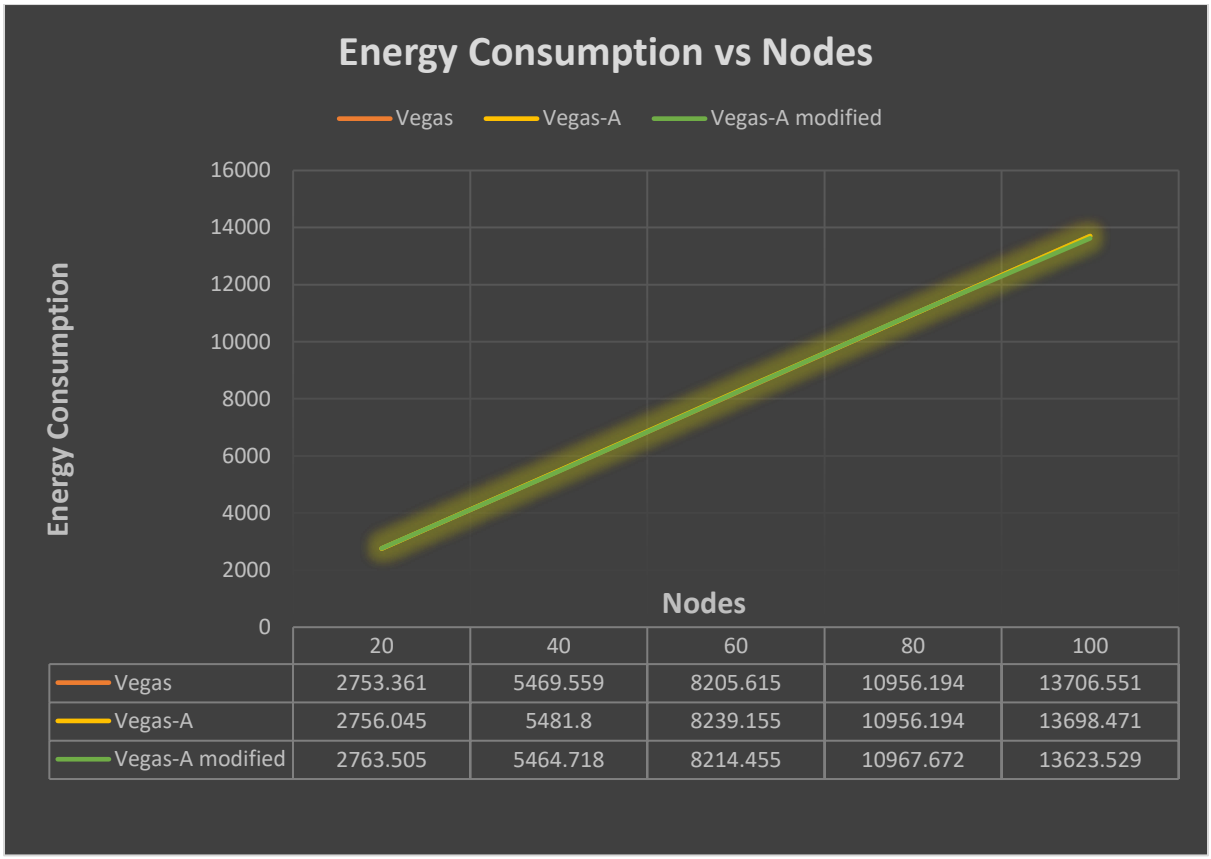
	20	40	60	80	100
Vegas	96.427743	97.144217	96.367465	95	96.001012
Vegas-A	96.564034	97.312073	96.661141	95	96.029713
Vegas-A modified	96.813725	97.285169	96.607556	95.200212	95.233918

Packet Drop Ratio:



	20	40	60	80	100
Vegas	3.572257	2.855783	3.632535	4.972826	3.948368
Vegas-A	3.413655	2.642369	3.338859	4.972826	3.970287
Vegas-A modified	3.186275	2.714831	3.341043	4.799788	4.707602

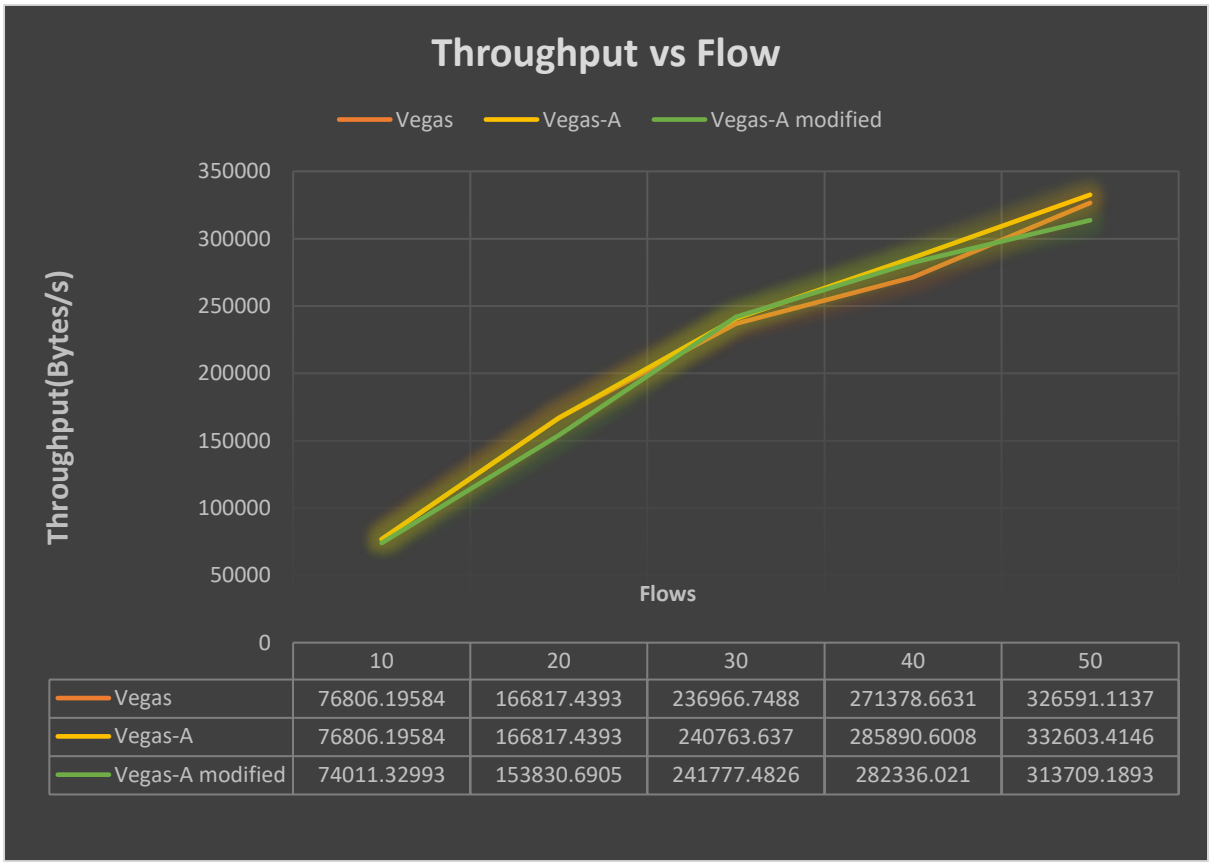
Energy Consumption:



	20	40	60	80	100
Vegas	2753.361	5469.559	8205.615	10956.194	13706.551
Vegas-A	2756.045	5481.8	8239.155	10956.194	13698.471
Vegas-A modified	2763.505	5464.718	8214.455	10967.672	13623.529

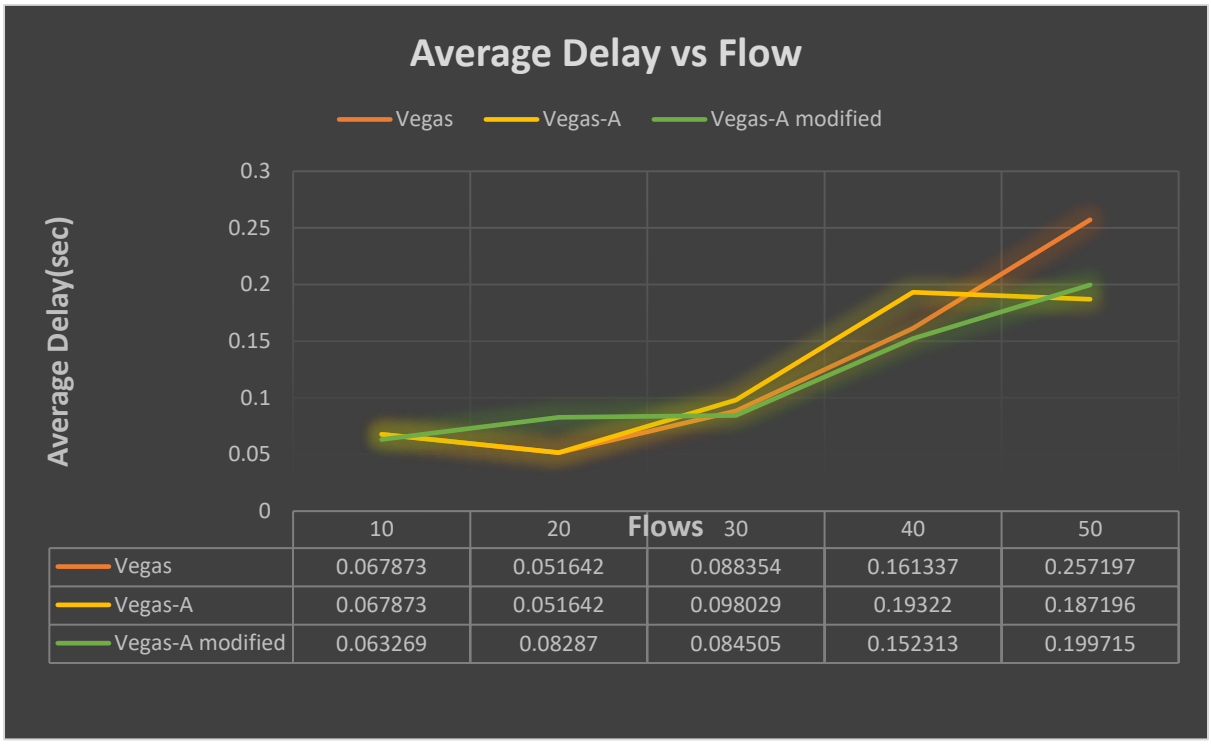
With respect to Number of Flows:

