Modelação e Desempenho de Redes e Serviços

Mini-Project - Nº 1

João Monteiro (102690), João Gaspar (107708)

Departamento de Eletrónica, Telecomunicações e Informática (DETI)

Universidade de Aveiro

29 de outubro de 2024

Conteúdo

Lista de Figuras

Introdução

Nos últimos anos, o crescimento das redes de comunicação e o aumento da procura por serviços em tempo real, como VoIP (Voz sobre IP) e streaming de vídeo, têm apresentado desafios significativos para garantir a Qualidade de Serviço (QoS) nessas redes. Com a crescente variedade de aplicações e a competição entre diferentes tipos de tráfego, é essencial entender o comportamento das redes sob diferentes condições de carga. Isso é particularmente importante quando diferentes tipos de tráfego, como dados e VoIP, competem pelos mesmos recursos de rede, uma vez que esses serviços possuem requisitos distintos de atraso e fiabilidade.

O objetivo principal deste projeto é realizar uma análise de desempenho de ligações ponto-a-ponto que suportam serviços baseados em pacotes. Para isso, foram realizadas simulações que analisam métricas de desempenho críticas, como o *atraso médio de pacotes*, a *perda de pacotes*, o *atraso máximo* e o *throughput total*, em função do número de fluxos VoIP ativos. Ao utilizar um simulador desenvolvido em MATLAB, foi possível obter resultados estatísticos com intervalos de confiança de 90%, proporcionando uma visão robusta do impacto do aumento do tráfego VoIP sobre o tráfego de dados e sobre a própria qualidade da transmissão VoIP.

Este relatório está organizado de forma a apresentar uma análise dos resultados obtidos. Cada secção inclui exercícios que abordam diferentes aspetos do desempenho da rede, contendo o código utilizado para as simulações e as respetivas análises de resultados e conclusões.

Capítulo 1

Task 1

1.1 Exercício 1.a.

1.1.1 Código

```
%% Exercise 1.a.

C = 10;      % Capacity of the link (Mbps)
f = 10000;   % Size of the queue (Bytes)
N = 20;      % Times to run the simulation
P = 100000; % Stopping criteria
alfa = 0.1; % 90% confidence interval

b = 10^-6;                                    % Bit error
    rate
arrival_rate = [1500 1600 1700 1800 1900];  % Arrival
    rate values (pps)

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

APD_values = zeros(1, length(arrival_rate));
APD_terms = zeros(1, length(arrival_rate));
PL_values = zeros(1, length(arrival_rate));
PL_terms = zeros(1, length(arrival_rate));

for i = 1:length(arrival_rate)
    lambda = arrival_rate(i);
    for j = 1:N
        [PL(j), APD(j), MPD(j), TT(j)] = Sim2(lambda, C,
            f, P, b);
    end

```

```
    media_PL = mean(PL);
    term_PL = norminv(1-alfa/2)*sqrt(var(PL)/N);
    PL_values(i) = media_PL;
    PL_terms(i) = term_PL;

    media_APD = mean(APD);
    term_APD = norminv(1-alfa/2)*sqrt(var(APD)/N);
    APD_values(i) = media_APD;
    APD_terms(i) = term_APD;
end

figure;
bar(arrival_rate, PL_values);
hold on;
grid on;
errorbar(arrival_rate, PL_values, PL_terms, '.');
title('Average Packet Loss');
xlabel('Arrival Rate (pps)');
ylabel('Packet Loss (%)');
hold off;

figure;
bar(arrival_rate, APD_values);
hold on;
grid on;
errorbar(arrival_rate, APD_values, APD_terms, '.');
title('Average Packet Delay');
xlabel('Arrival Rate (pps)');
ylabel('Average Delay (ms)');
hold off;
```

1.1.2 Resultados e Conclusões

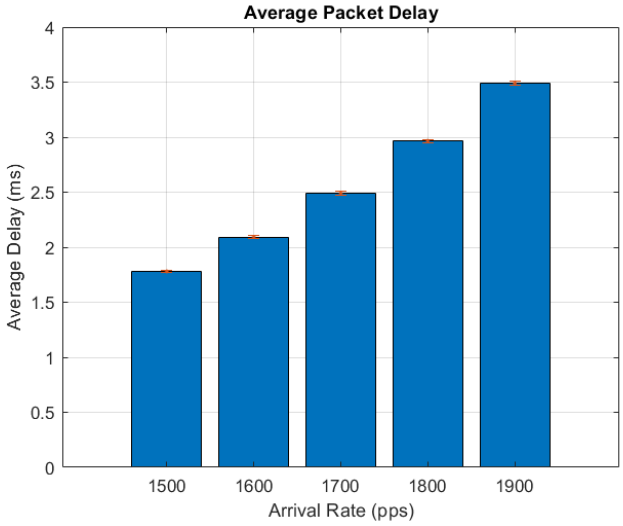

Figura 1.1: *Average Packet Delay* em função da taxa de chegada (*pps*). Com $b = 10^{-6}$.

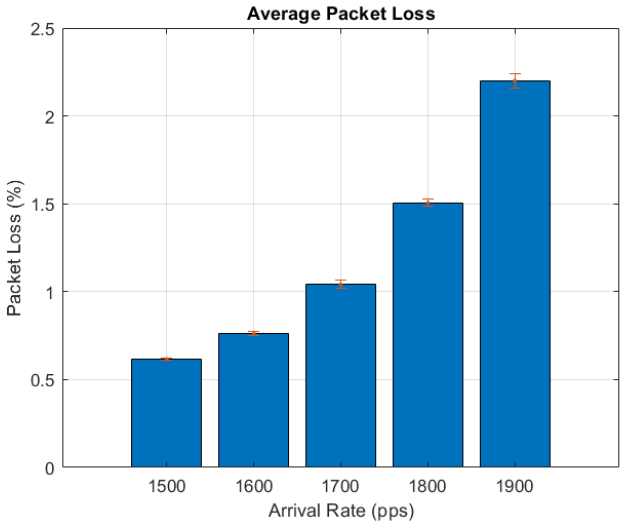

Figura 1.2: *Packet Loss* em função da taxa de chegada (*pps*). Com $b = 10^{-6}$.

Nesta secção, analisamos os resultados obtidos para o *average packet delay* e o *packet loss* em função da taxa de chegada de pacotes, utilizando uma fila com capacidade de 10.000 bytes e uma taxa de chegada variando entre 1500 e 1900 pps bem como um *bit error rate* de $b = 10^{-6}$.

No gráfico da figura 1.1, observa-se um aumento constante no *average packet delay* à medida que a taxa de chegada de pacotes aumenta. Este comportamento está de acordo com o esperado, uma vez que, com uma fila de tamanho fixo, à medida que mais pacotes chegam, eles precisam esperar mais tempo na fila antes de serem transmitidos, resultando em maiores atrasos.

Os valores do atraso médio dos pacotes aumentam significativamente com o aumento da taxa de chegada. Este aumento reflete o fato de que, com mais pacotes a seram processados, o tempo de espera acumula, especialmente nas taxas de chegada mais altas. Notamos também que as barras de erro nos gráficos são relativamente pequenas, o que sugere que os resultados são consistentes ao longo das 20 execuções da simulação.

O gráfico da figura 1.2 mostra o comportamento do *packet loss* para as mesmas taxas de chegada. Observa-se que, para taxas de chegada de 1500 pps e 1600 pps, a perda de pacotes é relativamente baixa. No entanto, à medida que a taxa de chegada aumenta, a perda de pacotes cresce de maneira acentuada, refletindo o impacto do congestionamento no sistema.

Os resultados apresentados demonstram que tanto o atraso médio dos pacotes quanto a perda de pacotes aumentam à medida que a taxa de chegada de pacotes se aproxima da capacidade máxima da rede. Com filas de tamanho maior, observou-se que, embora o atraso médio aumente, a perda de pacotes diminui, já que há mais capacidade para armazenar e processar pacotes antes de serem descartados.

1.2 Exercício 1.b.

1.2.1 Código

```
%% Exercise 1.b.

C = 10;      % Capacity of the link (Mbps)
f = 10000;   % Size of the queue (Bytes)
N = 20;      % Times to run the simulation
P = 100000; % Stopping criteria
alfa = 0.1; % 90% confidence interval

b = 10^-4;                                    % Bit error
    rate
arrival_rate = [1500 1600 1700 1800 1900];  % Arrival
    rate values (pps)

% Variables to store the simulation results
PL = zeros(1, N);
APD = zeros(1, N);
MPD = zeros(1, N);
TT = zeros(1, N);

```

```
% Variables to store bar graph data
APD_values = zeros(1, length(arrival_rate));
APD_terms = zeros(1, length(arrival_rate));
PL_values = zeros(1, length(arrival_rate));
PL_terms = zeros(1, length(arrival_rate));

for i = 1:length(arrival_rate)
    lambda = arrival_rate(i);
    for j = 1:N
        [PL(j), APD(j), MPD(j), TT(j)] = Sim2(lambda, C,
            f, P, b);
    end

    media_PL = mean(PL);
    term_PL = norminv(1-alfa/2)*sqrt(var(PL)/N);
    PL_values(i) = media_PL;
    PL_terms(i) = term_PL;

    media_APD = mean(APD);
    term_APD = norminv(1-alfa/2)*sqrt(var(APD)/N);
    APD_values(i) = media_APD;
    APD_terms(i) = term_APD;
end

% Plotting the results in separate bar charts with error
    bars
figure;
bar(arrival_rate, PL_values);
hold on;
grid on;
errorbar(arrival_rate, PL_values, PL_terms, '.');
title('Average Packet Loss');
xlabel('Arrival Rate (pps)');
ylabel('Packet Loss (%)');
hold off;

figure;
bar(arrival_rate, APD_values);
hold on;
grid on;
errorbar(arrival_rate, APD_values, APD_terms, '.');
title('Average Packet Delay');
xlabel('Arrival Rate (pps)');
ylabel('Average Delay (ms)');
hold off;
```

1.2.2 Resultados e Conclusões

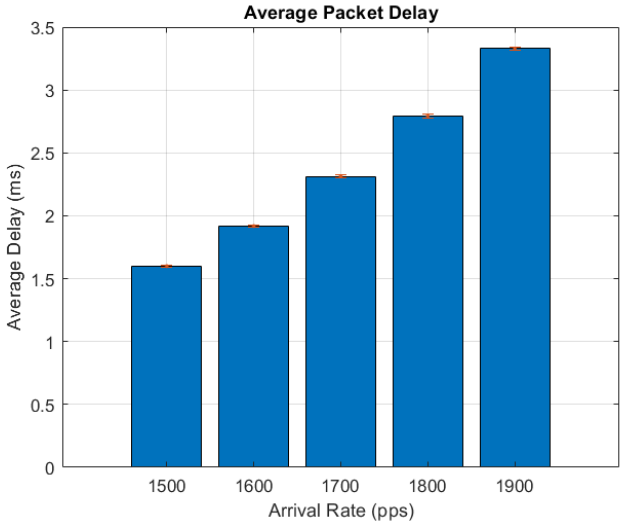

Figura 1.3: *Average Packet Delay* em função da taxa de chegada (*pps*). Com $b = 10^{-4}$.

