Universidade de Aveiro

Modelação e Desempenho de Redes e Serviços
Mini-Projeto 1

João Gameiro (93097), Pedro Abreu (93240)

Departamento de Electrónica, Telecomunicações e Informática

12 de janeiro de 2022

Conteúdo

Lista de Figuras

Capítulo 1

Introdução

O presente relatório visa descrever as decisões e conclusões tiradas durante a resolução do Mini-Projecto 1 desenvolvido no âmbito da unidade curricular de Modelação e Desempenho de Redes e Serviços. Este documento está dividido em dois capítulos sendo que num deles são descritas as decisões, implementação e conclusões tiradas da tarefa 1. O segundo capítulo diz respeito à tarefa 2.

É importante referir que no código desenvolvido apresentado neste relatório encontram-se comentários que ajudam numa melhor compreensão do mesmo.

Capítulo 2

Tarefa 1

2.1 Alínea a)

2.1.1 Abordagem Seguida

O simulador 1 é um simulador de eventos discretos, cujo seu principal objectivo é estimar a performance de uma ligação IP ponto a ponto entre um router de uma empresa e o seu ISP. Este simulador apenas considera o tráfego que vai na direcção do ISP para empresa. Os parâmetros de entrada do simulador incluem a taxa de chegada de pacotes, a capacidade da ligação, o tamanho da fila de espera e o número total de pacotes transmitidos numa simulação. Os seus contadores estatísticos que permitem analisar e tirar conclusões sobre o comportamento do link são a percentagem de perda de pacotes de dados, o atraso médio de pacotes de dados, o atraso máximo de um pacote de dados e o throughput transmitido.

No caso desta alínea, um dos parâmetros de performance que se pretende avaliar é o atraso médio dos pacotes nesta ligação para diferentes valores da taxa de chegada de pacotes de dados. Assim sendo para cada um dos valores da taxa de chegada de pacotes, foi corrido o simulador 50 vezes, e posteriormente foram recolhidos os parâmetros de performance resultantes da sua execução.

2.1.2 Código

```
1  P = 10000;                          %Stopping criterion
2  l = [400, 800, 1200, 1600, 2000]; %Values for packet rate in pps
3  C = 10;                             %Connection capacity in Mbps
4  f = 1000000;                        %Queue size in Bytes
5  %Vectors to store the average delays obtained from the simulations
6  %and their respective errors
7  delays = zeros(1, length(l));
8  errors = zeros(1, length(l));
9
10 %Number of Simulations
11 N = 50;
```

```
12
13 for j = 1:length(l)
14     %Vectors to store the performance parameters
15     PL = zeros(1, N);
16     APD = zeros(1, N);
17     MPD = zeros(1, N);
18     TT = zeros(1, N);
19     %Run the simulator and gather the performance parameters
20     for i = 1:N
21         [PL(i), APD(i), MPD(i), TT(i)] = Simulator1(l(j),C,f,P);
22     end
23     alfa = 0.1; %Confidence interval of 90%
24     %Calculation of the media and it's confidence interval
25     media = mean(APD);
26     term = norminv(1-alfa/2)*sqrt(var(APD)/N);
27     %Store the obtained values in the vectors
28     delays(j) = media;
29     errors(j) = term;
30 end
31
32 %Plot results
33 figure(1)
34 bar(l, delays)
35 xlabel("Packet Rate (pps)")
36 ylabel("Average Packet Delay (ms)")
37 hold on
38 %Adding error bars to the plot
39 er = errorbar(l, delays, errors, errors);
40 er.Color = [0 0 0];
41 er.LineStyle = 'none';
42 hold off
```

2.1.3 Conclusões

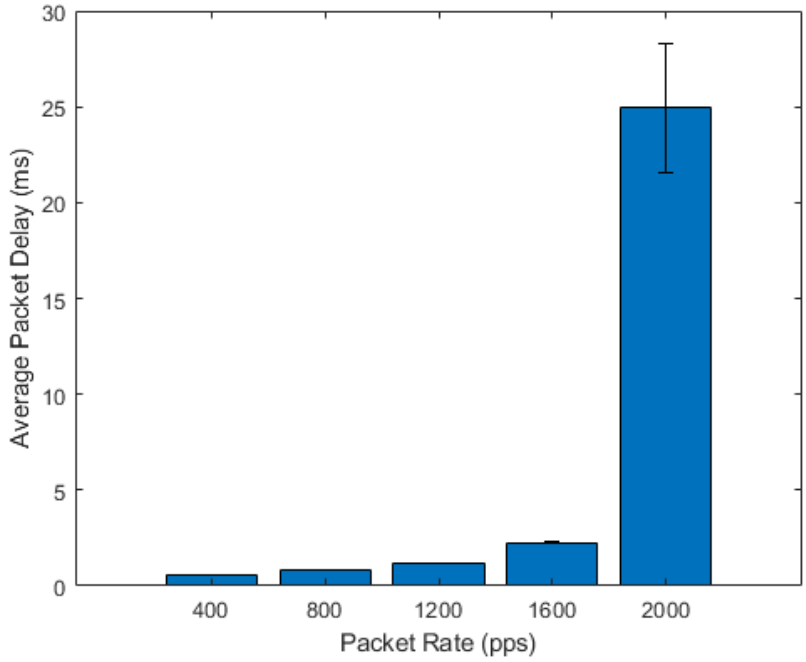

Figura 2.1: Resultado da execução da alínea 1.a).

É de esperar que para uma capacidade do link e tamanho de queue fixas, o atraso médio dos pacotes aumente à medida que a taxa de chegada dos mesmos aumenta. Podemos confirmar essa conclusão através da observação do gráfico 2.1. Assumindo também que para este simulador o tamanho médio de pacote em Bytes é 620,02 Bytes (calculado na Alínea c)) podemos calcular a taxa de chegada de pacotes em Bytes para analisar melhor os resultados obtidos.

Taxa de chegada de pacotes em pps	Taxa de chegada de pacotes em Bytes
400	248 008
800	496 016
1 200	744 024
1 600	992 032
2 000	1 240 040

Tabela 2.1: Taxa de chegada de pacotes em pps e Bytes

Considerando que a capacidade do link é de 10 Megabits podemos calcular a mesma em Bytes por segundo, obtendo assim 1250000 Bytes por segundo. Analisando os resultados obtidos na tabela reparamos que para uma taxa de chegada de 2 000 pacotes por segundo o

valor em bytes é bastante próximo da capacidade do link em Bytes por segundo. Este factor poderá justificar o atraso muito maior que se verifica no caso deste ultimo valor para a taxa de chegada de pacotes, pois o facto da capacidade ser tão próxima da taxa de chegada pode significar uma maior dificuldade no atendimento de pacotes (que estão à espera na fila), o que por sua vez vai aumentar o seu atraso médio. Os outros valores não têm atrasos tão grandes pelo facto da taxa de chegada ficar atrás do valor da capacidade.

2.2 Alínea b)

Para este caso foi usado o simulador da alínea anterior mas com o intuito de de medir não só o atraso médio dos pacotes, mas também a percentagem de perda dos mesmos. Deste modo o simulador foi corrido novamente 50 vezes, mas em vez de ser variada a taxa de chegada de pacotes, foram feitas as simulações para quatro valores diferentes do tamanho da fila de espera. No final da simulação foi recolhido não só atraso médio dos pacotes mas também a percentagem de perda dos mesmos.

2.2.1 Código

```
P = 10000; %Stopping criterion
l = 1800;  %Packet rate (pps)
C = 10;    %Link capacity (Mbps)
%Values for the queue size in bytes
f = [100000, 20000, 10000, 2000];
%Vectors to store the performance parameters
delays = zeros(1, length(f));
errorsd = zeros(1, length(f));
loss = zeros(1, length(f));
errorsl = zeros(1, length(f));

%Run the 50 simulations
N = 50;
for j = 1:length(f)

    PL = zeros(1, N);
    APD = zeros(1, N);
    MPD = zeros(1, N);
    TT = zeros(1, N);
    %Run and gather the simulation results
    for i = 1:N
        [PL(i), APD(i), MPD(i), TT(i)] = Simulator1(l,C,f(j),P);
    end
    alfa = 0.1; %Confidence interval of 90%

    %Calculation of the average Packet Loss and it's confidence interval
    media = mean(PL);
    term = norminv(1-alfa/2)*sqrt(var(PL)/N);
    fprintf("Packet Loss of data (%%)       = %.2e +- %.2e\n", media, term);
    loss(j) = media;
    errorsl(j) = term;
```

```
32
33      %Calculation of the average Packet delay and it's confidence interval
34      media = mean(APD);
35      term = norminv(1-alfa/2)*sqrt(var(APD)/N);
36      fprintf("Av. Packet Delay (ms)          = %.2e +- %.2e\n", media, term);
37      delays(j) = media;
38      errorsd(j) = term;
39  end
40
41  %Plot average packet delay results and it's error bars
42  figure(1)
43  bar(categorical(f), delays)
44  xlabel("Queue Size (Bytes)")
45  ylabel("Average Packet Delay (ms)")
46  hold on
47  er = errorbar(categorical(f), delays, errorsd, errorsd);
48  er.Color = [0 0 0];
49  er.LineStyle = 'none';
50  hold off;
51
52  %Plot packet loss results and it's error barrs
53  figure(2)
54  bar(categorical(f), loss)
55  xlabel("Queue Size (Bytes)")
56  ylabel("Packet Loss (%)")
57  hold on
58  er = errorbar(categorical(f), loss, errorsl, errorsl);
59  er.Color = [0 0 0];
60  er.LineStyle = 'none';
61  hold off;
```

