UNIVERSITÀ DI PISA

Master Degree in Computer Engineering

Performance Evaluation of Computer Systems and Networks
Project Report

Punch'em Videogame

Team Members:

Luca Tartaglia
Kamran Mehravar
Farzaneh Moghani

Academic Year 2021/2022

Table of Contents

1 Introduction **3**
- 1.1 Problem Description . . . 3
- 1.2 Objectives . . . 3
- 1.3 Performance Indexes . . . 3

2 Modeling **4**
- 2.1 Model Description . . . 4
  - 2.1.1 General Description . . . 4
  - 2.1.2 Opponents (Bosses and Minions) . . . 4
  - 2.1.3 The Player . . . 4
  - 2.1.4 Recovering of a Minion . . . 4
- 2.2 General Assumptions . . . 5
- 2.3 Preliminar Validation . . . 5
- 2.4 Factors . . . 5

3 Implementation **6**
- 3.1 Modules . . . 6
- 3.2 Messages . . . 6
- 3.3 Modules Behaviour . . . 6

4 Verification **8**
- 4.1 Deterministic and Simple Cases . . . 8
- 4.2 Degeneracy Test . . . 8
- 4.3 Consistency Test . . . 8
  - 4.3.1 Consistency Test on Throughput . . . 8
  - 4.3.2 Consistency Test on Mean Waiting Time E[W] . . . 9
- 4.4 Continuity Test . . . 10
  - 4.4.1 Continuity Test on Mean Service Time $\frac{1}{\mu}$ . . . 11
  - 4.4.2 Monotonicity Test on Recover Rate x . . . 11
- 4.5 Verification Against the Theoretical Model . . . 12

5 Factors Calibration **15**
- 5.1 Scenario Calibration . . . 15
  - 5.1.1 Bosses Factors Calibration . . . 15
  - 5.1.2 Minions Factors Calibration . . . 15
  - 5.1.3 Recover Rate Factor Calibration . . . 16
- 5.2 Calibration of Warm-Up Time . . . 17
- 5.3 Calibration of Simulation Time . . . 17

6 Simulation Experiments **18**
- 6.1 $2^k r$ Factorial Analysis . . . 18
  - 6.1.1 $2^k r$ Factorial Analysis on Throuhgput T (Easy Game Mode) . . . 19
  - 6.1.2 $2^k r$ Factorial Analysis on Mean Waiting Time E[W] (Hard Game Mode) . . . 20
- 6.2 Easy Game Mode Simulation Experiments . . . 23
  - 6.2.1 Simulation Experiments Results on Throughput T . . . 23
  - 6.2.2 Simulation Experiments Results on Mean Waiting Time E[W] . . . 24
- 6.3 Medium Game Mode Simulation Experiments . . . 25
  - 6.3.1 Simulation Experiments Results on Throughput T . . . 25
  - 6.3.2 Simulation Experiments Results on Mean Waiting Time E[W] . . . 26
- 6.4 Hard Game Mode Simulation Experiments . . . 27
  - 6.4.1 Simulation Experiments Results on Throughput T . . . 27
  - 6.4.2 Simulation Experiments Results on Mean Waiting Time E[W] . . . 28

7 Conclusions **30**

1 Introduction

1.1 Problem Description

Consider a punch'em videogame where your avatar meets two type of enemies: minions and bosses.
Fighting either of them until defeat takes some time (see later on) and they arrive with some interarrival times. The rule of the game is such that you can fight only one opponent at a time. When you are fighting a minion and a boss arrives, you stop fighting the minion and fight the boss instead.

A fight lasts until the opponent is defeated. However, when you fight a boss, the minion that you were fighting earlier recovers some health, at a rate which is x% of the rate at which you deplete it when fighting it.

A defeated opponent leaves the game. Opponents of the same class queue up one after the other and are fought in a FIFO order.

1.2 Objectives

The objective for the project is to study the **number of minion and bosses fought in the unit of time as a function of the arrival rates** of both types of opponents. Moreover study the **queueing time of both types of opponents**.

More in detail, at least the following scenarios must be evaluated:

- Exponential distribution of the interarrival times for the opponents and fighting times.

1.3 Performance Indexes

In order to define a metric of performance of the objective, the following Performance Indexes are define:

- **Throughput T** :
  Number of opponents defeated per unit of time (defeated opponents over seconds). Are considered two different Throughput for each kind of opponent: T_m and T_b.
  The throughput is computed with the following formula:

  $$T = \frac{N}{s}$$

  Where **N** is the number of **total opponents defeated** and **s** is the **total simulation time**.

- **Mean Waiting Time E[W]**:
  Is defined as the mean time between the arrival of an opponent (minion or boss) and its starting fight time.

- **Mean Response Time E[R]**:
  Is defined as the mean time between the arrival of an opponent (minion or boss) and its defeat time.

2 Modeling

2.1 Model Description

In this chapter is given a general description of the system and its behavior.

2.1.1 General Description

The system is modeled to reproduce a single player punch'em videogame, where there are two types of opponents, *Minions* and *Bosses*, sent by their respective bases to the *Player* that has to fight against them and try to defeat as much opponents as possible during a game.
Moreover, boss type opponents have a priority on minions and when a boss arrives the fighting minion has to be queued up again recovering some life (the recovering system is better described below).

2.1.2 Opponents (Bosses and Minions)

The opponents, sent respectively from the modules Boss and Minion to the module Player, are considered as messages (new type defined, called OpponentMessage) that are placed in a FIFO queue (different queues are used for each opponent type). The new type of opponent message has a parameter attached, the **service time**, that is expressed in seconds and represent the opponents life.

2.1.3 The Player

A fight for the Player consist of receive an opponent message (start of the fight) and when the time corresponding to the latter's life (service time) runs out, the battle is declared finished and the opponent defeated. The player continues to fight against each opponent until the end of the game time (simulation time).

2.1.4 Recovering of a Minion

The module Player serves the opponents jobs in order of arrival and when a Boss message arrives check if there is a minion message under service, if so it stops the service and activate the **recovering systems**, which performs the following steps:

1. Takes off the minion under service.
2. Computes the value to be added (in seconds) to the current life of the minion under service.

   This is done by computing the **recovering percentage**, which is defined as the maximum percentage that can be applied to the minion's current life for the computation of the life to be recovered. The recovering percentage of a minion represents a portion of the percentage of life lost at that specific instant of time based on the minion's original life.

   More specifically, this portion is defined according to the **recovering rate "x"**, i.e. an "x" percentage of the recovering percentage to be extracted, which is subsequently applied to the current life and will return the amount of life to be recovered from the minion. The recover rate x is a parameter that is defined during configuration and can take values $0 \leq x \leq 100$.

   The computation done in this way ensure that the **recovered life** added to the current one never exceed the original one.

   In general, the steps to determine the life to be recovered are the following:

   2.1. Computation of the **life lost** and **its percentage** based on the original life.

   2.2. Computation of a **"x" percentage of the lost life**, where the value of x is defined in the configuration. The latter value computed is called **recovering percentage**.

   2.3. Application of the recovering percentage to the current life, so as to **calculate the life to be recovered** (value to be added to the current life).

3. Adds to the opponent's current life the value computed in the step 2.

4. Queue up the recovered minion again.

Here there is **an example** to better understand the recovering system functioning:

1. A minion arrives with a life equal to 2 seconds.
2. A boss arrives after 0.5 seconds the beginning of the minion fight (current minion life is equal to 1.5 seconds and the lost life is 25% of the original life).
3. the recovering percentage can be at most 25% of the current life.
4. Suppose we choose in the configuration file the recover rate x = 50. It means that we want the minion recover half percentage of the lost life percentage. So the recovering percentage will be 12.5%(half of 25%) of the current life.
5. The life to be added is seconds is 0.1875s (12,5% of 1.5s)
6. The new minion life will be 1.6875s (the sum of the original life and the life to be recovered) and then the minion is queued up again, waiting for his turn to fight another time.

2.2 General Assumptions

The following general assumptions have been made:

- Boss and Minion messages coincides with the jobs of the system.
- The service time of an opponent is defined as their life.
- The time between the arrival of two different opponents to the player (interarrival time) and their service time (life) are described by an exponential RV.
- Minion and Boss messages are assigned to two different FIFO queues of unlimited capacity.
- The recovering system cannot produce a value of the life to recover that exceed the original life of a minion, ensuring that all minions at some point will be defeated.

2.3 Preliminar Validation

Before the implementation description, a preliminar validation is necessary to ensure that the model is correct. In particular some assumption made in the previous section is analyzed:

- The **unlimited capacity of the queues** for minions and bosses is reasonable, because the game is designed to let the player fight against as many opponents as possible, so the number of opponent placed in the queue can be very large.
- **Having no minions with infinite life** is reasonable, because if a minion could survive indefinitely the system would not be anymore realistic.

2.4 Factors

The following factors have been defined, which may influence the performance of the system:

- $\frac{1}{\lambda}$: mean of interarrival time for minion and bosses distribution.
- $\frac{1}{\mu}$: mean service time for minion and bosses distribution.
- **Recover rate x** : percentage of the maximum value a minion can recover (can assume values: $0 \leq x \leq 100$).

3 Implementation

3.1 Modules

The following modules have been defined:

- **Boss**: Simple module which randomly generates opponents of type "Boss" with interarrival and service time (life) exponential distributed or constant.
- **Minion**: Simple module which randomly generates opponents of type "Minion" with interarrival and service time (life) exponential distributed or constant.
- **Player** : Simple module which simulates the behaviour of the player. This module is in charge of receive and handle the opponent messages, implement the FIFO queues, manage the recovering system and compute all statistics.

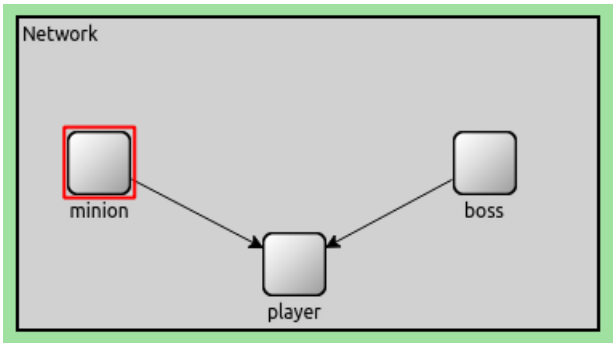


Figure 1: Punch'em System Model

3.2 Messages

A new messsage type has been defined: the **Opponent Message** type. It holds the service time for each opponent in **simtime_t** format. This new type of message is defined in order to hold the life for each opponent is been created.

