Relatório Guião 2
Desempenho e Dimensionamento de Redes

Dário Matos 89298
Pedro Almeida 89205

Maio de 2021

Pergunta 2

Simulador 2

O primeiro passo para resolver a tarefa 2 é desenvolver uma função MATLAB (simulator2), que simule a arquitetura de serviço baseada numa *server farm* com *n* servidores, onde cada servidor tem uma capacidade de *S* Mbps. A *server farm* fornece filmes em dois formatos: HD com uma taxa de transferência de 5 Mbps; e 4K com uma taxa de transferência de 25 Mbps.

Código:

```
function [b_hd b_4k]= simulator2(lambda, p, n, S, W, R, fname)
    % lambda - movies request rate (in requests/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 4Kmovies(in Mbps)
    % R - number of movie requests to stop simulation
    % fname - file name with the duration (in minutes) of the items

    C = n * S;

    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 a HD movie transmission
    DEPARTURE_4K = 2;  %termination of a 4K movie transmission

    %State variables initialization:
    STATE= zeros(1, n);
    STATE_HD= 0;

    %Statistical counters initialization:
    NARRIVALS= 0;   % total number of movie request up to current time
instant
    REQUESTS_HD= 0; % number of HD movie requests up to current time
instant
    REQUESTS_4K= 0; % number of 4K movie requests up to current time
instant
    BLOCKED_HD= 0;  % number of blocked HD movie requests up to current
time instant
    BLOCKED_4K= 0;  % number of blocked 4K movie requests up to current
```

```
time instant

     %Simulation Clock and initial List of Events:
    Clock= 0;
    EventList= [ARRIVAL exprnd(invlambda) 0];

     while NARRIVALS < R
        event= EventList(1,1);  % get event type from list
        Clock= EventList(1,2);  % get clock
        server_id = EventList(1,3); % get id from the last server
involved
        EventList(1,:)= []; % clear event

        % chegou um filme
        if event == ARRIVAL

            % definir evento de arrival e aumentar a contagem
            EventList= [EventList; ARRIVAL Clock+exprnd(invlambda) 0];
            NARRIVALS= NARRIVALS + 1;
            % filme 4k
            if rand() <= p/100

                % valor de ocupação do servidor e o seu índice
                [valor, index] = min(STATE);

                REQUESTS_4K = REQUESTS_4K + 1;

                % verificação da capacidade da interface
                if valor + 25 <= S

                    % definicção do próximo evento como saída(4k) com o
tempo e o servido que o proporciona
                    EventList= [EventList; DEPARTURE_4K
Clock+invmiu(randi(Nmovies)) index];

                    % aumento da ocupação do servidor
                    STATE(index) = STATE(index) + 25;
                else
                    BLOCKED_4K= BLOCKED_4K + 1;
                end
            % filme HD
            else

                % valor de ocupação do servidor e o seu índice
                [valor, index] = min(STATE);
```

```
                REQUESTS_HD=REQUESTS_HD + 1;

                % verificação da capacidade da interface e do limite da
conexão da internet da server farm
                if (valor + 5 <= S) && (STATE_HD + 5 <= C - W)

                    % definição do próximo evento como saída(4k) com o
tempo e o servido que o proporciona
                    EventList= [EventList; DEPARTURE_HD
Clock+invmiu(randi(Nmovies)) index];


                    % aumento de contador de tráfego HD e da ocupação do
servidor
                    STATE_HD = STATE_HD + 5;
                    STATE(index) = STATE(index) + 5;
                else
                    BLOCKED_HD = BLOCKED_HD + 1;
                end
            end
        % filme terminou, libertar recursos
        else
            if event == DEPARTURE_HD

                % diminuição da ocupação do counter total(hd) e do
servidor responsável
                STATE(server_id) = STATE(server_id) - 5;
                STATE_HD = STATE_HD - 5;
            elseif event == DEPARTURE_4K
                STATE(server_id) = STATE(server_id) - 25;
            end
        end
        EventList= sortrows(EventList,2);
     end

    b_hd= 100*BLOCKED_HD/REQUESTS_HD;    % blocking probability of HD
movie requests %
    b_4k= 100*BLOCKED_4K/REQUESTS_4K;    % blocking probability of 4K
movie requests %
end
```

Pergunta 2.a

Código:

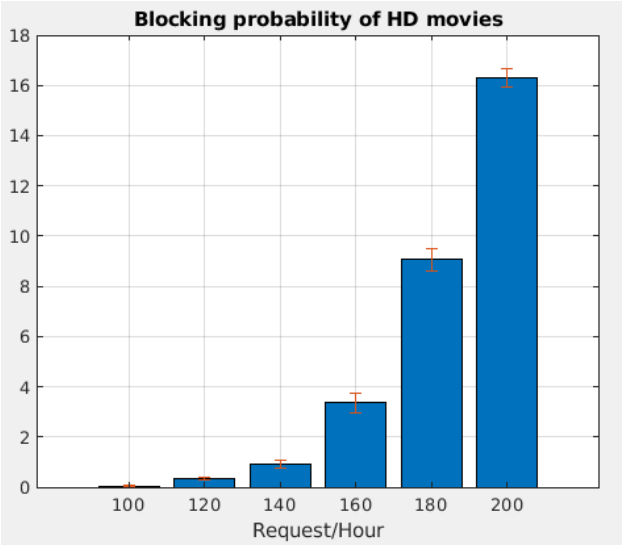
```
lambda = [100 120 140 160 180 200];
p = 20;
n = 10;
S = 100;
W = 0;
R = 10000;
fname = 'movies.txt';

% number of simulations
N = 10;

% vectors with N simulation results
block_hd = zeros(1,N);
block_4k = zeros(1,N);

mediaBlock_hd = zeros(1,size(lambda, 2));
termBlock_hd = zeros(1,size(lambda, 2));

mediaBlock_4k = zeros(1,size(lambda, 2));
termBlock_4k = zeros(1,size(lambda, 2));

for i = 1:size(lambda, 2)
    for it= 1:N
        [block_hd(it), block_4k(it)] = simulator2(lambda(i), p, n, S, W,
R, fname);
    end

    %90% confidence interval%
    alfa= 0.1;

    mediaBlock_hd(i) = mean(block_hd);
    termBlock_hd(i) = norminv(1-alfa/2)*sqrt(var(block_hd)/N);

    mediaBlock_4k(i) = mean(block_4k);
    termBlock_4k(i) = norminv(1-alfa/2)*sqrt(var(block_4k)/N);

end

figure(1)
bar(lambda, mediaBlock_hd)
title('Blocking probability of HD movies')
xlabel('Request/Hour')
```

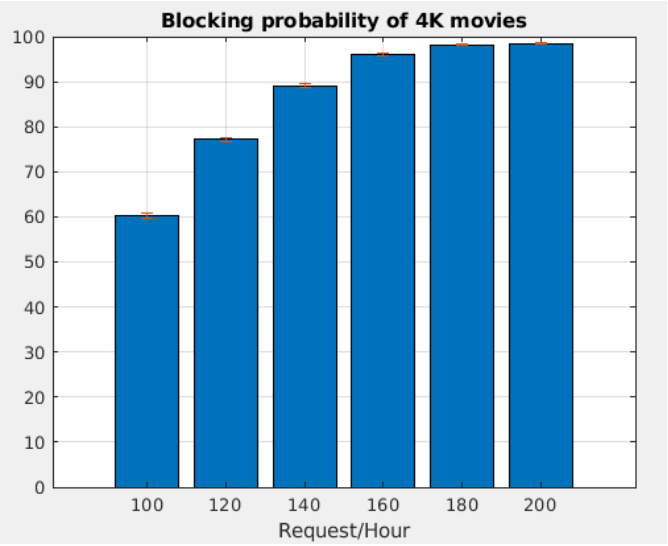
```
hold on
er = errorbar(lambda, mediaBlock_hd, termBlock_hd, termBlock_hd);
er.LineStyle = 'none';
hold off
grid on

figure(2)
bar(lambda, mediaBlock_4k)
title('Blocking probability of 4K movies')
xlabel('Request/Hour')
hold on
er = errorbar(lambda, mediaBlock_4k, termBlock_4k, termBlock_4k);
er.LineStyle = 'none';
hold off
grid on
```

Análise:

O primeiro passo é a criação de listas para guardar os resultados (tanto de probabilidades de bloqueio de todas as simulações, como as suas médias e valores de erro).
Seguidamente, efetuar, para cada valor de pedidos por hora (lambda), o número de simulações previamente definido, calcular a média dos resultados obtidos para cada valor de **lambda** e o erro para cada tipo de transmissão (HD/4K) associado.

Resultados:





Conclusões:

A partir dos gráficos acima apresentados é possível concluir que, naturalmente, a probabilidade de bloqueio, tanto para transmissões HD como 4K, é mais elevada quanto

maior for o número de pedidos por hora. É importante notar também que para os valores de **lambda** referidos, o máximo de probabilidade de bloqueio de transmissões HD é pouco superior a 16%, sendo que relativamente às 4k chega a ser um valor muito próximo da totalidade (100%). Este facto deve-se ao facto de estas transmissões requererem taxa de transmissão de 25Mbps e não ser feita qualquer reserva de recursos para filmes 4K.

Pergunta 2.b

Código:
De modo a não repetir o código já apresentado anteriormente, apenas serão explicitadas as alterações feitas.

É feita a simulação para três configurações diferentes da *server farm*:

```
[block_hd(it), block_4k(it)] = simulator2(lambda(i), p, 10, 100, W, R,
fname);
[block_hd2(it), block_4k2(it)] = simulator2(lambda(i), p, 4, 250, W, R,
fname);
[block_hd3(it), block_4k3(it)] = simulator2(lambda(i), p, 1, 1000, W, R,
fname);
```

