universidade de aveiro

DEPARTAMENTO DE ELETRÓNICA, TELECOMUNICAÇÕES E INFORMÁTICA

MESTRADO EM ENGENHARIA DE COMPUTADORES E TELEMÁTICA

ANO 2021/2022

MODELAÇÃO E DESEMPENHO DE REDES E SERVIÇOS

MINI-PROJECT 1

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

Tiago Dias (88896)

Rita Amante (89264)

TASK 1

Consider the event driven simulator *Simulator1* used in Task 5 of the Practical Guide.

1.a. Consider the case of *C* = 10 Mbps and *f* = 1.000.000 Bytes. Run *Simulator1* 50 times with a stopping criterion of *P* = 10000 each run and compute the estimated values and the 90% confidence intervals of the average delay performance parameter when λ = 400, 800, 1200, 1600 and 2000 pps. Present the average packet delay results in bar charts with the confidence intervals in error bars1. Justify the results and take conclusions concerning the impact of the packet rate in the obtained average packet delay.

Matlab code

```
1  lambda = [400, 800, 1200, 1600, 2000];
2  C = 10;
3  f = 1000000;
4  P = 10000;
5  N = 50;
6  alfa = 0.1;
7  PL = zeros(1,5);
8  APD = zeros(1,5);
9  MPD = zeros(1,5);
10 TT = zeros(1,5);
11 mediaAPD = zeros(1,5);
12 termAPD = zeros(1,5);
13 for i= 1:length(lambda)
14     for it= 1:N
15         [PL(it), APD(it), MPD(it), TT(it)] = Simulator1(lambda(i),C,f,P);
16     end
17     mediaAPD(i) = mean(APD);
18     termAPD(i) = norminv(1-alfa/2)*sqrt(var(APD)/N);
19 end
20 figure(1);
21 h = bar(lambda,mediaAPD);
22 hold on
23 er = errorbar(lambda,mediaAPD,termAPD);
24 er.Color = [0 0 0];
25 er.LineStyle = 'none';
26 grid on
27 title('Average Packet Delay');
28 xlabel('Lambda (pps)');
29 ylabel('Average packet delay (ms)');
30 hold off
```

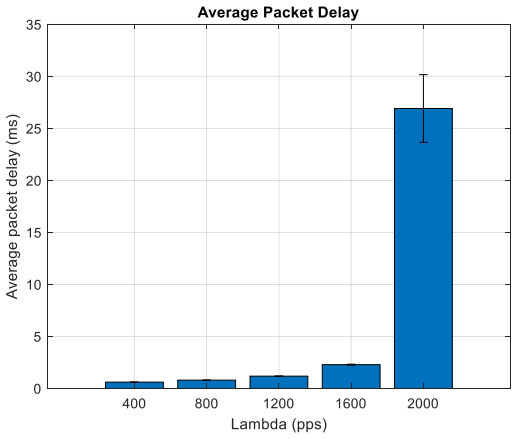
Code analysis

Primeiramente, foram atribuídos os valores às constantes do problema, de acordo com o pedido no enunciado (linhas 1 a 6). De seguida, foram inicializados quatro vetores a zeros (linhas 7 a 10), que irão conter os valores de retorno do *simulator1* e mais dois vetores a zeros (linhas 11 e 12) que irão conter os valores finais para a média e a taxa de ocupação das variáveis pretendidas.

Depois, para cada valor do vetor *lambdas* (linhas 13 a 19), executou-se o simulador N vezes (linhas 14 a 16) e, com os valores lidos do simulador, calculou-se a média do atraso médio de pacotes e a respetiva taxa de ocupação (linhas 17 e 18).

Por fim, desenhou-se o gráfico correspondente ao atraso médio de pacotes (linhas 20 a 30).

Result



Conclusions

Pode-se verificar que o atraso médio de pacotes é exponencialmente positivo e que para valores de, pelo menos 2000 pedidos por segundo, existe uma acentuação significativa, uma vez que o número de pacotes que estão na fila é substancialmente maior ao longo da simulação.

Como a fila de espera é do tipo FIFO, o tempo que leva a transmitir um pacote que entrou por último vai ser exponencialmente maior comparado com valores de lambda mais pequenos, logo o aumento será igualmente exponencial.

É importante salientar que os resultados obtidos podem não ser completamente fiáveis, dado que os intervalos de confiança são muito altos em comparação com os valores e suas variações.

1.b. Consider the case of ʎ = 1800 pps and C = 10 Mbps. Run *Simulator1* 50 times with a stopping criterion of P = 10000 each run and compute the estimated values and the 90% confidence intervals of the average delay and packet loss performance parameters when f = 100.000, 20.000, 10.000 and 2.000 Bytes. Present the average packet delay results in one figure and the average packet loss results in another figure (in both cases, in bar charts with the confidence intervals in error bars). Justify the results and take conclusions concerning the impact of the queue size in the obtained average packet delay and average packet loss.

Matlab code

```
1  lambda = 1800;
2  C = 10;
3  f = [100000, 20000, 10000, 2000];
4  P = 10000;
5  N = 50;
6  alfa = 0.1;
7  PL = zeros(1,4);
8  APD = zeros(1,4);
9  MPD = zeros(1,4);
10 TT = zeros(1,4);
11 mediaPL = zeros(1,4);
12 termPL = zeros(1,4);
13 mediaAPD = zeros(1,4);
14 termAPD = zeros(1,4);
15 for i= 1:length(f)
16     for it= 1:N
17         [PL(it), APD(it), MPD(it), TT(it)] = Simulator1(lambda,C,f(i),P);
18     end
19     mediaPL(i) = mean(PL);
20     termPL(i) = norminv(1-alfa/2)*sqrt(var(PL)/N);
21     mediaAPD(i) = mean(APD);
22     termAPD(i) = norminv(1-alfa/2)*sqrt(var(APD)/N);
23 end
24 figure(1);
25 h = bar(f,mediaPL);
26 hold on
27 er = errorbar(f,mediaPL,termPL);
28 er.Color = [0 0 0];
29 er.LineStyle = 'none';
30 grid on
31 title('Average Packet Loss');
32 xlabel('Queue size (bytes)');
33 ylabel('Average packet loss (%)');
34 hold off
35 figure(2);
36 h = bar(f,mediaAPD);
37 hold on
38 er = errorbar(f,mediaAPD,termAPD);
39 er.Color = [0 0 0];
40 er.LineStyle = 'none';
41 grid on
42 title('Average Packet Delay');
43 xlabel('Queue size (bytes)');
44 ylabel('Average packet delay (ms)');
45 hold off
```

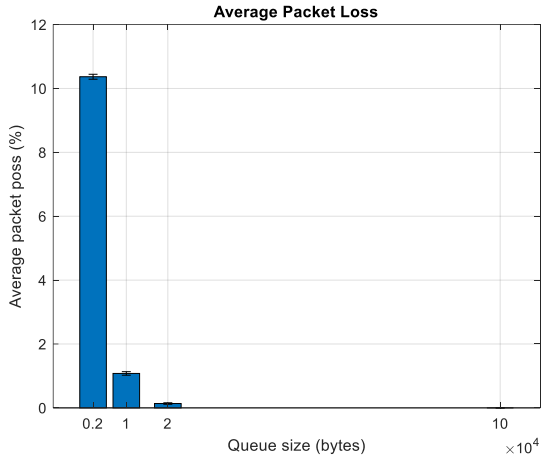
Code analysis

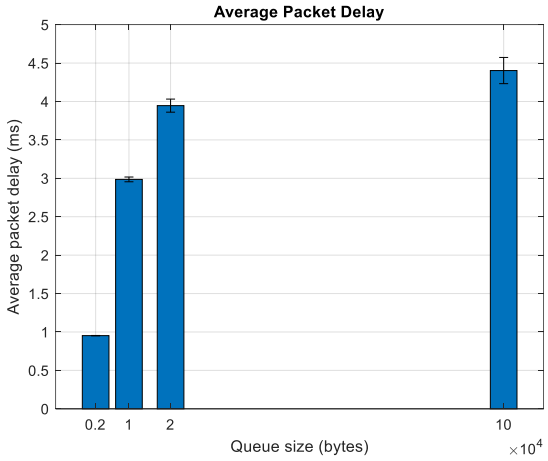
Primeiramente, foram atribuídos os valores às constantes do problema, de acordo com o pedido no enunciado (linhas 1 a 6). De seguida, foram inicializados quatro vetores a zeros (linhas 7 a 10), que irão conter os valores de retorno do *simulator1* e mais quatro vetores a zeros (linhas 11 a 14) que irão conter os valores finais para a média e a taxa de ocupação das variáveis pretendidas.

Depois, para cada valor do vetor *f* (linhas 15 a 23), executou-se o simulador N vezes (linhas 16 a 18) e, com os valores lidos do simulador, calculou-se a média de pacotes perdidos e a respetiva taxa de ocupação (linhas 19 e 20) e calculou-se a média do atraso médio de pacotes e a respetiva taxa de ocupação (linhas 21 e 22).

Por fim, desenharam-se os gráficos correspondentes à perda de pacotes (linhas 24 a 34) e ao atraso médio do pacote (linhas 35 a 45).

Result





Conclusions

É de esperar que haja uma substancial perda de pacotes, pois os valores do tamanho da fila de espera são consideravelmente inferiores ao valor fixo a aproximadamente 10 MBytes. Quanto menor for o tamanho da fila de espera, maior incapacidade de armazenamento de pacotes antes de serem atendidos, tendo que descartar alguns. Isto é provado pela tendência decrescente exponencial do valor proporcionalmente ao tamanho da fila.

Relativamente ao atraso médio de pacotes, apresenta uma tendência crescente com o aumento do tamanho da fila de espera. Como a fila de espera suporta mais pacotes à espera de serem processados, quando um pacote é inserido no fim da fila, permanecerá à espera durante mais tempo que numa fila de tamanho mais reduzido, na qual o pacote será mais rapidamente descartado, o que não possibilita que fique à espera tanto tempo.

1.c. Consider the case of ʎ = 1800 pps and f = 1.000.000 Bytes. Run *Simulator1* 50 times with a stopping criterion of P = 10000 at each run and compute the estimated values and the 90% confidence intervals of the average delay performance parameter when C = 10, 20, 30 and 40 Mbps. Present the average packet delay results in bar charts with the confidence intervals in error bars. Justify the results and take conclusions concerning the impact of the link capacity in the obtained average packet delay.