3.3 Modules Behaviour

- **Boss and Minion** :
  This two modules perform the same operations for Boss and Minion message types respectively:

  1. **Generate a random number from the chosen interarrival distribution** (exponential distribution or constant interarrival time) and set a timer accordingly.
  2. When the timer goes off, **generate a random number from the selected service time distribution** (exponential distribution or constant service time).
  3. **Send a message to Player module** (Opponent Message type) along with its service time.

  The functions defined are:

  - `scheduleNextArrival()`
  - `generateNewOpponent()`

- **Player**:

  1. When it **receives a message from Minion or Boss modules**, pushes them in its own FIFO queue (minion or boss queue):
     1.1. If **receives a Minion Message** it checks if the minion and boss queues are empty and if so, it starts to process the minion message (the fight starts).
     1.2. If **receives a Boss Message** it checks if a minion message is under service and if so, it stops the process and then starts the recovering phase of the minion (better described below) and then process the boss message (the fight starts).
  2. When a **minion needs to be recovered**, it computes the life to recover (by calling the specific function), sum up the current life and the life to be recovered and finally push again the minion message in its queue.
  3. **To process a message (fight against an opponent)** the module reads its service time (life) and set a timer (two different timer are defined, one for each kind of opponent). When the timer goes off it extracts the opponent message from its queue and deletes it (the opponent is defeated).
  4. It also **records statistics** every time a message (opponent) is sent to it, when a message is deleted (opponent defeated), when a message is queued up again (minion recovered) and finally when the simulation ends, in order to **compute the most meaningful KPIs**.

  The functions defined are:

  - `handleMinion(), handleBoss()`
  - `recoverMinion(), computeLifeRecovered()`
  - `defeatOpponent()`

4 Verification

In this section have been carried out several tests in order to verify if the implementation reflects correctly the model.

4.1 Deterministic and Simple Cases

This kind of verification consists of performing tests with predicted (in this case constant) values for the interarrival and service time in order to study the behaviour for simple cases and check the correctness of the mathematical procedures (such as the computation of the minion recover or the time of the fights).
All tests are made by using graphical environments (Qtenv) and debugging prints.

- **Test the correctness of fights**:
  The arrival and the queueing of the minions and bosses is correct and the fights last as long as calculated.
- **Test the correctness of the minions recovery (0%, 50% and 100%)**:
  The recovery method correctly add the 0% (no recovery), the 50% and the 100% (3 different simulation scenarios) of the computed life to recover to the current minion life. Also the new queuing of the recovering minion is correct.

4.2 Degeneracy Test

In the Degeneracy Test has been verified the behaviour of the system with parameters set to 0 values. The system works properly and the following observations can be made:
In such cases the system allows either to play with only one opponent type (minions or bosses) or it ends the simulation if all parameters (boss and minion service/arrival times) are set to 0.

- **Minion interarrival and/or service time set to 0**:
  The simulation proceeds by only having Boss type opponents.
- **Boss interarrival and/or service time set to 0**:
  The simulation proceeds by only having Minion type opponents.
- **Both opponents interarrival and/or serivce time set to 0**:
  The simulation ends at the first event.

4.3 Consistency Test

In the consistency test has been verified if the system react consistently with the output. In particular, has been verified across the **Throughput T** and the **Mean Waiting Time E[W]** analysis results (bosses and minions tested separately).

In order to perform the tests on bosses and minions separately, the game mode with only one type of opponent were used.

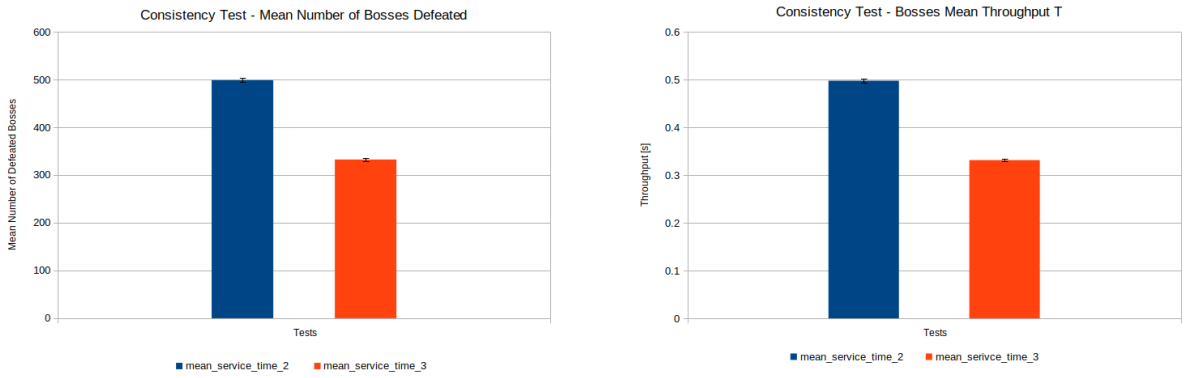
Following are shown only the tests made on the bosses but through the omnet configuration file and analysis charts is possible to see those made on minions, that produces similar results. The exponential distribution was used for all tests.

4.3.1 Consistency Test on Throughput

The mean arrival and service time values used are as follows:

Consistency Test on Throughput		
Test	**Mean Arrival Time**	**Mean Service Time**
Test 1	1s	2s
Test 2	1s	3s

The results can be seen in the following plots:



(a) Consistency Test on Bosses Defeated

(b) Consistency Test on Bosses Throughput

Figure 2: Consistency Test on Throughput T

The mean number of bosses defeated (Figure 2a) and the bosses throughput (Figure 2b) is greater in the test 1 and it is possible to conclude that, as expected, the throughput is higher when the service time decreases and vice-versa. In terms of studied model the player defeats more opponents if their life decreases.

4.3.2 Consistency Test on Mean Waiting Time E[W]

Tests were performed on the following values:

Consistency Test on Mean Waiting Time E[W]		
Test	**Mean Arrival Time**	**Mean Service Time**
Test 1	10 s	1 s
Test 2	10 s	5 s
Test 3	10 s	15 s

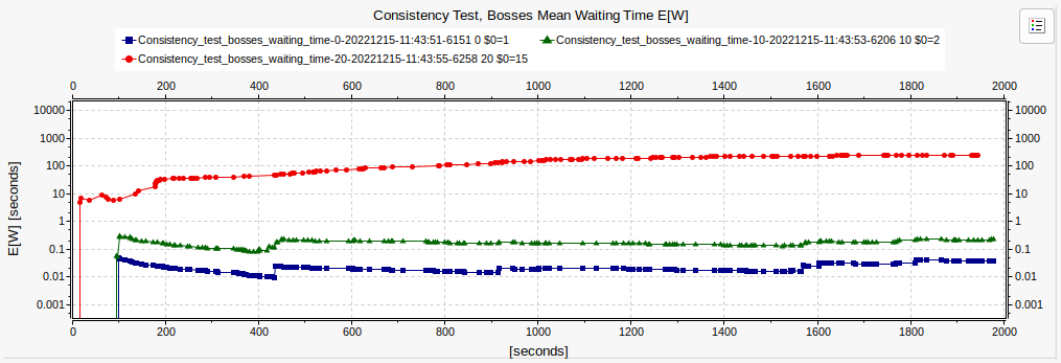


Figure 3: Omnet++ results sample on Boss Waiting Time

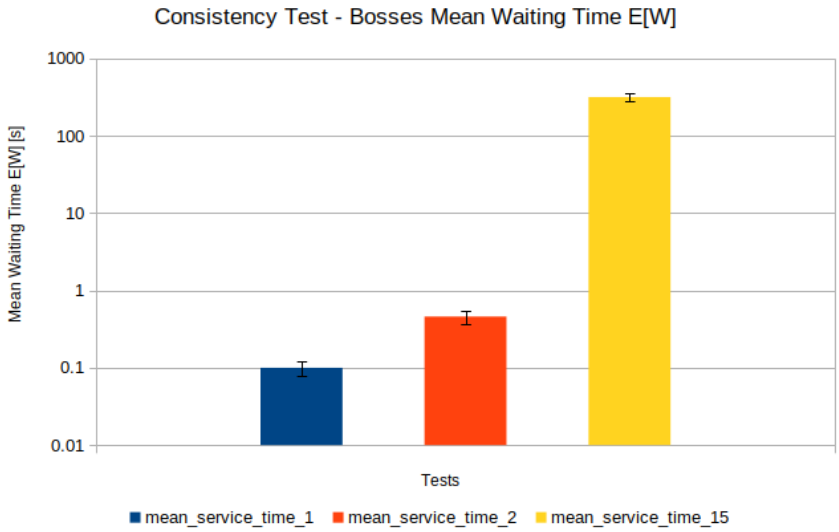


Figure 4: Consistency Test - Boss Mean Waiting Time results

As expected, when the service time is lower than the arrival time, the waiting time takes values around 0 (the player has enough time to fight and defeat the opponents before a new one arrives). On the contrary, if the service time is greater than the arrival time the mean waiting time increases considerably, until it reaches excessively high values that could impact on the overall performance of the system.

4.4 Continuity Test

The aim of the continuity test is to verify if changing slightly the input affect slightly the output.
Furthermore, it was also performed a **monotonicity test**. The latter consists in assess the monotonicity of some performance indexes using different combinations of factors.

To check the correctness of the system using these methods it has been evaluated the **Mean Response time E[R]** and the **Throughput T** by performing changes of the **Mean Service time** $\frac{1}{\mu}$ (for the continuity test) and the **Recover rate x** (for the monotonicity test).

