

# Progetto di Simulazione di Sistemi

Alternating server with non-zero switch-over times  
and opposite-queue threshold-based switching policy

Andrea Schinoppi, matricola 0001097628

## Indice

<b>1</b>	<b>Introduzione</b>	<b>3</b>
<b>2</b>	<b>Modello da realizzare</b>	<b>3</b>
<b>3</b>	<b>Descrizione del codice prodotto</b>	<b>3</b>
3.1	Struttura del progetto . . . . .	3
3.2	Rete realizzata . . . . .	4
3.2.1	Modifiche apportate ai file .ned . . . . .	4
3.3	Codice sviluppato . . . . .	4
3.3.1	Nuova SelectionStrategy . . . . .	4
3.3.2	Modifiche ad handleMessage . . . . .	5
<b>4</b>	<b>Simulazioni</b>	<b>7</b>
4.1	Configurazione 1 . . . . .	7
4.2	Configurazione 2 . . . . .	7
<b>5</b>	<b>Analisi dei risultati</b>	<b>8</b>
5.1	Approccio generale . . . . .	8
5.2	Analisi dei vettori . . . . .	8
5.2.1	$\lambda_1 = 1.0, \lambda_2 = 1.0$ . . . . .	9
5.2.2	$\lambda_1 = 1.0, \lambda_2 = 2.0$ . . . . .	9
5.2.3	$\lambda_1 = 2.0, \lambda_2 = 1.0$ . . . . .	10
5.2.4	$\lambda_1 = 2.0, \lambda_2 = 2.0$ . . . . .	11
5.2.5	$\lambda_1 = 4.0, \lambda_2 = 1.0$ . . . . .	11
5.2.6	$\lambda_1 = 4.0, \lambda_2 = 2.0$ . . . . .	12
5.2.7	$\lambda_1 = 10.0, \lambda_2 = 1.0$ . . . . .	13
5.2.8	$\lambda_1 = 10.0, \lambda_2 = 2.0$ . . . . .	13
5.2.9	$\lambda_1 = 20.0, \lambda_2 = 1.0$ . . . . .	14
5.2.10	$\lambda_1 = 20.0, \lambda_2 = 2.0$ . . . . .	15
5.2.11	$\lambda_1 = 1.0, \lambda_2 = 1.5$ . . . . .	16
5.2.12	$\lambda_1 = 1.0, \lambda_2 = 2.0$ . . . . .	16
5.2.13	$\lambda_1 = 2.0, \lambda_2 = 1.5$ . . . . .	17
5.2.14	$\lambda_1 = 2.0, \lambda_2 = 2.0$ . . . . .	18
5.2.15	$\lambda_1 = 3.0, \lambda_2 = 1.5$ . . . . .	18
5.2.16	$\lambda_1 = 3.0, \lambda_2 = 2.0$ . . . . .	19
5.2.17	$\lambda_1 = 3.5, \lambda_2 = 1.5$ . . . . .	20

5.2.18 $\lambda_1 = 3.5, \lambda_2 = 2.0$ . . . . .	20
5.3 Transiente iniziale . . . . .	21
5.4 Tempo medio di permanenza nel sistema dei job . . . . .	22
5.4.1 Configurazione 1 . . . . .	22
5.4.2 Configurazione 2 . . . . .	27
5.5 Numero medio di utenti nelle singole code . . . . .	30
5.5.1 Configurazione 1 . . . . .	30
5.5.2 Configurazione 2 . . . . .	35
5.6 Numero medio di utenti persi . . . . .	38
5.6.1 Configurazione 1 . . . . .	38
5.6.2 Configurazione 2 . . . . .	41
<b>6 Altre analisi</b>	<b>43</b>
6.1 Tempo medio trascorso in coda dai job . . . . .	43
6.2 Tempo medio di servizio dei job . . . . .	44
6.3 Tempo medio di vita dei job . . . . .	45
6.4 Tempo medio per cui il servente è occupato . . . . .	45
<b>7 Convalida del modello</b>	<b>46</b>
7.1 Analisi rispetto a $\lambda_1$ . . . . .	46
7.2 Analisi rispetto a $\lambda_2$ . . . . .	48
7.3 Analisi rispetto a $\mu_1$ . . . . .	49
7.4 Analisi rispetto a $\mu_2$ . . . . .	50
7.5 Analisi rispetto ad $\alpha$ . . . . .	51
7.6 Conclusioni . . . . .	51

# 1 Introduzione

Scopo di questo progetto era creare una variante del modello di simulazione descritto nelle sezioni 1 e 2 dell'articolo ”Alternating server with non-zero switch-over times and opposite-queue threshold-based switching policy” [JPY18] mediante piattaforma Omnet++.

Questa relazione sarà suddivisa come segue: nella sezione 2 verrà illustrato il modello da realizzare, nella sezione 3 verrà descritto brevemente il codice prodotto, nella sezione 4 saranno illustrate le configurazioni di simulazione, la sezione 5 sarà dedicata all’analisi dei risultati, la sezione 6 tratterà ulteriori analisi svolte e infine nella sezione 7 sarà convalidato il modello di simulazione realizzato.

## 2 Modello da realizzare

Il modello da realizzare prevede che un servente serva per un certo service time una di due code, secondo una politica basata su threshold: quando la coda non servita raggiunge la soglia impostata, in generale, il server dovrebbe eseguire uno switch-over con una certa durata per passare a servirla.

Tuttavia se durante uno switch-over, anche la coda da cui il server proviene raggiunge la propria soglia, il server dovrà continuare a servirla senza effettuare alcun cambio.

Inoltre, nel caso in cui entrambe le code superino il threshold contemporaneamente, lo switch-over non avverrà.

Infine, nel caso in cui le code fossero vuote, il servente attenderà che una abbia almeno un elemento per iniziare (o passare) a servirla.

Le due code sono M/M/1 e M/M/1/ $C_1$ , ciò significa che la prima ha capacità infinita mentre la seconda ha capacità  $C_1$ . Raggiunta la propria capacità, la seconda coda comincerà quindi a perdere i job ricevuti.

I tempi di interarrivo dei job alle code sono indicati da  $\lambda_1$  e  $\lambda_2$  e sono distribuzioni esponenziali con una certa media  $\frac{1}{\lambda}$ .

I tempi di servizio del servente in base alla coda di provenienza dei job sono distribuzioni uniformi di valori in due differenti intervalli.

Infine, i threshold delle due code sono definiti rispettivamente dai parametri K ed N e la durata di uno switch-over è determinata dal parametro  $\alpha$ .

## 3 Descrizione del codice prodotto

In questa sezione verrà descritto brevemente il codice prodotto.

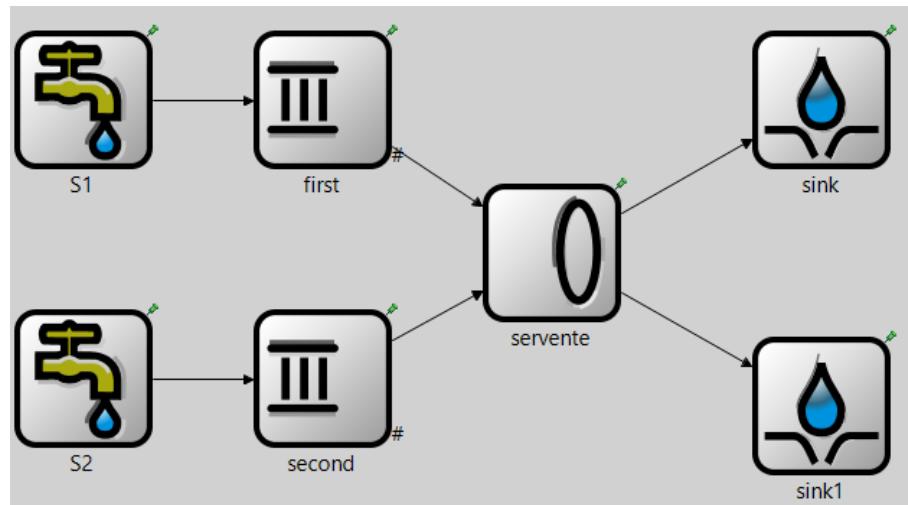
### 3.1 Struttura del progetto

La parte principale del progetto si trova nella cartella `/src`: essa contiene la cartella dei risultati `/results` e la cartella contenente gli elementi della queueinglib utili allo svolgimento del progetto `/queueing_utils`, oltre ai file `.ned` e `.ini`.

## 3.2 Rete realizzata

La rete realizzata per questo progetto (`/src/package.ned`) consiste in due sorgenti ("S1" ed "S2") che inviano job rispettivamente a due code ("first" e "second").

È presente un servente che secondo la politica basata su threshold precedentemente descritta serve una delle due code, per poi mandare i job serviti in un diverso sink ("sink" e "sink1") a seconda della provenienza.



**Figura 3.1:** Rete realizzata

La scelta di impiegare due sink è dettata dalla maggior quantità di dati che è possibile raccogliere.

### 3.2.1 Modifiche apportate ai file .ned

Per la realizzazione del progetto sono state necessarie alcune modifiche ai file .ned presenti nella cartella `/src/queueing_utils`.

Nel file `Server.ned`, per poter collegare l'uscita del servente a due sink distinti è stato aggiunto un gate di output denominato "out2".

Inoltre sono stati aggiunti i parametri per gestire i tempi di servizio  $\mu_1$  e  $\mu_2$  (`ServiceTime` e `ServiceTime_2`) e  $\alpha$  (`switchOverTime`).

Nel file `PassiveQueue.ned` è stato aggiunto il parametro `Threshold` che consente di impostare una soglia per le code.

## 3.3 Codice sviluppato

Di seguito sarà introdotto brevemente il codice sviluppato.

### 3.3.1 Nuova SelectionStrategy

Per prima cosa è stata aggiunta alla classe `SelectionStrategy`, ovvero quella contenente le diverse strategie con cui il servente può scegliere quale coda servire, la classe `ThresholdSelectionStrategy`, implementata in `/src/queueing_utils/SelectionStrategies.cc`.

L'implementazione è basata sulla verifica di tutti i possibili casi per decidere se continuare a servire una coda o avviare uno switch-over.

In particolare:

1. Se entrambe le code sono vuote il servente attende;
  - (a) Se le code erano vuote in precedenza viene verificato se lo siano ancora, controllando per prima quella che il server stava servendo ed avviando eventualmente uno switch-over.
2. Se solo la coda che il server sta attualmente servendo è vuota, avviene uno switch-over;
3. Se la coda non attualmente servita supera il threshold e l'altra no, avviene uno switch-over;
4. Se entrambe le code superano la propria soglia, il server continua a servire la stessa coda.

Inoltre ogni prima di ogni switch-over vengono nuovamente verificate tutte le condizioni per cui potrebbe dover essere annullato, in particolare:

- Viene verificato che la coda di destinazione non sia vuota;
- Viene verificato che la coda di destinazione non abbia superato la propria soglia nello stesso momento di quella di partenza.

Se una di queste verifiche non va a buon fine, lo switch-over è annullato.

### 3.3.2 Modifiche ad handleMessage

Un'altra modifica eseguita è stata quella alla funzione `Server::handleMessage`, che si trova nel file `/src/queueing_utils/Server.cc`.

In questo caso il codice è stato modificato per continuare a supportare le vecchie strategie di selezione ma anche la nuova `ThresholdSelectionStrategy`.

Porzioni di codice differente vengono eseguite in base al tipo di messaggio ricevuto dal servente:

1. Se il messaggio è un job, viene impostato un tempo di servizio in base alla sua provenienza, viene allocato e viene programmato un `endServiceMsg` al termine del service time.
2. Se il messaggio è un `endServiceMsg`, viene selezionato un gate di uscita in base alla provenienza del job attualmente gestito che viene poi deallocated e inviato al sink corretto.

In seguito viene chiamata la selection strategy per sapere da quale coda prelevare il prossimo job.

Da questo punto sono possibili 3 alternative:

- (a) Tutte le code sono vuote: il servente attende;
- (b) La coda restituita dalla selezione è la stessa che stava venendo servita in precedenza: viene richiesto un nuovo job dalla stessa coda;
- (c) La coda restituita dalla selezione è diversa da quella che stava venendo servita: switch-over.

In questo caso il server si invia un self-message chiamato `Begin_switchMsg`.

3. Se il messaggio è un `Begin_switchMsg`, lo stato del server viene impostato a occupato e viene programmato dopo  $\alpha$  secondi un messaggio `End_switchMsg`.
4. Se il messaggio è un `End_switchMsg`, lo stato del server torna libero e viene richiamata la strategia di selezione in modo da confermare o meno lo switch-over.

## 4 Simulazioni

Sono state eseguite due simulazioni con parametri differenti.

In tutti i casi sono stati eseguiti 20 esperimenti per ogni configurazione.

Il seed del generatore di numeri casuali è stato lasciato impostato a  $\$runnumber$ , ovvero al numero della run corrente.

Ogni esperimento è stato impostato per terminare automaticamente dopo 5000 secondi di tempo simulato.

### 4.1 Configurazione 1

I parametri in questa configurazione hanno assunto i seguenti valori:

Parametro	Valore	.ini
$\lambda_1$ (interarrivo 1)	1.0, 2.0, 4.0, 10.0, 20.0	exponential(\${1.0, 0.5, 0.25, 0.1, 0.05}s)
$\lambda_2$ (interarrivo 2)	2.0, 1.0	exponential(\${0.5, 1}s)
$\mu_1$ (service time 1)	[0.11, 0.55]	uniform(0.11s,0.55s)
$\mu_2$ (service time 2)	[0.1, 0.4]	uniform(0.1s,0.4s)
$\alpha$ (switch-over time)	[0.1, 0.3]	uniform(0.1s,0.3s)
$K$ (threshold coda 1)	10	10
Capacità coda 1	10	10
$N$ (threshold coda 2)	3	3
Capacità coda 2	infinita	-1

Tabella 4.1: Configurazione 1

### 4.2 Configurazione 2

I parametri in questa configurazione hanno assunto i seguenti valori:

Parametro	Valore	.ini
$\lambda_1$ (interarrivo 2)	1.0, 2.0, 3.0, 3.5	exponential(\${1.0,0.5,0.33,0.28}s)
$\lambda_2$ (interarrivo 1)	2.0, 1.5	exponential(\${0.5, 0.66}s)
$\mu_1$ (service time 1)	[0.22, 0.44]	uniform(0.22s,0.44s)
$\mu_2$ (service time 2)	[0.2,0.3]	uniform(0.2s,0.3s)
$\alpha$ (switch-over time)	[0.1, 0.3]	uniform(0.1s,0.3s)
$K$ (threshold coda 1)	10	10
Capacità coda 1	10	10
$N$ (threshold coda 2)	3	3
Capacità coda 2	infinita	-1

Tabella 4.2: Configurazione 2

## 5 Analisi dei risultati

In questa sezione sarà illustrato il processo di analisi dei risultati ottenuti.

### 5.1 Approccio generale

Per eseguire l'analisi dei dati è stato utilizzato il linguaggio Python su Jupyter notebook con librerie `pandas`, `numpy`, `matplotlib`, `scipy` e `math`.

Era richiesto di calcolare stima puntuale e intervalli di confidenza di:

1. Tempo medio di permanenza nel sistema dei job prelevati dalle singole code;
2. Numero medio di utenti nelle singole code;
3. Numero medio di utenti persi.

Inoltre era richiesto di studiare e discutere il problema del transiente iniziale.

Per prima cosa sono stati convertiti scalari e vettori in file .csv:

```
opp_scavetool.exe x *.sca -o scalars.csv  
opp_scavetool.exe x *.vec -o vectors.csv
```

In seguito gli scalari sono stati trasformati in numpy array da cui sono stati calcolati media e intervalli di confidenza al 95% mediante le funzioni `numpy.mean()` e `scipy.stats.norm.interval()`.

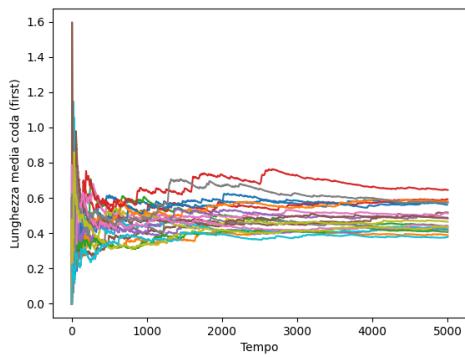
I vettori invece sono stati analizzati secondo quanto indicato nella documentazione (<https://docs.omnetpp.org/tutorials/pandas/>), ovvero tramite le funzioni `running_avg(x)` e `running_timeavg(t,x)` definite nei paragrafi 11 e 12.

### 5.2 Analisi dei vettori

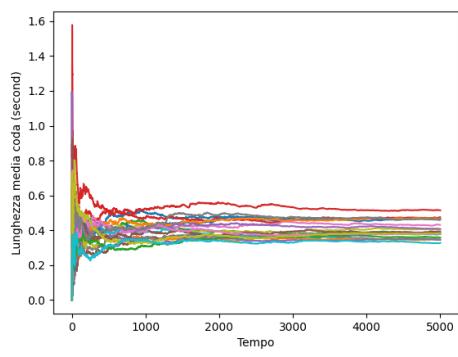
Per ogni configurazione e per ogni combinazione dei valori  $\lambda_1$  e  $\lambda_2$  sono stati mostrati come grafici i vettori `queueLength:vector` e `lifeTime:vector` forniti dai moduli `PassiveQueue` ("first" e "second") e `Sink` ("sink" e "sink1"). Di seguito saranno mostrati i valori della **Configurazione 1** al variare dei valori dei  $\lambda$  (tempi di interarrivo esponenziali di media data).

I tempi di servizio sono uniformemente distribuiti nell'intervallo **[0.11, 0.55]** per la coda "first" e **[0.1, 0.4]** per la coda "second".