Matlab code

```
1  lambda = 1800;
2  C = [10, 20, 30, 40];
3  f = 1000000;
4  P = 10000;
5  N = 50;
6  alfa = 0.1;
7  PL = zeros(1,4);
8  APD = zeros(1,4);
9  MPD = zeros(1,4);
10 TT = zeros(1,4);
11 mediaAPD = zeros(1,4);
12 termAPD = zeros(1,4);
13 for i= 1:length(C)
14     for it= 1:N
15         [PL(it), APD(it), MPD(it), TT(it)] = Simulator1(lambda,C(i),f,P);
16     end
17     mediaAPD(i) = mean(APD);
18     termAPD(i) = norminv(1-alfa/2)*sqrt(var(APD)/N);
19 end
20 figure(4);
21 h = bar(C,mediaAPD);
22 hold on
23 er = errorbar(C,mediaAPD,termAPD);
24 er.Color = [0 0 0];
25 er.LineStyle = 'none';
26 grid on
27 title('Average Packet Delay');
28 xlabel('Link bandwidth (Mbps)');
29 ylabel('Average packet delay (ms)');
30 hold off
```

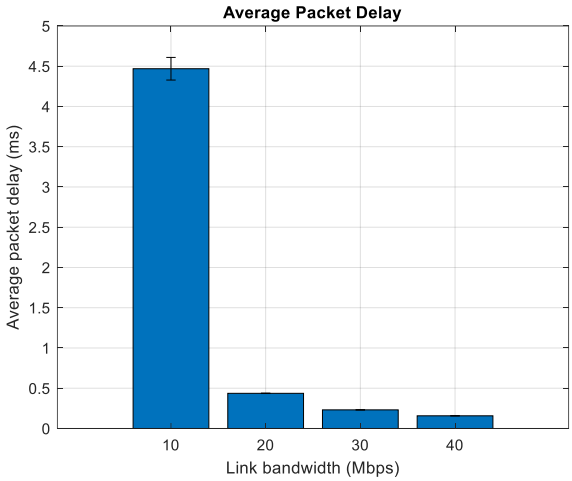
Code analysis

Primeiramente, foram atribuídos os valores às constantes do problema, de acordo com o pedido no enunciado (linhas 1 a 6). De seguida, foram inicializados quatro vetores a zeros (linhas 7 a 10), que irão conter os valores de retorno do *simulator1* e mais dois vetores a zeros (linhas 11 e 12) que irão conter os valores finais para a média e a taxa de ocupação das variáveis pretendidas.

Depois, para cada valor do vetor *f* (linhas 13 a 19), executou-se o simulador N vezes (linhas 14 a 16) e, com os valores lidos do simulador, calculou-se a média de pacotes perdidos e a respetiva taxa de ocupação (linhas 17 e 18).

Por fim, desenhou-se o gráfico correspondente ao atraso médio do pacote (linhas 20 a 30).

Result



Conclusions

É de esperar que à medida que a capacidade da ligação aumenta, o atraso médio de pacotes diminua exponencialmente. Uma vez que a taxa de chegada de pacotes é constante, a fila de espera é considerada infinita, é esperado uma diminuição exponencial com o aumento da capacidade da ligação cuja sua função é atender cada pacote que chega ao sistema. Logo, quanto maior a capacidade da ligação, maior capacidade de processamento de mais quantidade de bits.

1.d. Consider that the system is modelled by a M/G/1 queueing model. Determine the theoretical values of the average packet delay using the M/G/1 model for all cases of 1.c. Compare the theoretical values with the simulation results of experiments 1.c and take conclusions.

Matlab code

```
1  lambda = 1800;
2  C = [10, 20, 30, 40];
3  f = 1000000;
4  P = 10000;
5  N = 50;
6  alfa = 0.1;
7  PL = zeros(1,4);
8  APD = zeros(1,4);
9  MPD = zeros(1,4);
10 TT = zeros(1,4);
11 size = MeanPacketSize();
12 sim_APD = zeros(1,4);
13 the_APD = zeros(1,4);
14 the_APD_2 = zeros(1,4);
15 for i = 1:length(C)
16     for it= 1:N
17         [PL(it), APD(it), MPD(it), TT(it)] = Simulator1(lambda,C(i),f,P);
18         the_APD_2 = TheoAvgDelayMG1(lambda,C(i));
19     end
20     sim_APD(i) = mean(APD);
21     the_APD(i) = mean(the_APD_2);
22 end
23 figure(5);
24 h = bar(C,[sim_APD; the_APD]);
25 hold on
26 grid on
27 title("Average Packet Deplay (MG1 queueing model)");
28 legend('Simulation','Theoretical', 'location', 'northeast')
29 xlabel('Link bandwidth (Mbps)');
30 ylabel('Average packet delay (ms)');
31 hold off
```

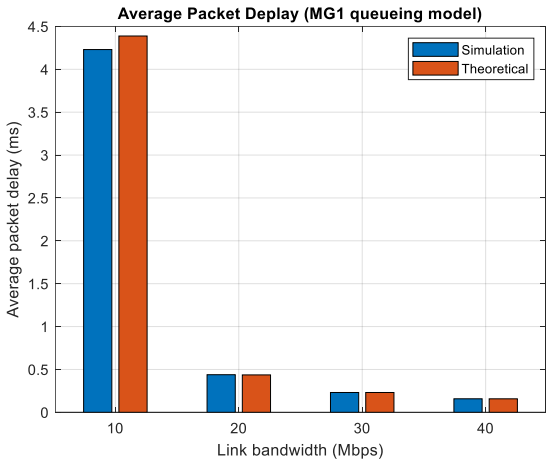
Code analysis

Primeiramente, foram atribuídos os valores às constantes do problema, de acordo com o pedido no enunciado (linhas 1 a 6). De seguida, foram inicializados quatro vetores a zeros (linhas 7 a 10), que irão conter os valores de retorno do *simulator1*. Na linha 11 foi definido o tamanho dos pacotes, com auxílio da função *MeanPacketSize.* Nas linhas 12 a 14 foram inicializados os vetores necessários para armazenar os valores que se pretendem reter da simulação.

Depois, para cada valor do vetor C (linhas 15 a 22), executou-se o simulador N vezes (linhas 16 a 19) e, com os valores lidos do simulador, calculou-se os valores teóricos para o APD no modelo M/G/1 com o auxílio da função *TheoAvgDelayMG1* e os valores de simulação para o APD no modelo M/G/1.

Por fim, desenhou-se o gráfico correspondente ao atraso médio do pacote, apresentando os valores de simulação e teóricos (linhas 23 a 31).

Result



Conclusions

Os valores teóricos de atraso médio de pacotes mantêm-se semelhantes aos valores simulados para capacidades de ligação maiores. No entanto, quando a capacidade da ligação é menor ou igual a 10Mbps, os valores simulados são menores que os valores teóricos. É notório que há um decréscimo exponencial à medida que a capacidade da ligação aumente.

1.e. Develop a new version of *Simulator1* to estimate 3 additional performance parameters: the average packet delay of the packets of size 64, 110 and 1518 Bytes, respectively. Consider the case of λ = 1800 pps and f = 1.000.000. Run the new version of Simulator1 50 times with a stopping criterion of P = 10000 at each run and compute the estimated values and the 90% confidence intervals of the 3 new average delay performance parameters when C = 10, 20, 50 and 100 Mbps. Present the average packet delay results in bar charts with the confidence intervals in error bars. Justify these results and the differences between them and the results of 1.c. Take conclusions concerning the impact of the link capacity in the obtained average packet delay of packets with different sizes.

Matlab code

```
1  lambda = 1800;
2  C = [10, 20, 30, 40];
3  f = 1000000;
4  P = 10000;
5  N = 50;
6  alfa = 0.1;
7  PL = zeros(1,4);
8  APD = zeros(1,4);
9  MPD = zeros(1,4);
10 TT = zeros(1,4);
11 APD64 = zeros(1,4);
12 APD110 = zeros(1,4);
13 APD1518 = zeros(1,4);
14 mediaAPD64 = zeros(1,4);
15 mediaAPD110 = zeros(1,4);
16 mediaAPD1518 = zeros(1,4);
17 % 64, 110, 1518 bytes error bars (NOT SHOWN IN THE GRAPH)
18 termAPD64 = zeros(1,4);
19 termAPD110 = zeros(1,4);
20 termAPD1518 = zeros(1,4);
21 for i= 1:length(C)
22     for it= 1:N
23         [PL(it), APD(it), MPD(it), TT(it), APD64(it), APD110(it), APD1518(it)] =
24 Simulator1New(lambda,C(i),f,P);
25     end
26     mediaAPD64(i) = mean(APD64);
27     mediaAPD110(i) = mean(APD110);
28     mediaAPD1518(i) = mean(APD1518);
29     % NOT SHOWN IN THE GRAPH
30     termAPD64(i) = norminv(1-alfa/2)*sqrt(var(APD64)/N);
31     termAPD110(i) = norminv(1-alfa/2)*sqrt(var(APD110)/N);
32     termAPD1518(i) = norminv(1-alfa/2)*sqrt(var(APD1518)/N);
33 end
34 figure(5);
35 h = bar(C, [mediaAPD64; mediaAPD110; mediaAPD1518]);
36 hold on
37 grid on
38 title('Average Packet Delay');
39 legend('64 Bytes','110 Bytes', '1518 Bytes', Location='northeast')
40 xlabel('Link bandwidth (Mbps)');
41 ylabel('Average packet delay (ms)');
42 hold off
```

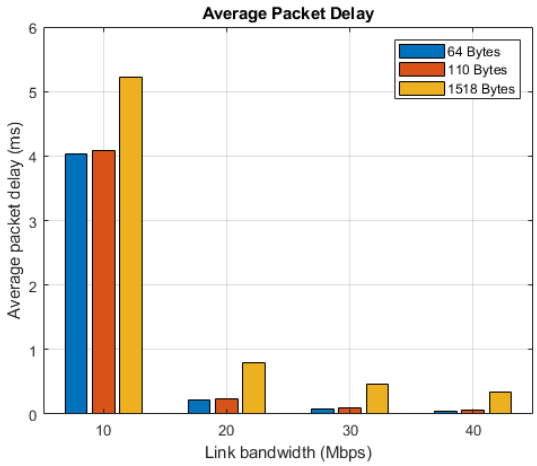
Code analysis

Primeiramente, foram atribuídos os valores às constantes do problema, de acordo com o pedido no enunciado (linhas 1 a 6). De seguida, foram inicializados sete vetores a zeros (linhas 7 a 13), que irão conter os valores de retorno do *Simulator1New* e mais seis vetores a zeros (linhas 14 a 20) que irão conter os valores finais para a média e o intervalo de confiança das barras de erro. Depois, para cada valor do vetor

C (linhas 21 a 3), executou-se o simulador N vezes (linhas 22 a 25) e, com os valores lidos do simulador, calculou-se a média do atraso dos pacotes com 64, 110 e 1518 bytes e a respetiva taxa de ocupação.

Por fim, desenhou-se o gráfico correspondente ao atraso médio dos respetivos pacotes (linhas 34 a 42).

Result



Conclusions

Seria de esperar que a relação entre o tamanho dos pacotes e o seu atraso médio fosse diretamente proporcional, ou seja, à medida que o tamanho dos pacotes aumenta, aumenta também o seu atraso médio. Foi concluído anteriormente no exercício 1c) que à medida que a capacidade da ligação aumenta, o atraso médio de pacotes diminui exponencialmente. Essa conclusão é também aplicada a este exercício em toda a sua íntegra.

