Modelação e Desempenho de Redes e Serviços

First mini-project report

Bruno Gomes, Diogo Silva

Universidade de Aveiro

Modelação e Desempenho de Redes e Serviços

DEPARTAMENTO DE ELETRÓNICA
TELECOMUNICAÇÕES E INFORMÁTICA

First mini-project report

Bruno Gomes, Diogo Silva
(103320) brunofgomes@ua.pt, (104341) diogobranco.as@ua.pt

23/10/2024

Contents

Chapter 1

Task 1

1.1 Exercise 1.a)

1.1.1 Results and Conclusions

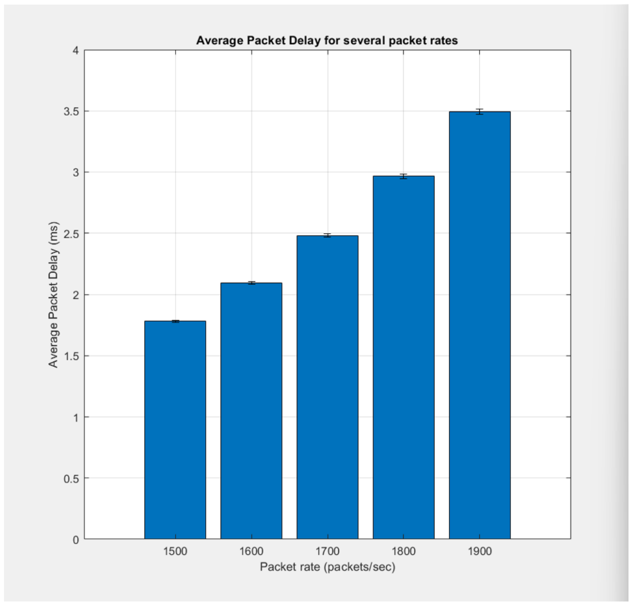


Figure 1.1: Average packet delay results with $b = 10^{-6}$

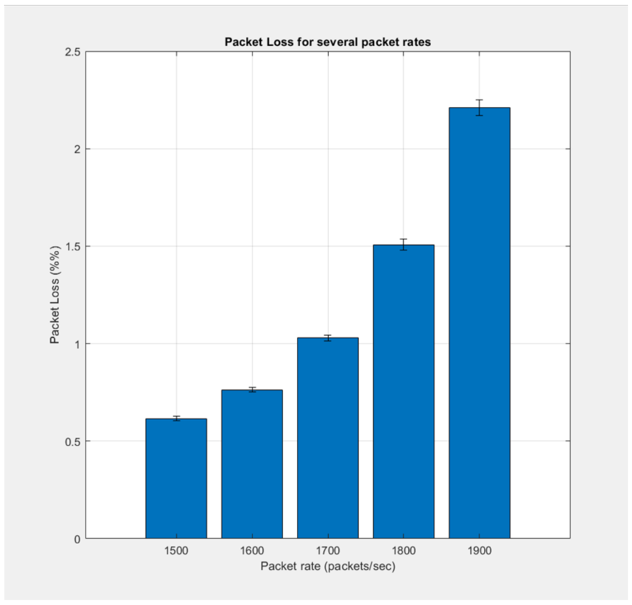


Figure 1.2: Packet loss results with $b = 10^{-6}$

Firstly, running *Sim2* allowed us to understand that higher packet rates will directly influence the average packet delay statistics, the higher the packet rate, the higher the average packet delay will be. This is due to the fact that higher packet rates will increase the traffic rate of the system, causing the queue to fill up quicker and eventually increasing the waiting times. Because if packets arrive at an instant where the link is occupied, they will have to wait in the queue before entering the link, the time that a packet waits in a queue contributes for the transmission delay of it, specially if the packet is further down the queue.

The packet delay depends mainly on two factors, the service time which is time that takes to transmit each packet, and the queuing delay which is the time that packets spend on the queue when the link is occupied with previous transmissions. We can calculate the service time by dividing the average packet size which is approximately **620 bytes**, by the capacity of the link which was defined as ***10 Mbps***.

$$D_t = \frac{620 \times 8}{10 \times 10^6} \approx 0.496\,\text{ms} \tag{1.1}$$

This means that in the absence of queuing, each packet takes about roughly 0.5 *ms* to transmit. However, as soon as packet arrival rates increase, and packets start entering the queue, the queue delay will start to add to the overall delay that each packet takes to transmit. The delay will also depend on how many packets are ahead in the queue. From the simulation results we got a linear growth in the average packet delay, it went from approximately 2 *ms* to 3.5 *ms*. Additionally, we can confirm this by calculating how much data is transferred in a second for the different packet rates that were tested in this simulation, consider for example the arrival rate of 1900 packets/sec, we can calculate the amount of data transferred doing the following:

$$Data_s = \frac{1900 \times 620 \times 8}{10^6} \approx 9.4\,\text{Mbps} \tag{1.2}$$

Packet Rate (pps)	**Data per second (Mbps)**
1500	7.44
1600	7.94
1700	8.43
1800	8.93
1900	9.42

Table 1.1: Packet Rate vs Data per second

The table above, explains why the increment of the average packet delay is linear, the increments of the data transferred are all around the value of 0.50 *Mbps*. Also, we can clearly see why the packet delay increases, because the higher the packet rate the more likely that the link will become congested, when the packet rate is 1900 packet/sec, the data that is transferred almost reaches the links capacity limit which is 10 *Mbps*, as a result, incoming packets will resort to wait in the queue before entering the link, therefore increasing their transmission delay.

Regarding the packet loss statistic, as we can observe in figure 1.2, the increase in packet loss is exponential meaning that when packet rate increases, the packet loss not only grows, but does so at an accelerating rate, indicating a steepening trend as the load on the network intensifies. In our simulations, the packet loss varies from approximately 0.6% to 2.2%.

Three main factors affect the packet loss statistic, which are the queue size, the packet arrival rate and bit error rate. To get into detail, few calculations can be made in order to explain why does packet loss increase. Firstly, it is important to know how many packets can the queue actually hold, since our queue has a size of 10 *KB*, which gives us exactly 10240 *Bytes* of queue size and given that the average packet size is approximately 620 *Bytes*, the queue can hold approximately 16 packets:

$$Queue\ size = \frac{10240}{620} \approx 16\,\text{packets} \tag{1.3}$$

At higher packets rates, such as 1900 packets/sec, packets arrival rate is very close to the service rate of the link which is approximately 2016 packets/second:

$$Service\ time = \frac{10 \times 10^6}{620 \times 8} \approx 2016\,\text{packets/second} \tag{1.4}$$

Given this fact, we will predict that the link will fill up quickly, and then packets will have to use the queue which only stores around 16 packets of data, which is not enough to accommodate all packets that reach at a rate of 1900 packets/sec. That is why packet rates like 1800 and 1900 packets/second have an increased packet loss. Finally, it is also important to note that packets have a small chance (b $= 10^{-6}$) of being sent to the link with errors, if a packet is sent with errors, it will automatically be discarded by the link, therefore contributing to packet loss.

