universidade de aveiro
Departamento de Eletrónica, Telecomunicações e Informática

Desempenho e Dimensionamento de Redes

Trabalho Prático 2
2020/2021

P1

Alunos:
Rodrigo Santos, 89180
Alexey Kononov, 89227

Introdução:

Neste relatório será usado em diversas alíneas um simulador de um serviço de reprodução de vídeo com o nome “simulator2”, para informação detalhada sobre a sua construção, código e funcionamento ver o apêndice A, no final do relatório.

Task 2:

a)

```
3 -    R = 10000;
4 -    alfa = 0.1;
5 -    lambda = 100:20:200;
6 -    N = 10;
7 -    p = 20;
8 -    servers = 10;
9 -    S = 100;
10 -   W = 0;
11 -   i = 1;
12 -   for lam = lambda

13
14 -       for it = 1:N
15 -           [results(it),av(it)]= simulator2(lam,p,servers,S,W,R,"movies.txt");
16 -       end
17
18 -       alfa= 0.1; %90% confidence interval%
19 -       mediaHD = mean(results);
20 -       termHD = norminv(1-alfa/2)*sqrt(var(results)/N);
21 -       media4K = mean(av);
22 -       term4K = norminv(1-alfa/2)*sqrt(var(av)/N);
23 -       fprintf('Blocking probability of HD = %.2e +- %.2e\n',mediaHD,termHD)
24 -       fprintf('Blocking probability of 4K = %.2e +- %.2e\n',media4K,term4K)
25 -       dataHD(i)= mediaHD;
26 -       errhighHD(i) = termHD;
27 -       errlowHD(i) = - termHD;
28 -       data4K(i)= media4K;
29 -       errhigh4K(i) = term4K;
30 -       errlow4K(i) = - term4K;
31 -       i=i+1;
32 -   end
33
34 -   data = [dataHD; data4K];
35 -   h = bar(lambda,data)
36 -   hold on
37 -   grid on
38 -   er = errorbar(lambda,dataHD,errlowHD,errhighHD);
39 -   er.Color = [0 0 0];
40 -   er.LineStyle = 'none';
41 -   er2 = errorbar(lambda,data4K,errlow4K,errhigh4K);
42 -   er2.Color = [0 0 0];
43 -   er2.LineStyle = 'none';
44 -   set(h, {'DisplayName'}, {'dataHD','data4K'}')
45 -   legend('Location','northwest')
46 -   hold off
```

Figura 1: Código Matlab Task 2.a

Análise de código:

Nas primeiras (3 - 11) linhas de código são inicializadas as variáveis da simulação conforme o necessário para a Task 2a, de seguida dentro de um ciclo *for,* linhas (14-16) são realizadas as 10 simulações para cada valor de pedidos por hora (*lambda*), no final de cada conjunto de simulações são calculadas as médias dos valores de probabilidade de bloqueio recebidos assim como o erro associado à simulação e armazenados esses valores (19 - 30). Por fim é feito o plot dos valores obtidos.

Resultados e análise:

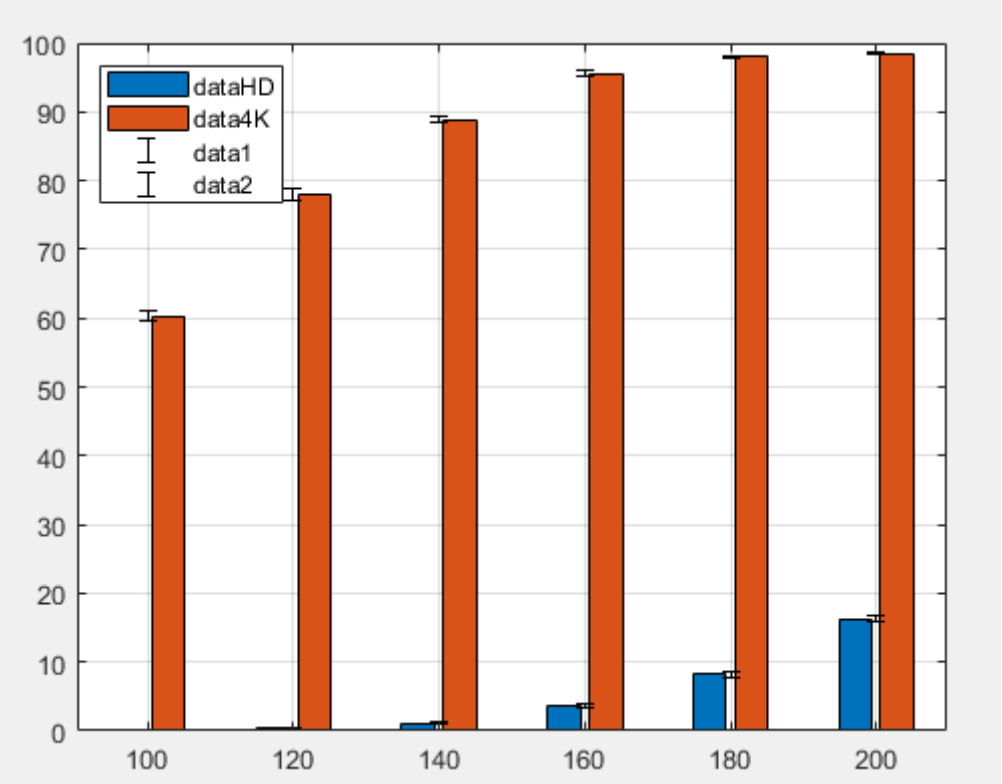

Figura 2: Gráfico de barras obtido

Conforme é possível verificar, tanto para filmes HD como para 4K a probabilidade de bloqueio aumenta à medida que o número de pedidos por hora também aumenta. Tendo no entanto um efeito mais evidente nos pedidos 4K, isto pode ser explicado pelo facto de estes ocuparem mais espaço nos servidores uma vez que necessitam de um throughput mais elevado.