4.4.1 Continuity Test on Mean Service Time $\frac{1}{\mu}$

This test was carried out using the following values for the factors of interest:

- **Bosses Mean Arrival time $\frac{1}{\lambda_b}$: 10 seconds**
- **Bosses Mean Service time values $\frac{1}{\mu_b}$: 5, 5.1, 5.2 seconds**
- **Repetitions: 50**

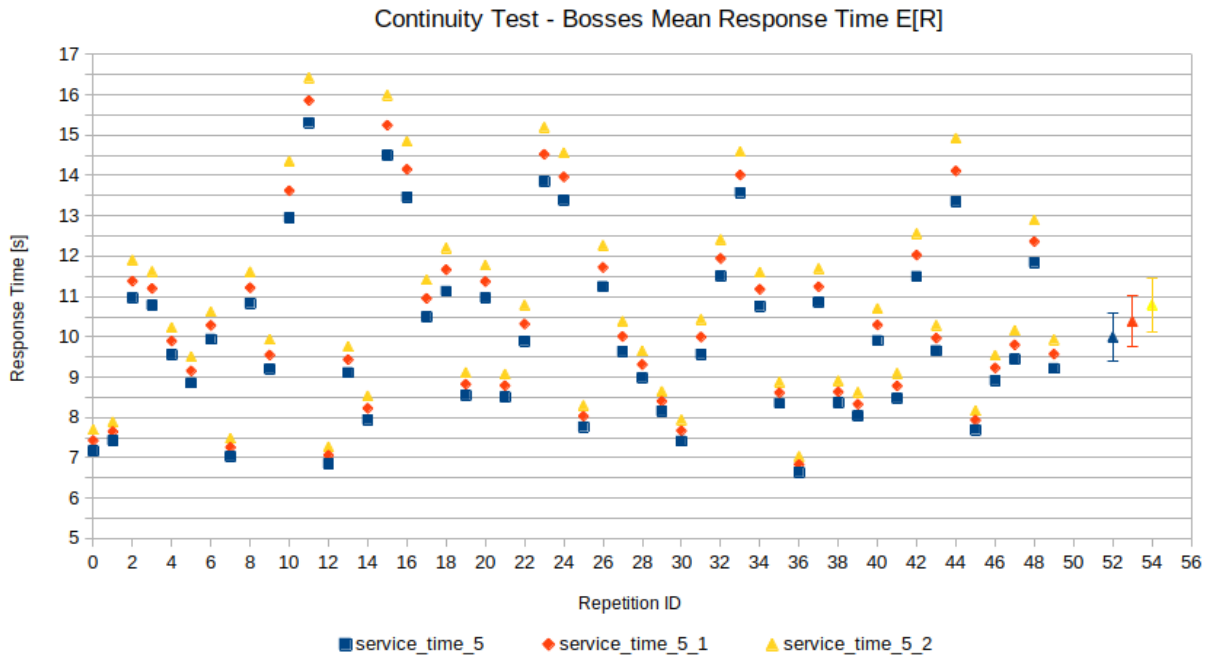


Figure 5: Continuity Test - Mean Response Time results

The full results are summarised in the following table:

Continuity Test on Mean Response Time E[R]	
Mean service time	**Mean Response Time E[R]**
5 s	9.988 s
5.1 s	10.378 s
5.2 s	10.788 s

As expected, increasing slightly the service time (opponent life) also the mean response time increases.

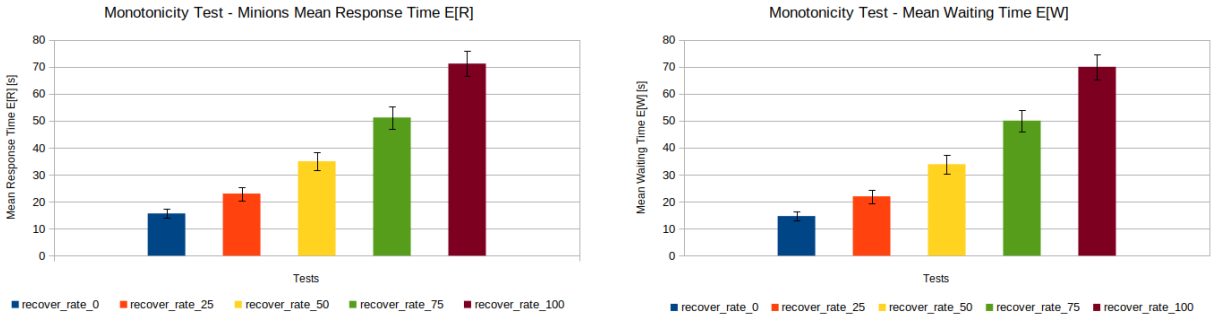
4.4.2 Monotonicity Test on Recover Rate x

This test has been carried out on minion type messages, by choosing specific values in order to reproduce scenarios when many minion messages have a life recovery and are queued again. The values used to perform this test are:

- **Recover Rate x values: 0%, 25%, 50%, 75% 100% .**
- **Repetitions : 100**

In a scenario where a lot of minions are recovered, increasing the percentage of the life to recover the following observations can be made:

1. The minions response and waiting time increases.



(a) Monotonicity Test on Mean Response Time E[R] (b) Monotonicity Test on Mean Waiting Time E[W]

Figure 6: Monotonicity Test - E[R] and E[W] results

2. The number of minions defeated decreases and consequently the throughput decreases.

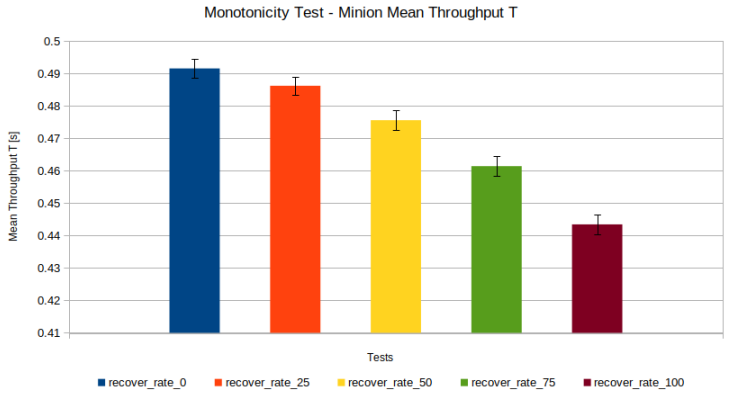


Figure 7: Monotonicity Test - Throughput T results

As expected, the monotonicity evaluation yielded good results and so, the test can be defined as successfully passed.

4.5 Verification Against the Theoretical Model

The goal of this kind of verification is to compare the output of a simplified version of our system against a theoretical model.
In order to create a simplified model of the system that reflects a queuing model were made the following assumptions:

- There is **only one opponent type** (e.g. bosses) and **no recovering system**.
- **Exponential interarrival and service time**.

The decision to have only bosses as opponent type lead us to discard the use of one of the two queues and then **reproduce a M/M/1 system** (queue + server), where the player is the "server" and the bosses are the "jobs".

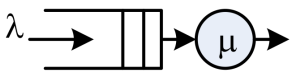


Figure 8: M/M/1 System Model

The metrics that can be computed both theoretically and by using the simulator are the **mean response time E[R]**, the **mean waiting time E[W]** and the **Throughput T**.

In an exponential distribution scenario the service time is: $f(x) = \mu e^{\mu x}$ where μ is the rate of the exponential distribution.

From queueing theory is possible to write the following formulas in order to compute the expected metrics:

- **System utilization**: $\rho = \frac{\lambda}{\mu}$
- **Number of jobs in the system**: $E[N] = \frac{\rho}{1-\rho}$
- **Mean Response Time**: $E[R] = \frac{E[N]}{\lambda}$
- **Mean Waiting Time**: $E[W] = E[R] - \frac{1}{\mu}$
- **Throughput T**: $\gamma = \mu\rho$ (where γ is the throughput formal denotation).

The system was tested in the following scenario:

- **Boss mean arrival time $\frac{1}{\lambda} = 1$**
- **Boss mean service time $\frac{1}{\mu} = 0.8$**
- **Simulation duration : 2000s**
- **Repetitions : 30**

The theoretical results obtained are the following:

$$\rho = 0.8,\ E[N] = 4,\ E[R] = 4,\ E[W] = 3.2,\ T = 1 \quad .$$

The results computed with the simulation are in line with expectations and are shown in the following table:

Verification Against Theoretical Model results		
Metric	**Value**	**Confidence Interval**
E[R]	4.0825	0.2814
E[W]	3.2818	0.2776
T	0.99845	0.00807

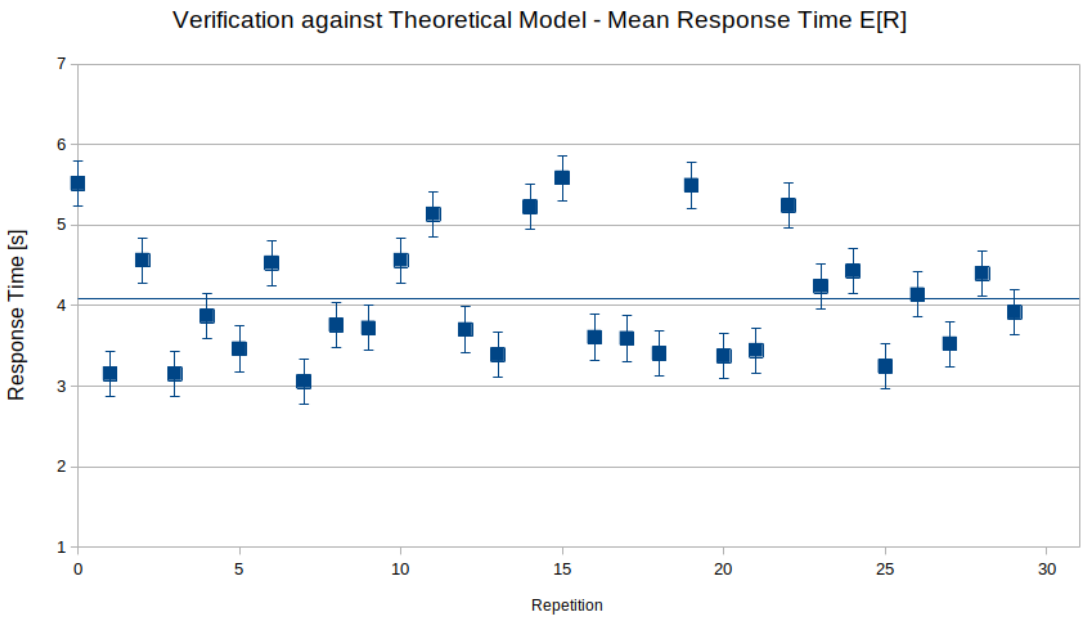


Figure 9: Verification against theoretical model - Mean Response Time E[R]

5 Factors Calibration

5.1 Scenario Calibration

The aim of this part is to fix the intervals of the factors to correctly reproduce the behaviour of the real system.
To do so, some hypotheses have to be made in order to create real life scenarios and give some insights for the system under study.

