Performance of a Packet Switching Connection

Desempenho e Dimensionamento de Redes
Universidade de Aveiro

Jorge Catarino, Ricardo Azevedo

Version 1.0

Performance of a Packet Switching Connection

DETI
Desempenho e Dimensionamento de Redes
Universidade de Aveiro

(85028) jorge.catarino@ua.pt

(84730) azevedoricardo98@ua.pt

12 de Junho, 2021

Conteúdo

Lista de Figuras

Capítulo 1

Tarefa 3

Neste capítulo iremos expor os resultados das várias alíneas da Tarefa 3, tal como o raciocínio para os alcançar. Será também apresentado o código desenvolvido e, quando considerado necessário, terá uma reflexão sobre os resultados obtidos.

1.1 Código Auxiliar

1.1.1 Função *runSimulator2*

Com o objectivo de reduzir a repetição de código no desenvolvimento desta tarefa, foi criada uma função que permite correr o simulador N vezes, calculando a médias e os erros associados a cada resultado.

O código desta função encontra-se reproduzido abaixo.

```
1  function [medias_PL, temp_PL , medias_APD, temp_APD ,
       medias_MDP, temp_MDP , medias_TT, temp_TT] =  ...
2      runSimulator2(n_times, alfa, lambda, C, f, P, b)
3
4  results_PL = zeros(0,n_times);
5  results_APD = zeros(0,n_times);
6  results_MDP = zeros(0,n_times);
7  results_TT = zeros(0,n_times);
8
9  for it = 1:n_times
10     [results_PL(it),results_APD(it),results_MDP(it),
           results_TT(it)] =  Simulator2(lambda, C, f, P,
           b);
11 end
12
```

```
13 medias_PL = mean(results_PL);
14 medias_APD = mean(results_APD);
15 medias_MDP = mean(results_MDP);
16 medias_TT = mean(results_TT);
17
18 temp_PL = norminv(1-alfa/2)*sqrt(var(results_PL)/
       n_times);
19 temp_APD = norminv(1-alfa/2)*sqrt(var(results_APD)/
       n_times);
20 temp_MDP = norminv(1-alfa/2)*sqrt(var(results_MDP)/
       n_times);
21 temp_TT = norminv(1-alfa/2)*sqrt(var(results_TT)/
       n_times);
```

1.2 Alínea a) e Alínea b)

1.2.1 Método e Resultados

Com o auxílio da função descrita anteriormente, foi então executado o simulador com os parâmetros $P = 10000$, $\lambda = [1500, 1600, 1700, 1800, 1900, 2000]$, $C = 10$, $f = 1e7$ e $b = 0$. Para obter os resultados, o simulador foi executado 10 vezes. Como se pode observar pelos resultados na Figura 1.1, este número de execuções produz intervalos de confiança demasiado elevados, sendo que estes resultados possuem pouca precisão estatística.

Para obter os resultados da alínea b) foram usados os mesmos parâmetros da alínea a), excepto que o simulador foi executado 40 vezes.

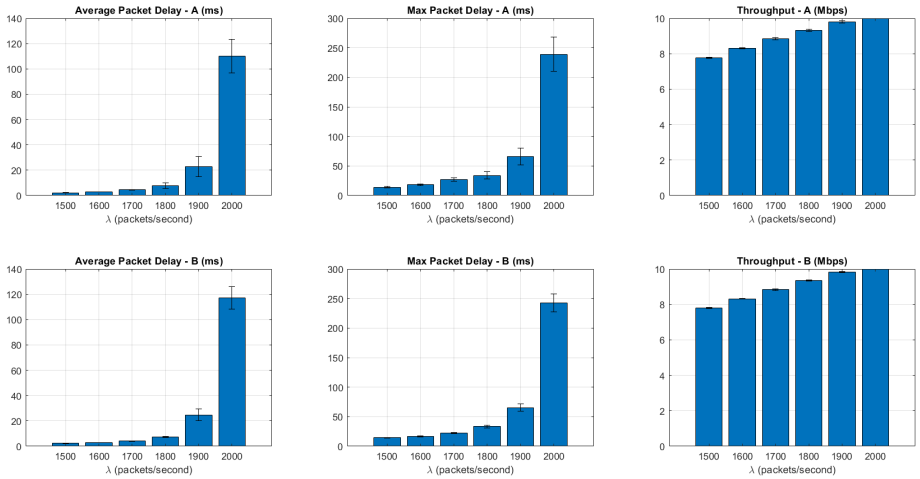

Figura 1.1: Resultado da alínea a) e alínea b)

1.2.2 Código

```
%% 3a) e 3b)
% Parameters
n_runs = 10;
P = 10000; %stoping criterion
alpha = 0.10; %confidence intervals
lambda_values = [1500, 1600, 1700, 1800, 1900, 2000];
    %packet rate
C = 10; %conection capacity
f = 10000000; %queue size
```

```
b = 0; %bit error rate

% Results arrays
sz = length(lambda_values);
medias_PL_a = zeros(1, sz);
temp_PL_a = zeros(1, sz);
medias_APD_a = zeros(1, sz);
temp_APD_a = zeros(1, sz);
medias_MDP_a = zeros(1, sz);
temp_MDP_a = zeros(1, sz);
medias_TT_a = zeros(1, sz);
temp_TT_a = zeros(1, sz);

% Run simulator n times for each value of lambda
for i = 1:sz
    [medias_PL_a(i), temp_PL_a(i) , medias_APD_a(i), temp_APD_a(i), ...
     medias_MDP_a(i), temp_MDP_a(i) , medias_TT_a(i), temp_TT_a(i)] = ...
    runSimulator2(n_runs, alpha, lambda_values(i), C, f, P,b);
end

% Parameters
n_runs = 40;
P = 10000; %stoping criterion
alpha = 0.10; %confidence intervals
lambda_values = [1500, 1600, 1700, 1800, 1900, 2000]; %packet rate
C = 10; %conection capacity
f = 10000000; %queue size
b = 0; %bit error rate

% Results arrays
sz = length(lambda_values);
medias_PL_b = zeros(1, sz);
temp_PL_b = zeros(1, sz);
medias_APD_b = zeros(1, sz);
temp_APD_b = zeros(1, sz);
medias_MDP_b = zeros(1, sz);
temp_MDP_b = zeros(1, sz);
medias_TT_b = zeros(1, sz);
temp_TT_b = zeros(1, sz);

% Run simulator n times for each value of lambda
for i = 1:sz
    [medias_PL_b(i), temp_PL_b(i) , medias_APD_b(i), temp_APD_b(i), ...
     medias_MDP_b(i), temp_MDP_b(i) , medias_TT_b(i), temp_TT_b(i)] = ...
```

```
    runSimulator2(n_runs, alpha, lambda_values(i), C, f, P,b);
end

figure(1)
tiledlayout(2,3)
% Average Packet Delay A
nexttile;
bar(lambda_values,medias_APD_a)
title("Average Packet Delay - A (ms)")
xlabel('\lambda (packets/second)')
grid on

hold on
er = errorbar(lambda_values, medias_APD_a, temp_APD_a);
er.Color = [0 0 0];
er.LineStyle = 'none';
hold off

% Max Packet Delay - A
nexttile;
bar(lambda_values,medias_MDP_a)
title("Max Packet Delay - A (ms)")
xlabel('\lambda (packets/second)')
grid on

hold on
er = errorbar(lambda_values, medias_MDP_a, temp_MDP_a);
er.Color = [0 0 0];
er.LineStyle = 'none';
hold off

% Throughput - A
nexttile;
bar(lambda_values,medias_TT_a)
title("Throughput - A (Mbps)")
xlabel('\lambda (packets/second)')
grid on

hold on
er = errorbar(lambda_values, medias_TT_a, temp_TT_a);
er.Color = [0 0 0];
er.LineStyle = 'none';
hold off


% Average Packet Delay B
nexttile;
```

```
bar(lambda_values,medias_APD_b)
title("Average Packet Delay - B (ms)")
xlabel('\lambda (packets/second)')
grid on

hold on
er = errorbar(lambda_values, medias_APD_b, temp_APD_b)
    ;
er.Color = [0 0 0];
er.LineStyle = 'none';
hold off

% Max Packet Delay - B
nexttile;
bar(lambda_values,medias_MDP_b)
title("Max Packet Delay - B (ms)")
xlabel('\lambda (packets/second)')
grid on

hold on
er = errorbar(lambda_values, medias_MDP_b, temp_MDP_b)
    ;
er.Color = [0 0 0];
er.LineStyle = 'none';
hold off

% Throughput - B
nexttile;
bar(lambda_values,medias_TT_b)
title("Throughput - B (Mbps)")
xlabel('\lambda (packets/second)')
grid on

hold on
er = errorbar(lambda_values, medias_TT_b, temp_TT_b);
er.Color = [0 0 0];
er.LineStyle = 'none';
hold off
```

1.2.3 Discussão

Se os resultados desta alínea b) forem comparados aos da alínea anterior, é possível observar então que houve uma diminuição no tamanho das barras de erro, o que por sua vez significa que os intervalos de confiança são mais reduzidos. Isto implica que ao executar o simulador 40 vezes é possível obter resultados mais precisos estatisticamente, do que se só for executado 10 vezes.