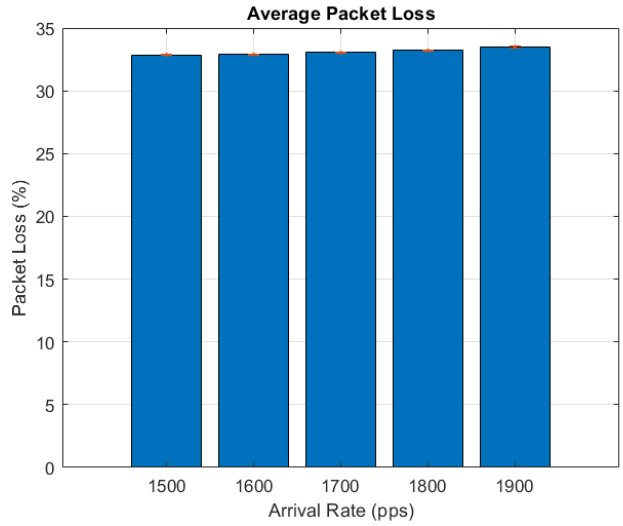

Figura 1.4: *Packet Loss* em função da taxa de chegada (*pps*). Com $b = 10^{-4}$.

Nesta secção, analisamos os resultados obtidos para o *average packet delay* e a *packet loss* em função da taxa de chegada de pacotes, utilizando uma fila com capacidade de 10.000 bytes e uma taxa de chegada variando entre 1500 e 1900 pps, agora considerando uma taxa de erro de bits de

$b = 10^{-4}$.

Os gráficos obtidos para o *average packet delay* e a *packet loss* foram plotados e comparados com os resultados do exercício 1.a. (Seção 1.1).

No gráfico da Figura 1.3, observa-se que o *average packet delay* permanece praticamente inalterado em relação ao exercício anterior, mesmo com a alteração na taxa de erro. Isso deve-se ao facto de que o atraso médio dos pacotes é mais influenciado pela taxa de chegada e pela capacidade de processamento da fila do que pela taxa de erro. Assim, a estrutura de enfileiramento e a taxa de chegada de pacotes continuam a dominar o comportamento do atraso no sistema.

Por outro lado, o gráfico da Figura 1.4 revela uma mudança significativa no comportamento de *packet loss*. Enquanto no exercício anterior a perda de pacotes aumentava gradualmente com a taxa de chegada, neste exercício o *packet loss* manteve-se constante em todos os valores de taxa de chegada, com uma média de aproximadamente 32,5%. Esta diferença drástica pode ser atribuída ao aumento da taxa de erro de bits, que leva a um maior número de pacotes a serem corrompidos antes de serem processados. Como resultado, a capacidade efetiva da fila é reduzida, resultando numa perda de pacotes mais acentuada.

Estes resultados indicam que, com um aumento na taxa de erro de bits, a performance da rede degrada-se significativamente, resultando num aumento constante da perda de pacotes independentemente da taxa de chegada.

1.3 Exercício 1.c.

1.3.1 Código

```
%% Exercise 1.c.

% Parameters
ber = [10^-6, 10^-4];  % Bit error rates for experiments 1.a and 1.b
pkt_sizes = [64, 110, 1518];  % Specific packet sizes
pkt_probs = [0.19, 0.23, 0.17];  % Probabilities of specific packet sizes
other_prob = (1 - sum(pkt_probs)) / ((109 - 65 + 1) + (1517 - 111 + 1));  % Probability of other sizes

% Initialize vector to store theoretical packet loss
ploss = zeros(1, length(ber));

% Calculate theoretical packet loss for each BER
for i = 1:length(ber)
    for size = 64:1518
        if size == 64
```

```
16              ploss(i) = ploss(i) + (1 - (1 - ber(i))^(size
                    * 8)) * 0.19;
17          elseif size == 110
18              ploss(i) = ploss(i) + (1 - (1 - ber(i))^(size
                    * 8)) * 0.23;
19          elseif size == 1518
20              ploss(i) = ploss(i) + (1 - (1 - ber(i))^(size
                    * 8)) * 0.17;
21          else
22              ploss(i) = ploss(i) + (1 - (1 - ber(i))^(size
                    * 8)) * other_prob;
23          end
24      end
25  end
26
27  % Normalize theoretical packet loss
28  ploss = (ploss ./ (0.19 + 0.23 + 0.17 + ((109 - 65 + 1) +
        (1517 - 111 + 1)) * other_prob)) * 100;
29
30  % Compare results
31  fprintf('Theoretical packet loss for b = 10^-6: %.4f%%\n'
        , ploss(1));
32  fprintf('Theoretical packet loss for b = 10^-4: %.4f%%\n'
        , ploss(2));
```

1.3.2 Código - Explicação

O código define os parâmetros necessários para calcular a perda de pacotes teórica devido à taxa de erro de bits (BER). As taxas de erro de bits são definidas como ber $= [10^{-6}, 10^{-4}]$, e os tamanhos específicos dos pacotes são definidos como pkt_sizes $= [64, 110, 1518]$ com as respetivas probabilidades pkt_probs $= [0.19, 0.23, 0.17]$. Estas probabilidades representam a distribuição dos tamanhos de pacotes: 19% dos pacotes têm 64 bytes, 23% têm 110 bytes e 17% têm 1518 bytes. A probabilidade dos outros tamanhos de pacotes é calculada como:

$$\text{other\_prob} = \frac{1 - \sum \text{pkt\_probs}}{(109 - 65 + 1) + (1517 - 111 + 1)}$$

De seguida, um vetor ploss é inicializado para armazenar a perda de pacotes teórica para cada taxa de erro de bits. O código entranum ciclo para calcular a perda de pacotes teórica para cada BER. Dentro deste ciclo, existe outro ciclo que itera sobre todos os tamanhos de pacotes possíveis (de 64 a 1518 bytes).

Para cada tamanho de pacote, o código verifica se o tamanho é 64, 110 ou 1518 bytes e calcula a probabilidade de que pelo menos um bit esteja errado no pacote, multiplicando pela probabilidade de ocorrência desse tamanho

de pacote. Para todos os outros tamanhos de pacotes, a probabilidade é multiplicada por other_prob.

A probabilidade de que pelo menos um bit esteja errado num pacote de tamanho L bits é calculada como:

$$1 - (1 - b)^L$$

Após calcular a perda de pacotes para todos os tamanhos de pacotes, o código normaliza a perda de pacotes teórica dividindo pelo somatório das probabilidades de todos os tamanhos de pacotes e multiplicando por 100 para obter a percentagem:

$$\text{ploss} = \left(\frac{\text{ploss}}{0.19 + 0.23 + 0.17 + ((109 - 65 + 1) + (1517 - 111 + 1)) \times \text{other\_prob}} \right) \times 100$$

Finalmente, os resultados da perda de pacotes teórica são apresentados para as taxas de erro de bits $b = 10^{-6}$ e $b = 10^{-4}$.

1.3.3 Resultados e Conclusões

Os valores teóricos de perda de pacotes devido à taxa de erro de bits (BER) foram calculados como segue:

```
>> ex1c
Theoretical packet loss for b = 10^-6: 0.4937%
Theoretical packet loss for b = 10^-4: 32.8278%
```

Figura 1.5: Perda de pacotes teórica devido à taxa de erro de bits (BER)

Os resultados simulados dos exercícios 1.a. e 1.b. (Seção 1.1 e Seção 1.2) foram comparados com os valores teóricos:

- **Perda de pacotes simulada para $b = 10^{-6}$ (aproximadamente):**
  - 1500 pps: 0.6%
  - 1600 pps: 0.75%
  - 1700 pps: 1.05%
  - 1800 pps: 1.5%
  - 1900 pps: 2.25%
- **Perda de pacotes simulada para $b = 10^{-4}$:** 32.5% (constante para todas as taxas de chegada)

Os resultados obtidos nos exercícios 1.a. e 1.b. são consistentes com os valores teóricos de perda de pacotes devido à taxa de erro de bits. Isso

sugere que a perda de pacotes nos exercícios 1.a e 1.b é predominantemente causada pela taxa de erro de bits.

Os valores teóricos e simulados são semelhantes, o que confirma a hipótese de que a perda de pacotes é principalmente devido à taxa de erro de bits, uma vez que o sistema consegue processar todos os pacotes a tempo. Este resultado indica que, sob as condições simuladas, o sistema é capaz de lidar com a carga de tráfego sem causar perdas adicionais de pacotes devido a congestionamento ou overflow da fila.

Capítulo 2

Task 2

2.1 Exercício 2.a.

2.1.1 Código - Sim3A

```
function [PLdata, PLVoIP, APDdata, APDVoIP, MPDdata,
    MPDVoIP, TT] = Sim3A(lambda, C, f, P, n, b)
% INPUT PARAMETERS:
% ... (mais codigo)
%  b       - bit error rate
% ... (mais codigo)
% Simulation loop:
while TRANSPACKETS_DATA + TRANSPACKETS_VOIP < P  %
    Stopping criterium
     % ... (mais codigo)
     switch Event
         case ARRIVAL_DATA  % If first event is an
             ARRIVAL_DATA
              % ... (mais codigo)
         case ARRIVAL_VOIP  % If first event is an
             ARRIVAL_VOIP
              % ... (mais codigo)
         case DEPARTURE  % If first event is a DEPARTURE
              if isPacketCorrupted(PacketSize, b)
                  if PacketType == ARRIVAL_DATA
                      LOSTPACKETS_DATA = LOSTPACKETS_DATA +
                          1;
                  else
                      LOSTPACKETS_VOIP = LOSTPACKETS_VOIP +
                          1;
                  end
              else
                  if PacketType == ARRIVAL_DATA
                      TRANSBYTES_DATA = TRANSBYTES_DATA +
                         PacketSize;
```

```
24                      DELAYS_DATA = DELAYS_DATA + (Clock -
                            ArrInstant);
25                      if Clock - ArrInstant > MAXDELAY_DATA
26                          MAXDELAY_DATA = Clock -
                                ArrInstant;
27                      end
28                      TRANSPACKETS_DATA = TRANSPACKETS_DATA
                            + 1;
29                  else
30                      TRANSBYTES_VOIP = TRANSBYTES_VOIP +
                            PacketSize;
31                      DELAYS_VOIP = DELAYS_VOIP + (Clock -
                            ArrInstant);
32                      if Clock - ArrInstant > MAXDELAY_VOIP
33                          MAXDELAY_VOIP = Clock -
                                ArrInstant;
34                      end
35                      TRANSPACKETS_VOIP = TRANSPACKETS_VOIP
                            + 1;
36                  end
37              end
38              % ... (mais codigo)
39      end
40  end
41
42  % ... (mais codigo)
43
44  function isCorrupted = isPacketCorrupted(PacketSize, b)
45      nBits = PacketSize * 8;
46      pNoError = (1 - b)^nBits;
47      isCorrupted = rand() > pNoError;
48  end
```

2.1.2 Diferenças e Justificações

As principais diferenças entre as simulações `Sim3` e `Sim3A` são as seguintes:

- **Parâmetros da Função:**
  Foi adicionado o parâmetro `b` como entrada na função `Sim3A`. Isto permite que a função considere a taxa de erro de bits (BER) ao estimar os parâmetros de desempenho.

