Computer Networks Sessional

CSE - 322

NS2 Project Report

DiffQ Congestion Control

Abdus Samee
ID: 1805021
Section: A
Dept: CSE

1 Personalized Parameters

The following parameters were assigned to me:

Selected Paper This paper has been selected for simulation in NS2.

Wireless Static Network A static network topology has been designed and the modified source code is compared with the existing one with varying parameters like node count, flow count, packet per second, and coverage area. I was assigned to display this network using MAC 802.11. The nodes were arranged in a grid in the network topology.

Wireless Mobile Network A wireless network topology has been designed with mobile nodes and modified source code is compared with the existing one by varying node count, flow count, packet per second, and speed of nodes. In this network, I was assigned MAC 802.15.4. But using this one made by throughput calculation negative. So, it was switched to 802.11. The nodes were arranged in a grid in the network topology.

Default parameters The values of the parameters by default: node count: 40, flow count: 20, packet rate: 200, Tx range: $1x$, and speed of a node: $10ms^{-1}$

Network Topology As no particular topology was assigned, a grid topology has been considered for demonstration purposes in NS2.

2 Modifications in NS2

The following modifications were made in the source code of NS2:

- A new queue file named **diff-queue.c** and its header file **diff-queue.h** have been added. This queue helps in backlog congestion by creating multiple queues in each agent for multiple destinations. The packets are dequeued based on the *DiffQ Priority*.
- Three new constants **sent-packets**, **ALPHA** = 0.125 and **BETA** = 0.25 have been included in the **tcp.h** file for manipulating the congestion window inside the **tcp.cc** file. The **sent-packets** variable is increased/decreased whenever a packet is sent and its **ACK** is received. Besides, some of the calculations and constants were manipulated using the constants mentioned above.

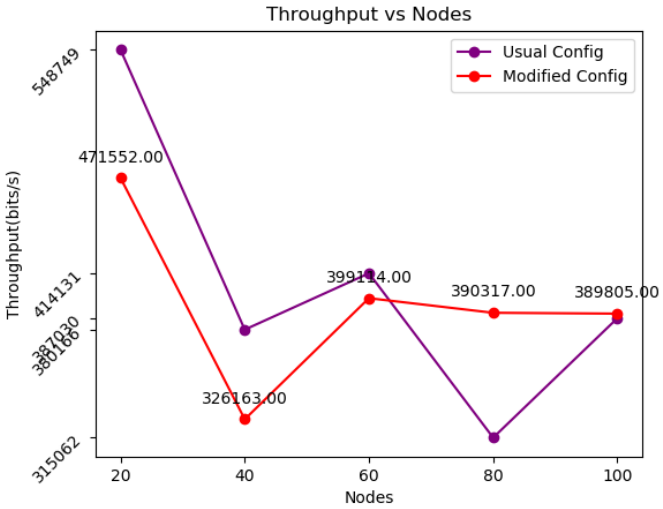
3 Wireless Network - Static

This network is simulated with MAC 802.11 by varying necessary parameters and generating graphs showcasing a comparison with existing implementation

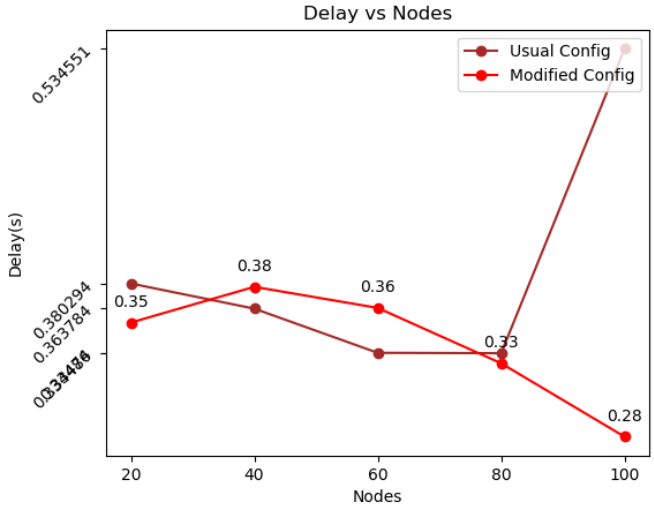
in NS2 with the modified source code. The modification has been made in the original `tcp.cc` file. So the difference lies in the queue used since a new queue has been implemented for the DiffQ congestion control and the original DropTail Priority Queue.

3.1 Graphs - Changing Node Count

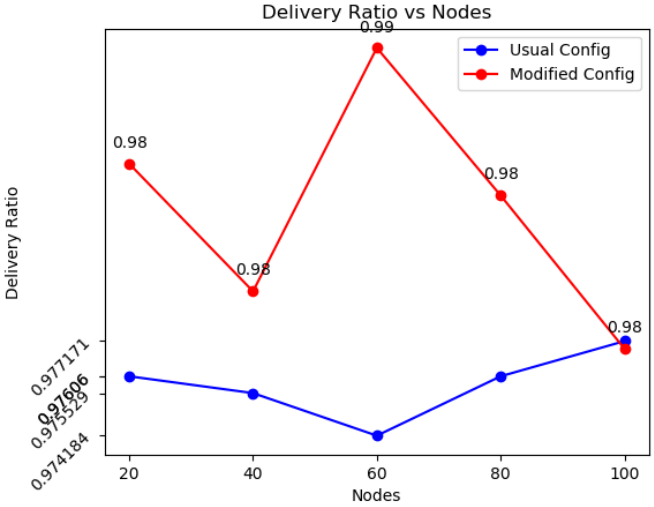
The **5** graphs are shown below:



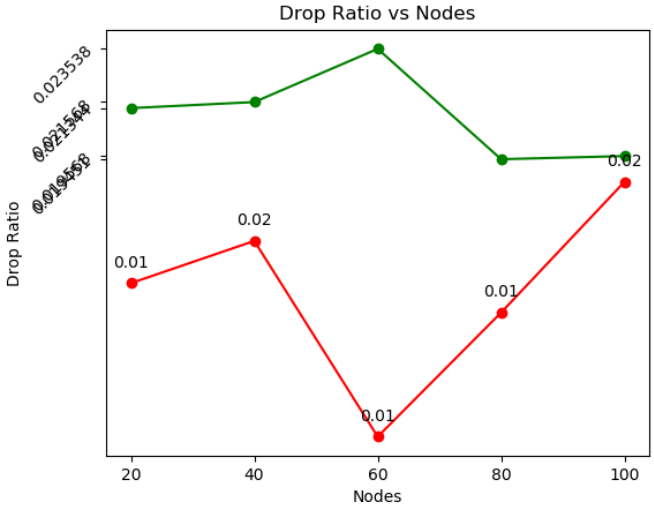
Generally, the throughput decreases with the increase in node count in the original and modified source code. If closely looked at, the throughput decreases more in the modified source code(red coloured).



