

Relazione PMCSN

Ingegneria Informatica Magistrale
A.A. 2021/2022

Gestione Settore Stadio

Valerio Crecco 0320452

Ludovico De Santis 0320460

Pierpaolo Spaziani 0316331



0. Manuale	1
1. Introduzione	1
2. Obiettivi.....	1
3. Modello Concettuale.....	2
3.1.Single Server Queue	3
3.2.Multi Server Queue	3
3.3.Multi Server No-Queue	4
3.4.Eventi	4
3.5.Variabili di stato.....	4
4. Modello delle Specifiche.....	4
5. Modello Computazionale *	6
6. Verifica	7
6.1. Verifica PARK ENTRY.....	8
6.2. Verifica TICKET BUY.....	8
6.3. Verifica DOC & PAT	9
6.4. Verifica DOC & PAT D.....	9
6.5. Verifica POLICE CONTROL	10
6.6. Verifica TURNSTILES.....	10
6.7. Verifica TURNSTILES D	11
7. Validazione	11
8. Validazione - Configurazioni	12
8.1. Fascia 13:00-14:00	13
8.2. Fascia 14:00-15:00	13
8.3. Fascia 15:00-16:00	14
9. Infinite Horizon Simulation	14
9.1 Fascia 13:00 – 14:00.....	15
9.2 Fascia 14:00 – 15:00.....	17

9.3 Fascia 15:00 – 16:00	20
10.Finite Horizon Simulation	23
10.1.Configurazione 1.....	24
10.2.Configurazione 2.....	25
10.3.Configurazione 3.....	26
11.Modello Migliorativo.....	28
11.1.Obiettivi.....	28
11.2.Modello Concettuale	29
11.3.Modello Computazionale.....	30
11.4.Verifica	30
11.5.Verifica Doc & Pat	31
11.6.Verifica Turnstiles	31
11.7.Validazione.....	32
11.8.Configurazioni.....	33
11.9.Fascia 13:00 – 14:00.....	33
11.10.Fascia 14:00 – 15:00.....	34
11.11.Fascia 15:00 – 16:00.....	34
11.12.Confronto Configurazioni.....	35
11.13.Infinite Horizon Simulation.....	35
11.14.Infinite Horizon Simulation – Fascia 13:00 – 14:00.....	36
11.15.Infinite Horizon Simulation – Fascia 14:00 – 15:00.....	38
11.16.Infinite Horizon Simulation – Fascia 15:00 – 16:00.....	40
11.17.Finite Horizon Simulation.....	42
12.Conclusioni	44
13.Implementazione	45

0. Manuale

Il sistema è stato implementato per piattaforme *Unix-like* usando il linguaggio C.

Per utilizzare l'applicazione è necessario compilare il *Makefile* ed eseguire il programma '*launcher*' come riportato nella seguente demo:

```
>> make  
    gcc -o launcher launcher.c  
>> ./launcher
```

Lanciata l'applicazione, è possibile selezionare una delle voci del menù che viene visualizzato, digitando un numero da 1 a 3:

```
*****  
          SIMULATION CONSOLE  
*****  
1) Base Simulation  
2) Improved Simulation  
3) Quit  
Select an option [1/2/3]:
```

Selezionata una delle opzioni, si accede ad un successivo menù nel quale è possibile effettuare, sia per il modello *Base* che per il modello *Migliorativo*, l'esecuzione di una delle seguenti simulazioni:

- *Simulazione ad Orizzonte Finito*;
- *Simulazione ad Orizzonte Infinito*;
- *Simulazione di una Singola Esecuzione*.

Per ciascuna di queste, è necessario selezionare una delle possibili configurazioni fornite a schermo.

È possibile inoltre, appena terminate le simulazioni ad orizzonte finito e infinito, previa installazione delle librerie Python utilizzate (*numpy*, *matplotlib* e *csv*), visualizzare i grafici degli andamenti di:

- *Tempo di risposta del sistema*;
- *Percentuale di controllo da parte della Polizia*;
- *Utilizzazioni dei serventi nei singoli centri*.

1. Introduzione

Prendendo spunto dalla nostra esperienza personale in ambito lavorativo, è stato deciso di modellare la gestione del flusso di arrivo presso un settore dello stadio.

L'organizzazione è spesso mal amministrata e non riesce a gestire il carico di utenti che si reca presso lo stadio, con il risultato di malcontenti poiché gli accessi avvengono ad evento già iniziato, nonostante gli utenti si siano recati allo stadio molto prima e abbiano trascorso il restante del tempo in coda.

Nel sistema sono state considerate due tipologie di utenti:

- utenti disabili;
- utenti non disabili.

È stata effettuata questa distinzione poiché, nel caso reale, i disabili hanno percorsi dedicati che permettono di non attendere molto tempo in coda.

Tuttavia, entrambe le tipologie hanno dei centri in comune, come l'accesso allo stadio tramite il parcheggio auto.

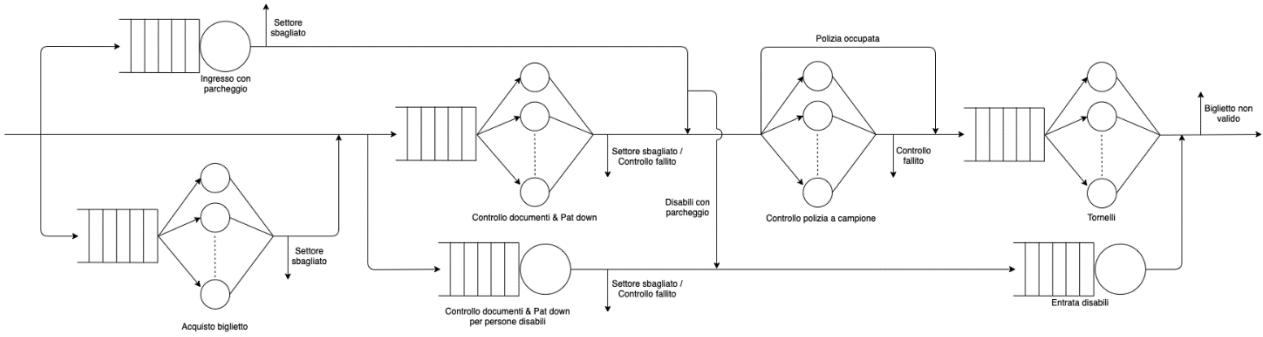
Un ulteriore argomento preso in considerazione è stato quello della sicurezza. Spesso gli stadi vengono ritenuti luoghi poco sicuri a causa dei numerosi spiacevoli eventi storicamente accaduti tra tifoserie. Per questo è stato scelto di inserire un tetto minimo di controlli a campione da parte della polizia del 25% sugli accessi.

2. Obiettivi

Gli obiettivi preposti sono:

- Minimizzare i tempi di accesso al settore dello stadio, con un tetto massimo di accesso pari a sei minuti e mezzo (*QoS 1*);
- Minimizzare i costi relativi al personale addetto da impiegare per l'erogazione del servizio;
- Garantire una copertura dei controlli delle forze dell'ordine di almeno il 25% del flusso di utenti provenienti dal centro *Doc & Pat* (*QoS 2*).

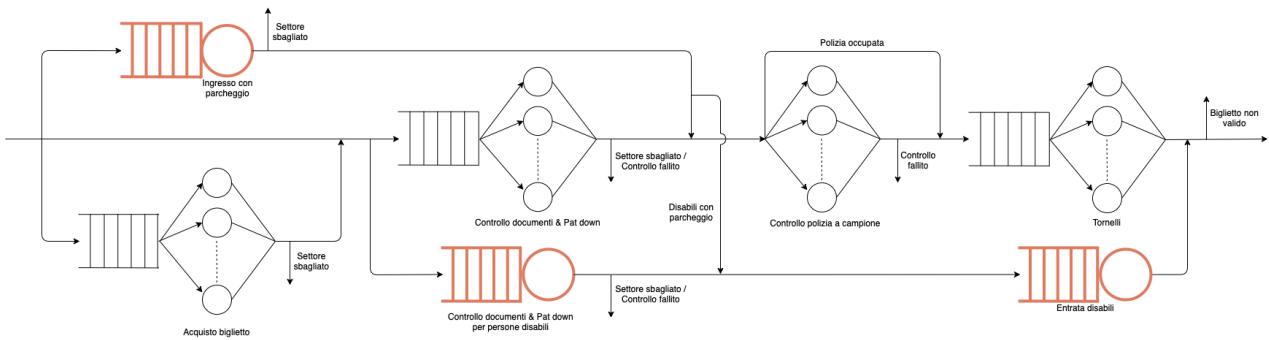
3. Modello Concettuale



I centri modellati nel nostro sistema sono i seguenti:

- **Park Entry**: rappresenta l’accesso al parcheggio del settore dello stadio, a cui possono accedere entrambe le tipologie di utenti;
- **Ticket Buy**: modella la possibilità di acquistare il biglietto in loco il giorno della partita. In questo centro possono accedere entrambe le tipologie di utenti;
- **Doc & Pat**: rappresenta la fase di controllo effettuato dal personale addetto alla verifica dei documenti ed al *pat-down*, relativa ai soli utenti non disabili;
- **Doc & Pat D.**: rappresenta la fase di controllo effettuato da personale addetto alla verifica dei documenti ed al *pat-down*, relativa ai soli utenti disabili;
- **Police Control**: descrive il controllo aggiuntivo di sicurezza da parte delle forze dell’ordine a campione, effettuato solo per gli utenti non disabili;
- **Turnstiles**: modella l’accesso al settore dello stadio attraverso il passaggio ai tornelli per i soli utenti non disabili.
- **Turnstiles D.**: modella l’accesso al settore dello stadio attraverso il passaggio ai tornelli per i soli utenti disabili.

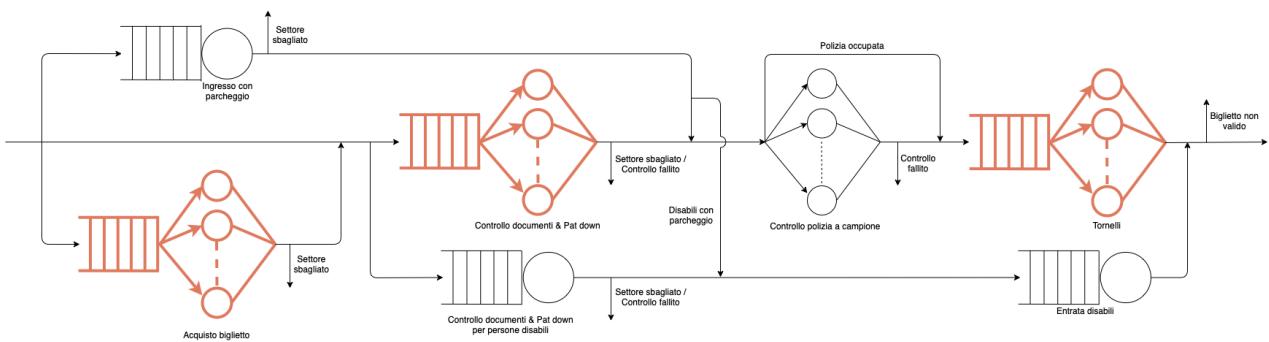
3.1. Single Server Queue



I centri **Park Entry**, **Doc & Pat D.** e **Turnstiles D.** sono stati modellati come una **M/M/1**.

Quando c’è un arrivo in uno di questi centri, se il servente è libero si andrà in servizio, altrimenti l’utente verrà messo in coda.

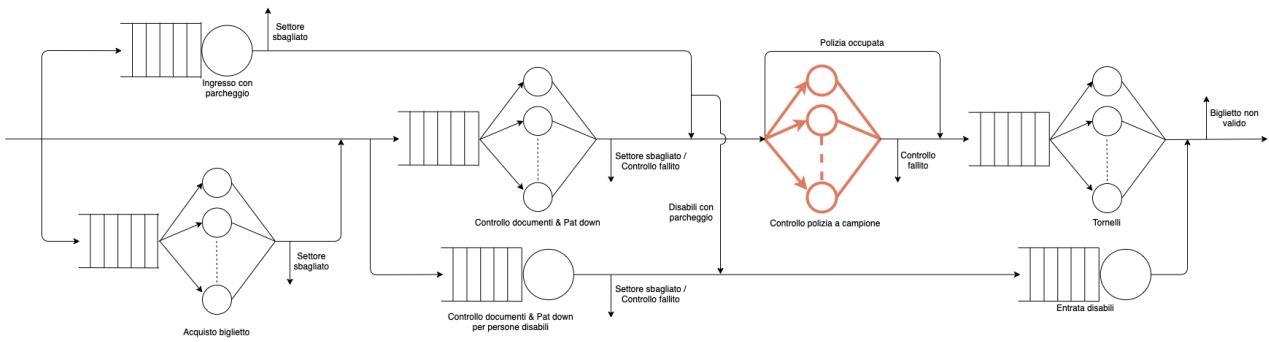
3.2. Multi Server Queue



I centri **Ticket Buy**, **Doc & Pat** e **Turnstiles** sono stati modellati come una **M/M/k**.