- **Verificação de Corrupção de Pacotes:**
  Foi adicionada uma chamada para `isPacketCorrupted(PacketSize, b)` no caso `DEPARTURE` para verificar se o pacote está corrompido. Isto garante que os pacotes sejam verificados quanto à corrupção devido a erros de bits antes de serem contabilizados como transmitidos com sucesso.

- **Tratamento de Pacotes Perdidos:**
  Se um pacote estiver corrompido, ele é contabilizado como perdido (incremento de `LOSTPACKETS_DATA` ou `LOSTPACKETS_VOIP`). Isto reflecte com precisão o impacto dos erros de bits na perda de pacotes.

- **Função Auxiliar:**
  Foi adicionada a função `isPacketCorrupted`, que determina se um pacote está corrompido com base na taxa de erro de bits `b`.

2.1.3 Código - 2.a.

```
%% Exercise 2.a.

% Parameters
lambda = 1500;                    % packet rate (packets/
    sec)
C = 10;                           % link bandwidth (Mbps)
f = 1000000;                      % queue size (Bytes)
b = 10^-5;                        % bit error rate
n_values = [10, 20, 30, 40];      % number of VoIP flows
N = 20;                           % number of runs
P = 100000;                       % number of packets (
    stopping criterion)
alfa = 0.1;                       % 90% confidence interval

% Variables to store the simulation results
PLdata = zeros(N, length(n_values));
PLVoIP = zeros(N, length(n_values));
APDdata = zeros(N, length(n_values));
APDVoIP = zeros(N, length(n_values));
MPDdata = zeros(N, length(n_values));
MPDVoIP = zeros(N, length(n_values));
TT = zeros(N, length(n_values));

% Run the simulation for each value of n
for i = 1:length(n_values)
    n = n_values(i);
    for j = 1:N
        [PLdata(j, i), PLVoIP(j, i), APDdata(j, i),
            APDVoIP(j, i), MPDdata(j, i), MPDVoIP(j, i),
            TT(j, i)] = Sim3A(lambda, C, f, P, n, b);
    end
end

% Calculate mean and confidence intervals
mean_PLdata = mean(PLdata);
mean_PLVoIP = mean(PLVoIP);
mean_APDdata = mean(APDdata);
```

```
mean_APDVoIP = mean(APDVoIP);
mean_MPDdata = mean(MPDdata);
mean_MPDVoIP = mean(MPDVoIP);
mean_TT = mean(TT);

ci_PLdata = norminv(1-alfa/2) * sqrt(var(PLdata) / N);
ci_PLVoIP = norminv(1-alfa/2) * sqrt(var(PLVoIP) / N);
ci_APDdata = norminv(1-alfa/2) * sqrt(var(APDdata) / N);
ci_APDVoIP = norminv(1-alfa/2) * sqrt(var(APDVoIP) / N);
ci_MPDdata = norminv(1-alfa/2) * sqrt(var(MPDdata) / N);
ci_MPDVoIP = norminv(1-alfa/2) * sqrt(var(MPDVoIP) / N);
ci_TT = norminv(1-alfa/2) * sqrt(var(TT) / N);

% Print the results
for i = 1:length(n_values)
    fprintf('Number of VoIP Flows: %d\n', n_values(i));
    fprintf('PLdata: %.4f%%    %.4f%%\n', mean_PLdata(i),
        ci_PLdata(i));
    fprintf('PLVoIP: %.4f%%    %.4f%%\n', mean_PLVoIP(i),
        ci_PLVoIP(i));
    fprintf('APDdata: %.4f ms    %.4f ms\n', mean_APDdata
       (i), ci_APDdata(i));
    fprintf('APDVoIP: %.4f ms    %.4f ms\n', mean_APDVoIP
       (i), ci_APDVoIP(i));
    fprintf('MPDdata: %.4f ms    %.4f ms\n', mean_MPDdata
       (i), ci_MPDdata(i));
    fprintf('MPDVoIP: %.4f ms    %.4f ms\n', mean_MPDVoIP
       (i), ci_MPDVoIP(i));
    fprintf('TT: %.4f Mbps    %.4f Mbps\n', mean_TT(i),
       ci_TT(i));
    fprintf('\n');
end
```

2.2 Exercício 2.b.

2.2.1 Código

```
%% Exercise 2.b.

% Parameters
lambda = 1500;                  % packet rate (packets/
   sec)
C = 10;                         % link bandwidth (Mbps)
f = 1000000;                    % queue size (Bytes)
b = 10^-5;                      % bit error rate
n_values = [10, 20, 30, 40];    % number of VoIP flows
N = 20;                         % number of runs
```

```
P = 100000;                         % number of packets (
    stopping criterion)
alfa = 0.1;                         % 90% confidence interval

% Variables to store the simulation results
PLdata = zeros(N, length(n_values));
PLVoIP = zeros(N, length(n_values));

% Run the simulation for each value of n
for i = 1:length(n_values)
    n = n_values(i);
    for j = 1:N
        [PLdata(j, i), PLVoIP(j, i), ~, ~, ~, ~, ~] =
            Sim3A(lambda, C, f, P, n, b);
    end
end

% Calculate mean and confidence intervals
mean_PLdata = mean(PLdata);
mean_PLVoIP = mean(PLVoIP);

ci_PLdata = norminv(1-alfa/2) * sqrt(var(PLdata) / N);
ci_PLVoIP = norminv(1-alfa/2) * sqrt(var(PLVoIP) / N);

% Plotting the results for packet loss of data packets
figure;
bar(n_values, mean_PLdata);
hold on;
errorbar(n_values, mean_PLdata, ci_PLdata, '.');
title('Average Packet Loss (Data)');
xlabel('Number of VoIP Flows');
ylabel('Packet Loss (%)');
grid on;
hold off;

% Plotting the results for packet loss of VoIP packets
figure;
bar(n_values, mean_PLVoIP);
hold on;
errorbar(n_values, mean_PLVoIP, ci_PLVoIP, '.');
title('Average Packet Loss (VoIP)');
xlabel('Number of VoIP Flows');
ylabel('Packet Loss (%)');
grid on;
hold off;
```

2.2.2 Resultados e Conclusões

Os resultados apresentados nas figuras (2.1 e 2.2), revelam uma diferença clara nas perdas de pacotes entre os fluxos de pacotes de dados e de VoIP, com uma maior percentagem de perda de pacotes para os pacotes de dados em comparação com os pacotes de VoIP. Esta discrepância pode ser atribuída ao tamanho dos pacotes. Dado que a taxa de erro de bits (BER) se mantém constante ao longo da simulação, é esperado que os pacotes de dados, que são substancialmente maiores do que os pacotes de VoIP, estejam mais suscetíveis a corrupção durante a transmissão.

A perda de pacotes para dados é cerca de 4,5%, enquanto para VoIP é cerca de 0,9%, o que está de acordo com a expectativa, visto que o tamanho médio dos pacotes de dados é aproximadamente 5 vezes maior do que o dos pacotes de VoIP. Como resultado, a perda de pacotes de dados é aproximadamente 5 vezes superior à dos pacotes de VoIP, refletindo diretamente essa diferença no tamanho dos pacotes.

Outra observação importante é a constância relativa das perdas de pacotes ao longo de diferentes números de fluxos de VoIP, tanto para os pacotes de dados quanto para VoIP. No entanto, os valores de perda mostram pequenas flutuações, sugerindo que o sistema está a operar perto do limite de saturação da sua capacidade. Este comportamento indica que, à medida que a taxa de chegada de pacotes aumenta, o sistema aproxima-se do ponto onde a fila começa a congestioná-lo, afetando assim o desempenho global.

Por fim, as margens de erro representadas pelos intervalos de confiança reforçam a estabilidade dos resultados obtidos, mostrando que, apesar das flutuações, a tendência geral se mantém constante. Isso sugere que a inclusão da taxa de erro de bits (BER) no modelo, além da análise dos pacotes corrompidos, oferece uma representação mais precisa das perdas de pacotes, especialmente quando o sistema se aproxima da sua capacidade máxima.

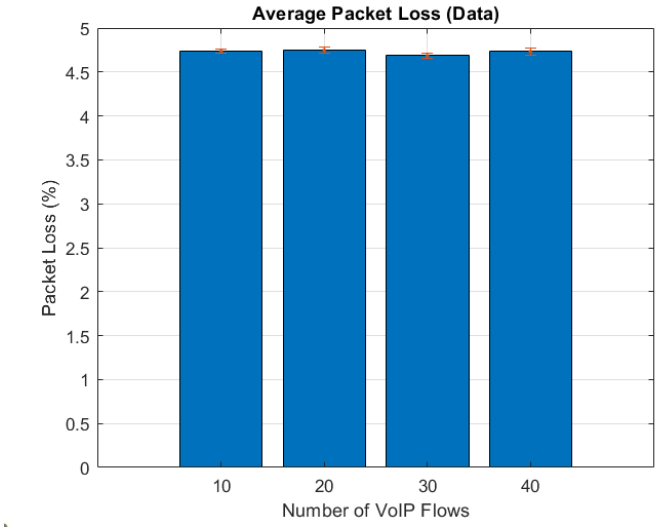

Figura 2.1: Perda média de pacotes para pacotes de dados.