The main goal of a videogame is to have a good game experience from the user's perspective, so the whole factor calibration phase turns on reach this result.

Through an investigating analysis on the mean response time E[R] as a function of the mean service time E[ts] has been found out that, to produce this kind of behaviour some rules must be observed. The ranges of interest for each factor are studied and then decided in detail below.

5.1.1 Bosses Factors Calibration

Having priority, the bosses are only influenced by their mean time of arrival and service.
As can be seen from the results below, the response time of the bosses only increases exponentially when their mean service time approaches or even exceeds their mean arrival time. Therefore, it is **reasonable to use values for the mean service time that do not exceed the mean arrival time**.

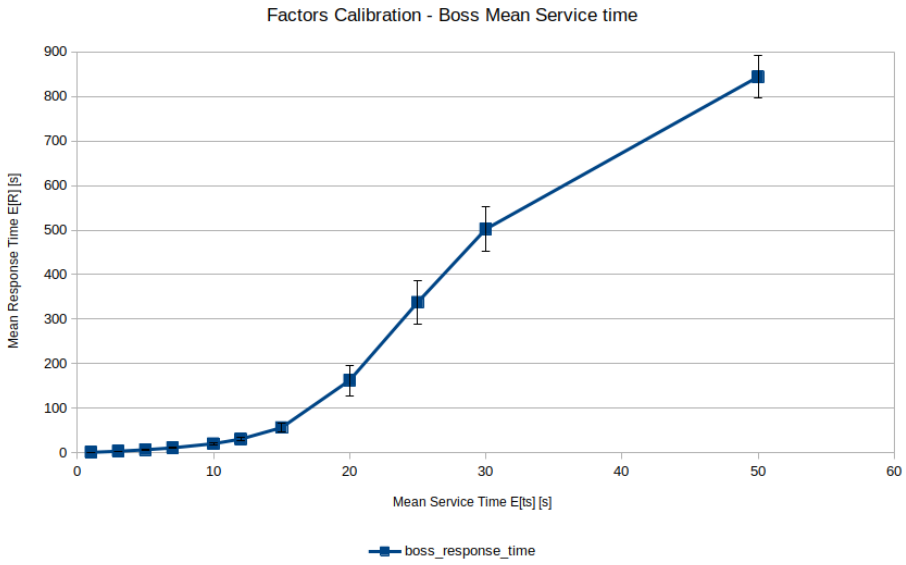
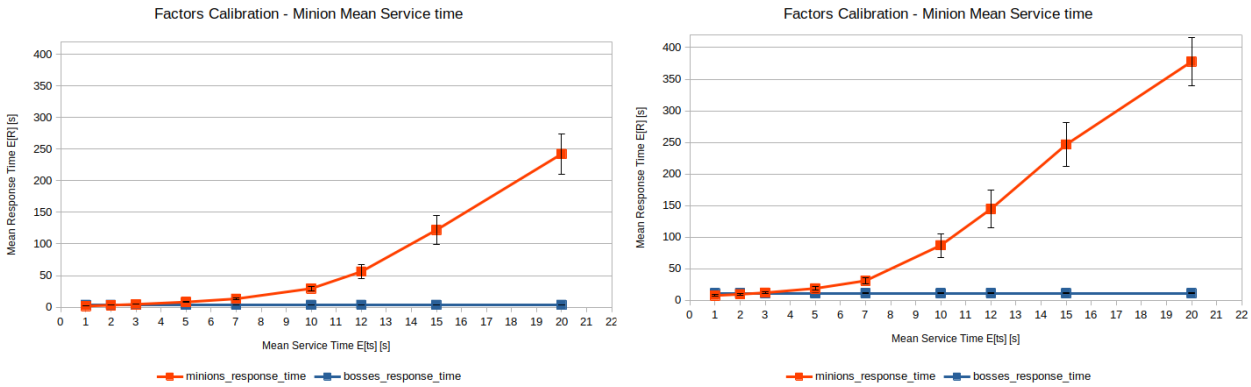


Figure 10: Example of Bosses Factors Calibration

5.1.2 Minions Factors Calibration

Unlike bosses, minions are influenced not only by their mean service and arrival time but also by those of the bosses and so, their behaviour changes accordingly.
To carry out this calibration, the behaviour of minions was studied both by increasing their service time and the service time of bosses. In fact can be seen that the more the boss service time increases, the more it would make the minions response time high and unrealistic.

Boss mean arrival time and minion mean arrival time values were fixed in this calibration ($\frac{1}{\lambda_b} = 20s$ and $\frac{1}{\lambda_m} = 20s$).



(a) Bosses mean service time 3 seconds

(b) Bosses mean service time 7 seconds

Figure 11: Example of Minions Factors Calibration

It may be concluded that a fairly realistic approach would be to **keep the mean service time of minions below their mean arrival time** and to **have a clear gap between the bosses mean arrival time and the minions mean arrival time**, in order to do not create queues for minions which would tend to take on excessively high values.

5.1.3 Recover Rate Factor Calibration

The recover rate x could have an impact on minion results. By using an higher rate the minions would recover more service time, impacting the overall performance of the system.
In particular, we would like to see which values could make significant changes on the minions mean response time E[R].
In this calibration, values in the range [0,100] (in steps of 25) were used for the recover rate x, while the remaining values were fixed as follows: $\frac{1}{\lambda_b} = 10s$; $\frac{1}{\mu_b} = 3s$; $\frac{1}{\lambda_m} = 5s$; $\frac{1}{\mu_m} = 2s$.

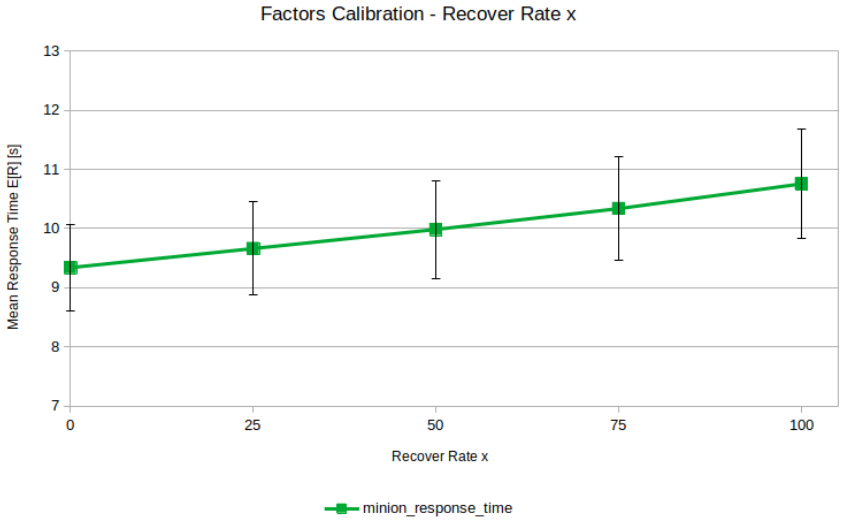


Figure 12: Example of Recover Rate x Factor Calibration

Through the results it can be seen that in this specific case using recover rate percentages not too distant from each other does not produce any major changes on the final results, but it could be observed that by using the 0% and the 100% the difference produced could be non-negligible in longer game simulations. Having stated this, is reasonable to **use only the mentioned extreme values** for the simulation experiments.

Having established these criteria the following ranges for the parameters were decided:

- **Boss Mean Arrival time $\frac{1}{\lambda_b}$:** [20s, 25s]
- **Boss Mean Service time $\frac{1}{\mu_b}$:** [5s, 10s]
- **Minion Mean Arrival time $\frac{1}{\lambda_m}$:** [10s, 15s]
- **Minion Mean Service time $\frac{1}{\mu_m}$:** [1s, 7s]
- **Recover Rate x:** [0, 100]

5.2 Calibration of Warm-Up Time

To calibrate on an optimum value for the warm-up time it is decided to use the response time E[R] performance index, selected as most reliable because is never null, unlike the waiting time E[W].

The technique used is the sliding window average, and the chosen warm-up time is when the values converges.

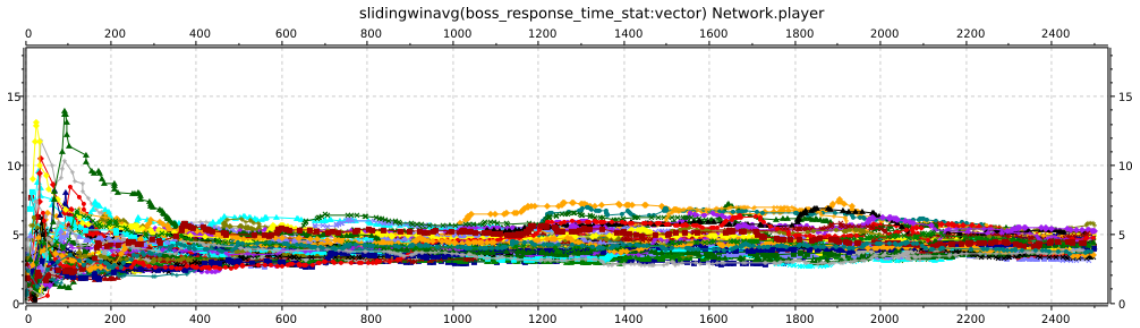


Figure 13: Warm-Up Time Calibration - Sliding window average on response time

Through the plots is possible to see that, even if the system never actually becomes stable, it keeps oscillating inside a well-defined band after a threshold of 350 seconds, and for this reason the value of the **warm-up time selected is 350 seconds**.

5.3 Calibration of Simulation Time

For what concern the simulation time calibration the study was made on the response time. In particular, were studied the mean value and the standard deviation on the samples collected.

Based on the tests made, it has been possible to observe that **2500 seconds is a good value to use for the simulation experiments**, based on the fact that the standard deviation and mean value for the response time are stabilized and the number of samples was deemed good enough to yield reliable conclusions on the experiments results.

6 Simulation Experiments

Given the assumptions highlighted in the previous chapter, i.e. that the main goal of the videogame is to have a good gaming experience from the user's point of view, it was decided to divide the final simulation experiments into three different categories, using an exponential distribution for mean service and arrival time in all three scenarios.
Usually for videogames of this kind the division can be made into **easy, medium and hard game modes**.