Network Throughput:



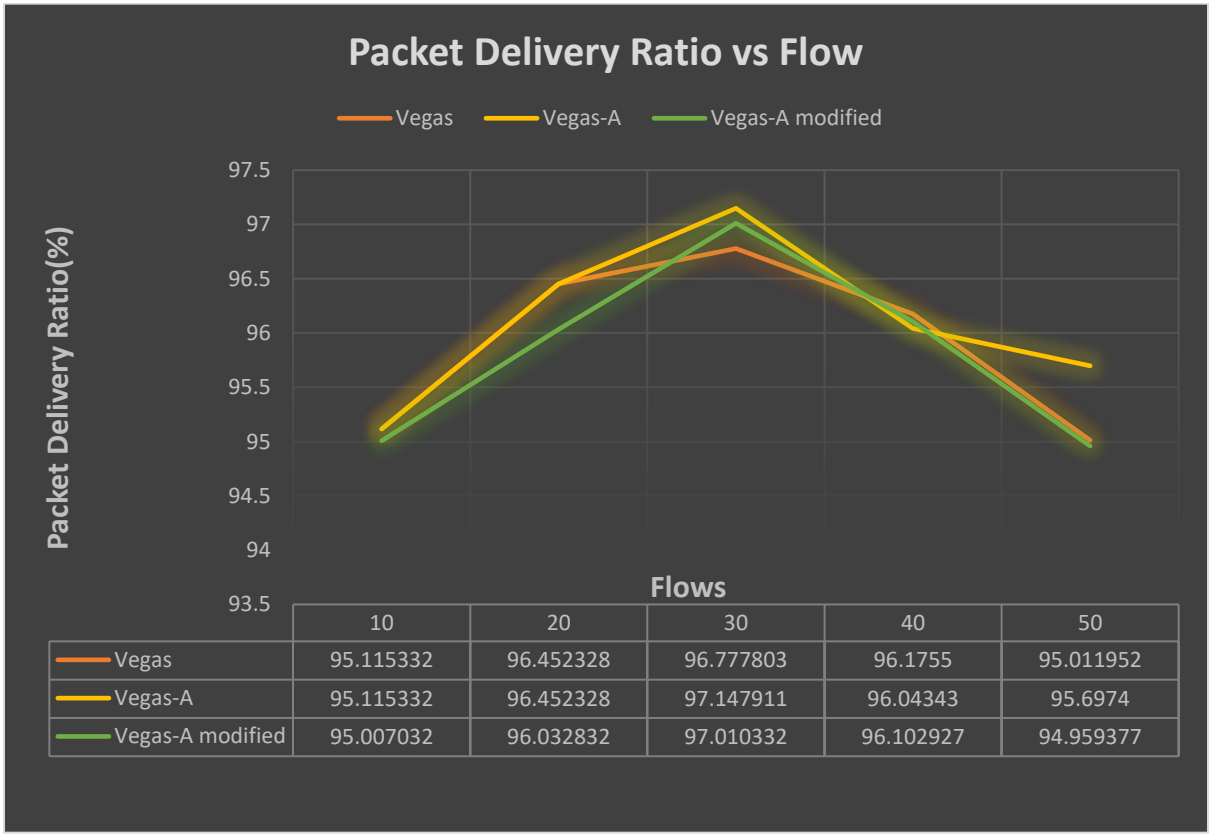
	10	20	30	40	50
Vegas	76806.19584	166817.4393	236966.7488	271378.6631	326591.1137
Vegas-A	76806.19584	166817.4393	240763.637	285890.6008	332603.4146
Vegas-A modified	74011.32993	153830.6905	241777.4826	282336.021	313709.1893

End-to-End Delay:



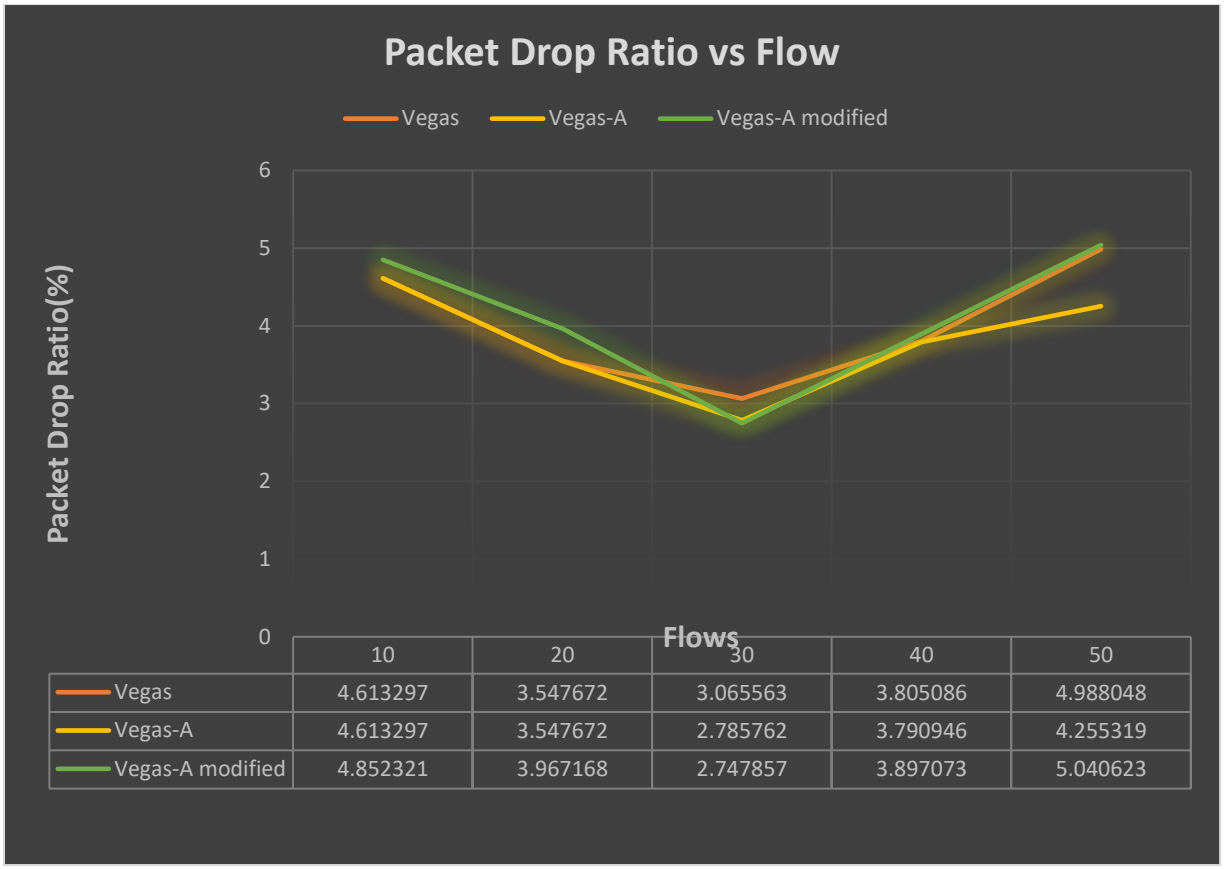
	10	20	30	40	50
Vegas	0.067873	0.051642	0.088354	0.161337	0.257197
Vegas-A	0.067873	0.051642	0.098029	0.19322	0.187196
Vegas-A modified	0.063269	0.08287	0.084505	0.152313	0.199715

Packet Delivery Ratio:



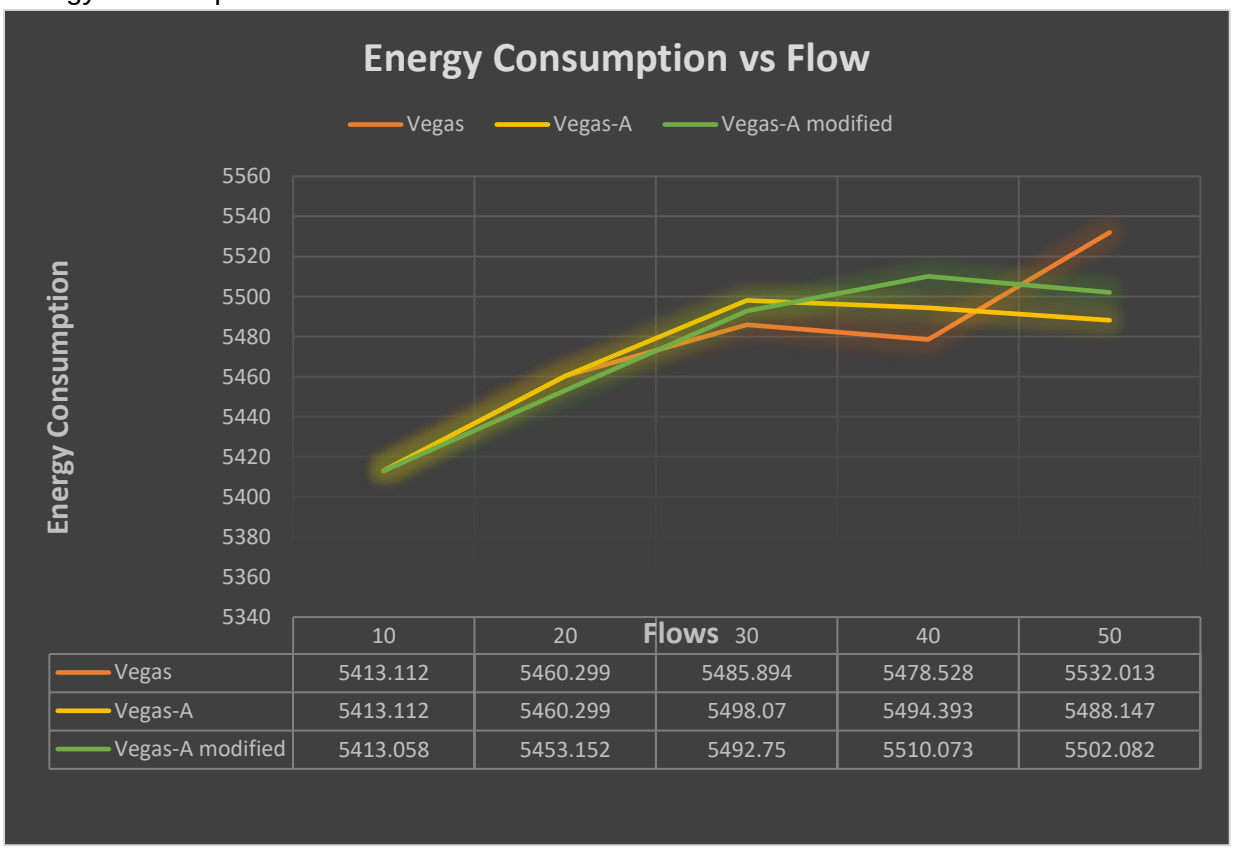
	10	20	30	40	50
Vegas	95.115332	96.452328	96.777803	96.1755	95.011952
Vegas-A	95.115332	96.452328	97.147911	96.04343	95.6974
Vegas-A modified	95.007032	96.032832	97.010332	96.102927	94.959377

