DEPARTAMENTO DE ELETRÓNICA, TELECOMUNICAÇÕES E INFORMÁTICA

MESTRADO INTEGRADO EM ENG. DE COMPUTADORES E TELEMÁTICA

ANO 2016/2017

DESEMPENHO E DIMENSIONAMENTO DE REDES

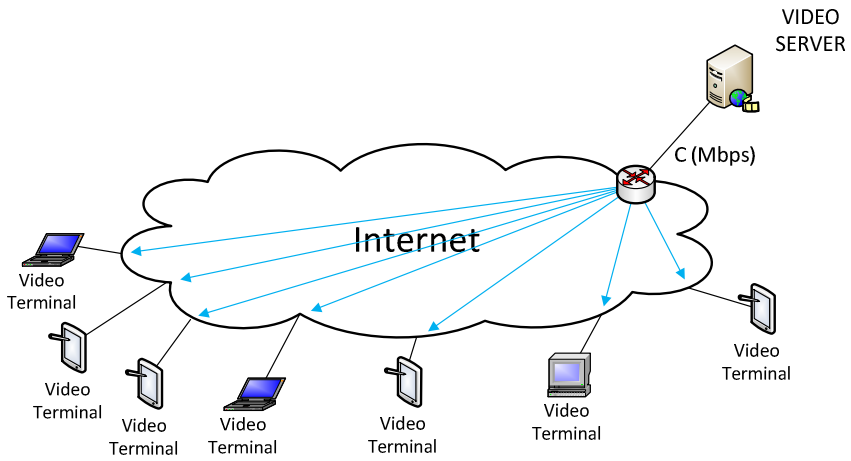
ASSIGNMENT GUIDE NO. 3

BLOCKING PERFORMANCE OF VIDEO-STREAMING SERVICES

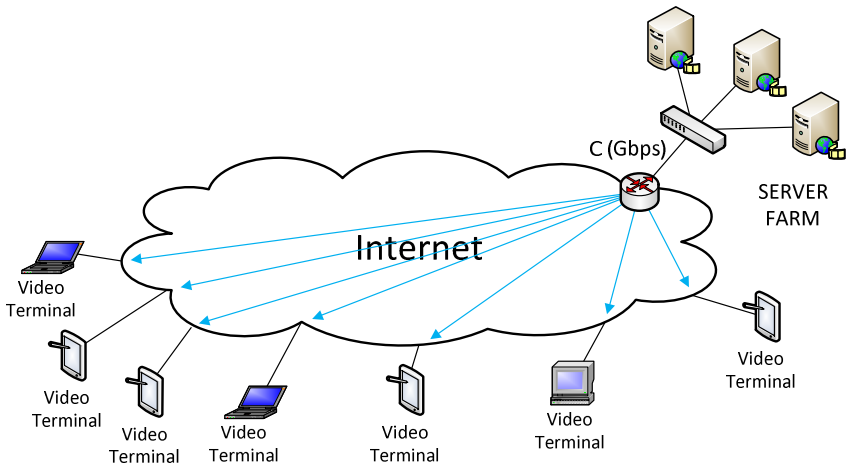
Assignment Guide No. 3

1. Preamble

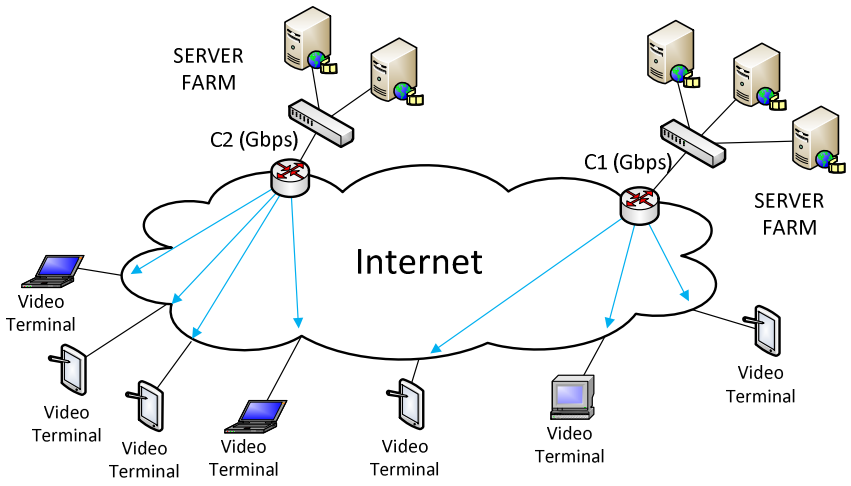
The aim of this assignment is to assess the blocking performance of video-streaming services. In its simplest form, such services are provided by a single video-streaming server, as illustrated in the following figure.



At each point in time, the server has a catalogue of movies, each one with a given duration, to be selected by the customers. Movies are available in one or more video formats, depending on the type of customers and/or revenue strategy of the company. For larger businesses, the service is provided by a farm of servers, located on a single site, as illustrated in the following figure.