This chapter presents the results obtained of a **$2^k r$ factorial analysis** and **simulation experiments on the three different games modes** and then final considerations on them.

6.1 $2^k r$ Factorial Analysis

Before proceeding with the experiments was carried out a $2^k r$ factorial analysis to get some insights and a deeper understanding of the system under analysis. This analysis was carried out with the aim of understanding the contribution of factors based on the results of minions and bosses KPIs.

It was decided to make more than one $2^k r$ analysis, changing the used ranges of values in order to reproduce easy and hard game modes scenarios.

The $2^k r$ Analysis is made on both metrics to be analyzed, the **Throughput T** and the **Mean Waiting Time E[W]**.

The **number of replicas used is $r = 5$** and the **number of factors in the system is $f = 5$**, which are listed below along with their ranges used:

$2^k r$ Factorial Analysis Factors and Ranges			
Factor	**Metric**	**Easy Game Mode Ranges**	**Hard Game Mode Ranges**
A	Boss Mean Arrival time $\frac{1}{\lambda_b}$	[20s, 30s]	[20s, 30s]
B	Boss Mean Service time $\frac{1}{\mu_b}$	[5s, 7.5s]	[7.5s, 10s]
C	Minion Mean Arrival time $\frac{1}{\lambda_m}$	[10s, 15s]	[10s, 15s]
D	Minion Mean Service time $\frac{1}{\mu_m}$	[1s, 3.5s]	[3.5s, 7s]
E	Recover Rate x	[0, 100]	[0, 100]

The objective for all factorial analysis is to test the hypothesis, i.e. that **the residuals (errors) are IID's normal RV with a null mean and a costant standard deviation.**

This is performed through the study of **QQ plots of residuals vs normal** (if residuals are normal distributed) and the study of **QQ plots of residuals vs predicted response** (if the standard deviation is constant, also known as **homoskedasticity**).

In the following, not all the $2^k r$ analysis performed are presented, but only the most relevant ones, which produced the most significant and useful results for experimental purposes (is possible to see the remaining graphs in the $2^k r$ analysis folder).

6.1.1 $2^k r$ Factorial Analysis on Throughput T (Easy Game Mode)

Regarding the first hypothesis, a **kind of linear trend can be seen in the graph** (Figure 14). On the other hand, as far as homoskedasticity is concerned, is possible to see that **no visible trends occurs** (Figure 15).

Based on the latter statement, it can be concluded that **the results are correct and the assumption is verified**. In this way, it is possible to study the impact percentages of the factors and derive some considerations on them.

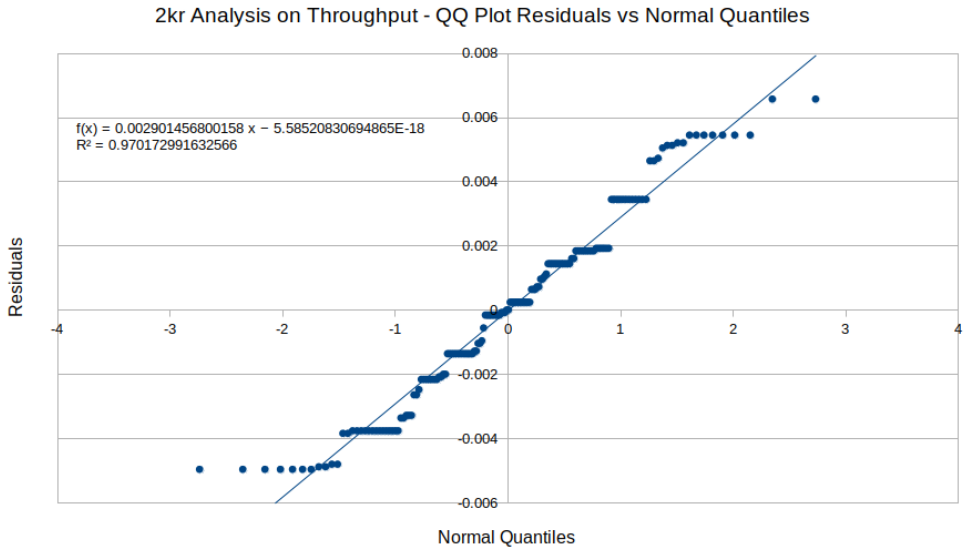


Figure 14: QQ Plot of residuals vs normal quantiles on Throughput $2^k r$ Factorial Analysis

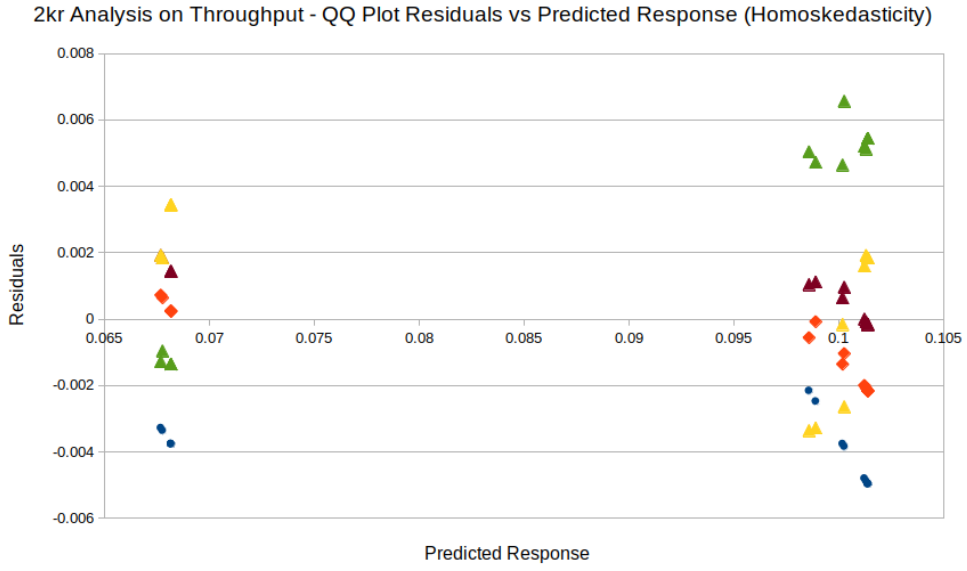


Figure 15: QQ Plot of residuals vs predicted response on Throughput $2^k r$ Factorial Analysis

It is now possible to analyze the results obtained, shown in the following table.

2kr Factorial Analysis on Throughput -Variation Results	
Factor	**Variation %**
A	0.19%
B	0.27%
C	48.18%
D	0.30%
E	0.42%
CD	11.22%
CE	10.56%
CDE	10.52%
Others	<3.5%

Through the results, it can be concluded that **the factor with the highest percentage of impact is the mean arrival time of minions (factor C)**, equal to 48%. Moreover, it can be seen that **the highest percentages are those where all factors concerning minions are present**. Instead, where the bosses factors are present the impact percentages are negligible.

For instance, combinations such as CD, CE and CDE (minions mean arrival, service time and minions recover rate) have a percentage >10% and are therefore not negligible.

6.1.2 $2^k r$ Factorial Analysis on Mean Waiting Time E[W] (Hard Game Mode)

The results produced on mean waiting time E[W] were unsatisfactory, in fact was possible to see from the residuals vs normal quantiles results (Figure 16) that **the trend is not linear but is heavy-tailed**.

Regarding the homoskedasticity, it is **possible to see a visible trend** and therefore in conclusion it **can be stated that the initial assumption is not met and the analysis is incorrect** providing impact percentages that cannot be considered in the study.

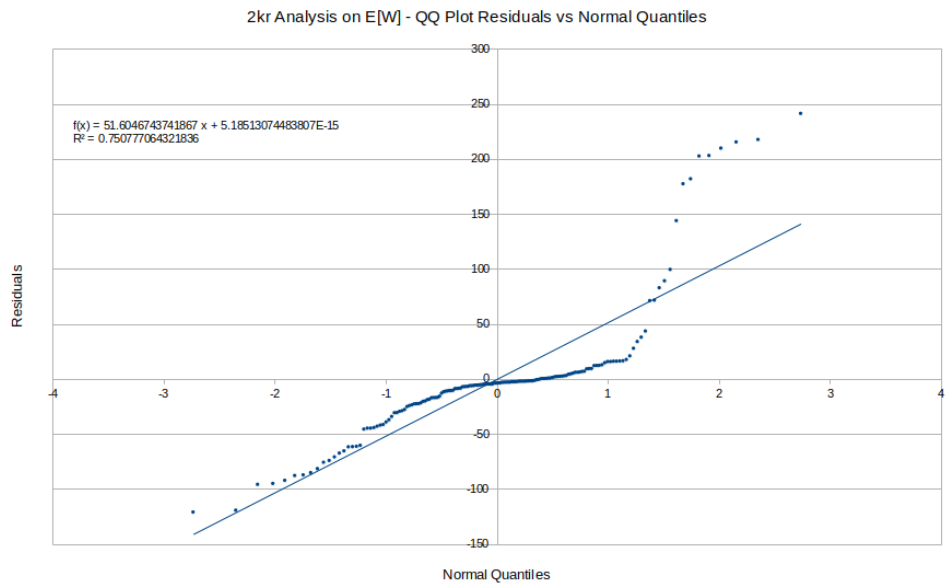


Figure 16: QQ Plot of residuals vs normal quantiles on Mean Waiting Time E[W] $2^k r$ Factorial Analysis