1.3 Alínea c)

1.3.1 Método e Resultados

Neste exercício foi então necessário reunir um conjunto de resultados de simulação e teóricos. Os valores de simulação já foram calculados na alínea anterior e são portanto aqui reutilizados. É preciso então calcular, para cada valor de λ, os valores teóricos de Atraso Médio de Pacotes e *Throughput* para os modelos de fila M/M/1 e M/G/1.

No caso do modelo M/M/1, temos que o atraso médio é dado por:

$$PacketDelay = \frac{1}{\mu - \lambda}$$

O μ corresponde a taxa a que a fila é capaz de servir os clientes, e é calculada por:

$$\mu = \frac{C}{AveragePacketSize}$$

O $AveragePacketSize$ para esta flow pode ser calculado a partir do sumatório de cada tamanho de pacote P_i a multiplicar pela sua probabilidade p_i

Com este valor podemos também calcular o $Throughput$ tanto para a fila M/M/1 e M/G/1, com a fórmula:

$$TT = 1e-6 * (\lambda * AveragePacketSize\_XXX);$$

Por fim é necessário calcular o valor de $PacketDelay$ para o modelo M/G/1. Tendo calculado o tempos de atendimento S_i, onde i corresponde ao tamanho do pacote que a fila vai servir é possível então calcular o $E[S]$ e $E[S^2]$. O atraso médio da fila de espera pode então ser calculado com a fórmula de Pollaczek-Khintchine. Na implementação, os valores de $PacketDelay$ são multiplicados por 1000 de forma a os converter a milissegundos.

Posto isto, foi necessário calcular os valores teóricos com cada valor de lambda dos parâmetros da alínea b).

Os resultados destes modelos teóricos serão comparados com os resultados vindos do simulador e podem então ser consultados na Figura 1.2

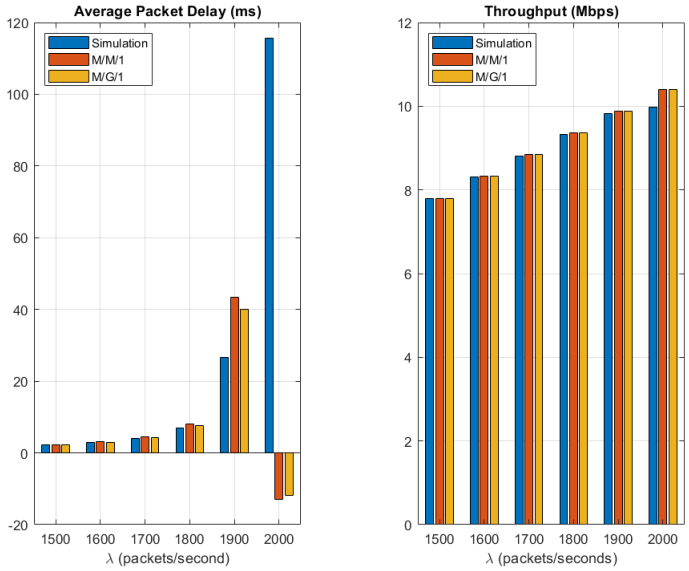

Figura 1.2: Resultado da alínea c)

1.3.2 Código

```
%% 3c)
% Parameters
lambda_values = [1500, 1600, 1700, 1800, 1900, 2000];
    %packet rate
C = 10; %conection capacity
f = 10000000; %queue size
% Calculate Theorical values for M/M/1

% 16% for 64 bytes, 25% for 110 bytes, 20% for 1518
    bytes,
% 39& for the rest

aux= [65:109 111:1517];

% B = sum(prob * S)
AveragePacketSize_MM1 = ((0.16 * 64) + (0.25 * 110) +
    (0.2 * 1518) + ...
     ((sum(aux)*0.39)/length(aux)))*8;

% mu = C / B
u = (C * 1000000)/AveragePacketSize_MM1 ;   % Mbps

PacketDelay_MM1 = zeros(1, sz);
```

```
Throughput_MM1 = zeros(1, sz);
for i=1:sz
    % packet delay = 1/mu - lambda
    PacketDelay_MM1(i) = 1 / (u - lambda_values(i)) *
        1000;

    %Throughput = 10-6 * TRANSMITTEDBYTES * 8
    Throughput_MM1(i) = 1e-6 * (lambda_values(i)*
        AveragePacketSize_MM1);
end

% Calculate Theorical values for M/G/1

S = C * 1000000;

AveragePacketSize_MG1 = ((0.16 * 64) + (0.25 * 110) +
    (0.2 * 1518) + ...
    ((sum(aux)*0.39)/length(aux)))*8;

ES = ((0.16 * 64) + (0.25 * 110) + (0.2 * 1518) + ...
    (0.39/length(aux)) *sum(aux)) * 8 / S;
ES2 = ((0.16 * (64*8/S)^2) + (0.25 * (110*8/S)^2) +
    (0.2 * (1518*8/S)^2) ...
    + (0.39/length(aux))*(sum((aux*8/S).^2)));

PacketDelay_MG1 = zeros(1, sz);
Throughput_MG1 = zeros(1, sz);
for i=1:sz
    % W = lambda*E[S^2]/2(1-lambdaE[s]) + E[s]
    PacketDelay_MG1(i) = ((lambda_values(i) * ES2)/(2
        * (1 - (lambda_values(i) * ES)))+ES)*1000;

    %Throughput = 10-6 * TRANSMITTEDBYTES * 8
    Throughput_MG1(i) = 1e-6 * (lambda_values(i)*
        AveragePacketSize_MG1);
end

figure(3);

tiledlayout(1,2)

nexttile;
bar(lambda_values, [medias_APD_b(:) PacketDelay_MM1(:)
    PacketDelay_MG1(:)])
legend("Simulation", "M/M/1", "M/G/1", "Location" ,"
    northwest")
title("Average Packet Delay (ms)")
xlabel('\lambda (packets/second)')
grid on

```

```
63 nexttile;
64 bar(lambda_values, [medias_TT_b(:) Throughput_MM1(:)
       Throughput_MG1(:)])
65 legend("Simulation", "M/M/1", "M/G/1", "Location" ,"
       northwest")
66 title("Throughput (Mbps)")
67 xlabel('\lambda (packets/seconds)')
68 grid on
```

1.3.3 Discussão

É possível observar nos resultados desta alínea que para taxas λ mais baixas, os resultados teóricos tanto do modelo M/M/1 e do M/G/1 aproximam-se dos resultados da simulação, tanto para o Atraso Médio de Pacotes como para o *Throughput*.

Porém, à medida que se aumenta a taxa e que esta se aproxima da taxa de recepção μ dos modelos (que é aproximadamente 1923), os resultados começam a ser bastante dispares, quando comparados com o da simulação.

No caso da taxa de recepção μ dos modelos teóricos ser ultrapassada, os cálculos do Atraso Médio de Pacotes devolvem valores negativos e os do *Throughput* devolvem valores superiores a 10. Como a fila não pode ter um throughput superior à sua capacidade e os atrasos não podem ser negativos, estes resultados não tem qualquer significado.

Com isto é possível então concluir que estes modelos teóricos não são válidos para prever a performance do sistema.

1.4 Alínea d)

1.4.1 Método e Resultados

Para esta alínea foi então executado o simulador com os parâmetros $P = 10000$, $\lambda = 1800$, $C = 10$, $f = [2500, 5000, 7500, 10000, 12500, 15000, 17500, 20000]$ e $b = 0$. Para obter os resultados, o simulador foi executado 40 vezes.

Os resultados podem ser consultados na Figura 1.3.

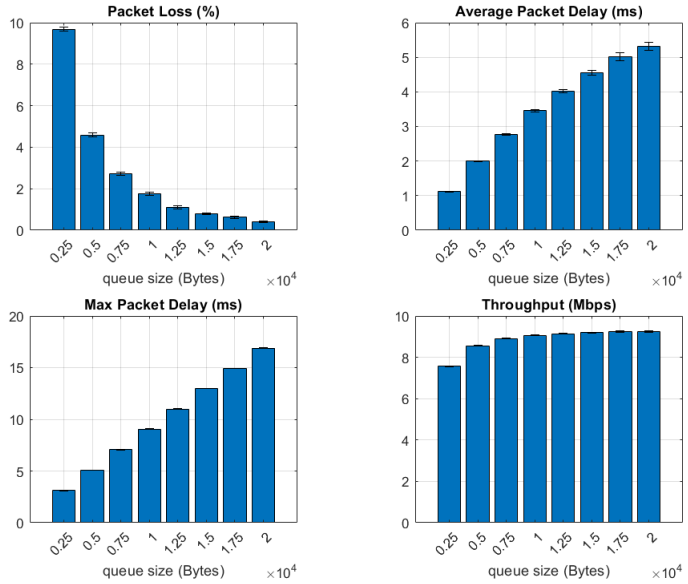

Figura 1.3: Resultado da alínea d)

