Tarefa #3

Desempenho de uma Ligação de Comutação de Pacotes

Martim Neves, 88904
Gabriel Saudade, 89304

27 de Junho 2021

Universidade de Aveiro
UC: Desempenho e Dimensionamento de Redes
Professor: Amaro Fernandes de Sousa

Task 3

Task 3.a - correr o Simulador 2 10 vezes com os parâmetros de entrada pedidos e calcular os valores estimados e os intervalos de confiança

Código Matlab

```
C=10;
f=10^7;
b=0;
P=10000;
N = 10; %number of simulations
lambda=[1500,1600,1700,1800,1900,2000];
PL=zeros(1,N);
APD=zeros(1,N);
MPD=zeros(1,N);
TT=zeros(1,N);

for i=1:length(lambda)
    for j= 1:N
        [PL(j),APD(j),MPD(j),TT(j)]= simulator2(lambda(i)
            ,C,f,P,b);
    end
    alfa= 0.1; %90% confidence interval%
    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(lambda,mediaapd)
hold on
erro=errorbar(lambda,mediaapd,termapd,termapd);
erro.Color=[0 0 0];
erro.LineStyle = 'none';
hold off
grid on
xlabel('lambda values')
ylabel('miliseconds')
```

```
37 title('Average Packet Delay (milliseconds)')
38
39 figure(2)
40 bar(lambda,mediampd)
41 hold on
42 errb=errorbar(lambda,mediampd,termmpd,termmpd);
43 errb.Color=[0 0 0];
44 errb.LineStyle = 'none';
45 hold off
46 grid on
47 xlabel('lambda values')
48 ylabel('miliseconds')
49 title('Maximum Packet Delay (milliseconds)')
50
51 figure(3)
52 bar(lambda,mediatt)
53 hold on
54 erro=errorbar(lambda,mediatt,termtt,termtt);
55 erro.Color=[0 0 0];
56 erro.LineStyle = 'none';
57 hold off
58 grid on
59 xlabel('lambda values')
60 ylabel('Mbps')
61 ylim([0 11])
62 title('Transmited Throughput (Mbps)')
```

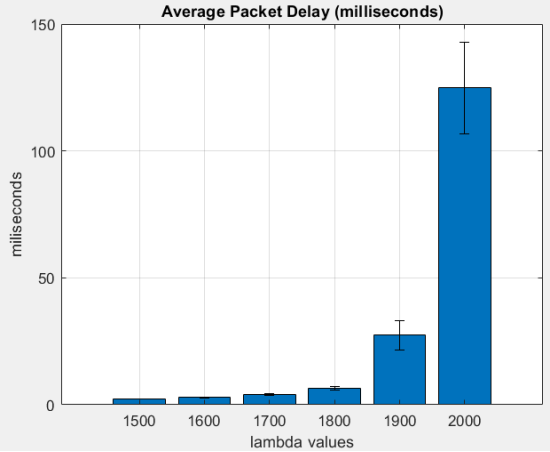
Análise do Código

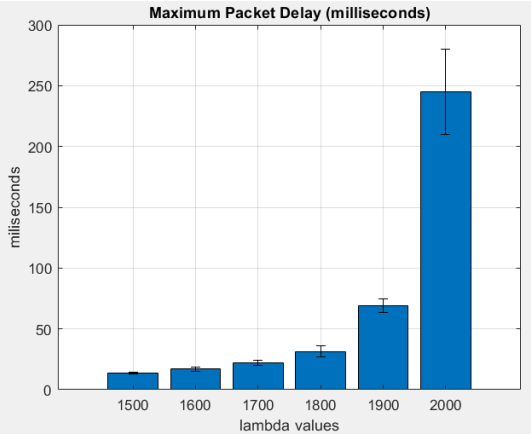
Inicialmente são definidos os valores para os parâmetros de entrada do simulador, valores estes dados no enunciado. Em seguida são criadas matrizes de zeros com dimensão igual ao número de corridas, de modo a guardar os valores provenientes do simulador. Este processo é repetido em todas as alíneas, pelo que apenas será explicado nesta alínea.

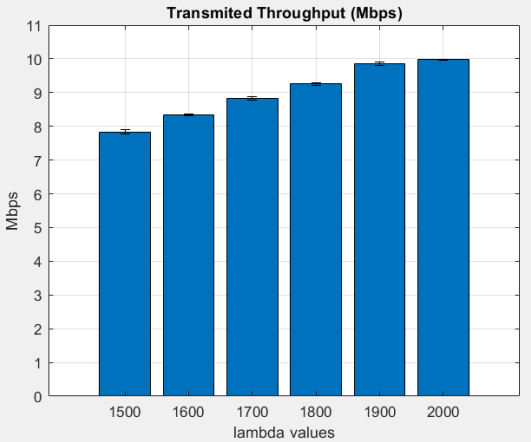
Em seguida são feitos dois ciclos for, um para percorrer os vários valores de lambda e outro para fazer as 10 corridas. Dentro desses ciclos é invocado o simulador 2 e os valores de retorno são armazenados nas matrizes anteriormente criadas. Depois disso são calculadas as métricas de desempenho e o processo é repetido para todos os valores de lambda.

Por fim são feitos gráficos de barras, com as respetivas barras de erro, para mostrar o desempenho para cada valor de lambda.

Resultados







Análises e Justificações

Como é possível observar pela figura, com o aumento da transmissão do número de pacotes por segundo, o average e o maximum packet delay aumentam substancialmente, pois o número de pacotes à espera na fila é gradualmente maior. Relativamente à taxa de transferência, dado que o tamanho de cada pacote varia entre 64 e 1518 bytes, à medida que o número de pacotes a serem transmitidos por segundo aumenta, a capacidade de transmissão chega perto do seu máximo.

Task 3.b - repetir a alínea anterior, mas desta vez para 40 corridas

Código Matlab

```
N=40;
PL=zeros(1,N);
APD=zeros(1,N);
MPD=zeros(1,N);
TT=zeros(1,N);

for i=1:length(lambda)
    for j= 1:N
        [PL(j),APD(j),MPD(j),TT(j)]= simulator2(lambda(i)
            ,C,f,P,b);
    end
    alfa= 0.1; %90% confidence interval%
    mediaplb(i) = mean(PL);
    termplb(i) = norminv(1-alfa/2)*sqrt(var(PL)/N);
    mediaapdb(i) = mean(APD);
    termapdb(i) = norminv(1-alfa/2)*sqrt(var(APD)/N);
    mediampdb(i) = mean(MPD);
    termmpdb(i) = norminv(1-alfa/2)*sqrt(var(MPD)/N);
    mediattb(i) = mean(TT);
    termttb(i) = norminv(1-alfa/2)*sqrt(var(TT)/N);
end

figure(4)
subplot(1,2,1)
bar(lambda,mediaapd)
hold on
erro=errorbar(lambda,mediaapd,termapd,termapd);
erro.Color=[0 0 0];
erro.LineStyle = 'none';
hold off
grid on
ylim([0 130])
xlabel('lambda values')
ylabel('miliseconds')
title('10 corridas')
subplot(1,2,2)
bar(lambda,mediaapdb)
hold on
erro=errorbar(lambda,mediaapdb,termapdb,termapdb);
erro.Color=[0 0 0];
```

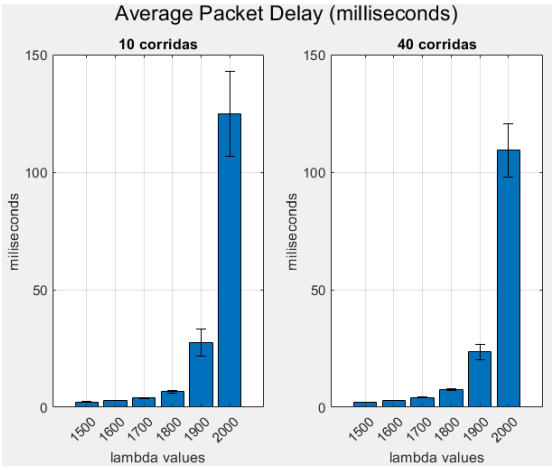
```
erro.LineStyle = 'none';
hold off
grid on
ylim([0 130])
xlabel('lambda values')
ylabel('miliseconds')
title('40 corridas')
sgtitle('Average Packet Delay (milliseconds)')

figure(5)
subplot(1,2,1)
bar(lambda,mediampd)
hold on
erro=errorbar(lambda,mediampd,termmpd,termmpd);
erro.Color=[0 0 0];
erro.LineStyle = 'none';
hold off
grid on
ylim([0 275])
xlabel('lambda values')
ylabel('milliseconds')
title('10 corridas')
subplot(1,2,2)
bar(lambda,mediampdb)
hold on
erro=errorbar(lambda,mediampdb,termmpdb,termmpdb);
erro.Color=[0 0 0];
erro.LineStyle = 'none';
hold off
grid on
ylim([0 275])
xlabel('lambda values')
ylabel('milliseconds')
title('40 corridas')
sgtitle('Maximum Packet Delay (milliseconds)')

figure(6)
subplot(1,2,1)
bar(lambda,mediatt)
hold on
erro=errorbar(lambda,mediatt,termtt,termtt);
erro.Color=[0 0 0];
erro.LineStyle = 'none';
hold off
grid on
xlabel('lambda values')
```

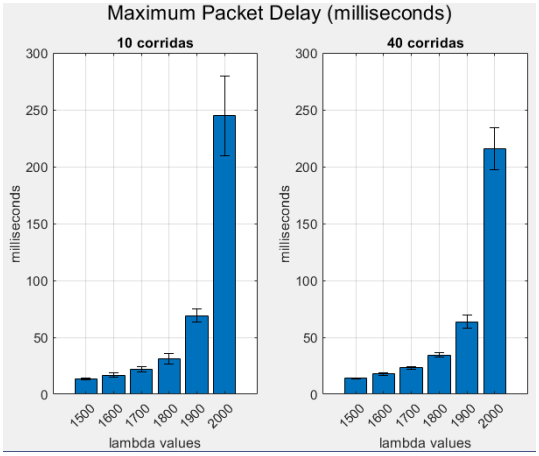
```
86  ylabel('Mbps')
87  ylim([0 11])
88  title('10 corridas')
89  subplot(1,2,2)
90  bar(lambda,mediattb)
91  hold on
92  erro=errorbar(lambda,mediattb,termttb,termttb);
93  erro.Color=[0 0 0];
94  erro.LineStyle = 'none';
95  hold off
96  grid on
97  ylim([0 11])
98  xlabel('lambda values')
99  ylabel('Mbps')
100 title('40 corridas')
101 sgtitle('Transmited Throughput (Mbps)')
```

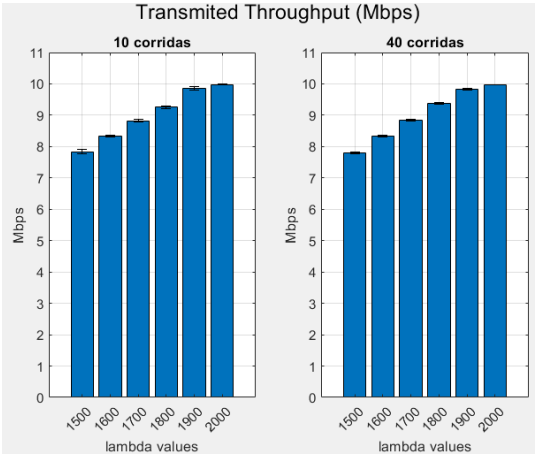
Análise do Código

Usando como parâmetros de entrada os valores definidos na alínea anterior, o número de corridas é mudado de 10 para 40 e são repetidos os ciclos for para percorrer os vários valores de lambda e para o número pedido de corridas, são calculadas as métricas de performance e são feitos os gráficos de barras com as respetivas barras de erro, sendo os gráficos de barras mostrados lado a lado com os resultados da alínea anterior, para se poderem perceber as diferenças entre fazer 10 e 40 corridas.

