Relatório Guião 3
Desempenho e Dimensionamento de Redes

Dário Matos 89298
Pedro Almeida 89205

Junho de 2021

Pergunta 3.a

Código

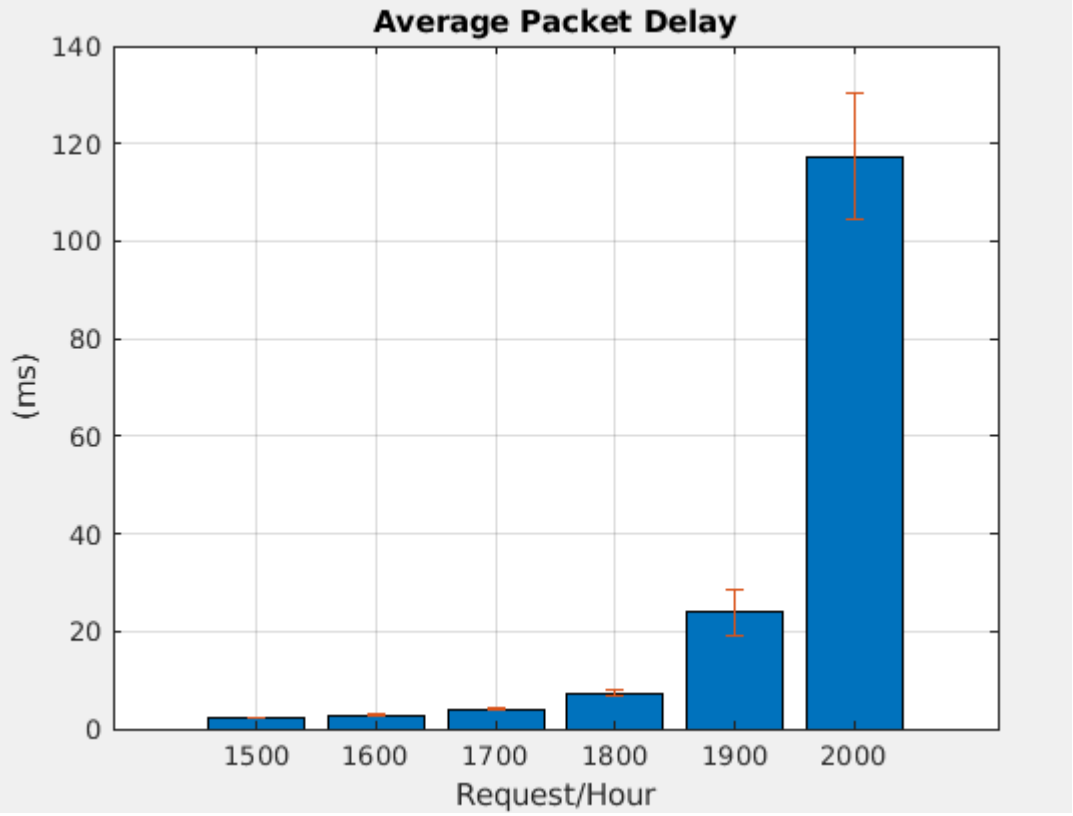
```
lambda = [1500, 1600, 1700, 1800, 1900, 2000];
C = 10;
P = 10000;
f = 10000000;
b = 0;

% number of simulations
N = 20; % a

PL = zeros(1,N);
APD = zeros(1,N);
MPD = zeros(1,N);
TT = zeros(1,N);

mediaPL = zeros(1, size(lambda, 2));
termPL = zeros(1, size(lambda, 2));
mediaAPD = zeros(1, size(lambda, 2));
termAPD = zeros(1, size(lambda, 2));
mediaMPD = zeros(1, size(lambda, 2));
termMPD = zeros(1, size(lambda, 2));
mediaTT = zeros(1, size(lambda, 2));
termTT = zeros(1, size(lambda, 2));

for i=1:size(lambda, 2)
    for it=1:N
       [PL(it), APD(it), MPD(it), TT(it)] = simulator2(lambda(i), C, f,
P, b);
    end

    % 90 confidence interval %
    alfa= 0.1;

    mediaPL(i) = mean(PL);
    termPL(i) = norminv(1-alfa/2)*sqrt(var(PL)/N);

    mediaAPD(i) = mean(APD);
    termAPD(i) = norminv(1-alfa/2)*sqrt(var(APD)/N);

    mediaMPD(i) = mean(MPD);
    termMPD(i) = norminv(1-alfa/2)*sqrt(var(MPD)/N);

    mediaTT(i) = mean(TT);
    termTT(i) = norminv(1-alfa/2)*sqrt(var(TT)/N);
```

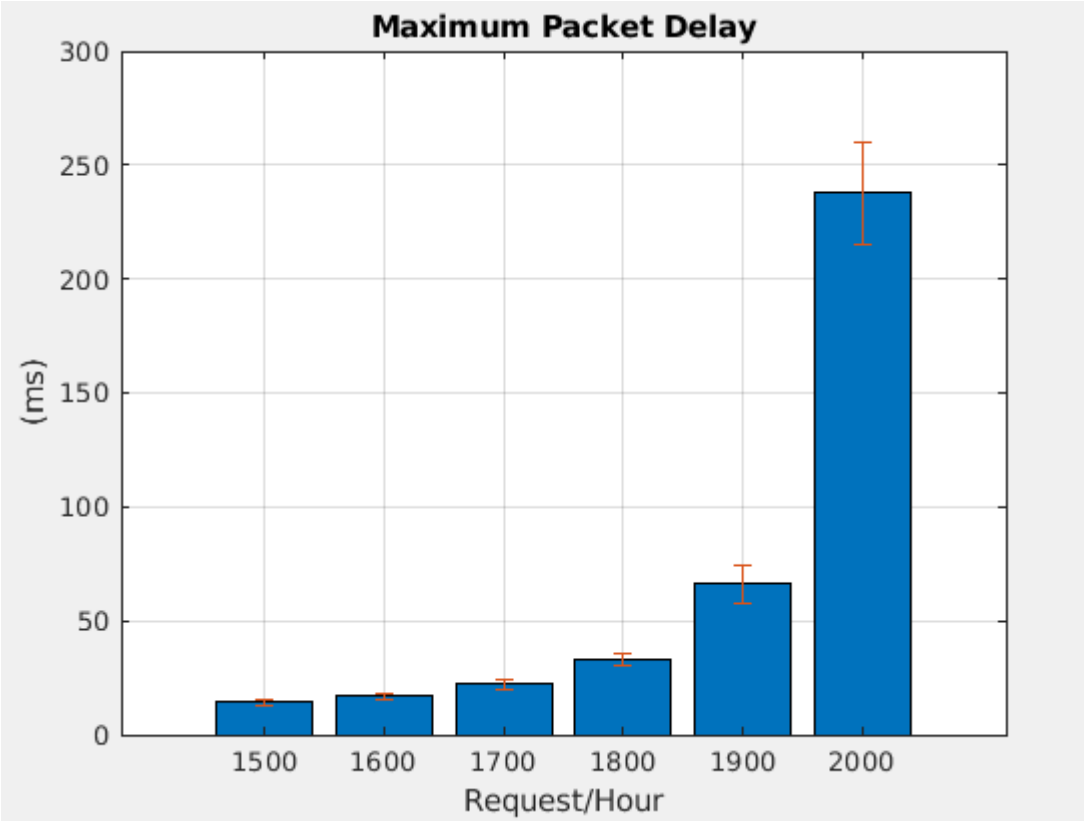
```
end

% graficos
figure(1)
bar(lambda, mediaAPD)
title('Average Packet Delay')
xlabel('Request/Second')
ylabel('(ms)')
hold on
er = errorbar(lambda, mediaAPD, termAPD, termAPD);
er.LineStyle = 'none';
hold off
grid on

figure(2)
bar(lambda, mediaMPD)
title('Maximum Packet Delay')
xlabel('Request/Second')
ylabel('(ms)')
hold on
er = errorbar(lambda, mediaMPD, termMPD, termMPD);
er.LineStyle = 'none';
hold off
grid on

figure(3)
bar(lambda, mediaTT)
title('Total Throughput')
xlabel('Request/Second')
ylabel('(Mbps)')
hold on
er = errorbar(lambda, mediaTT, termTT, termTT);
er.LineStyle = 'none';
hold off
grid on
```

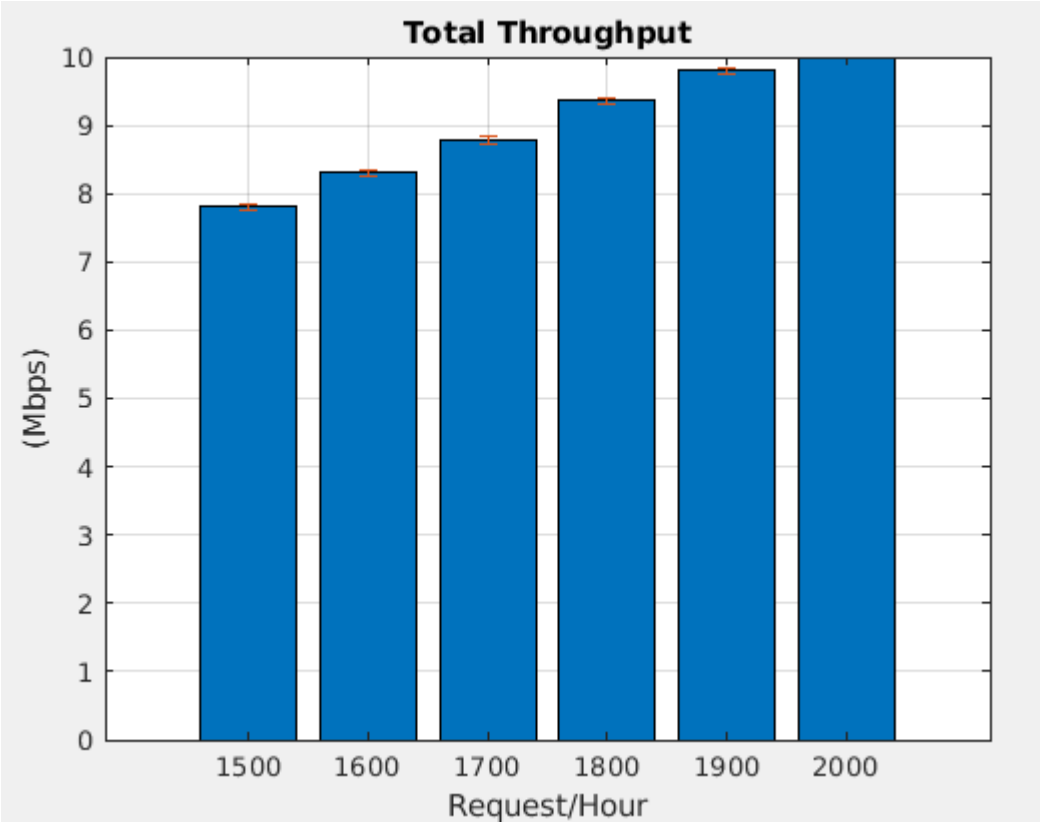
Análise

O processo inicia com a criação das variáveis correspondentes a parâmetros de cálculo providenciadas pelo enunciado. Seguidamente, dá-se aso à criação das variáveis para registar os valores de *Average Packet Delay, Maximum Packet Delay* e *Total Throughput*, o cálculos dos valores por simulação, e cálculo de média dos valores e dos erros. Por fim, dá-se a elaboração dos gráficos para mostrar os resultados.

Resultados


Average Packet Delay
140
120
100
80
60
40
20
0
(ms)
1500
1600
1700
1800
1900
2000
Request/Hour


Maximum Packet Delay
300
250
200
150
100
50
0
(ms)
1500
1600
1700
1800
1900
2000
Request/Hour



Conclusões

A partir dos gráficos acima apresentados, é possível concluir que a média e o máximo dos atrasos do fluxo de pacotes é atinge variações consideravelmente elevadas para um número de pedidos/segundo que atinja os 2 milhares. Ainda assim, verificam-se subidas, embora menos acentuadas, desde os valores de 1500 até 1900. Apenas no *Total Throughput* a subida é quase constante ao longo de todos os valores de requests/Second.

Pergunta 3.b

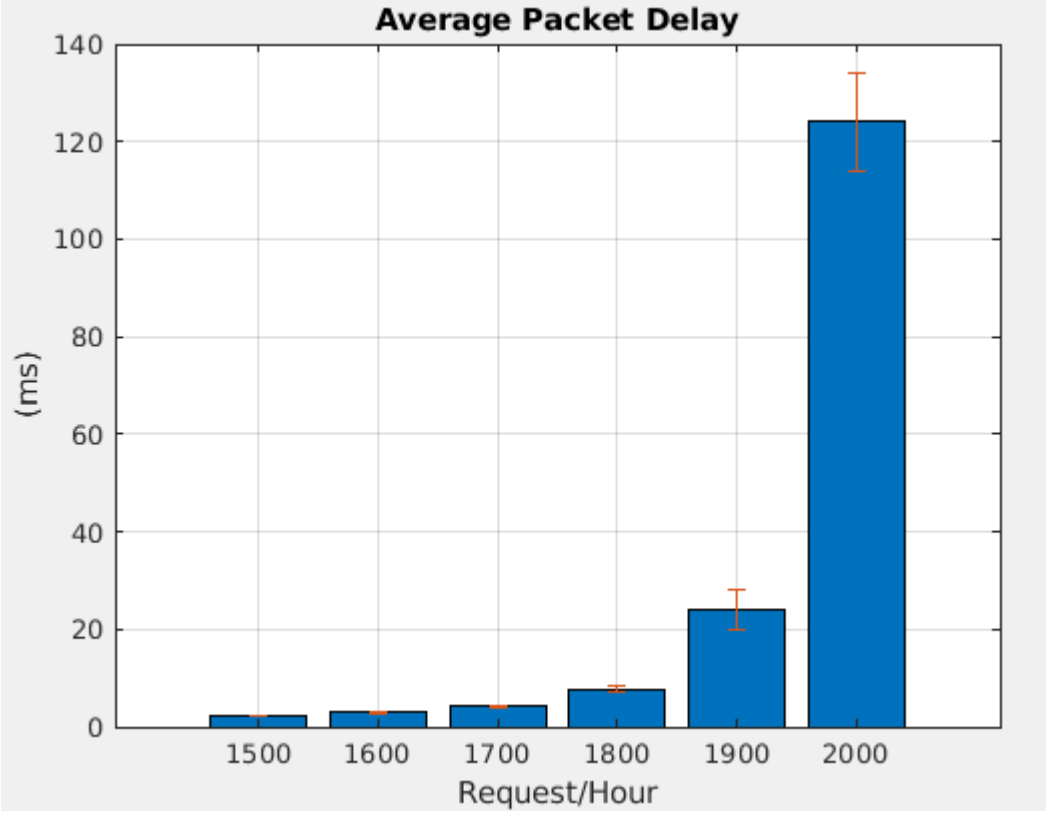
Código

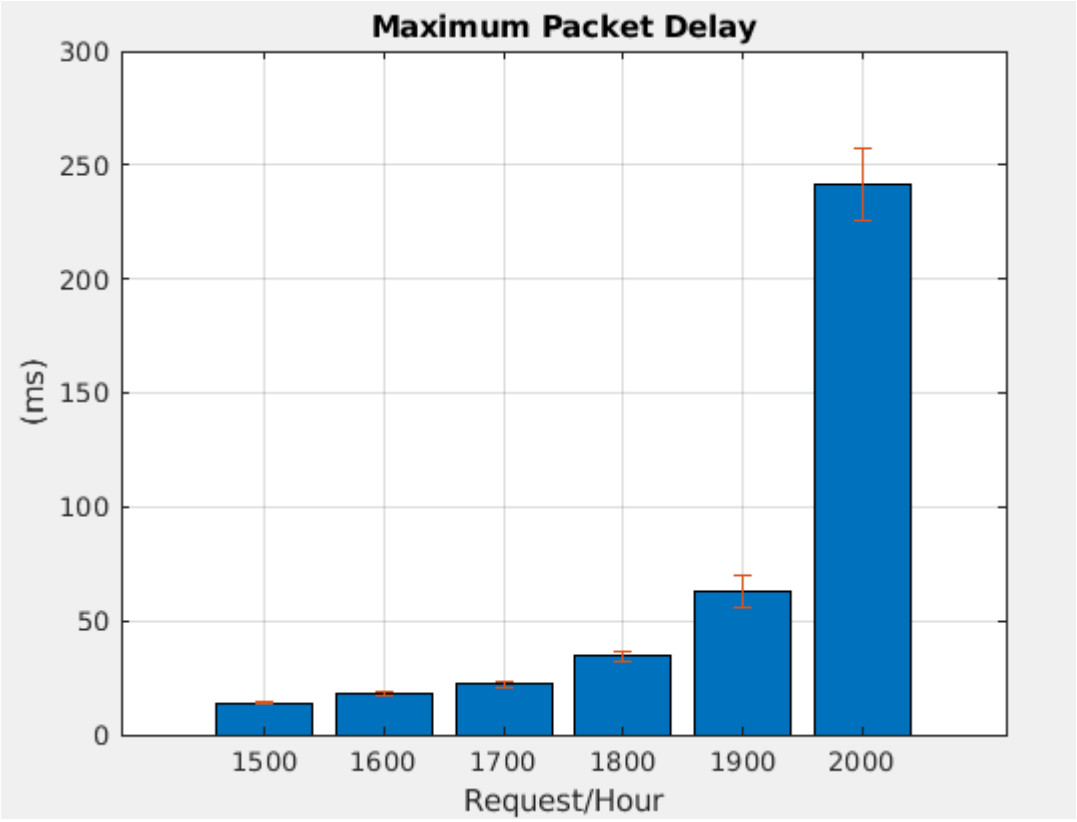
```
% number of simulations
N = 40; % b
```

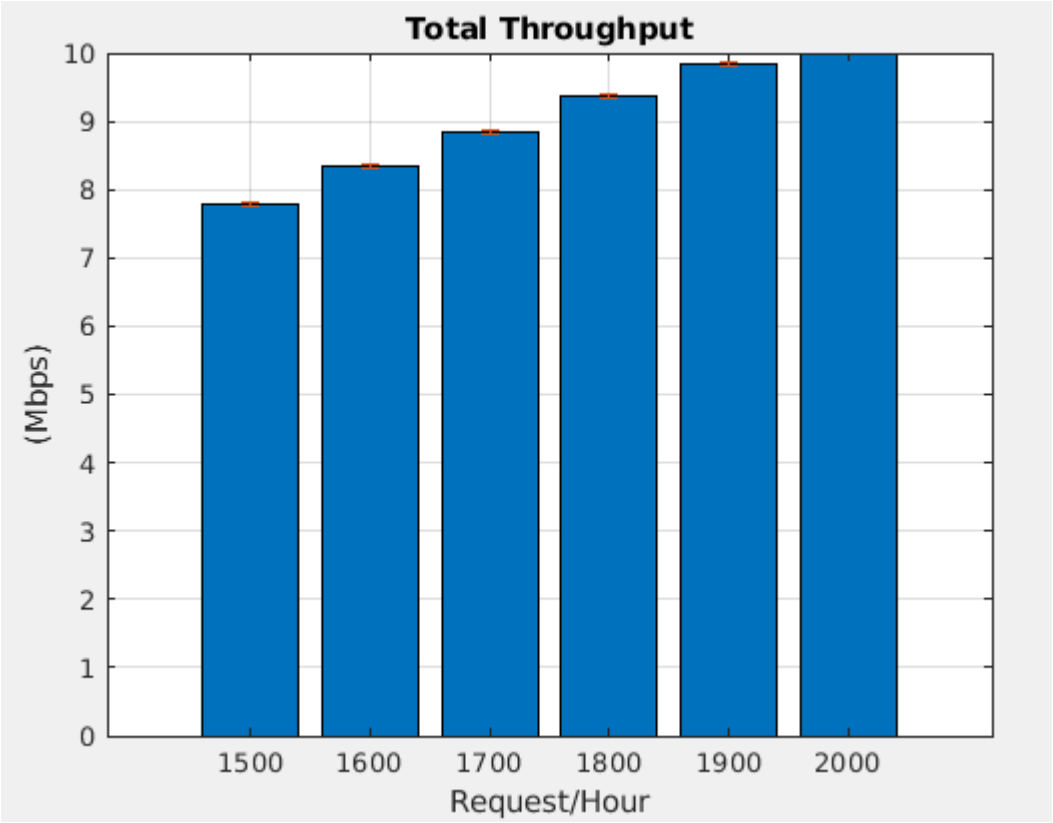
Análise

O código da alínea apenas varia no número de simulações efetuadas.

