DEPARTAMENTO DE ELETRÓNICA, TELECOMUNICAÇÕES
E INFORMÁTICA

MESTRADO INTEGRADO EM ENG. DE COMPUTADORES E TELEMÁTICA

ANO 2016/2017

DESEMPENHO E DIMENSIONAMENTO DE REDES

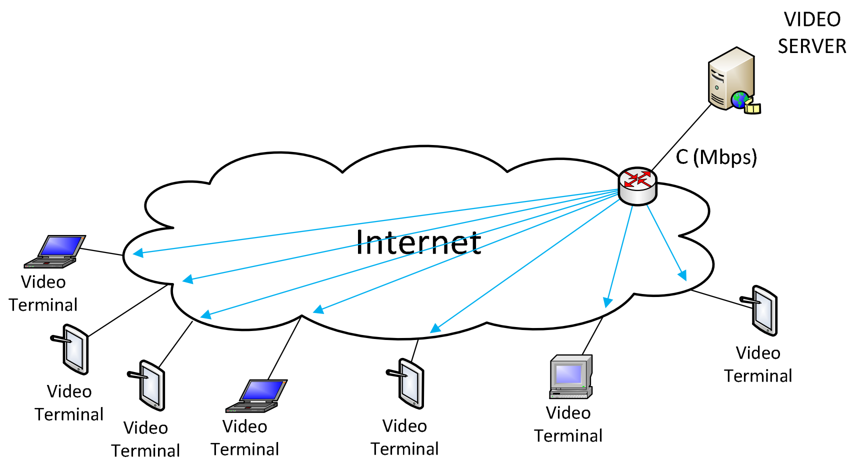
ASSIGNMENT GUIDE NO. 3

BLOCKING PERFORMANCE OF VIDEO-STREAMING SERVICES

Assignment Guide No. 3

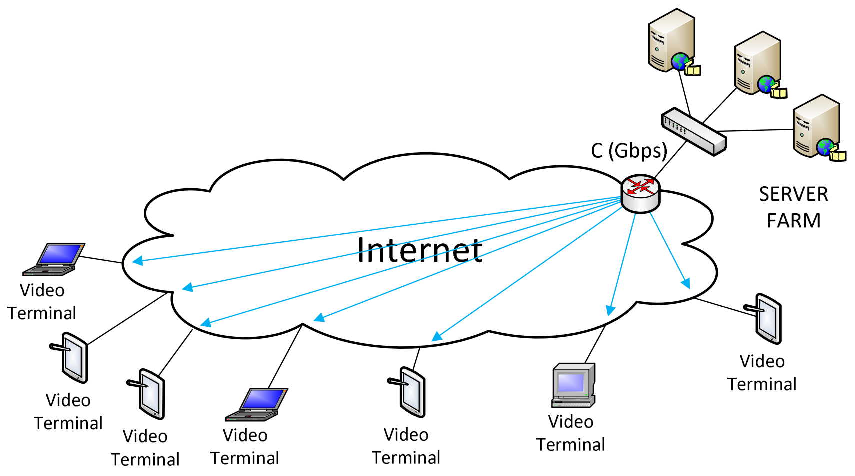
1. Preamble

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



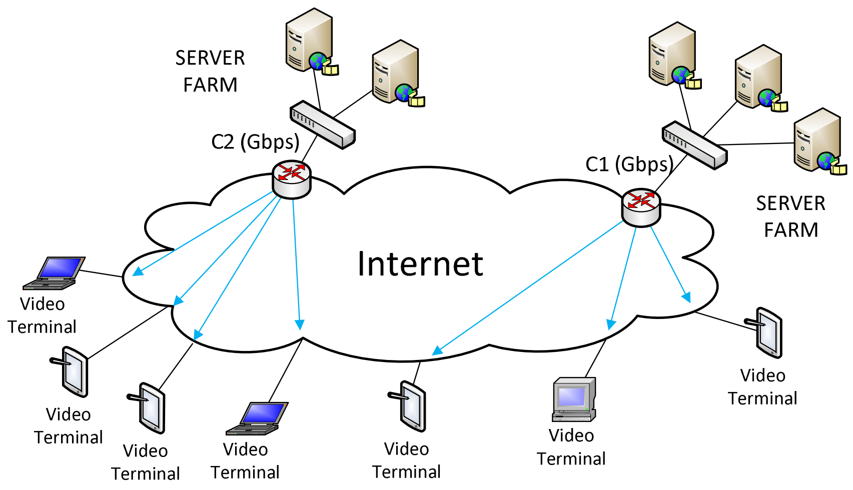
At each point in time, the server has a catalogue of movies, each one with a given duration, to be selected by the service subscribers. Movies are available in one or more video formats, depending on the targeted types of subscribers and/or the revenue strategy of the company.