Resultados







Análises e Justificações

Através da observação dos gráficos, é possível concluir que quando o número de simulações é maior, a precisão dos resultados torna-se também maior, como podemos ver pela diminuição do tamanho das barras de erro.

Task 3.c - calcular o atraso médio dos pacotes e throughput total da alínea 3.b assumindo que é modelado por um sistema M/M/1 e por um sistema M/G/1

Código Matlab

```
%M/M/1
aux1=[65:109  111:1517];
aux3=length(aux1);
aux4=(1-(0.16+0.25+0.2))/aux3;
B=(0.16*64+0.25*110+0.2*1518+aux4*sum(aux1))*8;
miu=C*1000000/B;
for j=1:length(lambda)
    packet_delay(j) = 1/(miu-lambda(j))*1000;
    throughput(j) = (lambda(j)*B)/1000000;
end

%M/G/1
aux1=[64:1518];
aux3=length(aux1)-3;
aux4=(1-(0.16+0.25+0.2))/aux3;
APD=zeros(1,length(lambda));
TT=zeros(1,length(lambda));
for i=1:length(aux1)
    Pi(i)=(1-b)^(8*aux1(i));
    if aux1(i)==64
        pn(i)=0.16;
    elseif aux1(i)==110
        pn(i)=0.25;
    elseif aux1(i)==1518
        pn(i)=0.2;
    else
        pn(i)=aux4;
    end
end
for j=1:length(lambda)
    ES(j)=0;
    ES2(j)=0;
    TT(j)=0;
    WQ(j)=0;
    Wi(j)=0;
    APD(j)=0;
    for i=1:length(aux1)
        ES(j)=ES(j)+pn(i)*(8*aux1(i)/(C*1e6));
        ES2(j)=ES2(j)+pn(i)*((8*aux1(i)/(C*1e6))^2);
```

```
40              TT(j)=TT(j)+(pn(i)*Pi(i)*lambda(j)*(8*aux1(i)));
41          end
42          WQ(j)=lambda(j)*ES2(j)/(2*(1-lambda(j)*ES(j)));
43          num(j)=0;
44          den(j)=0;
45          for i=1:length(aux1)
46              Wi(j)=WQ(j)+(8*aux1(i)/(C*1e6));
47              num(j)=num(j)+(pn(i)*Pi(i)*Wi(j));
48              den(j)=den(j)+(pn(i)*Pi(i));
49          end
50          APD(j)=num(j)/den(j);
51  end
52
53  figure(24)
54  bar(lambda,[mediaapdb;packet_delay;APD.*1000])
55  title('Average Packet Delay (milliseconds)')
56  legend('valor experimental','M/M/1','M/G/1')
57  xlabel('lambda values')
58  ylabel('milliseconds')
59
60  figure(25)
61  bar(lambda,[mediattb;throughput;TT/1000000])
62  title('Transmited Throughput (Mbps)')
63  legend('valor experimental','M/M/1','M/G/1')
64  xlabel('lambda values')
65  ylabel('Mbps')
```

Análise do Código

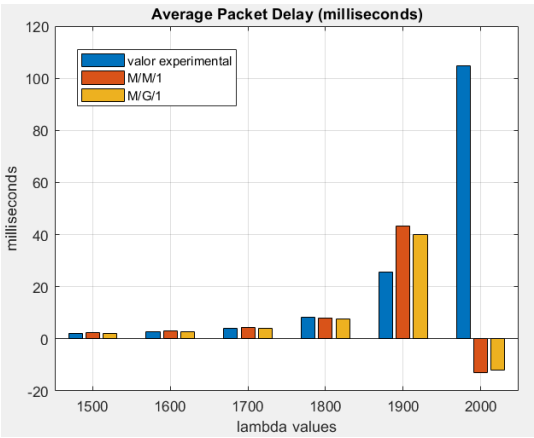
Para o modelo M/M/1, a matriz aux1 guarda todos os valores possíveis para o tamanho dos pacotes exceto os valores 64, 110 e 1518. A variável aux3 serve para armazenar o número total de pacotes para poder ser usada para calcular a probabilidade de o pacote ter um dos tamanhos presentes em aux1, visto que os valores mencionados anteriormente já têm uma probabilidade definida, e guardar essa probabilidade em aux4. De seguida é calculado o tamanho médio por pacote, armazenado na variável B, e a capacidade da ligação, miu. Por fim, são aplicadas as fórmulas para o cálculo da percentagem de perda de pacotes e do throughput total.

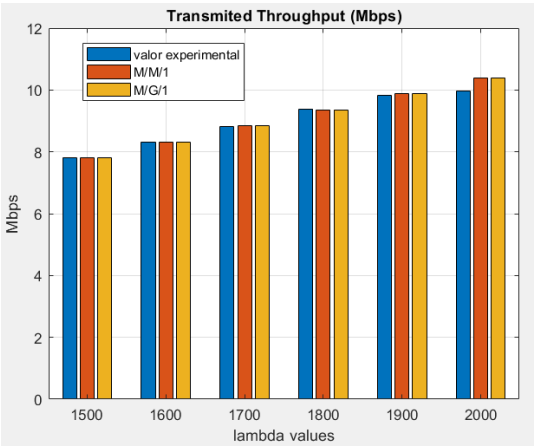
Para o modelo M/G/1, a matriz aux1 armazena todos os valores possíveis para o tamanho dos pacotes, a variável aux3, tal como para o modelo anterior, armazena o total de valores para o tamanho de pacotes que não têm uma probabilidade definida e a variável aux4 guarda a probabilidade de o pacote ter

um dos tamanho definidos em aux1, exceto os 3 mencionados anteriormente. Em seguida são inicializadas duas matrizes para guardar os valores finais. A seguir, dentro de um ciclo for, é calculada a probabilidade de cada pacote ser enviado sem erros e percorrida a matriz que guarda os possíveis tamanhos de cada pacote. Caso esse valor seja 64, 110 ou 1518, o valor da probabilidade para esse tamanho, 16, 25 e 20%, é guardado na matriz pn. Caso seja qualquer outro valor para o tamanho do pacote, o valor guardado na matriz pn é aux4. Em seguida é feito outro ciclo for, no qual são calculados os valores de média de transmissão de pacotes (E[S]) e o seu quadrado (E[S^2]), assim como o throughput total do sistema. A seguir, através da fórmula de Pollaczek - Khintchine, é calculado o atraso médio na fila de espera. Seguidamente é calculado o atraso médio de cada tamanho de pacote e é calculado o numerador e denominador, para finalmente serem usados no cálculo do atraso médio dos pacotes.

O passo final é fazer os gráficos de barras, sendo que para cada métrica, são apresentados os valores obtidos para cada modelo, assim como os valores obtidos na alínea anterior, para se poder comparar os valores obtidos por simulação com os valores teóricos para cada modelo.

Resultados





Análises e Justificações

Pela observação do gráfico, podemos concluir que os resultados teóricos dos sistemas M/M/1 e M/G/1 são praticamente iguais aos resultados obtidos por simulação tanto no *average packet delay* como no *trasmitted troughtput*, para um número de transmissão de pacotes por segundo inferior. No entanto, para lambdas superiores a discrepância torna-se maior e é possível observar principalmente no *average packet delay* quando o lambda é 2000 que os modelos não são validos, pois assumem uma fila infinita de pacotes de chegada e estes modelos são só validos quando a fila for maior que a taxa de transmissão de pacotes por segundo.

Task 3.d - correr o simulador 2 40 vezes com os valores pedidos no enunciado e calcular os valores estimados e os intervalos de confiança

Código Matlab

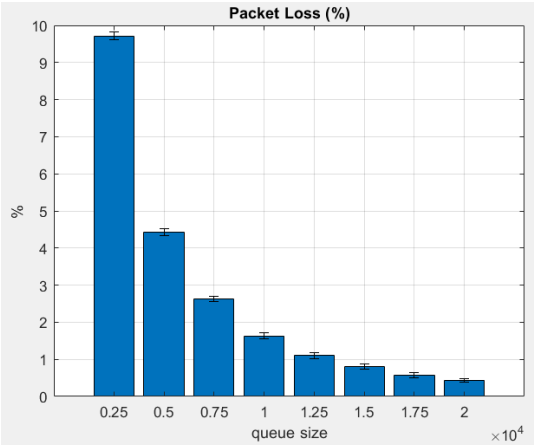
```
lambdad=1800;
fd=[2500,5000,7500,10000,12500,15000,17500,20000];
N=40;
PL=zeros(1,N);
APD=zeros(1,N);
MPD=zeros(1,N);
TT=zeros(1,N);

for i=1:length(fd)
    for j= 1:N
        [PL(j),APD(j),MPD(j),TT(j)]= simulator2(lambdad,C
            ,fd(i),P,b);
    end
    alfa= 0.1; %90% confidence interval%
    mediapld(i) = mean(PL);
    termpld(i) = norminv(1-alfa/2)*sqrt(var(PL)/N);
    mediaapdd(i) = mean(APD);
    termapdd(i) = norminv(1-alfa/2)*sqrt(var(APD)/N);
    mediampdd(i) = mean(MPD);
    termmpdd(i) = norminv(1-alfa/2)*sqrt(var(MPD)/N);
    mediattd(i) = mean(TT);
    termttd(i) = norminv(1-alfa/2)*sqrt(var(TT)/N);
end
figure(7)
bar(fd,mediapld)
```

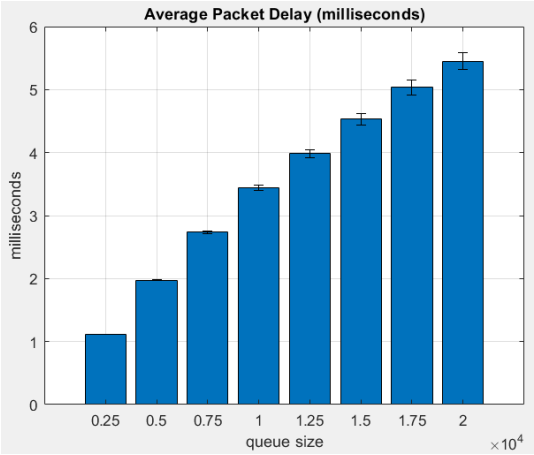
```
hold on
erro=errorbar(fd,mediapld,termpld,termpld);
erro.Color=[0 0 0];
erro.LineStyle = 'none';
hold off
grid on
xlabel('queue size')
ylabel('%')
title('Packet Loss (%)')

figure(8)
bar(fd,mediaapdd)
hold on
erro=errorbar(fd,mediaapdd,termapdd,termapdd);
erro.Color=[0 0 0];
erro.LineStyle = 'none';
hold off
grid on
xlabel('queue size')
ylabel('milliseconds')
title('Average Packet Delay (milliseconds)')

figure(9)
bar(fd,mediampdd)
hold on
errb=errorbar(fd,mediampdd,termmpdd,termmpdd);
errb.Color=[0 0 0];
errb.LineStyle = 'none';
hold off
grid on
xlabel('queue size')
ylabel('milliseconds')
title('Maximum Packet Delay (milliseconds)')

figure(10)
bar(fd,mediattd)
hold on
erro=errorbar(fd,mediattd,termttd,termttd);
erro.Color=[0 0 0];
erro.LineStyle = 'none';
hold off
grid on
xlabel('queue size')
ylabel('Mbps')
title('Transmited Throughput (Mbps)')
```

