Universidade de Aveiro

Modelação e Desempenho de Redes e Serviços

Mini-Project nº1

Pedro Carneiro (73775)

Inês Águia (73882)

October 29th 2024

Index

1. Introduction

Keeping in line with the requirements of the mini-project for the course of "Modeling and Performance of Networks and Services" we developed this report which includes key code snippets to demonstrante the reasoning behind the solutions and provides a summary of the conclusions drawn from the results of each exercise.

The report is divided into three chapters, one for each task and within each task into smaller sections for each exercise.

2. Task 1

2.1 Exercise: 1a

Required: Estimate by simulation the average packet delay and the average packet loss parameters when the bit error rate of the link is $b = 10^{-6}$ and for the arrival rate values λ = 1500, 1600, 1700, 1800 and 1900 pps. Plot the results in bar charts with the confidence intervals in error bars1. Justify the results and draw all relevant conclusions

The results on Table 2.1, show an increase both on the **Packet Loss (PL)** and also on the **Average Packet Delay (APD)**, directly related to the increasing value of **arrival rate (λ)**.

λ	**PL (%)**	**APD (ms)**
1500	0.63	1.78
1600	0.77	2.09
1700	1.01	2.46
1800	1.49	2.95
1900	2.20	3.48

Table 2.1: Table showing Simulation Results of 20 runs using **Sim2** with the following parameters : $\lambda = [1500, 1600, 1700, 1800, 1900]pps$, $C = 10Mbps$, $f = 10.000Bytes$, $b = 10^{-6}$, stop criterion of $P = 100.000$ and 90% confidence intervals

From the Figure 2.1 we can observe that as arrival rate increases, the system handles more packets per second, leading to higher queue occupancy and packets experiencing longer delays.

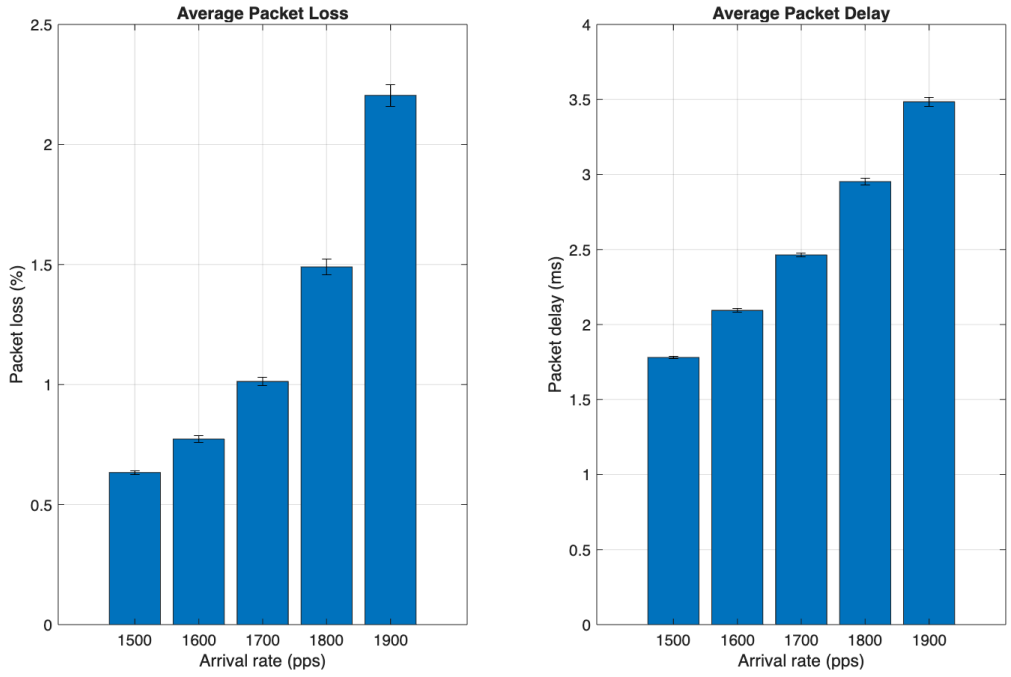

Figure 2.1: Bar chart with confident intervals in error bars from the simulation. Average Packet Loss on the Left and Average Packet Delay on the Right

$$\text{Link Capacity } (pps) = \frac{Capacity}{(\text{Average Byte Size} \times 8)} \tag{2.1}$$

Using the Equation 2.1 we calculate the link capacity ≈ 2016.1 pps. Since the max arrival rate ($\lambda = 1900$ pps) is lower than the link capacity, packet drops due to **queue overflow** are unlikely, meaning the packet loss observed in the Figure 2.1 is more likely due to **Bit Error Rate (BER)** rather than queue overload.

BER is a constant probability during the simulation, so as more packets are transmited the absolute number of packets experiencing errors increases, which are directly related to packet loss.

Conclusion: We find a relation between traffic load and network congestion, and the BER is directly related to the packet loss.

2.2 Exercise: 1b

Required: Repeat experiment $1.a$ *considering now a bit error rate* $b = 10^{-4}$*. Justify the differences between these results and the results of experiment* $1.a$ *and draw all relevant conclusions.*

The results on Table 2.2, show an almost stationary **Packet Loss (PL)** and an increase on the

Average Packet Delay (APD) directly related with **Arrival Rate** (λ).

λ	**PL (%)**	**APD (ms)**
1500	32.83	1.60
1600	32.88	1.91
1700	33.04	2.32
1800	33.21	2.81
1900	33.48	3.33

Table 2.2: Table showing Simulation Results of 20 runs using **Sim2** with the following parameters : $\lambda = [1500, 1600, 1700, 1800, 1900]pps$, $C = 10Mbps$, $f = 10.000Bytes$, $b = 10^{-4}$, stop criterion of $P = 100.000$ and 90% confidence intervals

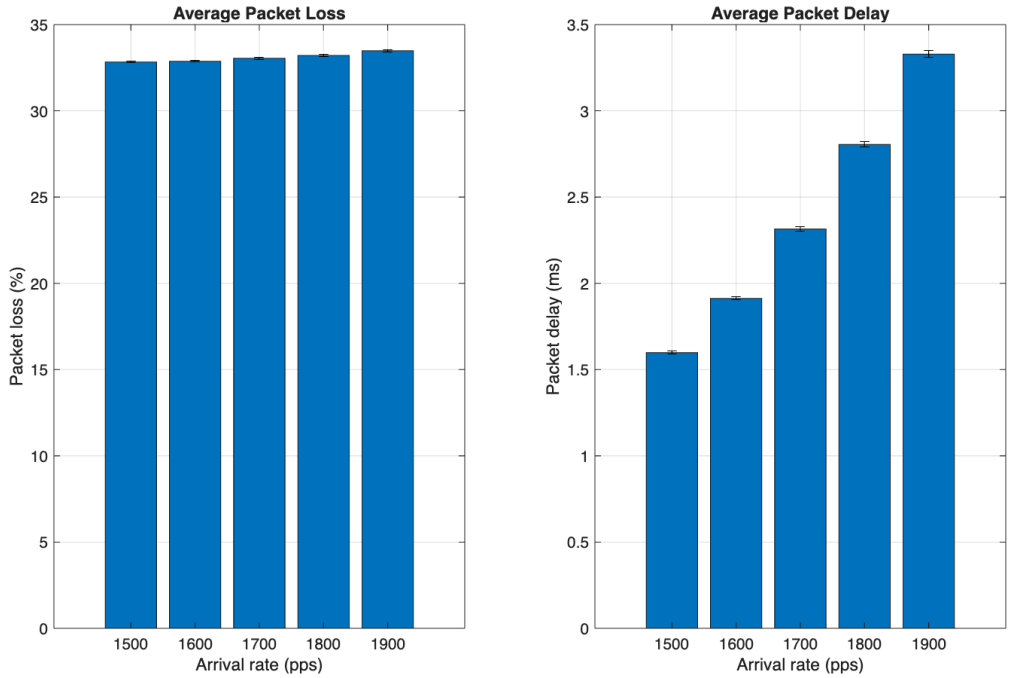

Figure 2.2: Bar chart with confident intervals in error bars from the simulation of the Average Packet Loss on the Left and Average Packet Delay on the Right

Analysing the Figure 2.2 we can see that Average Packet Delay values are similar to the ones on Figure 2.1. Thus we can infer that the package delay is barely influenced by the BER, or to be more precise, its influence is so little it can be can outlooked. With this we can now assume that the delay is linked to the transmission rate, particularly when it approaches the link's capacity - *As transmission increases, delays become more noticeable.*

The **packet loss increased significantly**, as displayed on Figure 2.2. As expected PL is now near 33%, compared with 3% from the previous simulation because BER was increased by a factor of 100x ($10^{-6} \times 100 = 10^{-4}$).

Conclusion: The **BER** influences directly the Packet Loss, due to the increasing amount of transmission errors which cause more packets to be dropped. On the other hand the BER is not responsible for Packet Delay, the small difference between values from APD can probably be atributted to quantity of packets being dropped.

2.3 Exercise: 1c

Required: Determine the theoretical average packet loss (in %) only due to the bit error rate for $b = 10^-6$ and $b = 10^-4$. Present and explain the MATLAB code developed for these calculations. Compare these values with the results obtained in 1.a and 1.b. What do you conclude?

