Report

Adaptive mean queue size and its rate of change: queue management with random dropping

Date of Submission: 27th February 2023

Submitted by:

Muhammad Ehsanul Kader
1805067
L-3/T-2
CSE-B1

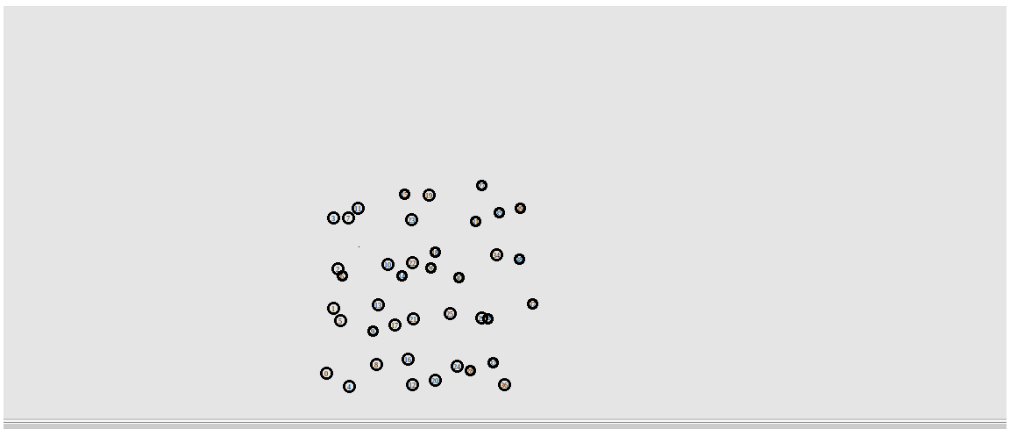
Introduction:

In high-speed networks with connections with large delay-bandwidth products, gateways are likely to be designed with correspondingly large maximum queues to accommodate transient congestion. A queuing technique called Random Early Detection (RED) was proposed for network schedulers to prevent congestion and avoid the need for large queues that are often full. The RED active queue management algorithm allows network operators to simultaneously achieve high throughput and low average delay. The random early detection active queue management (AQM) scheme uses the average queue size to calculate the dropping probability in terms of minimum and maximum thresholds. The effect of heavy load enhances the frequency of crossing the maximum threshold value resulting in frequent dropping of the packets. Our project involved the implementation of a novel algorithm, known as Adaptive Queue Management with Random Dropping (AQMRD), that utilizes not only the average queue size but also its rate of change. Additionally, it also incorporates another threshold value called mid threshold and divides the queue size into four segments based on which probability of whether the packet should be dropped or not is calculated" rewrite it.

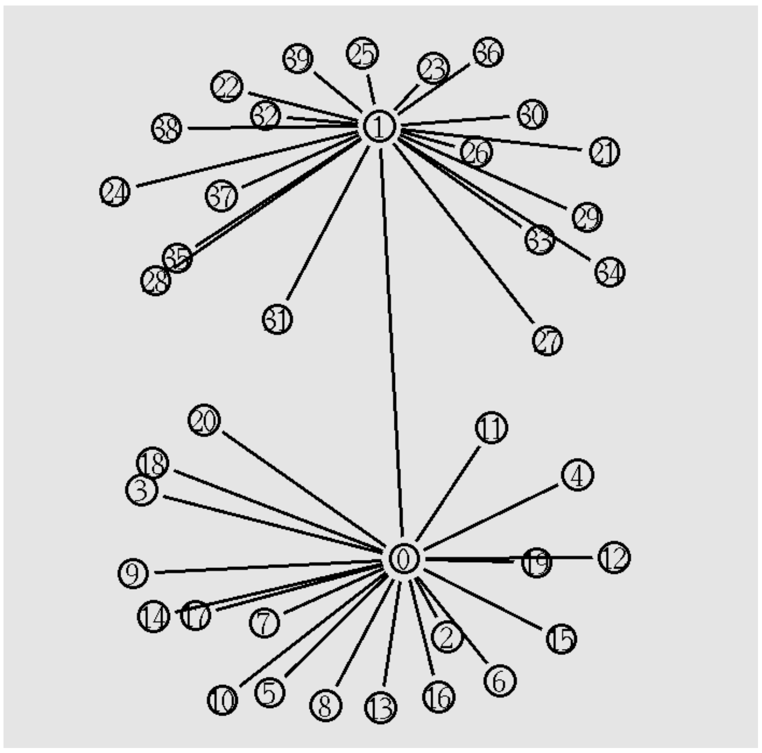
Network Topologies:

The network topology assigned to this project were:

1. Wireless dynamic nodes with MAC 802.11



2. Wired nodes



We have performed necessary simulations and derived results accordingly of these two kinds of networks.