Resultados


Average Packet Delay
140
120
100
80
60
40
20
0
(ms)
1500
1600
1700
1800
1900
2000
Request/Hour


Maximum Packet Delay
300
250
200
150
100
50
0
(ms)
1500
1600
1700
1800
1900
2000
Request/Hour



Conclusões

Após aumentar o número de simulações, analisa-se que os resultados são muito semelhantes. É só possível concluir que existem intervalos de confiança já bastante reduzidos, e que esta redução é mais acentuada ao longo do aumento do número de simulações, resultando assim em valores cada vez mais fidedignos, se este aumento fosse aumentando.

Pergunta 3.c

Código

```
% c)
% simulation values
disp('Simulation started');

lambda = [1500, 1600, 1700, 1800, 1900, 2000];
C = 10 * 1e6;
P = 10000;
f = 10000000;
b = 0;

% number of simulations
N = 40; % b
```

```
APD = zeros(1,N);
TT = zeros(1,N);

mediaAPD = zeros(1, size(lambda, 2));
mediaTT = zeros(1, size(lambda, 2));

for i=1:size(lambda, 2)
    for it=1:N
       [PL(it), APD(it), MPD(it), TT(it)] = simulator2(lambda(i), 10, f,
P, b);
    end

    mediaAPD(i) = mean(APD);
    mediaTT(i) = mean(TT);
end


% M/M/1 queueing model
disp('M/M/1 started');

tam= [65:109 111:1517];
pres = ((100 - (16+25+20)) / length(tam)) / 100;

p64 = 64 * 0.16;
p110 = 110 * 0.25;
p1518 = 1518 * 0.20;

% B - tamanho medio de pacotes
B = p64 + p110 + p1518;
for i=1:length(tam)
    B = B + (tam(i) * pres);
end
miu = (C) / (B * 8);

APD_mm1 = zeros(1, size(lambda, 2));
TT_mm1 = zeros(1, size(lambda, 2));

for i=1:size(lambda, 2)
    APD_mm1(i) = 1000 / (miu - lambda(i));
    TT_mm1(i) = (lambda(i) * (B * 8)) / 1e6;
end


% M/G/1 queueing model
disp('M/G/1 started');
```

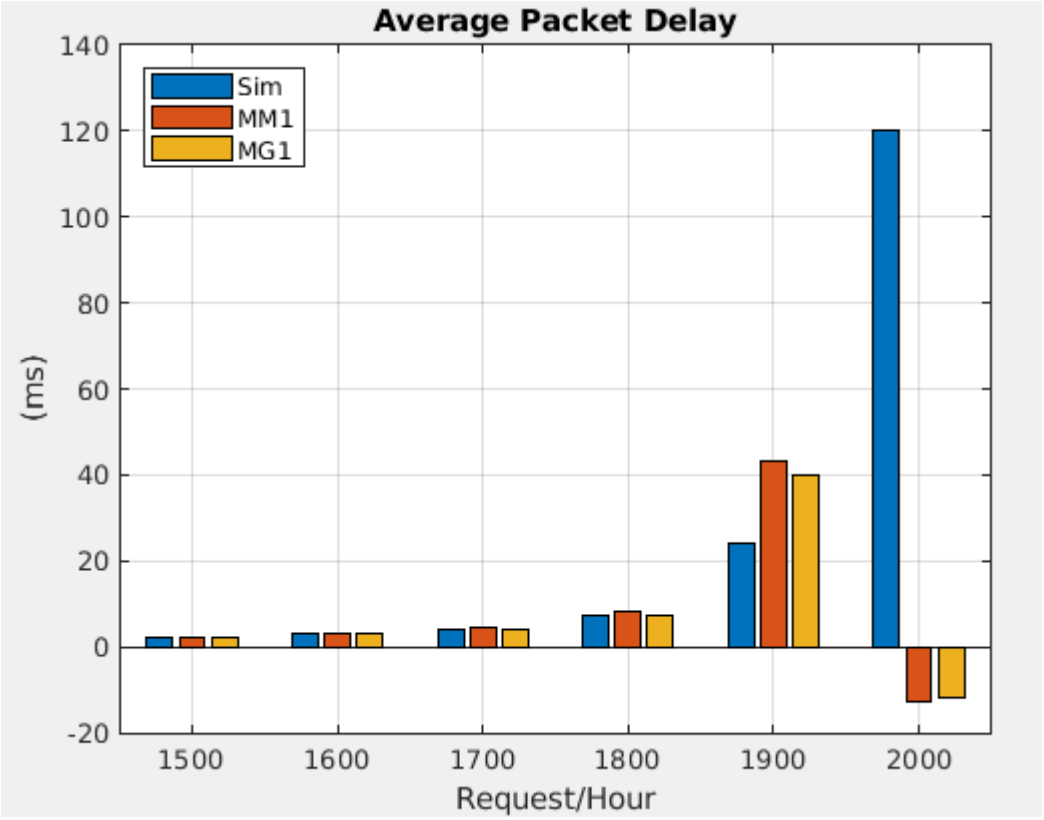
```
b = 0; % ber

P64 = (1 - b)^(8*64);
P110 = (1 - b)^(8*110);
P1518 = (1 - b)^(8*1518);

% avg packet delay
es64 = 0.16 * ((8 * 64) / C);
es110 = 0.25 * ((8 * 110) / C);
es1518 = 0.20 * ((8 * 1518) / C);
ess64 = 0.16 * ((8 * 64) / C)^2;
ess110 = 0.25 * ((8 * 110) / C)^2;
ess1518 = 0.20 * ((8 * 1518) / C)^2;

es = es64 + es110 + es1518;
ess = ess64 + ess110 + ess1518;

for i=1:length(tam)
    es = es + (pres * ((8 * tam(i)) / C));
    ess = ess + (pres * ((8 * tam(i)) / C)^2);
end


APD_mg1 = zeros(1, size(lambda, 2));
TT_mg1 = zeros(1, size(lambda, 2));

for i=1:size(lambda, 2)
    wq = (lambda(i) * ess) / (2 * (1 - lambda(i) * es));

    wi64 = wq + ((8*64)/C);
    wi110 = wq + ((8*110)/C);
    wi1518 = wq + ((8*1518)/C);

    apdUp64 = 0.16 * P64 * wi64;
    apdUp110 = 0.25 * P110 * wi110;
    apdUp1518 = 0.20 * P1518 * wi1518;
    apdDown64 = 0.16 * P64;
    apdDown110 = 0.25 * P110;
    apdDown1518 = 0.20 * P1518;

    apdUp = apdUp64 + apdUp110 + apdUp1518;
    apdDown = apdDown64 + apdDown110 + apdDown1518;

    for it=1:length(tam)
        Pi = (1 - b)^(8*tam(it));
        wi = wq + ((8*tam(it))/C);
```

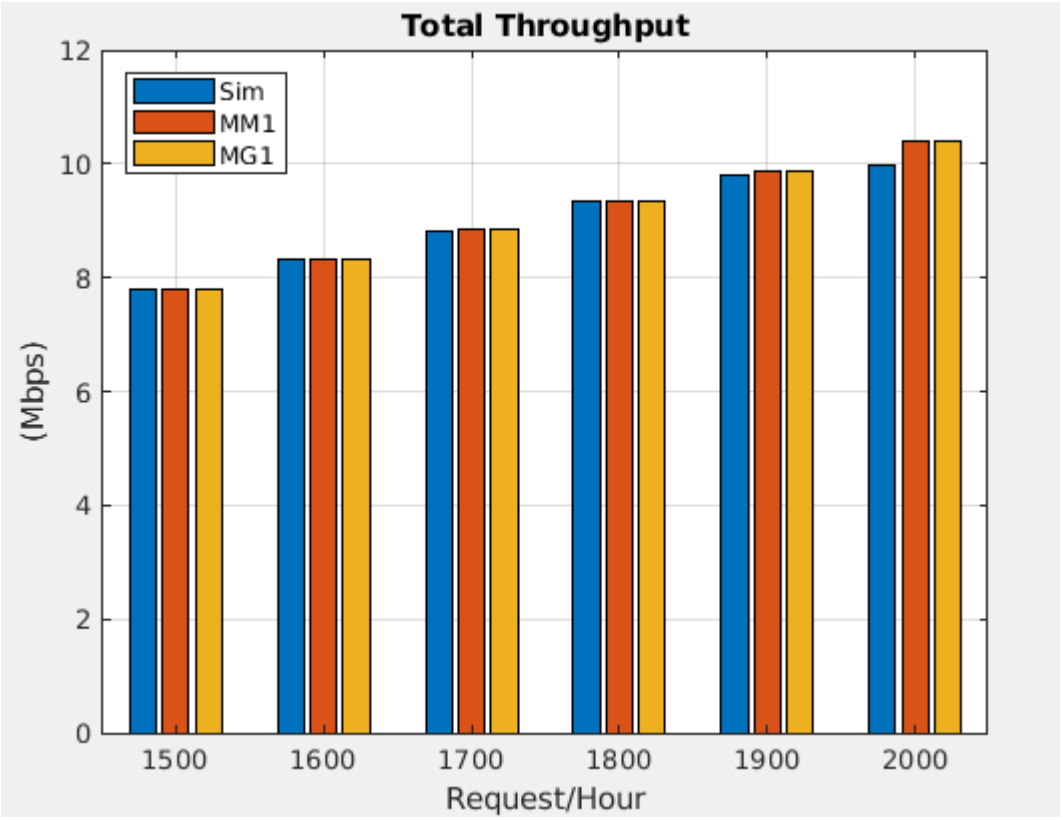
```
        apdUp = apdUp + pres * Pi * wi;
        apdDown = apdDown + pres * Pi;
    end

    APD_mg1(i) = 1e3 * (apdUp / apdDown);

    % total throughput
    T64= 0.16 * P64 * lambda(i)*8*64;
    T110= 0.25 * P110 * lambda(i)*8*110;
    T1518= 0.20 * P1518 * lambda(i)*8*1518;
    TT_mg1(i)= (T64 + T110 + T1518);

    for it=1:length(tam)
        Pi = (1 - b)^(8*tam(it));
        TT_mg1(i)= (TT_mg1(i) + (pres * Pi * lambda(i) * (8*tam(it))));
    end
    TT_mg1(i)= TT_mg1(i) / 1e6;
end

% plots
figure(1)
bar(lambda, [mediaAPD; APD_mm1; APD_mg1])
title('Average Packet Delay')
xlabel('Request/Second')
ylabel('(ms)')
legend('Sim', 'MM1', 'MG1', 'Location', 'northwest')
grid on

figure(2)
bar(lambda, [mediaTT; TT_mm1; TT_mg1])
title('Total Throughput')
xlabel('Request/Second')
ylabel('(Mbps)')
legend('Sim', 'MM1', 'MG1', 'Location', 'northwest')
grid on
```

Análise

Para esta alínea é necessário, analogamente às anteriores, calcular o valor de Average Packet Delay e Total Throughput através de simulação. Contudo, é necessário agora efetuar o cálculo destes valores através dos modelos teóricos *M/M/1* e *M/G/1*.

Resultados


Average Packet Delay
Sim
MM1
MG1
(ms)
140
120
100
80
60
40
20
0
-20
1500
1600
1700
1800
1900
2000
Request/Hour


Total Throughput
Sim
MM1
MG1
(Mbps)
12
10
8
6
4
2
0
1500
1600
1700
1800
1900
2000
Request/Hour

Conclusões

Para o *Average Packet Delay*, é possível concluir que existe uma subida gradual ao longe dos valores desde 1500 até 1800, uma subida maior para um número de *requests/Second* de 1900 e uma subida acentuada no valor da simulação para 2000 requests/Second. Ainda assim, os valores calculados a partir dos modelos teóricos apresentam valores reduzidos e negativos, o que pode indicar uma ultrapassagem do limite de pacotes por segundo dos limites dos modelos teóricos, apresentando assim valores sem sentido.

Quanto ao *Total Throughput*, à semelhança das alíneas que só abordaram cálculos por simulação, as subidas são graduais, tanto para o mesmo algoritmo de simulação como para modelos teóricos, desde 1500 até 2000 pedidos por segundo, como se tratasse de uma função linear. Regista-se apenas uma ligeira diferença do *throughput* para 2000 requests, tendo o valor simulado sido ligeiramente menor que os teóricos.

Pergunta 3.d

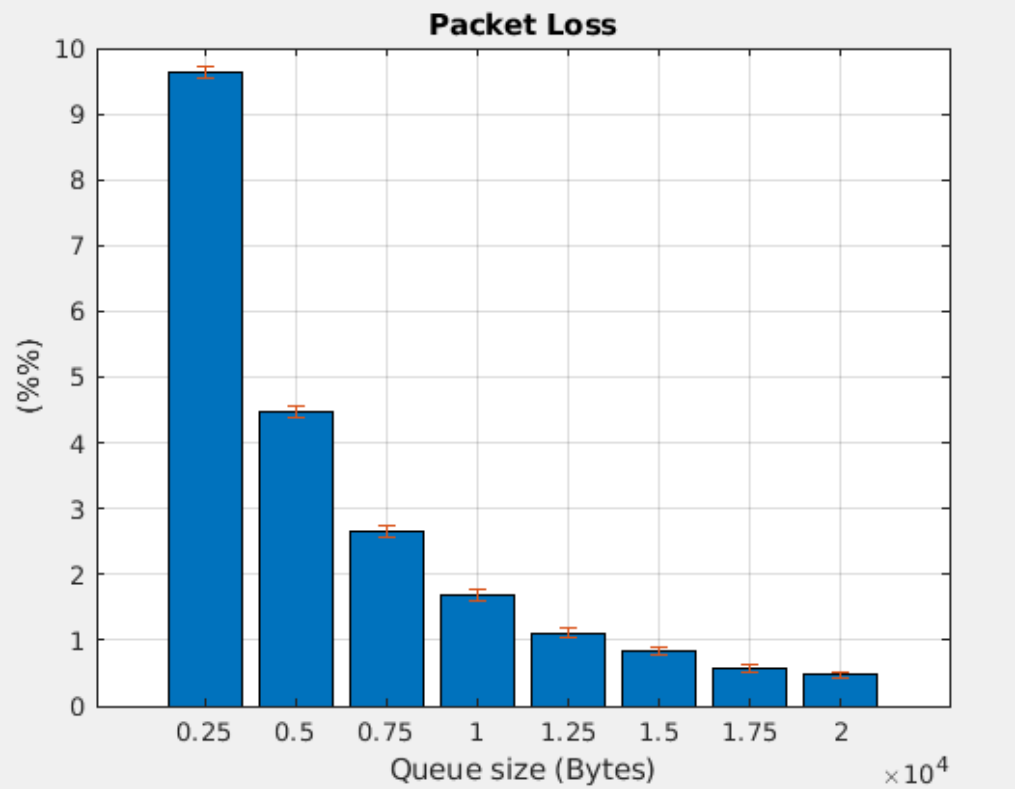
Código

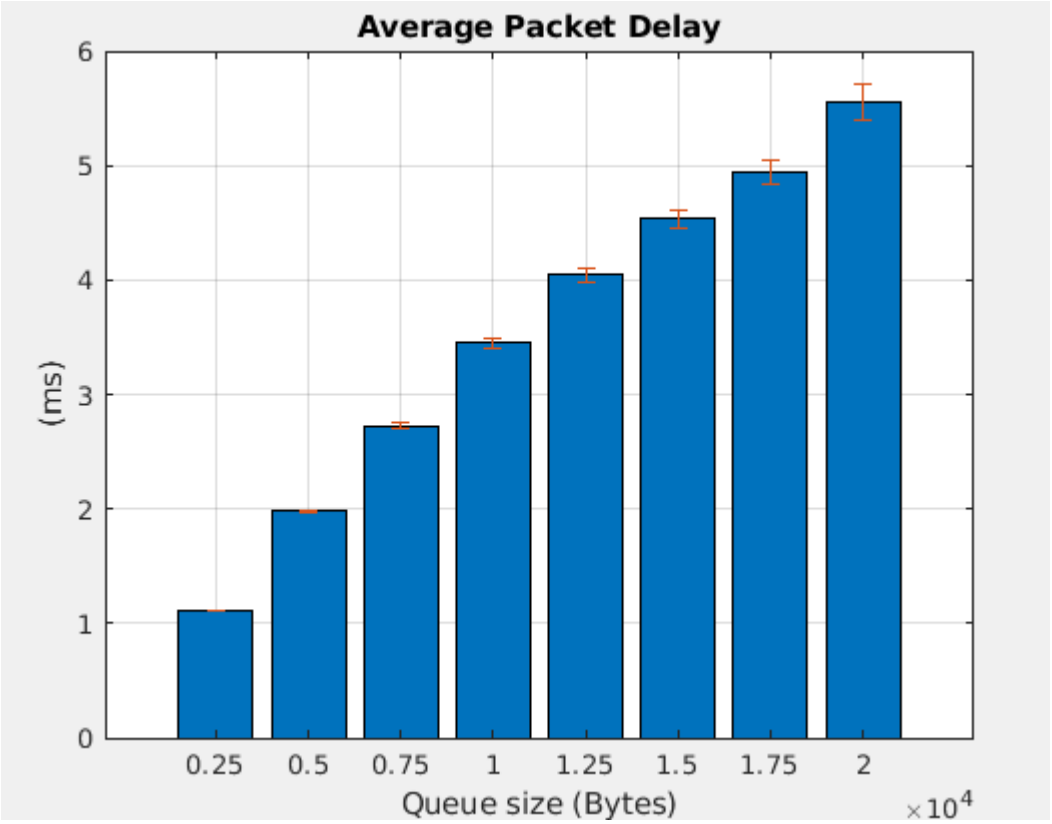
```
% d)
lambda = 1800;
C = 10;
P = 10000;
f = [2500, 5000, 7500, 10000, 12500, 15000, 17500, 20000];
b = 0; % d)
% b = 1e-5; % f)

% number of simulations
N = 40;

PL = zeros(1,N);
APD = zeros(1,N);
MPD = zeros(1,N);
TT = zeros(1,N);

mediaPL = zeros(1, size(f, 2));
termPL = zeros(1, size(f, 2));
mediaAPD = zeros(1, size(f, 2));
termAPD = zeros(1, size(f, 2));
mediaTT = zeros(1, size(f, 2));
termTT = zeros(1, size(f, 2));

for i=1:size(f, 2)
    for it=1:N
       [PL(it), APD(it), MPD(it), TT(it)] = simulator2(lambda, C, f(i),
P, b);
```