1.4.2 Código

```
%% 3d)

% Parameters
n_runs = 40;
P = 10000; %stoping criterion
alpha = 0.10; %confidence intervals
lambda = 1800; %packet rate
C = 10; %conection capacity
f_values = [2500, 5000, 7500, 10000, 12500, 15000,
    17500, 20000]; %queue size
b = 0; %bit error rate
```

```
11
12 % Results arrays
13 sz = length(f_values);
14 medias_PL = zeros(1, sz);
15 temp_PL = zeros(1, sz);
16 medias_APD = zeros(1, sz);
17 temp_APD = zeros(1, sz);
18 medias_MDP = zeros(1, sz);
19 temp_MDP = zeros(1, sz);
20 medias_TT = zeros(1, sz);
21 temp_TT = zeros(1, sz);
22
23 % Run simulator n times for each value of lambda
24 for i = 1:sz
25     [medias_PL(i), temp_PL(i) , medias_APD(i),
          temp_APD(i), ...
26      medias_MDP(i), temp_MDP(i) , medias_TT(i),
           temp_TT(i)] = ...
27     runSimulator2(n_runs, alpha, lambda, C, f_values(i
          ), P,b);
28 end
29
30 figure(4)
31 tiledlayout(2,2)
32 % Packet Loss
33 nexttile;
34 bar(f_values,medias_PL)
35 title("Packet Loss (%)")
36 xlabel('queue size (Bytes)')
37 ylim([0 100])
38 grid on
39
40 hold on
41
42 er = errorbar(f_values, medias_PL, temp_PL);
43 er.Color = [0 0 0];
44 er.LineStyle = 'none';
45
46 hold off
47
48 % Average Packet Delay
49 nexttile;
50 bar(f_values,medias_APD)
51 title("Average Packet Delay (ms)")
52 xlabel('queue size (Bytes)')
53 grid on
54
55 hold on
56
57 er = errorbar(f_values, medias_APD, temp_APD);
```

```
er.Color = [0 0 0];
er.LineStyle = 'none';

hold off

% Max Packet Delay
nexttile;
bar(f_values,medias_MDP)
title("Max Packet Delay (ms)")
xlabel('queue size (Bytes)')
grid on

hold on

er = errorbar(f_values, medias_MDP, temp_MDP);
er.Color = [0 0 0];
er.LineStyle = 'none';

hold off

% Throughput
nexttile;
bar(f_values,medias_TT)
title("Throughput (Mbps)")
xlabel('queue size (Bytes)')
grid on

hold on

er = errorbar(f_values, medias_TT, temp_TT);
er.Color = [0 0 0];
er.LineStyle = 'none';

hold off
```

1.4.3 Discussão

À medida que o comprimento da fila aumenta, é possível observar logo uma descida na percentagem de pacotes perdidos. A razão é trivial, se a fila é maior, mais dificilmente fica ocupada e portanto perde menos pacotes de chegada.

No caso dos atrasos máximo e médios estes são maiores pois, com o aumento do tamanho da fila e da taxa de transferência do sistema, o tempo que cada pacote se mantém lá depende do numero de pacotes á frente deste, com o aumento do numero de pacotes dentro da fila aumenta o tempo de espera de cada um.

1.5 Alínea e)

1.5.1 Método e Resultados

Para obter os resultados necessário foi preciso então fazer uma implementação do modelo teórico M/M/1/m. Para calcular o atraso médio de pacotes temos:

$$W = \frac{L}{\lambda \times (1 - \mu_m)}$$

O L, o número médio de pacotes no sistema, pode ser calculado a partir de:

$$L = \frac{\sum_{i=0}^{m} i \times (\lambda/\mu)^i}{\sum_{j=0}^{m} j(\lambda/\mu)^j}$$

O cálculo do μ já foi discutido em capítulos anteriores. Para o m, usamos a fórmula $m = round(((f_z * 8)/AveragePacketSize)) + 1$, onde f_z corresponde aos diferentes valores de tamanho de fila considerados.

A perda de pacotes é então calculada com:

$$PacketLoss = \frac{(\lambda/\mu)^m}{\sum_{j=0}^{m} j(\lambda/\mu)^j}$$

Foi então necessário fazer estes cálculos para todos os valores de tamanho de fila, o que na nossa implementação é representado por um loop for. O denominador e o numerador da expressão do número médio de pacotes foi calculada separadamente, de forma a permitir reutilizar os resultados para a expressão do Packet Loss.

Já o Throughput pode ser calculado subtraindo a percentagem de pacotes perdidos aos valor de dados transmitidos.

Por fim, juntando os resultados do simulador, executado com os parâmetros descritos na Secção 1.4, aos resultados teóricos, obtemos então os resultados da Figura 1.4

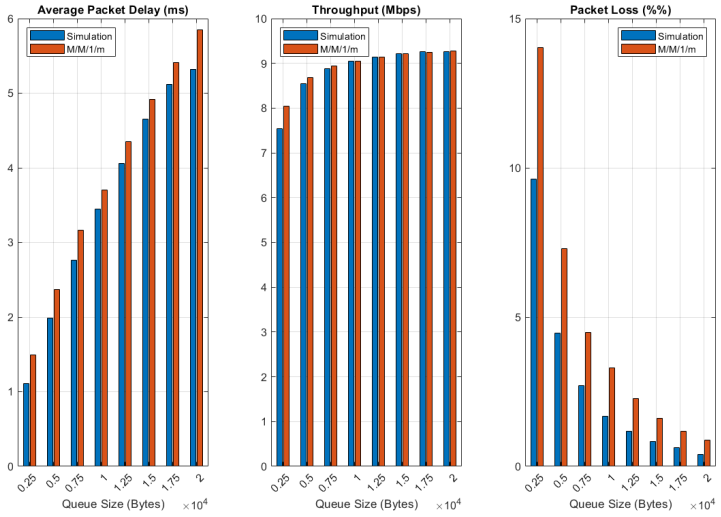

Figura 1.4: Resultado da alínea e)

1.5.2 Código

```
%% 3e)

% Parameters
lambda = 1800; %packet rate
C = 10 * 1000000; %conection capacity
f_values = [2500, 5000, 7500, 10000, 12500, 15000,
    17500, 20000]; %queue size

% Calculate Theorical Values for M/M/1/m
% 16% for 64 bytes, 25% for 110 bytes, 20% for 1518
    bytes,
% 39& for the rest

aux= [65:109 111:1517];

% B = sum(prob * S)
AveragePacketSize = ((0.16 * 64) + (0.25 * 110) + (0.2
    * 1518) + ...
```

```
    ((sum(aux)*0.39)/length(aux)))*8;

PacketLoss_MM1m = zeros(1, length(f_values));
PacketDelay_MM1m = zeros(1, length(f_values));
Throughput_MM1m = zeros(1, length(f_values));

for z=1:length(f_values)
    m = round(((f_values(z)*8)/AveragePacketSize))+1;

    % mu = C / B
    u = C/AveragePacketSize;    % Mbps
    % (lamda/mu)^m/sum((lamda/mu)^j)
    den = 0;
    for j = 0:m
        den = den + (lambda/u)^j;
    end
    PacketLoss_MM1m(z) = (((lambda/u)^(m))/den)*100;

    num_aux = 0;

    for i = 0:m
        num_aux = num_aux + (i * ((lambda/u)^i));
    end

    L = num_aux/den;

    PacketDelay_MM1m(z) = L/(lambda*(1 - (
       PacketLoss_MM1m(z)/100)))*1000;

    %Throughput = 10-6 * TRANSMITTEDBYTES * 8
    Throughput_MM1m(z) = (1e-6 * ((lambda*
       AveragePacketSize)-(lambda*AveragePacketSize*(
       PacketLoss_MM1m(z)/100))));
end

figure(5);

tiledlayout(1,3)

nexttile;
bar(f_values, [medias_APD(:) PacketDelay_MM1m(:)])
legend("Simulation", "M/M/1/m", "Location" ,"northwest
   ")
title("Average Packet Delay (ms)")
xlabel('Queue Size (Bytes)')
grid on

nexttile;
bar(f_values, [medias_TT(:) Throughput_MM1m(:)])
```

```
61 legend("Simulation", "M/M/1/m", "Location" ,"northwest
       ")
62 title("Throughput (Mbps)")
63 xlabel('Queue Size (Bytes)')
64 grid on
65
66 nexttile;
67 bar(f_values, [medias_PL(:) PacketLoss_MM1m(:)])
68 legend("Simulation", "M/M/1/m", "Location" ,"northeast
       ")
69 title("Packet Loss (%%)")
70 xlabel('Queue Size (Bytes)')
71 grid on
```

1.5.3 Discussão

Observando a Figura 1.4, é observável que os resultados da simulação são semelhantes aos do modelo teórico M/M/1/m pelo menos nos casos do atraso médio de pacotes e da taxa de transferência. Já no caso da perda de pacotes, existe uma maior discrepância entre os resultados, dado que os do modelo são mais ou menos o dobro para todos os casos excepto o primeiro. Sabendo isto, é possível afirmar que o modelo M/M/1/m é válido para prever a performance do sistema simulado.

1.6 Alínea f)

1.6.1 Método e Resultados

Nesta alínea foi executado o simulador com os parâmetros da alínea d), descritos na Secção 1.4, com a diferença que o $b = 10^{-5}$

Os resultados podem ser consultados na Figura 1.5.

