Universidade de Aveiro

Desempenho e Dimensionamento de Redes

deti, 2016/2017

Relatório do Trabalho no.3

Autores:
Pedro Coelho 59517
Nuno Silva 72708

Professores:
Amaro Fernandes de Sousa

18 de Maio de 2017

Conteúdo

Resumo

Os objetivos deste trabalho passam por determinar a performance de bloqueio de serviços de streaming de vídeo com recurso a um conjunto de simuladores que permitem analisar os efeitos da mudança de variáveis do sistema.

1 Primeira Parte

1.1 Análise Analítica

Table 2.1		
Case	BlockingProbability	Average ConnectionLoad (Mbps)
A	0,0142	2,9575
B	0,0171	3,1124
C	0,0521	4,2657
D	0,0606	4,4619
E	0,0087	74,3445
F	0,0167	77,8468
G	0,0541	85,1306
H	0,0782	87,5733
I	0,0012	898,889
J	0,0106	939,8955
K	0,0685	978,0748
L	0,1108	985,5607

1.2 Análise da Simulação

a.)

Table 1 2.2 a				
Case	BlockingProbability	BP Error	ConnectionLoad (Mbps)	AVL Error
A	0,013580	0,000826	2,936981	0,021173
B	0,016510	0,000325	3,118797	0,016500
C	0,056670	0,001549	4,242532	0,023359
D	0,061600	0,001970	4,469113	0,031906
E	0,008320	0,001437	74,014861	0,407929
F	0,015330	0,001626	77,145434	0,525025
G	0,051591	0,002394	84,602454	0,474908
H	0,076680	0,003473	87,401552	0,412952
I	0,000980	0,000650	856,669050	6,015189
J	0,005910	0,000721	895,768806	3,829742
K	0,065610	0,004515	938,722647	1,782291
L	0,101820	0,005131	944,805305	2,723552

b.) Foram introduzidas algumas alterações no simulador para que este não considere os primeiros N valores da simulação, a alteração reflete-se conforme exposto no segmento de código transcrito abaixo.

```
while NARRIVALS < N
    event= EventList(1,1);
    Previous_Clock= Clock;
    Clock= EventList(1,2);
    EventList(1,:)= [];
    LOAD= LOAD + STATE*(Clock-Previous_Clock);

    if event == ARRIVAL
        EventList= [EventList; ARRIVAL Clock+exprnd(invlambda)];
        NARRIVALS= NARRIVALS+1;
        if NARRIVALS >= N
            if STATE + M <= C
                STATE= STATE+M;
                EventList= [EventList; DEPARTURE Clock+exprnd(invmiu)];
            else
                %BLOCKED= BLOCKED+1;
            end
        end
    else
        STATE= STATE-M;
    end
    EventList= sortrows(EventList,2);
end

while NARRIVALS < R
    event= EventList(1,1);
    Previous_Clock= Clock;
    Clock= EventList(1,2);
    EventList(1,:)= [];
    LOAD= LOAD + STATE*(Clock-Previous_Clock);

    if event == ARRIVAL
        EventList= [EventList; ARRIVAL Clock+exprnd(invlambda)];
        NARRIVALS= NARRIVALS+1;
        if NARRIVALS >= N
            if STATE + M <= C
                STATE= STATE+M;
                EventList= [EventList; DEPARTURE Clock+exprnd(invmiu)];
            else
                BLOCKED= BLOCKED+1;
            end
        end
    else
        STATE= STATE-M;
    end
    EventList= sortrows(EventList,2);
end

b= BLOCKED/NARRIVALS;
o= LOAD/Clock;
```

end

Table 1 2.2b		
Case	BlockingProbability	Average ConnectionLoad (Mbps)
A	0,013	2,6538
B	0,015	2,8072
C	0,0478	3,8524
D	0,0551	4,0328
E	0,0058	65,68
F	0,014	69,9265
G	0,0457	76,2776
H	0,0689	78,4403
I	0,00021	766,4044
J	0,0086	802,8884
K	0,0509	840,2073
L	0,0912	848,4643

A alteração introduzida no simulador, que permite ignorar os primeiros N valores da simulação, permitiu que os valores obtidos se aproximassem mais dos valores determinados na análise analítica.

c.)
p(b) = 8.24e-03 +- 8.28e-04
p(o) = 8.03e+02 +- 1.53e+00

d.)
p(b) = 1.13e-02 +- 9.62e-04
p(o) = 9.28e+02 +- 1.30e+00

Os valores obtidos e a margem de erro são melhores quando existe recurso a um valor inferior de N.

