MINI-PROJECT 1:

PERFORMANCE EVALUATION OF POINT-TO-POINT LINKS SUPPORTING PACKET SERVICES

Universidade de Aveiro
Departamento de Eletrónica, Telecomunicações e Informática
Modelação e Desempenho de Redes e Serviços

Lúcia Sousa 93086, Rodrigo Martins 93264

Index

Task 1

Considering *Simulator1* presented in the practical classes to estimate the performance of a point-to-point IP link between a company router and its ISP.

Input parameters of *Simulator1*:

λ – packet rate (pps);

C – link capacity (Mbps);

f – queue size (Bytes);

P – total number of transmitted packets of a simulation run.

Performance parameters estimated by *Simulator1*:

PL – Packet Loss (%);

APD – Average Packet Delay (ms);

MPD – Maximum Packet Delay (ms);

TT – Transmitted Throughput (Mbps).

For this task, only the results of APD will be presented in bar charts.

The packet flow submitted to the link is characterized by: (i) an exponentially distributed time between packet arrivals with average 1/λ and (ii) a random packet size between 64 and 1518 bytes with the probabilities: 19% for 64 bytes, 23% for 110 bytes, 17% for 1518 bytes and an equal probability for all other values (i.e., from 65 to 109 and from 111 to 1517).

Exercise 1.a

Running *Simulator1* 50 times with the following input values, C = 10 Mbps, f = 1.000.000 Bytes, P = 10.000 packets and λ = 400, 800, 1200, 1600 and 2000 pps. The results obtained are presented in *Figure1* and *Table1*, with the respective associated errors.

It is possible to see that the higher the packet rate the higher the average packet delay. As mentioned above, the packet flow is characterized by an exponentially distributed time between packet arrivals with average 1/λ, which means, the more packets arriving per second, the shorter the time is between packet arrivals. However, more packets arrive in a second, for example 2000 packets arrive in a second, so there will be more packets to transmit, and therefore a longer delay, than if fewer packets arrive per second, for example if only 400 packets arrive in a second, fewer packets will be transmitted and therefore the delay will be shorter.

Packet Rate, λ (pps)	Average Packet Delay (ms)	Error
400	0.610685	±0.00176745
800	0.804863	±0.00367792
1200	1.18301	±0.0100787
1600	2.30704	±0.0398739
2000	26.7463	±3.92243

Table 1 - Average packet delay varying the packet rate and respective error.

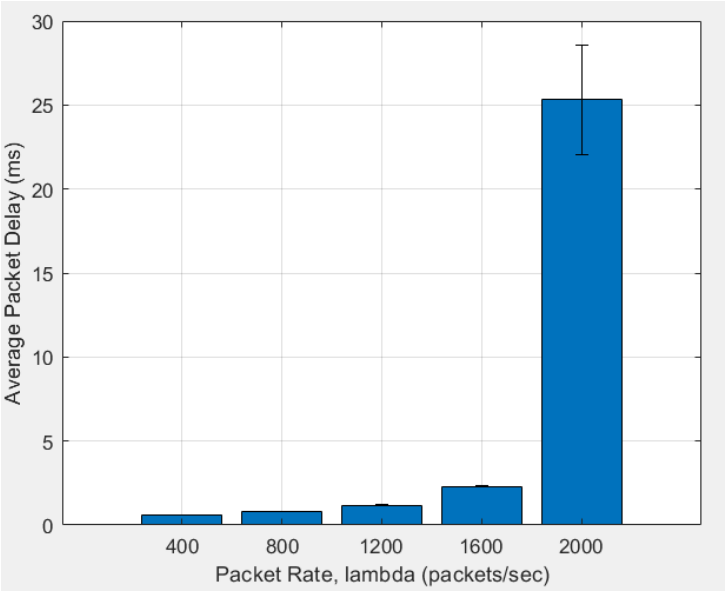

Figure 1 - Bar chart of the average packet delay varying the packet rate.

The code implemented for this exercise is presented as follows:

```
% Initializing variables

N = 50;
P = 10000;
conf = 0.9;
alfa=1-conf;
C = 10;
f = 1000000;

% Packet Rate = 400 pps
lambda1 = 400;
PacketLoss1 = zeros (1,N);
AvPacketDelay1 = zeros (1,N);
MaxPacketDelay1 = zeros (1,N);
Throughput1 = zeros (1,N);
% Run Simulator N times for lambda=400
for i=1:N
    [PacketLoss1(i), AvPacketDelay1(i), MaxPacketDelay1(i), Throughput1(i)] =
Simulator1(lambda1, C, f, P);
end
fprintf('Alinea a\n')
fprintf('\nlambda = 400\n')
```

```
media1=mean(AvPacketDelay1); % Calculation of the Average Packet Delay
term1=norminv(1-alfa/2)*sqrt(var(AvPacketDelay1)/N); % Calculation of the
error
fprintf('Av. Packet Delay (ms) = %.2e +- %.2e\n',media1,term1)

% Packet Rate = 800 pps
lambda2 = 800;
PacketLoss2 = zeros (1,N);
AvPacketDelay2 = zeros (1,N);
MaxPacketDelay2 = zeros (1,N);
Throughput2 = zeros (1,N);
% Run Simulator N times for lambda=800
for i=1:N
    [PacketLoss2(i), AvPacketDelay2(i), MaxPacketDelay2(i), Throughput2(i)] =
Simulator1(lambda2, C, f, P);
end
fprintf('\nlambda = 800\n')
media2=mean(AvPacketDelay2); % Calculation of the Average Packet Delay
term2=norminv(1-alfa/2)*sqrt(var(AvPacketDelay2)/N); % Calculation of the
error
fprintf('Av. Packet Delay (ms) = %.2e +- %.2e\n',media2,term2)

% Packet Rate = 1200 pps
lambda3 = 1200;
PacketLoss3 = zeros (1,N);
AvPacketDelay3 = zeros (1,N);
MaxPacketDelay3 = zeros (1,N);
Throughput3 = zeros (1,N);
% Run Simulator N times for lambda=1200
for i=1:N
    [PacketLoss3(i), AvPacketDelay3(i), MaxPacketDelay3(i), Throughput3(i)] =
Simulator1(lambda3, C, f, P);
end
fprintf('\nlambda = 1200\n')
media3=mean(AvPacketDelay3); % Calculation of the Average Packet Delay
term3=norminv(1-alfa/2)*sqrt(var(AvPacketDelay3)/N); % Calculation of the
error
fprintf('Av. Packet Delay (ms) = %.2e +- %.2e\n',media3,term3)

% Packet Rate = 1600 pps
lambda4 = 1600;
PacketLoss4 = zeros (1,N);
AvPacketDelay4 = zeros (1,N);
MaxPacketDelay4 = zeros (1,N);
Throughput4 = zeros (1,N);
% Run Simulator N times for lambda=1600
for i=1:N
    [PacketLoss4(i), AvPacketDelay4(i), MaxPacketDelay4(i), Throughput4(i)] =
Simulator1(lambda4, C, f, P);
end
fprintf('\nlambda = 1600\n')
media4=mean(AvPacketDelay4); % Calculation of the Average Packet Delay
term4=norminv(1-alfa/2)*sqrt(var(AvPacketDelay4)/N); % Calculation of the
error
fprintf('Av. Packet Delay (ms) = %.2e +- %.2e\n',media4,term4)

% Packet Rate = 2000 pps
lambda5 = 2000;
PacketLoss5 = zeros (1,N);
```

```
AvPacketDelay5 = zeros (1,N);
MaxPacketDelay5 = zeros (1,N);
Throughput5 = zeros (1,N);
% Run Simulator N times for lambda=2000
for i=1:N
    [PacketLoss5(i), AvPacketDelay5(i), MaxPacketDelay5(i), Throughput5(i)] =
Simulator1(lambda5, C, f, P);
end
fprintf('\nlambda = 2000\n')
media5=mean(AvPacketDelay5); % Calculation of the Average Packet Delay
term5=norminv(1-alfa/2)*sqrt(var(AvPacketDelay5)/N); % Calculation of the
error
fprintf('Av. Packet Delay (ms) = %.2e +- %.2e\n',media5,term5)

% Bar Graph
lambda = [lambda1 lambda2 lambda3 lambda4 lambda5];
avPacketDelay = [media1 media2 media3 media4 media5];
termhigh = [term1 term2 term3 term4 term5];
termlow = [-term1 -term2 -term3 -term4 -term5];
figure(1);
bar(lambda, avPacketDelay)
xlabel("Packet Rate, lambda (packets/sec)");
ylabel("Average Packet Delay (ms)");
title(" ");
grid("on");
hold on
er = errorbar(lambda, avPacketDelay, termlow, termhigh);
er.Color = [0 0 0];
er.LineStyle = 'none';
hold off
```

Exercise 1.b

Running *Simulator1* 50 times with the following input values, C = 10 Mbps, λ = 1800 pps, P = 10.000 packets and f = 100.000, 20.000, 10.000 and 2.000 Bytes. The results obtained are presented in *Figure2* and *Table2*, with the respective associated errors.

For smaller queue sizes the average packet delay is lower, this is because as packets arrive in a smaller queue size, the queue fills up faster and so the packets will have a smaller delay, because they will spend less time in the queue. On the other hand, packets arriving in a larger queue size may take longer in the queue, because the queue takes longer to fill and therefore has a longer delay.

Queue Size, f (Bytes)	Average Packet Delay (ms)	Error
2.000	0.954542	±0.00272
10.000	2.93743	±0.0353119
20.000	4.08258	±0.0926882
100.000	4.37957	±0.193863

Table 2 - Average packet delay varying the queue size and respective error.

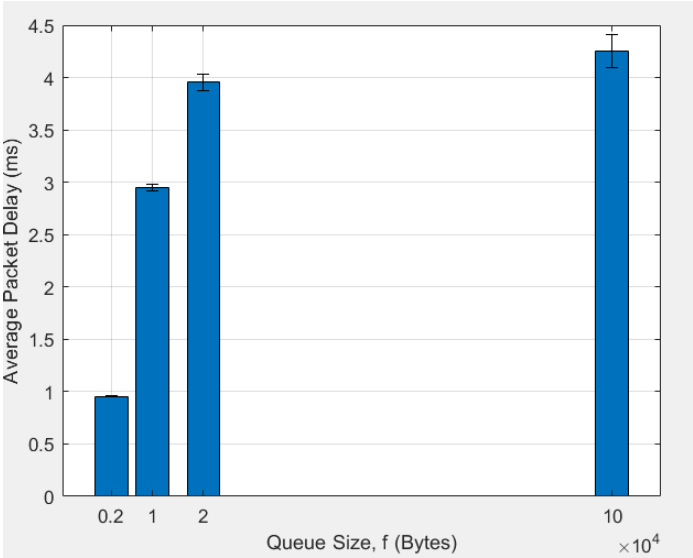

Figure 2 - Bar chart of the average packet delay varying the queue size.

The code implemented for this exercise is presented as follows:

```
% Initializing variables

N = 50;
P = 10000;
conf = 0.9;
alfa=1-conf;
C = 10;
lambda = 1800;

% Queue size = 100000 Bytes

f1 = 100000;
PacketLoss1 = zeros (1,N);
AvPacketDelay1 = zeros (1,N);
MaxPacketDelay1 = zeros (1,N);
Throughput1 = zeros (1,N);
% Run Simulator N times
for i=1:N
    [PacketLoss1(i), AvPacketDelay1(i), MaxPacketDelay1(i), Throughput1(i)] =
Simulator1(lambda, C, f1, P);
end
fprintf('Alinea b\n')
fprintf('\nf = 100000\n')
media1=mean(AvPacketDelay1); % Calculation of the Average Packet Delay
term1=norminv(1-alfa/2)*sqrt(var(AvPacketDelay1)/N); % Calculation of the
error
fprintf('Av. Packet Delay (ms) = %.2e +- %.2e\n',media1,term1)

% Queue size = 20000 Bytes

f2 = 20000;
PacketLoss2 = zeros (1,N);
AvPacketDelay2 = zeros (1,N);
MaxPacketDelay2 = zeros (1,N);
Throughput2 = zeros (1,N);
% Run Simulator N times
```

```
for i=1:N
    [PacketLoss2(i), AvPacketDelay2(i), MaxPacketDelay2(i), Throughput2(i)] =
Simulator1(lambda, C, f2, P);
end
fprintf('\nf = 20000\n')
media2=mean(AvPacketDelay2); % Calculation of the Average Packet Delay
term2=norminv(1-alfa/2)*sqrt(var(AvPacketDelay2)/N); % Calculation of the
error
fprintf('Av. Packet Delay (ms) = %.2e +- %.2e\n',media2,term2)

% Queue size = 10000 Bytes

f3 = 10000;
PacketLoss3 = zeros (1,N);
AvPacketDelay3 = zeros (1,N);
MaxPacketDelay3 = zeros (1,N);
Throughput3 = zeros (1,N);
% Run Simulator N times
for i=1:N
    [PacketLoss3(i), AvPacketDelay3(i), MaxPacketDelay3(i), Throughput3(i)] =
Simulator1(lambda, C, f3, P);
end
fprintf('\nf = 10000\n')
media3=mean(AvPacketDelay3); % Calculation of the Average Packet Delay
term3=norminv(1-alfa/2)*sqrt(var(AvPacketDelay3)/N); % Calculation of the
error
fprintf('Av. Packet Delay (ms) = %.2e +- %.2e\n',media3,term3)

% Queue size = 2000 Bytes

f4 = 2000;
PacketLoss4 = zeros (1,N);
AvPacketDelay4 = zeros (1,N);
MaxPacketDelay4 = zeros (1,N);
Throughput4 = zeros (1,N);
% Run Simulator N times
for i=1:N
    [PacketLoss4(i), AvPacketDelay4(i), MaxPacketDelay4(i), Throughput4(i)] =
Simulator1(lambda, C, f4, P);
end
fprintf('\nf = 2000\n')
media4=mean(AvPacketDelay4); % Calculation of the Average Packet Delay
term4=norminv(1-alfa/2)*sqrt(var(AvPacketDelay4)/N); % Calculation of the
error
fprintf('Av. Packet Delay (ms) = %.2e +- %.2e\n',media4,term4)

% Bar Chart
f = [f1 f2 f3 f4];
avPacketDelay = [media1 media2 media3 media4];
termhigh = [term1 term2 term3 term4];
termlow = [-term1 -term2 -term3 -term4];
figure(2);
bar(f, avPacketDelay)
xlabel("Queue Size, f (Bytes)");
ylabel("Average Packet Delay (ms)");
title(" ");
grid("on");
hold on
er = errorbar(f, avPacketDelay, termlow, termhigh);
er.Color = [0 0 0];
```

```
er.LineStyle = 'none';
hold off
```

Exercise 1.c

Running *Simulator1* 50 times with the following input values, λ = 1800 pps, f = 1.000.000 Bytes, P = 10.000 packets, and C = 10, 20, 30 and 40 Mbps. The results obtained are presented in *Figure3* and *Table3*, with the respective associated errors.

As the link capacity increases, the average packet delay decreases, because with a bigger capacity, more packets can be transmitted per second, and with a smaller capacity, less packets can be transmitted per second. So, the more packets are transmitted, the shorter the packet delay will be.

Capacity, C (Mbps)	Average Packet Delay (ms)	Error
10	4.31987	±0.153237
20	0.436335	±0.00231262
30	0.231325	±0.000907244
40	0.157986	±0.00045506

Table 3 - Average packet delay varying the capacity and respective error.

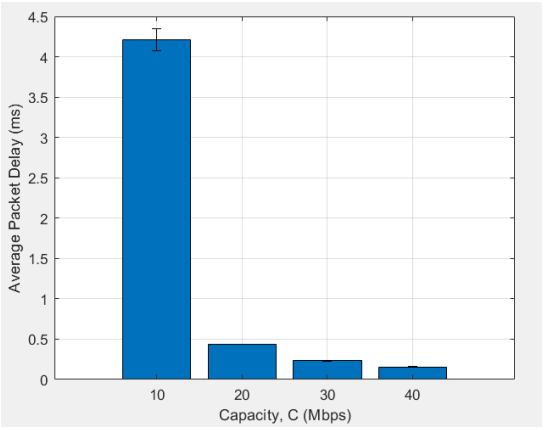

Figure 3 - Bar chart of the average packet delay varying the capacity.