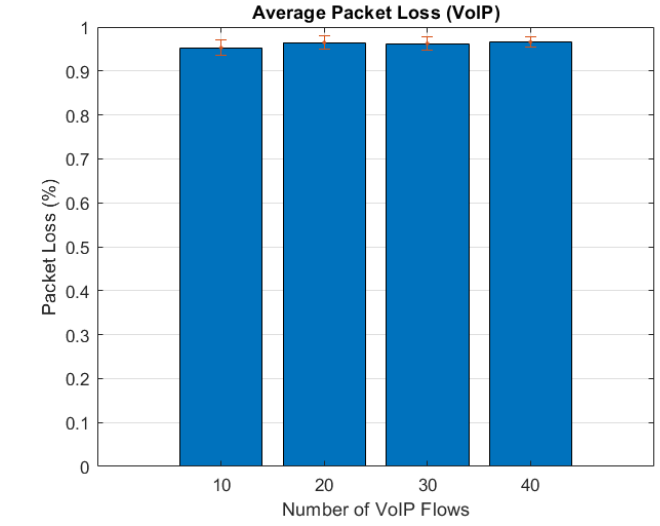

Figura 2.2: Perda média de pacotes para pacotes de VoIP.

2.3 Exercício 2.c.

2.3.1 Código

```
%% Exercise 2.c.

% Parameters
```

```
lambda = 1500;                    % packet rate (packets/
    sec)
C = 10;                           % link bandwidth (Mbps)
f = 1000000;                      % queue size (Bytes)
b = 10^-5;                        % bit error rate
n_values = [10, 20, 30, 40];      % number of VoIP flows
N = 20;                           % number of runs
P = 100000;                       % number of packets (
    stopping criterion)
alfa = 0.1;                       % 90% confidence interval

% Variables to store the simulation results
APDdata = zeros(N, length(n_values));
APDVoIP = zeros(N, length(n_values));

% Run the simulation for each value of n
for i = 1:length(n_values)
    n = n_values(i);
    for j = 1:N
        [~, ~, APDdata(j, i), APDVoIP(j, i), ~, ~, ~] =
            Sim3A(lambda, C, f, P, n, b);
    end
end

% Calculate mean and confidence intervals
mean_APDdata = mean(APDdata);
mean_APDVoIP = mean(APDVoIP);

ci_APDdata = norminv(1-alfa/2) * sqrt(var(APDdata) / N);
ci_APDVoIP = norminv(1-alfa/2) * sqrt(var(APDVoIP) / N);

% Plotting the results for average packet delay of data
    packets
figure;
bar(n_values, mean_APDdata);
hold on;
errorbar(n_values, mean_APDdata, ci_APDdata, '.');
title('Average Packet Delay (Data)');
xlabel('Number of VoIP Flows');
ylabel('Average Delay (ms)');
grid on;
hold off;

% Plotting the results for average packet delay of VoIP
    packets
figure;
bar(n_values, mean_APDVoIP);
hold on;
errorbar(n_values, mean_APDVoIP, ci_APDVoIP, '.');
```

```
48 title('Average Packet Delay (VoIP)');
49 xlabel('Number of VoIP Flows');
50 ylabel('Average Delay (ms)');
51 grid on;
52 hold off;
```

2.3.2 Resultados e Conclusões

Analisando as figuras 2.3 e 2.4, podemos observar que os pacotes de dados apresentam um atraso médio relativamente maior (porém pouco) em comparação com os pacotes VoIP. Este comportamento deve-se ao facto de os pacotes VoIP terem tamanhos mais pequenos e uniformemente distribuídos entre 110 e 130 bytes, enquanto os pacotes de dados variam significativamente em tamanho, com valores entre 64 e 1518 bytes e uma média de 620 bytes. Como esperado, pacotes maiores tendem a sofrer maiores atrasos, o que explica porque os pacotes de dados experimentam um atraso médio superior ao dos pacotes VoIP.

Além disso, observamos que à medida que o número de fluxos VoIP aumenta, o atraso médio de ambos os tipos de pacotes (dados e VoIP) também aumenta progressivamente. Este comportamento pode ser explicado pelo facto de que, à medida que o número de fluxos VoIP cresce, o sistema aproxima-se da sua capacidade máxima de processamento e transmissão. Como resultado, o tempo de espera nas filas aumenta, o que leva a um aumento no atraso médio dos pacotes, especialmente em cenários com tráfego intenso.

Esta tendência de aumento no atraso com o crescimento dos fluxos VoIP é um indicador claro de que o sistema está a aproximar-se do limiar da saturação. Embora os valores de atraso aumentem de forma consistente, a variação observada nos intervalos de confiança permanece relativamente estável, indicando que os resultados são confiáveis e apresentam uma margem de erro aceitável. No entanto, em condições de carga mais elevada, é provável que o desempenho do sistema se degrade ainda mais, resultando em atrasos significativamente maiores.

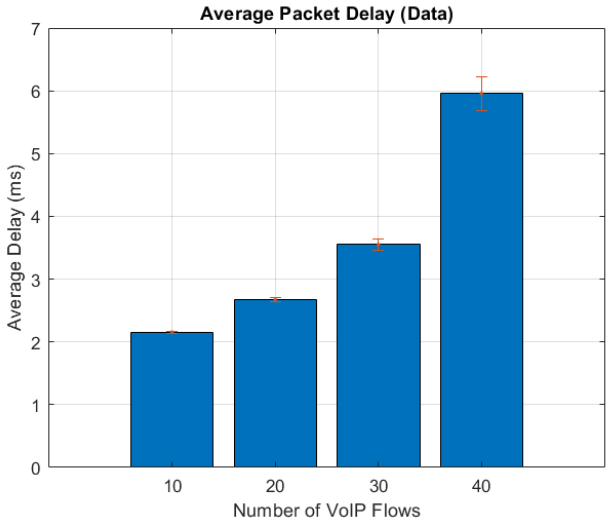

Figura 2.3: Atraso médio de pacotes para pacotes de dados.

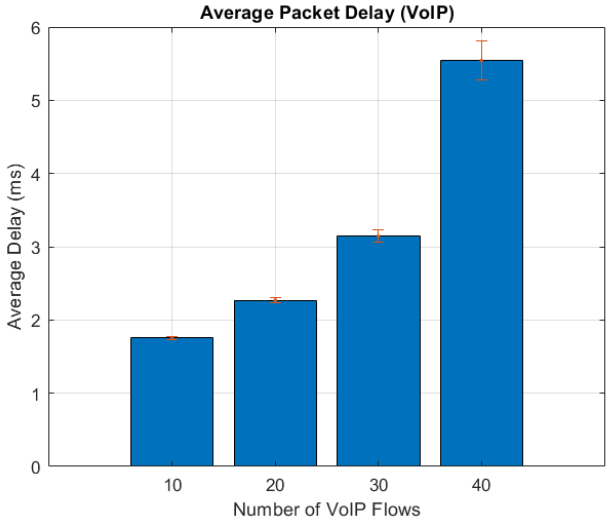

Figura 2.4: Atraso médio de pacotes para pacotes de VoIP.

2.4 Exercício 2.d.

2.4.1 Código

```
1 %% Exercise 2.d.
2
3 % Parameters
```

```
lambda = 1500;                    % packet rate (packets/
    sec)
C = 10;                           % link bandwidth (Mbps)
f = 1000000;                      % queue size (Bytes)
b = 10^-5;                        % bit error rate
n_values = [10, 20, 30, 40];      % number of VoIP flows
N = 20;                           % number of runs
P = 100000;                       % number of packets (
    stopping criterion)
alfa = 0.1;                       % 90% confidence interval

% Variables to store the simulation results
MPDdata = zeros(N, length(n_values));
MPDVoIP = zeros(N, length(n_values));

% Run the simulation for each value of n
for i = 1:length(n_values)
    n = n_values(i);
    for j = 1:N
        [~, ~, ~, ~, MPDdata(j, i), MPDVoIP(j, i), ~] =
            Sim3A(lambda, C, f, P, n, b);
    end
end

% Calculate mean and confidence intervals
mean_MPDdata = mean(MPDdata);
mean_MPDVoIP = mean(MPDVoIP);

ci_MPDdata = norminv(1-alfa/2) * sqrt(var(MPDdata) / N);
ci_MPDVoIP = norminv(1-alfa/2) * sqrt(var(MPDVoIP) / N);

% Plotting the results for maximum packet delay of data
    packets
figure;
bar(n_values, mean_MPDdata);
hold on;
errorbar(n_values, mean_MPDdata, ci_MPDdata, '.');
title('Maximum Packet Delay (Data)');
xlabel('Number of VoIP Flows');
ylabel('Maximum Delay (ms)');
grid on;
hold off;

% Plotting the results for maximum packet delay of VoIP
    packets
figure;
bar(n_values, mean_MPDVoIP);
hold on;
errorbar(n_values, mean_MPDVoIP, ci_MPDVoIP, '.');
```

```
48 title('Maximum Packet Delay (VoIP)');
49 xlabel('Number of VoIP Flows');
50 ylabel('Maximum Delay (ms)');
51 grid on;
52 hold off;
```

2.4.2 Resultados e Conclusões

Analisando as figuras 2.5 e 2.6, que mostram o atraso máximo dos pacotes para serviços de dados e VoIP em função do número de fluxos VoIP, observamos um aumento do atraso conforme o número de fluxos aumenta.

Na figura 2.5, podemos ver que o atraso máximo dos pacotes de dados cresce à medida que o número de fluxos VoIP aumenta. Com 10 fluxos, o atraso máximo está por volta de 20 ms, enquanto com 40 fluxos VoIP o atraso máximo atinge cerca de 35 ms. Esse comportamento é esperado, dado que o aumento do tráfego de VoIP faz com que o sistema opere mais próximo de sua capacidade máxima, resultando num maior atraso para todos os pacotes.

A figura 2.6 apresenta resultados semelhantes para o serviço VoIP, com o atraso máximo aumentando de aproximadamente 20 ms com 10 fluxos para quase 37 ms com 40 fluxos. Isto ocorre porque, à medida que o número de fluxos aumenta, a competição por recursos na rede também cresce, afetando tanto os pacotes de dados quanto os pacotes VoIP.

Comparando os atrasos máximos com os atrasos médios dos pacotes, observamos que o comportamento de ambos é semelhante, com o atraso máximo sempre sendo maior que o atraso médio, como esperado. À medida que o número de fluxos aumenta, o aumento do atraso médio acompanha o aumento do atraso máximo, refletindo a degradação geral da performance da rede.