2 Segunda Parte

O simulador anterior foi adaptado para que se possa considerar o pedido de filmes em alta definição.

```
function [ bs, bh ] = simu2( lambda, p, invmiu, S, W, Ms, Mh, R, N)
lambdas = lambda*(1-p);
lambdah = lambda*p;

invlambda_s=60/lambdas;
invlambda_h=60/lambdah;

ARRIVAL_S = 0;
ARRIVAL_H = 1;
DEPARTURE_S = 2;
DEPARTURE_H = 3;
STATE = zeros(1,S);
```

```
STATE_S = 0;

NARRIVALS = 0;
NARRIVALS_S = 0;
NARRIVALS_H = 0;
BLOCKED_S = 0;
BLOCKED_H = 0;

C=S * 100;
Clock = 0;
EventList = [ARRIVAL_S exprnd( invlambda_s ) 0;
             ARRIVAL_H exprnd(invlambda_h) 0];

EventList = sortrows( EventList, 2 );

while NARRIVALS < N
    event= EventList(1,1);
    %Previous_Clock= Clock;
    Clock= EventList(1,2);
    server = EventList (1,3);
    EventList(1,:)= [];
    %LOAD= LOAD + STATE*(Clock-Previous_Clock);

    if event == ARRIVAL_S
                EventList = [ EventList;
                              ARRIVAL_S Clock + exprnd(invlambda_s) 0];
                NARRIVALS = NARRIVALS + 1;
        %NARRIVALS_S = NARRIVALS_S + 1;
        aux=find(STATE==min(STATE));
        aux=aux(1);
                if STATE_S + Ms <= C-W && STATE(aux)+ Ms <= 100
                        STATE_S = STATE_S + Ms;
            STATE(aux) = STATE(aux) + Ms;
                        EventList = [EventList;
                                     DEPARTURE_S Clock + exprnd(invmiu) aux];
                else
                        %BLOCKED_S = BLOCKED_S + 1;
                end
    elseif event == ARRIVAL_H
        EventList = [ EventList;
                    ARRIVAL_H Clock + exprnd(invlambda_h) 0];
                NARRIVALS = NARRIVALS + 1;
        %NARRIVALS_H = NARRIVALS_H + 1;
        aux=find(STATE==min(STATE));
        aux=aux(1);
                if STATE(aux)+ Mh <= 100;
            STATE(aux) = STATE(aux) + Mh;
                        EventList = [EventList;
                                     DEPARTURE_H Clock + exprnd(invmiu) aux];
                else
```

```
                    %BLOCKED_H = BLOCKED_H + 1;
                end
    elseif event == DEPARTURE_S
        STATE_S= STATE_S-Ms;
        STATE(server) = STATE(server) - Ms;
    else
        STATE(server) = STATE(server) - Mh;
    end
    EventList= sortrows(EventList,2);
end

while NARRIVALS < R
    event= EventList(1,1);
    %Previous_Clock= Clock;
    Clock= EventList(1,2);
    server = EventList (1,3);
    EventList(1,:)= [];
    %LOAD= LOAD + STATE*(Clock-Previous_Clock);

    if event == ARRIVAL_S
                EventList = [ EventList;
                              ARRIVAL_S Clock + exprnd(invlambda_s) 0];
                NARRIVALS = NARRIVALS + 1;
        NARRIVALS_S = NARRIVALS_S + 1;
        aux=find(STATE==min(STATE));
        aux=aux(1);
                if STATE_S + Ms <= C-W && STATE(aux)+ Ms <= 100
                        STATE_S = STATE_S + Ms;
            STATE(aux) = STATE(aux) + Ms;
                        EventList = [EventList;
                                     DEPARTURE_S Clock + exprnd(invmiu) aux];
                else
                        BLOCKED_S = BLOCKED_S + 1;
                end
    elseif event == ARRIVAL_H
        EventList = [ EventList;
                    ARRIVAL_H Clock + exprnd(invlambda_h) 0];
                NARRIVALS = NARRIVALS + 1;
        NARRIVALS_H = NARRIVALS_H + 1;
        aux=find(STATE==min(STATE));
        aux=aux(1);
                if STATE(aux)+ Mh <= 100;
            STATE(aux) = STATE(aux) + Mh;
                        EventList = [EventList;
                        DEPARTURE_H Clock + exprnd(invmiu) aux];
                else
                        BLOCKED_H = BLOCKED_H + 1;
                end
    elseif event == DEPARTURE_S
        STATE_S= STATE_S-Ms;
```

```
        STATE(server) = STATE(server) - Ms;
    else
        STATE(server) = STATE(server) - Mh;
    end
    EventList= sortrows(EventList,2);
end

bs = BLOCKED_S/NARRIVALS_S;
bh = BLOCKED_H/NARRIVALS_H;

end
```