The code implemented for this exercise is presented as follows:

```
% Initializing variables

N = 50;
P = 10000;
conf = 0.9;
alfa=1-conf;
lambda = 1800;
f = 1000000;

% Capacity = 10 Mbps
C1 = 10;
PacketLoss1 = zeros (1,N);
```

```
AvPacketDelay1 = zeros (1,N);
MaxPacketDelay1 = zeros (1,N);
Throughput1 = zeros (1,N);
% Run Simulator N times
for i=1:N
    [PacketLoss1(i), AvPacketDelay1(i), MaxPacketDelay1(i), Throughput1(i)] =
Simulator1(lambda, C1, f, P);
end
fprintf('Alinea c\n')
fprintf('\nC = 10\n')
media1=mean(AvPacketDelay1); % Calculation of the Average Packet Delay
term1=norminv(1-alfa/2)*sqrt(var(AvPacketDelay1)/N); % Calculation of the
error
fprintf('Av. Packet Delay (ms) = %.2e +- %.2e\n',media1,term1)

% Capacity = 20 Mbps
C2=20;
PacketLoss2 = zeros (1,N);
AvPacketDelay2 = zeros (1,N);
MaxPacketDelay2 = zeros (1,N);
Throughput2 = zeros (1,N);
% Run Simulator N times
for i=1:N
    [PacketLoss2(i), AvPacketDelay2(i), MaxPacketDelay2(i), Throughput2(i)] =
Simulator1(lambda, C2, f, P);
end
fprintf('\nC = 20\n')
media2=mean(AvPacketDelay2); % Calculation of the Average Packet Delay
term2=norminv(1-alfa/2)*sqrt(var(AvPacketDelay2)/N); % Calculation of the
error
fprintf('Av. Packet Delay (ms) = %.2e +- %.2e\n',media2,term2)

% Capacity = 30 Mbps
C3=30;
PacketLoss3 = zeros (1,N);
AvPacketDelay3 = zeros (1,N);
MaxPacketDelay3 = zeros (1,N);
Throughput3 = zeros (1,N);
% Run Simulator N times
for i=1:N
    [PacketLoss3(i), AvPacketDelay3(i), MaxPacketDelay3(i), Throughput3(i)] =
Simulator1(lambda, C3, f, P);
end
fprintf('\nC = 30\n')
media3=mean(AvPacketDelay3); % Calculation of the Average Packet Delay
term3=norminv(1-alfa/2)*sqrt(var(AvPacketDelay3)/N); % Calculation of the
error
fprintf('Av. Packet Delay (ms) = %.2e +- %.2e\n',media3,term3)

% Capacity = 40 Mbps
C4=40;
PacketLoss4 = zeros (1,N);
AvPacketDelay4 = zeros (1,N);
MaxPacketDelay4 = zeros (1,N);
Throughput4 = zeros (1,N);
% Run Simulator N times
for i=1:N
    [PacketLoss4(i), AvPacketDelay4(i), MaxPacketDelay4(i), Throughput4(i)] =
Simulator1(lambda, C4, f, P);
```

```
end
fprintf('\nC = 40\n')
media4=mean(AvPacketDelay4); % Calculation of the Average Packet Delay
term4=norminv(1-alfa/2)*sqrt(var(AvPacketDelay4)/N); % Calculation of the
error
fprintf('Av. Packet Delay (ms) = %.2e +- %.2e\n',media4,term4)

% Bar Chart
C = [C1 C2 C3 C4];
avPacketDelay = [media1 media2 media3 media4];
termhigh = [term1 term2 term3 term4];
termlow = [-term1 -term2 -term3 -term4];
figure(3);
bar(C, avPacketDelay)
xlabel("Capacity, C (Mbps)");
ylabel("Average Packet Delay (ms)");
title(" ");
grid("on");
hold on
er = errorbar(C, avPacketDelay, termlow, termhigh);
er.Color = [0 0 0];
er.LineStyle = 'none';
hold off
```

Exercise 1.d

Considering that the system is modelled by a M/G/1 queueing model. The theoretical values of the average packet delay determined are in *Table4* and *Figure4*. The values in *Table3* are the average packet delay results from the simulation of *Exercise 1.c*.

The average theoretical packet delay is approximately equal to the average packet delay simulated in the previous exercise.

Capacity, C (Mbps)	Average Packet Delay (ms)
10	4.3883
20	0.4364
30	0.2313
40	0.1576

Table 4 - Theorical values of the average packet delay.

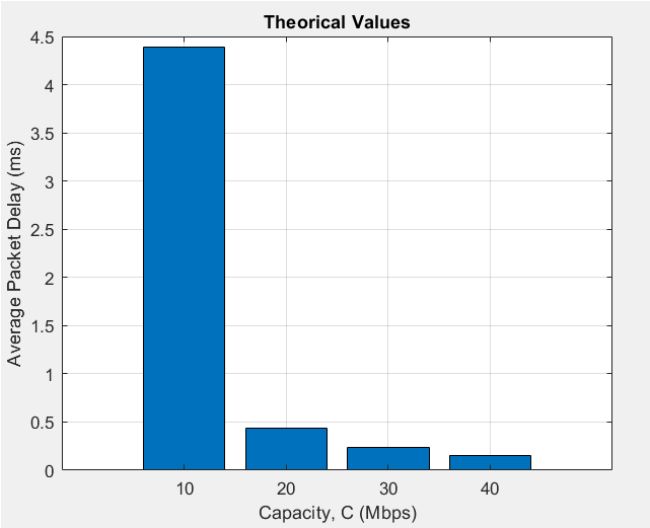

Figure 4 - Bar chart of the average packet delay varying the capacity.

The code implemented for this exercise is presented as follows:

```
% Initializing variables

N = 50;
P = 10000;
conf = 0.9;
alfa=1-conf;
lambda = 1800;
f = 1000000;
packetSize = 64:1518;
prob = zeros(1,1518);
prob(packetSize) = (1 - 0.19 - 0.23 -0.17) / (length(packetSize)-3);
prob(64) = 0.19;
prob(110) = 0.23;
prob(1518) = 0.17;
avgPacketSize = sum(prob(packetSize).*packetSize);

fprintf('\nAlinea d\n')

% Capacity = 10 Mbps
C1 = 10e6;
fprintf('\nC = 10\n')
S = ((packetSize.*8)./C1);
S_2 = S.^2;
E = sum(prob(packetSize).*S);
E_2 = sum(prob(packetSize).*S_2);
i=1:N;
u = C1/(avgPacketSize*8);
avgPacketDelay1 = (((lambda.*E_2) / (2*(1-lambda.*E))) + E) * 10^3;
fprintf('Av. Packet Delay = %.4f\n',avgPacketDelay1);

% Capacity = 20 Mbps
C2 = 20e6;
fprintf('\nC = 20\n')
S = ((packetSize.*8)./C2);
S_2 = S.^2;
E = sum(prob(packetSize).*S);
```

```
E_2 = sum(prob(packetSize).*S_2);
i=1:N;
u = C2/(avgPacketSize*8);
avgPacketDelay2 = (((lambda.*E_2) / (2*(1-lambda.*E))) + E) * 10^3;
fprintf('Av. Packet Delay = %.4f\n',avgPacketDelay2);

% Capacity = 30 Mbps
C3 = 30e6;
fprintf('\nC = 30\n')
S = ((packetSize.*8)./C3);
S_2 = S.^2;
E = sum(prob(packetSize).*S);
E_2 = sum(prob(packetSize).*S_2);
i=1:N;
u = C3/(avgPacketSize*8);
avgPacketDelay3 = (((lambda.*E_2) / (2*(1-lambda.*E))) + E) * 10^3;
fprintf('Av. Packet Delay = %.4f\n',avgPacketDelay3);

% Capacity = 40 Mbps
C4 = 40e6;
fprintf('\nC = 40\n')
S = ((packetSize.*8)./C4);
S_2 = S.^2;
E = sum(prob(packetSize).*S);
E_2 = sum(prob(packetSize).*S_2);
u = C4/(avgPacketSize*8);
avgPacketDelay4 = (((lambda.*E_2) / (2*(1-lambda.*E))) + E) * 10^3;
fprintf('Av. Packet Delay = %.4f\n',avgPacketDelay4);

% Bar Chart
avPacketDelay = [avgPacketDelay1 avgPacketDelay2 avgPacketDelay3
avgPacketDelay4];
C=[10 20 30 40];
figure(4);
bar(C, avPacketDelay)
xlabel("Capacity, C (Mbps)");
ylabel("Average Packet Delay (ms)");
title("Theorical Values");
grid("on");
```

Exercise 1.e

A new version of *Simulator1* was developed to estimate 3 additional performance parameters, the average packet delay of the packets of size 64, 110 and 1518 Bytes.

The first change was then to add three new outputs, average packet delay for each packet size, 64, 110 and 1518 Bytes. Then new variables were created, necessary to calculate the APD, these variables are TRANSMITTEDPACKETS and DELAYS for all the 3 different packet sizes. Since each event list contains the size of the packet, when it is an ARRIVAL, the size is checked and for each one the variable of the corresponding transmitted packets is incremented. When it is a DEPARTURE, the size is checked and for each one the variable of the corresponding delay is calculated. These changes are represented in bold in the following code section.

```
function [PL, APD_64, APD_110, APD_1518, MPD , TT] =
Simulator1_newVersion(lambda,C,f,P)
% INPUT PARAMETERS:
%  lambda - packet rate (packets/sec)
%  C      - link bandwidth (Mbps)
%  f      - queue size (Bytes)
%  P      - number of packets (stopping criterium)
% OUTPUT PARAMETERS:
%  PL   - packet loss (%)
%  APD_64  - average packet delay for packet size 64 (milliseconds)
%  APD_110  - average packet delay for packet size 110 (milliseconds)
%  APD_1518  - average packet delay for packet size 1518 (milliseconds)
%  MPD  - maximum packet delay (milliseconds)
%  TT   - transmitted throughput (Mbps)

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

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


%Statistical Counters:
TOTALPACKETS= 0;      % No. of packets arrived to the system
LOSTPACKETS= 0;       % No. of packets dropped due to buffer overflow
TRANSMITTEDPACKETS= 0;      % No. of transmitted packets
TRANSMITTEDPACKETS64= 0;    % No. of transmitted packets size 64
TRANSMITTEDPACKETS110= 0;   % No. of transmitted packets size 110
TRANSMITTEDPACKETS1518= 0;  % No. of transmitted packets size 64
TRANSMITTEDBYTES= 0;        % Sum of the Bytes of transmitted packets
DELAYS= 0;                  % Sum of the delays of transmitted packets
DELAYS64= 0;   % Sum of the delays of transmitted packets size 64
DELAYS110= 0;  % Sum of the delays of transmitted packets size 110
DELAYS1518= 0; % Sum of the delays of transmitted packets size 1518
MAXDELAY= 0;   % Maximum delay among all transmitted 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];

%Similation loop:
while TRANSMITTEDPACKETS<P               % Stopping criterium
    EventList= sortrows(EventList,2);    % Order EventList by time
    Event= EventList(1,1);               % Get first event and
    Clock= EventList(1,2);               %   and
    PacketSize= EventList(1,3);          %   associated
    ArrivalInstant= EventList(1,4);      %   parameters.
    EventList(1,:)= [];                  % Eliminate first event
      switch Event
         case ARRIVAL      % If first event is an ARRIVAL
           if PacketSize == 64
             TRANSMITTEDPACKETS64 = TRANSMITTEDPACKETS64 + 1;
           elseif PacketSize == 110
```

```
            TRANSMITTEDPACKETS110 = TRANSMITTEDPACKETS110 + 1;
          else
            TRANSMITTEDPACKETS1518 = TRANSMITTEDPACKETS1518 + 1;
          end
          TOTALPACKETS= TOTALPACKETS+1;
          tmp= Clock + exprnd(1/lambda);
          EventList = [EventList; ARRIVAL, tmp, GeneratePacketSize(), tmp];
          if STATE==0
            STATE= 1;
            EventList = [EventList; DEPARTURE, Clock +
      8*PacketSize/(C*10^6), PacketSize, Clock];
          else
            if QUEUEOCCUPATION + PacketSize <= f
             QUEUE= [QUEUE;PacketSize , Clock];
             QUEUEOCCUPATION= QUEUEOCCUPATION + PacketSize;
            else
              LOSTPACKETS= LOSTPACKETS + 1;
            end
          end
        case DEPARTURE             % If first event is a DEPARTURE
            if PacketSize == 64
              DELAYS64= DELAYS64 + (Clock - ArrivalInstant);
            elseif PacketSize == 110
              DELAYS110= DELAYS110 + (Clock - ArrivalInstant);
            else
              DELAYS1518= DELAYS1518 + (Clock - ArrivalInstant);
            end
            TRANSMITTEDBYTES= TRANSMITTEDBYTES + PacketSize;
            DELAYS= DELAYS + (Clock - ArrivalInstant);
            if Clock - ArrivalInstant > MAXDELAY
              MAXDELAY= Clock - ArrivalInstant;
            end
              TRANSMITTEDPACKETS= TRANSMITTEDPACKETS + 1;
            if QUEUEOCCUPATION > 0
               EventList = [EventList; DEPARTURE, Clock +
      8*QUEUE(1,1)/(C*10^6), QUEUE(1,1), QUEUE(1,2)];
              QUEUEOCCUPATION= QUEUEOCCUPATION - QUEUE(1,1);
              QUEUE(1,:)= [];
             else
              STATE= 0;
             end
        end
      end

%Performance parameters determination:
PL= 100*LOSTPACKETS/TOTALPACKETS;                    % in %
APD_64= 1000*DELAYS64/TRANSMITTEDPACKETS64;          % in milliseconds
APD_110= 1000*DELAYS110/TRANSMITTEDPACKETS110;
APD_1518= 1000*DELAYS1518/TRANSMITTEDPACKETS1518;
MPD= 1000*MAXDELAY;                                  % in milliseconds
TT= 10^(-6)*TRANSMITTEDBYTES*8/Clock;                % in Mbps
```

The results obtained are shown below for packet sizes 64, 110 and 1518 Bytes in *Figure5*, *Figure6* and *Figure7* respectively, as well as in *Table5*, *Table6* and *Table7*.

Comparing the results with the packet sizes, it is possible to see that they are not too dispersed from each other.

Capacity, C (Mbps)	Average Packet Delay (ms)	Error
10	3.84948	±0.153237
20	0.213628	±0.001986
50	0.0303404	±0.000364009
100	0.00974	±0.0000830

Table 5 - Average packet delay for packet size 64 Bytes, varying the capacity and respective error.

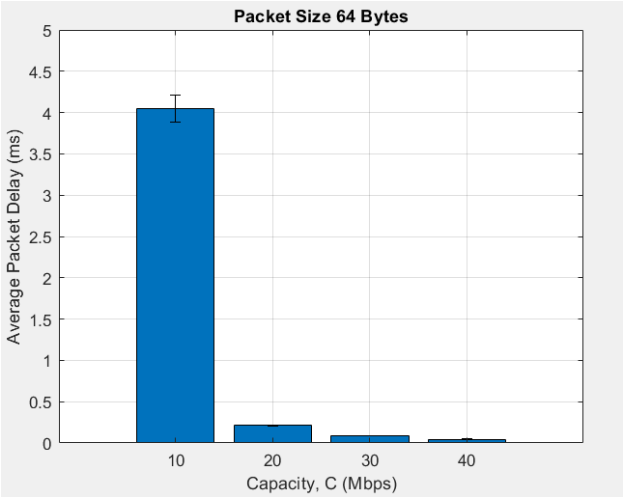

Figure 5 - Bar chart of the average packet delay for packet size of 64 bytes varying the capacity.

Capacity, C (Mbps)	Average Packet Delay (ms)	Error
10	3.89002	±0.148345
20	0.233099	±0.00211213
50	0.037948	±0.000341383
100	0.0133745	±0.00009185

Table 6 - Average packet delay for packet size 110 Bytes, varying the capacity and respective error.