TASK 2

Consider the event driven simulators *Simulator3* and *Simulator4* developed in Task 7 of the Practical Guide.

2.a. Consider the case of ʎ = 1500 pps, C = 10 Mbps and f = 1.000.000 Bytes. Run *Simulator3* 50 times with a stopping criterion of P = 10000 each run and compute the estimated values and the 90% confidence intervals of the average delay performance parameter of data packets and VoIP packets when n = 10, 20, 30 and 40 VoIP flows. Present the average data packet delay results in one figure and the average VoIP packet delay results in another figure (in both cases, in bar charts with the confidence intervals in error bars). Justify the results and take conclusions concerning the impact of the number of VoIP flows in the obtained average packet delay of each service when both services (data and VoIP) are statistically multiplexed in a single FIFO queue.

Matlab code

```
lambda = 1500;
C = 10;
f = 1000000;
P = 10000;
N = 50;
n = [10, 20, 30, 40];
alfa = 0.1;
PLdata = zeros(1,4);
APDdata = zeros(1,4);
MPDdata = zeros(1,4);
PLvoip = zeros(1,4);
APDvoip = zeros(1,4);
MPDvoip = zeros(1,4);
TT = zeros(1,4);
mediaAPDdata = zeros(1,4);
termAPDdata = zeros(1,4);
mediaAPDvoip = zeros(1,4);
termAPDvoip = zeros(1,4);
for i= 1:length(n)
    for it= 1:N
        [PLdata(it), APDdata(it), MPDdata(it), TT(it), PLvoip(it), APDvoip(it), MPDvoip(it)] =
Simulator3(lambda,C,f,P,n(i));
    end
    mediaAPDdata(i) = mean(APDdata);
    termAPDdata(i) = norminv(1-alfa/2)*sqrt(var(APDdata)/N);
    mediaAPDvoip(i) = mean(APDvoip);
    termAPDvoip(i) = norminv(1-alfa/2)*sqrt(var(APDvoip)/N);
end
figure(1);
h = bar(n,mediaAPDdata);
hold on
er = errorbar(n,mediaAPDdata,termAPDdata);
er.Color = [0 0 0];
er.LineStyle = 'none';
grid on
title('Average Data Packet Delay');
xlabel('n (number of Data packets flows)');
ylabel('Average Data packet delay (ms)');
hold off
figure(2);
h = bar(n,mediaAPDvoip);
hold on
er = errorbar(n,mediaAPDvoip,termAPDvoip);
er.Color = [0 0 0];
er.LineStyle = 'none';
grid on
title('Average VoIP Packet Delay');
xlabel('n (number of VoIP packets flows)');
```

```
49  ylabel('Average VoIP packet delay (ms)');
50  hold off
```

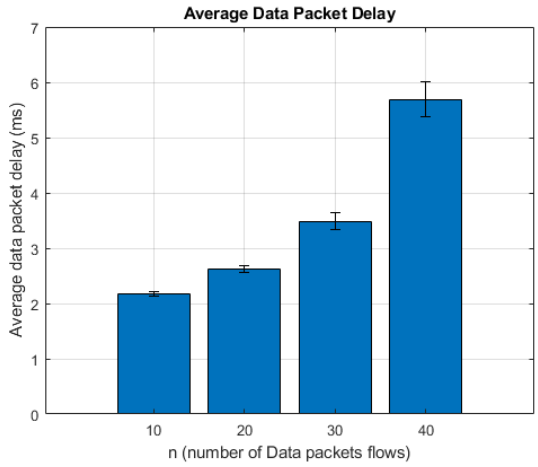
Code analysis

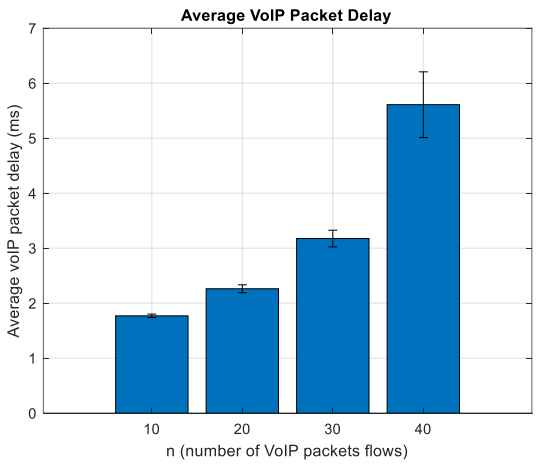
Primeiramente, foram atribuídos os valores às constantes do problema, de acordo com o pedido no enunciado (linhas 1 a 7). De seguida, foram inicializados sete vetores a zeros (linhas 8 a 14), que irão conter os valores de retorno do *simulator3* e mais quatro vetores a zeros (linhas 15 a 18) que irão conter os valores finais para a média e a taxa de ocupação das variáveis pretendidas.

Depois, para cada valor do vetor *n* (linhas 19 a 28), executou-se o simulador N vezes (linhas 20 a 23) e, com os valores lidos do simulador, calculou-se a média do atraso médio de pacotes e a respetiva taxa de ocupação (linhas 24 e 27), para dados e para VoIP.

Por fim, desenharam-se os gráficos correspondentes ao atraso médio de pacotes de dados (linhas 29 a 39) e ao atraso médio de pacotes de VoIP (linhas 40 a 50).

Result





Conclusions

Olhando para os resultados obtidos, conseguimos concluir que o atraso médio dos pacotes, independente do tipo, aumenta consoante o aumento do número de pacotes enviados. Não existe grande diferença nos resultados obtidos em relação á diferença de tipo de pacotes. Concluindo, assim, que o tipo de pacotes não influencia o seu atraso médio. É de referir ainda que ambos os tipos de pacotes têm a mesma prioridade, sendo tratados de forma igual.

2.b. Repeat experiment 2.a but now with *Simulator4*. Justify these results and the differences between them and the results of 2.a. Take conclusions concerning the impact of the number of VoIP flows in the obtained average packet delay of each service when VoIP service is supported with a priority which is higher than the data service.

Matlab code

```
lambda = 1500;
C = 10;
f = 1000000;
P = 10000;
N = 50;
n = [10, 20, 30, 40];
alfa = 0.1;
PLdata = zeros(1,4);
APDdata = zeros(1,4);
MPDdata = zeros(1,4);
PLvoip = zeros(1,4);
APDvoip = zeros(1,4);
MPDvoip = zeros(1,4);
TT = zeros(1,4);
mediaAPDdata = zeros(1,4);
termAPDdata = zeros(1,4);
mediaAPDvoip = zeros(1,4);
termAPDvoip = zeros(1,4);
for i= 1:length(n)
    for it= 1:N
        [PLdata(it), APDdata(it), MPDdata(it), TT(it), PLvoip(it), APDvoip(it), MPDvoip(it)] =
Simulator4(lambda,C,f,P,n(i));
    end
    mediaAPDdata(i) = mean(APDdata);
    termAPDdata(i) = norminv(1-alfa/2)*sqrt(var(APDdata)/N);
    mediaAPDvoip(i) = mean(APDvoip);
    termAPDvoip(i) = norminv(1-alfa/2)*sqrt(var(APDvoip)/N);
end
figure(3);
h = bar(n,mediaAPDdata);
hold on
er = errorbar(n,mediaAPDdata,termAPDdata);
er.Color = [0 0 0];
er.LineStyle = 'none';
grid on
title('Average Data Packet Delay');
xlabel('n (number of Data packets flows)');
ylabel('Average Data packet delay (ms)');
hold off
figure(4);
h = bar(n,mediaAPDvoip);
hold on
er = errorbar(n,mediaAPDvoip,termAPDvoip);
er.Color = [0 0 0];
er.LineStyle = 'none';
grid on
title('Average VoIP Packet Delay');
xlabel('n (number of VoIP packets flows)');
ylabel('Average VoIP packet delay (ms)');
hold off
```

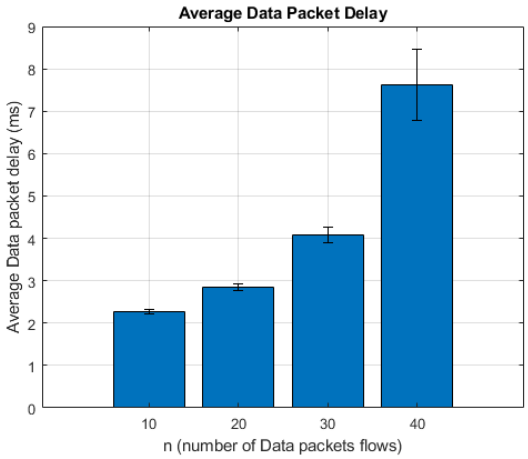
Code analysis

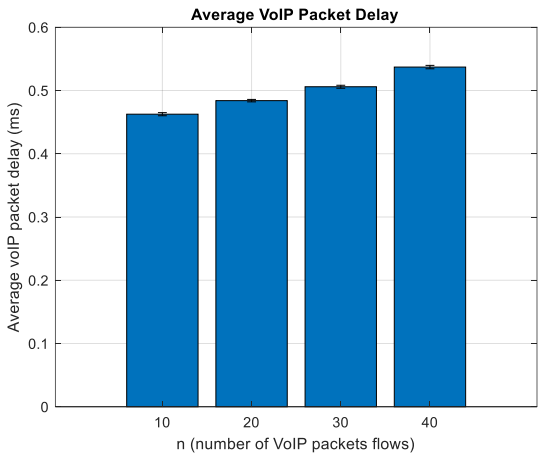
Primeiramente, foram atribuídos os valores às constantes do problema, de acordo com o pedido no enunciado (linhas 1 a 7). De seguida, foram inicializados sete vetores a zeros (linhas 8 a 14), que irão conter os valores de retorno do *simulator4* e mais quatro vetores a zeros (linhas 15 a 18) que irão conter os valores finais para a média e a taxa de ocupação das variáveis pretendidas.

Depois, para cada valor do vetor *n* (linhas 19 a 28), executou-se o simulador N vezes (linhas 20 a 23) e, com os valores lidos do simulador, calculou-se a média do atraso médio de pacotes e a respetiva taxa de ocupação (linhas 24 e 27), para dados e para VoIP.

Por fim, desenharam-se os gráficos correspondentes ao atraso médio de pacotes de dados (linhas 29 a 39) e ao atraso médio de pacotes de VoIP (linhas 40 a 50).

Result





Conclusions

Fazendo uma comparação direta com o exercício 2a) é possível analisar que os pacotes VoIP têm muito menos tempo de atraso em relação aos resultados obtidos anteriormente. Isto deve-se ao facto de neste exercício os pacotes VoIP terem uma prioridade mais alta em relação aos pacotes de dados. Por sua vez existe um ligeiro aumento no atraso médio dos pacotes de dados, visto estes serem colocados em fila de espera até que os pacotes mais prioritários (pacotes VoIP) sejam despachados.

2.c. Consider that the system is modelled by a M/G/1 queueing model with priorities. Determine the theoretical values of the average data packet delay and average VoIP packet delay using the M/G/1 model for all cases of 2.b. Compare the theoretical values with the simulation results of experiments 2.b and take conclusions.