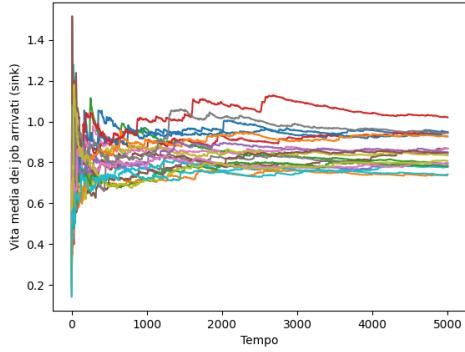
### 5.2.1 $\lambda_1 = 1.0, \lambda_2 = 1.0$



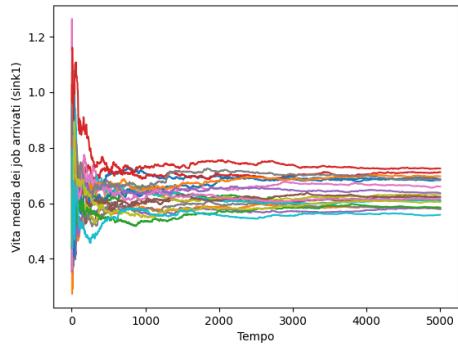
**Figura 5.1:** Conf. 1 - Lunghezza first



**Figura 5.2:** Conf. 1 - Lunghezza second

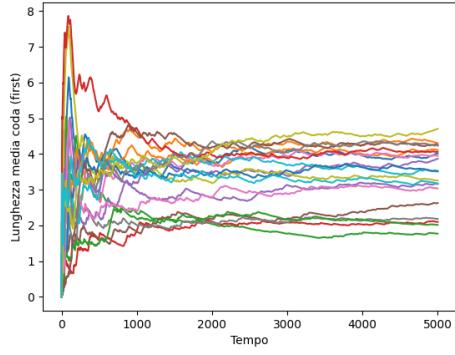


**Figura 5.3:** Conf. 1 - Lifetime first

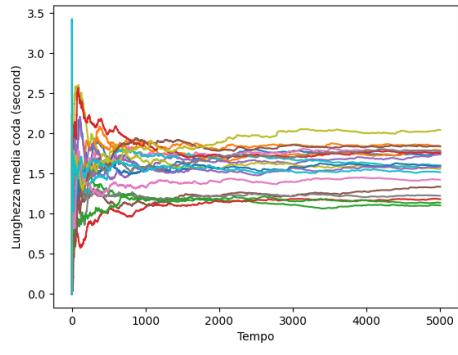


**Figura 5.4:** Conf. 1 - Lifetime second

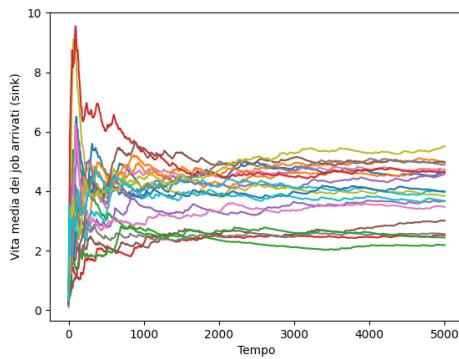
### 5.2.2 $\lambda_1 = 1.0, \lambda_2 = 2.0$



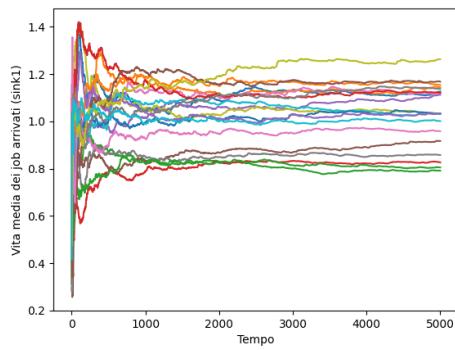
**Figura 5.5:** Conf. 1 - Lunghezza first



**Figura 5.6:** Conf. 1 - Lunghezza second

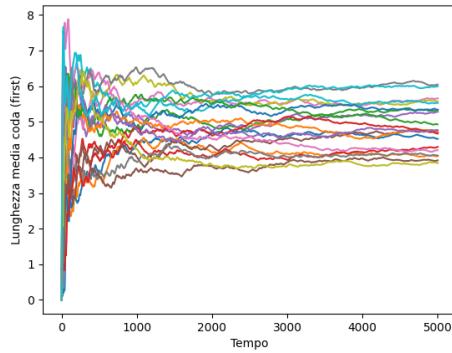


**Figura 5.7:** Conf. 1 - Lifetime first

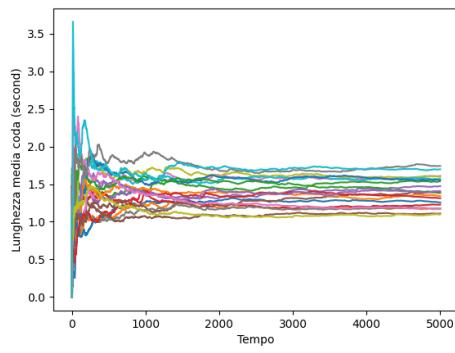


**Figura 5.8:** Conf. 1 - Lifetime second

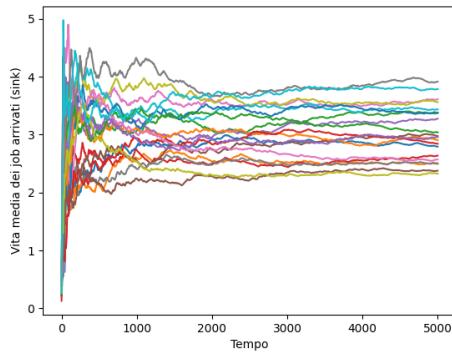
### 5.2.3 $\lambda_1 = 2.0, \lambda_2 = 1.0$



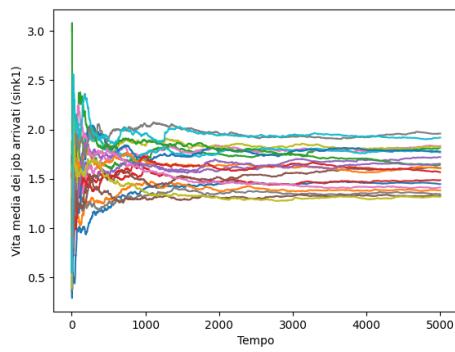
**Figura 5.9:** Conf. 1 - Lunghezza first



**Figura 5.10:** Conf. 1 - Lunghezza second

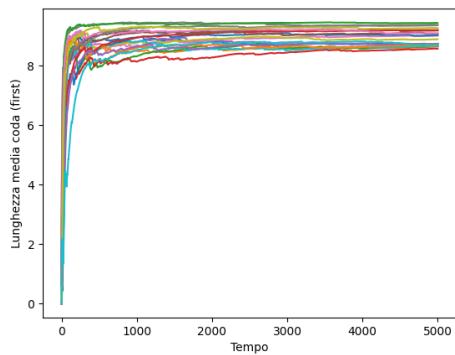


**Figura 5.11:** Conf. 1 - Lifetime first

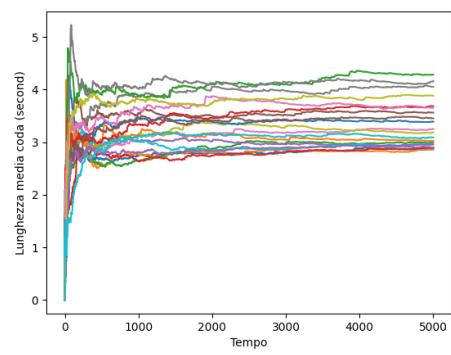


**Figura 5.12:** Conf. 1 - Lifetime second

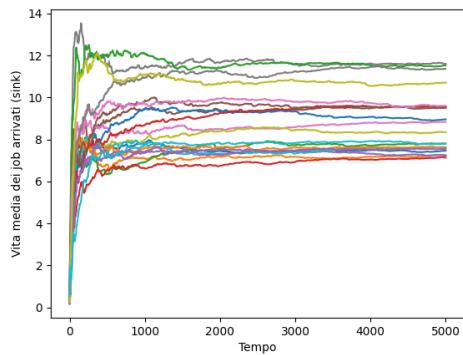
#### 5.2.4 $\lambda_1 = 2.0, \lambda_2 = 2.0$



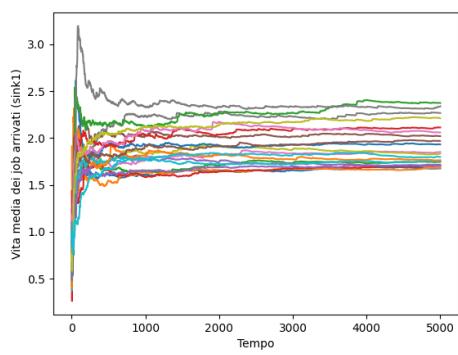
**Figura 5.13:** Conf. 1 - Lunghezza first



**Figura 5.14:** Conf. 1 - Lunghezza second

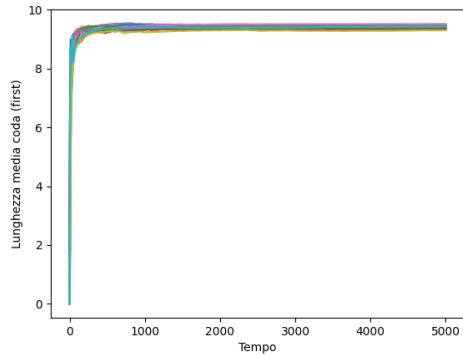


**Figura 5.15:** Conf. 1 - Lifetime first

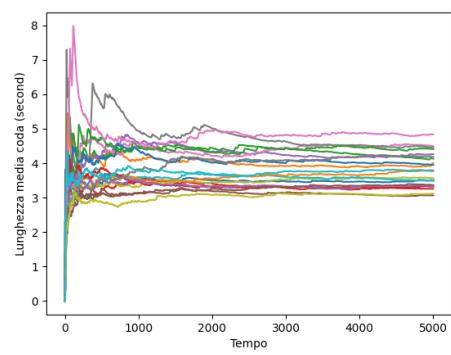


**Figura 5.16:** Conf. 1 - Lifetime second

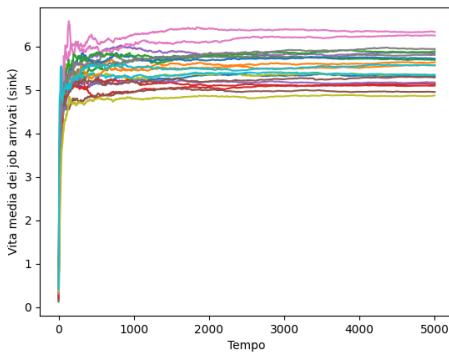
#### 5.2.5 $\lambda_1 = 4.0, \lambda_2 = 1.0$



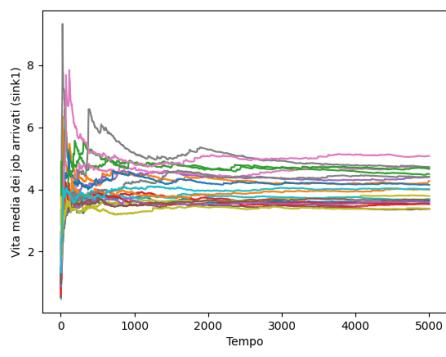
**Figura 5.17:** Conf. 1 - Lunghezza first



**Figura 5.18:** Conf. 1 - Lunghezza second

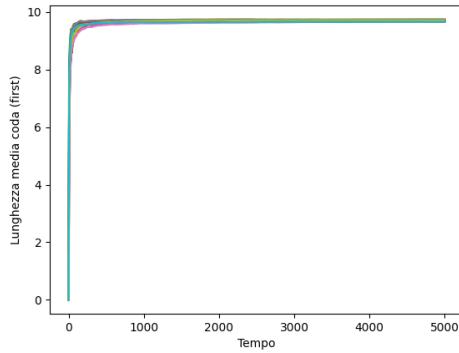


**Figura 5.19:** Conf. 1 - Lifetime first

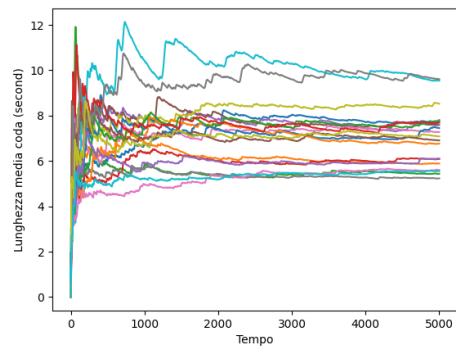


**Figura 5.20:** Conf. 1 - Lifetime second

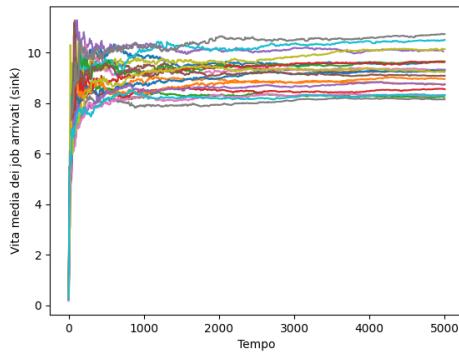
### 5.2.6 $\lambda_1 = 4.0, \lambda_2 = 2.0$



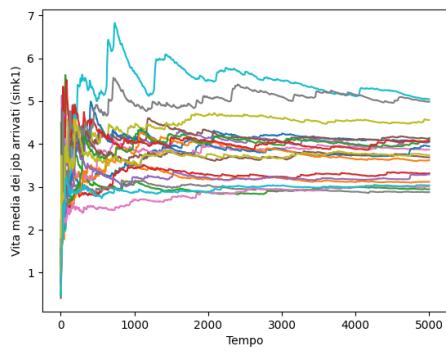
**Figura 5.21:** Conf. 1 - Lunghezza first



**Figura 5.22:** Conf. 1 - Lunghezza second

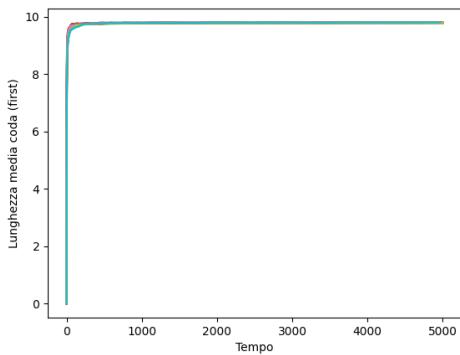


**Figura 5.23:** Conf. 1 - Lifetime first

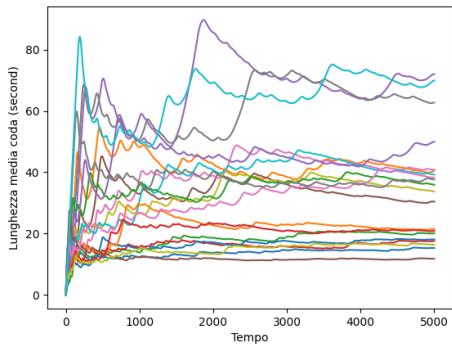


**Figura 5.24:** Conf. 1 - Lifetime second

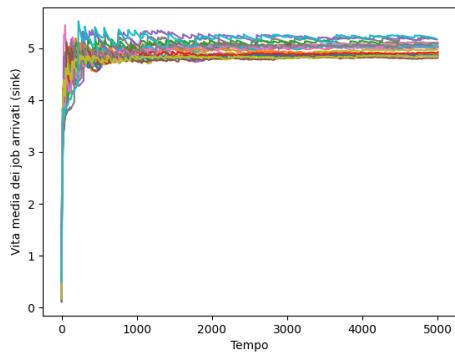
### 5.2.7 $\lambda_1 = 10.0, \lambda_2 = 1.0$



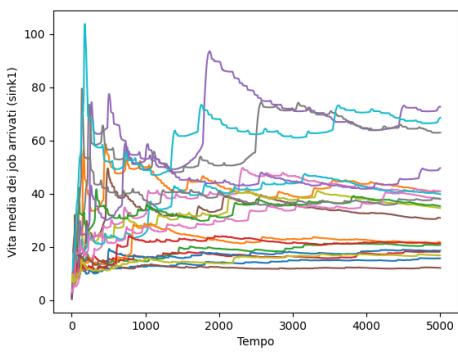
**Figura 5.25:** Conf. 1 - Lunghezza first



**Figura 5.26:** Conf. 1 - Lunghezza second

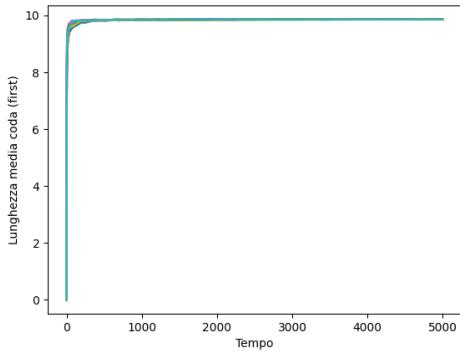


**Figura 5.27:** Conf. 1 - Lifetime first

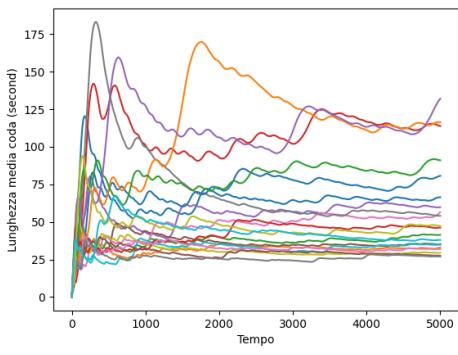


**Figura 5.28:** Conf. 1 - Lifetime second

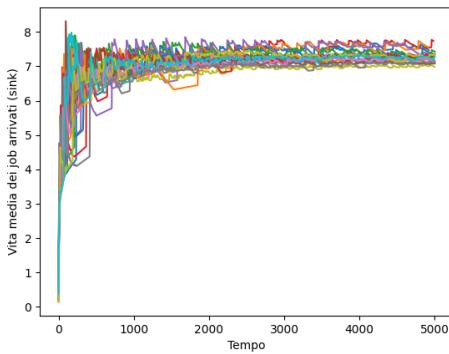
### 5.2.8 $\lambda_1 = 10.0, \lambda_2 = 2.0$



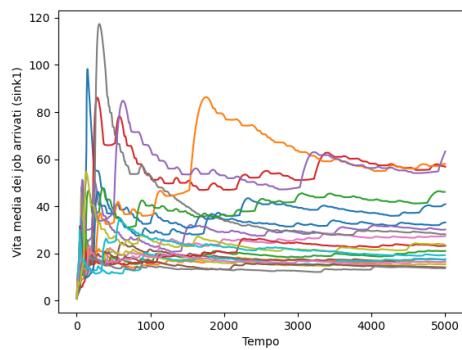
**Figura 5.29:** Conf. 1 - Lunghezza first



**Figura 5.30:** Conf. 1 - Lunghezza second

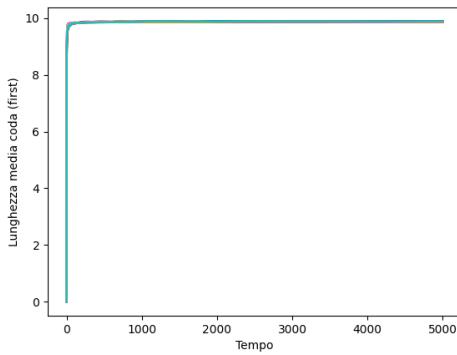


**Figura 5.31:** Conf. 1 - Lifetime first

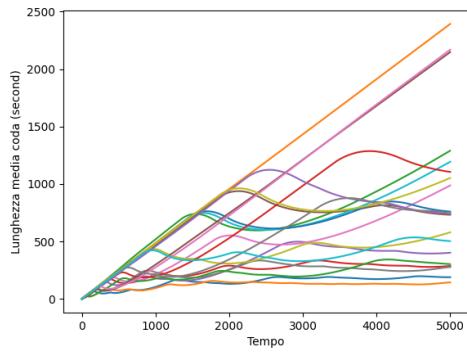


**Figura 5.32:** Conf. 1 - Lifetime second

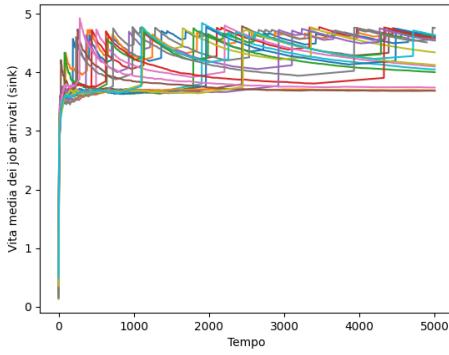
**5.2.9**  $\lambda_1 = 20.0$ ,  $\lambda_2 = 1.0$



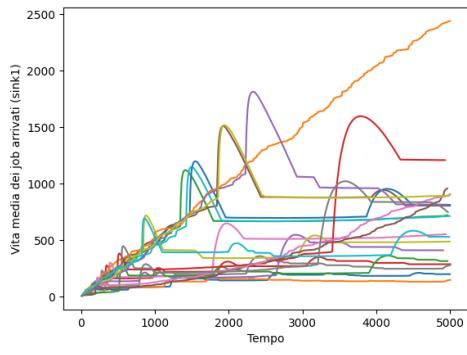
**Figura 5.33:** Conf. 1 - Lunghezza first