The aim of a server farm is (i) to scale the service to a higher number of clients and (ii) to make the service robust to single server failures. Finally, the most general case is to have multiple server farms on different sites, as illustrated in the figure on the next page. The aim of multiple server farms is (i) to scale the services to even higher number of clients, (ii) to lower the average routing distance (and, consequently, round-trip-time) between clients and servers and (iii) to make the service robust to single site failures. In this case, the number of server farms and, for each farm, its location, its number of servers and its Internet connection capacity is a layout problem which typically involves some sort of optimization.



2. First Part of the Assignment

Consider a video-streaming service provided by a single server whose Internet connection has a capacity of C (in Mbps). Consider that the server provides movies on a single video format and each movie has a throughput of M (in Mbps). When a movie is requested by a customer, it starts being transmitted by the server if the resulting total throughput is within the server Internet connection capacity; otherwise, the request is blocked. Consider that movie requests are a Poisson process with a total rate λ (in requests/hour) Consider also that the movies duration is an exponential distributed random variable with average duration $1/\mu$.(in minutes). This case can be modelled by a $M/M/m/m$ queuing system.

2.1. Analytical Analysis

Compute the analytical values of the blocking probability and the average server connection load (see Appendix A) for the cases defined in *Table 1*. Analyse these results and take conclusions on the impact of the different input parameters on the two performance parameters.

Table 1

Case	λ (requests/hour)	$1/\mu$ (minutes)	C (Mbps)	M (Mbps)	Blocking Probability	Average Connection Load (Mbps)
A	1.0	90	10	2		
B	1.0	95	10	2		
C	1.5	90	10	2		
D	1.5	95	10	2		
E	25	90	100	2		
F	25	95	100	2		
G	30	90	100	2		
H	30	95	100	2		
I	300	90	1000	2		
J	300	95	1000	2		
K	350	90	1000	2		
L	350	95	1000	2		

2.2. Simulation Analysis

Appendix B provides a MATLAB function named simulator1, implementing an event driven simulator for the video-streaming service based on a single server and providing movies with a single video format. The input parameters of this simulator are:

- λ – movies request rate (in requests per hour)
- $1/\mu$ – average duration of movies (in minutes)
- C – Internet server connection capacity (in Mbps)
- M – throughput of the movies (in Mbps)
- R – number of movie requests for the simulation to stop

The performance parameters estimated by this simulator are:

- b – blocking probability (percentage of movie requests that are blocked)
- o – average occupation of the Internet connection (in Mbps)

The stopping criteria is the time instant of the arrival of the movie request number R. This simulator considers:

- events: ARRIVAL (the time instant of a movie request) and DEPARTURE (the time instant of a movie termination);
- state variable: STATE (total throughput of the movies in transmission);
- statistical counters: LOAD (the integral of connection load up to the current time instant), NARRIVALS (number of movie requests) and BLOCKED (number of blocked requests).

Start by analysing carefully the provided MATLAB function. Then, develop a MATLAB script to run the simulator a given number of times and to compute the estimated values and the 90% confidence intervals of both performance parameters (confidence intervals in the form $a \pm b$).

a) For all cases defined in Table 1, run 10 times Simulator 1, each time with a stopping criterion of $R = 10000$. For each case, determine the estimated values and the 90% confidence intervals of both performance parameters (see slide 21 of theoretical presentation "Introduction to Discrete Event Simulation"). Compare the confidence intervals with the analytical values and check that the simulator results are not in accordance with them.

b) The reason for the previous results is the initial transient states of the simulation (the simulation takes a significant amount of simulated time to reach its steady state). Change the MATLAB function to consider an additional input parameter N and to start updating the statistical counters only at the time instant of the N^{th} movie request (do not forget to change accordingly the performance parameters calculation at the end of the simulation). Run again 10 times this version of the simulator for all cases in Table 1 with $R = 10000$ and $N = 1000$. What do you conclude?

c) Only for case J of Table I, run 100 times Simulator 1, each time with $R = 10000$ and $N = 1000$. Determine the estimated values and the 90% confidence intervals of both performance parameters. Compare and justify these results with the previous ones.

d) Again only for case J of Table I, run 10 times Simulator 1, each time with with $R = 100000$ and $N = 1000$. Determine the estimated values and the 90% confidence intervals of both performance parameters. Compare and justify these results with all the previous ones for this case J.

To be completed…

Appendix A – *M*/*M*/*m*/*m* queuing system

Blocking probability:

Consider an *M*/*M*/*m*/*m* queuing system with a capacity of *N* units and an offered load of ρ Erlangs where the load is composed by a flow of requests where each request is for 1 unit of the server. The ErlangB formula $E(\rho, N)$:

$$E(\rho, N) = \frac{\frac{\rho^N}{N!}}{\sum_{n=0}^{N} \frac{\rho^n}{n!}}$$

gives the probability of a new request being blocked because the server is fully occupied. In MATLAB, the straightforward implementation is:

```
numerator= ro^N/factorial(N);
denominator= 0;
for n= 0:N
    denominator= denominator + ro^n/factorial(n);
end
p= numerator/denominator
```