Matlab code

```
1  lambda = 1500;
2  C = 10;
3  f = 1000000;
4  P = 10000;
5  N = 50;
6  n = [10, 20, 30, 40];
7  alfa = 0.1;
8  PLdata = zeros(1,4);
9  APDdata = zeros(1,4);
10 MPDdata = zeros(1,4);
11 PLvoip = zeros(1,4);
12 APDvoip = zeros(1,4);
13 MPDvoip = zeros(1,4);
14 TT = zeros(1,4);
15 sim_mediaAPDdata = zeros(1,4);
16 sim_mediaAPDvoip = zeros(1,4);
17 the_mediaAPDdata = zeros(1,4);
18 the_mediaAPDvoip = zeros(1,4);
19 the_mediaAPDdata_2 = zeros(1,4);
20 the_mediaAPDvoip_2 = zeros(1,4);
21 for i= 1:length(n)
22     for it= 1:N
23         [PLdata(it), APDdata(it), MPDdata(it), TT(it), PLvoip(it), APDvoip(it), MPDvoip(it)] =
24 Simulator4(lambda,C,f,P,n(i));
25         [the_mediaAPDvoip_2, the_mediaAPDdata_2] =  TheoAvgDelayMG1_priorities(lambda, C,
26 n(i));
27     end
28     sim_mediaAPDdata(i) = mean(APDdata);
29     sim_mediaAPDvoip(i) = mean(APDvoip);
30     the_mediaAPDdata(i) = mean(the_mediaAPDdata_2);
31     the_mediaAPDvoip(i) = mean(the_mediaAPDvoip_2);
32 end
33 figure(3);
34 h = bar(n,[sim_mediaAPDdata; the_mediaAPDdata]);
35 hold on
36 grid on
37 title('Average Data Packet Delay');
38 legend('Simulation','Theoretical', 'location', 'northwest');
39 xlabel('n (number of Data packets flows)');
40 ylabel('Average Data packet delay (ms)');
41 hold off
42 figure(4);
43 h = bar(n,[sim_mediaAPDvoip; the_mediaAPDvoip]);
44 hold on
45 grid on
46 title('Average VoIP Packet Delay');
47 legend('Simulation','Theoretical', 'location', 'northwest');
48 xlabel('n (number of VoIP packets flows))');
49 ylabel('Average VoIP packet delay (ms)');
50 hold off
```

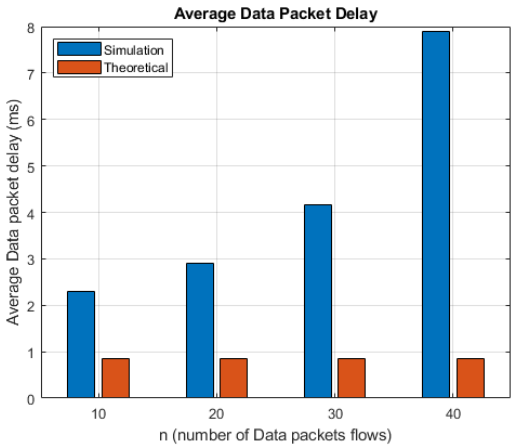
Code analysis

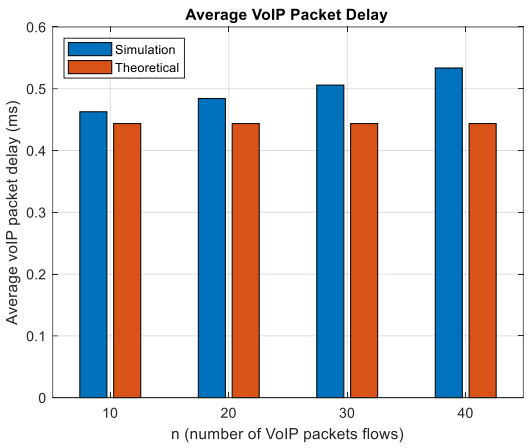
Primeiramente, foram atribuídos os valores às constantes do problema, de acordo com o pedido no enunciado (linhas 1 a 7). De seguida, foram inicializados seis vetores a zeros (linhas 8 a 14), que irão conter os valores de retorno do *simulator4* e mais seis vetores a zeros (linhas 15 a 20), que irão conter os valores de simulação e os valores teóricos do modelo M/G/1.

Depois, para cada valor do vetor *n* (linhas 21 a 32), executou-se o simulador N vezes (linhas 22 a 27) e, com os valores lidos do simulador, calculou-se os valores teóricos para o APD no modelo M/G/1 com o auxílio da função *TheoAvgDelayMG1_priorities* e os valores de simulação para o APD no modelo M/G/1.

Por fim, desenharam-se os gráficos correspondentes ao atraso médio de pacotes de dados (linhas 33 a 41) e ao atraso médio de pacotes VoIP (linhas 42 a 50), apresentando os valores de simulação e teóricos.

Result





Conclusions

No caso dos pacotes de dados, o valor teórico calculado é bastante inferior ao valor simulado. Já no caso dos pacotes VoIP os valores são mais próximos do valor teórico. Os valores simulados de ambos os pacotes possuem um atraso maior devido ao tempo de espera na fila de cada modelo.

2.d. Consider the case of λ = 1500 pps, C = 10 Mbps and f = 10.000 Bytes. Run *Simulator3* 50 times with a stopping criterion of P = 10000 each run and compute the estimated values and the 90% confidence intervals of the average delay and packet loss performance parameters of data packets and VoIP packets when n = 10, 20, 30 and 40 VoIP flows. Present the results of each of the 4 performance parameters (average data packet delay, average VoIP packet delay, data packet loss and VoIP packet loss) in different figures (in all cases, in bar charts with the confidence intervals in error bars). Justify the results and take conclusions concerning the impact of the number of VoIP flows in the obtained average packet delay and packet loss of each service when both services (data and VoIP) are statistically multiplexed in a single FIFO queue of small size.

Matlab code

```
lambda = 1500;
C = 10;
f = 10000;
P = 10000;
N = 50;
n = [10, 20, 30, 40];
alfa = 0.1;
PLdata = zeros(1,4);
APDdata = zeros(1,4);
MPDdata = zeros(1,4);
PLvoip = zeros(1,4);
APDvoip = zeros(1,4);
MPDvoip = zeros(1,4);
TT = zeros(1,4);
mediaAPDdata = zeros(1,4);
termAPDdata = zeros(1,4);
mediaAPDvoip = zeros(1,4);
termAPDvoip = zeros(1,4);
mediaPLdata = zeros(1,4);
termPLdata = zeros(1,4);
mediaPLvoip = zeros(1,4);
termPLvoip = zeros(1,4);
for i= 1:length(n)
    for it= 1:N
        [PLdata(it), APDdata(it), MPDdata(it), TT(it), PLvoip(it), APDvoip(it), MPDvoip(it)] =
Simulator3(lambda,C,f,P,n(i));
    end
    mediaPLdata(i) = mean(PLdata);
    termPLdata(i) = norminv(1-alfa/2)*sqrt(var(PLdata)/N);
    mediaPLvoip(i) = mean(PLvoip);
    termPLvoip(i) = norminv(1-alfa/2)*sqrt(var(PLvoip)/N);
    mediaAPDdata(i) = mean(APDdata);
    termAPDdata(i) = norminv(1-alfa/2)*sqrt(var(APDdata)/N);
    mediaAPDvoip(i) = mean(APDvoip);
    termAPDvoip(i) = norminv(1-alfa/2)*sqrt(var(APDvoip)/N);
end
figure(5);
h = bar(n,mediaPLdata);
hold on
er = errorbar(n,mediaPLdata,termPLdata);
er.Color = [0 0 0];
er.LineStyle = 'none';
grid on
title('Data Packet Loss');
xlabel('n (number of Data packets flows)');
ylabel('Data packet loss (%)');
hold off
figure(6);
h = bar(n,mediaPLvoip);
hold on
er = errorbar(n,mediaPLvoip,termPLvoip);
er.Color = [0 0 0];
```

```
53  er.LineStyle = 'none';
54  grid on
55  title('VoIP Packet Loss');
56  xlabel(' n (number of VoIP packets flows)');
57  ylabel(' VoIP packet loss (%)');
58  hold off
59  figure(7);
60  h = bar(n,mediaAPDdata);
61  hold on
62  er = errorbar(n,mediaAPDdata,termAPDdata);
63  er.Color = [0 0 0];
64  er.LineStyle = 'none';
65  grid on
66  title('Average Data Packet Delay');
67  xlabel('n (number of Data packets flows)');
68  ylabel('Average Data packet delay (ms)');
69  hold off
70  figure(8);
71  h = bar(n,mediaAPDvoip);
72  hold on
73  er = errorbar(n,mediaAPDvoip,termAPDvoip);
74  er.Color = [0 0 0];
75  er.LineStyle = 'none';
76  grid on
77  title('Average VoIP Packet Delay');
78  xlabel('n (number of VoIP packets flows)');
79  ylabel('Average VoIP packet delay (ms)');
80  hold off
```

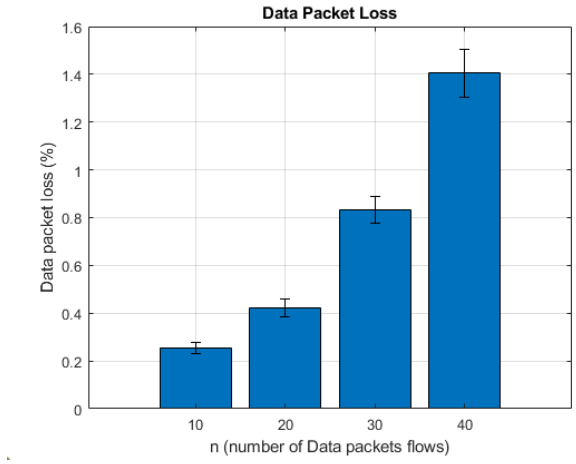
Code analysis

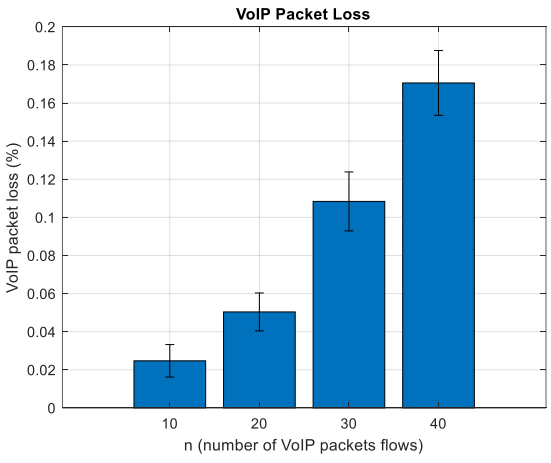
Primeiramente, foram atribuídos os valores às constantes do problema, de acordo com o pedido no enunciado (linhas 1 a 7). De seguida, foram inicializados sete vetores a zeros (linhas 8 a 14), que irão conter os valores de retorno do *simulator3* e mais oito vetores a zeros (linhas 15 a 22) que irão conter os valores finais para a média e a taxa de ocupação das variáveis pretendidas.