Packet Drop Ratio:



	10	20	30	40	50
Vegas	4.613297	3.547672	3.065563	3.805086	4.988048
Vegas-A	4.613297	3.547672	2.785762	3.790946	4.255319
Vegas-A modified	4.852321	3.967168	2.747857	3.897073	5.040623

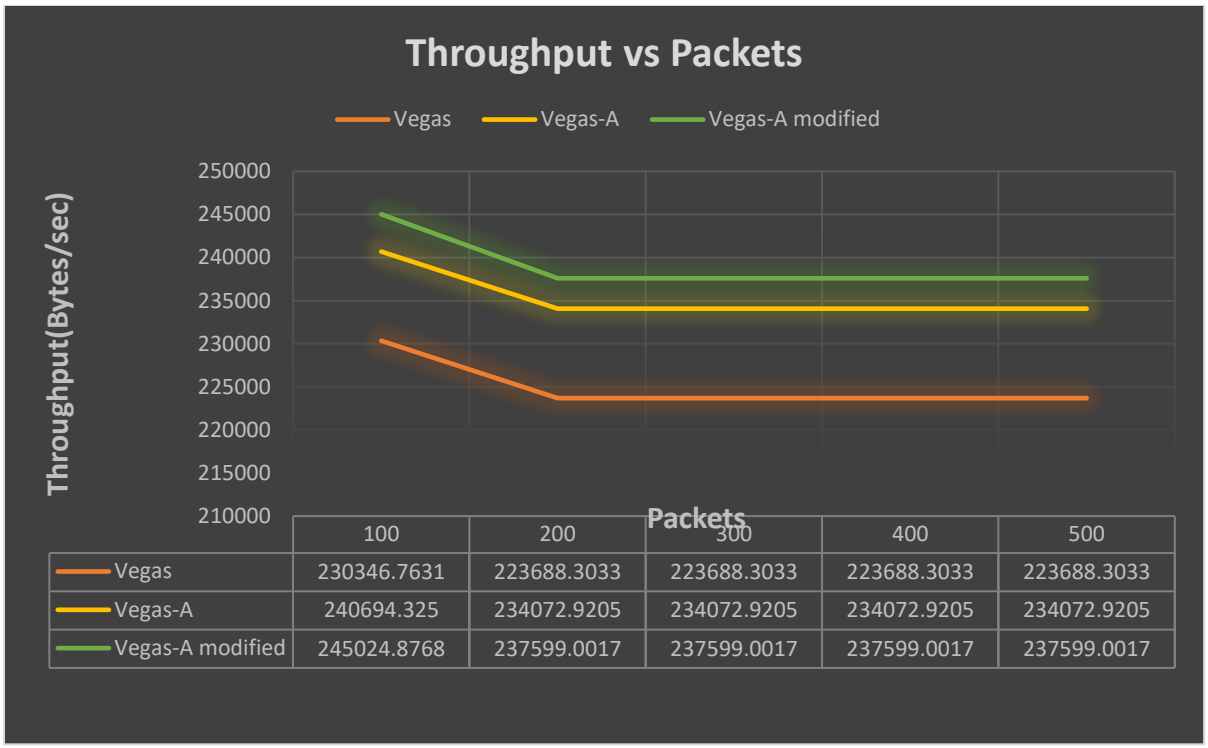
Energy Consumption:



	10	20	30	40	50
Vegas	5413.112	5460.299	5485.894	5478.528	5532.013
Vegas-A	5413.112	5460.299	5498.07	5494.393	5488.147
Vegas-A modified	5413.058	5453.152	5492.75	5510.073	5502.082

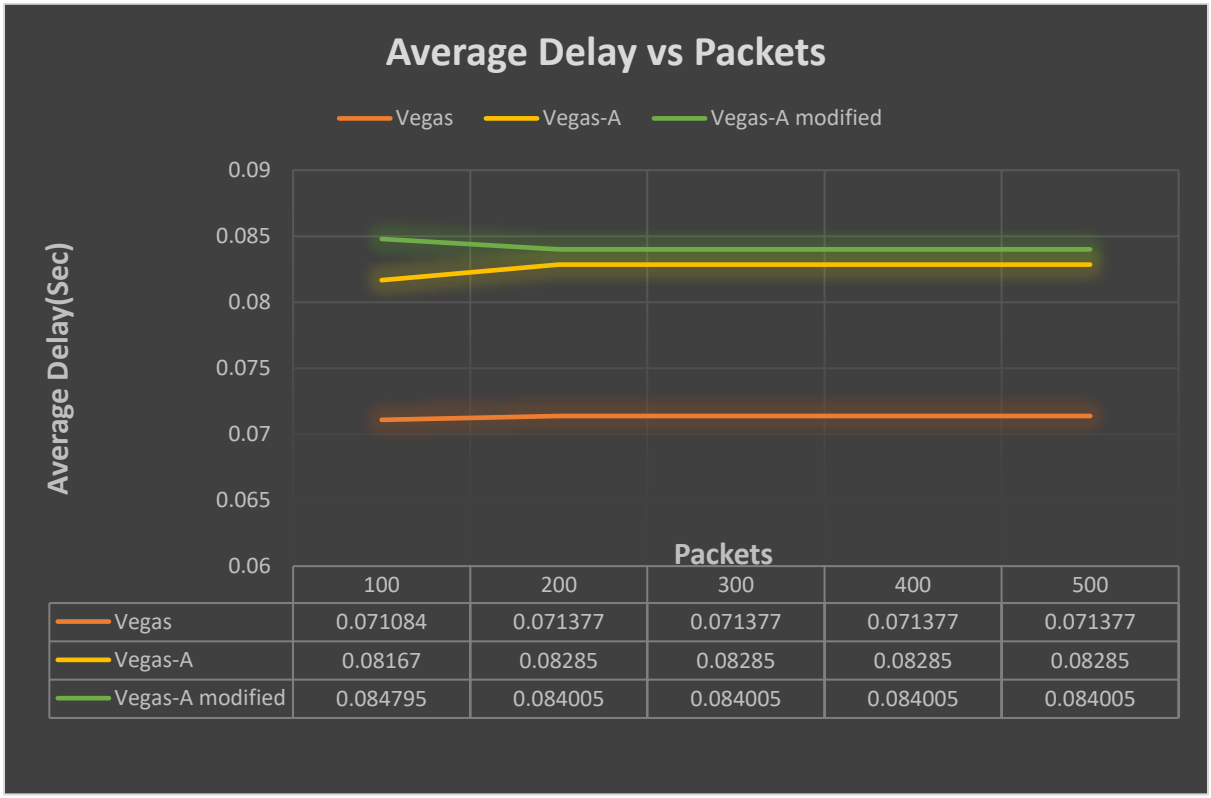
With respect to Number of Packets per second:

Network Throughput:



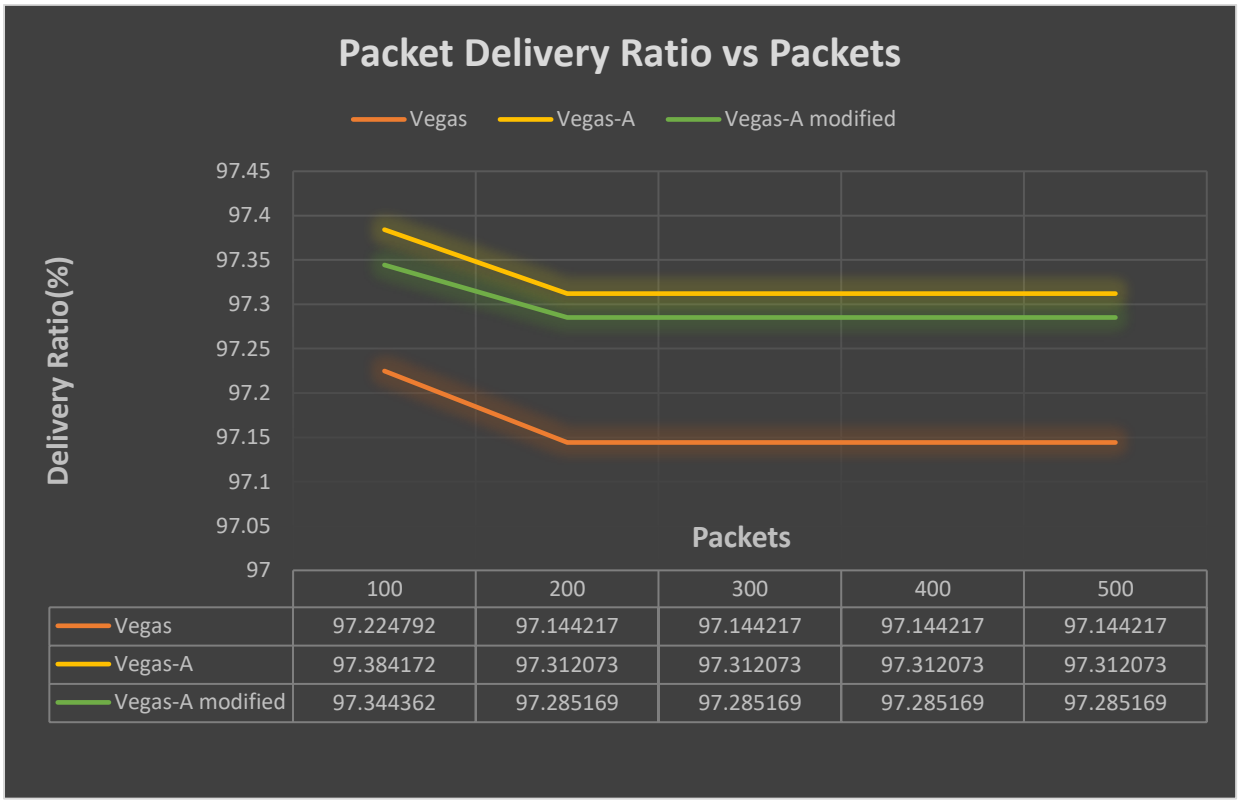
	100	200	300	400	500
Vegas	230346.7631	223688.3033	223688.3033	223688.3033	223688.3033
Vegas-A	240694.325	234072.9205	234072.9205	234072.9205	234072.9205
Vegas-A modified	245024.8768	237599.0017	237599.0017	237599.0017	237599.0017