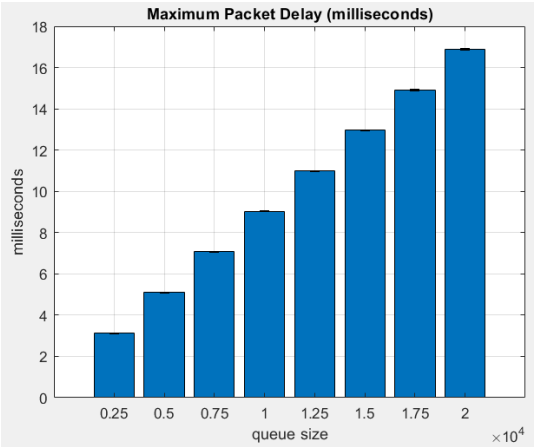
Análise do Código

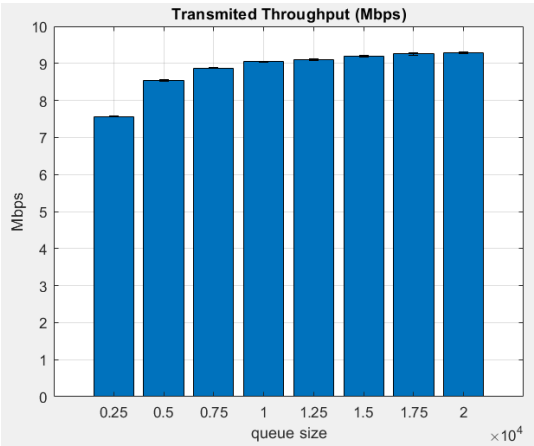
Tal como nas alíneas anteriores, são declarados os parâmetros de entrada cujos valores foram alterados, são feitos dois ciclos for, um para percorrer os valores de f (tamanho da fila de espera) e outro para fazer as corridas, é invocado o simulador 2 e os seus valores de retorno são guardados nas matrizes mencionadas anteriormente. De seguida são calculadas as métricas de desempenho e, por fim, são feitos gráficos de barras para cada métrica, juntamente com as respetivas barras de erro.

Resultados









Análises e Justificações

Relativamente ao *packet loss* com o aumento da fila de espera, naturalmente o número de pacotes perdidos na transmissão reduz e o *average e maximum packet delay* aumenta, pois, o número de pacotes na fila de espera aumenta. A taxa de transmissão aumenta relativamente pouco chegando perto da capacidade máxima de transmissão com o aumento da capacidade da fila de espera.

Task 3.e - calcular a percentagem de perda de pacotes, o atraso médio dos pacotes e throughput total da alínea 3.d assumindo que é modelado por um sistema M/M/1/m

Código Matlab

```
aux1=[65:109  111:1517];
aux3=length(aux1);
aux4=(1-(0.16+0.25+0.2))/aux3;
B=(0.16*64+0.25*110+0.2*1518+aux4*sum(aux1))*8;
miu=C*1000000/B;
for i=1:length(fd)
    m(i)=round(fd(i)*8/B)+1;
    num(i)=0;
    den(i)=0;
    for j=0:m(i)
        num(i)=num(i)+(j*(lambdad/miu)^j);
        den(i)=den(i)+(lambdad/miu)^j;
    end
    packet_loss2(i)=((lambdad/miu)^m(i))/den(i);
    L(i)=num(i)/den(i);
    packet_delay2(i)=L(i)/(lambdad*(1-packet_loss2(i)));
    throughput2(i) = ((lambdad*B)/1000000)*(1-
        packet_loss2(i));
end

figure(19)
bar(fd,[mediapld;packet_loss2*100])
grid on
title('Packet Loss (%)')
legend('valor experimental','M/M/1/m')
xlabel('queue size')
ylabel('%')

figure(20)
bar(fd,[mediaapdd;packet_delay2*1000])
grid on
title('Average Packet Delay (milliseconds)')
```

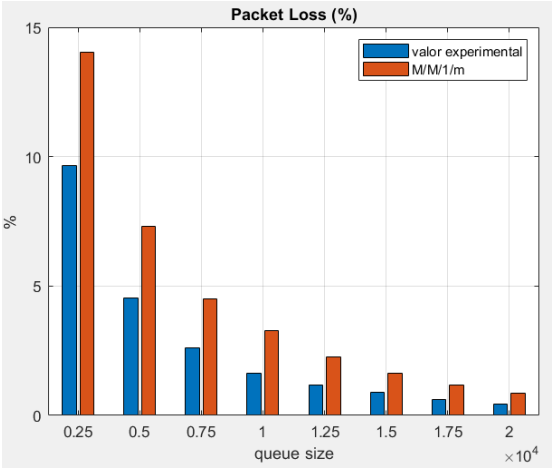
```
32 legend('valor experimental','M/M/1/m')
33 xlabel('queue size')
34 ylabel('milliseconds')
35
36 figure(21)
37 bar(fd,[mediattd;throughput2])
38 grid on
39 title('Transmited Throughput (Mbps)')
40 legend('valor experimental','M/M/1/m')
41 xlabel('queue size')
42 ylabel('Mbps')
```

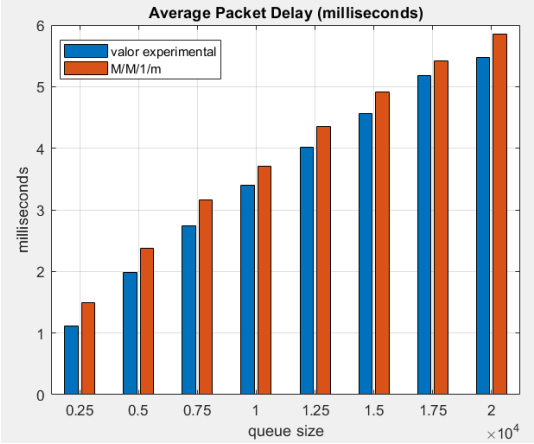
Análise do Código

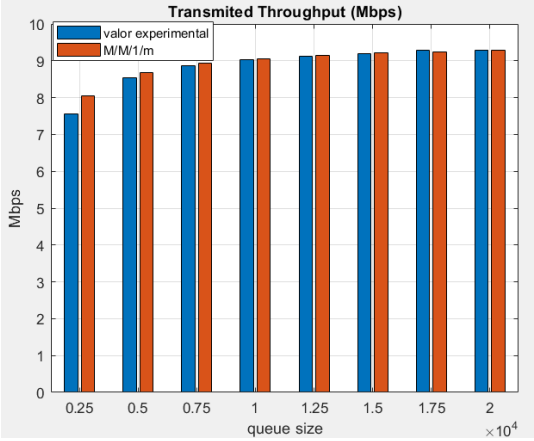
A matriz aux1 guarda todos os valores possíveis para o tamanho dos pacotes exceto os valores 64, 110 e 1518. A variável aux3 serve para armazenar o número total de pacotes para poder ser usado para calcular a probabilidade de o pacote ter um dos tamanhos presentes em aux1, visto que os valores mencionados anteriormente já têm uma probabilidade definida, e guardar essa probabilidade em aux4. De seguida é calculado o tamanho médio por pacote, armazenado na variável B, e a capacidade da ligação em pacotes por segundo, miu. De seguida é feito um ciclo for para calcular o número de clientes para os quais a fila de espera tem capacidade, m, e é feito outro ciclo for para calcular o numerador e denominador usados no cálculo da percentagem de perda de pacotes, atraso médio dos pacotes e throughput total.

Por fim são feitos os gráficos de barras, sendo que cada gráfico de barras corresponde a uma métrica e são apresentados em cada gráfico os valores teóricos e os valores obtidos por simulação, para poderem ser comparados.

Resultados







Análises e Justificações

Observando o *packet loss* e o *average packet delay*, podemos concluir que existe uma discrepância no resultado teórico do sistema M/M/1/m e do resultado da simulação, pois o sistema M/M/1/m acomoda um número finito de clientes, e consequentemente, o *packet loss* e o *average packet delay* é relativamente maior. No entanto, os valores obtidos da taxa de transmissão permanecem similares, chegando perto da capacidade total de transmissão.

Task 3.f - repetir a alínea 3.d, mas desta vez com bit error rate 10^{-5}

Código Matlab

```
bf=10^-5;
N=40;
PL=zeros(1,N);
APD=zeros(1,N);
MPD=zeros(1,N);
TT=zeros(1,N);

for i=1:length(fd)
    for j= 1:N
        [PL(j),APD(j),MPD(j),TT(j)]= simulator2(lambdad,C
            ,fd(i),P,bf);
    end
    alfa= 0.1; %90% confidence interval%
    mediaplf(i) = mean(PL);
    termplf(i) = norminv(1-alfa/2)*sqrt(var(PL)/N);
    mediaapdf(i) = mean(APD);
    termapdf(i) = norminv(1-alfa/2)*sqrt(var(APD)/N);
    mediampdf(i) = mean(MPD);
    termmpdf(i) = norminv(1-alfa/2)*sqrt(var(MPD)/N);
    mediattf(i) = mean(TT);
    termttf(i) = norminv(1-alfa/2)*sqrt(var(TT)/N);
end

figure(11)
subplot(1,2,1)
bar(fd,mediapld)
hold on
erro=errorbar(fd,mediapld,termpld,termpld);
erro.Color=[0 0 0];
erro.LineStyle = 'none';
hold off
grid on
ylim([0 40])
title('b=0')
xlabel('queue size')
ylabel('%')
subplot(1,2,2)
bar(fd,mediaplf)
hold on
erro=errorbar(fd,mediaplf,termplf,termplf);
```

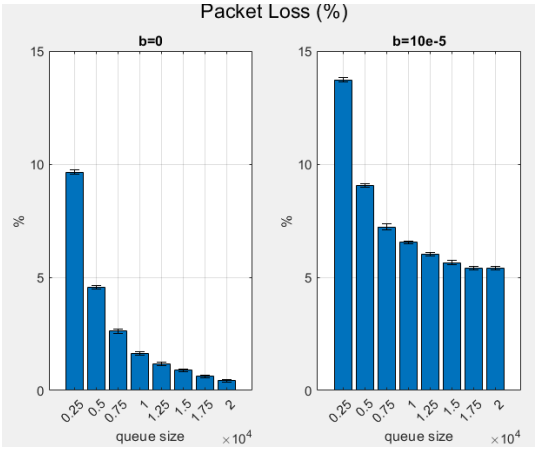
```
erro.Color=[0 0 0];
erro.LineStyle = 'none';
hold off
grid on
ylim([0 40])
title('b=10e-5')
xlabel('queue size')
ylabel('%')
sgtitle('Packet Loss (%)')

figure(12)
subplot(1,2,1)
bar(fd,mediaapdd)
hold on
erro=errorbar(fd,mediaapdd,termapdd,termapdd);
erro.Color=[0 0 0];
erro.LineStyle = 'none';
hold off
grid on
ylim([0 10])
title('b=0')
xlabel('queue size')
ylabel('milliseconds')
subplot(1,2,2)
bar(fd,mediaapdf)
hold on
erro=errorbar(fd,mediaapdf,termapdf,termapdf);
erro.Color=[0 0 0];
erro.LineStyle = 'none';
hold off
grid on
ylim([0 10])
title('b=10e-5')
xlabel('queue size')
ylabel('milliseconds')
sgtitle('Average Packet Delay (milliseconds)')

figure(13)
subplot(1,2,1)
bar(fd,mediampdd)
hold on
erro=errorbar(fd,mediampdd,termmpdd,termmpdd);
erro.Color=[0 0 0];
erro.LineStyle = 'none';
hold off
grid on
```

