UNIVERSITÀ
DEGLI STUDI
FIRENZE

Quantitative Evaluation of Stochastic Models

PARALLEL SERVER SEMANTIC WITH BERNSTEIN PHASE TYPES

Giacomo Magistrato

Introduction

Context: realization of a model characterized by a pool of containers dedicated to handling service requests.

- These requests follow an arrival process modeled according to a Poisson distribution, while service times are represented by means of a **Bernstein Phase-Type** distribution

- The **Bernstein Phase-Type (BPD) distribution** provide great flexibility in modeling stochastic phenomena with **Markovian processes**. This is particularly useful in contexts such as **GSPNs.**

- BPD models the time required for a Markov process to transition through a set of transient states. It consists of a combination of **exponential distributions**, each corresponding to a phase of the process.

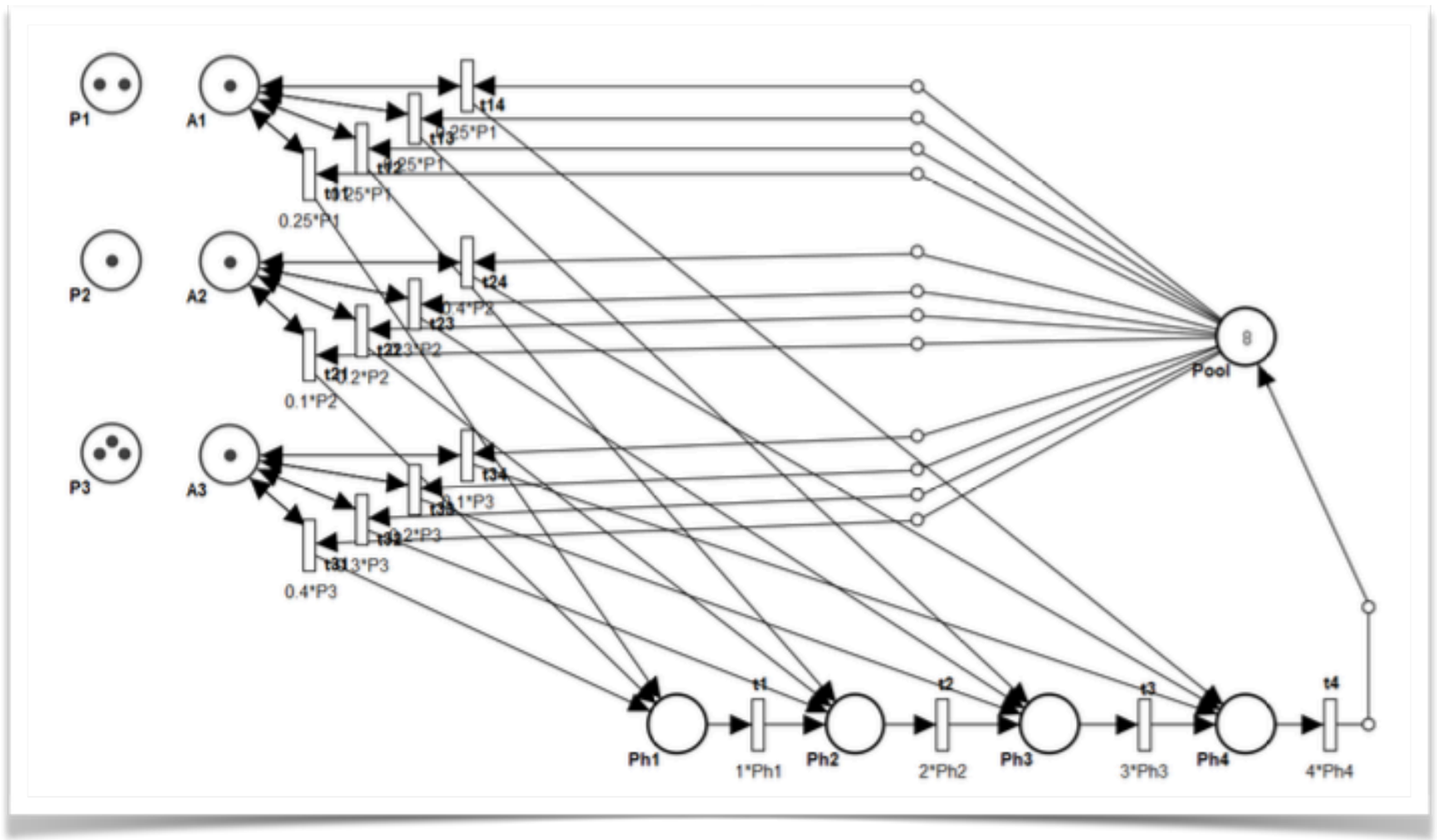
- By using **Bernstein polynomials** to **weight and combine multiple exponential distributions**, allows precise representation of complex multi-stage systems, where the total process completion time depends on the time spent in each phase.

The model

- The system has 3 input services defined as P1, P2, P3. Each of which has its own Poisson arrival rate.
- For each service, entry into the BPD, is determined by exponential transitions with different weights according to the service:
  - **P1**: same probability enters one of the stages.
  - **P2**: with higher probability enters the last phase of the BPD. It is the **fastest** service.
  - **P3**: with higher probability enters the first stage of BPD. It is the **slowest** service.

- **Pool**: place that presents tokens that represent the containers available to satisfy services.
- If the Pool does not present tokens, the firing of transitions to enter the BPD does **not occur**, resulting in rejection of the service request

The model



- Bernstein Phase consisting of **four phases**, tokens move from one phase to another via transitions with firing weights based on the transit of phases they reference.
- Upon exiting the BPD, the tokens **return** to the Pool, again available to be used in a new service request.
- Shown in the figure is the default network configuration, which will be taken as the base case for the analysis that will be performed

Goals of the model

The main goal of the system is to meet service demands, and to do so, the analysis on the system must go through the following points:

- **Sensitivity analysis**: how the system behaves in response to variation in arrival rates and poolsize. Fundamental analysis to understand the robustness, stability and performance of the system under different conditions.

- **Elasticity analysis**: starting from a steady state condition, assessments of the settling time with which the system responds to a perturbation(e.g., failure of a container, spike in a load).

- **Analysis on failures**: the ability of the system to maintain performance if there are failures that reduce the number of containers available to meet requests.

- **Pool size optimization**: finding a sizing of the number of pool replicas that leads to an optimal operating point in terms of deployment cost and performance.

Development

- The model was created on **Oris** and a program was then developed in Java that through the **Sirio** library allowed the model built and engineered to be recreated so that analysis could be performed in different configurations.

- Specific **rewards** were defined that can provide information on container availability and token distribution at various stages of BPD that are useful for understanding whether or not the system for a given configuration can satisfy incoming requests.

- **Pool Copies Rewards**: *If(Pool==0,1,0); If(Pool==1,1,0); If(Pool==2,1,0); If(Pool==3,1,0); If(Pool==4,1,0); If(Pool==5,1,0);If(Pool==6,1,0); If(Pool==7,1,0); If(Pool==8,1,0)*, through the probability vector they return we have information about the condition of the Pool.

- **Tokens Distribution Rewards**: *Pool; Ph1; Ph2; Ph3; Ph4;* provide information about the average availability of tokens in the system places. Useful for understanding the average availability of the Pool and the distribution of processes in the BPD

Development

- To observe the behavior of the system under various model configurations and parameterizations, **steady state** rewards were calculated and then through **transient analysis** of these it was possible to observe how the system evolved over time.

- In addition to analyses with predefined configurations, analyses have been carried out to observe how the system behaves if one or more **perturbations** to the system occur, such as a sudden spike in load or failure of one or more containers.

- To perform this type of analysis, a solution was developed to define a new network (**Steady Net**) that goes to recreate the steady state network condition and on this go to apply a perturbation, changing one or more parameters.

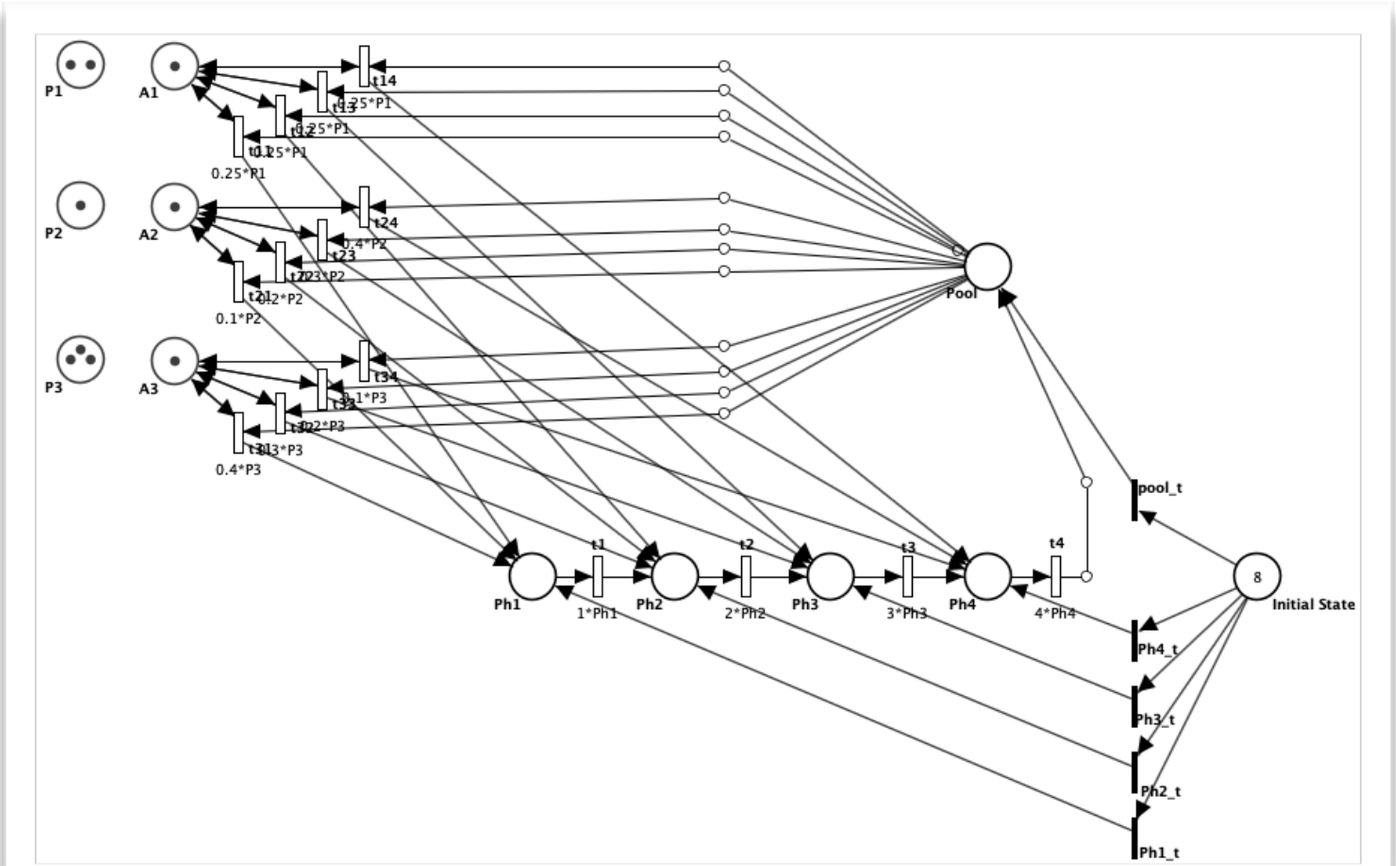
- By performing transient analysis on the new net, we see how the system responds to the perturbation. It is considered as the initial condition, the steady state condition because the perturbation is assumed to occur when the system is at steady state