**Figura 5.34:** Conf. 1 - Lunghezza second

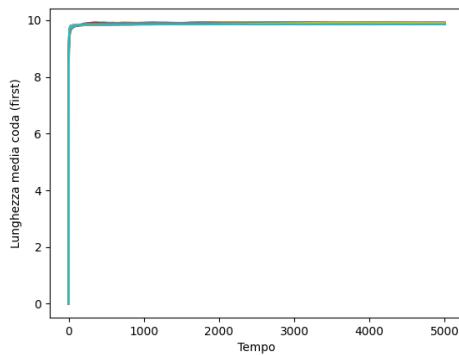


**Figura 5.35:** Conf. 1 - Lifetime first

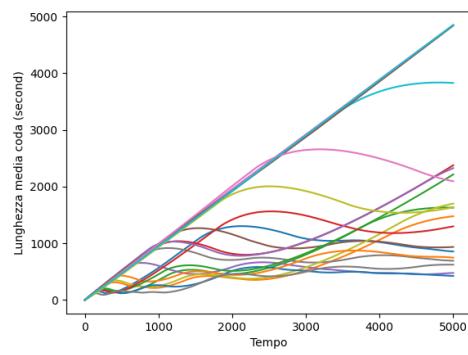


**Figura 5.36:** Conf. 1 - Lifetime second

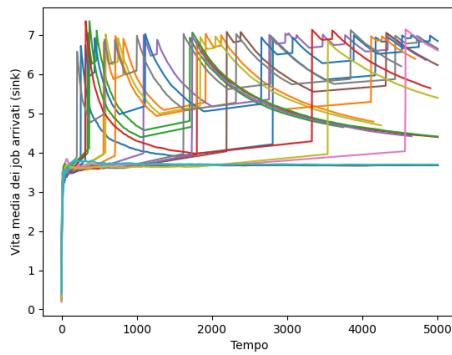
**5.2.10**  $\lambda_1 = 20.0$ ,  $\lambda_2 = 2.0$



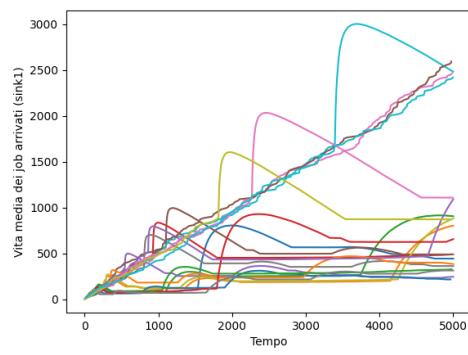
**Figura 5.37:** Conf. 1 - Lunghezza first



**Figura 5.38:** Conf. 1 - Lunghezza second



**Figura 5.39:** Conf. 1 - Lifetime first

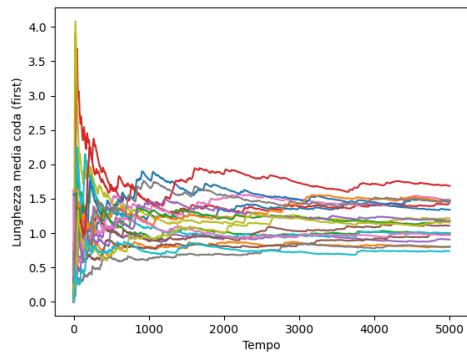


**Figura 5.40:** Conf. 1 - Lifetime second

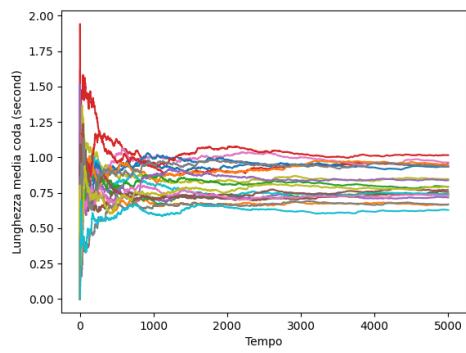
Di seguito saranno mostrati i valori della **Configurazione 2** al variare dei valori dei  $\lambda$  (tempi di interarrivo esponenziali di media data).

I tempi di servizio sono uniformemente distribuiti nell'intervallo [0.22, 0.44] per la coda "first" e [0.2, 0.3] per la coda "second".

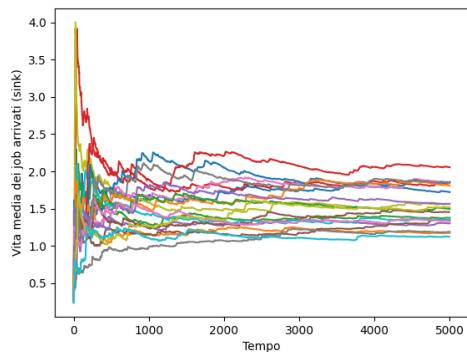
### 5.2.11 $\lambda_1 = 1.0, \lambda_2 = 1.5$



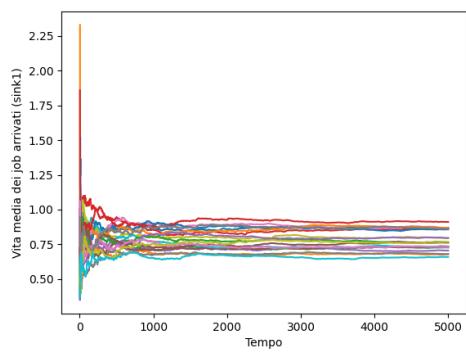
**Figura 5.41:** Conf. 2 - Lunghezza first



**Figura 5.42:** Conf. 2 - Lunghezza second

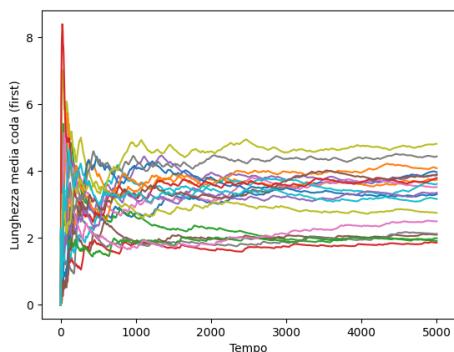


**Figura 5.43:** Conf. 2 - Lifetime first

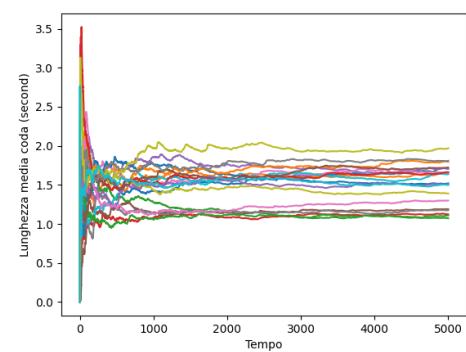


**Figura 5.44:** Conf. 2 - Lifetime second

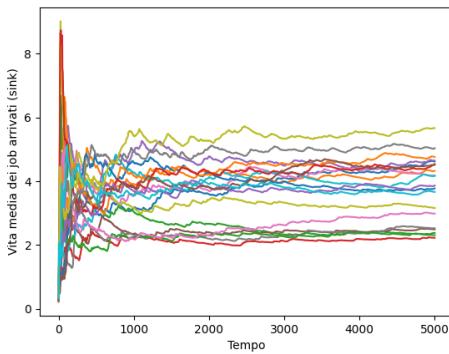
### 5.2.12 $\lambda_1 = 1.0, \lambda_2 = 2.0$



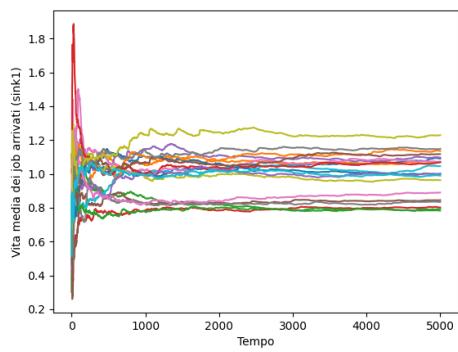
**Figura 5.45:** Conf. 2 - Lunghezza first



**Figura 5.46:** Conf. 2 - Lunghezza second

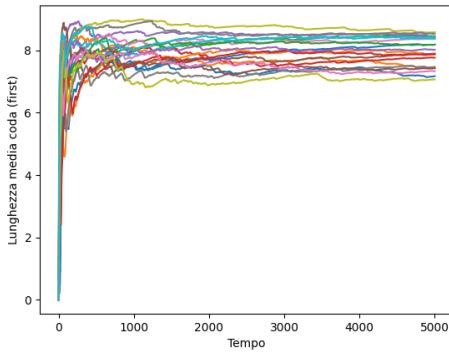


**Figura 5.47:** Conf. 2 - Lifetime first

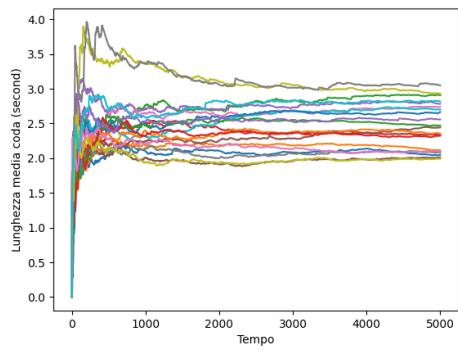


**Figura 5.48:** Conf. 2 - Lifetime second

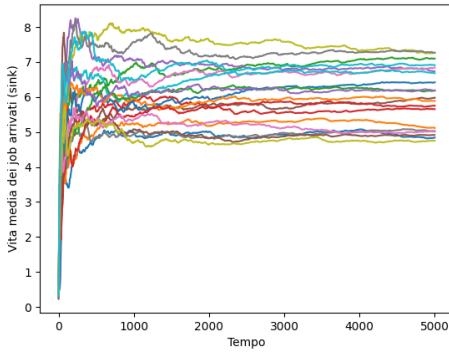
**5.2.13**  $\lambda_1 = 2.0$ ,  $\lambda_2 = 1.5$



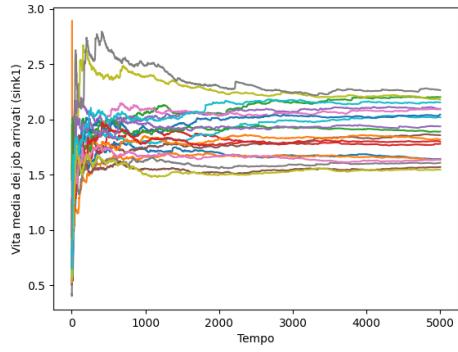
**Figura 5.49:** Conf. 2 - Lunghezza first



**Figura 5.50:** Conf. 2 - Lunghezza second

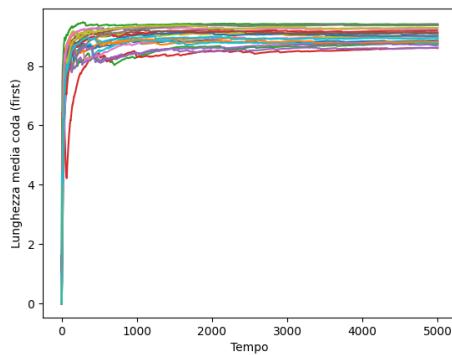


**Figura 5.51:** Conf. 2 - Lifetime first

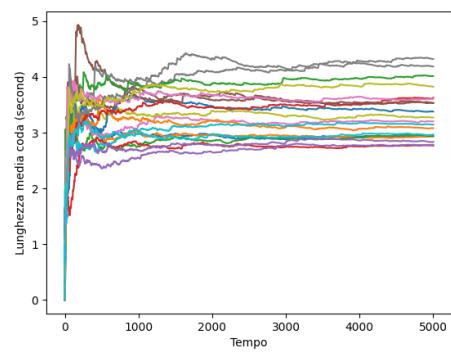


**Figura 5.52:** Conf. 2 - Lifetime second

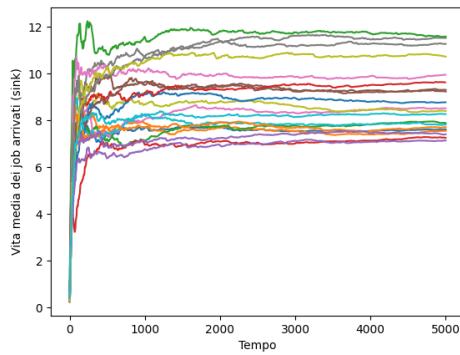
**5.2.14**  $\lambda_1 = 2.0, \lambda_2 = 2.0$



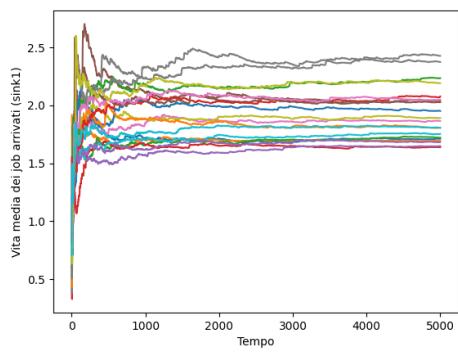
**Figura 5.53:** Conf. 2 - Lunghezza first



**Figura 5.54:** Conf. 2 - Lunghezza second

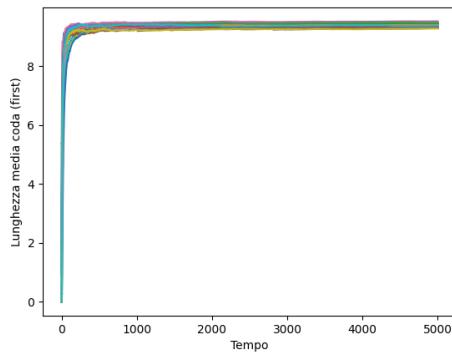


**Figura 5.55:** Conf. 2 - Lifetime first

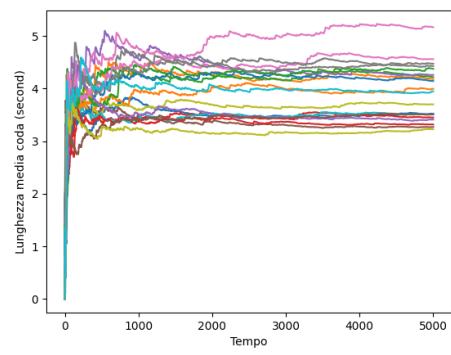


**Figura 5.56:** Conf. 2 - Lifetime second

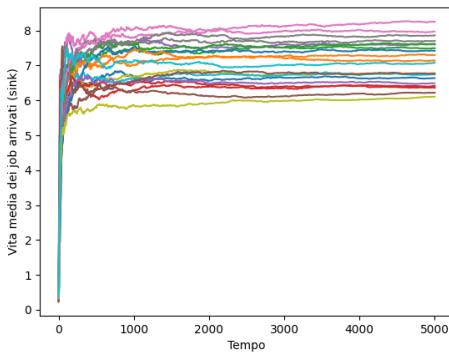
**5.2.15**  $\lambda_1 = 3.0, \lambda_2 = 1.5$



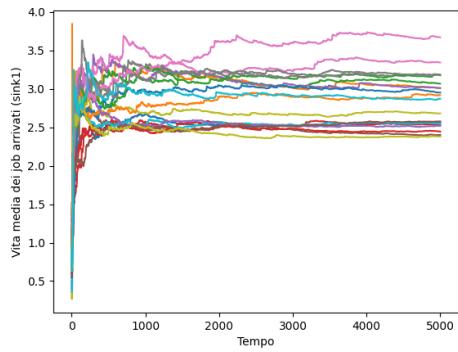
**Figura 5.57:** Conf. 2 - Lunghezza first



**Figura 5.58:** Conf. 2 - Lunghezza second

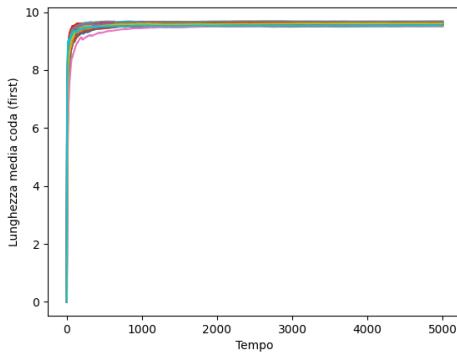


**Figura 5.59:** Conf. 2 - Lifetime first

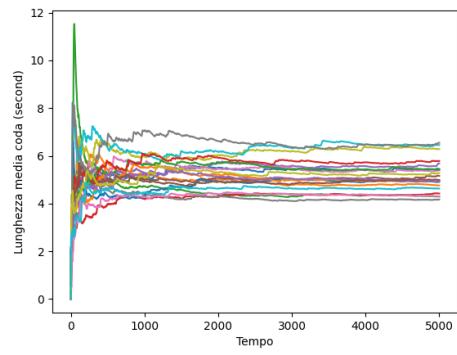


**Figura 5.60:** Conf. 2 - Lifetime second

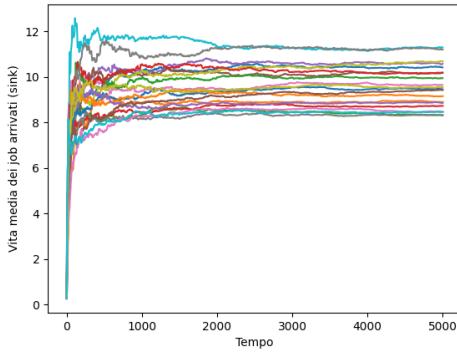
**5.2.16**  $\lambda_1 = 3.0$ ,  $\lambda_2 = 2.0$



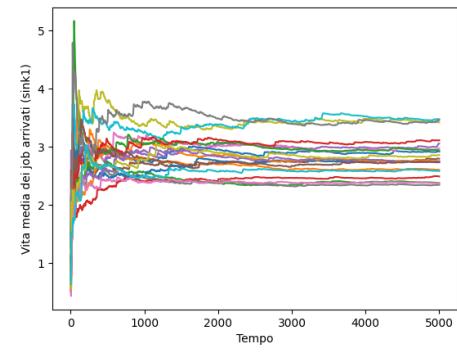
**Figura 5.61:** Conf. 2 - Lunghezza first



**Figura 5.62:** Conf. 2 - Lunghezza second

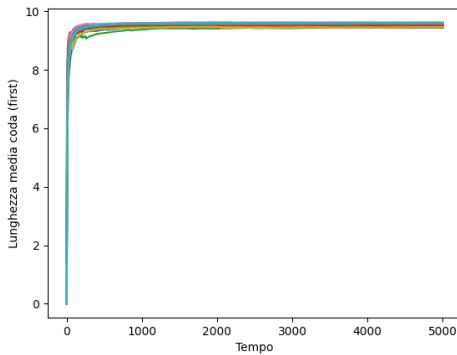


**Figura 5.63:** Conf. 2 - Lifetime first

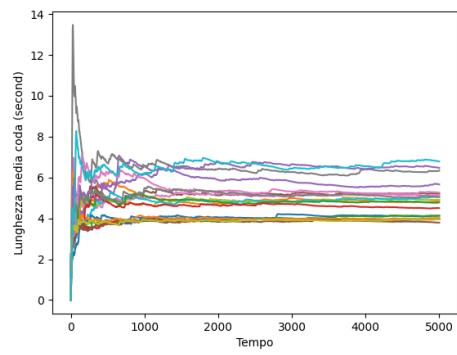


**Figura 5.64:** Conf. 2 - Lifetime second

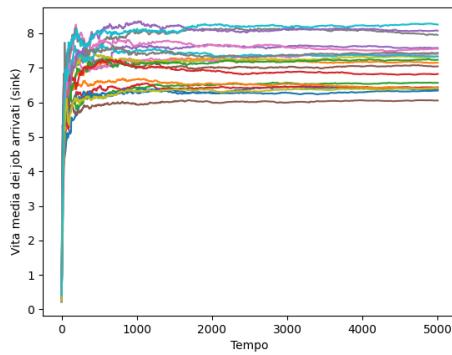
**5.2.17**  $\lambda_1 = 3.5, \lambda_2 = 1.5$