1.2 Exercise 1.b)

1.2.1 Results and Conclusions

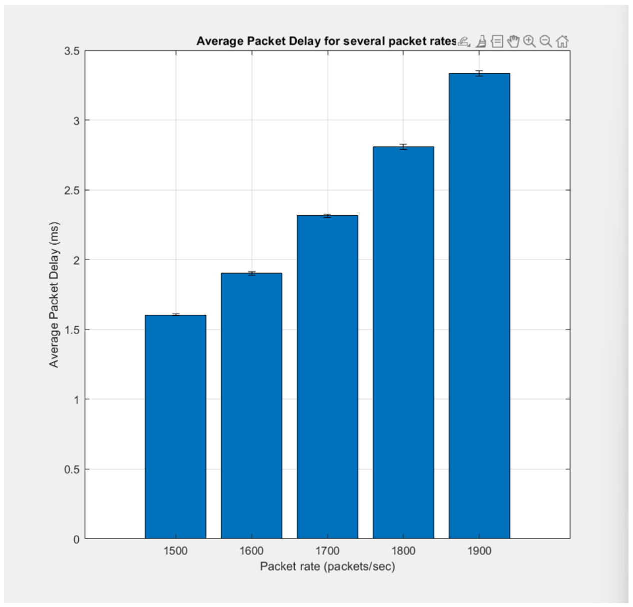


Figure 1.3: Average packet delay results with $b = 10^{-4}$

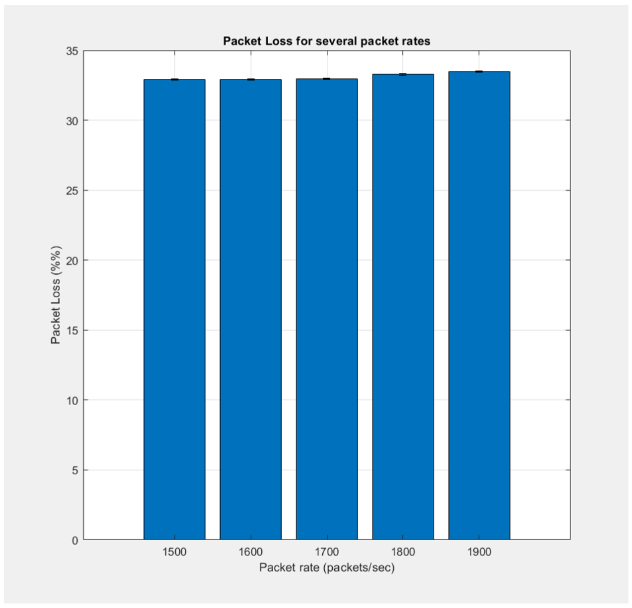


Figure 1.4: Packet loss results with $b = 10^{-4}$

Analyzing and comparing these results with the outcomes obtained in exercise **1.a**, starting with the average packet delay statistic we can actually see a small decrease in packet delay in exercise **1.b**, these values range from 1.6 ms to less than 3.5 ms approximately, these small differences occur because packets with errors are not serviced, since the bit error rate was increased by a order of $\times 100$, there are more chances of errors happening in packet transmission, therefore transmission delays of lost packets will not count for the total delay of the system, as a consequence the average packet delay will decrease by a small amount. These aspects are evident in the following code section of *Sim2*.

```

case DEPARTURE           % If first event is a DEPARTURE
    if (rand() < (1-b)^(PacketSize*8)) % prob. of sending packet without errors
        TRANSBYTES= TRANSBYTES + PacketSize;
        DELAYS= DELAYS + (Clock - ArrInstant);
        if Clock - ArrInstant > MAXDELAY
            MAXDELAY= Clock - ArrInstant;
        end
        TRANSPACKETS= TRANSPACKETS + 1;
    else  %packet will not be sent because of errors
        LOSTPACKETS = LOSTPACKETS + 1;
    end
```

The significant increase in packet loss in exercise **1.b** can be explained by the fact that, in addition of the queue being quite short (*10 KB*), now there is also the component of the bit error rate being substantially higher, since packets have higher chances of entering the link with errors, consequently there are more chances of packets being dropped, therefore contributing to overall packet loss. Since the bit error rate remains constant during all the transmission, the values of packet loss for different packet arrival rates, increase in equal proportions.

1.3 Exercise 1.c)

1.3.1 Code

```

range_probs = (1 - (0.19 + 0.23 + 0.17))/(109-65+1 + 1517-111+1);

x = 64:1518;

pack_loss = zeros(1,2); % BER = 10^-4 and BER = 10^-6 respectively

j = 1;

for i = 64:1518
    if i == 64
        pack_loss(j) = pack_loss(j) + (1-nchoosek(64*8,0)*((10^-4)^0)*(1-(10^-4))^(64*8)) * 0.19;
        pack_loss(j+1) = pack_loss(j+1) + (1-nchoosek(64*8,0)*((10^-6)^0)*(1-(10^-6))^(64*8)) * 0.19;

    elseif i == 110
        pack_loss(j) = pack_loss(j) + (1-nchoosek(110*8,0)*((10^-4)^0)*(1-(10^-4))^(110*8)) * 0.23;
        pack_loss(j+1) = pack_loss(j+1) + (1-nchoosek(110*8,0)*((10^-6)^0)*(1-(10^-6))^(110*8)) * 0.23;

    elseif i == 1518
        pack_loss(j) = pack_loss(j) + (1-nchoosek(1518*8,0)*((10^-4)^0)*(1-(10^-4))^(1518*8)) * 0.17;
        pack_loss(j+1) = pack_loss(j+1) + (1-nchoosek
           (1518*8,0)*((10^-6)^0)*(1-(10^-6))^(1518*8)
           ) * 0.17;

    else
        pack_loss(j) = pack_loss(j) + (1-nchoosek(i
           *8,0)*((10^-4)^0)*(1-(10^-4))^(i*8)) *
           range_probs;
        pack_loss(j+1) = pack_loss(j+1) + (1-nchoosek(
           i*8,0)*((10^-6)^0)*(1-(10^-6))^(i*8)) *
           range_probs;
    end
end

pack_loss = (pack_loss ./ (0.19 + 0.23 + 0.17 + ((109
   - 65 + 1) + (1517 - 111 + 1)) * range_probs)) *
   100;

fprintf("Theoretical packet loss for BER 10^-4 is %2.1
   f%%\nTheoretical packet loss for BER 10^-6 is %2.1f
   %%\n", pack_loss(1),pack_loss(2));
```

1.3.2 Results and Conclusions

We obtained a theoretical packet loss value for the bit error rate 10^{-6} of approximately 0.5% and for a bit error rate of 10^{-4} we obtained a packet loss value of 32.8%. The result we got for the bit error rate of 10^{-4} matched our simulation results that we obtained in exercise **1.b**, all bars appear to reach the value of 32.8% approximately, in this case the bit error rate is the only factor that seems to affect packet loss. However, the result we got for the other bit rate ($b = 10^{-6}$) does not match the simulation results obtained in exercise **1.a**, this is because in this specific case there are other factors that we should take into account that impact packet loss statistics such as queue size. Since the queue size we are using is very reduced, as soon as the packet arrival rates increase, so does the probability of packets being dropped, in exercise **1.a** that is the biggest influence in packet loss, since the bit error rate is very reduced.

