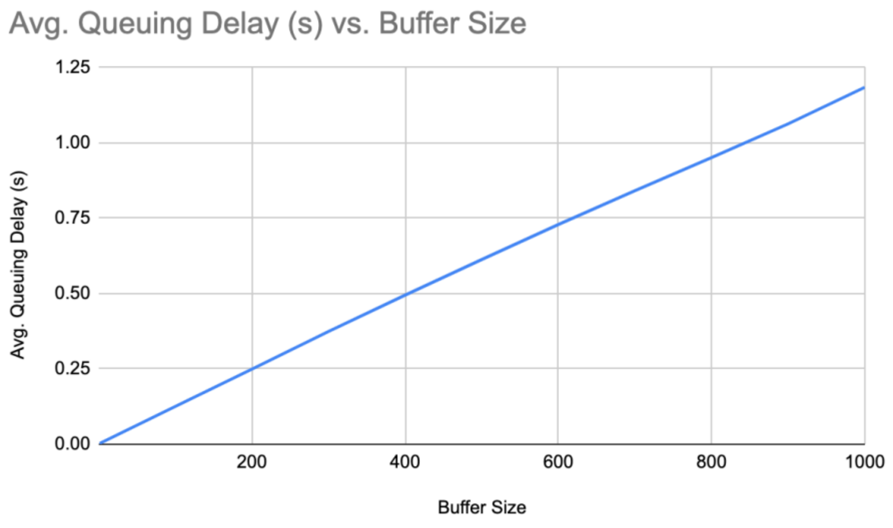
Router Queue Simulation

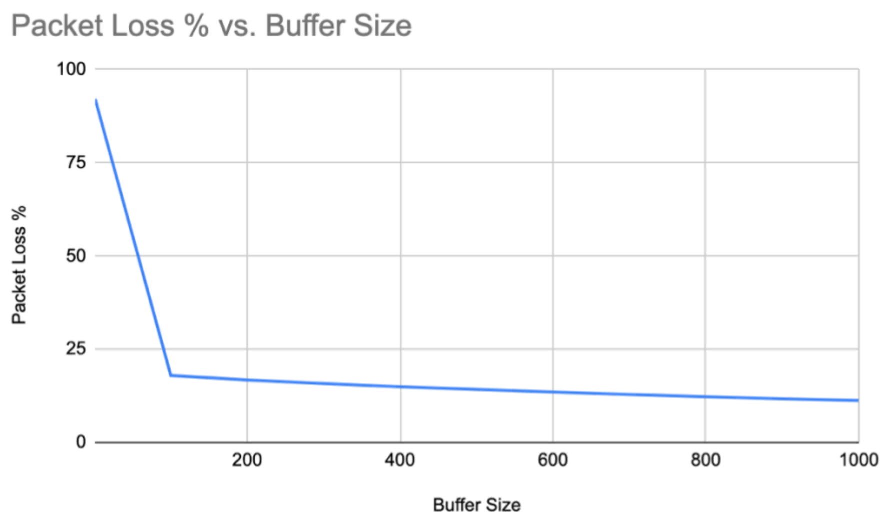
By: Mahtab Khan

Scenario 1: R = 5 Mbps, B will be varied.

For stream.txt trace:

B	Avg. Queuing Delay (s)	Packet Loss %
1	0.00238016	92.1131
100	0.125218	17.9561
200	0.249015	16.7567
300	0.373883	15.8017
400	0.494882	14.9501
500	0.612862	14.2014
600	0.729254	13.4942
700	0.841657	12.8341
800	0.951402	12.248
900	1.06343	11.7147
1000	1.18452	11.2638