Steady Net Building

- The new network is created with the same components as the original one, adding a Place, named **Initial State**, into which all tokens that in the initial state would reside in the Pool are moved.

- From this are defined **Immediate transitions** to all places in the BPD phases and to the Pool. These transitions will allow tokens to be distributed within the network.

- The firing of the new transitions will follow weights defined by the probability vector returned by the **Tokens Distribution Rewards**(*Pool; Ph1; Ph2; Ph3; Ph4)* from the Steady State analysis of the original network. In this way, the conditions of the steady state network can be re created.



Pool size optimization

- Keeping container replicas active but **idle** can be costly. At the same time, an inadequate pool size can lead to a large amount of rejections. Finding an optimal working point is one of the focuses of this project.

- To conduct this analysis, the rejection rate R, defined as the average value of rejected requests per unit time, is calculated.

- For a specific service request having arrival rate λ_{arr}, **R** is calculated as follows:

- $R = P(Pool == 0) * \lambda_{arr}$

- Since the service requests follow a Poisson distribution, the arrival rates being independent can be summed obtaining $\boldsymbol{\lambda_{tot}}$.. Therefore:

- $\boldsymbol{R_{tot} = P(Pool == 0) * \lambda_{tot}}$

- By establishing a desired **rejection threshold** for the system, we evaluate whether, for a given configuration, it is necessary to increase the pool size or decrease it

Development with Sirio

Taking advantage of the **Sirio** library, it was possible to recreate the model made with Oris and perform the analysis dynamically. Two main classes have been developed.

Through a **Builder** class and its functions, it is possible to create any configuration of the designed network or generate a Steady Net.

For greater malleability of the arrival rates, places P1, P2, P3 were replaced by an array, so that non-integer arrival rates could also be covered.

In the **ParallelServer** class, functions have been developed for:

- Steady state and transient calculation
- Dynamic modification of model parameters
- Calculating the rejection rate and finding the best pool size for the model.

Through various Python scripts, graphs are then created to display the results obtained

Analysis

- As a first analysis, the **2P1 1P2 3P3 8Pool** configuration is considered.
- The first step is to calculate the **steady state rewards**
- Looking at the *Tokens Distributions rewards* we see that the load distributes fairly evenly across the phases with a maximum in Ph1, in line with the input rate configuration.
- The average value of tokens in the Pool is 2,428 and looking at the *Pool Copies Rewards* values as well, we see that the probability of Pool being empty is only the fourth largest value.
- This leads one to think that the configuration is good but the resources are slightly overestimated.

Rewards	Valore
If(Pool==0,1,0)	0.1558
If(Pool==1,1,0)	0.1888
If(Pool==2,1,0)	0.2003
If(Pool==3,1,0)	0.1821
If(Pool==4,1,0)	0.1379
If(Pool==5,1,0)	0.0836
If(Pool==6,1,0)	0.0380
If(Pool==7,1,0)	0.0115
If(Pool==8,1,0)	0.0017

Pool Copies Rewards

Rewards	Valore
Pool	2.428
Ph1	1.519
Ph2	1.4350
Ph3	1.3506
Ph4	1.2662

Tokens distributions rewards

2P1 1P2 3P3 8Pool transient analysis

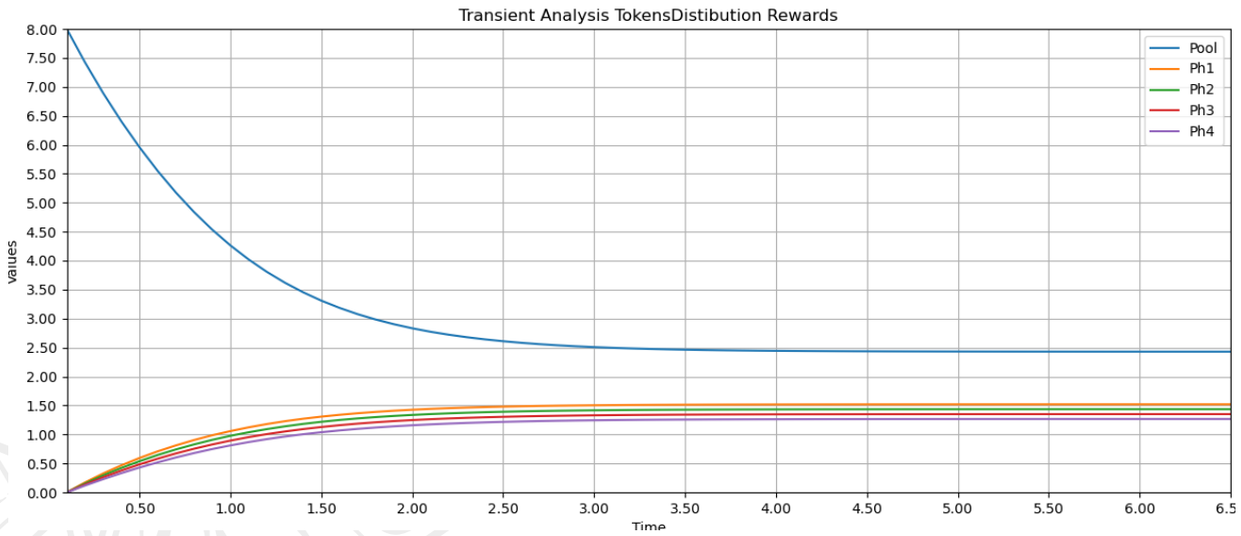
We continue the configuration analysis with **transient analysis**. Which allows the study of the behavior of the system before it reaches steady state.

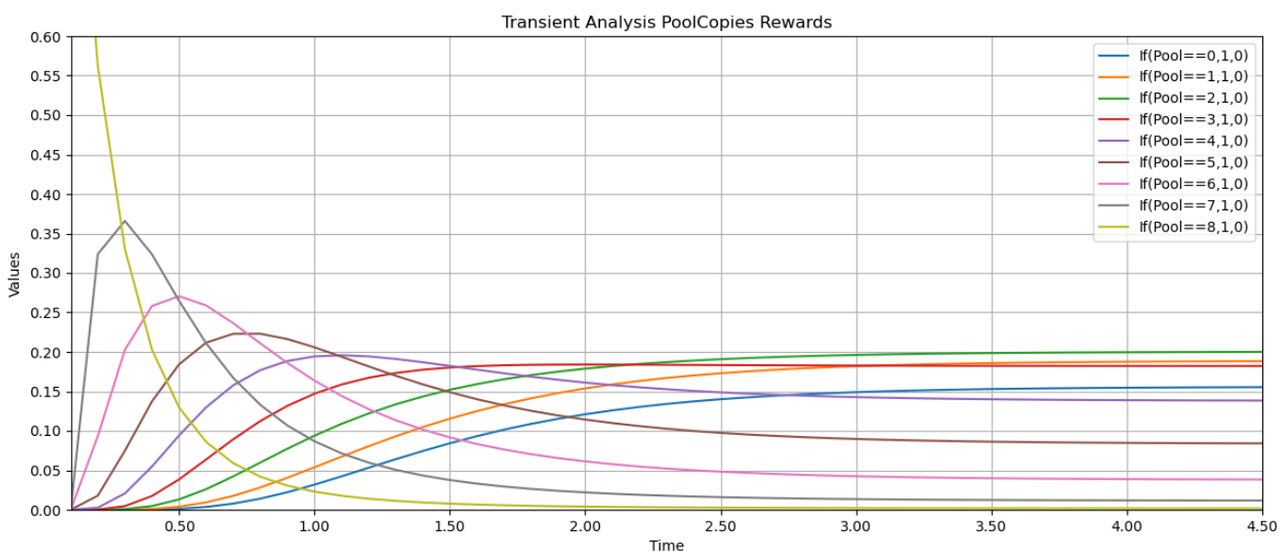
It considers time evolution and allows the system to be analyzed in non-stationary phases. This shows how and how soon the steady state is reached.

From the transient analyses in the figures, it is observed that the settling time is about 4 seconds.

A fairly rapid decrease in the number of replicas in the Pool is observed, due to the rate of arrival of requests, also considering that the most “time-consuming” requests are those with the highest rate.

