Formal Methods for System Verification Project Documentation

A dynamic server allocation for energy efficiency

Martina Donadi, 865737 **Fabio Dainese, 857661**

Ca' Foscari, University of Venice

December 1, 2019

***Keywords**: PEPA Language, Continuous Time Markov Chain, Server Power Consumption*

Fabio Dainese (email: 857661@stud.unive.it), **Martina Donadi** (email: 865737@stud.unive.it) - Master degree in 'Software Dependability and Cyber Security' (www.unive.it).

Table of Contents

1 Introduction **2**
- 1.1 Multi-Server Queue Model . . . 2

2 Components and Activities of the System **3**
- 2.1 Components of the System . . . 3
- 2.2 Activities of the System . . . 4
  - 2.2.1 Activities of the Queue . . . 4
  - 2.2.2 Activities of the Arrival Stream . . . 4
  - 2.2.3 Activities of the Server . . . 4

3 Interactions Between the Components **4**

4 PEPA Components of the System **5**
- 4.1 Queue PEPA Component . . . 5
- 4.2 Arrival Stream PEPA Component . . . 6
- 4.3 Server PEPA Component . . . 6

5 Entire System Expressed in PEPA **8**

6 Derivation Graph of the System **8**

7 Infinitesimal Generator Matrix **11**

8 Evaluation **12**
- 8.1 Global Balance Equations . . . 12
- 8.2 Sojourn Time . . . 13
- 8.3 Utilisation . . . 14

9 Evaluation of the System Using the PEPA Eclipse Plug-In **14**

10 References **19**

1 Introduction

Power consumption in data centres receive a huge concern by data centre providers and that's why in this project we are going to model a policy, in PEPA language, that it will dynamically perform the powering on or off of the servers in order to minimize the power consumption according to the demand.

The system taken into consideration consists of a given number (N) of homogeneous servers. These servers may be in any of the following five states:

- Powered up (wherein the server may be either active or idle);
- Powering up;
- Powered down;
- Powering down;
- Fault.

In the transitional conditions, i.e. *powering down* or *powering up*, and the *fault* condition, servers are unable to respond to service demands.
Although the *fault* condition could occur simultaneously with any of the other four conditions, in this case it's considered only in the transitional states (*powering up* and *powering down*) due to the increased likelihood of faults becoming apparent then.

There exists various types of heuristics that are used to organise the servers usage according to the demand, but in this paper we're considering only the 'High/Low' one.
This type of heuristic takes into consideration the arrival periods, switching time, processing time and the queue length. Furthermore, we assume that the system is very stable, meaning that the jobs' arrival rates are categorized in λ_{high} or λ_{low} binary form and that the jobs time/duration had a fixed value of μ.
The high/low is a sophisticated heuristic which incorporates the system performance and stability into its definition.

The PEPA model considered in this paper is the one described in [1] and can basically be regarded as a multi-server queue.

1.1 Multi-Server Queue Model

Variation in service and arrival rates causes queues. The problem of minimizing power consumption in data centres can be related to the one of modelling a queue to provide quality of service minimizing costs. In order to model a queue at least three parameters are necessary:

- Arrival rate : how fast a customer is showing up to reach a server in the data center;
- Service rate (μ) : how fast a server can handle a customer's request;
- Number of servers.

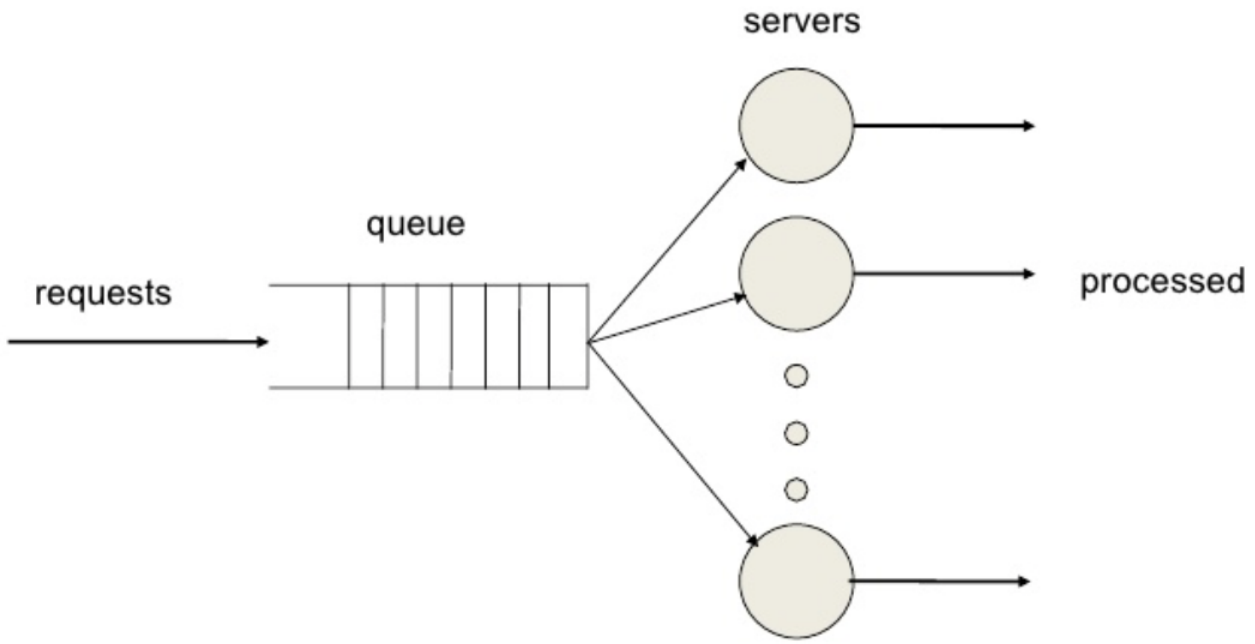


Figure 1: Multi server queue

For our model the number of servers is considered to be varying and bounded at N as well as the maximum number of jobs (i.e. the maximum number of customers in the queue) is bounded at N. The arrival rate of jobs in the system is varying: jobs arrive at either a high rate (λ_{high}) or at a low rate (λ_{low}). The service rate is varying according to the number of active servers: M servers out of the N are supposed to be static, i.e. to remain permanently active. When the queue is not full it engages in an arrival activity (with high/low rate of arrival depending on the current arrival rate of the system). When the queue is full the arrival process is suspended and jobs are lost. When there are i jobs ($0 < i \leq N$) in the queue, it engages in a serve activity at rate $i * \mu$.