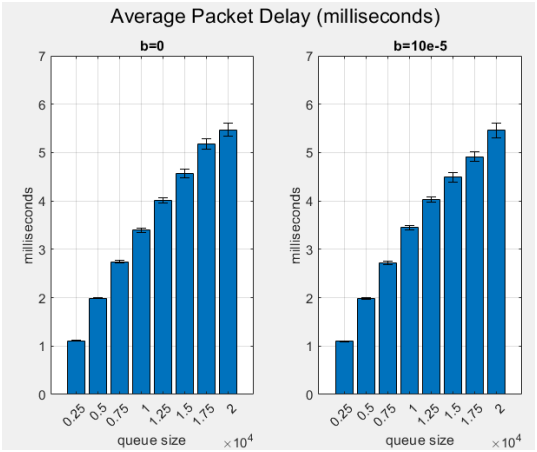
```
86  ylim([0 25])
87  title('b=0')
88  xlabel('queue size')
89  ylabel('milliseconds')
90  subplot(1,2,2)
91  bar(fd,mediampdf)
92  hold on
93  erro=errorbar(fd,mediampdf,termmpdf,termmpdf);
94  erro.Color=[0 0 0];
95  erro.LineStyle = 'none';
96  hold off
97  grid on
98  ylim([0 25])
99  title('b=10e-5')
100 xlabel('queue size')
101 ylabel('milliseconds')
102 sgtitle('Maximum Packet Delay (milliseconds)')
103
104 figure(14)
105 subplot(1,2,1)
106 bar(fd,mediattd)
107 hold on
108 erro=errorbar(fd,mediattd,termttd,termttd);
109 erro.Color=[0 0 0];
110 erro.LineStyle = 'none';
111 hold off
112 grid on
113 title('b=0')
114 xlabel('queue size')
115 ylabel('Mbps')
116 ylim([0 10])
117 subplot(1,2,2)
118 bar(fd,mediattf)
119 hold on
120 erro=errorbar(fd,mediattf,termttf,termttf);
121 erro.Color=[0 0 0];
122 erro.LineStyle = 'none';
123 hold off
124 grid on
125 ylim([0 10])
126 title('b=10e-5')
127 xlabel('queue size')
128 ylabel('Mbps')
129 sgtitle('Transmited Throughput (Mbps)')
```

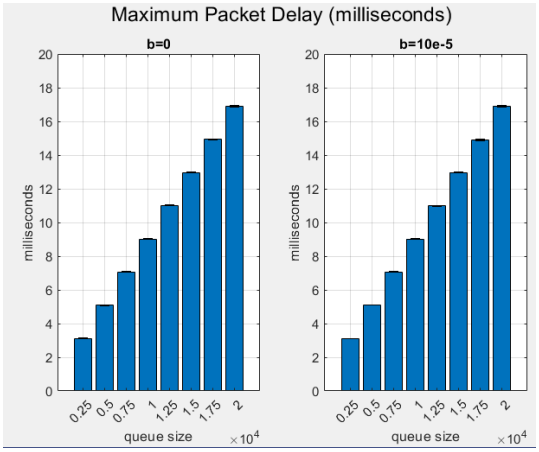
Análise do Código

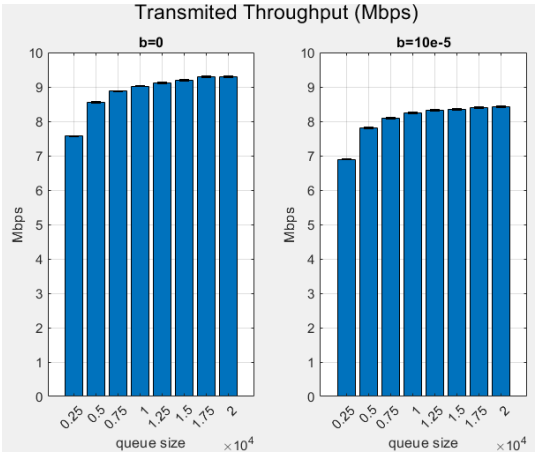
Primeiramente é dado valor aos parâmetros de entrada necessários, como mencionado anteriormente são usados dois ciclos for, é invocado o simulador 2 e são calculadas as métricas de desempenho. No fim são apresentados gráficos de barras para cada métrica, juntamente com o gráfico produzido na alínea 3.d correspondente a cada métrica, para podermos tirar conclusões acerca do impacto do bit error rate.

Resultados









Análises e Justificações

Introduzindo um valor de bit error rate, é possível observar que o packet loss naturalmente aumentou, pois, há uma maior probabilidade do pacote ter um erro e consequentemente não ser transmitido. Dessa forma, o average e maximum packet delays têm um pequeno aumento, e como são transmitidos menos pacotes, a taxa de transferência diminui comparativamente a um bit error rate de 0.

Task 3.g - repetir a alínea 3.b mas desta vez com bit error rate 10^{-5}

Código Matlab

```
N=40;
PL=zeros(1,N);
APD=zeros(1,N);
MPD=zeros(1,N);
TT=zeros(1,N);

for i=1:length(lambda)
    for j= 1:N
        [PL(j),APD(j),MPD(j),TT(j)]= simulator2(lambda(i)
            ,C,f,P,bf);
    end
    alfa= 0.1; %90% confidence interval%
    mediaplg(i) = mean(PL);
    termplg(i) = norminv(1-alfa/2)*sqrt(var(PL)/N);
    mediaapdg(i) = mean(APD);
    termapdg(i) = norminv(1-alfa/2)*sqrt(var(APD)/N);
    mediampdg(i) = mean(MPD);
    termmpdg(i) = norminv(1-alfa/2)*sqrt(var(MPD)/N);
    mediattg(i) = mean(TT);
    termttg(i) = norminv(1-alfa/2)*sqrt(var(TT)/N);
end

figure(15)
subplot(1,2,1)
bar(lambda,mediaplb)
hold on
erro=errorbar(lambda,mediaplb,termplb,termplb);
erro.Color=[0 0 0];
erro.LineStyle = 'none';
```

```
29 hold off
30 grid on
31 ylim([0 40])
32 title('b=0')
33 xlabel('lambda values')
34 ylabel('%')
35 subplot(1,2,2)
36 bar(lambda,mediaplg)
37 hold on
38 erro=errorbar(lambda,mediaplg,termplg,termplg);
39 erro.Color=[0 0 0];
40 erro.LineStyle = 'none';
41 hold off
42 grid on
43 ylim([0 40])
44 title('b=10e-5')
45 xlabel('lambda values')
46 ylabel('%')
47 sgtitle('Packet Loss (%)')
48
49 figure(16)
50 subplot(1,2,1)
51 bar(lambda,mediaapdb)
52 hold on
53 erro=errorbar(lambda,mediaapdb,termapdb,termapdb);
54 erro.Color=[0 0 0];
55 erro.LineStyle = 'none';
56 hold off
57 grid on
58 ylim([0 190])
59 title('b=0')
60 xlabel('lambda values')
61 ylabel('milliseconds')
62 subplot(1,2,2)
63 bar(lambda,mediaapdg)
64 hold on
65 erro=errorbar(lambda,mediaapdg,termapdg,termapdg);
66 erro.Color=[0 0 0];
67 erro.LineStyle = 'none';
68 hold off
69 grid on
70 ylim([0 190])
71 title('b=10e-5')
72 xlabel('lambda values')
73 ylabel('milliseconds')
74 sgtitle('Average Packet Delay (milliseconds)')
```

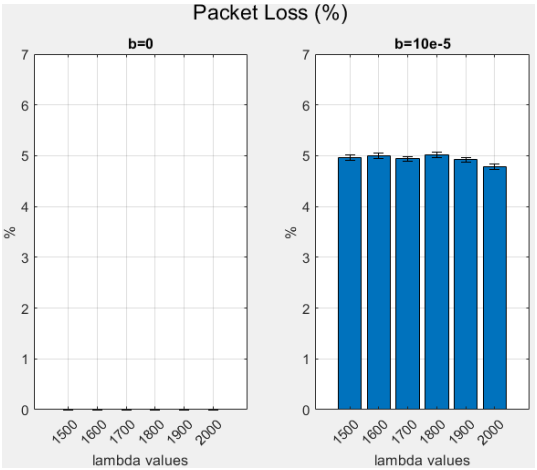
```
75
76  figure(17)
77  subplot(1,2,1)
78  bar(lambda,mediampdb)
79  hold on
80  erro=errorbar(lambda,mediampdb,termmpdb,termmpdb);
81  erro.Color=[0 0 0];
82  erro.LineStyle = 'none';
83  hold off
84  grid on
85  ylim([0 380])
86  title('b=0')
87  xlabel('lambda values')
88  ylabel('milliseconds')
89  subplot(1,2,2)
90  bar(lambda,mediampdg)
91  hold on
92  erro=errorbar(lambda,mediampdg,termmpdg,termmpdg);
93  erro.Color=[0 0 0];
94  erro.LineStyle = 'none';
95  hold off
96  grid on
97  ylim([0 380])
98  title('b=10e-5')
99  xlabel('lambda values')
100 ylabel('milliseconds')
101 sgtitle('Maximum Packet Delay (milliseconds)')
102
103 figure(18)
104 subplot(1,2,1)
105 bar(lambda,mediattb)
106 hold on
107 erro=errorbar(lambda,mediattb,termttb,termttb);
108 erro.Color=[0 0 0];
109 erro.LineStyle = 'none';
110 hold off
111 grid on
112 title('b=0')
113 xlabel('lambda values')
114 ylabel('Mbps')
115 ylim([0 15])
116 subplot(1,2,2)
117 bar(lambda,mediattg)
118 hold on
119 erro=errorbar(lambda,mediattg,termttg,termttg);
120 erro.Color=[0 0 0];
```

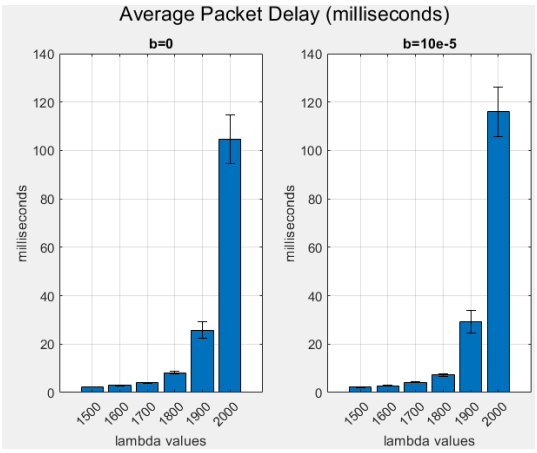
```
121 erro.LineStyle = 'none';
122 hold off
123 grid on
124 ylim([0 15])
125 title('b=10e-5')
126 xlabel('lambda values')
127 ylabel('Mbps')
128 sgtitle('Transmited Throughput (Mbps)')
```