2.3.1 Code

Listing 2.1: MATLAB code for calculate Average Packet Loss

```
1  b_values = [1e-6, 1e-4]; % Bit error rates
2
3  prob_left = (1 - (0.19 + 0.23 + 0.17)) / ((109 - 65 + 1) + (1517 - 111 + 1));
4  avg_packet_size = 0.19*64 + 0.23*110 + 0.17*1518 + sum((65:109)*(prob_left)) +
       sum((111:1517)*(prob_left));
5
6  % Display the average packet size
7  fprintf('\nAverage Packet Size: %.2f bytes\n\n', avg_packet_size);
8
9  % Initialize array to store packet loss for each bit error rate
10 packet_loss = zeros(size(b_values));
11
12 % Loop over each bit error rate
13 for i = 1:length(b_values)
14     b = b_values(i);  % Current bit error rate
15     % Calculate packet loss using the formula
16     packet_loss(i) = 1 - (1 - b)^(8 * avg_packet_size);  % 8 bits per byte
17     fprintf('Packet loss for b = %.1e: %.4f%%\n', b_values(i), packet_loss(i)
           * 100);
18 end
```

2.3.2 Results and Conclusions

Since we want to determine the theoretical average packet loss only due to the BER, the next logical step is to calculate the probability of the packet having at least one error. Given the packet is immediately dropped when one bit error is detected.

$$P_{noErrors} = \binom{n}{0} \times (1-p)^n = (1-p)^n, \text{ where n} = (\text{bytes} \times 8) \tag{2.2}$$

First we use the Equation 2.2 in order to calculate the probability of having "no errors". Having this we are able to calculate the probability of the packet having at least one error, as showned in Equation 2.3, using the complement rule $P(A') = 1 - P(A)$.

$$P_{atLeastOneError} = 1 - (1-p)^n, \text{ where n} = (\text{bytes} \times 8) \tag{2.3}$$

Now we need to calculate the value of n, and in this case we need the Average Packet Size from the simulation. For this we use the code previously made on the $4.a$ (from the pratical guide) to calculate it (≈ 620 bytes), Listing 2.2. The code calculates the weighted average of packet sizes in the ranges $(62:109)$ and $(111:1517)$.

Listing 2.2: MATLAB code for calculate Average Packet SIZE

```
1  prob_left = (1 - (0.19 + 0.23 + 0.17)) / ((109 - 65 + 1) + (1517 - 111 + 1));
2  avg_packet_size = 0.19*64 + 0.23*110 + 0.17*1518 + sum((65:109)*(prob_left)) +
       sum((111:1517)*(prob_left));
```

Once we have the Average Packet Size we can use the Equation 2.3 to get the theoretical value of the Average Packet Loss, because we already have the values of $p = BER$ (10^{-6} and 10^{-4}). As displayed on Listing 2.1 we iterate over each value of p and calculate the **Average Packet Loss** given a BER. We got the the following values:

1. $(p = 10^{-4}) \approx 0.4948\%$
2. $(p = 10^{-6}) \approx 39.1064\%$

Conclusion: The results are in the same order of magnitude of the previously simulated. This results reinforce that the values from the simulation are influenced by the BER only, as the system can process all the packets and we do not see a Queue Overload. In conclusion the **Packet Loss is mainly due to BER**.

3. Task 2

3.1 Exercise: 2a

Required: Using Sim3A, estimate all performance parameters when λ = 1500 pps, C = 10 Mbps, f = 1.000.000 Bytes, b = 10-5 and n = 10, 20, 30 and 40 VoIP flows, based on 20 runs of the simulator (with a stopping criterion of P = 100.000 on each run) and with 90% confidence intervals.

3.1.1 Code

> **Note**
>
> The complete code in the Appendix A

Listing 3.1: MATLAB code alteration on *Sim3* introducing a **bit error rate (BER)**

```
case DEPARTURE_DATA
    numBits = PacketSize * 8;
    if rand() < 1 - (1 - b)^numBits
        %(... Packet transmitted with ERRORS (BER) --> DROP)
    else
        %(... Package SUCCESSFULLY TRANSMITTED --> COUNT)
    end
    %(...)
case DEPARTURE_VOIP
    numBits = PacketSize * 8;
    if rand() < 1 - (1 - b)^numBits
        %(... Packet transmitted with ERRORS (BER) --> DROP)
    else
        %(... Package SUCCESSFULLY TRANSMITTED --> COUNT)
    end
    %(...)
```

3.1.2 Results and Conclusions

Started by implementing a random number generator to simulate random probabilities, P_{rand} and after a BER error check. The **error probability**, P_{error}, was calculated using the Equation 2.3. This expression represents the likelihood of at least one bit in the packet being erroneous.

- $P_{rand} < P_{error}$: packet contains errors so we must drop it
- $P_{rand} \geq P_{error}$: packet does not contain errors and is sucessfully transmitted

Note: Given our Simulator structure the implementation has to be done both for $Arrival_{DATA}$ and $Arrival_{VOIP}$.

VoIP Flows	PL_{DATA} (%)	PL_{VoIP} (%)	APD_{DATA} **(ms)**	APD_{VoIP} **(ms)**
10	4.72 ± 0.02	0.96 ± 0.03	2.13 ± 0.02	1.73 ± 0.02
20	4.73 ± 0.03	0.96 ± 0.01	2.67 ± 0.03	2.27 ± 0.02
30	4.72 ± 0.04	0.96 ± 0.02	3.73 ± 0.09	3.32 ± 0.08
40	4.72 ± 0.04	0.95 ± 0.01	5.75 ± 0.23	5.34 ± 0.22
VoIP Flows	MPD_{DATA} (%)	MPD_{VoIP} (%)	**Throughput (Mbps)**	
10	4.72 ± 0.02	0.96 ± 0.03	7.24 ± 0.01	
20	4.73 ± 0.03	0.96 ± 0.01	7.74 ± 0.02	
30	4.72 ± 0.04	0.96 ± 0.02	8.24 ± 0.02	
40	4.72 ± 0.04	0.95 ± 0.01	8.67 ± 0.02	

Table 3.1: Table showing Simulation Results of 20 runs using **Sim3A** with the following parameters : $\lambda = 1500pps$, $C = 10Mbps$, $f = 1.000.000Bytes$, $b = 10^{-5}$, $n = [10, 20, 30, 40]$ VoIP flows, stop criterion of $P = 100.000$ and 90% confidence intervals

3.2 Exercise: 2b

Required: Present the simulation results of 2.a concerning the packet loss of each service (data and VoIP) in bar charts with the confidence intervals in error bars. Justify the results and draw all relevant conclusions.

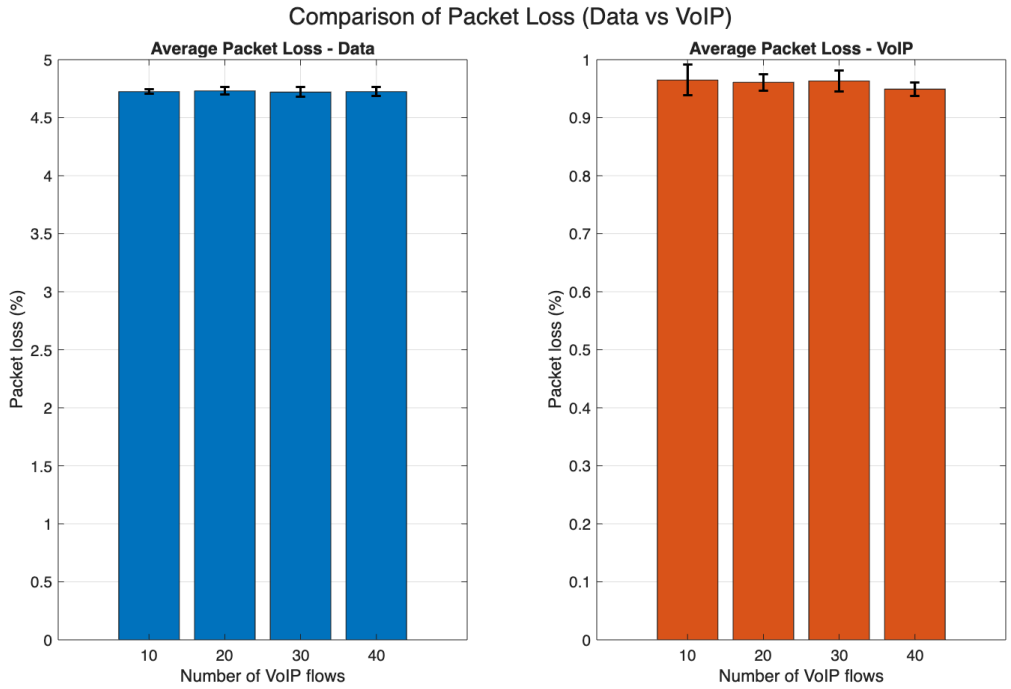

Figure 3.1: Comparison of Packet Loss between Data Packets(left bar chart) and VoIP Packets(right bar chart) across numbers of VoIP flows [10 - 40]

	Arrival Distribution (*ms*)	**Packet Sizes** (*bytes*)
Data Packets	Exponential distribution $\frac{1}{\lambda}$	Non-uniform distribution [64 - 1518]
VoIP Packets	Uniform distribution [16 - 24]	Uniform distribution [110 - 130]

Table 3.2: Table showing Packet(Data and VoIP) Characteristics

Table 3.2 is a brief summary of traffic characteristics.

Data packets arrive according to an exponential distribution with an average arrival time of $\frac{1}{\lambda}$, this very distribution tends to increase the likelihood of congestion. This can quickly fill the queue, leading to a higher packet loss for Data packets. Besides, data packets have a wide range of sizes given its non-uniform distribution, therefore, it is more likely to experience drops, particularly when the queue fills up with larger packets.

As for VoIP flows, the generate packets are uniformly distributed between 16 and 24 milliseconds, reducing the chances of queue congestion and sized wise, are between 110 and 130 bytes. Since each VoIP flow generates fewer packets (due to larger intervals between arrivals), there is generally less competition for queue space compared to the Data flow. This reduces the likelihood of VoIP packet loss, as they face less congestion pressure.

Conclusion: The increasing number of VoIP flows has little to none influence on Packet Loss. The main responsibles for this are *Arrival and Packet Size* distributions, since they influence the

queue congestion making Data Packets have a higher percentage of loss packets when comparing to VoIP packets.

3.3 Exercise: 2c

Required: Present the simulation results of 2.a concerning the average packet delay of each service in bar charts with the confidence intervals in error bars. Justify the results and draw all relevant conclusions.