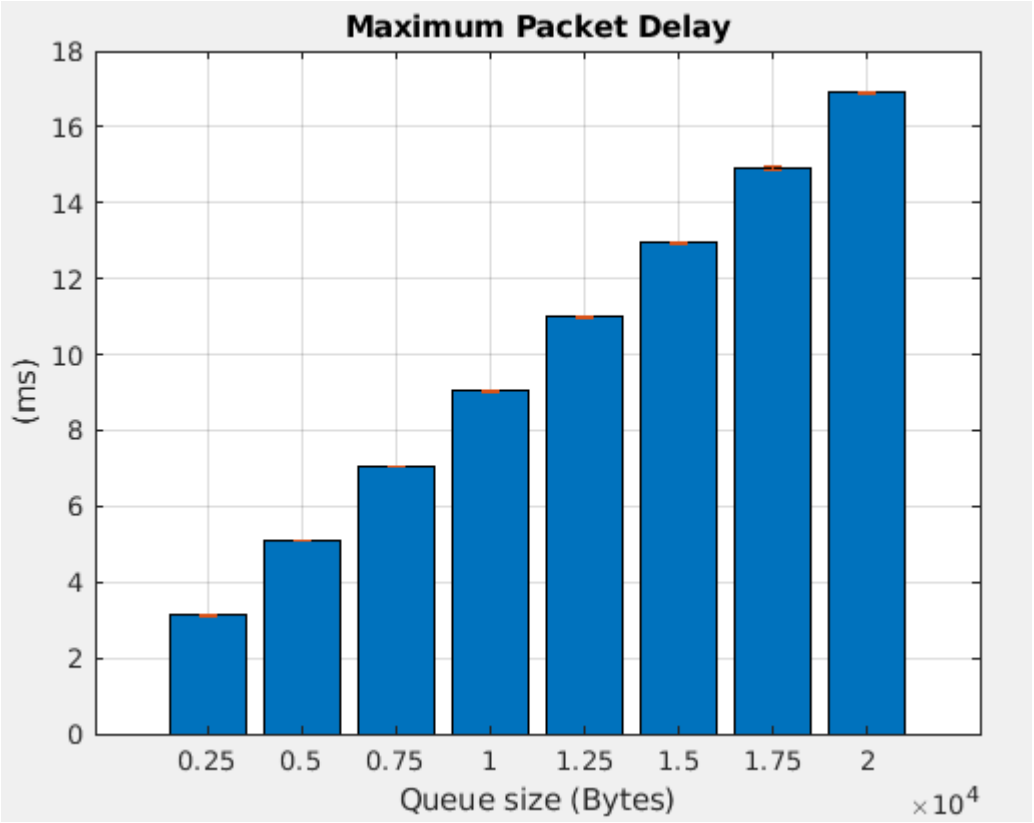
```
    end

    % 90 confidence interval %
    alfa= 0.1;

    mediaPL(i) = mean(PL);
    termPL(i) = norminv(1-alfa/2)*sqrt(var(PL)/N);

    mediaAPD(i) = mean(APD);
    termAPD(i) = norminv(1-alfa/2)*sqrt(var(APD)/N);

    mediaMPD(i) = mean(MPD);
    termMPD(i) = norminv(1-alfa/2)*sqrt(var(MPD)/N);

    mediaTT(i) = mean(TT);
    termTT(i) = norminv(1-alfa/2)*sqrt(var(TT)/N);
end

figure(1)
bar(f, mediaPL)
title('Packet Loss')
xlabel('Queue size (Bytes)')
ylabel('(%%)')
hold on
er = errorbar(f, mediaPL, termPL, termPL);
er.LineStyle = 'none';
hold off
grid on

figure(2)
bar(f, mediaAPD)
title('Average Packet Delay')
xlabel('Queue size (Bytes)')
ylabel('(ms)')
hold on
er = errorbar(f, mediaAPD, termAPD, termAPD);
er.LineStyle = 'none';
hold off
grid on

figure(3)
bar(f, mediaMPD)
title('Maximum Packet Delay')
xlabel('Queue size (Bytes)')
ylabel('(ms)')
hold on
```

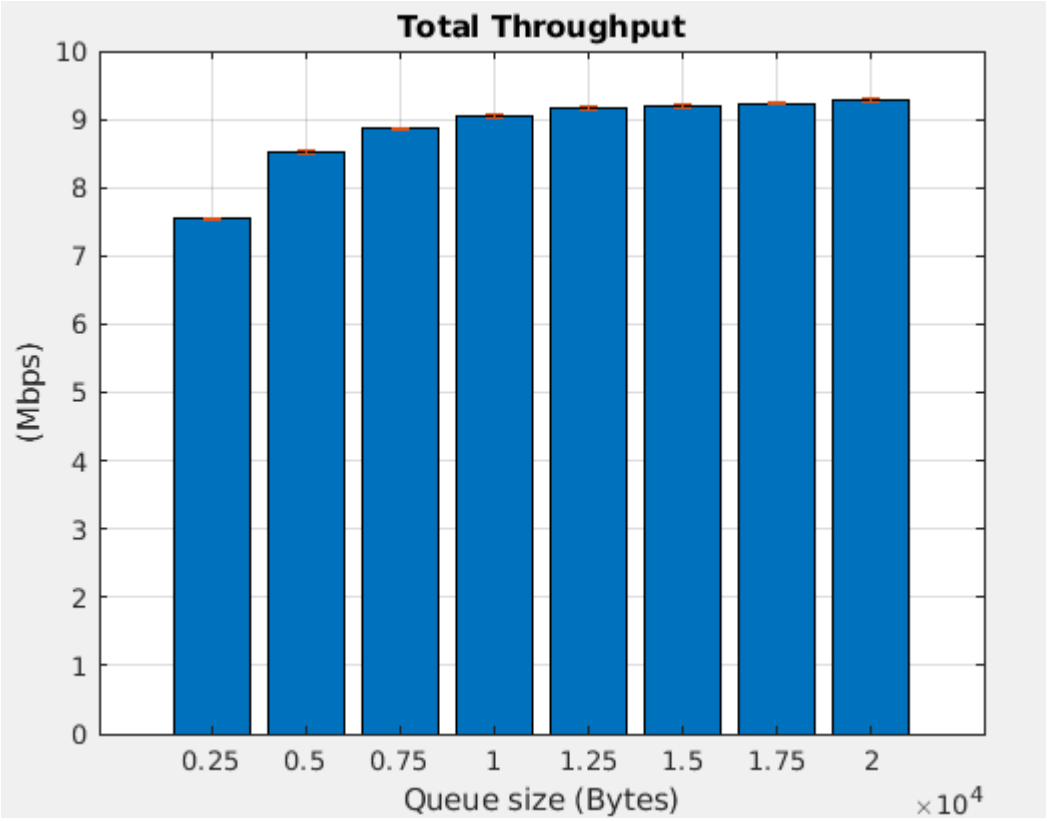
```
er = errorbar(f, mediaMPD, termMPD, termMPD);
er.LineStyle = 'none';
hold off
grid on

figure(4)
bar(f, mediaTT)
title('Total Throughput')
xlabel('Queue size (Bytes)')
ylabel('(Mbps)')
hold on
er = errorbar(f, mediaTT, termTT, termTT);
er.LineStyle = 'none';
hold off
grid on
```

Resultados




Average Packet Delay
(ms)
0
1
2
3
4
5
6
0.25
0.5
0.75
1
1.25
1.5
1.75
2
Queue size (Bytes)
×10^4


Maximum Packet Delay
(ms)
0
2
4
6
8
10
12
14
16
18
0.25
0.5
0.75
1
1.25
1.5
1.75
2
Queue size (Bytes)
×10^4



Conclusões

Como neste panorama estamos a utilizar filas de tamanho realistas e, portanto, limitados, conseguimos analisar resultados diferentes das alíneas anteriores. Esta característica leva à existência de pacotes perdidos (aqueles recebidos após a fila estar preenchida na totalidade) e à redução dos atrasos dos pacotes.
Ainda assim, de forma a tentar minimizar a perda de pacotes, o aumento da capacidade da fila resulta em maiores atrasos (médios e máximos) pois existem mais pacotes à espera.

Pergunta 3.e

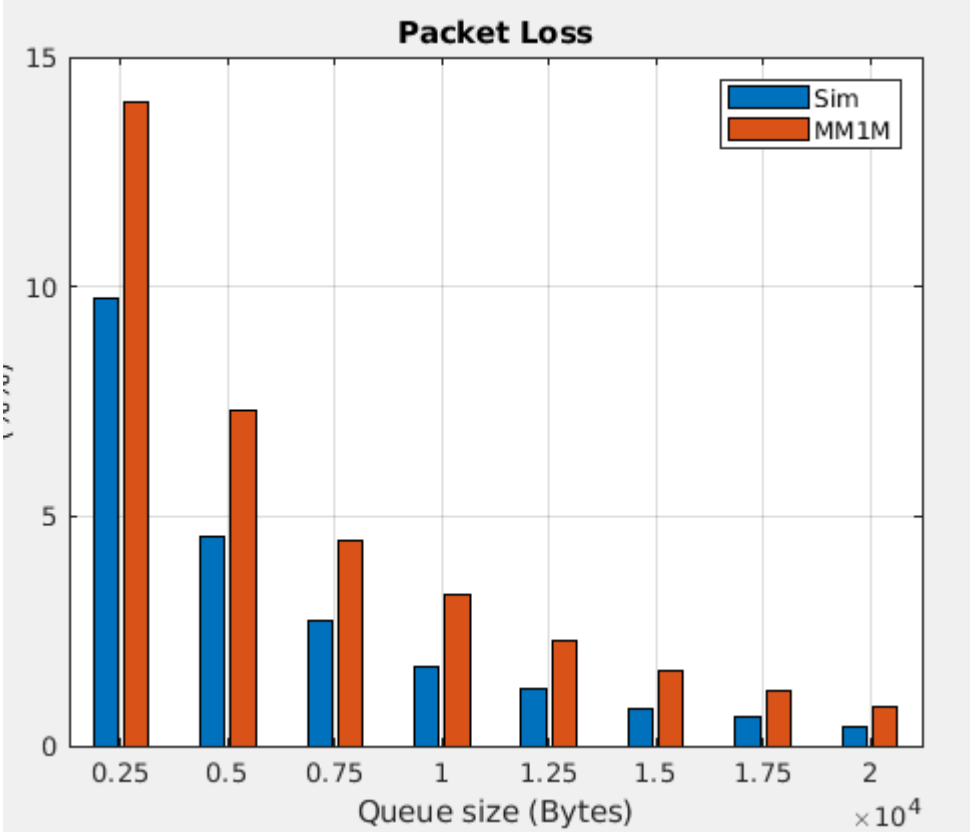
Código

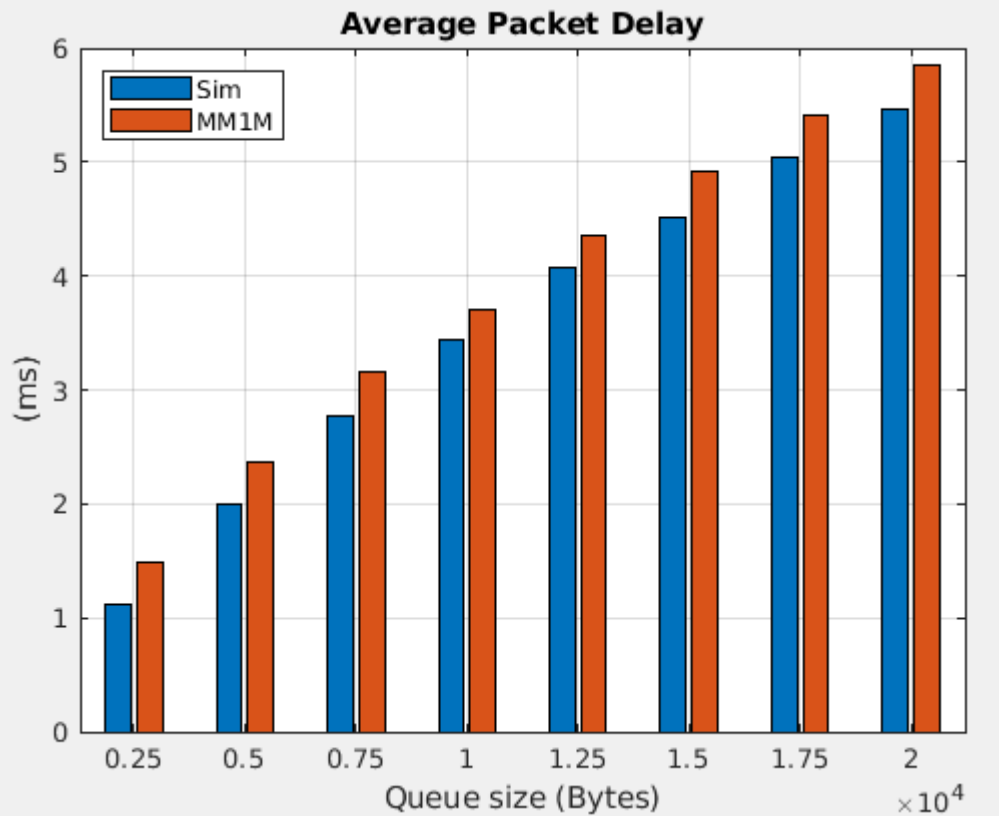
```
% e)
lambda = 1800;
C = 10;
P = 10000;
f = [2500, 5000, 7500, 10000, 12500, 15000, 17500, 20000];
b = 0;
```

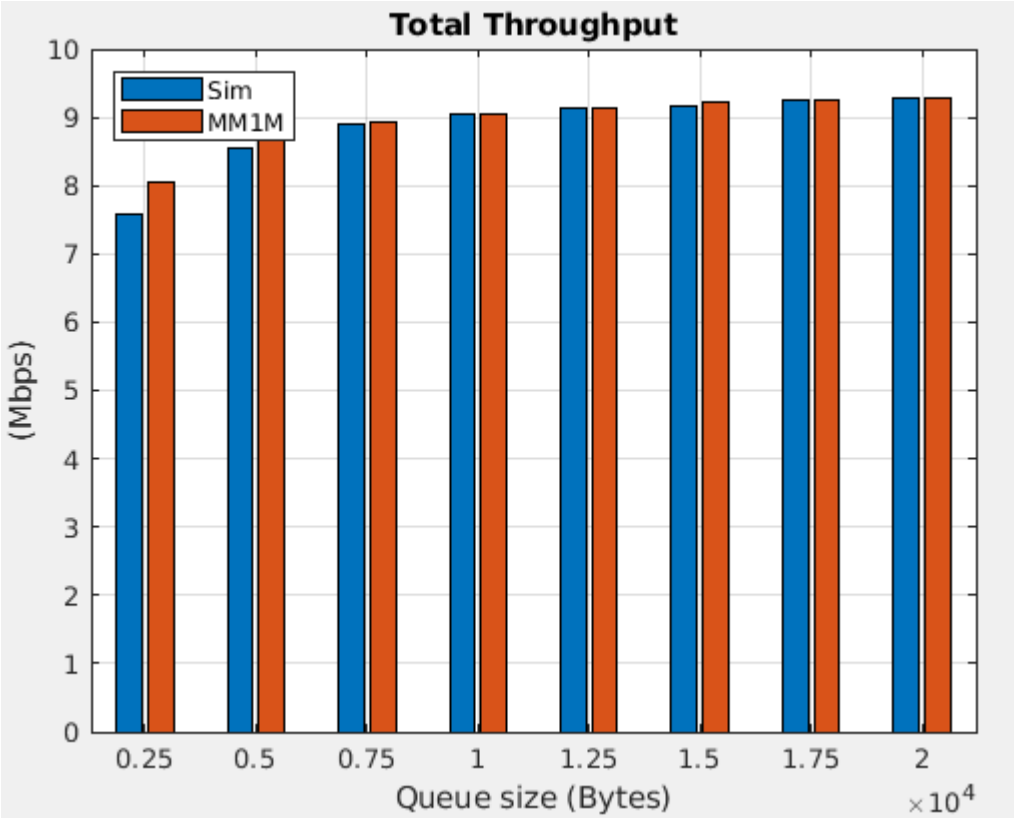
```
disp('Simulation started');
% number of simulations
N = 40;

PL = zeros(1,N);
APD = zeros(1,N);
MPD = zeros(1,N);
TT = zeros(1,N);

mediaPL = zeros(1, size(f, 2));
mediaAPD = zeros(1, size(f, 2));
mediaTT = zeros(1, size(f, 2));

for i=1:size(f, 2)
    for it=1:N
       [PL(it), APD(it), MPD(it), TT(it)] = simulator2(lambda, C, f(i),
P, b);
    end

    mediaPL(i) = mean(PL);
    mediaAPD(i) = mean(APD);
    mediaTT(i) = mean(TT);
end


% M/M/1/m queueing model
disp('MM1m started')
PL_mm1m = zeros(1, size(f,2));
APD_mm1m = zeros(1, size(f,2));
TT_mm1m = zeros(1, size(f,2));

aux2= [65:109 111:1517];
pres = (1-(0.2+0.16+0.25))/length(aux2);

B = 0.16*64 + 0.25*110 + 0.20*1518;
for i = 1:length(aux2)
    B = B + (aux2(i)*pres);
end

for i=1:size(f, 2)
    m = round(f(i)/B)+1;
    miu=(C*1e6)/(B*8);
    numerador = 0;
    denominador = 0;

    for j=0:m
```

```
        denominador = denominador + (lambda/miu)^j;
        numerador = numerador + j*((lambda/miu)^j);
    end

    PL_mm1m(i) = (((lambda/miu)^m)/denominador);
    L = numerador/denominador;
    APD_mm1m(i) = (L/(lambda*(1-PL_mm1m(i))));

    TT_mm1m(i) = 0.16*lambda*(8*64) + 0.25*lambda*(8*110) +
0.20*lambda*(8*1518);
    for j = 1:length(aux2)
        TT_mm1m(i) = TT_mm1m(i) + pres*lambda*(8*aux2(j));
    end
    TT_mm1m(i) = TT_mm1m(i)*(1-PL_mm1m(i));

    PL_mm1m(i) = PL_mm1m(i) * 100;
    APD_mm1m(i) = APD_mm1m(i) * 1000;
    TT_mm1m(i) = TT_mm1m(i) / (1e6);
end


% plots
figure(6)
bar(f, [mediaPL; PL_mm1m])
title('Packet Loss')
legend('Sim', 'MM1M', 'Location', 'northeast')
xlabel('Queue size (Bytes)')
ylabel('(%%)')
grid on

figure(7)
bar(f, [mediaAPD; APD_mm1m])
title('Average Packet Delay')
xlabel('Queue size (Bytes)')
ylabel('(ms)')
legend('Sim', 'MM1M', 'Location', 'northwest')
grid on

figure(8)
bar(f, [mediaTT; TT_mm1m])
title('Total Throughput')
xlabel('Queue size (Bytes)')
ylabel('(Mbps)')
legend('Sim', 'MM1M', 'Location', 'northwest')
grid on
```