2.2.2 Conclusões

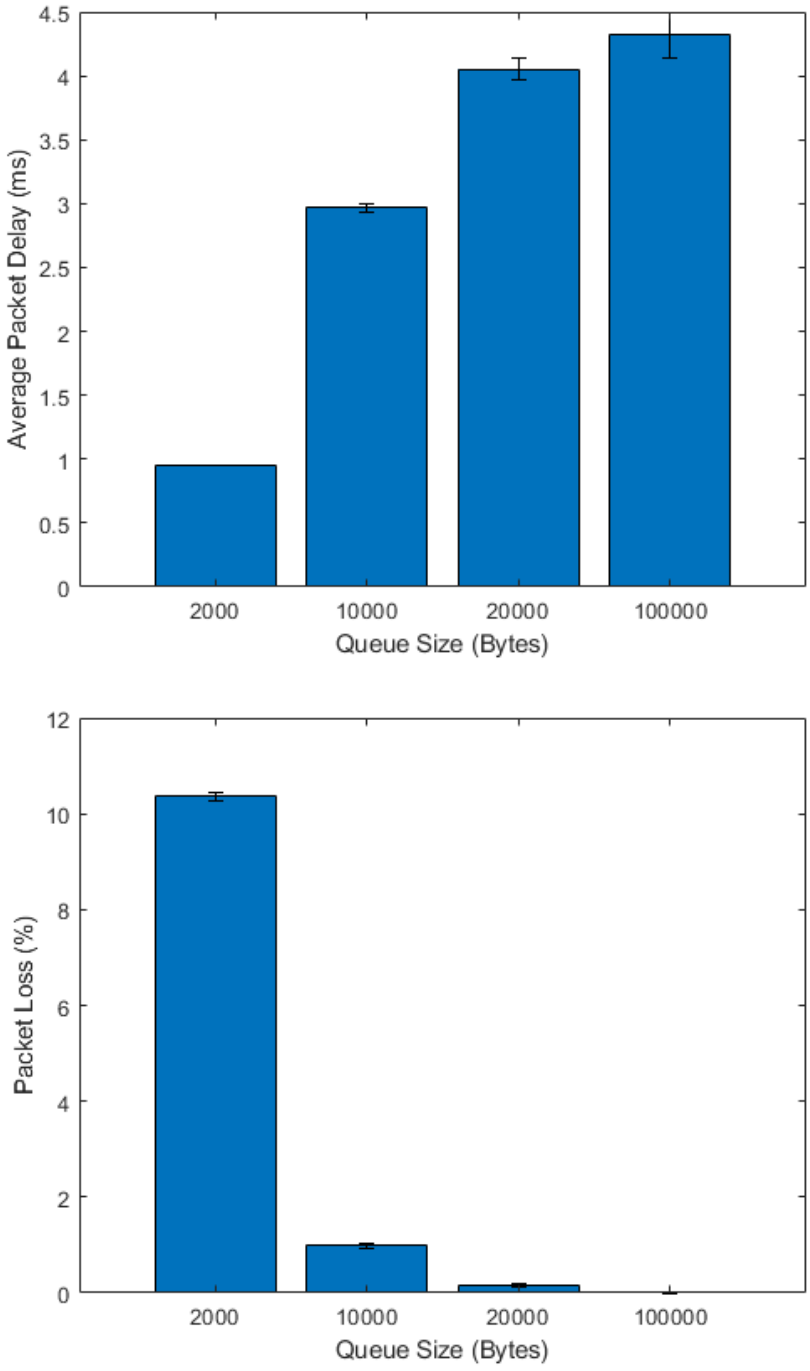

Figura 2.2: Resultado da execução da alínea 1.b).

Observando os resultados da simulação podemos concluir que quanto maior o tamanho da *queue*, maior atraso médio vão sofrer os pacotes pois o tempo de espera na fila aumenta proporcionalmente com o tamanho desta. Com um tamanho de *queue* mais pequeno a probabilidade de os pacotes serem descartados é maior pois esta ficará cheia mais rapidamente levando a uma maior percentagem de *Packet Loss* devido ao facto de não espaço para receber novos pacotes. Concluindo, o tamanho da fila de espera pressupõe um trade-off entre perda de pacotes e o atraso dos mesmos. Tamanhos muito elevados significam atrasos médios elevados, mas uma reduzida perda de pacotes. Ao invés, tamanhos mais pequenos de *queue* levam um número de descartado de pacotes mais elevado com atrasos médio mais baixos.

Estas conclusões foram tiradas tendo em conta que se está a considerar tamanhos diferentes para a fila de espera sem variar a capacidade do link.

2.3 Alínea c)

Nesta alínea foi novamente corrido o simulador 1 50 vezes, no entanto em vez de se variar o tamanho da fila de espera foi variada a capacidade do link. O parâmetro de performance recolhido foi o atraso médio dos pacotes.

2.3.1 Código

```
N = 50; % number of simulations

lambda = 1800;         % Packet rate pps
f = 1000000;           % Queue Size Bytes
C = [10, 20, 30, 40]; % Link capacity values in Mbps
P = 10000;             % packets

alfa = 0.1; %Confidence interval of 90%

%Vectors to store simulation results
APD = zeros(4, N);
media = zeros(4, 1);
term = zeros(4, 1);

%Run simulator and gather results
for i = [1:4]
    for n = [1:N]
        [¬, APD(i, n), ¬, ¬] = Simulator1(lambda, C(i), f, P);
    end
    %Calculation of average Packet delay and it's cofidence interval
    media(i, 1) = mean(APD(1,:));
    term(i, 1) = norminv(1-alfa/2) * sqrt(var(APD(1,:))/N);
end

%Plot results
bar(C, APD(:, 1))
hold on
er = errorbar(C, APD(:, 1), term(:, 1) * -1, term(:, 1));
```

```
er.Color = [0 0 0];
er.LineStyle = 'none';
hold off
title("Avg. Packet Delay");
xlabel("C, link capacity (Mbs)");
ylabel("APD, Avg. Packet Delay (ms)");
```

2.3.2 Conclusões

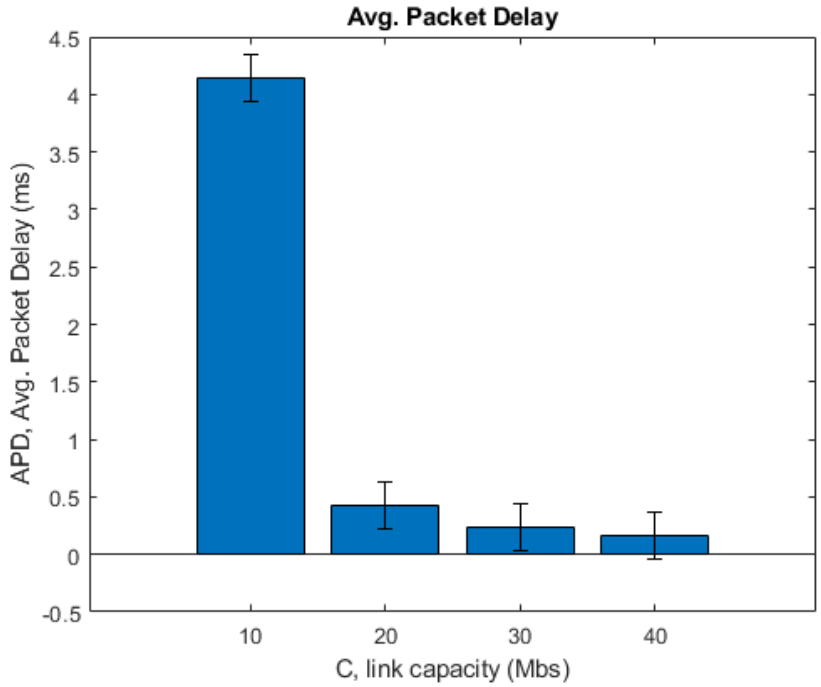

Figura 2.3: Resultado da execução da alínea 1.c).