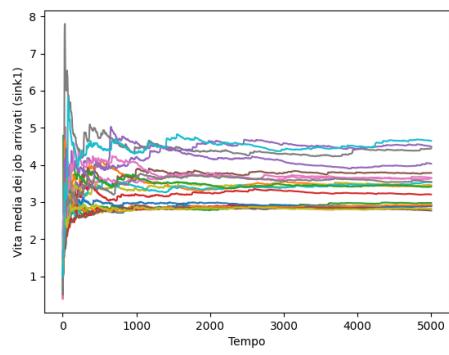
**Figura 5.65:** Conf. 2 - Lunghezza first



**Figura 5.66:** Conf. 2 - Lunghezza second

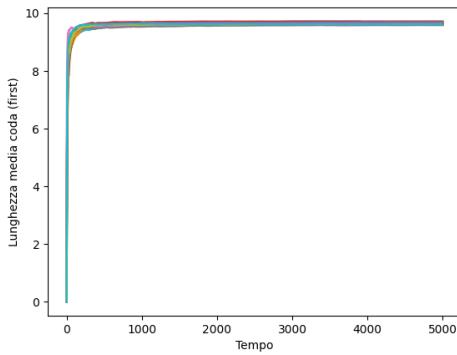


**Figura 5.67:** Conf. 2 - Lifetime first

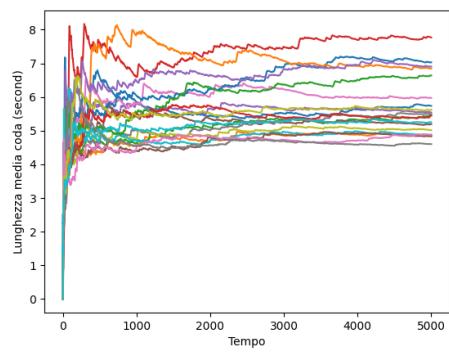


**Figura 5.68:** Conf. 2 - Lifetime second

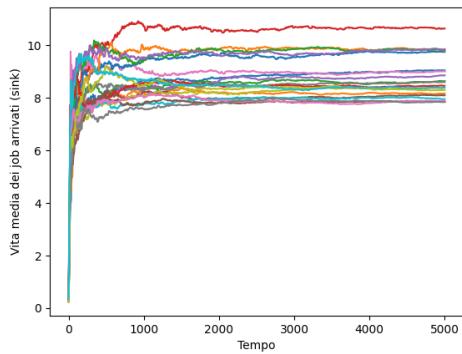
**5.2.18**  $\lambda_1 = 3.5, \lambda_2 = 2.0$



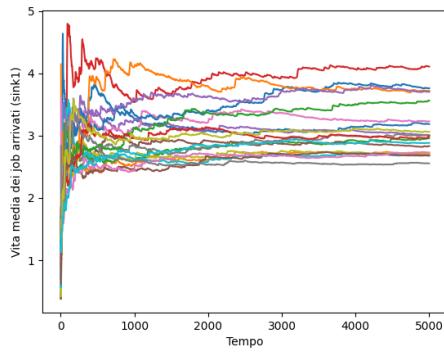
**Figura 5.69:** Conf. 2 - Lunghezza first



**Figura 5.70:** Conf. 2 - Lunghezza second



**Figura 5.71:** Conf. 2 - Lifetime first



**Figura 5.72:** Conf. 2 - Lifetime second

### 5.3 Transiente iniziale

Osservando i grafici dei vettori è possibile notare un punto di stabilizzazione dopo circa 1000s di tempo simulato.

È stato quindi inserito un `warmup-period = 1000s` in modo da eliminare il transiente iniziale prima di procedere alle analisi successive.

## 5.4 Tempo medio di permanenza nel sistema dei job

Per questa stima è stata calcolata la media del valore `queueingTime:mean` delle code per ogni run.

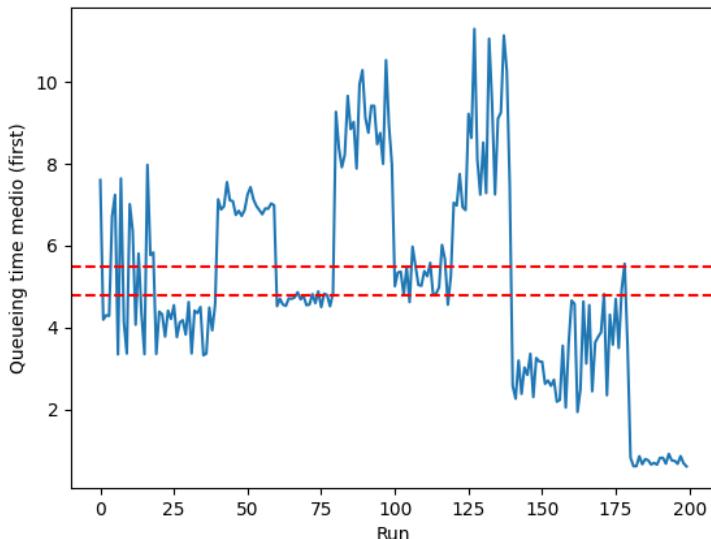
### 5.4.1 Configurazione 1

I risultati sono riportati di seguito:

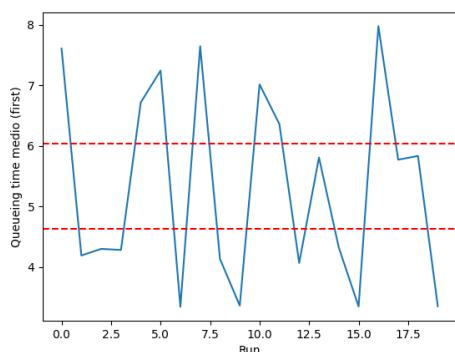
Modulo	Media	Intervallo di confidenza
Network.first	5.12 s	(4.77, 5.47)
Network.second	209.77 s	(131.87, 287.66)
Entrambe le code	107.44 s	(67.23, 147.66)

**Tabella 5.1:** Configurazione 1 - tempo medio di permanenza

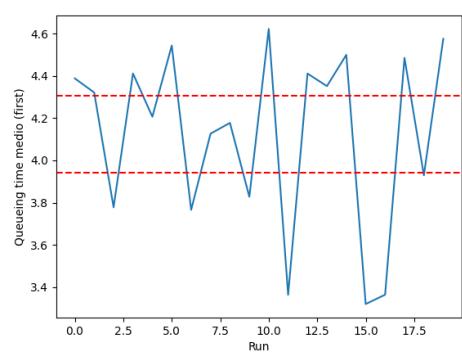
Di seguito i grafici che mostrano il tempo medio di permanenza per ogni coda in ogni run:



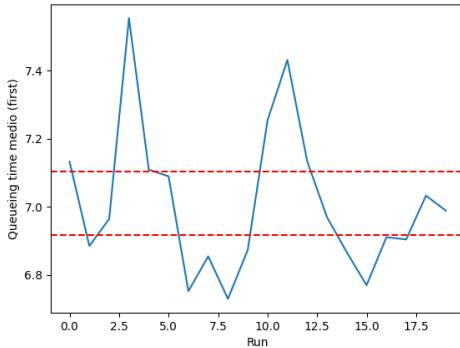
**Figura 5.73:** Configurazione 1 - tempo medio di permanenza (coda "first")



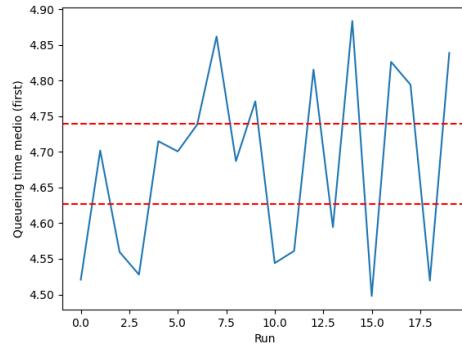
**Figura 5.74:** Conf. 1 - 0.05, 0.5



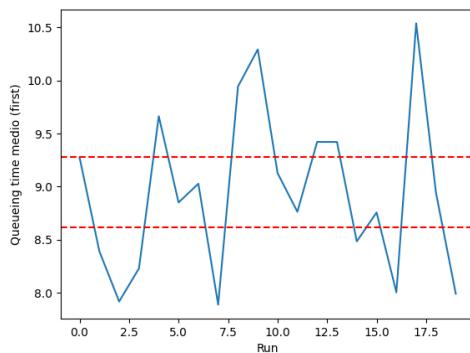
**Figura 5.75:** Conf. 1 - 0.05, 1



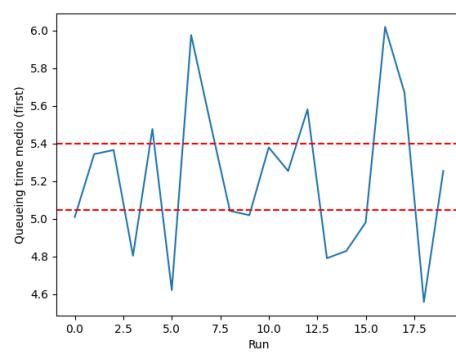
**Figura 5.76:** Conf. 1 - 0.1, 0.5



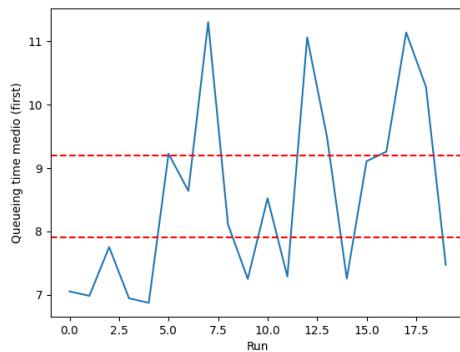
**Figura 5.77:** Conf. 1 - 0.1, 1.0



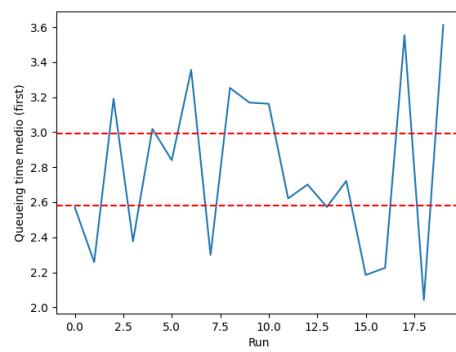
**Figura 5.78:** Conf. 1 - 0.25, 0.5



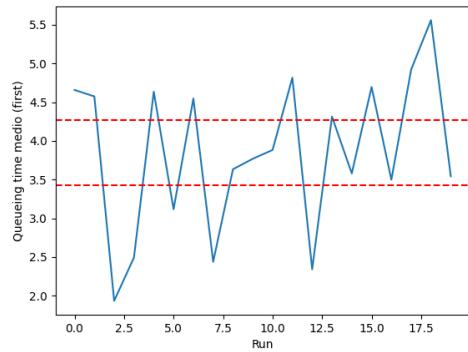
**Figura 5.79:** Conf. 1 - 0.25, 1.0



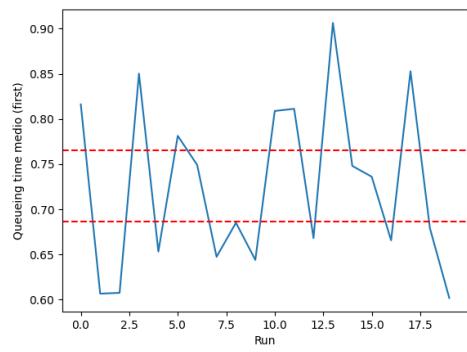
**Figura 5.80:** Conf. 1 - 0.5, 0.5



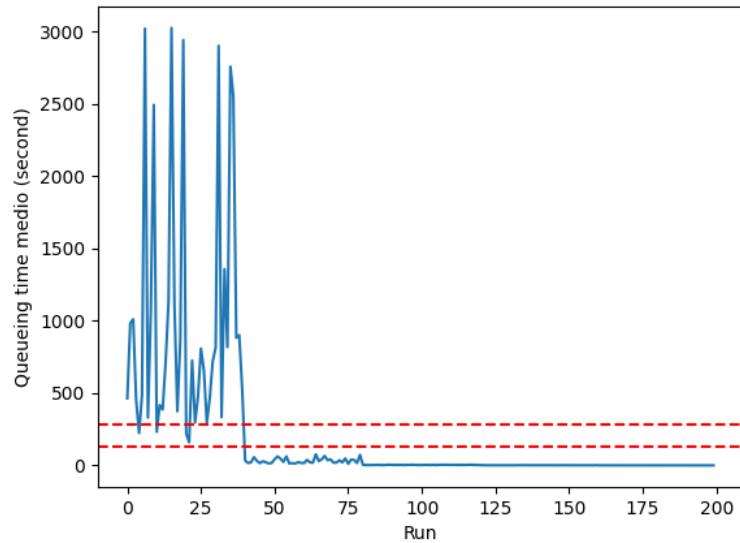
**Figura 5.81:** Conf. 1 - 0.5, 1.0



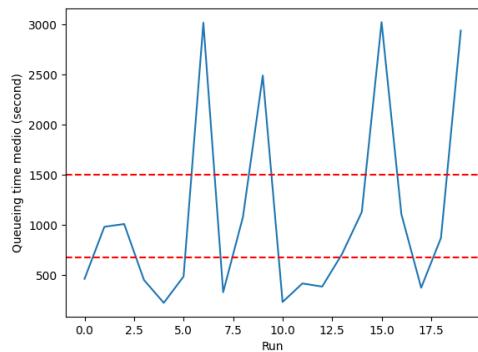
**Figura 5.82:** Conf. 1 - 1.0, 0.5



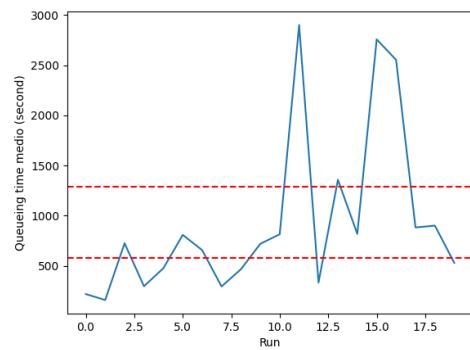
**Figura 5.83:** Conf. 1 - 1.0, 1.0



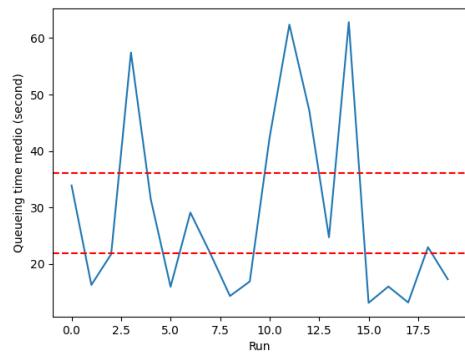
**Figura 5.84:** Configurazione 1 - tempo medio di permanenza (coda "second")



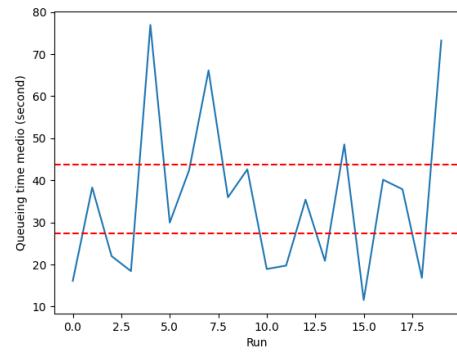
**Figura 5.85:** Conf. 1 - 0.05, 0.5



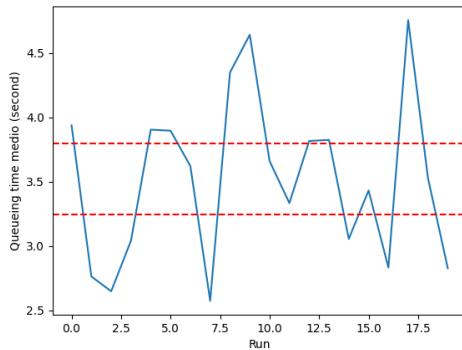
**Figura 5.86:** Conf. 1 - 0.05, 1



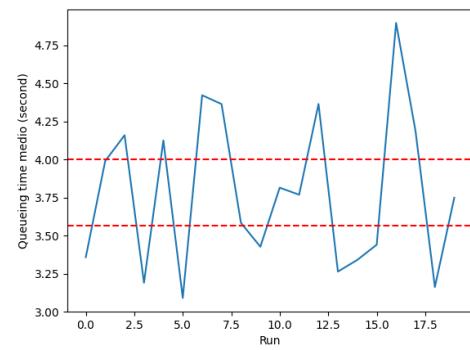
**Figura 5.87:** Conf. 1 - 0.1, 0.5



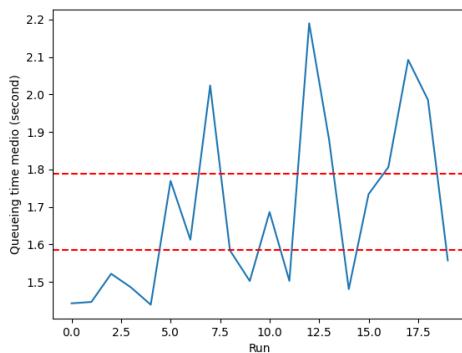
**Figura 5.88:** Conf. 1 - 0.1, 1.0



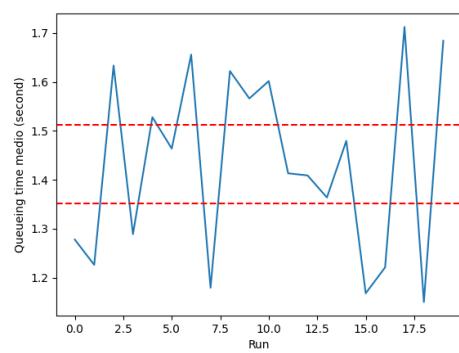
**Figura 5.89:** Conf. 1 - 0.25, 0.5



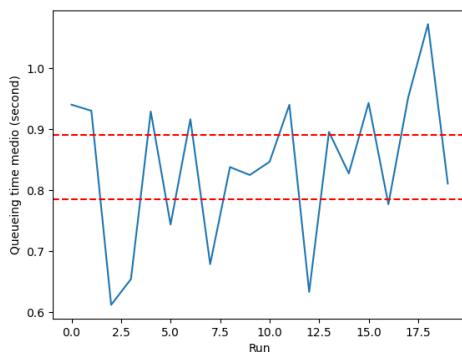
**Figura 5.90:** Conf. 1 - 0.25, 1.0



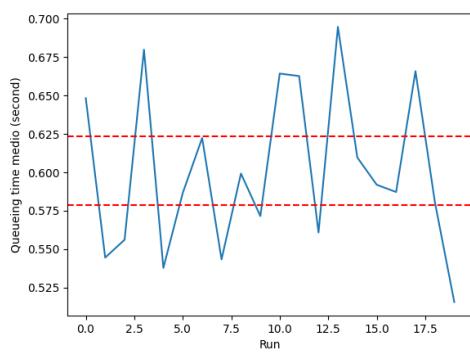
**Figura 5.91:** Conf. 1 - 0.5, 0.5



**Figura 5.92:** Conf. 1 - 0.5, 1.0



**Figura 5.93:** Conf. 1 - 1.0, 0.5



**Figura 5.94:** Conf. 1 - 1.0, 1.0

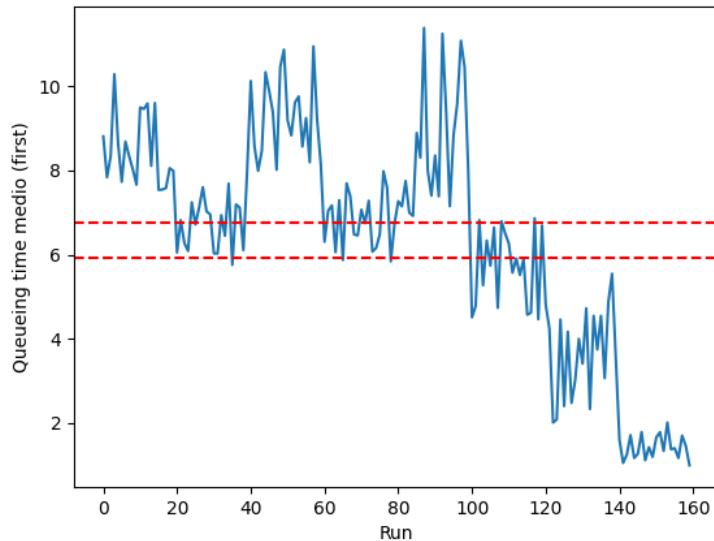
#### 5.4.2 Configurazione 2

I risultati sono riportati di seguito:

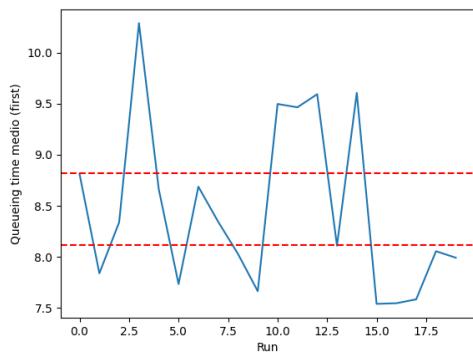
Modulo	Media	Intervallo di confidenza
Network.first	6.34 s	(5.93, 6.75)
Network.second	2.01 s	(1.86, 2.16)
Entrambe le code	4.17 s	(3.85, 4.50)

**Tabella 5.2:** Configurazione 2 - tempo medio di permanenza

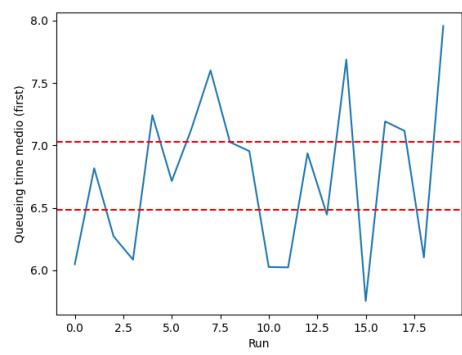
Di seguito i grafici che mostrano il tempo medio di permanenza per ogni coda in ogni run:



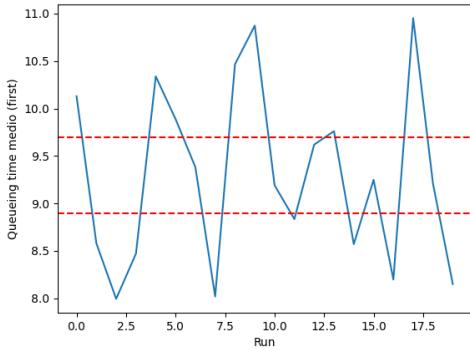
**Figura 5.95:** Configurazione 2 - tempo medio di permanenza (coda "first")



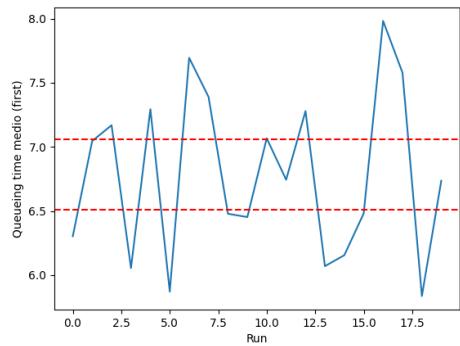
**Figura 5.96:** Conf. 2 - 0.28, 0.5



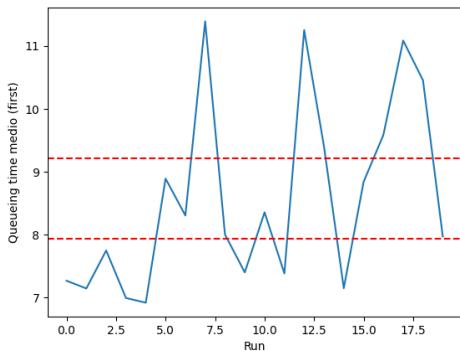
**Figura 5.97:** Conf. 2 - 0.28, 0.66



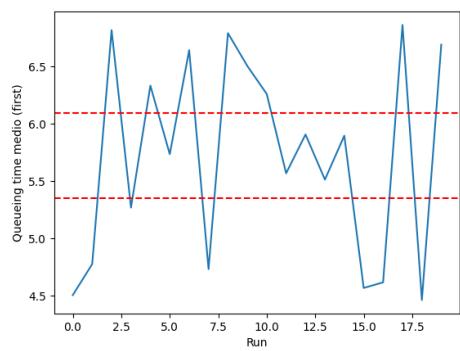
**Figura 5.98:** Conf. 2 - 0.33, 0.5



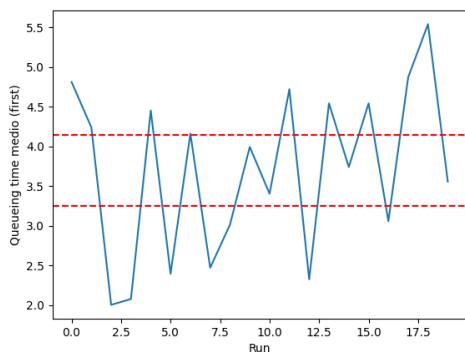
**Figura 5.99:** Conf. 2 - 0.33, 0.66



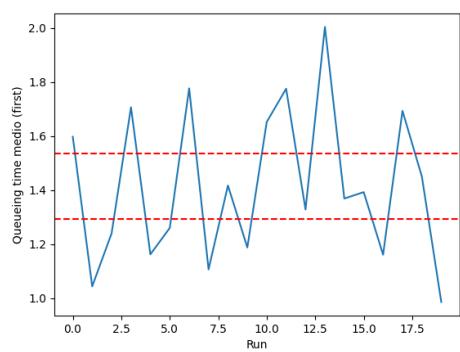
**Figura 5.100:** Conf. 2 - 0.5, 0.5



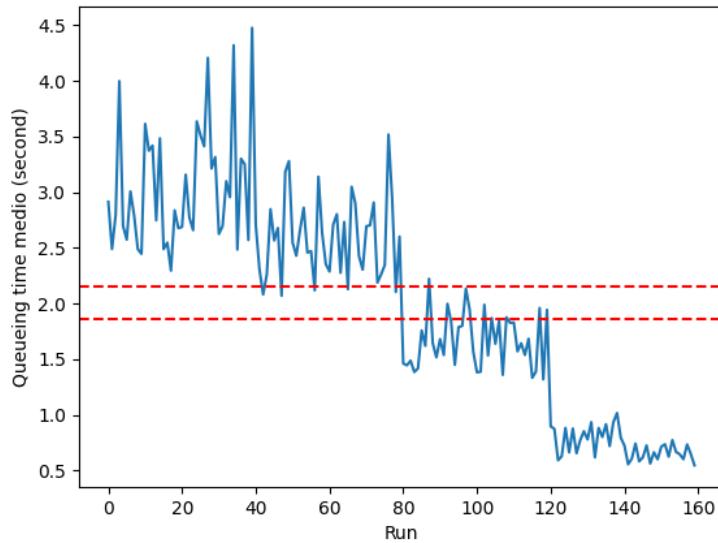
**Figura 5.101:** Conf. 2 - 0.5, 0.66



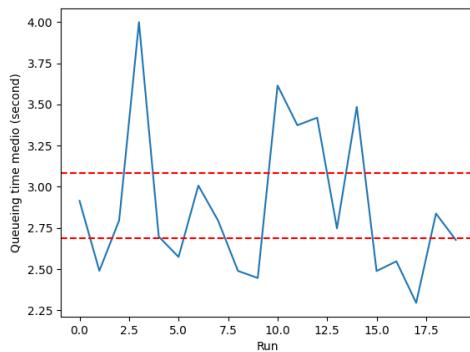
**Figura 5.102:** Conf. 2 - 1.0, 0.5



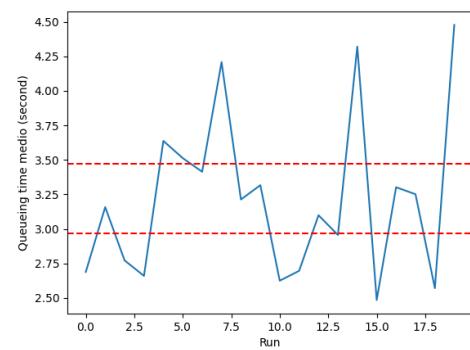
**Figura 5.103:** Conf. 2 - 1.0, 0.66



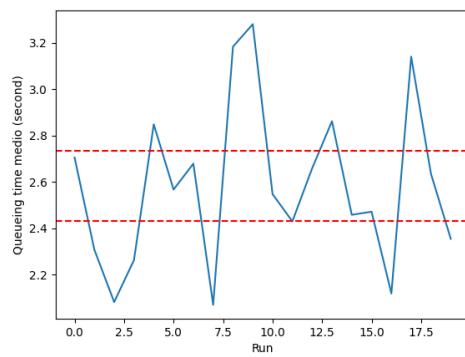
**Figura 5.104:** Configurazione 2 - tempo medio di permanenza (coda "second")



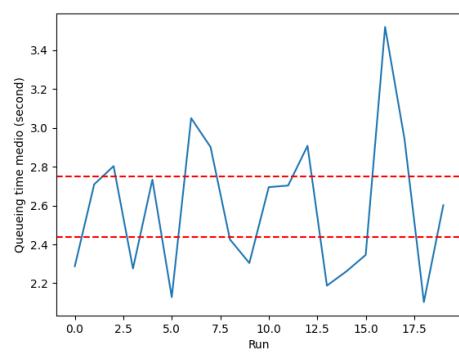
**Figura 5.105:** Conf. 2 - 0.28, 0.5



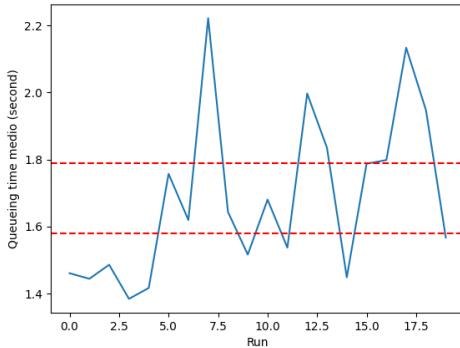
**Figura 5.106:** Conf. 2 - 0.28, 0.66



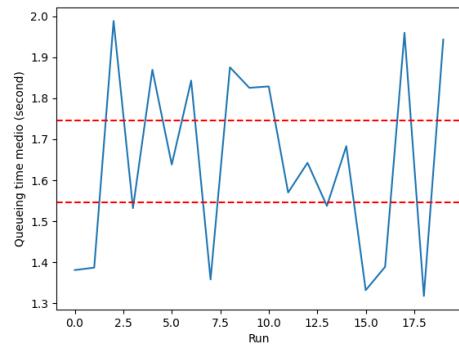
**Figura 5.107:** Conf. 2 - 0.33, 0.5



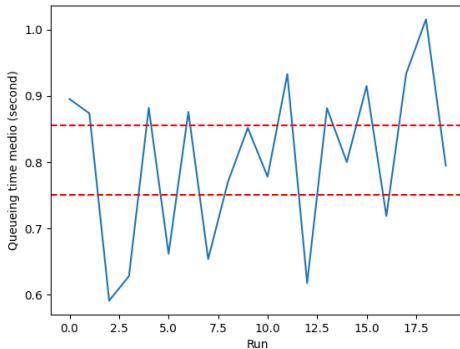
**Figura 5.108:** Conf. 2 - 0.33, 0.66



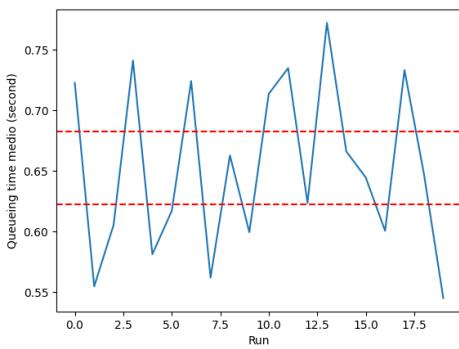
**Figura 5.109:** Conf. 2 - 0.5, 0.5



**Figura 5.110:** Conf. 2 - 0.5, 0.66



**Figura 5.111:** Conf. 2 - 1.0, 0.5



**Figura 5.112:** Conf. 2 - 1.0, 0.66

## 5.5 Numero medio di utenti nelle singole code

Per questa stima è stata calcolata la media del valore `queueLength:timeavg` delle code per ogni run.

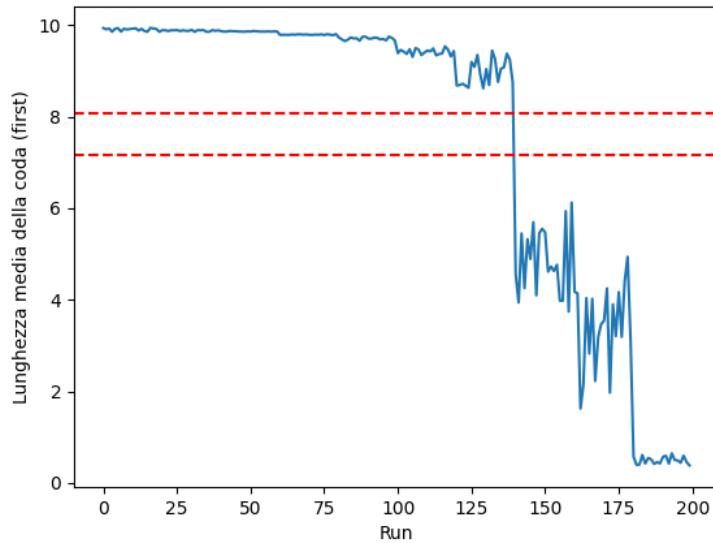
### 5.5.1 Configurazione 1

I risultati sono riportati di seguito:

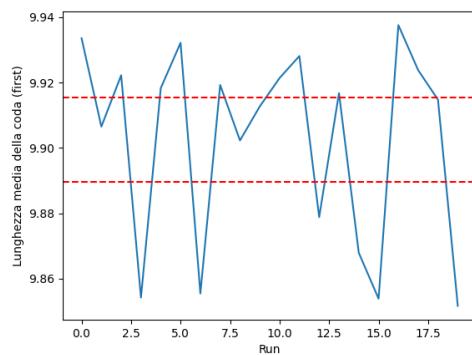
Modulo	Media	Intervallo di confidenza
Network.first	7.62 (8)	(7.18, 8.08)
Network.second	348.47 (348)	(216.75, 480,19)
Entrambe le code	178.05 (178)	(110.10, 246.00)

**Tabella 5.3:** Configurazione 1 - numero medio di utenti

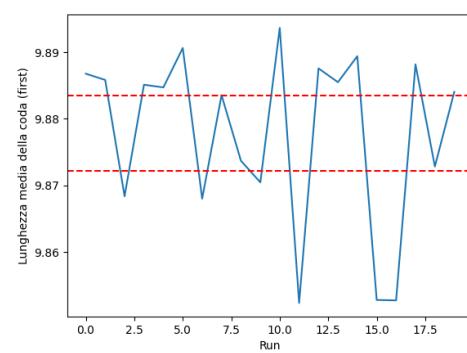
Di seguito i grafici che mostrano il numero medio di utenti per ogni coda in ogni run:



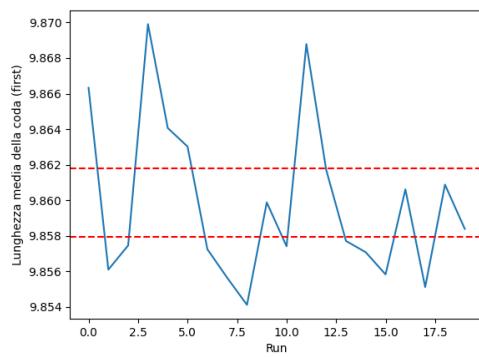
**Figura 5.113:** Configurazione 1 - numero medio di utenti (coda "first")