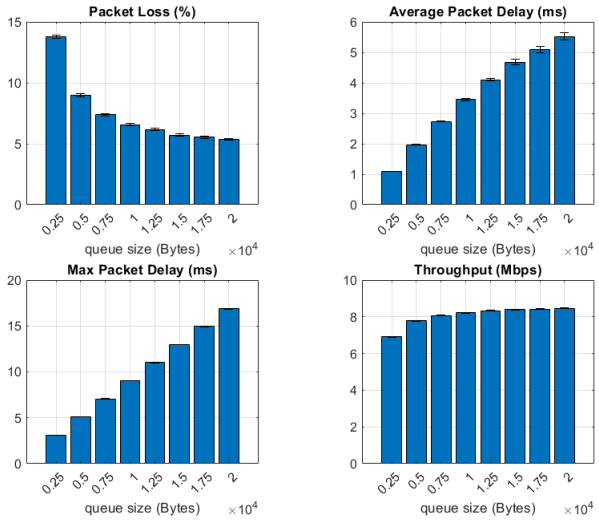

Figura 1.5: Resultado da alínea f)

1.6.2 Código

```
%% 3f)

% Parameters
n_runs = 40;
P = 10000; %stoping criterion
alpha = 0.10; %confidence intervals
lambda = 1800; %packet rate
C = 10; %conection capacity
f_values = [2500, 5000, 7500, 10000, 12500, 15000,
    17500, 20000]; %queue size
b = 1e-5; %bit error rate

% Results arrays
sz = length(f_values);
medias_PL = zeros(1, sz);
```

```
temp_PL = zeros(1, sz);
medias_APD = zeros(1, sz);
temp_APD = zeros(1, sz);
medias_MDP = zeros(1, sz);
temp_MDP = zeros(1, sz);
medias_TT = zeros(1, sz);
temp_TT = zeros(1, sz);

% Run simulator n times for each value of lambda
for i = 1:sz
    [medias_PL(i), temp_PL(i) , medias_APD(i),
       temp_APD(i), ...
     medias_MDP(i), temp_MDP(i) , medias_TT(i),
        temp_TT(i)] = ...
    runSimulator2(n_runs, alpha, lambda, C, f_values(i
       ), P,b);
end

figure(6)
tiledlayout(2,2)
% Packet Loss
nexttile;
bar(f_values,medias_PL)
title("Packet Loss (%)")
xlabel('queue size (Bytes)')
ylim([0 100])
grid on

hold on

er = errorbar(f_values, medias_PL, temp_PL);
er.Color = [0 0 0];
er.LineStyle = 'none';

hold off

% Average Packet Delay
nexttile;
bar(f_values,medias_APD)
title("Average Packet Delay (ms)")
xlabel('queue size (Bytes)')
grid on

hold on

er = errorbar(f_values, medias_APD, temp_APD);
er.Color = [0 0 0];
er.LineStyle = 'none';

hold off
```

```
62
63 % Max Packet Delay
64 nexttile;
65 bar(f_values,medias_MDP)
66 title("Max Packet Delay (ms)")
67 xlabel('queue size (Bytes)')
68 grid on
69
70 hold on
71
72 er = errorbar(f_values, medias_MDP, temp_MDP);
73 er.Color = [0 0 0];
74 er.LineStyle = 'none';
75
76 hold off
77
78 % Throughput
79 nexttile;
80 bar(f_values,medias_TT)
81 title("Throughput (Mbps)")
82 xlabel('queue size (Bytes)')
83 grid on
84
85 hold on
86
87 er = errorbar(f_values, medias_TT, temp_TT);
88 er.Color = [0 0 0];
89 er.LineStyle = 'none';
90
91 hold off
```

1.6.3 Discussão

Se compararmos os resultados da Figura 1.5 com os resultados da alínea d), na Figura 1.3, as únicas diferenças são relacionadas com a perda de pacotes e o Throughput, como seria de esperar. Ao ter um *bit error rate* diferente de 0, é possível que sejam perdidos pacotes já após a sua transmissão o que faz com que os valores de perda pacotes aumentem para todos os tamanhos de fila. O Throughput também é afetado ligeiramente, pois ao serem perdidos mais pacotes, poderá haver um decréscimo na quantidade de bytes transmitidos por segundo,

1.7 Alínea g)

1.7.1 Método e Resultados

Nesta alínea foi executado o simulador com os parâmetros da alínea b), descritos na Secção 1.2, com a diferença que o $b = 10^{-5}$

Os resultados podem ser consultados na Figura 1.6.

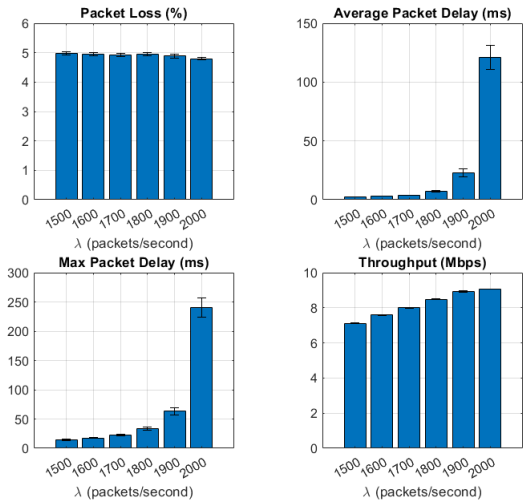

Figura 1.6: Resultado da alínea g)

1.7.2 Código

```
%% 3g)

% Parameters
n_runs = 40;
P = 10000; %stoping criterion
alpha = 0.10; %confidence intervals
lambda_values = [1500, 1600, 1700, 1800, 1900, 2000];
    %packet rate
C = 10; %conection capacity
f = 10000000; %queue size
b = 1e-5; %bit error rate

% Results arrays
sz = length(lambda_values);
medias_PL = zeros(1, sz);
```

```
temp_PL = zeros(1, sz);
medias_APD = zeros(1, sz);
temp_APD = zeros(1, sz);
medias_MDP = zeros(1, sz);
temp_MDP = zeros(1, sz);
medias_TT = zeros(1, sz);
temp_TT = zeros(1, sz);

% Run simulator n times for each value of lambda
for i = 1:sz
    [medias_PL(i), temp_PL(i) , medias_APD(i),
       temp_APD(i), ...
     medias_MDP(i), temp_MDP(i) , medias_TT(i),
        temp_TT(i)] = ...
    runSimulator2(n_runs, alpha, lambda_values(i), C,
       f, P,b);
end

figure(7)
tiledlayout(2,2)
% Packet Loss
nexttile;
bar(lambda_values,medias_PL)
title("Packet Loss (%)")
xlabel('\lambda (packets/second)')
ylim([0 100])
grid on

hold on

er = errorbar(lambda_values, medias_PL, temp_PL);
er.Color = [0 0 0];
er.LineStyle = 'none';

hold off

% Average Packet Delay
nexttile;
bar(lambda_values,medias_APD)
title("Average Packet Delay (ms)")
xlabel('\lambda (packets/second)')
grid on

hold on

er = errorbar(lambda_values, medias_APD, temp_APD);
er.Color = [0 0 0];
er.LineStyle = 'none';

hold off
```

```
62
63 % Max Packet Delay
64 nexttile;
65 bar(lambda_values,medias_MDP)
66 title("Max Packet Delay (ms)")
67 xlabel('\lambda (packets/second)')
68 grid on
69
70 hold on
71
72 er = errorbar(lambda_values, medias_MDP, temp_MDP);
73 er.Color = [0 0 0];
74 er.LineStyle = 'none';
75
76 hold off
77
78 % Throughput
79 nexttile;
80 bar(lambda_values,medias_TT)
81 title("Throughput (Mbps)")
82 xlabel('\lambda (packets/second)')
83 grid on
84
85 hold on
86
87 er = errorbar(lambda_values, medias_TT, temp_TT);
88 er.Color = [0 0 0];
89 er.LineStyle = 'none';
90
91 hold off
```

1.7.3 Discussão

As diferenças entre os resultados da alínea b), na Figura 1.1, e os da alínea g) prendem-se na perda de pacotes e no Throughput. Esta diferença faz sentido pois neste exercício o *bit error rate* é diferente de 0. O Throughput diminui ligeiramente, dado que ao perderem-se mais pacotes, diminui-se a quantidade bytes transmitidos por segundo da fila.

Outra observação que se pode retirar destes resultados é que a mudança da taxa de pacotes não afecta os valores de perde de pacotes, sendo que estes valores se mantém próximos para qualquer um dos valores de λ.

1.8 Alínea h)

1.8.1 Método e Resultados

Nesta alínea foi necessário calcular os resultados teóricos para um modelos de filas M/G/1 com ber. Para tal foram seguidos os passos presentes no Apêndice A do Guião fornecido. Na nossa implementação os valores de P_i para os tamanhos de pacotes de 64, 100 e 1518 bytes são calculados à cabeça, tal como os valores de $E[S]$ e $E[S^2]$. De seguida é feito um for loop, de forma a calcular os valores de Throughput e de atraso médio de pacotes para cada valor de lambda considerado. Os valores de P_i para os restante tamanho de pacotes são calculados em nested loops, quando necessários para os cálculos.