It is observed that Ph1 always remains the phase with the most tokens and consequently the release of containers is quite slow

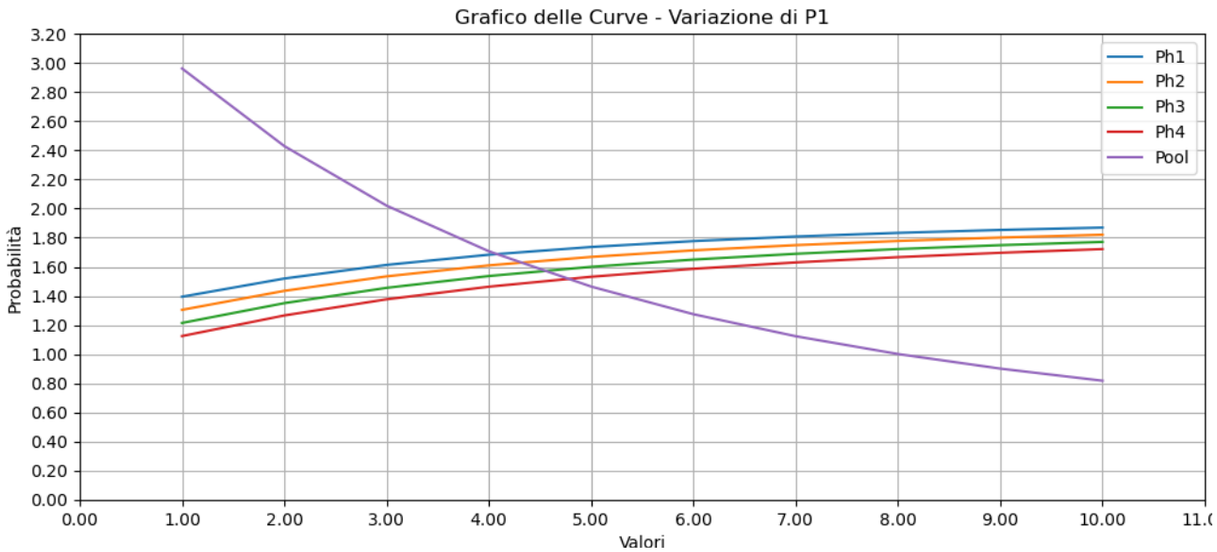


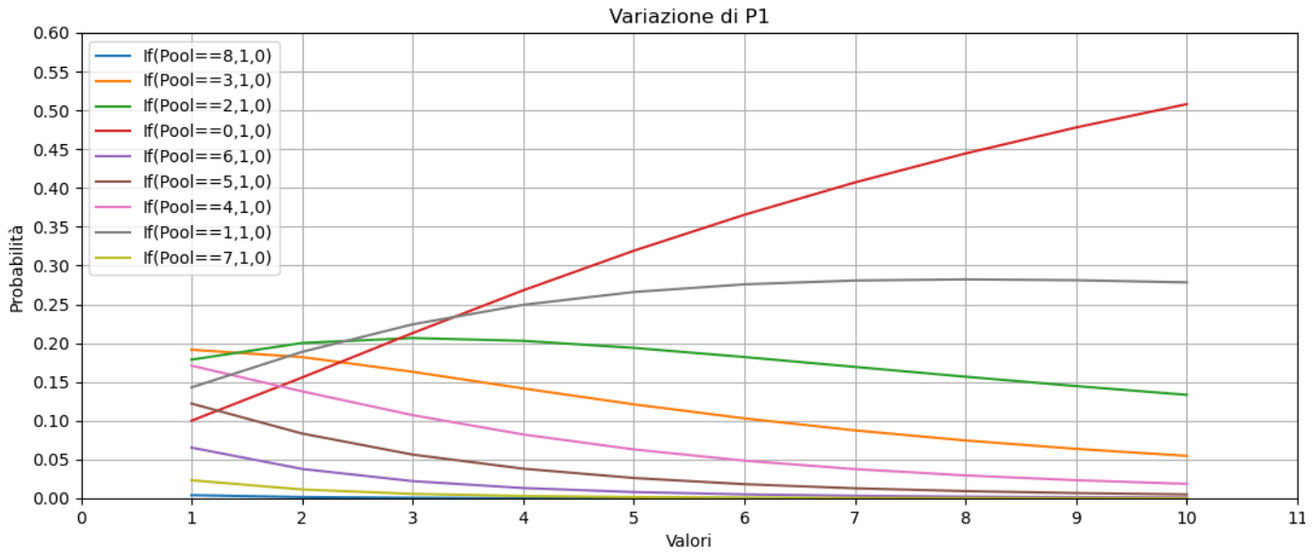


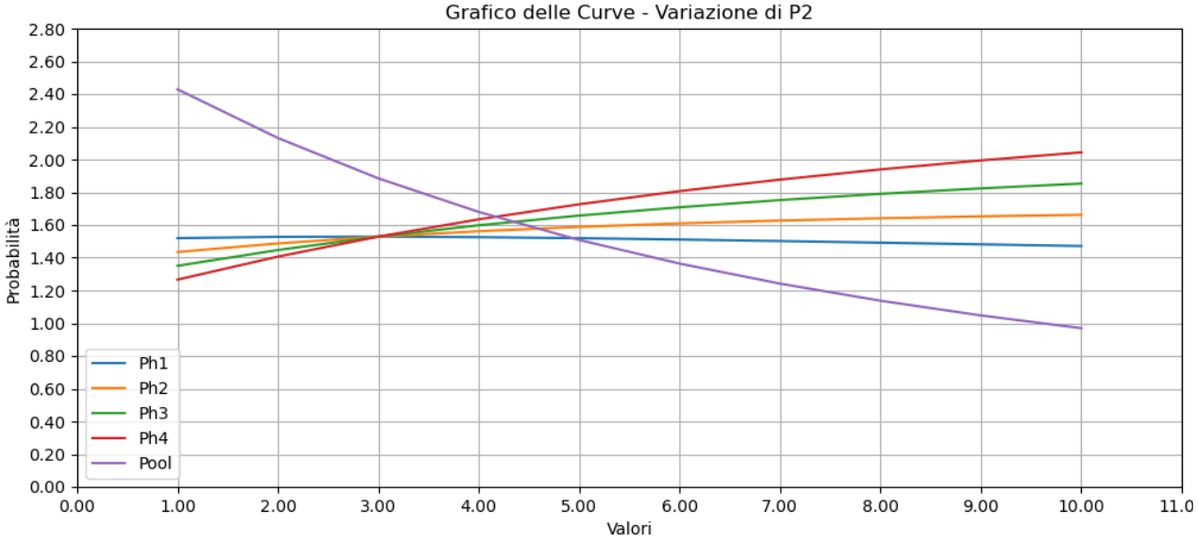
2P1 1P2 3P3 8Pool other analysis

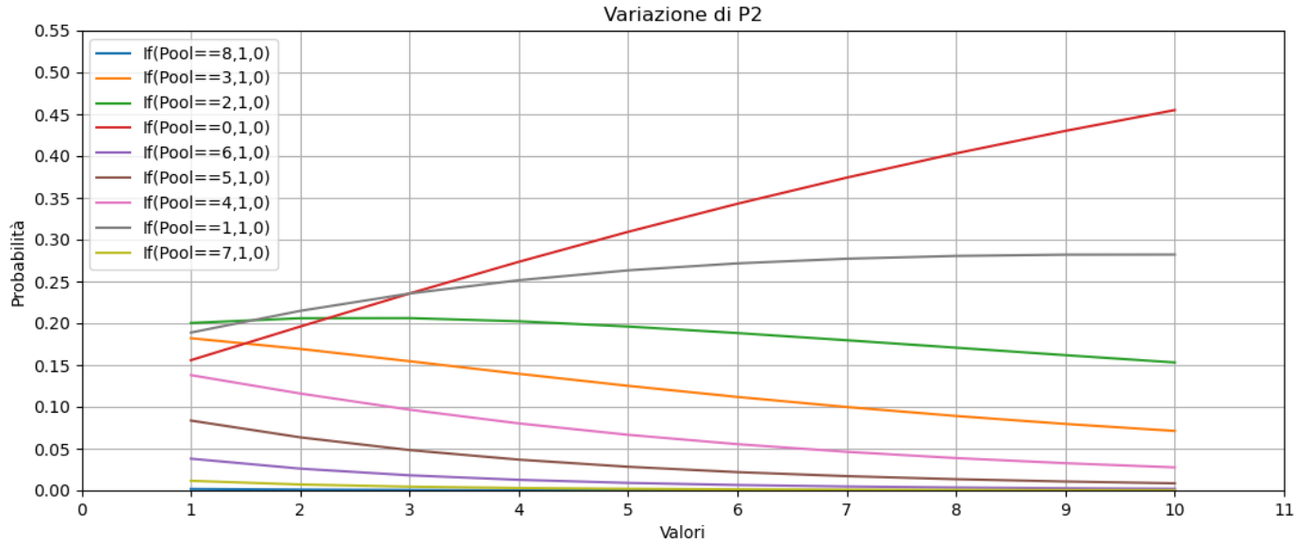
Let us now look at analyses in which steady-state rewards are calculated for **different values of a single parameter**. These analyses can be useful to get a general picture of the conditions in which the system may find itself if a single parameter has a different configuration

In particular, one can observe the different behavior of the system depending on the different arrival rate of a particular service. Remember that each service has a different service time.