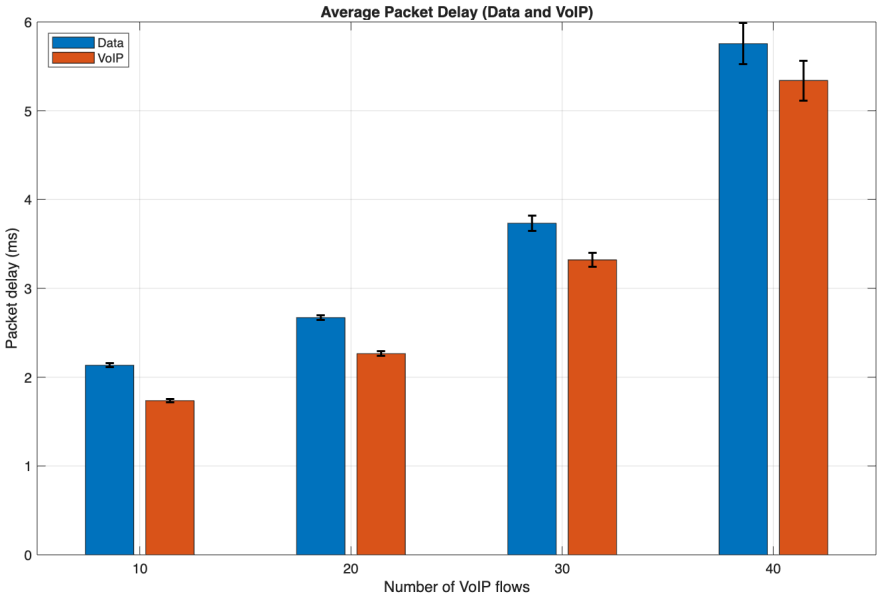

Figure 3.2: Bar chart showing the Average Packet delay of Data Packets (blue) and VoIP Packets(red) across number of VoIP flows [10 - 40]

Since Data Packets have larger, variable sizes and exponential arrival rate, they are more proun to queueing delays when network load is high. The exponential arrival rate of Data packets amplifies this delay when the system load is high, especially when more VoIP flows are competing for bandwidth.

VoIP Packets benefit from their small and more consistent size, as well as their uniformly distributted arrival rate lowering the queueing delay when compared to Data Packets, as presented in Figure 3.2.

Conclusion: The increasing number of VoIP flows influeces the Average Packet Delay as the traffic load makes packets compete for queue space. The differences between Data and VoIP packets are the results of *Arrival and Packet Size* distributions.

3.4 Exercise: 2d

Required: Present the simulation results of 2.a concerning the maximum packet delay of each service in bar charts with the confidence intervals in error bars. Justify the results and draw all relevant conclusions.

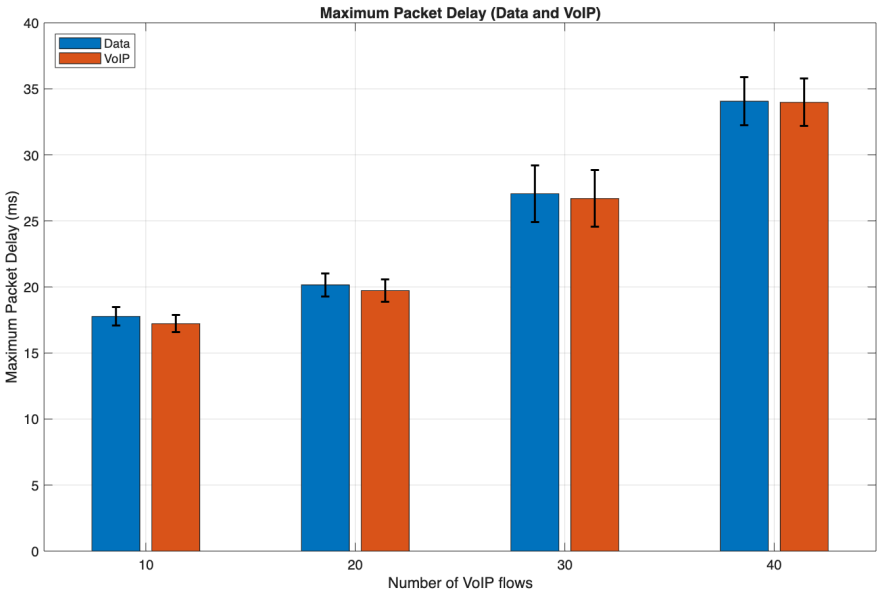

Figure 3.3: Bar chart showing the Maximum Packet delay of Data Packets (blue) and VoIP Packets(red) across number of VoIP flows [10 - 40]

As showned on Figure 3.3 results indicate that packets are more susceptible to increasing Maximum Delays as the number of VoIP flows rises, but there is no pronounced difference between Data and VoIP packets.

Conclusion: Number of VoIP flows is directly related to an increase on Maximum Packet Delay, because there is more traffic competing for queue space. The imperceptible difference between Data and VoIP tells us that the *Arrival and Packet Size* distributions has no influence on Maximum Packet Delay and that traffic is being treated the same way.

3.5 Exercise: 2e

Required: Present the simulation results of 2.a concerning the total throughput in bar charts with the confidence intervals in error bars. Justify the results and draw all relevant conclusions.

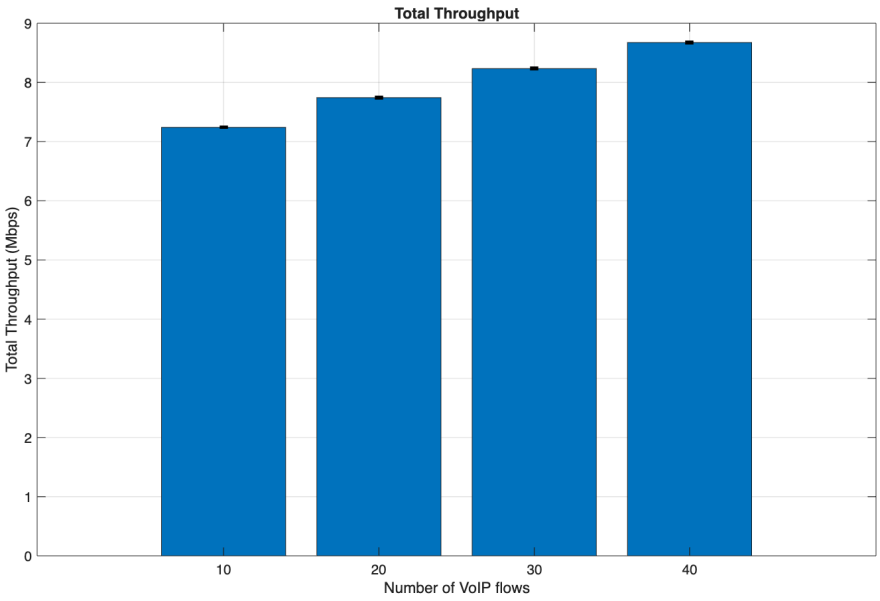

Figure 3.4: Bar chart showing the Throughput across number of VoIP flows [10 - 40]

The results on Figure 3.4 indicate that the network's throughput rises with the number of VoIP flows, as expected.

Having in mind that the $C = 10Mbps$, this results show that the network effectively uses, its capacity without experiencing a significant throughput degradation, almost 90% of traffic passes through the network.

Conclusion: The network handles up to 40 VoIP flows comfortably, further increases may require capacity upgrades to avoid congestion.

3.6 Exercise: 2f

Required: Determine the theoretical value of the total throughput for all cases simulated in experiment 2.a. Present and explain the MATLAB code developed for these calculations. Compare these values with the results obtained in 2.e. What do you conclude?

3.6.1 Code

Listing 3.2: MATLAB code for calculate theoretical Throughput for the N_{VoIP} Flows

```
% TT for DATA PACKETS
prob_left = (1 - (0.19 + 0.23 + 0.17)) / ((109 - 65 + 1) + (1517 - 111 + 1));
avgSIZE_data = 0.19*64 + 0.23*110 + 0.17*1518 + sum((65:109)*(prob_left)) + sum
    ((111:1517)*(prob_left));
rate_data = 1500;
```

```
data_TT = rate_data * avgSIZE_data * 8 / 10^6;

% TT for VoIP PACKETS
avgSIZE_voip = ( 110 + 130 ) / 2;
int_time_voip = (16 + 24) / 2;
rate_single_voip = 10^3 / int_time_voip;

% Define the number of VoIP flows to analyze
num_voip_flows = [10, 20, 30, 40];

% Loop through each VoIP flow scenario
for i = 1:length(num_voip_flows)
    n_voip = num_voip_flows(i);
    total_voip_rate = n_voip * rate_single_voip;
    voip_TT = total_voip_rate * avgSIZE_voip * 8 / 10^6;
    final_TT = data_TT + voip_TT; % Total throughput (Data + VoIP)

    % Display results for each VoIP flow scenario
    fprintf('For %d VoIP flows:\n', n_voip);
    fprintf('  Theoretical Total Throughput: %.2f Mbps\n\n', final_TT);
end
```

3.6.2 Results and Conclusions

Throughput is defined as the rate at which data is successfully transmitted over a network link, so we can calculate it by using the Equation 3.1 .

$$\text{Throughput (TT)} = \frac{\text{Total Transmitted Bytes} \times 8}{\text{Total Simulation Time}} \times 10^{-6} \tag{3.1}$$

Without access to the simulation time or transmitted Bytes, we relied on the transmission rates for Data and VoIP packets. These rates represent the number of packets transmitted per second (*pps*). By calculating the average packet size, we were able to convert this to bytes per second (*bps*), which enabled us to calculate the Throughput as required, using Equation 3.2.

$$\text{Throughput (bps)} = \text{Packet Arrival Rate (pps)} \times \text{Average Packet Size (bits)} \tag{3.2}$$

Typically, Link Capacity is measured in *Mbps*, so for easy comparison, we should express the Throughput (TT) in the same unit. To do this, we used Equation 3.3.

$$\text{Throughput (Mbps)} = \frac{\text{Packet Arrival Rate(pps)} \times AveragePacketSize(bits)}{10^6} \tag{3.3}$$

These formulas generally treat traffic as a single entity; however, in this scenario, we have distinct types of traffic, each with unique specifications as outlined in Table 3.2. Consequently, the Throughput (TT) must be calculated as shown in Equation 3.4.

$$TT_{Total} = TT_{Data} + TT_{VoIP} \tag{3.4}$$

So we need to calculate separately both Throughtputs (VoIP and Data). Let us start for Data using the Equation 3.5.

$$TT_{DATA} = \lambda_{DATA} \times (AvgPacketSize_{DATA} \times 8) \times 10^{-6} \tag{3.5}$$

We calculate the Average Packet Size for Data packets by considering their distribution as we did in 1*c.* and using the code in Listing 2.2. The rate of arrival, given in the problem statement, is $\lambda = 1500\,pps$.