Os resultados obtidos do modelo teórico, em conjunto com os resultados do simulador da alínea g), podem ser vistos então na Figura 1.7.

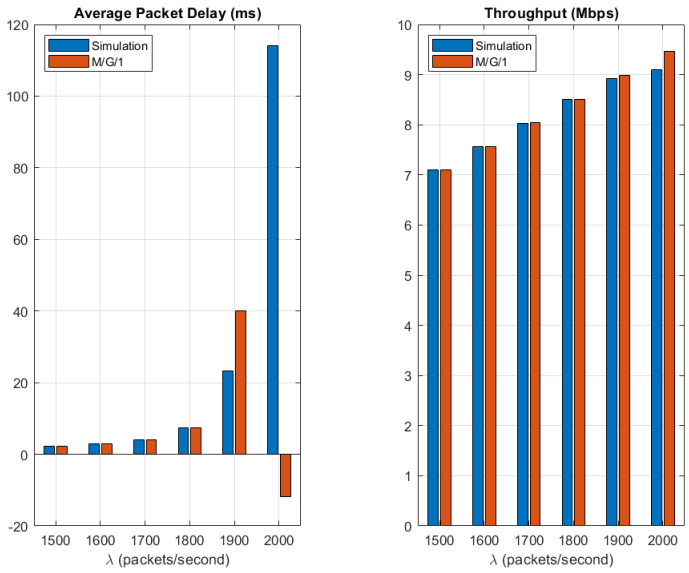

Figura 1.7: Resultado da alínea h)

1.8.2 Código

```
%% 3h)

% Calculating M/G/1 theoretical values with b
% Parameters
```

```
lambda_values = [1500, 1600, 1700, 1800, 1900, 2000]; %packet rate
C = 10; %conection capacity
b = 10^-5; %bit error rate

% sum(pi * (1 - Pi))
Pi_64 = (1 - b)^(8*64);
Pi_110 = (1 - b)^(8*110);
Pi_1518 = (1 - b)^(8*1518);

% Delay
S = C * 1000000;

ES = ((0.16 * 64) + (0.25 * 110) + (0.2 * 1518) + ...
    (0.39/length(aux)) *sum(aux)) * 8 / S;
ES2 = ((0.16 * (64*8/S)^2) + (0.25 * (110*8/S)^2) + (0.2 * (1518*8/S)^2) ...
    + (0.39/length(aux))*(sum((aux*8/S).^2)));

PacketDelay_MG1 = zeros(1, length(lambda_values));
Throughput_MG1 = zeros(1, length(lambda_values));

for z=1:length(lambda_values)
    % WQ = lambda*E[S^2]/2(1-lambdaE[s]) + E[s]
    WQ = ((lambda_values(z) * ES2)/(2 * (1 - (lambda_values(z) * ES))))*1000;

    Wi_64 = WQ + ((8*64)/S)*1000;
    Wi_110 = WQ + ((8*110)/S)*1000;
    Wi_1518 = WQ + ((8*1518)/S)*1000;

    APD_num = (0.16 * Pi_64 * Wi_64) + (0.25 * Pi_110 * Wi_110) ...
        + (0.2 * Pi_1518 * Wi_1518);

    APD_den = (0.16 * Pi_64) + (0.25 * Pi_110) + (0.2 * Pi_1518);

    APD_num_aux = 0;
    APD_den_aux = 0;
    for i=1:length(aux)
        Pi_aux = (1 - b)^(8*aux(i));
        Wi_aux = WQ + ((8*aux(i))/S)*1000;
        APD_num_aux = APD_num_aux + ((0.39/length(aux)) * Pi_aux * Wi_aux);
        APD_den_aux = APD_den_aux + ((0.39/length(aux)) * Pi_aux);
    end

```

```
47      PacketDelay_MG1(z) = (APD_num + APD_num_aux)/(
            APD_den + APD_den_aux);
48
49      % Throughput
50
51      Throughput = (0.16 * Pi_64 * lambda_values(z) * (8
            * 64)) ...
52          + (0.25 * Pi_110 * lambda_values(z) * (8 *
                110)) ...
53          + (0.2 * Pi_1518 * lambda_values(z) * (8 *
                1518));
54
55      Throughput_aux = 0;
56
57      for i=1:length(aux)
58          Pi_aux = (1 - b)^(8*aux(i));
59          Throughput_aux = Throughput_aux + ...
60              ((0.39/length(aux)) * Pi_aux *
                    lambda_values(z) * (8 * aux(i)));
61      end
62
63      Throughput_MG1(z) = (Throughput + Throughput_aux)
            * 1e-6;
64
65  end
66
67  figure(8);
68
69  tiledlayout(1,2)
70
71  nexttile;
72  bar(lambda_values, [medias_APD(:) PacketDelay_MG1(:)])
73  legend("Simulation", "M/G/1", "Location" ,"northwest")
74  title("Average Packet Delay (ms)")
75  xlabel('\lambda (packets/second)')
76  grid on
77
78  nexttile;
79  bar(lambda_values, [medias_TT(:) Throughput_MG1(:)])
80  legend("Simulation", "M/G/1", "Location" ,"northwest")
81  title("Throughput (Mbps)")
82  xlabel('\lambda (packets/second)')
83  grid on
```

1.8.3 Discussão

À semelhança do se pode observar na Discussão 1.3.3, à medida que se aumenta a taxa e que esta se aproxima da taxa de recepção μ dos modelos (que é aproximadamente 1923), os resultados começam a ser bastante dispares, quando comparados com o da simulação. E tal como na secção referida anteriormente, quando a taxa de recepção ultrapassa a dos modelos, os resultados do Throughput ficam acima da capacidade do sistema e os de atraso médio de pacotes ficam negativos, pelo que podemos concluir que o modelo M/G/1 não pode ser utilizado para prever a performance da sistema simulado.

Capítulo 2

Tarefa 4

Neste capítulo iremos expor os resultados das várias alíneas da Tarefa 4, tal como o raciocínio para os alcançar. Será também apresentado o código desenvolvido e, quando considerado necessário, terá uma reflexão sobre os resultados obtidos.

2.1 Simulador

2.1.1 Método

A fim de realizar os exercícios propostas na Tarefa 4, foi necessário proceder ao desenvolvimento de um novo Simulador. Neste simulador, ao contrário do que foi usado anteriormente, o fluxo de pacotes é modelado por uma cadeia de Markov de 3 estados, cada um correspondente a uma taxa de pacotes diferentes, 0.5λ, λ e 2λ.

Foi então necessário adicionar um evento novo, o TRANSITION, que sinaliza então uma transição de estado, e uma varíavel de estado, FLOWSTATE, que indica qual é o estado atual da cadeia, tomando valores de 1 a 3.

Com o objetivo de implementar a cadeira de Markov, foi calculado à cabeça as probabilidades limite de cada estado e o tempo médio de permanência (linhas 38 a 54 do Código 2.1.2). Os primeiro valores irão servir para decidir qual o valor inicial de FLOWSTATE e os últimos para marcar as próximas transições.

De seguida foi também calculado os três possíveis valores de λ e colocados num array, de forma a podermos aceder ao valor desejado usando FLOWSTATE como índice (linhas 58).

Foram criadas duas funções auxiliares para este simulador, a Calculate-

FirstState() (linhas 159 a 168 do Código 2.1.2 e CalculateNextState() (linhas 145 a 157).

A primeira função, tal como o nome indica, calcula o estado inicial da cadeia, a partir das probabilidades limite calculadas anteriormente.

Já a segunda é utilizada para calcular o valor seguinte de FLOWSTATE durante a execução do simulador. Como nas cadeias de Markov há sempre uma transição de estado, no caso da cadeia estar no estado 1 ou 3, transita sempre para o estado 2, e no caso de estar no estado 2, tem uma igual probabilidade de passar para o estado 1 ou 3.

No loop principal do simulador a principal mudança foi a adição de um novo case, que representa a transição (linhas 112 a 114), onde primeiro é calculado o próximo valor de FLOWSTATE, com uma chamada à função CalculateNextState() e é agendada uma nova transição.

Foi também modificada a maneira como se agendam as próximas chegadas de pacotes, pois agora o valor de lambda depende do estado da cadeia em que o simulador se encontra (linhas 65 e 79).

Para confirmar a correcção do simulador, comparamos os nossos resultados com aqueles os resultados exemplo das alíneas a) e b) desta Tarefa.