Parameters under variation:

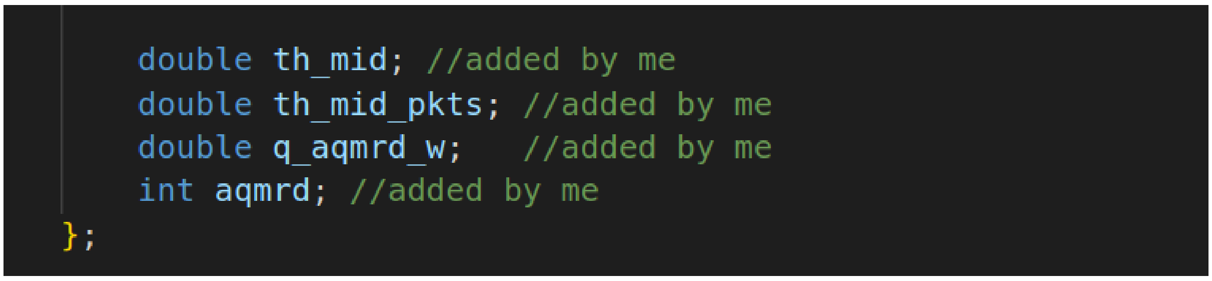
The parameters varied for the topologies in the simulation are as follows:

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. Coverage area: square of Transmission Range times 1, 2, 3, 4, 5
5. 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)

Modifications made:

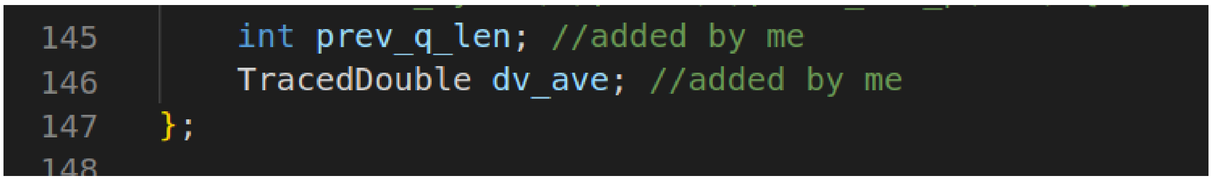
red.h

In *struct edp* following four variables were added:



```
    double th_mid; //added by me
    double th_mid_pkts; //added by me
    double q_aqmrd_w;   //added by me
    int aqmrd; //added by me
};
```

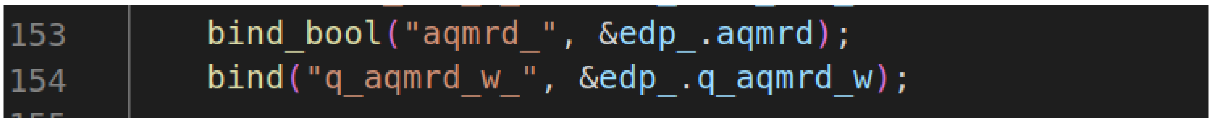
In *struct edv* following two variables were added:



```
    int prev_q_len; //added by me
    TracedDouble dv_ave; //added by me
};
```

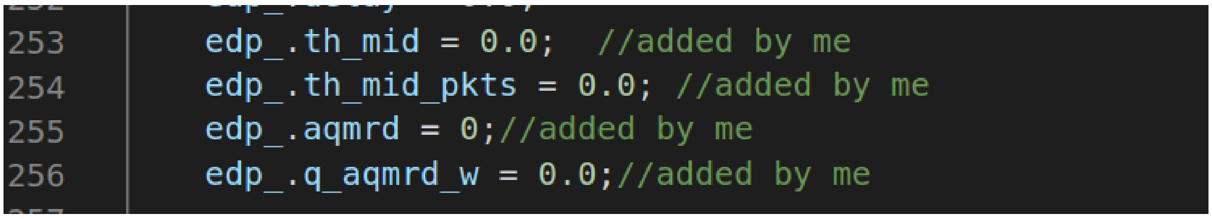
Red.cc

To pass parameter from tcl file to the class, the following code is added in *REDQueue::REDQueue()*

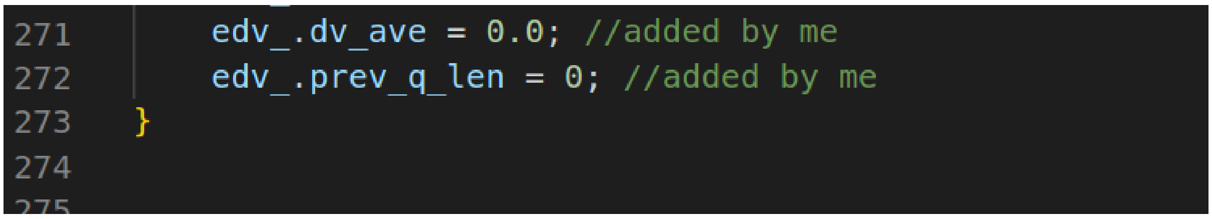


```
    bind_bool("aqmrd_", &edp_.aqmrd);
    bind("q_aqmrd_w_", &edp_.q_aqmrd_w);
```

The new parameters are initialized in *REDQueue::initparams()*



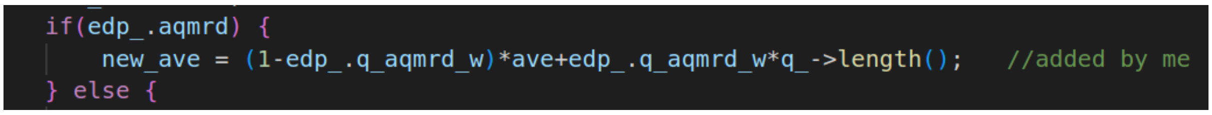
```
    edp_.th_mid = 0.0;  //added by me
    edp_.th_mid_pkts = 0.0; //added by me
    edp_.aqmrd = 0;//added by me
    edp_.q_aqmrd_w = 0.0;//added by me
```



```
    edv_.dv_ave = 0.0; //added by me
    edv_.prev_q_len = 0; //added by me
}
```