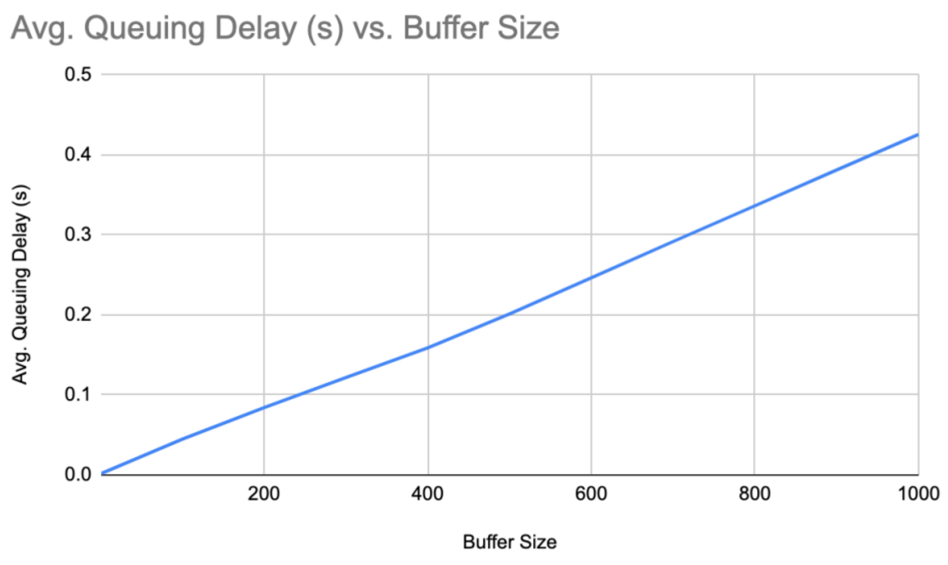


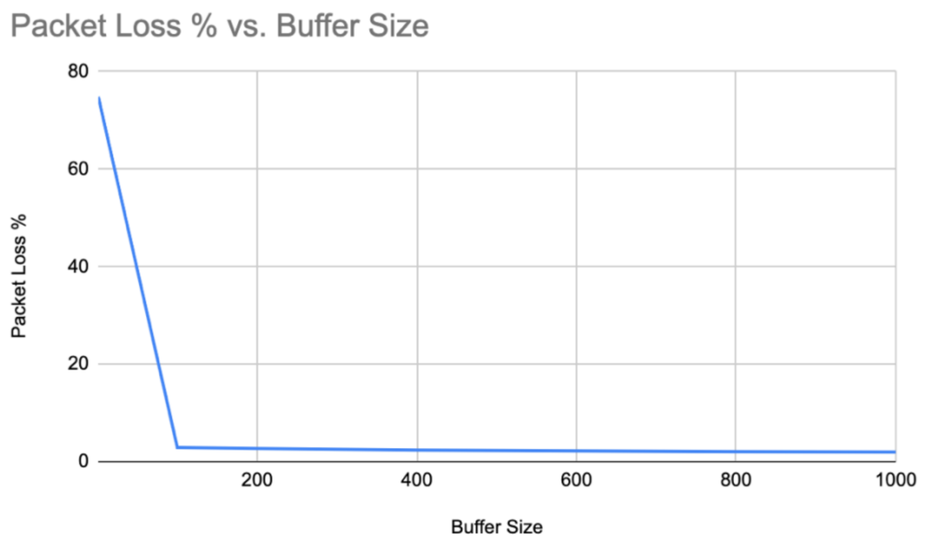


Comments on the results: The average queuing delay exhibits a consistent upward trend with an increase in the buffer size, indicating a linear relationship. Conversely, the packet loss percentage demonstrates a diminishing decrease as the buffer size expands. Graphical representations emphasize the linear increase in average queuing delay concerning buffer size, while the reduction in packet loss percentage diminishes notably after a buffer size of 100. Consequently, a buffer size of **200** emerges as an optimal choice at a 5 Mbps WLAN capacity, striking a favorable balance between delay and loss.

For zoom.txt trace:

B	Avg. Queuing Delay (s)	Packet Loss %
1	0.00187509	74.8511
100	0.044616	2.897
200	0.0842124	2.68124
300	0.121778	2.51753
400	0.158768	2.38286
500	0.201232	2.28887
600	0.246395	2.20049
700	0.291797	2.12867
800	0.336449	2.06245
900	0.38135	2.00704
1000	0.425797	1.95401