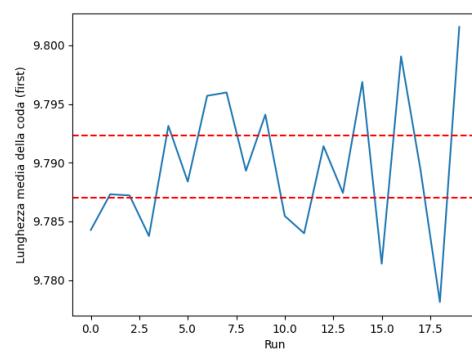
**Figura 5.114:** Conf. 1 - 0.05, 0.5



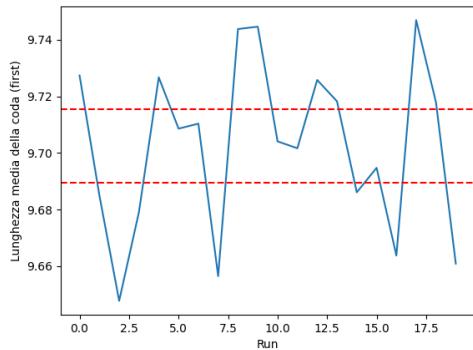
**Figura 5.115:** Conf. 1 - 0.05, 1



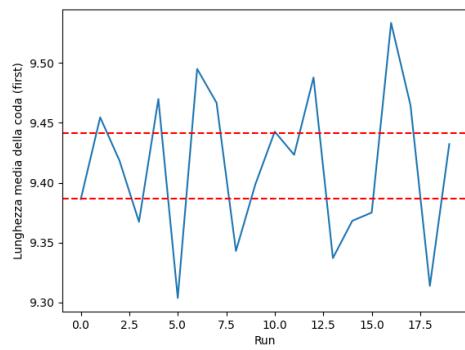
**Figura 5.116:** Conf. 1 - 0.1, 0.5



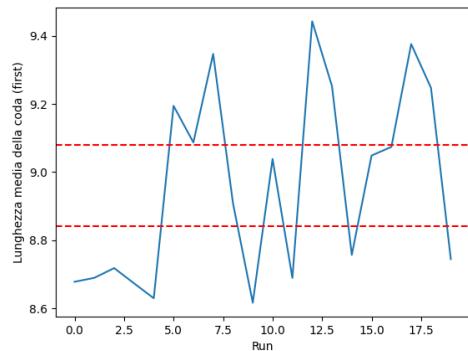
**Figura 5.117:** Conf. 1 - 0.1, 1.0



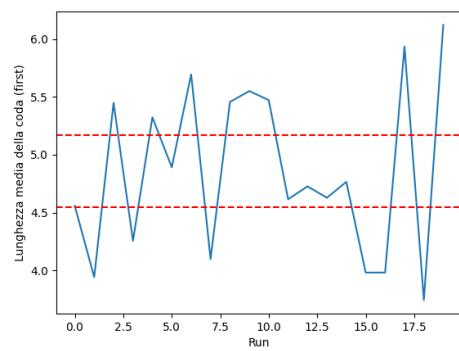
**Figura 5.118:** Conf. 1 - 0.25, 0.5



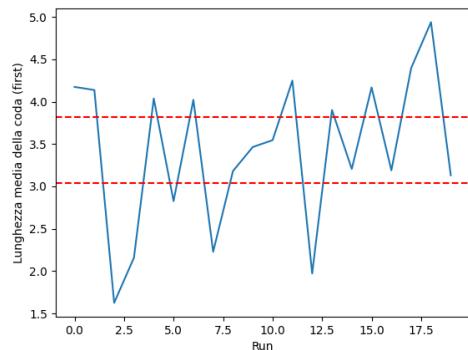
**Figura 5.119:** Conf. 1 - 0.25, 1.0



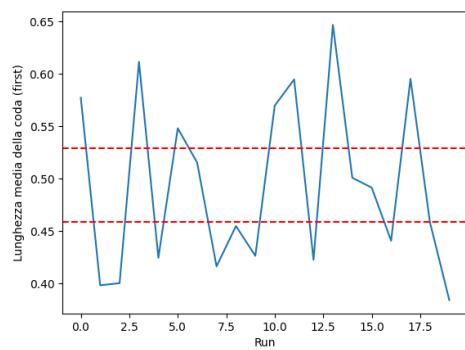
**Figura 5.120:** Conf. 1 - 0.5, 0.5



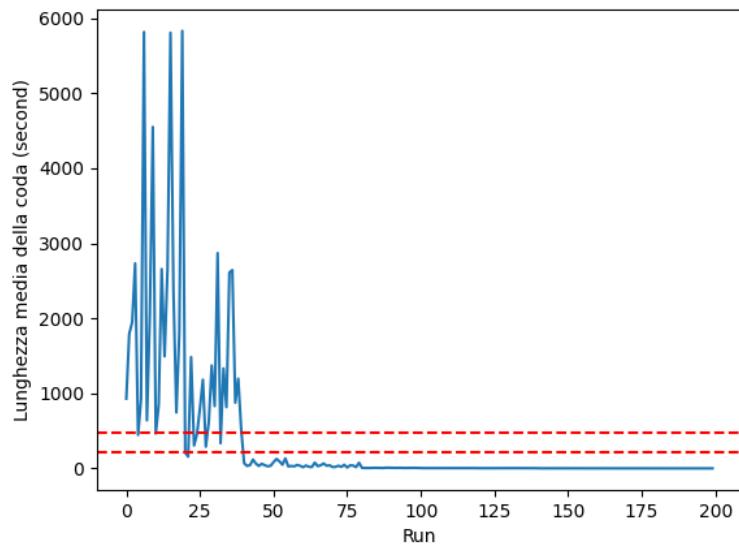
**Figura 5.121:** Conf. 1 - 0.5, 1.0



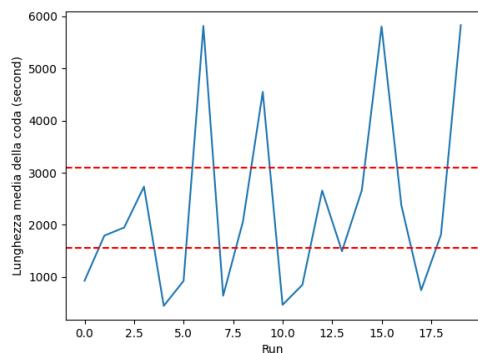
**Figura 5.122:** Conf. 1 - 1.0, 0.5



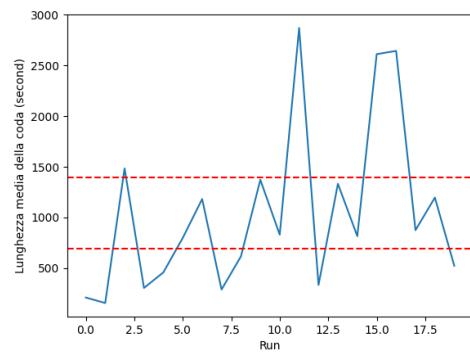
**Figura 5.123:** Conf. 1 - 1.0, 1.0



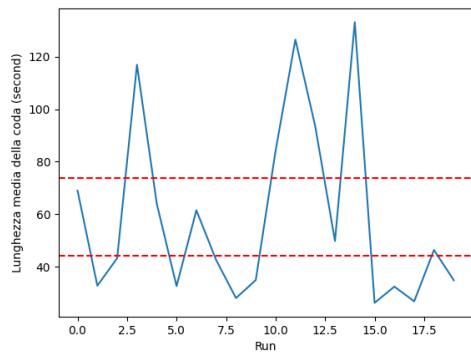
**Figura 5.124:** Configurazione 1 - numero medio di utenti (coda "second")



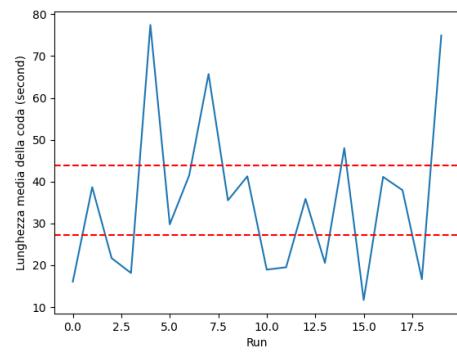
**Figura 5.125:** Conf. 1 - 0.05, 0.5



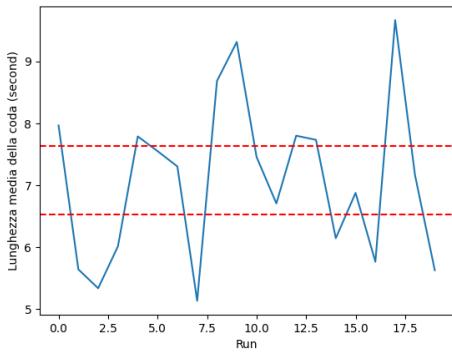
**Figura 5.126:** Conf. 1 - 0.05, 1



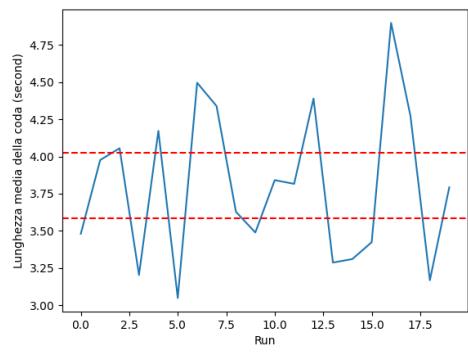
**Figura 5.127:** Conf. 1 - 0.1, 0.5



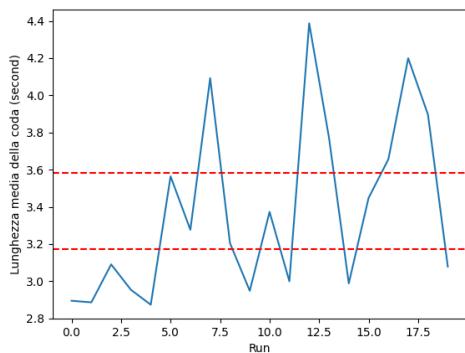
**Figura 5.128:** Conf. 1 - 0.1, 1.0



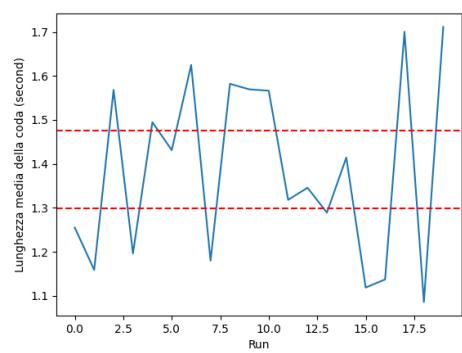
**Figura 5.129:** Conf. 1 - 0.25, 0.5



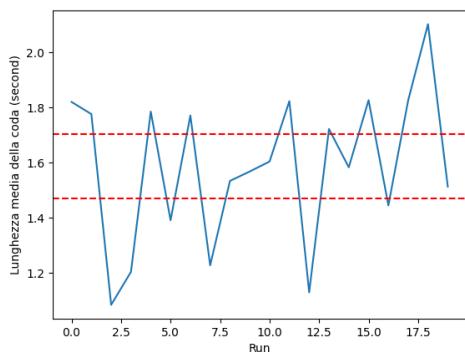
**Figura 5.130:** Conf. 1 - 0.25, 1.0



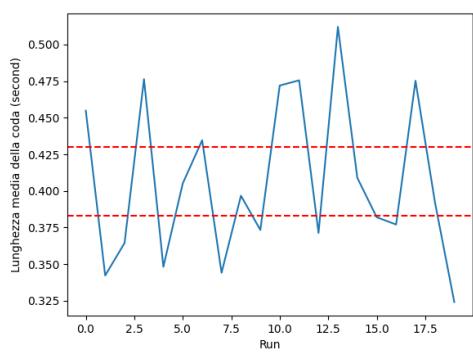
**Figura 5.131:** Conf. 1 - 0.5, 0.5



**Figura 5.132:** Conf. 1 - 0.5, 1.0



**Figura 5.133:** Conf. 1 - 1.0, 0.5



**Figura 5.134:** Conf. 1 - 1.0, 1.0

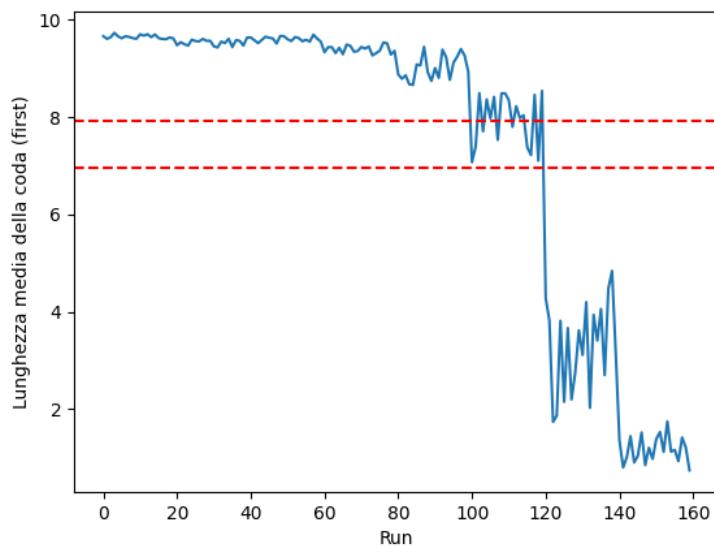
### 5.5.2 Configurazione 2

I risultati sono riportati di seguito:

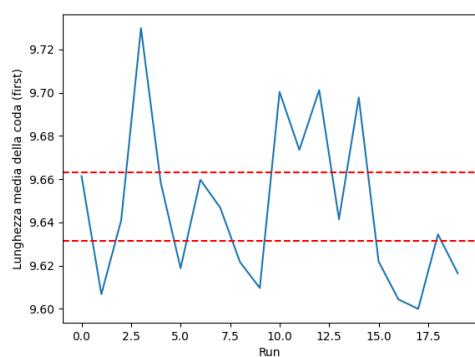
Modulo	Media	Intervallo di confidenza
Network.first	7.45 (7)	(6.97, 7.93)
Network.second	3.49 (3)	(3.22, 3.77)
Entrambe le code	5.47 (5)	(5.12, 5.82)

**Tabella 5.4:** Configurazione 2 - numero medio di utenti

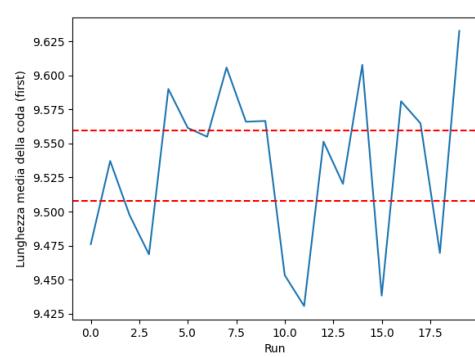
Di seguito i grafici che mostrano il numero medio di utenti per ogni coda in ogni run:



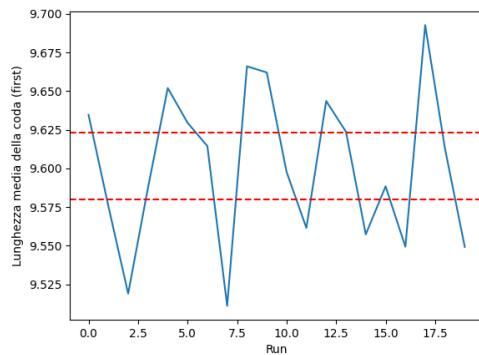
**Figura 5.135:** Configurazione 2 - numero medio di utenti (coda "first")



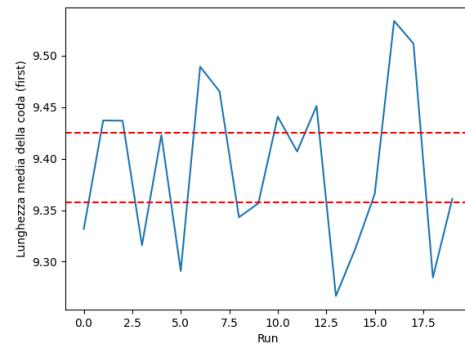
**Figura 5.136:** Conf. 2 - 0.28, 0.5



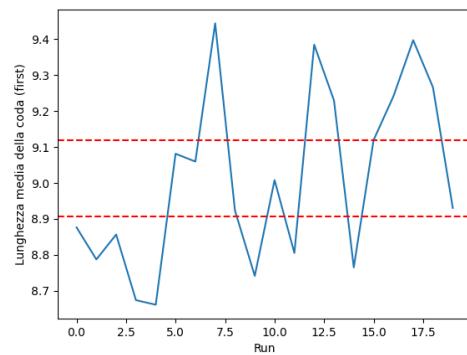
**Figura 5.137:** Conf. 2 - 0.28, 0.66



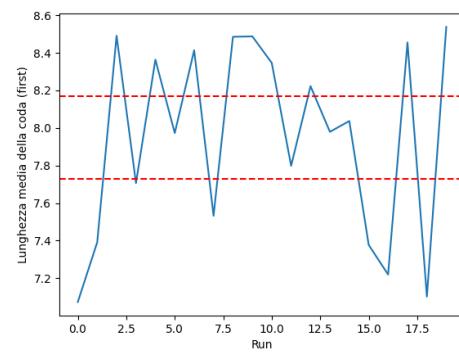
**Figura 5.138:** Conf. 2 - 0.33, 0.5



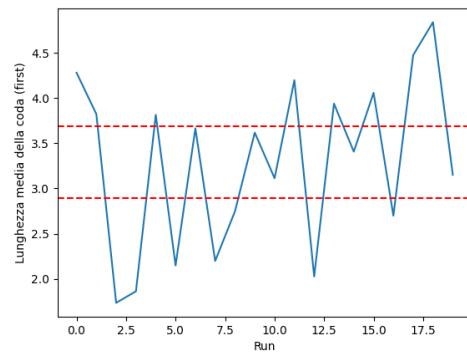
**Figura 5.139:** Conf. 2 - 0.33, 0.66



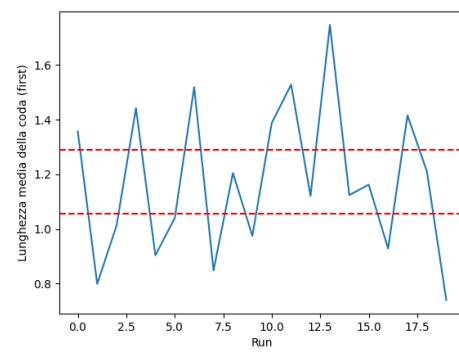
**Figura 5.140:** Conf. 2 - 0.5, 0.5



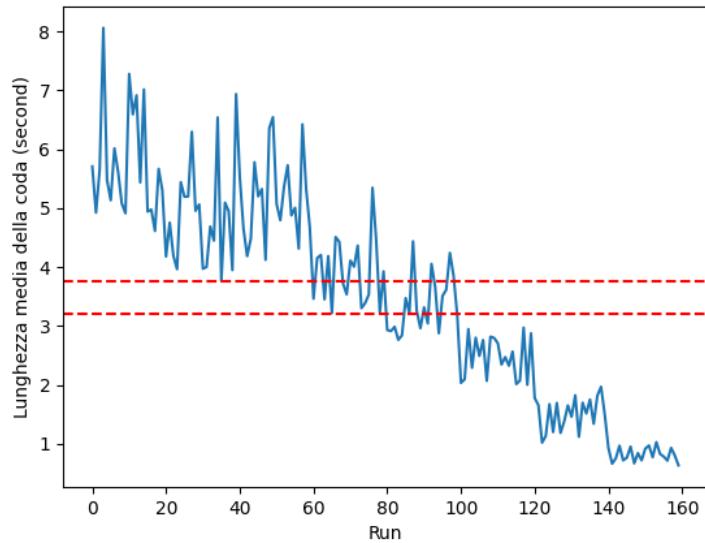
**Figura 5.141:** Conf. 2 - 0.5, 0.66



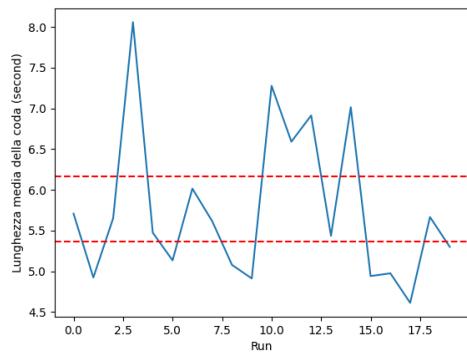
**Figura 5.142:** Conf. 2 - 1.0, 0.5



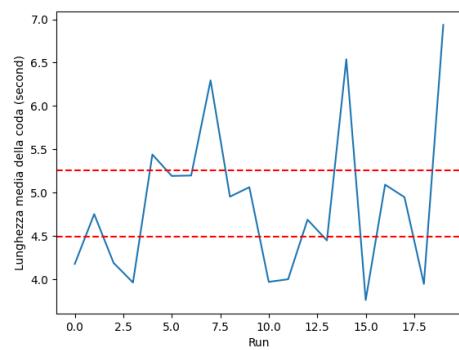
**Figura 5.143:** Conf. 2 - 1.0, 0.66



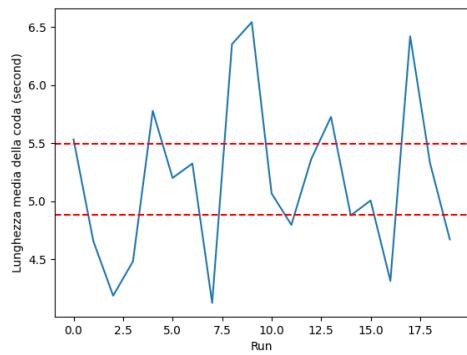
**Figura 5.144:** Configurazione 2 - numero medio di utenti (coda "second")