É importante notar que os intervalos de confiança em ambos os gráficos se mantêm relativamente estáveis, o que indica que os resultados são consistentes e confiáveis.

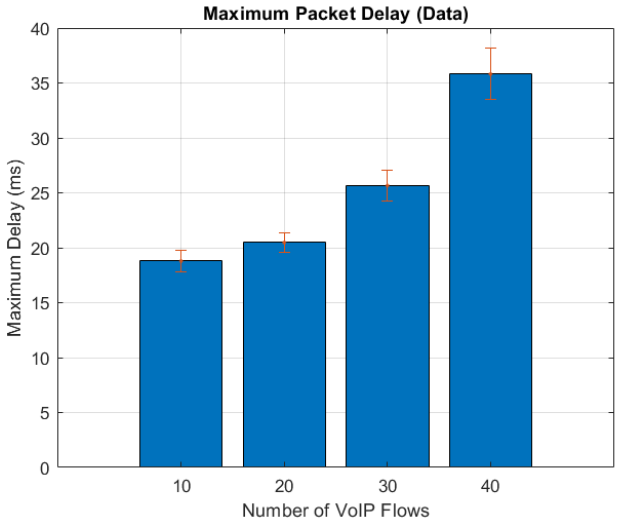

Figura 2.5: Atraso máximo dos pacotes de dados em função do número de fluxos VoIP

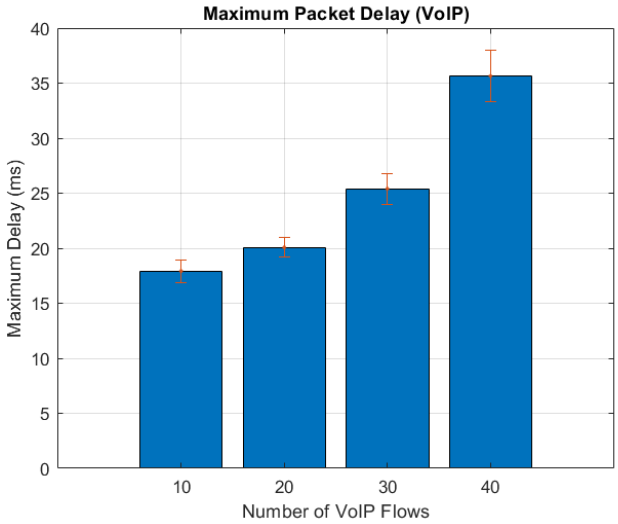

Figura 2.6: Atraso máximo dos pacotes VoIP em função do número de fluxos VoIP

2.5 Exercício 2.e.

2.5.1 Código

```
1 %% Exercise 2.e.
```

```
2
3  % Parameters
4  lambda = 1500;                    % packet rate (packets/
       sec)
5  C = 10;                           % link bandwidth (Mbps)
6  f = 1000000;                      % queue size (Bytes)
7  b = 10^-5;                        % bit error rate
8  n_values = [10, 20, 30, 40];      % number of VoIP flows
9  N = 20;                           % number of runs
10 P = 100000;                       % number of packets (
       stopping criterion)
11 alfa = 0.1;                       % 90% confidence interval
12
13 % Variables to store the simulation results
14 TT = zeros(N, length(n_values));
15
16 % Run the simulation for each value of n
17 for i = 1:length(n_values)
18     n = n_values(i);
19     for j = 1:N
20         [~, ~, ~, ~, ~, ~, TT(j, i)] = Sim3A(lambda, C, f
               , P, n, b);
21     end
22 end
23
24 % Calculate mean and confidence intervals
25 mean_TT = mean(TT);
26 ci_TT = norminv(1-alfa/2) * sqrt(var(TT) / N);
27
28 % Plotting the results for total throughput
29 figure;
30 bar(n_values, mean_TT);
31 hold on;
32 errorbar(n_values, mean_TT, ci_TT, '.');
33 title('Total Throughput');
34 xlabel('Number of VoIP Flows');
35 ylabel('Throughput (Mbps)');
36 grid on;
37 hold off;
```

2.5.2 Resultados e Conclusões

A Figura 2.7 apresenta os resultados da simulação em relação ao throughput total em função do número de fluxos VoIP, juntamente com os intervalos de confiança representados por barras de erro.

Observa-se que o throughput total aumenta conforme o número de fluxos VoIP cresce. Com 10 fluxos VoIP, o throughput total é de aproximadamente 7,4 Mbps, e à medida que o número de fluxos aumenta para 40, o throughput

chega a cerca de 8,6 Mbps. Esse aumento é esperado, uma vez que mais fluxos VoIP resultam num maior número de pacotes transmitidos pela rede, o que, por sua vez, eleva o volume de dados processados.

Os intervalos de confiança mostram que os valores são bastante consistentes ao longo dos diferentes cenários, com pouca variabilidade. Isso indica que os resultados da simulação são estáveis e confiáveis. O aumento no throughput está relacionado diretamente com a maior ocupação da largura de banda disponível à medida que mais fluxos VoIP são adicionados.

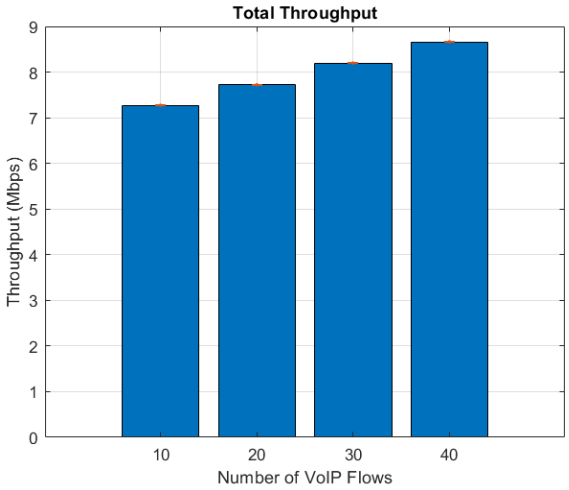

Figura 2.7: Throughput total em função do número de fluxos VoIP

2.6 Exercício 2.f.

2.6.1 Código

```
%% Exercise 2.f.

% Parameters
C = 10;                 % Link bandwidth in Mbps
lambda = 1500;          % packets per second for data

% Calculate B_avg
prob_64 = 0.19;
prob_110 = 0.23;
prob_1518 = 0.17;
mean_aux2 = mean([65:109 111:1517]);
B_avg = prob_64 * 64 + prob_110 * 110 + prob_1518 * 1518
    + (1 - prob_64 - prob_110 - prob_1518) * mean_aux2;

n_values = [10, 20, 30, 40]; % Number of VoIP flows
```

```
15
16 % Calculate theoretical throughput for each n
17 for i = 1:length(n_values)
18     n = n_values(i);
19     TT_theoretical = min(C, lambda * (B_avg * 8) / 1e6);
20     fprintf('Theoretical throughput for %d VoIP flows is:
           %.2f Mbps\n', n, TT_theoretical);
21 end
```

2.6.2 Código - Explicação

O código apresentado calcula o *throughput* teórico para diferentes números de fluxos de VoIP (10, 20, 30 e 40) com base nas características de largura de banda e distribuição de tamanhos dos pacotes. Abaixo, descrevemos os principais passos e fórmulas utilizadas. Primeiro, foram definidos os parâmetros principais:

- `C` é a largura de banda do link, fixada em 10 Mbps.
- `lambda` representa a taxa de chegada de pacotes de dados, que é de 1500 pacotes por segundo.

Em seguida, foi calculado o tamanho médio do pacote de dados (`B_avg`), considerando a distribuição de tamanhos fornecida pelo simulador `Sim1`. Esta distribuição é definida pelas seguintes probabilidades:

- `prob_64` = 0.19, correspondendo aos pacotes de 64 bytes.
- `prob_110` = 0.23, correspondendo aos pacotes de 110 bytes.
- `prob_1518` = 0.17, correspondendo aos pacotes de 1518 bytes.
- A média dos tamanhos de pacotes restantes (`mean_aux2`) é calculada como a média dos pacotes de tamanho entre 65 e 109 bytes e entre 111 e 1517 bytes.

Com o tamanho médio do pacote calculado, podemos determinar o *throughput* teórico (`TT_theoretical`) usando a seguinte fórmula:

$$\texttt{TT\_theoretical} = \min\left(C, \frac{\lambda \times (\texttt{B\_avg} \times 8)}{10^6}\right)$$

Aqui, multiplicamos o `B_avg` por 8 para converter o tamanho de bytes para bits e dividimos por 10^6 para obter o resultado em Mbps. O operador `min` garante que o *throughput* teórico não exceda a capacidade de banda do link (`C`).

2.6.3 Resultados e Conclusões

Nos resultados obtidos para o exercício 2.a. (Figura 2.8), observamos que o *throughput* total aumenta progressivamente com o número de fluxos VoIP, variando de 7.256 Mbps para 10 fluxos até 8.6729 Mbps para 40 fluxos. Este aumento no *throughput* é esperado, pois um maior número de fluxos VoIP leva a um aumento no tráfego total da rede.

Por outro lado, ao analisarmos os valores de *throughput* teórico obtidos no exercício 2.f. (Figura 2.9), verificamos que o *throughput* se mantém constante em 7.44 Mbps, independentemente do número de fluxos VoIP. Este valor fixo é uma limitação imposta pela largura de banda do link e pelo tamanho médio dos pacotes estimado. A teoria considera o valor máximo de *throughput* possível sem levar em conta a influência dinâmica do aumento dos fluxos VoIP, como acontece na simulação.