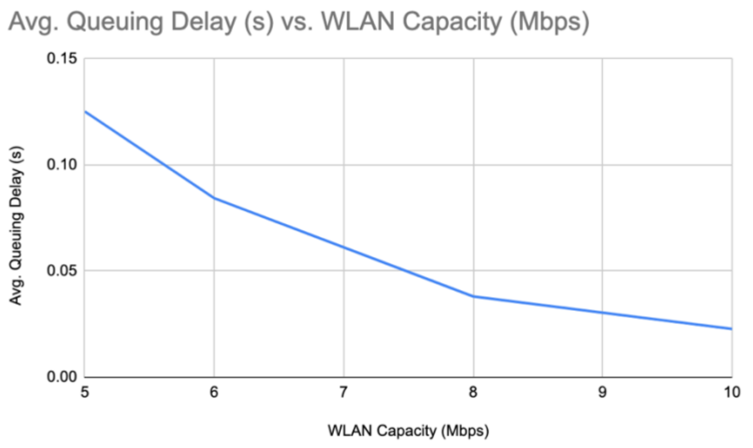


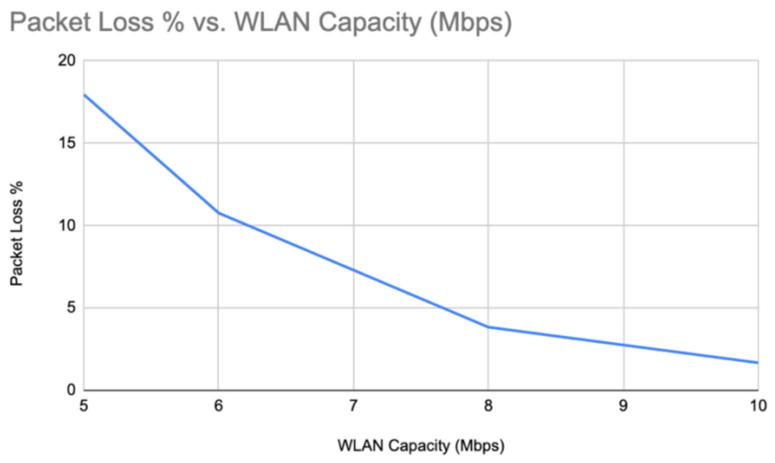
Comments on the results: The average queuing delay exhibits a consistent upward trend with an increase in the buffer size, indicating a linear relationship. Conversely, the packet loss percentage demonstrates a diminishing decrease as the buffer size expands. A buffer size of **200** emerges as an optimal choice at a 5 Mbps WLAN capacity, striking a favorable balance between delay and loss.

Scenario 2: B = 100 packets, R will be varied

For stream.txt:

WLAN Capacity (Mbps)	Avg. Queuing Delay (s)	Packet Loss %
5	0.125218	17.9561
6	0.0843446	10.7597
8	0.0380828	3.82983
10	0.0227822	1.66872