Resultados


Packet Loss
Sim
MM1M
15
10
5
0
0.25
0.5
0.75
1
1.25
1.5
1.75
2
Queue size (Bytes)
×10^4


Average Packet Delay
Sim
MM1M
(ms)
6
5
4
3
2
1
0
0.25
0.5
0.75
1
1.25
1.5
1.75
2
Queue size (Bytes)
×10^4



Conclusões

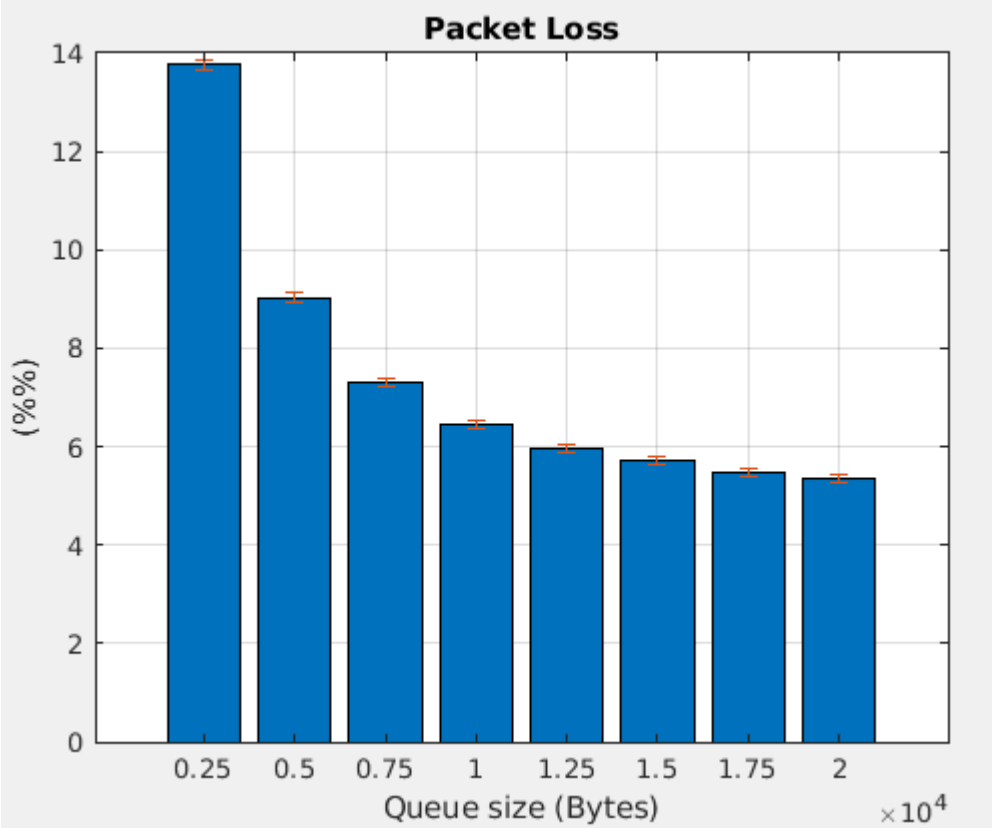
Após analisar os resultados da alínea e), é possível concluir apenas as perdas de pacotes divergem mais entre os resultados de simulação e os teóricos, embora o comportamento (neste caso, o decréscimo) se verifique para ambas as vertentes. As outras características mantêm valores aproximados.
A discrepância para o *Packet Loss* relaciona-se com o facto de utilizarmos a média do tamanho de pacotes para os cálculos teóricos, e isso poder resultado num preenchimento total da fila mais rápido, que resulta em maiores perdas, como vimos já noutras alíneas.

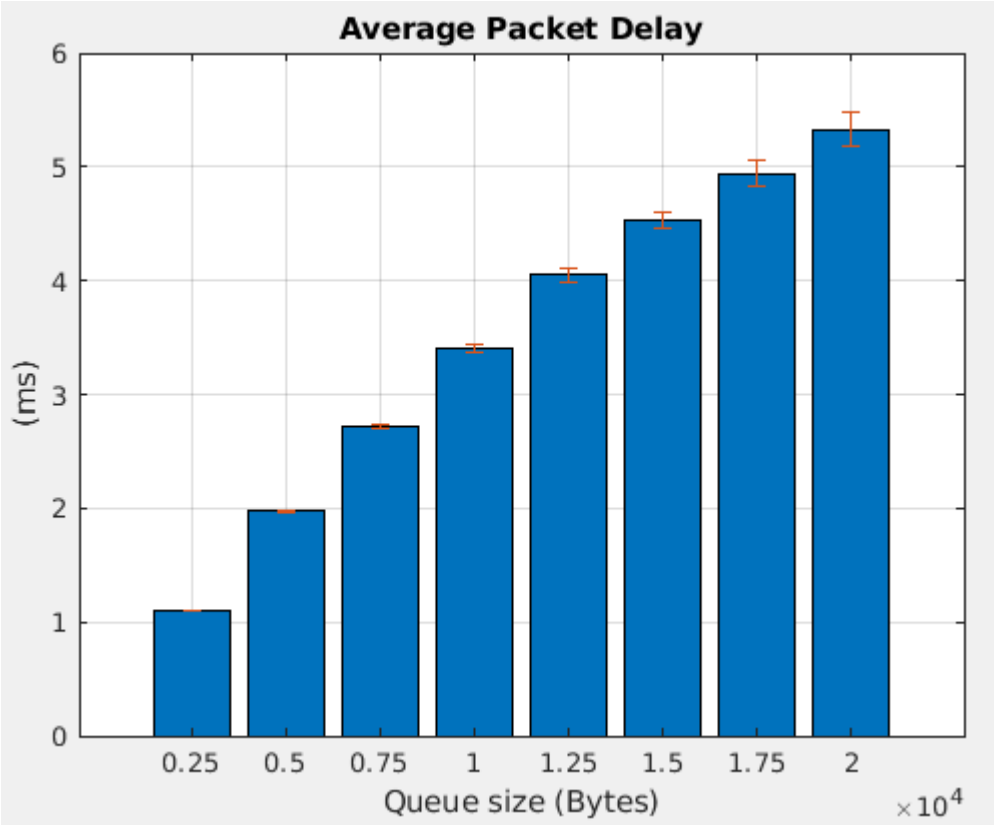
Pergunta 3.f

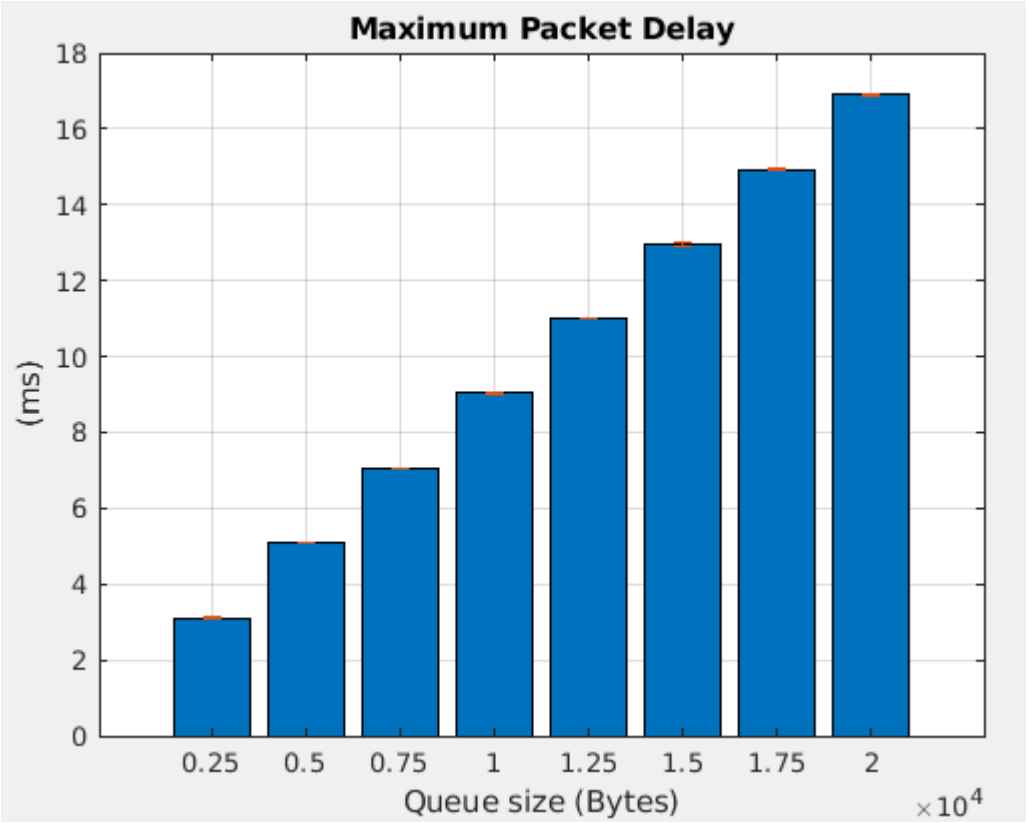
Código

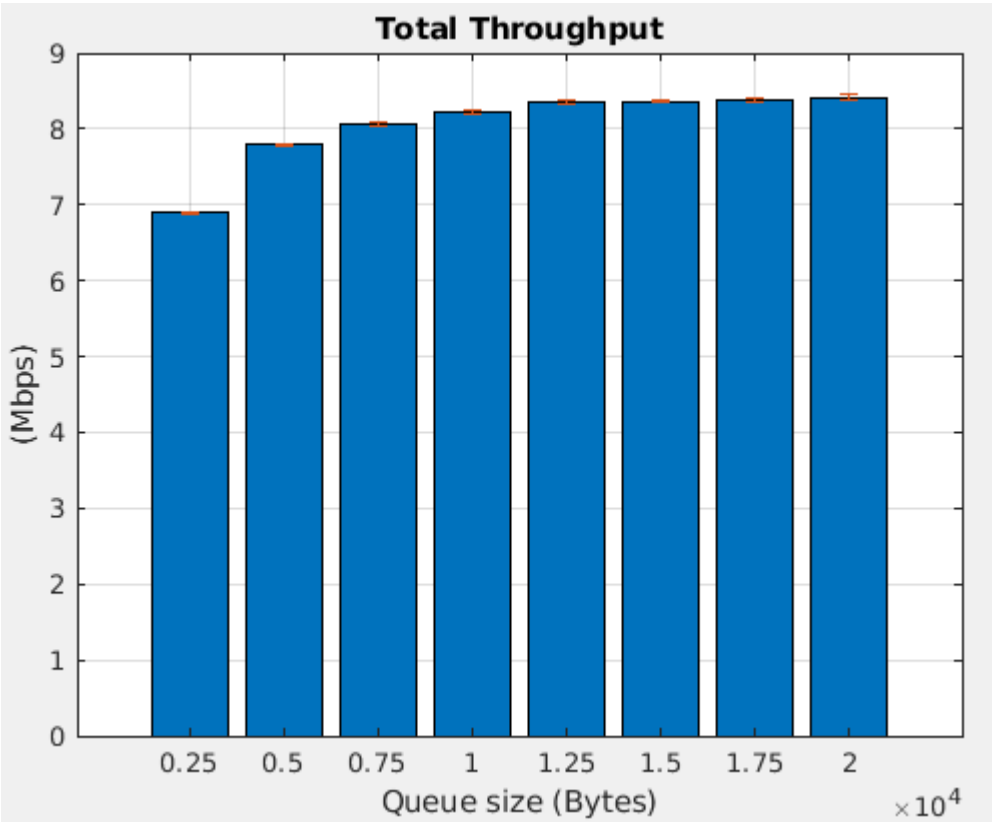
```
b = 1e-5; % f)
```

Resultados


Packet Loss
14
12
10
8
6
4
2
0
(%%)
0.25
0.5
0.75
1
1.25
1.5
1.75
2
Queue size (Bytes)
×10^4


Average Packet Delay
6
5
4
3
2
1
0
(ms)
0.25
0.5
0.75
1
1.25
1.5
1.75
2
Queue size (Bytes)
×10^4


Maximum Packet Delay
18
16
14
12
10
8
6
4
2
0
(ms)
0.25
0.5
0.75
1
1.25
1.5
1.75
2
Queue size (Bytes)
×10^4


Total Throughput
9
8
7
6
5
4
3
2
1
0
(Mbps)
0.25
0.5
0.75
1
1.25
1.5
1.75
2
Queue size (Bytes)
×10^4

Conclusões

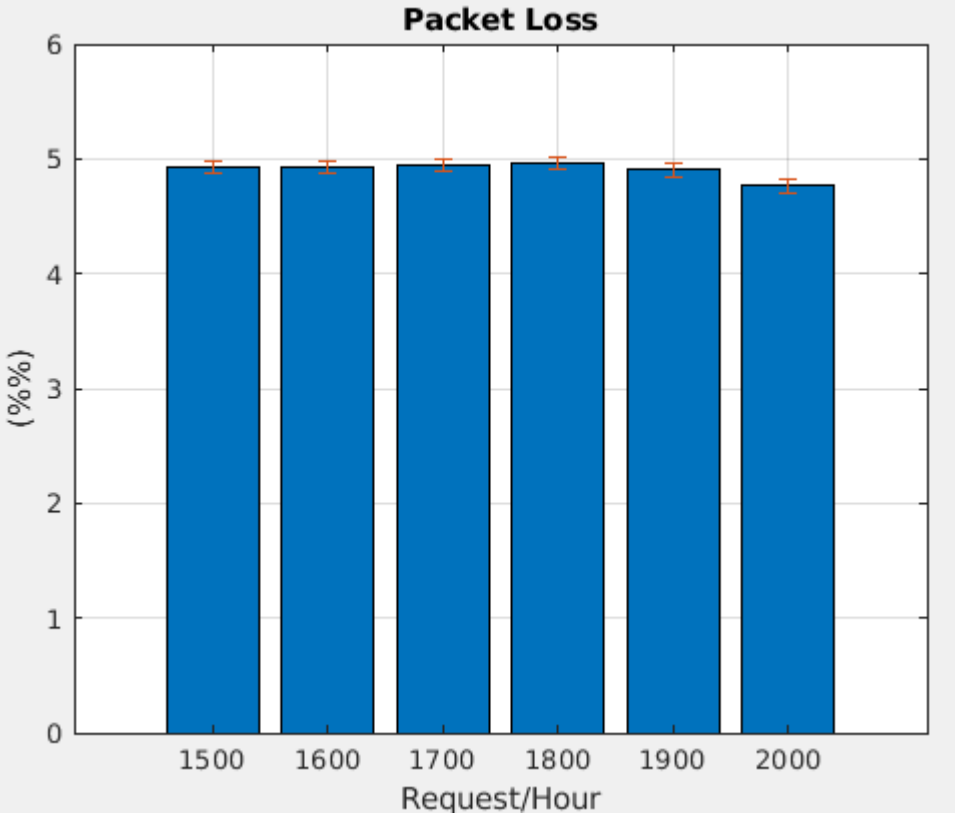
Com a introdução do ber, a perda de pacotes teve um aumento significativo independentemente do tamanho da fila de espera. Isto acontece pois o bit error rate introduz a possibilidade de haver erros na transmissão dos pacotes. Uma consequência do aumento da perda de pacotes é a diminuição do throughput.