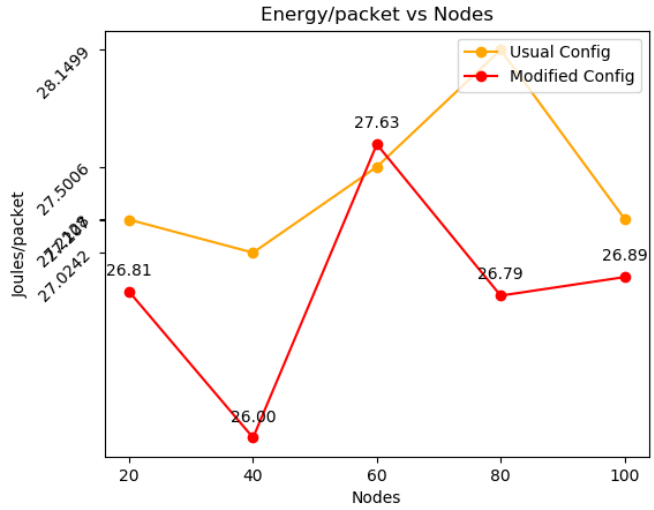
The delay in transmitting packets is less in the modified source code than in the original source code. It shows a gradual decrease in delay with increasing node count.



The delivery ratio is higher since the delay in packet transmission is lower in the modified source code. In fact, both the modified and original source code has a good delivery ratio since their basis is the original TCP.



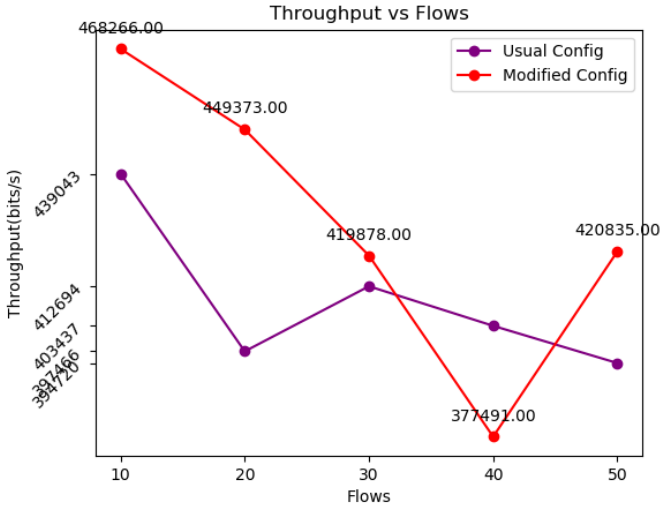
The drop ratio shows the mirror image of the delivery ratio in both versions of the code. Although modified source code is seen to have a lower drop ratio than original source code, their ranges are insignificant.



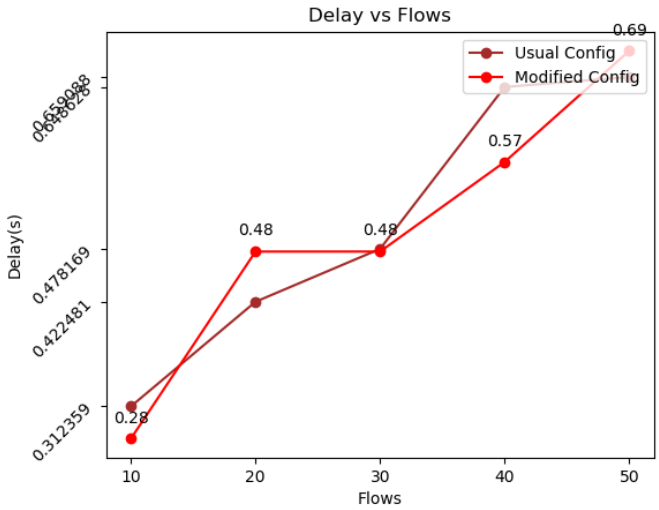
The rate of energy per packet in both versions of the code lies within the range of $(0.027, 0.028)$ Joules.

3.2 Graphs - Changing Flow Count

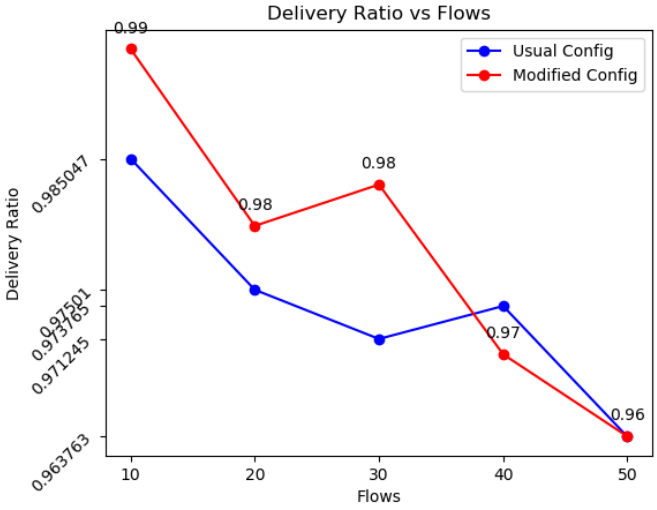
The **5** graphs are shown below:



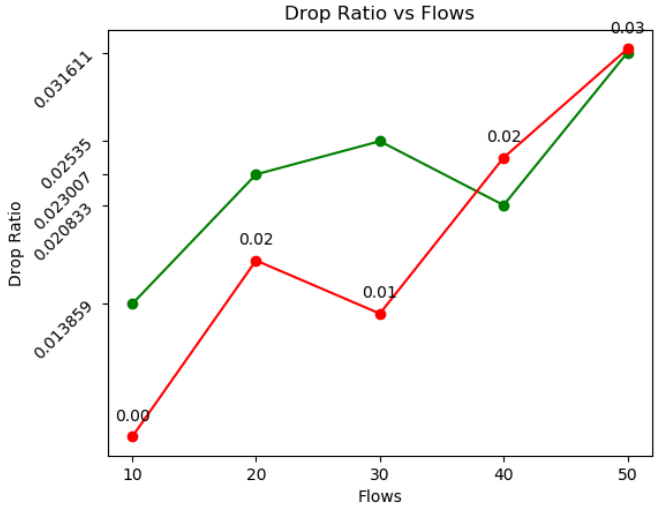
Generally, the throughput decreases with the increase in flow count in the original and modified source code. If closely looked at, the throughput decreases more in the original source code(purple coloured), except when the flow count is 40.