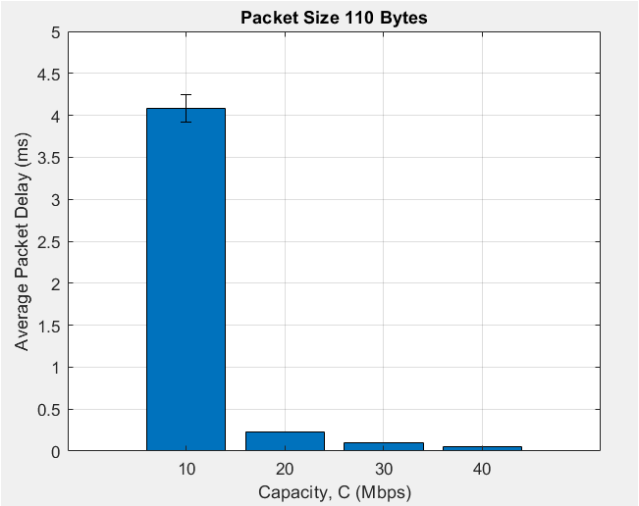

Figure 6 - Bar chart of the average packet delay for packet size of 110 bytes varying the capacity.

Capacity, C (Mbps)	Average Packet Delay (ms)	Error
10	4.59235	±0.152255
20	0.588574	±0.00236676
50	0.180747	±0.000351405
100	0.0848199	±0.000143254

Table 7 - Average packet delay for packet size 1518 Bytes, varying the capacity and respective error.

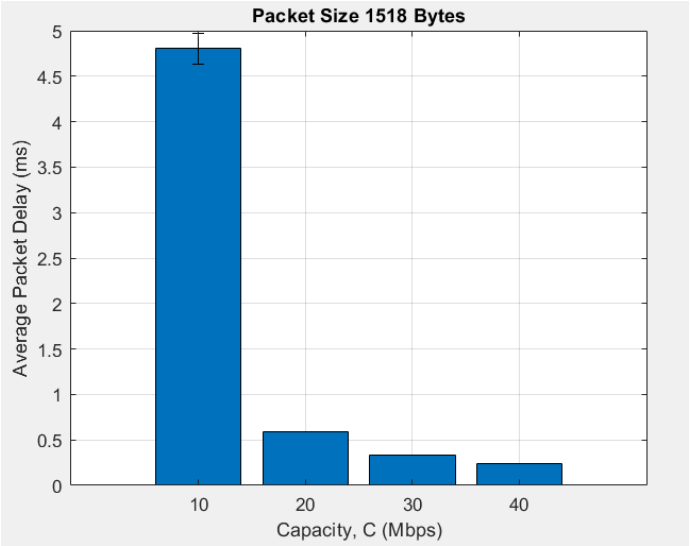

Figure 7 - Bar chart of the average packet delay for packet size of 1518 bytes varying the capacity.

The code implemented for this exercise is presented as follows:

```
% Initializing variables

N = 50;
P = 10000;
conf = 0.9;
alfa=1-conf;
lambda = 1800;
f = 1000000;

% Capacity = 10 Mbps
C1 = 10;
PacketLoss1 = zeros (1,N);
AvPacketDelay1_64 = zeros (1,N);
AvPacketDelay1_110 = zeros (1,N);
AvPacketDelay1_1518 = zeros (1,N);
MaxPacketDelay1 = zeros (1,N);
Throughput1 = zeros (1,N);
% Run Simulator N times
for i=1:N
    [PacketLoss1(i), AvPacketDelay1_64(i),AvPacketDelay1_110(i),
      AvPacketDelay1_1518(i), MaxPacketDelay1(i), Throughput1(i)] =
Simulator1_newVersion(lambda, C1, f, P);
end
fprintf('Alinea e\n')
fprintf('\nC = 10\n')
% Calculation of the Average Packet Delay for size 64
```

```
media1_64=mean(AvPacketDelay1_64);
% Calculation of the error for size 64
term1_64=norminv(1-alfa/2)*sqrt(var(AvPacketDelay1_64)/N);
fprintf('Av. Packet Delay size 64 (ms) = %.2e +- %.2e\n',media1_64,term1_64)
% Calculation of the Average Packet Delay for size 110
media1_110=mean(AvPacketDelay1_110);
% Calculation of the error for size 110
term1_110=norminv(1-alfa/2)*sqrt(var(AvPacketDelay1_110)/N);
fprintf('Av. Packet Delay size 110 (ms) = %.2e +-
%.2e\n',media1_110,term1_110)
% Calculation of the Average Packet Delay for size 1518
media1_1518=mean(AvPacketDelay1_1518);
% Calculation of the error for size 1518
term1_1518=norminv(1-alfa/2)*sqrt(var(AvPacketDelay1_1518)/N);
fprintf('Av. Packet Delay size 1518 (ms) = %.2e +-
%.2e\n',media1_1518,term1_1518)

% Capacity = 20 Mbps
C2=20;
PacketLoss2 = zeros (1,N);
AvPacketDelay2_64 = zeros (1,N);
AvPacketDelay2_110 = zeros (1,N);
AvPacketDelay2_1518 = zeros (1,N);
MaxPacketDelay2 = zeros (1,N);
Throughput2 = zeros (1,N);
% Run Simulator N times
for i=1:N
    [PacketLoss2(i), AvPacketDelay2_64(i), AvPacketDelay2_110(i),
AvPacketDelay2_1518(i), MaxPacketDelay2(i), Throughput2(i)] =
Simulator1_newVersion(lambda, C2, f, P);
end
fprintf('\nC = 20\n')
% Calculation of the Average Packet Delay for size 64
media2_64=mean(AvPacketDelay2_64);
% Calculation of the error for size 64
term2_64=norminv(1-alfa/2)*sqrt(var(AvPacketDelay2_64)/N);
fprintf('Av. Packet Delay size 64 (ms) = %.2e +- %.2e\n',media2_64,term2_64)
% Calculation of the Average Packet Delay for size 110
media2_110=mean(AvPacketDelay2_110);
% Calculation of the error for size 110
term2_110=norminv(1-alfa/2)*sqrt(var(AvPacketDelay2_110)/N);
fprintf('Av. Packet Delay size 110 (ms) = %.2e +-
%.2e\n',media2_110,term2_110)
% Calculation of the Average Packet Delay for size 1518
media2_1518=mean(AvPacketDelay2_1518);
% Calculation of the error for size 1518
term2_1518=norminv(1-alfa/2)*sqrt(var(AvPacketDelay2_1518)/N);
fprintf('Av. Packet Delay size 1518 (ms) = %.2e +-
%.2e\n',media2_1518,term2_1518)

% Capacity = 30 Mbps
C3=30;
PacketLoss3 = zeros (1,N);
AvPacketDelay3_64 = zeros (1,N);
AvPacketDelay3_110 = zeros (1,N);
AvPacketDelay3_1518 = zeros (1,N);
MaxPacketDelay3 = zeros (1,N);
Throughput3 = zeros (1,N);
% Run Simulator N times
```

```
for i=1:N
    [PacketLoss3(i), AvPacketDelay3_64(i), AvPacketDelay3_110(i),
AvPacketDelay3_1518(i), MaxPacketDelay3(i), Throughput3(i)] =
Simulator1_newVersion(lambda, C3, f, P);
end
fprintf('\nC = 30\n')
% Calculation of the Average Packet Delay for size 64
media3_64=mean(AvPacketDelay3_64);
% Calculation of the error for size 64
term3_64=norminv(1-alfa/2)*sqrt(var(AvPacketDelay3_64)/N);
fprintf('Av. Packet Delay size 64 (ms) = %.2e +- %.2e\n',media3_64,term3_64)
% Calculation of the Average Packet Delay for size 110
media3_110=mean(AvPacketDelay3_110);
% Calculation of the error for size 110
term3_110=norminv(1-alfa/2)*sqrt(var(AvPacketDelay3_110)/N);
fprintf('Av. Packet Delay size 110 (ms) = %.2e +-
%.2e\n',media3_110,term3_110)
% Calculation of the Average Packet Delay for size 1518
media3_1518=mean(AvPacketDelay3_1518);
% Calculation of the error for size 1518
term3_1518=norminv(1-alfa/2)*sqrt(var(AvPacketDelay3_1518)/N);
fprintf('Av. Packet Delay size 1518 (ms) = %.2e +-
%.2e\n',media3_1518,term3_1518)

% Capacity = 40 Mbps
C4=40;
PacketLoss4 = zeros (1,N);
AvPacketDelay4_64 = zeros (1,N);
AvPacketDelay4_110 = zeros (1,N);
AvPacketDelay4_1518 = zeros (1,N);
MaxPacketDelay4 = zeros (1,N);
Throughput4 = zeros (1,N);
% Run Simulator N times
for i=1:N
    [PacketLoss4(i), AvPacketDelay4_64(i), AvPacketDelay4_110(i),
AvPacketDelay4_1518(i), MaxPacketDelay4(i), Throughput4(i)] =
Simulator1_newVersion(lambda, C4, f, P);
end
fprintf('\nC = 40\n')
% Calculation of the Average Packet Delay for size 64
media4_64=mean(AvPacketDelay4_64);
% Calculation of the error for size 64
term4_64=norminv(1-alfa/2)*sqrt(var(AvPacketDelay4_64)/N);
fprintf('Av. Packet Delay size 64 (ms) = %.2e +- %.2e\n',media4_64,term4_64)
% Calculation of the Average Packet Delay for size 110
media4_110=mean(AvPacketDelay4_110);
% Calculation of the error for size 110
term4_110=norminv(1-alfa/2)*sqrt(var(AvPacketDelay4_110)/N);
fprintf('Av. Packet Delay size 110 (ms) = %.2e +-
%.2e\n',media4_110,term4_110)
% Calculation of the Average Packet Delay for size 1518
media4_1518=mean(AvPacketDelay4_1518);
% Calculation of the error for size 1518
term4_1518=norminv(1-alfa/2)*sqrt(var(AvPacketDelay4_1518)/N);
fprintf('Av. Packet Delay size 1518 (ms) = %.2e +-
%.2e\n',media4_1518,term4_1518)

% Bar Chart for Packet Size = 64
C = [C1 C2 C3 C4];
```

```
avPacketDelay_64 = [media1_64 media2_64 media3_64 media4_64];
termhigh_64 = [term1_64 term2_64 term3_64 term4_64];
termlow_64 = [-term1_64 -term2_64 -term3_64 -term4_64];
figure(5);
bar(C, avPacketDelay_64)
xlabel("Capacity, C (Mbps)");
ylabel("Average Packet Delay (ms)");
ylim([0 5]);
title("Packet Size 64 Bytes");
grid("on");
hold on
er = errorbar(C, avPacketDelay_64, termlow_64, termhigh_64);
er.Color = [0 0 0];
er.LineStyle = 'none';
hold off

% Bar Chart for Packet Size = 110
avPacketDelay_110 = [media1_110 media2_110 media3_110 media4_110];
termhigh_110 = [term1_110 term2_110 term3_110 term4_110];
termlow_110 = [-term1_110 -term2_110 -term3_110 -term4_110];
figure(6);
bar(C, avPacketDelay_110)
xlabel("Capacity, C (Mbps)");
ylabel("Average Packet Delay (ms)");
ylim([0 5]);
title("Packet Size 110 Bytes");
grid("on");
hold on
er = errorbar(C, avPacketDelay_110, termlow_110, termhigh_110);
er.Color = [0 0 0];
er.LineStyle = 'none';
hold off

% Bar Chart for Packet Size = 1518
avPacketDelay_1518 = [media1_1518 media2_1518 media3_1518 media4_1518];
termhigh_1518 = [term1_1518 term2_1518 term3_1518 term4_1518];
termlow_1518 = [-term1_1518 -term2_1518 -term3_1518 -term4_1518];
figure(7);
bar(C, avPacketDelay_1518)
xlabel("Capacity, C (Mbps)");
ylabel("Average Packet Delay (ms)");
ylim([0 5]);
title("Packet Size 1518 Bytes");
grid("on");
hold on
er = errorbar(C, avPacketDelay_1518, termlow_1518, termhigh_1518);
er.Color = [0 0 0];
er.LineStyle = 'none';
hold off
```

Task 2

Considering *Simulator3 and Simulator4* presented in the practical classes to estimate the performance of a point-to-point IP link between a company router and its ISP.

Input parameters of *Simulator3* and *Simulator4*:

λ – packet rate (pps);

C – link capacity (Mbps);

f – queue size (Bytes);

P – total number of transmitted packets of a simulation run;

n – number of VoIP packets;

Performance parameters estimated by *Simulator3* and *Simulator4*:

PLdata – Packet loss of data packets (%);

PLvoip – Packet loss of VoIP packets (%);

APDdata – Average packet delay of data packets (milliseconds);

APDvoip – Average packet delay of VoIP packets (milliseconds);

MPDdata – Maximum packet delay of data (milliseconds);

MPDvoip – Maximum packet delay of VoIP (milliseconds);

TT – Transmitted throughput (Mbps).

Queue type of *Simulator3*: FIFO

Queue type of *Simulator4*: Priority Queue

The packet flow submitted to the link is characterized by: (i) an exponentially distributed time between data packet arrivals with average 1/λ and (ii) a random data packet size between 64 and 1518 bytes with the probabilities: 19% for 64 bytes, 23% for 110 bytes, 17% for 1518 bytes and an equal probability for all other values (i.e., from 65 to 109 and from 111 to 1517) (iii) a random VoIP packet size uniformly distributed between 110 and 130 bytes (iv) the time between VoIP packet arrivals is uniformly distributed between 16 and 24 milliseconds.

Exercise 2.a

Running *Simulator3* 50 times with the following input values, C = 10 Mbps, f = 1.000.000 Bytes, P = 10.000 packets, λ = 1500 pps and n = 10, 20, 30, 40. The results obtained for the data packets are presented in *Figure8* and *Table8*, with the respective associated errors and the results for the VoIP packets are present in *Figure9* and *Table9, with the respective associated errors*.

It is possible to see that the higher the number of VoIP packets the higher the Average Packet Delay will be, both for data and VoIP packets, this effect comes from the fact that the more packets that must be transmitted, the longer will be the delay between their transmission. The average packet delay increases for both data and VoIP packets because the *Simulator3* uses a FIFO queue, this means that the data and VoIP packets have the same priority, which indicates that the number of packets in the queue will affect both packet types.

n	Average Data Packet Delay (ms)	Error
10	2.17	±0.00301
20	2.64	±0.00541
30	3.51	±0.00825
40	5.82	±0.0289

Table 8 - Average data packet delay varying the number of VoIp packets

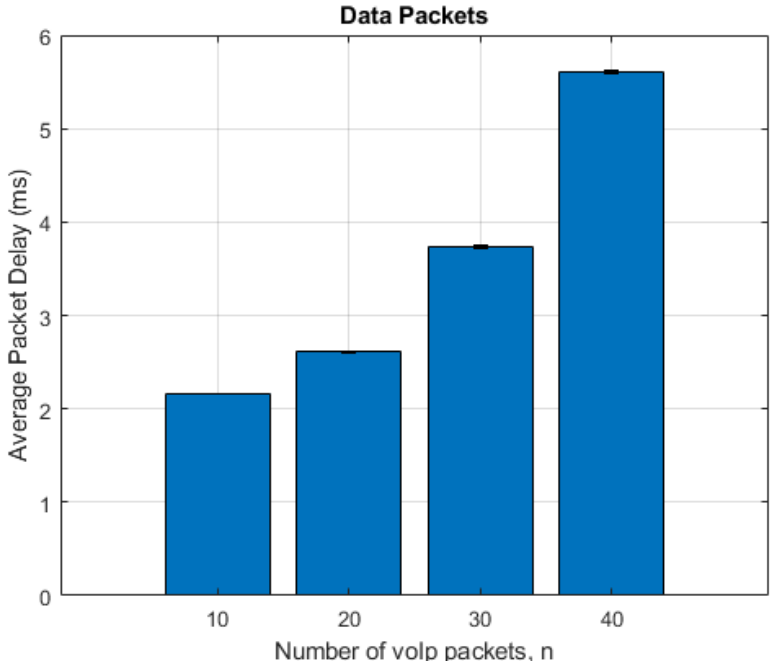

Figure 8 - Bar chart of the average data packet delay varying the number of VoIp packets

n	Average VoIP Packet Delay (ms)	Error
10	1.75	±0.00276
20	2.22	±0.00499
30	3.09	±0.00769
40	5.41	±0.0285