Chapter 2

Task 2

2.1 Exercise 2.a)

2.1.1 Code

```
    case DEPARTURE          % If first event is a DEPARTURE
            if(rand() < (1-b)^(PacketSize*8))
                if(PacketType == DATA)
                    TRANSBYTESD= TRANSBYTESD + PacketSize;
                    DELAYSD= DELAYSD + (Clock - ArrInstant);
                    if Clock - ArrInstant > MAXDELAYD
                        MAXDELAYD= Clock - ArrInstant;
                    end
                    TRANSPACKETSD= TRANSPACKETSD + 1;
                else %VoIP
                    TRANSBYTESVP= TRANSBYTESVP + PacketSize;
                    DELAYSVP= DELAYSVP + (Clock - ArrInstant);
                    if Clock - ArrInstant > MAXDELAYVP
                        MAXDELAYVP= Clock - ArrInstant;
                    end
                    TRANSPACKETSVP= TRANSPACKETSVP + 1;
                end
            else
                if(PacketType == DATA)
```

```
                    LOSTPACKETSD = LOSTPACKETSD + 1;
                else
                    LOSTPACKETSVP = LOSTPACKETSVP + 1;
                end
            end
```

2.1.2 Results and Conclusions

Sim3a is the same as the Sim3 version with the detail of the added Bit Error Rate (BER) from the Sim2.

So just like in Sim2, when we reach the end of the transmission of the packet, we check if a randomly generated value is less than the probability of the packet having been corrupted before continuing.

The key difference comes that when incrementing the LostPackets we must check if they are Data or VoIP.

2.2 Exercise 2.b)

2.2.1 Results and Conclusions

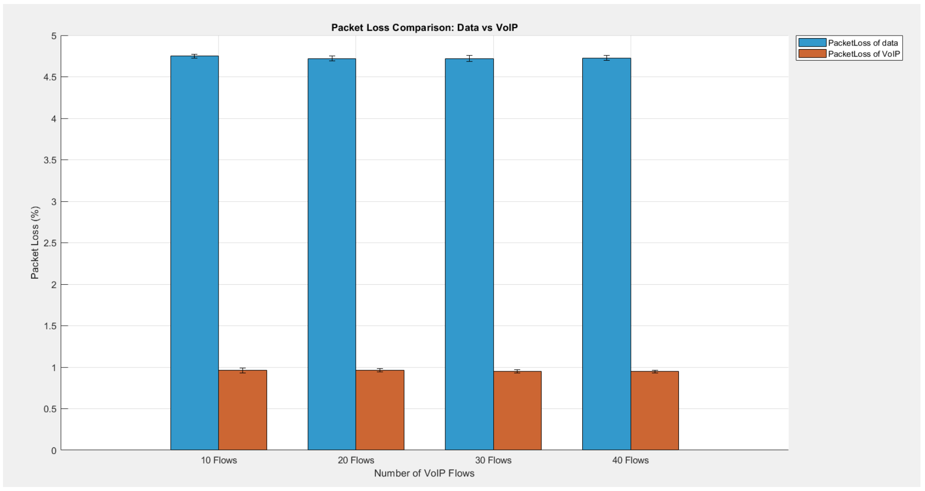


Figure 2.1: Packet loss through VoIP flows

The percentage of packet loss stays relatively the same through the different VoIP flows with no relevant changes. This might be due to the fact that we did not change any parameters like the BER, queue size or link bandwidth and suggests the queue size is big enough to handle the increased VoIP flows.

The fact that the packet loss percentage is smaller for VoIP packets ($\tilde{0}.9\%$) than for Data ones ($\tilde{4}.7\%$) is due to the fact that VoIP packets are generally smaller (110-130 bytes) compared to Data (65-1518 bytes). This means that for the same BER there is a higher chance for a data packet to be corrupted than a VoIP one.

2.3 Exercise 2.c)

2.3.1 Results and Conclusions

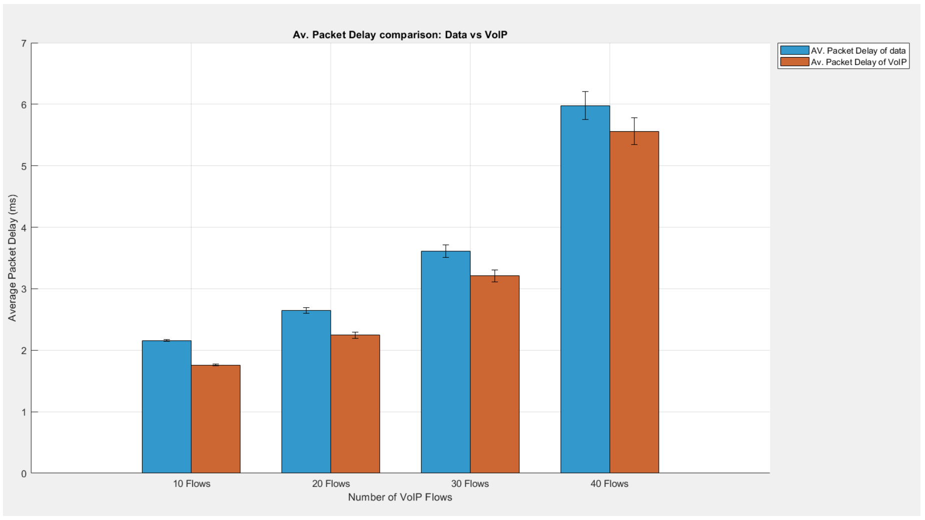


Figure 2.2: Packet loss through VoIP flows

As it can be seen by the 2.2 figure, with the increase of the VoIP flows, the average Packet delay increases, and so do the error intervals.

The increase of the average packet delay can be explained by the fact that we have all the same parameters in Sim3a except for the number of VoIP packets that increases. This results in a higher number of packets competing for the same resources, which results in a higher Packet Delay numbers. This congestion also results in higher confidence intervals as the delay numbers fluctuate more.