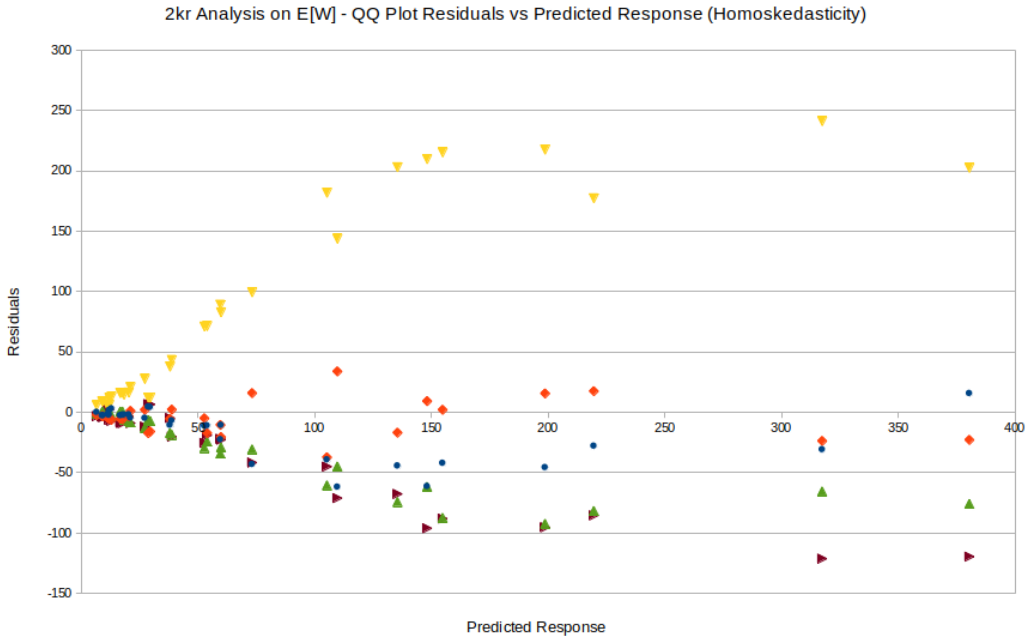


Figure 17: QQ Plot of residuals vs predicted response on Mean Waiting Time E[W] $2^k r$ Factorial Analysis

Trying to improve the final results and verify the initial assumption, was made an attempt **to perform a logarithmic transformation on the data** obtained from the simulation experiments.

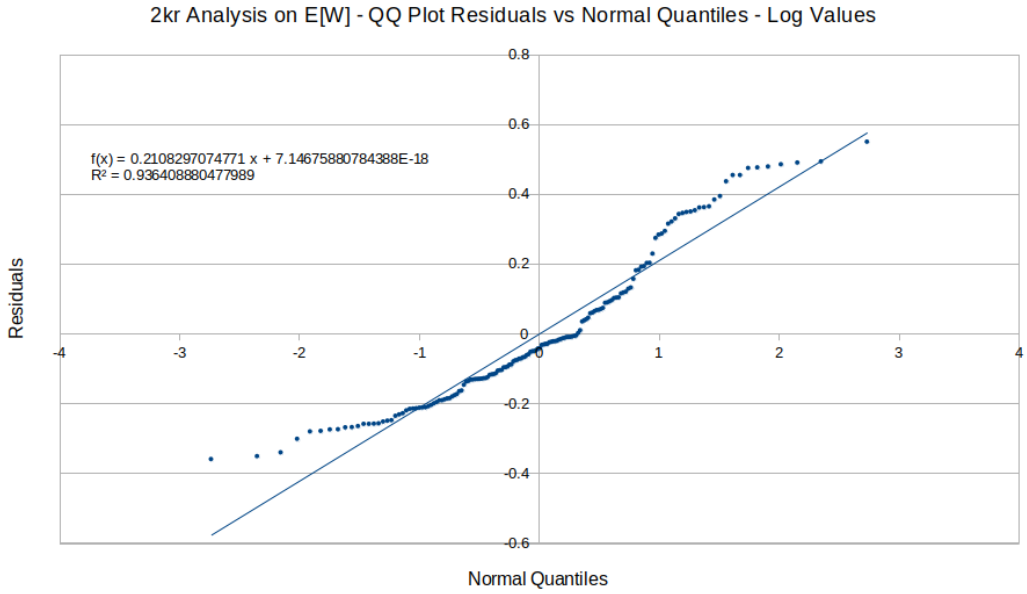


Figure 18: QQ Plot of residuals vs normal quantiles on Mean Waiting Time E[W] $2^k r$ Factorial Analysis - Logarithmic Transformation

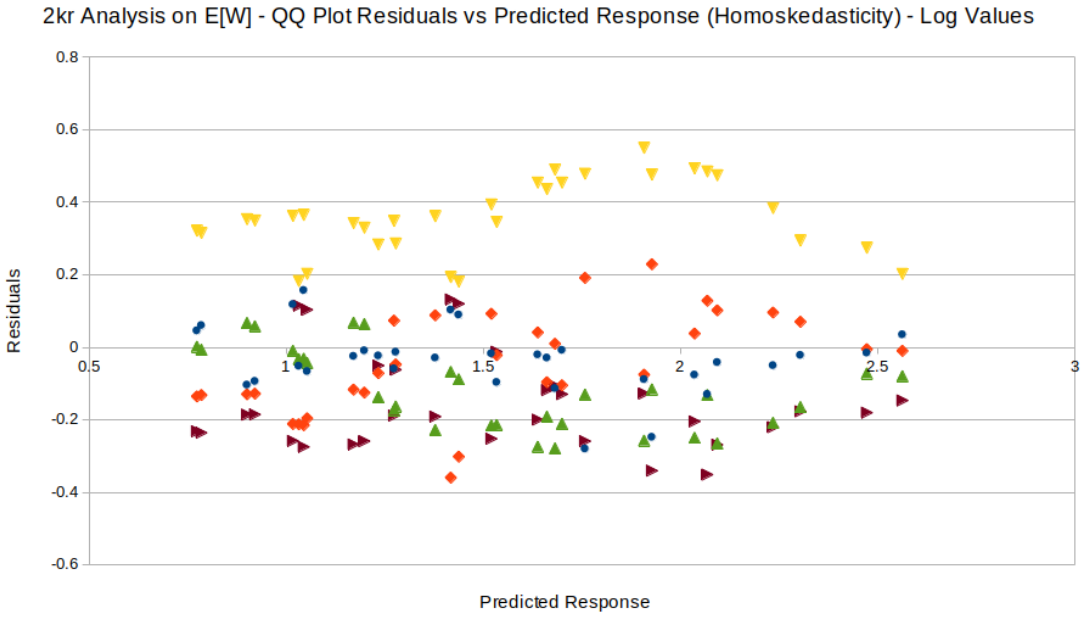


Figure 19: QQ Plot of residuals vs predicted response on Mean Waiting Time E[W] 2kr Factorial Analysis - Logaritmic Transformation

As can be seen from the graphs produced, the final results have improved, especially the homoskedasticity which no longer shows visible trends.

Unfortunately, the QQ plot of residuals vs normal quantiles does not show a normal distribution of errors but still an heavy-tailed distribution.

Although the method did not yield good results, it was still useful to employ this technique and a positive note that can be made is that in both cases (with and without logarithmic transformation) the factor with the greatest impact is the minions mean service time (factor D) and the error contribution is negligible.

6.2 Easy Game Mode Simulation Experiments

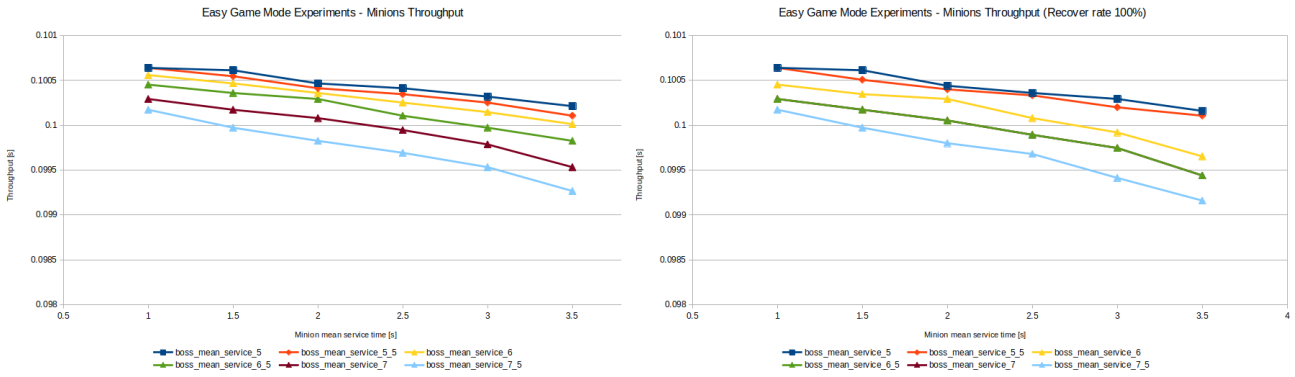
The ranges of values used for the system parameters are the following:

$\frac{1}{\lambda_b} = 20s$, $\frac{1}{\mu_b} = [5s, 7.5s]$ (steps of 0.5s), $\frac{1}{\lambda_m} = 10s$, $\frac{1}{\mu_m} = [1s, 3.5s]$ (steps of 0.5s), **Recover rate x** = [0%, 100%]

6.2.1 Simulation Experiments Results on Throughput T

In this case, the results were fairly predictable. Having priority for bosses, their **Throughput T_b remains consistent with their mean arrival times**.

Instead, for the minions having low values for their mean service time compared to their arrival time the **results of the Throughput T_m are almost identical among the various experiments**.
In fact, if the CI (Confidence Interval) values had been put in the graphs (not placed in order to show the slight difference between the various configurations) it would be seen that there is no difference on the several configurations results and the mean values can be considered identical, whether using the recover rate or not (Figure 20).