Table 9 - Average voIp packet delay varying the number of voIp packets

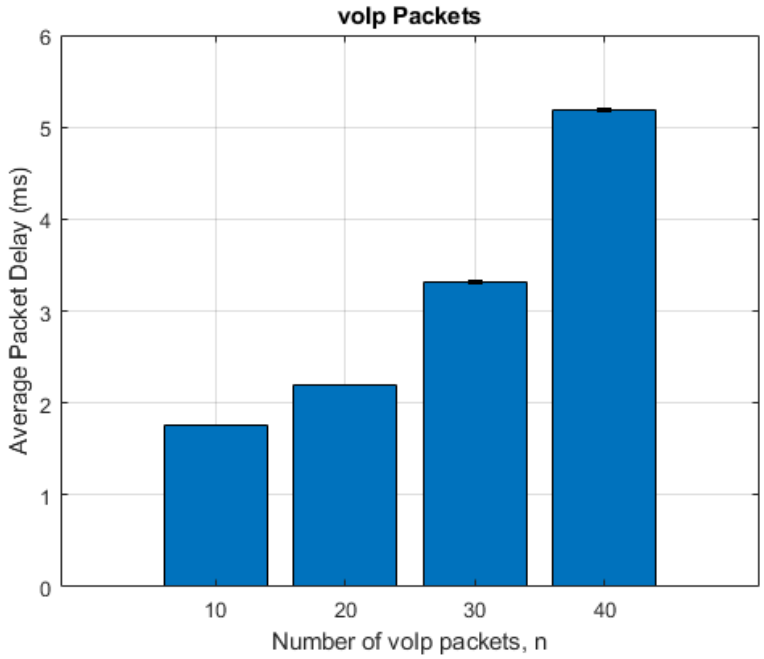

Figure 9 - Bar chart of the average voIp packet delay varying the number of VoIp packets

The code used to obtain these results is shown below:

```
% Initializing variables

P = 10000;
conf = 0.9;
lambda = 1500;
C = 10;
f = 1e6;
N = 50;
n = 10;


PacketLossdata = zeros(1,N);
PacketLossvoip = zeros(1,N);
AvgPacketDelaydata = zeros(1,N);
AvgPacketDelayvoip = zeros(1,N);
MaxPacketDelaydata = zeros(1,N);
MaxPacketDelayvoip = zeros(1,N);
Throughput = zeros(1,N);
```

```
for i=1:N

[PacketLossdata(i),PacketLossvoip(i),AvgPacketDelaydata(i),AvgPacketDelayvoip
(i),MaxPacketDelaydata(i),MaxPacketDelayvoip(i),Throughput(i)] =
Simulator3(lambda,C,f,P,n);
end


mAPDd = mean(AvgPacketDelaydata);
mAPDv = mean(AvgPacketDelayvoip);
term2d = norminv(1-conf/2)*sqrt(var(AvgPacketDelaydata)/N);
term2v = norminv(1-conf/2)*sqrt(var(AvgPacketDelayvoip)/N);


% Array with all average data packet delay

R1d = [mAPDd];

% Array with all average data packet delay errors

R2d = [term2d];

% Array with all average VoIP packet delay

R1v = [mAPDv];

% Array with all average VoIP packet delay errors

R2v = [term2v];

fprintf('Alinea A, n = 10')
fprintf(['\nAvgPacketDelay data= %.2e +- %.2e' ...
    '\nAvgPacketDelay voip= %.2e +- %.2e'], ...
    mAPDd, term2d, mAPDv, term2v)
fprintf('\n\n=========================================================\n\n')


% ------------------------------------------------------------ %
n = 20;


for i=1:N

[PacketLossdata(i),PacketLossvoip(i),AvgPacketDelaydata(i),AvgPacketDelayvoip
(i),MaxPacketDelaydata(i),MaxPacketDelayvoip(i),Throughput(i)] =
Simulator3(lambda,C,f,P,n);
end


mAPDd = mean(AvgPacketDelaydata);
mAPDv = mean(AvgPacketDelayvoip);
term2d = norminv(1-conf/2)*sqrt(var(AvgPacketDelaydata)/N);
term2v = norminv(1-conf/2)*sqrt(var(AvgPacketDelayvoip)/N);
```

```
% Array with all average data packet delay

R1d = [mAPDd];

% Array with all average data packet delay errors

R2d = [term2d];

% Array with all average VoIP packet delay

R1v = [mAPDv];

% Array with all average VoIP packet delay errors

R2v = [term2v];


fprintf('Alinea A, n = 20')
fprintf(['\nAvgPacketDelay data= %.2e +- %.2e' ...
    '\nAvgPacketDelay voip= %.2e +- %.2e'], ...
    mAPDd, term2d, mAPDv, term2v)
fprintf('\n\n=============================================================\n\n')


% ------------------------------------------------------------- %
n = 30;


for i=1:N

[PacketLossdata(i),PacketLossvoip(i),AvgPacketDelaydata(i),AvgPacketDelayvoip
(i),MaxPacketDelaydata(i),MaxPacketDelayvoip(i),Throughput(i)] =
Simulator3(lambda,C,f,P,n);
end


mAPDd = mean(AvgPacketDelaydata);
mAPDv = mean(AvgPacketDelayvoip);
term2d = norminv(1-conf/2)*sqrt(var(AvgPacketDelaydata)/N);
term2v = norminv(1-conf/2)*sqrt(var(AvgPacketDelayvoip)/N);


% Array with all average data packet delay

R1d = [mAPDd];

% Array with all average data packet delay errors


R2d = [term2d];


% Array with all average VoIP packet delay

R1v = [mAPDv];
```

```
% Array with all average VoIP packet delay errors


R2v = [term2v];


fprintf('Alinea A, n = 30')
fprintf(['\nAvgPacketDelay data= %.2e +- %.2e' ...
    '\nAvgPacketDelay voip= %.2e +- %.2e'], ...
    mAPDd, term2d, mAPDv, term2v)
fprintf('\n\n============================================================\n\n')

% ------------------------------------------------------------- %
n = 40;


for i=1:N

[PacketLossdata(i),PacketLossvoip(i),AvgPacketDelaydata(i),AvgPacketDelayvoip
(i),MaxPacketDelaydata(i),MaxPacketDelayvoip(i),Throughput(i)] =
Simulator3(lambda,C,f,P,n);
end


mAPDd = mean(AvgPacketDelaydata);
mAPDv = mean(AvgPacketDelayvoip);
term2d = norminv(1-conf/2)*sqrt(var(AvgPacketDelaydata)/N);
term2v = norminv(1-conf/2)*sqrt(var(AvgPacketDelayvoip)/N);


% Array with all average data packet delay

R1d = [mAPDd];

% Array with all average data packet delay errors

R2d = [term2d];

% Array with all average VoIP packet delay

R1v = [mAPDv];

% Array with all average VoIP packet delay errors

R2v = [term2v];

n = [10 20 30 40];


% Bar chart of average data packet delay

bar(n, R1d)
hold on
```

```
er = errorbar(n, R1d, R2d, R2d);
er.Color = [0 0 0];
er.LineStyle = 'none';
hold off
grid on
xlabel('Number of voIp packets, n')
ylabel('Average Packet Delay (ms)')
title('Data Packets')


% Bar chart of average VoIP packet delay

bar(n, R1v);

hold on

grid on
xlabel('Number of voIp packets, n')
ylabel('Average Packet Delay (ms)')
title('voIp Packets')
er = errorbar(n, R1v, R2v, R2v);
er.Color = [0 0 0];
er.LineStyle = 'none';
hold off


fprintf('Alinea A, n = 40')
fprintf(['\nAvgPacketDelay data= %.2e +- %.2e' ...
    '\nAvgPacketDelay voip= %.2e +- %.2e'], ...
    mAPDd, term2d, mAPDv, term2v)
fprintf('\n\n=========================================================\n\n')
```

Exercise 2.b

Running *Simulator4* 50 times with the following input values, C = 10 Mbps, f = 1.000.000 Bytes, P = 10.000 packets, λ = 1500 pps and n = 10, 20, 30, 40. The results obtained for the data packets are presented in *Figure10* and *Table10*, with the respective associated errors and the results for the VoIP packets are present in *Figure11* and *Table11*, with the respective associated errors.

It is possible to see that the higher the number of VoIP packets the higher the Average Packet Delay will be, both for data and VoIP packets, this effect comes from the fact that the more packets that must be transmitted, the longer will be the delay between their transmission. A major difference can be seen in the average VoIP packet delay by comparing the results from exercise 2.a and exercise 2.b and that is that the average VoIP packet delay in ex. 2.b is much lower than in ex. 2a and the average Data packet delay in ex. 2.b is higher than ex. 2a, this can be explained by the different types of queues, in ex. 2.a the queue is a FIFO and in ex. 2.b is a priority queue which means that in this queue the packets with higher priority, in this case the VoIP packets, are transmitted first even if they reach the queue after other lower priority

packets, in this case the Data packets, therefore lowering the Average packet delay for the higher priority packets.

n	Average Data Packet Delay (ms)	Error
10	2.27	±0.00324
20	2.85	±0.006
30	4.19	±0.0130
40	6.77	±0.0475

Table 10 - Average Data Packet Delay varying the number of VoIp packets

n	Average VoIP Packet Delay (ms)	Error
10	0.462	±0.000152
20	0.485	±0.000196
30	0.510	±0.000177
40	0.533	±0.000193

Table 11 - Average VoIP Packet Delay varying the number of VoIp packets

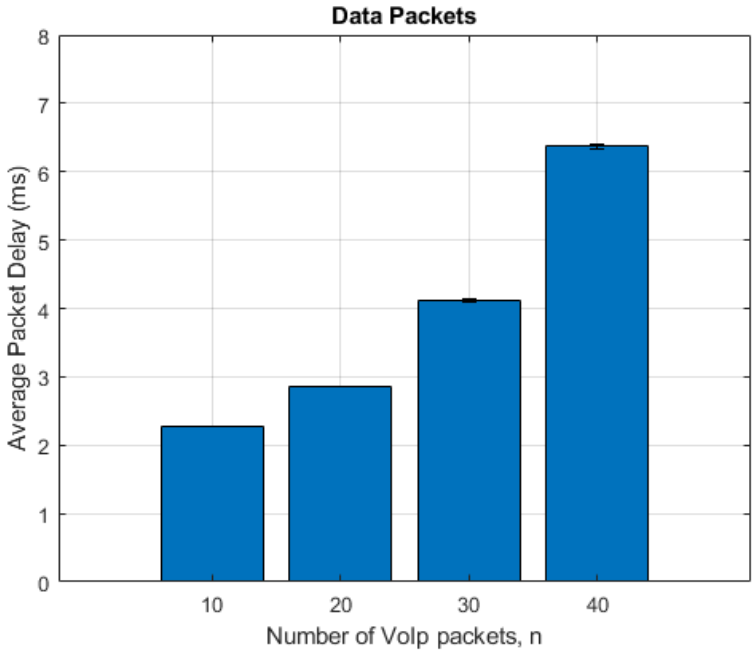

Figure 10 - Average Data Packet Delay varying the number of voIp packets

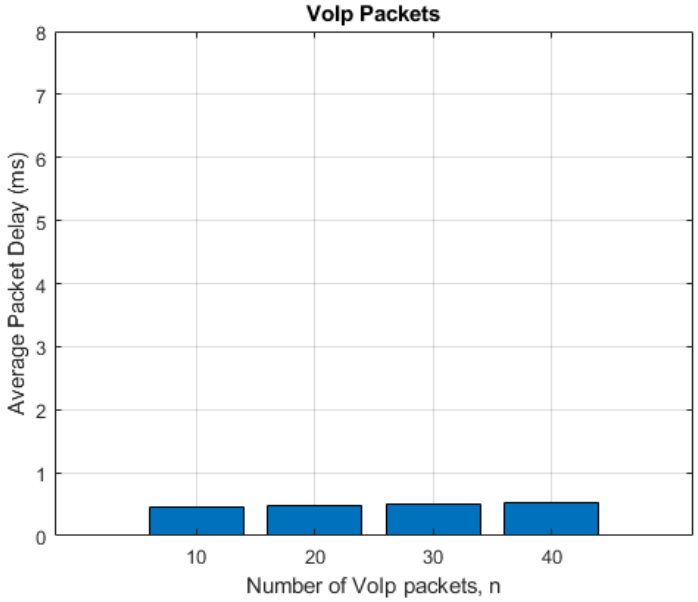

Figure 11 - Average Data Packet Delay varying the number of Volp packets

The code used to obtain these results is shown below:

```
% Initializing variables

P = 10000;
conf = 0.9;
lambda = 1500;
C = 10;
f = 1e6;
N = 50;
n = 10;


PacketLossdata = zeros(1,N);
PacketLossvoip = zeros(1,N);
AvgPacketDelaydata = zeros(1,N);
AvgPacketDelayvoip = zeros(1,N);
MaxPacketDelaydata = zeros(1,N);
MaxPacketDelayvoip = zeros(1,N);
Throughput = zeros(1,N);


for i=1:N

[PacketLossdata(i),PacketLossvoip(i),AvgPacketDelaydata(i),AvgPacketDelayvoip
(i),MaxPacketDelaydata(i),MaxPacketDelayvoip(i),Throughput(i)] =
Simulator4(lambda,C,f,P,n);
end


mAPDd = mean(AvgPacketDelaydata);
mAPDv = mean(AvgPacketDelayvoip);
```

```
term2d = norminv(1-conf/2)*sqrt(var(AvgPacketDelaydata)/N);
term2v = norminv(1-conf/2)*sqrt(var(AvgPacketDelayvoip)/N);


% Array with all average data packet delay

R1d = [mAPDd];

% Array with all average data packet delay errors

R2d = [term2d];

% Array with all average VoIP packet delay

R1v = [mAPDv];

% Array with all average data packet delay errors

R2v = [term2v];

fprintf('Alinea B, n = 10')
fprintf(['\nAvgPacketDelay data= %.2e +- %.2e' ...
    '\nAvgPacketDelay voip= %.2e +- %.2e'], ...
    mAPDd, term2d, mAPDv, term2v)
fprintf('\n\n=========================================================\n\n')


% ------------------------------------------------------------- %
n = 20;


for i=1:N

[PacketLossdata(i),PacketLossvoip(i),AvgPacketDelaydata(i),AvgPacketDelayvoip
(i),MaxPacketDelaydata(i),MaxPacketDelayvoip(i),Throughput(i)] =
Simulator4(lambda,C,f,P,n);
end


mAPDd = mean(AvgPacketDelaydata);
mAPDv = mean(AvgPacketDelayvoip);


term2d = norminv(1-conf/2)*sqrt(var(AvgPacketDelaydata)/N);
term2v = norminv(1-conf/2)*sqrt(var(AvgPacketDelayvoip)/N);


% Array with all average data packet delay

R1d = [R1d; mAPDd];

% Array with all average data packet delay errors

R2d = [R2d; term2d];
```

```
% Array with all average VoIP packet delay

R1v = [R1v; mAPDv];

% Array with all average VoIP packet delay errors


R2v = [R2v; term2v];


fprintf('Alinea B, n = 20')
fprintf(['\nAvgPacketDelay data= %.2e +- %.2e' ...
    '\nAvgPacketDelay voip= %.2e +- %.2e'], ...
    mAPDd, term2d, mAPDv, term2v)
fprintf('\n\n=========================================================\n\n')


% ------------------------------------------------------------- %
n = 30;


for i=1:N

[PacketLossdata(i),PacketLossvoip(i),AvgPacketDelaydata(i),AvgPacketDelayvoip
(i),MaxPacketDelaydata(i),MaxPacketDelayvoip(i),Throughput(i)] =
Simulator4(lambda,C,f,P,n);
end


mAPDd = mean(AvgPacketDelaydata);
mAPDv = mean(AvgPacketDelayvoip);
term2d = norminv(1-conf/2)*sqrt(var(AvgPacketDelaydata)/N);
term2v = norminv(1-conf/2)*sqrt(var(AvgPacketDelayvoip)/N);


% Array with all average data packet delay

R1d = [R1d; mAPDd];

% Array with all average data packet delay errors

R2d = [R2d; term2d];


% Array with all average VoIP packet delay

R1v = [R1v; mAPDv];

% Array with all average VoIP packet delay errors


R2v = [R2v; term2v];
```

```
fprintf('Alinea B, n = 30')
fprintf(['\nAvgPacketDelay data= %.2e +- %.2e' ...
    '\nAvgPacketDelay voip= %.2e +- %.2e'], ...
    mAPDd, term2d, mAPDv, term2v)
fprintf('\n\n=========================================================\n\n')


% ------------------------------------------------------------ %
n = 40;


for i=1:N

[PacketLossdata(i),PacketLossvoip(i),AvgPacketDelaydata(i),AvgPacketDelayvoip
(i),MaxPacketDelaydata(i),MaxPacketDelayvoip(i),Throughput(i)] =
Simulator4(lambda,C,f,P,n);
end

mAPDd = mean(AvgPacketDelaydata);
mAPDv = mean(AvgPacketDelayvoip);
term2d = norminv(1-conf/2)*sqrt(var(AvgPacketDelaydata)/N);
term2v = norminv(1-conf/2)*sqrt(var(AvgPacketDelayvoip)/N);


% Array with all average data packet delay

R1d = [R1d; mAPDd];

% Array with all average data packet delay errors

R2d = [R2d; term2d];


% Array with all average VoIP packet delay

R1v = [R1v; mAPDv];

% Array with all average VoIP packet delay errors


R2v = [R2v; term2v];
n = [10 20 30 40];


fprintf('Alinea B, n = 40')
fprintf(['\nAvgPacketDelay data= %.2e +- %.2e' ...
    '\nAvgPacketDelay voip= %.2e +- %.2e'], ...
    mAPDd, term2d, mAPDv, term2v)
fprintf('\n\n=========================================================\n\n')


% Bar chart of the average data packet delay

bar(n, R1d)
```

```
grid on
hold on
ylim([0 8])
xlabel('Number of VoIp packets, n')
ylabel('Average Packet Delay (ms)')
title('Data Packets')
er = errorbar(n, R1d, R2d, R2d);
er.Color = [0 0 0];
er.LineStyle = 'none';
hold off


% Bar chart of the average VoIP packet delay

bar(n, R1v);
grid on
hold on
ylim([0 8])
xlabel('Number of VoIp packets, n')
ylabel('Average Packet Delay (ms)')
title('VoIp Packets')
er = errorbar(n, R1v, R2v, R2v);
er.Color = [0 0 0];
er.LineStyle = 'none';
hold off
```