Conclui-se que quanto maior a capacidade do link, menor o atraso médio dos pacotes. Isto acontece pois para uma capacidade maior relativamente a um *throughput* menor o link é capaz de enviar os pacotes à medida que eles chegam minimizando o atraso na fila de espera.

No entanto é observável que para $n = 10$ Mbs o atraso é consideravelmente maior que para os outros valores de n. Considerando os valores teóricos do exercício:

- Tamanho médio de pacote de 620.02 Bytes. (ver Alínea c))
- Taxa de chegada de pacotes é 1800 pps.
- $(620.02 * 8) * 1800 \approx 8.9$ Mb.

Concluímos que este fluxo de dados gera 8.9 Mb de tráfego por segundo, sendo que a diferença para 10 Mb é baixa, (apenas 1.1 Mb). Isto justifica o atraso considerável em relação às outras

experiências com capacidade de link superior. Para as capacidades 20, 30 e 40 Mb a diferença para o tráfego do fluxo aumenta, é respectivamente 11.1, 21.1, 31.1 Mb levando a um atraso médio de pacote progressivamente menor.

2.4 Alínea d)

Para este caso estamos a considerar que o sistema é modelado por um modelo de queueing M/G/1. Isto significa que se verificam as seguintes condições:

- Chegada de pacotes ao sistema é um processo de Poisson com taxa λ;
- Tempo de atendimento tem uma distribuição genérica e independente das chegadas dos clientes;
- O sistema tem um servidor e acomoda um número infinito de pacotes.

Para obter o atraso médio no sistema era necessário usar a seguinte expressão

$$W_q = \left(\frac{\lambda E[S^2]}{2(1 - \lambda E[S])} \right) + E[S]$$

Analisando a expressão de cima percebemos que já sabemos a taxa de chegada de pacotes (λ) que é de 1800 pacotes por segundo, logo falta apenas saber $E[S]$ e $E[S^2]$.

Para esse efeito calculamos então o tempo de atendimento para cada tamanho de pacote e multiplicamos pela sua respectiva probabilidade. Finalmente somámos tudo para obter a média pesada do tempo de atendimento de pacotes no sistema. Foi calculado também o valor de $E[S^2]$ que é igual ao $E[S]$ com a diferença de que o tempo de atendimento é elevado ao quadrado antes de multiplicado pela sua respectiva probabilidade. Finalmente foi apenas necessário substituir os valores na expressão.

2.4.1 Código

```
1  l = 1800;                %Packet rate
2  C = [10, 20, 30, 40];    %Link Capacity
3  f = 1000000;             %Queue
4
5  %Calculation of the probability associated to each packet size
6  % (other than the sizes 64, 110 and 1518 bytes)
7  numelems = (109 - 65 + 1) + (1517 - 111 + 1);
8  probrestante = 100 - (19 + 23 + 17);
9  probcadaelem = (probrestante / numelems);
10
11 %Calculation of E[S] and E[S^2]
12 bytes = 64:1518;
13 for j = 1:length(C)
14     S = (bytes .* 8)./(C(j)*10^6);
```

```
15      S2 = (bytes .* 8)./(C(j)*10^6);
16      for i = 1:length(bytes)
17          if i == 1
18              S(i) = S(i)*0.19;
19              S2(i) = S2(i)^2*0.19;
20          elseif i == 110-64+1
21              S(i) = S(i)*0.23;
22              S2(i) = S2(i)^2*0.23;
23          elseif i == 1518-64+1
24              S(i) = S(i)*0.17;
25              S2(i) = S2(i)^2*0.17;
26          else
27              S(i) = S(i)*(probcadaelem/100);
28              S2(i) = S2(i)^2*(probcadaelem/100);
29          end
30      end
31
32      ES = sum(S);
33      ES2 = sum(S2);
34
35      %Calculation of the average packet delay
36      W = ((l*ES2/(2*(1-l*(ES)))) + ES)*10^3;
37      fprintf("Link Capacity: %2d Mbps -> ", C(j));
38      fprintf("Average Packet Delay:  %.2e ms\n", W);
39  end
```

2.4.2 Conclusões

```
Link Capacity: 10 Mbps -> Average Packet Delay:  4.39e+00 ms
Link Capacity: 20 Mbps -> Average Packet Delay:  4.36e-01 ms
Link Capacity: 30 Mbps -> Average Packet Delay:  2.31e-01 ms
Link Capacity: 40 Mbps -> Average Packet Delay:  1.58e-01 ms
```

Figura 2.4: Resultado da execução da alínea 1.d).

Analisando os resultados obtidos no gráfico da alínea c) e comparando com os que foram obtidos neste exercício, podemos observar que os valores são bastante semelhantes. Logo podemos tirar a conclusão que o Simulador apresenta-nos uma boa aproximação teórica ao comportamento da performance do link.

2.5 Alínea e)

2.5.1 Código

O simulador pedido é similar ao Simulador1, onde é acrescentada a lógica necessária para calcular o atrasos em pacotes de tamanho 64, 110 e 1518 bytes. Acrescentaram-se 6 contadores estatísticos para medir o número de pacotes transmitidos associados a cada tamanho e os seus atrasos (3 contadores TRANSMITTEDPACKETS e 3 contadores DELAYS

para os 3 tamanhos de pacotes). Adicionou-se a lógica necessária para manter os contadores actualizados ao longo da simulação. No fim desta, foi necessário acrescentar e calcular os parâmetros de performance: APD(Average Packet Delay) para os três tipos de pacotes.

```
function [PL , APD , APD_s64, APD_s110, APD_s1518,  MPD , TT] = ...
    ex1e_Simulator1(lambda,C,f,P)

[...]

%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
TRANSMITTEDPACKETS_s64 = 0;
TRANSMITTEDPACKETS_s110 = 0;
TRANSMITTEDPACKETS_s1518 = 0;
TRANSMITTEDBYTES= 0;   % Sum of the Bytes of transmitted packets
DELAYS= 0;             % Sum of the delays of transmitted packets
DELAYS_s64 = 0;
DELAYS_s110 = 0;
DELAYS_s1518 = 0;
MAXDELAY= 0;           % Maximum delay among all transmitted packets

[...]

        case DEPARTURE                      % If first event is a DEPARTURE
            TRANSMITTEDBYTES= TRANSMITTEDBYTES + PacketSize;
            aux =  (Clock - ArrivalInstant);
            DELAYS= DELAYS + aux;
            switch PacketSize
                case 110
                    DELAYS_s110 = DELAYS_s110 + aux;
                    TRANSMITTEDPACKETS_s110 = TRANSMITTEDPACKETS_s110 + 1;
                case 64
                    DELAYS_s64 = DELAYS_s64 + aux;
                    TRANSMITTEDPACKETS_s64 = TRANSMITTEDPACKETS_s64 + 1;
                case 1518
                    DELAYS_s1518 = DELAYS_s1518 + aux;
                    TRANSMITTEDPACKETS_s1518 = TRANSMITTEDPACKETS_s1518 + 1;
                otherwise
            end

[...]

%Performance parameters determination:
PL= 100*LOSTPACKETS/TOTALPACKETS;      % in %
APD= 1000*DELAYS/TRANSMITTEDPACKETS;   % in milliseconds
APD_s64 = 1000*DELAYS_s64/TRANSMITTEDPACKETS_s64;
APD_s110 = 1000*DELAYS_s110/TRANSMITTEDPACKETS_s110;
APD_s1518 = 1000*DELAYS_s1518/TRANSMITTEDPACKETS_s1518;
MPD= 1000*MAXDELAY;                    % in milliseconds
TT= 10^(-6)*TRANSMITTEDBYTES*8/Clock;  % in Mbps

[...]
```

```
% ex 1.e
N = 50;          % Number of simulations
lambda = 1800;   % Packet Rate in pps
f = 1000000;     % Queue size in bytes
C = [10, 20, 30, 40];  % Link Capacity in Mbps
P = 10000;       % packets
%Confidence interval of 90%
alfa = 0.1;

%Vectors to store results
APD_s64 = zeros(4, N);
APD_s110 = zeros(4, N);
APD_s1518 = zeros(4, N);

media = zeros(4, 3);
term = zeros(4, 3);
%Run simulations and gather results
for i = [1:4]
    for n = [1:N]
        [¬, ¬, APD_s64(i, n), APD_s110(i, n), APD_s1518(i, n), ¬, ¬] = ...
            ex1e_Simulator1(lambda, C(i), f, P);
    end
    %Calculate average packet delays and their confidence interval
    media(i, 1) = mean(APD_s64(i,:));
    media(i, 2) =  mean(APD_s110(i,:));
    media(i, 3) =  mean(APD_s1518(i,:));
    term(i, 1) = norminv(1-alfa/2) * sqrt(var(APD_s64(i,:))/N);
    term(i, 2) = norminv(1-alfa/2) * sqrt(var(APD_s110(i,:))/N);
    term(i, 3) = norminv(1-alfa/2) * sqrt(var(APD_s1518(i,:))/N);
end
%Plot Results
C = categorical({'10','20','30','40'});
C = reordercats(C,{'10','20','30','40'});
b = bar(C, media, "grouped");
hold on
% Calculate the number of groups and number of bars in each group
[ngroups,nbars] = size(media);
% Get the x coordinate of the bars
x = nan(nbars, ngroups);
for i = 1:nbars
    x(i,:) = b(i).XEndPoints;
end
er = errorbar(x', media, term, 'k','linestyle','none');
hold off

title("Avg. Packet Delay");
xlabel("C, link capacity (Mbs)");
ylabel("Delay (ms)");
legend({'64 bytes', '110 bytes', '1518 bytes'});
```

