Desempenho e Dimensionamento de Redes

Relatório 3

Blocking performance of video-streaming services

Autores:
João Quintanilha 68065
Bruno Henriques 68314

Professor:
Amaro de Sousa

May 15, 2018

Índice

1 Questions

1.1 Question 2

1.1.1 Question a

Assuming that the movies duration is an exponential distributed random variable with the same average duration of the items catalogue, i.e., 1/u = 86.3 minutes. In this case, the system can be modelled by an M/M/m/m queuing system. The theoretical values of the blocking probability and the average connection occupation using both algorithms on Appendix A, are shown on the table bellow.

Table 1: Theoretical values using algorithm from appendix A

Case	$\lambda/C/M$	Block Prob. (%)	Avg connect occ. (Mbps)
A	10/100/4	0.32	57.35
B	20/100/4	22.1	89.62
C	30/100/4	44.6	95.47
D	40/100/4	57.7	97.21
E	10/100/10	39.0	87.69
F	20/100/10	66.8	95.35
G	30/100/10	77.4	97.19
H	40/100/10	82.9	98.00
I	100/1000/4	0.0	575.33
J	200/1000/4	14.8	979.53
K	300/1000/4	42.3	994.63
L	400/1000/4	56.6	996.96
M	100/1000/10	31.8	979.82
N	200/1000/10	65.4	994.76
O	300/1000/10	76.8	997.01
P	400/1000/10	82.6	997.91

1.1.2 Question b

After developing a MATLAB script to run the simulator 10 times with the given parameters, we collected the results shown on table 2.

With these practical results and the analytic results collected on Question 2.a, we can confirm that both data are almost or exactly the same.

Checking the table 3 with the 90% confidence interval for both parameters, we can see that the practical and theoretical values differ at a extremely low percentage point, which means that the M/M/m/m queuing system is a good approximation of the simulated system.

Table 2: Simulation for appendix B with 90% confidence interval

Case	$\lambda/C/M$	Block Prob. (%)	Avg connect occ. (Mbps)
A	10/100/4	0.34 +- 1.60e-02	57.3 +- 1.25e-01
B	20/100/4	22.0 +- 1.16e-01	89.6 +- 6.40e-02
C	30/100/4	44.7 +- 8.84e-02	95.5 +- 2.48e-02
D	40/100/4	57.7 +- 1.23e-01	97.2 +- 1.82e-02
E	10/100/10	39.8 +- 1.06e-01	87.7 +- 3.72e-02
F	20/100/10	66.8 +- 8.69e-02	95.3 +- 3.04e-02
G	30/100/10	77.5 +- 6.78e-02	97.2 +- 1.52e-02
H	40/100/10	83.0 +- 5.22e-02	98.0 +- 1.85e-02
I	100/1000/4	0 +- 0.00e+00	576.3 +- 1.34e+00
J	200/1000/4	15.1 +- 2.58e-01	977.1 +- 3.67e-01
K	300/1000/4	42.3 +- 1.23e-01	992.2 +- 7.21e-02
L	400/1000/4	56.6 +- 7.62e-02	994.6 +- 1.07e-01
M	100/1000/10	31.8 +- 2.52e-01	979.4 +- 2.89e-01
N	200/1000/10	65.3 +- 3.63e-02	994.6 +- 5.24e-02
O	300/1000/10	76.8 +- 5.97e-02	996.0 +- 7.10e-02
P	400/1000/10	82.6 +- 6.11e-02	997.1 +- 8.23e-02

