

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

1 Introduzione	3
2 Modello da realizzare	3
3 Descrizione del codice prodotto	3
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
4 Simulazioni	7
4.1 Configurazione 1	7
4.2 Configurazione 2	7
5 Analisi dei risultati	8
5.1 Approccio generale	8
5.2 Tempo medio di permanenza nel sistema dei job	8
5.2.1 Configurazione 1	8
5.2.2 Configurazione 2	9
5.3 Numero medio di utenti nelle singole code	10
5.3.1 Configurazione 1	10
5.3.2 Configurazione 2	12
5.4 Numero medio di utenti persi	13
5.4.1 Configurazione 1	13
5.4.2 Configurazione 2	14
5.5 Transiente iniziale	15
5.5.1 Analisi di queueLength:vector	15
5.5.2 Analisi di queueLength:vector	16
6 Altre analisi	18
6.1 Tempo medio trascorso in coda dai job	18
6.2 Tempo medio di servizio dei job	19
6.3 Tempo medio di vita dei job	20
6.4 Tempo medio per cui il servente è occupato	20

7 Convalida del modello	21
7.1 Analisi rispetto a λ_1	21
7.2 Analisi rispetto a λ_2	23
7.3 Analisi rispetto a μ_1	24
7.4 Analisi rispetto a μ_2	25
7.5 Analisi rispetto ad α	26
7.6 Conclusioni	26

Appendici

Appendice A Risultati per gruppi di run

A.1 Utenti persi (coda "first") - Conf. 1
A.2 Utenti persi (coda "first") - Conf. 2
A.3 Tempo di permanenza (coda "first") - Conf. 1
A.4 Tempo di permanenza (coda "first") - Conf. 2
A.5 Tempo di permanenza (coda "second") - Conf. 1
A.6 Tempo di permanenza (coda "second") - Conf. 2
A.7 Numero medio di utenti (coda "first") - Conf. 1
A.8 Numero medio di utenti (coda "first") - Conf. 2
A.9 Numero medio di utenti (coda "second") - Conf. 1
A.10 Numero medio di utenti (coda "second") - Conf. 2

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 λ_1 e λ_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 indicati da μ_1 e μ_2 , 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 α .

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.

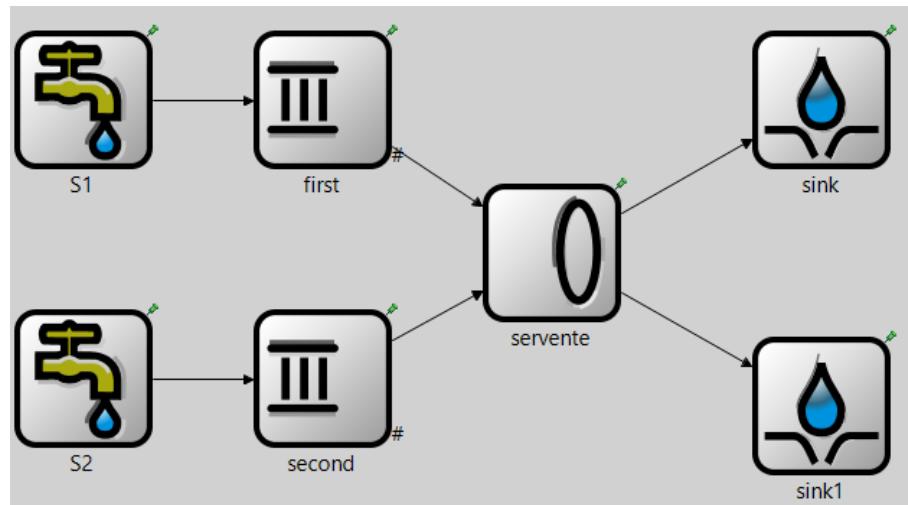


Figura 3.1: Rete realizzata

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 μ_1 e μ_2 (`ServiceTime` e `ServiceTime_2`) e α (`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 α 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 1000 secondi di tempo simulato.

4.1 Configurazione 1

I parametri in questa configurazione hanno assunto i seguenti valori:

Parametro	Valore	.ini
λ_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)
λ_2 (interarrivo 2)	2.0, 4.0	exponential(\$\{0.5, 0.25\}s)
μ_1 (service time 1)	[0.11, 0.55]	uniform(0.11s,0.55s)
μ_2 (service time 2)	[0.1, 0.4]	uniform(0.1s,0.4s)
α (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
λ_1 (interarrivo 1)	2.0, 2.5	exponential(\$\{0.5, 0.4\}s)
λ_2 (interarrivo 2)	1.0, 2.0, 3.0, 3.5	exponential(\$\{1.0,0.5,0.33,0.28\}s)
μ_1 (service time 1)	[0.22, 0.44]	uniform(0.22s,0.44s)
μ_2 (service time 2)	[0.2,0.3]	uniform(0.2s,0.3s)
α (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()`.

Inoltre sono stati realizzati i grafici per i risultati per tutte le run e per i singoli gruppi di run (disponibili in appendice A).

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 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.

5.2.1 Configurazione 1

I risultati sono riportati di seguito:

Modulo	Media	Intervallo di confidenza
Network.first	49.7005 s	(35.6207, 63.7803)
Network.second	90.6305 s	(64.8232, 116.4378)
Entrambe le code	70.2684 s	(55.3926, 85.1441)

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:

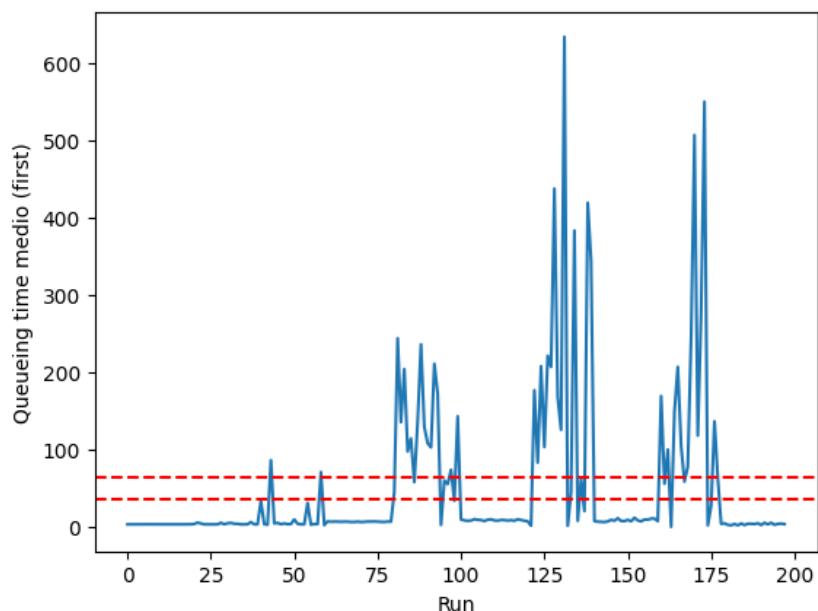


Figura 5.1: Configurazione 1 - tempo medio di permanenza (coda "first")

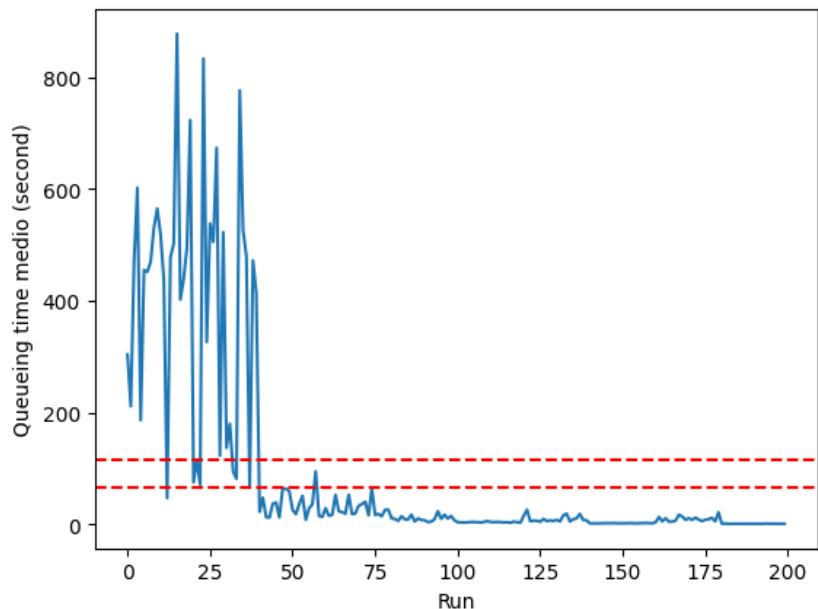


Figura 5.2: Configurazione 1 - tempo medio di permanenza (coda "second")

5.2.2 Configurazione 2

I risultati sono riportati di seguito:

Modulo	Media	Intervallo di confidenza
Network.first	20.8838 s	(17.9985, 23.7692)
Network.second	2.0308 s	(1.9633, 2.0984)
Entrambe le code	11.4573 s	(9.6827, 13.2319)

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:

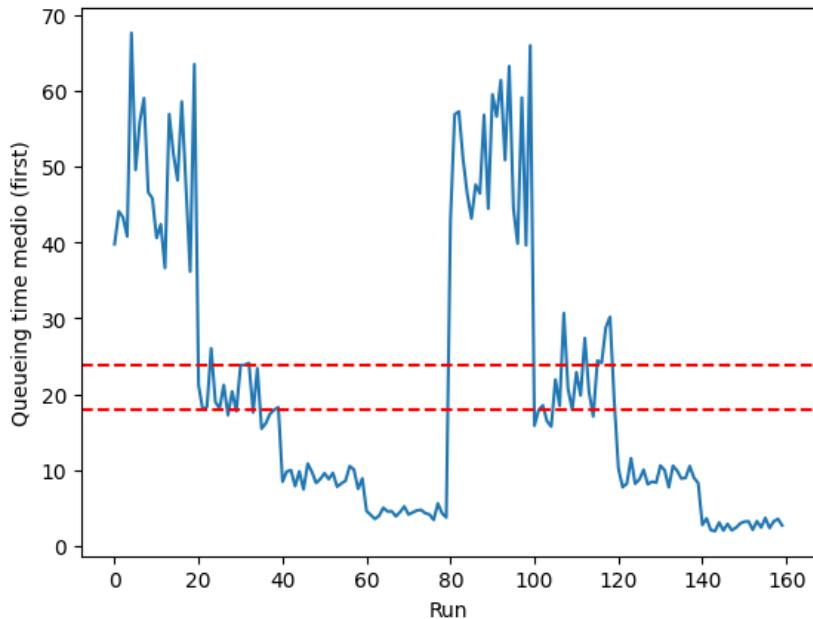


Figura 5.3: Configurazione 2 - tempo medio di permanenza (coda "first")