Quando c’è un arrivo in uno di questi centri, si controlla se c’è un servente libero, altrimenti l’utente verrà messo in coda. Nel caso ce ne fosse più di uno, in accordo alla politica **Equity**, si sceglierà quello libero da più tempo.

3.3. Multi Server No-Queue



Il centro ***Police Control*** è stato modellato come una **M/M/k/k** (*loss system*), in quanto, se tutti i poliziotti sono occupati, gli utenti accederanno direttamente al centro *Turnstiles*.

Il numero di serventi assegnato a tale centro, è stato individuato per ogni fascia oraria, in modo che il numero di utenti controllati sia almeno il **25%**.

3.4. Eventi

Gli eventi che possono verificarsi nel sistema sono:

- Arrivo di un utente in un centro;
- Completamento di una richiesta di servizio.

3.5. Variabili di stato

Le variabili di stato utilizzate per realizzare il sistema sono:

- Numero di serventi disponibili in ogni fascia oraria, che vengono allocati a seconda dell'affluenza prevista per tale fascia;
- Numero di utenti presenti in un centro;
- Numero di utenti in servizio in un centro;
- Rapporto *controlli su arrivi* nel centro ***Police Control***, che mostra la percentuale di copertura offerta per i controlli.

4. Modello delle Specifiche

I dati considerati per la simulazione fanno riferimento ad informazioni relative alla Curva dello stadio Olimpico, resi disponibili con cadenza annuale dal sito della Lega Calcio.

I tempi di interarrivo e di servizio sono stati modellati con una distribuzione ***Esponenziale***.

Il caso specifico preso in esame, considera una partita che ha luogo alle ore 15:00. Le attività di accesso al settore dello stadio si svolgono in un arco temporale tipicamente di tre ore, che vanno dalle ore 13:00 alle ore 16:00.

I dati utilizzati sono:

- Periodo di osservazione: **3 ore**;
- Numero di utenti al giorno: **7050 (94% capienza curva)**;
- Gli arrivi sono suddivisi in tre fasce orarie con i relativi flussi di arrivo:
 - ▶ **13:00 – 14:00: affluenza media**, con un λ_1 stimato di **0.5875 (30% degli utenti)**;
 - ▶ **14:00 – 15:00: affluenza alta**, con un λ_2 stimato di **1.273056 (65% degli utenti)**;
 - ▶ **15:00 – 16:00: affluenza bassa**, con un λ_3 stimato di **0.09778 (5% degli utenti)**;
- Tempi di servizio per ciascun centro:
 - ▶ **Park Entry: 30 sec/persona**;
 - ▶ **Ticket Buy: 90 sec/persona**;
 - ▶ **Doc & Pat: 20 sec/persona**;
 - ▶ **Doc & Pat D.: 20 sec/persona**;
 - ▶ **Police Control: 40 sec/persona**;
 - ▶ **Turnstiles: 15 sec/persona**;
 - ▶ **Turnstiles D.: 15 sec/persona**;
- Costi personale: **12,5 euro/ora** per membro dello staff;
- Costo medio poliziotto: **23,5 euro/ora** ciascuno;
- Le percentuali di instradamento sono:
 - ▶ **2%** degli arrivi si dirige al centro *Park Entry*;
 - ▶ **8 %** degli arrivi si dirige al centro *Ticket Buy*;
 - ▶ **90%** degli arrivi si divide tra i centri *Doc & Pat* e *Doc & Pat D.*:
 - ▶ **99%** si dirige al centro *Doc & Pat*, riservato agli utenti non disabili;
 - ▶ **1%** si dirige al centro *Doc & Pat D.*, riservato agli utenti disabili;
 - ▶ **65%** degli uscenti da *Park Entry*, si dirige al centro *Turnstiles D.*;
- Le percentuali di perdita del sistema sono:
 - ▶ *Park Entry*: prevede una percentuale di settore sbagliato pari all'**1%**;

- ▶ *Ticket Buy*: prevede una percentuale di settore sbagliato pari al **15%**;
- ▶ *Doc & Pat*: prevede una percentuale di settore sbagliato/controllo fallito pari al **5%**;
- ▶ *Doc & Pat D.*: prevede una percentuale di settore sbagliato/controllo fallito pari all'**1%**;
- ▶ *Police Control*: prevede una percentuale di controllo fallito pari al **5%**;
- ▶ *Turnstiles* e *Turnstiles D.*: prevedono una percentuale di biglietto sbagliato pari al **0.25%**.

5. Modello Computazionale *

A livello implementativo è stato utilizzato un approccio di tipo *Next-Event Simulation*, in cui l'avanzamento del tempo simulato viene effettuato tramite processamento dell'evento successivo.

Il sistema è stato sviluppato utilizzando il linguaggio C.

I file creati sono:

- ***main.c***: prevede l'implementazione di una singola simulazione del sistema nell'arco delle tre fasce orarie. Si occupa di gestire l'arrivo, il completamento e l'instradamento degli utenti versi i corretti centri del sistema e quindi di aggiornare le relative strutture dati;
- ***main_batched.c***: prevede l'implementazione del sistema nel caso di simulazione ad orizzonte infinito, con l'uso di k batches di dimensione b (k e b sono stati specificati nel file *config.h*). Si occupa di gestire l'arrivo, il completamento e l'instradamento degli utenti versi i corretti centri del sistema e quindi di aggiornare le relative strutture dati;
- ***better_call_main.c***: prevede le medesime funzionalità del *main.c* ma appositamente adattate al modello migliorativo. Sono, quindi, stati cambiati il routing degli utenti ed introdotta la gestione delle code di priorità dedicate alle persone disabili nei centri *Doc & Pat* e *Turnstiles*;
- ***better_call_batched.c***: prevede le stesse funzionalità del *better_call_main.c* ma basate su una simulazione ad orizzonte infinito, con l'uso di k batches di dimensione b (k e b sono stati specificati nel file *better_call_config.h*);
- ***config.h***: contiene le diverse possibili configurazioni di serventi da attivare per ciascuna fascia oraria del sistema, i tempi di servizio dei vari nodi, le percentuali di routing, le percentuali di perdita e le macro per indicare ciascuno dei centri come interi;

* per problemi di formattazione, la parte di implementazione è stata aggiunta a fine documento

- ***better_call_config.h***: contiene le medesime informazioni presenti in *config.h* ma adeguate al modello migliorativo;
- ***utils.h***: contiene le strutture dati utilizzate:
 - ▶ ***struct t_center***: modella le informazioni relative al singolo nodo del sistema (*num_servers*, *area*, *num_of_jobs*, etc...);
 - ▶ ***struct t_event_list***: modella le informazioni circa gli arrivi e i serventi del sistema (*status*, *last_event*, *current_t*, etc...);
 - ▶ ***struct sum***: modella le informazioni relative alle statistiche del singolo nodo, riguardanti il numero di job processati e la somma del tempo di servizio;
 - ▶ ***struct t_batch_stat***: modella le informazioni relative alle statistiche del singolo nodo per la versione basata su *batches*, riguardanti il numero di job processati e la somma del tempo di servizio per un singolo batch.

Alcuni campi di queste strutture sono stati duplicati per gestire al meglio il modello migliorativo. Nel modello base questi campi aggiunti non sono stati utilizzati.

- ***queue_time.py***: contiene uno script per il calcolo analitico dei tempi di coda dei vari centri;
- ***population.py***: contiene uno script per il calcolo analitico delle popolazioni medie nei vari centri.

6. Verifica

Il processo di verifica è stato effettuato sui dati scaturiti da una simulazione della fascia oraria centrale, 14:00 - 15:00, con $\lambda_2 = 1.273056$.

Per assicurare che i risultati siano conformi alle specifiche, sono stati osservati i seguenti criteri:

- Il numero di utenti in un centro deve essere uguale al numero di utenti in coda più il numero di utenti in servizio;
- Il tempo di risposta deve essere uguale al tempo trascorso in coda più il tempo di servizio;
- Il numero di arrivi deve essere uguale al numero di completamenti più il numero di jobs *dropped*;
- Il numero di utenti in ingresso in un centro è determinato dalla probabilità di routing indicata nel modello delle specifiche.

6.1. Verifica PARK ENTRY

```
for 114 jobs the service node statistics:
# arrivals ..... = 115
# dropped ..... = 1
# completions ..... = 114
avg interarrivals .. = 31.29885
avg # in node ..... = 4.06918
avg # in queue ..... = 3.24910
avg utilization .... = 0.82008
avg wait ..... = 134.04241
avg delay ..... = 107.02838
avg service ..... = 27.01403
```

$$E(N_S) = E(N_Q) + \rho = 3.24910 + 0.82008 = 4.06918$$

$$E(T_S) = E(T_Q) + E(S) = 107.02838 + 27.01403 = 134.04241$$

$$\text{Arrivi} = \text{Completamenti} + \text{Dropped} = 114 + 1 = 115$$

$$\#\text{expected_arrivals} = \lambda_2 \cdot 3600 \cdot p_{PE} = 1.273056 \cdot 3600 \cdot 0.02 = 91.66003$$

Gli arrivi effettivi (115) risultano outlier rispetto a quelli teorici (91.66003).

Effettuando la simulazione per 256 ripetizioni, i valori medi risultano: 92.25781.

6.2. Verifica TICKET BUY

```
for 299 jobs the service node statistics:
# arrivals ..... = 367
# dropped ..... = 68
# completions ..... = 299
avg interarrivals .. = 9.79972
avg # in node ..... = 10.77664
avg # in queue ..... = 1.97292
avg utilization .... = 0.80034
avg wait ..... = 113.92926
avg delay ..... = 20.85747
avg service ..... = 93.07180
```

$$E(N_S) = E(N_Q) + m\rho = 1.97292 + 11 \cdot 0.8003 = 10.77664$$

$$E(T_S) = E(T_Q) + E(S) = 20.85747 + 93.07180 = 113.92926$$

$$\text{Arrivi} = \text{Completamenti} + \text{Dropped} = 299 + 68 = 367$$

$$\#\text{expected_arrivals} = \lambda_2 \cdot 3600 \cdot p_{TB} = 1.273056 \cdot 3600 \cdot 0.08 = 366.6401$$

Gli arrivi effettivi (367) risultano outlier rispetto a quelli teorici (366.64013).

Effettuando la simulazione per 256 ripetizioni, i valori medi risultano: 366.89844.

6.3. Verifica DOC & PAT

```

for 4094 jobs the service node statistics:
# arrivals ..... = 4307
# dropped ..... = 213
# completions ..... = 4094
avg interarrivals .. = 0.83595
avg # in node ..... = 26.31702
avg # in queue ..... = 3.69777
avg utilization .... = 0.86997
avg wait ..... = 23.21443
avg delay ..... = 3.26183
avg service ..... = 19.95260

```

$$E(N_S) = E(N_Q) + m\rho = 3.69777 + 26 \cdot 0.87 = 26.31702$$

$$E(T_S) = E(T_Q) + E(S) = 3.26183 + 19.95260 = 23.21443$$

$$\text{Arrivi} = \text{Completamenti} + \text{Dropped} = 4094 + 213 = 4307$$

#expected_arrivals = 4391.98056 (*calcolo effettuato con ‘population.py’*)

Gli arrivi effettivi (4307) risultano outlier rispetto a quelli teorici (366.64013).

Effettuando la simulazione per 256 ripetizioni, i valori medi risultano: 4392.05859.

6.4. Verifica DOC & PAT D

```

for 27 jobs the service node statistics:
# arrivals ..... = 29
# dropped ..... = 2
# completions ..... = 27
avg interarrivals .. = 123.75219
avg # in node ..... = 0.15932
avg # in queue ..... = 0.01035
avg utilization .... = 0.14898
avg wait ..... = 19.90806
avg delay ..... = 1.29268
avg service ..... = 18.61538

```

$$E(N_S) = E(N_Q) + \rho = 0.01035 + 0.14898 = 0.15932$$

$$E(T_S) = E(T_Q) + E(S) = 1.29268 + 18.61538 = 19.90806$$

$$\text{Arrivi} = \text{Completamenti} + \text{Dropped} = 27 + 2 = 29$$

#expected_arrivals = 44.36344 (*calcolo effettuato con ‘population.py’*)

Gli arrivi effettivi (29) risultano outlier rispetto a quelli teorici (44.36344).

Effettuando la simulazione per 256 ripetizioni, i valori medi risultano: 44.55469.

6.5. Verifica POLICE CONTROL

```
for 1114 jobs the service node statistics:  
# arrivals ..... = 1171  
# dropped ..... = 57  
# completions ..... = 1114  
avg interarrivals .. = 3.24444  
avg # in node ..... = 12.15428  
avg # in queue ..... = 0.00000  
avg utilization .... = 0.93494  
avg wait ..... = 39.90212  
avg delay ..... = 0.00000  
avg service ..... = 39.90212
```

$$E(N_S) = m\rho = 13 \cdot 0.93494 = 12.15428$$

$$E(T_S) = E(S) = 39.90212 = 39.90212$$

Arrivi = Completamenti + Dropped = 1114 + 57 = 1171

#expected_arrivals = 1051.03543 (*calcolo effettuato con ‘population.py’*)

Gli arrivi effettivi (1171) rispettano il lower bound prefissato dal QoS (1051.03543, 25% degli arrivi).

