universidade de aveiro

Relatório do Guião nº 2 de Desempenho e Dimensionamento de Redes

Task 2

Simulator 2

Para resolver os exercícios da Task 2, foi necessário desenvolver um simulador, que seria posteriormente invocado nas alíneas da task 2, que ao receber vários parâmetros devolvesse a probabilidade de bloqueio de filmes em 4K e em HD.
Para isso, baseamo-nos no *simulator* 1 e no apêndice C.

No *simulator* 1, calculamos a probabilidade de bloqueio apenas para filmes com um determinada taxa de transferência. Neste *simulator,* pretendemos calcular a probabilidade de bloqueio para filmes com 2 tipos de taxa de transferência, filmes 4K (25 Mbps), e filmes HD (5 Mbps). Neste *simulator* também existe um argumento que será uma reserva de recursos para filmes em 4K.

```
function [b_hd b_4k]= simulator2(lambda,p,n,S,W,R,fname)
    %lambda = request arrival rate (in requests per hour)
    %p      = percentage of requests for 4K movies (in %)
    %n      = number of servers
    %S      = interface capacity of each server (in Mbps)
    %W      = resource reservation for 4K movies (in Mbps)
    %R      = number of movie requests to stop simulation
    %fname  = file name with the duration (in minutes) of the items

    invlambda=60/lambda;     % average time between requests (in minutes)
    invmiu= load(fname);     % duration (in minutes) of each movie
    Nmovies= length(invmiu); % number of movies

    %Events definition:
    ARRIVAL= 0;              % movie request
    DEPARTURE_HD= 1;         % termination of an hd movie transmission
    DEPARTURE_4K= 2;         % termination of a 4k movie transmission
    %State variables initialization:
    STATE= zeros(1,n);       % number of total Mbps in use
    STATE_HD= 0;             % n*S-W  number of HD movies Mbps in use
    %Statistical counters initialization:
    REQUESTS_HD= 0;
    REQUESTS_4K= 0;
    BLOCKED_HD= 0;
    BLOCKED_4K= 0;
    %Simulation Clock and initial List of Events:
    Clock= 0;
```

```
   EventList= [ARRIVAL exprnd(invlambda) 0];
   RR_HD = n*S-W; % resource reservation for HD movies (in Mbps)

   while REQUESTS_HD + REQUESTS_4K < R
      event= EventList(1,1); % identifies the event
      Clock= EventList(1,2);
      server= EventList(1,3); % identifies the server

      EventList(1,:)= [];
      x = rand;    % x defines if the request is in HD quality or in 4K
      if event == ARRIVAL
         [a, server] = min(STATE); % chooses the server with the most free space
         if(x>p) % if the request is in HD quality
            EventList= [EventList; ARRIVAL Clock+exprnd(invlambda) 0];
            REQUESTS_HD= REQUESTS_HD+1;
            if STATE(server) + 5 <= S && STATE_HD + 5 <= RR_HD % if the request can
be answered
               STATE(server) = STATE(server) + 5; % 5 Mbps added to the server STATE
               STATE_HD = STATE_HD + 5; % 5 Mbps added to STATE_HD
               EventList= [EventList; DEPARTURE_HD Clock+invmiu(randi(Nmovies))
server];
            else
               BLOCKED_HD= BLOCKED_HD+1;
            end
         else            % if the request is in 4K quality
            EventList= [EventList; ARRIVAL Clock+exprnd(invlambda) 0];
            REQUESTS_4K= REQUESTS_4K+1;
            if STATE(server) + 25 <= S                % if it is allowed to process the request
               STATE(server) = STATE(server)+25; % 25 Mbps added to the server STATE
               EventList= [EventList; DEPARTURE_4K Clock+invmiu(randi(Nmovies))
server];
            else % if not, it is blocked
               BLOCKED_4K= BLOCKED_4K+1;
            end
         end
      else
         if(event == DEPARTURE_HD)
            STATE(server) = STATE(server) - 5; % subtracts 5 Mbps to the server STATE
            STATE_HD = STATE_HD - 5; % subtracts 5 Mbps to STATE_HD
         else
            STATE(server) = STATE(server) - 25; % subtracts 25 Mbps to the server STATE
         end
      end
      EventList= sortrows(EventList,2);   % order by time
   end
   b_hd = 100*BLOCKED_HD/REQUESTS_HD; % blocking of HD movies probability in %
   b_4k = 100*BLOCKED_4K/REQUESTS_4K;   % blocking of 4K movies probability in %
end
```