2.5.2 Conclusões

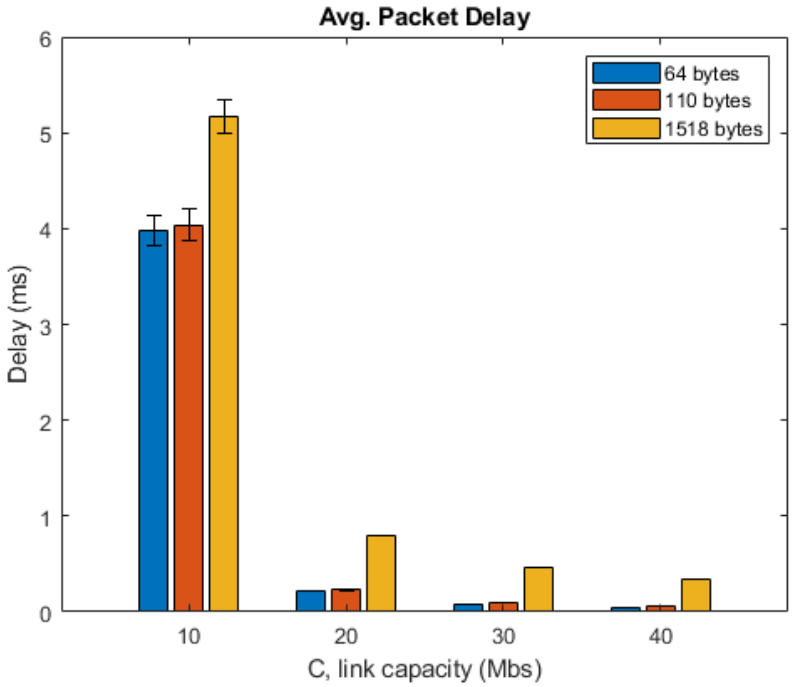

Figura 2.5: Resultado da execução da alínea 1.e).

Neste exercício considera-se os atraso médio para os tamanhos de pacotes mais comuns, 64, 110 e 1518 bytes, ou seja $19 + 23 + 15 = 59\%$ dos pacotes do fluxo considerados no exercício 1.c) são deste tipo. Assim, nas mesmas condições de experiência e de acordo com o concluído, para a capacidade de ligação $C = 10Mbps$, o link revela um atraso maior do que para as outras capacidades devido à considerável menor diferença entre o tráfego gerado e a capacidade do link. Também é possível observar que na mesma capacidade de ligação o atraso é maior quanto maior for o tamanho do pacote. Esta diferença é substancial nos pacotes de tamanho 64 e 110 bytes mas para o pacote de 1518 aumenta pelo menos $\frac{1518}{110} \approx 14$ vezes em relação aos outros dois. Isto justifica a diferença de atraso entre pacotes para mesma capacidade de queue.

No limite, para capacidades de ligação muito altas, p.e. 40, podemos afirmar que o atraso médio para um determinado tamanho de pacote se vai aproximar proporcionalmente do tempo de transmissão do pacote, por o atraso induzido na fila de espera ser mínimo.

Capítulo 3

Tarefa 2

3.1 Alínea a)

Para esta alínea era necessário correr o simulador 3 que é novamente um simulador de eventos estatísticos. É bastante parecido ao simulador 1 com a diferença que suporta um parâmetro a mais que representa o número de fluxos VoIP adicionais. Assim o simulador está pronto a funcionar com pacotes de dois tipos diferentes, VoIP e de dados, sendo que a política de escalonamento continua a ser FIFO (First-in-First-Out). É importante referir que para o bom funcionamento deste simulador todos os parâmetros de performance foram duplicados para suportar pacotes VoIP (uns parâmetros dizem respeito aos pacotes de dados outros aos VoIP).

Neste exercício foi então corrido o simulador 3 50 vezes para analisar o atraso médio dos pacotes de dados e de VoIP.

3.1.1 Código

```
%% a) Run Simulator 3 50 times with a stopping criterion of P = 10000
P = 10000;          %Stopping criterion
l = 1500;           %Packet rate
C = 10;             %Link capacity
f = 1000000;        %Queue size
n = [10, 20, 30, 40]; %number of VoIP flows
%Vectors to store results
delay = zeros(4, 2);
errors = zeros(4, 2);

N = 50;
for j = 1:length(n)
    PLd = zeros(1, N);
    PLv = zeros(1, N);
    APDd = zeros(1, N);
    APDv = zeros(1, N);
    MPDd = zeros(1, N);
```

```
18      MPDv = zeros(1, N);
19      TT = zeros(1, N);
20      %Run Simulation and gather results
21      for i = 1:N
22          [PLd(i), PLv(i), APDd(i), APDv(i), MPDd(i), MPDv(i), TT(i)] = ...
                Simulator3(l,C,f,P,n(j));
23      end
24      alfa = 0.1;
25      %Calculate data average delay and cofidence interval
26      media = mean(APDd);
27      term = norminv(1-alfa/2)*sqrt(var(APDd)/N);
28      fprintf("Av. Packet Delay data (ms)     = %.2e +- %.2e\n", media, term);
29      delay(j, 1) = media;
30      errors(j, 1) = term;
31      %Calculate VoIP average delay and cofidence interval
32      media = mean(APDv);
33      term = norminv(1-alfa/2)*sqrt(var(APDv)/N);
34      fprintf("Av. Packet Delay VoIP (ms)     = %.2e +- %.2e\n", media, term);
35      delay(j, 2) = media;
36      errors(j, 2) = term;
37  end
38
39  %Plot results
40  f = figure('Name','Ex. 2.a)','NumberTitle','off');
41  b = bar(n, delay, "grouped");
42  hold on
43  % Calculate the number of groups and number of bars in each group
44  [ngroups,nbars] = size(delay);
45  % Get the x coordinate of the bars
46  x = nan(nbars, ngroups);
47  for i = 1:nbars
48      x(i,:) = b(i).XEndPoints;
49  end
50  er = errorbar(x', delay, errors, 'k','linestyle','none');
51  grid on
52  hold off
53
54  title('Average Packet Delay')
55  xlabel('Number VoIP packet Flows(n)')
56  ylabel('Delay (ms)')
57  legend({'Data', 'VoIP'}, 'Location', 'northwest');
```

3.1.2 Conclusões

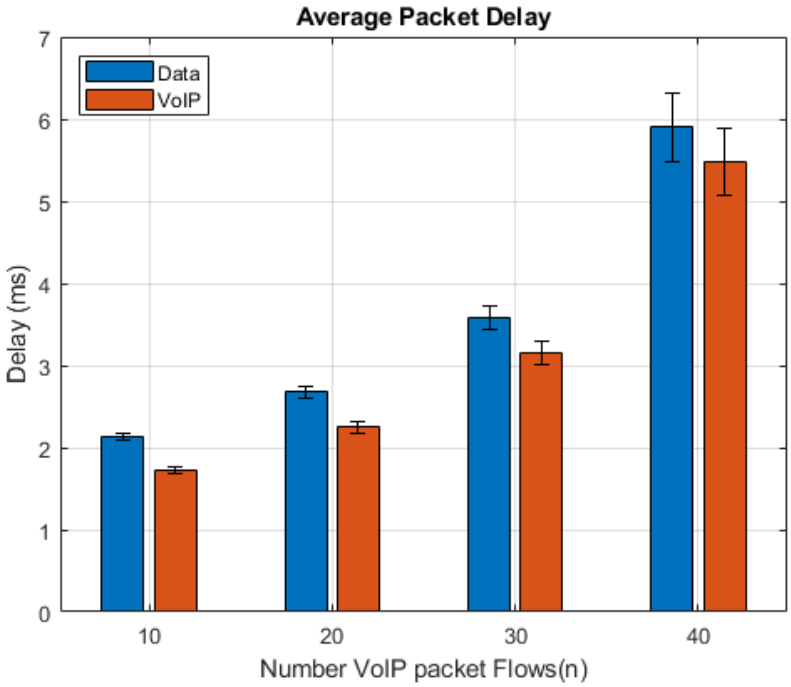

Figura 3.1: Resultado da execução da alínea 2.a).