To calculate the Throughput for VoIP packets, we apply a similar approach as with Data traffic, considering both the average arrival rates for VoIP packets and their average packet size distribution, as detailed in Table 3.2. The relevant formulas are provided in Equation 3.6.

$$AverageSize = \left(\frac{110+130}{2}\right) \text{(bits)} \ , \ ArrivalRate = \frac{10^3}{\left(\frac{16+24}{2}\right)} \text{ (s)} \tag{3.6}$$

However, this calculation is for a single VoIP flow. To obtain the overall Throughput for all VoIP flows, we need to multiply this value by the total number of VoIP flows, as indicated in Listing 3.2.

After this, we use the formula Equation 3.7 to calculate the Throughput for VoIP packets, ensuring that we convert the units to their appropriate values.

$$TT_{VoIP} = N_{VoIP}Flows \times \left(\lambda_{VoIP} \times (AvgPacketSize_{VoIP} \times 8) \times 10^{-6}\right) \tag{3.7}$$

Once we have calculated the Throughput for both Data and VoIP packets, we simply need to sum these values to derive the final Throughput, as shown in Equation 3.4.

Nº VoIP Flows	**10**	**20**	**30**	**40**
Theoretical	7.92	8.40	8.88	9.36
Simulation	7.24	7.74	8.24	8.67

Table 3.3: Table showing Throughtput ($Mbps$) values (Theoretical and Simulated) for different VoIP Flows

Conclusion: The theoretical throughput calculation assumes that every generated packet is successfully transmitted, without considering any potential packet loss. In real-world simulations or practical scenarios, as indicated by our simulation results, packet loss can occur, particularly under higher load conditions—such as when additional VoIP flows are introduced.

Consequently, the theoretical values Table 3.3 tend to be slightly higher than those obtained from the simulation, as the theoretical calculations do not account for packets that may be dropped during transmission.

4. Task 3

4.1 Exercise: 3a

Required: Use Sim3 to estimate the average packet delay and average packet loss of each service (data and VoIP). Recall that in Sim3, both services are statistically multiplexed in a single FIFO queue. Present each of the four performance parameters in a bar chart with the confidence intervals in error bars. Justify the differences in the performance values obtained for each service and draw all relevant conclusions

VoIP Flows	PL_{DATA} (%)	PL_{VoIP} (%)	APD_{DATA} **(ms)**	APD_{VoIP} **(ms)**
10	0.25 ± 0.01	0.03 ± 0.00	2.04 ± 0.01	1.63 ± 0.01
20	0.44 ± 0.02	0.05 ± 0.00	2.33 ± 0.02	1.93 ± 0.02
30	0.76 ± 0.02	0.09 ± 0.01	2.70 ± 0.02	2.32 ± 0.01
40	1.39 ± 0.07	0.17 ± 0.01	3.24 ± 0.3	2.88 ± 0.03
VoIP Flows	MPD_{DATA} (%)	MPD_{VoIP} (%)	**Throughput (Mbps)**	
10	9.00 ± 0.03	8.75 ± 0.07	7.24 ± 0.01	
20	9.05 ± 0.03	8.84 ± 0.05	7.74 ± 0.02	
30	9.07 ± 0.02	8.93 ± 0.03	8.24 ± 0.02	
40	9.08 ± 0.02	8.96 ± 0.04	8.67 ± 0.02	

Table 4.1: Table showing Simulation Results of 20 runs using **Sim3** with the following parameters : $\lambda = 1500pps$, $C = 10Mbps$, $f = 10.000Bytes$, $b = 10^{-5}$, $n = [10, 20, 30, 40]$ VoIP flows, stop criterion of $P = 100.000$ and 90% confidence intervals

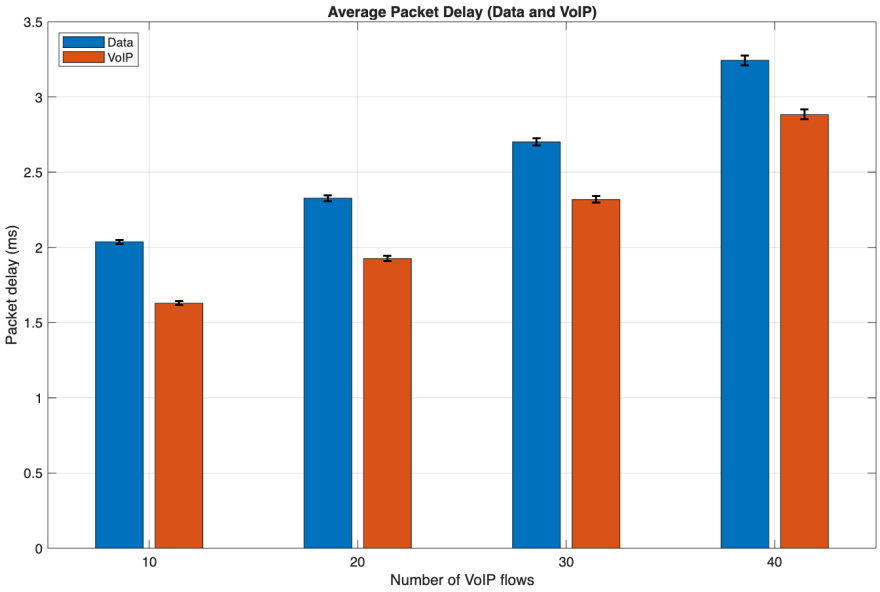

Figure 4.1: Bar chart showing the Average Packet delay of Data Packets (blue) and VoIP Packets(red) across number of VoIP flows [10 - 40], using Sim3

The first chart Figure 4.1 indicates that as the number of VoIP flows increases, both data and VoIP packet delays increase. However, data packets experience higher delays compared to VoIP packets for every VoIP flow level.

In a FIFO queue, packets are served in the order of arrival, regardless of their size or type. Since data packets have variable sizes, larger packets increase the delay, particularly for subsequent packets. VoIP packets, being more uniform in size and with arrival times in the range of 16-24 ms, tend to experience less delay.

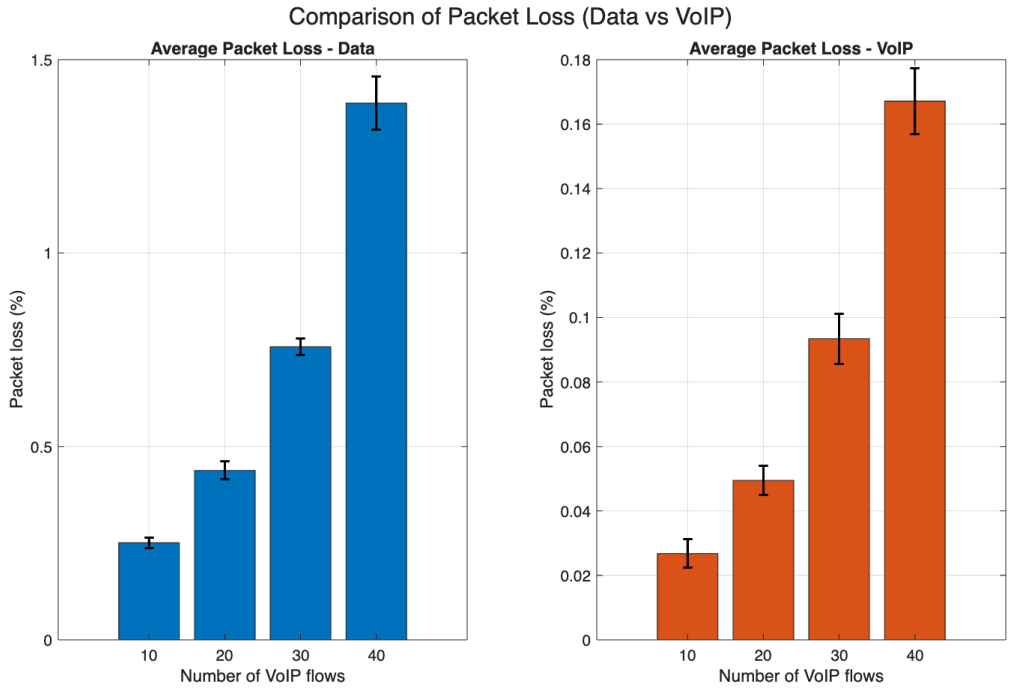

Figure 4.2: Comparison of Packet Loss between Data Packets(left bar chart) and VoIP Packets(right bar chart) across numbers of VoIP flows [10 - 40], usign $Sim3$

The packet loss results displayed in Figure 4.2 shows an increase in packet loss as the number of VoIP flows rises. However, data packets have significantly higher packet loss rates than VoIP packets across all levels of VoIP flows.

Since both data and VoIP packets are statistically multiplexed in a single FIFO queue, data packets have to compete with VoIP packets for queue space. With the increase in VoIP flows, the queue fills more quickly, leading to higher chances of packet loss, especially for the larger data packets.