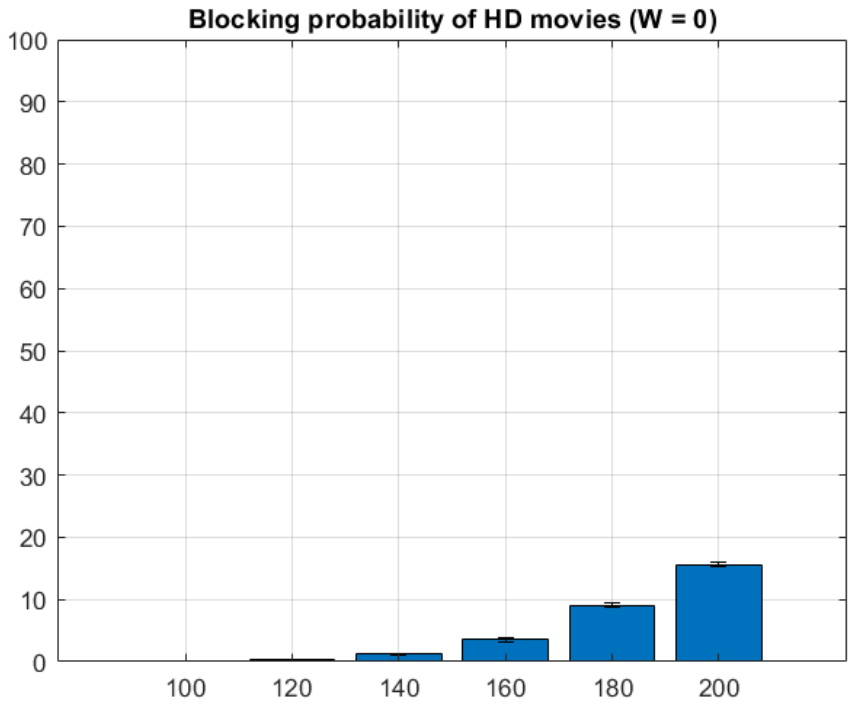
2.a. Develop a MATLAB script to run 10 times simulator2 with a stopping criterion of R = 10000 and to compute the estimated values and the 90% confidence intervals of both blocking probabilities. Consider Configuration 1 for = 100, 120, 140, 160, 180 and 200 requests/hour, p = 20%, and a resource reservation W = 0 Mbps. Present the results and the confidence intervals in bar charts with error bars. Analyse the results and take conclusions on (i) the impact of the arriving rate of the movie requests in the blocking probability of each movie format and (ii) if the stopping criterion value is large enough or should be larger.

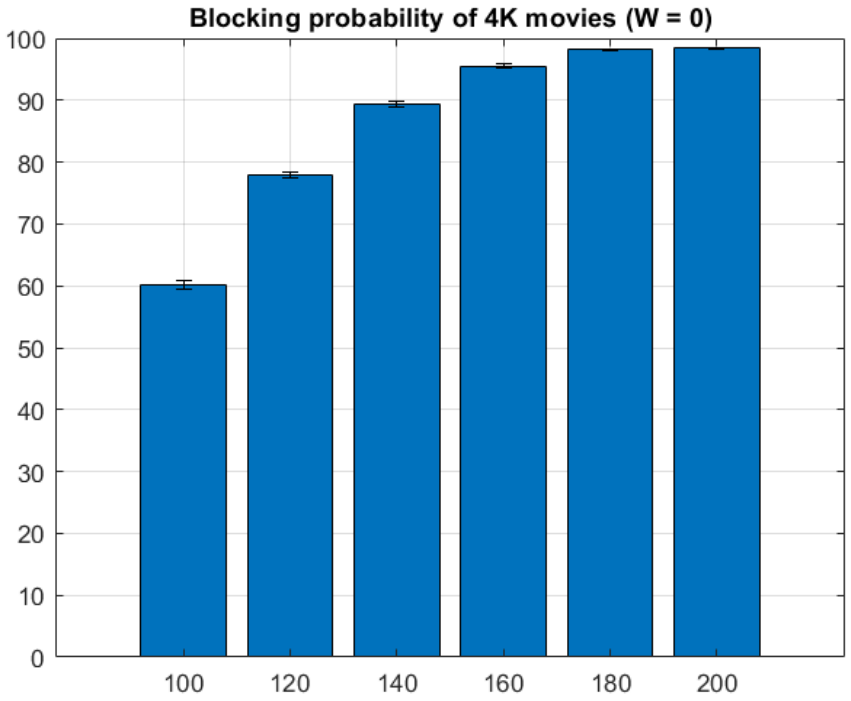
Na resolução desta alínea começamos por construir o *simulator* 2 a partir do *simulator* 1 previamente fornecido.
Em seguida, geramos um ciclo que é efetuado 6 vezes, 1 por cada valor do lambda e dentro desse ciclo efetuamos a simulação 10 vezes como pedido. Para além disso, calculamos a média das 10 simulações da probabilidade de bloqueio tanto para filmes HD como para filmes 4K e também a média do erro para os 2 casos.

```
R = 10000;  % Stopping criterion
lambda = 100:20:200;   % Requests per hour
p = 0.2;    % Probablity of being a 4K movie
alfa = 0.1;     % 90% confidence interval
W = 0;    % Resource reservation for 4K movies (Mbps)
fname = 'movies.txt';    % Textfile name
N = 10;     % Number of simulations

%  Configuration 1
n = 10;     % Number of servers
S = 100;      % Server capacity

for it = 1:6    % Six lambda values
   for i= 1:N    % Ten simulations for each lambda value
      [b_hd(i) b_4k(i)] = simulator2(lambda(it),p,n,S,W,R,fname);
   end
      % Average blocking probability for HD and 4K movies
      media_HD(it) = mean(b_hd);
      media_4K(it) = mean(b_4k);

      % Term for each movie format
      term_HD(it) = norminv(1-alfa/2)*sqrt(var(b_hd)/N);
      term_4K(it) = norminv(1-alfa/2)*sqrt(var(b_4k)/N);
end

% HD
figure(1)
bar(lambda, media_HD)
title('Blocking probability of HD movies (W = 0)')
ylim([0 100])
grid on
hold on
```

```
% Error bar
er = errorbar(lambda, media_HD, term_HD, term_HD);
er.Color = [0 0 0];
er.LineStyle = 'none';
hold off

% 4K
figure(2)
bar(lambda, media_4K)
title('Blocking probability of 4K movies (W = 0)')
ylim([0 100])
grid on
hold on

% Error bar
er = errorbar(lambda, media_4K, term_4K, term_4K);
er.Color = [0 0 0];
er.LineStyle = 'none';
hold off
```

Obtivemos então os seguintes gráficos:





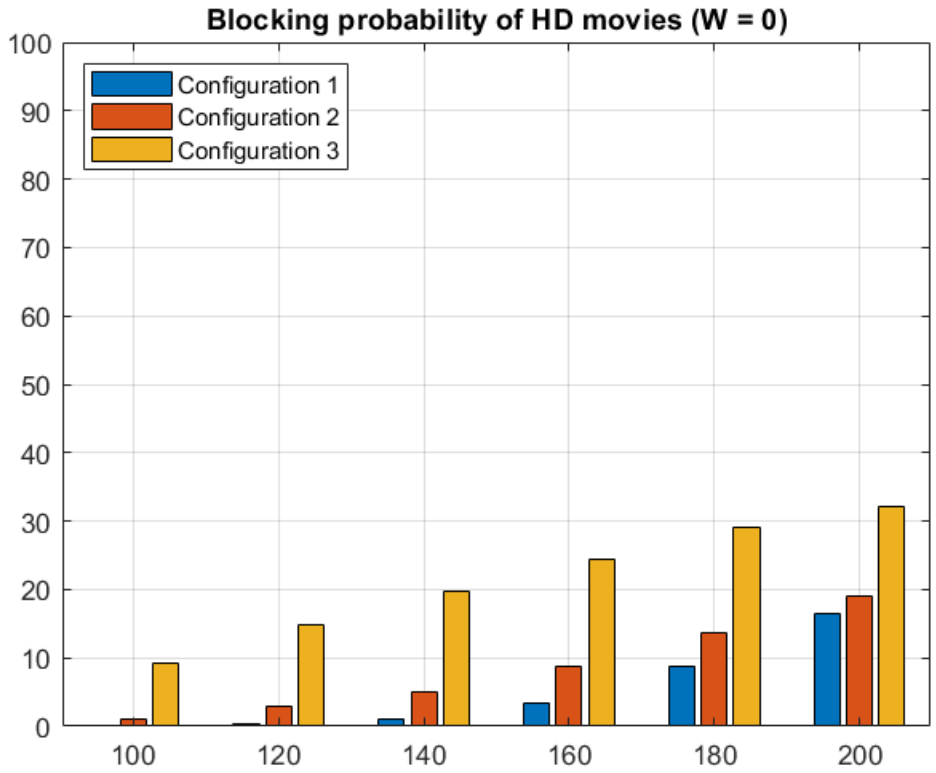
Perante os resultados obtidos concluímos que a probabilidade de bloqueio em filmes 4K é muito superior à probabilidade de bloqueio em filmes HD, o que faz sentido dado que o *throughput* dos primeiros é 5 vezes superior ao dos filmes HD. Podemos também observar que quanto maior o número de pedidos por hora, maior a probabilidade de haver bloqueios, tanto num caso como no outro. Relativamente ao *stopping criterion* pensamos que o valor é suficientemente grande para tirar as conclusões necessárias e para o erro não ser demasiado.

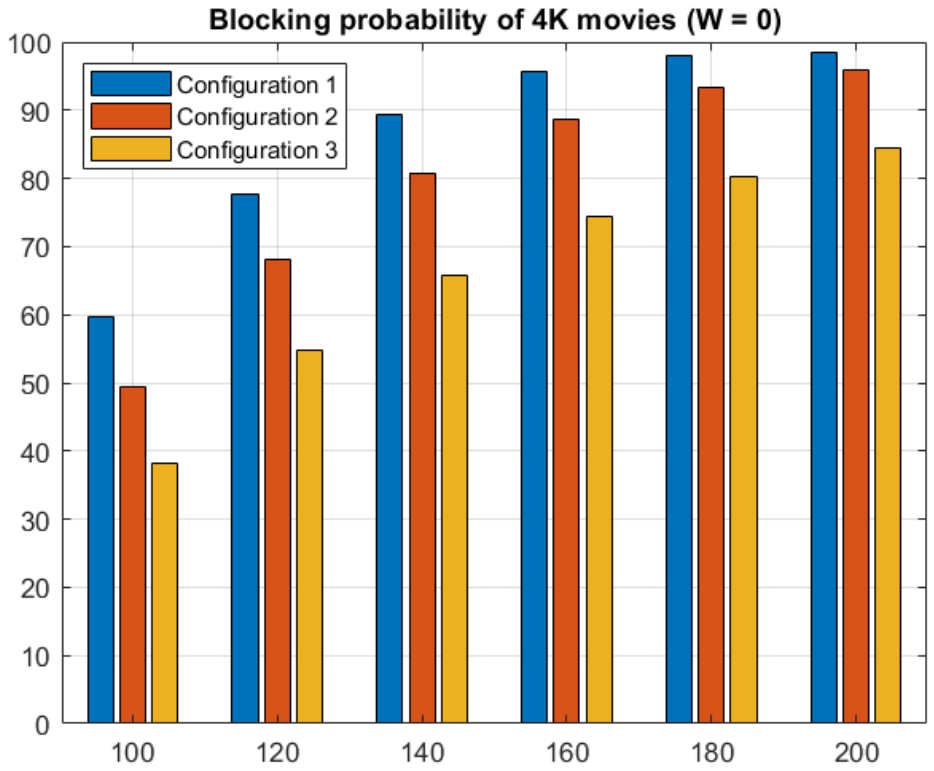
2.b. Repeat the simulations requested in question 2.a but now considering Configurations 2 and 3 (consider the same values of all other parameters). Present the bar charts of these results together with the previous results on a single figure for the blocking probability of HD movie requests and another single figure for the blocking probabilities of 4K movie requests (for clarity, do not include the error bars). Analyse the results and take conclusions on the impact of the 3 server farm configurations in the blocking probability of each movie format.