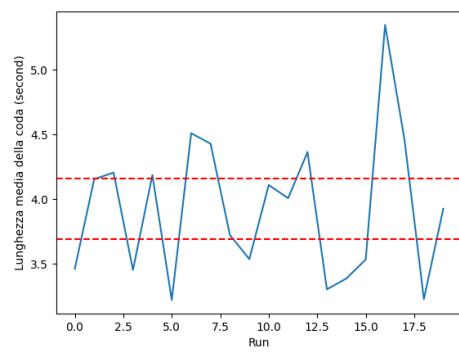
**Figura 5.145:** Conf. 2 - 0.28, 0.5



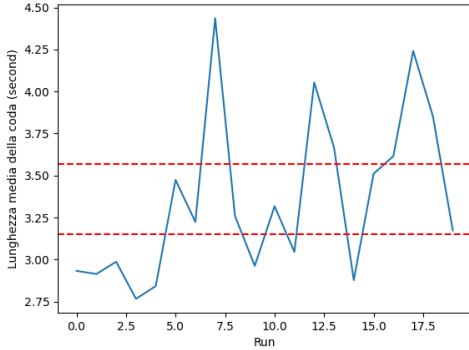
**Figura 5.146:** Conf. 2 - 0.28, 0.66



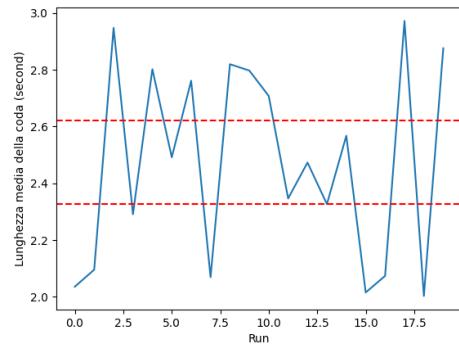
**Figura 5.147:** Conf. 2 - 0.33, 0.5



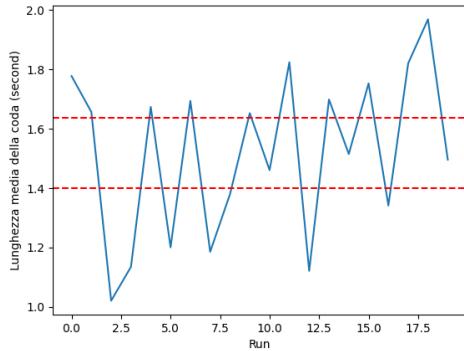
**Figura 5.148:** Conf. 2 - 0.33, 0.66



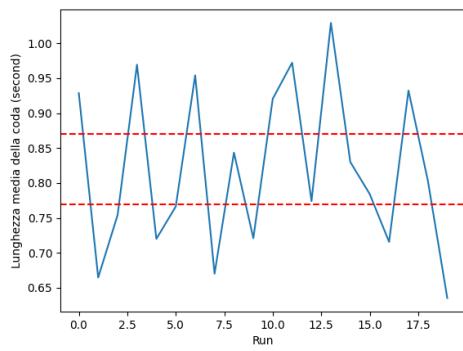
**Figura 5.149:** Conf. 2 - 0.5, 0.5



**Figura 5.150:** Conf. 2 - 0.5, 0.66



**Figura 5.151:** Conf. 2 - 1.0, 0.5



**Figura 5.152:** Conf. 2 - 1.0, 0.66

## 5.6 Numero medio di utenti persi

Per questa stima è stata calcolata la media del valore `dropped:count` delle code per ogni run.

### 5.6.1 Configurazione 1

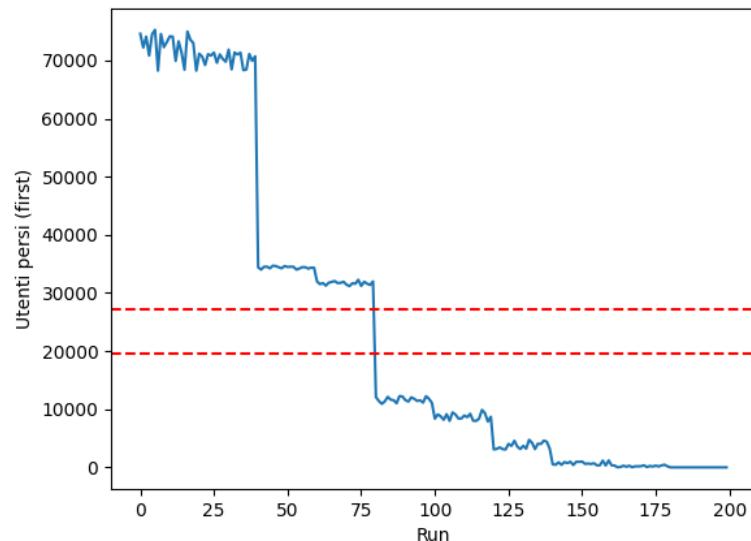
I risultati sono riportati di seguito:

Modulo	Media	Intervallo di confidenza
Network.first	23397.16 (23397)	(19688.32, 27106.00)
Network.second	0	/
Entrambe le code	11698.58 (11699)	(9518.40, 13878.76)

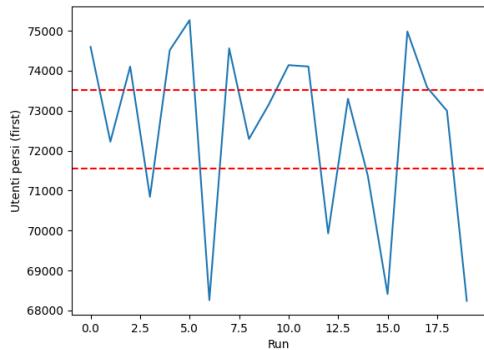
**Tabella 5.5:** Configurazione 1 - utenti persi

Per quanto riguarda la coda "second" il risultato era prevedibile dato che ha capacità infinita e quindi non perde mai utenti.

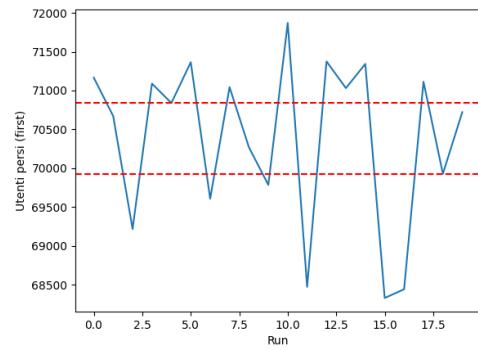
Passando invece alla coda "first", il grafico che segue mostra il numero di utenti persi per ogni run:



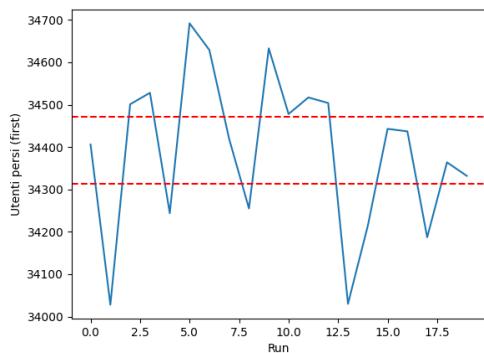
**Figura 5.153:** Configurazione 1 - utenti persi per run (coda "first")



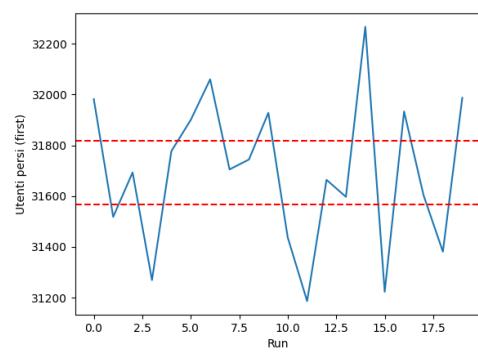
**Figura 5.154:** Conf. 1 - 0.05, 0.5



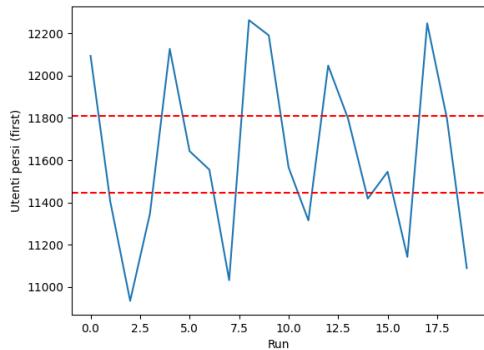
**Figura 5.155:** Conf. 1 - 0.05, 1



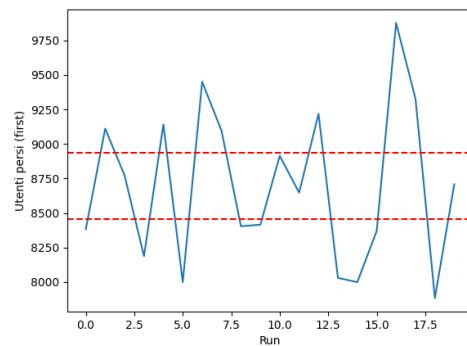
**Figura 5.156:** Conf. 1 - 0.1, 0.5



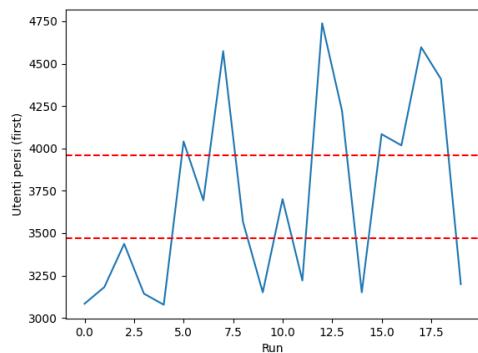
**Figura 5.157:** Conf. 1 - 0.1, 1.0



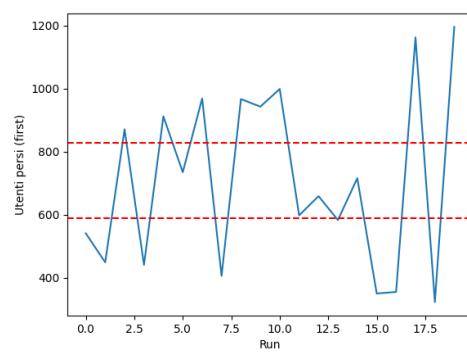
**Figura 5.158:** Conf. 1 - 0.25, 0.5



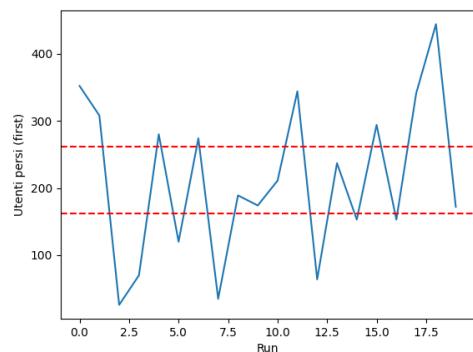
**Figura 5.159:** Conf. 1 - 0.25, 1.0



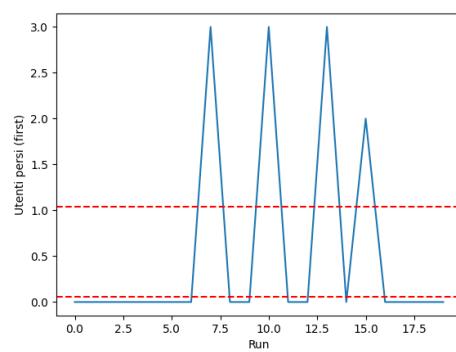
**Figura 5.160:** Conf. 1 - 0.5, 0.5



**Figura 5.161:** Conf. 1 - 0.5, 1.0



**Figura 5.162:** Conf. 1 - 1.0, 0.5



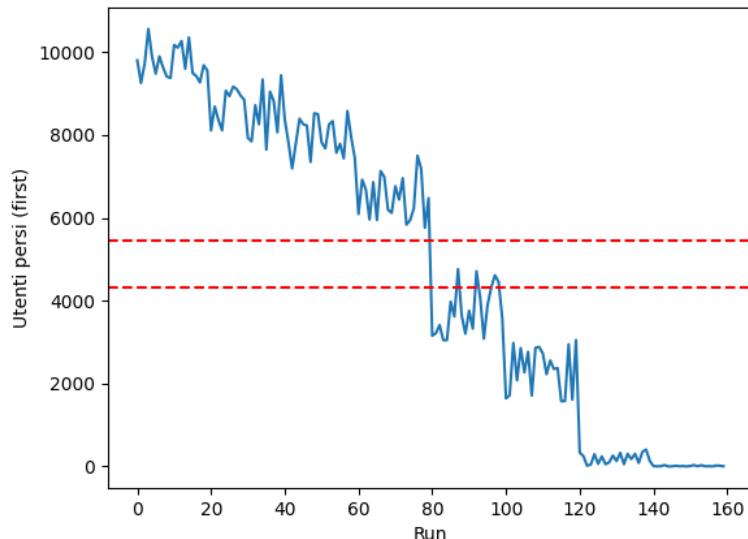
**Figura 5.163:** Conf. 1 - 1.0, 1.0

### 5.6.2 Configurazione 2

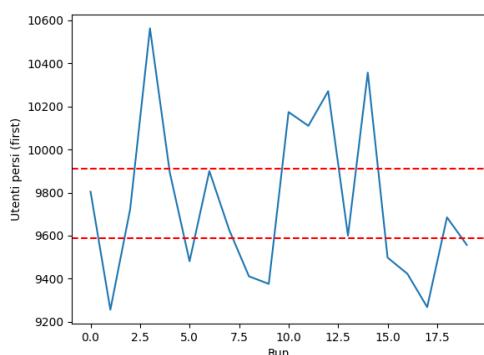
I risultati sono riportati di seguito:

Modulo	Media	Intervallo di confidenza
Network.first	4890.83 (4891)	(4330.1817, 5451.4933)
Network.second	0	/
Entrambe le code	2445.42 (2445)	(2057.6406, 2833.1969)

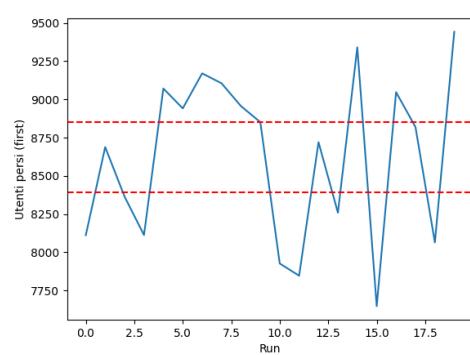
**Tabella 5.6:** Configurazione 2 - utenti persi



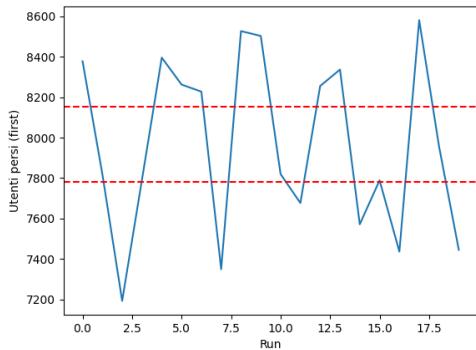
**Figura 5.164:** Configurazione 2 - utenti persi per run (coda "first")



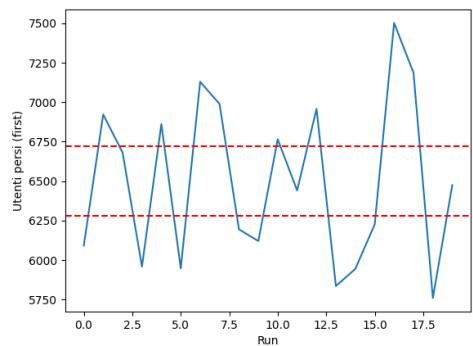
**Figura 5.165:** Conf. 2 - 0.28, 0.5



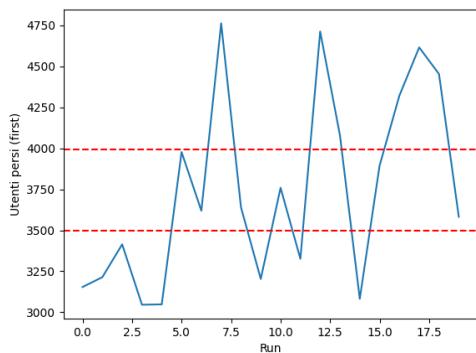
**Figura 5.166:** Conf. 2 - 0.28, 0.66



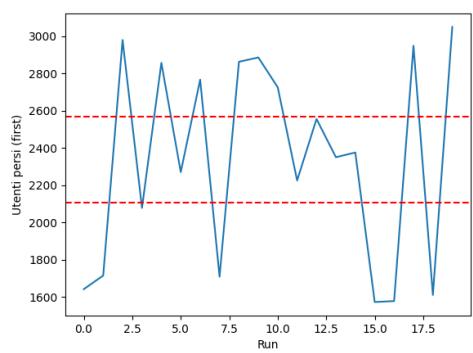
**Figura 5.167:** Conf. 2 - 0.33, 0.5



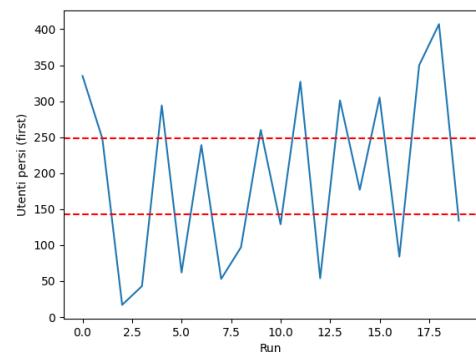
**Figura 5.168:** Conf. 2 - 0.33, 0.66



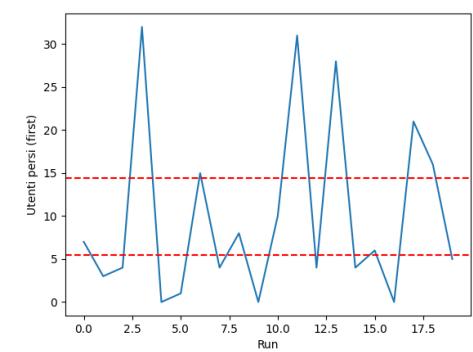
**Figura 5.169:** Conf. 2 - 0.5, 0.5



**Figura 5.170:** Conf. 2 - 0.5, 0.66



**Figura 5.171:** Conf. 2 - 1.0, 0.5



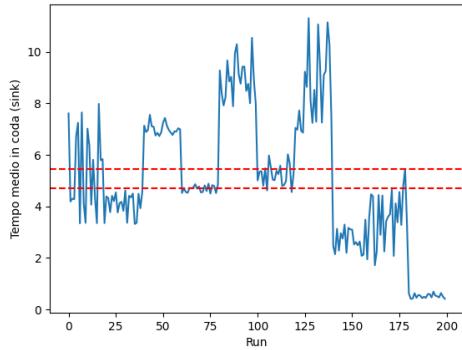
**Figura 5.172:** Conf. 2 - 1.0, 0.66

## 6 Altre analisi

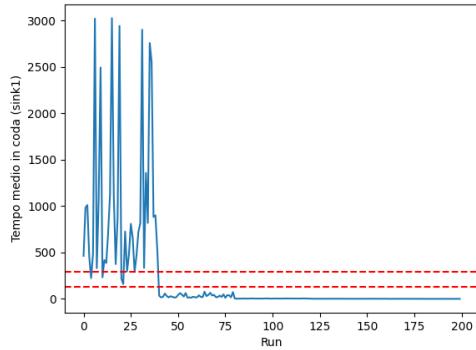
In questa sezione saranno rapidamente mostrati i risultati ricavabili dagli altri scalari forniti dal modello di simulazione, in particolari quelli forniti dai moduli Sink e Server.