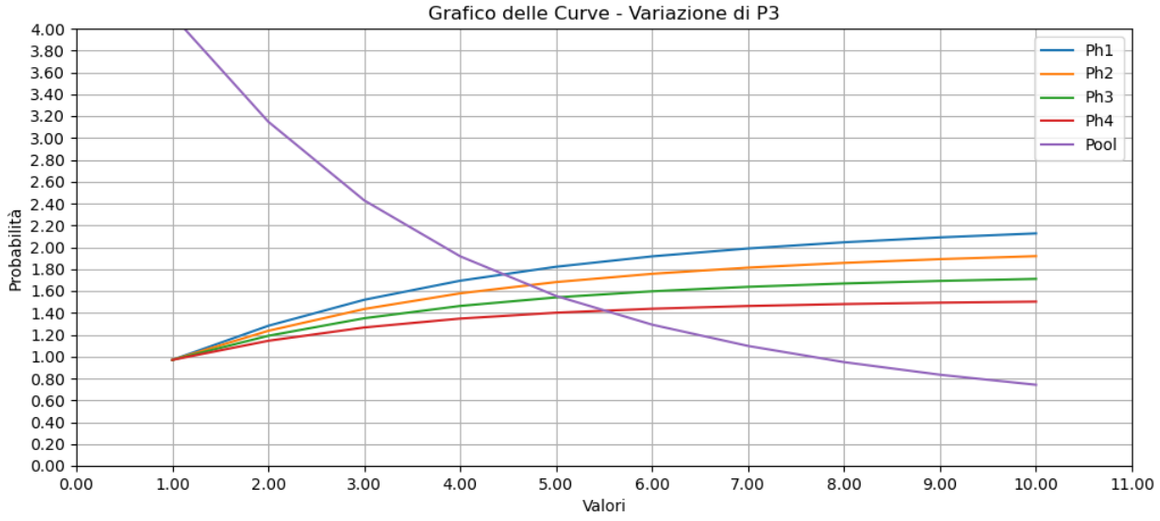


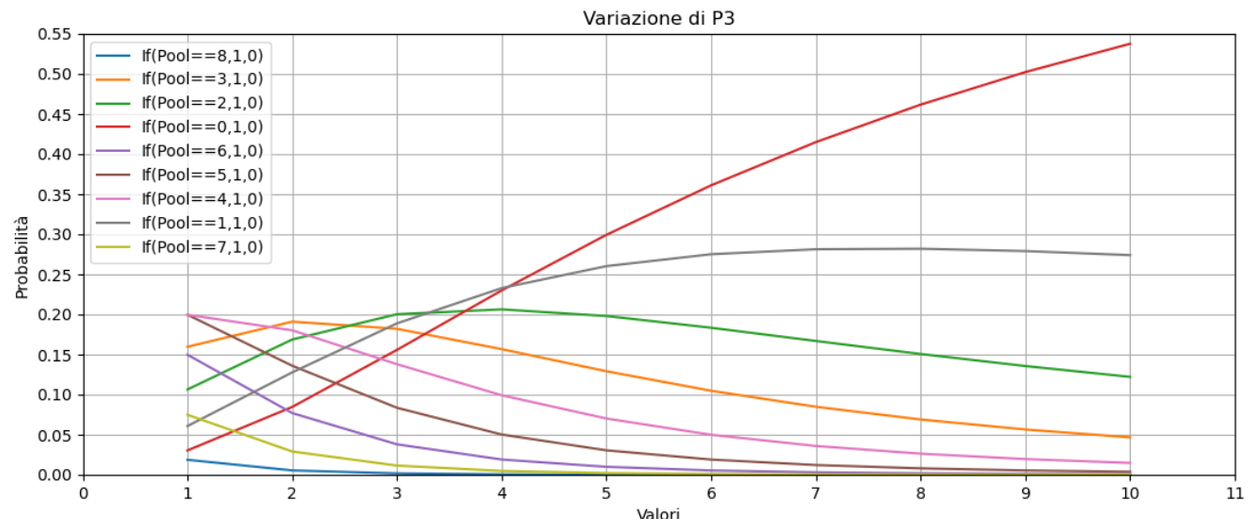


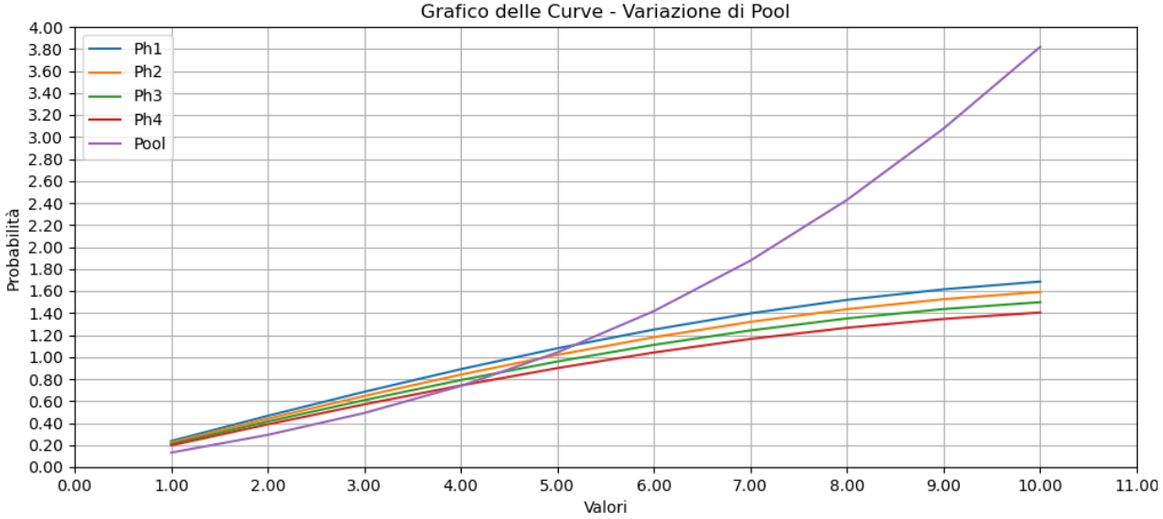


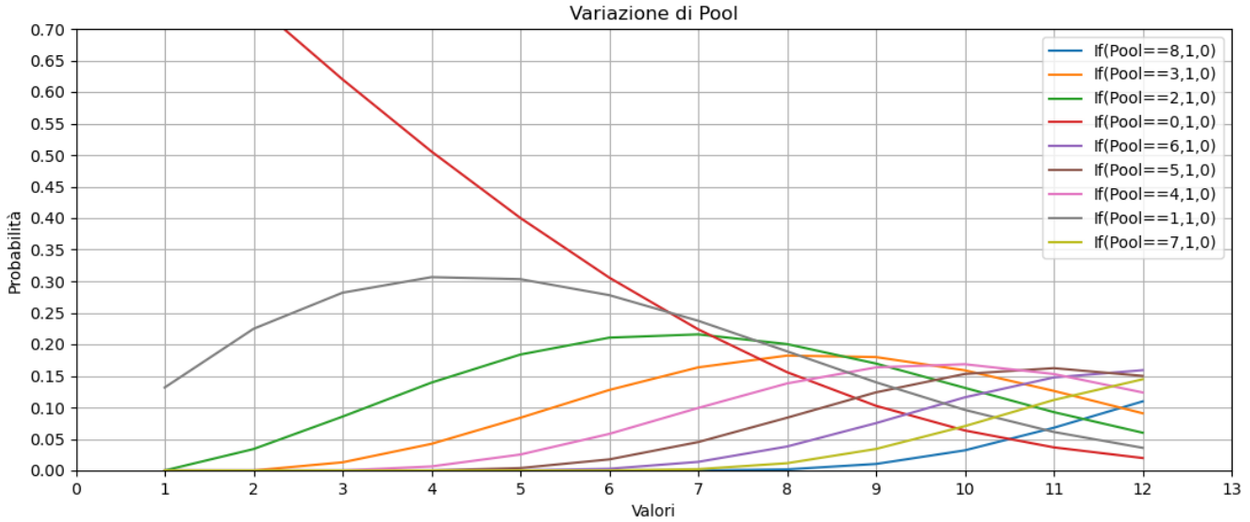


2P1 1P2 3P3 8Pool other analysis (2)









Such analyses are therefore useful for observing the margins a system has in terms of performance if there may be a change in arrival rates or pool size.

By looking at the graphs, one can study whether the system continues to maintain performance or goes under. They can also be useful in an optimization context.

Analysis for different configurations

Let us now consider the analysis using different configurations of the system: **2P1 1P2 3P3 10Pool**

Calculating the steady state rewards is obtained:

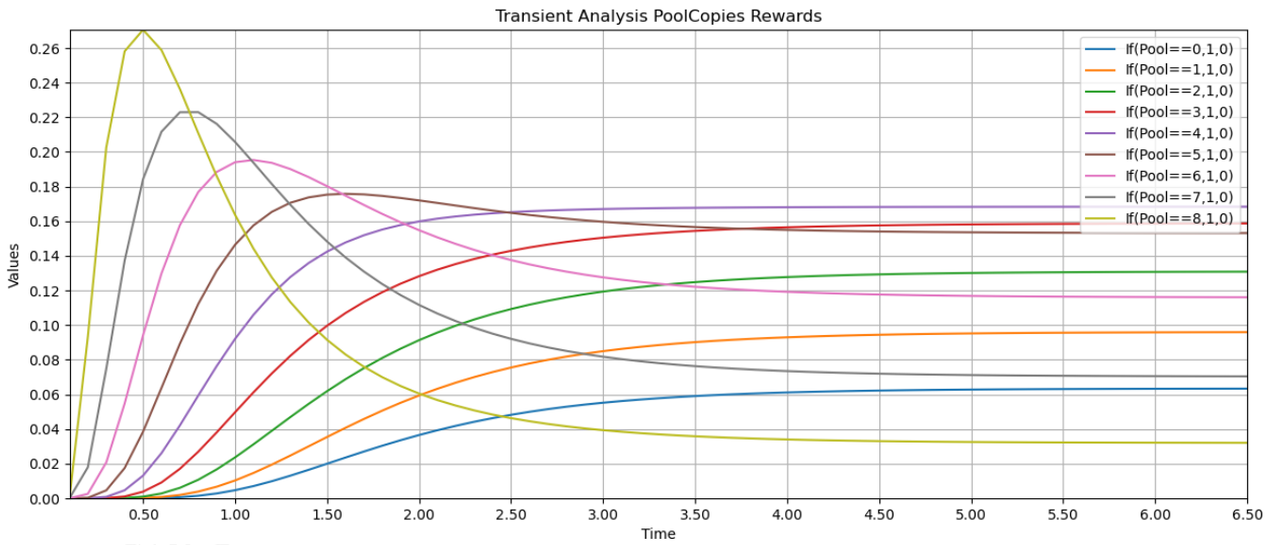
Pool Copies Rewards

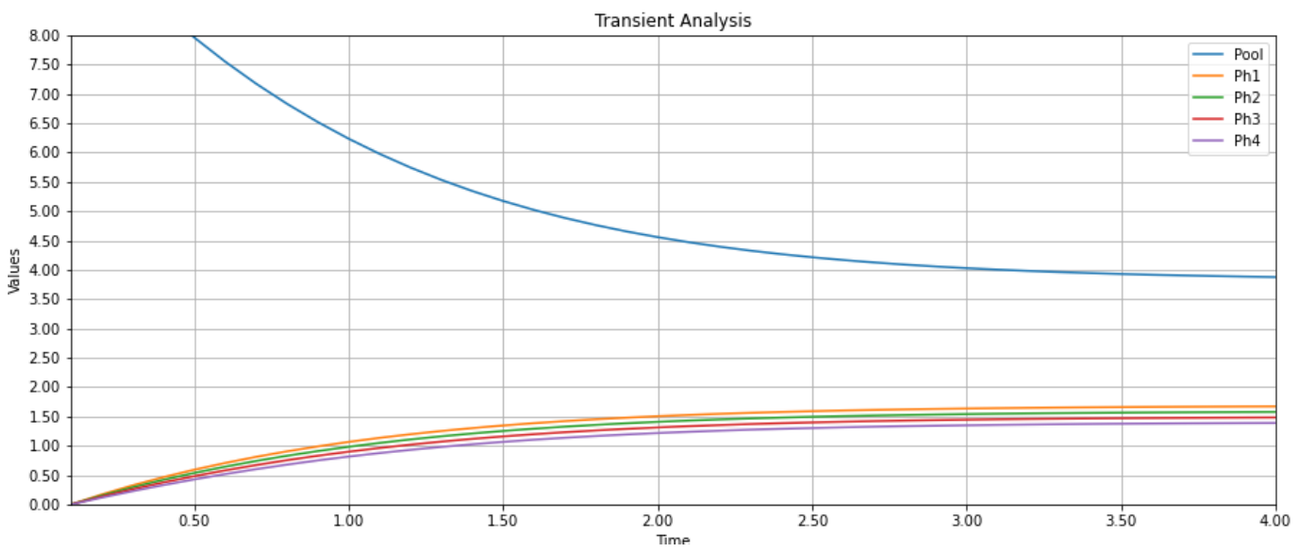
Rewards	Valore
If(Pool==0,1,0)	0.0633
If(Pool==1,1,0)	0.0960
If(Pool==2,1,0)	0.1309
If(Pool==3,1,0)	0.1587
If(Pool==4,1,0)	0.1683
If(Pool==5,1,0)	0.1530
If(Pool==6,1,0)	0.1159
If(Pool==7,1,0)	0.0702
If(Pool==8,1,0)	0.0319

Tokens Distribution Rewards

Rewards	Valore
Pool	3.983
Ph1	1.629
Ph2	1.5150
Ph3	1.5021
Ph4	1.4873

While the transient analysis is shown below:





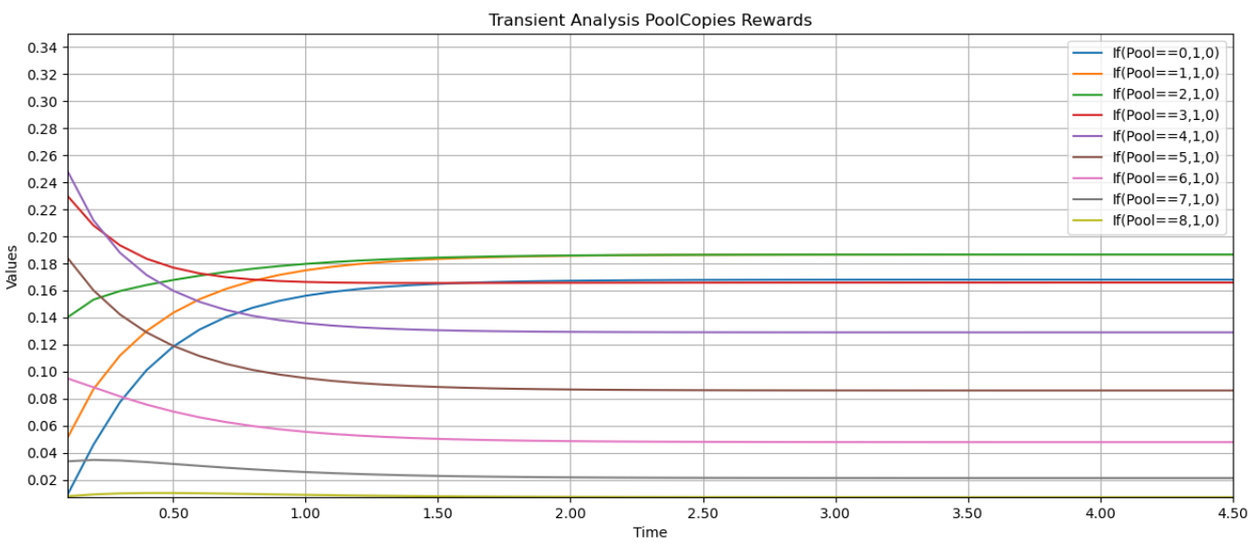
As can be seen, this configuration results in a significant waste of the resources reserved for the system since about 4 replicas on average remain idle and the probability that the Pool is empty is 0.063. We use this configuration as a starting point for another type of study on the system.

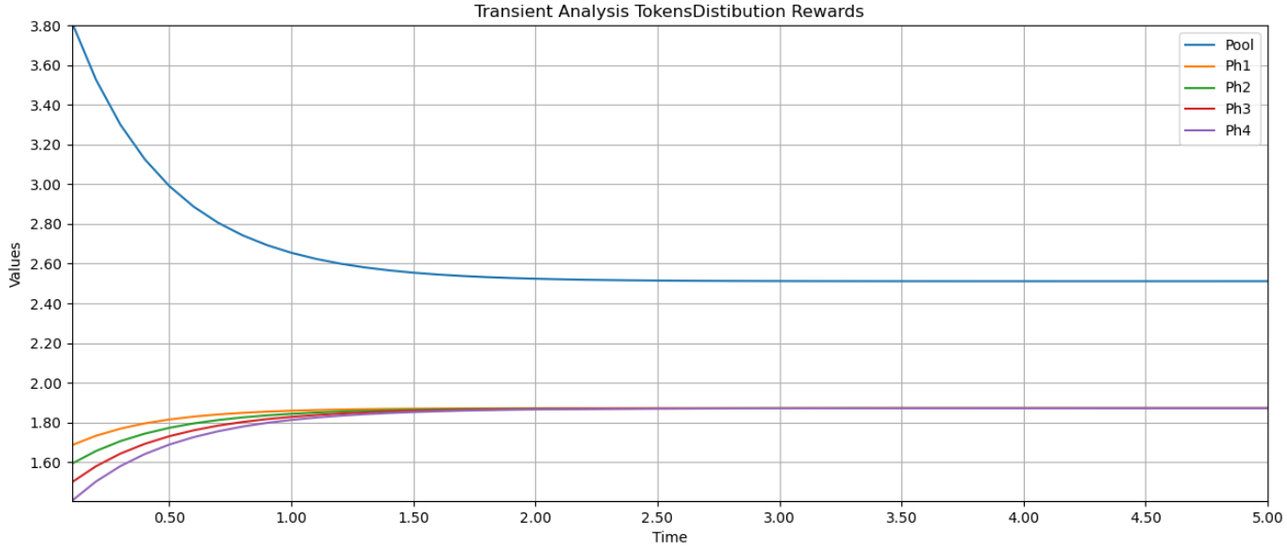
Perturbations on the system

Starting from the **2P1 1P2 3P3 10Pool** configuration that was found to be overestimated, we apply **perturbations** to the system to observe how it reacts, the settling time, and the new steady state it goes to once the transient is over

The first perturbation that is applied is an **increase in arrival rates**. We then consider an increase in the rates of P1 and P2 that brings the configuration into **3P1 3P2 3P3 10Pool**.

To study this perturbation, a transient analysis is performed from the steady state condition of the system in the initial configuration. Such a condition is obtained by constructing the **Steady Net**.





From the graphs, it is observed that the settling time is about 2.5s.

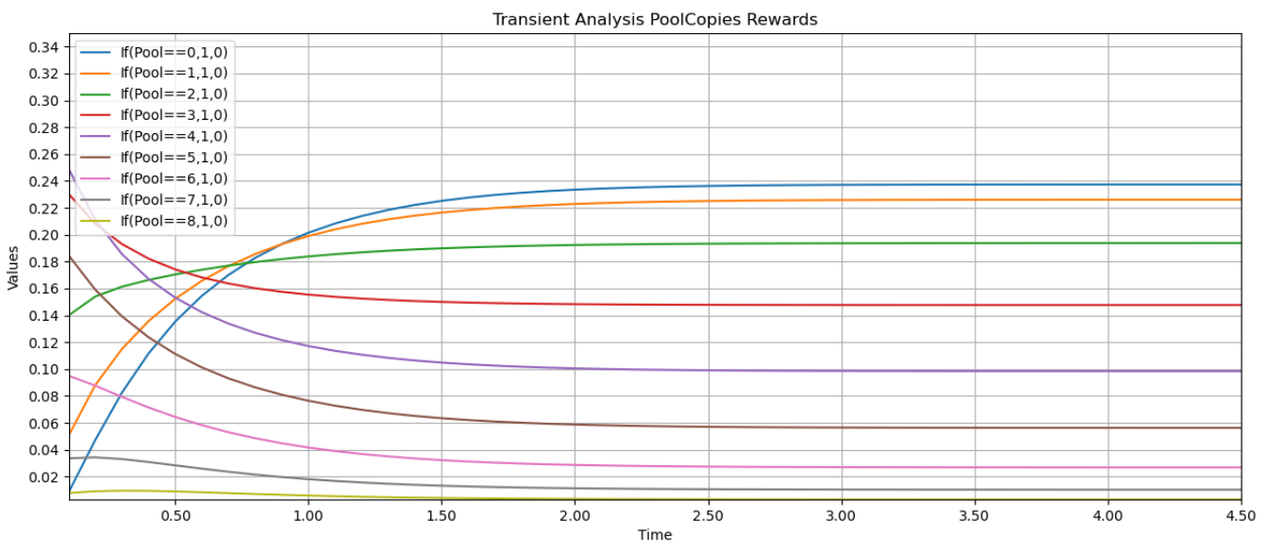
Since there is an increase in arrivals, but for less onerous services, the number of idle replications decreases even if only slightly. The probability of having the Pool empty turns out to be only the third highest.