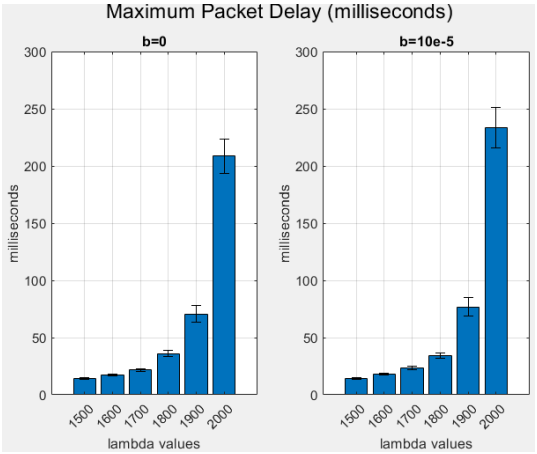
Análise do Código

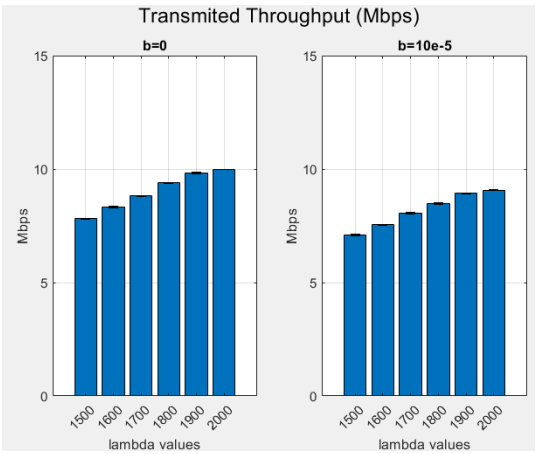
Tal como nas outras alíneas, são criados dois ciclos for, é chamado o simulador, os valores de retorno são guardados nas matrizes e são calculadas as métricas de performance. No final são apresentados os resultados de cada métrica, juntamente com os gráficos produzidos na alínea 3.b para cada métrica, para podermos tirar conclusões sobre o impacto do bit error rate.

Resultados









Análises e Justificações

A explicação da alínea anterior e desta são iguais, pois o bit error rate introduzido tem consequências semelhantes no packet loss, average e maximum packet delay e na taxa de transmissão dos pacotes independemente do tamanho da fila e do núemro de pacotes a serem transmitidos por segundo.

Task 3.h - calcular o atraso médio dos pacotes e throughput total da alínea 3.g assumindo que é modelado por um sistema M/G/1

Código Matlab

```
aux1=[64:1518];
aux3=length(aux1)-3;
aux4=(1-(0.16+0.25+0.2))/aux3;
APD=zeros(1,length(lambda));
TT=zeros(1,length(lambda));

for i=1:length(aux1)
    Pi(i)=(1-bf)^(8*aux1(i));
    if aux1(i)==64
        pn(i)=0.16;
    elseif aux1(i)==110
        pn(i)=0.25;
    elseif aux1(i)==1518
        pn(i)=0.2;
    else
        pn(i)=aux4;
    end
end
for j=1:length(lambda)
    ES(j)=0;
    ES2(j)=0;
    TT(j)=0;
    WQ(j)=0;
    Wi(j)=0;
    APD(j)=0;
    TT(j)=0;
    for i=1:length(aux1)
        ES(j)=ES(j)+pn(i)*(8*aux1(i)/(C*1e6));
        ES2(j)=ES2(j)+pn(i)*((8*aux1(i)/(C*1e6))^2);
        TT(j)=TT(j)+(pn(i)*Pi(i)*lambda(j)*(8*aux1(i)));
    end
    WQ(j)=lambda(j)*ES2(j)/(2*(1-lambda(j)*ES(j)));
    num(j)=0;
    den(j)=0;
    for i=1:length(aux1)
        Wi(j)=WQ(j)+(8*aux1(i)/(C*1e6));
        num(j)=num(j)+(pn(i)*Pi(i)*Wi(j));
        den(j)=den(j)+(pn(i)*Pi(i));
    end
```

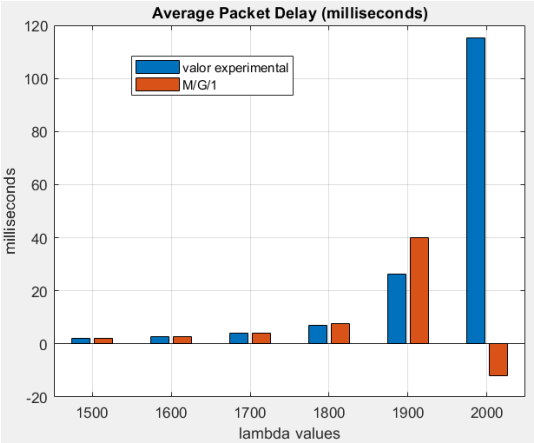
```
40      APD(j)=num(j)/den(j);
41  end
42
43  figure(22)
44  bar(lambda,[mediaapdg;APD.*1000])
45  grid on
46  title('Average Packet Delay (milliseconds)')
47  legend('valor experimental','M/G/1')
48  xlabel('lambda values')
49  ylabel('milliseconds')
50
51  figure(23)
52  bar(lambda,[mediattg;TT./1000000])
53  grid on
54  title('Transmited Throughput (Mbps)')
55  legend('valor experimental','M/G/1')
56  xlabel('lambda values')
57  ylabel('milliseconds')
```

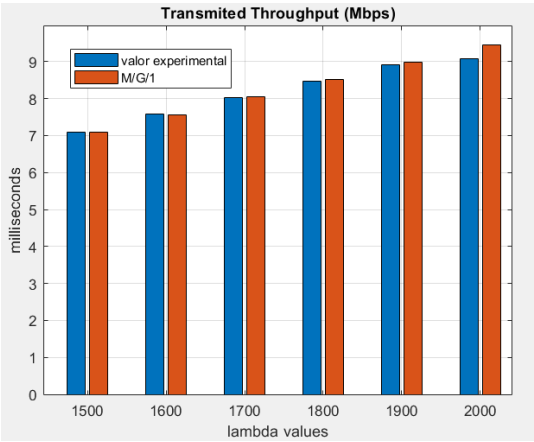
Análise do Código

A matriz aux1 armazena todos os valores possíveis para o tamanho dos pacotes, a variável aux3 armazena o total de valores para o tamanho de pacotes que não têm uma probabilidade definida e a variável aux4 guarda a probabilidade de o pacote ter um dos tamanhos definidos em aux1, exceto 64, 110 e 1518, que já têm probabilidaes definidas. De seguida são inicializadas matrizes para guardar os valores pretendidos e é feito um ciclo for para calcular a probabilidade de cada pacote ser enviado sem erros e percorrida a matriz que guarda os possíveis tamanhos de cada pacote. Caso esse valor seja 64, 110 ou 1518, o valor da probabilidade para esse tamanho, 16, 25 e 20%, é guardado na matriz pn. Caso seja qualquer outro valor para o tamanho do pacote, o valor guardado na matriz pn é aux4. Em seguida é feito outro ciclo for, no qual inicializadas novas matrizes para o armazenamento de valores necessários e são calculados os valores de média de transmissão de pacotes (E[S]) e o seu quadrado (E[S^2]), assim como o throughput total do sistema. A seguir, através da fórmula de Pollaczek - Khintchine, é calculado o atraso médio na fila de espera. Seguidamente é calculado o atraso médio de cada tamanho de pacote e calculado o numerador e denominador, para finalemente serem usados no cálculo do atraso médio dos pacotes.

Por fim são feitos os gráficos de barras, sendo que cada gráfico de barras corresponde a uma métrica e são apresentados em cada gráfico os valores teóricos e os valores obtidos por simulação, para poderem ser comparados.

Resultados





Análises e Justificações

Os resultados teóricos do modelo M/G/1 são semelhantes aos resultados obtidos por simulação, exceto quando o número de pacotes a serem transmitidos é superior ao tamanho da fila e consequentemente torna-se inválido usar o sistema M/G/1, já que este assume uma fila de espera infinita.

Task 4

Simulador 3

Código Matlab

```
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)
% OUTPUT PARAMETERS:
%  PL   - packet loss (%)
%  APD  - average packet delay (milliseconds)
%  MPD  - maximum packet delay (milliseconds)
%  TT   - transmitted throughput (Mbps)

% Initializing flow state:
time = 1/10;

FLOWSTATE =GetTransitionState()

%Events:
ARRIVAL= 0;        % Arrival of a packet
DEPARTURE= 1;      % Departure of a packet
TRANSITION = 2;    % Transition between states

%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
```

```
33 DELAYS= 0;                % Sum of the delays of transmitted
       packets
34 MAXDELAY= 0;              % Maximum delay among all
       transmitted packets
35
36 %Auxiliary variables:
37 % Initializing the simulation clock:
38 Clock= 0;
39
40 % Initializing the List of Events with the first ARRIVAL:
41 EventList = [ARRIVAL, Clock + exprnd(1/(rate(FLOWSTATE)))
       , GeneratePacketSize(), 0];
42 EventList = [EventList; TRANSITION, Clock + exprnd(time),
       0, 0];
43
44
45 %Similation loop:
46 while TRANSMITTEDPACKETS<P                    % Stopping
       criterium
47     EventList= sortrows(EventList,2);        % Order
           EventList by time
48     Event= EventList(1,1);                   % Get first
           event and
49     Clock= EventList(1,2);                   %   and
50     PacketSize= EventList(1,3);              %   associated
51     ArrivalInstant= EventList(1,4);          %   parameters.
52     EventList(1,:)= [];                      % Eliminate
           first event
53     switch Event
54         case ARRIVAL                         % If first event
               is an ARRIVAL
55             TOTALPACKETS= TOTALPACKETS+1;
56             EventList = [EventList; ARRIVAL, Clock +
                   exprnd(1/(rate(FLOWSTATE))),
                   GeneratePacketSize(), 0];
57             if STATE==0
58                 STATE= 1;
59                 EventList = [EventList; DEPARTURE, Clock
                       + 8*PacketSize/(C*10^6), PacketSize,
                       Clock];
60             else
61                 if QUEUEOCCUPATION + PacketSize <= f
62                     QUEUE= [QUEUE;PacketSize , Clock];
63                     QUEUEOCCUPATION= QUEUEOCCUPATION +
                           PacketSize;
64                 else
```

```
65                      LOSTPACKETS= LOSTPACKETS + 1;
66                  end
67              end
68          case DEPARTURE                          % If first
                event is a DEPARTURE
69              if rand() < (1-b)^(8*PacketSize)
70                  TRANSMITTEDPACKETS= TRANSMITTEDPACKETS +
                        1;
71                  TRANSMITTEDBYTES= TRANSMITTEDBYTES +
                        PacketSize;
72                  DELAYS= DELAYS + (Clock - ArrivalInstant)
                        ;
73                  if Clock - ArrivalInstant > MAXDELAY
74                      MAXDELAY= Clock - ArrivalInstant;
75                  end
76              else
77                 LOSTPACKETS = LOSTPACKETS + 1;
78              end
79
80              if QUEUEOCCUPATION > 0
81                  EventList = [EventList; DEPARTURE, Clock
                        + 8*QUEUE(1,1)/(C*10^6), QUEUE(1,1),
                        QUEUE(1,2)];
82                  QUEUEOCCUPATION= QUEUEOCCUPATION - QUEUE
                        (1,1);
83                  QUEUE(1,:)= [];
84              else
85                  STATE= 0;
86              end
87          case TRANSITION
88              EventList = [EventList; TRANSITION, Clock +
                    exprnd(time), 0, 0];
89              if FLOWSTATE ~= 2
90                 FLOWSTATE = 2;
91              else
92                  if rand<0.5
93                     FLOWSTATE = 3;
94                  else
95                     FLOWSTATE = 1;
96                  end
97              end
98      end
99  end
100
101 %Performance parameters determination:
102 PL= 100*LOSTPACKETS/TOTALPACKETS;       % in %
```

```
103 APD= 1000*DELAYS/TRANSMITTEDPACKETS;    % in milliseconds
104 MPD= 1000*MAXDELAY;                     % in milliseconds
105 TT= 10^(-6)*TRANSMITTEDBYTES*8/Clock;   % in Mbps
106
107 end
108
109 % determinar o estado incial
110 function out= GetTransitionState()
111     p1 = 1 / (1 + 10/5 + (10/5) * (5/10));
112     p2 = (10/5) / (1 + 10/5 + (10/5) * (5/10));
113     % p3 = ((10/5) * (5/10)) / (1 + 10/5 + (10/5) *
            (5/10));
114
115     aux = rand();
116
117     if aux <= p1
118         out = 1;
119     elseif aux <= p1 + p2
120         out = 2;
121     else
122         out = 3;
123     end
124 end
125
126 function out= GeneratePacketSize()
127     aux= rand();
128     aux2= [65:109 111:1517];
129     if aux <= 0.16
130         out= 64;
131     elseif aux <= 0.16 + 0.25
132         out= 110;
133     elseif aux <= 0.16 + 0.25 + 0.2
134         out= 1518;
135     else
136         out = aux2(randi(length(aux2)));
137     end
138 end
```