b)

```
3 -    R = 10000;
4 -    alfa = 0.1;
5 -    lambda = 100:20:200;
6 -    N = 10;
7 -    p = 20;
8 -    servers = 10;
9 -    S = 100;
10 -   W = 0;
11 -   i = 1;
12 -   for lam = lambda

14 -       for it = 1:N
15 -           [results(it),av(it)]= simulator2(lam,p,servers,S,W,R,"movies.txt");
16 -           [results2(it),av2(it)]= simulator2(lam,p,4,250,W,R,"movies.txt");
17 -           [results3(it),av3(it)]= simulator2(lam,p,1,1000,W,R,"movies.txt");
18 -       end

20 -       alfa= 0.1; %90% confidence interval%
21 -       mediaHD = mean(results);
22 -       termHD = norminv(1-alfa/2)*sqrt(var(results)/N);
23 -       media2HD = mean(results2);
24 -       term2HD = norminv(1-alfa/2)*sqrt(var(results2)/N);
25 -       media3HD = mean(results3);
26 -       term3HD = norminv(1-alfa/2)*sqrt(var(results3)/N);
27
28 -       media24K = mean(av2);
29 -       term24K = norminv(1-alfa/2)*sqrt(var(av2)/N);
30 -       media34K = mean(av3);
31 -       term34K = norminv(1-alfa/2)*sqrt(var(av3)/N);
32 -       media4K = mean(av);
33 -       term4K = norminv(1-alfa/2)*sqrt(var(av)/N);
34
35 -       dataHD(i)= mediaHD;
36 -       data2HD(i)= media2HD;
37 -       data3HD(i)= media3HD;
38
39 -       data4K(i)= media4K;
40
41 -       data24K(i)= media24K;
42
43 -       data34K(i)= media34K;
44
45 -       i=i+1;
46 -   end
```

Figura 3: Código Matlab Task 2.b

Análise de código:

Tal como na alínea anterior nas primeiras linhas são inicializadas as variáveis da simulação conforme a configuração 1 do simulador (3 - 10). Também de seguida são efetuadas as simulações para os diferentes valores de pedidos por hora no entanto são efetuadas

simulações com as configurações 2 e 3 também para as quais são feitos os ajustes necessários às variáveis (14 - 18). Por fim é feito o plot dos valores obtidos (não presente na figura, mas processo similar ao da alínea anterior).

Resultados e análise:

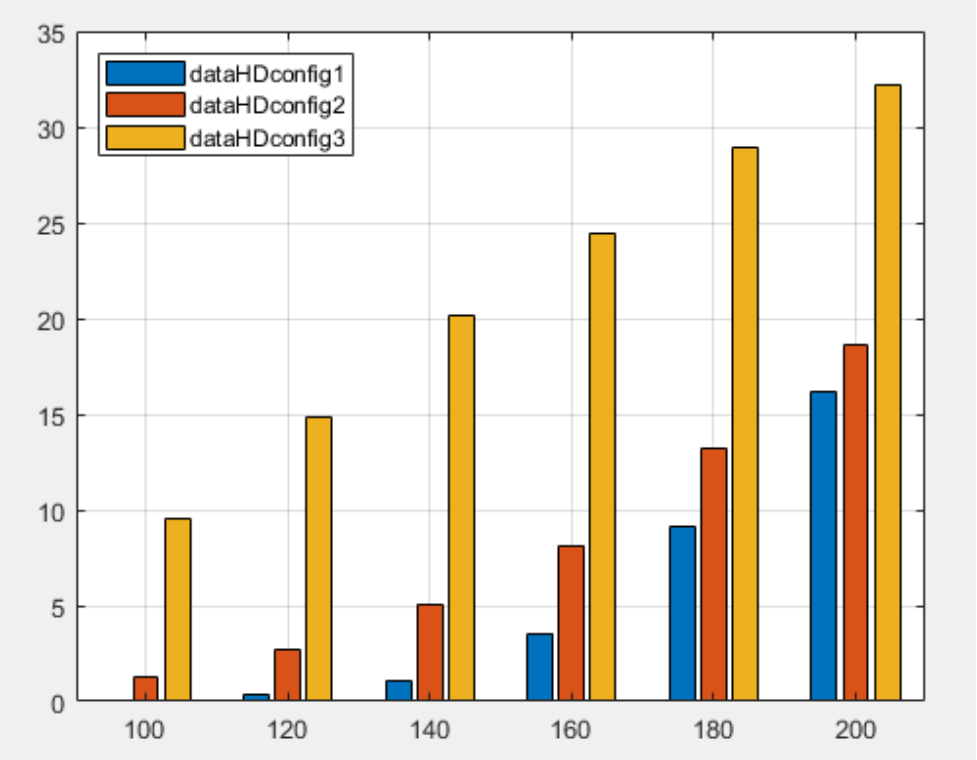

Figura 4: Probabilidade de bloqueio filmes HD em função do nº pedidos por hora

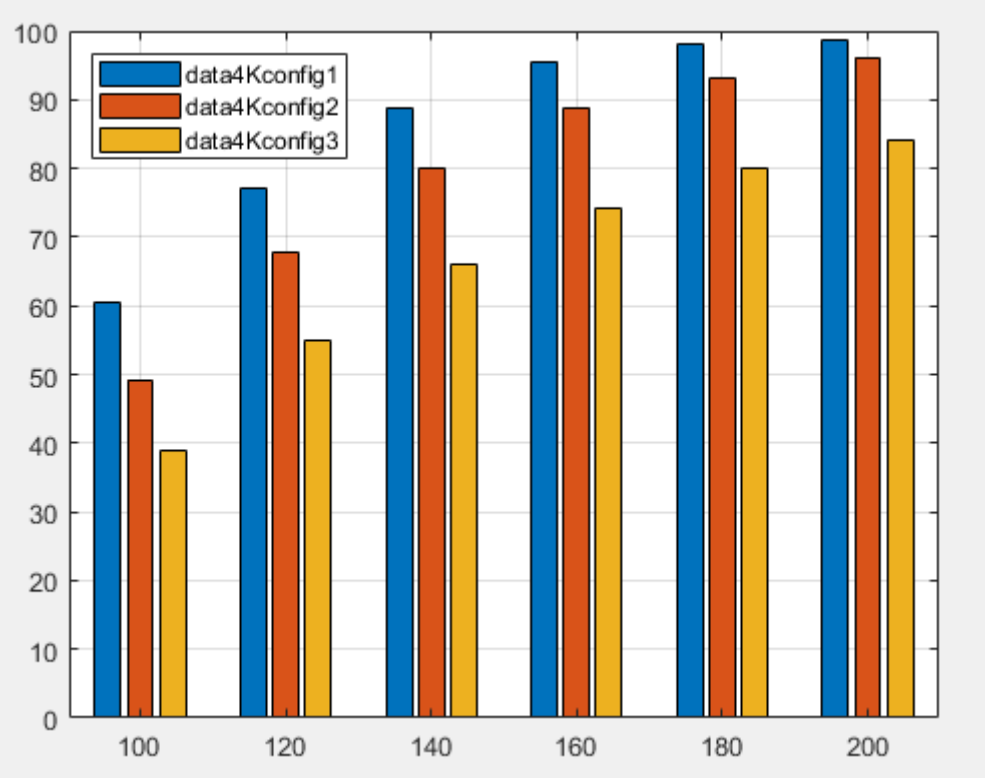

Figura 5: Probabilidade de bloqueio filmes 4K em função do nº pedidos por hora

Como é possível verificar através dos gráficos 4 e 5, independentemente da configuração a probabilidade de bloqueio aumenta sempre à medida que o número de pedidos por hora aumenta, algo que é previsível pois uma vez que chegam mais pedidos, há mais chance de os servidores já estarem cheios, acontecendo uma situação de bloqueio. No entanto podemos observar que as configurações do server farm, tem efeitos contrários para filmes HD e 4K, no caso de filmes HD a probabilidade de bloqueio aumenta significativamente à medida que o número de servidores de servidores diminui mesmo aumentando a interface da rede, comportamento oposto ao apresentado para filmes 4K nos quais diminui a probabilidade de bloqueio, isto deve se ao facto de os filmes 4K necessitarem de mais capacidade de throughput num servidor que os filmes HD, assim caso os servidores possuam um valor baixo de capacidade (Configuração 1), mais provável será acontecer um bloqueio. No caso dos filmes HD que requerem uma capacidade significativamente menor dos servidores, quanto mais servidores houver menor será a probabilidade de um pedido ficar bloqueado.

c)

```
R = 10000;
lambda = 100:20:200;
N = 10;
p = 20;
servers = 10;
S = 100;
W = 400;
i = 1;
for lam = lambda

    for it = 1:N
        [results(it),av(it)]= simulator2(lam,p,servers,S,W,R,"movies.txt");
        [results2(it),av2(it)]= simulator2(lam,p,4,250,W,R,"movies.txt");
        [results3(it),av3(it)]= simulator2(lam,p,1,1000,W,R,"movies.txt");
    end

    alfa= 0.1; %90% confidence interval%

    mediaHD = mean(results);
    termHD = norminv(1-alfa/2)*sqrt(var(results)/N);
    media2HD = mean(results2);
    term2HD = norminv(1-alfa/2)*sqrt(var(results2)/N);
    media3HD = mean(results3);
    term3HD = norminv(1-alfa/2)*sqrt(var(results3)/N);

    media24K = mean(av2);
    term24K = norminv(1-alfa/2)*sqrt(var(av2)/N);
    media34K = mean(av3);
    term34K = norminv(1-alfa/2)*sqrt(var(av3)/N);
    media4K = mean(av);
    term4K = norminv(1-alfa/2)*sqrt(var(av)/N);

    dataHD(i)= mediaHD;
    data2HD(i)= media2HD;
    data3HD(i)= media3HD;

    data4K(i)= media4K;
    data24K(i)= media24K;
    data34K(i)= media34K;
    i=i+1;
end
```

Figura 6: Código Matlab Task 2.c

Análise de código:

Código em todo similar ao código da alínea 2.b, única alteração presente no valor de reserva de recursos para filmes 4K, *W,* que nesta alínea possui o valor de 400.

Resultados e análise:

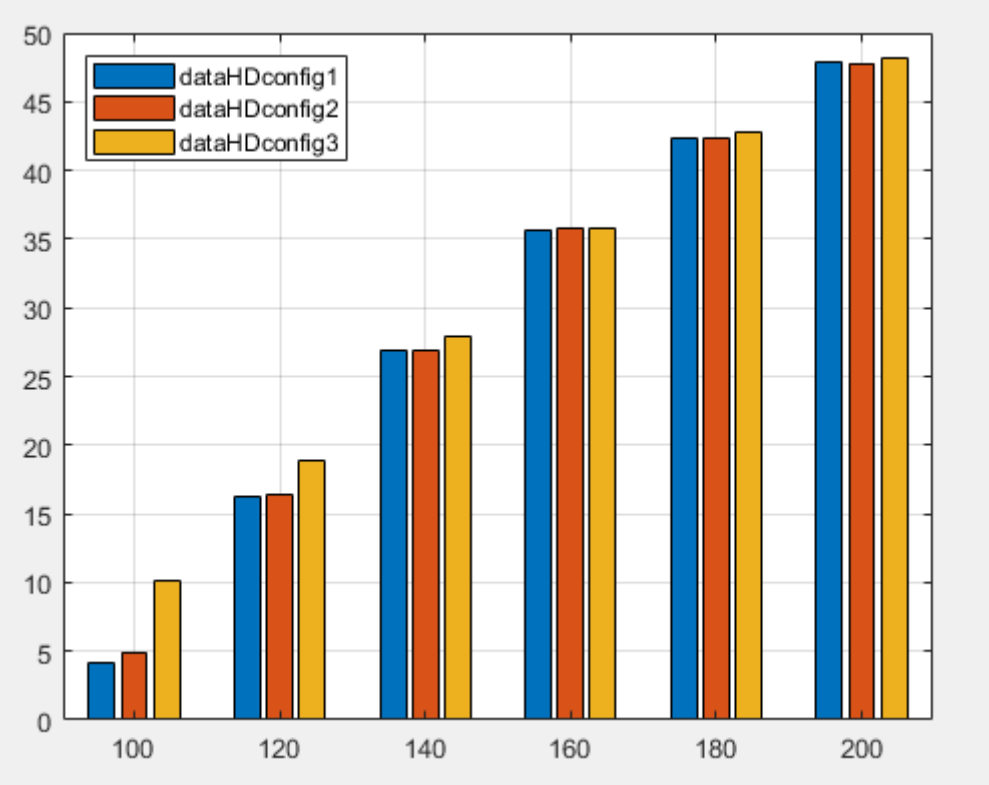

Figura 7: Probabilidade de bloqueio filmes HD em função do nº pedidos por hora

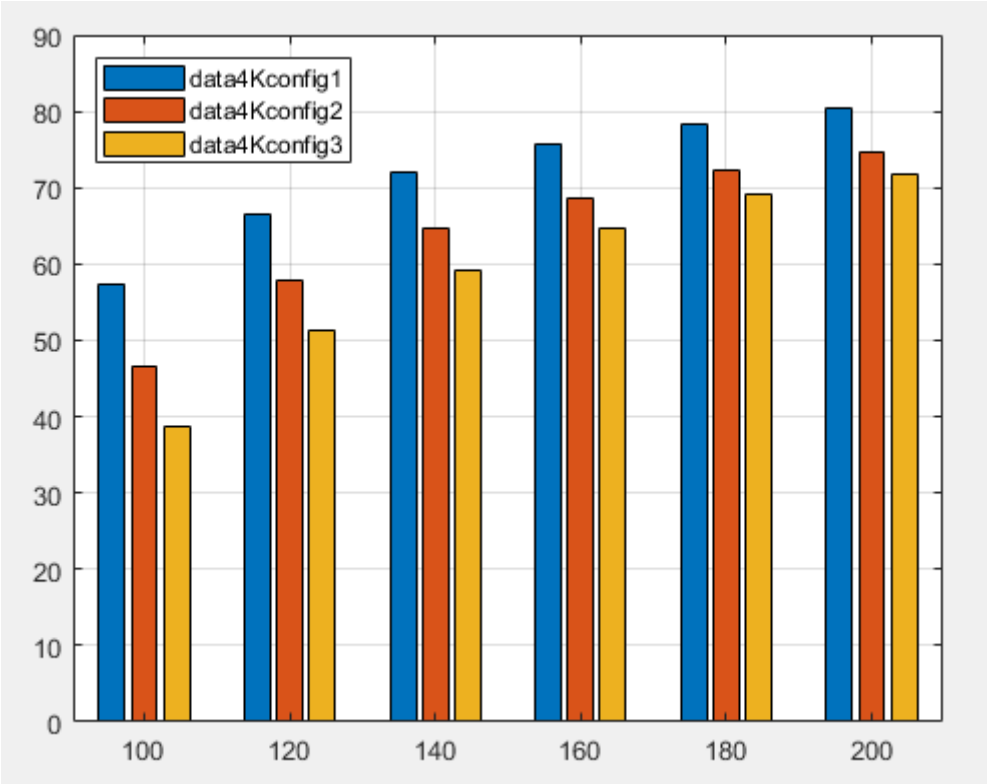

Figura 8: Probabilidade de bloqueio filmes 4K em função do nº pedidos por hora

Tal como na alínea anterior, o comportamento das probabilidades é parecido, no entanto, para filmes do formato HD não há tanta discrepância entre as diferentes configurações do server farm. Os valores das probabilidades de bloqueio para filmes HD aumentaram, visto

que os servidores têm agora menos capacidade, devido à reserva de capacidade para filmes 4K, já as probabilidades de bloqueio para filmes 4K diminuíram ligeiramente uma vez que existe mais capacidade alocada exclusivamente a este tipo de filmes.

d)

```
R = 10000;
alfa = 0.1;
lambda = 100:20:200;
N = 10;
p = 20;
servers = 10;
S = 100;
W = 600;
i = 1;
for lam = lambda

    for it = 1:N
        [results(it),av(it)]= simulator2(lam,p,servers,S,W,R,"movies.txt");
        [results2(it),av2(it)]= simulator2(lam,p,4,250,W,R,"movies.txt");
        [results3(it),av3(it)]= simulator2(lam,p,1,1000,W,R,"movies.txt");
    end

    alfa= 0.1; %90% confidence interval%

    mediaHD = mean(results);
    termHD = norminv(1-alfa/2)*sqrt(var(results)/N);
    media2HD = mean(results2);
    term2HD = norminv(1-alfa/2)*sqrt(var(results2)/N);
    media3HD = mean(results3);
    term3HD = norminv(1-alfa/2)*sqrt(var(results3)/N);

    media24K = mean(av2);
    term24K = norminv(1-alfa/2)*sqrt(var(av2)/N);
    media34K = mean(av3);
    term34K = norminv(1-alfa/2)*sqrt(var(av3)/N);
    media4K = mean(av);
    term4K = norminv(1-alfa/2)*sqrt(var(av)/N);

    dataHD(i)= mediaHD;
    data2HD(i)= media2HD;
    data3HD(i)= media3HD;

    data4K(i)= media4K;
    data24K(i)= media24K;
    data34K(i)= media34K;
    i=i+1;
end
```

Figura 9: Código Matlab Task 2.d

Análise de código:

Código em todo similar ao código da alínea 2.b, única alteração presente no valor de reserva de recursos para filmes 4K, *W*, que nesta alínea possui o valor de 600.