2.4 Exercise 2.d)

2.4.1 Results and Conclusions

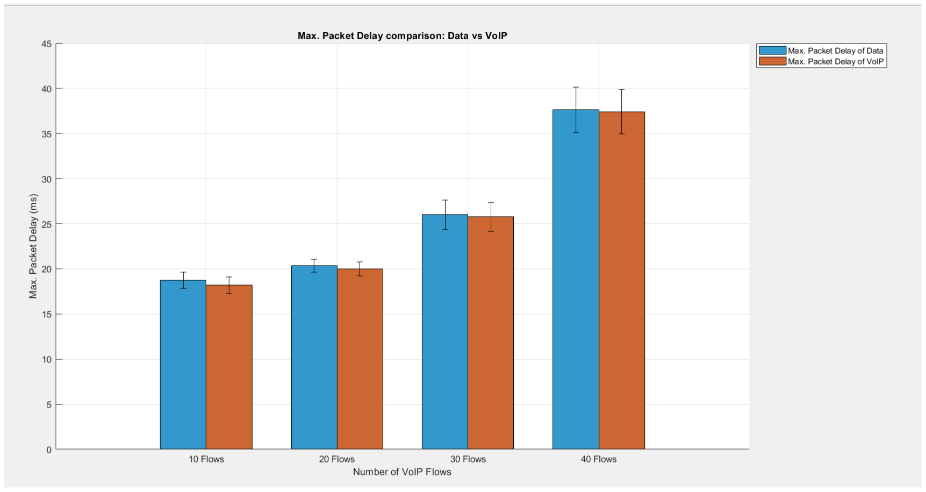


Figure 2.3: Maximum Packet Delay through VoIP flows

Just like with the 2.2 figure, we also notice a general increase in the time for the Maximum Packet Delay both for Data and VoIP packets. As we have more VoIP flows, we will increase the congestion since we didn't increase the other parameters of the simulation. This will result in what we see in the 2.3 figure. We can also notice the confidence intervals getting bigger as a result of the greater variability that leads to less predictability in the results.

2.5 Exercise 2.e)

2.5.1 Results and Conclusions

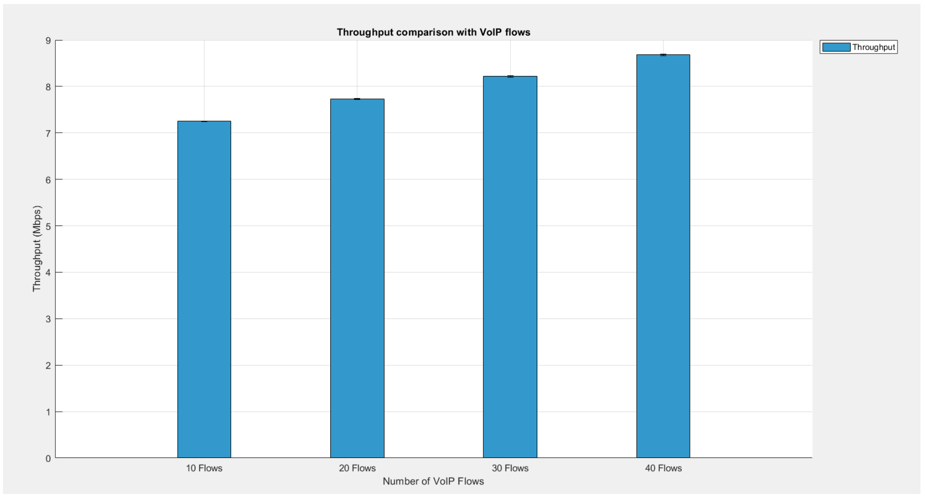


Figure 2.4: Transmitted Throughput across VoIP flows

The Transmitted Troughput (TT) is the sum of all the successfully transmitted (data and VoIP) packets trough the simulation divided by it's time. It is calculated by the following formula:

$$TT = \frac{10^{-6} \cdot (\text{TRANSBYTESVP} + \text{TRANSBYTESD}) \cdot 8}{\text{Clock}} \tag{2.1}$$

The results we get from the various simulations show us a small but gradual increase in the TT across the various VoIP flows.

While we could expect the TT to reduce because of the higher delays that increase the simulations duration, we also have various factors that increase it such as the constant packet loss and the increased arrival of the VoIP packets that means that we will have more packets being successfully transmitted with higher number of VoIP flows,

2.6 Exercise 2.f)

2.6.1 Code

```
f = 1e6;
prate = 1500;
capacity = 10 * 10^6;
b = 1e-5;
voip_flows = [10, 20, 30, 40];

pdata = [0.19, 0.23, 0.17];
sizes_data = [64, 110, 1518];

prob_left_data = (1 - sum(pdata)) / ((109 - 65 + 1) +
    (1517 - 111 + 1));
avg_size_data = sum(pdata .* sizes_data);

data_success_prob = sum(pdata .* (1 - b).^(sizes_data
    * 8));

left_packets = [65:109, 111:1517];
avg_size_data = avg_size_data + prob_left_data * sum(
    left_packets);
data_success_prob = data_success_prob + prob_left_data
     * sum((1 - b).^(left_packets * 8));

avg_size_voip = (110 + 130) / 2;
voip_success_prob = mean((1 - b).^(110:130 * 8));

lambda_voip = 1 / mean([0.016, 0.024]);

for i = 1:length(voip_flows)
    num_flows = voip_flows(i);
    throughput_data = (prate * avg_size_data * 8 *
        data_success_prob) / 1e6;
    throughput_voip = (lambda_voip * avg_size_voip * 8
         * voip_success_prob) / 1e6;

    total_throughput = throughput_data + (num_flows *
        throughput_voip);
    fprintf("Throughput (Mbps) for %d VoIP flows = %.4
        f\n", num_flows, total_throughput);
end
```

Firstly we introduce all the simulation variables relevant for the calculation of the throughput such as the packet rate, connection capacity, queue size, the bit error rate and the varying number of VoIP flows.