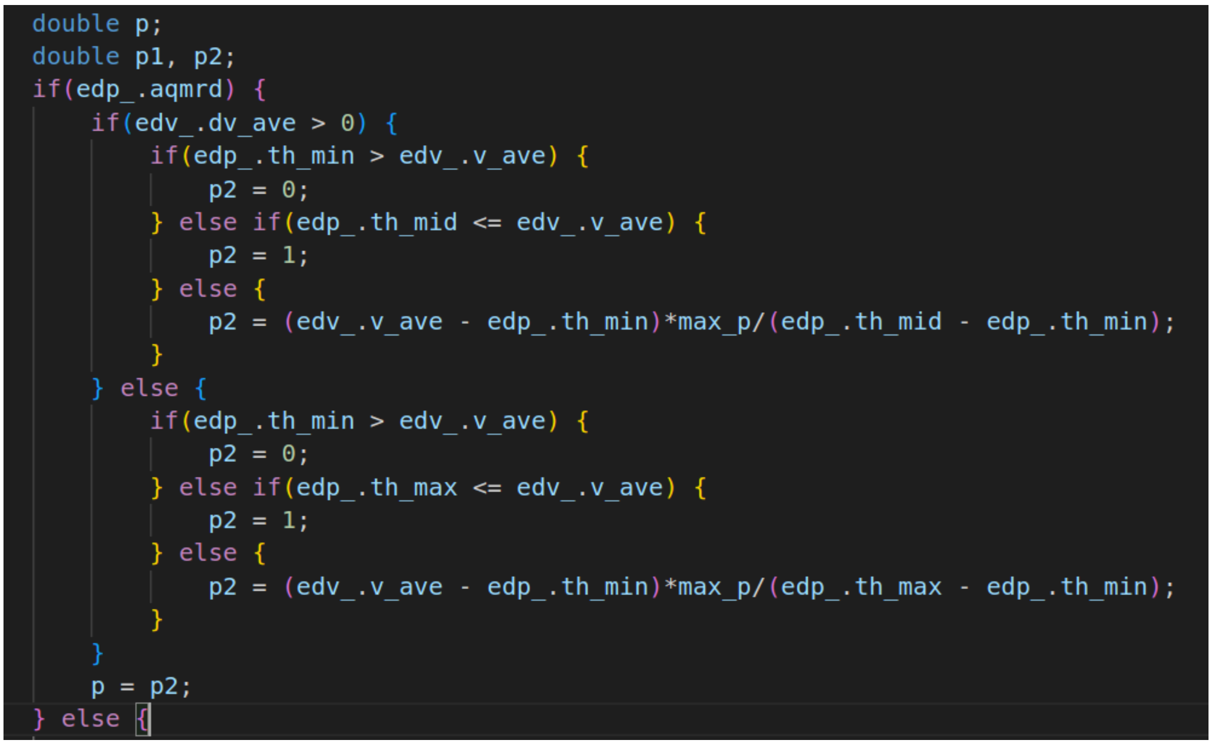
If else blocks were added where if edp_.aqmrd is true everything is done according to new algorithm, otherwise old algorithm is used.

To compute average queue size according to the new algorithm line inside if block were added int *REDQueue::estimator:*



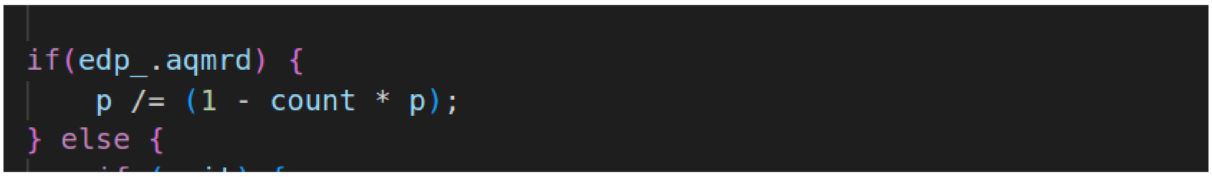
```
    if(edp_.aqmrd) {
        new_ave = (1-edp_.q_aqmrd_w)*ave+edp_.q_aqmrd_w*q_->length();   //added by me
    } else {
```

The *REDQueue::calculate_p_new()* is modified to calculate the probability function P_b of AQMRD



```
double p;
double p1, p2;
if(edp_.aqmrd) {
    if(edv_.dv_ave > 0) {
        if(edp_.th_min > edv_.v_ave) {
            p2 = 0;
        } else if(edp_.th_mid <= edv_.v_ave) {
            p2 = 1;
        } else {
            p2 = (edv_.v_ave - edp_.th_min)*max_p/(edp_.th_mid - edp_.th_min);
        }
    } else {
        if(edp_.th_min > edv_.v_ave) {
            p2 = 0;
        } else if(edp_.th_max <= edv_.v_ave) {
            p2 = 1;
        } else {
            p2 = (edv_.v_ave - edp_.th_min)*max_p/(edp_.th_max - edp_.th_min);
        }
    }
    p = p2;
} else {
```

