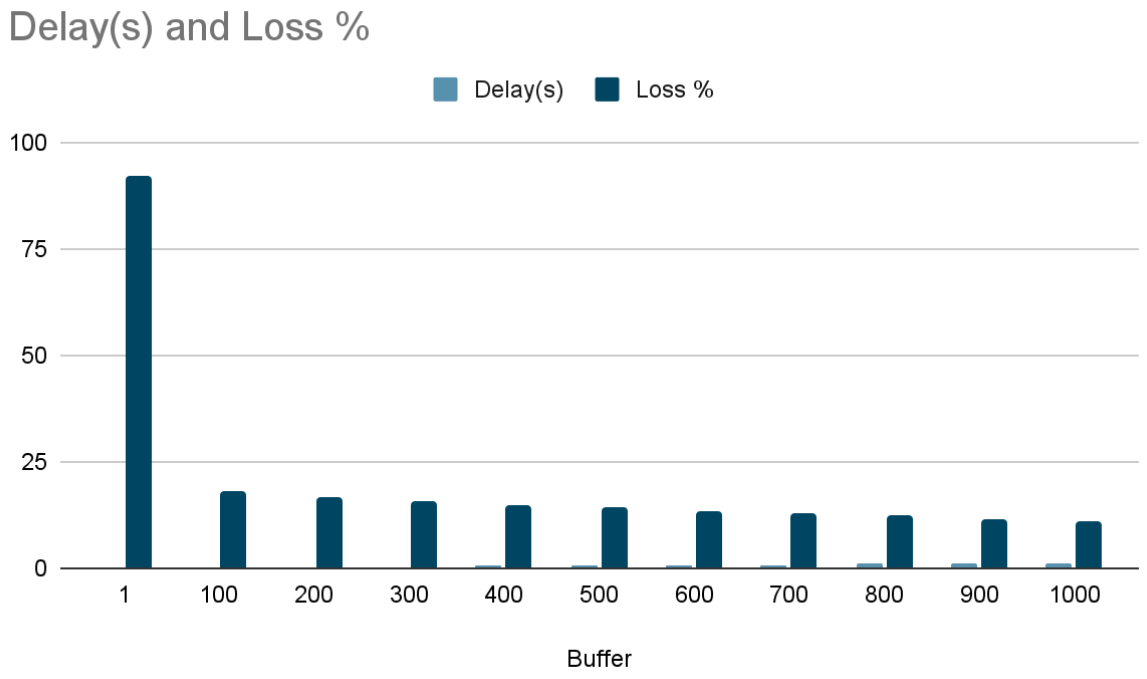
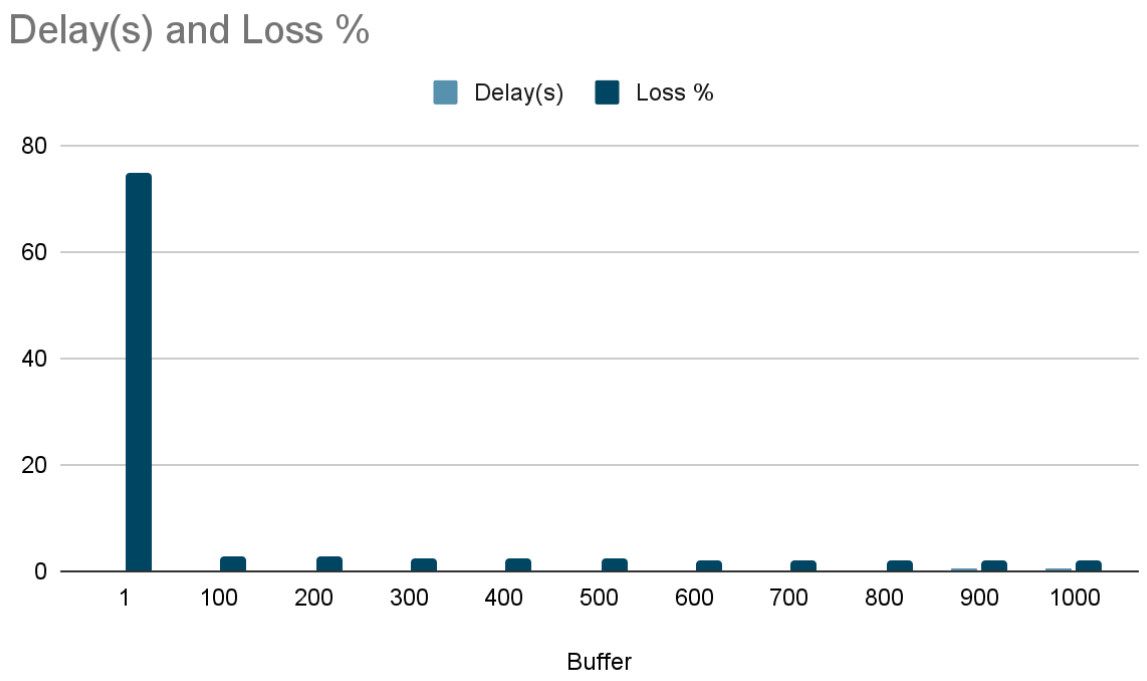
1
Stream.txt
5Mbps