Para a resolução deste exercício repetimos o processo da alínea a) para as 3 configurações (n=10 e S=100; n=4 e S=250; n=1 e S=1000).

```
%  Configuration 1
n = 10;
S = 100;

for it = 1:6
    for i= 1:N
        [b_hd(i) b_4k(i)] = simulator2(lambda(it),p,n,S,W,R,fname);
    end
        media1_HD(it) = mean(b_hd);
        media1_4K(it) = mean(b_4k);
end

%  Configuration 2
n = 4;
S = 250;
lambda = 100:20:200;

for it = 1:6
    for i= 1:N
        [b_hd(i) b_4k(i)] = simulator2(lambda(it),p,n,S,W,R,fname);
    end
        media2_HD(it) = mean(b_hd);
        media2_4K(it) = mean(b_4k);
end

%  Configuration 3
n = 1;
S = 1000;
lambda = 100:20:200;

for it = 1:6
    for i= 1:N
        [b_hd(i) b_4k(i)] = simulator2(lambda(it),p,n,S,W,R,fname);
    end
        media3_HD(it) = mean(b_hd);
        media3_4K(it) = mean(b_4k);
end
```

Obtivemos os gráficos abaixo:

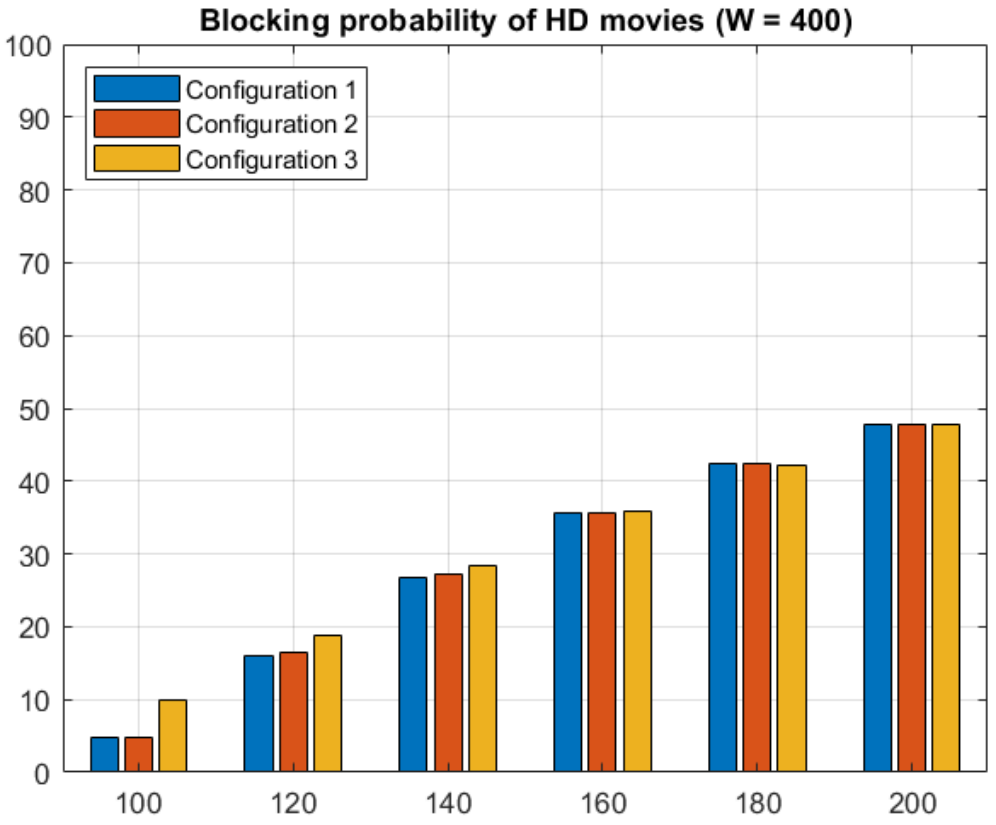

Blocking probability of HD movies (W = 0)
Configuration 1
Configuration 2
Configuration 3
100
90
80
70
60
50
40
30
20
10
0
100
120
140
160
180
200