Para a alínea a), um exemplo dos resultados obtidos é:

$$PacketLoss(\%) = 7.98e-03 + -1.31e-02$$

$$Av.PacketDelay(ms) = 1.17e+02 + -1.79e+01$$

$$Max.PacketDelay(ms) = 5.09e+02 + -7.21e+01$$

$$Throughput(Mbps) = 9.34e+00 + -9.97e-02$$

Para a alínea b), um exemplo dos resultados obtidos é:

$$PacketLoss(\%) = 1.07e+01 + -1.53e-01$$

$$Av.PacketDelay(ms) = 4.27e+00 + -4.85e-02$$

$$Max.PacketDelay(ms) = 9.17e+00 + -1.11e-02$$

$$Throughput(Mbps) = 7.55e+00 + -4.56e-02$$

2.1.2 Código

```
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)

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

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

%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;

% Calculating the probability of each state
```

```
38 transitions = [10/5 5/10];
39
40 % calculo da probabilidade limite de processos de
       nascimento e morte
41
42 sum = transitions(1) + (transitions(2) * transitions
       (1));
43
44 prob_1 = 1/(1+ sum);
45 prob_2 = prob_1*transitions(1);
46 prob_3 = prob_2*transitions(2);
47 probs = [prob_1 prob_2 prob_3];
48
49 % Calculo do tempo de perman ncia
50
51 time_1 = (1 / 10);
52 time_2 = (1 / (5 + 5));
53 time_3 = (1 / 10);
54 times = [time_1 time_2 time_3];
55
56 % Calculos dos valores de lambda por estado
57
58 lambda_values = [0.5*lambda lambda 1.5*lambda];
59
60 % Initialize first Transition
61 FLOWSTATE = CalculateFirstState(probs);
62 EventList = [TRANSITION, Clock + exprnd(times(
       FLOWSTATE)), 0, 0];
63
64 % Initializing the List of Events with the first
       ARRIVAL:
65 EventList = [EventList; ARRIVAL, Clock + exprnd(1/
       lambda_values(FLOWSTATE)), GeneratePacketSize(),
       0];
66
67 %Similation loop:
68 while TRANSMITTEDPACKETS<P                % Stopping
       criterium
69     EventList= sortrows(EventList,2);    % Order
          EventList by time
70     Event= EventList(1,1);               % Get first
          event and
71     Clock= EventList(1,2);               %   and
72     PacketSize= EventList(1,3);          %
          associated
73     ArrivalInstant= EventList(1,4);      %
          parameters.
74     EventList(1,:)= [];                  % Eliminate
          first event
```

```
75      ProbWithoutErros = CalculateProbabilityOfBER(
            PacketSize,b);
76      switch Event
77          case ARRIVAL                        % If first
                event is an ARRIVAL
78              TOTALPACKETS= TOTALPACKETS+1;
79              EventList = [EventList; ARRIVAL, Clock +
                    exprnd(1/lambda_values(FLOWSTATE)),
                    GeneratePacketSize(), 0];
80              if STATE==0
81                  STATE= 1;
82                  EventList = [EventList; DEPARTURE,
                        Clock + 8*PacketSize/(C*10^6),
                        PacketSize, Clock];
83              else
84                  if QUEUEOCCUPATION + PacketSize <= f
85                      QUEUE= [QUEUE;PacketSize , Clock];
86                      QUEUEOCCUPATION= QUEUEOCCUPATION +
                            PacketSize;
87                  else
88                      LOSTPACKETS= LOSTPACKETS + 1;
89                  end
90              end
91          case DEPARTURE                      % If first
                event is a DEPARTURE
92              % Verify if has erros; LOSTPACKETS=
                    LOSTPACKETS + 1;
93              if rand() > ProbWithoutErros
94                  LOSTPACKETS= LOSTPACKETS + 1;
95              else
96                  TRANSMITTEDBYTES= TRANSMITTEDBYTES +
                        PacketSize;
97                  DELAYS= DELAYS + (Clock -
                        ArrivalInstant);
98
99                  if Clock - ArrivalInstant > MAXDELAY
100                     MAXDELAY= Clock - ArrivalInstant;
101                 end
102                 TRANSMITTEDPACKETS= TRANSMITTEDPACKETS
                        + 1;
103             end
104
105             if QUEUEOCCUPATION > 0
106                 EventList = [EventList; DEPARTURE,
                        Clock + 8*QUEUE(1,1)/(C*10^6),
                        QUEUE(1,1), QUEUE(1,2)];
107                 QUEUEOCCUPATION= QUEUEOCCUPATION -
                        QUEUE(1,1);
108                 QUEUE(1,:)= [];
109             else
```

```
110                 STATE= 0;
111             end
112         case TRANSITION
113             FLOWSTATE = CalculateNextState(FLOWSTATE);
114             EventList = [EventList; TRANSITION, Clock
                    + exprnd(times(FLOWSTATE)), 0, 0];
115     end
116 end
117
118 %Performance parameters determination:
119 PL= 100*LOSTPACKETS/TOTALPACKETS;       % in %
120 APD= 1000*DELAYS/TRANSMITTEDPACKETS;    % in
        milliseconds
121 MPD= 1000*MAXDELAY;                     % in
        milliseconds
122 TT= 10^(-6)*TRANSMITTEDBYTES*8/Clock;  % in Mbps
123
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
139
140 function result= CalculateProbabilityOfBER(packetSize,
        ber)
141     n = packetSize * 8;
142     result = (1-ber)^n;
143 end
144
145 function newState= CalculateNextState(currState)
146     next_state_prob = rand();
147
148     if currState == 1 || currState == 3
149         newState = 2;
150     elseif currState == 2
151         if next_state_prob < 0.5
152             newState = 1;
153         else
154             newState = 3;
155         end
```

```
156     end
157 end
158 
159 function newState= CalculateFirstState(probs)
160     next_state_prob = rand();
161     if next_state_prob <= probs(1)
162         newState = 1;
163     elseif next_state_prob >= 1 - probs(3)
164         newState = 3;
165     else
166         newState = 2;
167     end
168 end
```

2.2 Alínea c)

2.2.1 Método e Resultados

Para esta alínea foi então executado os simuladores 2 e 3 com os parâmetros $P = 100000$, $\lambda = [1500, 1600, 1700, 1800, 1900, 2000]$, $C = 10$, $f = 1e7$ e $b = 0$. Para obter os resultados, os simuladores foram executados 10 vezes.

Os resultados podem ser vistos na Figura 2.1.

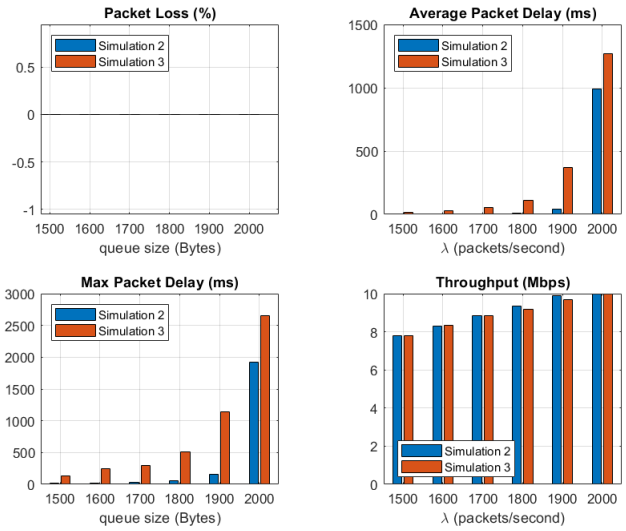

Figura 2.1: Resultado da alínea c)

2.2.2 Código

```
%% 4c)
% Parameters
n_runs = 10;
P = 100000; %stoping criterion
alpha = 0.10; %confidence intervals
lambda_values = [1500, 1600, 1700, 1800, 1900, 2000];
    %packet rate
C = 10; %conection capacity
f = 1e7; %queue size
b = 0; %bit error rate

sz = length(lambda_values);

```

```
% Results arrays Sim2
medias_PL_S2 = zeros(1, sz);
temp_PL_S2 = zeros(1, sz);
medias_APD_S2 = zeros(1, sz);
temp_APD_S2 = zeros(1, sz);
medias_MDP_S2 = zeros(1, sz);
temp_MDP_S2 = zeros(1, sz);
medias_TT_S2 = zeros(1, sz);
temp_TT_S2 = zeros(1, sz);

% Results arrays Sim3
medias_PL_S3 = zeros(1, sz);
temp_PL_S3 = zeros(1, sz);
medias_APD_S3 = zeros(1, sz);
temp_APD_S3 = zeros(1, sz);
medias_MDP_S3 = zeros(1, sz);
temp_MDP_S3 = zeros(1, sz);
medias_TT_S3 = zeros(1, sz);
temp_TT_S3 = zeros(1, sz);

% Run simulator n times for each value of lambda
for i = 1:sz
    [medias_PL_S2(i), temp_PL_S2(i) , medias_APD_S2(i), temp_APD_S2(i), ...
     medias_MDP_S2(i), temp_MDP_S2(i) , medias_TT_S2(i), temp_TT_S2(i)] = ...
    runSimulator2(n_runs, alpha, lambda_values(i), C, f, P,b);

    [medias_PL_S3(i), temp_PL_S3(i) , medias_APD_S3(i), temp_APD_S3(i), ...
     medias_MDP_S3(i), temp_MDP_S3(i) , medias_TT_S3(i), temp_TT_S3(i)] = ...
    runSimulator3(n_runs, alpha, lambda_values(i), C, f, P,b);
end

figure(1);

tiledlayout(2,2)

% Packet Loss
nexttile;
bar(lambda_values, [medias_PL_S2(:) medias_PL_S3(:)])
legend("Simulation 2", "Simulation 3", "Location" ,"northwest")
title("Packet Loss (%)")
xlabel('queue size (Bytes)')
grid on