Resultados e análise:

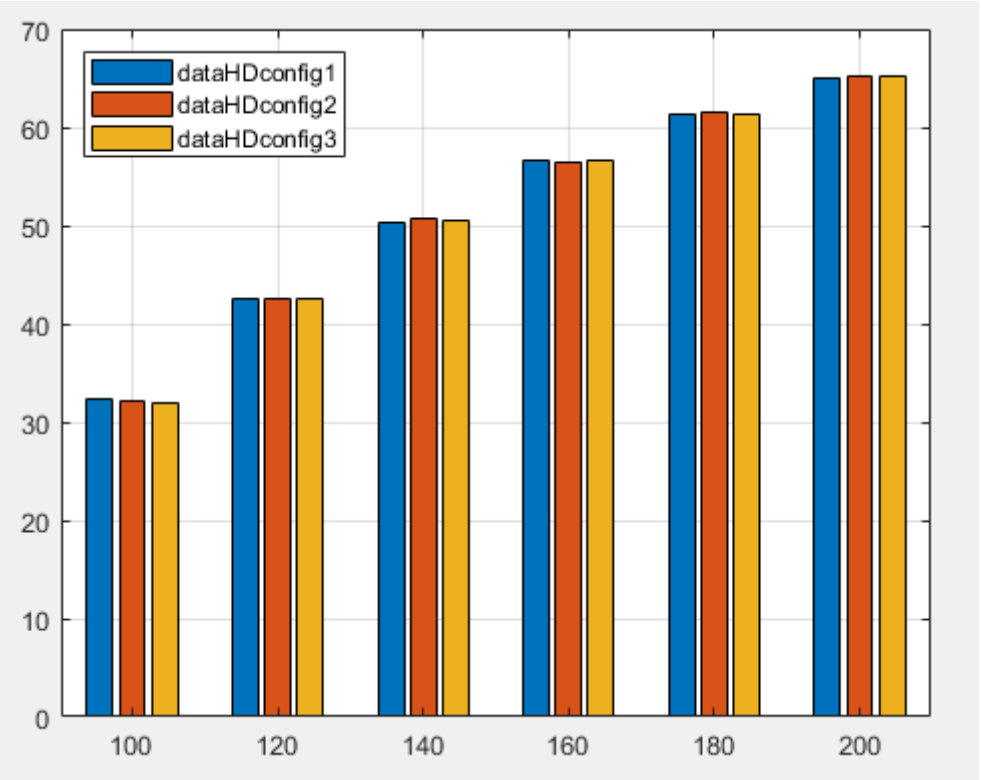

Figura 10: Probabilidade de bloqueio filmes HD em função do nº pedidos por hora

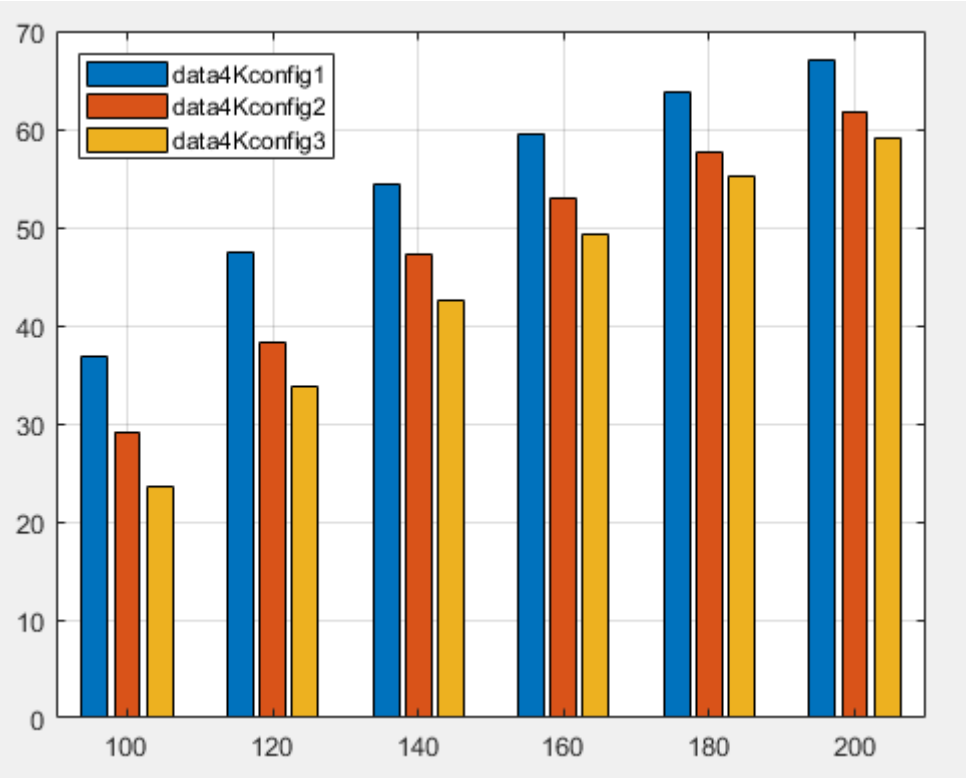

Figura 11: Probabilidade de bloqueio filmes 4K em função do nº pedidos por hora

Através dos gráficos obtidos nesta alínea, podemos observar um comportamento similar e esperado, em relação à alínea anterior, mais uma vez os valores das probabilidades de bloqueio para filmes HD subiram, tendo esta subida sido significativa e tendo rondado os 20 pontos percentuais em relação aos valores da alínea 2.c. Também para os filmes 4K as suas probabilidades desceram, descendo por volta de 10 pontos percentuais. Tanto o aumento da probabilidade de bloqueio em função do número de pedidos por hora para filmes HD como a descida da probabilidade de bloqueio em função do número de pedidos por hora para filmes 4K são explicadas pelo aumento da capacidade reservada para filmes 4K, uma vez que os filmes HD possuem menos recursos a seu dispor, havendo menos capacidade nos servidores para a sua reprodução é mais provável que haja um bloqueio de um pedido, pelo contrário para filmes 4K, havendo recursos destinados apenas a filmes deste tipo a probabilidade de um pedido ser bloqueado é menor.