In the following section the model is analyzed in details in terms of components, activities, PEPA representation and problems associated with it.

2 Components and Activities of the System

2.1 Components of the System

As we already mentioned before, the model taken into consideration uses a variant of the high/low policy, in which we have M static servers that remain permanently available to serve jobs, meanwhile the remaining servers might be turn on and off in response to the high and low periods of jobs' arrivals ($M - N$ servers).

The system will be composed by the following components:

- Q_i: Queue's current state, with $0 \leq i \leq N$, which represent the number of jobs in the system at the time i;
- $Arrival_{high}$, $Arrival_{low}$: Represent the two possible jobs' arrival stream states (high or low);
- All the possible server's state are represented by:
  - $Server_{on}$: State in which the server is either active or idle;

- $ServerPowering_{on}$: State in which the server is turning on;
- $Server_{off}$: State in which the server if fully turned off;
- $ServerPowering_{off}$: State in which the server is turning off;
- $Server_{failOn}$: State in which the server encountered an error performing the power-up activity;
- $Server_{failOff}$: State in which the server encountered an error performing the shutdown activity;
- $Server_{static}$: State representing a server that is always active (M servers).

2.2 Activities of the System

In this section we will describe all the possible behaviours, i.e. activities, that each component of the system can perform.

2.2.1 Activities of the Queue

The set of possible actions that the *queue* component can perform are:

- **Service**: When a request is successfully elaborated and the job leave the system (at a fixed rate of μ);
- **arrivalH**: When the arrivals into the system occurs at high rate λ;
- **arrivalL**: When the arrival into the system occurs at low rate ϵ.

2.2.2 Activities of the Arrival Stream

The activities of the *arrival stream*, identified by the components $Arrival_{high}$ and $Arrival_{low}$, are modeled by the previously defined queue activities, i.e. **arrivalH** and **arrivalL**, and also by the switching actions between the high and low rate defined through the activities:

- **highPeriodEnd** at rate β;
- **lowPeriodEnd** at rate γ.

2.2.3 Activities of the Server

The activities of the server are modeled by the previously defined activities **highPeriodEnd**, **lowPeriodEnd**, **service** and by new ones, such as:

- **powerup**: Used to turn on a server (η rate);
- **poweroff**: Used to turn off a server (ξ rate);
- **repair**: Used to fix the server in case of error/fault ($\sigma rate$);

3 Interactions Between the Components

To sum up the various behaviours of the previously described components, we can say that the:

- **Queue** will accept the incoming jobs, regardless the type of arrival, up until the queue if full ($i \leq n$), plus every time a job is served the queue index counter is decreased by 1;
- **Arrival Stream** will be in charge of detecting and switching accordingly to the appropriate arrival rate stream situation (high or low);

- **Server** will power-up or down accordingly to the high or low jobs arrival. To keep in mind that we have considered also M static servers.

The interaction between these components are governed by some common activities in which they have to cooperate, such as:

- $\mathbf{Arrival_{high}}$ and $\mathbf{Server_{on}}$ whenever the '*highPeriodEnd*' activity is performed (moving respectively the system states to $\mathbf{Arrival_{low}}$ and $\mathbf{ServerPowering_{off}}$);
- $\mathbf{Arrival_{low}}$ and $\mathbf{Server_{off}}$ whenever the '*lowPeriodEnd*' activity is performed (moving respectively the system states to $\mathbf{Arrival_{high}}$ and $\mathbf{ServerPowering_{on}}$);
- $\mathbf{Arrival_{high}/Arrival_{low}}$ and $\mathbf{Q_i}$ whenever the '*arrivalH/arrivalL*' activities are performed (moving respectively the system state to $\mathbf{Q_{i+1}}$ and remaining to the corresponding $\mathbf{Arrival_{high}/Arrival_{low}}$ state);
- $\mathbf{Server_{on}}$ and $\mathbf{Q_i}$ whenever the '*service*' activity is performed (moving respectively the system states to $\mathbf{Server_{on}}$ and $\mathbf{Q_{i-1}}$);

The rest of the activities will change the system's states as follows:

- $\mathbf{Server_{static}}$ will always perform the '*service*' activity and remains in its state;
- Whenever $\mathbf{ServerPowering_{on}}$ will perform the '*powerup*' activity it has a probability ρ of encountering a problem/fail that will lead it to the state $\mathbf{Server_{failOn}}$ (and therefore it has the probability of $(1-\rho)$ to perform correctly the activity leading it to the state $\mathbf{Server_{on}}$);
- Whenever $\mathbf{ServerPowering_{off}}$ will perform the '*poweroff*' activity it has a probability ρ of encountering a problem/fail that will lead it to the state $\mathbf{Server_{failOff}}$ (and therefore it has the probability of $(1-\rho)$ to perform correctly the activity leading it to the state $\mathbf{Server_{off}}$);
- Whenever the system is in the $\mathbf{Server_{failOn}}/\mathbf{Server_{failOff}}$ state it can only perform the '*repair*' activity, moving it to he corresponding $\mathbf{Server_{on}}/\mathbf{Server_{off}}$ state.

A more detailed and formal description of the interactions between the components will be presented in the next section.

4 PEPA Components of the System

4.1 Queue PEPA Component

The *Queue* PEPA component of the system is defined as follows:

$$
\begin{aligned}
Q_0 &\stackrel{\text{def}}{=} (arrivalH, \lambda).Q_1 + (arrivalL, \epsilon).Q_1 \\
Q_1 &\stackrel{\text{def}}{=} (arrivalH, \lambda).Q_2 + (arrivalL, \epsilon).Q_2 + (service, \mu).Q_1 \\
&\vdots \\
Q_i &\stackrel{\text{def}}{=} (arrivalH, \lambda).Q_{i+1} + (arrivalL, \epsilon).Q_{i+1} + (service, \mu).Q_{i-1} \\
&\vdots \\
Q_n &\stackrel{\text{def}}{=} (service, \mu).Q_{n-1}
\end{aligned}
$$

Meaning that whenever it arrives a new job into the system, either at a lower or higher rate, it will be added into the waiting queue up until the queue is full ($i \leq n$, where n is equals to the queue maximum capacity).
Meanwhile, when a job is served, it will leave the system (modeled by decreasing the queue state index i by 1).