1.2 Question 3

1.2.1 Question a

After creating a simulator2 (figure 2 + 3 + 4) similar to the first one but this time adding all the required variables mentioned in Appendix C and conditions mentioned on the question 3, we ran the script (figure 1)with 40 repetitions and a stopping criteria of R = 50000 for all the cases getting the following results on the table 3 below.

```
alfa= 0.1; %intervalo de confiança a 90%
R = 50000;
REP = 10; %%% METER A 40

b = zeros(1,REP);
o = zeros(1,REP);

Mhd = 4; %throughput of hd format
M4k = 10; %throughput of 4k format

for i=1:size(cases,1)

    letter = alphabet(i);
    fprintf('Case %s\n', letter);


      for j=1:REP

        [b(j), o(j)] = simulator2( cases(i,1), cases(i,2), cases(i,3), cases(i,4), Mhd, M4k, R);
      end
      mediaBlockingProb = mean(b);
      BlockingProb4k = mean(o);
      termo1 = norminv(1-alfa/2)*sqrt(var(b)/REP);
      termo2 = norminv(1-alfa/2)*sqrt(var(o)/REP);
      fprintf('HD Blocking Probability = %.2e +- %.2e\n', mediaBlockingProb,termo1)
      fprintf('4K Blocking probability = %.2e +- %.2e\n\n', BlockingProb4k, termo2)


 end
```

Figure 1: Script to run all cases for Simulator 2

```
function [bHD,b4K]= simulator2(lambda,S,W,p,Mhd,M4k,R)

  %lambda = request arrival rate (in requests per hour)
  %C= Internet connection capacity (in Mbps)
  C = 100*S; %Mbps
  %M= throughput of each movie (in Mbps)
  %R= stop simulation on ARRIVAL no. R
  p = p/100;   % para ser %
  invlambdaHD=60/(lambda *(1 - p));    %average time between requests (in minutes)
  invlambda4K=60/(lambda * p);        % FOR 4k
  invmiu= load('movies.txt'); %duration (in minutes) of each movie
  Nmovies= length(invmiu);    % number of movies

  %Events definition:
  ARRIVAL_HD = 0; % HD movie request
  ARRIVAL_4K = 1; % 4K movie request

  %{
  for i=1:S
    DEPARTURE_HD(i) = 2; %termination of an HD movie transmission
    DEPARTURE_4K(i) = 3; %termination of a 4K movie transmission
  end
  %}

   DEPARTURE_HD = 2; %termination of an HD movie transmission
   DEPARTURE_4K = 3; %termination of a 4K movie transmission

  %State variables initialization:
  STATE= zeros(1,S);
  STATE_HD = 0;   %total throughput of HD movies in transmission

  %Statistical counters initialization:

  NARRIVALS = 0;    %total number of movie requests
  NARRIVALS_HD = 0; %number of HD movie requests
  NARRIVALS_4K = 0; %number of 4K movie requests
  BLOCKED_HD = 0;   %number of blocked HD movie requests
  BLOCKED_4K = 0;   %number of blocked 4K movie requests

  %Simulation Clock and initial List of Events:
  Clock = 0;
  EventList= [ARRIVAL_HD exprnd(invlambdaHD) 0
              ARRIVAL_4K exprnd(invlambda4K) 0];
  EventList = sortrows(EventList,2);
```

Figure 2: Part 1 of Simulator 2

```
while NARRIVALS < R
    %EventList(1,1)

  event= EventList(1,1);
  Clock= EventList(1,2);
  server = EventList(1,3);
  EventList(1,:)= [];

  if event == ARRIVAL_HD %arrival of HD movie
     %"Arrival HD"
    %find server with lowest load
    server = find(STATE==min(STATE));
    server = server(1);

    EventList = [EventList; ARRIVAL_HD Clock+exprnd(invlambdaHD) 0];
    NARRIVALS_HD = NARRIVALS_HD+1;
    NARRIVALS = NARRIVALS +1;

    if STATE(server) + Mhd <= 100 && STATE_HD + Mhd <= C - W
        STATE(server) = STATE(server) + Mhd;
        STATE_HD = STATE_HD + Mhd;

        EventList= [EventList; DEPARTURE_HD Clock+invmiu(randi(Nmovies)) server];

    else
      BLOCKED_HD= BLOCKED_HD+1;
    end

  elseif event == ARRIVAL_4K %arrival of 4K movie
    %"ARRIVAL 4K"
    %find server with lowest load
    server = find(STATE==min(STATE));
    server = server(1);

    EventList = [EventList; ARRIVAL_4K Clock+exprnd(invlambda4K) 0];
    NARRIVALS_4K= NARRIVALS_4K+1;
    NARRIVALS = NARRIVALS +1;

    if STATE(server) + M4k <= 100
      STATE(server) = STATE(server) + M4k;
      EventList = [EventList; DEPARTURE_4K Clock+invmiu(randi(Nmovies)) server];
    else
      BLOCKED_4K= BLOCKED_4K+1;
    end
```