In alternative, 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 larger number of subscribers and (ii) to make the service robust to single server failures.

Finally, in its most general architecture, the service is provided by multiple server farms on different sites, as illustrated in the figure on the next page. The aim of having multiple server farms is (i) to scale the services to even larger number of subscribers, (ii) to lower the average routing distance (and, consequently, round-trip-time) between subscriber locations and server locations and (iii) to make the service robust not only to single server failures but also 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

In this 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 subscriber, it starts being transmitted by the server if the resulting total throughput is within the Internet connection capacity; otherwise, the request is blocked. Consider that movie requests are a Poisson process with an average rate of 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 an $M/M/m/m$ queuing system.

2.1. Analytical Analysis

Determine the analytical values of the blocking probability and the average 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 simulator1 are:

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

The performance parameters estimated by simulator1 are:

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

The stopping criterion is the time instant of the arrival of the movie request number R. Simulator1 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 simulator1 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 $a \pm b$ (see slide 21 of theoretical presentation "Introduction to Discrete Event Simulation").

a) For all cases defined in *Table 1*, run 10 times simulator1, 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. Compare the confidence intervals with the analytical values and check that the simulated results are not in accordance with the analytical values.

b) The reason for the previous results is the initial transient state 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 determination at the end of the simulation). Run again 10 times this version of the simulator for all cases defined in *Table 1* with R = 10000 and N = 1000. What do you conclude?

c) Only for case J of *Table 1*, run 100 times your version of simulator1, 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 for case J of *Table 1*, run 10 times your version of simulator1, each time 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 case J.

3. Second Part of the Assignment

In this second part of the assignment, consider a video-streaming service provided by one server farm with *S* video-streaming servers where each server has an interface of 100 Mbps (assume that the Internet connection of the server farm is always higher than $S \times 100$ Mbps). Consider also that the server farm provides movies on 2 possible video formats: a standard format whose throughput is 2 Mbps and a high-definition format whose throughput is 5 Mbps. All movies are available in both formats in all servers.

Consider a front-office system that assigns requests to servers using a load balancing strategy (i.e. each request is assigned to the least loaded server) and implements an admission control with a resource reservation of *W* (in Mbps) for high-definition movies (i.e., standard movies cannot occupy more than $C - W$ Mbps). In more detail, the admission control is as follows:

- When a high-definition movie is requested, it starts being transmitted by the least loaded server if it has at least 5 Mbps of unused capacity; otherwise, the request is blocked.
- When a standard movie is requested, it starts being transmitted by the least loaded server if the total throughput of standard movies does not become higher than $C - W$ Mbps and the least loaded server has at least 2 Mbps of unused capacity; otherwise, the request is blocked.

Consider that movie requests are a Poisson process with an average rate of λ (in requests/hour) and that *p*% of the requests are for high-definition movies. Consider also that the movies duration is an exponential distributed random variable with average duration $1/\mu$ (in minutes).

Develop a MATLAB function named simulator2, implementing an event driven simulator for this case to estimate the blocking probability of each type of movie requests (see Appendix C). Then, develop a MATLAB script to run simulator2 a given number of times and to compute the estimated values and the 90% confidence intervals of the two blocking probability parameters.

a) For all cases defined in *Table 2*, run 40 times simulator2, each time with a stopping criterion of $R = 10000$ and $N = 1000$. In all cases, consider the average movies duration $1/\mu = 90$ minutes. Analyse these results and take conclusions on the impact of the different input parameters on the two blocking probabilities.