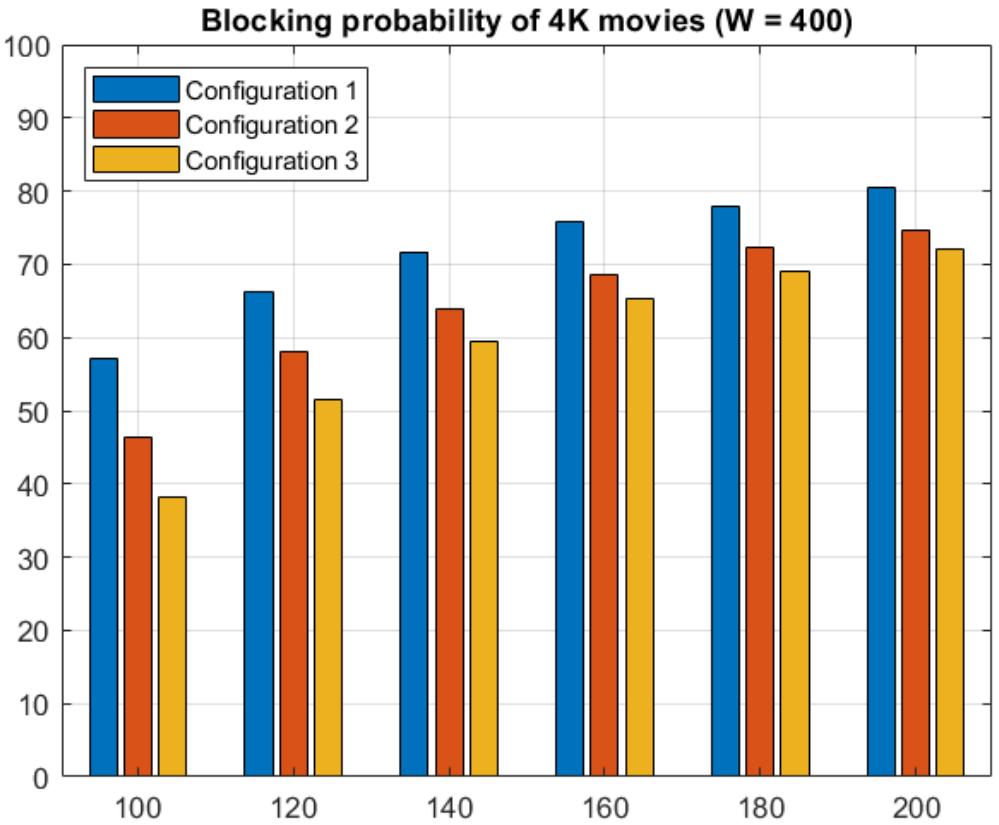

Blocking probability of 4K movies (W = 0)
Configuration 1
Configuration 2
Configuration 3
100
90
80
70
60
50
40
30
20
10
0
100
120
140
160
180
200

Podemos observar que as probabilidades de bloqueio em filmes 4K são bastante mais altas do que em filmes HD, isto porque a reserva de recursos para os filmes 4K é nula. Relativamente às configurações, concluímos que para os filmes 4K é mais benéfica a configuração 3 (1 servidor com interface de rede de 1000Mbps), por outro lado para os filmes HD é mais benéfica a configuração 1 (10 servidores com uma interface de rede de 100Mbps).
Isto deve-se ao facto de que, com 10 servidores de 100 Mbps haver probabilidade de no global ter espaço suficiente para responder a um *request* 4K, mas não ser possível responder ao pedido num servidor. Isto faz com que, nessa configuração possa ficar um espaço livre em cada servidor onde seja possível responder a pedidos HD mas não seja possível responder a pedidos 4K, favorecendo os pedidos HD e diminuindo a sua probabilidade de bloqueio.
Num único servidor, o que foi referido previamente sobre os filmes 4K não acontecerá, a única maneira de um filme 4K ser bloqueado é se não houver possibilidade global (neste caso apenas do único servidor) de responder ao pedido.

2.c. Repeat the simulations for all 3 server farm configurations but now considering a resource reservation W = 400 Mbps. Again, present the bar charts of these results on a single figure for the blocking probability of HD movie requests and another single figure for the blocking probabilities of 4K movie requests (and do not include the error bars). Compare these results with the previous results and take conclusions on the impact of the resource reservation in the blocking probability of each movie format for the 3 server farm configurations.

Nesta alínea, repetimos o processo da alínea anterior e a única alteração foi no valor da variável W (reserva de recursos para filmes 4K) que passou a ser igual a 400. Obtivemos os seguintes gráficos:

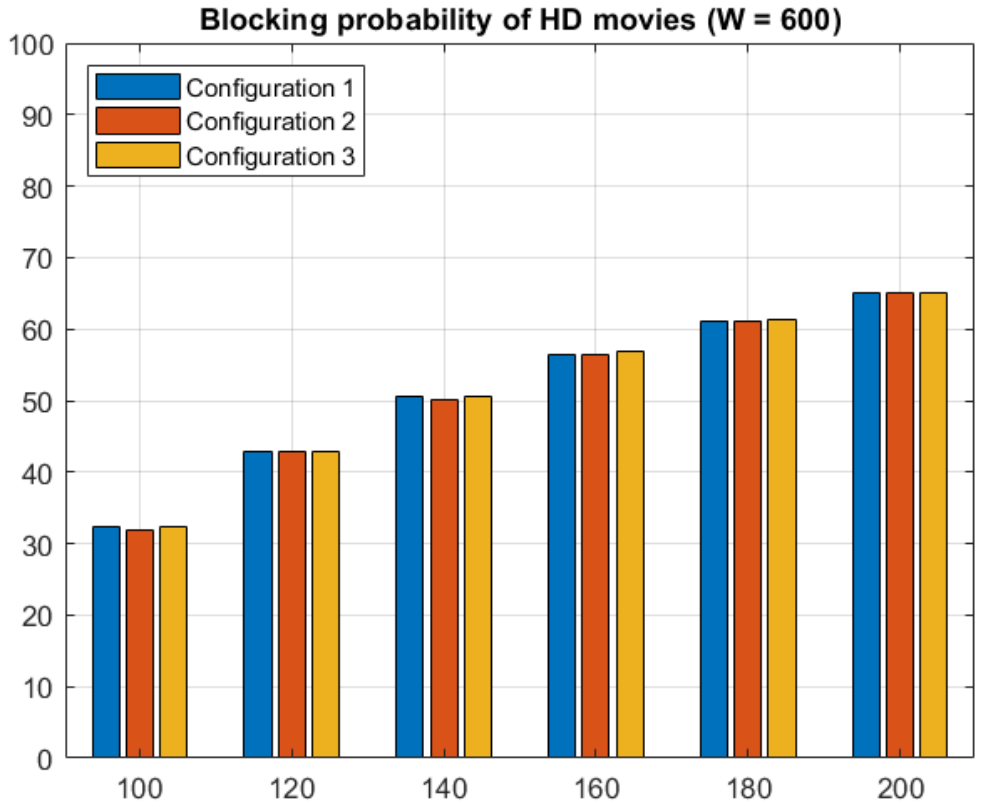

Blocking probability of HD movies (W = 400)
Configuration 1
Configuration 2
Configuration 3
100
90
80
70
60
50
40
30
20
10
0
100
120
140
160
180
200