This implementation has two problems. First, for larger values of *N* and ρ, it causes overflow problems. Second, it is inefficient because it requires a large number of elementary mathematical operations. An efficient way to compute ErlangB formula is as follows. If we divide both terms of the division by its numerator, we reformulate ErlangB formula in the following way:

$$E(\rho, N) = \frac{\frac{\rho^N}{N!}}{\sum_{n=0}^{N} \frac{\rho^n}{n!}} = \frac{1}{\sum_{n=0}^{N} \left(\frac{N!}{\rho^N} \times \frac{\rho^n}{n!} \right)}$$

$$E(\rho, N) = \frac{1}{\frac{N \times (N-1) \times \ldots \times 2 \times 1}{\rho^N} + \frac{N \times (N-1) \times \ldots \times 2}{\rho^{N-1}} + \cdots + \frac{N \times (N-1)}{\rho^2} + \frac{N}{\rho} + 1}$$

Now, if we define the sequence $a(n)$, with $n = N+1, N, N-1, \ldots, 2, 1$, in the following way:

$a(N+1) = 1$

$a(n) = a(n+1) \times n / \rho$, for $n = N, N-1, \ldots, 2, 1$

sum all $a(n)$ values and inverse the result, we obtain the ErlangB $E(\rho, N)$ value. In MATLAB, this algorithm can be implemented with the following code:

```
a= 1; p= 1;
for n= N:-1:1
    a= a*n/ro;
    p= p+a;
end
p= 1/p
```

Average Server Load:

On the other hand, the average server load is given by:

$$L(\rho, N) = \frac{\sum_{i=1}^{N} \frac{\rho^i}{(i-1)!}}{\sum_{n=0}^{N} \frac{\rho^n}{n!}}$$

In MATLAB, the straightforward implementation of the average server occupation is::

```
numerator= 0;
for i=1:C
    numerator= numerator+ro^i/factorial(i-1);
end
denominator= 0;
for n=0:C
    denominator= denominator+ro^n/factorial(n);
end
o= numerator/denominator
```

This implementation has the same problems as previously described for the ErlangB formula. To implement the average server load in an efficient way, we reformulate the expression as:

$$L(\rho, N) = \frac{\sum_{i=1}^{N} \frac{\rho^i}{(i-1)!}}{\sum_{n=0}^{N} \frac{\rho^n}{n!}} = \frac{\frac{\rho^N}{N!} \times \sum_{i=1}^{N} \left(\frac{N!}{\rho^N} \times \frac{\rho^i}{(i-1)!} \right)}{\sum_{n=0}^{N} \frac{\rho^n}{n!}} = E(\rho, N) \times \sum_{i=1}^{N} \left(\frac{N!}{\rho^N} \times \frac{\rho^i}{(i-1)!} \right)$$

$$L(\rho, N) = \frac{\frac{N \times (N-1) \times \ldots \times 2 \times 1}{\rho^{N-1}} + \frac{N \times (N-1) \times \ldots \times 2}{\rho^{N-2}} + \cdots + \frac{N \times (N-1)}{\rho} + N}{\frac{N \times (N-1) \times \ldots \times 2 \times 1}{\rho^{N}} + \frac{N \times (N-1) \times \ldots \times 2}{\rho^{N-1}} + \cdots + \frac{N \times (N-1)}{\rho^2} + \frac{N}{\rho} + 1}$$

If we define the sequence $a(n)$, with $n = N, N, N-1, \ldots, 2, 1$, in the following way:

$a(N) = N$

$a(n) = a(n+1) \times n / \rho$, for $n = N-1, \ldots, 2, 1$

and we sum all $a(n)$ values, we obtain the numerator of $L(\rho, N)$. The denominator is obtained in the same way as for the ErlangB formula $E(\rho, N)$. In MATLAB, this method can be implemented with the following code:

```
a= C;
numerator= a;
for i= C-1:-1:1
    a= a*i/ro;
    numerator= numerator+a;
end
a= 1;
denominator= a;
for i= C:-1:1
    a= a*i/ro;
    denominator= denominator+a;
end
o= numerator/denominator
```

Appendix B –Proposed MATLAB function for Simulator 1

```
function [b o]= simulator1(lambda,invmiu,C,M,R)
    %lambda = request arrival rate (in requests per hour)
    %invmiu= average movie duration (in minutes)
    %C= Internet connection capacity (in Mbps)
    %M= throughput of each movie (in Mbps)
    %R= stop simulation on ARRIVAL no. R

    invlambda=60/lambda; %average time between requests (in minutes)

    %Events definition:
    ARRIVAL= 0;         %movie request
    DEPARTURE= 1;       %termination of a movie transmission
    %State variables initialization:
    STATE= 0;
    %Statistical counters initialization:
    LOAD= 0;
    NARRIVALS= 0;
    BLOCKED= 0;
    %Simulation Clock and initial List of Events:
    Clock= 0;
    EventList= [ARRIVAL exprnd(invlambda)];

    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 STATE + M <= C
                STATE= STATE+M;
                EventList= [EventList; DEPARTURE Clock+exprnd(invmiu)];
            else
                BLOCKED= BLOCKED+1;
            end
        else
            STATE= STATE-M;
        end

        EventList= sortrows(EventList,2);
    end

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