Análise do Código

Começámos por calcular a **probabilidade do estado inicial através das "Probabilidades limite de processos de nascimento e morte"**. De

seguida, inicializámos os valores da taxa de pacotes para cada estado e comparando um valor rand com as probabilidades calculadas anteriormente, conseguimos então definir o seu estado inicial.
Ainda antes do início da transmissão dos pacotes definimos a que instante será efetuada a próxima transição.
Diferente da versão anterior do simulador, desta vez no estado de ARRIVAL, utilizamos o valor de lambda correspondente ao estado atual, em vez de um valor geral previamente definido.
Por último, sempre que será efetuada uma transição de estado, verificamos o estado atual, e se estiver no estado 1 ou estado 3, será automaticamente transitado para o estado 2. Quando o estado atual é o estado 2, dado que a taxa de partida para cada um dos outros estados é igual, então comparando um rand com um 0.5 conseguimos calcular para que estado irá ser feita a transição.

Task 4.a - correr o Simulador 3 10 vezes com os parâmetros de entrada pedidos e calcular os valores estimados e os intervalos de confiança

Código Matlab

```
C=10;
f=10^6;
b=0;
P=100000;
N = 10; %number of simulations
lambda=1800;
PL=zeros(1,N);
APD=zeros(1,N);
MPD=zeros(1,N);
TT=zeros(1,N);
for j= 1:N
    [PL(j),APD(j),MPD(j),TT(j)]= simulator3(lambda,C,f,P,
        b);
end
alfa= 0.1; %90% confidence interval%
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);

```

```
24 fprintf('Packet Loss (%%) = %.4e +- %.4e\n',mediapl,
       termpl)
25 fprintf('Av. Packet delay (ms) = %.4e +- %.4e\n',mediaapd
       ,termapd)
26 fprintf('Max. Packet Delay (ms) = %.4e +- %.4e\n',
       mediampd,termmpd)
27 fprintf('Throughput (Mbps) = %.4e +- %.4e\n',mediatt,
       termtt)
```

Análise do Código

Inicialmente são declarados os parâmetros de entrada para o simulador3, é feito um ciclo for para o número de corridas, é invocado o simulador e são guardados os valores de retorno em matrizes. De seguida são calculadas métricas de performance e são apresentadas.

Resultados

```
Packet Loss (%) = 4.0861e-03 +- 6.7210e-03
Av. Packet delay (ms) = 1.4362e+02 +- 3.4062e+01
Max. Packet Delay (ms) = 5.9568e+02 +- 1.0280e+02
Throughput (Mbps) = 9.3241e+00 +- 1.0301e-01
```

Task 4.b - repetir a alínea anterior mas desta vez para diferentes valores de f e b

Código Matlab

```
1  fb=10^4;
2  bb=10^-5;
3  PL=zeros(1,N);
4  APD=zeros(1,N);
5  MPD=zeros(1,N);
6  TT=zeros(1,N);
7
8  for j= 1:N
9      [PL(j),APD(j),MPD(j),TT(j)]= simulator3(lambda,C,fb,P
           ,bb);
10 end
11 alfa= 0.1; %90% confidence interval%
12 mediaplb = mean(PL);
13 termplb = norminv(1-alfa/2)*sqrt(var(PL)/N);
```

```
14 mediaapdb = mean(APD);
15 termapdb = norminv(1-alfa/2)*sqrt(var(APD)/N);
16 mediampdb = mean(MPD);
17 termmpdb = norminv(1-alfa/2)*sqrt(var(MPD)/N);
18 mediattb = mean(TT);
19 termttb = norminv(1-alfa/2)*sqrt(var(TT)/N);
20
21 fprintf('Packet Loss (%%) = %.4e +- %.4e\n',mediaplb,
       termplb)
22 fprintf('Av. Packet delay (ms) = %.4e +- %.4e\n',
       mediaapdb,termapdb)
23 fprintf('Max. Packet Delay (ms) = %.4e +- %.4e\n',
       mediampdb,termmpdb)
24 fprintf('Throughput (Mbps) = %.4e +- %.4e\n',mediattb,
       termttb)
```

Análise do Código

Tal como na alínea anterior, são declarados os parâmetros de entrada do simulador, é feito um ciclo for para as corridas, é chamado o simulador, são guardados os valores de retorno em matrizes e são calculadas as métricas de desempenho, para serem apresentadas de seguida.

Resultados

```
Packet Loss (%) = 1.0676e+01 +- 1.7151e-01
Av. Packet delay (ms) = 4.2151e+00 +- 5.0786e-02
Max. Packet Delay (ms) = 9.1735e+00 +- 1.0907e-02
Throughput (Mbps) = 7.5001e+00 +- 4.4968e-02
```

Task 4.c - correr o Simulador 2 e o Simulador 3 10 vezes cada com os parâmetros de entrada pedidos e calcular os valores estimados e os intervalos de confiança, apresentando ambos os resultados de cada parâmetro no mesmo gráfico de barras

Código Matlab

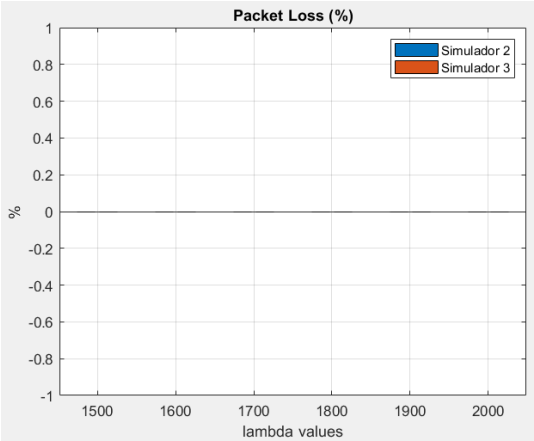
```
fc=10^7;
lambdac=[1500,1600,1700,1800,1900,2000];
PL3=zeros(1,N);
APD3=zeros(1,N);
MPD3=zeros(1,N);
TT3=zeros(1,N);
PL2=zeros(1,N);
APD2=zeros(1,N);
MPD2=zeros(1,N);
TT2=zeros(1,N);

for i=1:length(lambdac)
    for j= 1:N
        [PL2(j),APD2(j),MPD2(j),TT2(j)]= simulator2(
            lambdac(i),C,fc,P,b);
        [PL3(j),APD3(j),MPD3(j),TT3(j)]= simulator3(
            lambdac(i),C,fc,P,b);
    end
    alfa= 0.1; %90% confidence interval%
    mediaplc2(i) = mean(PL2);
    termplc2(i) = norminv(1-alfa/2)*sqrt(var(PL2)/N);
    mediaapdc2(i) = mean(APD2);
    termapdc2(i) = norminv(1-alfa/2)*sqrt(var(APD2)/N);
    mediampdc2(i) = mean(MPD2);
    termmpdc2(i) = norminv(1-alfa/2)*sqrt(var(MPD2)/N);
    mediattc2(i) = mean(TT2);
    termttc2(i) = norminv(1-alfa/2)*sqrt(var(TT2)/N);

    mediaplc3(i) = mean(PL3);
    termplc3(i) = norminv(1-alfa/2)*sqrt(var(PL3)/N);
    mediaapdc3(i) = mean(APD3);
    termapdc3(i) = norminv(1-alfa/2)*sqrt(var(APD3)/N);
    mediampdc3(i) = mean(MPD3);
    termmpdc3(i) = norminv(1-alfa/2)*sqrt(var(MPD3)/N);
    mediattc3(i) = mean(TT3);
    termttc3(i) = norminv(1-alfa/2)*sqrt(var(TT3)/N);
end
```

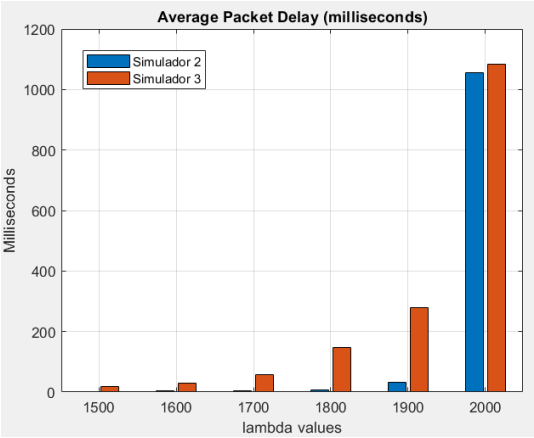
```
36
37 figure(1)
38 bar(lambdac,[mediaplc2;mediaplc3])
39 grid on
40 title('Packet Loss (%)')
41
42 figure(2)
43 bar(lambdac,[mediaapdc2;mediaapdc3])
44 grid on
45 title('Average Packet Delay (milliseconds)')
46
47 figure(3)
48 bar(lambda,[mediampdc2;mediampdc3])
49 grid on
50 title('Maximum Packet Delay (milliseconds)')
51
52 figure(4)
53 bar(lambda,[mediattc2;mediattc3])
54 grid on
55 title('Transmited Throughput (Mbps)')
```