Table 2

λ (requests/hour)	*S* (No. servers)	*W* (Mbps)	Standard Blocking Probability (%)	High-Definition Blocking Probability (%)
13	1	0		
13	1	60		
13	1	80		
50	3	0		
50	3	180		
50	3	240		

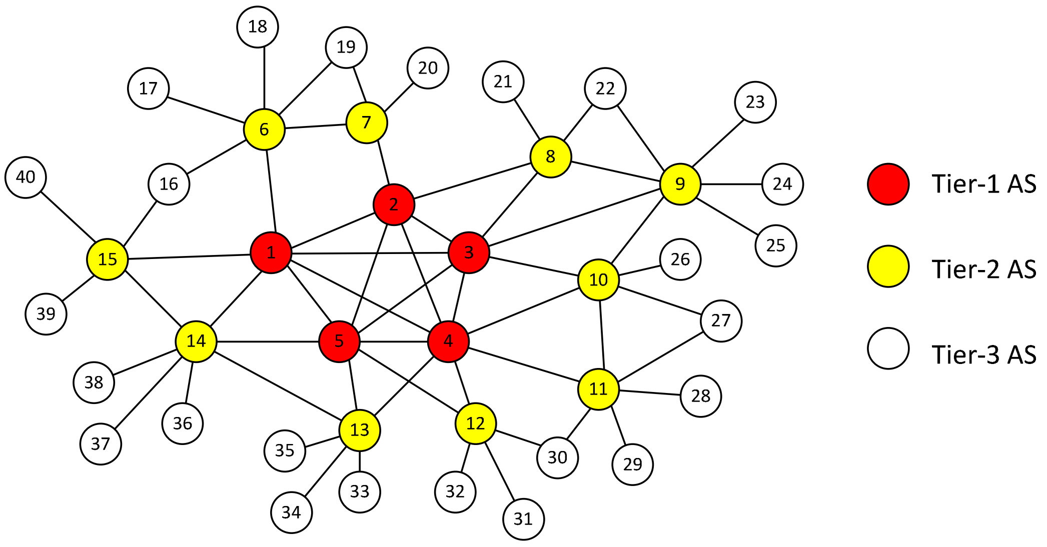
b) Consider that currently the company has a server farm with 2 servers and 8000 subscribers. In the prime time, each subscriber requests, on average, 1 movie per week and around 10% of all movie requests are for high-definition movies. Determine the reservation value *W* that should be used in the front-office to minimize the worst blocking probability between the two video format requests. What are the blocking probabilities for such value of *W*.

c) Consider that the company is launching a marketing campaign aiming to reach 17000 subscribers and 20% of high-definition requests (it is still expected that each subscriber requests, on average, 1 movie per week). Moreover, the company aims to have a robust solution for single server failures: (i) by default, the worst blocking probability must be not higher than 0.1%, (ii) when one server fails, the worst blocking probability must be not higher than 1%. Determine how many additional video-streaming servers are required for the targeted goals of the marketing campaign. Determine also the best reservation value *W* to be set in the front-office.

4. Third Part of the Assignment

In this third part of the assignment, consider a service provided by multiple server farms where, again, movies are available in the 2 previous possible video formats and both formats are available for all movies on all servers of all farms. In all farms, each server has an interface of 100 Mbps.

Consider that the Internet part that covers the company targeted subscribers is given by the following figure, which specifies the different types of Autonomous Systems (ASs) and how they are connected (the list of pairs of connected ASs is provided in MATLAB format in Appendix D):



Besides determining the total number of servers, the company also needs (i) to identify the ASs where the different server farms must be connected and (ii) to decide how many servers should be placed on each farm. Only Tier-2 of Tier-3 ASs provide the Internet access service. The average OPEX costs (Operational Costs) of a server farm is 8 when it is connected to a Tier-2 AS and 6 when it is connected to a Tier-3 AS. The company assumes that it can reach 2500 subscribers on each Tier-2 AS and 1000 subscribers on each Tier-3 AS with average requests of 2 movies per week per customer in the prime time and 20% of requests for high-definition movies.

a) Select a minimum cost set of ASs to install the server farms (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 OPEX cost of the solution?

b) Use simulator2 to determine the total number of required servers and the appropriate reservation *W* to provide a blocking probability around 1% for both types of movie requests.

c) To define the final solution, split the total number of servers determined in **b)** by the locations determined in **a)** in a proportion as close as possible to the number of subscribers that are closer to each server farm.