Conclusion: Data traffic experiences higher packet loss due to its larger and more variable packet sizes and arrival patterns. VoIP traffic maintains relatively low packet loss due to its smaller, consistent packet sizes and more uniform arrival intervals.

4.2 Exercise: 3b

Required: Use Sim4 to estimate the same performance parameters as in 3.a. Recall that in Sim4, VoIP service has higher priority than data service. Present each of the four performance parameters in a bar chart with the confidence intervals in error bars. Justify the differences in the performance values obtained for each service, and the differences between these results and the results of experiment 3.a. Draw all relevant conclusions

VoIP Flows	PL_{DATA} (%)	PL_{VoIP} (%)	APD_{DATA} (ms)	APD_{VoIP} (ms)
10	0.24 ± 0.01	0.03 ± 0.00	2.10 ± 0.01	0.46 ± 0.00
20	0.43 ± 0.02	0.05 ± 0.00	2.50 ± 0.02	0.48 ± 0.00
30	0.77 ± 0.03	0.09 ± 0.01	3.09 ± 0.02	0.50 ± 0.00
40	1.42 ± 0.05	0.17 ± 0.01	3.92 ± 0.04	0.52 ± 0.0

VoIP Flows	MPD_{DATA} (%)	MPD_{VoIP} (%)	Throughput (Mbps)
10	9.48 ± 0.04	1.40 ± 0.01	7.24 ± 0.01
20	10.01 ± 0.04	1.43 ± 0.01	7.74 ± 0.02
30	10.68 ± 0.07	1.48 ± 0.02	8.24 ± 0.02
40	11.33 ± 0.05	1.52 ± 0.02	8.67 ± 0.02

Table 4.2: Table showing Simulation Results of 20 runs using **Sim4** with the following parameters : $\lambda = 1500pps$, $C = 10Mbps$, $f = 10.000Bytes$, $n = [10, 20, 30, 40]$ VoIP flows, stop criterion of $P = 100.000$ and 90% confidence intervals

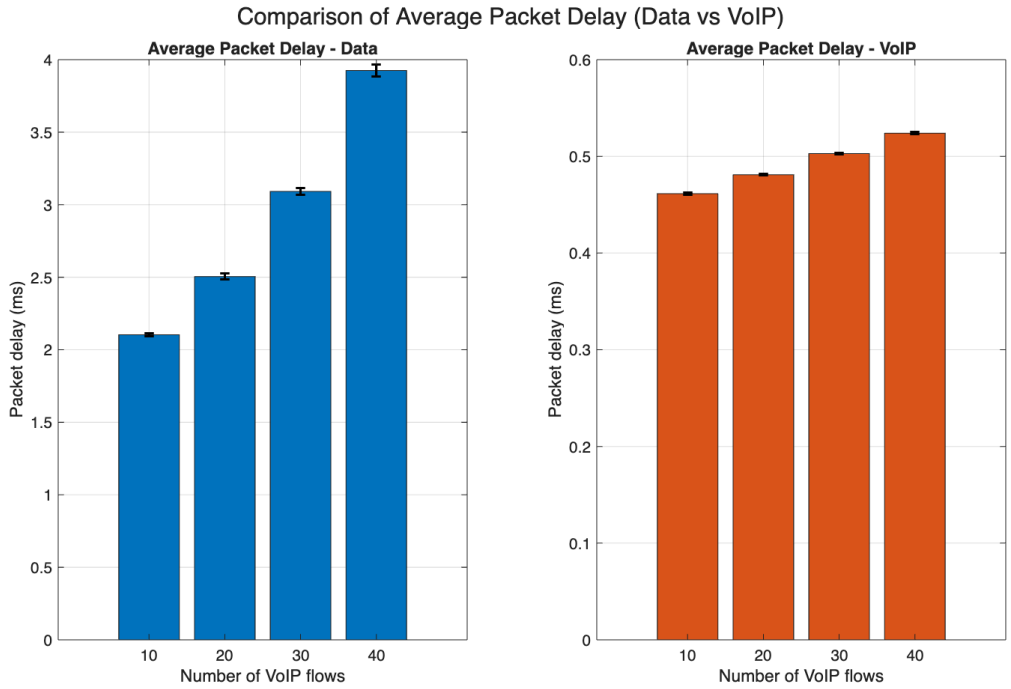

Figure 4.3: Comparison of Packet Delay between Data Packets(left bar chart) and VoIP Packets(right bar chart) across numbers of VoIP flows [10 - 40], using $Sim4$

The priority discard algorithm in $Sim4$ allows VoIP packets to bypass Data packets in the queue, resulting in lower delays, as displayed on Figure 4.3, for VoIP at the cost of higher delays for Data. This demonstrates the benefit of prioritization for time-sensitive traffic (like VoIP), but it also shows

the impact on Data packets, which experience longer wait times.

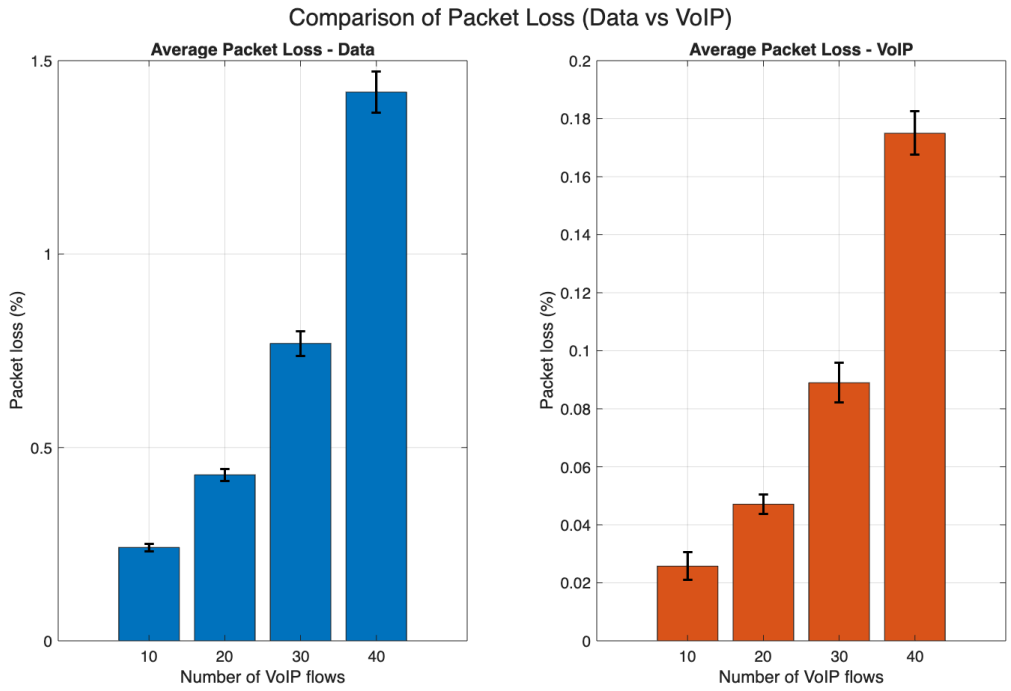

Figure 4.4: Comparison of Packet Loss between Data Packets(left bar chart) and VoIP Packets(right bar chart) across numbers of VoIP flows [10 - 40], using *Sim*4

Because of the priority algorithm, VoIP packets are less likely to be dropped, Figure 4.4, while Data packets are more prone to being discarded when the queue becomes full. This results in better reliability for VoIP at the expense of Data packet loss, especially under high load conditions.

Conclusion: The prioritization of VoIP packets in Sim4 demonstrates a significant improvement in delay and reliability for VoIP services, which is beneficial for real-time applications like voice calls. However, this comes at the expense of Data traffic, highlighting a common trade-off in network QoS management.

4.3 Exercise: 3c

Required: Develop a new version of Sim4, named Sim4A, to estimate the same performance parameters as Sim4 changing the queue packet discard algorithm as follows: arriving VoIP packets are always accepted in the queue (if there is enough space) but arriving data packets are accepted in the queue only if the total queue occupation does not become higher than p (in %) of the queue size2 (parameter p should be a new input parameter of Sim4A). Present the developed MATLAB function of Sim4A highlighting and justifying the introduced changes

4.3.1 Code

> **Note**
>
> The complete code in the Appendix B

Listing 4.1: MATLAB code for the Discard algorithm: Accepting Data packets in the queue only if the total queue occupation does not become higher than $p(\%)$ of the queue size and VoIP packets are always accepted in the queue (if there is enough space)

```
case ARRIVAL_DATA
    % (...)
    if STATE == 0
        % (...)
    else
        if QUEUEOCCUPATION + PacketSize <= f * (p/100)
           % (... ADD PACKETS)
        else
            % (... DROP PACKETS)
        end
    end
```

4.3.2 Results and Conclusions

Introducing a New Discard Algorithm: arriving VoIP packets are always accepted in the queue (if there is enough space) but arriving Data packets are accepted in the queue only if the total queue occupation does not become higher than $p(\%)$ of the queue size.

- Case $ARRIVAL_{VoIP}$: we did not change anything since for the VoIP packets nothing changed, only needing to make sure there is enough space like we already did in *Sim4*, as can be seen in Listing 4.2

Listing 4.2: MATLAB code for the Discard algorithm for Case $ARRIVAL_{VOIP}$

```
case ARRIVAL_VOIP
    %(...)
        if QUEUEOCCUPATION + PacketSize <= f
           % (... ADD PACKETS)
        else
            % (... DROP PACKETS)
        end
    end
```

- Case $ARRIVAL_{DATA}$: we altered the previous verification that is equal to Listing 4.2 to only add the packet to the queue if the sum of the queue ocuppation with the packet size

was less or equal than the percentage chosen previously. As we can see on Listing B.1, the verification follows the mathematical expression $u \leq v * (w/100)$, where u represents the sized combined from queue occupation and the packet size to add, v represents the maximum queue size and x represents the percentage of queue which Data packets can not overcome.