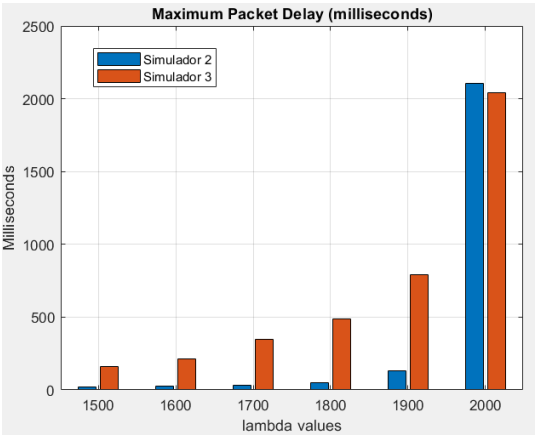
Análise do Código

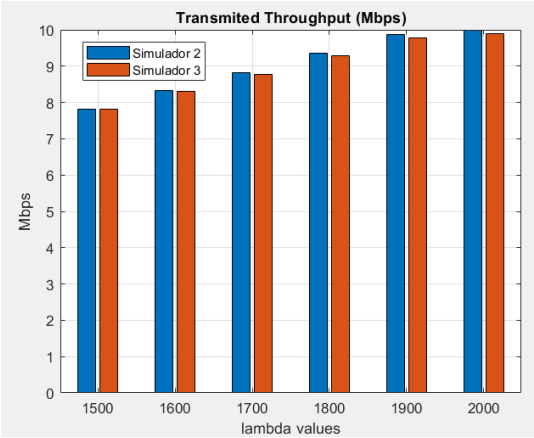
Primeiramente são declarados os valores dos parâmetros de entrada que mudam de valor em relação às alíneas anteriores e em seguida são feitos dois ciclos for, um para percorrer todos os valores possíveis para o tamanho da fila de espera e outro para fazer as corridas pedidas. Dentros desses ciclos são chamados os simuladores 2 e 3 com os mesmos parâmetros de entrada. Em seguida são calculadas as métricas de desempenho para cada um dos simuladores. Por fim são apresentados os valores de cada métrica para cada simulador, sendo que a mesma métrica obtida por cada um dos simuladores é apresentada no mesmo gráfico.

Resultados









Análises e Justificações

Pela observação do gráfico é possível concluir que dado um bit error rate igual a 0 a percentagem de pacotes perdidos também é 0. A maior discrepância observável é no average e maximum packet delay onde o simulador 3 evidentemente introduz um packed delay superior, pois trata-se de um processo de Poisson onde é introduzida uma taxa de ocorrência de um dado evento que neste caso se traduz na transição para um estado de chegada ou de saída. Dado os mesmos parâmetros de entrada, a taxa total de transferência de pacotes é

praticamente igual nos dois simuladores, no entanto, o simulador 3, devido aos delays introduzidos, tem uma taxa total de transferência ligeiramente menor.

Task 4.d - repetir a alínea anterior para diferentes parâmetros de entrada

Código Matlab

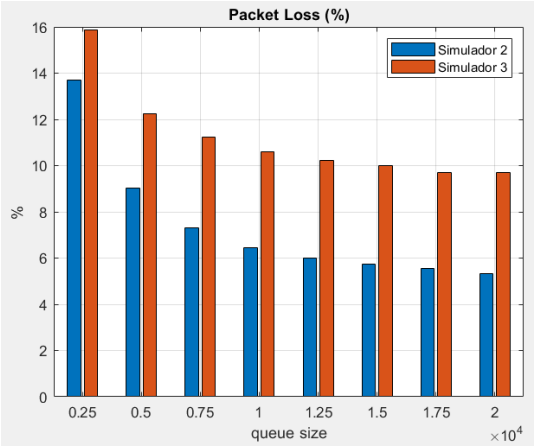
```
fd=[2500,5000,7500,10000,12500,15000,17500,20000];
PL3=zeros(1,N);
APD3=zeros(1,N);
MPD3=zeros(1,N);
TT3=zeros(1,N);
PL2=zeros(1,N);
APD2=zeros(1,N);
MPD2=zeros(1,N);
TT2=zeros(1,N);

for i=1:length(fd)
    for j= 1:N
        [PL2(j),APD2(j),MPD2(j),TT2(j)]= simulator2(
            lambda,C,fd(i),P,b);
        [PL3(j),APD3(j),MPD3(j),TT3(j)]= simulator3(
            lambda,C,fd(i),P,b);
    end
    alfa= 0.1; %90% confidence interval%
    mediapld2(i) = mean(PL2);
    termpld2(i) = norminv(1-alfa/2)*sqrt(var(PL2)/N);
    mediaapdd2(i) = mean(APD2);
    termapdd2(i) = norminv(1-alfa/2)*sqrt(var(APD2)/N);
    mediampdd2(i) = mean(MPD2);
    termmpdd2(i) = norminv(1-alfa/2)*sqrt(var(MPD2)/N);
    mediattd2(i) = mean(TT2);
    termttd2(i) = norminv(1-alfa/2)*sqrt(var(TT2)/N);

    mediapld3(i) = mean(PL3);
    termpld3(i) = norminv(1-alfa/2)*sqrt(var(PL3)/N);
    mediaapdd3(i) = mean(APD3);
    termapdd3(i) = norminv(1-alfa/2)*sqrt(var(APD3)/N);
    mediampdd3(i) = mean(MPD3);
    termmpdd3(i) = norminv(1-alfa/2)*sqrt(var(MPD3)/N);
    mediattd3(i) = mean(TT3);
    termttd3(i) = norminv(1-alfa/2)*sqrt(var(TT3)/N);
end

```

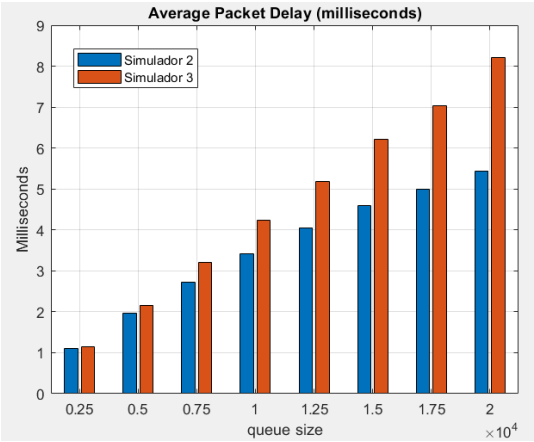
```
figure(5)
bar(fd,[mediapld2;mediapld3])
grid on
title('Packet Loss (%)')

figure(6)
bar(fd,[mediaapdd2;mediaapdd3])
grid on
title('Average Packet Delay (milliseconds)')

figure(7)
bar(fd,[mediampdd2;mediampdd3])
grid on
title('Maximum Packet Delay (milliseconds)')

figure(8)
bar(fd,[mediattd2;mediattd3])
grid on
title('Transmited Throughput (Mbps)')
```

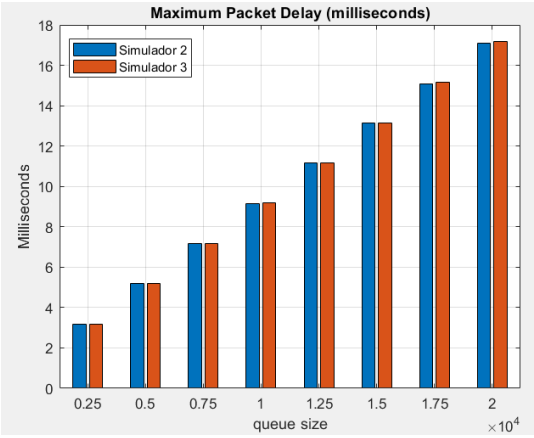
Análise do Código

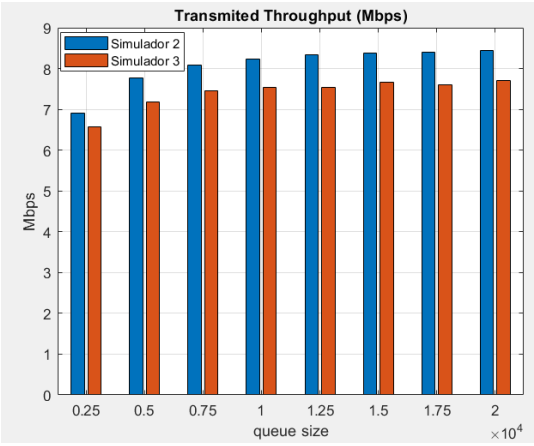
Nesta alínea foram seguidos os mesmos passos que para a alínea anterior (declaração dos parâmetros de entrada, ciclos for, invocação de ambos os simuladores, cálculo das métricas de desempenho para ambos os simuladores e apresentação dos gráficos de barras) mas para valores diferentes do tamanho da fila de espera e valor fixo de lambda.

Resultados









Análises e Justificações

É expetável que com o aumento da capacidade da fila de espera, a percentagem de pacotes perdidos diminua em ambos os casos e os packets delays tenham o mesmo comportamento dos resultados obtidos na alínea anterior. Portanto, pela observação do gráfico, é possível concluir que este comportamento realmente se verifica, isto é, o average e maximum packet delay do simulador 3 são superiores, mas neste caso com uma discrepância inferior, e a taxa total de transmissão de pacotes é superior no simulador 2.

Task 4.e - repetir a alínea 4.c com b=10^{-5} e comparar os resultados, tirando conclusões sobre o impacto de ambos os modelos de chegada de pacotes nos parâmetros de performance

Código Matlab

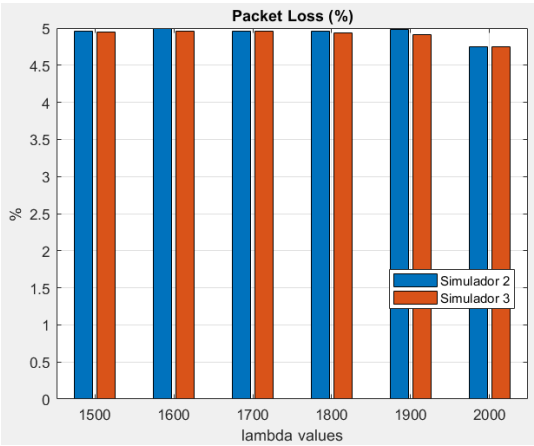
```
PL3=zeros(1,N);
APD3=zeros(1,N);
MPD3=zeros(1,N);
TT3=zeros(1,N);
PL2=zeros(1,N);
APD2=zeros(1,N);
MPD2=zeros(1,N);
TT2=zeros(1,N);

for i=1:length(lambdac)
    for j= 1:N
        [PL2(j),APD2(j),MPD2(j),TT2(j)]= simulator2(
            lambdac(i),C,fc,P,bb);
        [PL3(j),APD3(j),MPD3(j),TT3(j)]= simulator3(
            lambdac(i),C,fc,P,bb);
    end
    alfa= 0.1; %90% confidence interval%
    mediaple2(i) = mean(PL2);
    termple2(i) = norminv(1-alfa/2)*sqrt(var(PL2)/N);
    mediaapde2(i) = mean(APD2);
    termapde2(i) = norminv(1-alfa/2)*sqrt(var(APD2)/N);
    mediampde2(i) = mean(MPD2);
    termmpde2(i) = norminv(1-alfa/2)*sqrt(var(MPD2)/N);
    mediatte2(i) = mean(TT2);
    termtte2(i) = norminv(1-alfa/2)*sqrt(var(TT2)/N);

    mediaple3(i) = mean(PL3);
    termple3(i) = norminv(1-alfa/2)*sqrt(var(PL3)/N);
    mediaapde3(i) = mean(APD3);
    termapde3(i) = norminv(1-alfa/2)*sqrt(var(APD3)/N);
    mediampde3(i) = mean(MPD3);
    termmpde3(i) = norminv(1-alfa/2)*sqrt(var(MPD3)/N);
    mediatte3(i) = mean(TT3);
    termtte3(i) = norminv(1-alfa/2)*sqrt(var(TT3)/N);
end

figure(9)
bar(lambdac,[mediaple2;mediaple3])
```