End-to-End Delay:



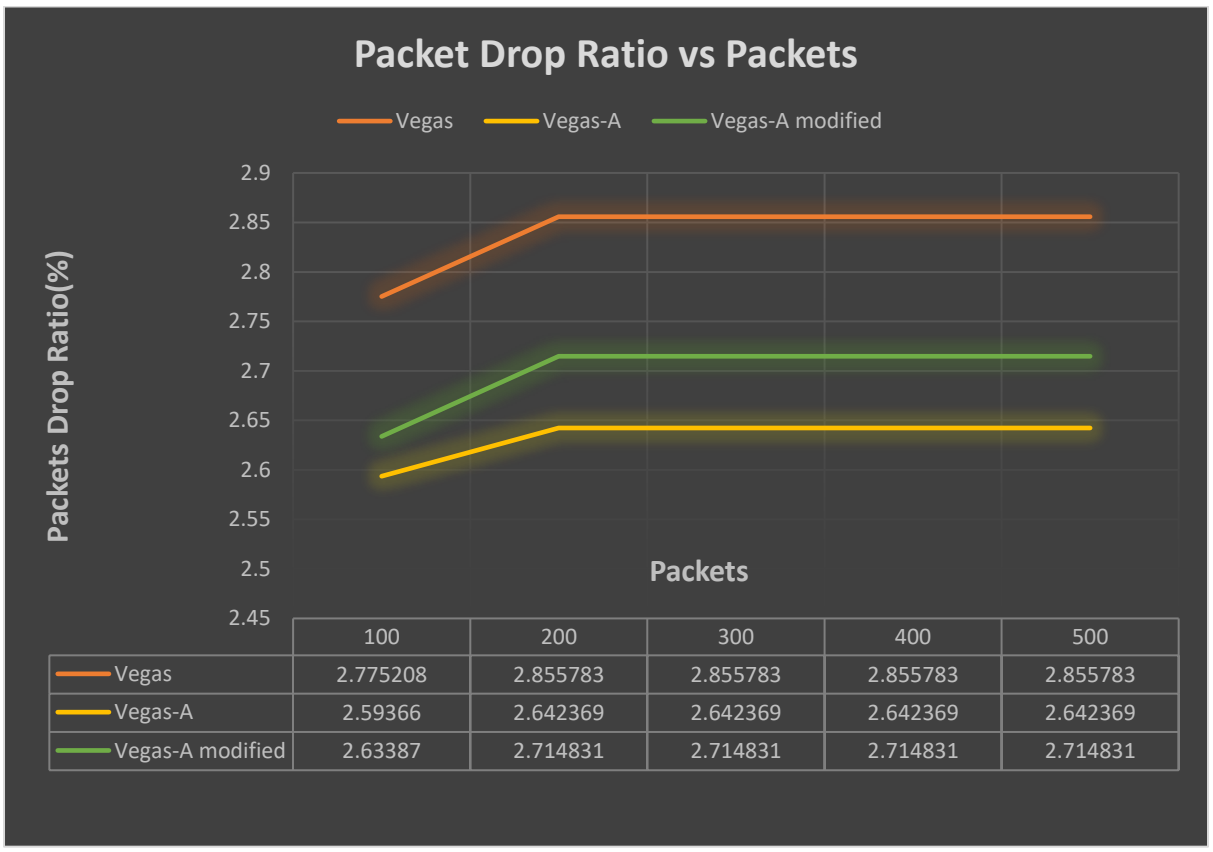
	100	200	300	400	500
Vegas	0.071084	0.071377	0.071377	0.071377	0.071377
Vegas-A	0.08167	0.08285	0.08285	0.08285	0.08285
Vegas-A modified	0.084795	0.084005	0.084005	0.084005	0.084005

Packet Delivery Ratio:



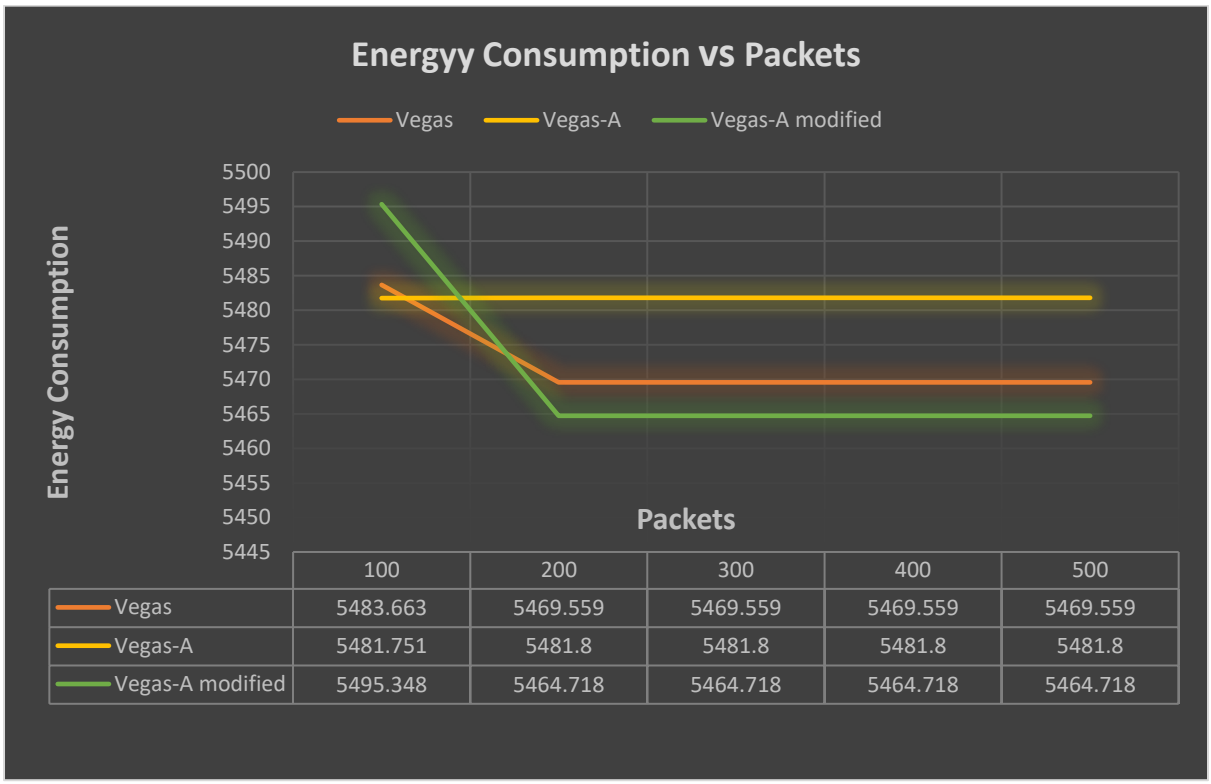
	100	200	300	400	500
Vegas	97.224792	97.144217	97.144217	97.144217	97.144217
Vegas-A	97.384172	97.312073	97.312073	97.312073	97.312073
Vegas-A modified	97.344362	97.285169	97.285169	97.285169	97.285169

Packet Drop Ratio:



	100	200	300	400	500
Vegas	2.775208	2.855783	2.855783	2.855783	2.855783
Vegas-A	2.59366	2.642369	2.642369	2.642369	2.642369
Vegas-A modified	2.63387	2.714831	2.714831	2.714831	2.714831

Energy Consumption:

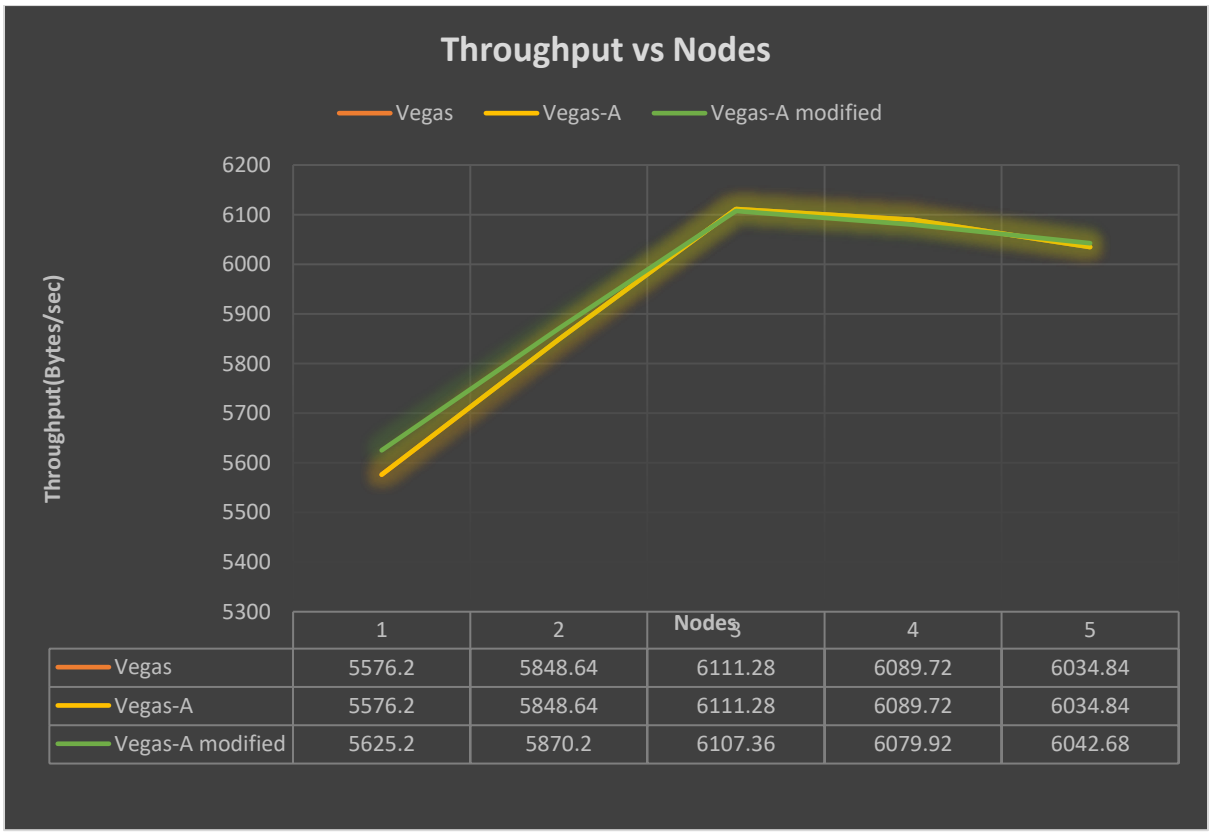


	100	200	300	400	500
Vegas	5483.663	5469.559	5469.559	5469.559	5469.559
Vegas-A	5481.751	5481.8	5481.8	5481.8	5481.8
Vegas-A modified	5495.348	5464.718	5464.718	5464.718	5464.718

2. Wired nodes:

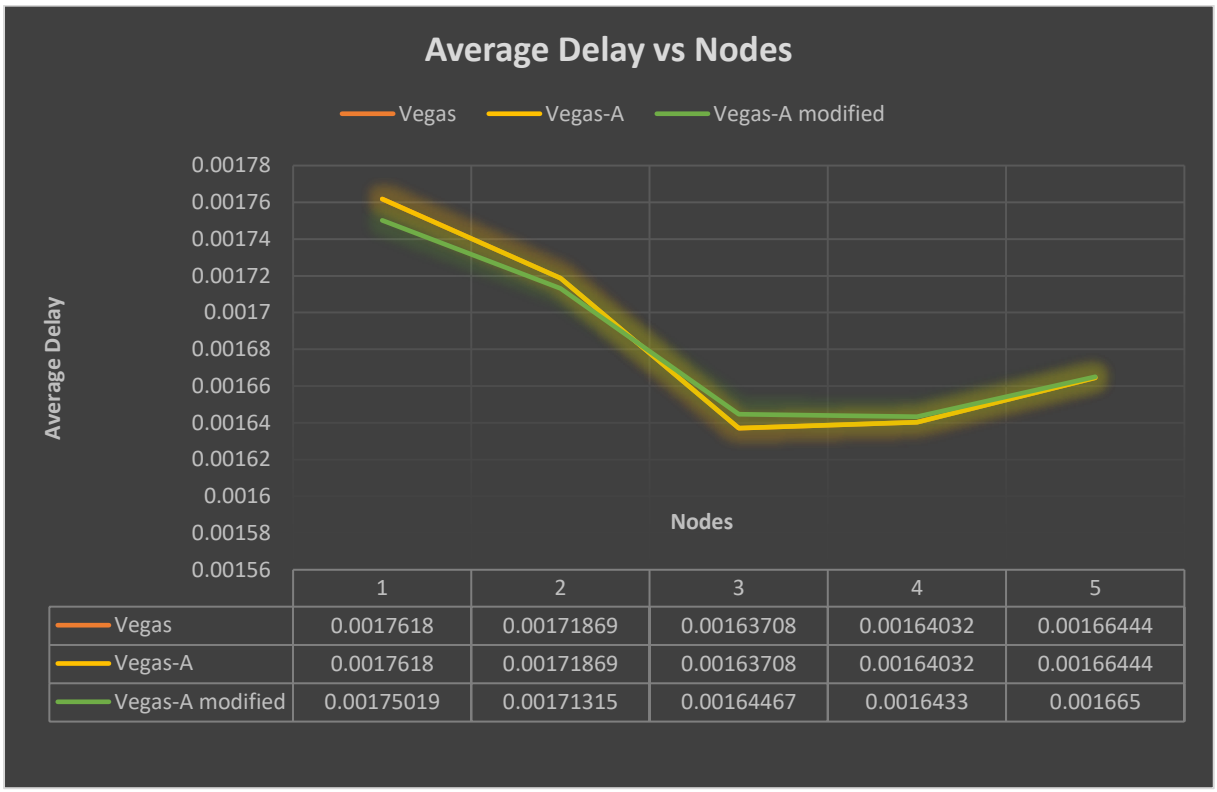
With respect to Number of Nodes:

Network Throughput:



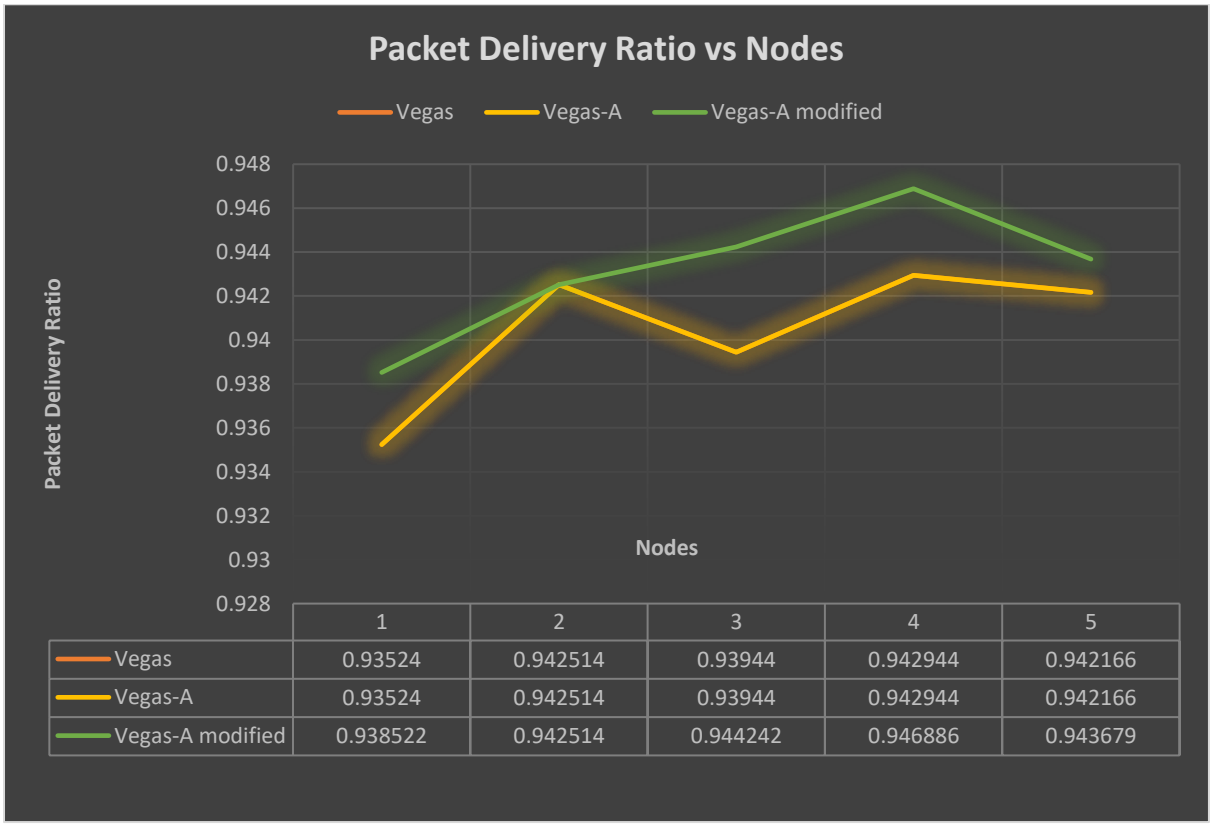
	1	2	3	4	5
Vegas	5576.2	5848.64	6111.28	6089.72	6034.84
Vegas-A	5576.2	5848.64	6111.28	6089.72	6034.84
Vegas-A modified	5625.2	5870.2	6107.36	6079.92	6042.68

End-to-End Delay:



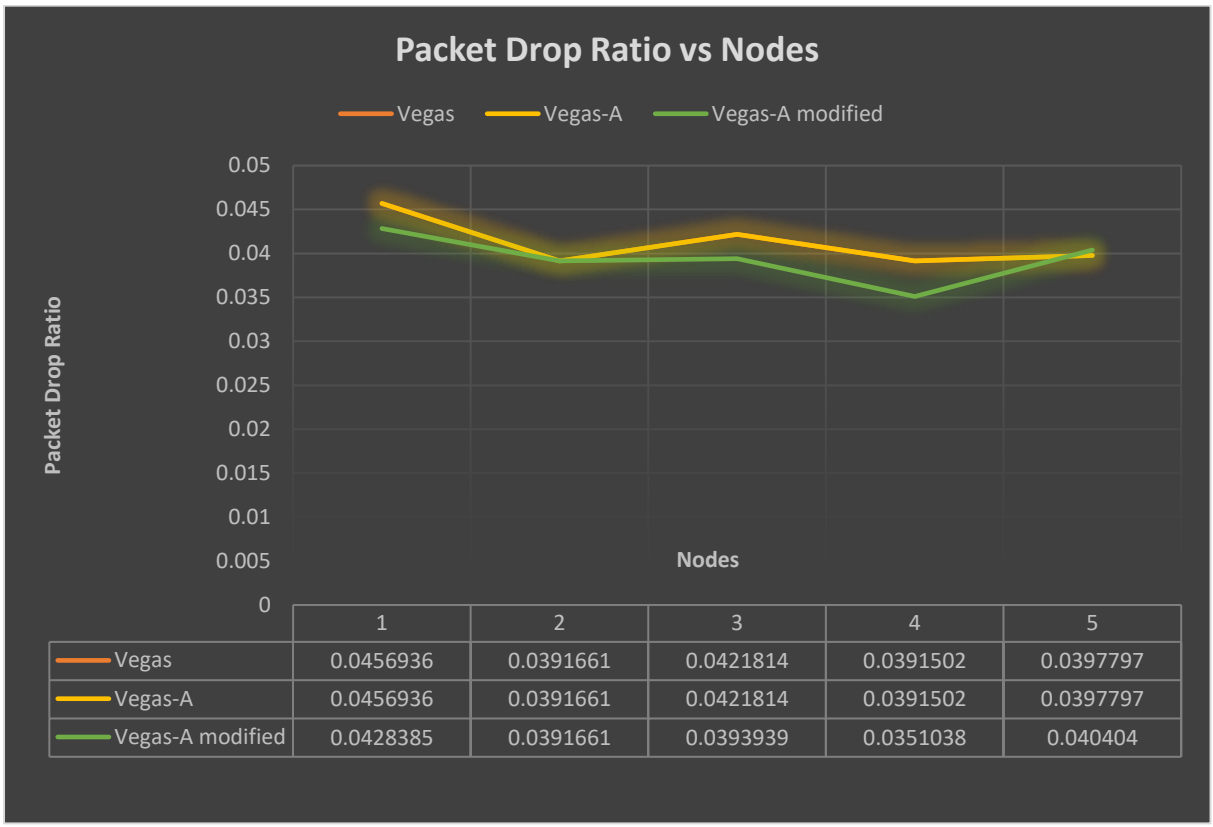
	1	2	3	4	5
Vegas	0.0017618	0.00171869	0.00163708	0.00164032	0.00166444
Vegas-A	0.0017618	0.00171869	0.00163708	0.00164032	0.00166444
Vegas-A modified	0.00175019	0.00171315	0.00164467	0.0016433	0.001665

Packet Delivery Ratio:



	1	2	3	4	5
Vegas	0.93524	0.942514	0.93944	0.942944	0.942166
Vegas-A	0.93524	0.942514	0.93944	0.942944	0.942166
Vegas-A modified	0.938522	0.942514	0.944242	0.946886	0.943679

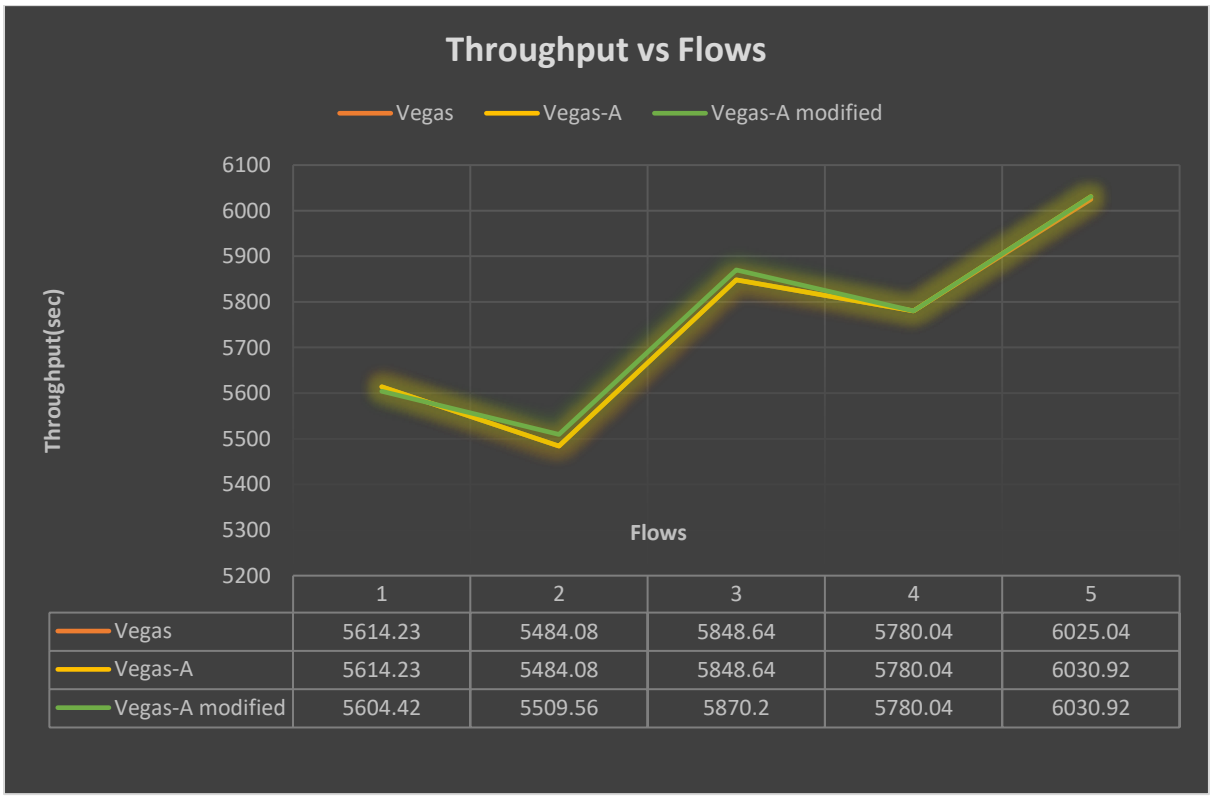
Packet Drop Ratio:



	1	2	3	4	5
Vegas	0.0456936	0.0391661	0.0421814	0.0391502	0.0397797
Vegas-A	0.0456936	0.0391661	0.0421814	0.0391502	0.0397797
Vegas-A modified	0.0428385	0.0391661	0.0393939	0.0351038	0.040404

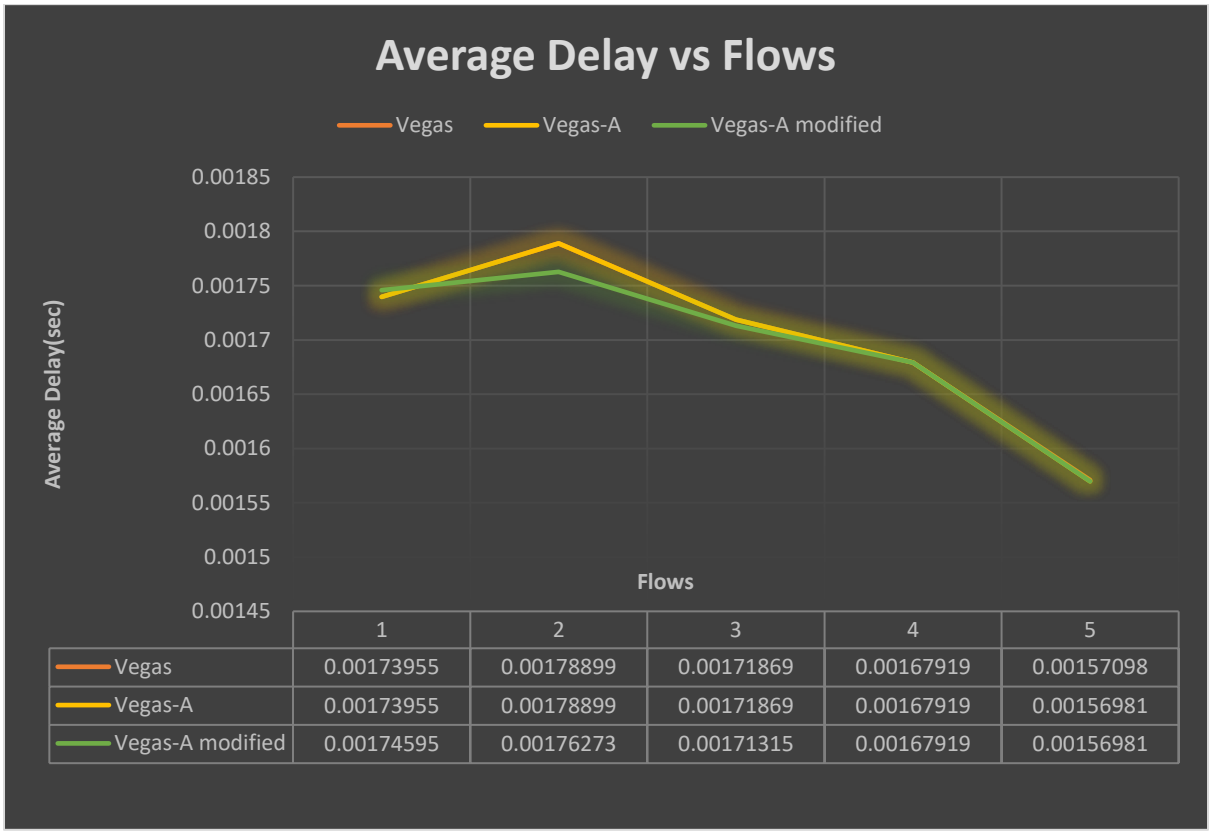
With respect to Number of Flows:

Network Throughput:



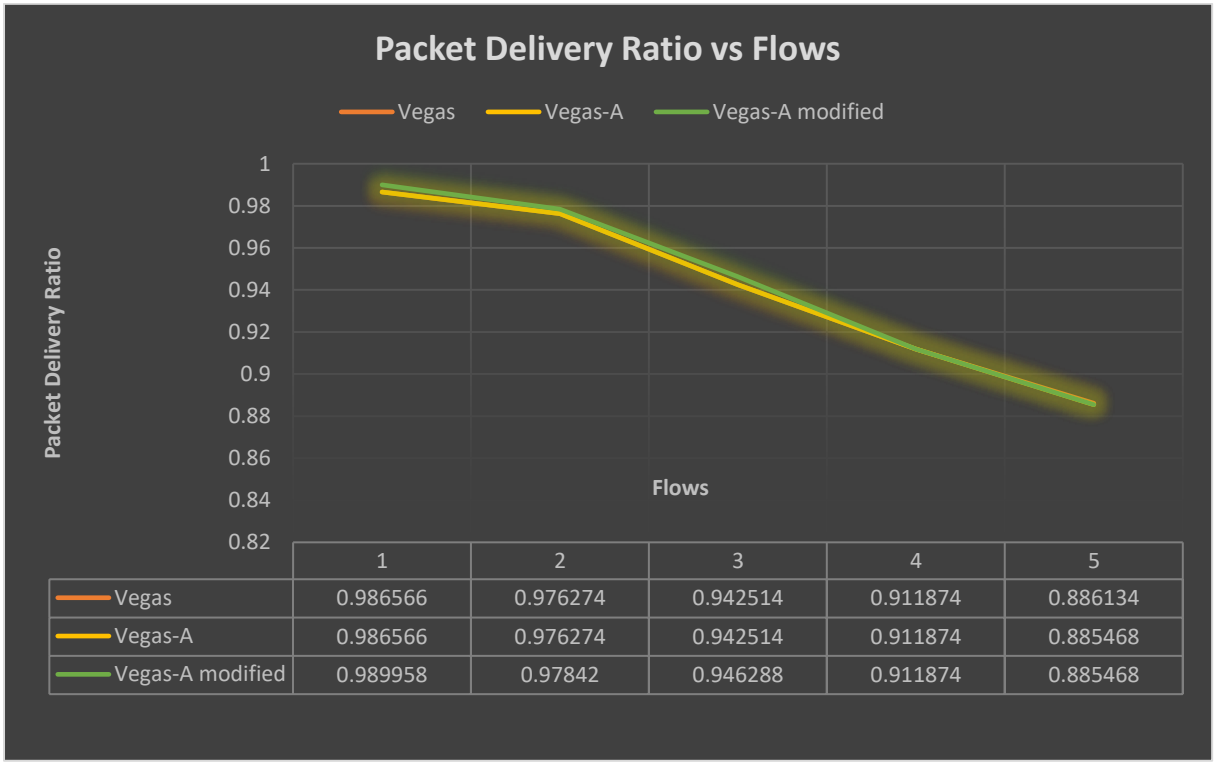
	1	2	3	4	5
Vegas	5614.23	5484.08	5848.64	5780.04	6025.04
Vegas-A	5614.23	5484.08	5848.64	5780.04	6030.92
Vegas-A modified	5604.42	5509.56	5870.2	5780.04	6030.92

End-to-End Delay:



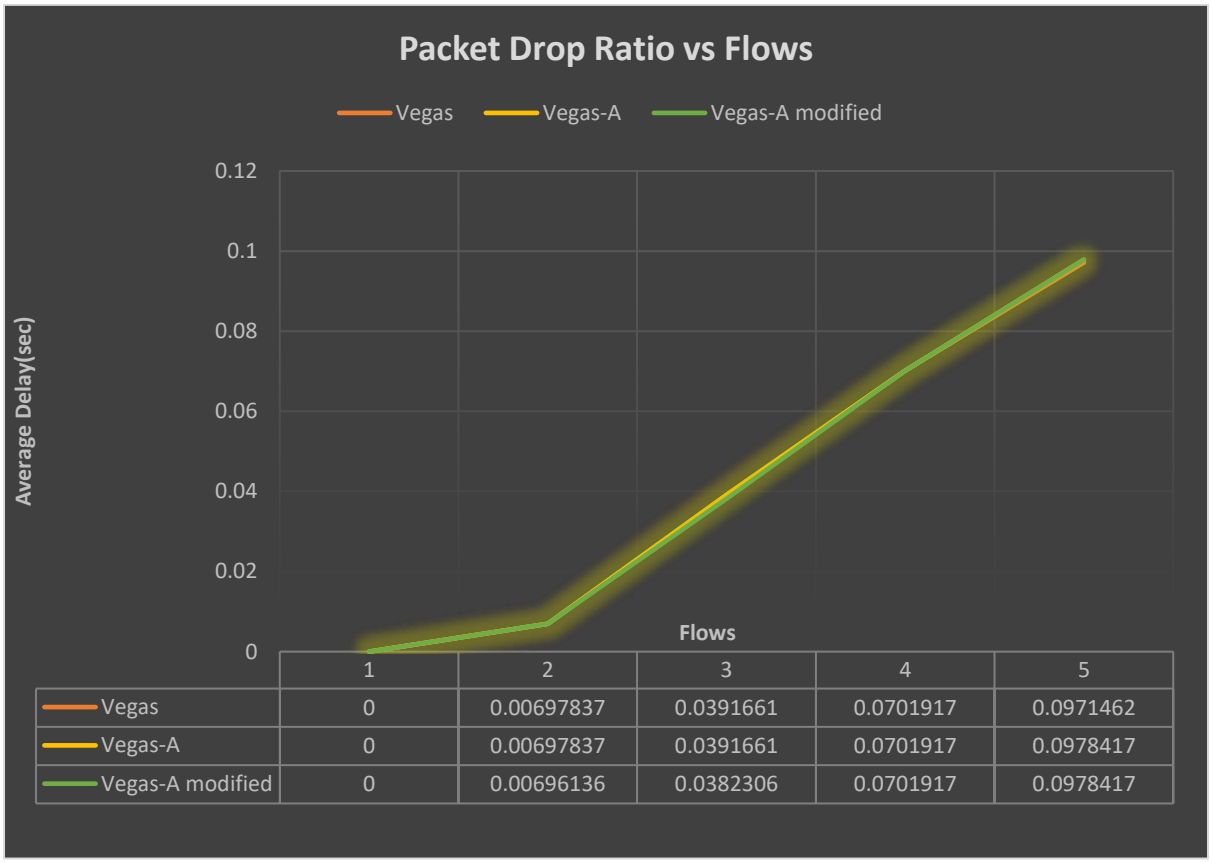
	1	2	3	4	5
Vegas	0.00173955	0.00178899	0.00171869	0.00167919	0.00157098
Vegas-A	0.00173955	0.00178899	0.00171869	0.00167919	0.00156981
Vegas-A modified	0.00174595	0.00176273	0.00171315	0.00167919	0.00156981

Packet Delivery Ratio:



	1	2	3	4	5
Vegas	0.986566	0.976274	0.942514	0.911874	0.886134
Vegas-A	0.986566	0.976274	0.942514	0.911874	0.885468
Vegas-A modified	0.989958	0.97842	0.946288	0.911874	0.885468

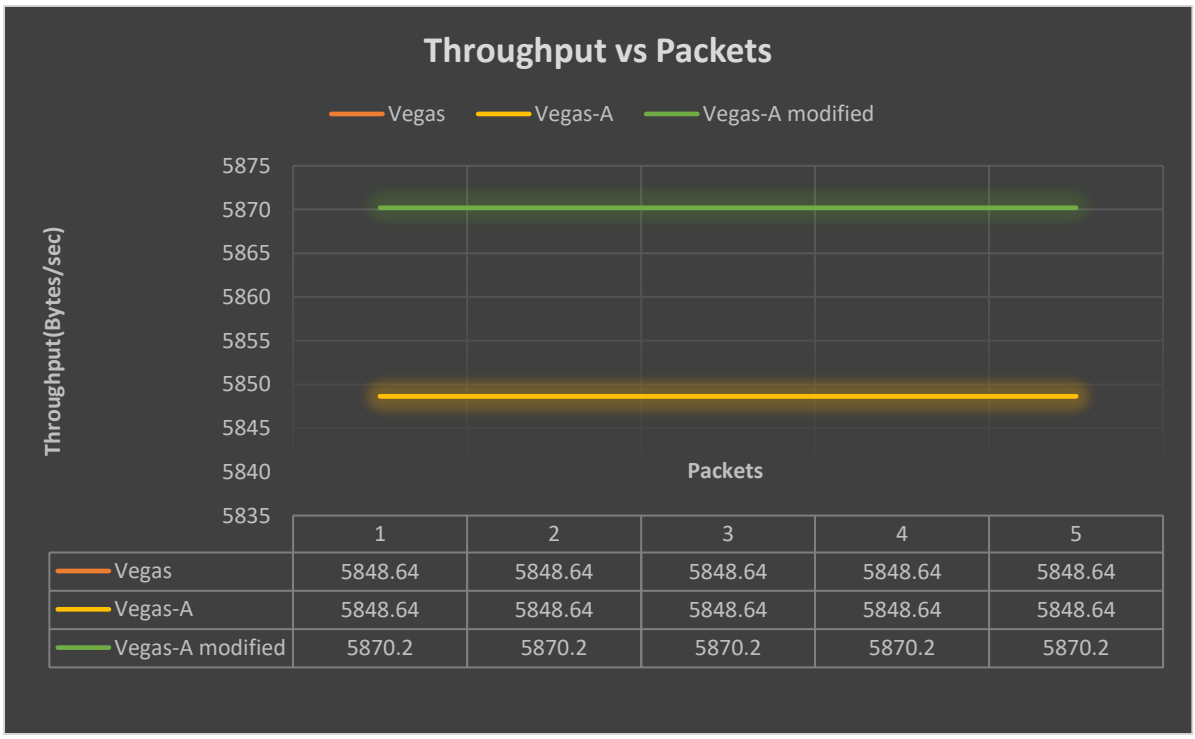
Packet Drop Ratio:



	1	2	3	4	5
Vegas	0	0.00697837	0.0391661	0.0701917	0.0971462
Vegas-A	0	0.00697837	0.0391661	0.0701917	0.0978417
Vegas-A modified	0	0.00696136	0.0382306	0.0701917	0.0978417