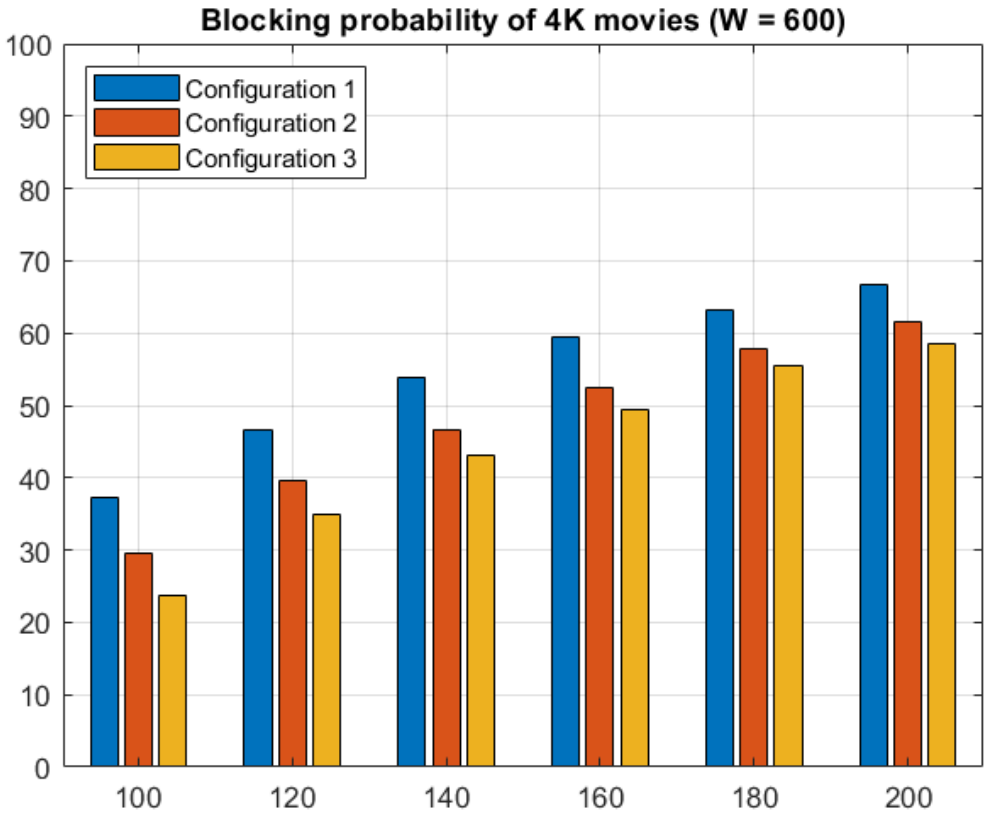

Blocking probability of 4K movies (W = 400)
Configuration 1
Configuration 2
Configuration 3
100
90
80
70
60
50
40
30
20
10
0
100
120
140
160
180
200

Perante estes resultados confirmamos o que já esperávamos, que a probabilidade de bloqueio em filmes 4K é mais baixa do que quando o W é igual a 0 e a probabilidade de bloqueio em filmes HD é mais alta. Isto deve-se ao facto de os filmes HD não poderem ocupar mais do que 600Mbps, ou seja, se o throughput total de filmes HD for superior a este valor, os filmes HD ficam bloqueados, dando prioridade aos filmes 4K.

2.d. Repeat the previous question 2.c but now considering a resource reservation W = 600 Mbps and present the results in the same way as before. Compare these results with the previous results and take conclusions on the impact of the different resource reservation values in the blocking probability of each movie format for the 3 server farm configurations.

Para a resolução desta alínea, repetimos o processo das duas alíneas anteriores e a única alteração foi no valor da variável W (reserva de recursos para filmes 4K) que passou a ser igual a 600. Obtivemos os gráficos abaixo:





Observando estes resultados podemos concluir o que, mais uma vez, era expectável, que a probabilidade de bloqueio em filmes 4K é mais baixa do que quando o W é igual a 0 ou 400 e a probabilidade de bloqueio em filmes HD é mais alta.
A razão para isto acontecer é a que já vimos na alínea acima, dado que os filmes HD não podem ocupar mais do que 400Mbps, os filmes 4K têm prioridade logo a partir deste valor, portanto a probabilidade de bloqueio em filmes 4K vai ser ainda menor e a de filmes HD vai ser maior, ficando até bastante equilibradas.

2.e. Consider a video-service company with 100000 (one hundred thousand) subscribers, among which 24% are golden subscribers and 76% are regular subscribers (i.e., a golden subscriber is provided with movies in 4K format while a regular subscriber is provided with movies in HD format). Both types of subscribers are expected to request, on average, 1 movie per day.
The company aims to have a robust server farm solution such that the worst blocking probability among the two types of subscribers is not higher than 0.1% when all servers are working and 1% when one server fails. Consider that each server has an interface of S = 10 Gbps (= 10000 Mbps). Determine by simulation (using a stopping criterion of R = 100000) the minimum number of required servers and a proper reservation value W to be set in the front-office of the service to meet the required blocking performance.