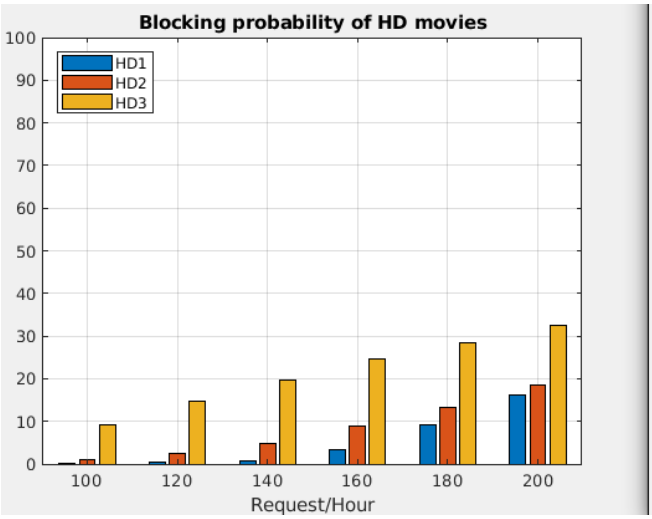
Geração dos gráficos de colunas com múltiplas colunas:

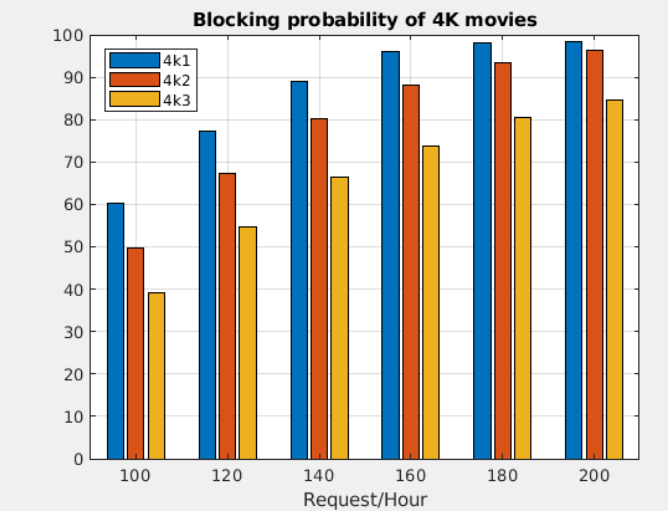
```
figure(3)
bar(lambda,[mediaBlock_hd; mediaBlock_hd2; mediaBlock_hd3])
title('Blocking probability of HD movies')
xlabel('Request/Hour')
legend('HD1', 'HD2', 'HD3', 'Location', 'northwest')
ylim([0 100])
grid on
```

Análise:

Em semelhança à pergunta 2.a, o procedimento inicia por criar as listas para reservar os resultados. Todo o algoritmo é idêntico, tirando o facto de serem efetuadas simulações para as configurações do enunciado, todas correspondentes a um total de 1000 Mbps de taxa transmissão.

Resultados:





Conclusões:

Através dos gráficos abaixo apresentados, é possível concluir que, para transmissões HD, o valor de probabilidade de bloqueio é mais elevado para uma configuração com menos servidores possível, com uma maior capacidade de transmissão nas interfaces. Ainda é possível concluir também que, para diferentes valores de pedidos por hora, o crescimento das probabilidades de bloqueio é superior para configurações com vários servidores do que para a configuração de 1 só servidor.

Quanto às transmissões 4K, o crescimento de probabilidade de bloqueio é semelhante para as 3 configurações, para os diferentes valores de pedidos. Contudo, a configuração que apresenta menores probabilidades de bloqueio, para qualquer **lambda**, é a terceira, indicando que para transmissões de maior taxa, existe menor bloqueio com um número reduzido de servidores e maior capacidade da interface em si.

Pergunta 2.c

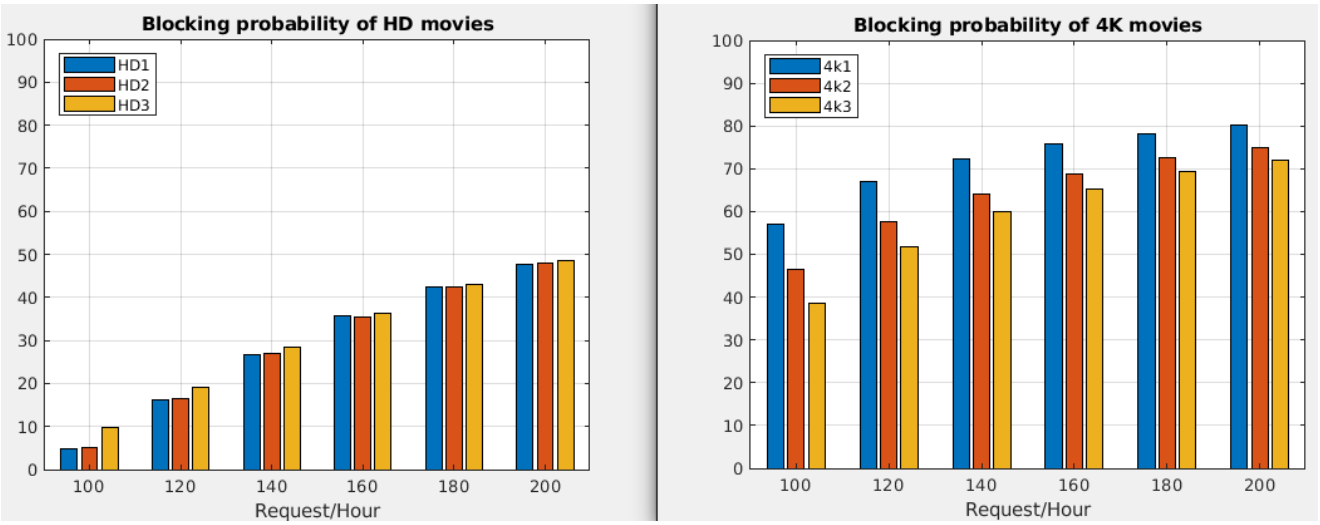
Código:

A única alteração necessária é atualizar a reservação de recursos de 0 para 400 Mbps.

```
W = 400;  % c
```

Análise:
Em relação à pergunta anterior, a única diferença a notar no procedimento é uma reserva de recursos de 400 Mbps.