6.6. Verifica TURNSTILES

```
for 4071 jobs the service node statistics:  
# arrivals ..... = 4078  
# dropped ..... = 7  
# completions ..... = 4071  
avg interarrivals .. = 0.94271  
avg # in node ..... = 18.71583  
avg # in queue ..... = 3.04852  
avg utilization .... = 0.82460  
avg wait ..... = 17.68643  
avg delay ..... = 2.88084  
avg service ..... = 14.80559
```

$$E(N_S) = E(N_Q) + m\rho = 3.04852 + 19 \cdot 0.8246 = 18.71583$$

$$E(T_S) = E(T_Q) + E(S) = 2.88084 + 14.80559 = 17.68643$$

Arrivi = Completamenti + Dropped = 4071 + 7 = 4078

#expected_arrivals = 4151.58995 (*calcolo effettuato con ‘population.py’*)

Gli arrivi effettivi (4078) risultano outlier rispetto a quelli teorici (4151.58995).

Effettuando la simulazione per 256 ripetizioni, i valori medi risultano: 4147.52343.

6.7. Verifica TURNSTILES D

```

for 99 jobs the service node statistics:
# arrivals ..... = 100
# dropped ..... = 1
# completions ..... = 99
avg interarrivals .. = 37.88205
avg # in node ..... = 0.42644
avg # in queue ..... = 0.09024
avg utilization .... = 0.33620
avg wait ..... = 16.16843
avg delay ..... = 3.42142
avg service ..... = 12.74700

```

$$E(N_S) = E(N_Q) + \rho = 0.09024 + 0.33620 = 0.42644$$

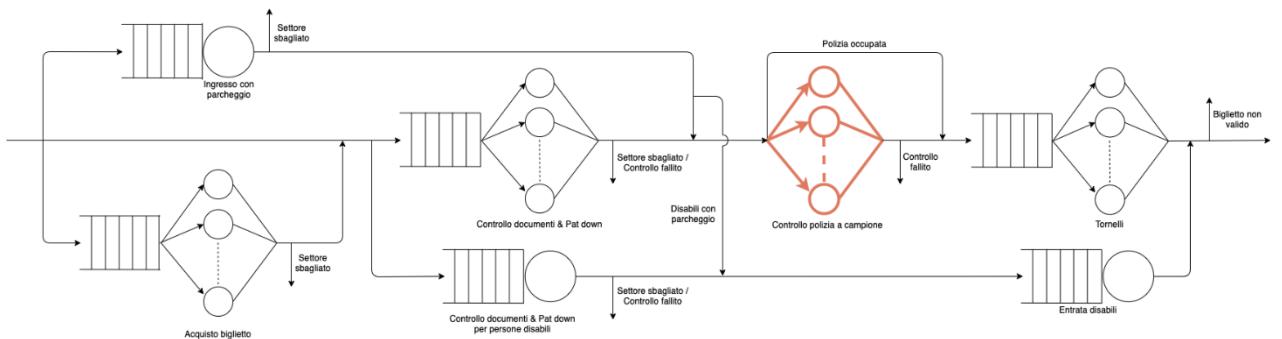
$$E(T_S) = E(T_Q) + E(S) = 3.42142 + 12.74700 = 16.16843$$

$$\text{Arrivi} = \text{Completamenti} + \text{Dropped} = 99 + 1 = 100$$

#expected_arrivals = 101.12848 (*calcolo effettuato con 'population.py'*)

Gli arrivi effettivi (100) risultano outlier rispetto a quelli teorici (101.12848). Effettuando la simulazione per 256 ripetizioni, i valori medi risultano: 101.69141.

7. Validazione



Ciò che ci si aspetta dal sistema è che, aumentando il numero di serventi nel centro “Doc & Pat”, la percentuale relativa al QoS del controllo della polizia (25%) diminuisca:

Mettendo **22** serventi sul centro *Doc & Pat* e **11** su *Police Control*:

POLICE CONTROL

```

# arrivals ..... = 1057
# dropped ..... = 44
# completions ..... = 1013

```

TURNSTILES

```

# arrivals ..... = 4152
# dropped ..... = 7
# completions ..... = 4145

```

$$\frac{arrivi_{PC}}{arrivi_T} = \frac{1057}{4152} = 0,255$$

Mettendo **30** serventi sul centro Doc & Pat e **11** su Police Control:

POLICE CONTROL

arrivals = 1019
dropped = 48
completions = 971

TURNSTILES

arrivals = 4120
dropped = 10
completions = 4110

$$\frac{arrivi_{PC}}{arrivi_T} = \frac{1019}{4120} = 0,247$$

Sono stati confrontati i risultati delle simulazioni con i valori teorici scaturiti dalle seguenti formule:

$$E(T_S)_{KP} = \frac{\rho E(S)}{1 - \rho} + E(S) \quad p(0) = \left[\sum_{i=0}^{m-1} \frac{(m\rho)^i}{i!} + \frac{(m\rho)^m}{m!(1-\rho)} \right]^{-1} \quad \lambda' = \lambda(1 - p_{loss})$$

$$E(T_S)_{Erlang} = \frac{P_Q E(S)}{1 - \rho} + E(S_i) \quad P_Q = \frac{(m\rho)^m}{m!(1-\rho)} p(0) \quad \rho = \frac{\lambda'}{\mu}$$

$$E(T_S) = \sum_{i=0}^m v_i \cdot E(T_{S_i}) \quad E(T_Q) = \frac{P_Q E(S)}{1 - \rho} \quad \pi_m = \frac{\left(\frac{\lambda}{\mu}\right)^m}{\sum_{j=0}^m \left(\frac{\lambda}{\mu}\right)^j \frac{1}{j!}}$$

8. Validazione - Configurazioni

Per le diverse fasce orarie, sono state utilizzate diverse configurazioni di serventi attivi al fine di garantire un efficiente erogazione del servizio:

	Park Entry	Ticket Buy	Doc & Pat	Doc & Pat D.	Police Control	Turnstiles	Turnstiles D.
13:00 – 14:00	1	5	13	1	6	9	1
14:00 – 15:00	1	11	26	1	13	19	1
15:00 – 16:00	1	2	3	1	2	2	1

Per il calcolo del *Risultato Sperimentale*, sono state mediati i risultati di 256 simulazioni.

Il valore presente nella colonna *Risultato Analitico* è stato ottenuto eseguendo lo script *queue_time.py*.

8.1. Fascia 13:00-14:00

Statistica	Risultato Analitico	Risultato Sperimentale ($\alpha = 0.05$)
$E(T_{Q_{PARK_ENTRY}})$	16.33205	15.97357 ± 0.77346
$E(T_{Q_{TICKET_BUY}})$	75.52453	77.72653 ± 6.08385
$E(T_{Q_{DOC_PAT}})$	6.00957	5.96558 ± 0.19859
$E(T_{Q_{DOC_PAT_D}})$	2.56674	2.38105 ± 0.19344
$E(T_{Q_{POLICE_CONTROL}})$	0.00000	0.00000 ± 0.00000
$E(T_{Q_{TURNSTILES}})$	9.55838	9.67299 ± 0.37824
$E(T_{Q_{TURNSTILES_D}})$	3.62097	3.40926 ± 0.22181

8.2. Fascia 14:00-15:00

Statistica	Risultato Analitico	Risultato Sperimentale ($\alpha = 0.05$)
$E(T_{Q_{PARK_ENTRY}})$	97.02908	92.91735 ± 7.42203
$E(T_{Q_{TICKET_BUY}})$	22.94744	22.78496 ± 2.04092
$E(T_{Q_{DOC_PAT}})$	8.33916	8.11687 ± 0.43658
$E(T_{Q_{DOC_PAT_D}})$	6.54152	6.16607 ± 0.38180
$E(T_{Q_{POLICE_CONTROL}})$	0.00000	0.00000 ± 0.00000
$E(T_{Q_{TURNSTILES}})$	5.2815	5.26622 ± 0.27694
$E(T_{Q_{TURNSTILES_D}})$	10.92325	10.59737 ± 0.49544

8.3. Fascia 15:00-16:00

Statistica	Risultato Analitico	Risultato Sperimentale ($\alpha = 0.05$)
$E(T_{Q_{PARK_ENTRY}})$	1.86973	1.76706 ± 0.11201
$E(T_{Q_{TICKET_BUY}})$	12.72913	12.81082 ± 0.40134
$E(T_{Q_{DOC_PAT}})$	6.87433	6.88404 ± 0.08748
$E(T_{Q_{DOC_PAT_D}})$	0.38591	0.36980 ± 0.05437
$E(T_{Q_{POLICE_CONTROL}})$	0.00000	0.00000 ± 0.00000
$E(T_{Q_{TURNSTILES}})$	11.84891	11.72581 ± 0.13766
$E(T_{Q_{TURNSTILES_D}})$	0.50170	0.48458 ± 0.03605

9. Infinite Horizon Simulation

La simulazione ad orizzonte infinito è stata effettuata su un lasso di tempo molto più lungo rispetto al tempo reale, andando a prendere in considerazione una singola fascia per volta.

Si è considerato, quindi, un sistema *statico*, con un numero di serventi e flusso di arrivo che rimangono invariati al cambio di fascia oraria.

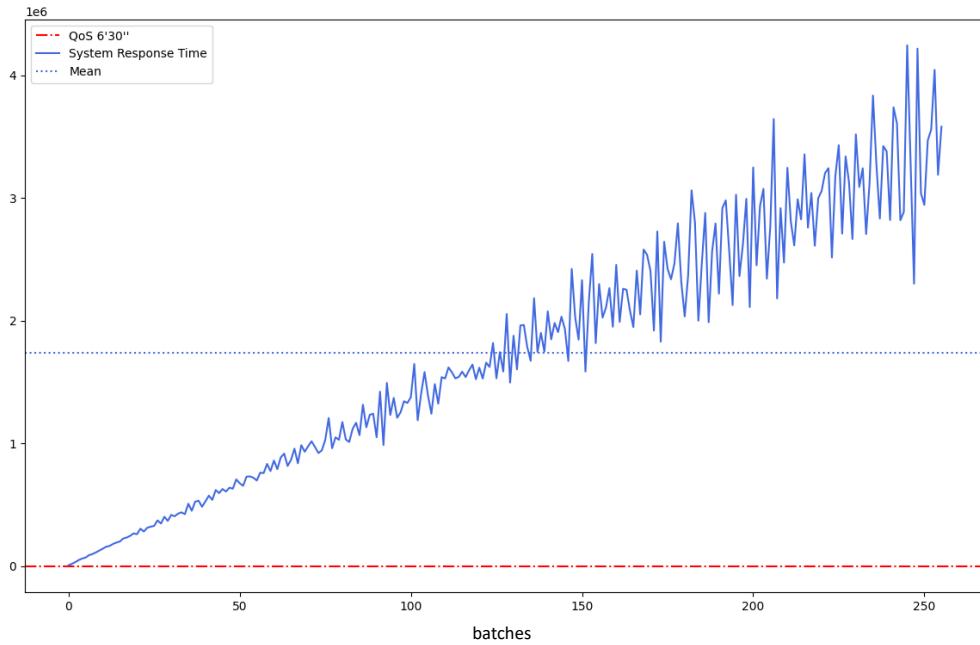
Per ogni fascia, si è cercato di raggiungere i seguenti obiettivi:

- Analizzare il comportamento del sistema allo stato stazionario;
- Individuare la configurazione migliore per rispettare i QoS e minimizzare i costi.

Per far ciò è stato utilizzato il metodo delle ***Batch Means***. È stata considerata un'esecuzione suddivisa in $k = 256$ batches di dimensione $b = 8192$. Tali valori sono stati scelti cercando il valore di b per cui *l'autocorrelazione* del campione fosse minore di 0,2 per un lag $j = 1$. A questo punto, sono state calcolate le statistiche per ogni batch per un'ulteriore analisi attraverso i successivi grafici.