The previously described *queue* PEPA component can be also graphically represented by the following derivation graph:

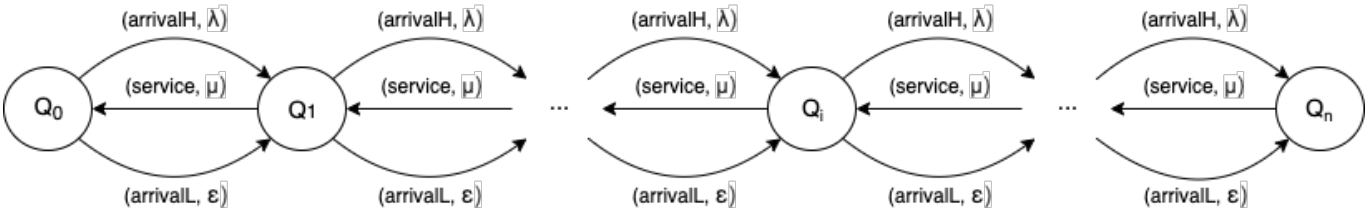


Figure 2: Queue PEPA derivation graph

4.2 Arrival Stream PEPA Component

The *Arrival Stream* PEPA component of the system is defined as follows:

$$Arrival_{high} \stackrel{\text{def}}{=} (arrivalH, \lambda).Arrival_{high} + (highPeriodEnd, \beta).Arrival_{low}$$
$$Arrival_{low} \stackrel{\text{def}}{=} (arrivalL, \epsilon).Arrival_{low} + (lowPeriodEnd, \gamma).Arrival_{high}$$

Meaning that whenever the arrival rate is set in either *high* or *low*, the system will remain in its current state until that specific period ends and the system will adjust itself by using the switching operations *highPeriodEnd* or *lowPeriodEnd* accordingly.

The previously described *arrival stream* PEPA component can be also graphically represented by the following derivation graph:

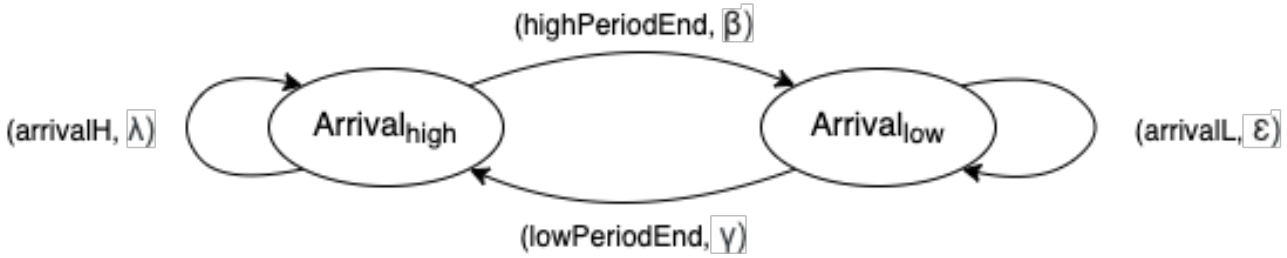


Figure 3: Arrival Stream PEPA derivation graph

4.3 Server PEPA Component

The *Server* PEPA component of the system is defined as follows:

$$ServerPowering_{on} \stackrel{\text{def}}{=} (powerup, \eta * (1-\rho)).Server_{on} + (powerup, \eta * \rho).Server_{failOn}$$
$$Server_{on} \stackrel{\text{def}}{=} (service, \mu).Server_{on} + (highPeriodEnd, \beta).ServerPowering_{off}$$
$$ServerPowering_{off} \stackrel{\text{def}}{=} (poweroff, \xi * (1-\rho)).Server_{off} + (poweroff, \xi * \rho).Server_{failOff}$$
$$Server_{off} \stackrel{\text{def}}{=} (lowPeriodEnd, \gamma).ServerPowering_{on}$$

$$Server_{static} \stackrel{\text{def}}{=} (service, \mu).Server_{static}$$
$$Server_{failOn} \stackrel{\text{def}}{=} (repair, \sigma).Server_{on}$$
$$Server_{failOff} \stackrel{\text{def}}{=} (repair, \sigma).Server_{off}$$

The presented states definitions illustrate the normal flow between all the possible server's states.
To point out that we defined ρ as the probability that an error occurs during the execution of the *powerup* and/or *poweroff* activities.

The previously described *server* PEPA component can be also graphically represented by the following derivation graphs:

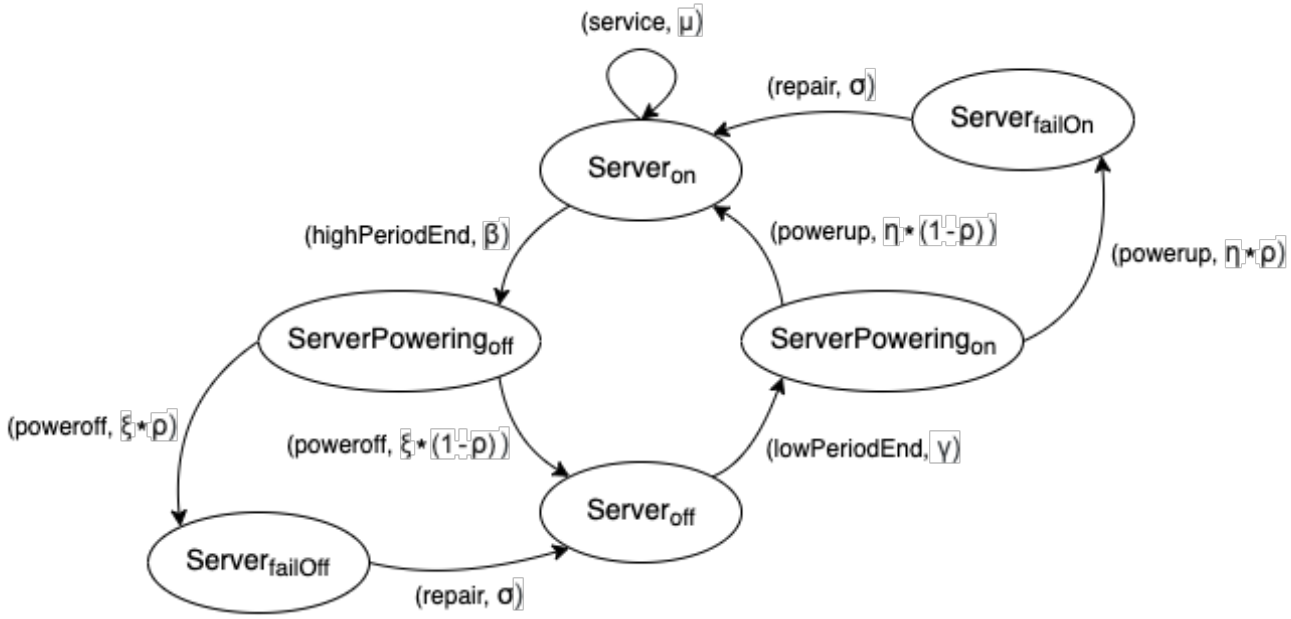


Figure 4: Server PEPA derivation graph

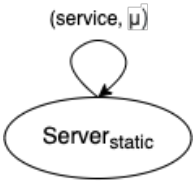


Figure 5: Static server PEPA derivation graph

5 Entire System Expressed in PEPA

The entire system in the PEPA process algebra is defined as follows:

$$System \stackrel{\text{def}}{=} ((Arrival_{high} \underset{K}{\bowtie} Server_{on} \underset{K}{\bowtie} ... \underset{K}{\bowtie} Server_{on}) \underset{Z}{\bowtie} Server_{static}[M]) \underset{L}{\bowtie} Q_0$$

Where respectively the sets K, Z, L are defined as follows:

- $K = \{highPeriodEnd\}$;
- $Z = \emptyset$;
- $L = \emptyset$.

Just to recall, the number of '$Server_{on}$' present on the '$System$' PEPA definition are equals to $N - M$, where N are the total number of servers and M are the static ones.

So, for example if we choose $N = 6$ and $M = 3$, we are considering a system with a total of 6 servers, in which 3 of them are static and the other 3 are dynamic.
So, given those specifications the system PEPA definition is defined as follows:

$$System \stackrel{\text{def}}{=} ((Arrival_{high} \underset{K}{\bowtie} Server_{on} \underset{K}{\bowtie} Server_{on} \underset{K}{\bowtie} Server_{on}) \underset{Z}{\bowtie} Server_{static}[M]) \underset{L}{\bowtie} Q_0$$

In which $K = \{highPeriodEnd\}, Z = L = \emptyset$ and $M = 3$.

6 Derivation Graph of the System

The simplest way to represent the Markov Process underlying the PEPA model is in terms of its state transition diagram, which can be derived from the model's derivation graph.

The derivation graph related to the considered system grows, in terms of components a therefore in its complexity, very fast due to the presence of many states involved in the system definition, so for simplicity we are going to consider only a sub-part of the entire system. Just for prospective, considering just one static server, one dynamic server and a queue component of length 2 would result in a 36 states system. For this reason we decided to analyse just the interaction between the components $Arrival_{high}$ and $Server_{on}$ as defined by the following PEPA equation:

$$Arrival_{high} \underset{\{highPeriodEnd\}}{\bowtie} Server_{on}$$

This generates a 12 states diagram as summarized in the following image:

12 states

1	Arrival_high	Server_on	0.7511560761484473
2	Arrival_low	ServerPowering_off	0.03755780380742236
3	Arrival_high	ServerPowering_off	0.037557803807422366
4	Arrival_low	Server_off	0.017839956808525625
5	Arrival_low	Server_failOff	0.0018778901903711179
6	Arrival_high	Server_off	0.0572756508063191
7	Arrival_high	Server_failOff	0.005633670571113353
8	Arrival_low	ServerPowering_on	0.00891997840426281
9	Arrival_high	ServerPowering_on	0.0661956292105819
10	Arrival_low	Server_on	0.00847397948404967
11	Arrival_low	Server_failOn	4.459989202131406E-4
12	Arrival_high	Server_failOn	0.007065561841271393

Figure 6: Space state view for the system equation - PEPA Eclipse plug-in

For readability reasons the provided derivation graph (Figure 7) will have in each node a number corresponding to the associated states row present in 'Figure 6'.

The derivation graph for the considered PEPA equation is illustrated as follows:

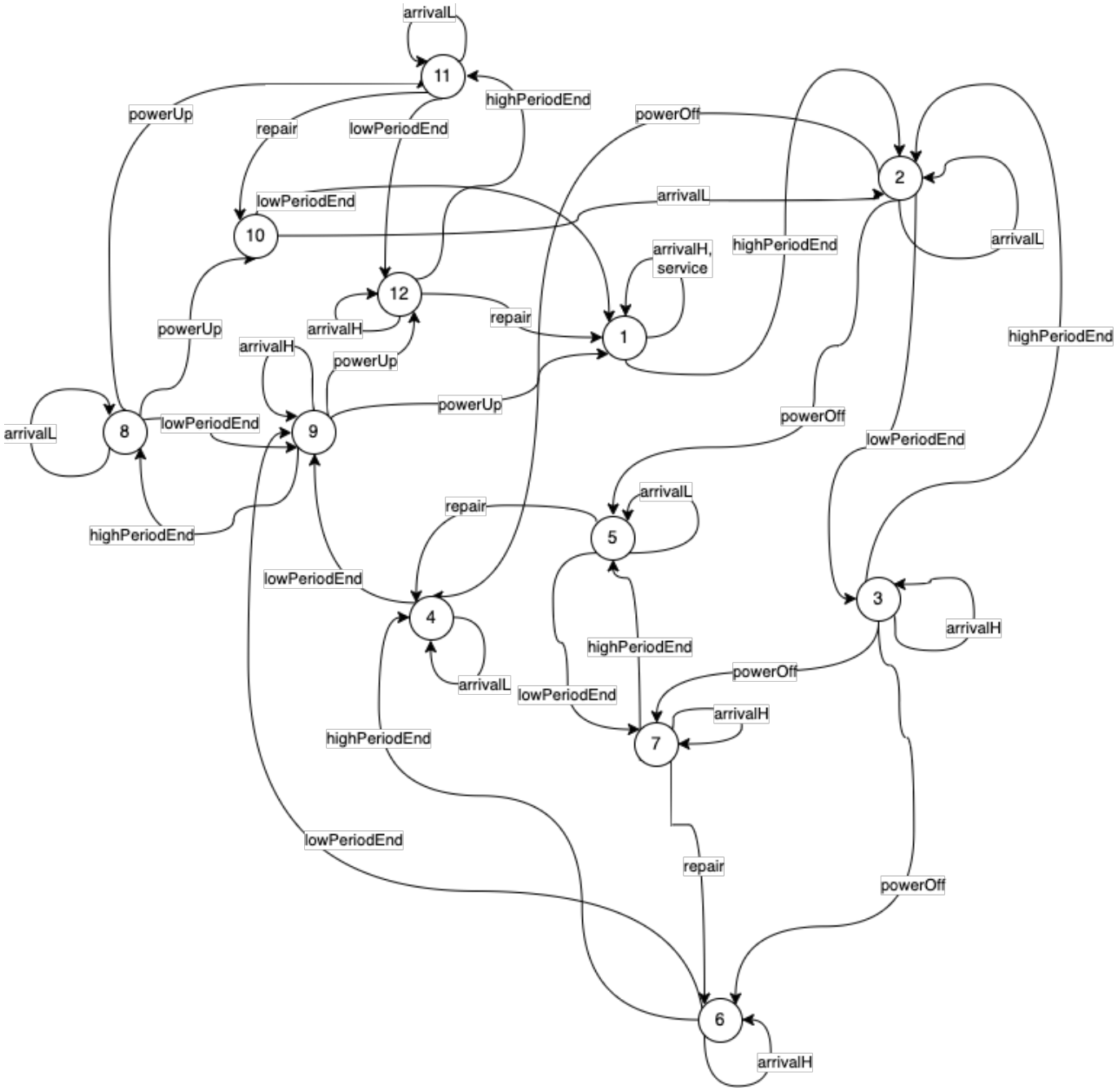


Figure 7: Sub-system derivation graph

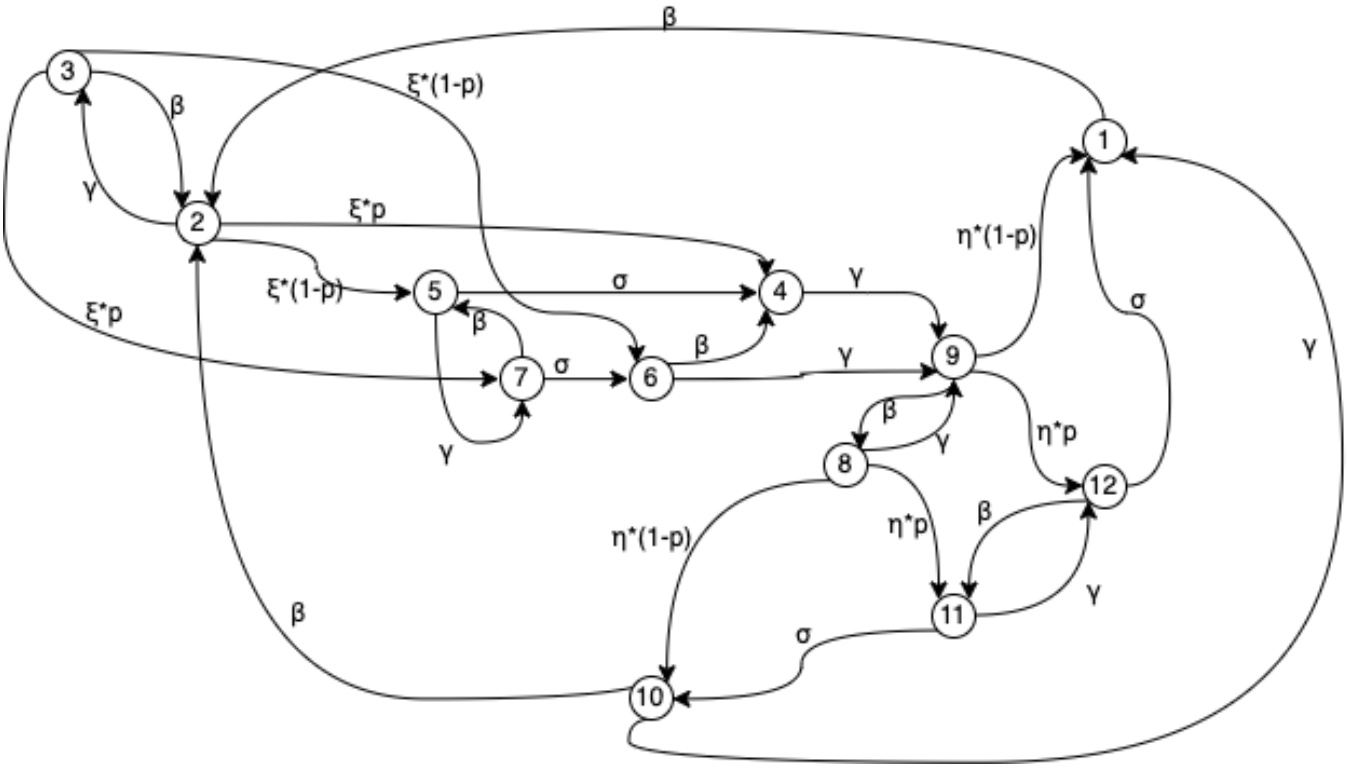


Figure 8: Continuous Time Markov Chain graph

7 Infinitesimal Generator Matrix

In this section we are going to present the infinitesimal generator matrix Q of the Markov process associated to the previous described components' subset of the system (section 6).

For space and complexity reasons ($|S| = 12$) we will not report the associated *Continuous Time Markov Chain* representation graph for the chosen components' subset.

Just to recall, the *infinitesimal generator matrix* is constructed such that for every element q_{ij} (with $i \neq j$, where i, j start from 1 up to the number of states involved) denotes the rate departing from i and arriving in state j (if there are multiple arrow starting from i and arriving in j, just consider the sum of the rates). Meanwhile the diagonal elements q_{ii} are defined such that:

$$q_{ii} = -\sum_{j \neq i} q_{ij}$$

The infinitesimal generator matrix associated to the analyzed components is defined as follows:

$$Q = \begin{bmatrix}
-\beta & \beta & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -(\gamma+\xi) & \gamma & \xi*(1-\rho) & \xi*\rho & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \beta & -(\beta+\xi) & 0 & 0 & \xi*(1-\rho) & \xi*\rho & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -\gamma & 0 & 0 & 0 & 0 & \gamma & 0 & 0 & 0 \\
0 & 0 & 0 & \sigma & -(\sigma+\gamma) & 0 & \gamma & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \beta & 0 & -(\beta+\gamma) & 0 & 0 & \gamma & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \beta & \sigma & -(\beta+\sigma) & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -(\gamma+\eta) & \gamma & \eta*(1-\rho) & \eta*\rho & 0 \\
\eta*(1-\rho) & 0 & 0 & 0 & 0 & 0 & 0 & \beta & -(\beta+\eta) & 0 & 0 & \eta*\rho \\
\gamma & \beta & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -(\beta+\gamma) & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \sigma & -(\sigma+\gamma) & \gamma \\
\sigma & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \beta & -(\sigma+\beta)
\end{bmatrix}$$