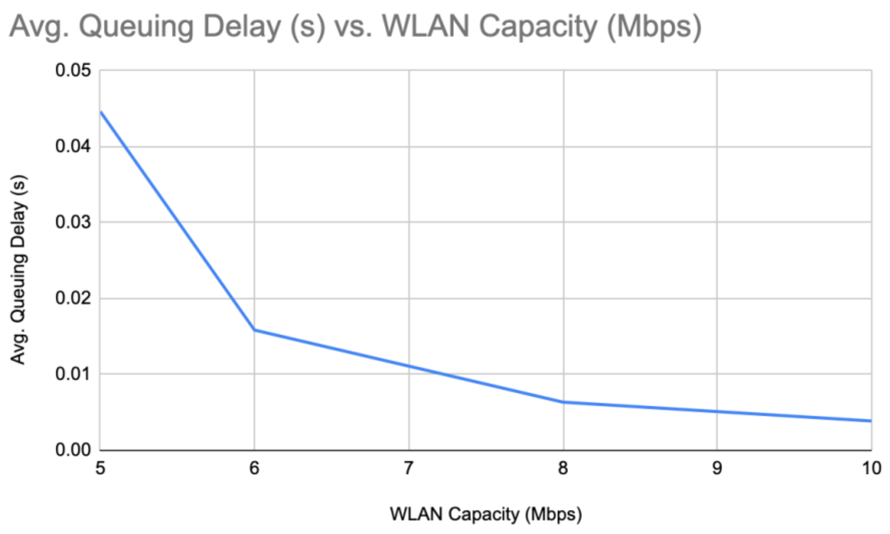


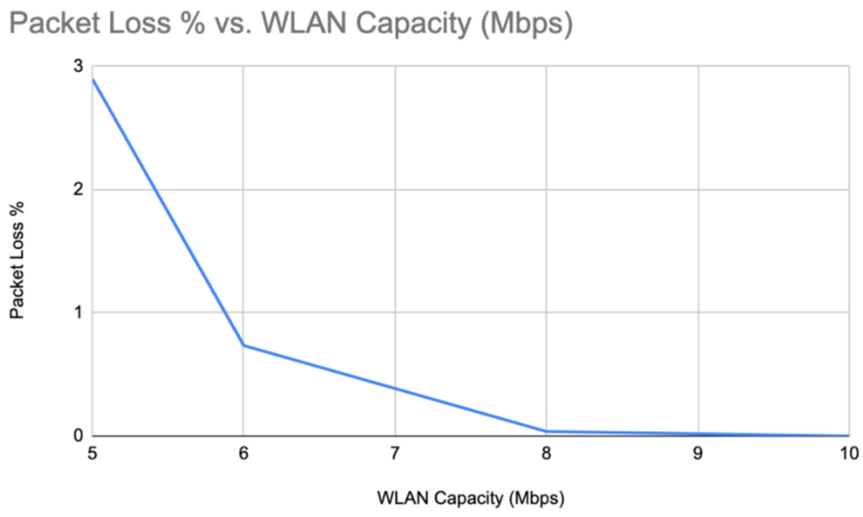


Comments on the result: The table presents a comparative analysis of network performance metrics across different WLAN capacities, ranging from 5 Mbps to 10 Mbps. The average queuing delay, representing the time packets spend in the network queue before transmission, increases with higher WLAN capacities. Conversely, the packet loss percentage demonstrates a diminishing trend as WLAN capacity expands. Notably, at **6 Mbps** WLAN capacity, the system exhibits a balance between queuing delay and packet loss, with an optimal trade-off observed at this specific capacity. The data underscores the critical interplay between WLAN capacity and network performance metrics, aiding in the identification of optimal configurations for minimizing delays and packet losses.

For zoom.txt:

WLAN Capacity (Mbps)	Avg. Queuing Delay (s)	Packet Loss %
5	0.044616	2.897
6	0.0158577	0.735087
8	0.00636103	0.0366141
10	0.00387264	0





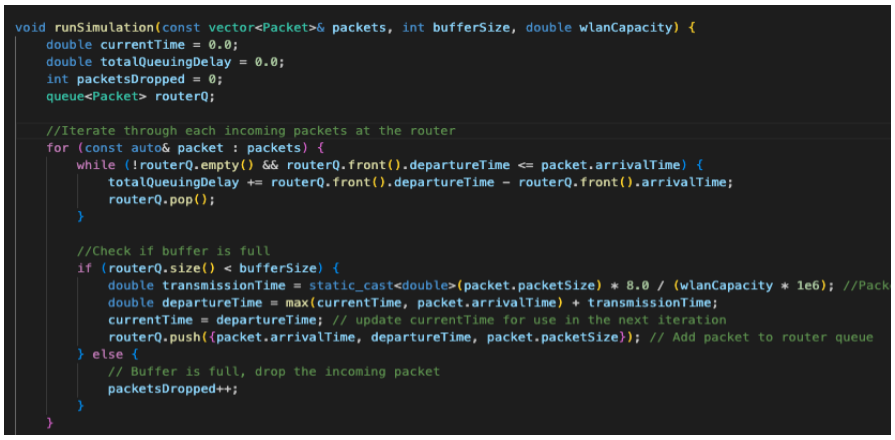
Comments on the result: The average queuing delay, representing the time packets spend in the network queue before transmission, exhibits a substantial decrease as WLAN capacity increases. Simultaneously, the packet loss percentage diminishes, reaching zero at 10 Mbps WLAN capacity. These findings underscore an inverse relationship between WLAN capacity and both queuing delay and packet loss, with higher capacities correlating with enhanced network efficiency and reliability. The data supports the conclusion that an increase in WLAN capacity contributes to a marked reduction in queuing delays and ensures minimal packet loss. Therefore, a WLAN capacity of **6 Mbps** provides an optimum balance between delay and loss.