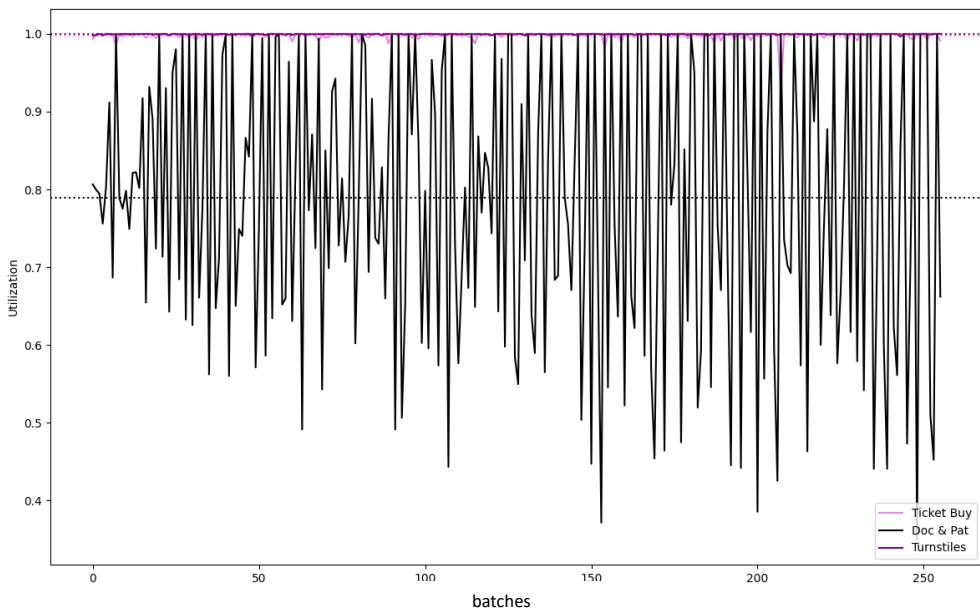
9.1 Fascia 13:00 – 14:00

Configurazione: {1,3,10,1,6,5,1}



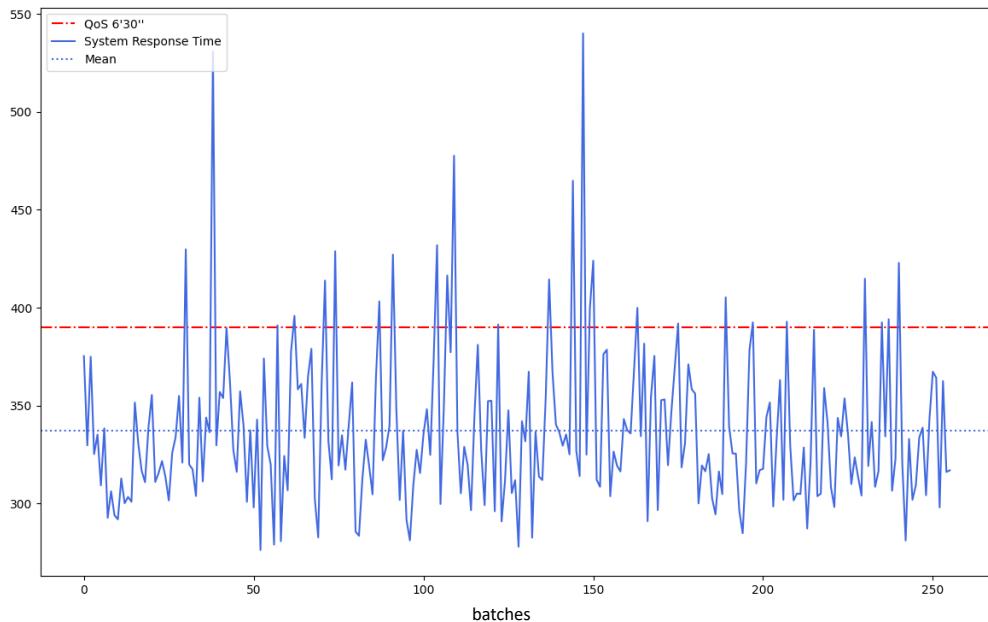
Nella prima fascia oraria, utilizzando la suddetta configurazione si può osservare come il sistema sia fortemente sottodimensionato e NON stazionario. Ciò si può osservare dall’andamento del tempo di risposta che cresce fino all’infinito.

Tale crescita del tempo di risposta è giustificata anche dal successivo grafico delle utilizzazioni. Infatti, si può notare che per i centri *Ticket Buy* e *Turnstiles* si ha un’utilizzazione pari a 1. Il centro *Doc & Pat* ha media di utilizzazione inferiore allo 0.8, tuttavia è molto variabile e spesso arriva al valore 1. Questo andamento è legato al fatto che il centro ha arrivi che dipendono anche da *Ticket Buy*:

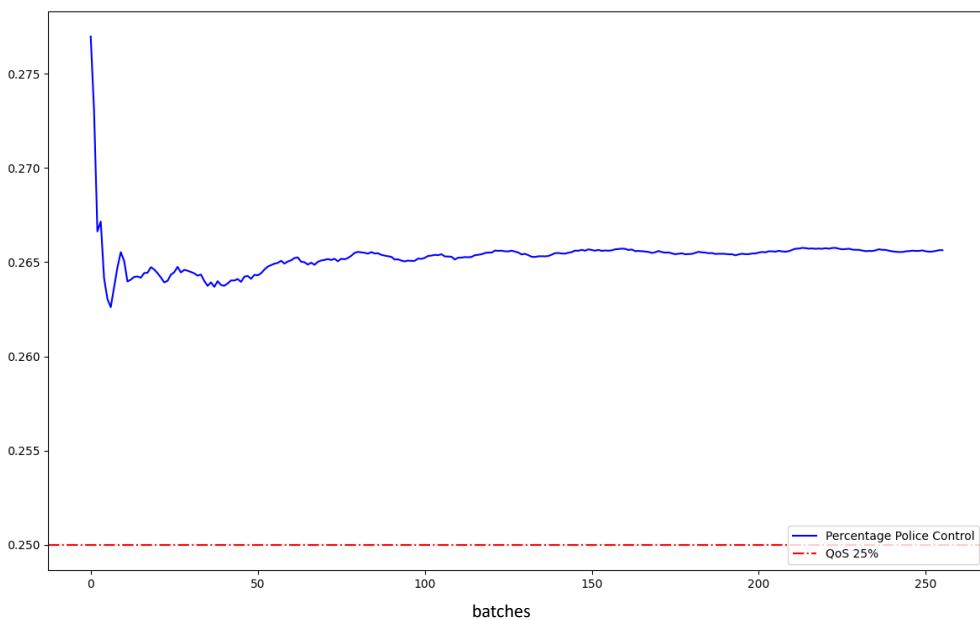


Al fine di rendere il sistema stazionario nella fascia 13:00 – 14:00, è stato fatto un ridimensionamento dei serventi giungendo alla configurazione:
{1,5,13,1,6,9,1}.

Passando a tale configurazione, è possibile avere un tempo di risposta medio inferiore a 350 secondi, che quindi rispetta il QoS previsto (6 minuti e mezzo):

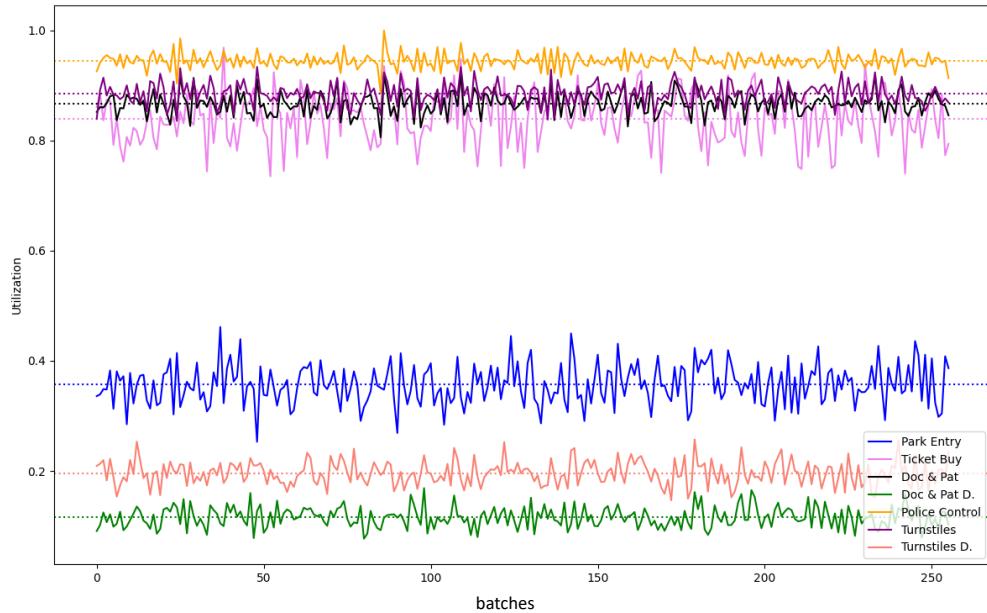


Con questa configurazione viene rispettato anche il QoS dei controlli della polizia, che prevedeva una percentuale minima di controlli del 25%:



Come si può osservare dal grafico, con tale configurazione si ottiene una percentuale di circa il 26.5 % di controlli, che soddisfa il QoS. Inoltre, il numero di poliziotti considerati risulta essere il minimo sufficiente per garantire il QoS desiderato. La percentuale iniziale di controlli è inizialmente molto alta poiché i primi utenti vengono controllati tutti.

La stazionarietà del sistema è anche evidenziata dall'andamento delle utilizzazioni successivamente riportato:

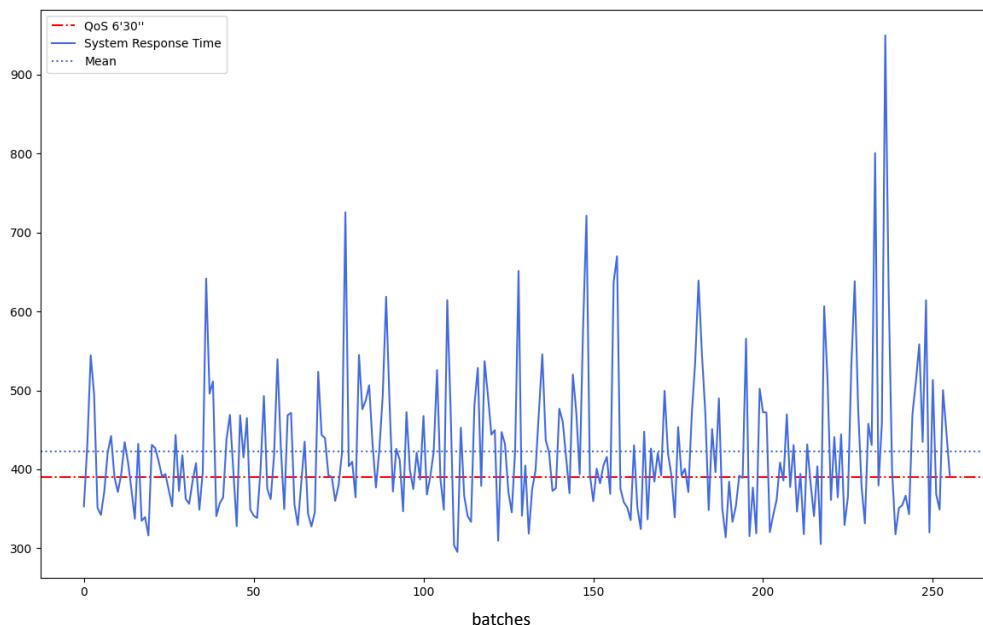


Come illustrato nel grafico, il centro **Police Control** ha l'utilizzazione prossima al valore 1 e, non avendo coda, ne implica il corretto funzionamento. Gli altri centri multi-server hanno un'utilizzazione compresa tra 0.8 e 0.9.

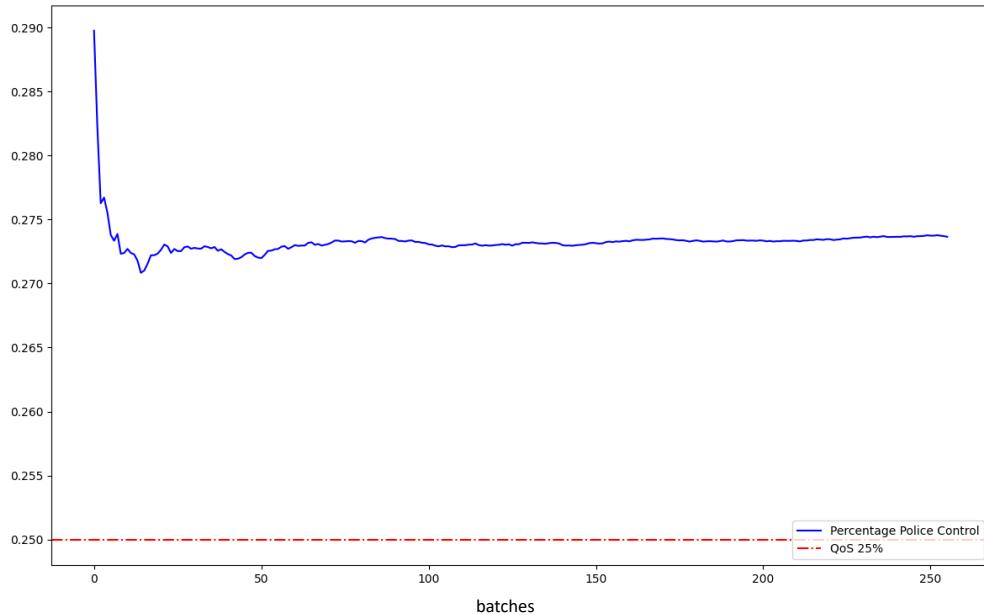
Per la prima fascia, la configurazione $\{1,5,13,1,6,9,1\}$, risulta minima per avere la stazionarietà, quindi ottima.