$$Q_N^T = \begin{bmatrix}
-\beta & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \eta*(1-\rho) & \gamma & 0 & \sigma \\
\beta & -(\gamma+\xi) & \beta & 0 & 0 & 0 & 0 & 0 & 0 & \beta & 0 & 0 \\
0 & \gamma & -(\beta+\xi) & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \xi*(1-\rho) & 0 & -\gamma & \sigma & \beta & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \xi*\rho & 0 & 0 & -(\sigma+\gamma) & 0 & \beta & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \xi*(1-\rho) & 0 & 0 & -(\beta+\gamma) & \sigma & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \xi*\rho & 0 & \gamma & 0 & -(\beta+\sigma) & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -(\gamma+\eta) & \beta & 0 & 0 & 0 \\
0 & 0 & 0 & \gamma & 0 & \gamma & 0 & \gamma & -(\beta+\eta) & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \eta*(1-\rho) & 0 & -(\beta+\gamma) & \sigma & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \eta*\rho & 0 & 0 & -(\sigma+\gamma) & \beta \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1
\end{bmatrix}$$

8 Evaluation

8.1 Global Balance Equations

In this section we are going to present how to calculate the *global balance equation*, but we are not presenting the actual calculation due to complexity reasons.

Just to remind that finding the *global balance equation* means calculating the following formula:

$$\pi Q = 0$$

Moreover, since the previously presented state space had 12 states, the global balance equations will have $12 + 1$ equations (the 13^{th} one is the normalization condition) in 12 unknowns as represented below:

$$\begin{cases}
\pi_1 \cdot \beta = \pi_9 \cdot (\eta \cdot (1-\rho)) + \pi_{10} \cdot \gamma + \pi_{12} \cdot \sigma \\
\pi_2 \cdot (\gamma + \xi) = \pi_1 \cdot \beta + \pi_3 \cdot \beta + \pi_{10} \cdot \beta \\
\pi_3 \cdot (\beta + \xi) = \pi_2 \cdot \gamma \\
\pi_4 \cdot \gamma = \pi_2 \cdot (\xi \cdot (1-p)) + \pi_5 \cdot \sigma + \pi_6 \cdot \beta \\
\pi_5 \cdot (\sigma + \gamma) = \pi_2 \cdot (\xi \cdot \rho) + \pi_7 \cdot \beta \\
\pi_6 \cdot (\beta + \gamma) = \pi_3 \cdot (\xi \cdot (1-\rho)) + \pi_7 \cdot \sigma \\
\pi_7 \cdot (\beta + \sigma) = \pi_3 \cdot (\xi \cdot \rho) + \pi_5 \cdot \gamma \\
\pi_8 \cdot (\gamma + \eta) = \pi_9 \cdot \beta \\
\pi_9 \cdot (\beta + \eta) = \pi_4 \cdot \gamma + \pi_6 \cdot \gamma + \pi_8 \cdot \gamma \\
\pi_{10} \cdot (\beta + \gamma) = \pi_8 \cdot (\xi \cdot (1-\rho)) + \pi_{11} \cdot \sigma \\
\pi_{11} \cdot (\sigma + \gamma) = \pi_8 \cdot (\xi \cdot \rho) + \pi_{12} \cdot \beta \\
\pi_{12} \cdot (\sigma + \beta) = \pi_9 \cdot (\xi \cdot \rho) + \pi_{11} \cdot \gamma \\
\pi_1 + \pi_2 + \pi_3 + \pi_4 + \pi_5 + \pi_6 + \pi_7 + \pi_8 + \pi_9 + \pi_{10} + \pi_{11} + \pi_{12} = 1
\end{cases}$$

So, once the above system equations is resolved, i.e. you find out the values for each π_i for $i = 1, ..., 12$, those values will tell you the portion of time, expressed in probability, that the system behaves like the π_i components/state.

8.2 Sojourn Time

The dynamic behaviour of the system is represented by the transitions between its states and the times spent in states: the sojourn times. The sojourn times are memoryless.

Just to recall, the *exit rate* from a component P is the rate at which the system leaves the state corresponding to the component P.
So, regarding our sub-system, the various *exit rates* are:

$$\begin{aligned} q(1) &= \beta \\ q(2) &= \gamma + \xi \\ q(3) &= \beta + \xi \\ q(4) &= \gamma \\ q(5) &= \sigma + \gamma \\ q(6) &= \beta + \gamma \\ q(7) &= \beta + \sigma \\ q(8) &= \sigma + \eta \\ q(9) &= \beta + \eta \\ q(10) &= \beta + \gamma \\ q(11) &= \sigma + \gamma \\ q(12) &= \sigma + \beta \end{aligned}$$

Now that we have all the various *exit rates* we can calculate the *sojourn time* for each state. As before, we recall that the *sojourn time* of a component P is equals to $ST(p) = (q(p))^{-1}$, where $q(p)$ identifies the exit rate of the component 'p'.

The relative *sojourn times* of the analyzed sub-part of the system are presented below:

$$
\begin{aligned}
ST(1) &= \frac{1}{\beta} \\
ST(2) &= \frac{1}{\gamma + \xi} \\
ST(3) &= \frac{1}{\beta + \xi} \\
ST(4) &= \frac{1}{\gamma} \\
ST(5) &= \frac{1}{\sigma + \gamma} \\
ST(6) &= \frac{1}{\beta + \gamma} \\
ST(7) &= \frac{1}{\beta + \sigma} \\
ST(8) &= \frac{1}{\sigma + \eta} \\
ST(9) &= \frac{1}{\beta + \eta} \\
ST(10) &= \frac{1}{\beta + \gamma} \\
ST(11) &= \frac{1}{\sigma + \gamma} \\
ST(12) &= \frac{1}{\sigma + \beta}
\end{aligned}
$$

8.3 Utilisation

The *utilisation* measure is the total probability that the model is in one of the states in which the resource is in use. So, regarding our sub-set of the system the utilization variables are reported below:

$$U_{Arrival_{high}} = \pi_1 + \pi_3 + \pi_6 + \pi_7 + \pi_9 + \pi_{12}$$

$$U_{Server_{on}} = \pi_1 + \pi_{10}$$

9 Evaluation of the System Using the PEPA Eclipse Plug-In

In this final section we have modeled our entire system using the PEPA Eclipse plug-in in order to examine and extrapolate further meaningful information that there will be presented below.