Resultados:



Conclusões:

Através dos gráficos apresentados, é possível concluir que existe maior probabilidade de bloqueio para transmissões HD, devido ao facto de a taxa de transmissão disponível para as mesmas ser mais reduzida. No que toca às transmissões 4K, a probabilidade de bloqueio é menor, naturalmente, pois existe agora uma reserva de recursos específicos para este tipo de transmissão, pelo que não se coloca a hipótese das transmissões HD "consumirem" toda a capacidade disponível.

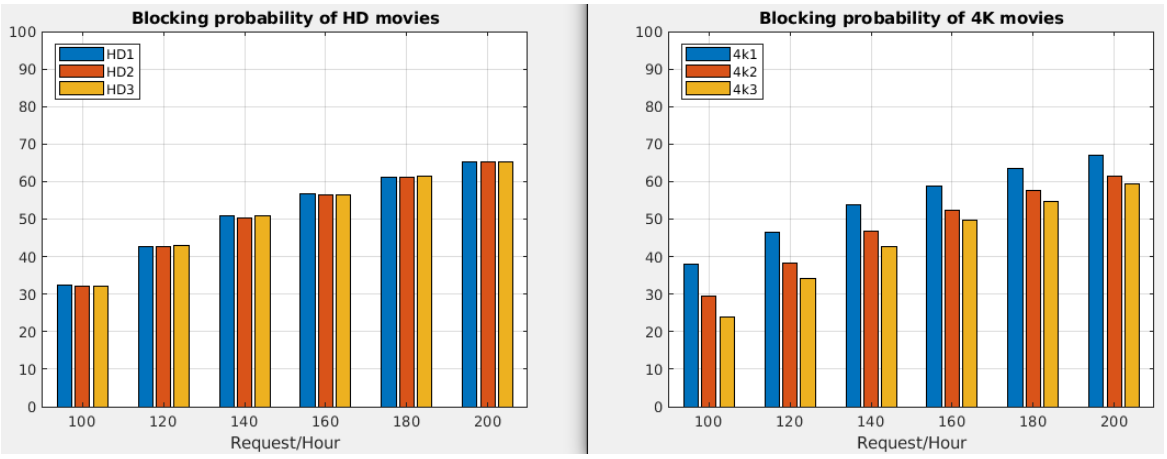
Pergunta 2.d

Código:

Novamente, a única alteração necessária é atualizar a reservação de recursos de 400 para 600 Mbps.

```
W = 600;  % d
```

Resultados:



Conclusões:

À semelhança da questão 2.c, é possível concluir que uma maior reserva de recursos para transmissões 4K aumenta as probabilidades de bloqueio das transmissões HD e diminui as de 4K. Ainda assim, retira-se a conclusão que esta reserva não é aconselhável, pois apresenta uma probabilidade de bloqueio muito elevada para HD (superior a 60% para um número elevado de pedidos por hora), sendo que os bloqueios nas transmissões 4K permanecem elevados (também acima de 60% nos piores casos).
É necessário encontrar um equilíbrio entre o bloqueio dos dois tipos de transmissão, talvez estudando qual o tipo mais requerido pelos clientes e adequando a reserva de recursos 4K à preferência geral.