9.2 Fascia 14:00 – 15:00

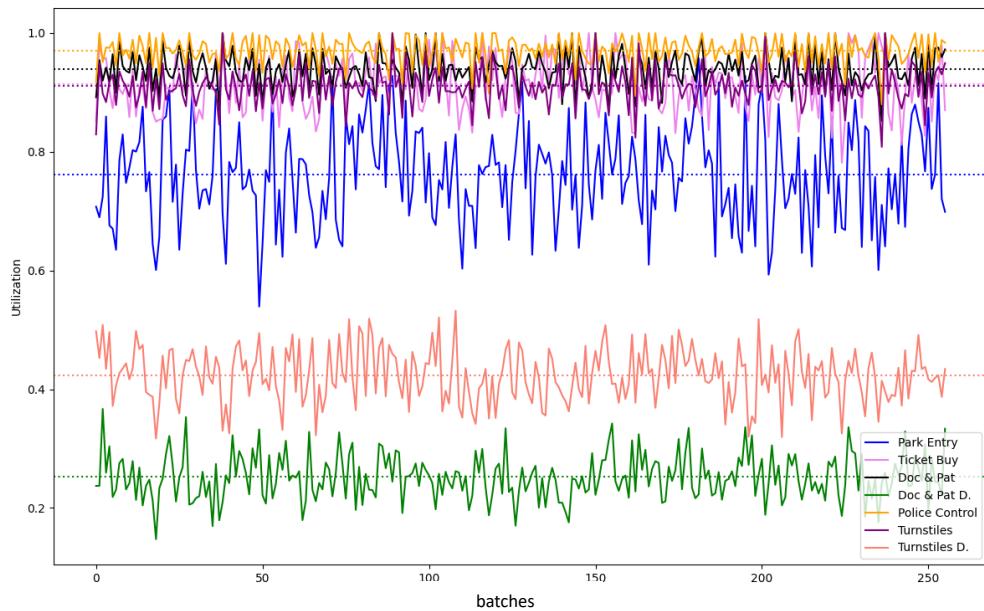
Configurazione: $\{1,10,26,1,13,19,1\}$



Nella seconda fascia oraria, con questa configurazione, il sistema è stazionario. Il tempo di risposta medio risulta essere superiore al QoS (6 minuti e mezzo). Questa configurazione permette anche di rispettare il QoS dei controlli della polizia del 25%. Anche in questo caso, il numero di serventi scelto per il centro *Police Control* risulta essere il numero minimo sufficiente per soddisfare il QoS desiderato:

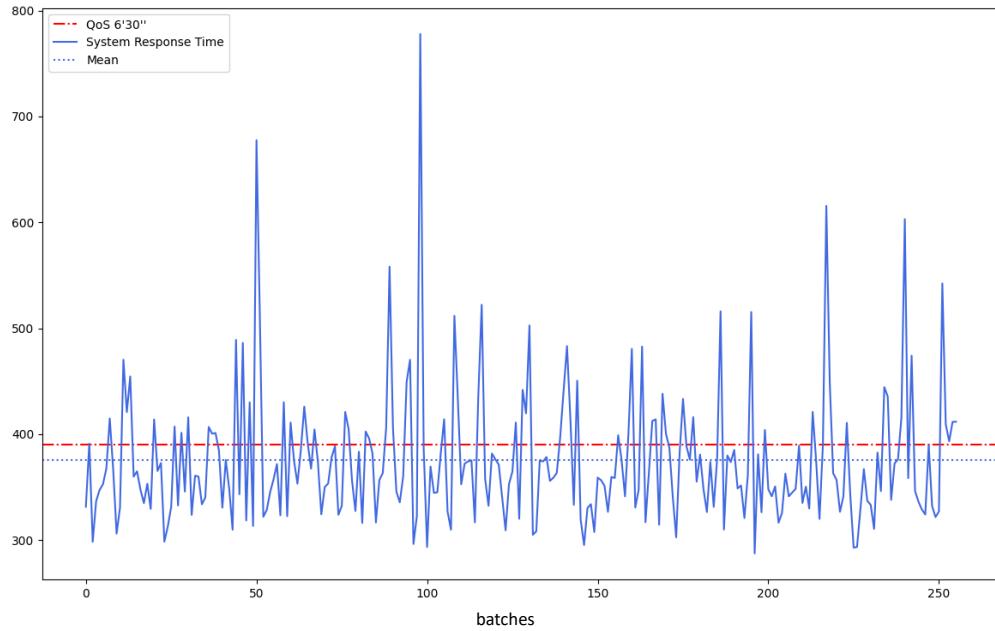


La stazionarietà della seconda fascia è esplicata anche dal grafico delle utilizzazioni:

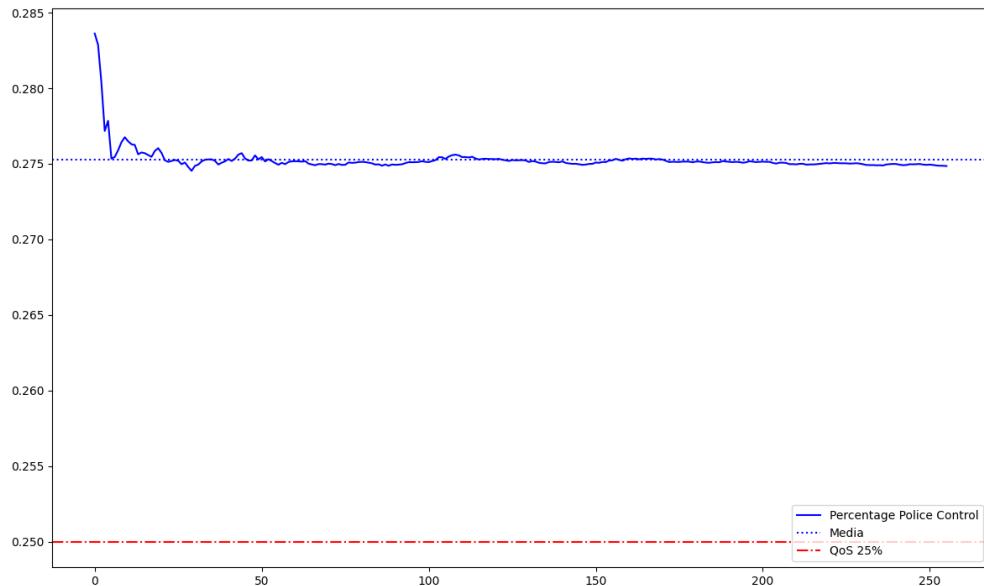


Come mostrato in figura, i centri multi-server hanno un'utilizzazione prossima allo 0.9, quindi la configurazione scelta è la minima necessaria per garantire la stazionarietà senza tuttavia rispettare il QoS del tempo di risposta.

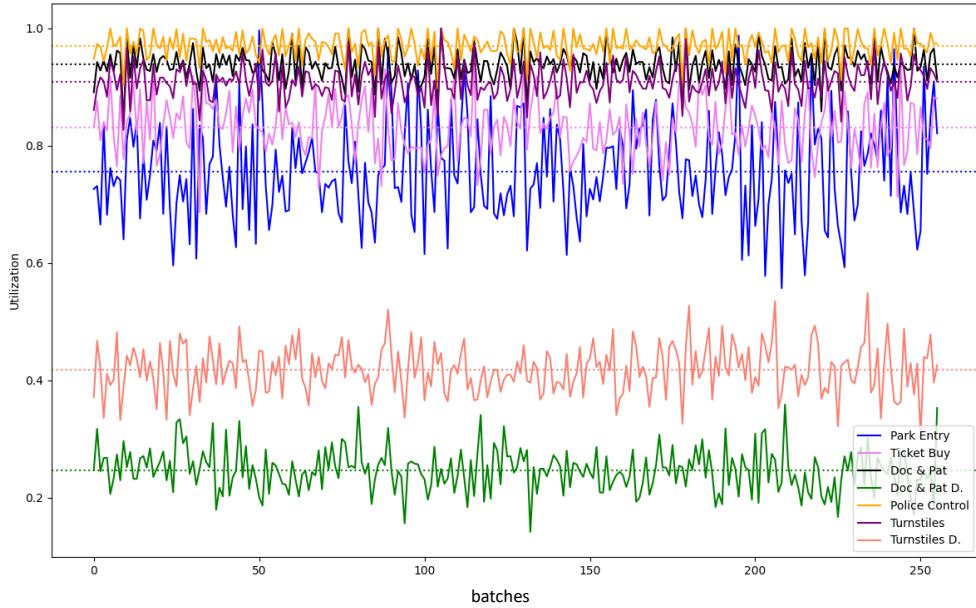
Passando alla configurazione $\{1,11,26,1,13,19,1\}$, aggiungendo un singolo servente nel centro *Ticket Buy*, si riesce a soddisfare il QoS del tempo medio di risposta (6 minuti e mezzo):



Tale configurazione continua a soddisfare il QoS dei controlli della polizia di almeno il 25%. Dai dati raccolti si evince che la percentuale di utenti controllati è del 27.5%, che è appena superiore alla soglia minima. Anche in questa situazione la configurazione è minima, in quanto con un impiego minore di poliziotti non è possibile soddisfare il QoS:



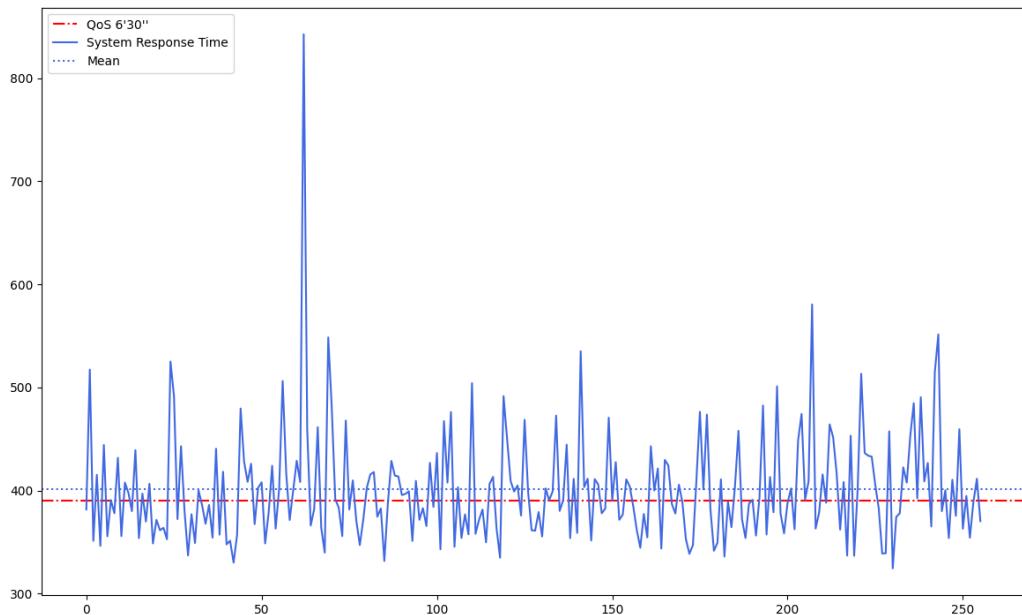
Anche per questa configurazione, il grafico delle utilizzazioni ci mostra la stazionarietà del sistema, con la differenza che in questo caso il QoS del tempo di risposta viene rispettato:



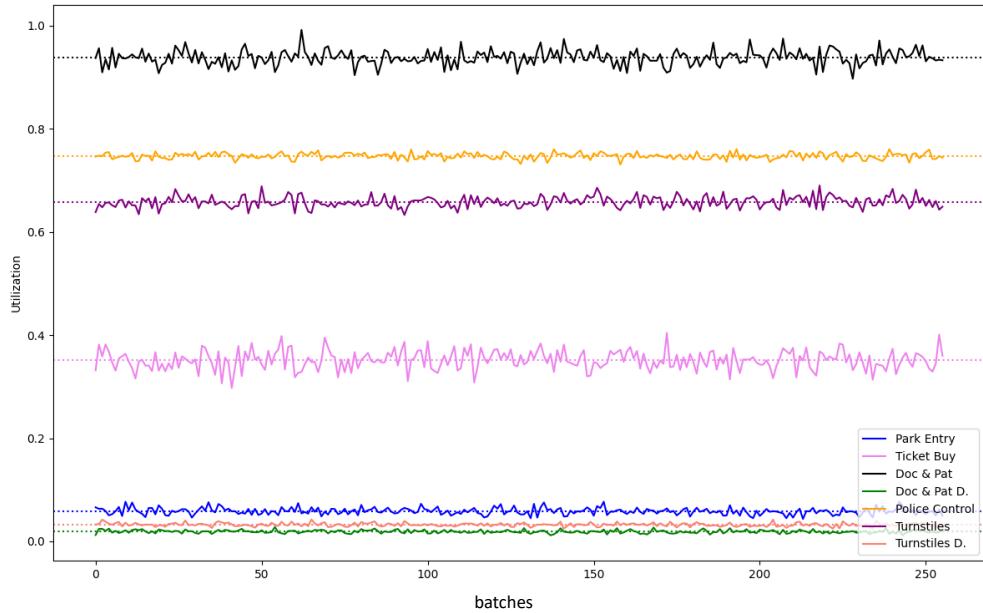
9.3 Fascia 15:00 – 16:00

Configurazione: $\{1,2,2,1,2,2,1\}$

Nella terza fascia, con la suddetta configurazione, il sistema risulta stazionario ma ha un tempo di risposta medio superiore al QoS (6 minuti e mezzo):