Analisando os resultados podemos ver que à medida que aumentam os fluxos de pacotes VoIP também aumenta o atraso médio de ambos os tipos de pacotes. Esta conclusão deve-se ao facto de quantos mais pacotes houverem no sistema, maior é a probabilidade de a fila de espera estar cheia e o link ocupado, o que implica um maior atraso no atendimento dos pacotes.

No entanto uma particularidade deste gráfico é o facto de os pacotes de dados terem sempre um atraso ligeiramente maior que os de VoIP. Tendo em conta que os tamanhos dos pacotes VoIP são uniformemente distribuídos entre 110 e 130 Bytes e que os pacotes Data têm da valores variados de tamanho maior entre 64 e 1518 Bytes, de acordo com o capítulo anterior, em que se concluiu que pacotes com maior tamanho tendem a sofrer um maior atraso médio, podemos afirmar que o atraso no fluxo de Data será sempre maior que nos fluxos de VoIP. De notar que a média de tamanhos de pacotes Voip é 120 Bytes, mas para dados é 620,02 Bytes o que sustenta também a conclusão tirada anteriormente.

3.2 Alínea b)

Nesta alínea o simulador usado já foi o número 4, que suporta fluxos de pacotes de dados e de VoIP. No entanto neste caso a fila de espera não segue uma organização tipo FIFO, mas uma organização em que os pacotes de VoIP têm prioridade sobre os de dados. Quando um

pacote VoIP chega à fila de espera é posto à frente dos pacotes de dados que já lá estão à espera. Neste caso o simulador 4 foi corrido nas mesmas condições que o da alínea anterior.

3.2.1 Código

O código é igual ao exercício anterior 2.a), 3.1.1, com mudança em que é usado o Simulador4 e alterado o nome da janela e o título do gráfico. Acrescenta-se ainda, um gráfico, 3.2.2, com os valores exprimentais do *delay* médio dos pacotes VoIP em função de n. Reparo que os valores Average Packet Delay para os n fluxos VoIP no primeiro gráfico são os mesmos expressos no segundo.

```
[...]

    for i = 1:N
        [PLd(i), PLv(i), APDd(i), APDv(i), MPDd(i), MPDv(i), TT(i)] = ...
            Simulator4(l,C,f,P,n(j));
    end

[...]

f = figure('Name','Ex. 2.b)','NumberTitle','off');

[...]

figure(2)
bar(n, delay(:, 2))
title('Average Voip Packet Delay (Exprimental)')
xlabel('Number VoIP packet Flows(n)')
ylabel('Delay (ms)')
hold on
er = errorbar(n, delay(:, 2), errors(:, 2), errors(:, 2));
er.Color = [0 0 0];
er.LineStyle = 'none';
hold off;
```

3.2.2 Conclusões

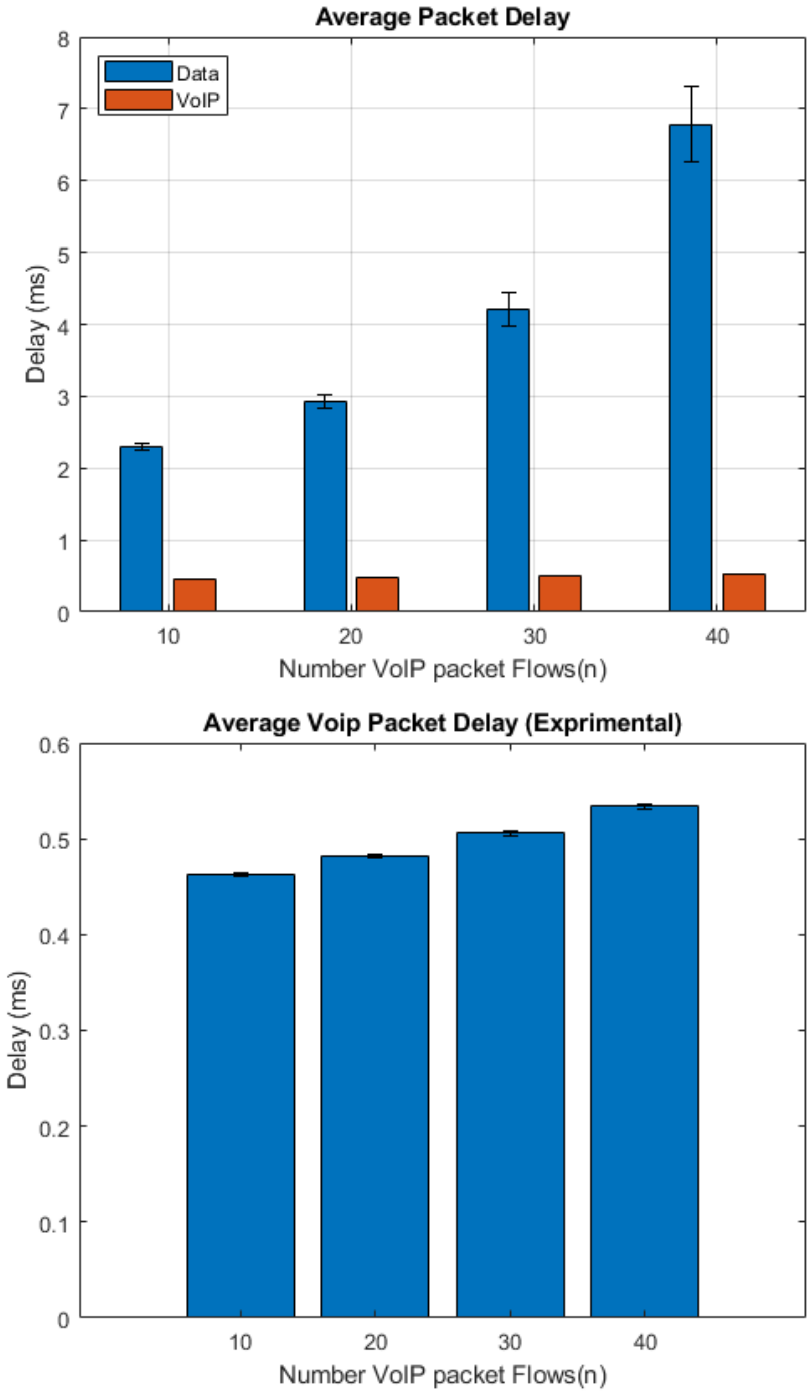

Figura 3.2: Resultado da execução da alínea 2.b).

Tal como esperado, quanto mais fluxos de pacotes VoIP maior o atraso médio de ambos os tipos de pacote. Comparando estes resultados com os obtidos na alínea a) podemos observar uma discrepância no que diz respeito ao atraso médio dos pacotes VoIP, que é consideravelmente menor. Sendo esta experiência nas mesmas condições, para capacidades de ligação iguais o atraso dos pacotes de dados aumentou ligeiramente em relação à alínea anterior. Isto justifica-se pelos os pacotes VoIP terem prioridade maior implicando que são servidos primeiro que os de prioridade mais baixa, neste caso Data. Assim os pacotes de Data tenderão a passar mais tempo no buffer à espera de serem processados levando a atrasos maiores. De notar que no limite, numa situação de n » 40 fluxos VoIP o atraso do fluxo de Data deteriorar-se-ia exponencialmente.