4.4 Exercise: 3d

Required: Use simulator Sim4A to estimate the same performance parameters as in 3.a and 3.b for p = 90%. Justify the differences in the performance values obtained for each service, and the differences between these results and the results of experiment 3.b. Draw all relevant conclusions.

VoIP Flows	PL_{DATA} (%)	PL_{VoIP} (%)	APD_{DATA} **(ms)**	APD_{VoIP} **(ms)**
10	0.37 ± 0.02	0.00 ± 0.00	2.05 ± 0.02	0.46 ± 0.00
20	0.61 ± 0.02	0.00 ± 0.00	2.41 ± 0.01	0.48 ± 0.00
30	1.06 ± 0.03	0.00 ± 0.00	2.95 ± 0.02	0.50 ± 0.00
40	1.78 ± 0.02	0.00 ± 0.00	3.62 ± 0.02	0.52 ± 0.00
VoIP Flows	MPD_{DATA} (%)	MPD_{VoIP} (%)	**Throughput (Mbps)**	
10	8.76 ± 0.05	1.41 ± 0.02	7.24 ± 0.01	
20	9.26 ± 0.05	1.43 ± 0.01	7.74 ± 0.02	
30	9.82 ± 0.05	1.47 ± 0.01	8.24 ± 0.02	
40	10.51 ± 0.05	1.51 ± 0.02	8.67 ± 0.02	

Table 4.3: Table showing Simulation Results of 20 runs using $Sim4A$ with the following parameters : $\lambda = 1500pps$, $C = 10Mbps$, $f = 10.000Bytes$, $n = [10, 20, 30, 40]$ VoIP flows, stop criterion of $P = 100.000$, 90% confidence intervals and $p = 90\%$

As explained on the previous exercices, Sim4A implements a new queue packet discard algorithm where arriving Data packets are only accepted in the queue if the total queue occupation does not become higher than a given value. In contrast, VoIP packets are always accepted as long as there is available space in the queue. For this scenario, the *p(total queue occupation accepted for data packets) = 90%*, what only accentuates the previous behaviour of giving preference to VoIP packets.

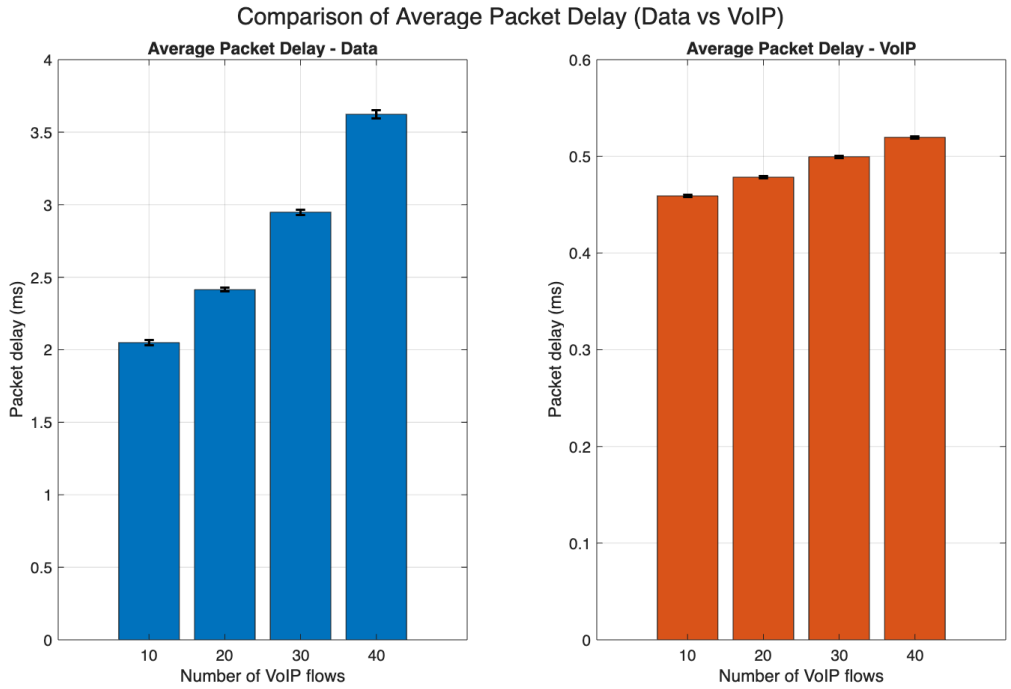

Figure 4.5: Comparison of Packet Delay between Data Packets(left bar chart) and VoIP Packets(right bar chart) across numbers of VoIP flows [10 - 40], using $Sim4A$ with $p = 90\%$

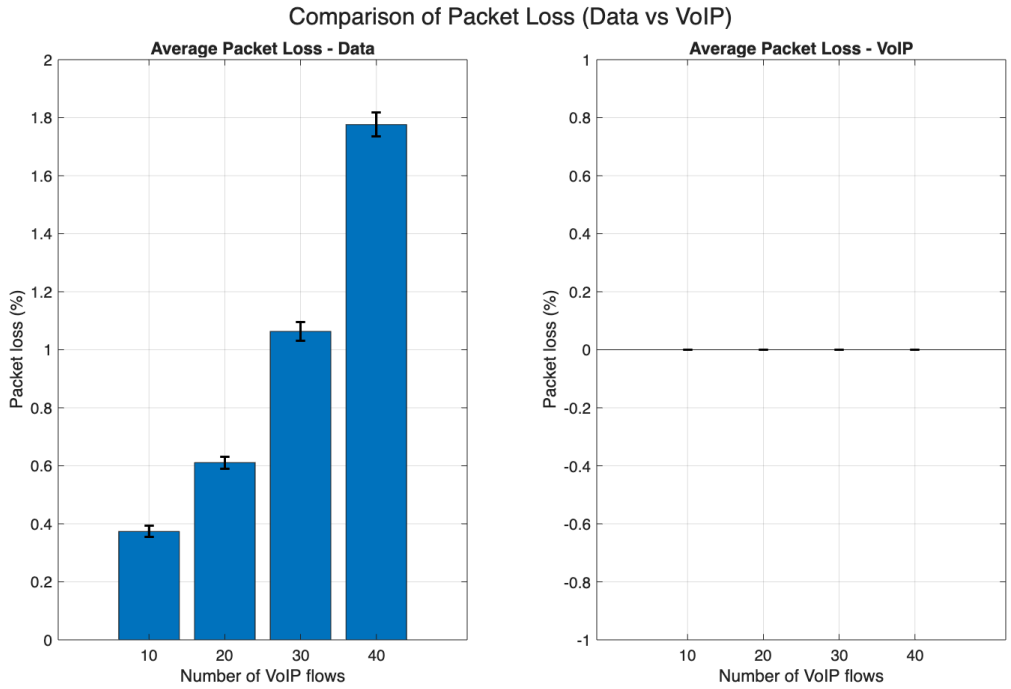

Figure 4.6: Comparison of Packet Loss between Data Packets(left bar chart) and VoIP Packets(right bar chart) across numbers of VoIP flows [10 - 40] , using $Sim4A$ with $p = 90\%$

With this approach, we anticipated that the primary impact would be on the average packet loss rate, while average packet delay should remain unaffected, Figure 4.5, given that all other conditions influencing delay have been kept constant.

We would expect an increase in packet loss for Data packets, as there is now less available space allocated to them in the queue. This, combined with the existing priority algorithm, drives the average packet loss for VoIP packets closer to zero or even eliminates it entirely, ensuring nearly uninterrupted transmission quality for VoIP traffic Figure 4.6.

4.5 Exercise: 3e

Required: Repeat experiment 3.d considering now p = 60%. Justify the differences in the performance values obtained for each service, and the differences between these results and the results of experiments 3.b and 3.d. Draw all relevant conclusions.

VoIP Flows	PL_{DATA} (%)	PL_{VoIP} (%)	APD_{DATA} (ms)	APD_{VoIP} (ms)
10	1.26 ± 0.02	0.00 ± 0.00	1.74 ± 0.01	0.45 ± 0.00
20	1.74 ± 0.04	0.00 ± 0.00	1.97 ± 0.01	0.47 ± 0.00
30	2.38 ± 0.05	0.00 ± 0.00	2.24 ± 0.01	0.49 ± 0.00
40	3.27 ± 0.05	0.00 ± 0.00	2.60 ± 0.01	0.50 ± 0.00

VoIP Flows	MPD_{DATA} (%)	MPD_{VoIP} (%)	**Throughput (Mbps)**
10	6.34 ± 0.04	1.40 ± 0.01	7.24 ± 0.01
20	6.72 ± 0.05	1.43 ± 0.01	7.74 ± 0.02
30	7.18 ± 0.06	1.48 ± 0.02	8.24 ± 0.02
40	7.65 ± 0.09	1.52 ± 0.02	8.67 ± 0.02

Table 4.4: Table showing Simulation Results of 20 runs using $Sim4A$ with the following parameters : $\lambda = 1500pps$, $C = 10Mbps$, $f = 10.000Bytes$, $n = [10, 20, 30, 40]$ VoIP flows, stop criterion of $P = 100.000$, 90% confidence intervals and $p = 60\%$

For this exercise, we began by setting the probability of total queue occupation allocated to Data packets to 60%. This implies that data packets are limited to occupying only 60% of the queue's capacity, which we expect will lead to higher packet loss rates for Data packets.

Analyzing Figure 4.8, we observe, as expected, that packet loss for Data packets increases compared to when the maximum queue occupation was set to 90%. The packet loss for VoIP packets remains consistently at zero, which is also anticipated given that VoIP packets are prioritized and do not face the same queue limitations as data packets. Since VoIP packet loss was zero in previous scenarios, it's reasonable that the same result is observed here.