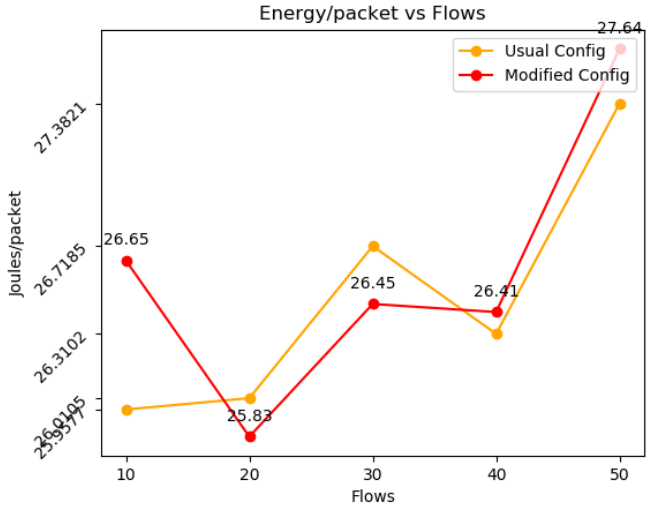
The delays in packet transmission in both the original and modified configurations go hand-in-hand, increasing with increasing flow count.



The delivery ratio decreases in both versions of the code with increasing flow count because of the congestion in the network due to excessive flows. Still, the ratio remains above 95%.



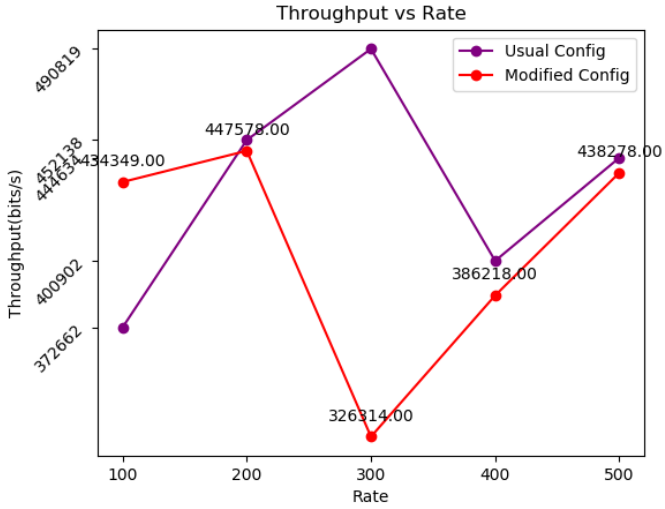
The drop ratio is the exact mirror image of the delivery ratio in the respective codes.



The energy consumption per packet increases with the increasing number of flows.

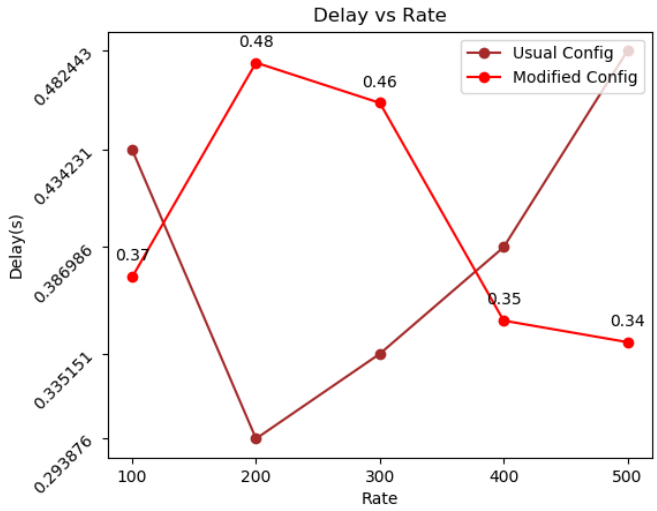
3.3 Graphs - Changing Packet Rate

The **5** graphs are shown below:

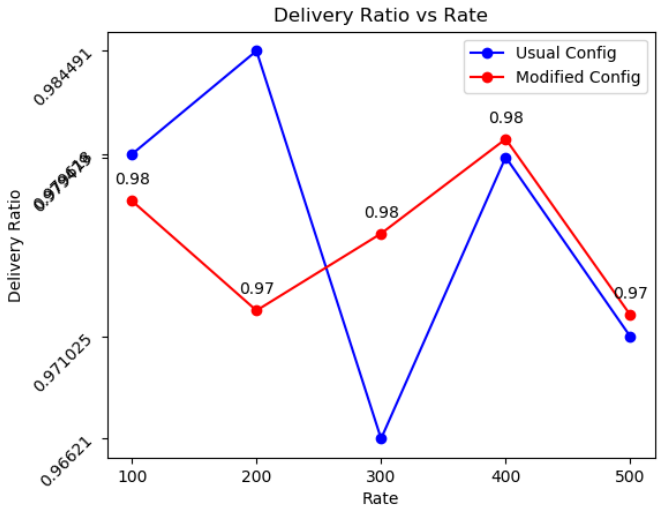


The throughput is on the rise for original source code but declines for the modified source code when the packet rate is increased. They have the highest

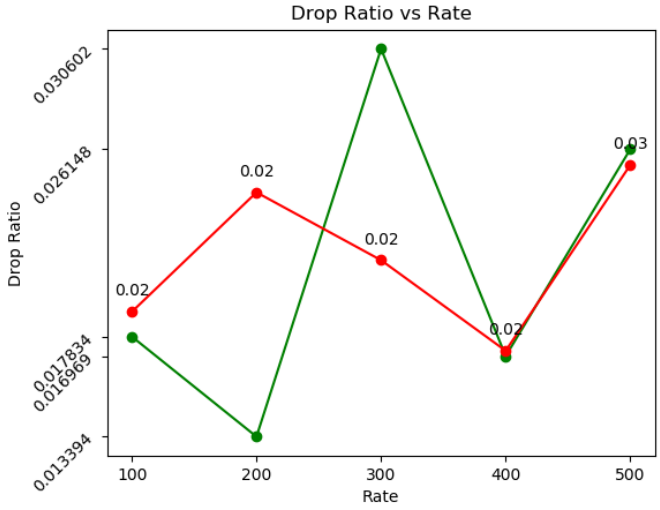
contrast at a packet rate of 300 but after that, they increase, even though original source code performs better.