The *REDQueue::modify_p()* is modified to calculate the probability function P_a of AQMRD



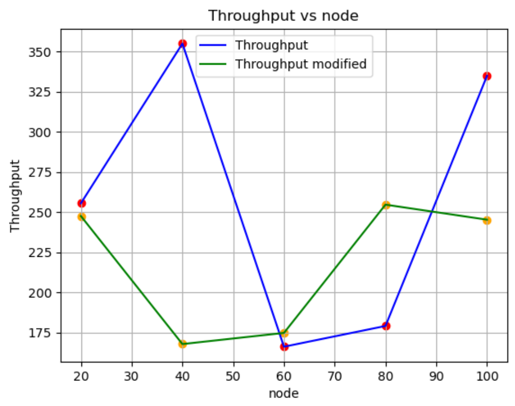
```
if(edp_.aqmrd) {
    p /= (1 - count * p);
} else {
```

Results with graphs:

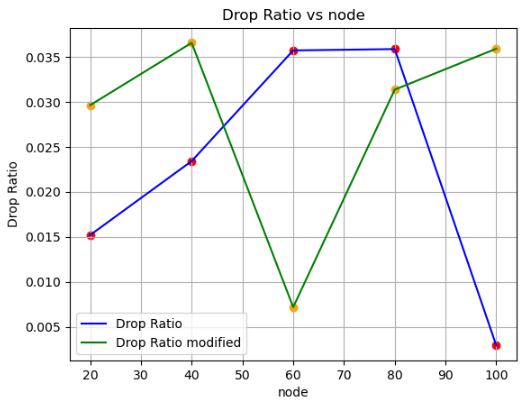
1. Wireless 802.11 mobile nodes:

With respect to Number of Nodes:

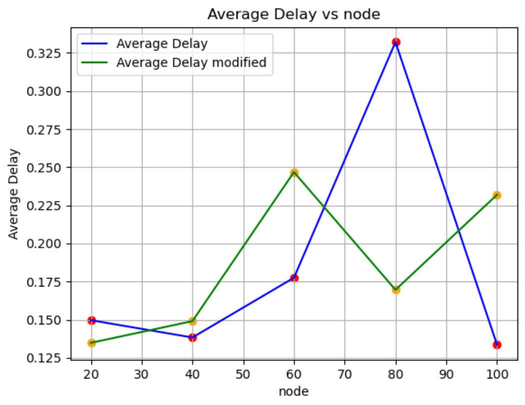
Network Throughput:



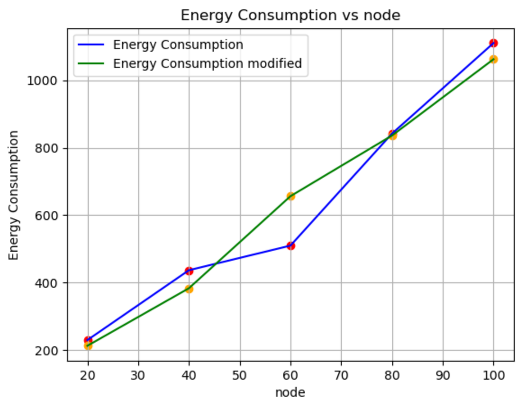
Packet Drop Ratio:



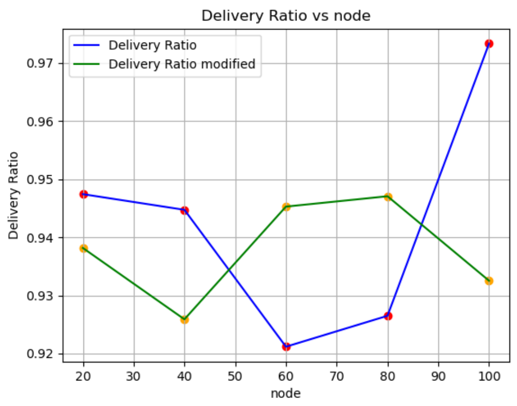
End-to-End Delay:



Energy Consumption:

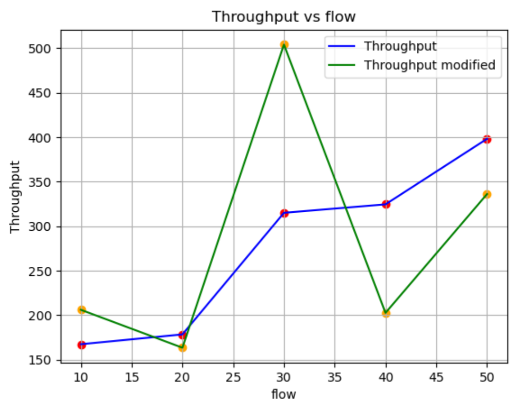


Packet Delivery Ratio:

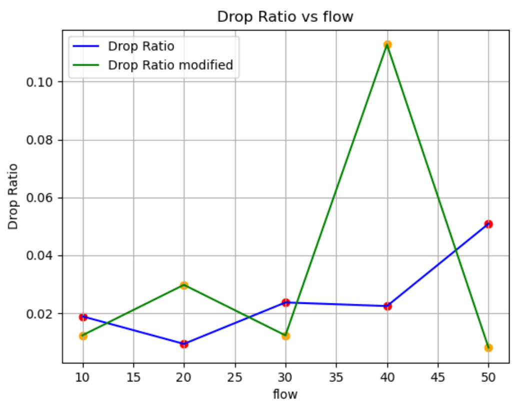


With respect to Number of Flows:

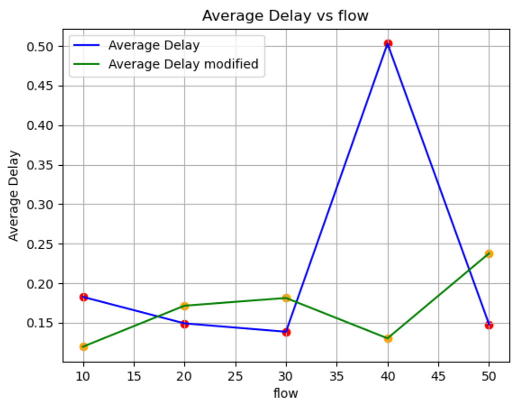
Network Throughput:



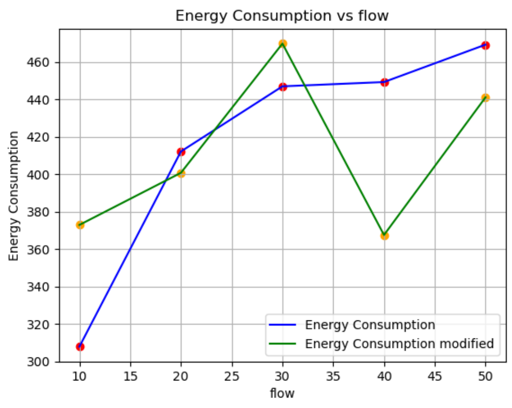
Packet Drop Ratio:



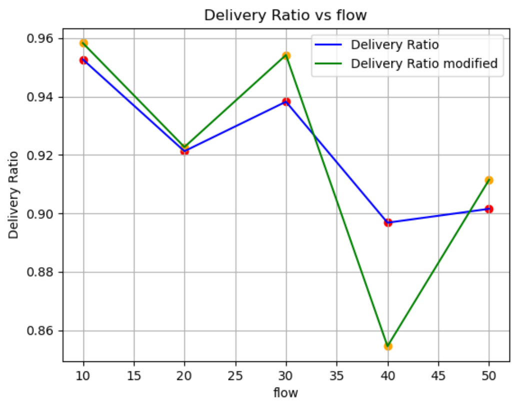
End-to-End Delay:



Energy Consumption:

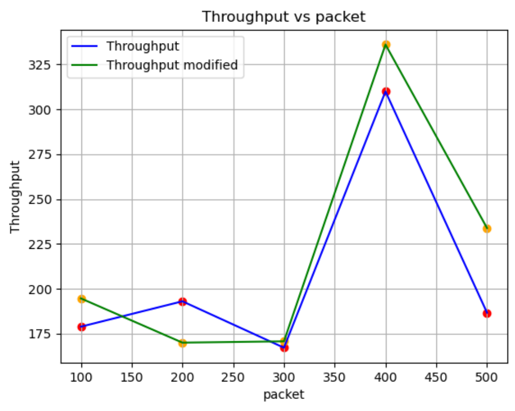


Packet Delivery Ratio:

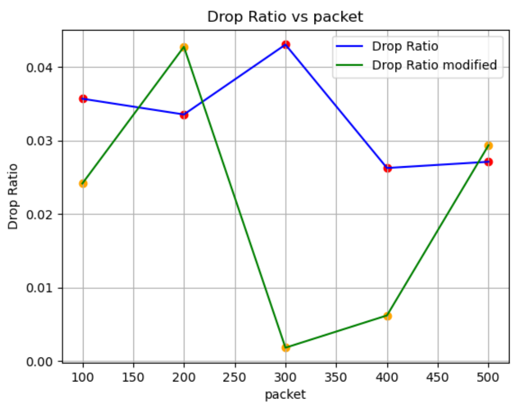


With respect to Number of Packets per second:

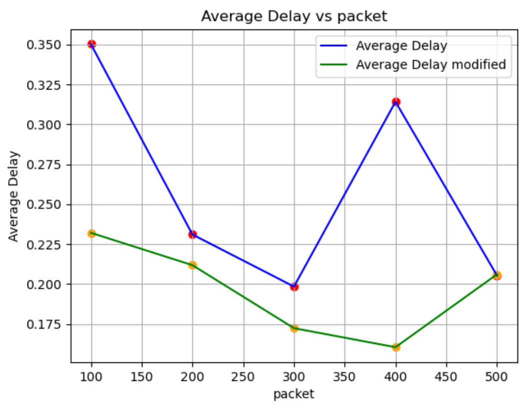
Network Throughput:



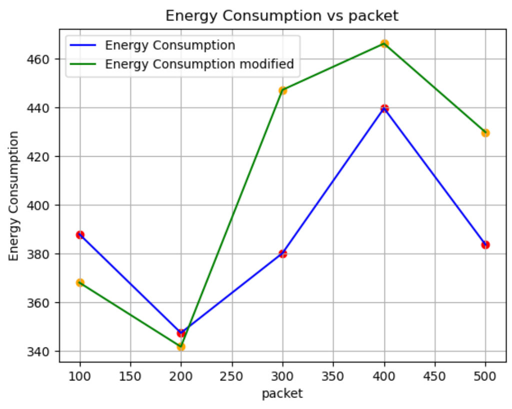
Packet Drop Ratio:



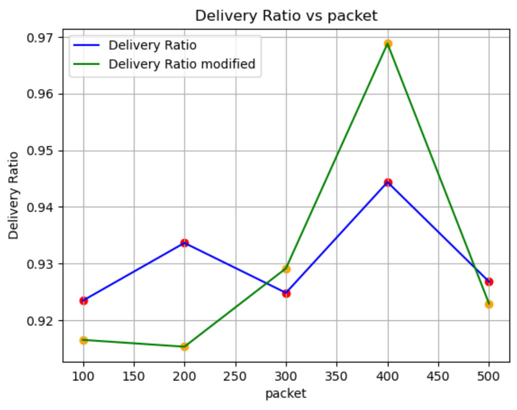
End-to-End Delay:



Energy Consumption:

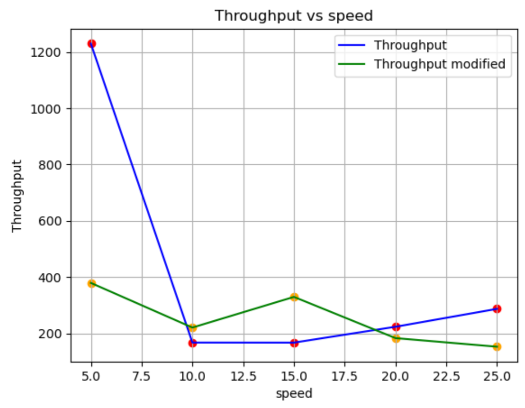


Packet Delivery Ratio:

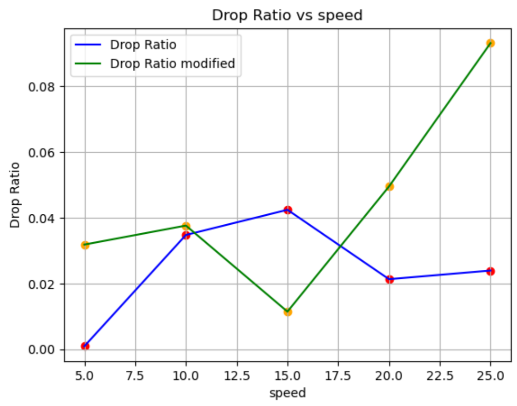


With respect to speed:

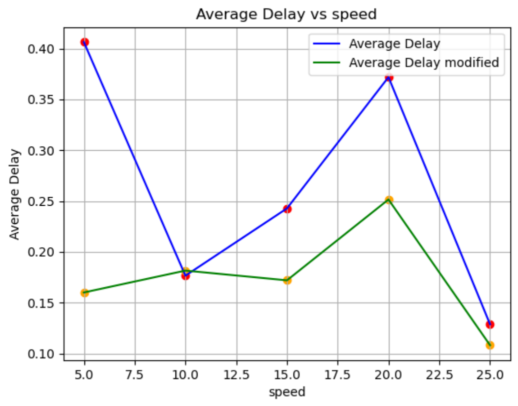
Network Throughput:



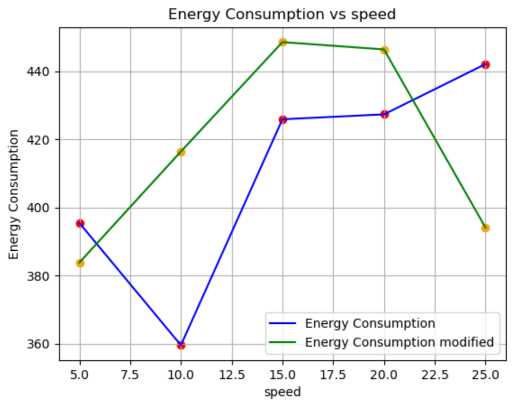
Packet Drop Ratio:



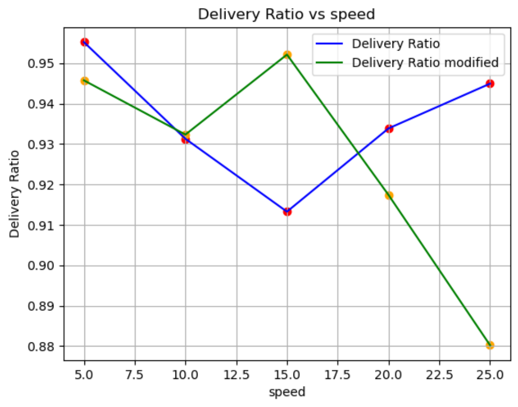
End-to-End Delay:



Energy Consumption:

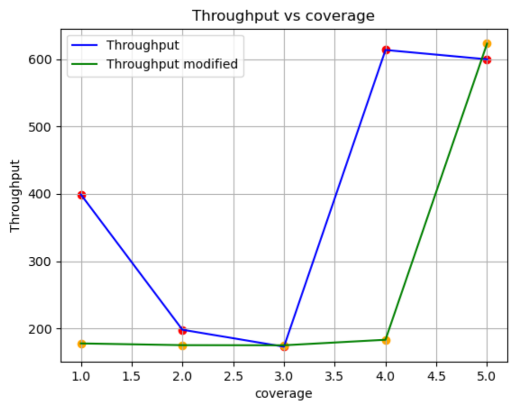


Packet Delivery Ratio:

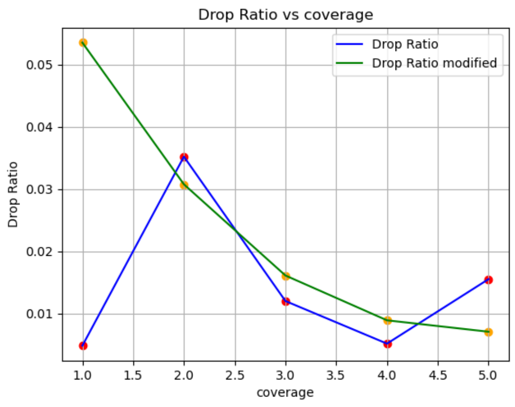


With respect to Coverage Area:

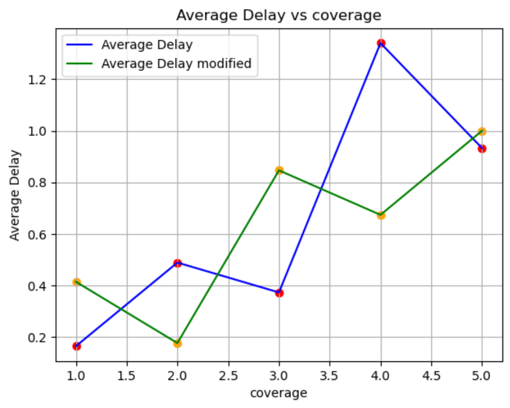
Network Throughput:



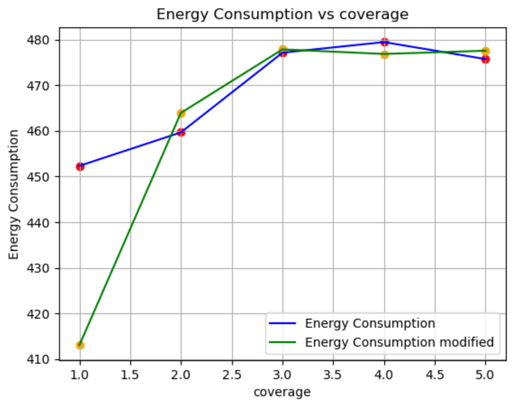
Packet Drop Ratio:



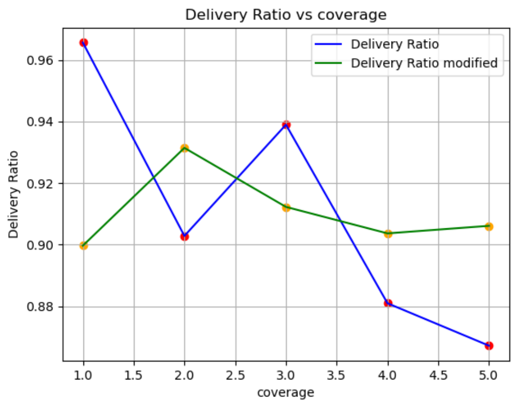
End-to-End Delay:



Energy Consumption:



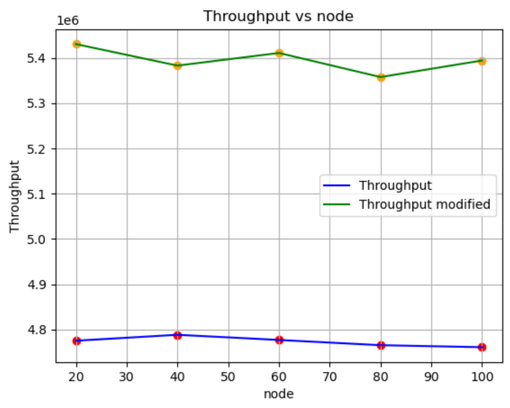
Packet Delivery Ratio:



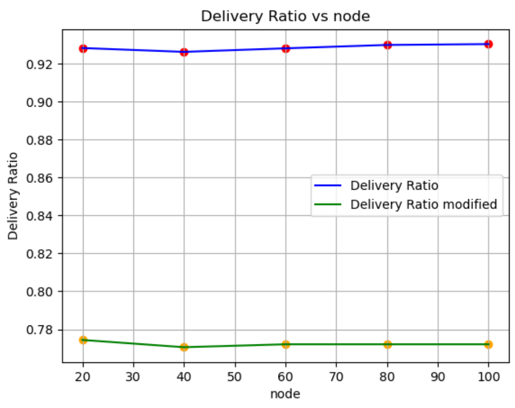
2. Wired nodes:

With respect to Number of Nodes:

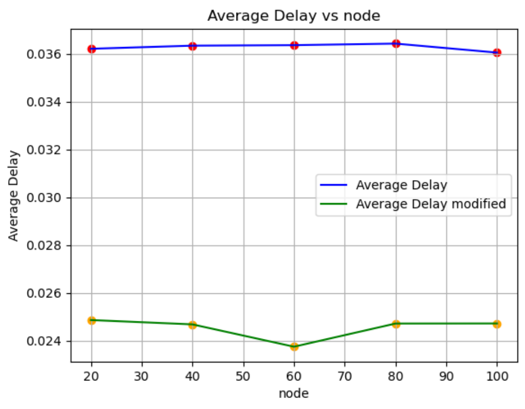
Network Throughput:



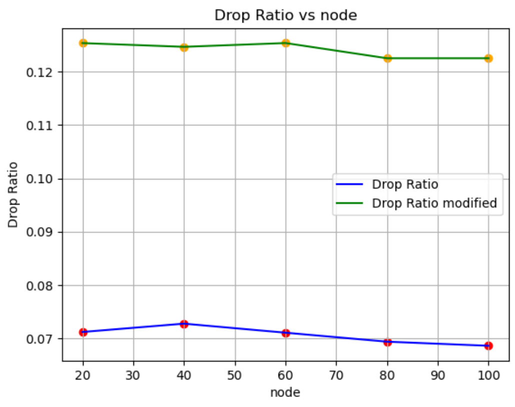
Packet Delivery Ratio:



End-to-End Delay:

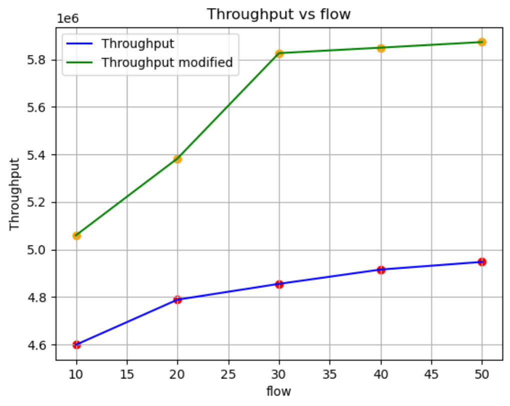


Drop Ratio:

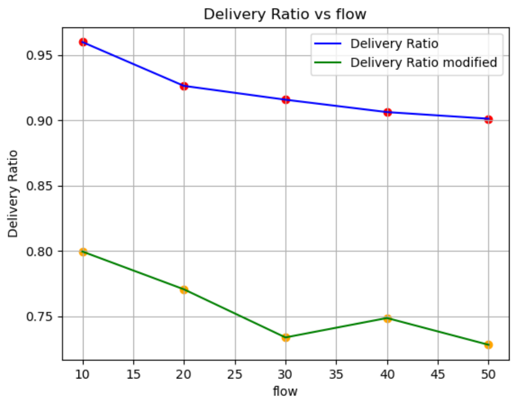


With respect to Number of Flows:

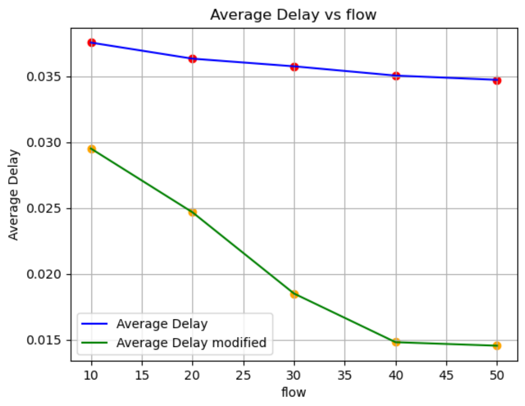
Network Throughput:



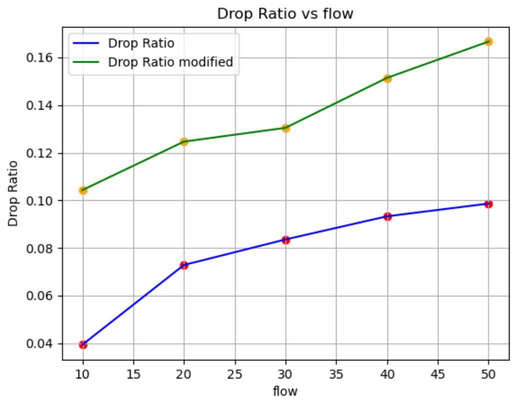
Packet Delivery Ratio:



End-to-End Delay:



Drop Ratio:



Observation:

The analysis shows that the proposed algorithm has better end-to-end average delay performance in wireless networks but slightly worse drop ratio and lower throughput. In terms of energy consumption, both the old and new algorithms have similar performance on average.

Meanwhile, in wired networks, the algorithm achieves higher throughput and better end-to-end delay but has poor delivery ratio and drop ratio performance. These findings are significant for optimizing network performance and reliability.