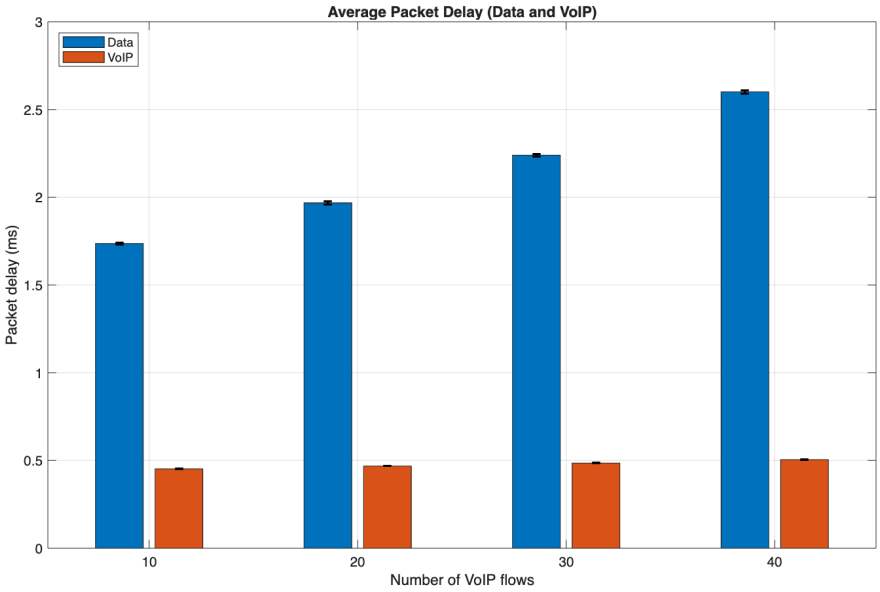

Figure 4.7: Bar chart showing the Average Packet delay of Data Packets (blue) and VoIP Packets(red) across number of VoIP flows [10 - 40], using $Sim4A$ with $p = 60\%$

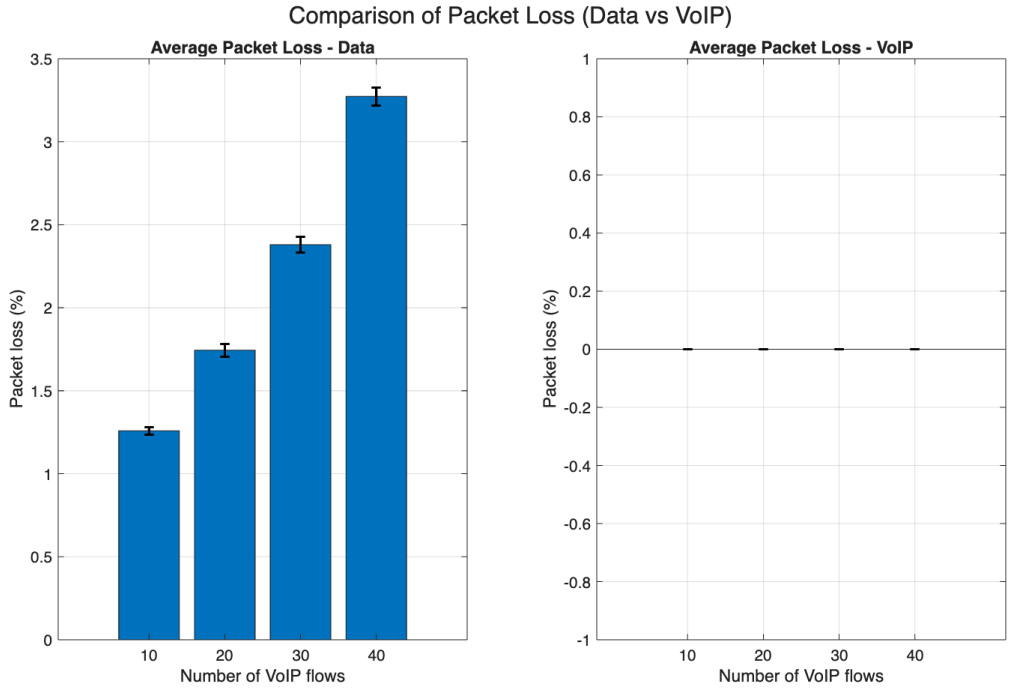

Figure 4.8: Comparison of Packet Loss between Data Packets(left bar chart) and VoIP Packets(right bar chart) across numbers of VoIP flows [10 - 40] , using $Sim4A$ with $p = 60\%$

Examining Figure 4.7, we notice a slight reduction in the average packet delay for Data packets. This decrease can be attributed to the increased packet dropping rate: fewer Data packets are

queued, leading to a lighter load in the queue and consequently shorter wait times for those that are processed. For VoIP packets, the average packet delay remains almost constant, even as the number of VoIP flows increases. This stability suggests that the delay for VoIP packets is primarily determined by processing and transmission times, rather than queuing delays, reinforcing the notion that VoIP traffic is minimally impacted by queue restrictions placed on Data packets.

5. Information

The members' contributions were equal and the link for the project's repository can be found here: `https://github.com/PedroMiguelTorresCarneiro/MDRS`. Our auto-evaluation is 16. An invitation has been sent to the professor's email: `asousa@ua.pt`, to be part of the repository.

A. Appendix

Listing A.1: Sim3A

```
function [PLdata, PLVoIP, APDdata, APDVoIP, MPDdata, MPDVoIP, TT] = Sim3A(lambda,
    C, f, P, n, b)
    ARRIVAL_DATA = 0;
    ARRIVAL_VOIP = 2;
    DEPARTURE_DATA = 1;
    DEPARTURE_VOIP = 3;

    STATE = 0;
    QUEUEOCCUPATION = 0;
    QUEUE = [];

    TOTALPACKETS_DATA = 0;
    LOSTPACKETS_DATA = 0;
    TRANSPACKETS_DATA = 0;
    TRANSBYTES_DATA = 0;
    DELAYS_DATA = 0;
    MAXDELAY_DATA = 0;

    TOTALPACKETS_VOIP = 0;
    LOSTPACKETS_VOIP = 0;
    TRANSPACKETS_VOIP = 0;
    TRANSBYTES_VOIP = 0;
    DELAYS_VOIP = 0;
    MAXDELAY_VOIP = 0;

    Clock = 0;
    tmp = Clock + exprnd(1/lambda);
    EventList = [ARRIVAL_DATA, tmp, GeneratePacketSizeData(), tmp];

    for i = 1:n
        tmpVoIP = Clock + 0.02*rand();
        EventList = [EventList; ARRIVAL_VOIP, tmpVoIP, GeneratePacketSizeVoIP(),
            tmpVoIP];
```

```
    end

    while TRANSPACKETS_DATA + TRANSPACKETS_VOIP < P
        EventList = sortrows(EventList,2);
        Event = EventList(1,1);
        Clock = EventList(1,2);
        PacketSize = EventList(1,3);
        ArrInstant = EventList(1,4);
        EventList(1,:) = [];

        switch Event
            case ARRIVAL_DATA
                TOTALPACKETS_DATA = TOTALPACKETS_DATA + 1;
                tmp = Clock + exprnd(1/lambda);
                EventList = [EventList; ARRIVAL_DATA, tmp, GeneratePacketSizeData
                    (), tmp];
                if STATE == 0
                    STATE = 1;
                    EventList = [EventList; DEPARTURE_DATA, Clock + 8*PacketSize/(
                        C*10^6), PacketSize, Clock];
                else
                    if QUEUEOCCUPATION + PacketSize <= f
                        QUEUE = [QUEUE; PacketSize, Clock, ARRIVAL_DATA];
                        QUEUEOCCUPATION = QUEUEOCCUPATION + PacketSize;
                    else
                        LOSTPACKETS_DATA = LOSTPACKETS_DATA + 1;
                    end
                end

            case ARRIVAL_VOIP
                TOTALPACKETS_VOIP = TOTALPACKETS_VOIP + 1;
                tmpVoIP = Clock + 0.016 + (0.024 - 0.016)*rand();
                EventList = [EventList; ARRIVAL_VOIP, tmpVoIP,
                    GeneratePacketSizeVoIP(), tmpVoIP];
                if STATE == 0
                    STATE = 1;
                    EventList = [EventList; DEPARTURE_VOIP, Clock + 8*PacketSize/(
                        C*10^6), PacketSize, Clock];
                else
                    if QUEUEOCCUPATION + PacketSize <= f
                        QUEUE = [QUEUE; PacketSize, Clock, ARRIVAL_VOIP];
                        QUEUEOCCUPATION = QUEUEOCCUPATION + PacketSize;
                    else
                        LOSTPACKETS_VOIP = LOSTPACKETS_VOIP + 1;
                    end
                end
```

```
74
75              case DEPARTURE_DATA
76                  numBits = PacketSize * 8;
77                  if rand() < 1 - (1 - b)^numBits
78                      LOSTPACKETS_DATA = LOSTPACKETS_DATA + 1;
79                  else
80                      TRANSBYTES_DATA = TRANSBYTES_DATA + PacketSize;
81                      DELAYS_DATA = DELAYS_DATA + (Clock - ArrInstant);
82                      if Clock - ArrInstant > MAXDELAY_DATA
83                          MAXDELAY_DATA = Clock - ArrInstant;
84                      end
85                      TRANSPACKETS_DATA = TRANSPACKETS_DATA + 1;
86                  end
87
88                  if QUEUEOCCUPATION > 0
89                      if QUEUE(1,3) == ARRIVAL_DATA
90                          EventList = [EventList; DEPARTURE_DATA, Clock + 8*QUEUE
                                (1,1)/(C*10^6), QUEUE(1,1), QUEUE(1,2)];
91                      else
92                          EventList = [EventList; DEPARTURE_VOIP, Clock + 8*QUEUE
                                (1,1)/(C*10^6), QUEUE(1,1), QUEUE(1,2)];
93                      end
94                      QUEUEOCCUPATION = QUEUEOCCUPATION - QUEUE(1,1);
95                      QUEUE(1,:) = [];
96                  else
97                      STATE = 0;
98                  end
99
100             case DEPARTURE_VOIP
101                 numBits = PacketSize * 8;
102                 if rand() < 1 - (1 - b)^numBits
103                     LOSTPACKETS_VOIP = LOSTPACKETS_VOIP + 1;
104                 else
105                     TRANSBYTES_VOIP = TRANSBYTES_VOIP + PacketSize;
106                     DELAYS_VOIP = DELAYS_VOIP + (Clock - ArrInstant);
107                     if Clock - ArrInstant > MAXDELAY_VOIP
108                         MAXDELAY_VOIP = Clock - ArrInstant;
109                     end
110                     TRANSPACKETS_VOIP = TRANSPACKETS_VOIP + 1;
111                 end
112
113                 if QUEUEOCCUPATION > 0
114                     if QUEUE(1,3) == ARRIVAL_VOIP
115                         EventList = [EventList; DEPARTURE_VOIP, Clock + 8*QUEUE
                                (1,1)/(C*10^6), QUEUE(1,1), QUEUE(1,2)];
116                     else
```

```
                        EventList = [EventList; DEPARTURE_DATA, Clock + 8*QUEUE
                            (1,1)/(C*10^6), QUEUE(1,1), QUEUE(1,2)];
                    end
                    QUEUEOCCUPATION = QUEUEOCCUPATION - QUEUE(1,1);
                    QUEUE(1,:) = [];
                else
                    STATE = 0;
                end
            end
        end

        PLdata = 100*LOSTPACKETS_DATA/TOTALPACKETS_DATA;
        APDdata = 1000*DELAYS_DATA/TRANSPACKETS_DATA;
        MPDdata = 1000*MAXDELAY_DATA;

        PLVoIP = 100*LOSTPACKETS_VOIP/TOTALPACKETS_VOIP;
        APDVoIP = 1000*DELAYS_VOIP/TRANSPACKETS_VOIP;
        MPDVoIP = 1000*MAXDELAY_VOIP;

        TT = 1e-6*(TRANSBYTES_DATA + TRANSBYTES_VOIP)*8/Clock;

    end

    function out = GeneratePacketSizeData()
        aux = rand();
        aux2 = [65:109 111:1517];
        if aux <= 0.19
            out = 64;
        elseif aux <= 0.19 + 0.23
            out = 110;
        elseif aux <= 0.19 + 0.23 + 0.17
            out = 1518;
        else
            out = aux2(randi(length(aux2)));
        end
    end

    function out = GeneratePacketSizeVoIP()
        out = 109 + randi(21);
    end
```