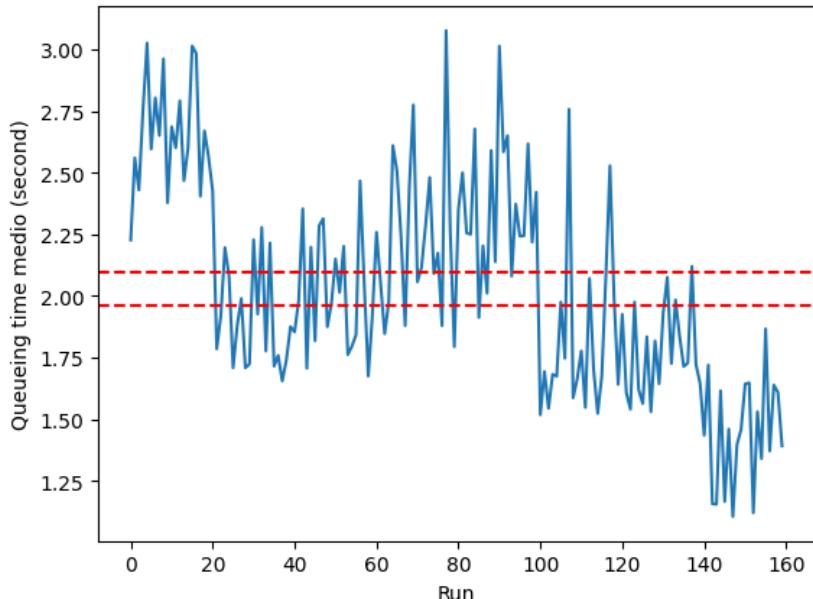


Figura 5.4: Configurazione 2 - tempo medio di permanenza (coda "second")

5.3 Numero medio di utenti nelle singole code

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

5.3.1 Configurazione 1

I risultati sono riportati di seguito:

Modulo	Media	Intervallo di confidenza
Network.first	9.1197 s	(8.8419, 9.3974)
Network.second	261.2103 s	(187.3792, 335.0414)
Entrambe le code	135.1650 s	(96.2374, 174.0925)

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:

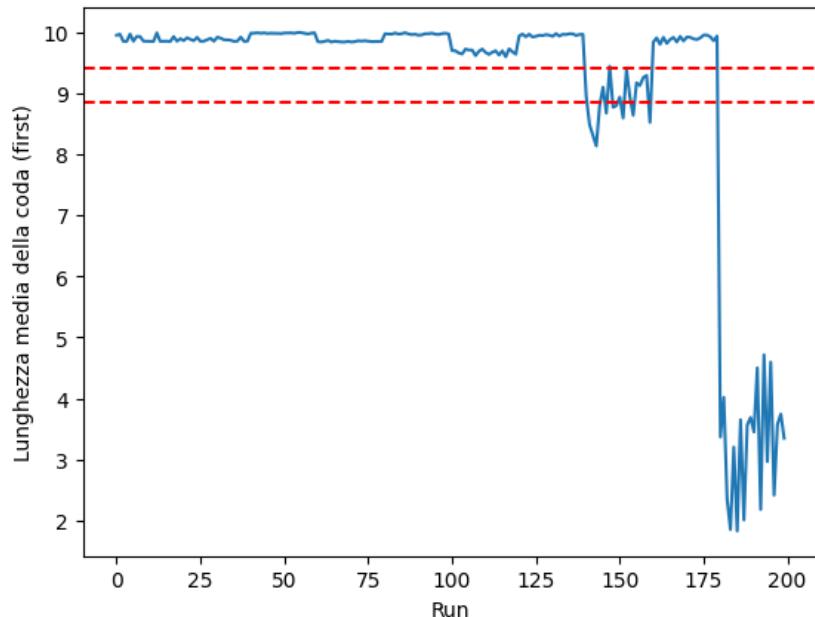


Figura 5.5: Configurazione 1 - numero medio di utenti (coda "first")

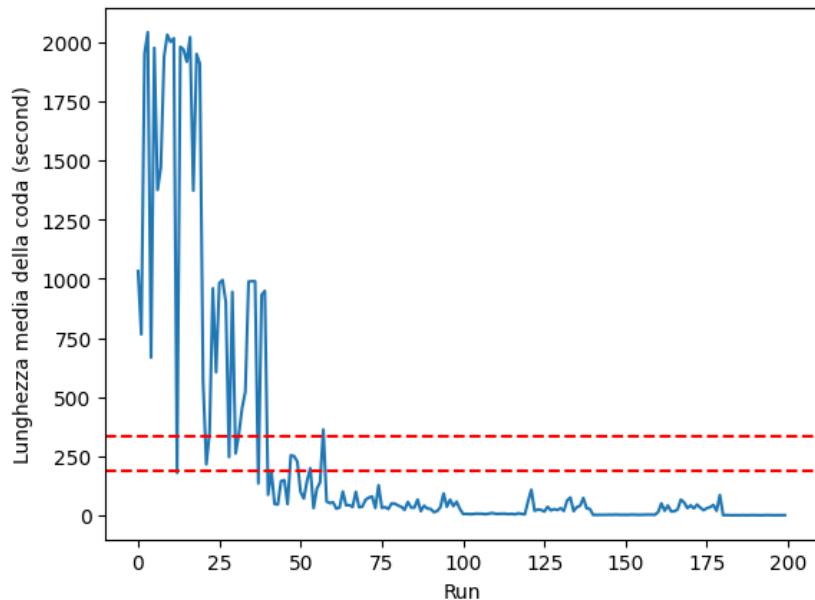


Figura 5.6: Configurazione 1 - numero medio di utenti (coda "second")

5.3.2 Configurazione 2

I risultati sono riportati di seguito:

Modulo	Media	Intervallo di confidenza
Network.first	8.8007 s	(8.5462, 9.0551)
Network.second	5.0807 s	(4.6513, 5.5100)
Entrambe le code	6.9407 s	(6.6185, 7.2628)

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:

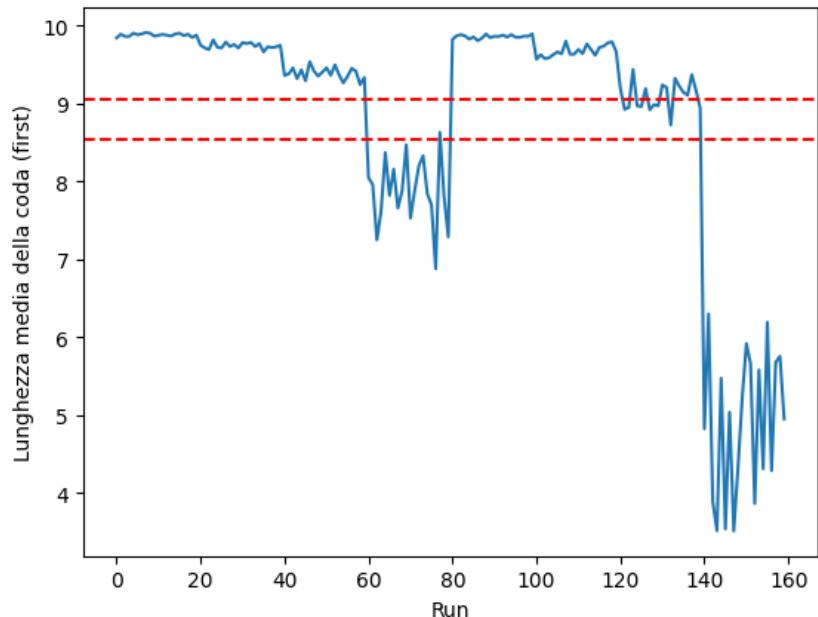


Figura 5.7: Configurazione 2 - numero medio di utenti (coda "first")

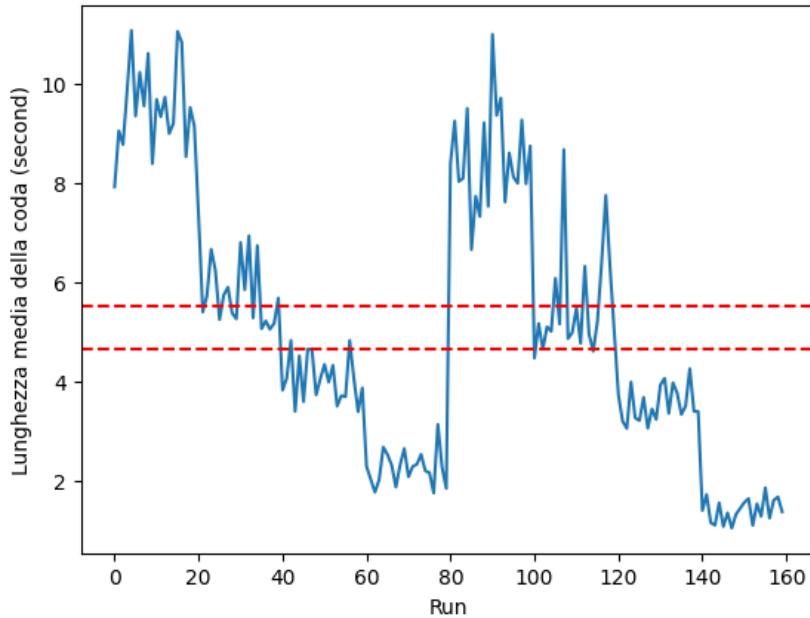


Figura 5.8: Configurazione 2 - numero medio di utenti (coda "second")

5.4 Numero medio di utenti persi

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

5.4.1 Configurazione 1

I risultati sono riportati di seguito:

Modulo	Media	Intervallo di confidenza
Network.first	6456.9300 (6457)	(5568.0493, 7345.8107)
Network.second	0	/
Entrambe le code	3228.4650 (3228)	(2682.9140, 3774.0160)

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:

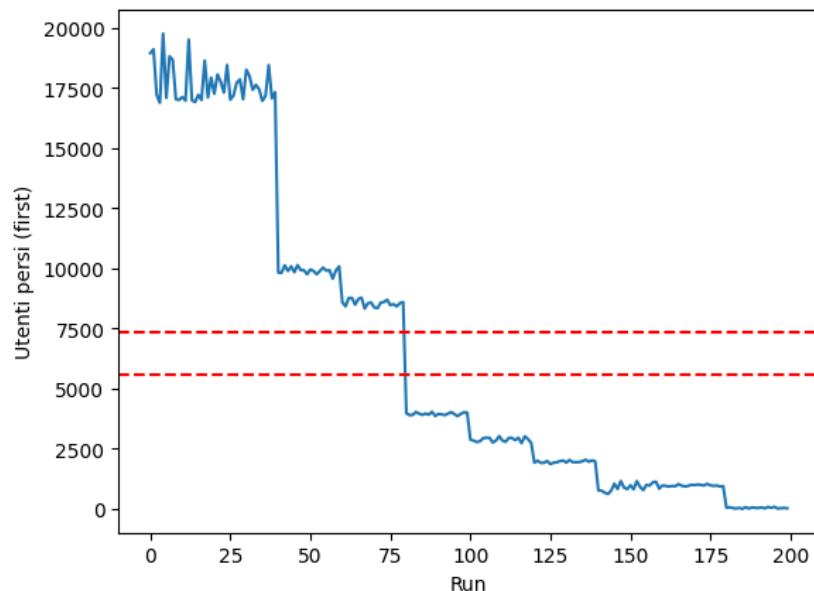


Figura 5.9: Configurazione 1 - utenti persi per run

5.4.2 Configurazione 2

I risultati sono riportati di seguito:

Modulo	Media	Intervallo di confidenza
Network.first	1355.0812 (1355)	(1252.0166, 1458.1459)
Network.second	0	/
Entrambe le code	677.5406 (678)	(587.1724, 767.9088)

Tabella 5.6: Configurazione 2 - utenti persi

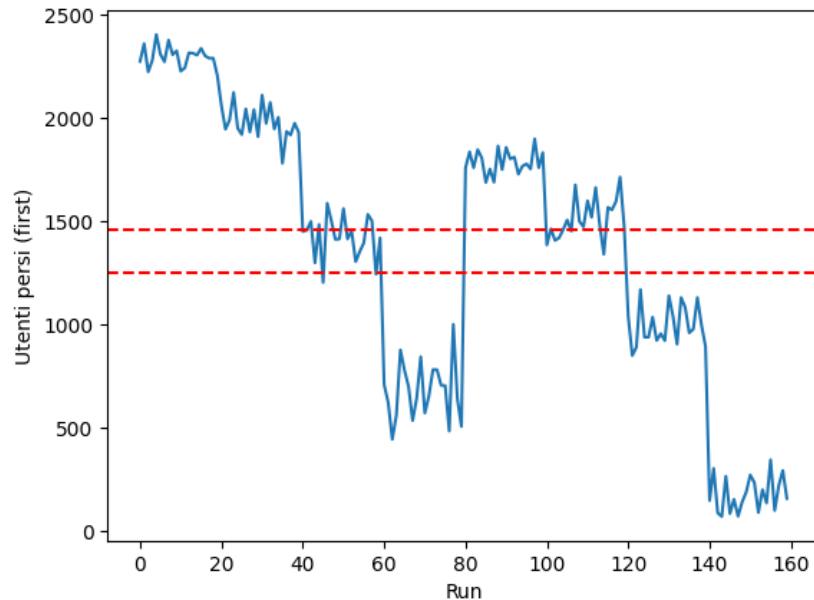


Figura 5.10: Configurazione 2 - utenti persi per run (coda "first")

5.5 Transiente iniziale

In questa sezione si discuterà il problema del transiente iniziale mediante il grafico dei vettori `queueLength:vector` e `lifeTime:vector` forniti dai moduli `PassiveQueue` ("first" e "second") e `Sink` ("sink" e "sink1").

5.5.1 Analisi di `queueLength:vector`

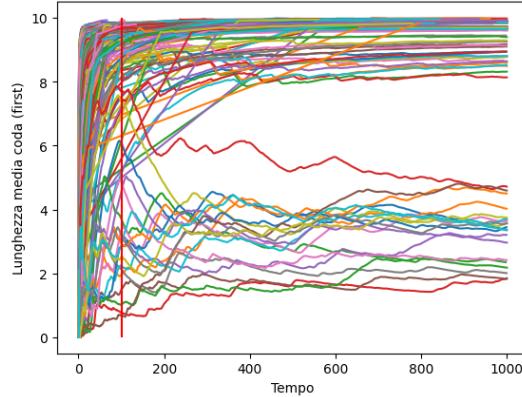


Figura 5.11: Conf. 1 - lunghezza first

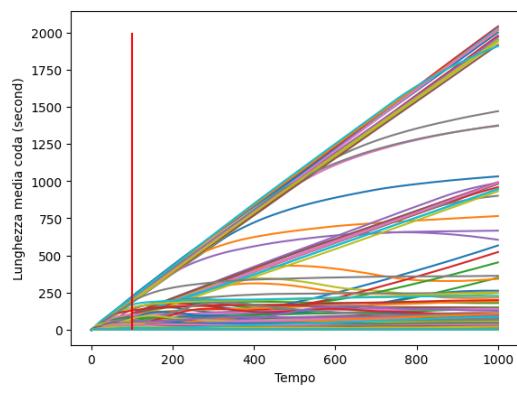


Figura 5.12: Conf. 1 - lunghezza second

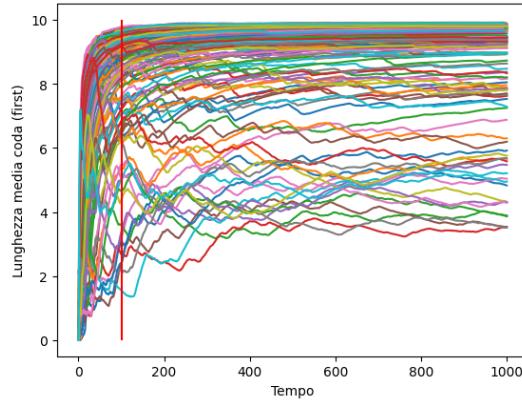


Figura 5.13: Conf. 2 - lunghezza first

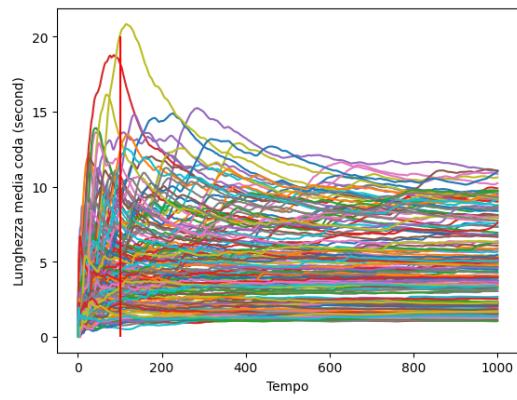


Figura 5.14: Conf. 2 - lunghezza second

In tutti e quattro i casi è possibile notare come l'andamento delle curve si stabilizzi in un punto compreso nell'intervallo di tempo tra 0 e 200.

In particolare, in ogni grafico è stata disegnata una linea verticale rossa in corrispondenza di 100 unità di tempo ed è possibile notare come per quanto riguarda la lunghezza delle code si tratti di un transiente iniziale adatto.

Per dimostrare ciò è stato inserito un `warmup-period = 100s`:

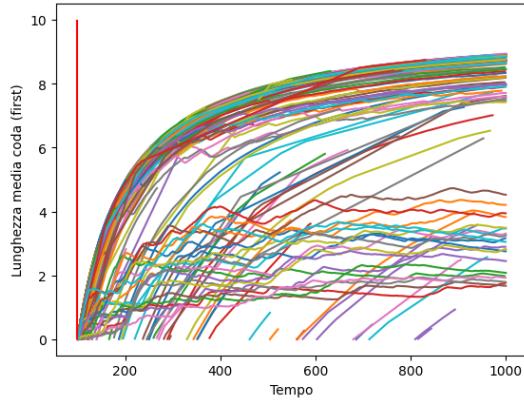


Figura 5.15: Conf. 1 - lunghezza first

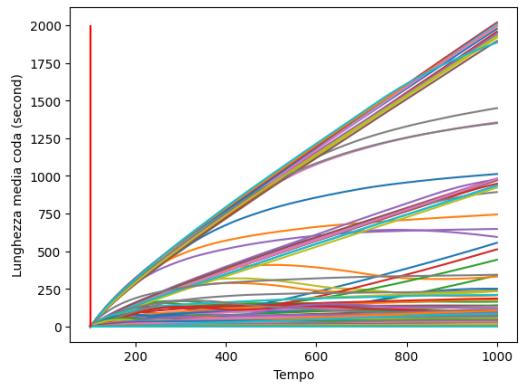


Figura 5.16: Conf. 1 - lunghezza second

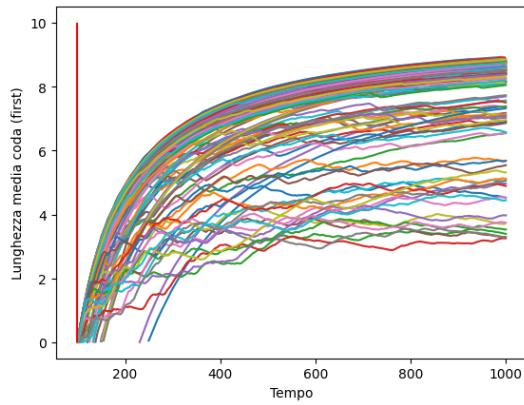


Figura 5.17: Conf. 2 - lunghezza first

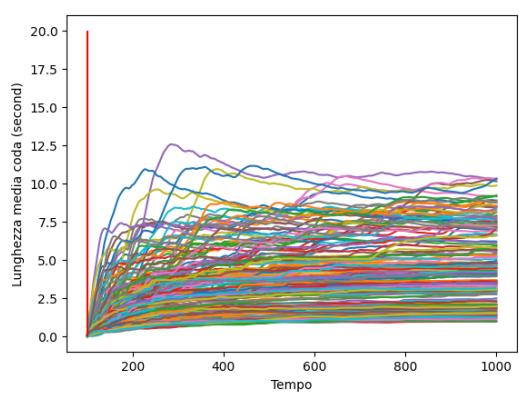


Figura 5.18: Conf. 2 - lunghezza second

5.5.2 Analisi di queueLength:vector

Lo stesso tipo di analisi è stato eseguito su `queueLength:vector`:

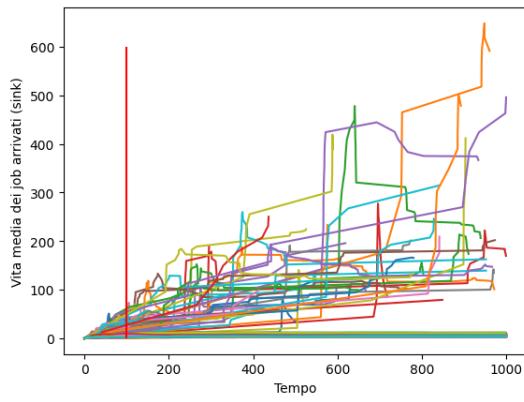


Figura 5.19: Conf. 1 - lifetime first

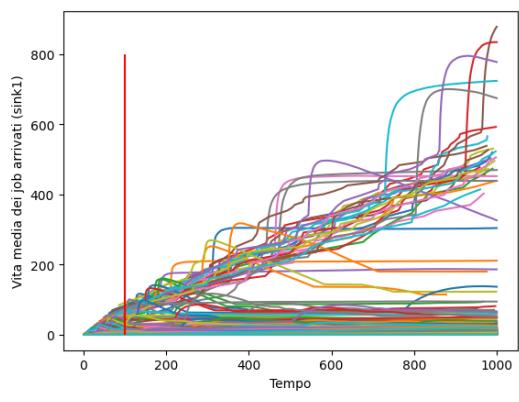


Figura 5.20: Conf. 1 - lifetime second

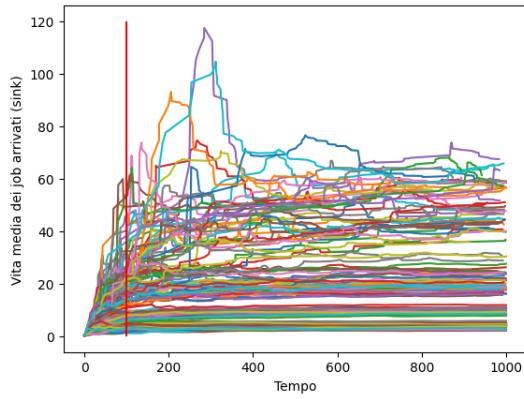


Figura 5.21: Conf. 2 - lifetime first

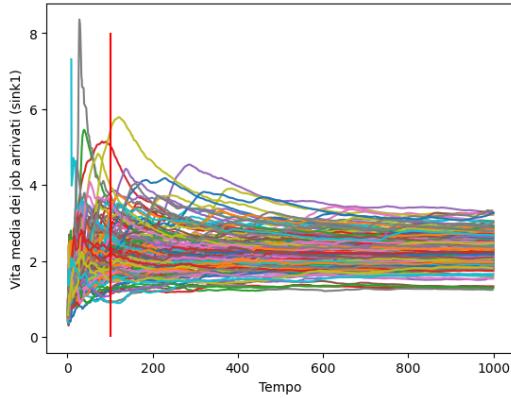


Figura 5.22: Conf. 2 - lifetime second

Analogamente a quanto avvenuto in precedenza è stata disegnata una linea verticale rossa in corrispondenza di 100 unità di tempo ed è possibile notare come anche in questo caso si possa considerare un transiente iniziale adatto.

I risultati con `warmup-period = 100s` sono i seguenti:

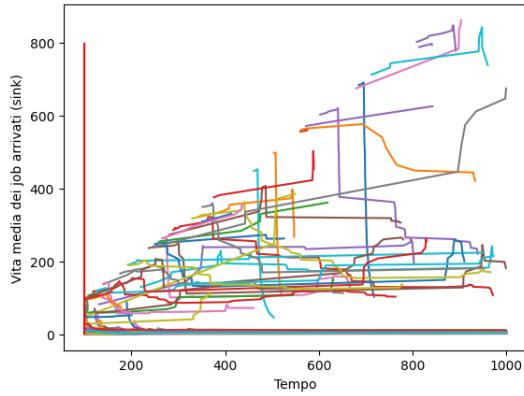


Figura 5.23: Conf. 1 - lifetime first

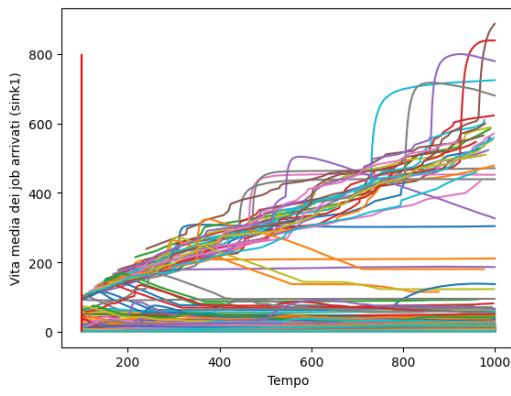


Figura 5.24: Conf. 1 - lifetime second

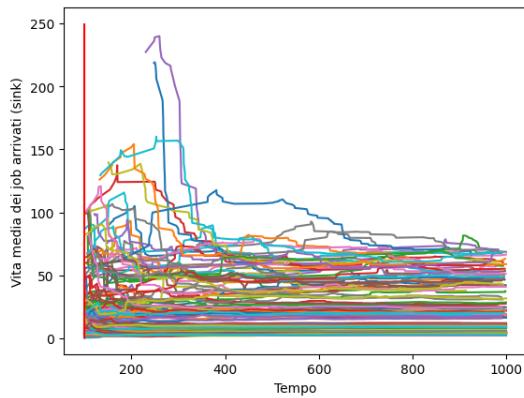


Figura 5.25: Conf. 2 - lifetime first

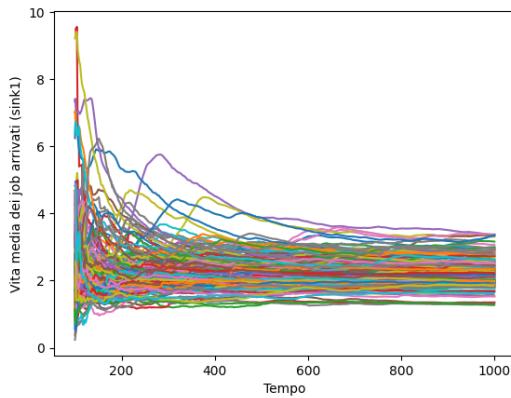


Figura 5.26: Conf. 2 - lifetime second

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

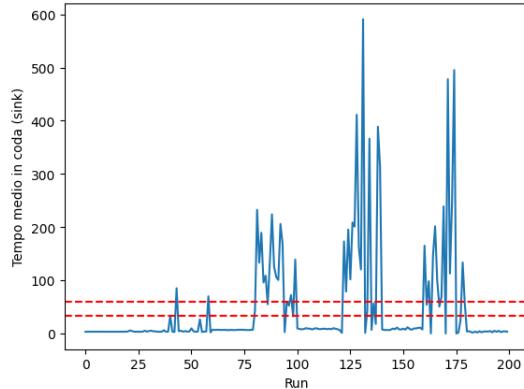


Figura 6.1: Conf. 1 - Queue time ("sink")

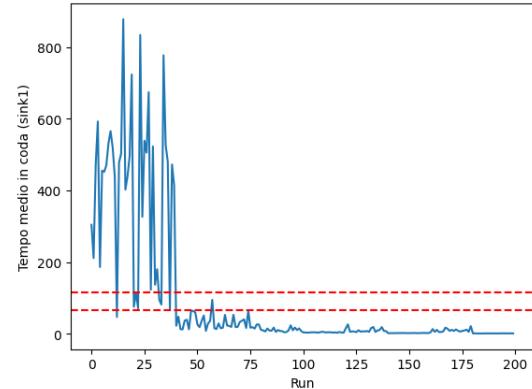


Figura 6.2: Conf. 1 - Queue time ("sink1")

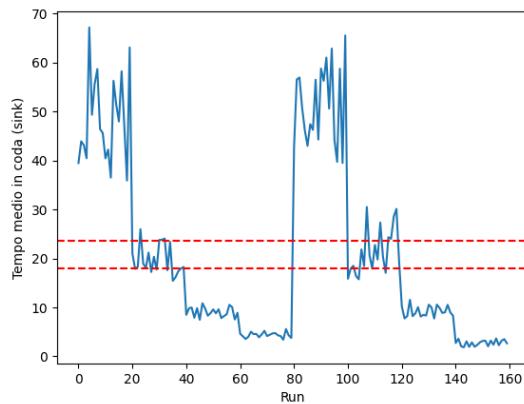


Figura 6.3: Conf. 2 - Queue time ("sink")

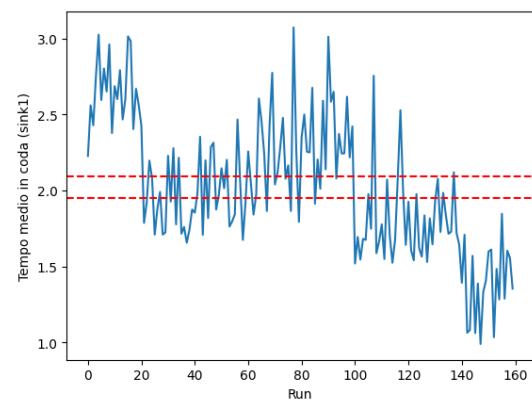


Figura 6.4: Conf. 2 - Queue time ("sink1")

6.2 Tempo medio di servizio dei job

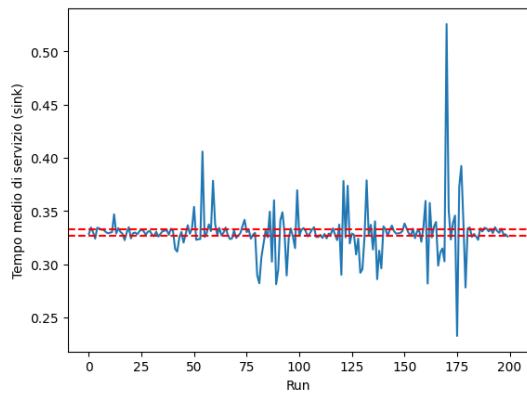


Figura 6.5: Conf. 1 - Service time ("sink")

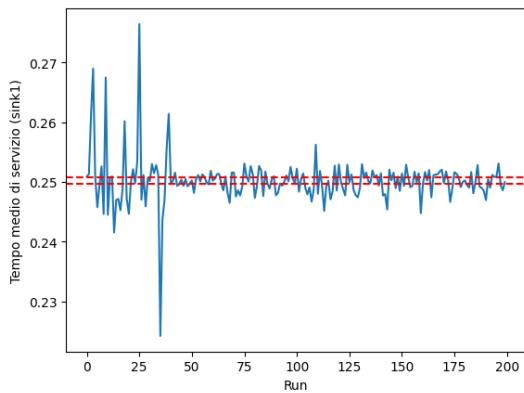


Figura 6.6: Conf. 1 - Service time ("sink1")

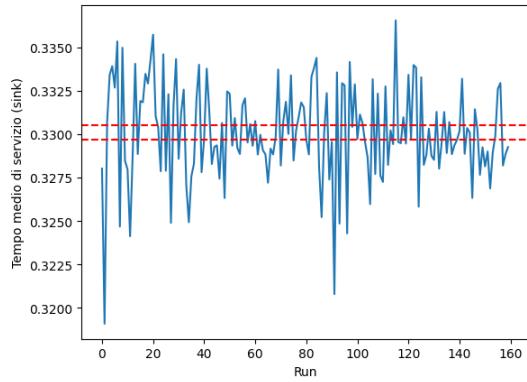


Figura 6.7: Conf. 2 - Service time ("sink")