Comparando os valores simulados com o valor teórico, notamos que o valor teórico aproxima-se mais do valor simulado para 10 e 20 fluxos e começa a subestimar o *throughput* conforme o número de fluxos VoIP aumenta. Este comportamento indica que o modelo teórico não considera completamente os efeitos de congestionamento, que se tornam mais relevantes à medida que a ocupação da rede aumenta, causando uma redução no *throughput* efetivo nos cenários de maior carga.

```
>> ex2a
Number of VoIP Flows: 10
PLdata: 4.7322% ± 0.0209%
PLVoIP: 0.9412% ± 0.0223%
APDdata: 2.1589 ms ± 0.0223 ms
APDVoIP: 1.7587 ms ± 0.0198 ms
MPDdata: 18.4260 ms ± 0.9422 ms
MPDVoIP: 17.6126 ms ± 0.9612 ms
TT: 7.2562 Mbps ± 0.0143 Mbps

Number of VoIP Flows: 20
PLdata: 4.7342% ± 0.0416%
PLVoIP: 0.9314% ± 0.0172%
APDdata: 2.6704 ms ± 0.0321 ms
APDVoIP: 2.2688 ms ± 0.0297 ms
MPDdata: 20.3731 ms ± 0.7652 ms
MPDVoIP: 20.1034 ms ± 0.8147 ms
TT: 7.7320 Mbps ± 0.0175 Mbps

Number of VoIP Flows: 30
PLdata: 4.7262% ± 0.0371%
PLVoIP: 0.9464% ± 0.0167%
APDdata: 3.6593 ms ± 0.0896 ms
APDVoIP: 3.2499 ms ± 0.0853 ms
MPDdata: 25.7754 ms ± 1.4150 ms
MPDVoIP: 25.5617 ms ± 1.4126 ms
TT: 8.2165 Mbps ± 0.0146 Mbps

Number of VoIP Flows: 40
PLdata: 4.7780% ± 0.0331%
PLVoIP: 0.9580% ± 0.0150%
APDdata: 5.8140 ms ± 0.2036 ms
APDVoIP: 5.4048 ms ± 0.2014 ms
MPDdata: 35.5349 ms ± 2.2049 ms
MPDVoIP: 35.3441 ms ± 2.1796 ms
TT: 8.6729 Mbps ± 0.0199 Mbps
```

Figura 2.8: Resultados simulados do *throughput*, atraso e perda de pacotes para diferentes fluxos VoIP.

```
>> ex2f
Theoretical throughput for 10 VoIP flows is: 7.44 Mbps
Theoretical throughput for 20 VoIP flows is: 7.44 Mbps
Theoretical throughput for 30 VoIP flows is: 7.44 Mbps
Theoretical throughput for 40 VoIP flows is: 7.44 Mbps
```

Figura 2.9: *Throughput* teórico para diferentes fluxos VoIP.

Capítulo 3

Task 3

3.1 Exercício 3.a.

3.1.1 Código

```
%% Exercise 3.a.

% Parameters
lambda = 1500;  % pps (packets/sec)
C = 10;         % Link bandwidth (Mbps)
f = 10000;      % Queue size (Bytes)
P = 100000;     % Stop criterion (number of packets)
nVoIPs = [10, 20, 30, 40];  % VoIP flows
N = 20;         % Number of simulations
alpha = 0.1;    % for 90% confidence interval

% Results arrays
PLdata_results = zeros(N, length(nVoIPs));
PLVoIP_results = zeros(N, length(nVoIPs));
APDdata_results = zeros(N, length(nVoIPs));
APDVoIP_results = zeros(N, length(nVoIPs));

% Simulation loop
for i = 1:length(nVoIPs)
    for j = 1:N
        % Run Sim3 for each VoIP flow count
        [PLdata_results(j,i), PLVoIP_results(j,i),
           APDdata_results(j,i), APDVoIP_results(j,i)] =
            ...
            Sim3(lambda, C, f, P, nVoIPs(i));
    end
end

% Calculate mean and confidence intervals
mean_PLdata = mean(PLdata_results);
```

```
mean_PLVoIP = mean(PLVoIP_results);
mean_APDdata = mean(APDdata_results);
mean_APDVoIP = mean(APDVoIP_results);

% Standard error and 90% confidence interval (Z = 1.645)
Z = norminv(1 - alpha / 2);
ci_PLdata = Z * std(PLdata_results) / sqrt(N);
ci_PLVoIP = Z * std(PLVoIP_results) / sqrt(N);
ci_APDdata = Z * std(APDdata_results) / sqrt(N);
ci_APDVoIP = Z * std(APDVoIP_results) / sqrt(N);

% Plot Average Packet Delay for Data and VoIP with error bars
figure;
hold on; grid on;
b = bar(nVoIPs, [mean_APDdata; mean_APDVoIP]', 'grouped'); % Plot bar chart
% Get x-coordinates of each bar for accurate error bar placement
xData = b(1).XEndPoints;  % Data bar positions
errorbar(xData, mean_APDdata, ci_APDdata, 'k.', 'linestyle', 'none'); % Data errors centered
xData = b(2).XEndPoints;  % VoIP bar positions
errorbar(xData, mean_APDVoIP, ci_APDVoIP, 'k.', 'linestyle', 'none'); % VoIP errors centered
xlabel('Number of VoIP Flows');
ylabel('Average Packet Delay (ms)');
title('Average Packet Delay for Data and VoIP');
legend('Data', 'VoIP');
hold off;

% Plot Packet Loss for Data and VoIP with error bars
figure;
hold on; grid on;
b = bar(nVoIPs, [mean_PLdata; mean_PLVoIP]', 'grouped'); % Plot bar chart
% Get x-coordinates of each bar for accurate error bar placement
xData = b(1).XEndPoints;  % Data bar positions
errorbar(xData, mean_PLdata, ci_PLdata, 'k.', 'linestyle', 'none'); % Data errors centered
xData = b(2).XEndPoints;  % VoIP bar positions
errorbar(xData, mean_PLVoIP, ci_PLVoIP, 'k.', 'linestyle', 'none'); % VoIP errors centered
xlabel('Number of VoIP Flows');
ylabel('Packet Loss (%)');
title('Packet Loss for Data and VoIP');
legend('Data', 'VoIP');
hold off;
```

3.1.2 Resultados e Conclusões

No exercício 3.a, foi utilizado o `Sim3` para estimar o atraso médio de pacotes e a perda de pacotes para os dois tipos de serviços: dados e VoIP. No `Sim3`, ambos os serviços são multiplexados estatisticamente numa fila única em regime FIFO (First In, First Out). Os parâmetros simulados incluem o número de fluxos VoIP que varia entre 10 e 40, com uma taxa de chegada de pacotes de dados de 1500 pps, uma largura de banda do link de 10 Mbps e uma fila com capacidade de 10.000 Bytes.

Relativamente ao atraso médio dos pacotes:

- A figura 3.1 revela que o atraso médio dos pacotes aumenta progressivamente com o número de fluxos VoIP. Isto é esperado, pois o aumento de fluxos leva a uma maior competição por espaço na fila e pelo acesso ao canal de transmissão.
- Comparando os serviços, observou-se que os pacotes de dados apresentam um atraso médio maior do que os pacotes VoIP. Este comportamento pode ser atribuído ao fato de que os pacotes de dados são frequentemente maiores e a ausência de prioridade significa que os pacotes VoIP têm o mesmo tratamento na fila.

Relativamente à perda de pacotes:

- A figura 3.2 mostra a perda de pacotes para ambos os serviços. À medida que o número de fluxos VoIP aumenta, a perda de pacotes para o serviço de dados torna-se significativamente maior. Este resultado é consistente com a sobrecarga na fila, onde pacotes adicionais de dados são descartados quando a fila atinge a sua capacidade máxima.
- O serviço VoIP, embora também afetado, apresenta uma perda de pacotes relativamente menor, mas ainda assim crescente com o aumento de fluxos.