Nesta alínea adotamos uma abordagem de tentativa erro. Fomos procurando o número ideal de servidores, e após termos o número ideal de servidores ajustamos o valor de W (reserva para filmes em resolução 4K).
Até aos 5 servidores, com qualquer valor de W, a probabilidade de bloqueio para filmes HD e 4K era sempre maior que 1%, logo ficamos a saber que menos que 7 servidores era impossível para obtermos as taxas de bloqueio que pretendíamos.

```
clear all % Clears the cache

fprintf('Starting simulation\n');
fname = 'movies.txt';    % Textfile name

W = 36800;     % Resource reservation for 4K movies (Mbps)
lambda = 4167;  %100 000 / 24 (Requests per hour)
percent_golden = 0.24;

alfa = 0.1;    % 90% confidence interval
n = 7;      % number of servers
S = 10000;  % 10 GB (10000 MB)
R = 100000;   % Stopping criterion
N = 10;   % Number of simulations

for it= 1:N
    [b_hd(it) b_4k(it)] = simulator2(lambda,percent_golden,n,S,W,R,fname)
end

% 1 server fails
n = n - 1;

for it= 1:N
    [b_hd_fail(it) b_4k_fail(it)] = simulator2(lambda,percent_golden,n,S,W,R,fname)
end

% All servers working
media_HD = mean(b_hd);
media_4K = mean(b_4k);

% One server fails
media_HD_fail = mean(b_hd_fail);
media_4K_fail = mean(b_4k_fail);

% All servers working
term_HD = norminv(1-alfa/2)*sqrt(var(b_hd)/N);
term_4K = norminv(1-alfa/2)*sqrt(var(b_4k)/N);

% One server fails
term_HD_fail = norminv(1-alfa/2)*sqrt(var(b_hd_fail)/N);
term_4K_fail = norminv(1-alfa/2)*sqrt(var(b_4k_fail)/N);

fprintf('\nWith %d servers:\n', n+1);
fprintf('Blocking probability HD (%%) = %.6f +- %.6f\n', media_HD, term_HD);
fprintf('Blocking probability 4K (%%) = %.6f +- %.6f\n', media_4K, term_4K);
```

```
fprintf('\nWhen 1 server fails (%d servers):\n', n);
fprintf('Blocking probability HD (%%) = %.6f +- %.6f\n', media_HD_fail, term_HD_fail);
fprintf('Blocking probability 4K (%%) = %.6f +- %.6f\n', media_4K_fail, term_4K_fail);
```

Após alguns testes com o W a 36800, obtivemos este resultado:

```
With 7 servers:
Blocking probability HD (%) = 0.000000 +- 0.000000
Blocking probability 4K (%) = 0.000000 +- 0.000000

When 1 server fails (6 servers):
Blocking probability HD (%) = 0.224768 +- 0.062116
Blocking probability 4K (%) = 0.386475 +- 0.098542
```

Como podemos ver, com 7 servidores a probabilidade de bloqueio é inferior a 0.1%, e, caso algum falhe, a probabilidade é menor que 1% já tendo em consideração o erro.

Task 3

3.a. Compute the ASs to connect the server farms such that the total cost of the Internet connections is minimized (see Appendix E). The solution must guarantee that the shortest path from any Tier-2 or Tier3 AS to the closest server farm has no more than 1 intermediate AS. What is the total Internet connections cost of the solution? How many server farms are required and in which ASs they must be connected to? Analyse the selected ASs and take conclusions.

Para a resolução desta alínea, tivemos em conta o apêndice E, fazendo um ciclo *for* para obter o custo mínimo da solução, associando a cada *server farm* o custo da sua conexão de *internet*.
Posteriormente, utilizamos 2 ciclos para iterar por todos os server farms e verificar se o seu shortest path era igual ou inferior a 3, pois apenas poderiam ter um AS intermédio entre eles.
Por fim, definimos os identificadores dos AS's *Tier-2* e *Tier-3.*
A linha 1 (matrix) importa a matriz G fornecida no apêndice D.

```
matrix;
s = G(:,1);
t = G(:,2);
D = graph(s,t);
```

```
plot(D);

I = zeros(40);
C = zeros(1,40);

r = 6:1:40; %tier-2 and tier-3 interfaces
c = [12 12 12 12 12 12 12 12 12 12 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8]; %cost
n = length(r);

fid = fopen('3a.lp','wt'); %open file

fprintf(fid,'Minimize\n');
for i=1:n
    fprintf(fid,' + %d x%d',c(i),r(i));  %print the identifier and the respective cost
end
fprintf(fid,'\nSubject To\n');

for j = 6:1:40
    for i = 6:1:40
        P = shortestpath(D,j,i);  % calculate the nodes from the shortest path
        if (length(P) <= 3)          % if the nodes quantity is less or equal to three
            fprintf(fid,' + x%d', i);
        end
    end
    fprintf(fid,' >= 1 \n');
end


fprintf(fid,'Binary\n');
for i=1:n
    fprintf(fid,' x%d\n',r(i));    %print all the identifiers
end


fprintf(fid,'End\n');
fclose(fid);
```

Ao correr este script, obtivemos o seguinte ficheiro (3a.lp):

```
Minimize
 + 12 x6 + 12 x7 + 12 x8 + 12 x9 + 12 x10 + 12 x11 + 12 x12 + 12 x13 + 12 x14 + 12 x15 + 8 x16 + 8 x17 + 8 x18 + 8 x19 +
8 x20 + 8 x21 + 8 x22 + 8 x23 + 8 x24 + 8 x25 + 8 x26 + 8 x27 + 8 x28 + 8 x29 + 8 x30 + 8 x31 + 8 x32 + 8 x33 + 8 x34 + 8
x35 + 8 x36 + 8 x37 + 8 x38 + 8 x39 + 8 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
Binary
 x6
 x7
 x8
 x9
 x10
 x11
 x12
 x13
 x14
 x15
 x16
 x17
 x18
 x19
 x20
 x21
 x22
 x23
 x24
 x25
 x26
 x27
 x28
 x29
 x30
 x31
 x32
 x33
 x34
 x35
 x36
 x37
 x38
 x39
 x40
End
```

Ao introduzirmos o ficheiro 3a.lp no servidor, obtivemos a seguinte solution file:

```
# 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
```

Ao analisar o resultado, concluímos que a solução de menor custo inclui 2 AS's de *Tier-2* (AS's 9 e 13) e 3 AS's de *Tier-3* (AS's 16, 21 e 30), obtendo um custo total de 48.