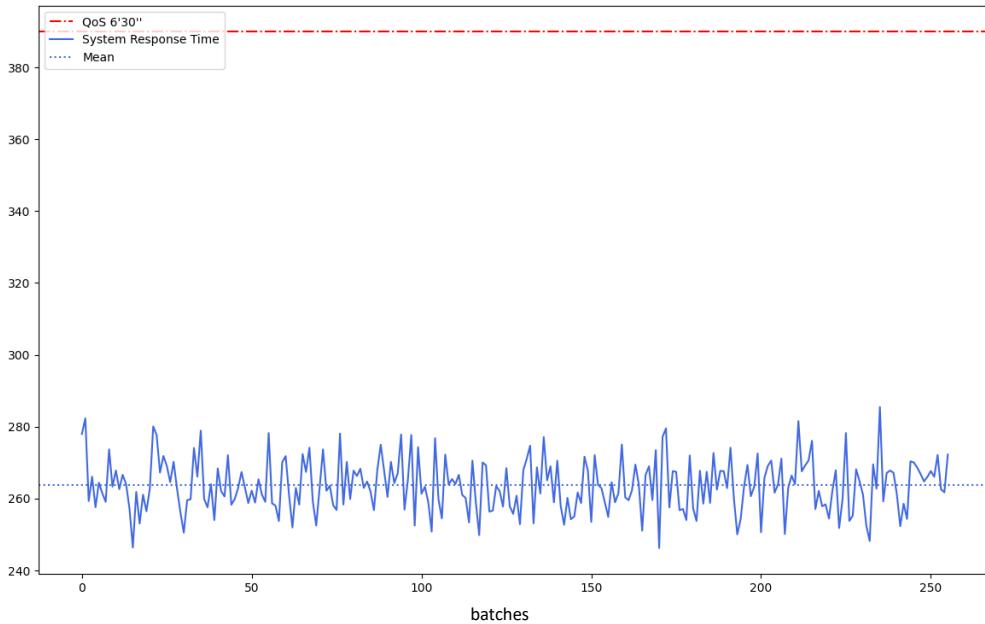


La stazionarietà di questa configurazione è mostrata anche dal seguente grafico delle utilizzazioni:

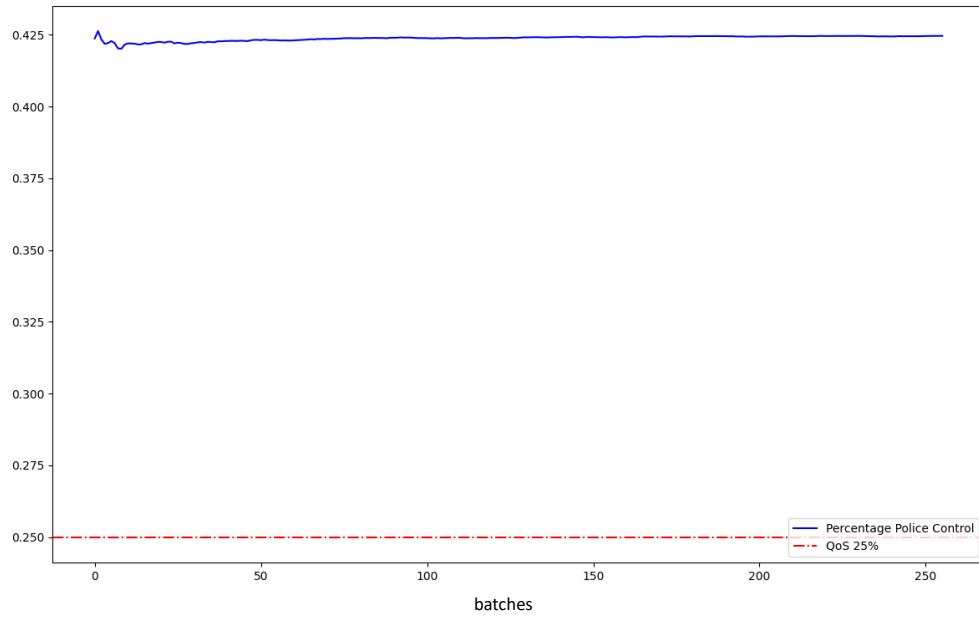


Dal grafico è possibile notare che la configurazione $\{1,2,2,1,2,2,1\}$, risulta essere minima per garantire la stazionarietà del sistema e per non variare la struttura del modello, perché togliendo un servente al centro *Doc & Pat*, il sistema diventerebbe instabile.

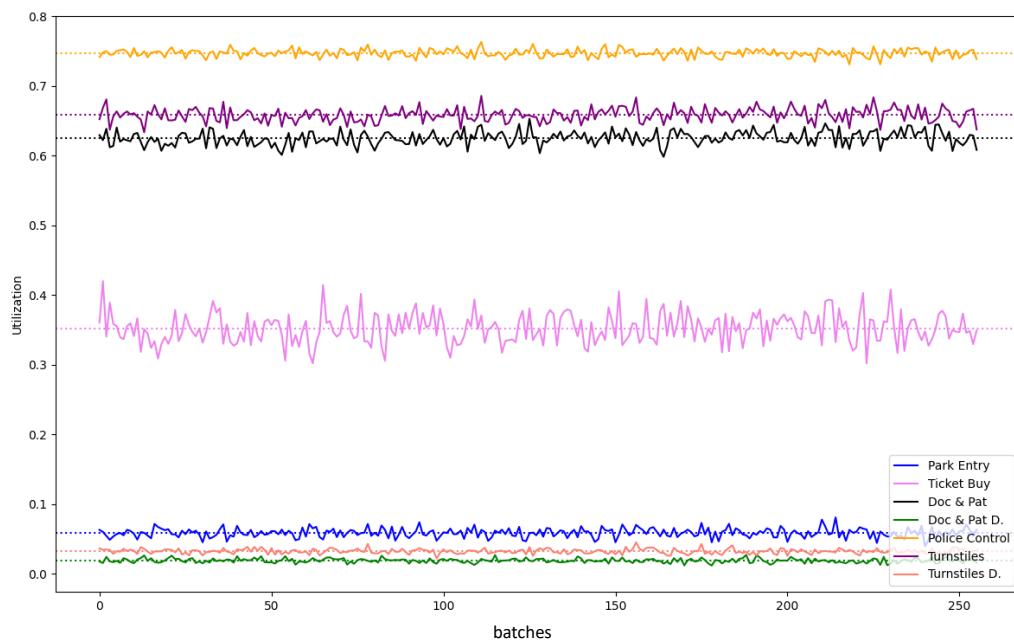
Passando alla configurazione $\{1,2,3,1,2,2,1\}$, aggiungendo un servente per il centro *Doc & Pat*, si riesce a mantenere il sistema stazionario e a garantire un tempo di risposta del sistema inferiore al QoS desiderato:



Tale configurazione soddisfa, anche, il QoS dei controlli della polizia del 25%. Si ottiene una percentuale del 42,5%, quindi anche in questo caso il numero di poliziotti impiegati è minimo per rispettare il QoS e non variare il modello:



La stazionarietà del sistema si evince anche dal grafico delle utilizzazioni successivamente riportato:



10. Finite Horizon Simulation

Con la simulazione ad orizzonte finito è stata effettuata un'analisi su tutte le tre fasce orarie, cercando di individuare una configurazione ottima dei serventi, al fine di raggiungere gli obiettivi prefissati relativamente al QoS dei controlli ed al tempo di risposta minimo del sistema.

I risultati sono stati mediati eseguendo 256 ripetizioni, in cui:

- Il sistema è vuoto ad inizio e fine simulazione;
- Ogni ripetizione fornisce statistiche rappresentanti lo studio del transiente del sistema;
- L'insieme delle statistiche rappresenta un punto del campione.

Ad ogni cambio di fascia oraria, il sistema si comporta in modo dinamico, ovvero, il numero di serventi viene aumentato/diminuito ed anche il flusso di arrivo varia come indicato nel modello delle specifiche.

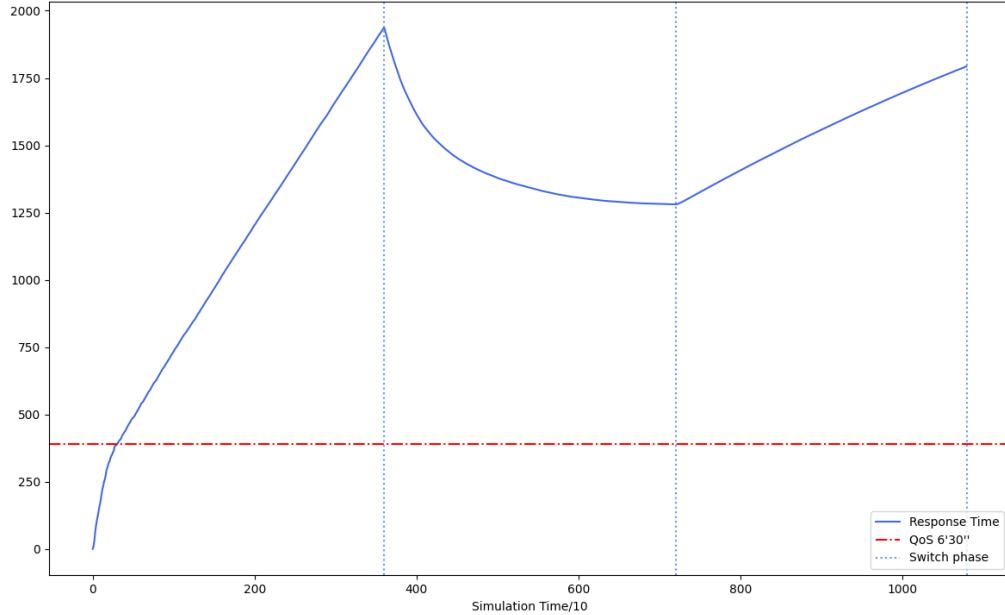
Per ogni ripetizione, sono state effettuate le misurazioni del tempo di risposta del sistema ogni dieci secondi (come somma dei tempi di risposta dei centri), per un totale di 1080 misurazioni per ripetizione.

Gli obiettivi principali della simulazione ad orizzonte finito sono:

- analizzare il comportamento del sistema al cambio fascia;
- verificare che le configurazioni individuate siano valide;
- individuare la configurazione migliore per rispettare i QoS e minimizzare i costi;
- analizzare il comportamento del sistema nel transiente.

10.1. Configurazione 1

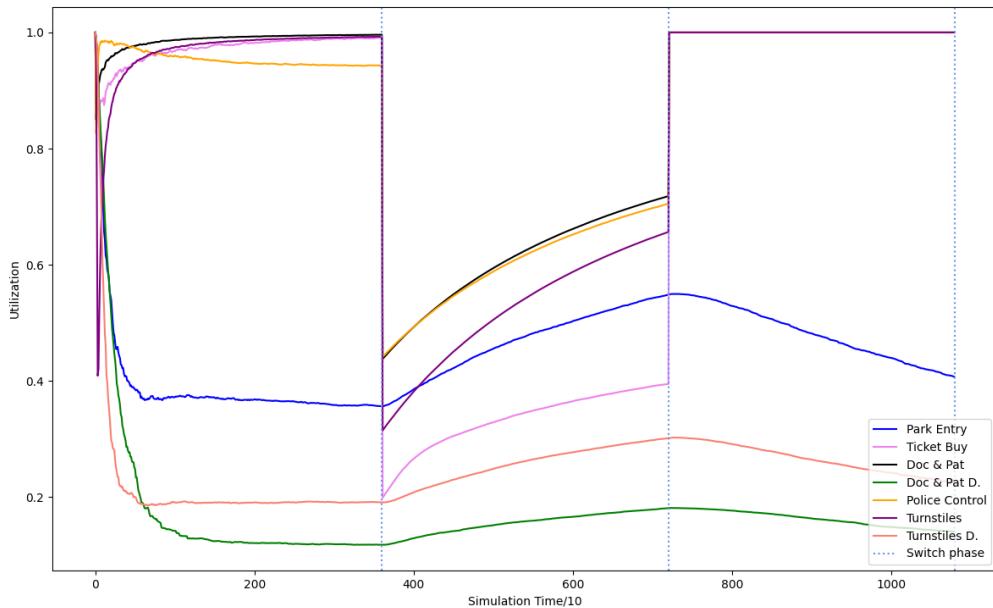
Configurazioni: $\{1,3,10,1,6,5,1\} — \{1,17,23,1,13,16,1\} — \{1,2,2,1,2,2,1\}$



Il QoS che richiede di avere tempi massimi di risposta del sistema di 6 minuti e mezzo, non viene rispettato.