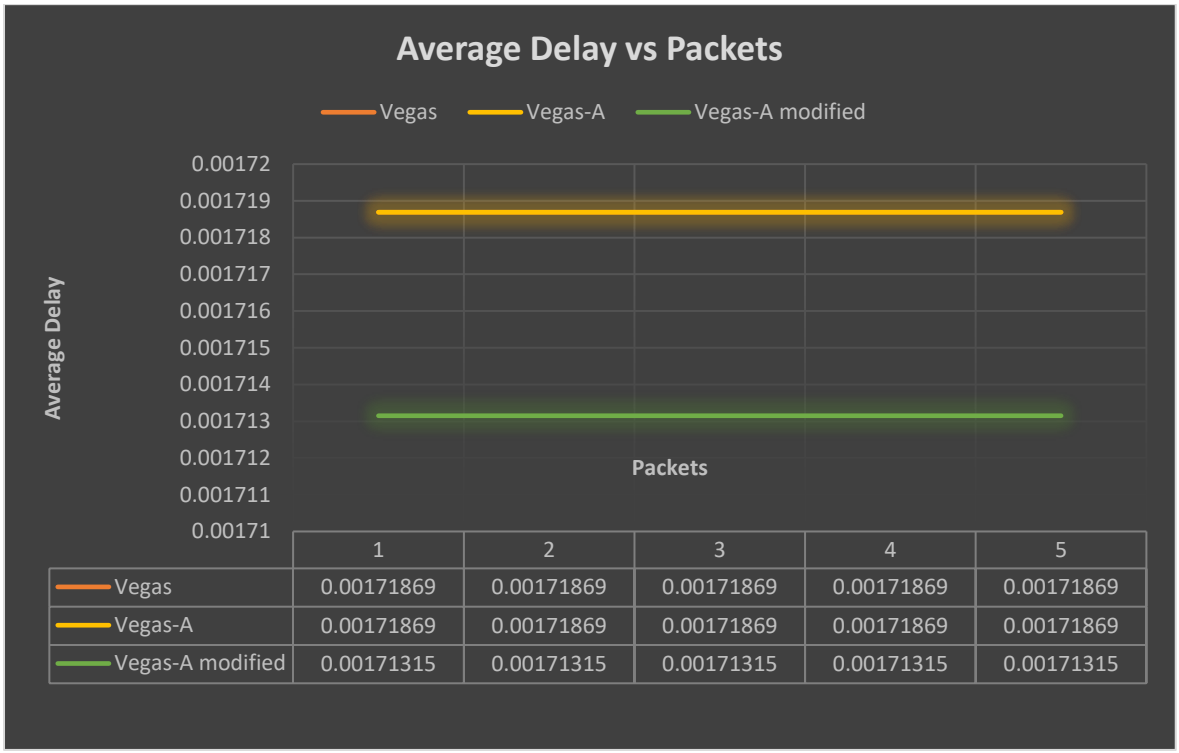
With respect to Number of Packets per second:

Network Throughput:



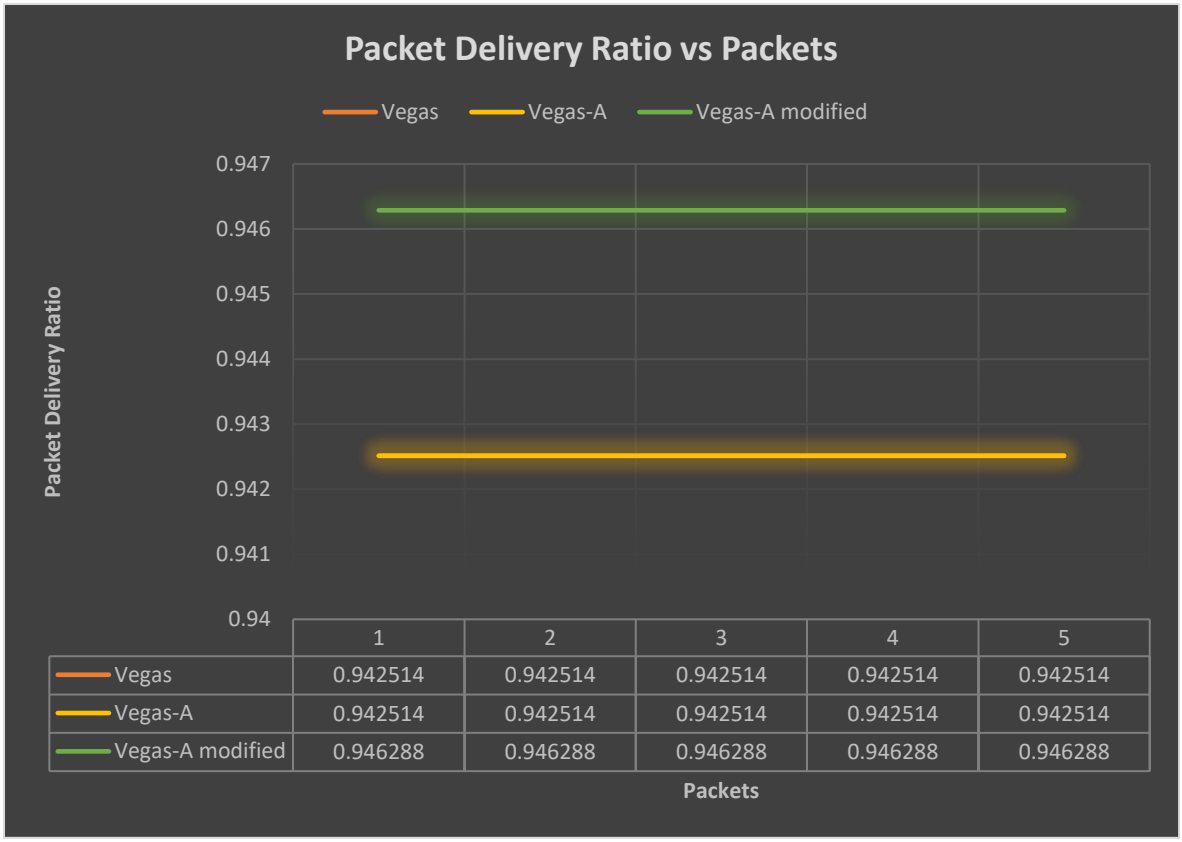
	1	2	3	4	5
Vegas	5848.64	5848.64	5848.64	5848.64	5848.64
Vegas-A	5848.64	5848.64	5848.64	5848.64	5848.64
Vegas-A modified	5870.2	5870.2	5870.2	5870.2	5870.2

End-to-End Delay:



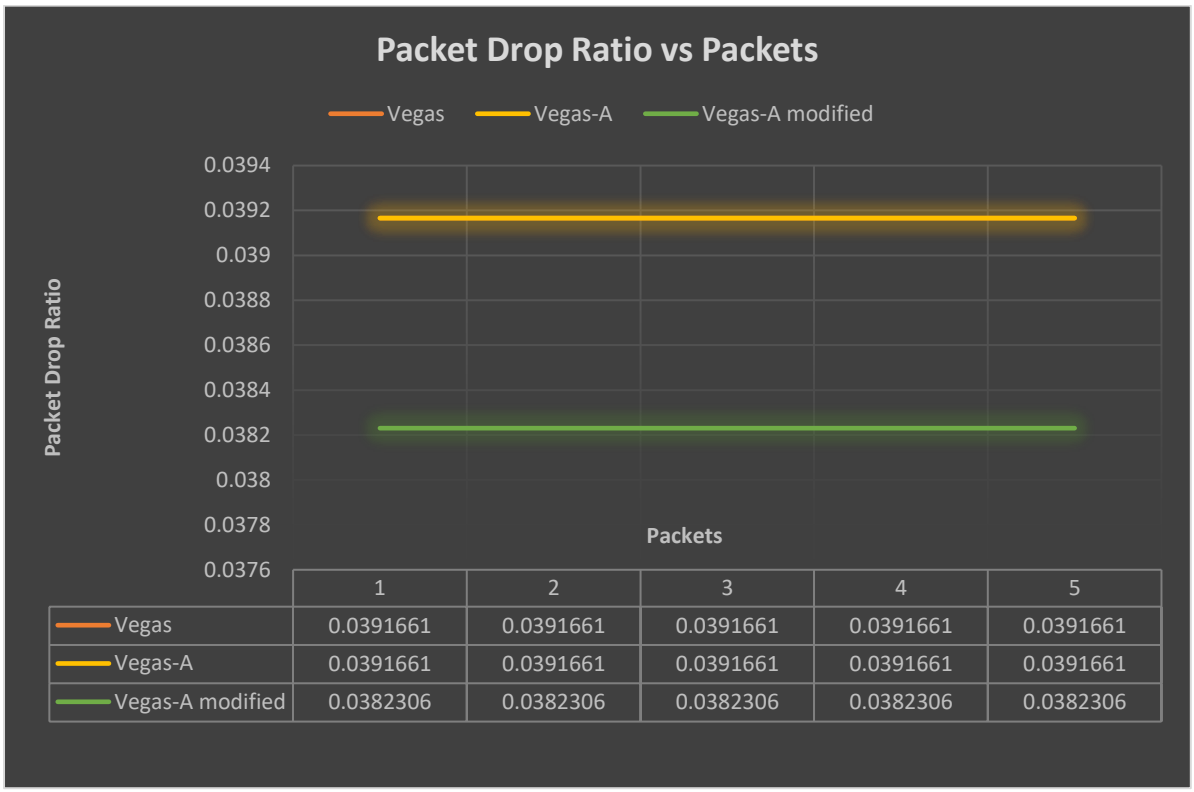
	1	2	3	4	5
Vegas	0.00171869	0.00171869	0.00171869	0.00171869	0.00171869
Vegas-A	0.00171869	0.00171869	0.00171869	0.00171869	0.00171869
Vegas-A modified	0.00171315	0.00171315	0.00171315	0.00171315	0.00171315

Packet Delivery Ratio:



	1	2	3	4	5
Vegas	0.942514	0.942514	0.942514	0.942514	0.942514
Vegas-A	0.942514	0.942514	0.942514	0.942514	0.942514
Vegas-A modified	0.946288	0.946288	0.946288	0.946288	0.946288

Packet Drop Ratio:



	1	2	3	4	5
Vegas	0.0391661	0.0391661	0.0391661	0.0391661	0.0391661
Vegas-A	0.0391661	0.0391661	0.0391661	0.0391661	0.0391661
Vegas-A modified	0.0382306	0.0382306	0.0382306	0.0382306	0.0382306

<u>Summary of Vegas-A vs Vegas:</u>

TCP Vegas-A shows improvement over the tradition TCP Vegas in cases of:

- Throughput, packet delivery ratio and packet drop ratio for wireless
- Throughput (in some cases) in wired

TCP Vegas-A lacks behind in case of:

- Average Delay and Energy Consumption for wireless

<u>Summary of Modified Vegas-A vs Vegas:</u>

TCP Vegas-A(Modified) shows improvement over the tradition TCP Vegas in cases of:

- Throughput, Packet Delivery Ratio and Packet Drop Ratio, Energy Consumption while varying packets sent per second for wireless
- All four parameters almost (Throughput, Packet Delivery Ratio and Packet Drop Ratio, Average Delay) for wired