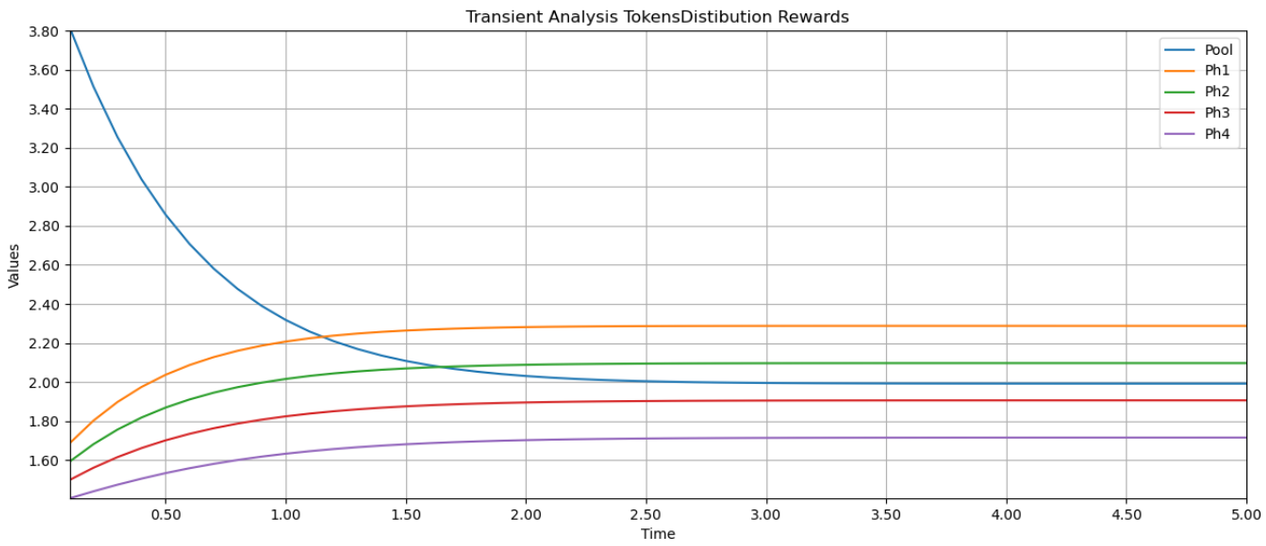
It can be concluded that the system has no problem handling this increase in arrival rates and still maintains a fairly high number of idle replicas.

Perturbations on the system(cont.)

Starting again from the **2P1 1P2 3P3 10Pool** configuration, we now apply other perturbations.

The second type of perturbation that is applied is a **spike** in arrival rates for the most onerous service, P3, which brings the configuration into **2P1 1P2 6P3 10Pool.**





The settling time in this case is longer, about 3s.

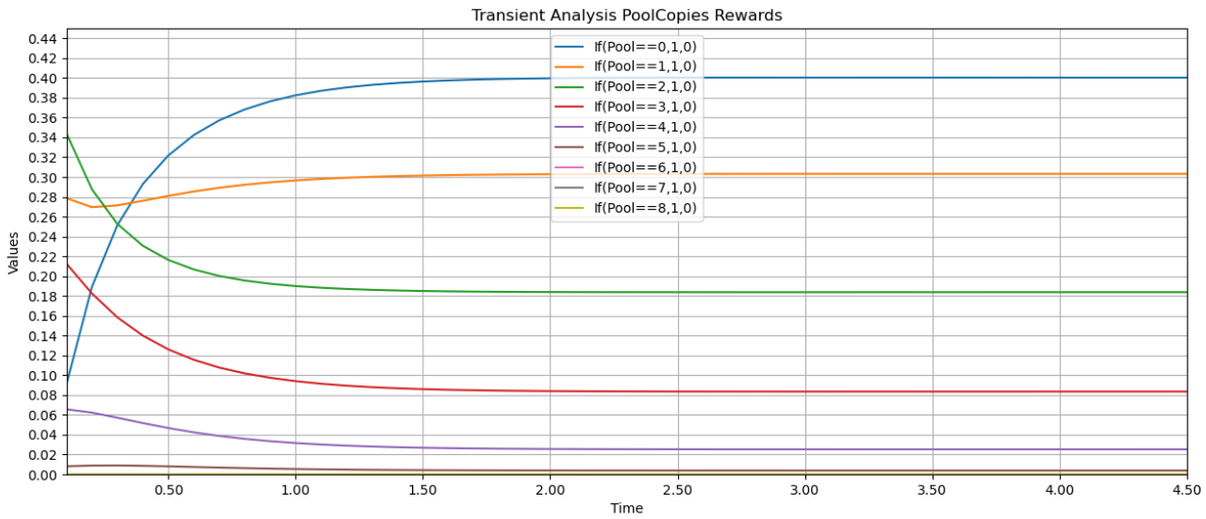
With this spike, the average amount of replicas in the Pool goes to half and the probability that the system has zero available becomes the maximum probability in Copies Rewards Pools. The release of replicas will be much slower; in fact, it can be seen that most are engaged with more onerous services.

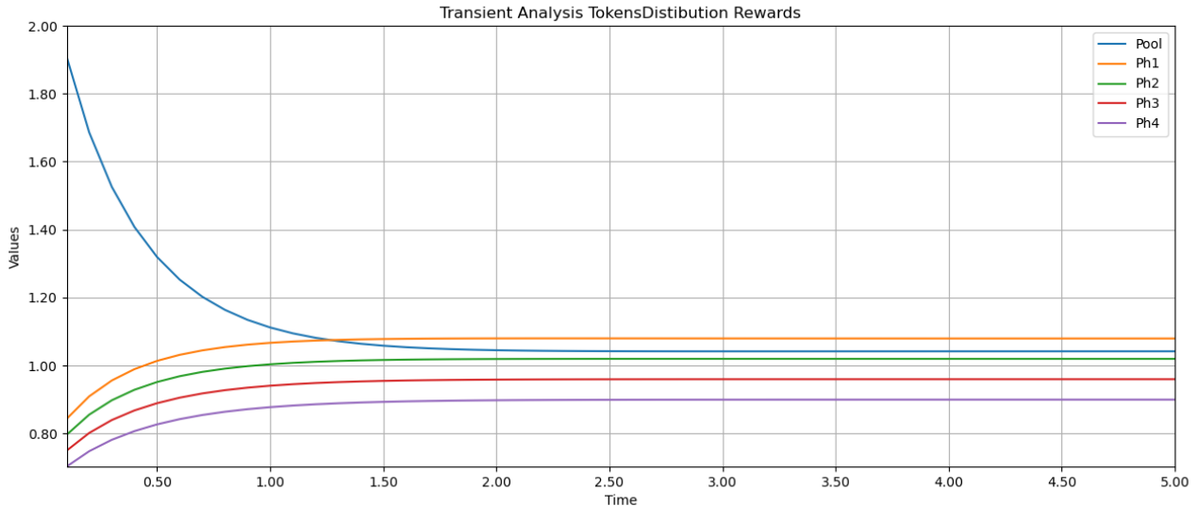
It can be concluded that the system with this perturbation will tend to reject more service requests..

Perturbations on the system(cont.)

Still starting from the **2P1 1P2 3P3 10Pool** configuration, let us apply a perturbation that simulates the failure of multiple containers.

We assume a severe **failure** to the system that makes half of the containers in the Pool unavailable, bringing the system into the **2P1 1P2 3P3 5Pool** configuration.





The settling time is about 3s.

Under this condition, the probability that the Pool is empty grows very rapidly, arriving at steady state to have a value of 0.40.

It can be concluded that the system with this perturbation is in distress and will tend to reject more service requests.

Pool size optimization

To perform this analysis, a case study was defined in which there are 3 configurations:

- **2P1 2P2 1P3 8Pool**: low onerous configuration
- **2P1 1P2 3P3 8Pool**: onerous configuration
- **3P1 1P2 5P3 8Pool**: very onerous configuration

Transitions from one configuration to another are performed, going to observe how the system reacts, how the rejection rate changes, and going to evaluate a possible optimal number of replicas in the Pool.

The **rejection rates** are calculated at steady state, and graphs of the rejection rate trend based on the Pool size value will be proposed for each configuration so that appropriate considerations can be drawn.

A maximum threshold is defined within which the rejection rate is considered acceptable, based on the arrival rate a maximum rejection rate of 1.00 is considered

From onerous configuration to very onerous

2P1 1P2 3P3 8Pool

Rewards	Valore
If(Pool==0,1,0)	0.1558
If(Pool==1,1,0)	0.1888
If(Pool==2,1,0)	0.2003
If(Pool==3,1,0)	0.1821
If(Pool==4,1,0)	0.1379
If(Pool==5,1,0)	0.0836
If(Pool==6,1,0)	0.0380
If(Pool==7,1,0)	0.0115
If(Pool==8,1,0)	0.0017

Rewards	Valore
Pool	2.428
Ph1	1.519
Ph2	1.4350
Ph3	1.3506
Ph4	1.2662

Tasso totale richieste	Tasso Rejection
6	0.9349

3P1 1P2 5P3 8Pool

Rewards	Valore
If(Pool==0,1,0)	0.3475
If(Pool==1,1,0)	0.2722
If(Pool==2,1,0)	0.1870
If(Pool==3,1,0)	0.1103
If(Pool==4,1,0)	0.0534
If(Pool==5,1,0)	0.0211
If(Pool==6,1,0)	0.0062
If(Pool==7,1,0)	0.0012
If(Pool==8,1,0)	0.0001