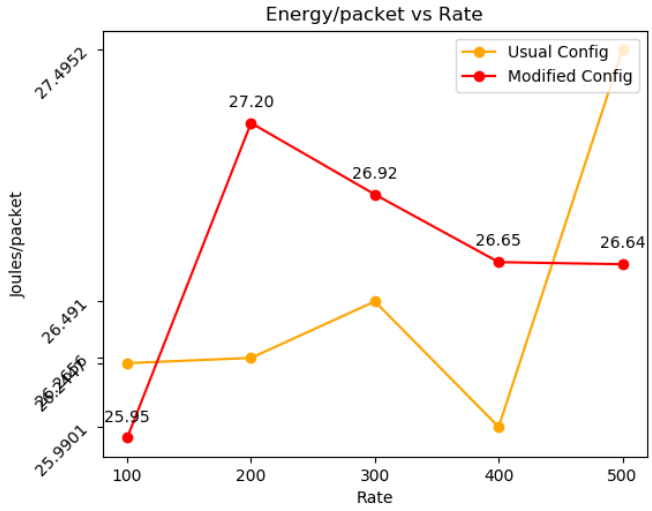
With the increase in packet rate, the delay in packet transmission usually decreases in modified source code but increases in original source code, with some exceptions at some points.



The delivery ratio shows unusual behaviour with an increasing rate with no standard pattern in its change. But one thing is constant, that is, both versions of code have a higher delivery ratio.



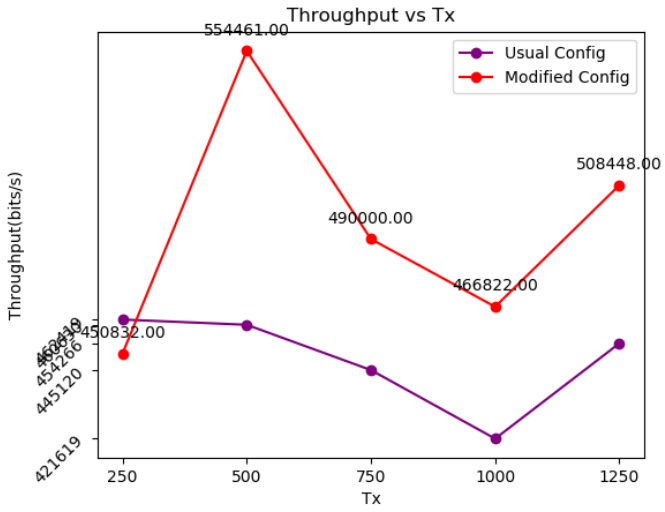
Since the delivery ratio is higher, the drop ratio is significantly lower in both versions of the code.



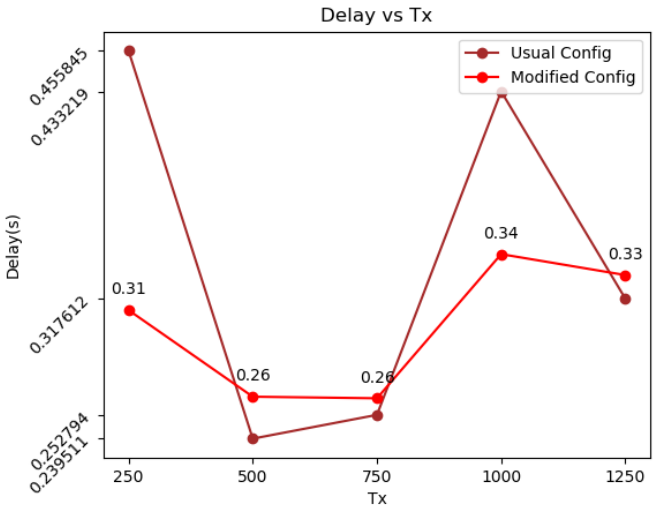
The rate of energy consumption per packet is within the range of $(0.025, 0.028)$ for both versions of the code. Also, it has unprecedented ups and downs.

3.4 Graphs - Changing Tx Range

The **5** graphs are shown below:

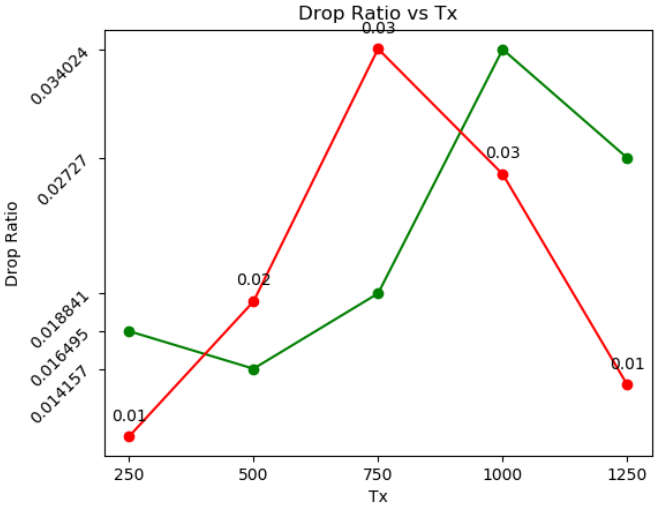


The modified source code gives high throughput when the Tx range is on the rise. The cause is the updated DiffQueue which is used to control the congestion and reduce packet drop and delay.

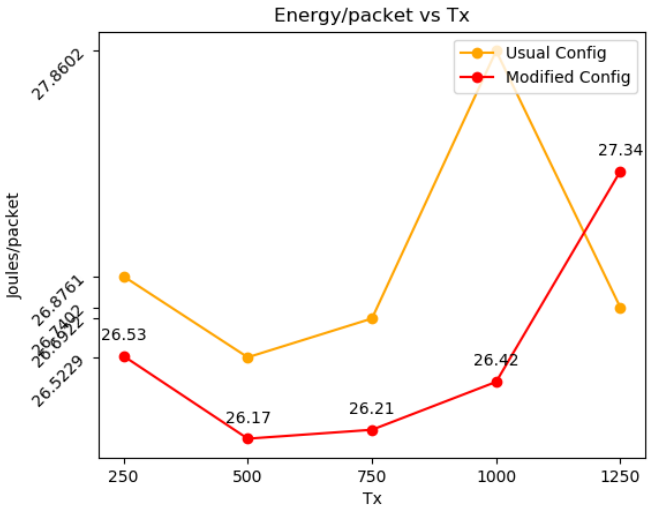


The delay graph has ups and downs with increasing Tx range. Generally, we can see a descent because as the transmission range increases, there is a delay in transmitting packets to a longer distance.

The delivery ratio in both codes has a similar pattern due to the usage of the basic TCP congestion control. But the modified code shows a little lower ratio than the original counterpart.



The drop ratio is a bit higher in the modified NS2 code than the original one owing to the lower delivery ratio with increasing Tx range.