3.3 Alínea c)

De acordo com Simulador4 e os dados do exercício 2.a), existem 2 fluxos de prioridades diferentes: Data e VoIP, caracterizados por:

- VoIP, prioridade mais alta (n= 1), o intervalo entre chegada de pacotes é definido por uma distribuição uniforme de entre 16 e 24 ms. A média numa distribuição uniforme é dada por $\frac{a+b}{2}$, neste caso 20 ms. Assim a taxa de chegada de pacotes é $\frac{1}{20e-3}$ pps.
- O tamanho dos pacotes é também definido por uma distribuição uniforme entre 110 e 130 Bytes. Assim o tamanho médio de pacotes é 120 Bytes.
- Data, prioridade mais baixa (n= 2), a chegada de pacotes é definido por uma distribuição exponencial com uma taxa de 1500 pps.
- O tamanho dos pacotes é definido por uma distribuição aleatória de acordo com as seguintes percentagens:

  19% para pacotes de 64 bytes

  23% para pacotes de 110 bytes

  17% para pacotes de 1518 bytes

  igual probabilidade para todos os outros tamanhos de pacote

Num sistema M/G/1 com prioridades do tipo não preemptivo o atraso na fila de espera para a prioridade k é dado por:

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

sendo ρ_k dado por:

$$\rho_k = \lambda_k . E[S_k]$$

Para obter o atraso médio dos pacotes do sistema numa determinada prioridade k é preciso somar o tempo de transmissão médio dos pacotes considerados, $E[S_k]$

3.3.1 Código

Legenda

λ = lambda | ρ = p | μ = u | $E[S_k]$ = ESk | $E[S_k^2]$ = ESk_2

```
% 2 prioridades:
%   n = 1 -> VoIP
%   n = 2 -> data

% tamanho medio dos pacotes de data
prob = (1 - (0.19 + 0.23 + 0.17)) / ((109 - 65 + 1) + (1517 - 111 + 1));
AVGpckt_size = round(64*0.19 + 110*0.23 + 1518*0.17 + sum([65:109] * prob) ...
    + sum([111:1517] * prob), 2);
% tamanho medio dos pacotes de VoIP = 120
% taxa de chegada de pacotes do tipo VoIP e 1 a cada 20 ms em media
lambda1 = (1/20e-3);   % pps
lambda2 = 1500;     % pps
u1 = 10e6/(8 * 120);    % pps
u2 = 10e6/(8 * AVGpckt_size);   %pps
ES1= 1/u1;  % seg
ES2 = 1/u2;     % seg
ES1_2 = 2/(u1^2);   % seg^2
ES2_2 = 2/(u2^2);   % seg^2

n = [10, 20, 30, 40];
y = zeros(1,4);

for i = n
    lambda1 = (1/20e-3) * i;
    p1 = lambda1 * ES1;
    w1 = ((lambda1 * ES1_2 + lambda2 * ES2_2) / (2 * (1 - p1))) + ES1;
    % to ms
    w1 = w1 * 1e3;
    fprintf('n=%i -> w1 = %0.2f ms\n', i, w1);
    y(i/10) = w1;
end

figure(2);
bar(n, y);
title('Average Voip Packet Delay (Teorical)')
xlabel('Number of VoIP packet Flows')
ylabel('Average Voip Packet Delay (ms)')
```

3.3.2 Conclusões

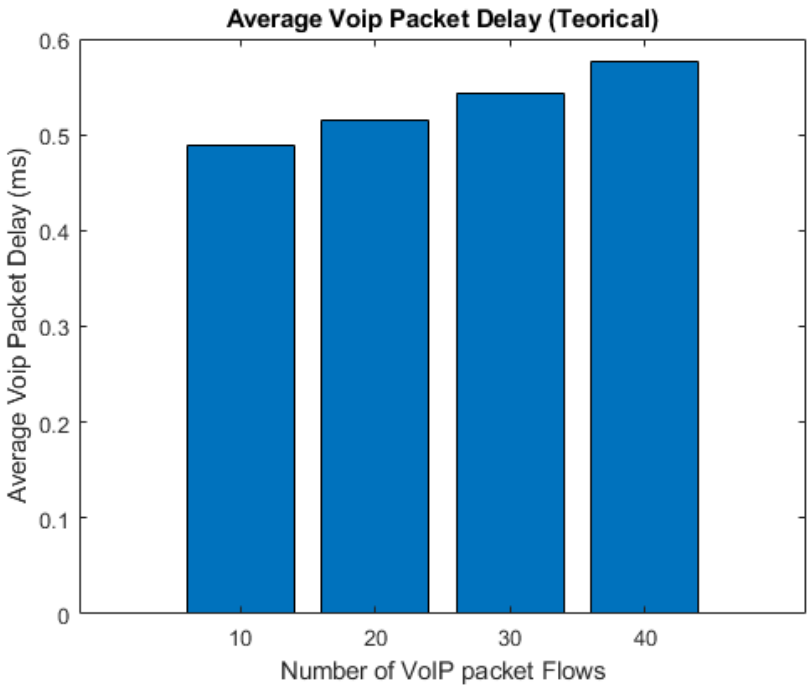

Figura 3.3: Resultado da execução da alínea 2.c).

Comparando os resultados obtidos com os da alínea anterior, chegamos à conclusão de que os valores teóricos são similares com os obtidos pelo simulador. Logo podemos então concluir que o simulador oferece uma boa aproximação teórica relativamente ao atraso médio nos pacotes VoIP.

3.4 Alínea d)

Para esta alínea, foi corrido o simulador 3 50 vezes que tal como já referido anteriormente suporta fluxos de pacotes VoIP e contém contadores estatísticos relativos a ambos os dois tipos de pacotes. Nesta situação foi variado novamente o número de fluxos VoIP e foi analisado o atraso médio e a percentagem de perda de pacotes para ambos os dois tipos (dados e VoIP).

3.4.1 Código

```
% ex2d
N = 50;

lambda = 1500;  % pps
f = 10000;
```

```
C = 10;          % Mbps
P = 10000;       % packets

alfa = 0.1;

n = [10, 20, 30, 40];

APD = zeros(4, N);
APDv = zeros(4, N);
PL = zeros(4, N);
PLv =zeros(4, N);

media = zeros(4, 4);
term = zeros(4, 4);

for i = [1:4]
    for x = [1:N]
        [PL(i, x), PLv(i, x), APD(i, x), APDv(i, x), ¬, ¬, ¬] = ...
            Simulator3(lambda, C, f, P, n(i));
    end

    media(i, 1) = mean(APD(i,:));
    media(i, 2) = mean(APDv(i,:));
    media(i, 3) = mean(PL(i,:));
    media(i, 4) = mean(PLv(i,:));

    term(i, 1) = norminv(1-alfa/2) * sqrt(var(APD(i,:))/N);
    term(i, 2) = norminv(1-alfa/2) * sqrt(var(APDv(i,:))/N);
    term(i, 3) = norminv(1-alfa/2) * sqrt(var(PL(i,:))/N);
    term(i, 4) = norminv(1-alfa/2) * sqrt(var(PLv(i,:))/N);
end

% display
n = categorical({'APD','APDv','PL','PLv'});
n = reordercats(n,{'APD','APDv','PL','PLv'});
f = figure('Name','Ex. 2.d)','NumberTitle','off');
f.Position = [680   558   1120   420];
t = tiledlayout(1,4);
t1 = nexttile;
bar(n, media(1, :));
hold on
er = errorbar(n,  media(1, :), term(1, :) * -1, term(1, :));
er.Color = [0 0 0];
er.LineStyle = 'none';
xlabel(t1, "n = 10");
grid on
hold off

t2 = nexttile;
bar(n, media(2, :));
hold on
er = errorbar(n,  media(2, :), term(2, :) * -1, term(2, :));
er.Color = [0 0 0];
er.LineStyle = 'none';
xlabel(t2, "n = 20");
grid on
```

```
hold off

t3 = nexttile;
bar(n, media(3, :));
hold on
er = errorbar(n,  media(3, :), term(3, :) * -1, term(3, :));
er.Color = [0 0 0];
er.LineStyle = 'none';
xlabel(t3, "n = 30");
grid on
hold off

t4 = nexttile;
bar(n, media(4, :));
hold on
er = errorbar(n,  media(4, :), term(4, :) * -1, term(4, :));
er.Color = [0 0 0];
er.LineStyle = 'none';
xlabel(t4, "n = 40");
grid on
hold off

linkaxes([t1, t2, t3, t4], 'y');

title(t, 'APD and PL for data and VoIP with diferent number of VoIP ...
    flows(n), considering single FIFO Queue');
ylabel(t, 'APD, APDv (ms) | PL, PLv (%)')
```