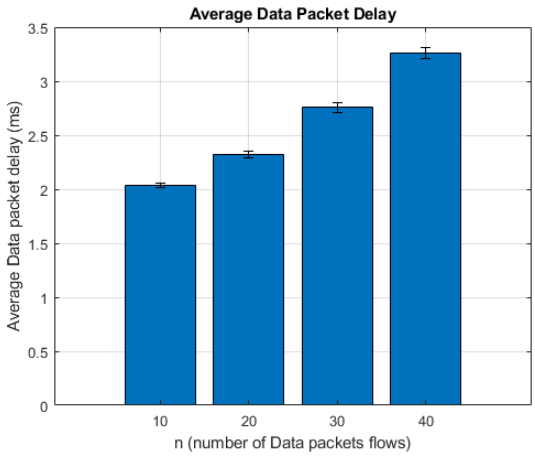
Depois, para cada valor do vetor *n* (linhas 23 a 36), executou-se o simulador N vezes (linhas 24 a 27) e, com os valores lidos do simulador, calculou-se os pacotes de dados perdidos e de VoIP e as respetivas taxas de ocupação (linhas 28 a 31) e a média do atraso médio de pacotes de dados e de VoIP e a respetiva taxa de ocupação (linhas 32 a 35).

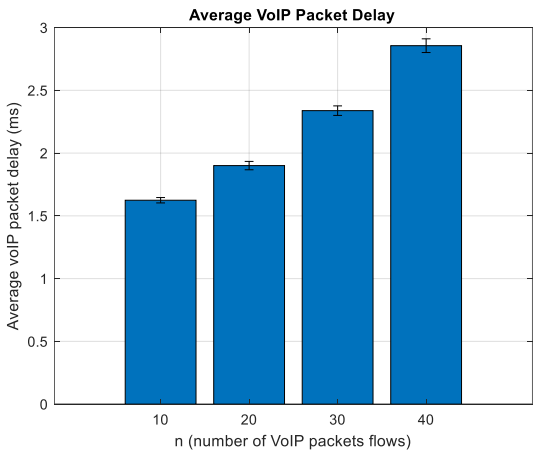
Por fim, desenharam-se os gráficos correspondentes aos pacotes de dados perdidos (linhas 37 a 47), aos pacotes de VoIP perdidos (linhas 48 a 58), ao atraso médio de pacotes de dados (linhas 59 a 69) e ao atraso médio de pacotes de VoIP (linhas 70 a 80).

Result









Conclusions

Relativamente à perda de pacotes, verifica-se um aumento exponencial à medida que o número de fluxos de pacotes VoIP aumenta. Uma vez que a fila de espera tem um tamanho finito, vão existir sempre perdas de pacotes, pois poderá haver pacotes que já não cabem na fila de espera.

Quando a fila estiver a atingir a sua capacidade máxima e não seja possível inserir um pacote de dados, pois o tamanho dos pacotes de dados pode ser muito superior ao tamanho dos pacotes VoIP, é inserido um pacote VoIP em vez de um pacote de dados, aumentando o valor de perda de pacotes de dados.

Após uma análise dos gráficos, é notório que os pacotes VoIP sobrem um atraso menor que os pacotes de dados, isto deve-se a vários motivos:

- O tamanho dos pacotes VoIP é menor que o tamanho dos pacotes de dados;
- De acordo com a alínea 1e), pacotes com tamanhos maiores, sofrem atrasos superiores em comparação com pacotes de tamanho menores;
- Existem mais fluxos VoIP, o que faz com que os pacotes de dados levem mais tempo na fila de espera.

Como o número de fluxos de dados VoIP aumentam, a quantidade de pacotes na fila de espera também aumenta e consecutivamente também aumenta o tempo de espera de cada pacote, tanto para pacotes de dados, como para pacotes VoIP.

2.e. Repeat experiment 2.d but now with *Simulator4*. Justify these results and the differences between them and the results of 2.d. Take conclusions concerning the impact of the number of VoIP flows in the obtained average packet delay and packet loss of each service when VoIP service is supported with a priority which is higher than the data service and the queue is of small size.

Matlab code

```
lambda = 1500;
C = 10;
f = 10000;
P = 10000;
N = 50;
n = [10, 20, 30, 40];
alfa = 0.1;
PLdata = zeros(1,4);
APDdata = zeros(1,4);
MPDdata = zeros(1,4);
PLvoip = zeros(1,4);
APDvoip = zeros(1,4);
MPDvoip = zeros(1,4);
TT = zeros(1,4);
mediaAPDdata = zeros(1,4);
termAPDdata = zeros(1,4);
mediaAPDvoip = zeros(1,4);
termAPDvoip = zeros(1,4);
mediaPLdata = zeros(1,4);
termPLdata = zeros(1,4);
mediaPLvoip = zeros(1,4);
termPLvoip = zeros(1,4);
for i= 1:length(n)
    for it= 1:N
        [PLdata(it), APDdata(it), MPDdata(it), TT(it), PLvoip(it), APDvoip(it), MPDvoip(it)] =
Simulator4(lambda,C,f,P,n(i));
    end
    mediaPLdata(i) = mean(PLdata);
    termPLdata(i) = norminv(1-alfa/2)*sqrt(var(PLdata)/N);
    mediaPLvoip(i) = mean(PLvoip);
    termPLvoip(i) = norminv(1-alfa/2)*sqrt(var(PLvoip)/N);
    mediaAPDdata(i) = mean(APDdata);
    termAPDdata(i) = norminv(1-alfa/2)*sqrt(var(APDdata)/N);
    mediaAPDvoip(i) = mean(APDvoip);
    termAPDvoip(i) = norminv(1-alfa/2)*sqrt(var(APDvoip)/N);
end
figure(9);
h = bar(n,mediaPLdata);
hold on
er = errorbar(n,mediaPLdata,termPLdata);
er.Color = [0 0 0];
er.LineStyle = 'none';
grid on
title('Data Packet Loss');
xlabel('n (number of Data packets flows)');
ylabel('Data packet loss (%)');
hold off
figure(10);
h = bar(n,mediaPLvoip);
hold on
er = errorbar(n,mediaPLvoip,termPLvoip);
er.Color = [0 0 0];
er.LineStyle = 'none';
grid on
title('VoIP Packet Loss');
xlabel('n (number of VoIP packets flows)');
ylabel('VoIP packet loss (%)');
hold off
```

```
59  figure(11);
60  h = bar(n,mediaAPDdata);
61  hold on
62  er = errorbar(n,mediaAPDdata,termAPDdata);
63  er.Color = [0 0 0];
64  er.LineStyle = 'none';
65  grid on
66  title('Average Data Packet Delay');
67  xlabel('n (number of Data packets flows)');
68  ylabel('Average data packet delay (ms)');
69  hold off
70  figure(12);
71  h = bar(n,mediaAPDvoip);
72  hold on
73  er = errorbar(n,mediaAPDvoip,termAPDvoip);
74  er.Color = [0 0 0];
75  er.LineStyle = 'none';
76  grid on
77  title('Average VoIP Packet Delay');
78  xlabel('n (number of VoIP packets flows)');
79  ylabel('Average VoIP packet delay (ms)');
80  hold off
81
```

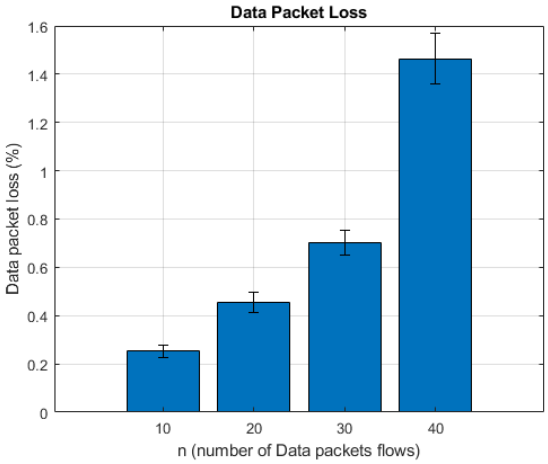
Code analysis

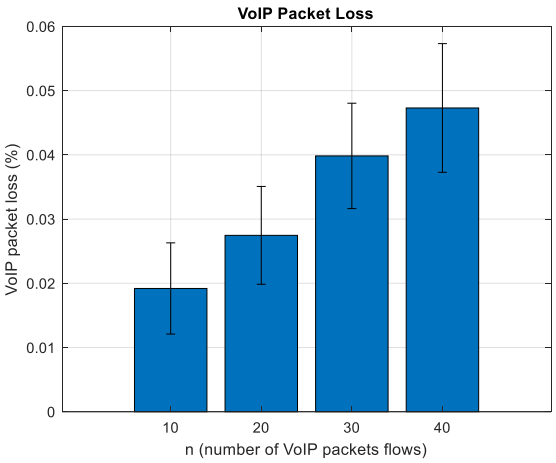
Primeiramente, foram atribuídos os valores às constantes do problema, de acordo com o pedido no enunciado (linhas 1 a 7). De seguida, foram inicializados sete vetores a zeros (linhas 8 a 14), que irão conter os valores de retorno do *simulator4* e mais oito vetores a zeros (linhas 15 a 22) que irão conter os valores finais para a média e a taxa de ocupação das variáveis pretendidas.

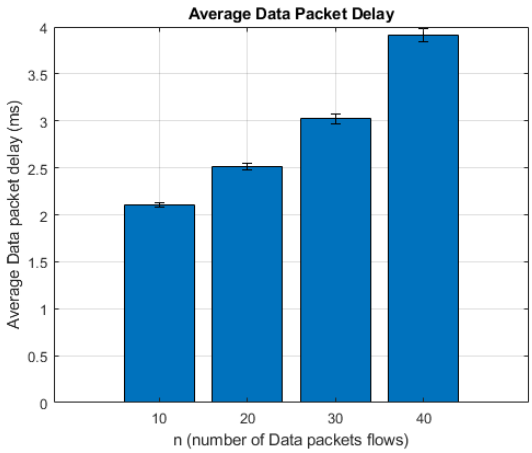
Depois, para cada valor do vetor *n* (linhas 23 a 36), executou-se o simulador N vezes (linhas 24 a 27) e, com os valores lidos do simulador, calculou-se os pacotes de dados perdidos e de VoIP e as respetivas taxas de ocupação (linhas 28 a 31) e a média do atraso médio de pacotes de dados e de VoIP e a respetiva taxa de ocupação (linhas 32 a 35).