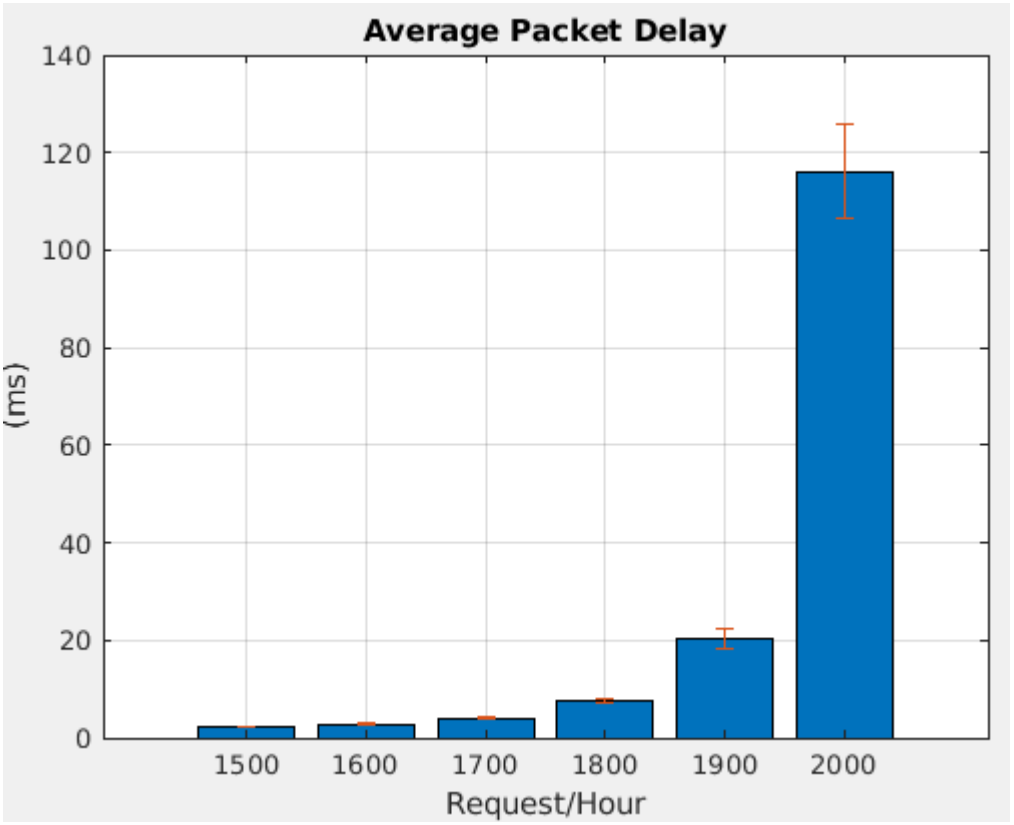
Pergunta 3.g

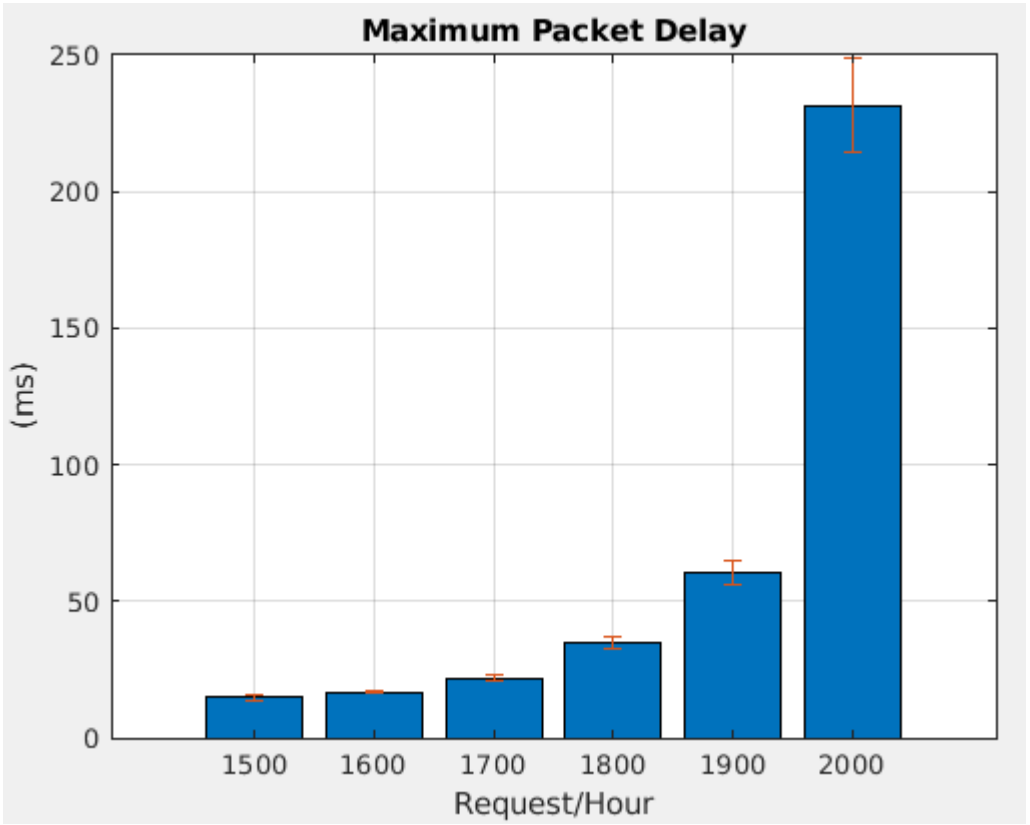
Código

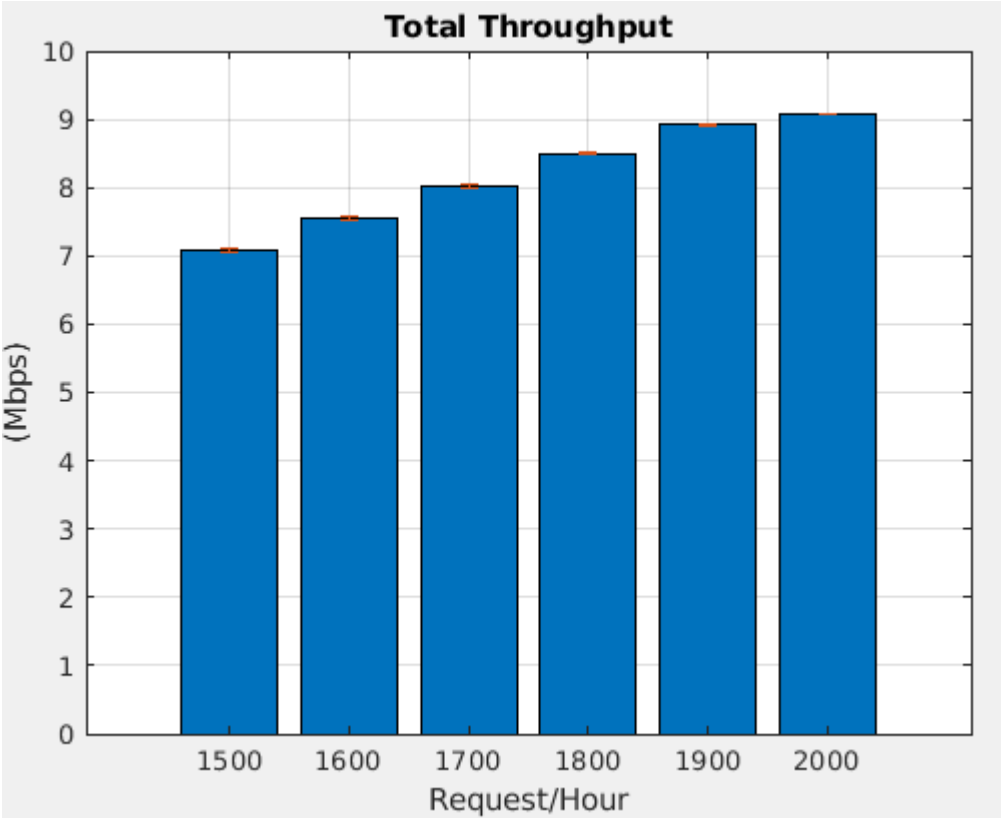
```
b = 1e-5; % g
```

Resultados




Average Packet Delay
140
120
100
80
60
40
20
0
(ms)
1500
1600
1700
1800
1900
2000
Request/Hour


Maximum Packet Delay
250
200
150
100
50
0
(ms)
1500
1600
1700
1800
1900
2000
Request/Hour



Conclusões

A maior diferença nos resultados obtidos agora e entre a pergunta 3.b é a perda de pacotes. Como foi analisado, na pergunta 3.b não havia perda de pacotes, mas visto que foi introduzido o valor BER, agora podem surgir erros na transmissão de pacotes.

Uma outra conclusão é que agora, com a existência de perda de pacotes, o throughput baixou ligeiramente.

Quanto ao tempo médio e máximo de atraso de pacotes não houve alterações.

Pergunta 3.h

Código

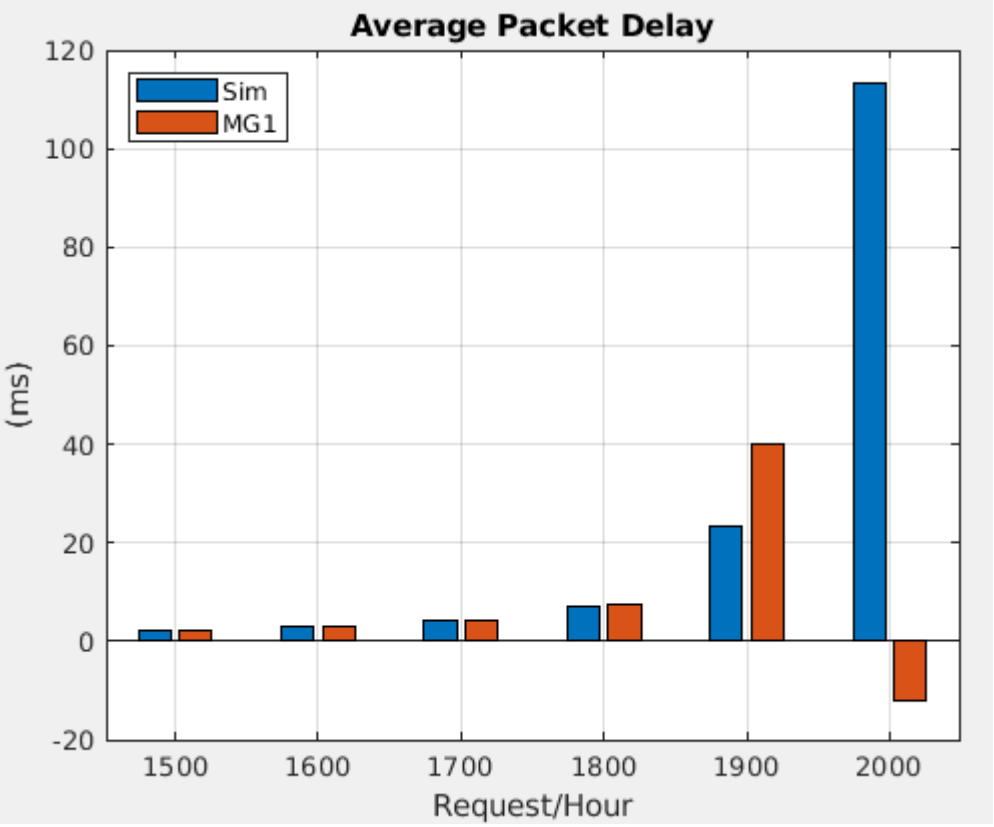
```
%h
lambda = [1500, 1600, 1700, 1800, 1900, 2000];
C = 10 * 1e6;
P = 10000;
f = 10000000;
b = 1e-5; % g

% number of simulations
N = 40; % b
```

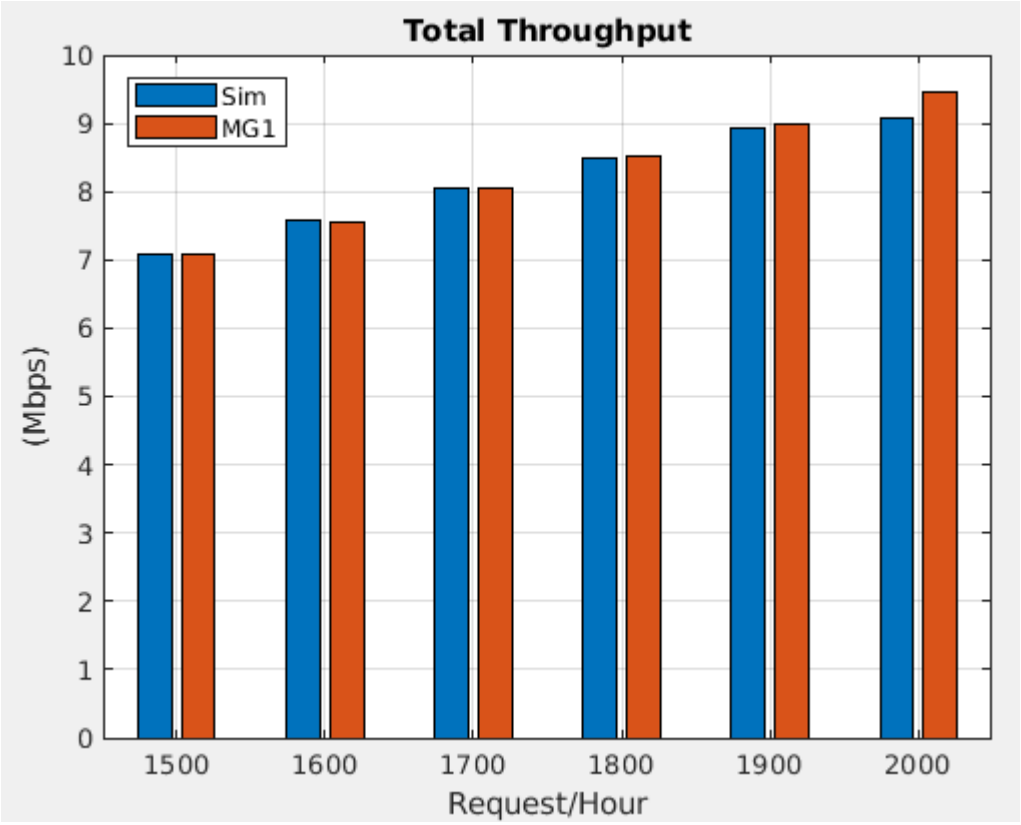
```
PL = zeros(1,N);
APD = zeros(1,N);
MPD = zeros(1,N);
TT = zeros(1,N);

mediaPL = zeros(1, size(lambda, 2));
mediaAPD = zeros(1, size(lambda, 2));
mediaMPD = zeros(1, size(lambda, 2));
mediaTT = zeros(1, size(lambda, 2));

disp('Simulation started...');
for i=1:size(lambda, 2)
    for it=1:N
       [PL(it), APD(it), MPD(it), TT(it)] = simulator2(lambda(i), 10, f,
P, b);
    end

    mediaPL(i) = mean(PL);
    mediaAPD(i) = mean(APD);
    mediaMPD(i) = mean(MPD);
    mediaTT(i) = mean(TT);

end

% M/G/1 queueing model
disp('M/G/1 started...');

P64 = (1 - b)^(8*64);
P110 = (1 - b)^(8*110);
P1518 = (1 - b)^(8*1518);

% avg packet delay
es64 = 0.16 * ((8 * 64) / C);
es110 = 0.25 * ((8 * 110) / C);
es1518 = 0.20 * ((8 * 1518) / C);
ess64 = 0.16 * ((8 * 64) / C)^2;
ess110 = 0.25 * ((8 * 110) / C)^2;
ess1518 = 0.20 * ((8 * 1518) / C)^2;

es = es64 + es110 + es1518;
ess = ess64 + ess110 + ess1518;

for i=1:length(tam)
    es = es + (pres * ((8 * tam(i)) / C));
    ess = ess + (pres * ((8 * tam(i)) / C)^2);
end
```

```
APD_mg1 = zeros(1, size(lambda, 2));
TT_mg1 = zeros(1, size(lambda, 2));

for i=1:size(lambda, 2)
    wq = (lambda(i) * ess) / (2 * (1 - lambda(i) * es));

    wi64 = wq + ((8*64)/C);
    wi110 = wq + ((8*110)/C);
    wi1518 = wq + ((8*1518)/C);

    apdUp64 = 0.16 * P64 * wi64;
    apdUp110 = 0.25 * P110 * wi110;
    apdUp1518 = 0.20 * P1518 * wi1518;
    apdDown64 = 0.16 * P64;
    apdDown110 = 0.25 * P110;
    apdDown1518 = 0.20 * P1518;

    apdUp = apdUp64 + apdUp110 + apdUp1518;
    apdDown = apdDown64 + apdDown110 + apdDown1518;

    for it=1:length(tam)
        Pi = (1 - b)^(8*tam(it));
        wi = wq + ((8*tam(it))/C);
        apdUp = apdUp + pres * Pi * wi;
        apdDown = apdDown + pres * Pi;
    end

    APD_mg1(i) = 1e3 * (apdUp / apdDown);

    % total throughput
    T64= 0.16 * P64 * lambda(i)*8*64;
    T110= 0.25 * P110 * lambda(i)*8*110;
    T1518= 0.20 * P1518 * lambda(i)*8*1518;
    TT_mg1(i)= (T64 + T110 + T1518);

    for it=1:length(tam)
        Pi = (1 - b)^(8*tam(it));
        TT_mg1(i)= (TT_mg1(i) + (pres * Pi * lambda(i) * (8*tam(it))));
    end
    TT_mg1(i)= TT_mg1(i) / 1e6;
end

% graficos
figure(1)
```

```
bar(lambda, [mediaAPD; APD_mg1])
title('Average Packet Delay')
xlabel('Request/Second')
ylabel('(ms)')
legend('Sim', 'MG1', 'Location', 'northwest')
grid on

figure(2)
bar(lambda, [mediaTT; TT_mg1])
title('Total Throughput')
xlabel('Request/Second')
ylabel('(Mbps)')
legend('Sim', 'MG1', 'Location', 'northwest')
grid on
```

Resultados





Conclusões

Depois de uma análise dos resultados gerados acima, conclui-se que a comparação entre os valores de *Total Throughput* é simples, sendo os valores bastante semelhantes, variando ligeiramente para os 2000 requests/Second.

Para o atraso médio de pacotes, os valores são equivalentes até aos 1900 pedidos por segundo, onde é possível verificar um efeito do cálculo médio dos pacotes para um maior número de pedidos, devido ao aumento de atraso para os resultados provenientes do modelo teórico.
Para o máximo de 2000 pedidos por segundo volta a acontecer o resultado negativo de *delay* devido à transmissão de um número de pacotes superior à média o que, no cálculo relativo ao modelo matemático, apresenta valores negativos.

Pergunta 4

Nas alíneas a) e b) servem maioritariamente para verificar que o *Simulador3* funciona corretamente.