Exercise 2.c

Considering that the system is modelled by a M/G/1 queueing model with priorities the theoretical values of the average data packet delay and average VoIP packet delay were calculated following the formula shown below:

$$W_{Qk} = \begin{cases} \dfrac{\sum_{i=1}^{n}(\lambda_i E[S_i^2])}{2(1-\rho_1)} & ,k = 1 \\ \dfrac{\sum_{i=1}^{n}(\lambda_i E[S_i^2])}{2(1-\rho_1-\cdots-\rho_{k-1})(1-\rho_1-\cdots-\rho_k)} & ,k > 1 \end{cases} \qquad \rho_k = \lambda_k E[S_k]$$

Which resulted in the following results:

n	Average Data Packet Delay (ms)
10	2.38
20	3.11
30	4.49
40	7.99

Table 12 - Average Data Packet Delay varying the number of VoIp packets

n	Average VoIP Packet Delay (ms)
10	0.488
20	0.514
30	0.543
40	0.576

Table 13 - Average VoIp Packet Delay varying the number of VoIp packets

The theoretical values obtained are very similar to the results obtained in exercise 2.b and this proves that the priority queue implemented in *simulator4* is implemented correctly.

The following matlab code was used to calculate these values:

```
% 20 ms = 0,02 s

t = 0.02;
% VoIP lambdas

lambdas = [10/t 20/t 30/t 40/t];
C = 10e6;
lambdab = 1500;


% Average Data packet size

packetSizedata = 64:1518;
prob = zeros(1,1518);
prob(packetSizedata) = (1 - 0.19 - 0.23 -0.17) / (length(packetSizedata)-3);
prob(64) = 0.19;
prob(110) = 0.23;
prob(1518) = 0.17;
avgPacketSizedata = sum(prob(packetSizedata).*packetSizedata);


% Mew of Data packets

u = C/(avgPacketSizedata*8);
% Average VoIP packet size

packetSizevoip = 110:130;
avgPacketSizevoip = sum(packetSizevoip)/length(packetSizevoip);
pk = avgPacketSizevoip*8;


% Mew of VoIP packets

mewA = C/pk;
ESA = 1/mewA;
ESA2 = 2/(mewA^2);
ESB = 1/u;
```

```
ESB2 = 2/(u^2);
for i=1:4
    Wa = (((lambdas(i) * ESA2 + lambdab*ESB2) / (2 * (1 - lambdas(i)*ESA))) +
ESA) * 10^3;
    Wb = (((lambdas(i) * ESA2 + lambdab*ESB2) / (2 * (1 - lambdas(i)*ESA) *
(1 - lambdas(i)*ESA - lambdab*ESB))) + ESB) * 10^3;
    fprintf('n = %d', lambdas(i)*t)
    fprintf(['\nAvgPacketDelay VoIP= %.2e' ...
        '\nAvgPacketDelay Data= %.2e'], ...
       Wa, Wb)

fprintf('\n\n=========================================================\n\n')
end
```

Exercise 2.d

Running *Simulator3* 50 times with the following input values, C = 10 Mbps, f = 10.000 Bytes, P = 10.000 packets, λ = 1500 pps and n = 10, 20, 30, 40. The results obtained for the data packets are presented in *Figure12, Figure13, Table14 and Table16*, with the respective associated errors and the results for the VoIP packets are present in *Figure14, Figure15*, *Table15 and Table17, with the respective associated errors*.

By reducing the queue size of a FIFO queue less packets can be placed in the queue, which means that if less packets can be put on hold in the queue, the queue will get full more frequently and more packets will be lost, this effect can be observed in table 14 and table 15. When the number of VoIP packets is low (n = 10), the queue is big enough to handle the flow of packets without losing neither data or VoIP packets, but when the number of VoIP packets is greater than 10 (n > 10) some packet loss starts to appear in the system, the percentage of packet loss is greater in data packets than in VoIP packets because there are more data packets than VoIP packets, which means that if one packet is lost, the probability of that packet being a data packet is greater therefore the amount of packet losses for data packets must be greater than the packet loss of the VoIP packets, as shown in table14 and table 15.

The packet delay doesn't depend on the queue size but on the type of queue the system uses, in this case the queue is a FIFO, which means that all packets have the same priority, therefore is expected that all packet types will have a very similar average packet delay, as shown in table 16 and table 17.

n	Data Packet Loss (%)	Error
10	0.00	±0.00
20	0.0673	±0.00439
30	0.840	±0.0331
40	3.38	±0.0208

Table 14 - Data Packet Loss varying the number of VoIp packets

n	VoIP Packet Loss (%)	Error
10	0.00	±0.00
20	0.00843	±0.000757
30	0.128	±0.00546
40	0.453	±0.00550

Table 15 - VoIp Packet Loss varying the number of VoIp packets

n	Average Data Packet Delay (ms)	Error
10	6.86	±0.0435
20	22.7	±0.556
30	43.7	±0.345
40	61.9	±0.105

Table 16 - Average Data Packet Delay varying the number of VoIp packets

n	Average VoIP Packet Delay (ms)	Error
10	6.45	±0.0423
20	22.3	±0.562
30	43.4	±0.346
40	62.0	±0.105