B. Appendix

Listing B.1: Sim4A

```
function [PLdata, PLVoIP, APDdata, APDVoIP, MPDdata, MPDVoIP, TT] = Sim4A(lambda,
    C, f, P, n, p)
    ARRIVAL_DATA = 0;
    ARRIVAL_VOIP = 2;
    DEPARTURE_DATA = 1;
    DEPARTURE_VOIP = 3;

    STATE = 0;
    QUEUEOCCUPATION = 0;
    QUEUE = [];

    TOTALPACKETS_DATA = 0;
    LOSTPACKETS_DATA = 0;
    TRANSPACKETS_DATA = 0;
    TRANSBYTES_DATA = 0;
    DELAYS_DATA = 0;
    MAXDELAY_DATA = 0;

    TOTALPACKETS_VOIP = 0;
    LOSTPACKETS_VOIP = 0;
    TRANSPACKETS_VOIP = 0;
    TRANSBYTES_VOIP = 0;
    DELAYS_VOIP = 0;
    MAXDELAY_VOIP = 0;

    Clock = 0;
    tmp = Clock + exprnd(1/lambda);
    EventList = [ARRIVAL_DATA, tmp, GeneratePacketSizeData(), tmp];

    for i = 1:n
        tmpVoIP = Clock + 0.02*rand();
        EventList = [EventList; ARRIVAL_VOIP, tmpVoIP, GeneratePacketSizeVoIP(),
            tmpVoIP];
```

```
32      end
33
34      while TRANSPACKETS_DATA + TRANSPACKETS_VOIP < P
35          EventList = sortrows(EventList,2);
36          Event = EventList(1,1);
37          Clock = EventList(1,2);
38          PacketSize = EventList(1,3);
39          ArrInstant = EventList(1,4);
40          EventList(1,:) = [];
41
42          switch Event
43              case ARRIVAL_DATA
44                  TOTALPACKETS_DATA = TOTALPACKETS_DATA + 1;
45                  tmp = Clock + exprnd(1/lambda);
46                  EventList = [EventList; ARRIVAL_DATA, tmp, GeneratePacketSizeData
                        (), tmp];
47
48                  if STATE == 0
49                      STATE = 1;
50                      EventList = [EventList; DEPARTURE_DATA, Clock + 8*PacketSize/(
                            C*10^6), PacketSize, Clock];
51                  else
52                      if QUEUEOCCUPATION + PacketSize <= f * (p/100)
53                          QUEUE = [QUEUE; PacketSize, Clock, ARRIVAL_DATA];
54                          QUEUEOCCUPATION = QUEUEOCCUPATION + PacketSize;
55                      else
56                          LOSTPACKETS_DATA = LOSTPACKETS_DATA + 1;
57                      end
58                  end
59
60              case ARRIVAL_VOIP
61                  TOTALPACKETS_VOIP = TOTALPACKETS_VOIP + 1;
62                  tmpVoIP = Clock + 0.016 + (0.024 - 0.016)*rand();
63                  EventList = [EventList; ARRIVAL_VOIP, tmpVoIP,
                        GeneratePacketSizeVoIP(), tmpVoIP];
64                  if STATE == 0
65                      STATE = 1;
66                      EventList = [EventList; DEPARTURE_VOIP, Clock + 8*PacketSize/(
                            C*10^6), PacketSize, Clock];
67                  else
68                      if QUEUEOCCUPATION + PacketSize <= f
69                          QUEUE = [QUEUE; PacketSize, Clock, ARRIVAL_VOIP];
70                          QUEUEOCCUPATION = QUEUEOCCUPATION + PacketSize;
71                      else
72                          LOSTPACKETS_VOIP = LOSTPACKETS_VOIP + 1;
73                      end
```

```
74                  end
75
76              case DEPARTURE_DATA
77                  TRANSBYTES_DATA = TRANSBYTES_DATA + PacketSize;
78                  DELAYS_DATA = DELAYS_DATA + (Clock - ArrInstant);
79                  if Clock - ArrInstant > MAXDELAY_DATA
80                      MAXDELAY_DATA = Clock - ArrInstant;
81                  end
82                  TRANSPACKETS_DATA = TRANSPACKETS_DATA + 1;
83
84                  if QUEUEOCCUPATION > 0
85                      QUEUE = sortrows(QUEUE, 3, "descend");
86
87                      if QUEUE(1,3) == ARRIVAL_DATA
88                          EventList = [EventList; DEPARTURE_DATA, Clock + 8*QUEUE
                                (1,1)/(C*10^6), QUEUE(1,1), QUEUE(1,2)];
89                      else
90                          EventList = [EventList; DEPARTURE_VOIP, Clock + 8*QUEUE
                                (1,1)/(C*10^6), QUEUE(1,1), QUEUE(1,2)];
91                      end
92
93                      QUEUEOCCUPATION = QUEUEOCCUPATION - QUEUE(1,1);
94                      QUEUE(1,:) = [];
95                  else
96                      STATE = 0;
97                  end
98
99              case DEPARTURE_VOIP
100                 TRANSBYTES_VOIP = TRANSBYTES_VOIP + PacketSize;
101                 DELAYS_VOIP = DELAYS_VOIP + (Clock - ArrInstant);
102                 if Clock - ArrInstant > MAXDELAY_VOIP
103                     MAXDELAY_VOIP = Clock - ArrInstant;
104                 end
105                 TRANSPACKETS_VOIP = TRANSPACKETS_VOIP + 1;
106
107                 if QUEUEOCCUPATION > 0
108                     QUEUE = sortrows(QUEUE, 3, "descend");
109
110                     if QUEUE(1,3) == ARRIVAL_VOIP
111                         EventList = [EventList; DEPARTURE_VOIP, Clock + 8*QUEUE
                                (1,1)/(C*10^6), QUEUE(1,1), QUEUE(1,2)];
112                     else
113                         EventList = [EventList; DEPARTURE_DATA, Clock + 8*QUEUE
                                (1,1)/(C*10^6), QUEUE(1,1), QUEUE(1,2)];
114                     end
115
```

```
                    QUEUEOCCUPATION = QUEUEOCCUPATION - QUEUE(1,1);
                    QUEUE(1,:) = [];
                else
                    STATE = 0;
                end
        end
    end

    PLdata = 100*LOSTPACKETS_DATA/TOTALPACKETS_DATA;
    APDdata = 1000*DELAYS_DATA/TRANSPACKETS_DATA;
    MPDdata = 1000*MAXDELAY_DATA;

    PLVoIP = 100*LOSTPACKETS_VOIP/TOTALPACKETS_VOIP;
    APDVoIP = 1000*DELAYS_VOIP/TRANSPACKETS_VOIP;
    MPDVoIP = 1000*MAXDELAY_VOIP;

    TT = 1e-6*(TRANSBYTES_DATA + TRANSBYTES_VOIP)*8/Clock;

end

function out = GeneratePacketSizeData()
    aux = rand();
    aux2 = [65:109 111:1517];
    if aux <= 0.19
        out = 64;
    elseif aux <= 0.19 + 0.23
        out = 110;
    elseif aux <= 0.19 + 0.23 + 0.17
        out = 1518;
    else
        out = aux2(randi(length(aux2)));
    end
end

function out = GeneratePacketSizeVoIP()
    out = 109 + randi(21);
end
```