Task 3:

a)

```
G=[ 1 2
    1 3
    1 4
    1 5
    1 6
    1 14
    1 15
    2 3
    2 4
```

... (declaração da variável G, tal como no apêndice D do guia)

```
% VARIABLES
n = 6:1:40;
c = zeros(40);
c = [repelem(0, 5) repelem(12, 15-5) repelem(8, 40-15)];
vars = {};
% GRAPH
internet = graph(G(:,1), G(:,2));
plot(internet)

% ILP FILE WRITING
f = fopen('task3a.lp', 'wt');

% OBJECTIVE FUNCTION
fprintf(f, 'Minimize\n');
for i=n
    fprintf(f, ' + %f x%d', c(i), i);
end

% CONSTRAINTS
fprintf(f, '\nSubject To\n');
write = 0;
for i = n
    for j = n
        [sp, len] = shortestpath(internet, i, j);
        if len <= 2
            write = 1;
            fprintf(f, ' + x%d', j);
        end
    end
    if write == 1
        fprintf(f, ' >= 1 \n');
    end
    write= 0;
end

% ILP BINARY VARIABLES
fprintf(f, '\nBinary\n');
for i = n
    fprintf(f, 'x%d', i);
    fprintf(f, '\n');
end

% END
fprintf(f, '\nEnd\n');
fclose(f);
```

Figuras 12 e 13: Código Matlab Task 3.a

Análise de código:

Em primeiro lugar, (linhas 67 - 69) foram declaradas as variáveis necessárias tais como os id's dos ASs tier 2 e 3, *n,* e o array de custos de cada nó, *c,* e de seguida foi criado o grafo dos ASs (72 - 73). Entre as linhas 76 e 110, foi realizada a construção do ficheiro .lp conforme a definição do modelo de Programação Linear Inteira presente no apêndice E do guia. Nas linhas 79 - 82 encontra-se a escrita da função objetivo, que é minimizar o custo total na disponibilização dos filmes pelos ASs. A partir da linha 85 até à linha 99 são escritas as condições do problema, no qual se inclui que todos ASs tenham pelo menos 1 AS onde se encontra um server farm com a distância máxima de 1 AS intermédio (linha 89 - 92). Por fim, é feita a escrita das variáveis binárias que representam os ASs, nas quais, caso lhes seja atribuído o valor 1 significa que esse AS contém um server farm (linhas 102 - 106), e também o final do ficheiro .lp (linhas 109 - 110).

Resultados e análise:

Ficheiro "task3a.lp" gerado:

```
Minimize
 + 12.000000 x6 + 12.000000 x7 + 12.000000 x8 + 12.000000 x9 + 12.000000 x10 + 12.000000 x11
 + 12.000000 x12 + 12.000000 x13 + 12.000000 x14 + 12.000000 x15 + 8.000000 x16 + 8.000000 x17
 + 8.000000 x18 + 8.000000 x19 + 8.000000 x20 + 8.000000 x21 + 8.000000 x22 + 8.000000 x23 + 8.000000 x24
 + 8.000000 x25 + 8.000000 x26 + 8.000000 x27 + 8.000000 x28 + 8.000000 x29 + 8.000000 x30 + 8.000000 x31
 + 8.000000 x32 + 8.000000 x33 + 8.000000 x34 + 8.000000 x35 + 8.000000 x36 + 8.000000 x37 + 8.000000 x38
 + 8.000000 x39 + 8.000000 x40
Subject To
 + x6 + x7 + x14 + x15 + x16 + x17 + x18 + x19 + x20 >= 1
 + x6 + x7 + x8 + x16 + x17 + x18 + x19 + x20 + x21 >= 1
 + x7 + x8 + x9 + x10 + x20 + x21 + x22 + x23 + x24 + x25 >= 1
 + x8 + x9 + x10 + x11 + x21 + x22 + x23 + x24 + x25 + x26 + x27 >= 1
 + x8 + x9 + x10 + x11 + x12 + x13 + x22 + x23 + x24 + x25 + x26 + x27 + x28 + x29 + x30 >= 1
 + x9 + x10 + x11 + x12 + x13 + x26 + x27 + x28 + x29 + x30 >= 1
 + x10 + x11 + x12 + x13 + x14 + x30 + x31 + x32 >= 1
 + x10 + x11 + x12 + x13 + x14 + x33 + x34 + x35 + x36 + x37 + x38 >= 1
 + x6 + x12 + x13 + x14 + x15 + x33 + x34 + x35 + x36 + x37 + x38 >= 1
 + x6 + x14 + x15 + x16 + x39 + x40 >= 1
 + x6 + x7 + x15 + x16 + x17 + x18 + x19 + x39 + x40 >= 1
 + x6 + x7 + x16 + x17 + x18 + x19 >= 1
 + x6 + x7 + x16 + x17 + x18 + x19 >= 1
 + x6 + x7 + x16 + x17 + x18 + x19 + x20 >= 1
 + x6 + x7 + x8 + x19 + x20 + x21 >= 1
 + x7 + x8 + x9 + x20 + x21 + x22 >= 1
 + x8 + x9 + x10 + x21 + x22 + x23 + x24 + x25 >= 1
 + x8 + x9 + x10 + x22 + x23 + x24 + x25 >= 1
 + x8 + x9 + x10 + x22 + x23 + x24 + x25 >= 1
 + x8 + x9 + x10 + x22 + x23 + x24 + x25 >= 1
 + x9 + x10 + x11 + x26 + x27 >= 1
 + x9 + x10 + x11 + x26 + x27 + x28 + x29 + x30 >= 1
 + x10 + x11 + x27 + x28 + x29 + x30 >= 1
 + x10 + x11 + x27 + x28 + x29 + x30 >= 1
 + x10 + x11 + x12 + x27 + x28 + x29 + x30 + x31 + x32 >= 1
 + x12 + x30 + x31 + x32 >= 1
 + x12 + x30 + x31 + x32 >= 1
 + x13 + x14 + x33 + x34 + x35 >= 1
 + x13 + x14 + x33 + x34 + x35 >= 1
 + x13 + x14 + x33 + x34 + x35 >= 1
 + x13 + x14 + x36 + x37 + x38 >= 1
 + x13 + x14 + x36 + x37 + x38 >= 1
 + x13 + x14 + x36 + x37 + x38 >= 1
 + x15 + x16 + x39 + x40 >= 1
 + x15 + x16 + x39 + x40 >= 1
```

```
44
45  Binary
46  x6
47  x7
48  x8
49  x9
50  x10
51  x11
52  x12
53  x13
54  x14
55  x15
56  x16
57  x17
58  x18
59  x19
60  x20
61  x21
62  x22
63  x23
64  x24
65  x25
66  x26
67  x27
68  x28
69  x29
70  x30
71  x31
72  x32
73  x33
74  x34
75  x35
76  x36
77  x37
78  x38
79  x39
80  x40
81
82  End
```