First of all, the system expressed using the PEPA plug-in is equals to the following snippet of code:

```
l = 50.0;
e = 10.0;
m = 10.0;
b = 10.0;
```

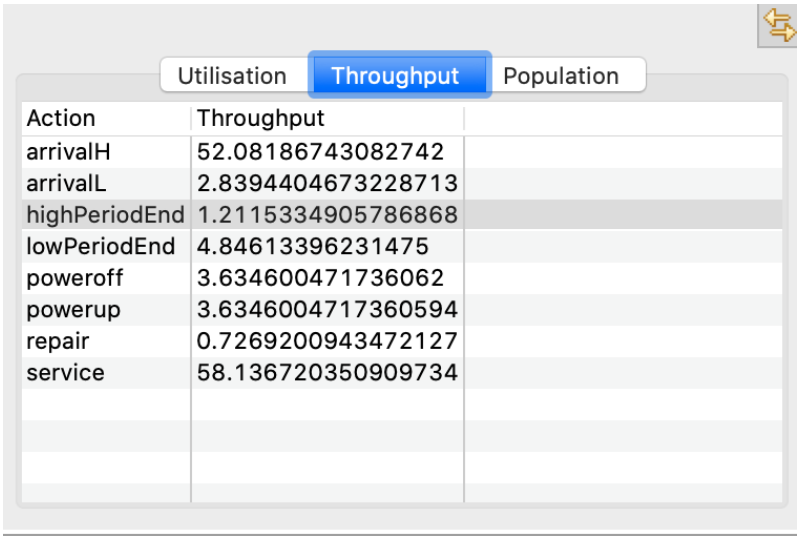
```
g = 10.0;
n = 100.0;
r = 0.1;
x = 100.0;
s = 1.0;

Q0 = (arrivalH,l).Q1 + (arrivalL,e).Q1;
Q1 = (arrivalH,l).Q2 + (arrivalL,e).Q2 + (service,m).Q0;
Q2 = (service,m).Q1;

Arrival_high = (arrivalH,l).Arrival_high + (highPeriodEnd,b).Arrival_low;
Arrival_low = (arrivalL,e).Arrival_low + (lowPeriodEnd,g).Arrival_high;

ServerPowering_on = (powerup, n * (1-r)).Server_on + (powerup, n*r).Server_failOn;
Server_on = (service,m).Server_on + (highPeriodEnd,b).ServerPowering_off;
ServerPowering_off = (poweroff,x*(1-r)).Server_off + (poweroff,x*r).Server_failOff;
Server_off = (lowPeriodEnd,g).ServerPowering_on;

Server_static = (service,m).Server_static;
Server_failOn = (repair,s).Server_on;
Server_failOff = (repair,s).Server_off;

((Arrival_high <highPeriodEnd> Server_on <highPeriodEnd> Server_on <highPeriodEnd> Server_on) <>
    Server_static <> Server_static <> Server_static ) <> Q0
```

Listing 1: PEPA Eclipse Plug-In of the System

To point out that we have modeled a system having $N = 6$ servers, which 3 of them are *dynamic* and the rest are *static* (i.e. always on).
For complexity reasons the queue capacity has been bounded at 2.
Finally the various rates were set as follows:

- **High arrival rate**: 50;
- **Low arrival rate**: 10;
- **Powering-up and down**: 100 in equally *high* and *low* period of job arrivals.

Using this specific system configuration on the plug-in environment, we retrieved the **throughput** as illustrated in the following image:



Action	Throughput
arrivalH	52.08186743082742
arrivalL	2.8394404673228713
highPeriodEnd	1.2115334905786868
lowPeriodEnd	4.84613396231475
poweroff	3.634600471736062
powerup	3.6346004717360594
repair	0.7269200943472127
service	58.136720350909734

Figure 9: Throughput evaluation of the system

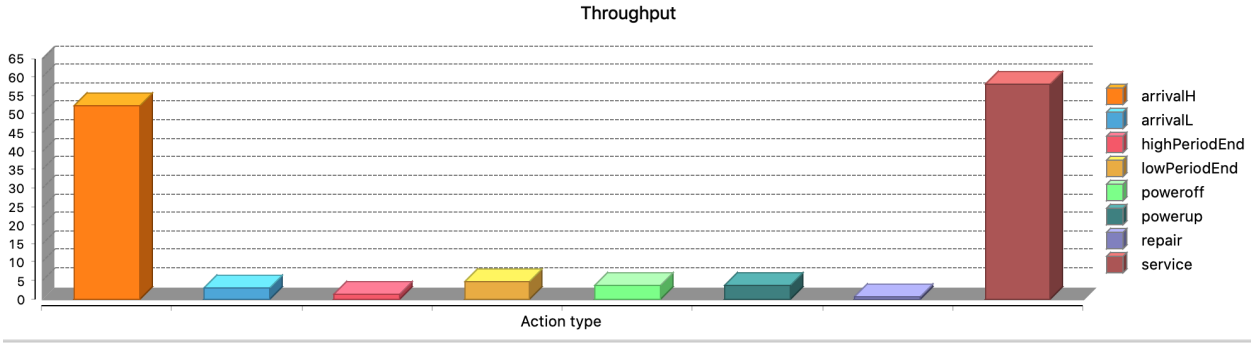


Figure 10: Throughput evaluation of the system

Moreover, in the following images you can appreciate the **utilization** regarding the various system's states.