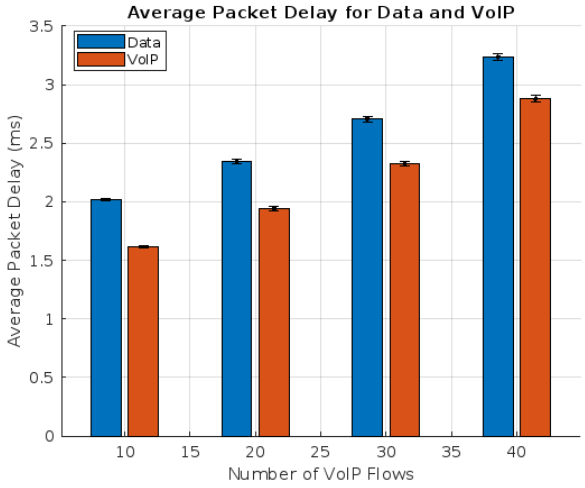

Figura 3.1: Atraso médio de pacotes para dados e VoIP

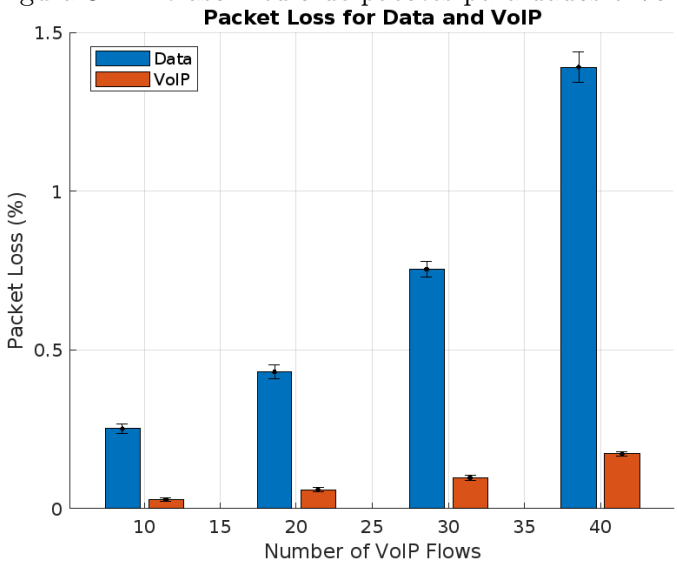

Figura 3.2: Perda média de pacotes para dados e VoIP

3.2 Exercício 3.b.

3.2.1 Código

```
%% Exercise 3.b.

% Parameters
lambda = 1500;   % packets per second for data
C = 10;          % Link bandwidth in Mbps
f = 10000;       % Queue size in Bytes
```

```
P = 100000;      % Stopping criterion (number of packets)
nVoIPs = [10, 20, 30, 40];  % Number of VoIP flows
N = 20;          % Number of simulations
alpha = 0.1;     % for 90% confidence interval

% Results arrays for data and VoIP metrics
PLdata_results = zeros(N, length(nVoIPs));
PLVoIP_results = zeros(N, length(nVoIPs));
APDdata_results = zeros(N, length(nVoIPs));
APDVoIP_results = zeros(N, length(nVoIPs));

% Simulation loop
for i = 1:length(nVoIPs)
    for j = 1:N
        % Run Sim4 for each VoIP flow count
        [PLdata_results(j,i), PLVoIP_results(j,i),
           APDdata_results(j,i), APDVoIP_results(j,i)] =
            ...
            Sim4(lambda, C, f, P, nVoIPs(i));
    end
end

% Calculate mean and confidence intervals
mean_PLdata = mean(PLdata_results);
mean_PLVoIP = mean(PLVoIP_results);
mean_APDdata = mean(APDdata_results);
mean_APDVoIP = mean(APDVoIP_results);

% Calculate 90% confidence intervals (Z = 1.645 for 90%
   confidence)
Z = norminv(1 - alpha / 2);
ci_PLdata = Z * std(PLdata_results) / sqrt(N);
ci_PLVoIP = Z * std(PLVoIP_results) / sqrt(N);
ci_APDdata = Z * std(APDdata_results) / sqrt(N);
ci_APDVoIP = Z * std(APDVoIP_results) / sqrt(N);

% Plot Average Packet Delay for Data and VoIP with error
   bars
figure;
hold on; grid on;
b = bar(nVoIPs, [mean_APDdata; mean_APDVoIP]', 'grouped')
   ;
% Get x-coordinates for error bars on each group
xData = b(1).XEndPoints;  % Data bar positions
errorbar(xData, mean_APDdata, ci_APDdata, 'k.', '
   linestyle', 'none');
xData = b(2).XEndPoints;  % VoIP bar positions
errorbar(xData, mean_APDVoIP, ci_APDVoIP, 'k.', '
   linestyle', 'none');
```

```
49 xlabel('Number of VoIP Flows');
50 ylabel('Average Packet Delay (ms)');
51 title('Average Packet Delay for Data and VoIP with VoIP
       Priority');
52 legend('Data', 'VoIP');
53 hold off;
54
55 % Plot Packet Loss for Data and VoIP with error bars
56 figure;
57 hold on; grid on;
58 b = bar(nVoIPs, [mean_PLdata; mean_PLVoIP]', 'grouped');
59 % Get x-coordinates for error bars on each group
60 xData = b(1).XEndPoints;  % Data bar positions
61 errorbar(xData, mean_PLdata, ci_PLdata, 'k.', 'linestyle'
       , 'none');
62 xData = b(2).XEndPoints;  % VoIP bar positions
63 errorbar(xData, mean_PLVoIP, ci_PLVoIP, 'k.', 'linestyle'
       , 'none');
64 xlabel('Number of VoIP Flows');
65 ylabel('Packet Loss (%)');
66 title('Packet Loss for Data and VoIP with VoIP Priority')
       ;
67 legend('Data', 'VoIP');
68 hold off;
```

3.2.2 Resultados e Conclusões

No exercício 3.b, utilizou-se o `Sim4`, onde o serviço VoIP tem prioridade sobre o serviço de dados, para avaliar os mesmos parâmetros de desempenho do exercício 3.a. Esta configuração permite observar o impacto de priorizar VoIP numa rede que também serve pacotes de dados.

Relativamente ao atraso médio dos pacotes:

- A figura 3.3 ilustra que o atraso médio dos pacotes de VoIP é significativamente menor comparado ao serviço de dados, com uma tendência a manter-se baixo mesmo com o aumento de fluxos VoIP. Este resultado é esperado devido à prioridade concedida aos pacotes VoIP, que permite que esses pacotes sejam processados antes dos pacotes de dados na fila.

- Por outro lado, o atraso médio dos pacotes de dados aumenta consideravelmente com o número de fluxos VoIP, refletindo o impacto negativo desta priorização nos pacotes de dados, que ficam na fila enquanto os pacotes VoIP são atendidos.

Relativamente à perda de pacotes:

- A figura 3.4 mostra que a perda de pacotes para o serviço VoIP é reduzida em comparação com o exercício 3.a, pois os pacotes VoIP têm maior probabilidade de serem transmitidos sem serem descartados. No entanto, a perda de pacotes de dados aumenta de forma mais acentuada em comparação com o cenário sem prioridade, uma vez que os pacotes de dados são os primeiros a serem descartados quando a fila atinge a sua capacidade máxima.

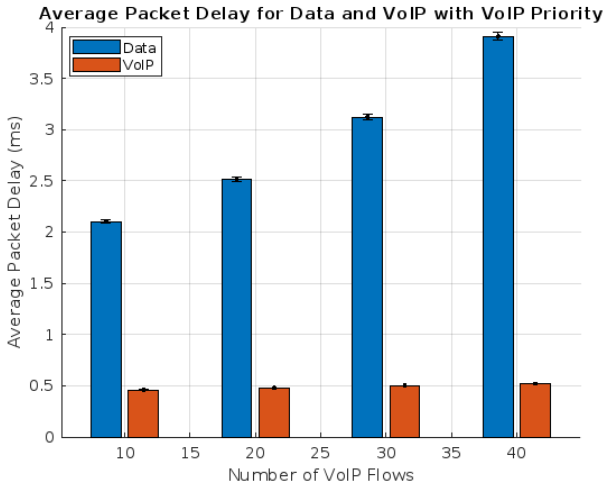

Figura 3.3: Atraso médio de pacotes para dados e VoIP com prioridade

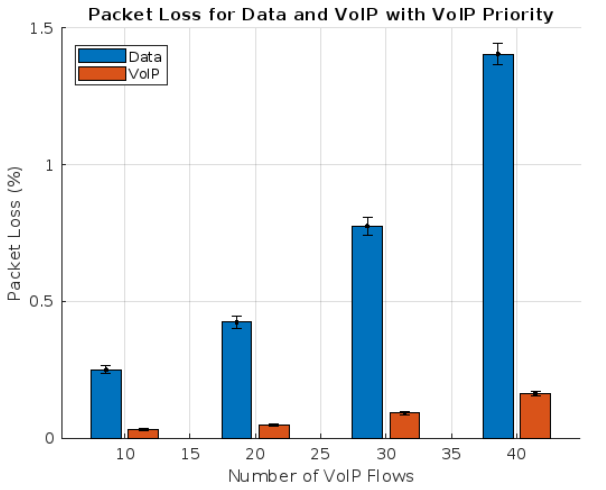

Figura 3.4: Perda média de pacotes para dados e VoIP com prioridade

3.3 Exercício 3.c.

3.3.1 Código

```
function [PLdata, PLVoIP, APDdata, APDVoIP, MPDdata,
    MPDVoIP, TT] = Sim4A(lambda, C, f, P, n, p)
% INPUT PARAMETERS:
%  lambda - packet rate (packets/sec)
%  C      - link bandwidth (Mbps)
%  f      - queue size (Bytes)
%  P      - number of packets (stopping criterium)
%  n      - number of VoIP flows
%  p      - maximum queue occupation percentage for data
    packets
% ... (mais codigo)
% Simulation loop:
while TRANSPACKETS_DATA + TRANSPACKETS_VOIP < P  %
    Stopping criterium
    % ... (mais codigo)
    switch Event
        case ARRIVAL_DATA
            % ... (mais codigo)
            else
                if QUEUEOCCUPATION + PacketSize <= f * (p
                    / 100)
                    QUEUE = [QUEUE; PacketSize, Clock,
                        ARRIVAL_DATA];
                    QUEUEOCCUPATION = QUEUEOCCUPATION +
                        PacketSize;
                else
                    LOSTPACKETS_DATA = LOSTPACKETS_DATA +
                        1;
                end
            end
        case ARRIVAL_VOIP  % If first event is an
            ARRIVAL_VOIP
            % ... (mais codigo)
        case DEPARTURE  % If first event is a DEPARTURE
            % ... (mais codigo)
    end
end
% ... (mais codigo)
```

3.3.2 Diferenças e Justificações

As principais diferenças entre as simulações `Sim4` e `Sim4A`:

- **Parâmetros da Função:**
  Foi adicionado o parâmetro `p` para limitar a ocupação máxima da fila

para pacotes de dados, garantindo a prioridade ao tráfego VoIP em situações de congestionamento. Este parâmetro controla o limite máximo de ocupação da fila para aceitação de pacotes de dados, priorizando pacotes VoIP quando a fila está próxima da capacidade.

- **Algoritmo de Descarte:**
  O algoritmo agora aceita sempre pacotes VoIP, desde que haja espaço, enquanto pacotes de dados só são aceites se a ocupação da fila não ultrapassar `p%` da capacidade total. O ajuste no descarte melhora a qualidade de serviço para VoIP ao limitar a aceitação de pacotes de dados com base na ocupação da fila, permitindo sempre a entrada de pacotes VoIP quando houver espaço.

3.4 Exercício 3.d.

3.4.1 Código

```
%% Exercise 3.d.

% Parameters
lambda = 1500;  % packets per second for data
C = 10;         % Link bandwidth in Mbps
f = 10000;      % Queue size in Bytes
P = 100000;     % Stopping criterion (number of packets)
nVoIPs = [10, 20, 30, 40];  % Number of VoIP flows
N = 20;         % Number of simulations
alpha = 0.1;    % for 90% confidence interval
p = 90;         % Maximum queue occupation percentage for
    data packets

% Results arrays for data and VoIP metrics
PLdata_results = zeros(N, length(nVoIPs));
PLVoIP_results = zeros(N, length(nVoIPs));
APDdata_results = zeros(N, length(nVoIPs));
APDVoIP_results = zeros(N, length(nVoIPs));

% Simulation loop
for i = 1:length(nVoIPs)
    for j = 1:N
        % Run Sim4A for each VoIP flow count
        [PLdata_results(j,i), PLVoIP_results(j,i),
           APDdata_results(j,i), APDVoIP_results(j,i)] =
            ...
            Sim4A(lambda, C, f, P, nVoIPs(i), p);
    end
end

```

```
% Calculate mean and confidence intervals
mean_PLdata = mean(PLdata_results);
mean_PLVoIP = mean(PLVoIP_results);
mean_APDdata = mean(APDdata_results);
mean_APDVoIP = mean(APDVoIP_results);

% Calculate 90% confidence intervals (Z = 1.645 for 90% confidence)
Z = norminv(1 - alpha / 2);
ci_PLdata = Z * std(PLdata_results) / sqrt(N);
ci_PLVoIP = Z * std(PLVoIP_results) / sqrt(N);
ci_APDdata = Z * std(APDdata_results) / sqrt(N);
ci_APDVoIP = Z * std(APDVoIP_results) / sqrt(N);

% Plot Average Packet Delay for Data and VoIP with error bars
figure;
hold on; grid on;
b = bar(nVoIPs, [mean_APDdata; mean_APDVoIP]', 'grouped');
% Get x-coordinates for error bars on each group
xData = b(1).XEndPoints;  % Data bar positions
errorbar(xData, mean_APDdata, ci_APDdata, 'k.', 'linestyle', 'none');
xData = b(2).XEndPoints;  % VoIP bar positions
errorbar(xData, mean_APDVoIP, ci_APDVoIP, 'k.', 'linestyle', 'none');
xlabel('Number of VoIP Flows');
ylabel('Average Packet Delay (ms)');
title('Average Packet Delay for Data and VoIP with 90% Queue Occupation Limit');
legend('Data', 'VoIP');
hold off;

% Plot Packet Loss for Data and VoIP with error bars
figure;
hold on; grid on;
b = bar(nVoIPs, [mean_PLdata; mean_PLVoIP]', 'grouped');
% Get x-coordinates for error bars on each group
xData = b(1).XEndPoints;  % Data bar positions
errorbar(xData, mean_PLdata, ci_PLdata, 'k.', 'linestyle', 'none');
xData = b(2).XEndPoints;  % VoIP bar positions
errorbar(xData, mean_PLVoIP, ci_PLVoIP, 'k.', 'linestyle', 'none');
xlabel('Number of VoIP Flows');
ylabel('Packet Loss (%)');
title('Packet Loss for Data and VoIP with 90% Queue Occupation Limit');
```

```
68 legend('Data', 'VoIP');
69 hold off;
```