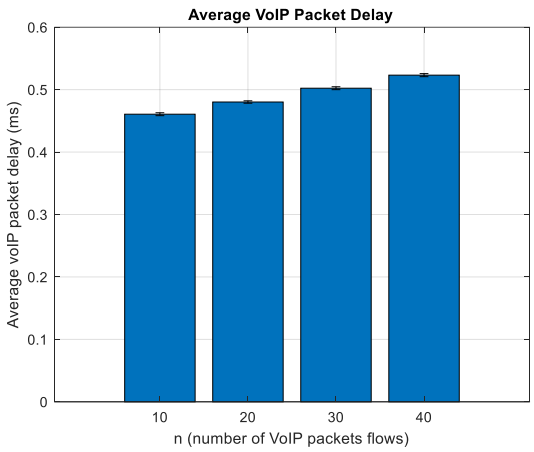
Por fim, desenharam-se os gráficos correspondentes aos pacotes de dados perdidos (linhas 37 a 47), aos pacotes de VoIP perdidos (linhas 48 a 58), ao atraso médio de pacotes de dados (linhas 59 a 69) e ao atraso médio de pacotes de VoIP (linhas 70 a 81).

Result









Conclusions

Neste exercício os pacotes VoIP têm uma prioridade mais elevada em relação aos pacotes de dados. Isto permite que os pacotes VoIP sejam enviados mais rapidamente que os pacotes de dados. Confrontando os dados da experiência, é possível observar que a perda percentual de pacotes do tipo VoIP é inferior á perda de pacotes de dados, assim como o atraso médio dos mesmos. Sendo os pacotes VoIP tratados primeiramente, o seu atraso será inferior ao atraso dos pacotes de dados.

De referir ainda que nos gráficos referentes aos pacotes de VoIP os resultados no eixo dos y's crescem de uma forma linear com um declive bastante inferior ao declive da reta de subida dos gráficos correspondentes aos pacotes de dados.

2.f. Develop a new version of *Simulator4* 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). Repeat experiment 2.e but now with the new version of Simulator4. Justify these results and the differences between them and the results of 2.e. Take conclusions concerning the impact of the number of VoIP flows in the obtained average packet delay and packet loss of each service when (i) VoIP service is supported with a priority which is higher than the data service and (ii) the packet acceptance in the queue is differentiated.

Matlab code

```
lambda = 1500;
C = 10;
f = 10000;
P = 10000;
N = 50;
n = [10, 20, 30, 40];
alfa = 0.1;
PLdata = zeros(1,4);
APDdata = zeros(1,4);
MPDdata = zeros(1,4);
PLvoip = zeros(1,4);
APDvoip = zeros(1,4);
MPDvoip = zeros(1,4);
TT = zeros(1,4);
mediaAPDdata = zeros(1,4);
termAPDdata = zeros(1,4);
mediaAPDvoip = zeros(1,4);
termAPDvoip = zeros(1,4);
mediaPLdata = zeros(1,4);
termPLdata = zeros(1,4);
mediaPLvoip = zeros(1,4);
termPLvoip = zeros(1,4);
for i= 1:length(n)
    for it= 1:N
        [PLdata(it), APDdata(it), MPDdata(it), TT(it), PLvoip(it), APDvoip(it), MPDvoip(it)] =
Simulator4New(lambda,C,f,P,n(i));
    end
    mediaPLdata(i) = mean(PLdata);
    termPLdata(i) = norminv(1-alfa/2)*sqrt(var(PLdata)/N);
    mediaPLvoip(i) = mean(PLvoip);
    termPLvoip(i) = norminv(1-alfa/2)*sqrt(var(PLvoip)/N);
    mediaAPDdata(i) = mean(APDdata);
    termAPDdata(i) = norminv(1-alfa/2)*sqrt(var(APDdata)/N);
    mediaAPDvoip(i) = mean(APDvoip);
    termAPDvoip(i) = norminv(1-alfa/2)*sqrt(var(APDvoip)/N);
end
figure(9);
h = bar(n,mediaPLdata);
hold on
er = errorbar(n,mediaPLdata,termPLdata);
er.Color = [0 0 0];
er.LineStyle = 'none';
grid on
title('Data Packet Loss');
xlabel('n (number of Data packets flows)');
ylabel('Data packet loss (%)');
hold off
figure(10);
h = bar(n,mediaPLvoip);
hold on
er = errorbar(n,mediaPLvoip,termPLvoip);
er.Color = [0 0 0];
er.LineStyle = 'none';
grid on
```

```
55  title('VoIP Packet Loss');
56  xlabel('n (number of VoIP packets flows)');
57  ylabel('VoIP packet loss (%)');
58  hold off
59  figure(11);
60  h = bar(n,mediaAPDdata);
61  hold on
62  er = errorbar(n,mediaAPDdata,termAPDdata);
63  er.Color = [0 0 0];
64  er.LineStyle = 'none';
65  grid on
66  title('Average Data Packet Delay');
67  xlabel('n (number of Data packets flows');
68  ylabel('Average Data packet delay (ms)');
69  hold off
70  figure(12);
71  h = bar(n,mediaAPDvoip);
72  hold on
73  er = errorbar(n,mediaAPDvoip,termAPDvoip);
74  er.Color = [0 0 0];
75  er.LineStyle = 'none';
76  grid on
78  title('Average VoIP Packet Delay');
79  xlabel('n (number of VoIP packets flows)');
80  ylabel('Average VoIP packet delay (ms)');
81  hold off
```

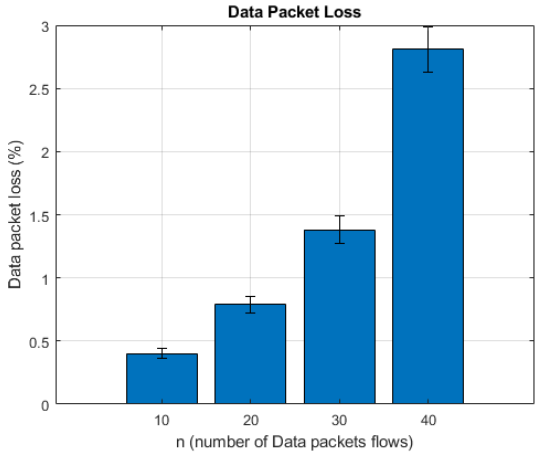
Code analysis

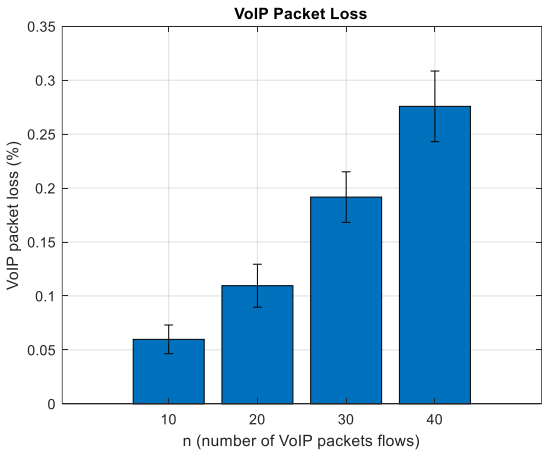
Primeiramente, foram atribuídos os valores às constantes do problema, de acordo com o pedido no enunciado (linhas 1 a 7). De seguida, foram inicializados sete vetores a zeros (linhas 8 a 14), que irão conter os valores de retorno do *simulator4New* e mais oito vetores a zeros (linhas 15 a 22) que irão conter os valores finais para a média e a taxa de ocupação das variáveis pretendidas.

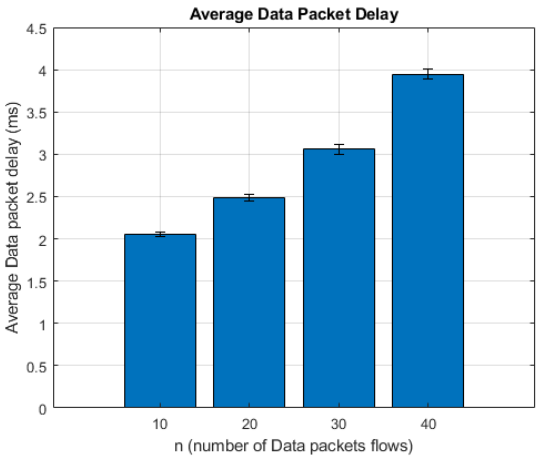
Depois, para cada valor do vetor *n* (linhas 23 a 36), executou-se o simulador N vezes (linhas 24 a 27) e, com os valores lidos do simulador, calculou-se os pacotes de dados perdidos e de VoIP e as respetivas taxas de ocupação (linhas 28 a 31) e a média do atraso médio de pacotes de dados e de VoIP e a respetiva taxa de ocupação (linhas 32 a 35).

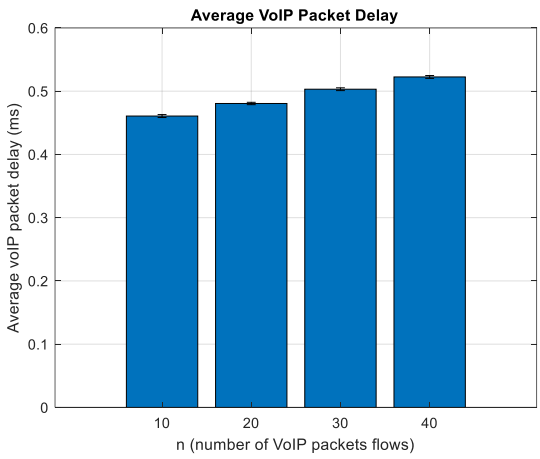
Por fim, desenharam-se os gráficos correspondentes aos pacotes de dados perdidos (linhas 37 a 47), aos pacotes de VoIP perdidos (linhas 48 a 58), ao atraso médio de pacotes de dados (linhas 59 a 69) e ao atraso médio de pacotes de VoIP (linhas 70 a 81).

Result









Conclusions

Relativamente ao atraso médio dos pacotes os resultados são bastante semelhantes ao exercício 2e). A limitação da capacidade de pacotes de dados para 90% leva a que menos pacotes sejam recebidos, aumentando assim a velocidade de resposta. Colocando agora a questão em relação aos pacotes VoIP é possível verificar que pouco ou nada mudaram, isto acontece devido á redução do número de pacotes do tipo data não influenciar os pacotes VoIP devido á sua prioridade mais elevada.

Falando agora sobre a perda de pacotes, é possivel verificar que os pacotes de dados sofreram um aumento devido à limitação de 90% dos mesmos. Já os pacotes VoIP não são tão perdidos e existe uma taxa de receção maior relativamente ao exercício 2e) visto os pacotes de dados terem sido limitados, ficando assim mais espaço na fila e existindo uma probabilidade maior destes serem aceites.