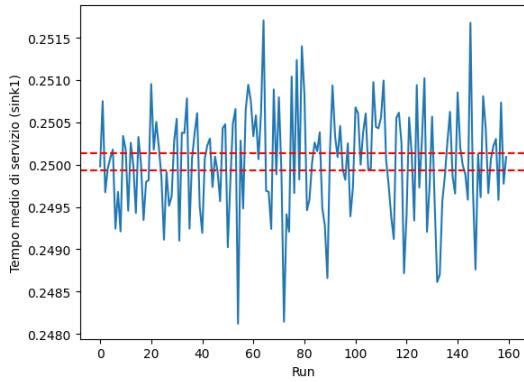


Figura 6.8: Conf. 2 - Service time ("sink1")

6.3 Tempo medio di vita dei job

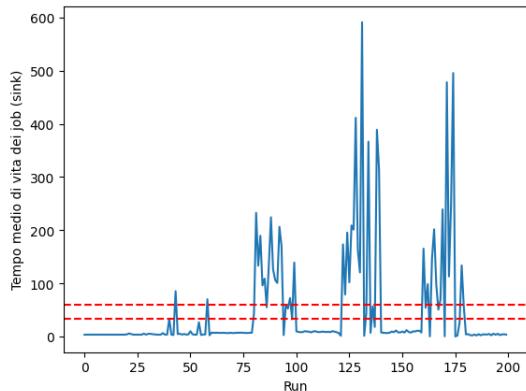


Figura 6.9: Conf. 1 - Lifetime ("sink")

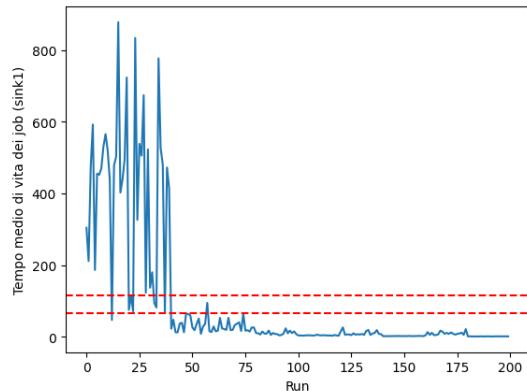


Figura 6.10: Conf. 1 - Lifetime ("sink1")

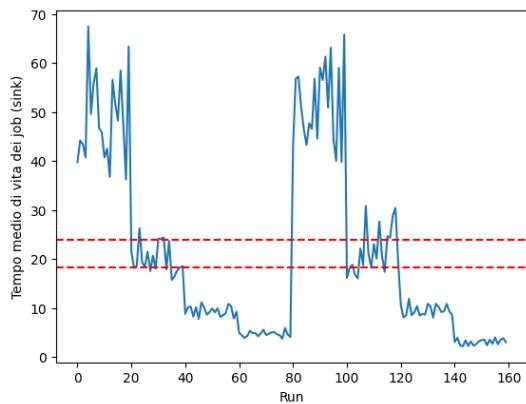


Figura 6.11: Conf. 2 - Lifetime ("sink")

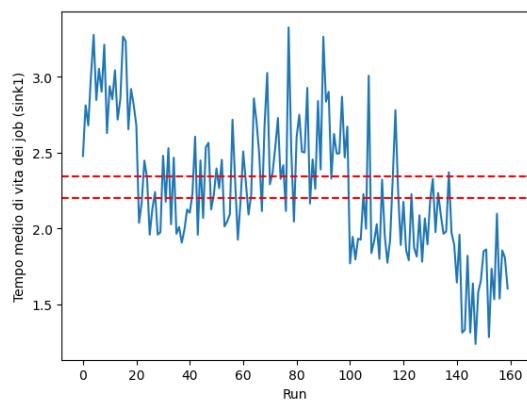


Figura 6.12: Conf. 2 - Lifetime ("sink1")

6.4 Tempo medio per cui il servente è occupato

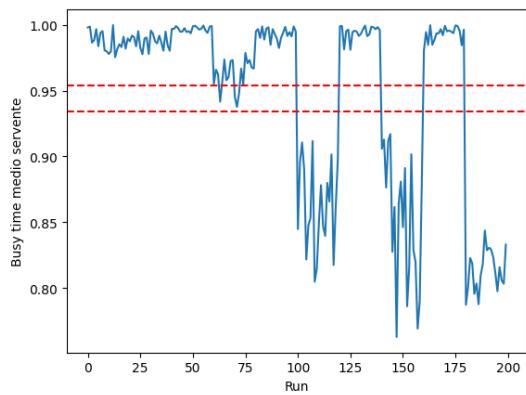


Figura 6.13: Conf. 1 - Busy time

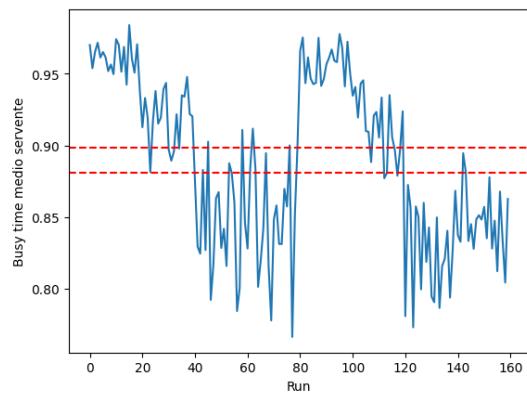


Figura 6.14: Conf. 2 - Busy time

7 Convalida del modello

Per la convalida sono stati confrontati i risultati presenti nel capitolo 5 dell'articolo^[JPY18] (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 1000 secondi di tempo simulato l'una.

7.1 Analisi rispetto a λ_1

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

Parametro	Valore
λ_2	3
μ_1	3
μ_2	4
α	5
K	10
Capacità coda 1	10
N	3
Capacità coda 2	infinita

Tabella 7.1: Analisi rispetto a λ_1

Di seguito i risultati ottenuti:

Valori di λ_1	$E[L_1]$ ^[JPY18]	$E[L_1]$ modello	$E[L_2]$ ^[JPY18]	$E[L_2]$ 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 λ_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 λ_1 succede sempre più rapidamente.

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

Valori di λ_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 λ_1

Un discorso analogo al precedente può essere fatto per $E[W_i]$: è infatti possibile notare come all'aumentare del valore di λ_1 , $E[W_1]$ diminuisca in confronto a quello dell'articolo, mentre $E[W_2]$ aumenti.

7.2 Analisi rispetto a λ_2

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

Parametro	Valore
λ_1	2
μ_1	3
μ_2	4
α	5
K	10
Capacità coda 1	10
N	3
Capacità coda 2	infinita

Tabella 7.4: Analisi rispetto a λ_2

Di seguito i risultati ottenuti:

Valori di λ_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 λ_2

Valori di λ_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 λ_2

7.3 Analisi rispetto a μ_1

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

Parametro	Valore
λ_1	2
λ_2	3
μ_2	4
α	5
K	10
Capacità coda 1	10
N	3
Capacità coda 2	infinita

Tabella 7.7: Analisi rispetto a μ_1

Di seguito i risultati ottenuti:

Valori di μ_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 μ_1

Valori di μ_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 μ_1

Similmente a quanto accaduto precedentemente con λ_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 μ_2

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

Parametro	Valore
λ_1	2
λ_2	3
μ_1	3
α	5
K	10
Capacità coda 1	10
N	3
Capacità coda 2	infinita

Tabella 7.10: Analisi rispetto a μ_2

Di seguito i risultati ottenuti:

Valori di μ_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 μ_2

Valori di μ_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 μ_2

7.5 Analisi rispetto ad α

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

Parametro	Valore
λ_1	2
λ_2	3
μ_1	3
μ_2	4
K	10
Capacità coda 1	10
N	3
Capacità coda 2	infinita

Tabella 7.13: Analisi rispetto ad α

Di seguito i risultati ottenuti:

Valori di α	$E[L_1]$ [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 α

Valori di α	$E[W_1]$ [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 α

7.6 Conclusioni

Visti i risultati a confronto è possibile affermare che il modello realizzato sia simile al modello dell'articolo^[JPY18], 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

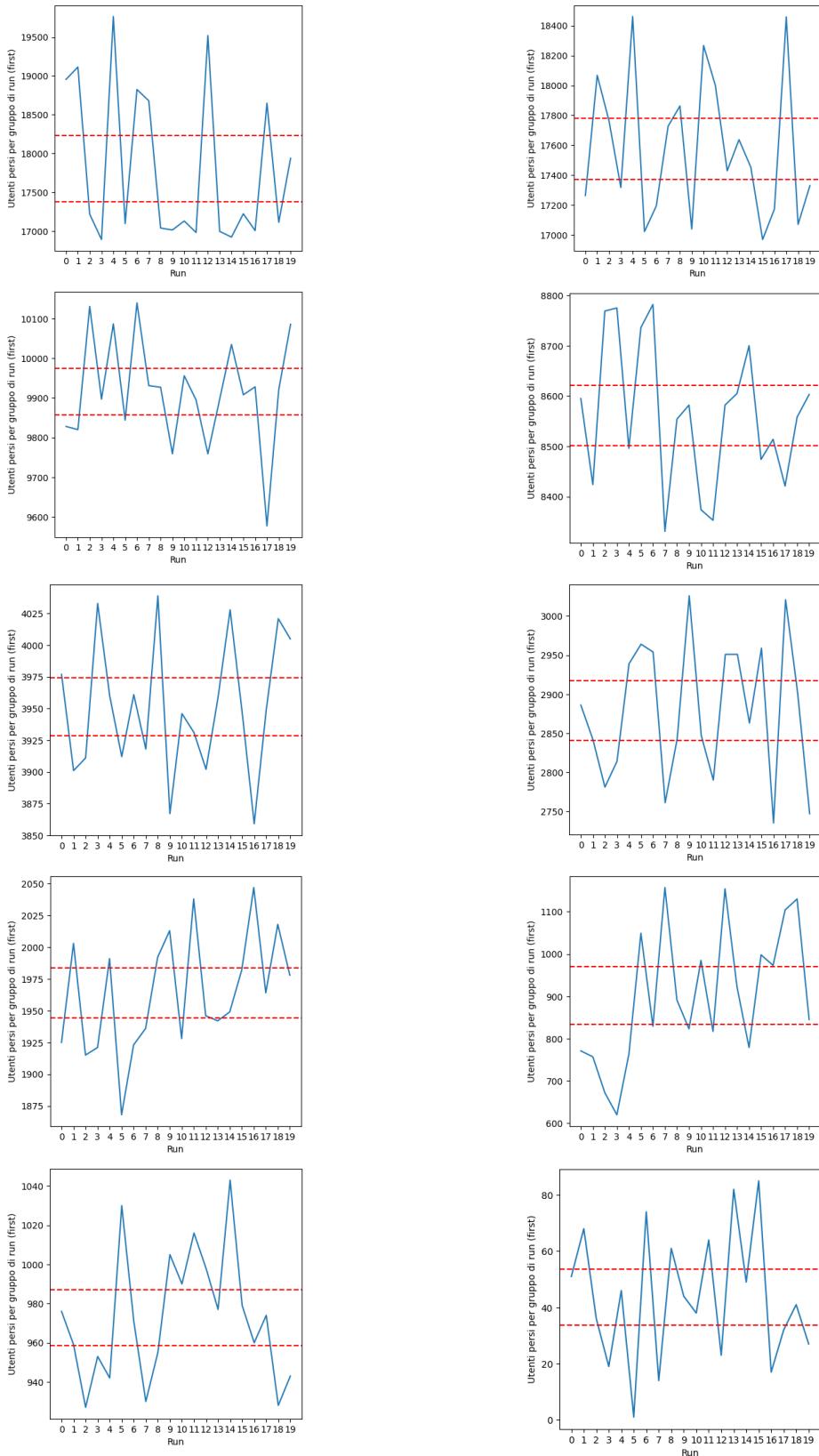
- [JPY18] Amit Jolles, Efrat Perel, and Uri Yechiali. Alternating server with non-zero switch-over times and opposite-queue threshold-based switching policy. *Performance Evaluation*, 126:22–38, 2018.