```

```
56 nexttile;
57 bar(lambda_values, [medias_APD_S2(:) medias_APD_S3(:)
       ])
58 legend("Simulation 2", "Simulation 3", "Location" ,"
       northwest")
59 title("Average Packet Delay (ms)")
60 xlabel('\lambda (packets/second)')
61 grid on
62
63 % Max Packet Delay
64 nexttile;
65 bar(lambda_values, [medias_MDP_S2(:) medias_MDP_S3(:)
       ])
66 legend("Simulation 2", "Simulation 3", "Location" ,"
       northwest")
67 title("Max Packet Delay (ms)")
68 xlabel('queue size (Bytes)')
69 grid on
70
71 nexttile;
72 bar(lambda_values, [medias_TT_S2(:) medias_TT_S3(:)])
73 legend("Simulation 2", "Simulation 3", "Location" ,"
       northwest")
74 title("Throughput (Mbps)")
75 xlabel('\lambda (packets/second)')
76 grid on
```

2.2.3 Discussão

Ao comparar a Simulação 3 com a Simulação 2, é possível observar que os atrasos médios de pacotes do Simulador 3 são superiores as do Simulador 2 para todas as taxas de pacotes consideradas. Isto deve-se a maneira como o fluxo de pacotes é modelado no Simulador 3, que permite que existam momentos na simulação em que o sistema está a funcionar a 2λ, isto é, o dobro do valor de taxa definido, o que aumenta o número de pacotes na fila e aumenta o tempo que o sistema demora a tratar deles.

O Throughput do Simulador 3 oscila no sentido em que em alguns casos é inferior ao do Simulador 2 e noutros é superior. Estes resultados seriam mais uniformes se o simulador fosse executado um maior número de vezes, e com isto seria possível observar que o Simulador 3 geralmente deve ter um Throughput mais baixo, pois os atrasos maiores diminuem a capacidade de transmissão por segundo.

2.3 Alínea d)

2.3.1 Método e Resultados

Para esta alínea foi então executado o simulador com os parâmetros $P = 100000$, $\lambda = 1800$, $C = 10$, $f = [2500, 5000, 7500, 10000, 12500, 15000, 17500, 20000]$ e $b = 0$. Para obter os resultados, o simulador foi executado 10 vezes.

Os resultados podem ser vistos na Figura 2.2.

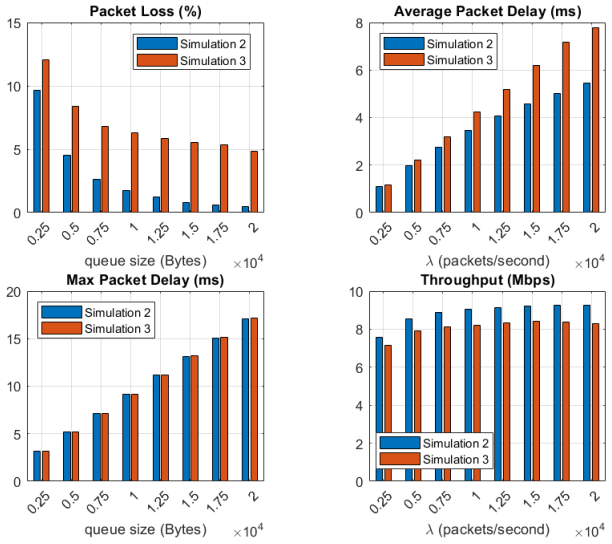

Figura 2.2: Resultado da alínea d)

2.3.2 Código

```
%% 4d)
% Parameters
n_runs = 10;
P = 100000; %stoping criterion
alpha = 0.10; %confidence intervals
lambda = 1800; %packet rate
C = 10; %conection capacity
f_values = [2500, 5000, 7500, 10000, 12500, 15000,
    17500, 20000]; %queue size
b = 0; %bit error rate

sz = length(f_values);

```

```
% Results arrays Sim2
medias_PL_S2 = zeros(1, sz);
temp_PL_S2 = zeros(1, sz);
medias_APD_S2 = zeros(1, sz);
temp_APD_S2 = zeros(1, sz);
medias_MDP_S2 = zeros(1, sz);
temp_MDP_S2 = zeros(1, sz);
medias_TT_S2 = zeros(1, sz);
temp_TT_S2 = zeros(1, sz);

% Results arrays Sim3
medias_PL_S3 = zeros(1, sz);
temp_PL_S3 = zeros(1, sz);
medias_APD_S3 = zeros(1, sz);
temp_APD_S3 = zeros(1, sz);
medias_MDP_S3 = zeros(1, sz);
temp_MDP_S3 = zeros(1, sz);
medias_TT_S3 = zeros(1, sz);
temp_TT_S3 = zeros(1, sz);

% Run simulator n times for each value of lambda
for i = 1:sz
    [medias_PL_S2(i), temp_PL_S2(i) , medias_APD_S2(i), temp_APD_S2(i), ...
     medias_MDP_S2(i), temp_MDP_S2(i) , medias_TT_S2(i), temp_TT_S2(i)] = ...
    runSimulator2(n_runs, alpha, lambda, C, f_values(i), P,b);

    [medias_PL_S3(i), temp_PL_S3(i) , medias_APD_S3(i), temp_APD_S3(i), ...
     medias_MDP_S3(i), temp_MDP_S3(i) , medias_TT_S3(i), temp_TT_S3(i)] = ...
    runSimulator3(n_runs, alpha, lambda, C, f_values(i), P,b);
end

figure(2);

tiledlayout(2,2)

% Packet Loss
nexttile;
bar(f_values, [medias_PL_S2(:) medias_PL_S3(:)])
legend("Simulation 2", "Simulation 3", "Location" ,"northwest")
title("Packet Loss (%)")
xlabel('queue size (Bytes)')
grid on

```

```
nexttile;
bar(f_values, [medias_APD_S2(:) medias_APD_S3(:)])
legend("Simulation 2", "Simulation 3", "Location" ,"
    northwest")
title("Average Packet Delay (ms)")
xlabel('\lambda (packets/second)')
grid on

% Max Packet Delay
nexttile;
bar(f_values, [medias_MDP_S2(:) medias_MDP_S3(:)])
legend("Simulation 2", "Simulation 3", "Location" ,"
    northwest")
title("Max Packet Delay (ms)")
xlabel('queue size (Bytes)')
grid on

nexttile;
bar(f_values, [medias_TT_S2(:) medias_TT_S3(:)])
legend("Simulation 2", "Simulation 3", "Location" ,"
    northwest")
title("Throughput (Mbps)")
xlabel('\lambda (packets/second)')
grid on
```

2.3.3 Discussão

Os resultados para o atraso de pacote máximo são semelhantes tanto para ambos os simuladores. No entanto, o atraso médio de pacotes é substancialmente maior no simulador 3, o que significa que este atinge valores mais altos de atraso com mais frequência. Isto deve-se à maneira como é definido o fluxo de pacotes, que permite o sistema funciona a duas vezes o valor de λ definido.

Dado que as filas tem tamanhos baixos, existe uma percentagem de perda de pacotes em ambas as simulações, porém a da Simulação 3 é consistentemente mais alta para todos os tamanhos considerados. Isto ocorre porque a fila é preenchida mais rapidamente na Simulação 3, pois existe a possibilidade do sistema funciona duas vezes o valor de λ definido, e chegam mais pacotes nas alturas em que a fila está preenchida, que consequentemente são considerados como perdidos. Este valores poderiam ser maiores se não houvesse um estado que deixa o sistema funcionar a metade do valor de lambda definido.

2.4 Alínea e)

2.4.1 Método e Resultados

Nesta alínea foi executado o simulador com os parâmetros da alínea c), descritos na Secção 2.2, com a diferença que o $b = 10^{-5}$.

Os resultados podem ser vistos na Figura 2.3.