Simulador 3

As alterações feitas ao *Simulador2* apresentam-se devidamente comentadas.

```
function [PL , APD , MPD , TT] = Simulator3(lambda,C,f,P,b)
% INPUT PARAMETERS:
%  lambda - packet rate (packets/sec)
%  C      - link bandwidth (Mbps)
%  f      - queue size (Bytes)
%  P      - number of packets (stopping criterium)
%  b      - Bit error rate
% OUTPUT PARAMETERS:
%  PL   - packet loss (%)
%  APD  - average packet delay (milliseconds)
%  MPD  - maximum packet delay (milliseconds)
%  TT   - transmitted throughput (Mbps)

 denominador = (1 + (10/5) + ((10/5)*(5/10)));

 % prob de cada estado
 s1 = 1 / denominador;
 s2 = (10/5) / denominador;
 s3 = ((10/5)*(5/10)) / denominador;

 % tempo de permanencia em cada estado
 t1 = 1 / 10;
 t2 = 1 / (5 + 5);
 t3 = 1 / 10;

%Events:
ARRIVAL= 0;          % Arrival of a packet
DEPARTURE= 1;        % Departure of a packet
TRANSITION= 2;       % transition of a state in the packet arriving
Markov chain

%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
```

```
%Auxiliary variables:
% Initializing the simulation clock:
Clock= 0;

rate = 0;   % "lambda" consoante o estado

temp = rand;
% escolhe estado inicial, atualiza o rate adequado ao estado e "agenda"
o
% novo evento de transicao
if temp <= s1
    FlowState= 1;
    rate = lambda * 0.5;
    EventList = [TRANSITION, Clock + exprnd(t1), 0, 0];
elseif temp <= s1 + s2
    FlowState= 2;
    rate = lambda;
    EventList = [TRANSITION, Clock + exprnd(t2), 0, 0];
else
    FlowState= 3;
    rate = lambda * 1.5;
    EventList = [TRANSITION, Clock + exprnd(t3), 0, 0];
end

% Initializing the List of Events with the first ARRIVAL:
EventList = [EventList; ARRIVAL, Clock + exprnd(1/rate),
GeneratePacketSize(), 0];


%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;
            EventList = [EventList; ARRIVAL, Clock + exprnd(1/rate),
GeneratePacketSize(), 0];
            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
            error = rand() > ((1 - b)^(8*PacketSize));
            if error
                LOSTPACKETS= LOSTPACKETS + 1;
            else
                TRANSMITTEDBYTES= TRANSMITTEDBYTES + PacketSize;
                DELAYS= DELAYS + (Clock - ArrivalInstant);
                if Clock - ArrivalInstant > MAXDELAY
                    MAXDELAY= Clock - ArrivalInstant;
                end
                TRANSMITTEDPACKETS= TRANSMITTEDPACKETS + 1;
            end

            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

        case TRANSITION     % If first event is a TRANSITION
            % se o estado é diferente do 2, entao o novo estado é o dois
            % atualiza o rate adequado ao estado
            % e "agenda" novo evento de transicao
            if FlowState ~= 2   % transiçao para os estados 1 e 3
                % so pode ir para s2
                FlowState = 2;
                rate = lambda;
                EventList = [EventList; TRANSITION, Clock + exprnd(t2),
0, 0];

            else % FlowState == 2
                % decidir se vai para s1 ou s3
                temp = rand;
```

```
                if temp < 0.5   % taxa de transicao e igual para os dois
estados possiveis (ou seja 50-50)
                    FlowState = 1;
                    EventList = [EventList; TRANSITION, Clock +
exprnd(t1), 0, 0];
                    rate = lambda * 0.5;
                else
                    FlowState = 3;
                    EventList = [EventList; TRANSITION, Clock +
exprnd(t3), 0, 0];
                    rate = lambda * 1.5;
                end
            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 <= 0.16
        out= 64;
    elseif aux <= 0.16 + 0.25
        out= 110;
    elseif aux <= 0.16 + 0.25 + 0.2
        out= 1518;
    else
        out = aux2(randi(length(aux2)));
    end
end
```

Pergunta 4.a

Código

```
lambda = 1800;  % lambda - packet rate (packets/sec)
C = 10;         % link bandwidth (Mbps)
P = 100000;     % number of packets (stopping criterium)
f = 1e6; % a
b = 0;   % a

N = 10;  % number of simulations

PL = zeros(1,N);
APD = zeros(1,N);
MPD = zeros(1,N);
TT = zeros(1,N);

for it= 1:N
    [PL(it), APD(it), MPD(it), TT(it)] = simulator3(lambda, C, f, P, b);
end

% 90confidence interval %
alfa= 0.1;

mediaPL = mean(PL);
termPL = norminv(1-alfa/2)*sqrt(var(PL)/N);

mediaAPD = mean(APD);
termAPD = norminv(1-alfa/2)*sqrt(var(APD)/N);

mediaMPD = mean(MPD);
termMPD = norminv(1-alfa/2)*sqrt(var(MPD)/N);

mediaTT = mean(TT);
termTT = norminv(1-alfa/2)*sqrt(var(TT)/N);

fprintf('\nPacket Loss: %.4e + %.4e\n', mediaPL, termPL);
fprintf('Avg packet delay (ms): %.4e + %.4e\n', mediaAPD, termAPD);
fprintf('Max packet delay (ms): %.4e + %.4e\n', mediaMPD, termMPD);
fprintf('Throughput (Mbps): %.4e +- %.4e\n', mediaTT, termTT);
```

Resultados

```
Packet Loss: 0.0000e+00 + 0.0000e+00
Avg packet delay (ms): 1.4391e+02 + 2.7045e+01
Max packet delay (ms): 5.7777e+02 + 7.8117e+01
Throughtput (Mbps): 9.3200e+00 +- 1.0319e-01
```

Pergunta 4.b

Código

```
f = 1e4; % b
b = 1e-5; % b
```

Resultados

```
Packet Loss: 1.0730e+01 + 2.2296e-01
Avg packet delay (ms): 4.2582e+00 + 8.2765e-02
Max packet delay (ms): 9.1777e+00 + 1.3495e-02
Throughtput (Mbps): 7.5347e+00 +- 8.3868e-02
```

Pergunta 4.c

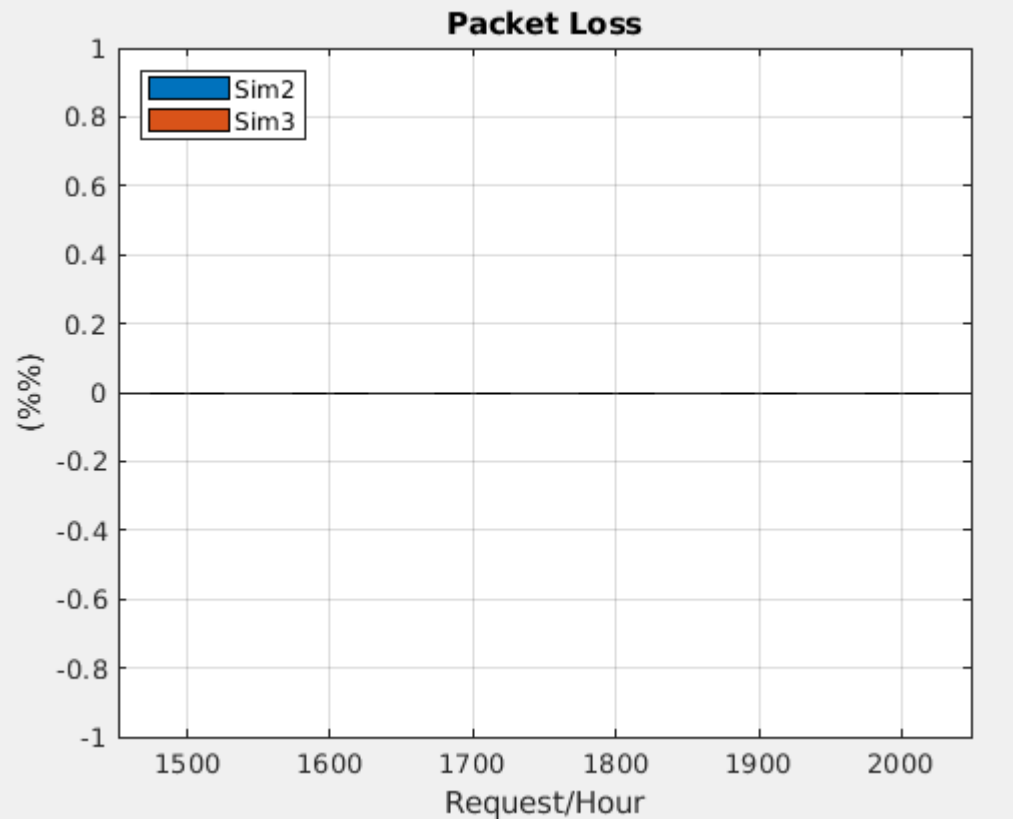
Código

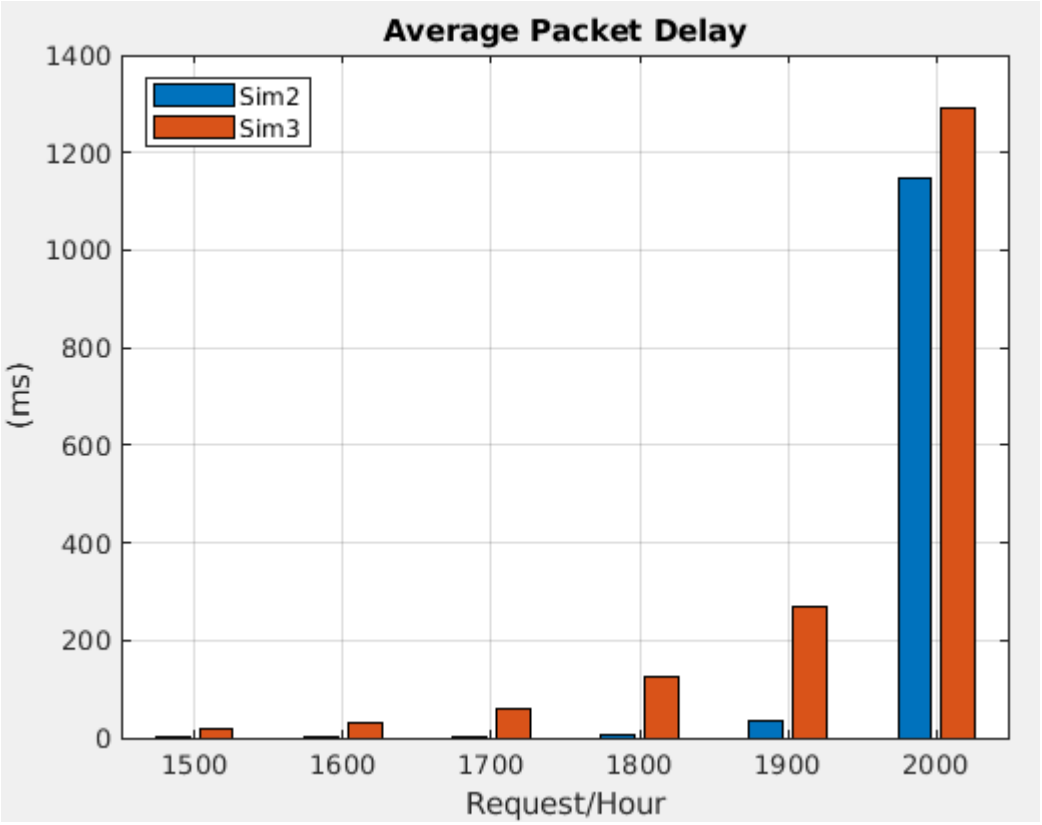
```
lambda = [1500, 1600, 1700, 1800, 1900, 2000];  % lambda - packet rate
(packets/sec)
C = 10;          % link bandwidth (Mbps)
P = 100000;      % number of packets (stopping criterium)
f = 1e7;
b = 0; % c)

mediaPL = zeros(1, size(lambda, 2));
mediaAPD = zeros(1, size(lambda, 2));
mediaMPD = zeros(1, size(lambda, 2));
mediaTT = zeros(1, size(lambda, 2));

mediaPL3 = zeros(1, size(lambda, 2));
mediaAPD3 = zeros(1, size(lambda, 2));
mediaMPD3 = zeros(1, size(lambda, 2));
mediaTT3 = zeros(1, size(lambda, 2));

for i=1:size(lambda, 2)
    for it=1:N
       [PL(it), APD(it), MPD(it), TT(it)] = simulator2(lambda(i), C, f,
P, b);
```

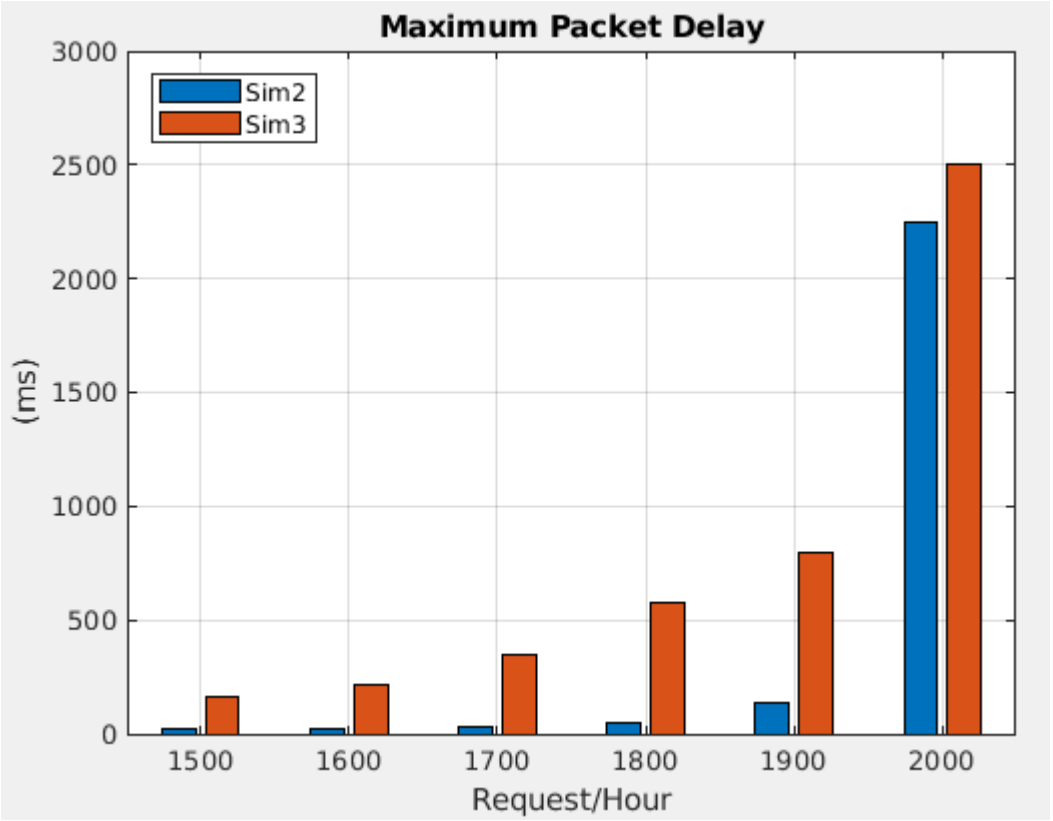
```
    end

    mediaPL(i) = mean(PL);
    mediaAPD(i) = mean(APD);
    mediaMPD(i) = mean(MPD);
    mediaTT(i) = mean(TT);

    for it=1:N
       [PL(it), APD(it), MPD(it), TT(it)] = simulator3(lambda(i), C, f,
P, b);
    end

    mediaPL3(i) = mean(PL);
    mediaAPD3(i) = mean(APD);
    mediaMPD3(i) = mean(MPD);
    mediaTT3(i) = mean(TT);
end

% plots
figure(1)
bar(lambda, [mediaPL; mediaPL3])
title('Packet Loss')
xlabel('Request/Second')
ylabel('(%%)')
legend('Sim2', 'Sim3', 'Location', 'northwest')
grid on
figure(2)
bar(lambda, [mediaAPD; mediaAPD3])
title('Average Packet Delay')
xlabel('Request/Second')
ylabel('(ms)')
legend('Sim2', 'Sim3', 'Location', 'northwest')
grid on

figure(3)
bar(lambda, [mediaMPD; mediaMPD3])
title('Maximum Packet Delay')
xlabel('Request/Second')
ylabel('(ms)')
legend('Sim2', 'Sim3', 'Location', 'northwest')
grid on

figure(4)
bar(lambda, [mediaTT; mediaTT3])
title('Throughput')
xlabel('Request/Second')
```

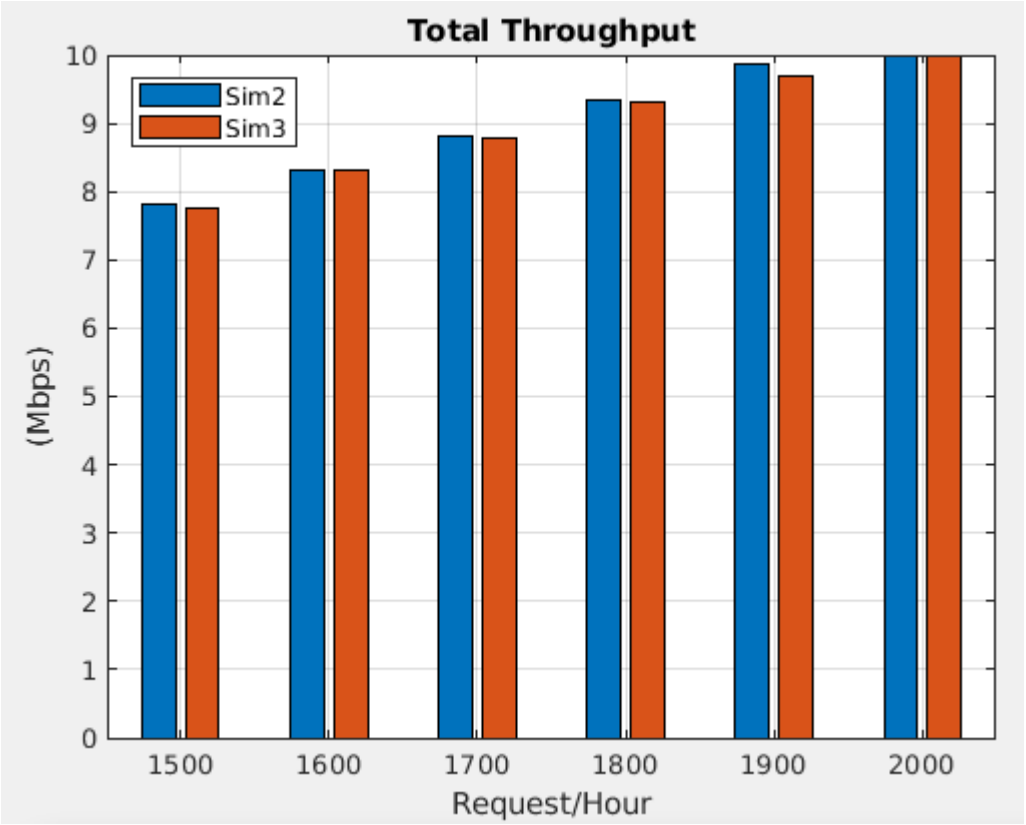
```
ylabel('(Mbps)')
legend('Sim2', 'Sim3', 'Location', 'northwest')
grid on
```

Resultados




Average Packet Delay
Sim2
Sim3
1400
1200
1000
800
600
400
200
0
(ms)
1500
1600
1700
1800
1900
2000
Request/Hour


Maximum Packet Delay
Sim2
Sim3
3000
2500
2000
1500
1000
500
0
(ms)
1500
1600
1700
1800
1900
2000
Request/Hour



Conclusões

De um modo geral pode concluir-se que o *Simulador3* tem um tempo de atraso, tanto médio como máximo, superior aos tempos de atraso do *Simulador2*.

Uma vez que o *Simulador3* é modelado através de uma cadeia de Markov em que dependendo do estado o ritmo a que se recebe pedidos (lambda) varia, leva a um comportamento diferente do *Simulador2*.

Uma justificação possível para o observado é que quando o sistema se apresenta no estado em que o *rate* é 1.5 lambda, este vai receber pedidos a um ritmo mais elevado o que vai levar a que a fila de espera encha e consequentemente aumenta o tempo de processamento a aumentar.