Buffer	Delay(s)	Loss %
1	0	92.113
100	0.122914	17.9561
200	0.246711	16.7567
300	0.371579	15.8017
400	0.492578	14.9501
500	14.2014	0.610558
600	0.72695	13.4942
700	0.839353	12.8341
800	0.949098	12.248
900	1.06112	11.7147
1000	1.18222	11.2638



Zoom.txt
5Mbps

Buffer	Delay(s)	Loss %
1	0	74.8839
100	0.0426761	2.89658
200	0.0822907	2.67886
300	0.119816	2.51374
400	0.156904	2.38132
500	0.199489	2.28634
600	0.244616	2.2075
700	0.290403	2.13217
800	0.33514	2.07241
900	0.3798	2.009
1000	0.424123	1.96032



Both steam and zoom has very high loss at B = 1 but there are no time delay but it won't be efficient since there are too many loss packets that we have to send them again.
After 100 packets, stream and zoom both has significantly low loss rate and slowly decreasing in loss rate as packets increased by 100. From 100 packet for stream, it has about 17.96 % loss rate and 1000 packets, it has about 11.26%, it is decreased only by 6.7 % which is about 37% increased efficiency while B is increased by 10 times. Also delay is increased by 10 times compared to 100 and 1000 which isn't ideal to increase buffer if it cost alot of money. Same for Zoom that delay is increased about 10 times but zoom already has low loss rate after 100. From 100 to 1000, it only decreased from 2.89% to 1.96%, which has increased efficiency of 32%. Stream is more challenging since there are more packet loss and higher delay which is harder to handle. Zoom has very low loss rate and low delay that so that loss rate wouldn't matter, there probably no needs to decrease even more from there.

2.
Stream.txt
B = 100

WLAN(Mbps)	Delay(s)	Loss %
5	0.122914	17.9561
6	0.0824071	10.7597
8	0.0366274	3.82983
10	0.0216178	1.66872

Zoom.txt
B = 100

WLAN(Mbps)	Delay(s)	Loss %
5	0.0426761	2.89658
6	0.0142326	0.736069
8	0.0051496	0.0368946
10	0.00289724	0

```
// while event is empty
   while (!events.empty()) {
       Event event = events.top();
       events.pop();

       double event_time = event.time;
       Packet& packet = event.packet;

       switch (event.type) {
           case PACKET_ARRIVAL:
               // all the packets are added to the packet arrival
               totalPacketsIn++;
               totalPacketsInByte += packet.size;

               // if buffer can take packets
               if (buffer.size() < bufferSize) {
                   buffer.push(packet);

                   // if buffer has size of 1, it goes to the packet
departure
                   if (buffer.size() == 1) {
                       // calculate exact time of packet that goes to
the packet_departure
                       packet.departureTime = event_time + (packet.size
* 8.0 / (wlan * 1000000));
                       events.push({PACKET_DEPARTURE,
packet.departureTime, packet});
                   }

               }

               //if buffer is full then we drop packets
               else {
                   events.push({PACKET_DROP, event_time, packet});
               }
               break;


           case PACKET_DEPARTURE:
           {
               // get current time
               packet.arrivalTime = event.time;
```

```
                // receive departure time from arrival so current time -
departureTime will give queuing delay
                totalQueuingDelay += packet.arrivalTime -
packet.departureTime;
                // calculating number of packets that come to packet
departure
                totalPacketsOut++;
                totalPacketsOutByte += packet.size;
                // remove front packet
                buffer.pop();
                // if buffer is not empty
                if (!buffer.empty()) {
                    // next packets to go to calculate departuretime and
push that to departure
                    Packet& next_packet = buffer.front();
                    // updating the departure time for the next packet
so we know the next packet time now for each packets
                    next_packet.departureTime = event_time +
(next_packet.size * 8.0 / (wlan * 1000000));
                    events.push({PACKET_DEPARTURE,
next_packet.departureTime, next_packet});
                    totalQueuingDelay += packet.arrivalTime -
buffer.front().timestamp;
                }
            }
            break;
            // when buffer is full, it drops packets that calculating
number of packets lost
            case PACKET_DROP:
                totalPacketsLost++;
                totalPacketsLostByte += packet.size;
                break;
        }
    }
```

We first put all the packets in the packet_arrival since all the packets needs to be at there so we calculate the total packets .
If buffer size is lower than the max buffer size (B) we get the packets in the buffer and if buffer size is 1 which means that once we get the packet we send that to the Packet_departure. If buffer is full then we will send those packets to the packet drop. In the packet_departure, we will be calculating the number of packets thats is sent and we pop once it has been done. In the packet drop, we are calculating the number of packets those are lost.

It has way more efficient to buy bandwidth than buffer since as Mbps increased by 20 percent from 5 to 6, delay decreased by 67% for zoom and loss rate also decreased by 75% Stream doesn’t have same rate of decrease however as it has has way more efficiency than increasing buffer. By only increasing twice of Mbps stream has 6 times shorter delay time also more than 10 times decreasing loss rate. While even increasing the buffer by 10 times only decreased rate by 37 percent but it increased delay time. Increasing Mbps has benefit of decreasing the delay time and decrease loss rate however increasing buffer only decrease little bit of loss rate and increasing the delay.