SIMULATOR 1

Matlab code

```
function [PL , APD , MPD , TT] = Simulator1(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    - average packet delay (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
TRANSMITTEDBYTES= 0;    % Sum of the Bytes of transmitted packets
DELAYS= 0;              % Sum of the delays of transmitted packets
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
            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
            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= 1000*DELAYS/TRANSMITTEDPACKETS;   % in milliseconds
MPD= 1000*MAXDELAY;                    % in milliseconds
TT= 10^(-6)*TRANSMITTEDBYTES*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
```

SIMULATOR 1 (NEW VERSION – 1.e)

Matlab code

```
function [PL , APD , MPD , TT, APD64, APD110, APD1518] = Simulator1(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  - average packet delay (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
TRANSMITTEDBYTES = 0;       % Sum of the Bytes of transmitted packets
DELAYS = 0;                 % Sum of the delays of transmitted packets
MAXDELAY = 0;               % Maximum delay among all transmitted packets
TRANSMITTED64BYTES = 0;      % No. of transmitted packets with 64 bytes
TRANSMITTED110BYTES = 0;     % No. of transmitted packets with 110 bytes
TRANSMITTED1518BYTES = 0;    % No. of transmitted packets with 1518 bytes
DELAY64 = 0;                % Sum of the delays of transmitted packets with 64 bytes
DELAY110 = 0;               % Sum of the delays of transmitted packets with 110 bytes
DELAY1518 = 0;              % Sum of the delays of transmitted packets with 1518 bytes
% 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
            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
            TRANSMITTEDBYTES = TRANSMITTEDBYTES + PacketSize;
```

```
            if (PacketSize == 64)
                TRANSMITTED64BYTES = TRANSMITTED64BYTES + 1;
                DELAY64 = DELAY64 + (Clock - ArrivalInstant);
            elseif (PacketSize == 110)
                TRANSMITTED110BYTES = TRANSMITTED110BYTES + 1;
                DELAY110 = DELAY110 + (Clock - ArrivalInstant);
            elseif (PacketSize == 1518)
                TRANSMITTED1518BYTES = TRANSMITTED1518BYTES + 1;
                DELAY1518 = DELAY1518 + (Clock - ArrivalInstant);
            else
                DELAYS = DELAYS + (Clock - ArrivalInstant);
            end
            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 = 1000*DELAYS/TRANSMITTEDPACKETS;           % in milliseconds
MPD = 1000*MAXDELAY;                            % in milliseconds
TT = 10^(-6)*TRANSMITTEDBYTES*8/Clock;          % in Mbps
APD64 = 1000*DELAY64/TRANSMITTED64BYTES;        % in milliseconds
APD110 = 1000*DELAY110/TRANSMITTED110BYTES;     % in milliseconds
APD1518 = 1000*DELAY1518/TRANSMITTED1518BYTES;  % in milliseconds
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
```

SIMULATOR 3

Matlab code

```
function [PLdata , APDdata, MPDdata , TT , PLvoip, APDvoip, MPDvoip ] =
Simulator3(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      - additional VoIP packet flows (sec)
% OUTPUT PARAMETERS:
%  PLdata   - packet loss of data packets (%)
%  APDdata  - average delay of data packets (milliseconds)
%  MPDdata  - maximum delay of data packets (milliseconds)
%  PLvoip   - packet loss of voip packets (%)
%  APDvoip  - average delay of voip 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
DATA = 2;
VOIP = 3;
% 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
TOTALVOIPPACKETS = 0;        % No. of voip packets arrived to the systeM
LOSTPACKETS = 0;             % No. of packets dropped due to buffer overflow
LOSTVOIPPACKETS = 0;         % No. of voip packets dropped due to buffer overflow
TRANSMITTEDPACKETS = 0;      % No. of transmitted packets
TRANSMITTEDVOIPPACKETS = 0;  % No. of voip transmitted packets
TRANSMITTEDBYTES = 0;        % Sum of the Bytes of transmitted packets
DELAYS = 0;                  % Sum of the delays of transmitted packets
VOIPDELAYS = 0;              % Sum of the delays of transmitted voip packets
MAXDELAY = 0;                % Maximum delay among all transmitted packets
MAXVOIPDELAY = 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];
for i=1:n
    tmp = Clock + 0.02*rand();
    EventList = [EventList; ARRIVAL, tmp, GeneratePacketSizeVoip(), tmp, VOIP];
end
% SIMILATION LOOP:
while (TRANSMITTEDPACKETS + TRANSMITTEDVOIPPACKETS) < P   % Stopping criterium
    EventList = sortrows(EventList,2);    % Order EventList by time
    Event = EventList(1,1);               % Get first event
    Clock = EventList(1,2);
    PacketSize = EventList(1,3);
    ArrivalInstant = EventList(1,4);
    PacketType = EventList(1,5);
    EventList(1,:)= [];                  % Eliminate first event
    switch Event
        case ARRIVAL        % If first event is an ARRIVAL
            if PacketType == DATA
                TOTALPACKETS = TOTALPACKETS+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
                        QUEUE = [QUEUE;PacketSize , Clock, PacketType];
                        QUEUEOCCUPATION = QUEUEOCCUPATION + PacketSize;
                    else
                        LOSTPACKETS = LOSTPACKETS + 1;
                    end
                end
            else
                TOTALVOIPPACKETS = TOTALVOIPPACKETS+1;
                tmp = Clock + 0.008*rand() + 0.016;
                EventList = [EventList; ARRIVAL, tmp, GeneratePacketSizeVoip(), tmp,
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];
                        QUEUEOCCUPATION = QUEUEOCCUPATION + PacketSize;
                    else
                        LOSTVOIPPACKETS = LOSTVOIPPACKETS + 1;
                    end
                end
            end
        case DEPARTURE              % If first event is a DEPARTURE
            if PacketType == DATA
                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), QUEUE(1,3)];
                    QUEUEOCCUPATION = QUEUEOCCUPATION - QUEUE(1,1);
                    QUEUE(1,:) = [];
                else
                    STATE = 0;
                end
            end
            if PacketType == VOIP
                TRANSMITTEDBYTES = TRANSMITTEDBYTES + PacketSize;
                VOIPDELAYS = VOIPDELAYS + (Clock - ArrivalInstant);
                if Clock - ArrivalInstant > MAXVOIPDELAY
                    MAXVOIPDELAY = 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:
PLdata = 100*LOSTPACKETS/TOTALPACKETS;              % in %
APDdata = 1000*DELAYS/TRANSMITTEDPACKETS;           % in milliseconds
MPDdata = 1000*MAXDELAY;                            % in milliseconds
```

```
PLvoip = 100*LOSTVOIPPACKETS/TOTALVOIPPACKETS;      % packet loss VOIP
APDvoip = 1000*VOIPDELAYS/TRANSMITTEDVOIPPACKETS;   % average packet delay VOIP
MPDvoip = 1000*MAXVOIPDELAY;                        % maximum packet delay VOIP
TT = 10^(-6)*TRANSMITTEDBYTES*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 = GeneratePacketSizeVoip()
    out = randi([110 130]);
end
```

SIMULATOR 4

Matlab code

```
function [PLdata , APDdata, MPDdata , TT , PLvoip, APDvoip, MPDvoip ] =
Simulator4(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      - additional VoIP packet flows (sec)
% OUTPUT PARAMETERS:
%  PLdata   - packet loss of data packets (%)
%  APDdata  - average delay of data packets (milliseconds)
%  MPDdata  - maximum delay of data packets (milliseconds)
%  PLvoip   - packet loss of voip packets (%)
%  APDvoip  - average delay of voip 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
DATA = 2;
VOIP = 3;
% STATE VARIABLES:
STATE = 0;              % 0 - connection free; 1 - connection bysy
QUEUEOCCUPATION = 0;    % Occupation of the queue (in Bytes)
QUEUEVOIPOCCUPATION = 0;% Occupation of the voip queue (in Bytes)
QUEUE = [];              % Size and arriving time instant of each packet in the queue
QUEUEVOIP = [];
% STATISTICAL COUNTERS:
TOTALPACKETS = 0;            % No. of packets arrived to the system
TOTALVOIPPACKETS = 0;        % No. of voip packets arrived to the systeM
LOSTPACKETS = 0;             % No. of packets dropped due to buffer overflow
LOSTVOIPPACKETS = 0;         % No. of voip packets dropped due to buffer overflow
TRANSMITTEDPACKETS = 0;      % No. of transmitted packets
TRANSMITTEDVOIPPACKETS = 0;  % No. of voip transmitted packets
TRANSMITTEDBYTES = 0;        % Sum of the Bytes of transmitted packets
DELAYS = 0;                  % Sum of the delays of transmitted packets
VOIPDELAYS = 0;              % Sum of the delays of transmitted voip packets
MAXDELAY = 0;                % Maximum delay among all transmitted packets
MAXVOIPDELAY = 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];
for i=1:n
    tmp = Clock + 0.02*rand();
    EventList = [EventList; ARRIVAL, tmp, GeneratePacketSizeVoip(), tmp, VOIP];
end
% SIMILATION LOOP:
while (TRANSMITTEDPACKETS + TRANSMITTEDVOIPPACKETS) < P   % Stopping criterium
    EventList = sortrows(EventList,2);    % Order EventList by time
    Event = EventList(1,1);               % Get first event
    Clock = EventList(1,2);
    PacketSize = EventList(1,3);
    ArrivalInstant = EventList(1,4);
    PacketType = EventList(1,5);
    EventList(1,:)= [];                  % Eliminate first event
    switch Event
        case ARRIVAL            % If first event is an ARRIVAL
            if PacketType == DATA
                TOTALPACKETS = TOTALPACKETS+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 + QUEUEVOIPOCCUPATION + PacketSize <= f
                        QUEUE = [QUEUE; PacketSize , Clock, PacketType];
                        QUEUEOCCUPATION = QUEUEOCCUPATION + PacketSize;
                    else
                        LOSTPACKETS = LOSTPACKETS + 1;
                    end
                end
            else
                TOTALVOIPPACKETS = TOTALVOIPPACKETS + 1;
                tmp = Clock + 0.008*rand() + 0.016;
                EventList = [EventList; ARRIVAL, tmp, GeneratePacketSizeVoip(), tmp,
PacketType];
                if STATE == 0
                    STATE = 1;
                    EventList = [EventList; DEPARTURE, Clock + 8*PacketSize/(C*10^6),
PacketSize, Clock, PacketType];
                else
                    if QUEUEOCCUPATION + PacketSize <= f
                        QUEUEVOIP = [QUEUEVOIP; PacketSize , Clock, PacketType];
                        QUEUEVOIPOCCUPATION = QUEUEVOIPOCCUPATION + PacketSize;
                    else
                        LOSTVOIPPACKETS = LOSTVOIPPACKETS + 1;
                    end
                end
            end
        case DEPARTURE          % If first event is a DEPARTURE
            if PacketType == DATA
                TRANSMITTEDBYTES = TRANSMITTEDBYTES + PacketSize;
                DELAYS = DELAYS + (Clock - ArrivalInstant);
                if Clock - ArrivalInstant > MAXDELAY
                    MAXDELAY = Clock - ArrivalInstant;
                end
                TRANSMITTEDPACKETS = TRANSMITTEDPACKETS + 1;
                if QUEUEVOIPOCCUPATION > 0
                    EventList = [EventList; DEPARTURE, Clock + 8*QUEUEVOIP(1,1)/(C*10^6),
QUEUEVOIP(1,1), QUEUEVOIP(1,2), QUEUEVOIP(1,3)];
                    QUEUEVOIPOCCUPATION = QUEUEVOIPOCCUPATION - QUEUEVOIP(1,1);
                    QUEUEVOIP(1,:) = [];
                else
                    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
            if PacketType == VOIP
                TRANSMITTEDBYTES = TRANSMITTEDBYTES + PacketSize;
                VOIPDELAYS = VOIPDELAYS + (Clock - ArrivalInstant);
                if Clock - ArrivalInstant > MAXVOIPDELAY
                    MAXVOIPDELAY = Clock - ArrivalInstant;
                end
                TRANSMITTEDVOIPPACKETS = TRANSMITTEDVOIPPACKETS + 1;
                if QUEUEVOIPOCCUPATION > 0
                    EventList = [EventList; DEPARTURE, Clock + 8*QUEUEVOIP(1,1)/(C*10^6),
QUEUEVOIP(1,1), QUEUEVOIP(1,2), QUEUEVOIP(1,3)];
                    QUEUEVOIPOCCUPATION = QUEUEVOIPOCCUPATION - QUEUEVOIP(1,1);
                    QUEUEVOIP(1,:) = [];
                else
```

```
                    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
end
% PERFORMANCE PARAMETERS DETERMINATION:
PLdata = 100*LOSTPACKETS/TOTALPACKETS;              % in %
PLvoip = 100*LOSTVOIPPACKETS/TOTALVOIPPACKETS;      % in %
APDdata = 1000*DELAYS/TRANSMITTEDPACKETS;           % in milliseconds
APDvoip = 1000*VOIPDELAYS/TRANSMITTEDVOIPPACKETS;   % in milliseconds
MPDdata = 1000*MAXDELAY;                            % in milliseconds
MPDvoip = 1000*MAXVOIPDELAY;                        % in milliseconds
TT = 10^(-6)*TRANSMITTEDBYTES*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 = GeneratePacketSizeVoip()
    out = randi([110 130]);
end
```