Table 17 - Average VoIp Packet Delay varying the number of VoIp packets

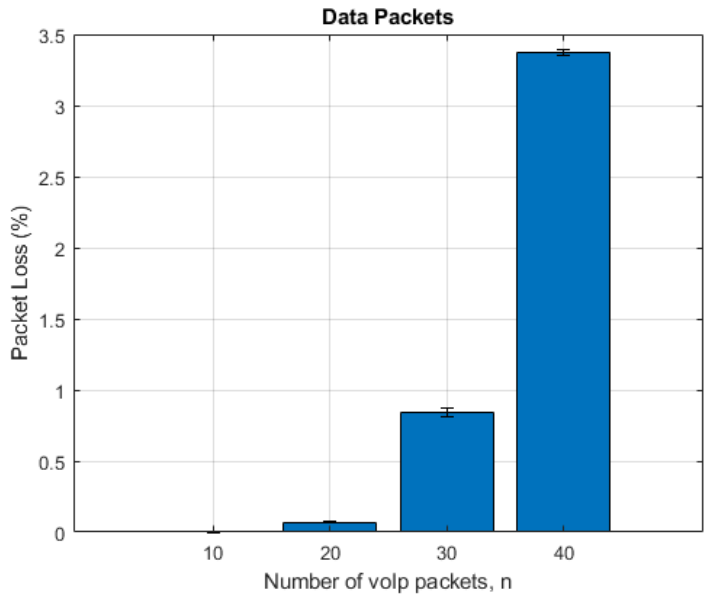

Figure 12 - Data Packet Loss varying the number of VoIp packets

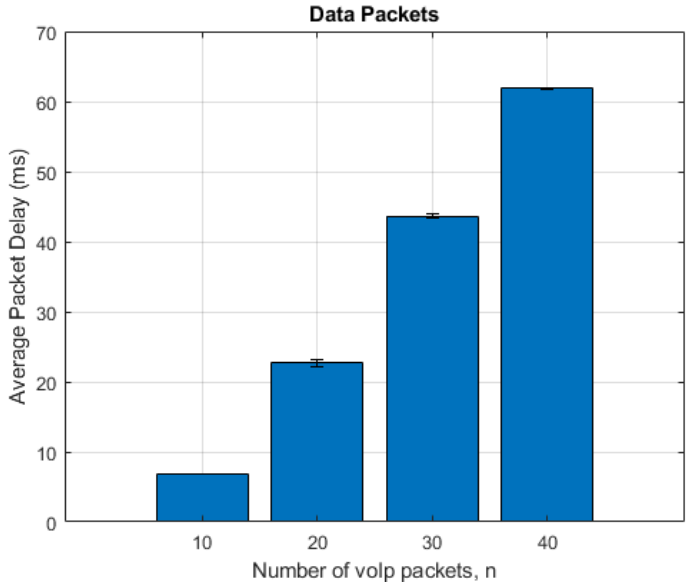

Figure 13 - Average Data Packet Delay varying the number of VoIp packets

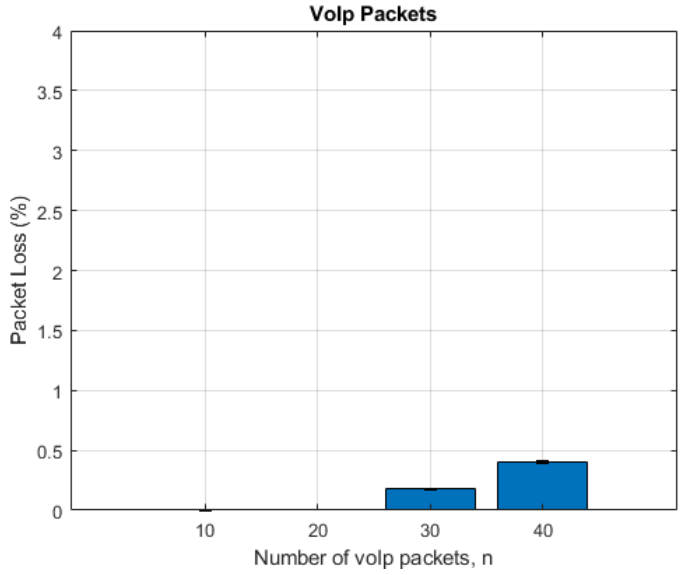

Figure 14 - VoIp Packet Loss varying the number of VoIp packets

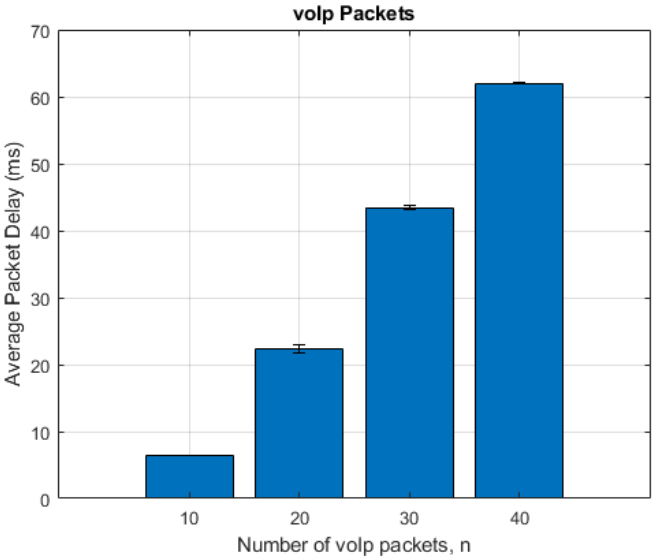

Figure 15 - Average Volp Packet Delay varying the number of Volp packets

The code used to obtain these results is shown below:

```
% Initializing variables

P = 10000;
conf = 0.9;
lambda = 1800;
C = 10;
f = 1e5;
N = 10;
n = 10;
PacketLossdata = zeros(1,N);
PacketLossvoip = zeros(1,N);
AvgPacketDelaydata = zeros(1,N);
AvgPacketDelayvoip = zeros(1,N);
MaxPacketDelaydata = zeros(1,N);
MaxPacketDelayvoip = zeros(1,N);
Throughput = zeros(1,N);


for i=1:N

[PacketLossdata(i),PacketLossvoip(i),AvgPacketDelaydata(i),AvgPacketDelayvoip
(i),MaxPacketDelaydata(i),MaxPacketDelayvoip(i),Throughput(i)] =
Simulator3(lambda,C,f,P,n);
end


mPLd = mean(PacketLossdata);
mPLv = mean(PacketLossvoip);
mAPDd = mean(AvgPacketDelaydata);
```

```
mAPDv = mean(AvgPacketDelayvoip);
term1d = norminv(1-conf/2)*sqrt(var(PacketLossdata)/N);
term1v = norminv(1-conf/2)*sqrt(var(PacketLossvoip)/N);
term2d = norminv(1-conf/2)*sqrt(var(AvgPacketDelaydata)/N);
term2v = norminv(1-conf/2)*sqrt(var(AvgPacketDelayvoip)/N);


% data packets
% Array with all packet losses of data packet delay

RPLd = [mPLd];

% Array with all average data packet delay

RAPDd = [mAPDd];

% Array with all packet losses of data packet delay errors

Rt1d = [term1d];

% Array with all average data packet delay errors

Rt2d = [term2d];


%voIp packets
% Array with all packet losses of VoIP packet delay

RPLv = [mPLv];

% Array with all average VoIP packet delay

RAPDv = [mAPDv];

% Array with all packet losses of VoIP packet delay errors


Rt1v = [term1v];

% Array with all average VoIP packet delay errors

Rt2v = [term2v];


fprintf('Alinea D, n = 10')
fprintf(['PacketLoss data= %.2e +- %.2e\n' ...
    'PacketLoss voip= %.2e +- %.2e\n' ...
    'AvgPacketDelay data= %.2e +- %.2e\n' ...
    'AvgPacketDelay voip= %.2e +- %.2e\n'], ...
    mPLd, term1d, mPLv, term1v, mAPDd, term2d, mAPDv, term2v)
fprintf('\n\n=======================================================\n\n')


% ------------------------------------------------------------- %
```

```
n = 20;
for i=1:N

[PacketLossdata(i),PacketLossvoip(i),AvgPacketDelaydata(i),AvgPacketDelayvoip
(i),MaxPacketDelaydata(i),MaxPacketDelayvoip(i),Throughput(i)] =
Simulator3(lambda,C,f,P,n);
end


mPLd = mean(PacketLossdata);
mPLv = mean(PacketLossvoip);
mAPDd = mean(AvgPacketDelaydata);
mAPDv = mean(AvgPacketDelayvoip);
term1d = norminv(1-conf/2)*sqrt(var(PacketLossdata)/N);
term1v = norminv(1-conf/2)*sqrt(var(PacketLossvoip)/N);
term2d = norminv(1-conf/2)*sqrt(var(AvgPacketDelaydata)/N);
term2v = norminv(1-conf/2)*sqrt(var(AvgPacketDelayvoip)/N);


% data packets
% Array with all packet losses of data packet delay

RPLd = [RPLd; mPLd];

% Array with all average data packet delay

RAPDd = [RAPDd; mAPDd];

% Array with all packet losses of data packet delay errors

Rt1d = [Rt1d; term1d];

% Array with all average data packet delay errors

Rt2d = [Rt2d; term2d];


%VoIP packets
% Array with all packet losses of VoIP packet delay

RPLv = [RPLv; mPLv];

% Array with all average VoIP packet delay

RAPDv = [RAPDv; mAPDv];

% Array with all packet losses of VoIP packet delay errors

Rt1v = [Rt1v; term1v];

% Array with all average VoIP packet delay errors
```

```
Rt2v = [Rt2v; term2v];


fprintf('Alinea D, n = 20')
fprintf(['PacketLoss data= %.2e +- %.2e\n' ...
    'PacketLoss voip= %.2e +- %.2e\n' ...
    'AvgPacketDelay data= %.2e +- %.2e\n' ...
    'AvgPacketDelay voip= %.2e +- %.2e\n'], ...
    mPLd, term1d, mPLv, term1v, mAPDd, term2d, mAPDv, term2v)
fprintf('\n\n=======================================================\n\n')


% ------------------------------------------------------------- %
n = 30;


for i=1:N

[PacketLossdata(i),PacketLossvoip(i),AvgPacketDelaydata(i),AvgPacketDelayvoip
(i),MaxPacketDelaydata(i),MaxPacketDelayvoip(i),Throughput(i)] =
Simulator3(lambda,C,f,P,n);
end


mPLd = mean(PacketLossdata);
mPLv = mean(PacketLossvoip);
mAPDd = mean(AvgPacketDelaydata);
mAPDv = mean(AvgPacketDelayvoip);
term1d = norminv(1-conf/2)*sqrt(var(PacketLossdata)/N);
term1v = norminv(1-conf/2)*sqrt(var(PacketLossvoip)/N);
term2d = norminv(1-conf/2)*sqrt(var(AvgPacketDelaydata)/N);
term2v = norminv(1-conf/2)*sqrt(var(AvgPacketDelayvoip)/N);


% data packets
% Array with all packet losses of data packet delay

RPLd = [RPLd; mPLd];

% Array with all average data packet delay

RAPDd = [RAPDd; mAPDd];

% Array with all packet losses of data packet delay errors

Rt1d = [Rt1d; term1d];

% Array with all average data packet delay errors

Rt2d = [Rt2d; term2d];


%VoIP packets
% Array with all packet losses of VoIP packet delay
```

```
RPLv = [RPLv; mPLv];

% Array with all average VoIP packet delay

RAPDv = [RAPDv; mAPDv];

% Array with all packet losses of VoIP packet delay errors

Rt1v = [Rt1v; term1v];

% Array with all average VoIP packet delay errors

Rt2v = [Rt2v; term2v];


fprintf('Alinea D, n = 30')
fprintf(['PacketLoss data= %.2e +- %.2e\n' ...
    'PacketLoss voip= %.2e +- %.2e\n' ...
    'AvgPacketDelay data= %.2e +- %.2e\n' ...
    'AvgPacketDelay voip= %.2e +- %.2e\n'], ...
    mPLd, term1d, mPLv, term1v, mAPDd, term2d, mAPDv, term2v)
fprintf('\n\n=========================================================\n\n')


% ------------------------------------------------------------- %
n = 40;


for i=1:N

[PacketLossdata(i),PacketLossvoip(i),AvgPacketDelaydata(i),AvgPacketDelayvoip
(i),MaxPacketDelaydata(i),MaxPacketDelayvoip(i),Throughput(i)] =
Simulator3(lambda,C,f,P,n);
end


mPLd = mean(PacketLossdata);
mPLv = mean(PacketLossvoip);
mAPDd = mean(AvgPacketDelaydata);
mAPDv = mean(AvgPacketDelayvoip);
term1d = norminv(1-conf/2)*sqrt(var(PacketLossdata)/N);
term1v = norminv(1-conf/2)*sqrt(var(PacketLossvoip)/N);
term2d = norminv(1-conf/2)*sqrt(var(AvgPacketDelaydata)/N);
term2v = norminv(1-conf/2)*sqrt(var(AvgPacketDelayvoip)/N);


% data packets
% Array with all packet losses of data packet delay

RPLd = [RPLd; mPLd];

% Array with all average data packet delay

RAPDd = [RAPDd; mAPDd];
```

```
% Array with all packet losses of data packet delay errors

Rt1d = [Rt1d; term1d];

% Array with all average data packet delay errors

Rt2d = [Rt2d; term2d];


%VoIP packets
% Array with all packet losses of VoIP packet delay

RPLv = [RPLv; mPLv];

% Array with all average VoIP packet delay

RAPDv = [RAPDv; mAPDv];

% Array with all packet losses of VoIP packet delay errors

Rt1v = [Rt1v; term1v];

% Array with all average VoIP packet delay errors

Rt2v = [Rt2v; term2v];


% ------------------------------------------------------------- %


n = [10 20 30 40];


fprintf('Alinea D, n = 40')
fprintf(['PacketLoss data= %.2e +- %.2e\n' ...
    'PacketLoss voip= %.2e +- %.2e\n' ...
    'AvgPacketDelay data= %.2e +- %.2e\n' ...
    'AvgPacketDelay voip= %.2e +- %.2e\n'], ...
    mPLd, term1d, mPLv, term1v, mAPDd, term2d, mAPDv, term2v)
fprintf('\n\n=========================================================\n\n')


% data packets
% bar chart of the Packet Loss of data packets
bar(n, RPLd)
hold on
grid on
ylim([0 4])
title('Data Packets')
xlabel('Number of VoIp packets, n')
ylabel('Packet Loss (%)')
er = errorbar(n, RPLd, Rt1d, Rt1d);
er.Color = [0 0 0];
```

```
er.LineStyle = 'none';
hold off


% bar chart of the Average Packet Delay of data packets
bar(n, RAPDd)
hold on
grid on
title('Data Packets')
xlabel('Number of VoIp packets, n')
ylabel('Average Packet Delay (ms)')
er = errorbar(n, RAPDd, Rt2d, Rt2d);
er.Color = [0 0 0];
er.LineStyle = 'none';
hold off


% voIp packets
% bar chart of the Packet Loss of VoIP packets
bar(n, RPLv)
hold on
grid on
ylim([0 4])
title('VoIp Packets')
xlabel('Number of voIp packets, n')
ylabel('Packet Loss (%)')
er = errorbar(n, RPLv, Rt1v, Rt1v);
er.Color = [0 0 0];
er.LineStyle = 'none';
hold off


% bar chart of the Average Packet Delay of VoIP packets
bar(n, RAPDv)
hold on
grid on
title('VoIp Packets')
xlabel('Number of voIp packets, n')
ylabel('Average Packet Delay (ms)')
er = errorbar(n, RAPDv, Rt2v, Rt2v);
er.Color = [0 0 0];
er.LineStyle = 'none';
hold off
```

Exercise 2.e

Running *Simulator4* 50 times with the following input values, C = 10 Mbps, f = 10.000 Bytes, P = 10.000 packets, λ = 1500 pps and n = 10, 20, 30, 40. The results obtained for the data packets are presented in *Figure16, Figure18* and *Table18 and Table20*, with the respective associated errors and the results for the VoIP packets are present in *Figure17, Figure19. Table19* and *Table21, with the respective associated errors*.

In *simulator4* the queue type is a priority queue which means that the packets with the highest priority in the queue will be transmitted first, in this case the VoIP packets have the highest priority, this will influence the average packet delay of each packet type, because if one packet has more priority than the others will have to wait less time on the queue to be transmitted, therefore lowering the average packet delay for that packet type and subsequently the packets with the lowest priority will experience the highest average packet delay. This effect is shown in the tables 20 and 21, this results contrast with the ones obtained in exercise 2.d where the average packet delay for each packet type was very similar.

The priority queue won't affect the packet losses for each packet, because this type of queue only affects the order by each packets leaves the queue and the packet loss depends on the space available in the queue when the packet arrives, so it's to be expected that the packet losses obtained in exercise 2.d will remain similar in this exercise, as shown in tables 18 and 19.

n	Data Packet Loss (%)	Error
10	0.00	±0.00
20	0.0122	±0.00154
30	0.820	±0.0291
40	3.57	±0.0274

Table 18 - Data Packet Loss varying the number of VoIp packets

n	VoIP Packet Loss (%)	Error
10	0.00	±0.00
20	0.00278	±0.000349
30	0.105	±0.00538
40	0.514	±0.00647

Table 19 - VoIp Packet Loss varying the number of VoIp packets

n	Average Data Packet Delay (ms)	Error
10	7.60	±0.0677
20	19.7	±0.3000
30	53.5	±0.7990
40	77.7	±0.2660

Table 20 - Average Data Packet Delay varying the number of VoIp packets

n	Average VoIP Packet Delay (ms)	Error
10	0.534	±0.000268
20	0.559	±0.000313
30	0.566	±0.000367
40	0.568	±0.000213