Rewards	Valore
Pool	1.3453
Ph1	1.8592
Ph2	1.7289
Ph3	1.5984
Ph4	1.4679

Tasso totale richieste	Tasso Rejection
9	3.1283

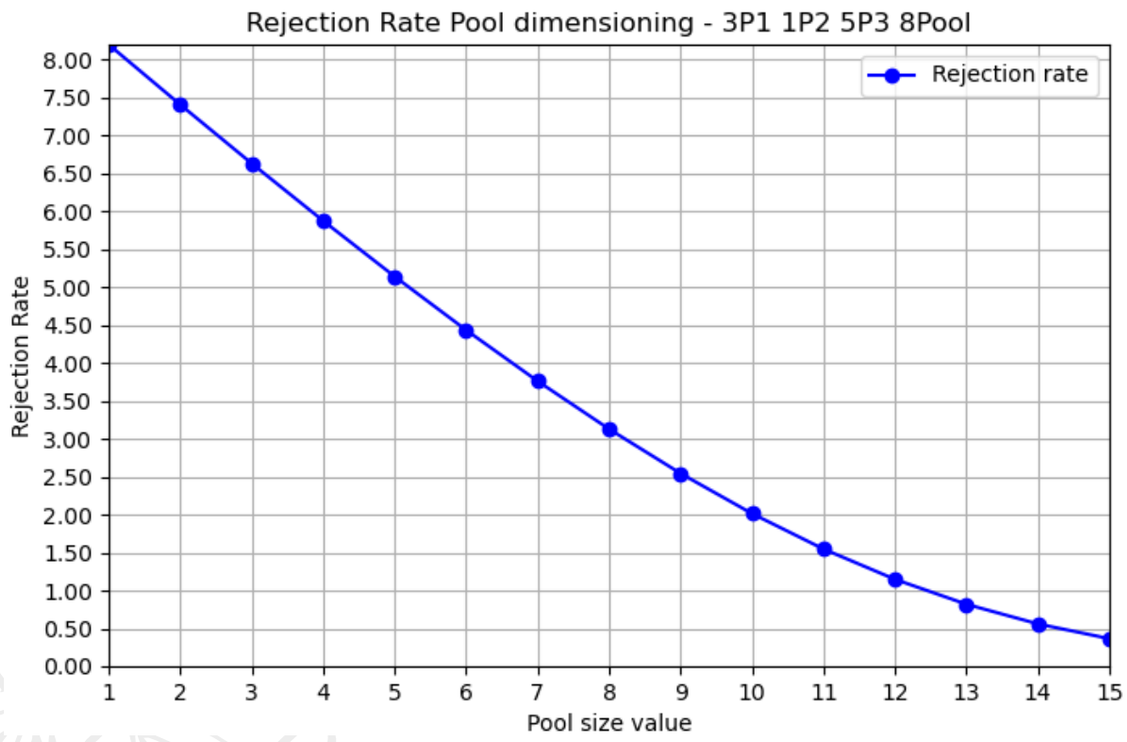
The change from onerous to very onerous configuration brings the rejection rate to 3.12, thus bringing the system into a critical condition in terms of rejected requests.

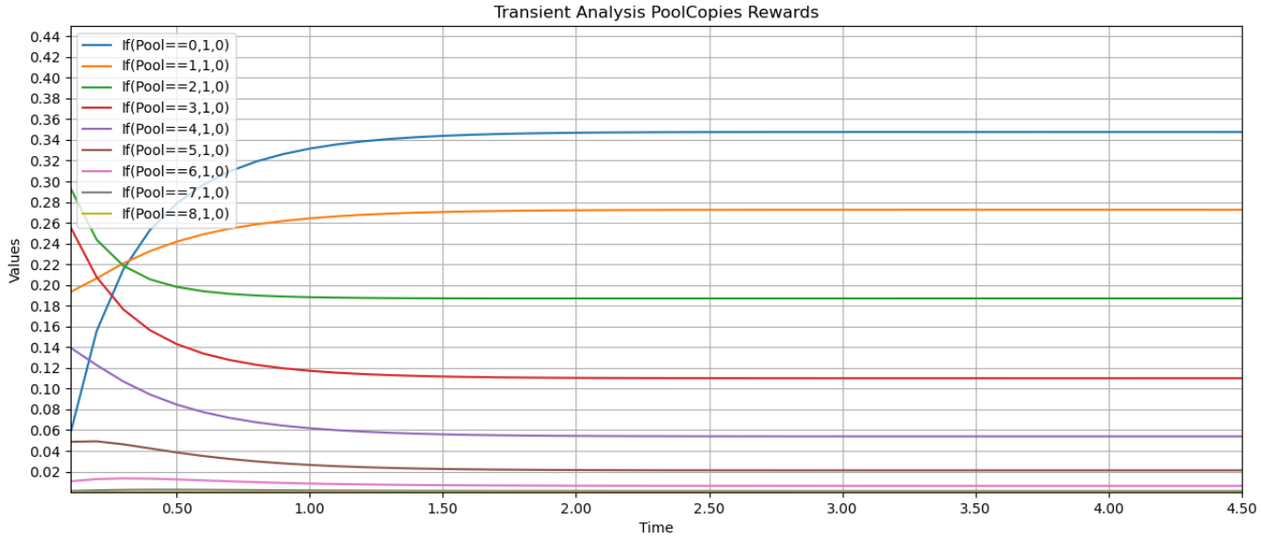
It is therefore necessary to go to optimize the number of replications to allow it to fall within the predefined rejection threshold

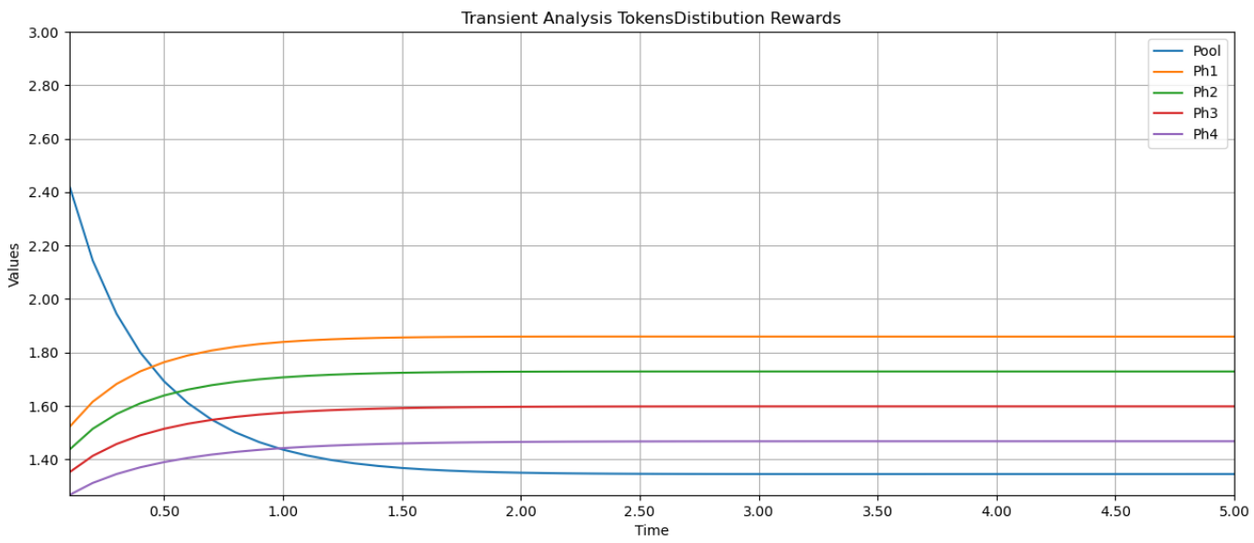
From onerous configuration to very onerous(cont.)

From the transient analysis of the configuration change, a settling time of about 2.5s is observed

From the Pool size sizing graph for 3P1 1P2 5P3 8Pool it can be seen that a Pool size value of 12 allows the system to stay around the rejection rate threshold. With a Pool size value of **13** in fact, the rejection rate would drop to **0.85.**



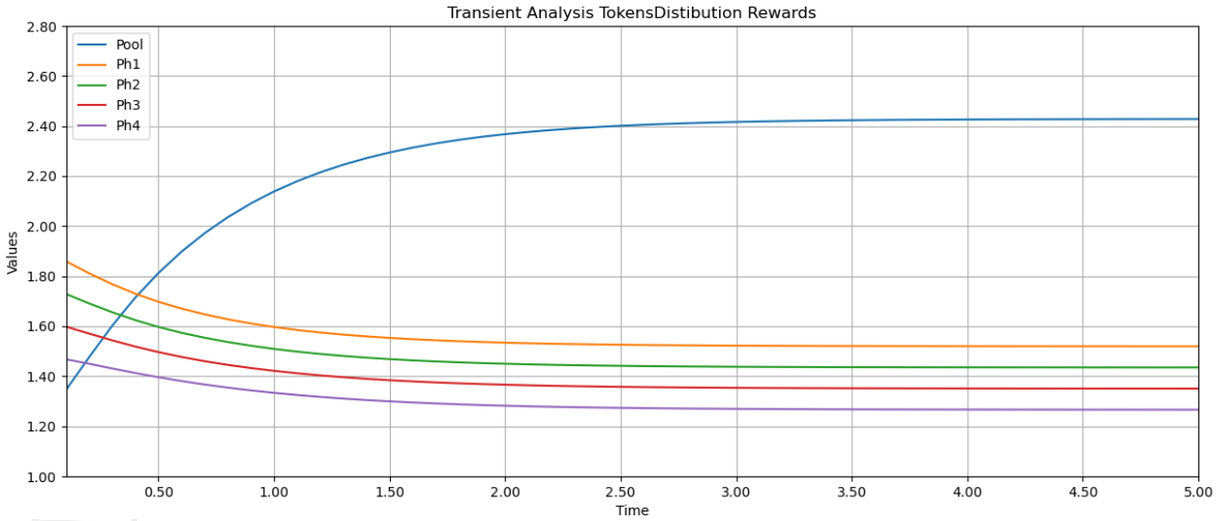


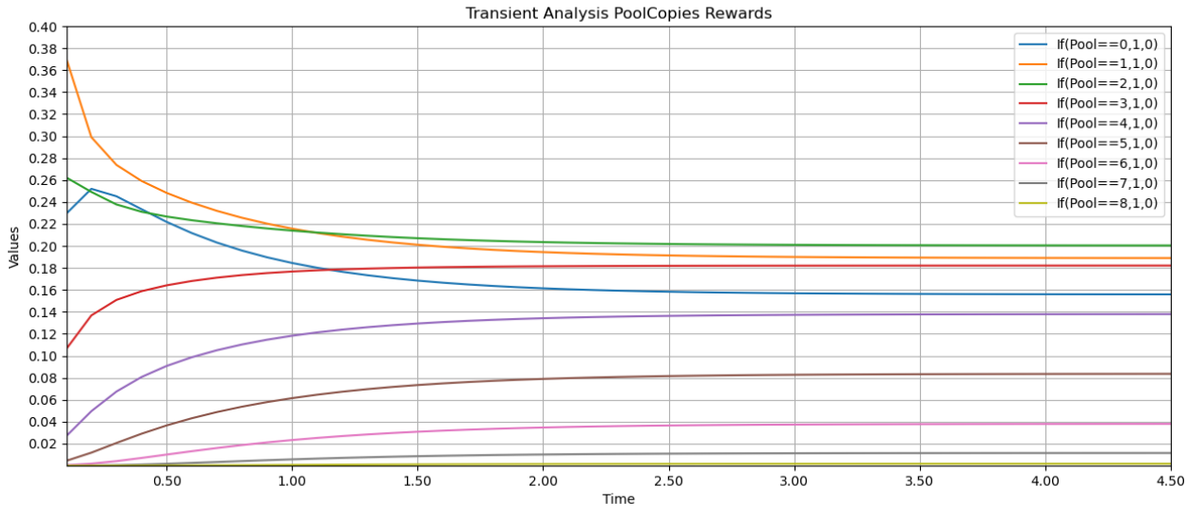


From very onerous configuration to onerous

Let us now analyze what happens when the peak load ends and we return to the average configuration.