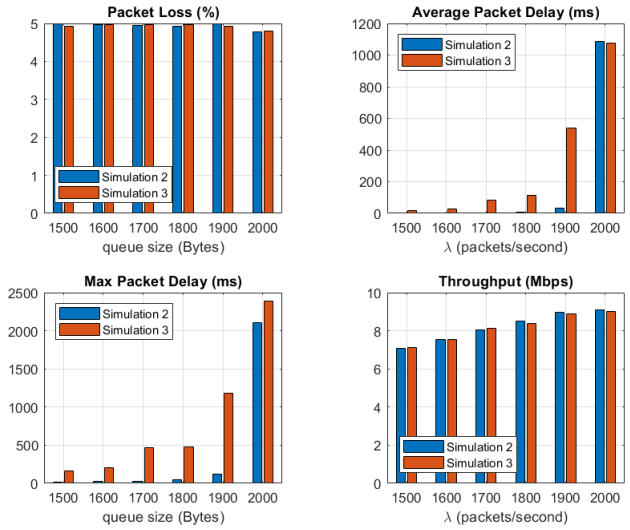

Figura 2.3: Resultado da alínea e)

2.4.2 Código

```
%% 4e)
% Parameters
n_runs = 10;
P = 100000; %stoping criterion
alpha = 0.10; %confidence intervals
lambda_values = [1500, 1600, 1700, 1800, 1900, 2000];
    %packet rate
C = 10; %conection capacity
f = 1e7; %queue size
b = 1e-5; %bit error rate

sz = length(lambda_values);

% Results arrays Sim2
```

```
14 medias_PL_S2 = zeros(1, sz);
15 temp_PL_S2 = zeros(1, sz);
16 medias_APD_S2 = zeros(1, sz);
17 temp_APD_S2 = zeros(1, sz);
18 medias_MDP_S2 = zeros(1, sz);
19 temp_MDP_S2 = zeros(1, sz);
20 medias_TT_S2 = zeros(1, sz);
21 temp_TT_S2 = zeros(1, sz);
22
23 % Results arrays Sim3
24 medias_PL_S3 = zeros(1, sz);
25 temp_PL_S3 = zeros(1, sz);
26 medias_APD_S3 = zeros(1, sz);
27 temp_APD_S3 = zeros(1, sz);
28 medias_MDP_S3 = zeros(1, sz);
29 temp_MDP_S3 = zeros(1, sz);
30 medias_TT_S3 = zeros(1, sz);
31 temp_TT_S3 = zeros(1, sz);
32
33 % Run simulator n times for each value of lambda
34 for i = 1:sz
35     [medias_PL_S2(i), temp_PL_S2(i) , medias_APD_S2(i)
          , temp_APD_S2(i), ...
36      medias_MDP_S2(i), temp_MDP_S2(i) , medias_TT_S2(i
          ), temp_TT_S2(i)] = ...
37     runSimulator2(n_runs, alpha, lambda_values(i), C,
          f, P,b);
38
39     [medias_PL_S3(i), temp_PL_S3(i) , medias_APD_S3(i)
          , temp_APD_S3(i), ...
40      medias_MDP_S3(i), temp_MDP_S3(i) , medias_TT_S3(i
          ), temp_TT_S3(i)] = ...
41     runSimulator3(n_runs, alpha, lambda_values(i), C,
          f, P,b);
42 end
43
44 figure(3);
45
46 tiledlayout(2,2)
47
48 % Packet Loss
49 nexttile;
50 bar(lambda_values, [medias_PL_S2(:) medias_PL_S3(:)])
51 legend("Simulation 2", "Simulation 3", "Location" ,"
       northwest")
52 title("Packet Loss (%)")
53 xlabel('queue size (Bytes)')
54 grid on
55
56 nexttile;
```

```
57 bar(lambda_values, [medias_APD_S2(:) medias_APD_S3(:)
       ])
58 legend("Simulation 2", "Simulation 3", "Location" ,"
       northwest")
59 title("Average Packet Delay (ms)")
60 xlabel('\lambda (packets/second)')
61 grid on
62
63 % Max Packet Delay
64 nexttile;
65 bar(lambda_values, [medias_MDP_S2(:) medias_MDP_S3(:)
       ])
66 legend("Simulation 2", "Simulation 3", "Location" ,"
       northwest")
67 title("Max Packet Delay (ms)")
68 xlabel('queue size (Bytes)')
69 grid on
70
71 nexttile;
72 bar(lambda_values, [medias_TT_S2(:) medias_TT_S3(:)])
73 legend("Simulation 2", "Simulation 3", "Location" ,"
       northwest")
74 title("Throughput (Mbps)")
75 xlabel('\lambda (packets/second)')
76 grid on
```

2.4.3 Discussão

Comparando os resultados da alínea c) com os da alínea e) reparamos que a única diferença se encontra na percentagem de pacotes perdidos ser maior que zero. Ambos os simuladores produzem os mesmos resultados pois o descarte de pacotes ocorre da mesma maneira em ambos os simuladores, dado que o *bit error rate* é o mesmo e o tamanho da fila não afeta a perda de pacotes neste exercício.

2.5 Alínea f)

2.5.1 Método e Resultados

Nesta alínea foi executado o simulador com os parâmetros da alínea d), descritos na Secção 2.3, com a diferença que o $b = 10^{-5}$.

Os resultados podem ser vistos na Figura 2.4.

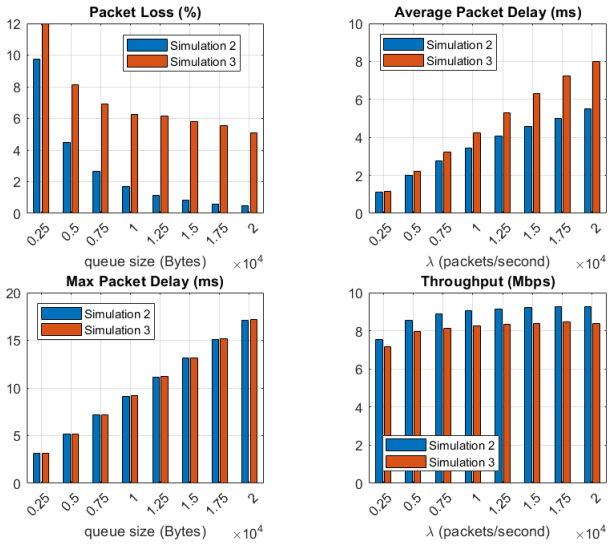

Figura 2.4: Resultado da alínea f)

2.5.2 Código

```
%% 4f)
% Parameters
n_runs = 10;
P = 100000; %stoping criterion
alpha = 0.10; %confidence intervals
lambda = 1800; %packet rate
C = 10; %conection capacity
f_values = [2500, 5000, 7500, 10000, 12500, 15000,
    17500, 20000]; %queue size
b = 0; %bit error rate

sz = length(f_values);

% Results arrays Sim2
```

```
14 medias_PL_S2 = zeros(1, sz);
15 temp_PL_S2 = zeros(1, sz);
16 medias_APD_S2 = zeros(1, sz);
17 temp_APD_S2 = zeros(1, sz);
18 medias_MDP_S2 = zeros(1, sz);
19 temp_MDP_S2 = zeros(1, sz);
20 medias_TT_S2 = zeros(1, sz);
21 temp_TT_S2 = zeros(1, sz);
22
23 % Results arrays Sim3
24 medias_PL_S3 = zeros(1, sz);
25 temp_PL_S3 = zeros(1, sz);
26 medias_APD_S3 = zeros(1, sz);
27 temp_APD_S3 = zeros(1, sz);
28 medias_MDP_S3 = zeros(1, sz);
29 temp_MDP_S3 = zeros(1, sz);
30 medias_TT_S3 = zeros(1, sz);
31 temp_TT_S3 = zeros(1, sz);
32
33 % Run simulator n times for each value of lambda
34 for i = 1:sz
35     [medias_PL_S2(i), temp_PL_S2(i) , medias_APD_S2(i)
          , temp_APD_S2(i), ...
36      medias_MDP_S2(i), temp_MDP_S2(i) , medias_TT_S2(i
          ), temp_TT_S2(i)] = ...
37     runSimulator2(n_runs, alpha, lambda, C, f_values(i
          ), P,b);
38
39     [medias_PL_S3(i), temp_PL_S3(i) , medias_APD_S3(i)
          , temp_APD_S3(i), ...
40      medias_MDP_S3(i), temp_MDP_S3(i) , medias_TT_S3(i
          ), temp_TT_S3(i)] = ...
41     runSimulator3(n_runs, alpha, lambda, C, f_values(i
          ), P,b);
42 end
43
44 figure(4);
45
46 tiledlayout(2,2)
47
48 % Packet Loss
49 nexttile;
50 bar(f_values, [medias_PL_S2(:) medias_PL_S3(:)])
51 legend("Simulation 2", "Simulation 3", "Location" ,"
       northwest")
52 title("Packet Loss (%)")
53 xlabel('queue size (Bytes)')
54 grid on
55
56 nexttile;
```

```
57 bar(f_values, [medias_APD_S2(:) medias_APD_S3(:)])
58 legend("Simulation 2", "Simulation 3", "Location" ,"
      northwest")
59 title("Average Packet Delay (ms)")
60 xlabel('\lambda (packets/second)')
61 grid on
62
63 % Max Packet Delay
64 nexttile;
65 bar(f_values, [medias_MDP_S2(:) medias_MDP_S3(:)])
66 legend("Simulation 2", "Simulation 3", "Location" ,"
      northwest")
67 title("Max Packet Delay (ms)")
68 xlabel('queue size (Bytes)')
69 grid on
70
71 nexttile;
72 bar(f_values, [medias_TT_S2(:) medias_TT_S3(:)])
73 legend("Simulation 2", "Simulation 3", "Location" ,"
      northwest")
74 title("Throughput (Mbps)")
75 xlabel('\lambda (packets/second)')
76 grid on
```

2.5.3 Discussão

Comparando os resultados da alínea d) com os da alínea f) reparamos que a única diferença se encontra na percentagem de pacotes perdidos ser maior na alínea f). Isto seria de esperar porque o *bit error rate* é diferente de zero, o que aumenta o numero de pacotes perdidos.

Neste caso a perda de pacotes é superior para a Simulação 3 pois, como já foi referido anteriormente, o tamanho das filas reduzido afecta mais este simulador, devido à forma como é modelado o fluxo de pacotes, que permite ter uma taxa duas vezes maior à definida.