Pergunta 2.e

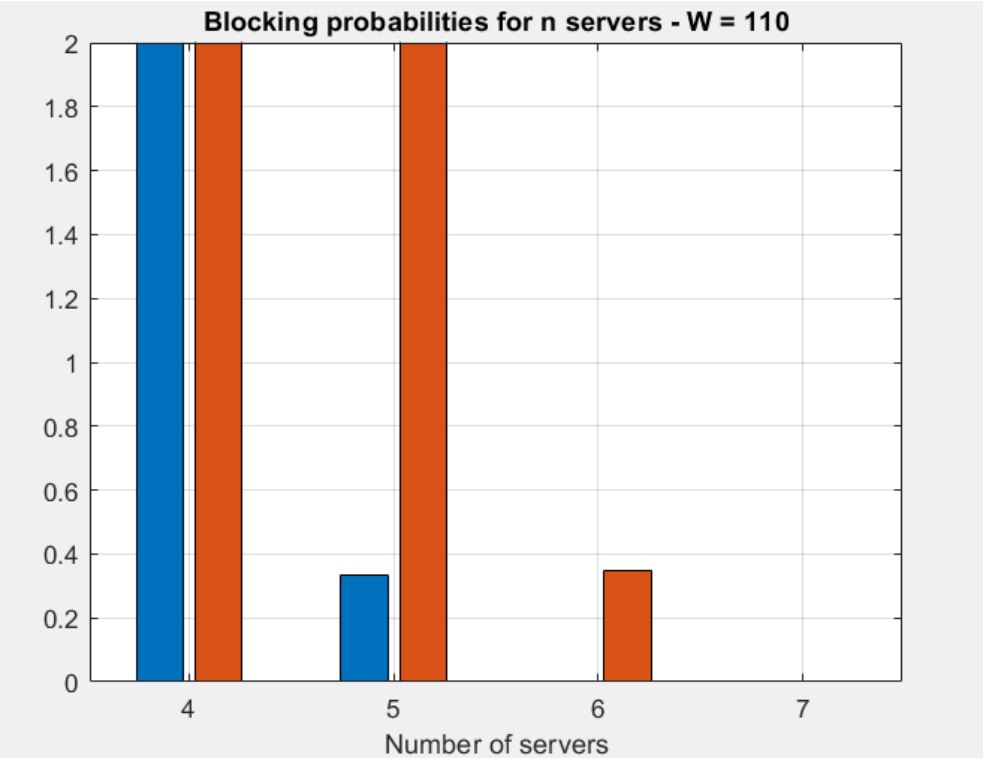
Código:

```
results= 0; %zeros(1,N); %vector with N simulation results
results2= 0; %zeros(1,N); %vector with N simulation results

lambda = 100000/24;

results=zeros(1,4);
results2=zeros(1,4);
n=[4 5 6 7];
W=110;
R=100000;
for i=1:4
    [results(i) results2(i)] =
simulator2(lambda,24,n(i),10000,W,R,'movies.txt');
end

figure(1)

bar(n, [results; results2])
title('Blocking probabilities for n servers - W = 110')
xlabel('Number of servers')

ylim([0 2])


grid on
```

Análise:

Para esta alínea, o procedimento inicia criando as variáveis de armazenamento dos resultados. Seguidamente, define-se a reserva de recursos e o critério de paragem, para depois se efetuarem simulações para vários valores de números de servidores e obter as probabilidades de paragem. A utilização de vários valores de *n* tem como objetivo a verificação da transição, em termos de bloqueios, para o número de servidores ideal, visível num gráfico.

Resultados:



Conclusões:

Do gráfico acima representado é possível concluir que, para manter probabilidades de bloqueio inferiores a 0.1%, o número ideal de servidores é 7. Contudo, para satisfazer a condição de as probabilidades serem inferiores a 1% quando falha um servidor, nestas condições, apenas existe bloqueio de transmissões 4K.

Pergunta 3.a

Código:
Geração do ficheiro *lp*:

```
g = graph(G(:,1), G(:,2));

fid = fopen('ex3.lp', 'wt');

fprintf(fid, 'Minimize\n');
for i= 6:40
   if i <= 15
       fprintf(fid, ' + %f x%d', 12, i);    % tier2
   else
       fprintf(fid, ' + %f x%d', 8, i);     % tier3
   end
end

fprintf(fid, '\nSubject To\n');
for j = 6:40
    for i = 6:40
        [p, d] = shortestpath(g, j, i);
        if (d <= 2)
            fprintf(fid, ' + x%d', i);
        end
    end
    fprintf(fid, ' >= %f\n', 1);
end


fprintf(fid,'Binary\n');
for i = 6:40
    fprintf(fid, ' x%d', i);
end

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

Análise:

Sendo este exercício um problema de otimização pode ser resolvido com *Integer Linear Programming (ILP) Model*.
Com a lista de ASs conectados (G) disponibilizado pelo professor, é construído o grafo (g) com a parte da Internet que é coberta pelo serviço de streaming. De seguida, é escrito o ficheiro com o formato *lp*.
É usado a *keyword* “Minimize” uma vez que se quer encontrar a solução com menor custo.
Já na parte de restrições, é calculada o caminho entre dois ASs através da função auxiliar *shortestpath* e caso a distância (d) entre os ASs que se estão a considerar (*j* e *i*) seja menor ou igual que 2, isto é, ou é o mesmo AS (d=0), ou está diretamente ligado (d=1) ou tem apenas um AS intermédio (d=2), então escreve-se esse AS (variável *x* com o número do

AS) no ficheiro. No final da linha acrescenta-se “>= 1” para garantir que o AS *j* está conectado a pelo menos um AS *i* onde está uma *server farm*.
No final do ficheiro *lp* apresentam-se as variáveis binárias utilizadas.

Resultados:

Com o ficheiro *lp* (em anexo) gerado, este foi submetido no *solver Gurobi*.
Os resultados obtidos foram os seguintes:

- O custo total das ligações da Internet da solução: 48
- Número de *server farms* necessárias: 5
- ASs em que as server farms estão conectadas: 9, 13, 16, 21, 30

Conclusões:

Verificando os ASs selecionados para conectar as *server farms*, pode ser confirmado que todos os ASs estão no máximo a um AS intermédio de chegar a uma *server farm*, cumprindo assim as restrições impostas. Apenas esta tarefa seria de uma complexidade elevada caso fosse resolvida por um humano. Complicando ainda mais para encontrar a solução ótima.

Pergunta 3b

Código:

```
r_tier2 = 10 * 5000;    % numero de AS tier2 * subscribers
r_tier3 = 25 * 2500;    % numero de AS tier3 * subscribers

lambda = ((r_tier2 + r_tier3) / 24);    % lambda - movies request rate
(in requests/hour)
p = 30;                 % p - percentage of requests for 4K movies (in %)
n = 76;                 % n - number of servers
S = 1000;               % S - interface capacity of each server(in Mbps)
W = 52150;              % W - resource reservation for 4K movies(in Mbps)
R = 50000;              % R - number of movie requests to stop simulation
fname = 'movies.txt'; % fname - file name with the duration (in minutes)
of the items

N = 5;  % number of simulations

block_hd = zeros(1,N);
block_4k = zeros(1,N);

for it= 1:N
    [block_hd(it), block_4k(it)] = simulator2(lambda, p, n, S, W, R,
```

```
fname);
end

% 90 confidence interval %
alfa= 0.1;

mediaBlock_hd = mean(block_hd);
termBlock_hd = norminv(1-alfa/2)*sqrt(var(block_hd)/N);

mediaBlock_4k = mean(block_4k);
termBlock_4k = norminv(1-alfa/2)*sqrt(var(block_4k)/N);

fprintf('\nNumber of servers: %.0f\n', n);
fprintf('Reservation for 4K: %.0f\n', W);
fprintf('block_hd: %.4f +- %.4f\n', mediaBlock_hd, termBlock_hd);
fprintf('block_4k: %.4f +- %.4f\n', mediaBlock_4k, termBlock_4k);
```

Análise:

O código apresentado é muito semelhante ao apresentado da Pergunta 2, onde se configura a *server farm* e se simula com auxílio do *simulador2*, com diferença na apresentação de resultados onde agora apenas se imprime os resultados na consola.

Resultados:

Para chegar à solução, foi feito um processo de tentativa-erro.
O primeiro passo foi aumentar o número de servidores até que a probabilidade de bloqueio de filmes no formato HD baixasse de 1%. De seguida aumentava-se sucessivamente a reserva de recursos para filmes 4K com o objetivo de diminuir a sua probabilidade de bloqueio mas quando se ultrapassa um certo limiar, a probabilidade de filmes HD aumentava significativamente. Neste momento, voltava se ao primeiro passo.

A configuração da *server farm* para a qual se conseguiu que o bloqueio de filmes de ambos os formatos fosse abaixo de 1% foi:

```
Number of servers: 76
Reservation for 4K: 52150
block_hd: 0.3168 +- 0.1328
block_4k: 0.5583 +- 0.2014
```

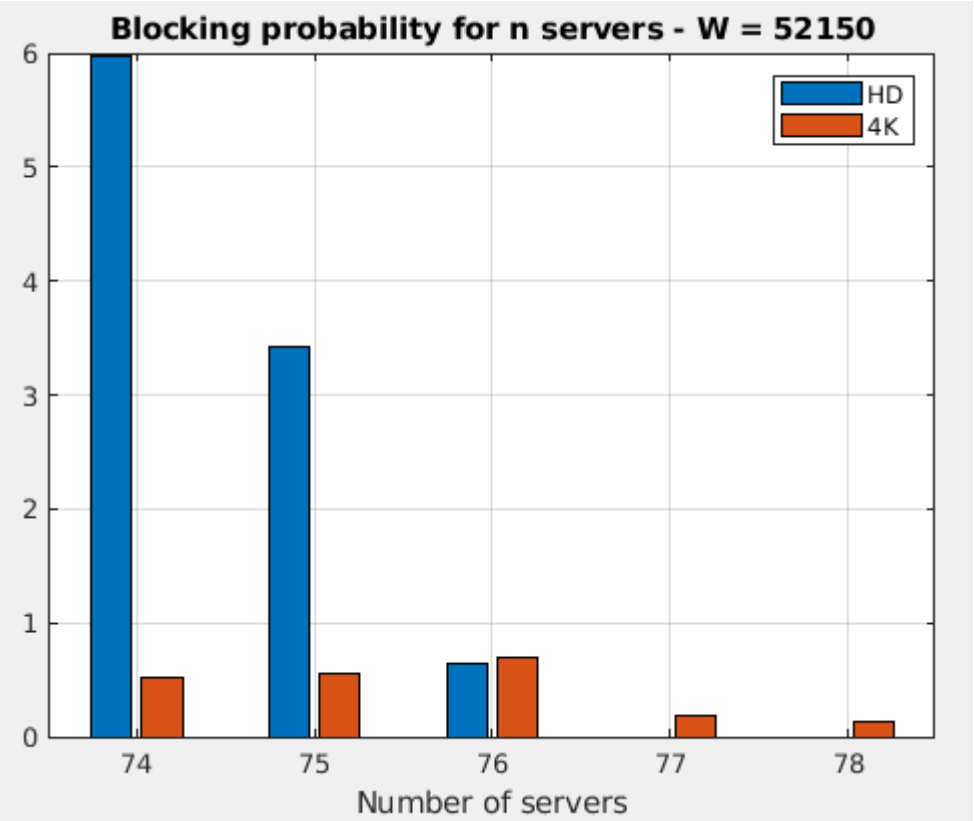
Para (n-1) servidores, ambas as probabilidades de bloqueio estão acima de 1% (somando erro), logo mesmo que se aumente a reserva de recursos, nunca se vai conseguir baixar as duas probabilidades.