Come si può osservare, la prima fascia, con tali configurazioni, non riesce a smaltire i jobs in arrivo. La seconda fascia, invece, inizia in stato di sovraccarico e migliora leggermente i tempi di risposta, grazie all'aggiunta di 45 nuovi serventi. La terza fascia, infine, inizia in uno stato di sovraccarico e non si stabilizza con tale configurazione.

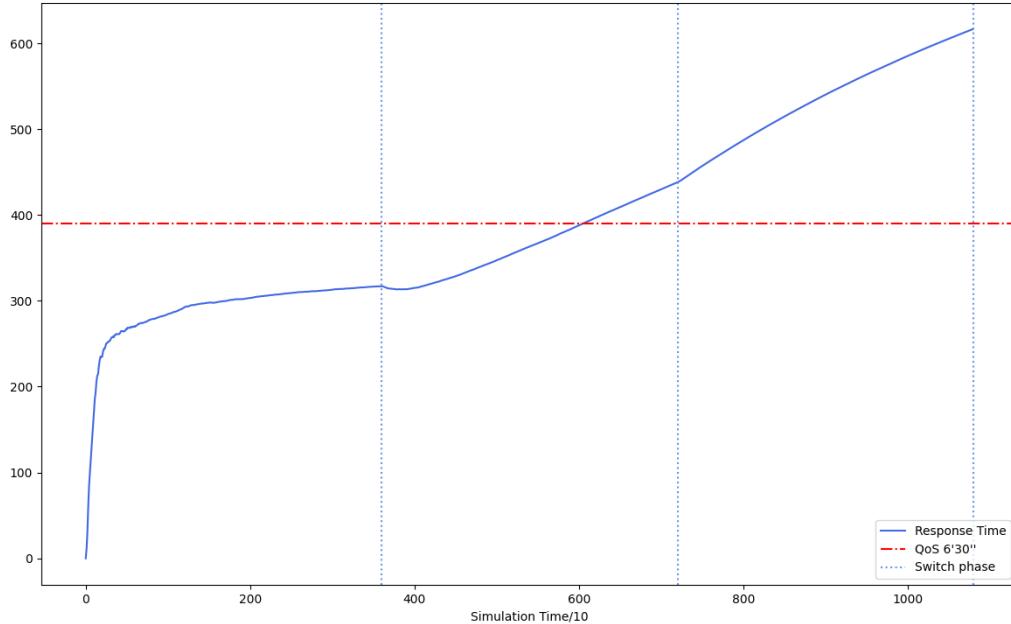
Analizzando le utilizzazioni riportate nel seguente grafico è possibile osservare come nella prima fascia i centri *Doc & Pat*, *Ticket Buy* e *Turnstiles* si trovino in uno stato di *under provisioning*, mentre, le altre fasce risultano già sovraccaricate e non valide per l'analisi.



10.2. Configurazione 2

In questo caso è stata cambiata la configurazione dei serventi per la prima fascia passando da: $\{1,3,10,1,6,5,1\} \rightarrow \{1,5,13,1,6,9,1\}$

Configurazioni: $\{1,5,13,1,6,9,1\} — \{1,17,23,1,13,16,1\} — \{1,2,2,1,2,2,1\}$



Il QoS, che richiede di avere tempi massimi di risposta del sistema di 6 minuti e mezzo, non viene rispettato.

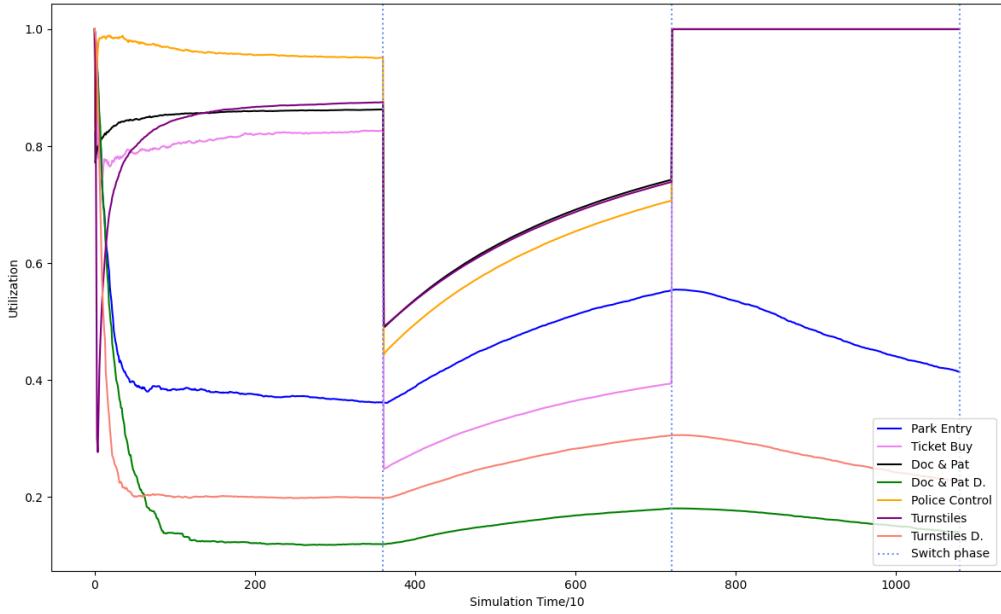
Come si può osservare, tale configurazione, permette di mantenere la prima fascia stabile.

La seconda fascia, invece, non riesce a smaltire gli utenti in servizio a causa dell'elevato numero di arrivi.

Infine, la terza fascia inizia in stato di sovraccarico e non si stabilizza a causa del basso numero di serventi.

Analizzando il grafico delle utilizzazioni è possibile notare come:

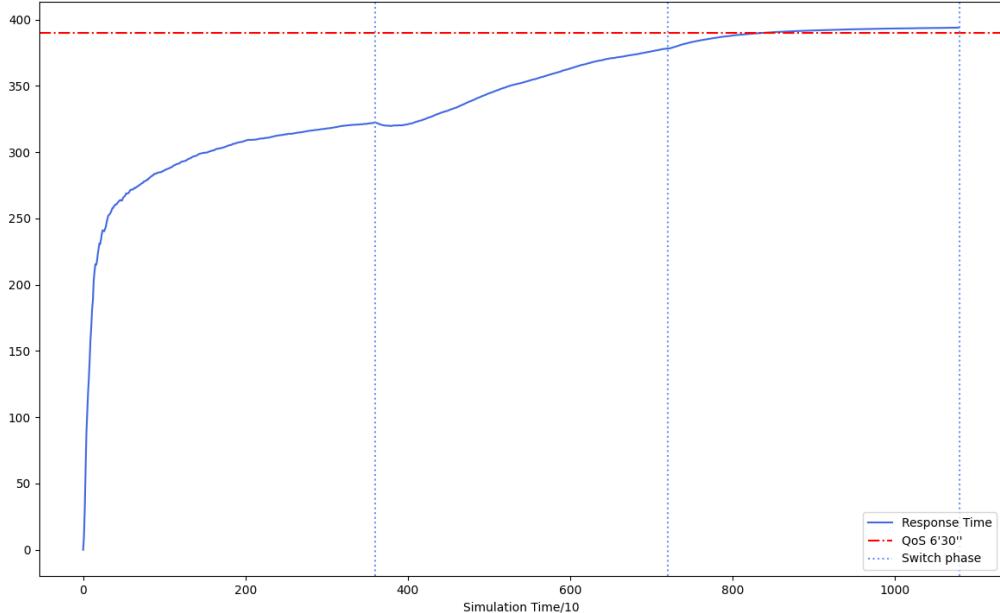
- La prima fascia sia stazionaria;
- La seconda fascia sia stazionaria ma non rispetti il QoS, dunque, si necessita un ridimensionamento del numero di serventi;
- La terza fascia risulta già sovraccarica, in quanto andando a ridurre il numero di serventi si ha che i centri dovranno smaltire un numero elevato di jobs in coda.



10.3. Configurazione 3

In questo caso è stato cambiato la configurazione dei serventi per la seconda fascia passando da: $\{1,17,23,1,13,16,1\} \rightarrow \{1,10,26,1,13,19,1\}$

Configurazioni: $\{1,5,13,1,6,9,1\} — \{1,10,26,1,13,19,1\} — \{1,2,2,1,2,2,1\}$

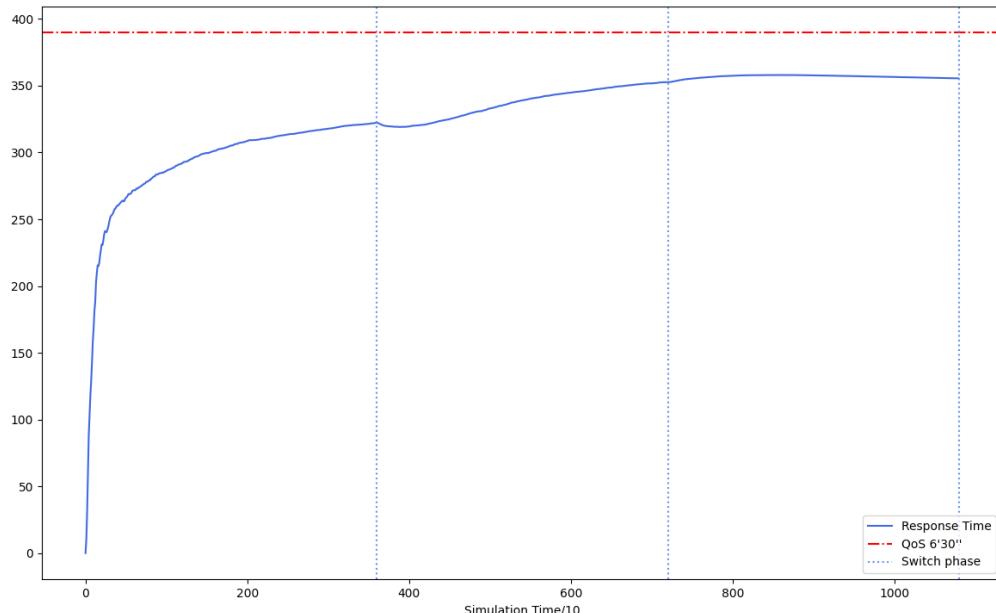


Utilizzando la suddetta configurazione che, nella simulazione ad orizzonte infinito, permetteva di rendere il sistema stazionario ma NON di rispettare il QoS del tempo di risposta medio, è possibile notare come anche nella simulazione ad orizzonte finito, tale QoS non venga rispettato. Dal grafico mostrato nella precedente immagine, è possibile notare come la seconda fascia si avvicini all'*upper bound* e la terza lo superi.

Quindi, rispetto alla configurazione ottimale per la simulazione ad orizzonte infinito, mancano due serventi al fine di rispettare il QoS del tempo di risposta. Andando a ridimensionare la seconda e terza fascia nel seguente modo:

2° fascia: $\{1, 10, 26, 1, 13, 19, 1\} \rightarrow \{1, 11, 26, 1, 13, 19, 1\}$

3° fascia: $\{1, 2, 2, 1, 2, 2, 1\} \rightarrow \{1, 2, 3, 1, 2, 2, 1\}$



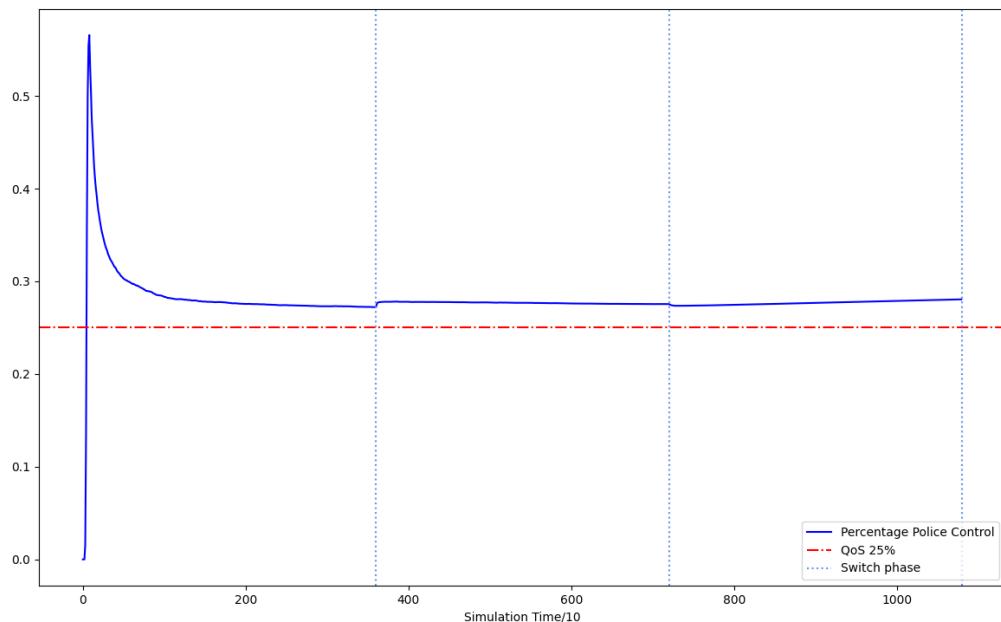
È possibile, in questa maniera, soddisfare il QoS prefissato.