3.b. Use simulator2 to determine the total number of required servers and the appropriate reservation W to provide a blocking probability of at most 1% for movie requests of both formats. Present the simulation results with confidence intervals guaranteeing that both intervals are fully below the required blocking probability value. What was the required stopping criterion value of the simulations?

Nesta alínea, mais uma vez adotamos uma abordagem tentativa erro. Após definirmos os diferentes argumentos do simulador previamente fornecidos (lambda, S), tentámos descobrir qual seria o correto número de servidores.

Chegámos à conclusão que o número mínimo, e portanto ideal de servidores é 76, visto que tanto a probabilidade de bloqueio de filmes HD como a probabilidade de bloqueio em filmes 4K é inferior a 1%.
Concluímos também que uma reserva de recursos de 52200Mbps para filmes 4K mantém as probabilidades de bloqueio de ambos os formatos de filmes equilibrados. Por fim, usamos um critério de paragem de 100000 visto que reduz bastante o erro.

```
clear all    % Clears the cache

% Max blocking probability = 1%

R = 100000;     % Stopping criterion
lambda = (5000*10+2500*25) / 24; % Requests per hour
p = 0.3;    % Probablity of being a 4K movie
alfa = 0.1;     % 90% confidence interval
W = 52200;  % Resource reservation for 4K movies (Mbps)
fname = 'movies.txt';   % Textfile name
N = 10;     % Number of simulations

n = 76;     % Number of servers
S = 1000;   % Server capacity

for i= 1:N
    [b_hd(i) b_4k(i)] = simulator2(lambda,p,n,S,W,R,fname)
end

media_HD = mean(b_hd);
media_4K = mean(b_4k);

term_HD = norminv(1-alfa/2)*sqrt(var(b_hd)/N);
term_4K = norminv(1-alfa/2)*sqrt(var(b_4k)/N);

fprintf('\nWith %d servers:\n', n);
fprintf('Blocking probability HD (%%) = %.6f +- %.6f\n', media_HD, term_HD);
fprintf('Blocking probability 4K (%%) = %.6f +- %.6f\n', media_4K, term_4K);
```

Tendo isto em conta, obtivemos os seguintes resultados:

```
With 76 servers:
Blocking probability HD (%) = 0.710380 +- 0.087491
Blocking probability 4K (%) = 0.618629 +- 0.153505
```

3.c. To define the final solution (i.e., how many servers are put in operation on each server farm), split the total number of servers determined in 3.b by the ASs determined in 3.a in a proportion as close as possible to the number of subscribers that are closer to each server farm. How many servers are installed on each server farm? Analyse the final solution and take conclusions.

Para este exercício, começamos por calcular o número de subscritores que vão estar associados a cada *server farm.* Posteriormente, através de uma regra de 3 simples obtivemos o número de servidores necessários em cada *server farm*. Como é fisicamente impossível ter um número não inteiro de servidores, arredondamos o número para cima, através da função *ceil* para obter o resultado final.

```
n9 = [9, 10, 22, 23, 24, 25, 26, 27];   % Closest AS to AS 9
n9_subs = 2 * 5000 + 6 * 2500;   % Number of subscribers close to AS 9
n13 = [13, 14, 33, 34, 35, 36, 37, 38];    % Closest AS to AS 13
n13_subs = 2 * 5000 + 6 * 2500;   % Number of subscribers close to AS 13
n16 = [6, 15, 16, 17, 18, 19, 39, 40];    % Closest AS to AS 16
n16_subs = 2 * 5000 + 6 * 2500;   % Number of subscribers close to AS 16
n21 = [7, 8, 20, 21];     % Closest AS to AS 21
n21_subs = 2 * 5000 + 2 * 2500;   % Number of subscribers close to AS 21
n30 = [11, 12, 28, 29, 30, 31, 32];   % Closest AS to AS 30
n30_subs = 2 * 5000 + 5 * 2500;    % Number of subscribers close to AS 30

% Total number of subscribers
subscribers = n9_subs + n13_subs + n16_subs + n21_subs + n30_subs;

n9_servers = n9_subs * 76 / subscribers;     % Necessary servers on AS 9
n13_servers = n13_subs * 76 / subscribers; % Necessary servers on AS 13
n16_servers = n16_subs * 76 / subscribers; % Necessary servers on AS 16
n21_servers = n21_subs * 76 / subscribers; % Necessary servers on AS 21
n30_servers = n30_subs * 76 / subscribers; % Necessary servers on AS 30

% Total number of servers necessary
total_servers = ceil(n9_servers) + ceil(n13_servers) + ceil(n16_servers) +
ceil(n21_servers) + ceil(n30_servers);
```

```
AS 9  needs 17 servers (16.8889).
AS 13 needs 17 servers (16.8889).
AS 16 needs 17 servers (16.8889).
AS 21 needs 11 servers (10.1333).
AS 30 needs 16 servers (15.2000).
Total servers needed: 78.
```

Concluímos que serão necessários 78 servidores em vez dos 76 calculados na alínea b).

Pedro Gonçalves 88859
Pedro Silva 89228