Conclusion

In the context of streaming, it's crucial to strike a balance between having enough internet speed and a well-sized buffer. In the scenario described in the assignment, where a tournament is being streamed in a boardroom, the expectation is that there won't be excessive network congestion. Therefore, opting for purchasing additional bandwidth is a prudent choice. This decision aims to ensure a seamless streaming experience during the tournament, allowing for smooth playback without interruptions.

How the simulation works

The parseInputFile function reads packet information from a trace file, initializes a Packet object for each line, and stores them in a vector.



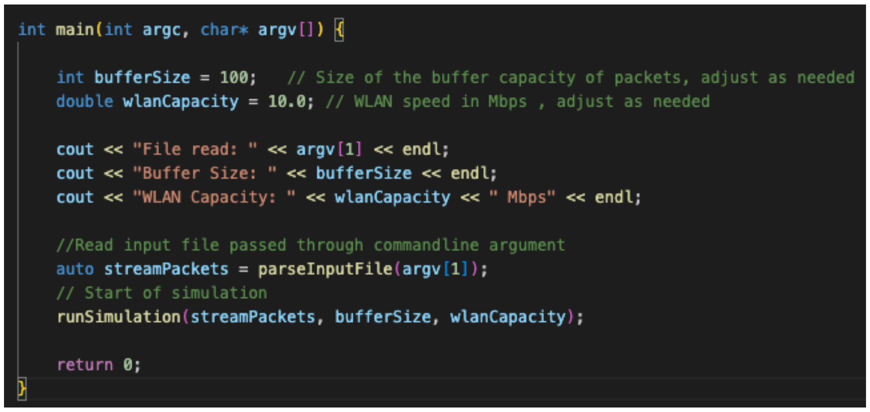
```cpp
void runSimulation(const vector<Packet>& packets, int bufferSize, double wlanCapacity) {
    double currentTime = 0.0;
    double totalQueuingDelay = 0.0;
    int packetsDropped = 0;
    queue<Packet> routerQ;

    //Iterate through each incoming packets at the router
    for (const auto& packet : packets) {
        while (!routerQ.empty() && routerQ.front().departureTime <= packet.arrivalTime) {
            totalQueuingDelay += routerQ.front().departureTime - routerQ.front().arrivalTime;
            routerQ.pop();
        }

        //Check if buffer is full
        if (routerQ.size() < bufferSize) {
            double transmissionTime = static_cast<double>(packet.packetSize) * 8.0 / (wlanCapacity * 1e6); //Pack
            double departureTime = max(currentTime, packet.arrivalTime) + transmissionTime;
            currentTime = departureTime; // update currentTime for use in the next iteration
            routerQ.push({packet.arrivalTime, departureTime, packet.packetSize}); // Add packet to router queue
        } else {
            // Buffer is full, drop the incoming packet
            packetsDropped++;
        }
    }
```

The runSimulation function (as shown above) processes the packets based on the specified buffer size and WLAN capacity. It maintains a queue (routerQ) to simulate the router buffer.

The function (main loop) iterates through each incoming packet in the packets vector. This models the arrival of a packet. Inside the loop, there's a while loop that dequeues packets from the front of the routerQ until the buffer is empty or until the departure time of the front packet is later than the arrival time of the current packet. This simulates the process of packets leaving the buffer as they are transmitted.

If the buffer is full, the incoming packet is not enqueued, and the packetsDropped count is incremented.

Packets are dequeued when their departure time is reached, and new packets are enqueued if the buffer is not full. The function calculates queuing delay and tracks dropped packets.



```cpp
int main(int argc, char* argv[]) {

    int bufferSize = 100;   // Size of the buffer capacity of packets, adjust as needed
    double wlanCapacity = 10.0; // WLAN speed in Mbps , adjust as needed

    cout << "File read: " << argv[1] << endl;
    cout << "Buffer Size: " << bufferSize << endl;
    cout << "WLAN Capacity: " << wlanCapacity << " Mbps" << endl;

    //Read input file passed through commandline argument
    auto streamPackets = parseInputFile(argv[1]);
    // Start of simulation
    runSimulation(streamPackets, bufferSize, wlanCapacity);

    return 0;
}
```

The main function (as shown above) sets the WLAN bandwidth and the buffer size for the simulation. These values are passed to the call to the runSimulation function, which uses it for calculating the transmission time and to check if the buffer is full.