Figure 3: Part 2 of Simulator 2

```
    elseif event == DEPARTURE_HD %departure of HD movie
        %"DEPARTURE HD"
      STATE(server) = STATE(server) - Mhd;
      STATE_HD = STATE_HD - Mhd;

    elseif event == DEPARTURE_4K %departure of 4K movie
        %"DEPARTURE 4K"
      STATE(server) = STATE(server) - M4k;
    end

    EventList= sortrows(EventList,2);

  end
  bHD= 100*BLOCKED_HD/NARRIVALS_HD; % HD blocking probability in %
  b4K= 100*BLOCKED_4K/NARRIVALS_4K; % 4K blocking probability in %
end
```

Figure 4: Part 3 of Simulator 2

Table 3: HD & 4K Blocking probability and 90% confidence interval for each

Case	$\lambda/S/W/p$	HD Block Prob %	4K Block Prob %
A	10/1/0/20	4.00e+00 +- 4.11e-02	1.15e+01 +- 1.01e-01
B	10/1/30/20	5.49e+00 +- 3.95e-02	1.07e+01 +- 9.65e-02
C	10/1/60/20	2.82e+01 +- 7.08e-02	3.67e+00 +- 7.11e-02
D	10/1/0/40	8.44e+00 +- 6.97e-02	2.19e+01 +- 8.32e-02
E	10/1/30/40	8.55e+00 +- 5.52e-02	2.21e+01 +- 1.04e-01
F	10/1/60/40	1.77e+01 +- 6.05e-02	1.92e+01 +- 8.46e-02
G	30/3/0/20	3.25e-01 +- 1.27e-02	3.38e+00 +- 8.07e-02
H	30/3/120/20	1.56e+00 +- 3.11e-02	2.80e+00 +- 6.65e-02
I	30/3/180/20	2.12e+01 +- 8.96e-02	3.16e-01 +- 2.63e-02
J	30/3/0/40	1.62e+00 +- 2.61e-02	1.40e+01 +- 1.08e-01
K	30/3/120/40	1.64e+00 +- 3.80e-02	1.41e+01 +- 9.66e-02
L	30/3/180/40	6.99e+00 +- 7.36e-02	1.22e+01 +- 8.44e-02

1.2.2 Question b

Assuming there are 20.000 subscribers and 30% of 4K requests it leads us to:

$$\lambda = \frac{20000(req/week)}{7days * 24hours} \simeq 119 \text{ and } p = 30\%$$

We created a little script 5 that assumes the values described above as fixed, and searches for the smallest amount of servers needed for the company to fulfill its requirements. After running it , we conclude that the minimum amount of servers necessary are 13 and an adequate reservation value W to be set in the front-office is between 160 and 180 Mbps.

```matlab
R = 50000;

Mhd = 4; %throughput of hd format
M4k = 10; %throughput of 4k format

% i = nr of servers , Starting on 10
for i=11:25
        % j = w going always from 0 to 200 jumping 20
        for j=0:20:200

         % case with lambda always 119 , servers , W , p
        cases = [ 119 i j 30 ];
        [b, o] = simulator2( cases(1), cases(2), cases(3), cases(4), Mhd, M4k, R);


            % if the difference between the two formats is smaller than 0.1
            if abs(o-b) <= 0.1

                %simulation with one less server (server failure)
                [x, y] = simulator2( cases(1), cases(2)-1, cases(3), cases(4), Mhd, M4k, R);

                % if the difference between the two formats with 1 less server is smaller than 1
                if abs(y-x) <= 1

                fprintf('Servers: %d \n', i)
                fprintf('W: %d \n', + j)
                fprintf('HD Blocking Probability = %.6e\n', b)
                fprintf('4K Blocking probability = %.6e\n\n', o)


                fprintf('Servers: %d \n', i-1)
                fprintf('W: %d \n', + j)
                fprintf('HD Blocking Probability = %.6e\n', x)
                fprintf('4K Blocking probability = %.6e\n\n', y)

                fprintf('---------------------------\n\n')
                end
            end

      end

end
```