Appendici

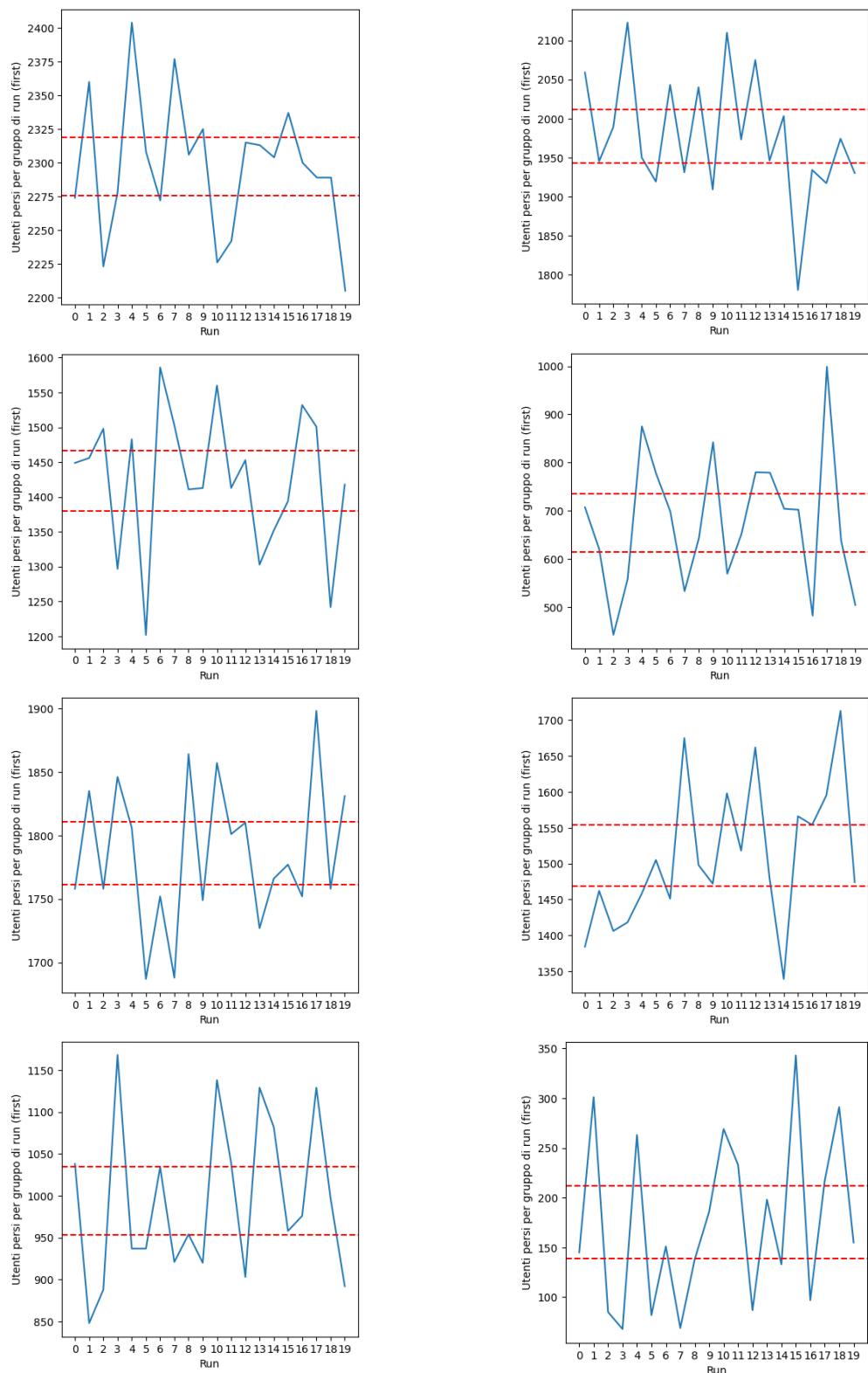
Appendice A Risultati per gruppi di run

In questa appendice saranno mostrati i risultati per gruppi di 20 run, dato che ogni run di simulazione è stata ripetuta 20 volte con gli stessi parametri.

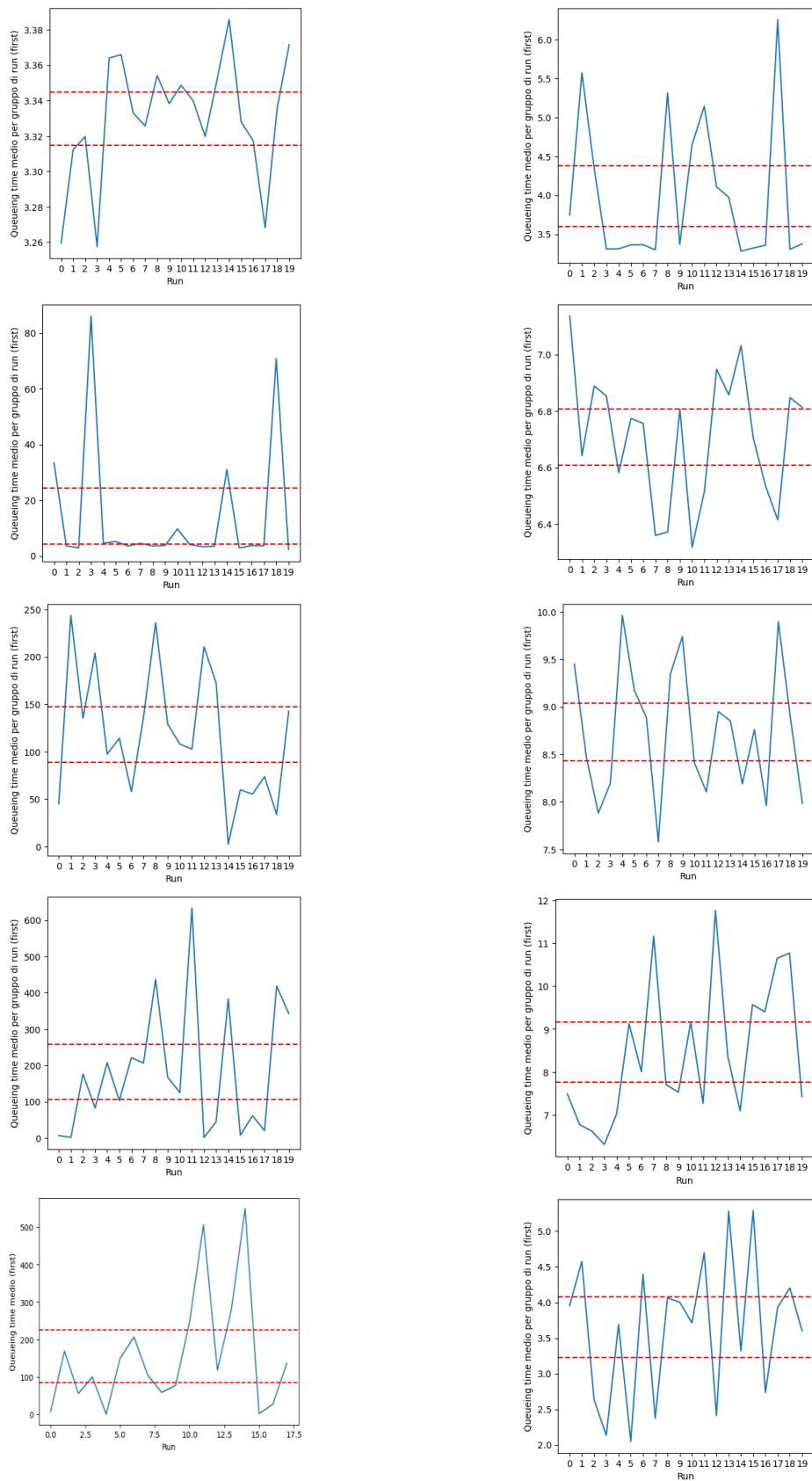
A.1 Utenti persi (coda "first") - Conf. 1