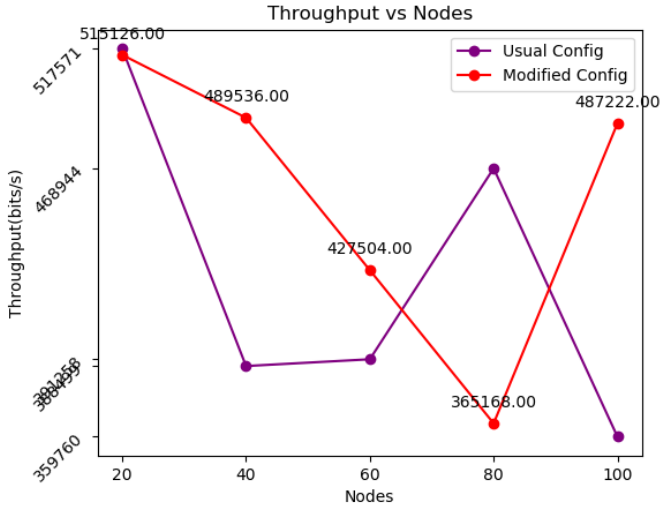
With increasing Tx range, the energy consumed per packet increases. It is due to the fact that more energy is required to transmit packets to farther distances.

4 Wireless Network - Mobile

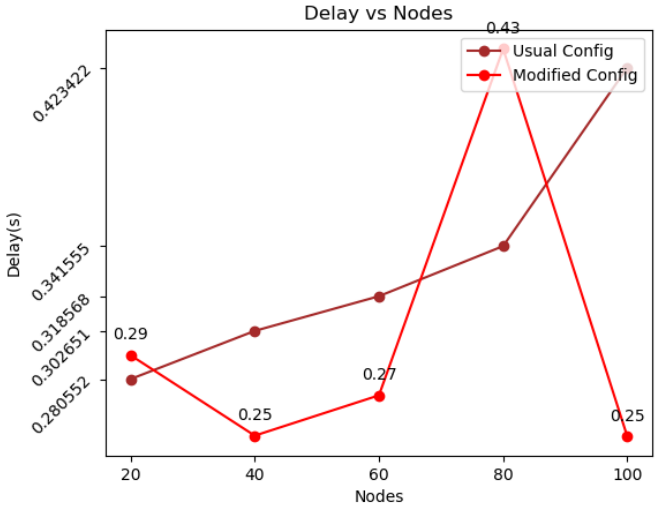
This network is simulated with MAC 802.11 by varying necessary parameters and generating graphs showcasing a comparison with existing implementation in NS2 with the modified source code. The original assignment was MAC 802.15.4, but that gave the throughput negative. So it was changed. The modification has been made in the original `tcp.cc` file. So the difference lies in the queue used since a new queue has been implemented for the DiffQ congestion control.

4.1 Graphs - Changing Node Count

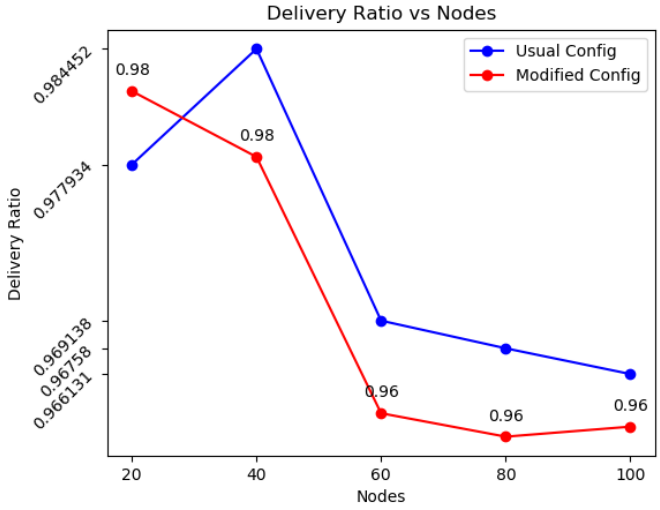
The **5** graphs are shown below:



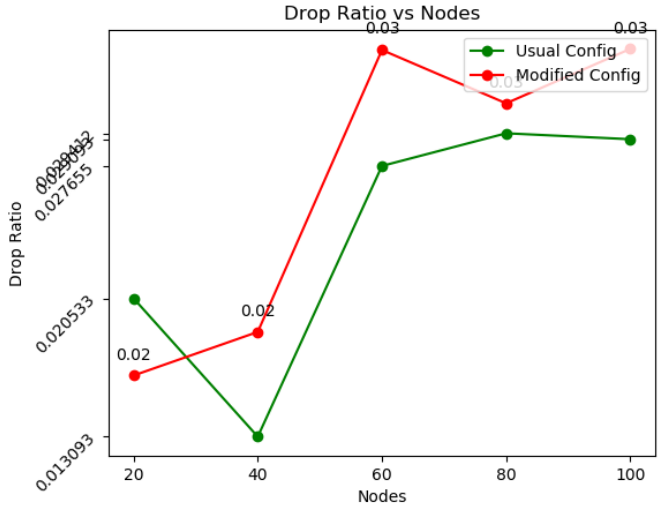
Generally, the throughput decreases with the increase in node count in both the original and modified source code. If closely looked at, the throughput decreases more in the modified source code(red coloured). Moreover, the throughput also shows a rise with increasing node count for original source code(purple coloured).



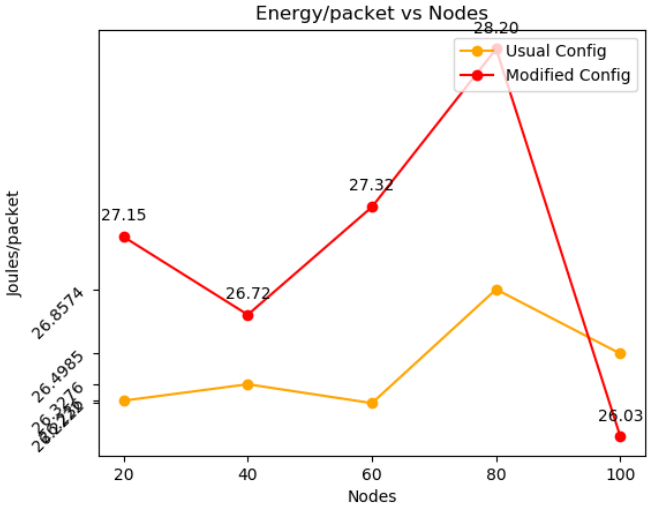
The delay is higher at some points in the modified DiffQ congestion control than in the usual TCP congestion control.