3.4.2 Conclusões

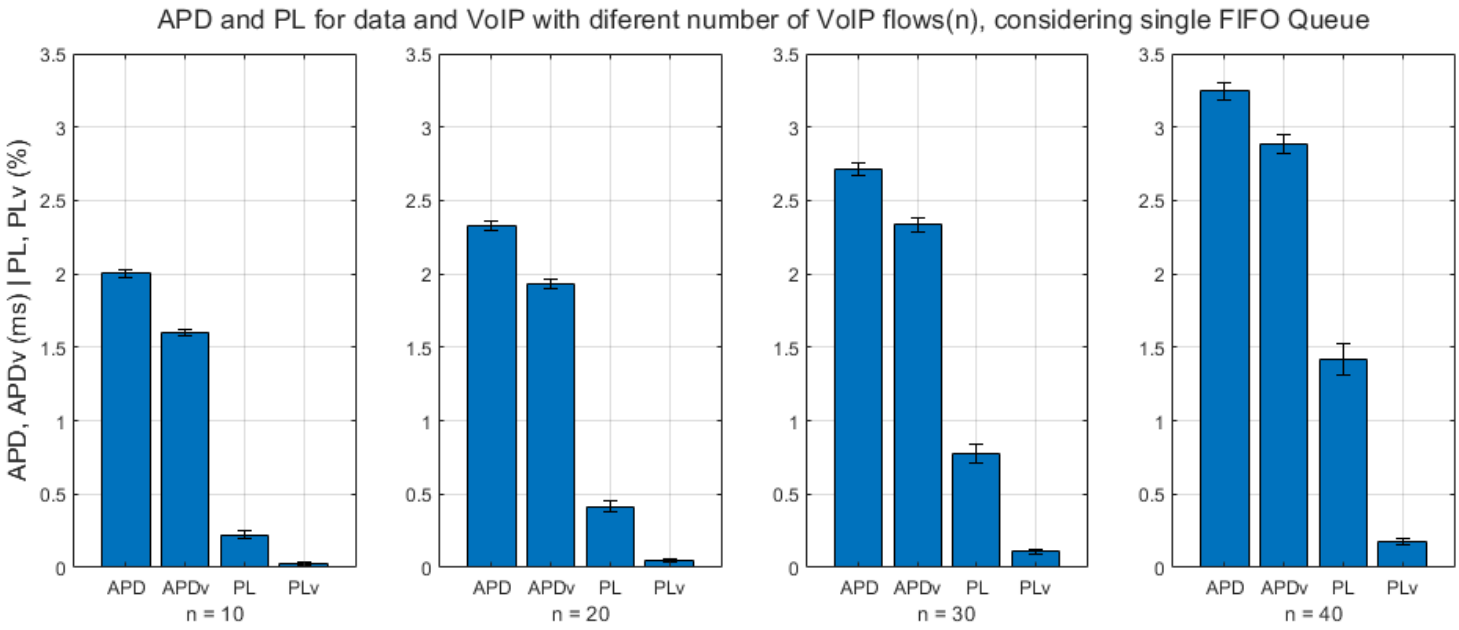

Figura 3.4: Resultado da execução da alínea 2.d).

Sabendo que nesta experiência os pacotes são multiplexados estatisticamente numa queue do tipo FIFO, à medida que o número de fluxos VoIP aumenta, também o atraso médio e a percentagem de perda de pacotes para todos fluxos do sistema aumentam (neste caso VoIP e Data). De facto, mais fluxos de pacotes implica uma maior quantidade de tráfego a ser gerado que leva a maiores atrasos e perdas.

3.5 Alínea e)

Foi repetida a experiência da alínea anterior mas desta vez com o simulador 4. Foi corrido 50 vezes e foram extraídos os valores resultantes da sua execução.

3.5.1 Código

O código é igual ao exercício anterior 2.d), 3.4.1, com mudança é usado o Simulador4 e alterado o nome da janela e o título do gráfico.

```
[...]

for i = [1:4]
    for x = [1:N]
```

```
            [PL(i, x), PLv(i, x), APD(i, x), APDv(i, x), ¬, ¬, ¬] = ...
                Simulator4(lambda, C, f, P, n(i));
        end

    [...]

    f = figure('Name','Ex. 2.e)','NumberTitle','off');

    [...]

    title(t, "APD and PL for data and VoIP with diferent number of VoIP ...
        flows(n), considering Priority Queuing(Voip > data)");
    ylabel(t, "APD, APDv (ms) | PL, PLv (%)")
```

3.5.2 Conclusões

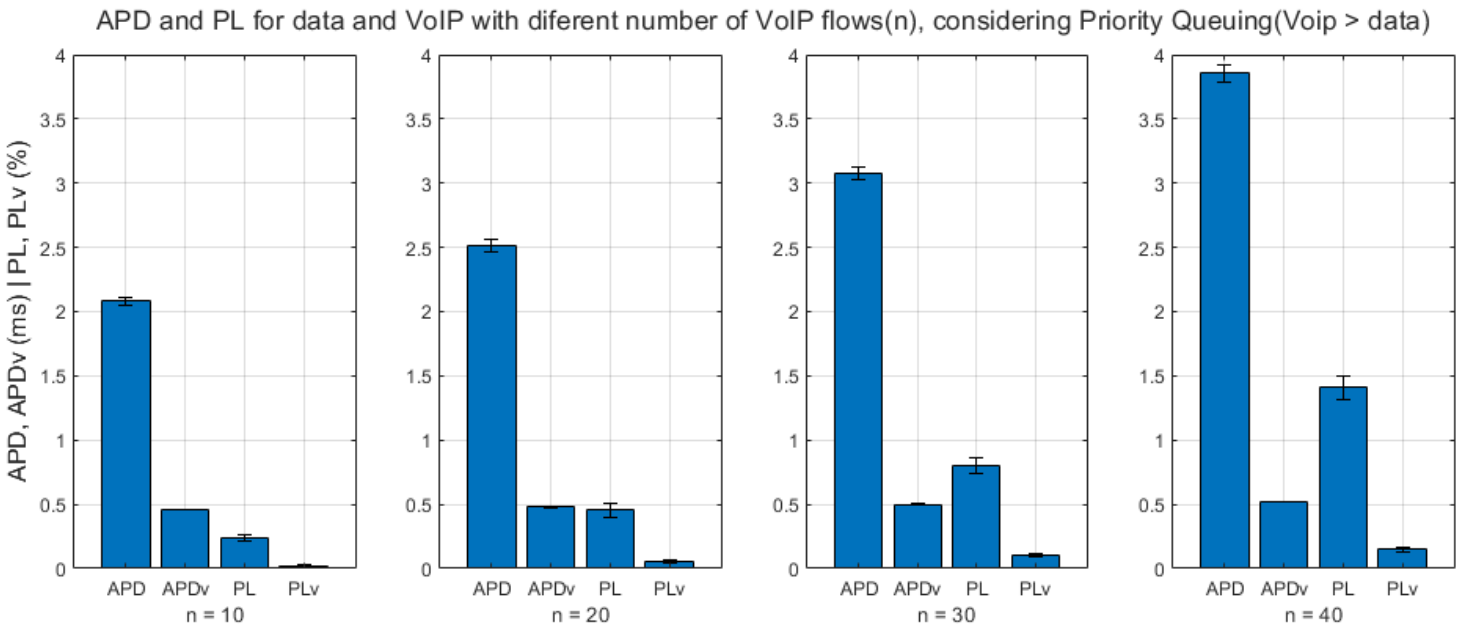

Figura 3.5: Resultado da execução da alínea 2.e).

Esta experiência é similar à anterior contudo os pacotes de VoIP tem prioridade superior aos de Data. Conclui-se então, que à medida que aumentam o número de fluxos de pacotes VoIP, aumentam o atraso médio e perda de pacotes, mas ao invés de aumentarem equitativamente verifica-se o aumento muito maior nos pacotes Data. Isto deve-se ao facto de os pacotes VoIP serem servidos primeiro que os de Data. Podemos constatar ainda que apesar de o número de fluxos VoIP aumentar(n = 10, 20, 30 e 40) o atraso médio é praticamente igual, perto de 0.5 ms, e que nunca se regista uma perda de pacotes superior a 0.2%. Ainda em relação ao exercício anterior, focando no gráfico de ambos, podemos concluir que nas 2 disciplinas(tipo FIFO e Priority queuing) os valores apresentados para o PL e PLv são, em

relação aos valores do outro gráfico, similares, significando que o tamanho da queue nesta experiência não afecta o resultado para a perda de pacotes(é grande o suficiente para as duas disciplinas). É no atraso médio em que a prioridade dada ao pacotes tipo VoIP se faz notar. O atraso médio dos pacotes VoIP mantém-se constante, como referido anteriormente(≈ 0.5 ms), mas o atraso médio dos pacotes tipo Data é sempre superior aos valores registos na disciplina FIFO.

3.6 Alínea f)

Para esta alínea era necessário alterar o simulador 4 para que os pacotes de dados não serem aceites caso o tamanho da fila se tornasse maior que 90% da sua ocupação. Logo foi apenas necessário fazer uma pequena alteração no simulador 4 e esta foi numa expressão condicional em que se verifica se um pacote de dados vai ser aceite ou não (na versão anterior apenas era aceite se houvesse espaço para ele). A alteração realizada passou então por substituir a linha desse *if* por uma versão em que se verifica a condição referida anteriormente.