Table 21 - Average VoIp Packet Delay varying the number of VoIp packets

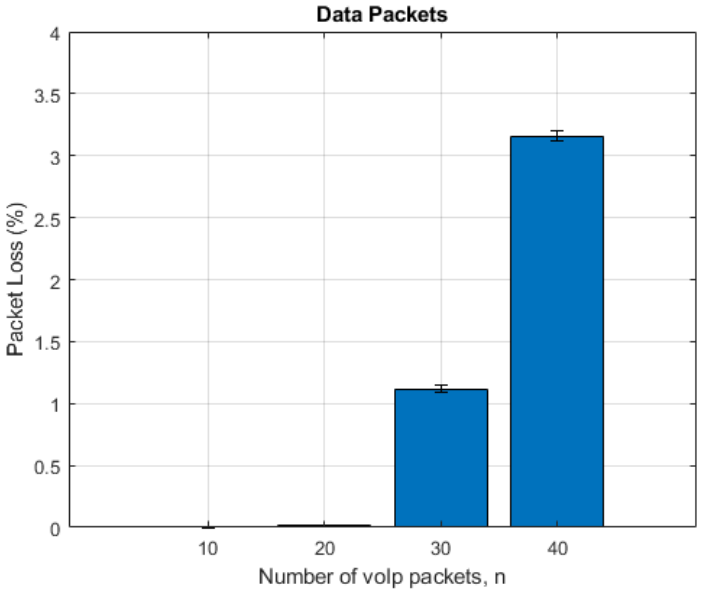

Figure 16 - Data Packet Loss varying the number of VoIp packets

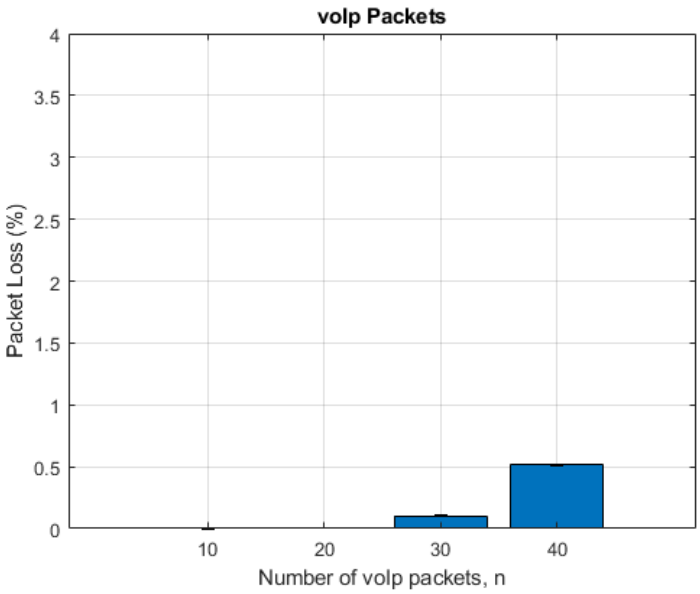

Figure 17 - VoIp Packet Loss varying the number of VoIp packets

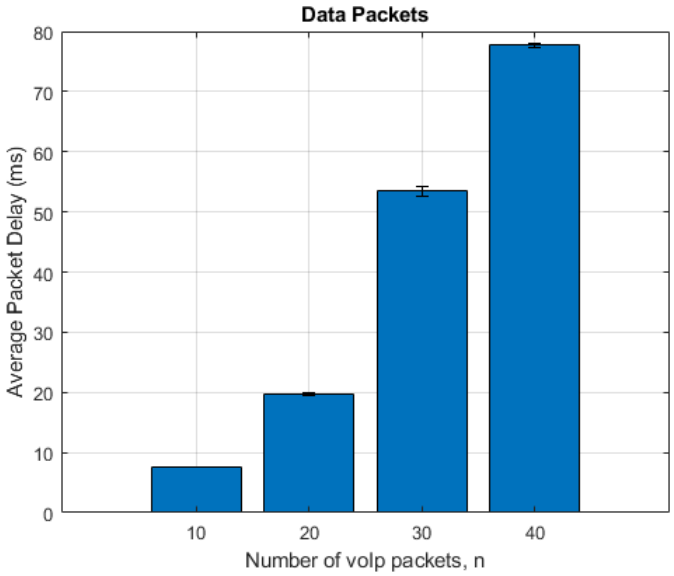

Figure 18 – Average Data Packet Delay varying the number of VoIp packets

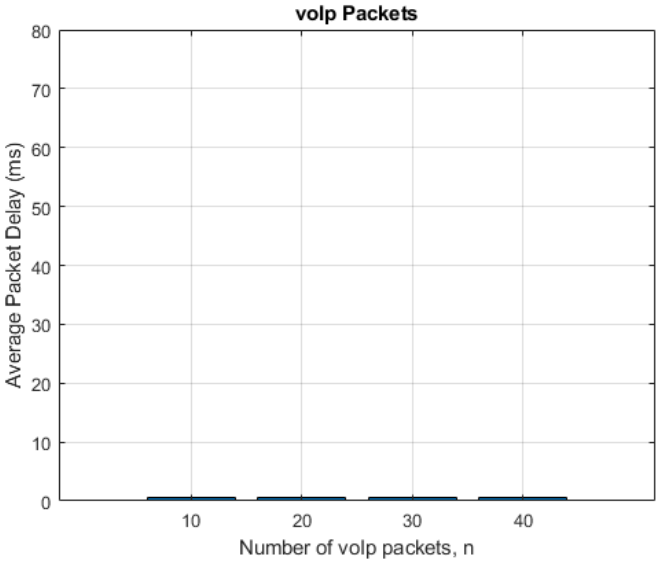

Figure 19 – Average VoIp Packet Delay varying the number of VoIp packets

The code used to obtain these results is shown below:

```
% Initializing variables

P = 10000;
conf = 0.9;
lambda = 1800;
C = 10;
f = 1e5;
N = 10;
n = 10;


PacketLossdata = zeros(1,N);
PacketLossvoip = zeros(1,N);
AvgPacketDelaydata = zeros(1,N);
AvgPacketDelayvoip = zeros(1,N);
MaxPacketDelaydata = zeros(1,N);
MaxPacketDelayvoip = zeros(1,N);
Throughput = zeros(1,N);


for i=1:N

[PacketLossdata(i),PacketLossvoip(i),AvgPacketDelaydata(i),AvgPacketDelayvoip
(i),MaxPacketDelaydata(i),MaxPacketDelayvoip(i),Throughput(i)] =
Simulator4(lambda,C,f,P,n);
end
```

```
mPLd = mean(PacketLossdata);
mPLv = mean(PacketLossvoip);
mAPDd = mean(AvgPacketDelaydata);
mAPDv = mean(AvgPacketDelayvoip);
term1d = norminv(1-conf/2)*sqrt(var(PacketLossdata)/N);
term1v = norminv(1-conf/2)*sqrt(var(PacketLossvoip)/N);
term2d = norminv(1-conf/2)*sqrt(var(AvgPacketDelaydata)/N);
term2v = norminv(1-conf/2)*sqrt(var(AvgPacketDelayvoip)/N);


% data packets
% Array with all packet losses of data packet delay

RPLd = [mPLd];

% Array with all average data packet delay

RAPDd = [mAPDd];

% Array with all packet losses of data packet delay errors

Rt1d = [term1d];

% Array with all average data packet delay errors

Rt2d = [term2d];


%voIp packets
% Array with all packet losses of VoIP packet delay

RPLv = [mPLv];

% Array with all average VoIP packet delay

RAPDv = [mAPDv];

% Array with all packet losses of VoIP packet delay errors


Rt1v = [term1v];

% Array with all average VoIP packet delay errors

Rt2v = [term2v];


fprintf('Alinea E, n = 10')
fprintf(['PacketLoss data= %.2e +- %.2e\n' ...
    'PacketLoss voip= %.2e +- %.2e\n' ...
    'AvgPacketDelay data= %.2e +- %.2e\n' ...
    'AvgPacketDelay voip= %.2e +- %.2e\n'], ...
    mPLd, term1d, mPLv, term1v, mAPDd, term2d, mAPDv, term2v)
fprintf('\n\n=======================================================\n\n')
```

```
% ------------------------------------------------------------- %


n = 20;


for i=1:N

[PacketLossdata(i),PacketLossvoip(i),AvgPacketDelaydata(i),AvgPacketDelayvoip
(i),MaxPacketDelaydata(i),MaxPacketDelayvoip(i),Throughput(i)] =
Simulator4(lambda,C,f,P,n);
end


mPLd = mean(PacketLossdata);
mPLv = mean(PacketLossvoip);
mAPDd = mean(AvgPacketDelaydata);
mAPDv = mean(AvgPacketDelayvoip);
term1d = norminv(1-conf/2)*sqrt(var(PacketLossdata)/N);
term1v = norminv(1-conf/2)*sqrt(var(PacketLossvoip)/N);
term2d = norminv(1-conf/2)*sqrt(var(AvgPacketDelaydata)/N);
term2v = norminv(1-conf/2)*sqrt(var(AvgPacketDelayvoip)/N);


% data packets
% Array with all packet losses of data packet delay

RPLd = [RPLd; mPLd];

% Array with all average data packet delay

RAPDd = [RAPDd; mAPDd];

% Array with all packet losses of data packet delay errors

Rt1d = [Rt1d; term1d];

% Array with all average data packet delay errors

Rt2d = [Rt2d; term2d];


%VoIP packets
% Array with all packet losses of VoIP packet delay

RPLv = [RPLv; mPLv];

% Array with all average VoIP packet delay

RAPDv = [RAPDv; mAPDv];

% Array with all packet losses of VoIP packet delay errors
```

```
Rt1v = [Rt1v; term1v];

% Array with all average VoIP packet delay errors

Rt2v = [Rt2v; term2v];

fprintf('Alinea E, n = 20')
fprintf(['PacketLoss data= %.2e +- %.2e\n' ...
    'PacketLoss voip= %.2e +- %.2e\n' ...
    'AvgPacketDelay data= %.2e +- %.2e\n' ...
    'AvgPacketDelay voip= %.2e +- %.2e\n'], ...
    mPLd, term1d, mPLv, term1v, mAPDd, term2d, mAPDv, term2v)
fprintf('\n\n=========================================================\n\n')


% ------------------------------------------------------------- %
n = 30;


for i=1:N

[PacketLossdata(i),PacketLossvoip(i),AvgPacketDelaydata(i),AvgPacketDelayvoip
(i),MaxPacketDelaydata(i),MaxPacketDelayvoip(i),Throughput(i)] =
Simulator4(lambda,C,f,P,n);
end


mPLd = mean(PacketLossdata);
mPLv = mean(PacketLossvoip);
mAPDd = mean(AvgPacketDelaydata);
mAPDv = mean(AvgPacketDelayvoip);
term1d = norminv(1-conf/2)*sqrt(var(PacketLossdata)/N);
term1v = norminv(1-conf/2)*sqrt(var(PacketLossvoip)/N);
term2d = norminv(1-conf/2)*sqrt(var(AvgPacketDelaydata)/N);
term2v = norminv(1-conf/2)*sqrt(var(AvgPacketDelayvoip)/N);


% data packets
% Array with all packet losses of data packet delay

RPLd = [RPLd; mPLd];

% Array with all average data packet delay

RAPDd = [RAPDd; mAPDd];

% Array with all packet losses of data packet delay errors

Rt1d = [Rt1d; term1d];

% Array with all average data packet delay errors

Rt2d = [Rt2d; term2d];
```

```
%VoIP packets
% Array with all packet losses of VoIP packet delay

RPLv = [RPLv; mPLv];

% Array with all average VoIP packet delay

RAPDv = [RAPDv; mAPDv];

% Array with all packet losses of VoIP packet delay errors

Rt1v = [Rt1v; term1v];

% Array with all average VoIP packet delay errors

Rt2v = [Rt2v; term2v];


fprintf('Alinea E, n = 30')
fprintf(['PacketLoss data= %.2e +- %.2e\n' ...
    'PacketLoss voip= %.2e +- %.2e\n' ...
    'AvgPacketDelay data= %.2e +- %.2e\n' ...
    'AvgPacketDelay voip= %.2e +- %.2e\n'], ...
    mPLd, term1d, mPLv, term1v, mAPDd, term2d, mAPDv, term2v)
fprintf('\n\n=========================================================\n\n')


% ------------------------------------------------------------- %
n = 40;


for i=1:N

[PacketLossdata(i),PacketLossvoip(i),AvgPacketDelaydata(i),AvgPacketDelayvoip
(i),MaxPacketDelaydata(i),MaxPacketDelayvoip(i),Throughput(i)] =
Simulator4(lambda,C,f,P,n);
end


mPLd = mean(PacketLossdata);
mPLv = mean(PacketLossvoip);
mAPDd = mean(AvgPacketDelaydata);
mAPDv = mean(AvgPacketDelayvoip);
term1d = norminv(1-conf/2)*sqrt(var(PacketLossdata)/N);
term1v = norminv(1-conf/2)*sqrt(var(PacketLossvoip)/N);
term2d = norminv(1-conf/2)*sqrt(var(AvgPacketDelaydata)/N);
term2v = norminv(1-conf/2)*sqrt(var(AvgPacketDelayvoip)/N);


% data packets
% Array with all packet losses of data packet delay

RPLd = [RPLd; mPLd];
```

```
% Array with all average data packet delay

RAPDd = [RAPDd; mAPDd];

% Array with all packet losses of data packet delay errors

Rt1d = [Rt1d; term1d];

% Array with all average data packet delay errors

Rt2d = [Rt2d; term2d];


%VoIP packets
% Array with all packet losses of VoIP packet delay

RPLv = [RPLv; mPLv];

% Array with all average VoIP packet delay

RAPDv = [RAPDv; mAPDv];

% Array with all packet losses of VoIP packet delay errors

Rt1v = [Rt1v; term1v];

% Array with all average VoIP packet delay errors

Rt2v = [Rt2v; term2v];


% ------------------------------------------------------------- %


n = [10 20 30 40];


fprintf('Alinea E, n = 40')
fprintf(['PacketLoss data= %.2e +- %.2e\n' ...
    'PacketLoss voip= %.2e +- %.2e\n' ...
    'AvgPacketDelay data= %.2e +- %.2e\n' ...
    'AvgPacketDelay voip= %.2e +- %.2e\n'], ...
    mPLd, term1d, mPLv, term1v, mAPDd, term2d, mAPDv, term2v)
fprintf('\n\n=======================================================\n\n')


% data packets
% bar chart of the Packet Loss of Data packets
bar(n, RPLd)
hold on
grid on
ylim([0 4])
title('Data Packets')
xlabel('Number of voIp packets, n')
```

```
ylabel('Packet Loss (%)')
er = errorbar(n, RPLd, Rt1d, Rt1d);
er.Color = [0 0 0];
er.LineStyle = 'none';
hold off


% bar chart of the Average Packet Delay of Data packets
bar(n, RAPDd)
hold on
ylim([0 80])
grid on
title('Data Packets')
xlabel('Number of voIp packets, n')
ylabel('Average Packet Delay (ms)')
er = errorbar(n, RAPDd, Rt2d, Rt2d);
er.Color = [0 0 0];
er.LineStyle = 'none';
hold off


% voIp packets
% bar chart of the Packet Loss of VoIP packets
bar(n, RPLv)
hold on
grid on
ylim([0 4])
title('voIp Packets')
xlabel('Number of voIp packets, n')
ylabel('Packet Loss (%)')
er = errorbar(n, RPLv, Rt1v, Rt1v);
er.Color = [0 0 0];
er.LineStyle = 'none';
hold off


% bar chart of the Average Packet Delay of VoIP packets
bar(n, RAPDv)
hold on
ylim([0 80])
grid on
title('voIp Packets')
xlabel('Number of voIp packets, n')
ylabel('Average Packet Delay (ms)')
er = errorbar(n, RAPDv, Rt2v, Rt2v);
er.Color = [0 0 0];
er.LineStyle = 'none';
hold off
```

Exercise 2.f

A new version of *Simulator4* was developed to consider that VoIP packets are always accepted in the queue (if there is enough space) but data packets are accepted in the queue only if the total queue occupation does not become higher than 90% (a simplified version of WRED – Weighted Random Early Discard).

To implement this type of queue the method by each data packet was added to the queue had to be changed, first the simulator needs to calculate if the queue occupation doesn't become higher than 90%, if it becomes the packet will be discarded, in order to achieve this the simulator compares the queue occupation plus the packet size to 0,9 * f(size of the queue), if the queue occupation plus the packet size is inferior to or equal to 0.9 * f the data packet is added. The resulting code is as follows:

```
function [PLd, PLv , APDd , APDv, MPDd, MPDv , TT] =
Simulator5(lambda,C,f,P,n)
% 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
% OUTPUT PARAMETERS:
%  PLdata   - packet loss of data packets (%)
%  PLvoip   - packet loss of VoIP packets (%)
%  APDdata  - average delay of data packets (milliseconds)
%  APDvoip  - average delay of VoIP packets (milliseconds)
%  MPDdata  - maximum delay of data packets (milliseconds)
%  MPDvoip  - maximum delay of voip packets (milliseconds)
%  TT   - transmitted throughput (Mbps)

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

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

%Statistical Counters:
TOTALDATAPACKETS= 0;       % No. of data packets arrived to the system
TOTALVOIPPACKETS= 0;       % No. of voip packets arrived to the system
LOSTDATAPACKETS= 0;        % No. of data packets dropped due to buffer
overflow
LOSTVOIPPACKETS= 0;        % No. of voip packets dropped due to buffer
overflow
TRANSMITTEDDATAPACKETS= 0; % No. of transmitted data packets
TRANSMITTEDVOIPPACKETS= 0; % No. of transmitted voip packets
TRANSMITTEDDATABYTES= 0;   % Sum of the Bytes of transmitted data packets
TRANSMITTEDVOIPBYTES= 0;   % Sum of the Bytes of transmitted voip packets
DELAYSDATA= 0;             % Sum of the delays of transmitted data packets
DELAYSVOIP= 0;             % Sum of the delays of transmitted voip packets
MAXDELAYDATA= 0;           % Maximum delay among all transmitted data packets
```

```
MAXDELAYVOIP= 0;           % Maximum delay among all transmitted voip packets
TRANSMITTEDPACKETS = 0;

% Initializing the simulation clock:
Clock= 0;
% Initializing the List of Events with the first ARRIVAL:
tmp= Clock + exprnd(1/lambda);
packetType=0; % 0->Data
EventList = [ARRIVAL, tmp, GeneratePacketSize(), tmp, packetType];
packetType=1; % 1->VoIP
for i=1:n
    time = unifrnd(0,0.02);
    EventList = [EventList; ARRIVAL, time, Size(), time, packetType];
end

%Similation loop:
while TRANSMITTEDPACKETS<P           % Stopping criterium
    EventList= sortrows(EventList,2);    % Order EventList by time
    Event= EventList(1,1);               % Get first event and
    Clock= EventList(1,2);               %   and
    PacketSize= EventList(1,3);          %   associated
    ArrivalInstant= 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 == 0
                TOTALDATAPACKETS= TOTALDATAPACKETS+1;
                tmp= Clock + exprnd(1/lambda);
                EventList = [EventList; ARRIVAL, tmp, GeneratePacketSize(),
tmp, PacketType];
                if STATE==0
                    STATE= 1;
                    EventList = [EventList; DEPARTURE, Clock +
8*PacketSize/(C*10^6), PacketSize, Clock, PacketType];
                else
                    if QUEUEOCCUPATION + PacketSize <= (f*0.9) % 90% check
                        QUEUE= [QUEUE;PacketSize , Clock, PacketType];
                        QUEUEOCCUPATION= QUEUEOCCUPATION + PacketSize;
                    else
                        LOSTDATAPACKETS= LOSTDATAPACKETS + 1;
                    end
                end
            else
                TOTALVOIPPACKETS= TOTALVOIPPACKETS+1;
                time = Clock+unifrnd(0.016,0.024);
                EventList = [EventList; ARRIVAL, time, Size(), time,
PacketType];
                if STATE==0
                    STATE= 1;
                    EventList = [EventList; DEPARTURE, Clock +
8*PacketSize/(C*10^6), PacketSize, Clock, PacketType];
                else
                    if QUEUEOCCUPATION + PacketSize <= f
                        QUEUE= [QUEUE;PacketSize , Clock, PacketType];
                        % sort QUEUE by packetType (1) primeiro (VoIP) (0)
depois (Data)
                        QUEUE = sortrows(QUEUE,3,"descend");
                        QUEUEOCCUPATION= QUEUEOCCUPATION + PacketSize;
```

```
                    else
                        LOSTVOIPPACKETS= LOSTVOIPPACKETS + 1;
                    end
                end
            end
        case DEPARTURE                      % If first event is a DEPARTURE
            TRANSMITTEDPACKETS = TRANSMITTEDPACKETS + 1;
            if PacketType == 0
                TRANSMITTEDDATABYTES= TRANSMITTEDDATABYTES + PacketSize;
                DELAYSDATA= DELAYSDATA + (Clock - ArrivalInstant);
                if Clock - ArrivalInstant > MAXDELAYDATA
                    MAXDELAYDATA= Clock - ArrivalInstant;
                end
                TRANSMITTEDDATAPACKETS= TRANSMITTEDDATAPACKETS + 1;
                if QUEUEOCCUPATION > 0
                    EventList = [EventList; DEPARTURE, Clock +
8*QUEUE(1,1)/(C*10^6), QUEUE(1,1), QUEUE(1,2), QUEUE(1,3)];
                    QUEUEOCCUPATION= QUEUEOCCUPATION - QUEUE(1,1);
                    QUEUE(1,:)= [];
                else
                    STATE= 0;
                end
            else
                TRANSMITTEDVOIPBYTES= TRANSMITTEDVOIPBYTES + PacketSize;
                DELAYSVOIP= DELAYSVOIP + (Clock - ArrivalInstant);
                if Clock - ArrivalInstant > MAXDELAYVOIP
                    MAXDELAYVOIP= Clock - ArrivalInstant;
                end
                TRANSMITTEDVOIPPACKETS= TRANSMITTEDVOIPPACKETS + 1;
                if QUEUEOCCUPATION > 0
                    EventList = [EventList; DEPARTURE, Clock +
8*QUEUE(1,1)/(C*10^6), QUEUE(1,1), QUEUE(1,2), QUEUE(1,3)];
                    QUEUEOCCUPATION= QUEUEOCCUPATION - QUEUE(1,1);
                    QUEUE(1,:)= [];
                else
                    STATE= 0;
                end
            end
    end
end

%Performance parameters determination:
PLd= 100*LOSTDATAPACKETS/TOTALDATAPACKETS;                          % in %
PLv= 100*LOSTVOIPPACKETS/TOTALVOIPPACKETS;                          % in %
APDd= 1000*DELAYSDATA/TRANSMITTEDDATAPACKETS;                       % in
milliseconds
APDv= 1000*DELAYSVOIP/TRANSMITTEDVOIPPACKETS;                       % in
milliseconds
MPDd= 1000*MAXDELAYDATA;                                            % in
milliseconds
MPDv= 1000*MAXDELAYVOIP;                                            % in
milliseconds
TT= 10^(-6)*(TRANSMITTEDDATABYTES+TRANSMITTEDVOIPBYTES)*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

function out= Size()
    out = randi([110,130]);
end
```

The results obtained in this exercise match the expectations for this type of queue, that is it would be expected that the Data Packet Loss would increase relatively to the exercise 2.e and the VoIp Packet Loss would decrease to 0 or values close to 0 and for the Average Data or VoIp Packet Delay remain the same as exercise 2.e. The Data Packet Loss increases because if the simulator can discard Data Packets before they enter the queue this will create more opportunities for Data Packets to be discarded, the opposite effect is present in the VoIp packets, that is the simulator will a lot of the time have at least 10% of its queue empty which creates more opportunities to accept VoIp packets and this makes the VoIp Packet Loss decrease considerably. The Average Packet Delay remains almost the same as exercise 2.e because this type of queue only affects the insertion of new packets in the queue and not their transmission time, therefore the Average Packet Delay will remain the same.

When the number of VoIp packets increases in a priority queue the packet loss of both services will not be different than a normal FIFO queue, but the average packet delay will be significantly lower in the packets which have the higher priority.