Then we calculate the avg size of the data packets and VoIP ones. Following that we must check if the packet will be successfully transmitted by factoring in the BER

$$(1 - \text{BER})^{\text{packet size in bits}} \tag{2.2}$$

Once that is done we use the following formulas to calculate the Throughput for data and VoIP:

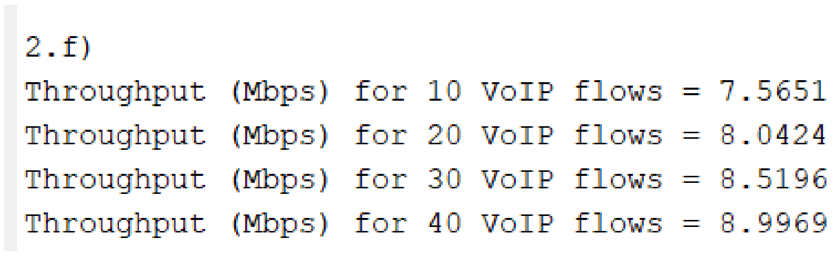
$$\text{Throughput}_{\text{data}} = \frac{\text{prate} \times \text{avg\_size\_data} \times 8 \times \text{data\_success\_prob}}{10^6} \tag{2.3}$$

$$\text{Throughput}_{\text{VoIP}} = \frac{\lambda_{\text{voip}} \times \text{avg\_size\_voip} \times 8 \times \text{voip\_success\_prob}}{10^6} \tag{2.4}$$

Finally we get the transmitted throughput by multiplying the VoIP throughput with the number o VoIP flows

$$\text{Total Throughput} = \text{Throughput}_{\text{da-ta}} + (\text{num\_flows} \times \text{Throughput}_{\text{VoIP}}) \tag{2.5}$$

Results and Conclusions



```
2.f)
Throughput (Mbps) for 10 VoIP flows = 7.5651
Throughput (Mbps) for 20 VoIP flows = 8.0424
Throughput (Mbps) for 30 VoIP flows = 8.5196
Throughput (Mbps) for 40 VoIP flows = 8.9969
```

Figure 2.5: TT theoretical results

The results we received from the theoretical calculations are expected as they fit in with the ones we obtained in the simulations, as seen in the 2.4 image.

Chapter 3

Task 3

3.1 Exercise 3.a)

3.1.1 Results and Conclusions

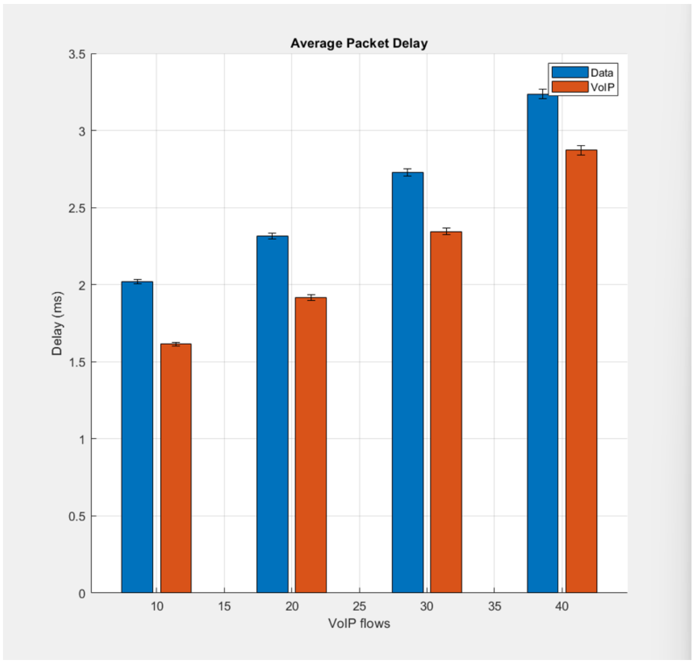


Figure 3.1: Average packet delay calculated by *Sim3*

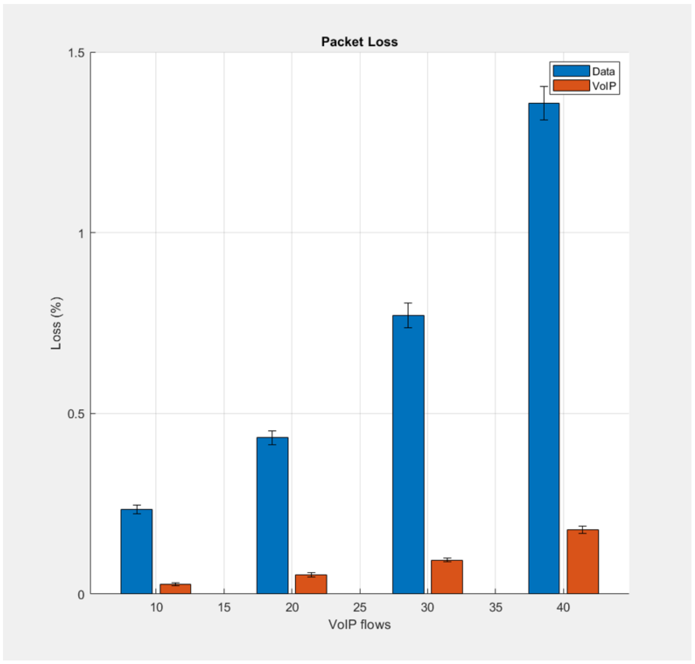


Figure 3.2: Packet loss calculated by *Sim3*

To start with, regarding the average packet delay we can observe that when the number of VoIP packet flows increases, both delays from data and VoIP packets increase, however, data packets seem to suffer a bigger delay when compared to VoIP packets, this happens because since VoIP packets have a smaller size (average of 120 bytes) when compared to data packets which have an average size of 620 bytes, this factor makes a significant difference when transmitting packets using a link that uses First In First Out (FIFO) discipline, we know that smaller packets have an easier time getting inside the link due to their size when compared to data packets that usually have bigger sizes. Additionally, it should be noted that VoIP packets have lower arrival rates, these packets arrive according to a uniform distribution between 16 ms and 24 ms, which gives us and average interval arriving time of 20 ms, we can obtain the average packet arrival rate for VoIP packets by doing the following:

$$Arrival\ rate = \frac{1}{0.02} = 50\,\text{packets/second} \tag{3.1}$$

This value is 30× lower than the data packets arrival rate, which means that VoIP packets do not load the link as much as data packets and are a less likely to encounter full queues, which leads to lower delays for these type of packets. On the other hand, because data packets have higher arrival rates and bigger sizes they are more likely to encounter a full queue therefore increasing delays for these type of packets.

Concerning the packet loss statistic, with everything that was said before, it is predicted that data packets would be way more affected by packet loss, since bigger packets have more difficulty entering the link, that would also translate in an increased chance of those type of packets being dropped.