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

Appendix C – Specification of Simulator 2

Develop a MATLAB function named simulator2, implementing an event driven simulator for the service architecture based on one server farm with S servers, providing movies in 2 video formats and with a resource reservation of W for high-definition movies. As starting point, use the simulator1 MATLAB function that you have obtained in the first part of the assignment with the input parameter *N*. The input parameters of simulator2 must be:

λ –	movies request rate (in requests/hour)
p –	percentage of requests for high-definition movies (in %)
$1/\mu$ –	average duration of movies (in minutes)
S –	number of servers (each server with a capacity of 100 Mbps)
W –	resource reservation for high-definition movies (in Mbps)
M_S –	throughput of movies in standard definition (2 Mbps)
M_H –	throughput of movies in high definition (5 Mbps)
R –	number of movie requests to stop simulation
N –	movie request number to start updating the statistical counters

The performance parameters estimated by simulator2 must be:

b_S –	blocking probability of standard movie requests
b_H –	blocking probability of high-definition movie requests

The stopping criteria must be the time instant of the arrival of the movie request number *R*. In the simulator development, consider the following events:

ARRIVAL_S -	time instant of a standard movie request
ARRIVAL_H -	time instant of a high-definition movie request
DEPARTURE_S(i) -	time instant of a standard movie termination on server i (i = 1,…,S)
DEPARTURE_H(i) -	time instant of a high-definition movie termination on server i (i = 1,…,S)

Consider the following state variables:

STATE(i) -	total throughput of the movies in transmission by server i (i = 1,…,S)
STATE_S -	total throughput of standard movies in transmission

Consider the following statistical counters:

NARRIVALS -	total number of movie requests
NARRIVALS_S -	number of standard movie requests
NARRIVALS_H -	number of high-definition movie requests
BLOCKED_S -	number of blocked standard movie requests
BLOCKED_H -	number of blocked high-definition movie requests

Appendix D – List of pairs of connected ASs

```
G= [ 1  2
     1  3
     1  4
     1  5
     1  6
     1 14
     1 15
     2  3
     2  4
     2  5
     2  7
     2  8
     3  4
     3  5
     3  8
     3  9
     3 10
     4  5
     4 10
     4 11
     4 12
     4 13
     5 12
     5 13
     5 14
     6  7
     6 16
     6 17
     6 18
     6 19
     7 19
     7 20
     8  9
     8 21
     8 22
     9 10
     9 22
     9 23
     9 24
     9 25
    10 11
    10 26
    10 27
    11 27
    11 28
    11 29
    11 30
    12 30
    12 31
    12 32
    13 14
    13 33
    13 34
    13 35
    14 15
    14 36
    14 37
    14 38
    15 16
    15 39
    15 40];
```

Appendix E – Solving the server farm location problem using ILP (Integer Linear Programming)

We have a set of Autonomous Systems (ASs) and we aim to select a subset of ASs to connect one server farm on each selected AS. The solution must guarantee that in the network of ASs, there is a path between each Tier-2 (and Tier-3) AS and at least one server farm with no more than one intermediate AS. Consider the following notation:

n –	number of Tier-2 (and Tier-3) ASs where server farms can be connected to;
c_i –	OPEX cost of connecting a server farm to AS i, with $1 \leq i \leq n$;
$I(j)$ –	set of Tier-2 (and Tier-3) ASs such that there is a shortest path between AS j and each AS $i \in I(j)$ with at most one intermediate AS.