The delivery ratio drops in both cases with increasing node count, the modified having a lower ratio at every point.



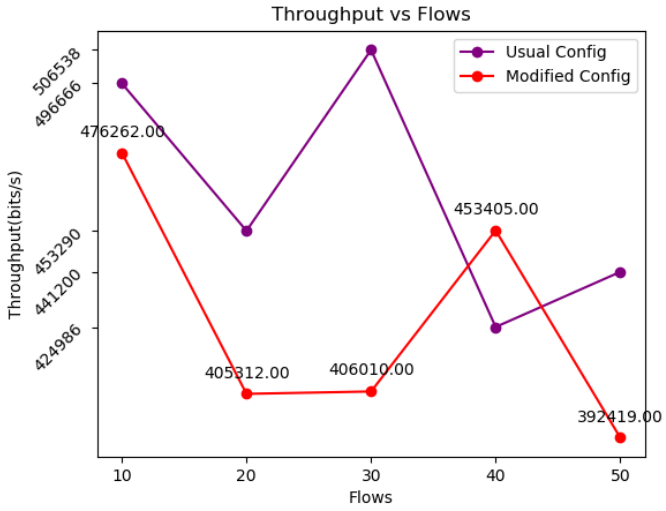
The drop ratio increases with increasing node count in both congestion controls.



The rate of energy consumption in DiffQ congestion control lies at a higher range than the original TCP control with DropTail Priority Queue with increasing node count.

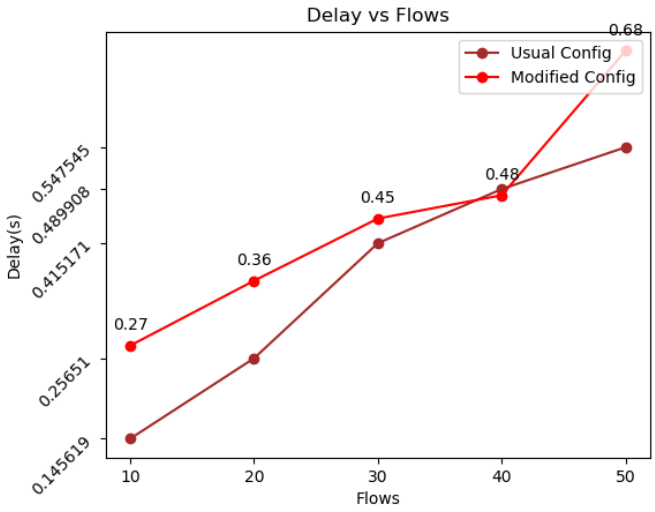
4.2 Graphs - Changing Flow Count

The **5** graphs are shown below:

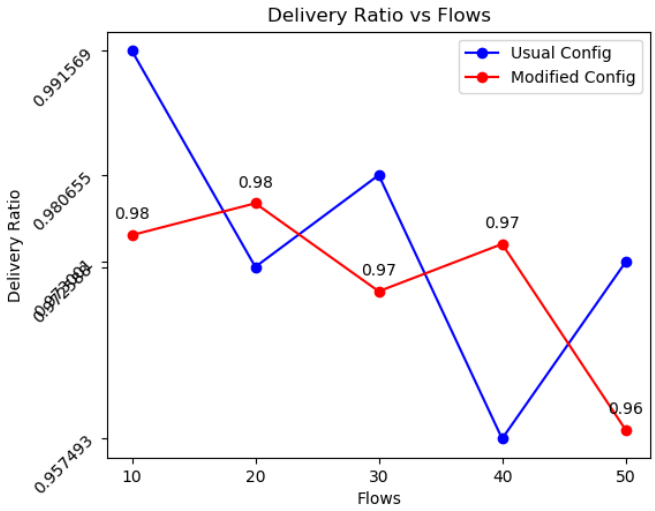


Generally, the throughput decreases with the increase in flow count in the original and modified source code. If closely looked at, the throughput decreases

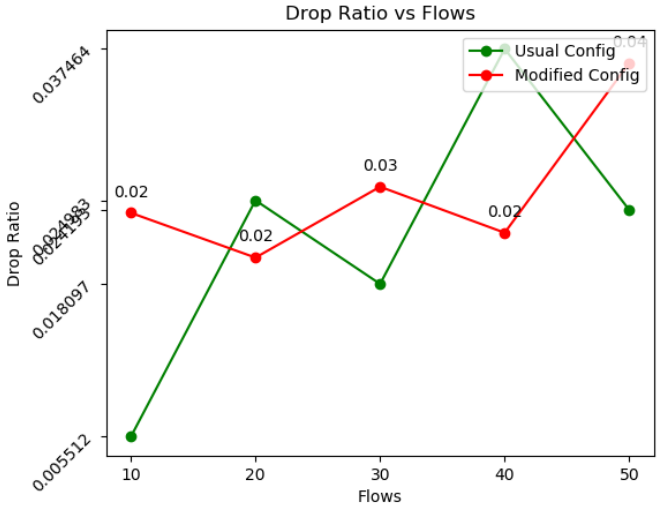
more in the modified source code(red coloured). Most of the time the original source code performs better.



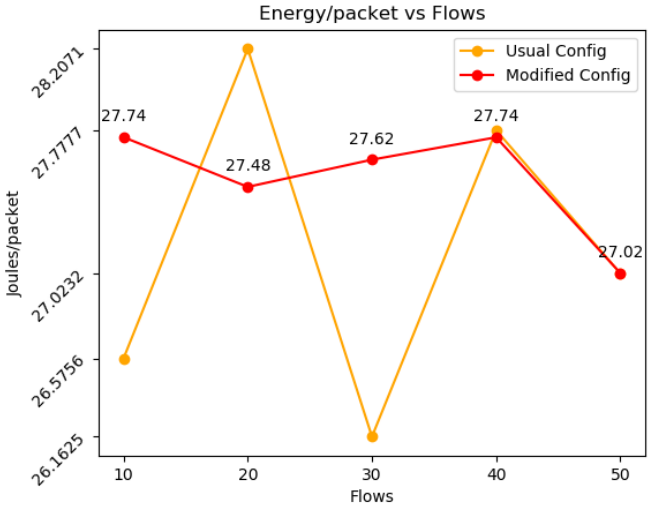
The delay in packet transmission is quite the same in both versions of the code, with subtle differences.



Delivery ratio drops in general with increasing flow count.