Figure 5: Script to find the least amount of servers

1.3 Question 4

1.3.1 Question a

After using the algorithm to create matrix I as follows in 6 we made the following code in figure 7 in order to generate the .lp file represented partially in figure 8. After introducing it to NEOS Gurobi, we obtained the following result from figure 9

```
% Matrix full of -1
I = zeros(40,40) -1;

% Iteration for all AS Tier 2 and 3
for i=6:size(I,2)

    % assigning label 0 to AS j
    I(i,i) = 0;

    for a=0:1
        % for each AS pair(i,j)
        for j=1:size(G,1)
            p1 = I(i,G(j,1));
            p2 = I(i,G(j,2));

            % if one AS has label a and the other AS has label -1
            if (p1 == a && p2 == -1)
                I(i,G(j,2)) = a+1;
            % assign label a+1 to the AS that has label -1
            elseif (p1 == -1 && p2 == a)
                I(i,G(j,1)) = a+1;
            end

        end
    end

end
```

Figure 6: Algorithm for indexing matrix

```
%% gerar ILP
% custos
C(6:15) = 10;
C(16:40) = 8;
% Minimize
fid = fopen('ex3_minimize.lp','wt');
fprintf(fid,'Minimize\n');
for i=6:40
    fprintf(fid,' + %f x%d',C(i),i);
end
% Subject to
fprintf(fid,'\nSubject To\n');
% Constrain (2)
for j=6:40
    for i=6:40
        if I(i,j) > -1
            fprintf(fid,' + y%d,%d',j,i);
        end
    end
    fprintf(fid,' =  1\n');
end
% Constrain (3)
for j=6:40
    for i=6:40
        if I(i,j) > -1
            fprintf(fid,' + y%d,%d - x%d <= 0\n',j,i,i);
        end
    end
end
% Binary
fprintf(fid,'Binary\n');
for i=6:40
    fprintf(fid,' x%d\n',i);
end
for j=6:40
    for i=6:40
        if I(i,j) > -1
            fprintf(fid,' y%d,%d\n',j,i);
        end
    end
end
fprintf(fid,'End\n');
fclose(fid);
```

Figure 7: Code used for generating .lp file

```
Minimize
 + 10.000000 x6 + 10.000000 x7 + 10.000000 x8 + 10.000000 x9 + 1
Subject To
 + y6,6 + y6,7 + y6,14 + y6,15 + y6,16 + y6,17 + y6,18 + y6,19 +
 + y7,6 + y7,7 + y7,8 + y7,16 + y7,17 + y7,18 + y7,19 + y7,20 +
 + y8,7 + y8,8 + y8,9 + y8,10 + y8,20 + y8,21 + y8,22 + y8,23 +
 + y9,8 + y9,9 + y9,10 + y9,11 + y9,21 + y9,22 + y9,23 + y9,24 +
 + y10,8 + y10,9 + y10,10 + y10,11 + y10,12 + y10,13 + y10,22 +
 + y11,9 + y11,10 + y11,11 + y11,12 + y11,13 + y11,26 + y11,27 +
 + y12,10 + y12,11 + y12,12 + y12,13 + y12,14 + y12,30 + y12,31
 + y13,10 + y13,11 + y13,12 + y13,13 + y13,14 + y13,33 + y13,34
 + y14,6 + y14,12 + y14,13 + y14,14 + y14,15 + y14,33 + y14,34 +
 + y15,6 + y15,14 + y15,15 + y15,16 + y15,39 + y15,40 =  1
 + y16,6 + y16,7 + y16,15 + y16,16 + y16,17 + y16,18 + y16,19 +
 + y17,6 + y17,7 + y17,16 + y17,17 + y17,18 + y17,19 =  1
 + y18,6 + y18,7 + y18,16 + y18,17 + y18,18 + y18,19 =  1
 + y19,6 + y19,7 + y19,16 + y19,17 + y19,18 + y19,19 + y19,20 =
 + y20,6 + y20,7 + y20,8 + y20,19 + y20,20 + y20,21 =  1
 + y21,7 + y21,8 + y21,9 + y21,20 + y21,21 + y21,22 =  1
 + y22,8 + y22,9 + y22,10 + y22,21 + y22,22 + y22,23 + y22,24 +
 + y23,8 + y23,9 + y23,10 + y23,22 + y23,23 + y23,24 + y23,25 =
 + y24,8 + y24,9 + y24,10 + y24,22 + y24,23 + y24,24 + y24,25 =
 + y25,8 + y25,9 + y25,10 + y25,22 + y25,23 + y25,24 + y25,25 =
 + y26,9 + y26,10 + y26,11 + y26,26 + y26,27 =  1
 + y27,9 + y27,10 + y27,11 + y27,26 + y27,27 + y27,28 + y27,29 +
 + y28,10 + y28,11 + y28,27 + y28,28 + y28,29 + y28,30 =  1
 + y29,10 + y29,11 + y29,27 + y29,28 + y29,29 + y29,30 =  1
 + y30,10 + y30,11 + y30,12 + y30,27 + y30,28 + y30,29 + y30,30
 + y31,12 + y31,30 + y31,31 + y31,32 =  1
 + y32,12 + y32,30 + y32,31 + y32,32 =  1
 + y33,13 + y33,14 + y33,33 + y33,34 + y33,35 =  1
 + y34,13 + y34,14 + y34,33 + y34,34 + y34,35 =  1
 + y35,13 + y35,14 + y35,33 + y35,34 + y35,35 =  1
 + y36,13 + y36,14 + y36,36 + y36,37 + y36,38 =  1
 + y37,13 + y37,14 + y37,36 + y37,37 + y37,38 =  1
```