a.)

Table 2				
λ	S	W	Blocking Proability (%)	HD Blocking Probability (%)
13	1	0	0,00395	0,0116
13	1	60	0,015	0,012
13	1	80	0,291	0,00333
50	3	0	0,00129	0,0158
50	3	180	0,00702	0,0157
50	3	240	0,369	0,00109

É fácil observar que o aumento do número de servidores disponíveis se traduz numa diminuição da probabilidade de bloqueio.

b.)

```
invmiu = 90;
p = 0.1;
S = 2;
lambda =8000/(7*24);
W = 5:5:80;
Ms = 2;
Mh = 5;
R = 10000;
N = 1000;

iterations = 40;
bSD = zeros(length(W), iterations);
bHD = zeros(length(W), iterations);
for j=1:length(W)
    for i=1:iterations
        [bsh, bhd] = simu2(lambda, p, invmiu, S, W(j), Ms, Mh, R, N);
        bSH(j, i) = bsh;
        bHD(j, i) = bhd;
    end
end
```

W	bSD	bSDError	bHD	bHDError
5	0.007781	0.000714	0.036533	0.003233
10	0.006826	0.000599	0.037121	0.002374
15	0.008513	0.000630	0.040261	0.002855
20	0.008235	0.000646	0.041515	0.002660
25	0.008314	0.000713	0.038396	0.003224
30	0.008814	0.000696	0.040216	0.002592
35	0.009409	0.000900	0.038640	0.002218
40	0.012066	0.001134	0.033219	0.001941
45	0.016148	0.001111	0.029907	0.001935
50	0.022685	0.001611	0.028178	0.002460
55	0.035663	0.001802	0.018287	0.002103
60	0.047803	0.001784	0.012417	0.001803
65	0.069463	0.002087	0.009464	0.001352
70	0.084852	0.002757	0.005187	0.001100
75	0.116492	0.002433	0.003078	0.000785
80	0.139083	0.002587	0.001967	0.000639

Os valores de W para os quais as probabilidades de bloqueio dos dois modos se equilibram situam-se entre os 50 e 60 Mb.

c.)

```
invmiu = 90;
p = 0.2;
S = 1:6;
S = repmat(S, 1, 3)';
W = [repmat([40], 6, 1); repmat([50], 6, 1); repmat([60], 6, 1)];
lambda =17000 / (7 * 24);
Ms = 2;
Mh = 5;
R = 10000;
N = 1000;

iterations = 40;
bsD = zeros(length(W), iterations);
bHD = zeros(length(W), iterations);

for j = 1:length(W)
    for i = 1:iterations
        [sd, hd] = simu2(lambda, p, invmiu, S, W(j), Ms, Mh, R, N);
        bsD(j, i) = sd;
        bHD(j, i) = hd;
    end

    fprintf('%d;%d;', W(j), S(j));
    fprintf('%.6f;', mean(bsD(j, :)));
    fprintf('%.6f\n', mean(bHD(j, :)));
end
```

Tabela 1: My caption

W	S	bSD	bSDError	bHD	bHDError
40	1	0,754892	0,00168	0,746161	0,003788
	2	0,370633	0,002335	0,720057	0,003061
	3	0,113016	0,001871	0,510047	0,005485
	4	0,015806	0,000773	0,15189	0,004366
	5	0,000056	0,000039	0,002809	0,000987
	6	0			
	7				
	8				
	9				
	10				
50	1	0,797163	0,001458	0,684077	0,003853
	2	0,402206	0,00322	0,673515	0,003806
	3	0,118339	0,002118	0,503414	0,005362
	4	0,015386	0,000921	0,15421	0,005892
	5	0,000076	0,000036	0,002328	0,000808
	6	0			
	7				
	8				
	9				
	10				
60	1	0,836788	0,001397	0,620391	0,003958
	2	0,441544	0,002768	0,623758	0,002744
	3	0,123252	0,001383	0,501121	0,004437
	4	0,016062	0,000974	0,157417	0,005331
	5	0,000049	0,00003	0,001521	0,000525
	6	0			
	7				
	8				
	9				
	10				

Para o caso sugerido em c.) o melhor valor de W é 50.