3.6.1 Código

Alterações ao Simulador 4

```
function [PLd, PLv, APDd, APDv, MPDd, MPDv, TT] = ...
    ex2f_Simulator4(lambda,C,f,P,n)

[...]

%Similation loop:
while (TRANSMITTEDPACKETS+TRANSMITTEDPACKETSV)<P
    [ . . . ]
    switch Event
        case ARRIVAL                    % If first event is an ARRIVAL
            if type == 0  % Data packet
                TOTALPACKETS= TOTALPACKETS+1;
                tmp= Clock + exprnd(1/lambda);
                EventList = [EventList; ARRIVAL, tmp, ...
                    GeneratePacketSize(), tmp, 0];
                if STATE==0
                    STATE= 1;
                    EventList = [EventList; DEPARTURE, Clock + ...
                        8*PacketSize/(C*10^6), PacketSize, Clock, 0];
                else                                        % Queue ...
                    occupation must not become higher than 90%
                    if ((QUEUEOCCUPATION+PacketSize)/f) ≤ 0.9
                        QUEUE= [QUEUE;PacketSize , Clock, 0];
                        QUEUEOCCUPATION= QUEUEOCCUPATION + PacketSize;
                    else
                        LOSTPACKETS= LOSTPACKETS + 1;
                    end
                end
                [ ... ]
```

```
[ ... ]
```

Código para correr o simulador

```
N = 50;          % Number of Simulations
lambda = 1500;   % Packet Rate pps
f = 10000;       % Queue Size Bytes
C = 10;          % Link Capacity in Mbps
P = 10000;       % Stopping criterion
alfa = 0.1;      % Cofidence interval of 90%
%Values for the VoIP flows
n = [10, 20, 30, 40];

% Vectors to store simulation values
APD = zeros(4, N);
APDv = zeros(4, N);
PL = zeros(4, N);
PLv =zeros(4, N);
media = zeros(4, 4);
term = zeros(4, 4);

% Run and gather simulation results
for i = [1:4]
    for x = [1:N]
        [PL(i, x), PLv(i, x), APD(i, x), APDv(i, x), ¬, ¬, ¬] = ...
            ex2f_Simulator4(lambda, C, f, P, n(i));
    end
    % Calculate Average packet delay and loss and their cofidence interval
    media(i, 1) = mean(APD(i,:));
    media(i, 2) = mean(APDv(i,:));
    media(i, 3) = mean(PL(i,:));
    media(i, 4) = mean(PLv(i,:));
    term(i, 1) = norminv(1-alfa/2) * sqrt(var(APD(i,:))/N);
    term(i, 2) = norminv(1-alfa/2) * sqrt(var(APDv(i,:))/N);
    term(i, 3) = norminv(1-alfa/2) * sqrt(var(PL(i,:))/N);
    term(i, 4) = norminv(1-alfa/2) * sqrt(var(PLv(i,:))/N);
end

% display results
n = categorical({'APD','APDv','PL','PLv'});
n = reordercats(n,{'APD','APDv','PL','PLv'});
f = figure('Name','Ex. 2.e)','NumberTitle','off');
f.Position = [100 100 1050 400];

t = tiledlayout(1,4);
t1 = nexttile;
bar(n, media(1, :));
hold on
er = errorbar(n,  media(1, :), term(1, :) * -1, term(1, :));
er.Color = [0 0 0];
er.LineStyle = 'none';
xlabel(t1, "n = 10");
grid on
hold off
```

```
t2 = nexttile;
bar(n, media(2, :));
hold on
er = errorbar(n,  media(2, :), term(2, :) * -1, term(2, :));
er.Color = [0 0 0];
er.LineStyle = 'none';
xlabel(t2, "n = 20");
grid on
hold off

t3 = nexttile;
bar(n, media(3, :));
hold on
er = errorbar(n,  media(3, :), term(3, :) * -1, term(3, :));
er.Color = [0 0 0];
er.LineStyle = 'none';
xlabel(t3, "n = 30");
grid on
hold off

t4 = nexttile;
bar(n, media(4, :));
hold on
er = errorbar(n,  media(4, :), term(4, :) * -1, term(4, :));
er.Color = [0 0 0];
er.LineStyle = 'none';
xlabel(t4, "n = 40");
grid on
hold off

linkaxes([t1, t2, t3, t4], 'y');
% Graph title
title(t, 'APD and PL for data and VoIP with diferent number of VoIP ...
    flows(n), considering Priority Queuing (Simplified WRED)');
ylabel(t, 'APD, APDv (ms) | PL, PLv (%)')
```

3.6.2 Conclusões

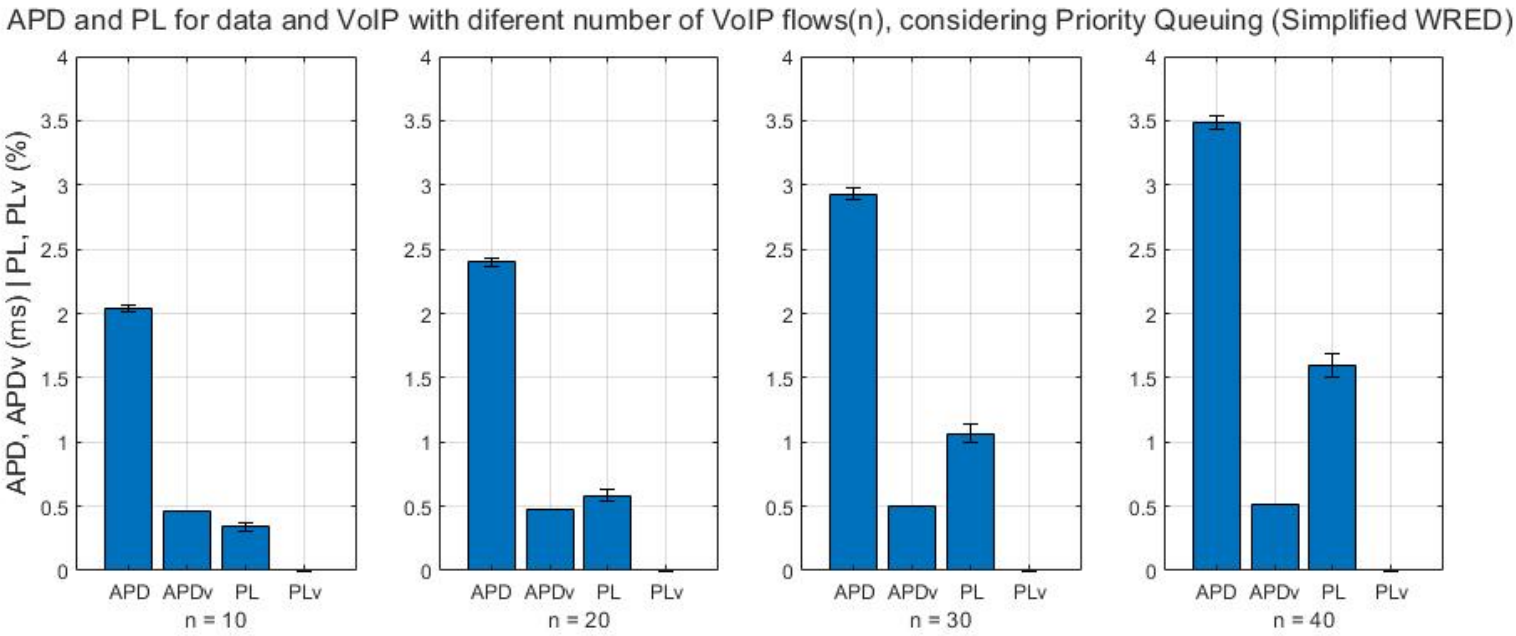

Figura 3.6: Resultado da execução da alínea 2.f).

Analisando os resultados obtidos concluímos que a taxa de atrasos e perdas de pacotes aumenta à medida que aumentam o número de fluxos de pacotes VoIP. No entanto tendo em conta que os pacotes de dados não são aceites se a taxa de ocupação da fila de espera for maior que 90%, observamos então que os atrasos de pacotes VoIP aumentam muito pouco, enquanto que a perda de pacotes do mesmo tipo é nula. Isto ocorre não só pela condição referida anteriormente mas também pelo facto dos pacotes VoIP terem prioridade sobre os de dados.

Ou seja para pacotes de dados que estejam à espera de ser atendidos na queue, vão ver a sua vez constantemente adiada pela chegada de novos pacotes de VoIP. A perda de pacotes de dados também sofre um aumento significativamente maior do que os da alínea anterior pois quanto mais cheia a queue estiver, maior é a probabilidade de estes serem descartados pelo facto da ocupação ultrapassar os 90%. Este tipo de configuração da fila de espera quase que deixa sempre espaço livre para novos pacotes VoIP, o que justifica as perdas nulas dos pacotes VoIP.