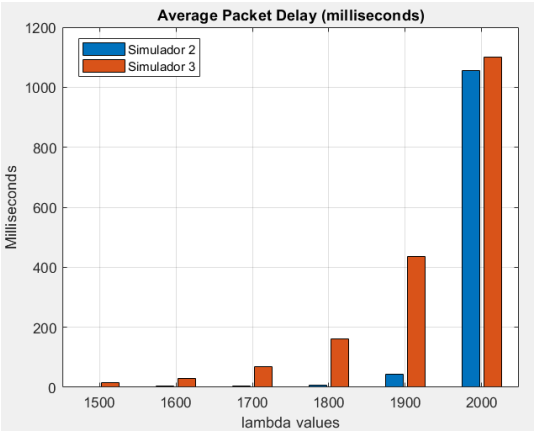
```
grid on
title('Packet Loss (%)')

figure(10)
bar(lambdac,[mediaapde2;mediaapde3])
grid on
title('Average Packet Delay (milliseconds)')

figure(11)
bar(lambda,[mediampde2;mediampde3])
grid on
title('Maximum Packet Delay (milliseconds)')

figure(12)
bar(lambda,[mediatte2;mediatte3])
grid on
title('Transmited Throughput (Mbps)')
```

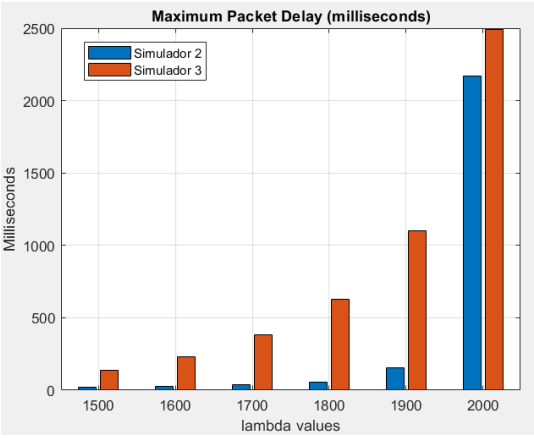
Análise do Código

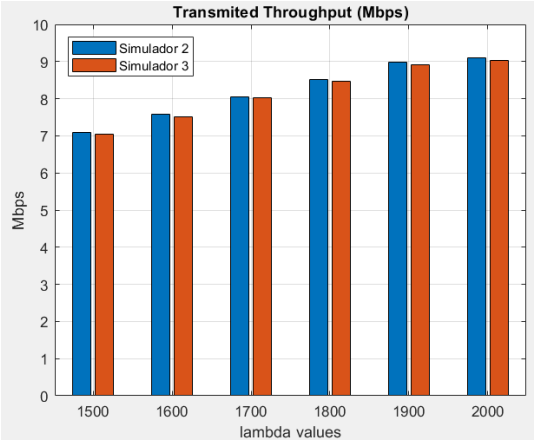
Como os valores para os parâmetros de entrada necessários nesta alínea já foram todos declarados anteriormente, o primeiro passo é fazer os ciclos for para percorrer os vários valores de lambda e fazer as várias corridas. De seguida são invocados ambos os simuladores e os valores de retorno são guardados em matrizes. De seguida são calculadas as métricas de performance. Por fim, são apresentados os gráficos de barras para cada métrica obtidos em cada simulador para melhor comparação e ser mais fácil tirar conclusões sobre o impacto do bit error rate.

Resultados









Análises e Justificações

Com a introdução do bit error rate, comparativamente aos resultados obtidos na alínea 4.c, o packet loss deixa de ser 0 e passa a ter uma percentagem relativamente pequena em ambos os simuladores. O average e maximum pakcet delay deverão ter um comportamento semelhante à alinea 4.c, com um pequeno aumento nos seus tempos de saída introduzidos pela perda de alguns pacotes. A transmissão total de pacotes também tem um comportamento semelhante, mas com uma ligeira diminuição devido ao *ber* introduzido.

Task 4.f - repetir a alínea 4.d com b=10^{-5} e comparar os resultados, tirando conclusões sobre o impacto de ambos os modelos de chegada de pacotes nos parâmetros de performance

Código Matlab

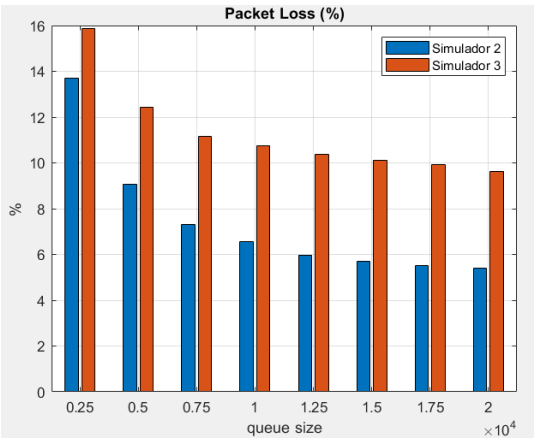
```
PL3=zeros(1,N);
APD3=zeros(1,N);
MPD3=zeros(1,N);
TT3=zeros(1,N);
PL2=zeros(1,N);
APD2=zeros(1,N);
MPD2=zeros(1,N);
TT2=zeros(1,N);

for i=1:length(fd)
    for j= 1:N
        [PL2(j),APD2(j),MPD2(j),TT2(j)]= simulator2(
            lambda,C,fd(i),P,bb);
        [PL3(j),APD3(j),MPD3(j),TT3(j)]= simulator3(
            lambda,C,fd(i),P,bb);
    end
    alfa= 0.1; %90% confidence interval%
    mediaplf2(i) = mean(PL2);
    termplf2(i) = norminv(1-alfa/2)*sqrt(var(PL2)/N);
    mediaapdf2(i) = mean(APD2);
    termapdf2(i) = norminv(1-alfa/2)*sqrt(var(APD2)/N);
    mediampdf2(i) = mean(MPD2);
    termmpdf2(i) = norminv(1-alfa/2)*sqrt(var(MPD2)/N);
    mediattf2(i) = mean(TT2);
    termttf2(i) = norminv(1-alfa/2)*sqrt(var(TT2)/N);

    mediaplf3(i) = mean(PL3);
    termplf3(i) = norminv(1-alfa/2)*sqrt(var(PL3)/N);
    mediaapdf3(i) = mean(APD3);
    termapdf3(i) = norminv(1-alfa/2)*sqrt(var(APD3)/N);
    mediampdf3(i) = mean(MPD3);
    termmpdf3(i) = norminv(1-alfa/2)*sqrt(var(MPD3)/N);
    mediattf3(i) = mean(TT3);
    termttf3(i) = norminv(1-alfa/2)*sqrt(var(TT3)/N);
end

figure(13)
bar(fd,[mediaplf2;mediaplf3])
```

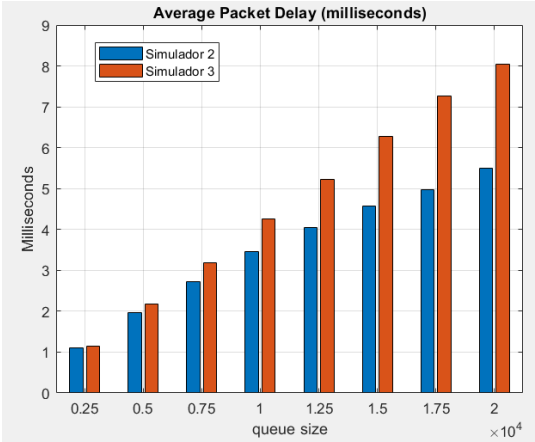
```
37 grid on
38 title('Packet Loss (%)')
39
40 figure(14)
41 bar(fd,[mediaapdf2;mediaapdf3])
42 grid on
43 title('Average Packet Delay (milliseconds)')
44
45 figure(15)
46 bar(fd,[mediampdf2;mediampdf3])
47 grid on
48 title('Maximum Packet Delay (milliseconds)')
49
50 figure(16)
51 bar(fd,[mediattf2;mediattf3])
52 grid on
53 title('Transmited Throughput (Mbps)')
```

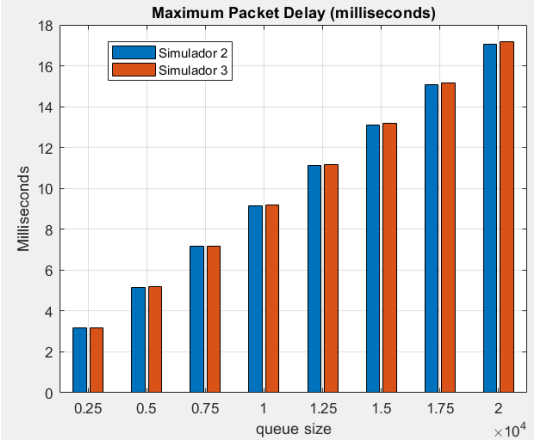
Análise do Código

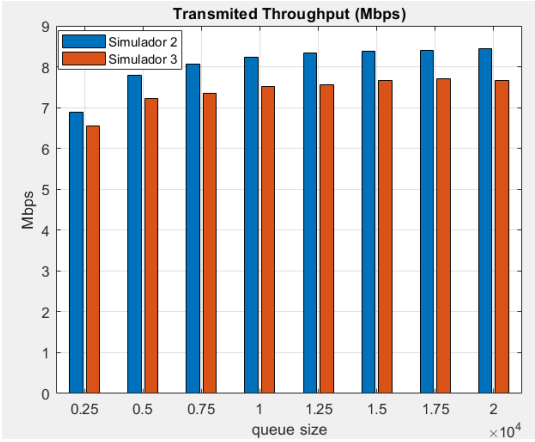
Tal como na alínea anterior, todos os parâmetros de entrada necessários já foram declarados anteriormente e, por isso, a primeira coisa a fazer são os ciclos for para percorrer os vários valores para o tamanho da fila de espera e para fazer as corridas pedidas. Em seguida são invocados ambos os simuladores e são calculadas as métricas de desempenho para cada um deles. Por fim são criados os gráficos de barras para cada métrica, sendo que valores de simuladores diferentes mas que dizem respeito à mesma métrica aparecem no mesmo gráfico.

Resultados









Análises e Justificações

Pela observação dos gráficos, os simuladores apresentam um comportamento semelhante à alinea 4.d. Apenas existe um aumento pouco significativo no delay minimo e máximo dos pacotes a serem transmitidos. Os restantes resultados são praticamente iguais, não havendo uma diferença significativa com a introdução do bit error rate.