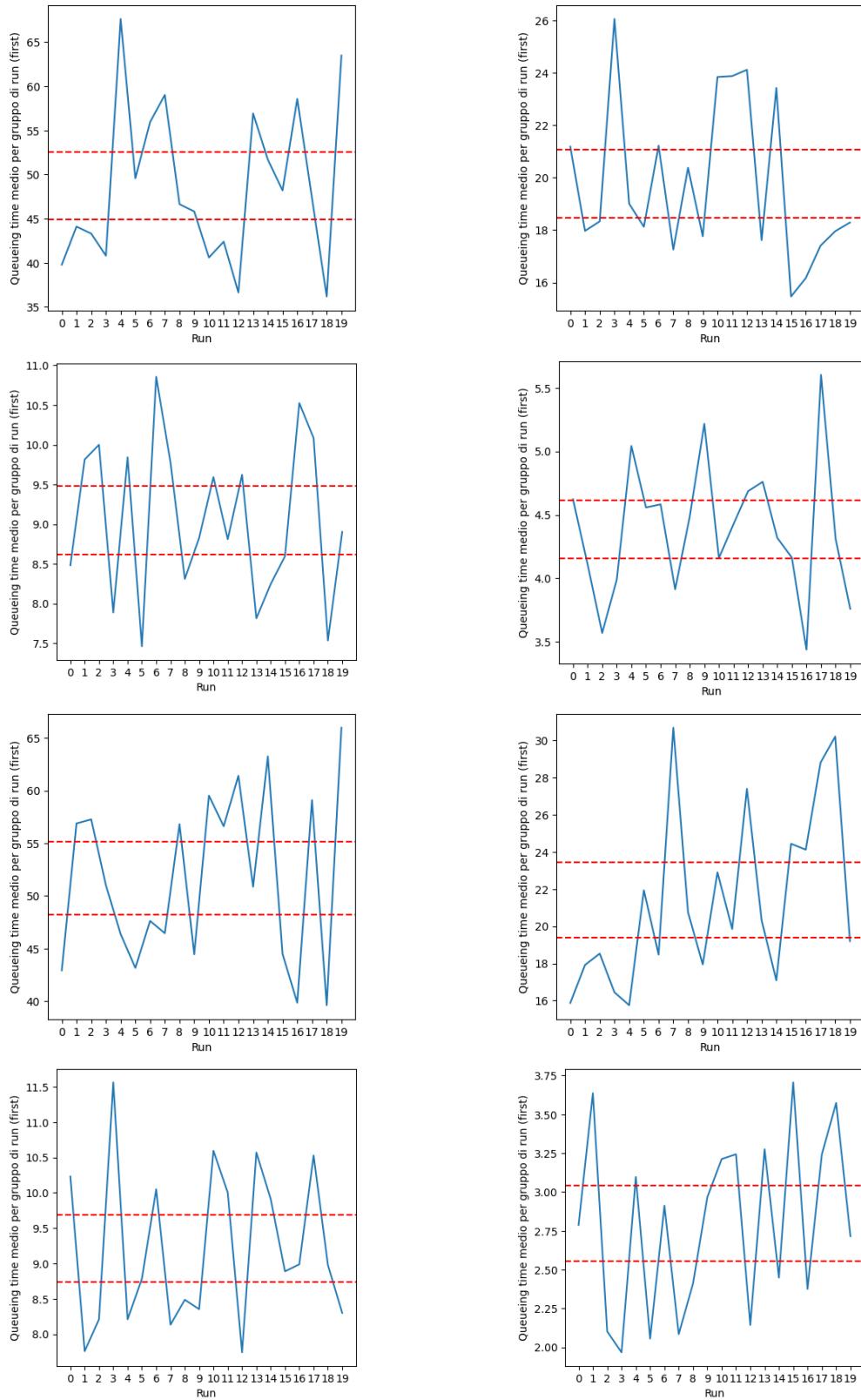
A.2 Utenti persi (coda "first") - Conf. 2



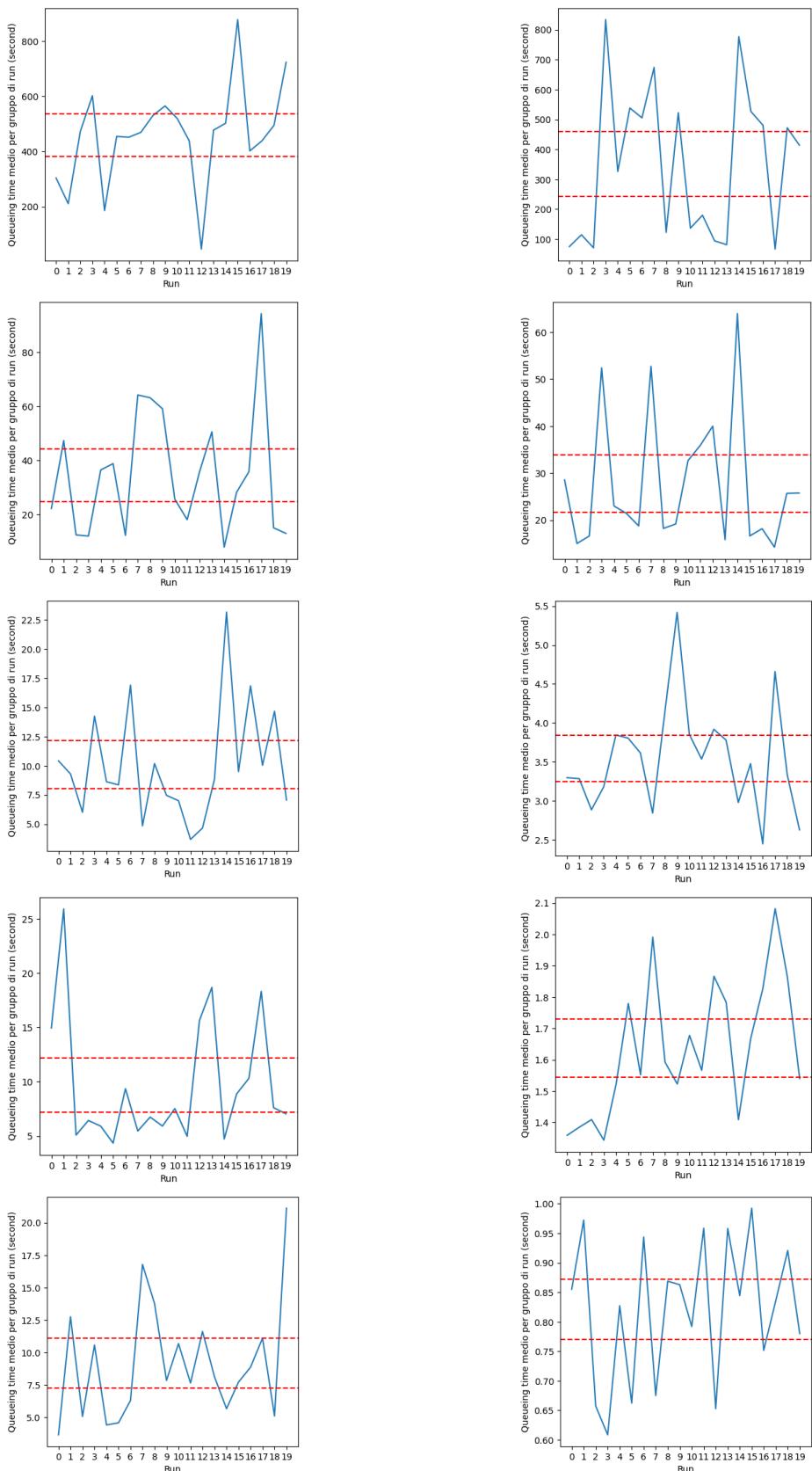
A.3 Tempo di permanenza (coda "first") - Conf. 1



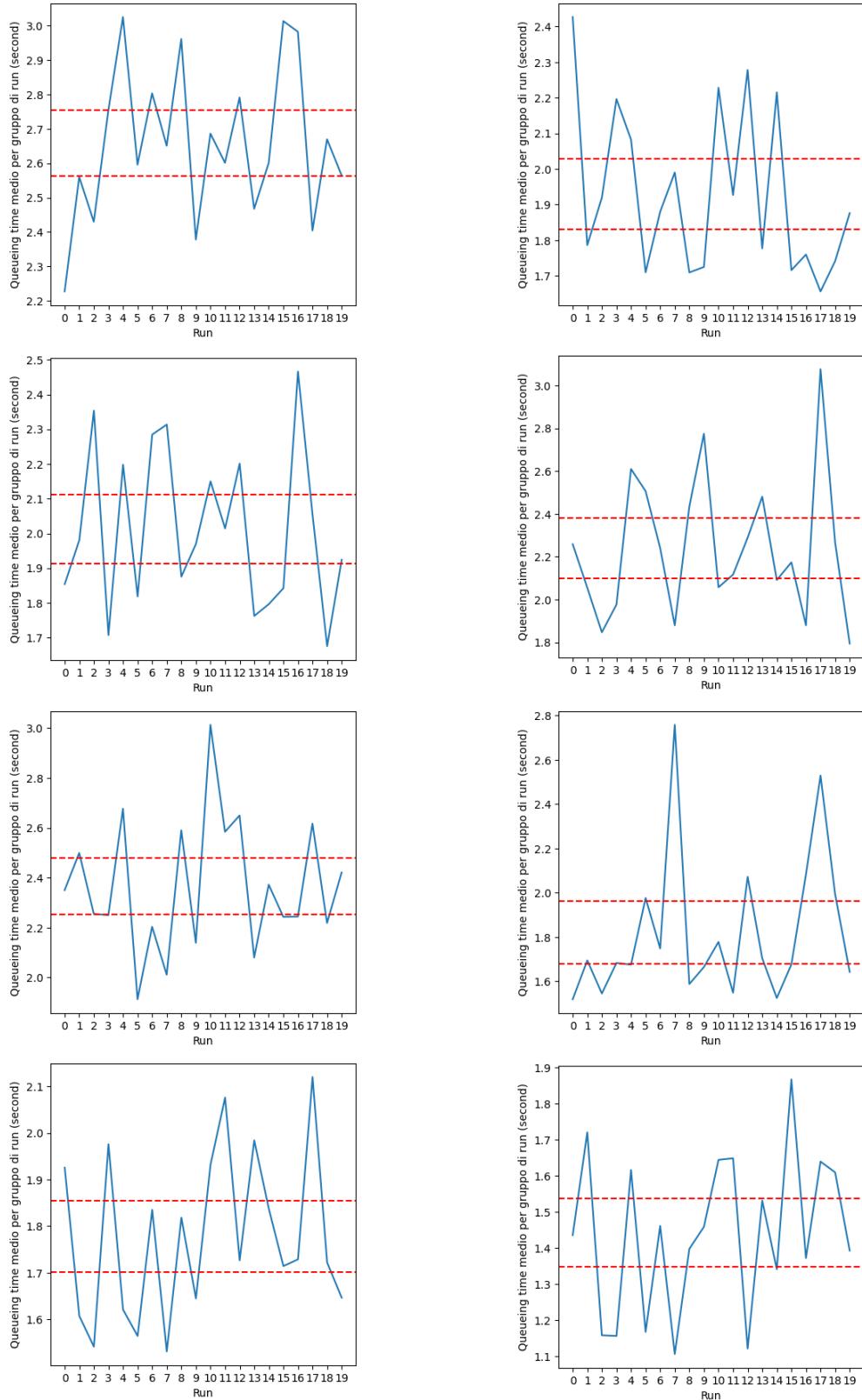
A.4 Tempo di permanenza (coda "first") - Conf. 2