3.2 Exercise 3.b)

3.2.1 Results and Conclusions

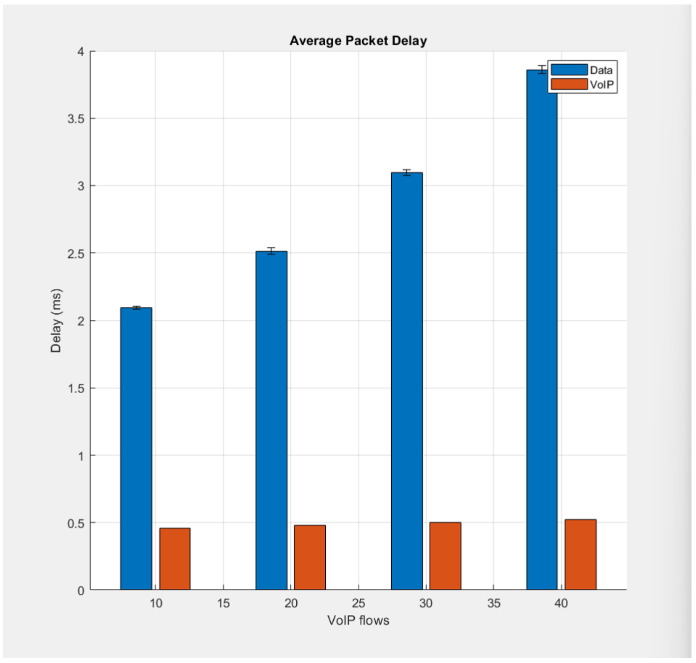


Figure 3.3: Average packet delay calculated by *Sim4*

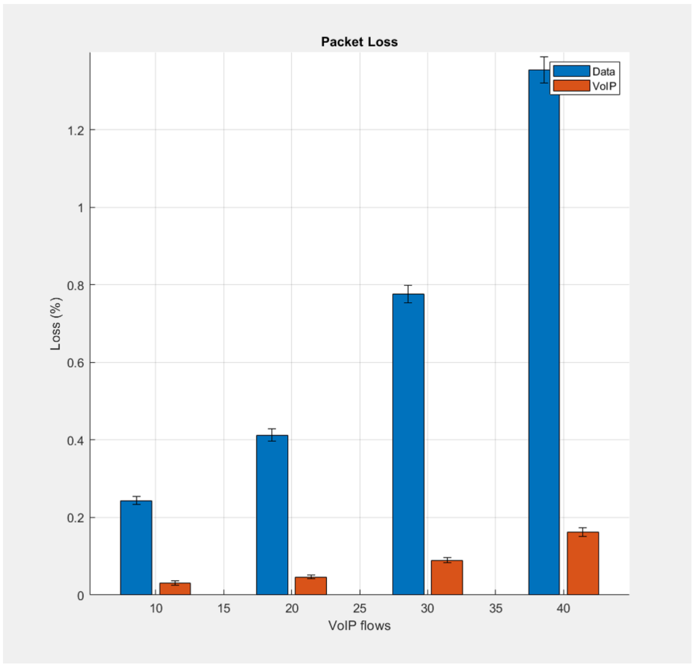


Figure 3.4: Packet loss calculated by *Sim4*

Comparing the packet loss obtained in this experiment with the packet loss statistics obtained in exercise **3.a** we can see that the packet loss remains unchanged, this means that the VoIP packet priority queuing mechanism did not introduce any significant changes in the packet loss aspect. VoIP packets will always have the advantage of being small which makes the entrance in the queue almost guaranteed when compared to bigger data packets. Also, the mechanism we introduced in *Sim4* influences mostly the transmission time of packets, not the probability of packets entering the queue of the link.

However, we do notice a big difference in average packet delay statistics, with *Sim4* we obtain much lower values of average packet delay for VoIP packets and a slightly higher value of packet delay for data packets, this happens because in *Sim4* there is a queuing mechanism which gives priority to all VoIP packets present in the queue, basically the code sorts the queue contents in a descending order, and since VoIP packets have a tag with the number 1 and data packets have a tag with number 0, the first packets to be taken out of the queue will be VoIP packets because they have a higher tag value than data packets. These aspects will influence greatly the delay of VoIP packets because they spend less time waiting in the queue since they have higher priority than data packets. Furthermore, it is also important to note that there are no interval errors present on the bars that represent the average packet delay for VoIP packets, this is because the variance of error values is close to zero.