When the number of VoIp packets increases in a queue where the packet acceptance is differentiated the packet loss for the lower priority service will increase while the highest priority service will have 0 or almost 0 packet losses, but the average packet delay will be similar to all services, if the queue does not have priority in the transmission.

n	Data Packet Loss (%)	Error
10	0.00	±0.00
20	0.0154	±0.00193
30	1.29	±0.0347
40	3.74	±0.0591

Table 22 - VoIp Packet Loss varying the number of VoIp packets

n	VoIP Packet Loss (%)	Error
10	0.00	±0.00
20	0.00	±0.00
30	0.00	±0.00
40	0.00	±0.00

Table 23 - VoIp Packet Loss varying the number of VoIp packets

n	Average Data Packet Delay (ms)	Error
10	8.47	±0.114
20	19.9	±0.375
30	53.1	±0.550
40	72.4	±0.168

Table 24 - Average Data Packet Delay varying the number of VoIp packets

n	Average VoIP Packet Delay (ms)	Error
10	0.533	±0.000358
20	0.557	±0.000421
30	0.569	±0.000105
40	0.571	±0.000218

Table 25 – Average VoIp Packet Delay varying the number of VoIp packets

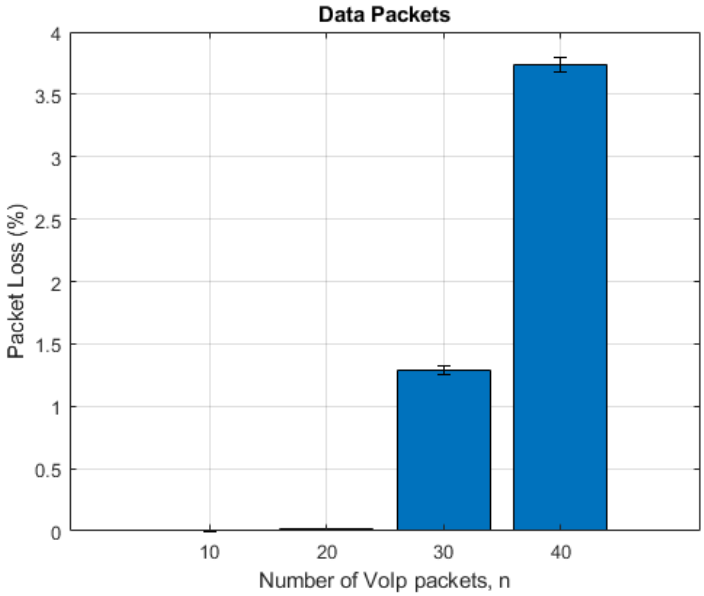

Figure 20 – Data Packet Loss varying the number of VoIp packets

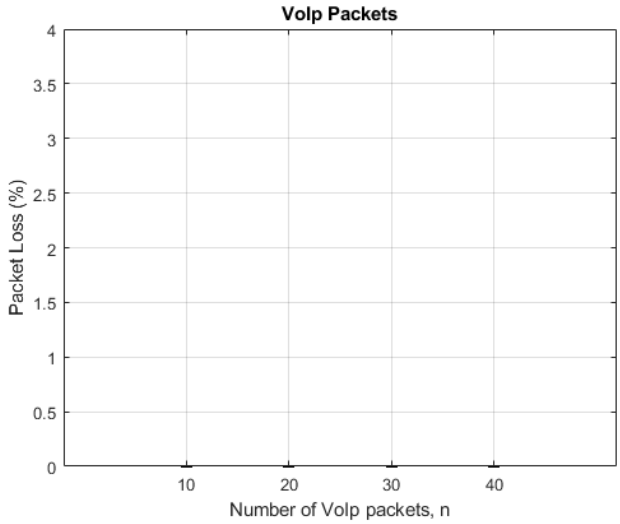

Figure 21 – Volp Packet Loss varying the number of Volp packets

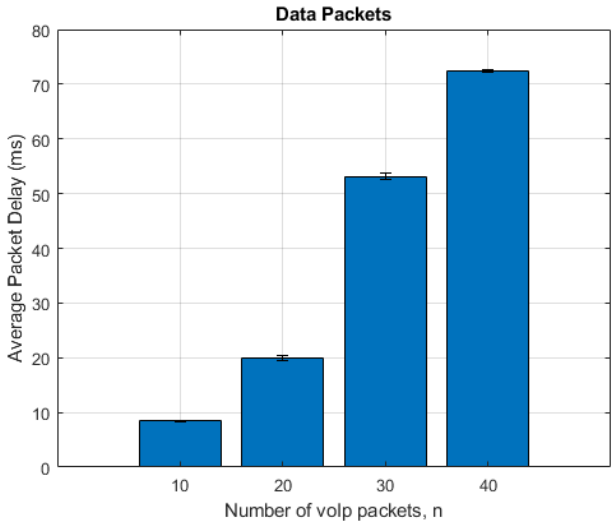

Figure 22 – Average Data Packet Delay varying the number of Volp packets

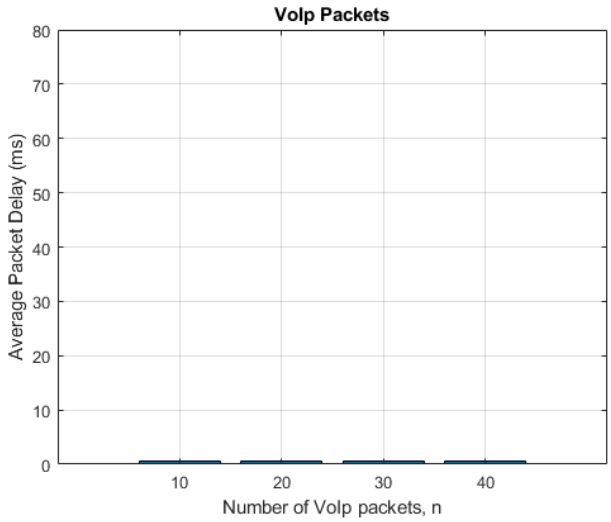

Figure 23 – Average Volp Packet Delay varying the number of Volp packets

The code used to get these results is shown below:

```
% Initializing variables

P = 10000;
conf = 0.9;
lambda = 1800;
C = 10;
f = 1e5;
N = 10;
n = 10;


PacketLossdata = zeros(1,N);
PacketLossvoip = zeros(1,N);
AvgPacketDelaydata = zeros(1,N);
AvgPacketDelayvoip = zeros(1,N);
MaxPacketDelaydata = zeros(1,N);
MaxPacketDelayvoip = zeros(1,N);


Throughput = zeros(1,N);


for i=1:N

[PacketLossdata(i),PacketLossvoip(i),AvgPacketDelaydata(i),AvgPacketDelayvoip
(i),MaxPacketDelaydata(i),MaxPacketDelayvoip(i),Throughput(i)] =
Simulator5(lambda,C,f,P,n);
end


mPLd = mean(PacketLossdata);
mPLv = mean(PacketLossvoip);
mAPDd = mean(AvgPacketDelaydata);
mAPDv = mean(AvgPacketDelayvoip);
term1d = norminv(1-conf/2)*sqrt(var(PacketLossdata)/N);
term1v = norminv(1-conf/2)*sqrt(var(PacketLossvoip)/N);
term2d = norminv(1-conf/2)*sqrt(var(AvgPacketDelaydata)/N);
term2v = norminv(1-conf/2)*sqrt(var(AvgPacketDelayvoip)/N);


% data packets
% Array with all packet losses of data packet delay

RPLd = [mPLd];

% Array with all average data packet delay

RAPDd = [mAPDd];

% Array with all packet losses of data packet delay errors
```

```
Rt1d = [term1d];

% Array with all average data packet delay errors

Rt2d = [term2d];


%voIp packets
% Array with all packet losses of VoIP packet delay

RPLv = [mPLv];

% Array with all average VoIP packet delay

RAPDv = [mAPDv];

% Array with all packet losses of VoIP packet delay errors


Rt1v = [term1v];

% Array with all average VoIP packet delay errors

Rt2v = [term2v];


fprintf('Alinea F, n = 10')
fprintf(['PacketLoss data= %.2e +- %.2e\n' ...
    'PacketLoss voip= %.2e +- %.2e\n' ...
    'AvgPacketDelay data= %.2e +- %.2e\n' ...
    'AvgPacketDelay voip= %.2e +- %.2e\n'], ...
    mPLd, term1d, mPLv, term1v, mAPDd, term2d, mAPDv, term2v)
fprintf('\n\n=========================================================\n\n')


% ------------------------------------------------------------- %


n = 20;


for i=1:N

[PacketLossdata(i),PacketLossvoip(i),AvgPacketDelaydata(i),AvgPacketDelayvoip
(i),MaxPacketDelaydata(i),MaxPacketDelayvoip(i),Throughput(i)] =
Simulator5(lambda,C,f,P,n);
end


mPLd = mean(PacketLossdata);
mPLv = mean(PacketLossvoip);
mAPDd = mean(AvgPacketDelaydata);
mAPDv = mean(AvgPacketDelayvoip);
term1d = norminv(1-conf/2)*sqrt(var(PacketLossdata)/N);
```

```
term1v = norminv(1-conf/2)*sqrt(var(PacketLossvoip)/N);
term2d = norminv(1-conf/2)*sqrt(var(AvgPacketDelaydata)/N);
term2v = norminv(1-conf/2)*sqrt(var(AvgPacketDelayvoip)/N);


% data packets
% Array with all packet losses of data packet delay

RPLd = [RPLd; mPLd];

% Array with all average data packet delay

RAPDd = [RAPDd; mAPDd];

% Array with all packet losses of data packet delay errors

Rt1d = [Rt1d; term1d];

% Array with all average data packet delay errors

Rt2d = [Rt2d; term2d];


%VoIP packets
% Array with all packet losses of VoIP packet delay

RPLv = [RPLv; mPLv];

% Array with all average VoIP packet delay

RAPDv = [RAPDv; mAPDv];

% Array with all packet losses of VoIP packet delay errors

Rt1v = [Rt1v; term1v];

% Array with all average VoIP packet delay errors

Rt2v = [Rt2v; term2v];


fprintf('Alinea F, n = 20')
fprintf(['PacketLoss data= %.2e +- %.2e\n' ...
    'PacketLoss voip= %.2e +- %.2e\n' ...
    'AvgPacketDelay data= %.2e +- %.2e\n' ...
    'AvgPacketDelay voip= %.2e +- %.2e\n'], ...
    mPLd, term1d, mPLv, term1v, mAPDd, term2d, mAPDv, term2v)
fprintf('\n\n=======================================================\n\n')


% ------------------------------------------------------------- %
n = 30;


for i=1:N
```

```
[PacketLossdata(i),PacketLossvoip(i),AvgPacketDelaydata(i),AvgPacketDelayvoip
(i),MaxPacketDelaydata(i),MaxPacketDelayvoip(i),Throughput(i)] =
Simulator5(lambda,C,f,P,n);
end


mPLd = mean(PacketLossdata);
mPLv = mean(PacketLossvoip);
mAPDd = mean(AvgPacketDelaydata);
mAPDv = mean(AvgPacketDelayvoip);
term1d = norminv(1-conf/2)*sqrt(var(PacketLossdata)/N);
term1v = norminv(1-conf/2)*sqrt(var(PacketLossvoip)/N);
term2d = norminv(1-conf/2)*sqrt(var(AvgPacketDelaydata)/N);
term2v = norminv(1-conf/2)*sqrt(var(AvgPacketDelayvoip)/N);


% data packets
% Array with all packet losses of data packet delay

RPLd = [RPLd; mPLd];

% Array with all average data packet delay

RAPDd = [RAPDd; mAPDd];

% Array with all packet losses of data packet delay errors

Rt1d = [Rt1d; term1d];

% Array with all average data packet delay errors

Rt2d = [Rt2d; term2d];


%VoIP packets
% Array with all packet losses of VoIP packet delay

RPLv = [RPLv; mPLv];

% Array with all average VoIP packet delay

RAPDv = [RAPDv; mAPDv];

% Array with all packet losses of VoIP packet delay errors

Rt1v = [Rt1v; term1v];

% Array with all average VoIP packet delay errors

Rt2v = [Rt2v; term2v];


fprintf('Alinea F, n = 30')
```

```
fprintf(['PacketLoss data= %.2e +- %.2e\n' ...
    'PacketLoss voip= %.2e +- %.2e\n' ...
    'AvgPacketDelay data= %.2e +- %.2e\n' ...
    'AvgPacketDelay voip= %.2e +- %.2e\n'], ...
    mPLd, term1d, mPLv, term1v, mAPDd, term2d, mAPDv, term2v)
fprintf('\n\n=========================================================\n\n')


% ------------------------------------------------------------- %
n = 40;

for i=1:N

[PacketLossdata(i),PacketLossvoip(i),AvgPacketDelaydata(i),AvgPacketDelayvoip
(i),MaxPacketDelaydata(i),MaxPacketDelayvoip(i),Throughput(i)] =
Simulator5(lambda,C,f,P,n);
end

mPLd = mean(PacketLossdata);
mPLv = mean(PacketLossvoip);
mAPDd = mean(AvgPacketDelaydata);
mAPDv = mean(AvgPacketDelayvoip);
term1d = norminv(1-conf/2)*sqrt(var(PacketLossdata)/N);
term1v = norminv(1-conf/2)*sqrt(var(PacketLossvoip)/N);
term2d = norminv(1-conf/2)*sqrt(var(AvgPacketDelaydata)/N);
term2v = norminv(1-conf/2)*sqrt(var(AvgPacketDelayvoip)/N);


% data packets
% Array with all packet losses of data packet delay

RPLd = [RPLd; mPLd];

% Array with all average data packet delay

RAPDd = [RAPDd; mAPDd];

% Array with all packet losses of data packet delay errors

Rt1d = [Rt1d; term1d];

% Array with all average data packet delay errors

Rt2d = [Rt2d; term2d];


%VoIP packets
% Array with all packet losses of VoIP packet delay

RPLv = [RPLv; mPLv];

% Array with all average VoIP packet delay

RAPDv = [RAPDv; mAPDv];
```

```
% Array with all packet losses of VoIP packet delay errors

Rt1v = [Rt1v; term1v];

% Array with all average VoIP packet delay errors

Rt2v = [Rt2v; term2v];


% ------------------------------------------------------------- %

fprintf('Alinea F, n = 40')
fprintf(['PacketLoss data= %.2e +- %.2e\n' ...
    'PacketLoss voip= %.2e +- %.2e\n' ...
    'AvgPacketDelay data= %.2e +- %.2e\n' ...
    'AvgPacketDelay voip= %.2e +- %.2e\n'], ...
    mPLd, term1d, mPLv, term1v, mAPDd, term2d, mAPDv, term2v)
fprintf('\n\n=======================================================\n\n')



n = [10 20 30 40];

% data packets
% bar chart of the Packet Loss of Data packets
bar(n, RPLd)
hold on
grid on
ylim([0 4])
title('Data Packets')
xlabel('Number of VoIp packets, n')
ylabel('Packet Loss (%)')
er = errorbar(n, RPLd, Rt1d, Rt1d);
er.Color = [0 0 0];
er.LineStyle = 'none';
hold off


% bar chart of the Average Packet Delay of Data packets
bar(n, RAPDd)
hold on
grid on
ylim([0 80])
title('Data Packets')
xlabel('Number of voIp packets, n')
ylabel('Average Packet Delay (ms)')
er = errorbar(n, RAPDd, Rt2d, Rt2d);
er.Color = [0 0 0];
er.LineStyle = 'none';
hold off
```

```
% voIp packets
% bar chart of the Packet Loss of VoIP packets
bar(n, RPLv)
hold on
grid on
ylim([0 4])
title('VoIp Packets')
xlabel('Number of VoIp packets, n')
ylabel('Packet Loss (%)')
er = errorbar(n, RPLv, Rt1v, Rt1v);
er.Color = [0 0 0];
er.LineStyle = 'none';
hold off


% bar chart of the Average Packet Delay of VoIP packets
bar(n, RAPDv)
hold on
grid on
ylim([0 80])
title('VoIp Packets')
xlabel('Number of VoIp packets, n')
ylabel('Average Packet Delay (ms)')
er = errorbar(n, RAPDv, Rt2v, Rt2v);
er.Color = [0 0 0];
er.LineStyle = 'none';
hold off
```