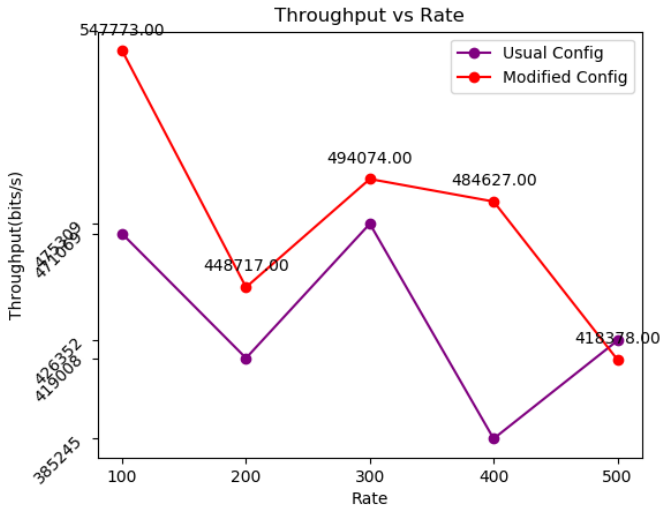
The drop ratio increases in general with increasing flow count.



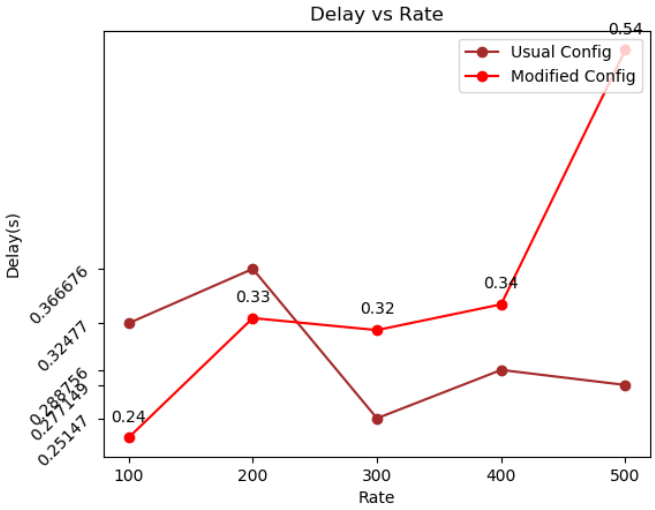
The energy consumption rate in DiffQ congestion control remains at a constant range while it has ups and downs in the TCP congestion control with DropTail Priority Queue.

4.3 Graphs - Changing Packet Rate

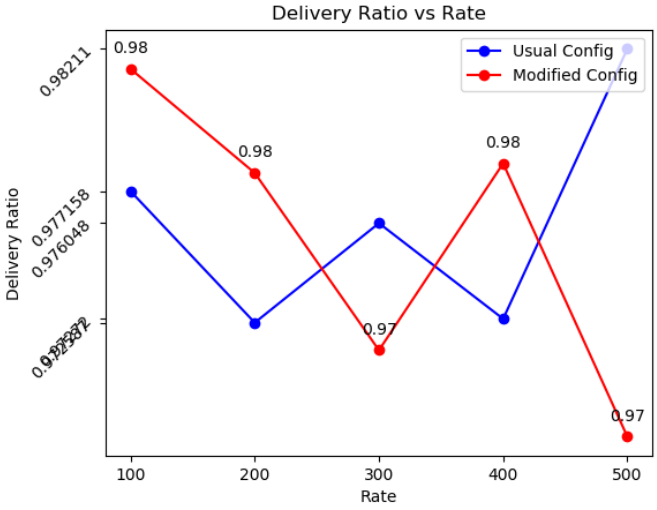
The **5** graphs are shown below:



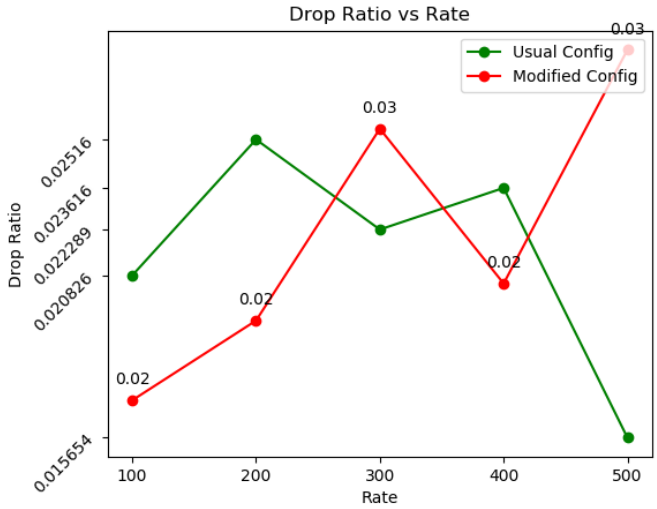
The throughput is on the rise for original source code but declines for the modified source code when the packet rate is increased. They have the highest contrast at a packet rate of 300 but after that, they increase, even though original source code performs better.



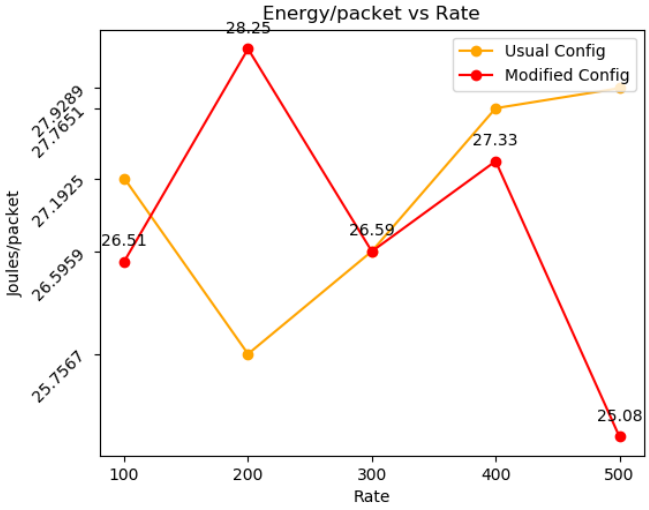
With an increasing rate of packet transmission, the delay in packet transmission continuously grows in both cases.



The delivery ratio in both cases has no definite pattern. It has ups and downs.