3.3 Exercise 3.c)

3.3.1 Code

```
function [PLd, PLv , APDd, APDv] = Sim4A(lambda,C,f,P,
    n,p)
% INPUT PARAMETERS:
%  lambda - packet rate (packets/sec)
%  C      - link bandwidth (Mbps)
%  f      - queue size (Bytes)
%  P      - number of packets (stopping criterium)
%  n      - number of VoIP packet flows
%  p      - queue occupation threshold
% OUTPUT PARAMETERS:
%  PLd   - packet loss of data packets(%)
%  PLv   - packet loss of VoIP packets(%)
%  APDd  - average packet delay of data packets(
    milliseconds)
%  APDv  - average packet delay of VoIP packets(
    milliseconds)
%  MPDd  - maximum delay of data packets(milliseconds)
%  MPDv  - maximum delay of VoIP packets(milliseconds)
%  TT   - transmitted throughput (data+VoIP) (Mbps)

%Events:
ARRIVAL= 0;       % Arrival of a packet
DEPARTURE= 1;     % Departure of a packet

%Packet type
DATA = 0;
VOIP = 1;

%State variables:
STATE = 0;           % 0 - connection is free; 1 -
    connection is occupied
QUEUEOCCUPATION= 0; % Occupation of the queue (in
    Bytes)
QUEUE= [];           % Size and arriving time instant
    of each packet in the queue

dataQueueLimit = (p/100)*f;

%Statistical Counters:
TOTALPACKETSD= 0;      % No. of data packets arrived to
     the system
```

```
TOTALPACKETSV= 0;      % No. of voip packets arrived to the system
LOSTPACKETSD= 0;       % No. of data packets dropped due to buffer overflow
LOSTPACKETSV= 0;       % No. of voip packets dropped due to buffer overflow
TRANSPACKETSD= 0;      % No. of transmitted data packets
TRANSPACKETSV= 0;      % No. of transmitted voip packets
TRANSBYTESD= 0;        % Sum of the Bytes of transmitted data packets
TRANSBYTESV= 0;        % Sum of the Bytes of transmitted voip packets
DELAYSD= 0;            % Sum of the delays of transmitted data packets
DELAYSV= 0;            % Sum of the delays of transmitted voip packets
MAXDELAYD= 0;          % Maximum delay among all transmitted data packets
MAXDELAYV= 0;          % Maximum delay among all transmitted voip packets

% Initializing the simulation clock:
Clock= 0;

% Initializing the List of Events with the first ARRIVAL:
tmp= Clock + exprnd(1/lambda);
EventList = [ARRIVAL, tmp, GeneratePacketSize(), tmp, DATA]; % DATA is used to identify type of packet

%Initializing VoIP packets
for i = 1:n
    tmp = unifrnd(0, 0.02);     % time packet arrivals is uniform distribution between 0 ms and 20 ms
    EventList = [EventList; ARRIVAL, tmp, randi([110, 130]), tmp, VOIP];
end

%Simulation loop:
while (TRANSPACKETSD + TRANSPACKETSV)<P    % Stopping criterium
    EventList= sortrows(EventList,2);  % Order EventList by time
    Event= EventList(1,1);                % Get first event
    Clock= EventList(1,2);                %    and all
    PacketSize= EventList(1,3);         %    associated
    ArrInstant= EventList(1,4);     %    parameters.
    PacketType = EventList(1,5);
    EventList(1,:)= [];                   % Eliminate first event
    switch Event
        case ARRIVAL          % If first event is an ARRIVAL
          if(PacketType == DATA)
              TOTALPACKETSD= TOTALPACKETSD+1;
              tmp= Clock + exprnd(1/lambda);
              EventList = [EventList; ARRIVAL, tmp, GeneratePacketSize(), tmp,DATA];
              if STATE==0
                  STATE= 1;
                  EventList = [EventList; DEPARTURE, Clock + 8*PacketSize/(C*10^6), PacketSize, Clock,DATA];
              else
                  if (QUEUEOCCUPATION + PacketSize) <= dataQueueLimit %define queue threshold for data packets
                      QUEUE= [QUEUE;PacketSize , Clock, DATA];
                      QUEUEOCCUPATION= QUEUEOCCUPATION + PacketSize;
                  else
                      LOSTPACKETSD= LOSTPACKETSD + 1;
                  end
              end
          else % PacketType == VoIP
              TOTALPACKETSV= TOTALPACKETSV+1;
              tmp= Clock + unifrnd(0.016, 0.024);
              EventList = [EventList; ARRIVAL, tmp, randi([110,130]), tmp,VOIP];
              if STATE==0
                  STATE= 1;
                  EventList = [EventList; DEPARTURE, Clock + 8*PacketSize/(C*10^6), PacketSize, Clock,VOIP];
              else
```

```
                    if QUEUEOCCUPATION + PacketSize <= f
                        QUEUE= [QUEUE;PacketSize , Clock, VOIP];
                        QUEUEOCCUPATION= QUEUEOCCUPATION + PacketSize;
                    else
                        LOSTPACKETSV= LOSTPACKETSV + 1;
                    end
                end
            end
        case DEPARTURE          % If first event is a DEPARTURE
            if(PacketType == DATA)
                TRANSBYTESD= TRANSBYTESD + PacketSize;
                DELAYSD= DELAYSD + (Clock - ArrInstant);
                if Clock - ArrInstant > MAXDELAYD
                    MAXDELAYD= Clock - ArrInstant;
                end
                TRANSPACKETSD= TRANSPACKETSD + 1;
            else
                TRANSBYTESV= TRANSBYTESV + PacketSize;
                DELAYSV= DELAYSV + (Clock - ArrInstant);
                if Clock - ArrInstant > MAXDELAYV
                    MAXDELAYV= Clock - ArrInstant;
                end
                TRANSPACKETSV= TRANSPACKETSV + 1;
            end

            if QUEUEOCCUPATION > 0
                QUEUE = sortrows(QUEUE,3,"descend"); % voip packets will be the first to be removed from the queue
                EventList = [EventList; DEPARTURE, Clock + 8*QUEUE(1,1)/(C*10^6), QUEUE(1,1), QUEUE(1,2), QUEUE(1,3)]; %QUEUE(1,3) is the type of packet
                QUEUEOCCUPATION= QUEUEOCCUPATION - QUEUE(1,1);
                QUEUE(1,:)= [];
            else
```

```
                STATE= 0;
            end
    end
end

%Performance parameters determination:
PLd= 100*LOSTPACKETSD/TOTALPACKETSD;  % in percentage
PLv= 100*LOSTPACKETSV/TOTALPACKETSV;  % in percentage
APDd= 1000*DELAYSD/TRANSPACKETSD;     % in milliseconds
APDv= 1000*DELAYSV/TRANSPACKETSV;     % in milliseconds
MPDd= 1000*MAXDELAYD;                % in milliseconds
MPDv= 1000*MAXDELAYV;                % in milliseconds
TT= 1e-6*(TRANSBYTESD + TRANSBYTESV)*8/Clock;    % in Mbps

end

function out= GeneratePacketSize()
    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
```

3.3.2 Results and Conclusions

The main changes made to *Sim4A* were:

```
dataQueueLimit = (p/100)*f;
```

This line was added in order to define a queue size limit for incoming data packets, the parameter p is the limit that the queue will have for data packets, for instance if $p = 90$, it means that 10% of that queue size will never be used by data packets, therefore VoIP packets will have more allocated queue size for them.

The following code enforces the previously referenced rule, if an incoming data packet causes the queue to exceed its defined capacity, the packet is automatically discarded. This approach helps maintain the queue within its limit but also increases data packet loss statistics as a result.

```
if (QUEUEOCCUPATION + PacketSize) <= dataQueueLimit %
    define queue threshold for data packets
     QUEUE= [QUEUE;PacketSize , Clock, DATA];
     QUEUEOCCUPATION= QUEUEOCCUPATION + PacketSize;
```

By limiting the queue size available for data packets, it is expected that data packet loss will increase, while VoIP packet loss will decrease.