3 Terceira Parte

a.) Para calcular a conectividade entre AS com base na matriz G foi necessário desenvolver um script e outro para escrever o ficheiro LP para ser usado no NEOS Server.

```
for j = 6:40
    I(j, j) = 0;

    for i = 1:40
        if i ~= j
            for k = 1:length(G)
                if (G(k, 1) == j && G(k, 2) == i)
                    || (G(k, 2) == j && G(k, 1) == i)
                    I(j, i) = 1;
                    for l = 1:length(G)
```

```
                        if G(l, 1) == i
                            n = G(l,2);
                            if I(j, n) == -1
                                I(j, n) = 2;
                            end;
                        elseif G(l, 2) == i
                            n = G(l,1);
                            if I(j, n) == -1
                                I(j, n) = 2;
                            end;
                        end;
                    end;
                end;
            end;
        end;
    end;
end;

%OPEX costs
costT2 = 8;
costT3 = 6;

%Number of AS
n = 40;

file = fopen('problem.lp','w');

%Minimize cost
fprintf(file, 'Minimize\n');
for j = 6:15
    fprintf(file, ' + %d x%d', costT2, j);
end;
for j = 16:40
    fprintf(file, ' + %d x%d', costT3, j);
end;

%Constrains
fprintf(file, '\nSubject To\n');

%sum Yji = 4
for j = 6:40
    for i = 6:40
        if I(j, i) > -1
            fprintf(file, ' + y%d,%d' ,j ,i);
        end
    end
    fprintf(file, ' = 1\n');
end

%Yji <= Xi
```

```
for j = 6:40
    for i = 6:40
        if I(j, i) > -1
            fprintf(file, ' + y%d,%d - x%d <= 0\n', j, i, i);
        end;
    end
end

%Binary variables
fprintf(file, 'Binary\n');
for j = 6:40
    fprintf(file, ' x%d\n', j);
    for i = 6:40
        if I(j, i) > -1
            fprintf(file, ' y%d,%d\n', j, i);
        end;
    end;
end;

%End
fprintf(file, 'End\n');
fclose(file);

%Configuration
file = fopen('display.txt','w');
fprintf(file, 'display solution variables -');
fclose(file);
```

Isto permitiu-nos obter o seguinte ficheiro, do qual omitimos algumas partes que achamos não ter interesse:

```
*************************************************************

Variable Name            Solution Value
x6                             1.000000
x7                             1.000000
x8                             1.000000
x9                             1.000000
x10                            1.000000
x11                            1.000000
x12                            1.000000
x13                            1.000000
x14                            1.000000
x15                            1.000000
y6,1                           1.000000
y7,2                           1.000000
y8,2                           1.000000
y9,3                           1.000000
y10,3                          1.000000
y11,4                          1.000000
```

y12,4	1.000000
y13,4	1.000000
y14,1	1.000000
y15,1	1.000000
y16,6	1.000000
y17,6	1.000000
y18,6	1.000000
y19,6	1.000000
y20,7	1.000000
y21,8	1.000000
y22,8	1.000000
y23,9	1.000000
y24,9	1.000000
y25,9	1.000000
y26,10	1.000000
y27,10	1.000000
y28,11	1.000000
y29,11	1.000000
y30,11	1.000000
y31,12	1.000000
y32,12	1.000000
y33,13	1.000000
y34,13	1.000000
y35,13	1.000000
y36,14	1.000000
y37,14	1.000000
y38,14	1.000000
y39,15	1.000000
y40,15	1.000000
x1	1.000000
x2	1.000000
x3	1.000000
x4	1.000000
y16,2	1.000000
y16,3	1.000000
y16,4	1.000000
y17,2	1.000000
y17,3	1.000000
y17,4	1.000000
y18,2	1.000000
y18,3	1.000000
y18,4	1.000000
y19,3	1.000000
y19,4	1.000000
y20,1	1.000000
y20,3	1.000000
y20,4	1.000000
y21,1	1.000000
y21,4	1.000000
y22,1	1.000000