Figure 8: One part of the .lp file

```
Variable types: 0 continuous, 284 integer (284 binary)

Root relaxation: objective 4.400000e+01, 222 iterations, 0.00 seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

*    0     0               0      44.0000000   44.00000  0.00%     -    0s

Explored 0 nodes (222 simplex iterations) in 0.01 seconds
Thread count was 1 (of 64 available processors)

Solution count 2: 44 218

Optimal solution found (tolerance 1.00e-04)
Best objective 4.400000000000e+01, best bound 4.400000000000e+01, gap 0.0000%
Optimal objective: 44.0
********** Begin .sol file *************
# Objective value = 44
x6 0
x7 0
x8 0
x9 0
x10 1
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
y6,6 0
y6,7 0
y6,14 0
```

Figure 9: Gurobi test for .lp file

We conclude from the results that the ASs that provide us with the shortest path from any Tier-2 or Tier-3 to the nearest server farm (that doest cross through more than one AS) are: **10, 13, 16, 21, 30**. We also concluded that the total OPEX cost is **44**.

1.3.2 Question b

After running the script from figure 10, we concluded that the minimum servers required to have around 1% blocking for both formats, is 62 with a value of W = 960.

```matlab
Mhd = 4; %throughput of hd format
M4k = 10; %throughput of 4k format


p = 20; % 20% of requests are HD

subscribers = 10*3000 + 24*1500;
% requests per hour
lambda = 2 / (24 * 7);  % 2 request / 24h * 7days
lambda = lambda * subscribers;

limit_servers = 100;
limit_w = 1200;

Mhd = 4; %throughput of hd format
M4k = 10; %throughput of 4k format

for i=50:limit_servers
      for j=500:20:limit_w

        [b, o] = simulator2( lambda, i, j, p, Mhd, M4k, R);
        if (o <= 1 && b <= 1)
            fprintf('Servers: %d \n', i)
            fprintf('W: %d \n', + j)
            fprintf('HD Blocking Probability = %.2e \n', b)
            fprintf('4K Blocking probability = %.2e \n\n', o)
        end
      end




 end
```

```
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
>> script4
Servers: 62
W: 960
HD Blocking Probability = 0.00e+00
4K Blocking probability = 9.94e-01

Servers: 64
W: 520
HD Blocking Probability = 0.00e+00
4K Blocking probability = 6.16e-01

Servers: 64
W: 580
HD Blocking Probability = 0.00e+00
4K Blocking probability = 4.09e-01

Servers: 64
W: 600
HD Blocking Probability = 0.00e+00
4K Blocking probability = 7.10e-01
```

Figure 10: Script to run exercise 3b