Figuras 14 e 15: Ficheiro .lp gerado

De salientar que o conteúdo das linhas 2 a 7 foi todo escrito em apenas uma linha aquando da geração do ficheiro, foram apenas separadas para a apresentação do ficheiro.

Ficheiro .sol obtido após execução no solver:

```
# Objective value = 48
x6 0
x7 0
x8 0
x9 1
x10 0
x11 0
x12 0
x13 1
x14 0
x15 0
x16 1
x17 0
x18 0
x19 0
x20 0
x21 1
x22 0
x23 0
x24 0
x25 0
x26 0
x27 0
x28 0
x29 0
x30 1
x31 0
x32 0
x33 0
x34 0
x35 0
x36 0
x37 0
x38 0
x39 0
x40 0
```

Figura 16: Ficheiro .sol resultante

Após a execução do modelo ILP, presente no ficheiro gerado, no solver, foram obtidos os resultados apresentados na figura acima. Podemos observar que o custo total da solução é de 48, e que são necessários 5 server farms, localizados nos ASs 9, 13, 16, 21 e 30. Estes ASs são ASs com ligação a muitos outros ASs em seu redor; a exemplo o AS 9, que analisando o grafo apresentado na figura a seguir, verificamos que consegue providenciar 11 dos 40 ASs existentes tendo em consideração as condições ao qual o problema está sujeito; logo faz sentido que tenham sido selecionados estes ASs.

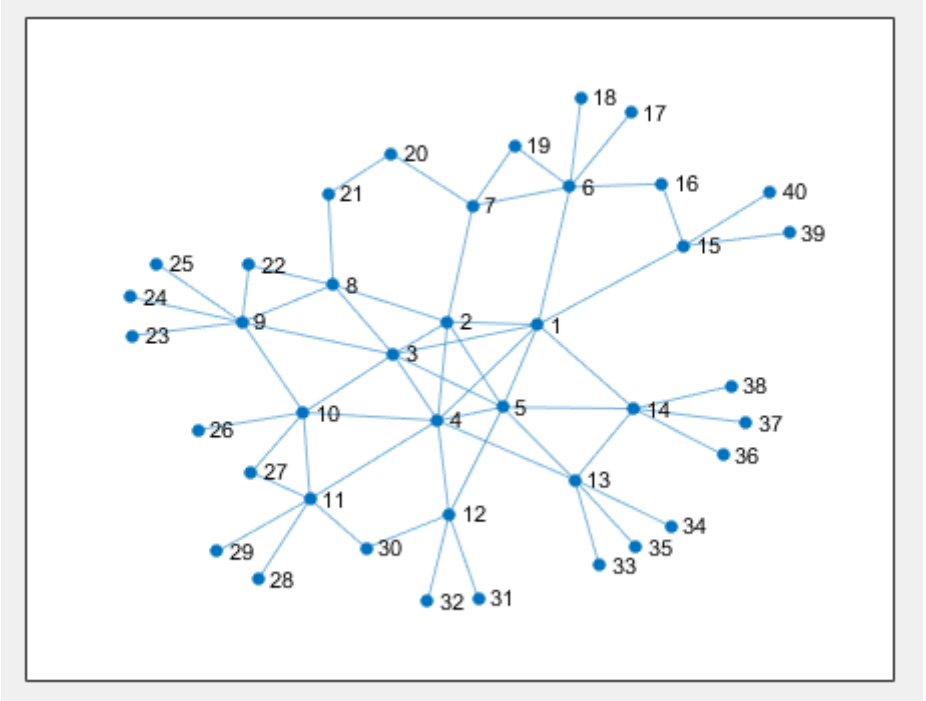

Figura 17: Grafo obtido

Apêndice A:

Simulator2.m:

```
1   function [bHD, b4K]= simulator2(lambda, p, n, S, W, R, fname)
2       % lambda - movies request rate (in requests per hour)
3       % p = percentage of requests for 4K movies (in %)
4       % n = Number of servers
5       % S = capacity of each server
6       % W = Resource reservation for 4K movies (in Mbps)
7       % R = Number of movie requests to stop simulation
8       % fname = filename
9
10      invlambda=60/lambda; %average time between requests (in minutes)
11      invmiu= load(fname); %duration (in minutes) of each movie
12      Nmovies= length(invmiu); % number of movies
13
14      % Events Definition
15      DEPARTURE_HD = 1; % HD movie termination
16      DEPARTURE_4K = 2; % 4K movie termination
17      ARRIVAL_HD = 0;  % HD movie arrival
18      ARRIVAL_4K = 3; % 4K movie arrival
19
20      % State Variables
21      STATE = zeros(1, n); % Throughput of movies in transmission by server
22      STATE_HD = 0; % Throughput of HD movies in transmission
23
24      % Statistical counters
25      NARRIVALS = 0; % Movie requests
26      REQUESTS_HD = 0; % HD movie requests
27      REQUESTS_4K = 0; % 4K movie requests
28      BLOCKED_HD = 0; % Blocked HD movie requests
29      BLOCKED_4K = 0; % Blocked 4K movie requests
30
31      %Simulation Clock
32      Clock = 0;
33      C = S*n; % Bandwidth for a whole server farm
```

Figura 18: Simulator2 inicialização de variáveis

Nesta secção de código são apenas inicializadas as variáveis necessárias ao simulador, tendo por base o apêndice C do guia, no entanto, optamos por definir variáveis distintas para um eventos do tipo arrival HD e arrival 4K (linhas 17 -18).

```
35 -        if(rand(1)<p/100)
36 -            new_event = ARRIVAL_4K;
37 -        else
38 -            new_event = ARRIVAL_HD;
39 -        end
40 -        EventList= [new_event exprnd(invlambda) 0];
41 -
42 -        while NARRIVALS < R
43 -            event= EventList(1,1);
44 -            Clock= EventList(1,2);
45 -            idx2 = EventList(1,3);
46 -            EventList(1,:)= [];
47 -            [Min, Idx] = min(STATE);
48 -            if event == ARRIVAL_HD
49 -                if(rand(1)<p/100)
50 -                    new_event = ARRIVAL_4K;
51 -                else
52 -                    new_event = ARRIVAL_HD;
53 -                end
54 -
55 -                EventList= [EventList; new_event Clock+exprnd(invlambda) 0];
56 -                NARRIVALS = NARRIVALS + 1;
57 -                REQUESTS_HD = REQUESTS_HD + 1;
58 -                if(Min+5 <= S && STATE_HD+5 <= C-W)
59 -                    STATE(1, Idx) = STATE(1, Idx) + 5;
60 -                    STATE_HD = STATE_HD + 5;
61 -                    EventList= [EventList; DEPARTURE_HD Clock+invmiu(randi(Nmovies)) Idx];
62 -                else
63 -                    BLOCKED_HD = BLOCKED_HD + 1;
64 -                end
```

Figura 19: Geração de eventos e procedimento para eventos ARRIVAL_HD

Nesta secção é gerado um novo evento e adicionado há *EventList* tendo em conta a probabilidade de ser 4K ou HD. De seguida, caso seja um evento *ARRIVAL_HD*, são atualizadas as respectivas variáveis e é feita a atribuição deste evento ao servidor com menos carga nesse momento, caso nenhum servidor consiga alojar o pedido este fica bloqueado e é atualizada a variável *BLOCKED_HD* respetiva.

```
66 -            elseif event==ARRIVAL_4K
67 -                if(rand(1)<p/100)
68 -                    new_event = ARRIVAL_4K;
69 -                else
70 -                    new_event = ARRIVAL_HD;
71 -                end
72 -                EventList= [EventList; new_event Clock+exprnd(invlambda) 0];
73 -                NARRIVALS = NARRIVALS + 1;
74 -                REQUESTS_4K = REQUESTS_4K + 1;
75 -
76 -                if(Min <= S-25)
77 -                    STATE(1, Idx) = STATE(1, Idx) + 25;
78 -                    EventList= [EventList; DEPARTURE_4K Clock+invmiu(randi(Nmovies)) Idx];
79 -
80 -                else
81 -                    BLOCKED_4K = BLOCKED_4K + 1;
82 -                end
83
```

Figura 20: Procedimiento para eventos ARRIVAL_4K

Neste excerto é feito um processo semelhante para caso o evento gerado anteriormente seja *ARRIVAL_4K*, são atualizadas as variáveis respetivas e é feito a atribuição do filme a um servidor disponível, novamente caso não haja nenhum servidor disponível o evento ficará bloqueado e é atualizada a variável *BLOCKED_4K* respetiva.

```
84 -            elseif event==DEPARTURE_HD
85 -                STATE(1, idx2) = STATE(1, idx2) - 5;
86 -                STATE_HD = STATE_HD - 5;
```

Figura 21: Procedimiento para eventos DEPARTURE_HD

Caso o evento gerado seja um evento *DEPARTURE_HD*, é retirado o filme ao servidor onde este se encontrava alocado e atualizado o throughput total de filmes HD (linhas 85 e 86).

```
87 -            else
88 -                STATE(1, idx2) = STATE(1, idx2) - 25;
89 -            end
90
91 -            EventList= sortrows(EventList,2);
92 -        end
93
94 -        bHD = 100*BLOCKED_HD/REQUESTS_HD; % blocking probability in %
95 -        b4K = 100*BLOCKED_4K/REQUESTS_4K; % blocking probability in %
96 -    end
```

Figura 22: Procedimiento para eventos DEPARTURE_4K e cálculo final de probabilidades de bloqueio

Por último, caso seja evento *DEPARTURE_4K*, é retirado o filme ao servidor onde se encontrava (linha 88), uma vez concluído o ciclo de processamento do evento, passa-se para o próximo evento na lista (linha 91).
Os valores de retorno do simulador, são calculados por fim nas linhas 94 e 95, onde é calculada em porcentagem a probabilidade de bloqueio de filmes HD e 4K.