In this case, the settling time is 4s. The probability of the Pool being empty slowly decreases over this interval. The load lightens but still remains onerous





From very onerous configuration to low onerous

2P1 2P2 1P3 8Pool

Rewards	Valore
If(Pool==0,1,0)	0.0218
If(Pool==1,1,0)	0.0472
If(Pool==2,1,0)	0.0893
If(Pool==3,1,0)	0.1448
If(Pool==4,1,0)	0.1957
If(Pool==5,1,0)	0.2116
If(Pool==6,1,0)	0.1715
If(Pool==7,1,0)	0.0927
If(Pool==8,1,0)	0.0250

Rewards	Valore
Pool	4.380
Ph1	0.8312
Ph2	0.8803
Ph3	0.929
Ph4	0.978

Tasso totale richieste	Tasso Rejection
5	0.2827

3P1 1P2 5P3 8Pool

Rewards	Valore
If(Pool==0,1,0)	0.3475
If(Pool==1,1,0)	0.2722
If(Pool==2,1,0)	0.1870
If(Pool==3,1,0)	0.1103
If(Pool==4,1,0)	0.0534
If(Pool==5,1,0)	0.0211
If(Pool==6,1,0)	0.0062
If(Pool==7,1,0)	0.0012
If(Pool==8,1,0)	0.0001

Rewards	Valore
Pool	1.3453
Ph1	1.8592
Ph2	1.7289
Ph3	1.5984
Ph4	1.4679

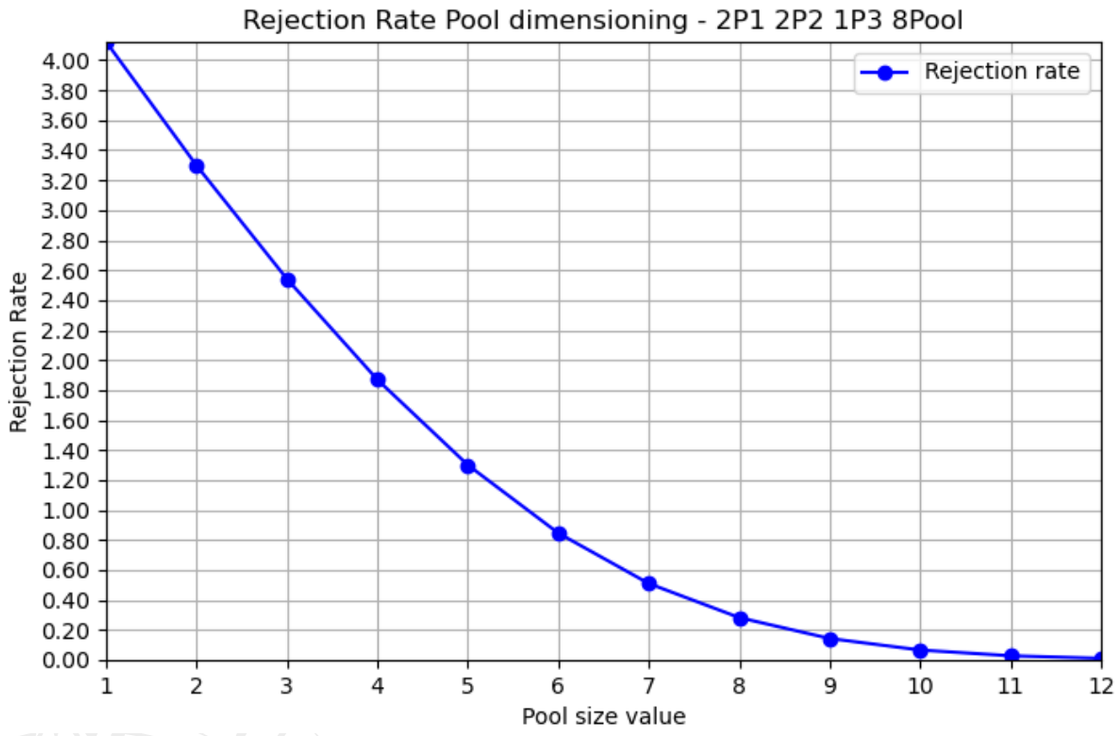
Tasso totale richieste	Tasso Rejection
9	3.1283

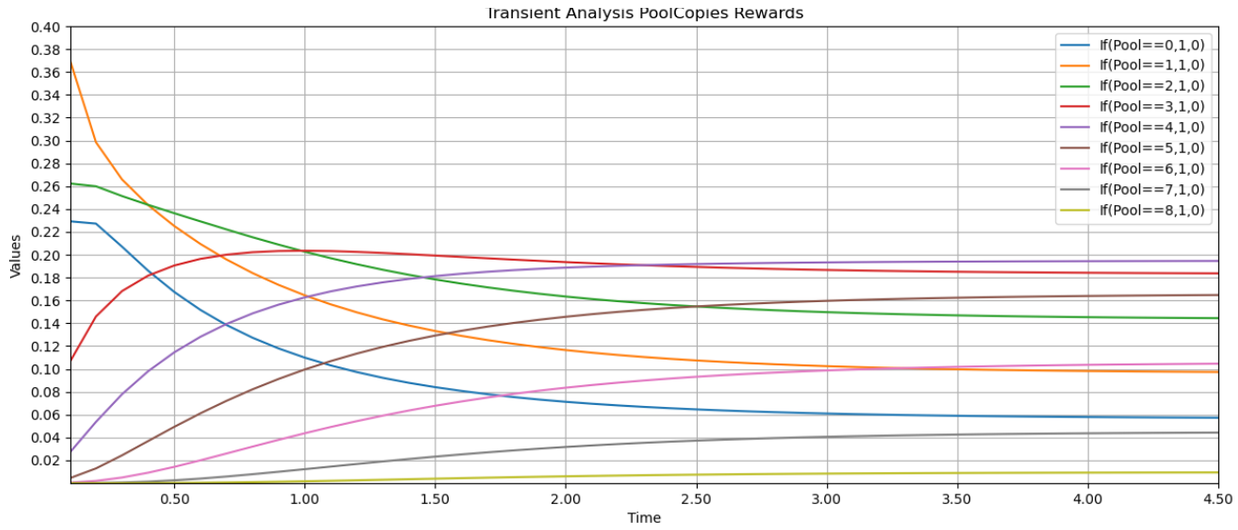
Suppose a sudden load collapse occurs at the time when the system is in the most onerous configuration.

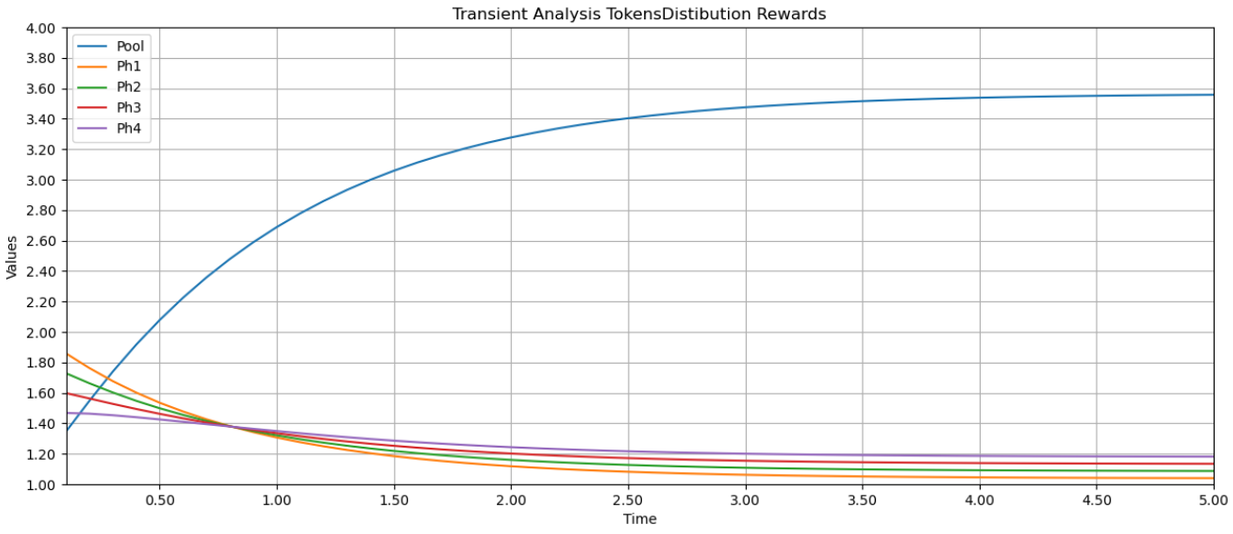
The rejection rate in this configuration is low, so we can think of going to reduce the pool size.

From very onerous configuration to low onerous (cont.)

A settling time of about 5s is observed from the transient analysis. The load change is much sharper than in the intermediate case.
From the Pool size sizing plot for 2P1 2P2 1P3 8Pool, it can be seen that reducing the Pool size to 6 keeps the system within the rejection rate threshold. With a Pool size value of **6** in fact, the rejection rate would drop to **0.81.**







Conclusions

- In this presentation, some of the analyses addressed in this study were presented.
- Using Oris Tool and the Sirio library, it was possible to approach the studies in two complementary ways.
- The program developed in Java, through model engineering allows analyses with customizable parameters to be able to study the model in steady state conditions and during transients for specific configurations or in case perturbations occur.
- Finally, through the study of rejection rates, it is possible to study pool size adaptation techniques.