Pergunta 4.d

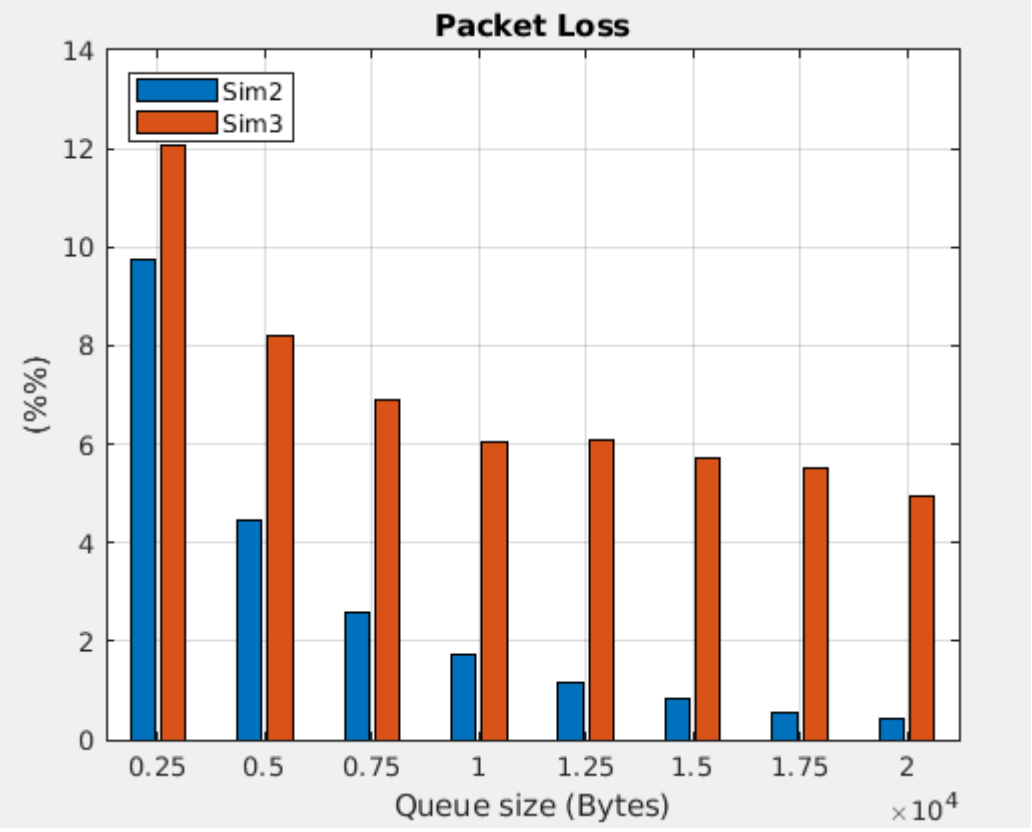
Código

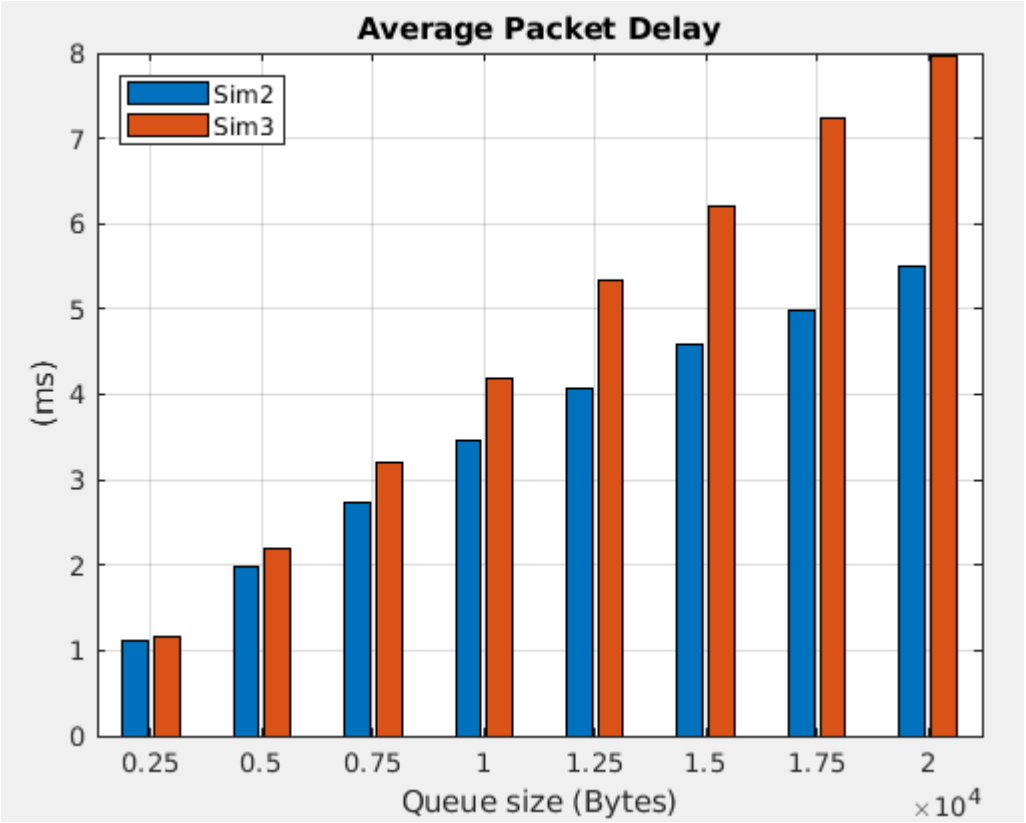
```
lambda = 1800;  % lambda - packet rate (packets/sec)
C = 10;         % link bandwidth (Mbps)
P = 100000;     % number of packets (stopping criterium)
f = [2500, 5000, 7500, 10000, 12500, 15000, 17500, 20000];
b = 0; % d)

mediaPL = zeros(1, size(f, 2));
mediaAPD = zeros(1, size(f, 2));
mediaMPD = zeros(1, size(f, 2));
mediaTT = zeros(1, size(f, 2));
```

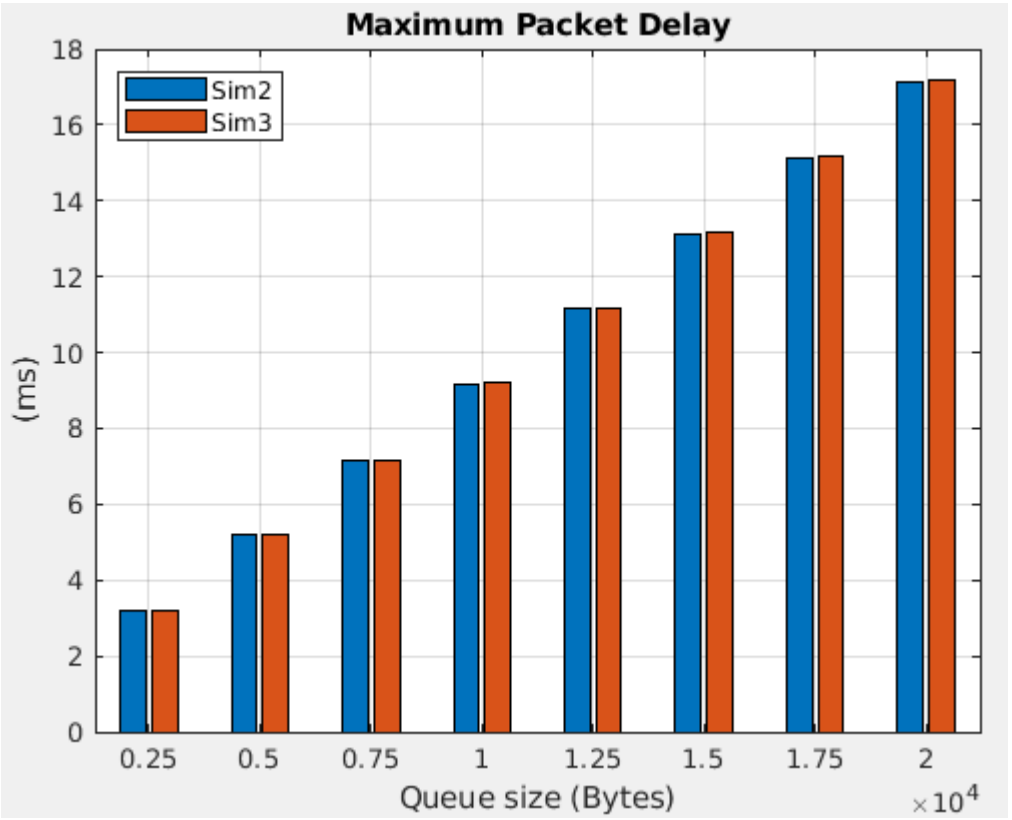
```
mediaPL3 = zeros(1, size(f, 2));
mediaAPD3 = zeros(1, size(f, 2));
mediaMPD3 = zeros(1, size(f, 2));
mediaTT3 = zeros(1, size(f, 2));

for i=1:size(f, 2)
    for it=1:N
        [PL(it), APD(it), MPD(it), TT(it)] = simulator2(lambda, C, f(i),
P, b);
    end

    mediaPL(i) = mean(PL);
    mediaAPD(i) = mean(APD);
    mediaMPD(i) = mean(MPD);
    mediaTT(i) = mean(TT);

    for it=1:N
        [PL(it), APD(it), MPD(it), TT(it)] = simulator3(lambda, C, f(i),
P, b);
    end

    mediaPL3(i) = mean(PL);
    mediaAPD3(i) = mean(APD);
    mediaMPD3(i) = mean(MPD);
    mediaTT3(i) = mean(TT);
end

% plots
figure(1)
bar(f, [mediaPL; mediaPL3])
title('Packet Loss')
xlabel('Queue size (Bytes)')
ylabel('(%%)')
legend('Sim2', 'Sim3', 'Location', 'northwest')
grid on

figure(2)
bar(f, [mediaAPD; mediaAPD3])
title('Average Packet Delay')
xlabel('Queue size (Bytes)')
ylabel('(ms)')
legend('Sim2', 'Sim3', 'Location', 'northwest')
grid on

figure(3)
```

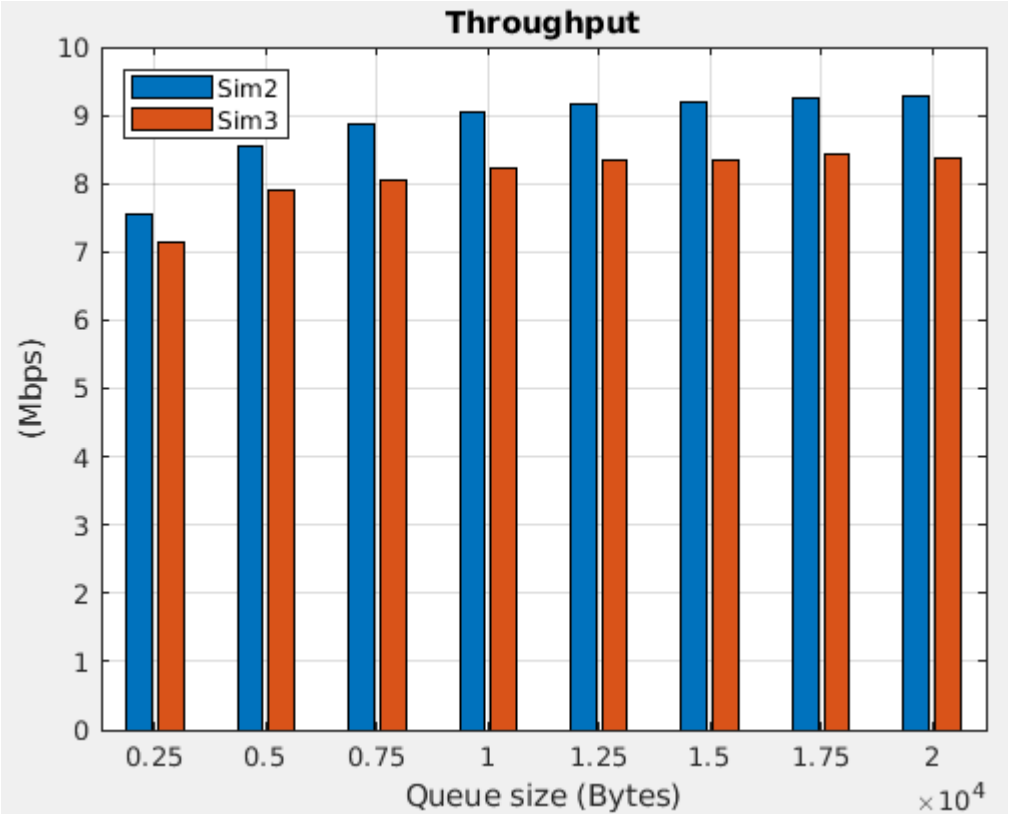
```
bar(f, [mediaMPD; mediaMPD3])
title('Maximum Packet Delay')
xlabel('Queue size (Bytes)')
ylabel('(ms)')
legend('Sim2', 'Sim3', 'Location', 'northwest')
grid on

figure(4)
bar(f, [mediaTT; mediaTT3])
title('Throughput')
xlabel('Queue size (Bytes)')
ylabel('(Mbps)')
legend('Sim2', 'Sim3', 'Location', 'northwest')
grid on
```

Resultados




Average Packet Delay
Sim2
Sim3
(ms)
0
1
2
3
4
5
6
7
8
0.25
0.5
0.75
1
1.25
1.5
1.75
2
Queue size (Bytes)
×10^4


Maximum Packet Delay
Sim2
Sim3
(ms)
0
2
4
6
8
10
12
14
16
18
0.25
0.5
0.75
1
1.25
1.5
1.75
2
Queue size (Bytes)
×10^4



Conclusões

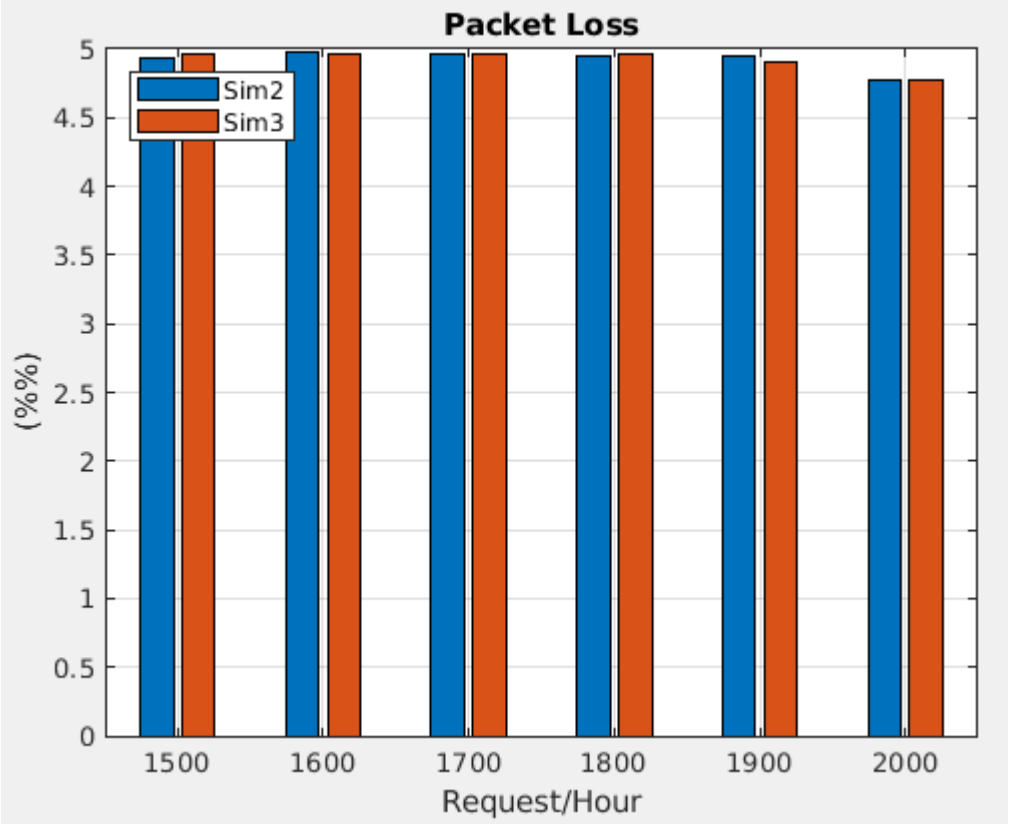
De um modo geral pode-se concluir que o *Simulador3* tem um desempenho pior que o *Simulador2*. O tempo máximo de atraso é igual nos dois simuladores uma vez que ambos atingem a capacidade máxima da fila de espera. Já no tempo médio de atraso é mais alto no *Simulador3* pois atinge a capacidade máxima da fila de espera mais cedo que o *Simulador2*. Uma outra consequência de atingir a capacidade máxima da fila de espera é a perda de pacotes, uma vez que atingido esse limite, os pacotes que chegarem depois serão perdidos. Assim, uma vez que o *Simulador3* atinge primeiro esse limite, perde mais pacotes que o *Simulador2*, como se pode confirmar no gráfico. Consequentemente, o *Simulador3*, ao perder mais pacotes, vai ter um throughput mais baixo.