(a) Minions Throughput without the Recover Rate (b) Minions Throughput employing the Recover Rate

Figure 20: Easy Game Mode Experiments - Minions Throughput with and without Recover Rate

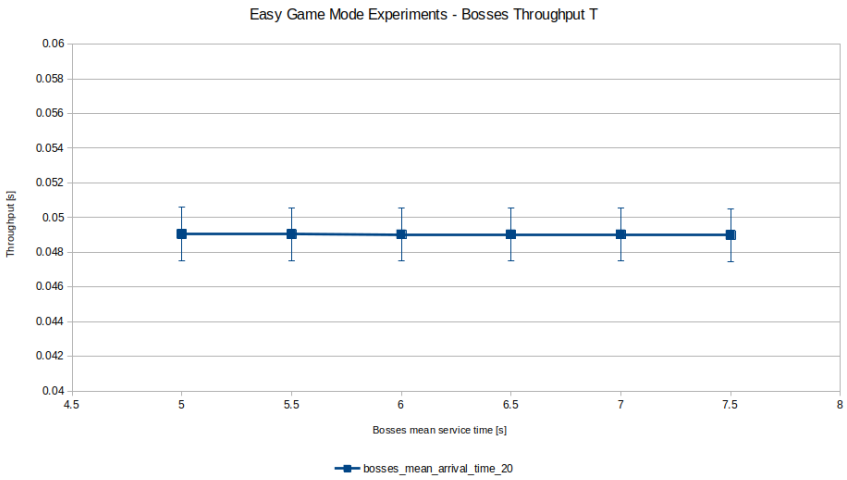
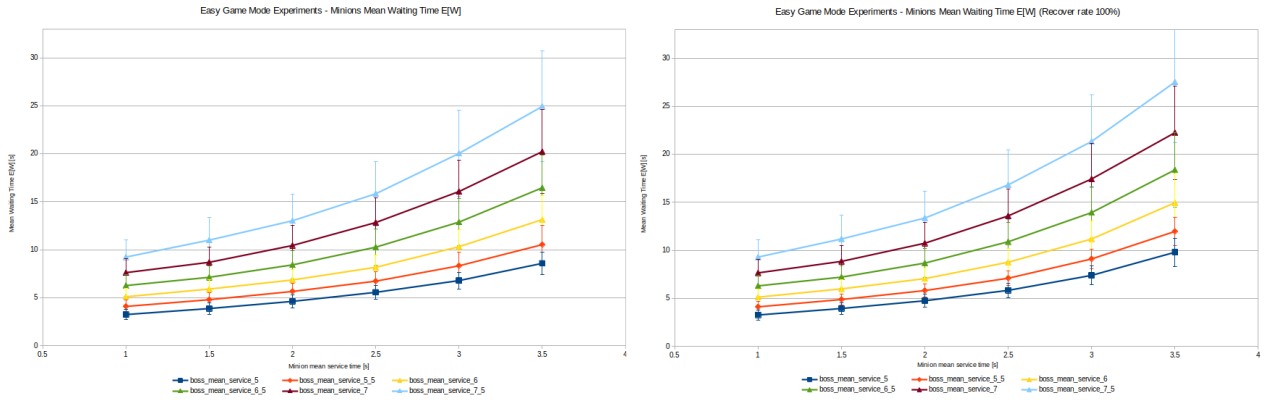


Figure 21: Easy Game Mode Experiments - Bosses Throughput

6.2.2 Simulation Experiments Results on Mean Waiting Time E[W]

In this case the results show relatively **small mean waiting time E[W] values for both bosses and minions**. Moreover, the use of recover rate again does not have a significant impact on the minion results.



(a) Minions Mean Waiting Time without recover rate (b) Minions Mean Waiting Time employing the recover rate

Figure 22: Easy Game Mode Experiments - Minions Mean Waiting Time with and without Recover Rate

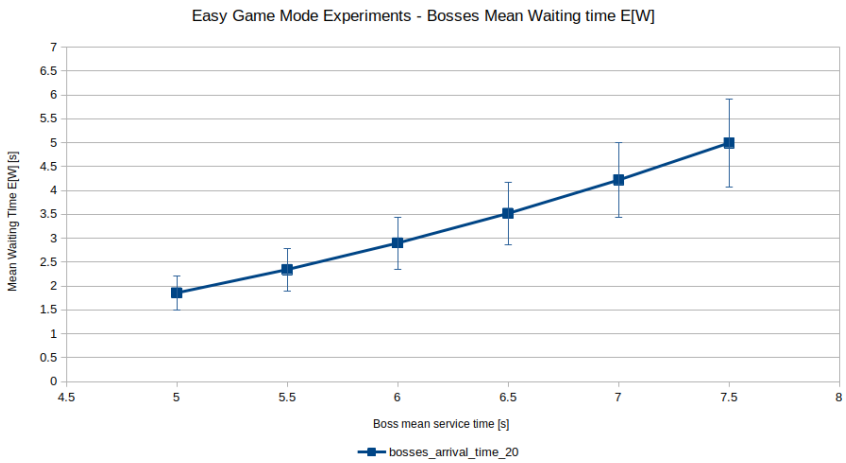


Figure 23: Easy Game Mode Experiments - Bosses Mean Waiting Time

From the point of view of the system, this scenario can be properly considered as an easy game mode because **the player has enough time to fight against his opponents without creating long queues** and **defeats a fairly large number of them**, regardless of the type of opponents he faces (defeats 1 boss every 20 seconds and 1 minion every 10 seconds, i.e. their mean arrival times).

6.3 Medium Game Mode Simulation Experiments

The ranges of values used for the system parameters are the following:

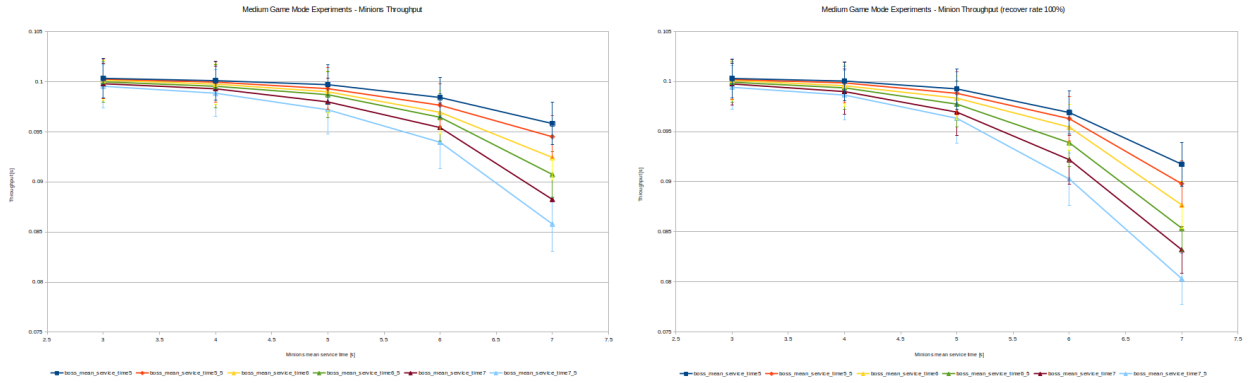
$\frac{1}{\lambda_b}$ = 20s, $\frac{1}{\mu_b}$ = [5s, 7.5s] (steps of 0.5s), $\frac{1}{\lambda_m}$= 10s, $\frac{1}{\mu_m}$ = [3s, 7s] (steps of 1s), **Recover rate x** = [0%, 100%]

Compared to easy game mode, non-negligible differences can be noticed for both Throughput T and Mean Waiting Time E[W] results.

6.3.1 Simulation Experiments Results on Throughput T

The most relevant consideration is that by changing the value of the minion mean service time, the value of Throughput is no longer the same.

In particular, depending on the chosen configuration, it could take up to 12.5 seconds to defeat a minion (throughput equal to 0.08s), which, for long simulations, could be relevant for the overall performance of the system.



(a) Minions Throughput without recover rate (b) Minions Throughput employing the Recover Rate

Figure 24: Medium Game Mode Experiments - Minions Throughput with and without recover

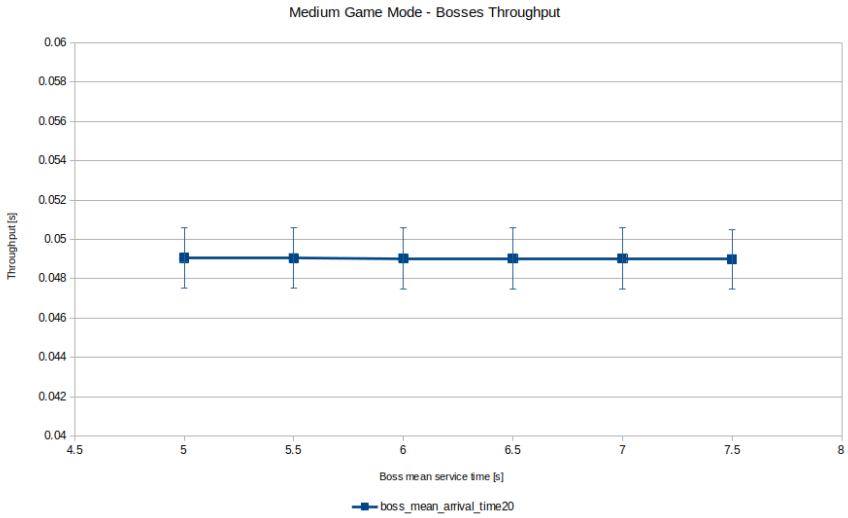


Figure 25: Medium Game Mode Experiments - Bosses Throughput

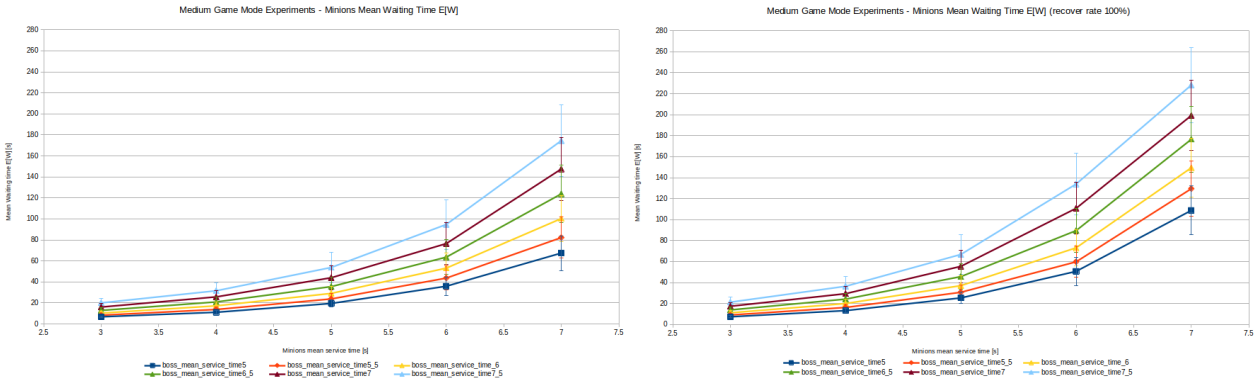
6.3.2 Simulation Experiments Results on Mean Waiting Time E[W]

As with throughput, the mean waiting time also varied.

In this case, **the mean service time of the minions has an even more visible impact on the mean waiting time of the minions**, which when combined with the mean service time of the bosses, that already produced an increase in the mean waiting time, yields results that vary greatly (Figure 26a).