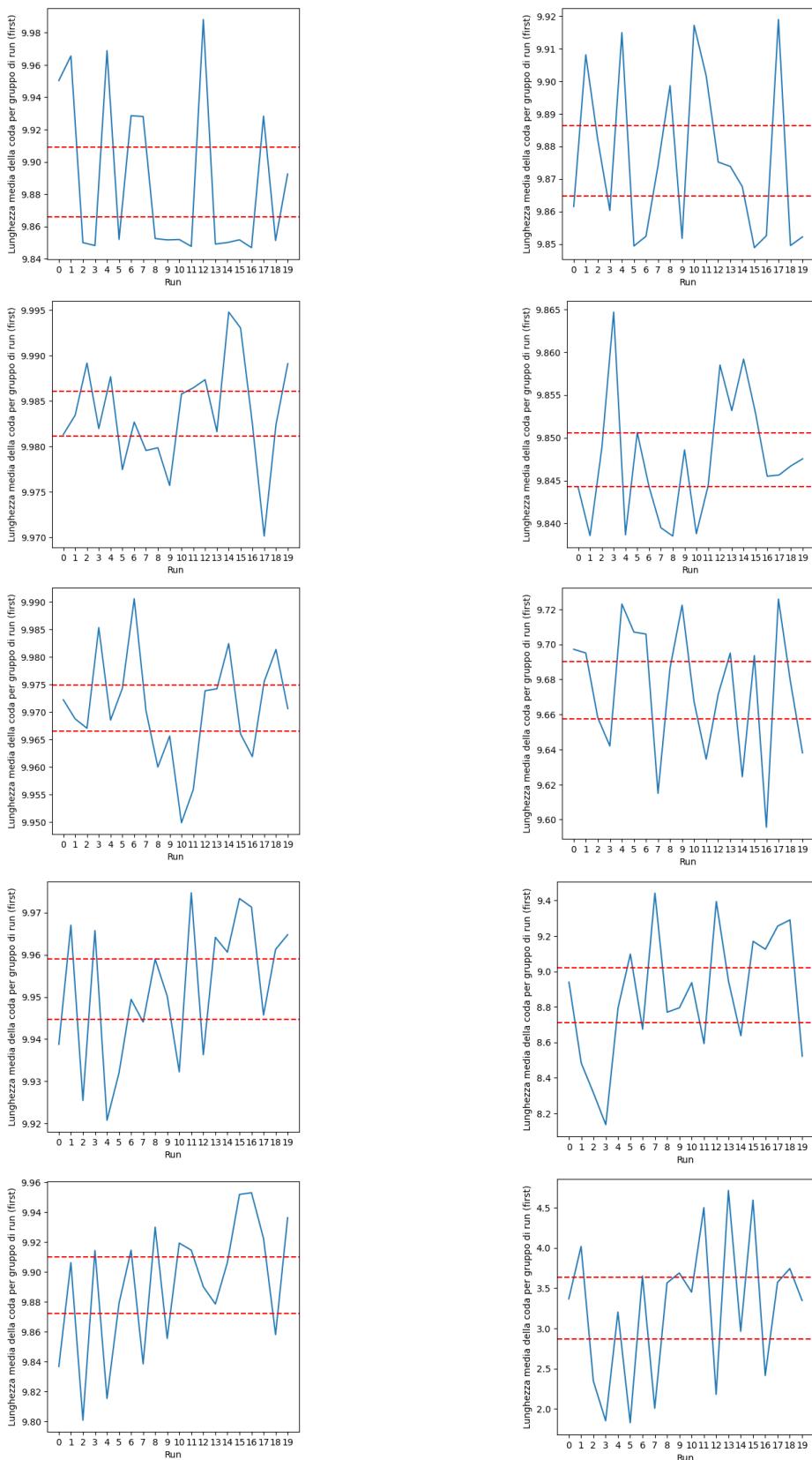
A.5 Tempo di permanenza (coda "second") - Conf. 1



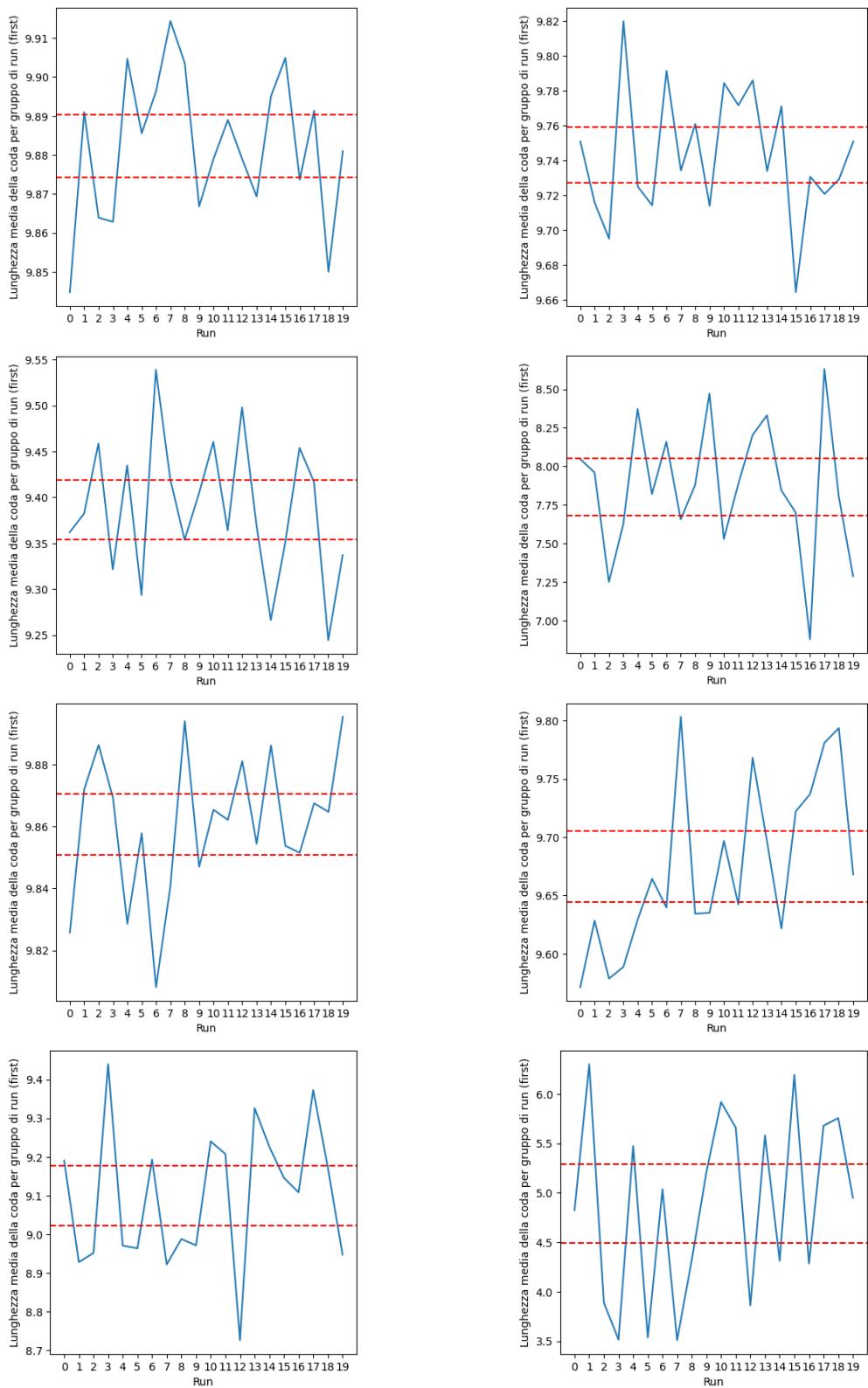
A.6 Tempo di permanenza (coda "second") - Conf. 2



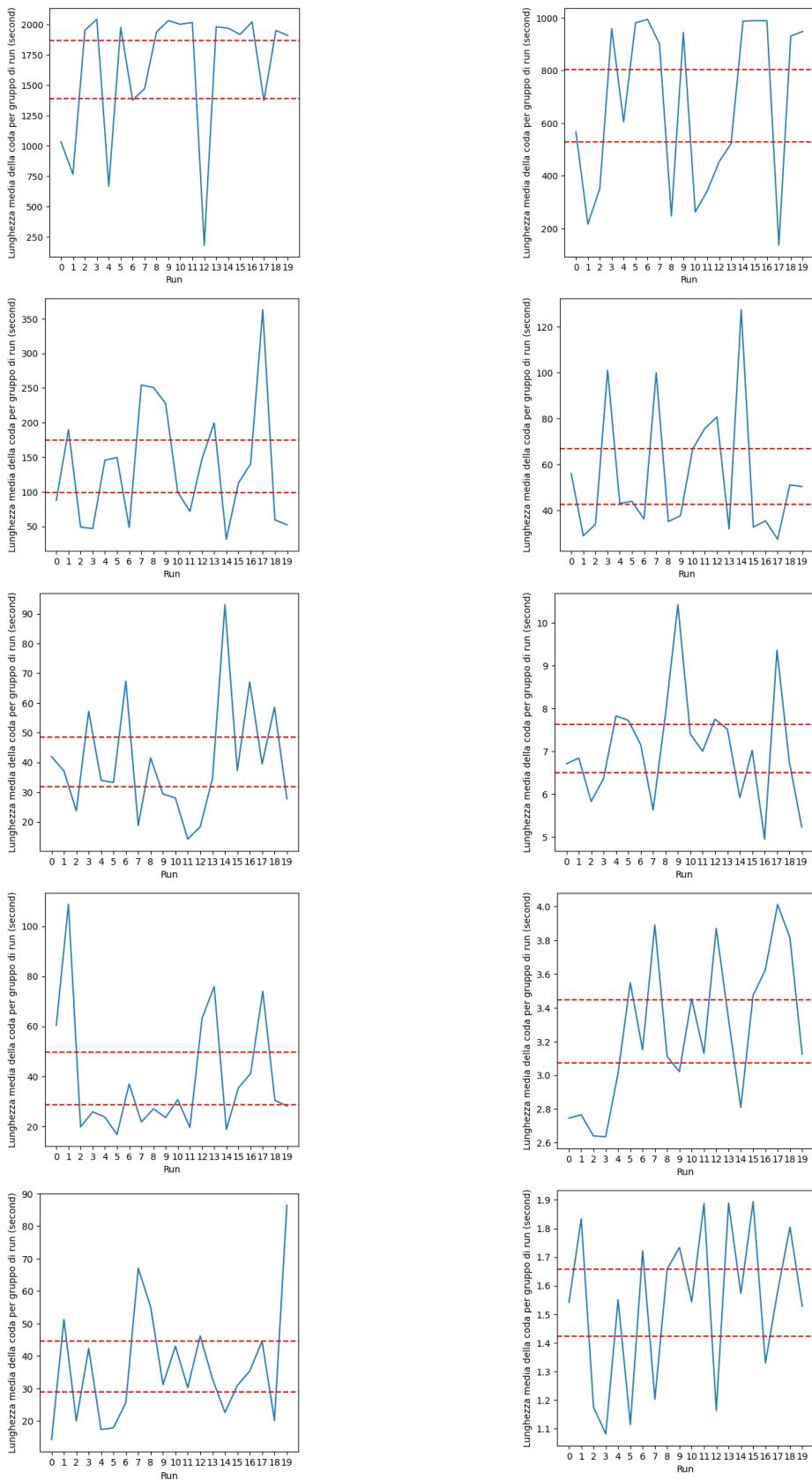
A.7 Numero medio di utenti (coda "first") - Conf. 1



A.8 Numero medio di utenti (coda "first") - Conf. 2



A.9 Numero medio di utenti (coda "second") - Conf. 1



A.10 Numero medio di utenti (coda "second") - Conf. 2