I dati riguardanti "sink" fanno riferimento ai job prelevati dalla coda "first", quelli riguardanti "sink1", fanno invece riferimento ai job prelevati da "second".

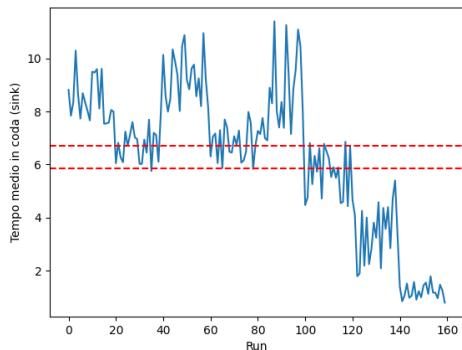
### 6.1 Tempo medio trascorso in coda dai job



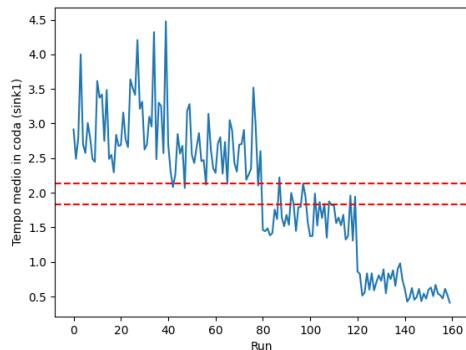
**Figura 6.1:** Conf. 1 - Queue time ("sink")



**Figura 6.2:** Conf. 1 - Queue time ("sink1")

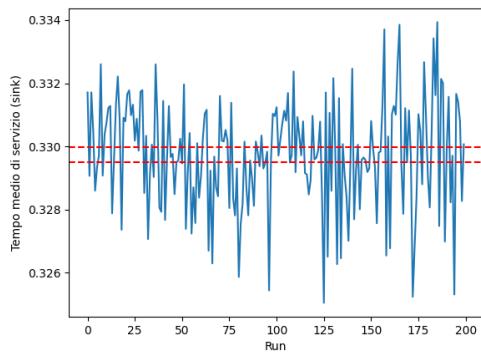


**Figura 6.3:** Conf. 2 - Queue time ("sink")

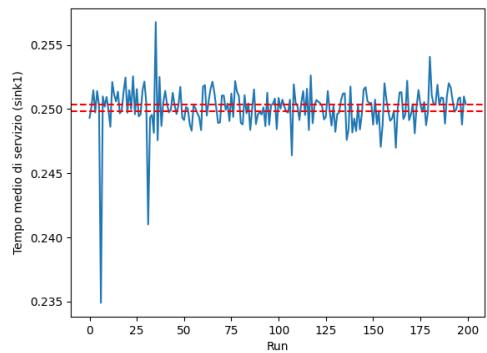


**Figura 6.4:** Conf. 2 - Queue time ("sink1")

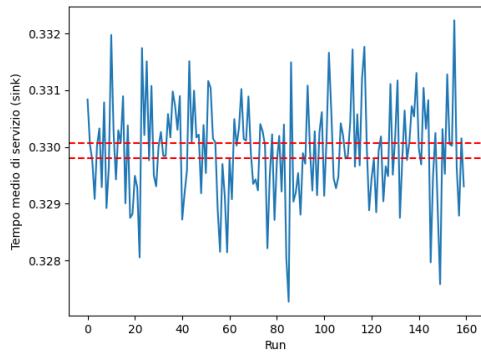
## 6.2 Tempo medio di servizio dei job



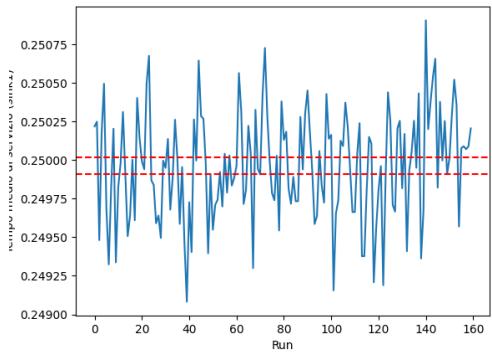
**Figura 6.5:** Conf. 1 - Service time ("sink")



**Figura 6.6:** Conf. 1 - Service time ("sink1")

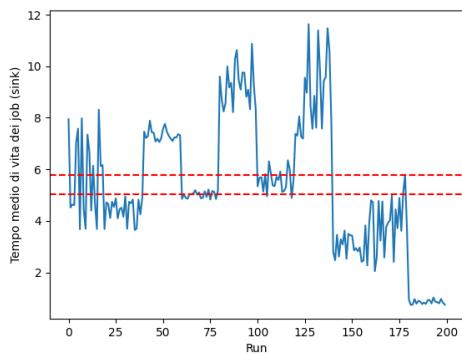


**Figura 6.7:** Conf. 2 - Service time ("sink")

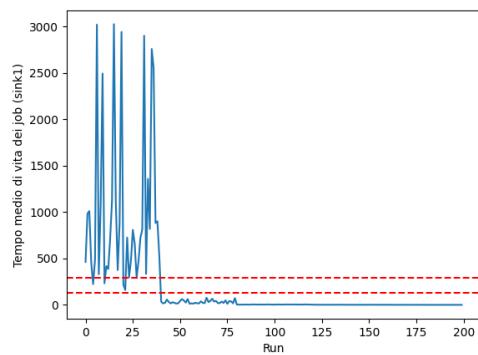


**Figura 6.8:** Conf. 2 - Service time ("sink1")

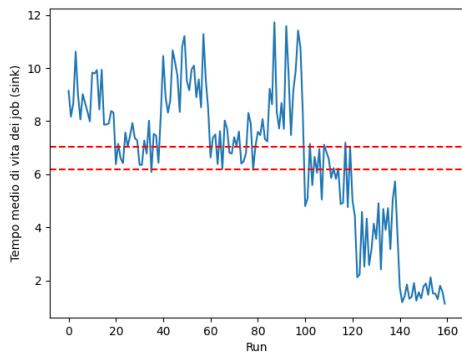
### 6.3 Tempo medio di vita dei job



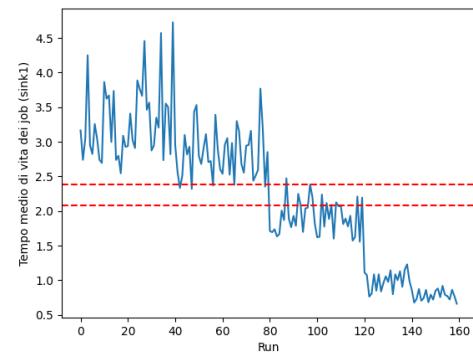
**Figura 6.9:** Conf. 1 - Lifetime ("sink")



**Figura 6.10:** Conf. 1 - Lifetime ("sink1")

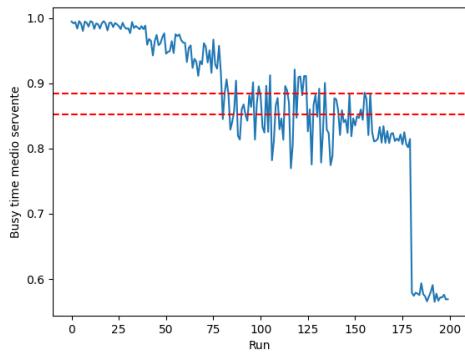


**Figura 6.11:** Conf. 2 - Lifetime ("sink")

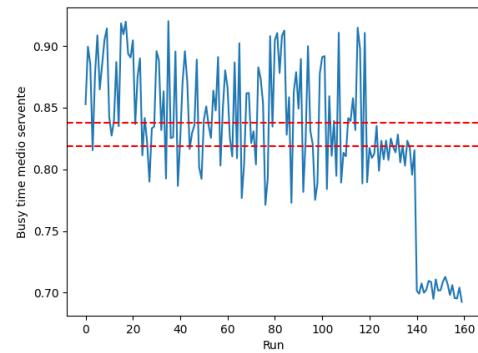


**Figura 6.12:** Conf. 2 - Lifetime ("sink1")

### 6.4 Tempo medio per cui il servente è occupato



**Figura 6.13:** Conf. 1 - Busy time



**Figura 6.14:** Conf. 2 - Busy time

## 7 Convalida del modello

Per la convalida sono stati confrontati i risultati presenti nel capitolo 5 dell'articolo<sup>[JPY18]</sup> (in particolare nelle tabelle 5.1 - 5.5) con quelli ottenuti nelle stesse condizioni dal modello realizzato.

Come affermato nell'articolo, i modelli sono stabili se  $\lambda_2 < \mu_2$ , per questo motivo durante il processo di analisi questa condizione sarà sempre rispettata.

I parametri analizzati sono stati  $E[L_i]$ , ovvero il numero medio di utenti nella coda  $i$  e  $E[W_i]$ , ovvero il tempo medio passato nella coda  $i$  dagli utenti.

Per ogni esperimento sono state effettuate 20 ripetizioni della durata di 5000 secondi di tempo simulato l'una.

### 7.1 Analisi rispetto a $\lambda_1$

Per questa analisi i parametri della simulazione hanno acquisito i seguenti valori:

Parametro	Valore
$\lambda_2$	3
$\mu_1$	3
$\mu_2$	4
$\alpha$	5
$K$	10
Capacità coda 1	10
$N$	3
Capacità coda 2	infinita

**Tabella 7.1:** Analisi rispetto a  $\lambda_1$

Di seguito i risultati ottenuti:

Valori di $\lambda_1$	$E[L_1]$ <sup>[JPY18]</sup>	$E[L_1]$ modello	$E[L_2]$ <sup>[JPY18]</sup>	$E[L_2]$ modello
0.01	0.048	0.02	3.0362	2.44
0.1	0.6055	0.53	3.3292	2.77
0.5	5.4876	4.88	4.4380	4.21
1	8.5837	8.15	5.2767	5.23
2	9.5894	9.52	5.9541	6.89
4	9.8400	9.82	6.6118	11.76
10	9.9389	9.92	7.9745	111.41
100	9.9924	9.97	25.907	1121.89

**Tabella 7.2:** Variazione di  $E[L_i]$  rispetto  $\lambda_1$

Le significative differenze tra  $E[L_2]$  dell'articolo e quello del modello proposto quando  $\lambda_1 > \lambda_2$  sono dovute al fatto che il server non effettua uno switch-over se la coda che sta servendo (in questo caso la coda "first") ha superato il suo threshold, fenomeno che all'aumentare di  $\lambda_1$  succede sempre più rapidamente.

Con queste condizioni la coda "second" non viene quindi quasi mai servita, accumulando utenti in coda.

Valori di $\lambda_1$	$E[W_1]$ [JPY18]	$E[W_1]$ modello	$E[W_2]$ [JPY18]	$E[W_2]$ modello
0.01	4.7980	3.93	1.0121	1.05
0.1	6.0575	6.26	1.1097	1.12
0.5	13.694	14.86	1.4793	1.44
1	18.438	19.08	1.7590	1.74
2	20.148	22.01	1.9847	2.26
4	19.537	19.81	2.2039	3.87
10	17.481	15.84	2.6582	36.59
100	13.173	4.80	8.6356	401.30

**Tabella 7.3:** Variazione di  $E[W_i]$  rispetto  $\lambda_1$

Un discorso analogo al precedente può essere fatto per  $E[W_i]$ : è infatti possibile notare come all'aumentare del valore di  $\lambda_1$ ,  $E[W_1]$  diminuisca in confronto a quello dell'articolo, mentre  $E[W_2]$  aumenti.

## 7.2 Analisi rispetto a $\lambda_2$

Per questa analisi i parametri della simulazione hanno acquisito i seguenti valori:

Parametro	Valore
$\lambda_1$	2
$\mu_1$	3
$\mu_2$	4
$\alpha$	5
$K$	10
Capacità coda 1	10
$N$	3
Capacità coda 2	infinita

**Tabella 7.4:** Analisi rispetto a  $\lambda_2$

Di seguito i risultati ottenuti:

Valori di $\lambda_2$	$E[L_1]$ [JPY18]	$E[L_1]$ modello	$E[L_2]$ [JPY18]	$E[L_2]$ modello
0.01	1.8891	1.24	0.0224	0.02
0.1	2.0625	1.35	0.2113	0.16
0.5	3.2757	2.53	0.8230	0.72
1	5.4280	4.78	1.5370	1.57
2	8.5720	8.35	3.2026	3.88
2.5	9.2212	9.06	4.2734	4.94
3	9.5894	9.52	5.9541	6.89
3.5	9.8260	9.74	10.256	10.82

**Tabella 7.5:** Variazione di  $E[L_i]$  rispetto  $\lambda_2$

Valori di $\lambda_2$	$E[W_1]$ [JPY18]	$E[W_1]$ modello	$E[W_2]$ [JPY18]	$E[W_2]$ modello
0.01	0.9502	0.93	2.2407	3.04
0.1	1.0390	0.96	2.1136	2.29
0.5	1.6975	1.56	1.6459	1.70
1	3.1400	3.06	1.5370	1.67
2	8.1286	8.36	1.6013	1.95
2.5	12.350	13.16	1.7094	1.97
3	20.148	22.01	1.9847	2.26
3.5	42.635	43.18	2.9302	3.09

**Tabella 7.6:** Variazione di  $E[W_i]$  rispetto  $\lambda_2$

### 7.3 Analisi rispetto a $\mu_1$

Per questa analisi i parametri della simulazione hanno acquisito i seguenti valori:

Parametro	Valore
$\lambda_1$	2
$\lambda_2$	3
$\mu_2$	4
$\alpha$	5
$K$	10
Capacità coda 1	10
$N$	3
Capacità coda 2	infinita

**Tabella 7.7:** Analisi rispetto a  $\mu_1$

Di seguito i risultati ottenuti:

Valori di $\mu_1$	$E[L_1]$ [JPY18]	$E[L_1]$ modello	$E[L_2]$ [JPY18]	$E[L_2]$ modello
0.01	9.9964	9.96	421.50	1498.73
0.1	9.9847	9.94	46.149	656.92
0.5	9.9439	9.91	12.962	44.90
1	9.8935	9.85	8.7647	18.01
2	9.7749	9.71	6.6644	8.91
4	9.3026	9.13	5.5780	6.03
10	6.9733	6.74	4.6262	4.43
100	4.3079	2.66	3.8023	1.87

**Tabella 7.8:** Variazione di  $E[L_i]$  rispetto  $\mu_1$

Valori di $\mu_1$	$E[W_1]$ [JPY18]	$E[W_1]$ modello	$E[W_2]$ [JPY18]	$E[W_2]$ modello
0.01	1375.0	466.93	140.50	482.24
0.1	326.36	141.33	15.383	222.43
0.5	90.189	78.15	4.3206	14.85
1	49.720	47.04	2.9216	5.91
2	28.289	27.71	2.2215	2.95
4	15.440	17.29	1.8593	1.99
10	6.0813	9.17	1.5421	1.51
100	2.7715	3.95	1.2674	1.09

**Tabella 7.9:** Variazione di  $E[W_i]$  rispetto  $\mu_1$

Similmente a quanto accaduto precedentemente con  $\lambda_1$ , anche in questo caso la coda "first" resta servita per più tempo con una conseguente ricaduta su  $E[L_2]$  ed  $E[W_2]$ .

## 7.4 Analisi rispetto a $\mu_2$

Per questa analisi i parametri della simulazione hanno acquisito i seguenti valori:

Parametro	Valore
$\lambda_1$	2
$\lambda_2$	3
$\mu_1$	3
$\alpha$	5
$K$	10
Capacità coda 1	10
$N$	3
Capacità coda 2	infinita

**Tabella 7.10:** Analisi rispetto a  $\mu_2$

Di seguito i risultati ottenuti:

Valori di $\mu_2$	$E[L_1]$ [JPY18]	$E[L_1]$ modello	$E[L_2]$ [JPY18]	$E[L_2]$ modello
3.25	9.8830	9.82	14.882	13.49
3.5	9.7796	9.72	9.0270	9.66
3.75	9.6819	9.58	6.9922	8.54
4	9.5894	9.52	5.9541	6.89
10	7.8261	6.84	2.5769	3.55
100	5.2089	2.68	1.4877	1.98

**Tabella 7.11:** Variazione di  $E[L_i]$  rispetto  $\mu_2$

Valori di $\mu_2$	$E[W_1]$ [JPY18]	$E[W_1]$ modello	$E[W_2]$ [JPY18]	$E[W_2]$ modello
3.25	67.120	67.30	4.9606	4.49
3.5	36.186	37.09	3.0090	3.19
3.75	25.512	25.78	2.3307	2.83
4	20.148	22.01	1.9847	2.26
10	5.6290	5.67	0.8590	1.20
100	2.8689	2.61	0.4959	0.94

**Tabella 7.12:** Variazione di  $E[W_i]$  rispetto  $\mu_2$

## 7.5 Analisi rispetto ad $\alpha$

Per questa analisi i parametri della simulazione hanno acquisito i seguenti valori:

Parametro	Valore
$\lambda_1$	2
$\lambda_2$	3
$\mu_1$	3
$\mu_2$	4
$K$	10
Capacità coda 1	10
$N$	3
Capacità coda 2	infinita

**Tabella 7.13:** Analisi rispetto ad  $\alpha$

Di seguito i risultati ottenuti:

Valori di $\alpha$	$E[L_1]$ [JPY18]	$E[L_1]$ modello	$E[L_2]$ [JPY18]	$E[L_2]$ modello
3.25	9.6784	9.57	6.3537	7.95
3.5	9.6642	9.55	6.2747	8.155
3.75	9.6505	9.51	6.2053	7.64
4	9.6373	9.52	6.1439	7.133
10	9.4332	9.33	5.5400	6.13
100	9.1523	9.18	5.1009	5.83

**Tabella 7.14:** Variazione di  $E[L_i]$  rispetto ad  $\alpha$

Valori di $\alpha$	$E[W_1]$ [JPY18]	$E[W_1]$ modello	$E[W_2]$ [JPY18]	$E[W_2]$ modello
3.25	23.608	27.86	2.1179	2.63
3.5	22.951	25.89	2.0916	2.70
3.75	22.362	25.43	2.0685	2.52
4	21.831	24.55	2.0480	2.35
10	16.375	16.78	1.8467	2.02
100	12.625	13.28	1.7003	1.91

**Tabella 7.15:** Variazione di  $E[W_i]$  rispetto ad  $\alpha$

## 7.6 Conclusioni

Visti i risultati a confronto è possibile affermare che il modello realizzato sia simile al modello dell'articolo<sup>[JPY18]</sup>, ad eccezione di alcuni valori che dipendono dal fatto che la coda "first" tende a saturarsi molto in fretta, impedendo al servente di realizzare uno switch-over per servire la coda "second".

## Riferimenti bibliografici

- [JPY18] Amit Jolles, Efrat Perel, and Uri Yechiali. Alternating server with non-zero switch-over times and opposite-queue threshold-based switching policy. *Performance Evaluation*, 126:22–38, 2018.