Although not by much, it can already be seen that **the recovery rate has a greater impact on the overall system performance, that can let the mean waiting time increase by up to 40 seconds in the worst case** (Figure 26b).

For what concern the **bosses mean waiting time**, it is possible to see that it **takes same values as previous experiments**, since both mean service time and mean arrival time have not changed (Figure 27).



(a) Minions Mean Waiting Time without recover rate (b) Minions Mean Waiting Time employing the recover rate

Figure 26: Medium Game Mode Experiments - Minions Mean Waiting Time with and without recover

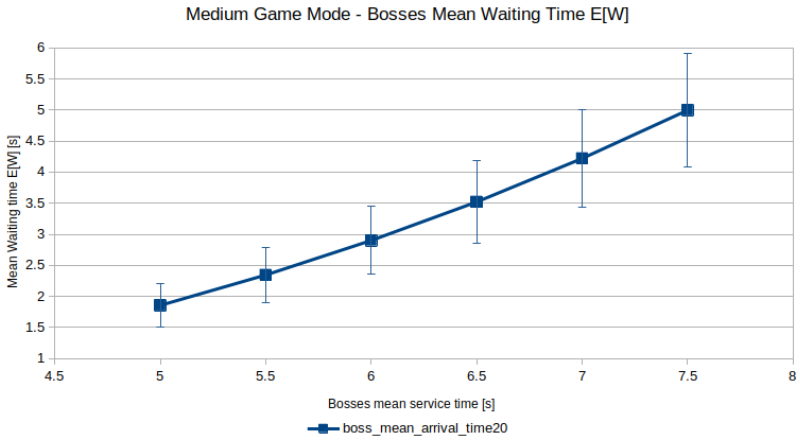


Figure 27: Medium Game Mode Experiments - Bosses Mean Waiting Time

6.4 Hard Game Mode Simulation Experiments

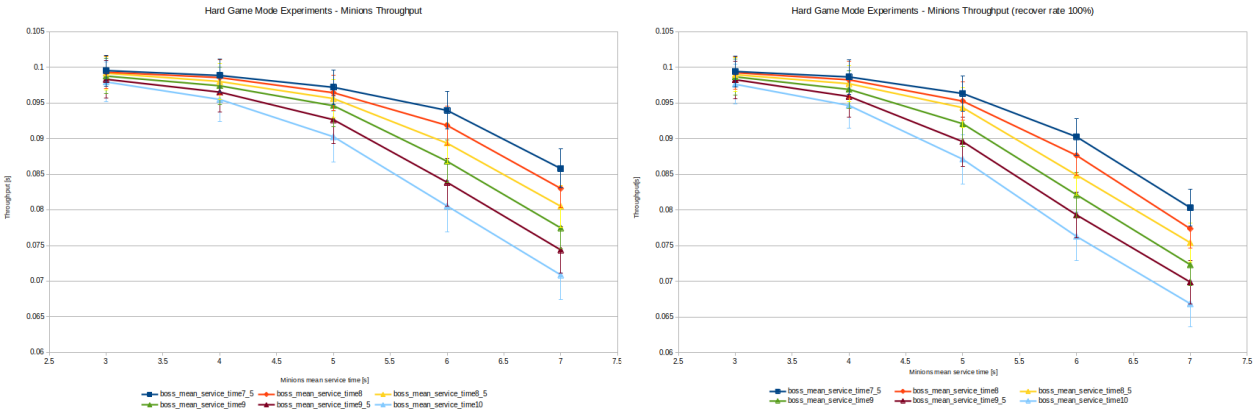
The ranges of values used for the system parameters are the following:

$\frac{1}{\lambda_b}$ = 20s, $\frac{1}{\mu_b}$ = [7.5s, 10s] (steps of 0.5s), $\frac{1}{\lambda_m}$= 10s, $\frac{1}{\mu_m}$ = [3s, 7s] (steps of 1s), **Recover rate x** = [0%, 100%]

6.4.1 Simulation Experiments Results on Throughput T

In this case, **the minion throughput shows a further drop, due to the increase in the bosses mean service time values** employed in the hard game mode.

This change resulted in minion throughput values of 0.07s, i.e. 1 minion defeated every 14 seconds (Figure 28a) and applying the recover rate the throughput value reached approximately 0.065s, i.e. 1 minion defeated every 15 seconds (Figure 28b).



(a) Minions Throughput without recover rate

(b) Minions Throughput employing the Recover Rate

Figure 28: Hard Game Mode Experiments - Minions Throughput with and without recover

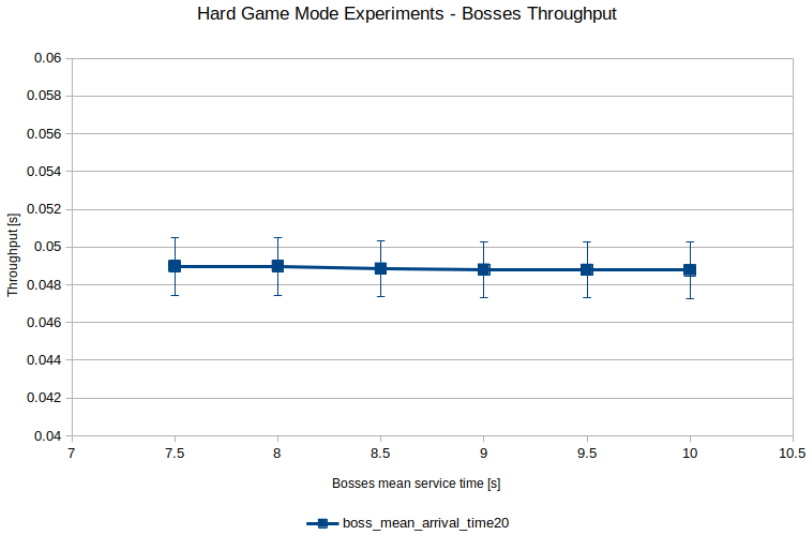


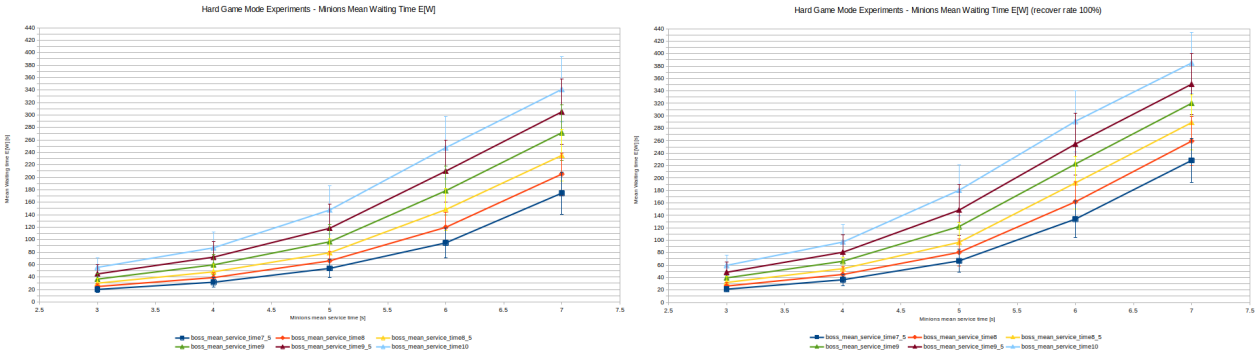
Figure 29: Hard Game Mode Experiments - Boss Throughput

6.4.2 Simulation Experiments Results on Mean Waiting Time E[W]

With regard to mean waiting time, the hard game mode configuration has effects on both boss and minion results.

The mean waiting time of bosses gets to double respect to the previous configurations, being able to let the player wait up to 12 seconds between one boss fight and another (Figure 31).

The mean waiting time of minions also doubles its results, and in this case even using the 100% recover rate produces noticeable changes (Figure 30b). The wait between the start of one fight and another could reach 400 seconds, which can be considered unlikely in this kind of videogames.



(a) Minions Mean Waiting Time without recover rate (b) Minions Mean Waiting Time employing the recover rate

Figure 30: Hard Game Mode Experiments - Minions Mean Waiting Time with and without recover

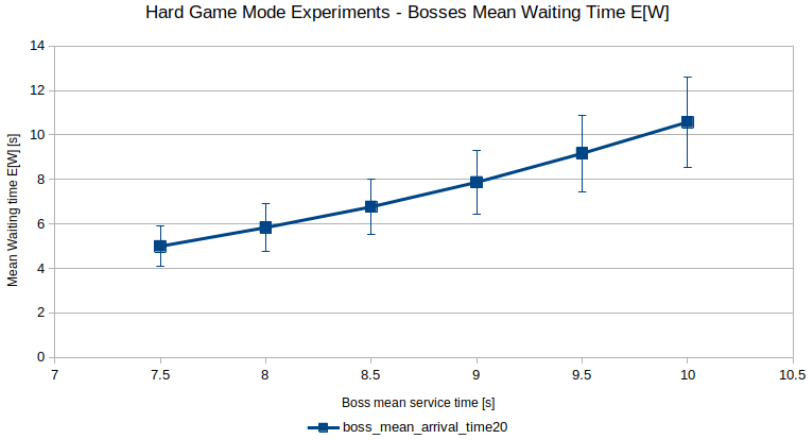


Figure 31: Hard Game Mode Experiments - Bosses Mean Waiting Time

7 Conclusions

It was seen how play by selecting different game modes could change the user's gaming experience.

In general it was possible to point out that using the easy game mode could satisfy any type of player quite well, due to limited waiting time and throughput of both opponent types.

On the contrary, choosing a medium game mode or even the hard game mode the waiting time can considerably increase, having a negative impact on the number of opponents defeated per second and then on the user game experience, lowering the satisfaction level of the players.

The question we want to answer is, why is this study useful and which benefits can it give us?

Knowing these results, **average game experiences can be estimated, predicting how a game might take place and which goals users would achieve**.
Hence, these results could prove to be a powerful tool in one of the most delicate phases of the development of a "punch'em videogame", i.e. deciding the key values which, as we have seen, can be: opponents arrival time, life strength, life recovery mode and many others.

Being able to study the possible outcomes of a game in advance and set the basic parameters of the game appropriately can improve users performance and, above all, their level of satisfaction and fulfilment, which can be key factors in the success of a videogame.