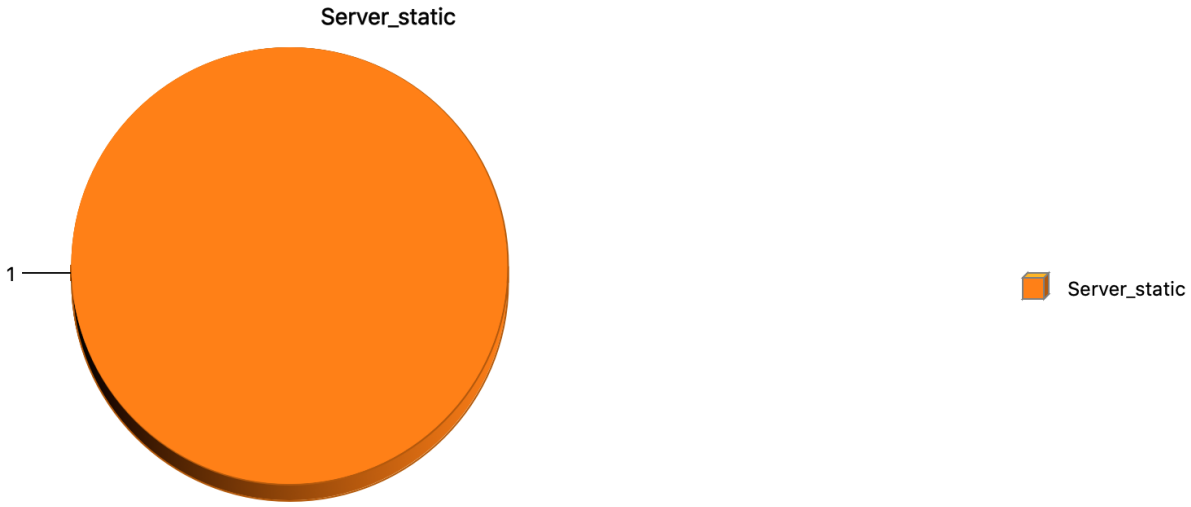


Figure 11: Utilisation Static Server component

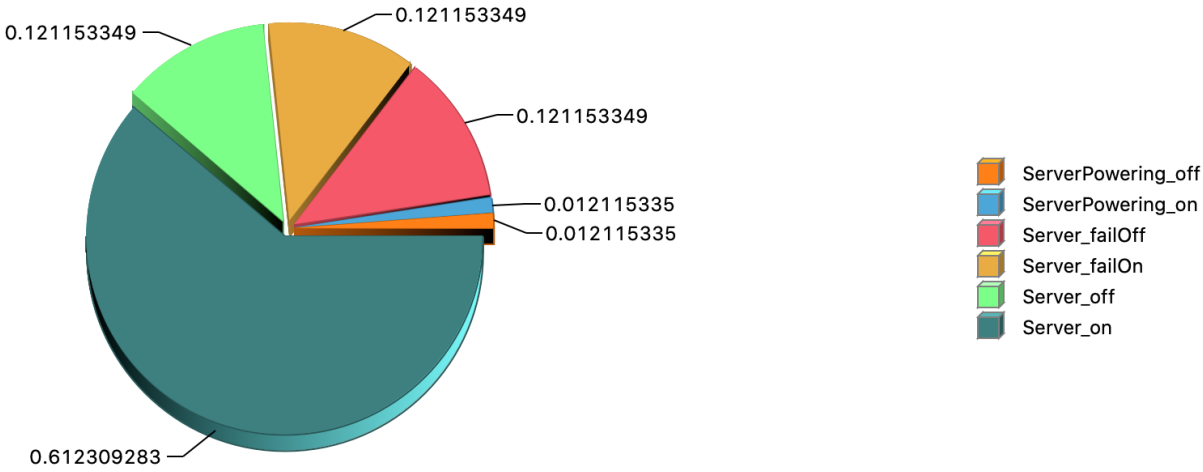


Figure 12: Utilisation Dynamic Server component

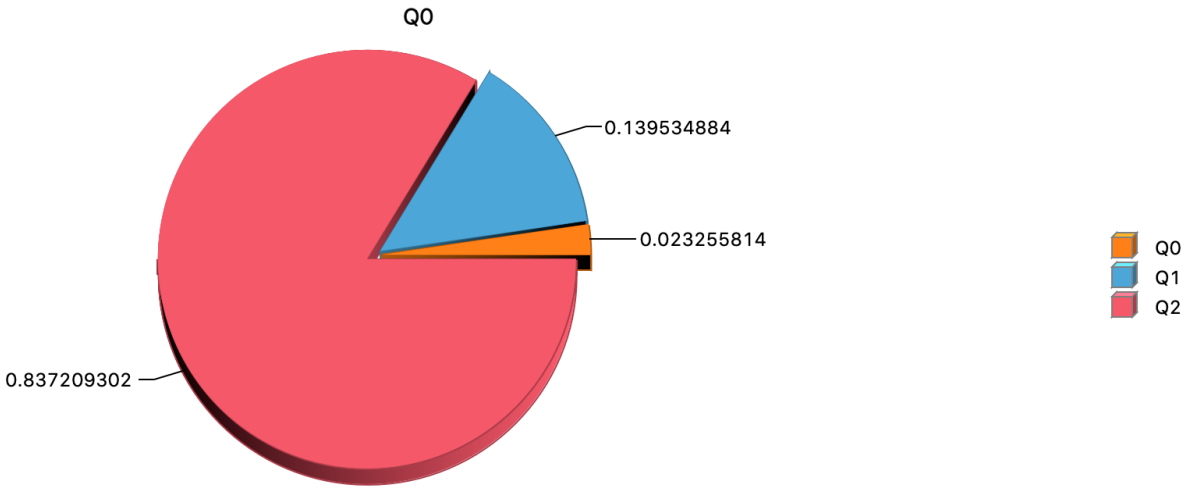


Figure 13: Utilisation Queue component

Finally in the following image it will be presented the **population** for the considered system's configuration.

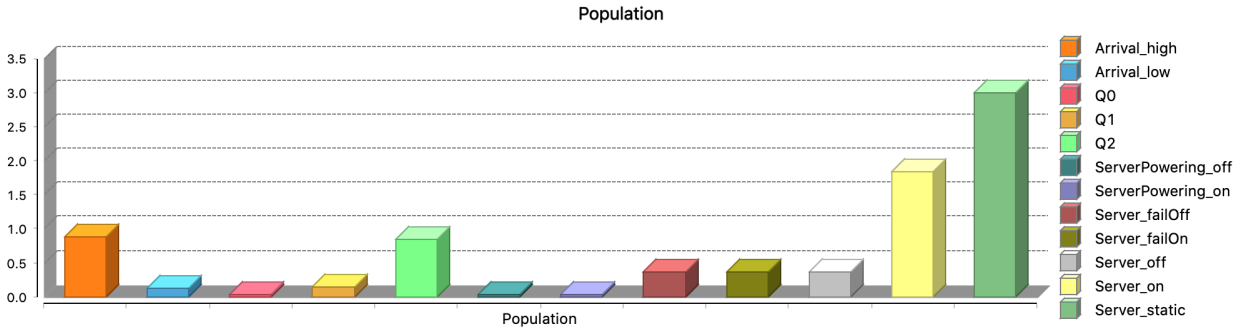


Figure 14: Population for a 3 dynamic + 3 static servers configuration

10 References

[1] Ali Alssaiari and Nigel Thomas. Performance modelling of dynamic server allocation for energy efficiency using pepa. September 2016.