3.4.2 Resultados e Conclusões

Neste exercício, foram analisados os atrasos e perdas de pacotes em cenários com limites de ocupação da fila de 90% para pacotes de dados e VoIP. Fazendo uso da função `Sim4A`, com o parâmetro de ocupação de fila limitado, os resultados destacaram as diferenças no desempenho de pacotes de dados e VoIP. Os resultados observados foram os seguintes:

- **Atraso Médio de Pacotes:**
  Na figura 3.5 observou-se que o atraso médio para pacotes VoIP manteve-se baixo, mesmo com o aumento do número de fluxos VoIP, evidenciando a prioridade de processamento desses pacotes. Já os pacotes de dados apresentaram um aumento de atraso proporcional à quantidade de fluxos VoIP, devido ao impacto da ocupação controlada da fila para dados.

- **Perda de Pacotes:**
  Na figura 3.6 os resultados mostram que a perda de pacotes de dados aumentou com o aumento dos fluxos VoIP. A ocupação de fila limitada a 90% impediu o descarte de pacotes VoIP em congestionamentos, priorizando a sua transmissão e causando uma taxa de perda significativamente menor para esses pacotes em comparação com os dados.

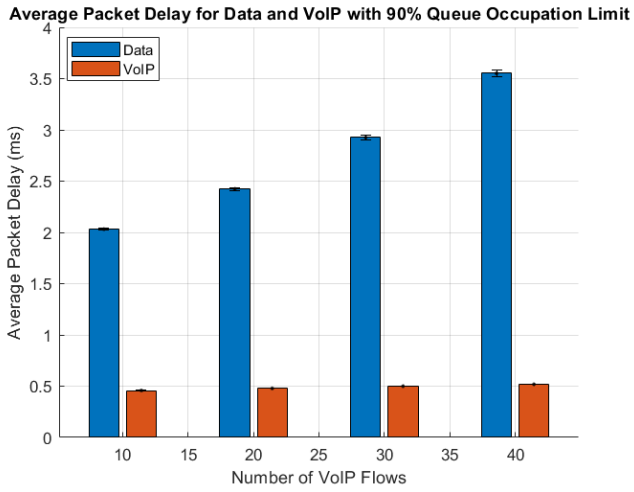

Figura 3.5: Atraso médio de pacotes para dados e VoIP com limite de ocupação da fila de 90%

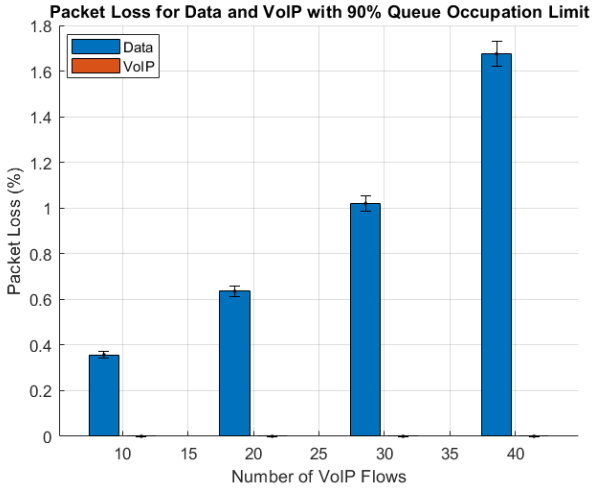

Figura 3.6: Perda média de pacotes para dados e VoIP com limite de ocupação da fila de 90%

3.5 Exercício 3.e.

3.5.1 Código

```
%% Exercise 3.e.

% Parameters
lambda = 1500;  % packets per second for data
C = 10;         % Link bandwidth in Mbps
```

```
f = 10000;       % Queue size in Bytes
P = 100000;      % Stopping criterion (number of packets)
nVoIPs = [10, 20, 30, 40];  % Number of VoIP flows
N = 20;          % Number of simulations
alpha = 0.1;     % for 90% confidence interval
p = 60;          % Maximum queue occupation percentage for
    data packets

% Results arrays for data and VoIP metrics
PLdata_results = zeros(N, length(nVoIPs));
PLVoIP_results = zeros(N, length(nVoIPs));
APDdata_results = zeros(N, length(nVoIPs));
APDVoIP_results = zeros(N, length(nVoIPs));

% Simulation loop
for i = 1:length(nVoIPs)
    for j = 1:N
        % Run Sim4A for each VoIP flow count
        [PLdata_results(j,i), PLVoIP_results(j,i),
           APDdata_results(j,i), APDVoIP_results(j,i)] =
            ...
            Sim4A(lambda, C, f, P, nVoIPs(i), p);
    end
end

% Calculate mean and confidence intervals
mean_PLdata = mean(PLdata_results);
mean_PLVoIP = mean(PLVoIP_results);
mean_APDdata = mean(APDdata_results);
mean_APDVoIP = mean(APDVoIP_results);

% Calculate 90% confidence intervals (Z = 1.645 for 90%
   confidence)
Z = norminv(1 - alpha / 2);
ci_PLdata = Z * std(PLdata_results) / sqrt(N);
ci_PLVoIP = Z * std(PLVoIP_results) / sqrt(N);
ci_APDdata = Z * std(APDdata_results) / sqrt(N);
ci_APDVoIP = Z * std(APDVoIP_results) / sqrt(N);

% Plot Average Packet Delay for Data and VoIP with error
   bars
figure;
hold on; grid on;
b = bar(nVoIPs, [mean_APDdata; mean_APDVoIP]', 'grouped')
   ;
% Get x-coordinates for error bars on each group
xData = b(1).XEndPoints;  % Data bar positions
errorbar(xData, mean_APDdata, ci_APDdata, 'k.', '
   linestyle', 'none');
```

```
xData = b(2).XEndPoints;  % VoIP bar positions
errorbar(xData, mean_APDVoIP, ci_APDVoIP, 'k.', '
    linestyle', 'none');
xlabel('Number of VoIP Flows');
ylabel('Average Packet Delay (ms)');
title('Average Packet Delay for Data and VoIP with 60%
    Queue Occupation Limit');
legend('Data', 'VoIP');
hold off;

% Plot Packet Loss for Data and VoIP with error bars
figure;
hold on; grid on;
b = bar(nVoIPs, [mean_PLdata; mean_PLVoIP]', 'grouped');
% Get x-coordinates for error bars on each group
xData = b(1).XEndPoints;  % Data bar positions
errorbar(xData, mean_PLdata, ci_PLdata, 'k.', 'linestyle'
    , 'none');
xData = b(2).XEndPoints;  % VoIP bar positions
errorbar(xData, mean_PLVoIP, ci_PLVoIP, 'k.', 'linestyle'
    , 'none');
xlabel('Number of VoIP Flows');
ylabel('Packet Loss (%)');
title('Packet Loss for Data and VoIP with 60% Queue
    Occupation Limit');
legend('Data', 'VoIP');
hold off;
```

3.5.2 Resultados e Conclusões

Neste exercício, como o $p = 60\%$, o tempo de espera para pacotes de dados melhora, mas à custa de um aumento na taxa de perda. Para o tráfego VoIP, a priorização contínua manteve o desempenho estável, com baixo atraso e perda de pacotes.

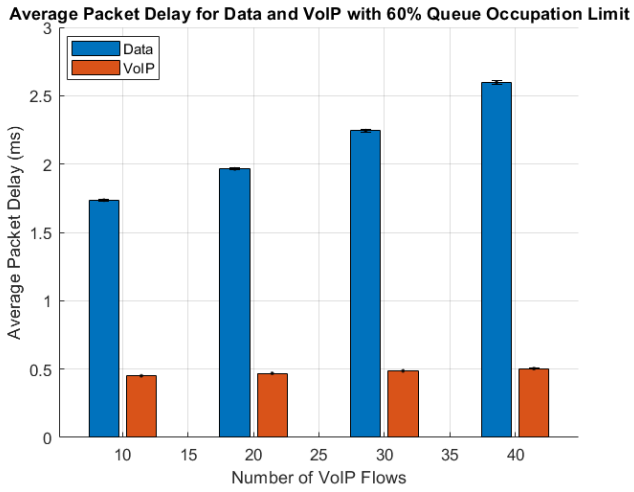

Figura 3.7: Atraso médio de pacotes para dados e VoIP com limite de ocupação da fila de 60%

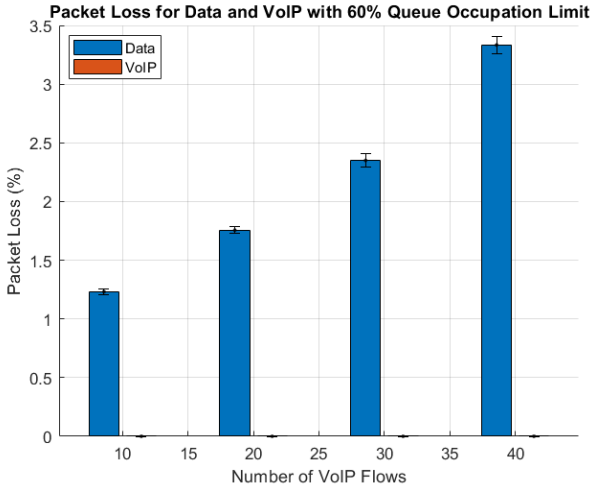

Figura 3.8: Perda de pacotes para dados e VoIP com limite de ocupação da fila de 60%

Autoavaliação

Neste projeto, o esforço foi distribuído igualmente entre os membros do grupo. Cada um contribuiu com 50% do trabalho, colaborando em todas as fases do projeto, desde a implementação das simulações até a análise dos resultados e elaboração do relatório.

- João Monteiro: 50% do trabalho
- João Gaspar: 50% do trabalho