Consider the list G of AS pairs (i, j), as provided in the previous Appendix D. To compute each set $I(j)$, use the following labelling algorithm:

- Start by assigning label 0 to AS j and label –1 to all other ASs.
- Then, run for a=0 and a=1 the following cycle: for each AS pair $(i, j) \in G$, if one AS has label a and the other AS has label –1, assign label a+1 to the AS that has label –1.
- At the end: (i) the shortest path between AS j and each AS with label 1 has no intermediate ASs and (ii) the shortest path between AS j and each AS with label 2 has one intermediate AS.
- The set $I(j)$ is composed by all Tier-2 (and Tier-3) ASs whose label at the end of the algorithm is not negative.

Consider the following variables:

x_i –	binary variable, with $1 \leq i \leq n$, that when is equal to 1 means that AS i must be connected to one server farm;
y_{ji} –	binary variable, with $1 \leq j \leq n$ and $i \in I(j)$, that when is equal to 1 means that AS j is associated with AS i.

The Integer Linear Programming (ILP) model defining the optimization problem is defined as:

Minimize $\sum_{i=1}^{n} c_i x_i$ (1)

Subject to:

$\sum_{i \in I(j)} y_{ji} = 1 \qquad , j = 1 \dots n$ (2)

$y_{ji} \leq x_i \qquad , j = 1 \dots n, i \in I(j)$ (3)

$x_i \in \{0,1\} \qquad , i = 1 \dots n$ (4)

$y_{ji} \in \{0,1\} \qquad , j = 1 \dots n, i \in I(j)$ (5)

The objective function (1) is the minimization of the OPEX costs of the selected server farms. Constraints (2) guarantee that each AS j is associated with one AS $i \in I(j)$ while constraints (3) guarantee that an associated AS $i \in I(j)$ must have one server farm connected (when y_{ji} is 1, x_i must be also equal to 1 in constraints (3)). Therefore, constraints (2–3) guarantee that each AS j has always one server farm whose shortest path has at most one intermediate AS. Constraints (4–5) are the constraints that define all variables as binary ones.

Any set of values assigned to variables x_i and y_{ji} compliant with constraints (2–5) defines a feasible solution. A set of assigned values that provides the minimum value for the objective function (1) is an optimal solution: the variables x_i set to 1 define the selected ASs to be connected by server farms and the value of (1) is the minimum possible OPEX cost.

To solve the ILP model, develop a code to write an ASCII file using the LP format. The LP format is illustrated in the following example:

```
Minimize
 + 2 u1,1 + 3 u1,2 - 2.0 u1,3 + 5.2 u1,4 + 4.3 u2,1 + u2,2 - u2,3 + 2.5 u2,4
Subject To
\ restricao 1
 + u1,1 + 3 u1,2 - u2,1 - u2,2 = 0
\ restricao 2
 - 4.3 u1,1 + 3.5 u2,1 + ff1 - ff2 >= 3.54
\ restricao 3
 + 5 u1,4 - 3 u2,4 + 4.54 ff1 <= 0
Binary
 ff1
 ff2
General
 u1,1
 u1,2
 u1,3
 u1,4
End
```

- Variables can be named anything provided that the name does not exceed 255 characters, all of which must be alphanumeric (a-z, A-Z, 0-9) or one of the symbols ! " # $ % & ( ) , . ; ? @ _ ‘ ’ { } ~. A variable name cannot begin with a number or a period.
- In the line after `Minimize` (or `Maximize`), specify the objective function.
- In the lines after `Subject To`, specify the problem constraints.
- Constraints must be in canonical format, i.e., the variables before ‘=’ , ‘<=’ or ‘>=’ and the constant afterwards.
- Anything that follows a backslash (\) is a comment and is ignored until a return is encountered (blank lines are also ignored).
- In the lines after `Binary`, list all binary variables and in the lines after `General`, list all integer (non-binary) variables. All variables not listed in these fields are assumed to be real variables.

With the ASCII file defining the optimization problem in LP format, you can use any standard solver to solve the problem. There are some public sites that enable to solve ILP problems in the cloud. As a suggestion, use CPLEX, as provided in https://neos-server.org/neos/solvers/index.html. In this case, you also need to prepare another ASCII file with the content:

```
display solution variables -
```

to configure the server to feedback the values of the variables of the optimal solution at the end of the problem resolution.