y22 ,4	1.000000
y23 ,1	1.000000
y23 ,2	1.000000
y23 ,4	1.000000
y24 ,1	1.000000
y24 ,2	1.000000
y24 ,4	1.000000
y25 ,1	1.000000
y25 ,2	1.000000
y25 ,4	1.000000
y26 ,1	1.000000
y26 ,2	1.000000
y27 ,1	1.000000
y27 ,2	1.000000
y28 ,1	1.000000
y28 ,2	1.000000
y28 ,3	1.000000
y29 ,1	1.000000
y29 ,2	1.000000
y29 ,3	1.000000
y30 ,1	1.000000
y30 ,2	1.000000
y30 ,3	1.000000
y31 ,1	1.000000
y31 ,2	1.000000
y31 ,3	1.000000
y32 ,1	1.000000
y32 ,2	1.000000
y32 ,3	1.000000
y33 ,1	1.000000
y33 ,2	1.000000
y33 ,3	1.000000
y34 ,1	1.000000
y34 ,2	1.000000
y34 ,3	1.000000
y35 ,1	1.000000
y35 ,2	1.000000
y35 ,3	1.000000
y36 ,2	1.000000
y36 ,3	1.000000
y36 ,4	1.000000
y37 ,2	1.000000
y37 ,3	1.000000
y37 ,4	1.000000
y38 ,2	1.000000
y38 ,3	1.000000
y38 ,4	1.000000
y39 ,2	1.000000
y39 ,3	1.000000
y39 ,4	1.000000

```
y40,2                                   1.000000
y40,3                                   1.000000
y40,4                                   1.000000
All other variables in the range 1-1440 are 0
```

b.)

```
invmiu = 90;
R = 10000;
N = 1000;
Ms = 2;
Mh = 5;

p = 0.2;
lambda = 2 * subscribers / (7 * 24);
S = 20:30;
W = 90;

iterations = 40;
bSD = zeros(length(W), iterations);
bHS = zeros(length(W), iterations);

for j = 1:length(S)
    for i = 1:iterations
        [bsh, bhd] = simu2(lambda, p, invmiu, S(j), W, Ms, Mh, R, N);
        bSD(j, i) = bsd;
        bHS(j, i) = bhd;
    end

    fprintf('%d;', S(j));
    fprintf('%.6f;', mean(bSD(j, :)));
    fprintf('%.6f\n', mean(bHS(j, :)));
end
```

S	bSD	bHD
20	0,001232	0,382501
21	0,000740	0,289839
22	0,000038	0,190175
23	0,000030	0,110990
24	0,000011	0,046365
25	0,000000	0,013556
26	0,000000	0,001669

A tabela construida com base no script para esta alinea permite apurar que para obter valores inferiores a 1% são necessários pelo menos 25 servidores.

c.)

```
AS = [9, 14, 19, 30];
cost = 28;
```

```
S = zeros(40, 1);

for i = 1:length(S)
    closer = -1;
    for j = 1:length(AS)
        if I(AS(j), i) ~= -1
            if closer == -1 || I(AS(j), i) < I(closer, i)
                closer = AS(j);
            end;
        end;
    end;

    S(i) = closer;
end;

subs = zeros(length(AS), 1);
server = zeros(length(AS), 1);

totalServers = 25;

subsT2 = 2500;
subsT3 = 1000;

for i = 1:length(AS)
    S(S == AS(i)) = i;
end;

%Calculate how many subs per AS
for i = 6:15
    subs(S(i)) = subs(S(i)) + subsT2;
end;
for i = 16:40
    subs(S(i)) = subs(S(i)) + subsT3;
end;

totalSubscrivers = sum(subs);

%How many server to place in each AS
for i = 1:length(AS)
    server(i) = round(totalServers * subs(i)/totalSubscrivers);
end;
```

Este script permitiu-nos obter as seguintes relações entre AS, servidores e número de subscritores: AS 9 com 7 servidores e 14500 subs, AS 14 com 8 servidores e 16500 subs, AS 30 e 19 com 5 servidores e 10000 e 9000 subs respetivamente.