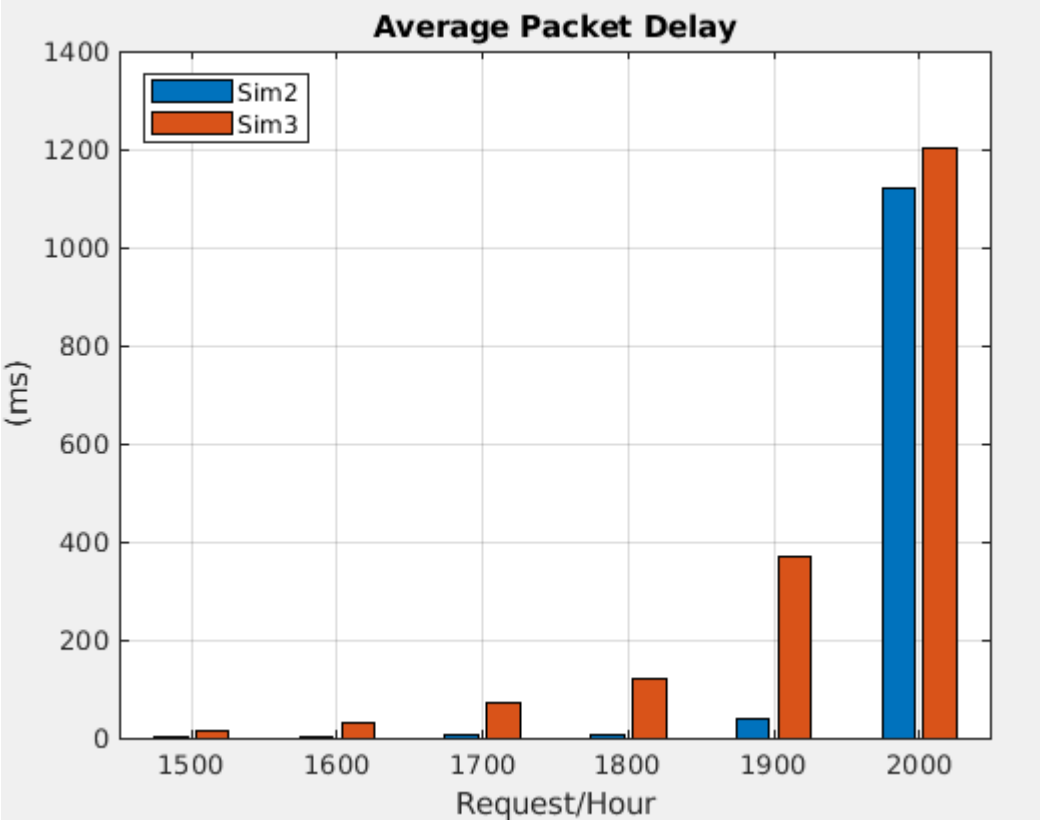
Pergunta 4.e

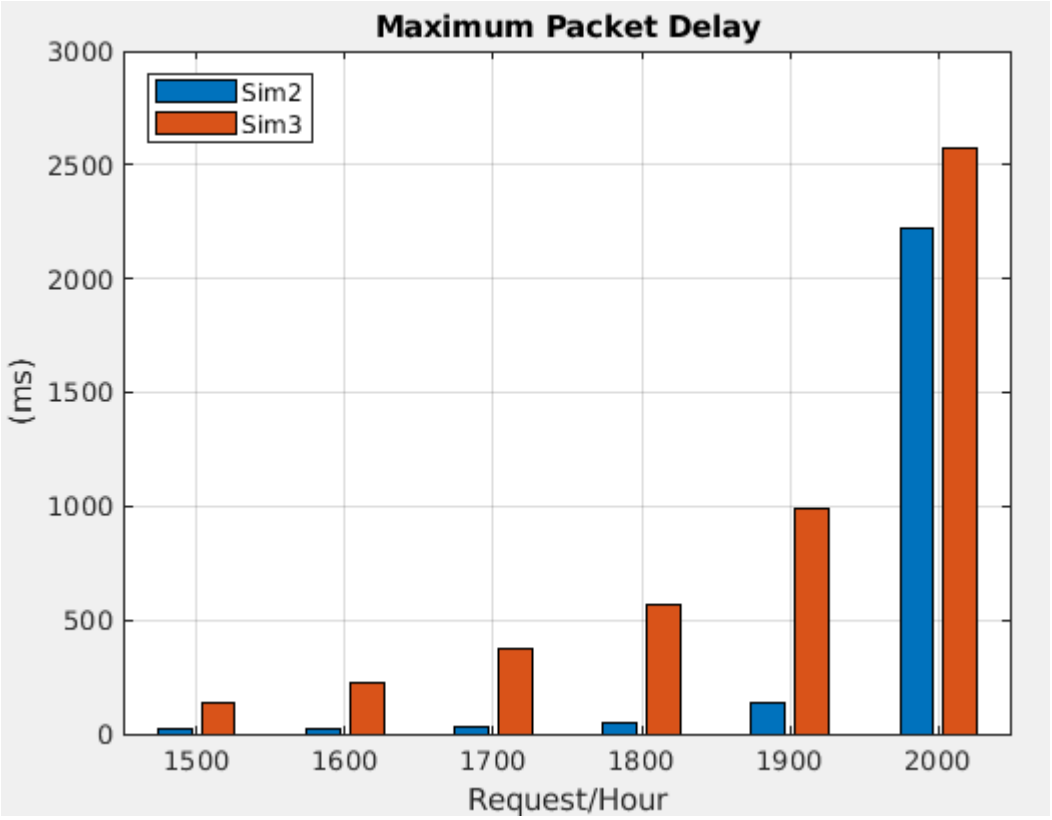
Código

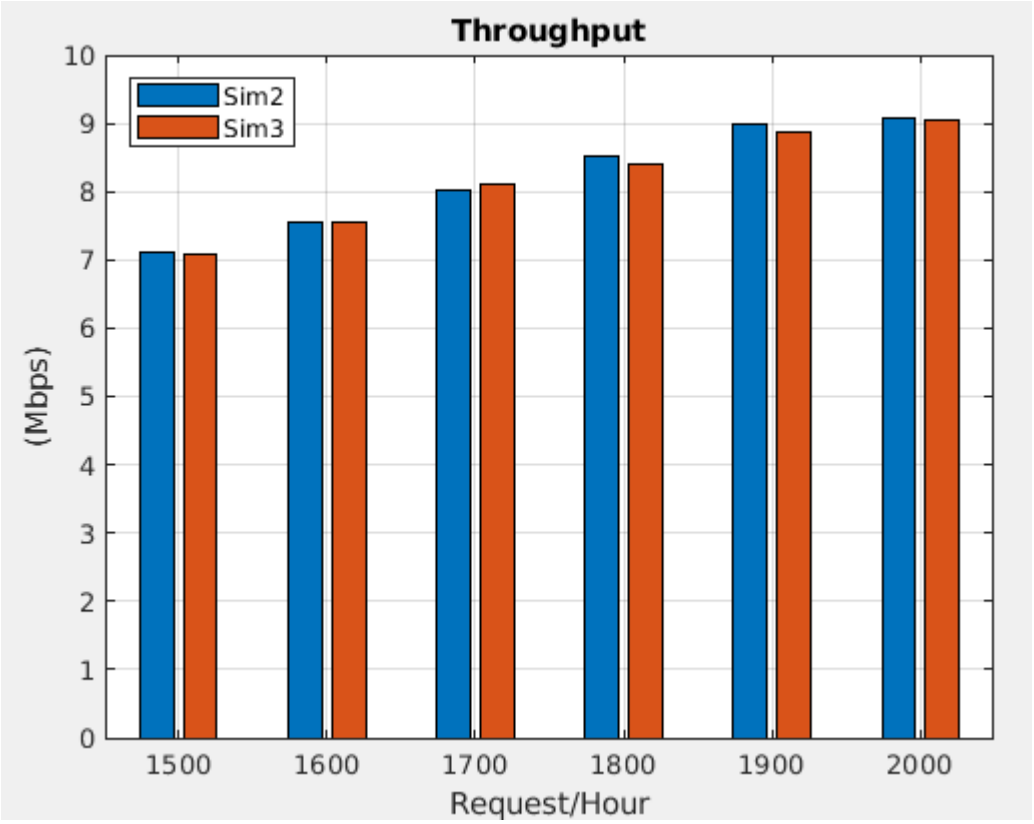
```
b = 1e-5; % e)
```

Resultados


Packet Loss
Sim2
Sim3
(%%)
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
1500
1600
1700
1800
1900
2000
Request/Hour


Average Packet Delay
Sim2
Sim3
(ms)
0
200
400
600
800
1000
1200
1400
1500
1600
1700
1800
1900
2000
Request/Hour





Conclusões

Como seria de esperar, em comparação com a pergunta 3.c, ao aumentar o Bit Error Rate (BER) é a existência de perda de pacotes (na 3.c não havia perda de pacotes) e

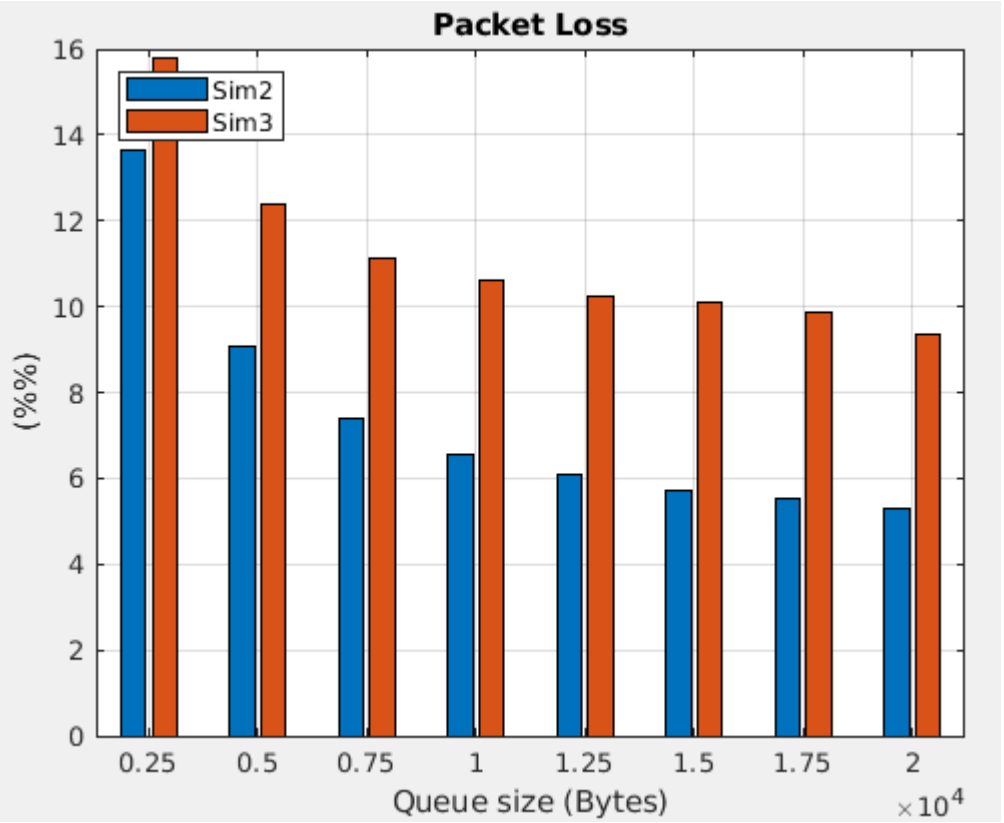
consequentemente uma diminuição do throughput. Quanto ao atraso médio e máximo de pacotes não houve mudanças significativas.

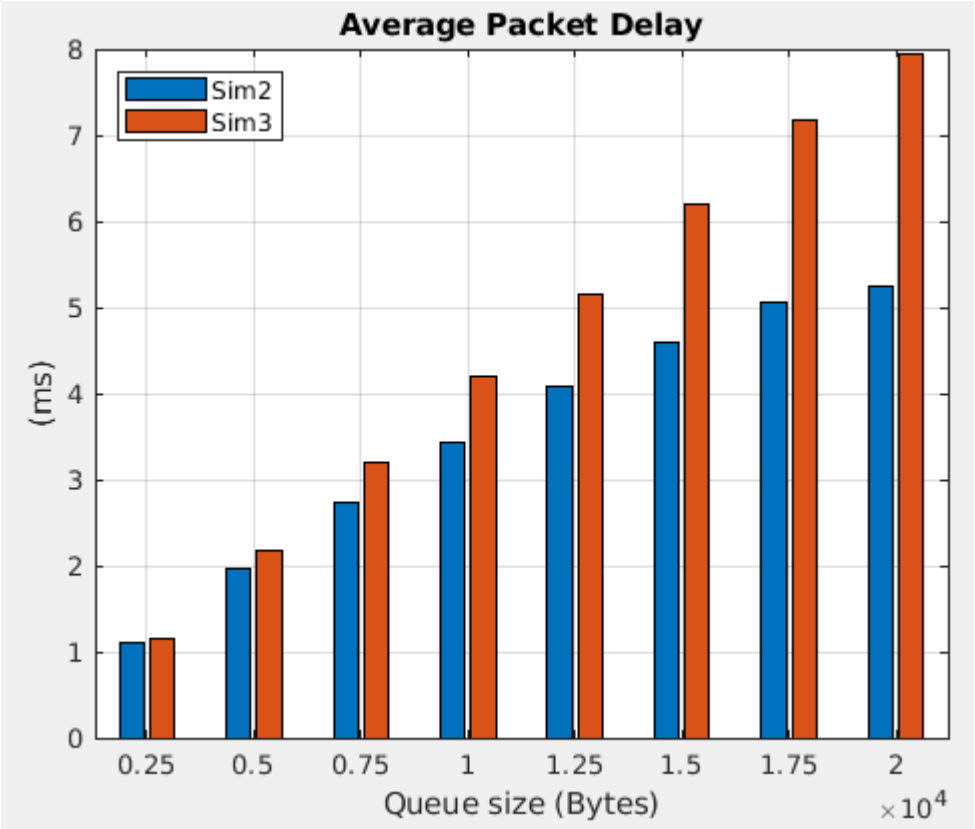
Pergunta 4.f

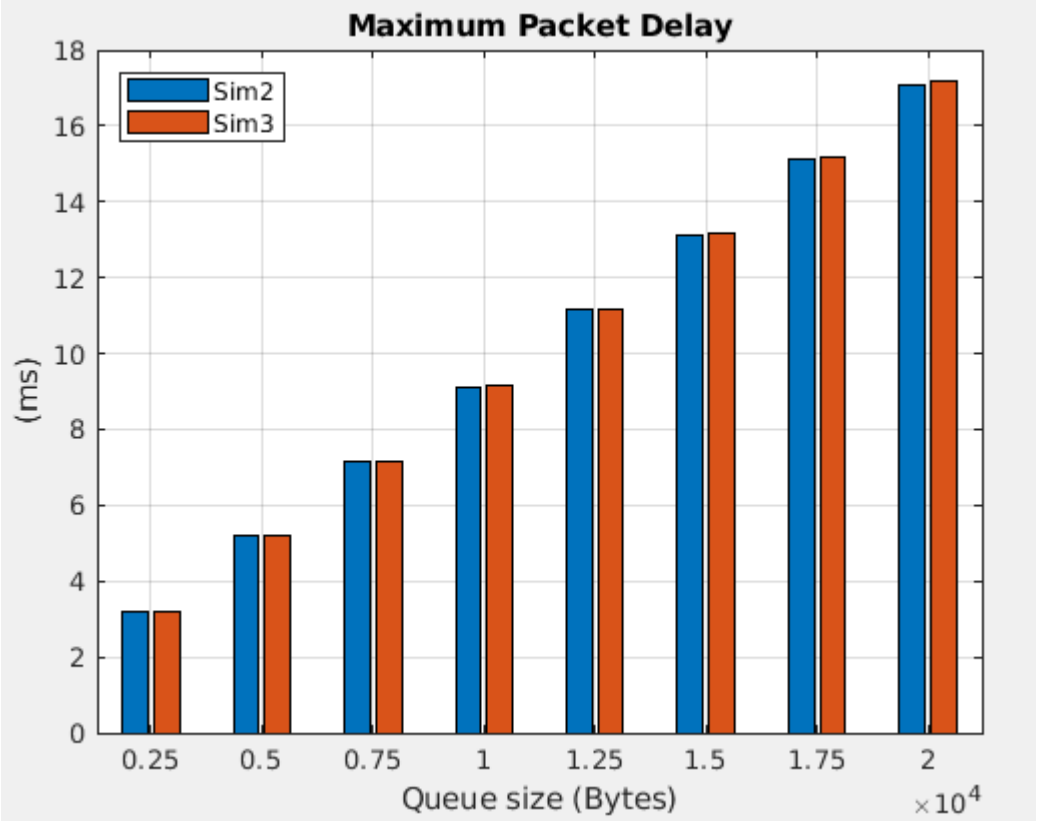
Código

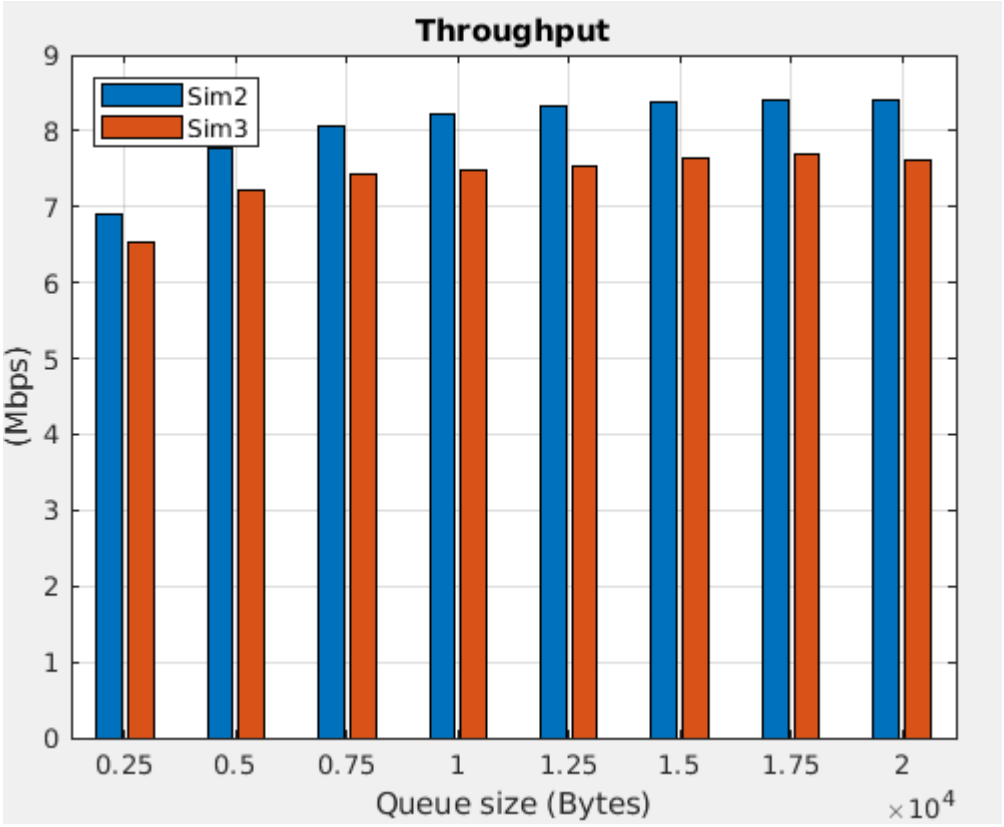
```
b = 1e-5; % f)
```

Resultados




Average Packet Delay
Sim2
Sim3
(ms)
0
1
2
3
4
5
6
7
8
0.25
0.5
0.75
1
1.25
1.5
1.75
2
Queue size (Bytes)
×10^4


Maximum Packet Delay
Sim2
Sim3
(ms)
0
2
4
6
8
10
12
14
16
18
0.25
0.5
0.75
1
1.25
1.5
1.75
2
Queue size (Bytes)
×10^4



Conclusões

Em semelhança com as conclusões retiradas da pergunta anterior, o aumento do BER teve um impacto no aumento no número de pacotes perdidos e consequentemente na diminuição do throughput.