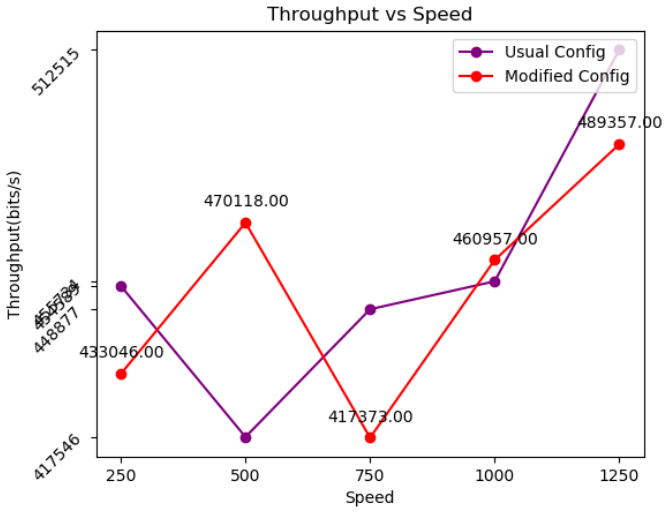
The drop ratio in both versions of the code has no definite pattern. Sometimes it grows and sometimes it falls.



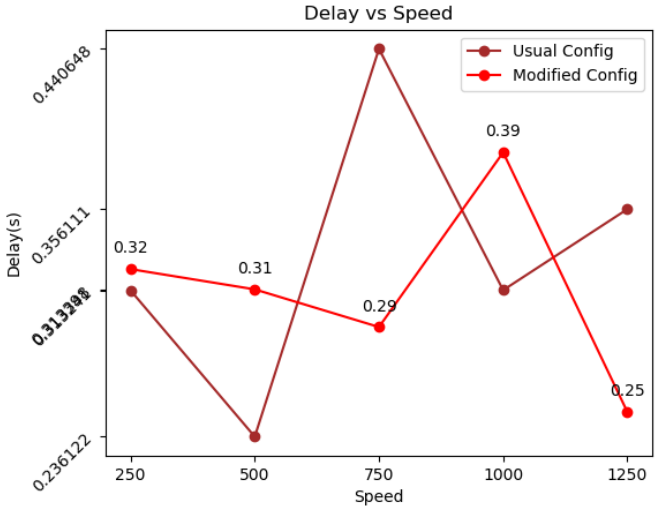
The same goes for the energy consumption rate as it has ups and downs in both versions of the code.

4.4 Graphs - Changing Node Speed

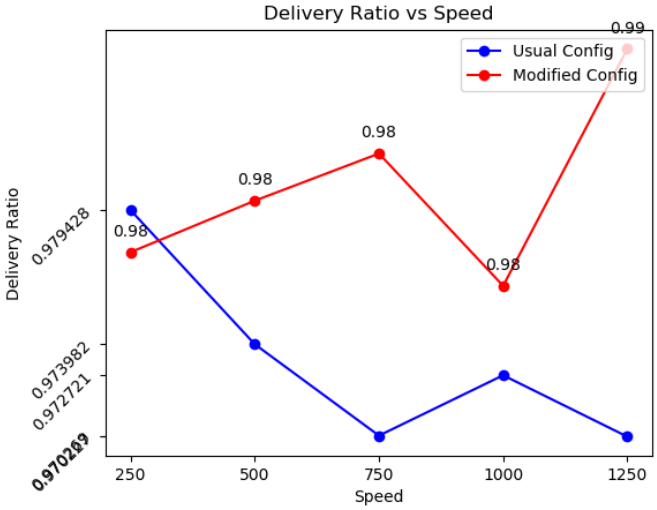
The **5** graphs are shown below:



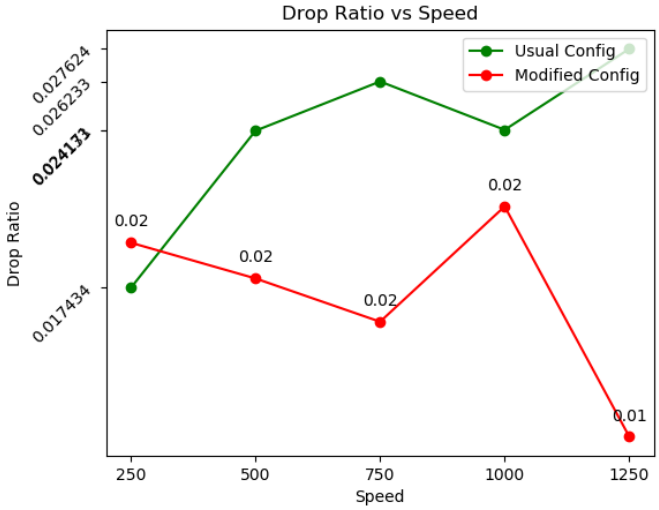
The modified source code gives high throughput when the Tx range is on the rise. The cause is the updated DiffQueue which is used to control the congestion and reduce packet drop and delay.



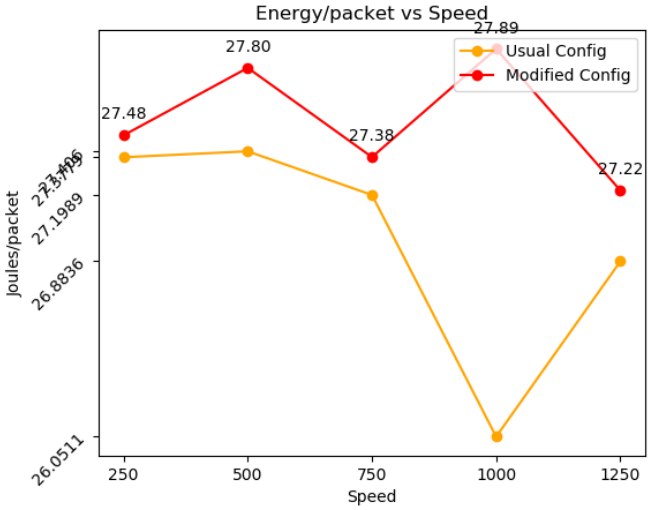
The delay in the usual config has a criss-cross pattern where it sometimes grows and sometimes falls. In the modified config, the delay falls gradually, with a spike in between.



Although the delivery ratio is higher in both cases, the modified version of the code lies at a higher range than the original version.



The drop ratio has a subtle gap between the two versions. In the original version, we can say that it gradually increases while in the modified version it gradually decreases.



The energy consumption rate is a bit higher in the modified version of the code than in the original version and is on the rise with the increasing speed of the nodes.

5 Summary

As the DiffQ algorithm is added on top of the existing TCP algorithm, they most of the time lie side by side in the graphs. But one noticeable thing from the graphs is that the DiffQ algorithm takes up more energy per packet than the existing one. In some cases, it shows a better Delivery ratio and less Drop ratio owing to the fact that packets are now handled in the queues of agents, and only the packet with the highest DiffQ priority whose release will not add to the congestion of the network is dequeued. Moreover, increasing node and flow count decreases the throughput of the network. As a whole, DiffQ congestion control separately cannot compete with TCP but when merged, it will work almost like TCP, sometimes better than TCP.