SIMULATOR 4 (NEW VERSION – 2.f)

Matlab code

```
function [PLdata , APDdata, MPDdata , TT , PLvoip, APDvoip, MPDvoip ] =
Simulator4(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      - additional VoIP packet flows (sec)
% OUTPUT PARAMETERS:
%  PLdata   - packet loss of data packets (%)
%  APDdata  - average delay of data packets (milliseconds)
%  MPDdata  - maximum delay of data packets (milliseconds)
%  PLvoip   - packet loss of voip packets (%)
%  APDvoip  - average delay of voip 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
DATA = 2;
VOIP = 3;
% STATE VARIABLES:
STATE = 0;              % 0 - connection free; 1 - connection bysy
QUEUEOCCUPATION = 0;    % Occupation of the queue (in Bytes)
QUEUEVOIPOCCUPATION = 0;% Occupation of the voip queue (in Bytes)
QUEUE = [];              % Size and arriving time instant of each packet in the queue
QUEUEVOIP = [];
% STATISTICAL COUNTERS:
TOTALPACKETS = 0;            % No. of packets arrived to the system
TOTALVOIPPACKETS = 0;        % No. of voip packets arrived to the systeM
LOSTPACKETS = 0;             % No. of packets dropped due to buffer overflow
LOSTVOIPPACKETS = 0;         % No. of voip packets dropped due to buffer overflow
TRANSMITTEDPACKETS = 0;      % No. of transmitted packets
TRANSMITTEDVOIPPACKETS = 0;  % No. of voip transmitted packets
TRANSMITTEDBYTES = 0;        % Sum of the Bytes of transmitted packets
DELAYS = 0;                  % Sum of the delays of transmitted packets
VOIPDELAYS = 0;              % Sum of the delays of transmitted voip packets
MAXDELAY = 0;                % Maximum delay among all transmitted packets
MAXVOIPDELAY = 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];
for i=1:n
    tmp = Clock + 0.02*rand();
    EventList = [EventList; ARRIVAL, tmp, GeneratePacketSizeVoip(), tmp, VOIP];
end
% SIMILATION LOOP:
while (TRANSMITTEDPACKETS + TRANSMITTEDVOIPPACKETS) < P   % Stopping criterium
    EventList = sortrows(EventList,2);    % Order EventList by time
    Event = EventList(1,1);               % Get first event
    Clock = EventList(1,2);
    PacketSize = EventList(1,3);
    ArrivalInstant = EventList(1,4);
    PacketType = EventList(1,5);
    EventList(1,:)= [];                  % Eliminate first event
    switch Event
        case ARRIVAL            % If first event is an ARRIVAL
            if PacketType == DATA
                TOTALPACKETS = TOTALPACKETS+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 + QUEUEVOIPOCCUPATION <= f*0.9 && QUEUEOCCUPATION +
QUEUEVOIPOCCUPATION + PacketSize
                        QUEUE = [QUEUE; PacketSize , Clock, PacketType];
                        QUEUEOCCUPATION = QUEUEOCCUPATION + PacketSize;
                    else
                        LOSTPACKETS = LOSTPACKETS + 1;
                    end
                end
            else
                TOTALVOIPPACKETS = TOTALVOIPPACKETS + 1;
                tmp = Clock + 0.008*rand() + 0.016;
                EventList = [EventList; ARRIVAL, tmp, GeneratePacketSizeVoip(), tmp,
PacketType];
                if STATE == 0
                    STATE = 1;
                    EventList = [EventList; DEPARTURE, Clock + 8*PacketSize/(C*10^6),
PacketSize, Clock, PacketType];
                else
                    if QUEUEOCCUPATION + PacketSize <= f
                        QUEUEVOIP = [QUEUEVOIP; PacketSize , Clock, PacketType];
                        QUEUEVOIPOCCUPATION = QUEUEVOIPOCCUPATION + PacketSize;
                    else
                        LOSTVOIPPACKETS = LOSTVOIPPACKETS + 1;
                    end
                end
            end
        case DEPARTURE          % If first event is a DEPARTURE
            if PacketType == DATA
                TRANSMITTEDBYTES = TRANSMITTEDBYTES + PacketSize;
                DELAYS = DELAYS + (Clock - ArrivalInstant);
                if Clock - ArrivalInstant > MAXDELAY
                    MAXDELAY = Clock - ArrivalInstant;
                end
                TRANSMITTEDPACKETS = TRANSMITTEDPACKETS + 1;
                if QUEUEVOIPOCCUPATION > 0
                    EventList = [EventList; DEPARTURE, Clock + 8*QUEUEVOIP(1,1)/(C*10^6),
QUEUEVOIP(1,1), QUEUEVOIP(1,2), QUEUEVOIP(1,3)];
                    QUEUEVOIPOCCUPATION = QUEUEVOIPOCCUPATION - QUEUEVOIP(1,1);
                    QUEUEVOIP(1,:) = [];
                else
                    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
            if PacketType == VOIP
                TRANSMITTEDBYTES = TRANSMITTEDBYTES + PacketSize;
                VOIPDELAYS = VOIPDELAYS + (Clock - ArrivalInstant);
                if Clock - ArrivalInstant > MAXVOIPDELAY
                    MAXVOIPDELAY = Clock - ArrivalInstant;
                end
                TRANSMITTEDVOIPPACKETS = TRANSMITTEDVOIPPACKETS + 1;
                if QUEUEVOIPOCCUPATION > 0
                    EventList = [EventList; DEPARTURE, Clock + 8*QUEUEVOIP(1,1)/(C*10^6),
QUEUEVOIP(1,1), QUEUEVOIP(1,2), QUEUEVOIP(1,3)];
                    QUEUEVOIPOCCUPATION = QUEUEVOIPOCCUPATION - QUEUEVOIP(1,1);
                    QUEUEVOIP(1,:) = [];
```

```
                else
                    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
end
% PERFORMANCE PARAMETERS DETERMINATION:
PLdata = 100*LOSTPACKETS/TOTALPACKETS;              % in %
PLvoip = 100*LOSTVOIPPACKETS/TOTALVOIPPACKETS;      % in %
APDdata = 1000*DELAYS/TRANSMITTEDPACKETS;           % in milliseconds
APDvoip = 1000*VOIPDELAYS/TRANSMITTEDVOIPPACKETS;   % in milliseconds
MPDdata = 1000*MAXDELAY;                            % in milliseconds
MPDvoip = 1000*MAXVOIPDELAY;                        % in milliseconds
TT = 10^(-6)*TRANSMITTEDBYTES*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 = GeneratePacketSizeVoip()
    out = randi([110 130]);
end
```

AUXILIAR FUNCTIONS

Matlab code

```
function W = TheoAvgDelayMG1(lambda,C)
    [es, es2] = ES_data(C);
    W = (((lambda*es2 ) / (2 * (1 - lambda*es))) + es) * 1000;
end

function [es, es2] = ES_data(C)
    k = (0.41/((109 - 65 + 1)+(1517 - 111 + 1)));
    es = 0.19*((64*8)/(C*10^6)) + 0.23*((110*8)/(C*10^6)) + 0.17*((1518*8)/(C*10^6));
    es2 = 0.19*((64*8)/(C*10^6))^2 + 0.23*((110*8)/(C*10^6))^2 + 0.17*((1518*8)/(C*10^6))^2;
    for n = 65:109
        es = es + k * ((n*8)/(C*10^6));
        es2 = es2 + k * ((n * 8)/(C*10^6))^2;
    end
    for n = 111:1517
         es = es + k * ((n*8)/(C*10^6));
         es2 = es2 + k * ((n * 8)/(C*10^6))^2;
    end
end

function mean = MeanPacketSize()
    mean = 0.19 * (64*8) + 0.23 * (110*8) + 0.17 * (1518*8);
    for n = 65:109
        mean = mean + ((0.41/((109 - 65 + 1)+(1517 - 111 + 1))) * (n*8));
    end
    for j = 111:1517
        mean = mean + ((0.41/((109 - 65 + 1)+(1517 - 111 + 1))) * (j*8));
    end
end

function [es, es2] = ES_voip(C, v)
    es = 0;
    es2 = 0;
    for i = 1:size(v, 2)
        es = es + (((v(i)*8)/(C*10^6)))*(1/21);
        es2 = es2 + (((v(i)*8)/(C*10^6))^2)*(1/21);
    end
end

function [W1, W2] = TheoAvgDelayMG1_priorities(lambda, C, n)
    meanPacketVoipSize = (110+130)/2;
    bytesVoip = 110:130;
    lambdaVoip = (1/(20*10^3))*n;
    lambdaData = lambda;
    [esData, es2Data] =  ES_data(C);
    [esVoip, es2Voip] = ES_voip(C,bytesVoip);
    uVoip = (C*10^6) / (meanPacketVoipSize*8);
    uData = (C*10^6) / (esData);
    p1 = lambdaVoip / uVoip;
    p2 = lambdaData / uData;
    WQ1 = ((lambdaVoip*es2Voip) + (lambdaData.*es2Data)) / (2*(1-p1));
    WQ2 = ((lambdaVoip*es2Voip) + (lambdaData.*es2Data)) / (2*(1-p1)*(1-p1-p2));
    W1 = (WQ1 + esVoip) * 1000;
    W2 = (WQ2 + esData) * 1000;
end
```