3.4 Exercise 3.d)

3.4.1 Results and Conclusions

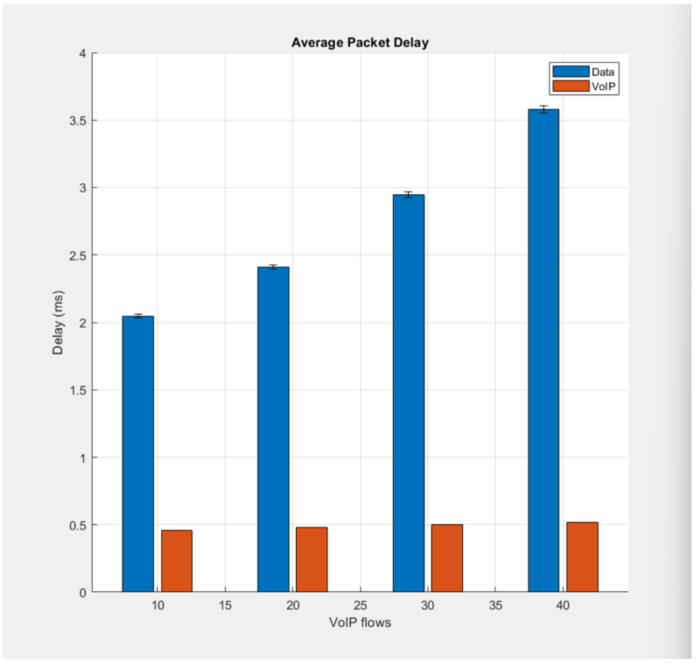


Figure 3.5: Average packet delay calculated by *Sim4A* with $p = 90$

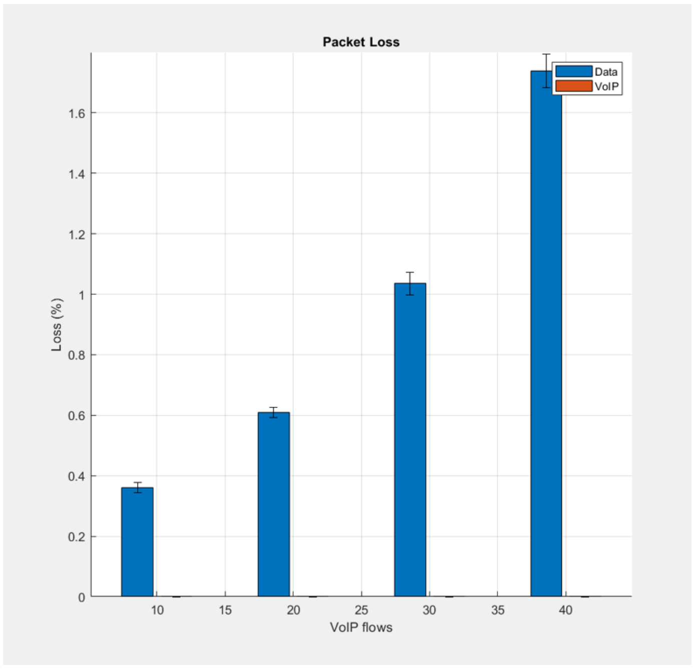


Figure 3.6: Packet loss calculated by *Sim4A* with $p = 90$

Firstly, regarding the average packet delay in this simulation we did not see much changes apart from the fact of the average packet delay of data packets being a little bit reduced when compared to exercise **3.b**, this is due to the fact that since the queue size is now limited for data packets, it consequently increases the chances of data packets being dropped, moreover this means that the system will not need to waste time transmitting some of these packets because they were dropped, which means that those transmissions times will not add up to the total system delay for data packets, therefore decreasing the overall average data packet delay.

The packet loss results obtained in this simulation showed that the new queue threshold for data packets resulted in an increased data packet lost percentage, because data packets now have only a smaller portion of the queue size that they have to share with other VoIP packets, as we have seen before, data packets have a bigger difficulty of entering the queue because they are bigger packets that try to enter the link at a very high rate when compared to VoIP packets, now that the queue is reduced for data packets this makes it even harder for these type of packets to enter the queue successfully. The 10% of queue size that is not being used by data packets creates a good buffer for VoIP packets to be transmitted peacefully without being dropped, the new feature that we added in *Sim4A* basically adds an "exclusive" zone for VoIP packets, and since the packets arrive at a slower rate when compared to data packets there is almost a guarantee that there will be space for VoIP packets when they need to enter the queue.

3.5 Exercise 3.e)

3.5.1 Results and Conclusions

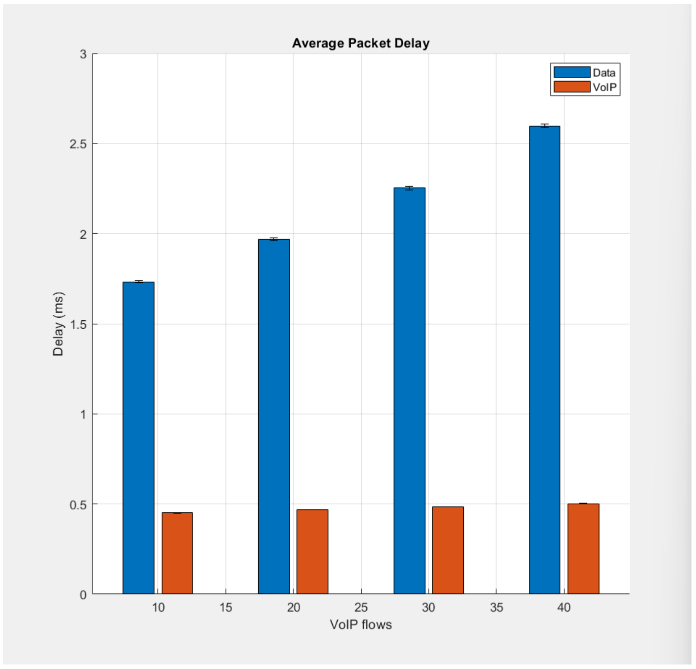


Figure 3.7: Average packet delay calculated by *Sim4A* with $p = 60$

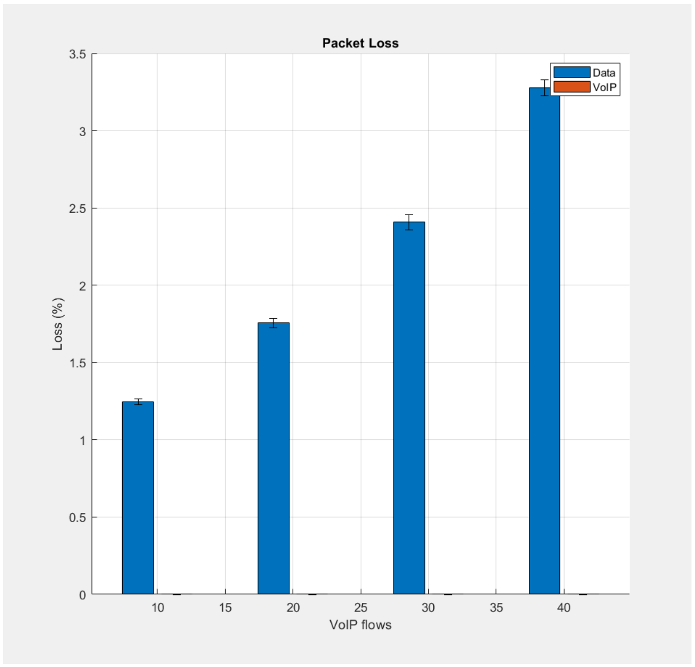


Figure 3.8: Packet loss calculated by *Sim4A* with $p = 60$

As predicted packet loss increased even more for data packets because now the queue size has limit of 60% for these kind of packets, also there is no packet loss for VoIP packets because these packets have 40% of the queue size reserved for them which is more than enough to avoid network congestion.

The average packet delay remained the same for VoIP packets, and for data packets it was reduced because since even more data packets are being dropped that also translates for lower transmissions delays since lost packets do not contribute for the overall delay of the system.

The results were basically the same as in exercise **3.d** but amplified.

To conclude, in exercise **3.b** we only had the aspect of priority queuing for VoIP packets which impacted the delay of the transmission of data packets negatively and affected the transmission of VoIP packets positively, then in exercises **3.d** and **3.e** it was introduced a new mechanism that created a queue size limit for data packets, this new feature increased packet loss for data packets in both exercises, it also created a good buffer for VoIP packets, and it decreased the average packet delay for data packets because more packets of that type were being dropped.

Author's Contribution

Each member contributed equally to the development of this project, therefore, we decided to allocate 50% to each.

Acrónimos

BER Bit Error Rate

FIFO First In First Out

TT Transmitted Troughput