```
Number of servers: 75
Reservation for 4K: 52150
block_hd: 3.0569 +- 0.2339
block_4k: 0.7546 +- 0.3004
```

O número necessário de filmes requisitados para parar a simulação utilizado foi 50000.

Depois de encontrada a solução para a qual a percentagem de bloqueio de ambos os formatos de filmes fosse menor que 1%, gerou-se um gráfico de modo a facilitar a visualização da influência que o número de servidores tem no bloqueio de filmes. Na geração do gráfico apenas foi feita uma simulação e não se apresentou o fator de confiança.

```
block_hd = zeros(1,5);
block_4k = zeros(1,5);

n = [74 75 76 77 78];
for i=1:5
    [block_hd(i), block_4k(i)] = simulator2(lambda, p, n(i), S, W, R,
fname);
end

figure(1)
bar(n, [block_hd; block_4k])
title('Blocking probability for n servers - W = 52150')
xlabel('Number of servers')
legend('HD', '4K')
grid on
```



Conclusões:
Com a resolução deste exercício pode-se concluir que para além de se decidir onde colocar as *server farms* (calculado no exercício anterior) é também de grande importância definir os requisitos pretendidos e calcular a configuração ideal para alcançar tais requisitos. Uma solução simples é aumentar o número de servidores. Como se pode verificar ao analisar o gráfico apresentado acima, ao aumentar o número de servidores, baixa a probabilidade de bloqueio dos pedidos dos filmes. Contudo, esta solução não é a mais económica. Assim, é de grande importância calcular a melhor configuração possível (neste caso a reserva de recursos para filmes 4K) podendo este trabalho poupar muito dinheiro às Operadoras de Rede.

Pergunta 3.c

Código:
Calcular número de servidores por AS:

```
% clientes por AS
n9 = 7*2500 + 4*5000;
n13 = 6*2500 + 5*5000;
n16 = 6*2500 + 3*5000;
```

```
n21 = 3*2500 + 3*5000;
n30 = 6*2500 + 3*5000;

% total de clientes
totalClients=10*5000+25*2500;

% percentagem de clientes por AS
perc9 = n9/totalClients;
perc13 = n13/totalClients;
perc16 = n16/totalClients;
perc21 = n21/totalClients;
perc30 = n30/totalClients;

% n servidores necessários, calculado no b)
totalServers = 76;

% distribuicao dos servers por AS com base no num de clientes desse AS
servers9 = perc9*totalServers;
servers13 = perc13*totalServers;
servers16 = perc16*totalServers;
servers21 = perc21*totalServers;
servers30 = perc30*totalServers;

totalServersNeeded = round(servers9) + round(servers13) +
round(servers16) + round(servers21) + round(servers30);

realServers9 = round(servers9)*totalServers / totalServersNeeded;
realServers13 = round(servers13)*totalServers / totalServersNeeded;
realServers16 = round(servers16)*totalServers / totalServersNeeded;
realServers21 = round(servers21)*totalServers / totalServersNeeded;
realServers30 = round(servers30)*totalServers / totalServersNeeded;

fprintf('Servers in operation in AS 9: %d\n', round(realServers9));
fprintf('Servers in operation in AS 13: %d\n', round(realServers13));
fprintf('Servers in operation in AS 16: %d\n', round(realServers16));
fprintf('Servers in operation in AS 21: %d\n', round(realServers21));
fprintf('Servers in operation in AS 30: %d\n', round(realServers30));
fprintf('Total servers: %d\n', totalServers);
```

Gráfico:

```
clientes = [perc9 perc13 perc16 perc21 perc30];
servers = [round(realServers9) round(realServers13) round(realServers16)
round(realServers21) round(realServers30)];

figure(2)
```

```
bar([9 13 16 21 30], clientes)
title('Percentage of clients per AS')
xlabel('AS number')
grid on

figure(3)
bar([9 13 16 21 30], servers)
title('Number of servers per AS')
xlabel('AS number')
grid on
```

Análise:

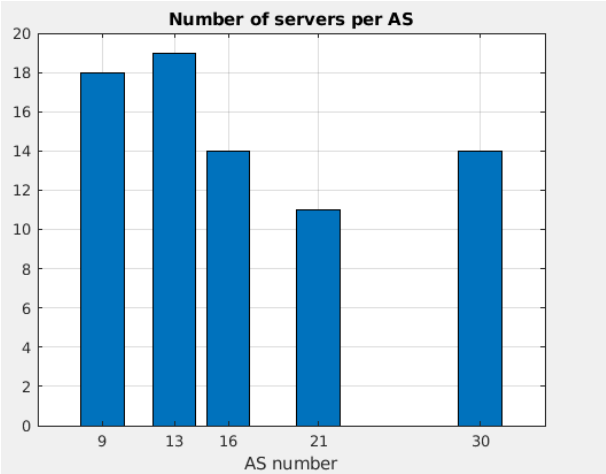
Para calcular o número de servidores ideal em cada *server farm* nos ASs selecionados no exercício 3.a, o primeiro passo foi calcular a percentagem de clientes de cada AS sobre todos os clientes. De seguida, foram calculados os servidores necessários para cobrir os clientes representados nas percentagens.

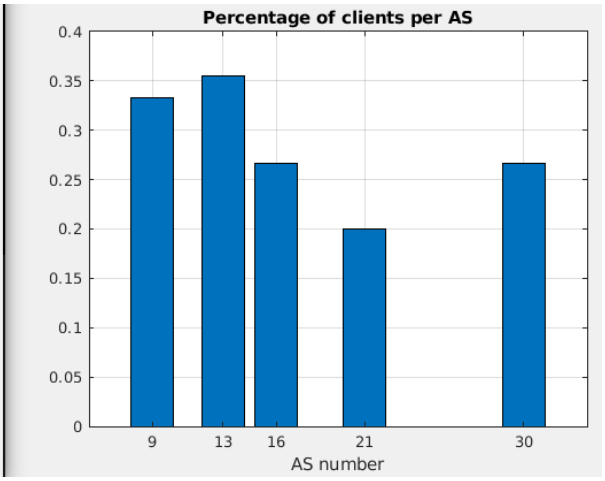
Contudo, e como diferentes *server farms* (utilizadas) cobriam o tráfego de *server farms* (não utilizadas) repetidas, a soma das percentagens é superior a 100%. Assim, o cálculo de servidores necessários para a distribuição de clientes resultava num número superior ao disponível, sendo posteriormente necessário efetuar um cálculo de proporcionalidade direta para limitar os servidores utilizados aos possíveis utilizar (máximo de 76).

Os gráficos apresentados são apenas de apoio para as conclusões tiradas.

Resultados:

```
Servers in operation in AS 9: 18
Servers in operation in AS 13: 19
Servers in operation in AS 16: 14
Servers in operation in AS 21: 11
Servers in operation in AS 30: 14
Total servers: 76
```





Conclusões:

Para além de ser de grande importância calcular a configuração ideal e assim encontrar o número mínimo de servidores necessários para os requisitos impostos, é também crítico dividir esses servidores de forma pensada. Nos ASs com mais clientes deverão estar mais servidores em funcionamento de modo a suportar mais pedidos de filmes. Como se pode verificar na análise dos gráficos acima, o número de servidores calculado para cada ASs é diretamente proporcional ao número de clientes desse mesmo AS.

Anexo

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 >= 1.000000
+ x6 + x7 + x8 + x16 + x17 + x18 + x19 + x20 + x21 >= 1.000000
+ x7 + x8 + x9 + x10 + x20 + x21 + x22 + x23 + x24 + x25 >= 1.000000
+ x8 + x9 + x10 + x11 + x21 + x22 + x23 + x24 + x25 + x26 + x27 >= 1.000000
+ x8 + x9 + x10 + x11 + x12 + x13 + x22 + x23 + x24 + x25 + x26 + x27 + x28 + x29 + x30 >= 1.000000
+ x9 + x10 + x11 + x12 + x13 + x26 + x27 + x28 + x29 + x30 >= 1.000000
+ x10 + x11 + x12 + x13 + x14 + x30 + x31 + x32 >= 1.000000
+ x10 + x11 + x12 + x13 + x14 + x33 + x34 + x35 + x36 + x37 + x38 >= 1.000000
+ x6 + x12 + x13 + x14 + x15 + x33 + x34 + x35 + x36 + x37 + x38 >= 1.000000
+ x6 + x14 + x15 + x16 + x39 + x40 >= 1.000000
+ x6 + x7 + x15 + x16 + x17 + x18 + x19 + x39 + x40 >= 1.000000

```
 + x6 + x7 + x16 + x17 + x18 + x19 >= 1.000000
 + x6 + x7 + x16 + x17 + x18 + x19 >= 1.000000
 + x6 + x7 + x16 + x17 + x18 + x19 + x20 >= 1.000000
 + x6 + x7 + x8 + x19 + x20 + x21 >= 1.000000
 + x7 + x8 + x9 + x20 + x21 + x22 >= 1.000000
 + x8 + x9 + x10 + x21 + x22 + x23 + x24 + x25 >= 1.000000
 + x8 + x9 + x10 + x22 + x23 + x24 + x25 >= 1.000000
 + x8 + x9 + x10 + x22 + x23 + x24 + x25 >= 1.000000
 + x8 + x9 + x10 + x22 + x23 + x24 + x25 >= 1.000000
 + x9 + x10 + x11 + x26 + x27 >= 1.000000
 + x9 + x10 + x11 + x26 + x27 + x28 + x29 + x30 >= 1.000000
 + x10 + x11 + x27 + x28 + x29 + x30 >= 1.000000
 + x10 + x11 + x27 + x28 + x29 + x30 >= 1.000000
 + x10 + x11 + x12 + x27 + x28 + x29 + x30 + x31 + x32 >= 1.000000
 + x12 + x30 + x31 + x32 >= 1.000000
 + x12 + x30 + x31 + x32 >= 1.000000
 + x13 + x14 + x33 + x34 + x35 >= 1.000000
 + x13 + x14 + x33 + x34 + x35 >= 1.000000
 + x13 + x14 + x33 + x34 + x35 >= 1.000000
 + x13 + x14 + x36 + x37 + x38 >= 1.000000
 + x13 + x14 + x36 + x37 + x38 >= 1.000000
 + x13 + x14 + x36 + x37 + x38 >= 1.000000
 + x15 + x16 + x39 + x40 >= 1.000000
 + x15 + x16 + x39 + x40 >= 1.000000
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
```