Per tale configurazione, si ha un **costo totale** pari a **1731 €**.

Dopo il primo cambio di fascia oraria, c'è un decremento del tempo di risposta poiché vengono *aggiunti* 36 serventi al sistema.

Subito dopo il secondo cambio di fascia, c'è un aumento del tempo di risposta poiché vengono *rimossi* 60 serventi dal sistema.

Per quanto riguarda il QoS relativo ai controlli effettuati dalla polizia si ha che tale configurazione ci permette di soddisfarlo:



La percentuale di controllo inizialmente è molto alta, poiché i primi utenti verranno tutti controllati, essendo il centro *Police Control* ancora vuoto. Successivamente il valore si stabilizza sul suo valore medio.

11. Modello Migliorativo

Per la realizzazione del modello migliorativo si è deciso di unificare i centri:

- *Doc & Pat e Doc & Pat D.;*
- *Turnstiles e Turnstiles D.;*

andandoli a sostituire con due *multi-server con code di priorità differente senza preemption (M/M/m NP_priority)*, pur mantenendo il percorso separato per gli utenti disabili.

Tale scelta è stata fatta in quanto le coppie di centri considerate, hanno tempi di servizio uguale, ed inoltre i due centri dedicati ai disabili (*Doc & Pat D.*, *Turnstiles D.*) hanno un'utilizzazione molto bassa.

11.1. Obiettivi

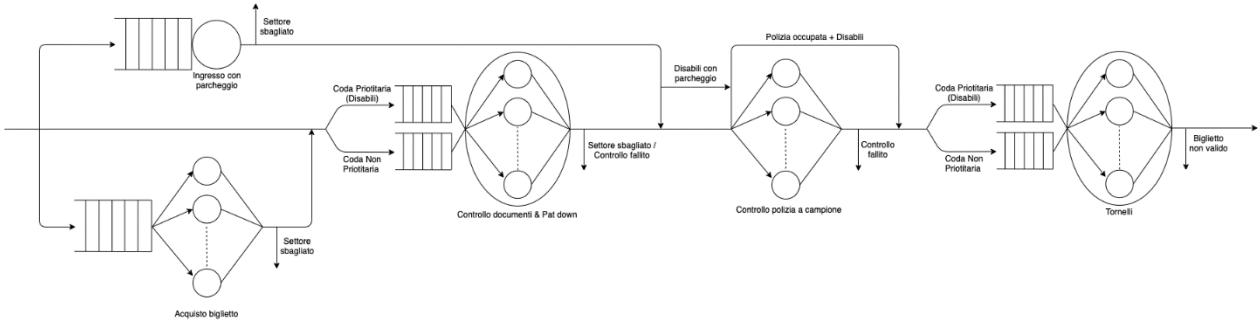
Per il modello migliorativo si hanno i medesimi obiettivi del modello base:

- Minimizzare i tempi di accesso al settore dello stadio, con un tetto massimo di accesso pari a sei minuti e mezzo (*QoS 1*);
- Minimizzare i costi relativi al personale addetto da impiegare per l'erogazione del servizio;
- Garantire una copertura dei controlli delle forze dell'ordine di almeno il 25% del flusso di utenti provenienti dal centro *Doc & Pat* (*QoS 2*).

Tuttavia in aggiunta, il modello migliorativo permette di soddisfare una delle seguenti caratteristiche:

- *mantenere lo stesso numero di serventi* del modello precedente e *migliorare il tempo di risposta* del sistema;
- *diminuire il numero di serventi* rispetto al modello precedente, mantenere gli stessi tempi di risposta ma *diminuendo i costi*.

11.2. Modello Concettuale



I centri modellati nel sistema migliorativo sono i seguenti:

- **Park Entry**: rappresenta l'accesso al parcheggio del settore dello stadio, a cui possono accedere entrambe le tipologie di utenti;
- **Ticket Buy**: modella la possibilità di acquistare il biglietto in loco il giorno della partita. In questo centro possono accedere entrambe le tipologie di utenti;
- **Doc & Pat**: rappresenta la fase di controllo effettuato dal personale addetto alla verifica dei documenti ed al *pat-down*, relativa ad entrambe le tipologie di utenti.
- **Police Control**: descrive il controllo aggiuntivo di sicurezza da parte delle forze dell'ordine a campione, effettuato solo per gli utenti non disabili;
- **Turnstiles**: modella l'accesso al settore dello stadio attraverso il passaggio ai tornelli per entrambe le tipologie di utenti.

Il centro **Park Entry** è stato modellato come una **M/M/1**. Quando c'è un arrivo in uno di questi centri, se il servente è libero si andrà in servizio, altrimenti l'utente verrà messo in coda.

I centri **Ticket Buy** è stato modellato come una **M/M/k**. Quando c'è un arrivo, si controlla se c'è un servente libero, altrimenti l'utente verrà messo in coda. Nel caso ce ne fosse più di uno, in accordo alla politica **Equity**, si sceglierà quello libero da più tempo.

I centri **Doc & Pat** e **Turnstiles** sono stati modellati come **M/M/m NP_priority** con l'uso di due code con diversa priorità, in modo da dedicare la coda con priorità maggiore agli utenti con disabilità.

Il centro **Police Control** è stato modellato come una **M/M/k/k (loss system)**, in quanto, se tutti i poliziotti sono occupati, gli utenti accederanno direttamente al centro **Turnstiles**.

11.3. Modello Computazionale

A livello implementativo è stato utilizzato un approccio di tipo *Next-Event Simulation*, in cui l'avanzamento del tempo simulato viene effettuato tramite processamento dell'evento successivo.

I file relativi al modello migliorativo (descritti precedentemente) sono:

- ***better_call_main.c***;
- ***better_call_batched_main.c***;
- ***better_config.h***;
- ***utils.h***;

L'implementazione del sistema è molto simile al modello base con l'aggiunta degli appositi campi nelle strutture dati, utili per la gestione degli utenti disabili.

Inoltre, è stato modificato la funzione che realizza l'instradamento degli utenti verso i centri, in modo da indirizzare le persone con disabilità verso la corretta coda prioritaria nei centri *Doc & Pat* e *Turnstiles*.

11.4. Verifica

Per assicurare che i risultati siano conformi alle specifiche, sono stati osservati gli stessi criteri del modello precedente:

- Il tempo di risposta deve essere uguale alla somma del tempo trascorso in coda più il tempo di servizio;
- Il numero di arrivi deve essere uguale al numero di completamenti più il numero di dropped;
- Il numero di utenti in ingresso in un centro è determinato dalla probabilità di routing indicata nel modello delle specifiche.

Il processo di verifica è stato effettuato sui dati scaturiti da una simulazione della fascia oraria centrale, 14:00 - 15:00, con $\lambda_2 = 1.273056$.

Si può osservare come il tempo di attesa (*avg delay*) per la coda prioritaria, sia molto più basso rispetto a quella non prioritaria.

11.5. Verifica Doc & Pat

```

for 4199 jobs the service node statistics:
# arrivals ..... = 4469
# dropped ..... = 270
# completions .... = 4199
                                         PRIORITÀ      NO-PRIORITÀ
processed ..... =           41           4428
avg interarrivals .. = 87.80966       0.81305
avg # in node ..... = 0.24032       29.35080
avg # in queue ..... = 0.00634       5.74908
avg utilization .... = 0.00934       0.87414
avg wait ..... = 22.10646       24.99957
avg delay ..... = 0.54189       4.89679
avg service ..... = 21.56456       20.10278

```

$$E(T_{S_p}) = E(T_{Q_p}) + E(S) = 0.54189 + 21.56456 = 22.10646$$

$$E(T_{S_{N_P}}) = E(T_{Q_{N_P}}) + E(S) = 4.89679 + 20.10278 = 24.99957$$

$$\text{Arrivi} = \text{Completamenti} + \text{Dropped} = 4199 + 270 = 4469$$

#expected_arrivals = 4436.344 (*calcolo effettuato con 'population.py'*)

11.6. Verifica Turnstiles

```

for 4229 jobs the service node statistics:
# arrivals ..... = 4242
# dropped ..... = 13
# completions .... = 4229
                                         PRIORITÀ      NO-PRIORITÀ
processed ..... =           106           4136
avg interarrivals .. = 37.34415       0.95708
avg # in node ..... = 0.37798       20.29476
avg # in queue ..... = 0.01071       4.62815
avg utilization .... = 0.01927       0.78333
avg wait ..... = 14.12728       19.44034
avg delay ..... = 0.38212       4.43330
avg service ..... = 13.74516       15.00704

```

$$E(T_{S_p}) = E(T_{Q_p}) + E(S) = 0.38212 + 13.74516 = 14.12728$$

$$E(T_{S_{N_P}}) = E(T_{Q_{N_P}}) + E(S) = 4.43330 + 15.00704 = 19.44034$$

$$\text{Arrivi} = \text{Completamenti} + \text{Dropped} = 4242 + 13 = 4229$$

#expected_arrivals = 4252.71843 (*calcolo effettuato con 'population.py'*)

11.7. Validazione

Sono stati confrontati i valori dei nuovi centri a code prioritarie con i loro relativi nel modello precedente:

Doc & Pat

Modello Migliorativo

Configurazione: {1,11,27,13,20}

```
DOC & PAT (PRIO)
# tot. arrivals ..... = 4469
# prio arrivals ..... = 41
# no-prio arrivals .. = 4428
# dropped ..... = 270
# completions ..... = 4199
avg delay prio ..... = 0.41237
avg delay no-prio ... = 4.66989
```

Modello Base

Configurazione: {1,11,26,1,13,19,1}

DOC & PAT	
# arrivals	= 4307
# dropped	= 213
# completions	= 4094
avg delay	= 8.11687

DOC & PAT D.	
# arrivals	= 29
# dropped	= 2
# completions	= 27
avg delay	= 6.16607

Turnstiles

Modello Migliorativo

Configurazione: {1,11,27,13,20}

```
TURNSTILES (PRIO)
# tot. arrivals ..... = 4242
# prio arrivals ..... = 106
# no-prio arrivals .. = 4136
# dropped ..... = 13
# completions ..... = 4229
avg delay prio ..... = 0.38723
avg delay no-prio ... = 3.30892
```

Modello Base

Configurazione: {1,11,26,1,13,19,1}

TURNSTILES	
# arrivals	= 4078
# dropped	= 7
# completions	= 4071
avg delay	= 5.26622

TURNSTILES D.	
# arrivals	= 100
# dropped	= 1
# completions ...	= 99
avg delay	= 10.59737

Sono stati confrontati i risultati delle simulazioni per i multi-server con priorità, con i valori teorici scaturiti dalle seguenti formule:

$$p(0) = \left[\sum_{i=0}^{m-1} \frac{(m\rho)^i}{i!} + \frac{(m\rho)^m}{m!(1-\rho)} \right]^{-1} \quad P_Q = \frac{(m\rho)^m}{m!(1-\rho)} p(0)$$

$$E(T_{Q_k}) = \frac{P_Q \cdot E(S)}{(1 - \sum_{i=1}^k \rho_i)(1 - \sum_{i=1}^{k-1} \rho_i)}$$

11.8. Configurazioni

Le diverse configurazioni di serventi utilizzate per la validazione, sono quelle con lo stesso numero di serventi totali del modello precedente:

	Park Entry	Ticket Buy	Doc & Pat	Police Control	Turnstiles
13:00 – 14:00	1	5	14	6	10
14:00 – 15:00	1	11	27	13	20
15:00 – 16:00	1	2	4	2	3

Per il calcolo del Risultato Sperimentale, sono state mediati i risultati di 256 simulazioni.

I valori presenti nella colonna Risultato Analitico sono stati calcolati con lo stesso numero di serventi del modello base e confrontato con i valori ottenuti dallo script ‘queue_time.py’.

11.9. Fascia 13:00 – 14:00

Statistica	Risultato Analitico	Risultato Sperimentale ($\alpha = 0.05$)
$E(T_{Q_{PARK_ENTRY}})$	16.33205	15.54437 ± 0.80040
$E(T_{Q_{TICKET_BUY}})$	75.52453	71.06907 ± 5.97436
$E(T_{Q_{DOC_PAT}})_{PRIO}$	0.52382	0.52261 ± 0.02395
$E(T_{Q_{DOC_PAT}})_{NO_PRIO}$	2.79264	2.64942 ± 0.20092
$E(T_{Q_{POLICE_CONTROL}})$	0.00000	0.00000 ± 0.00000
$E(T_{Q_{TURNSTILES}})_{PRIO}$	0.68044	0.69890 ± 0.01927
$E(T_{Q_{TURNSTILES}})_{NO_PRIO}$	3.73337	3.71195 ± 0.12083

11.10. Fascia 14:00 – 15:00

Statistica	Risultato Analitico	Risultato Sperimentale ($\alpha = 0.05$)
$E(T_{Q_{PARK_ENTRY}})$	97.02908	97.06647 ± 0.96097
$E(T_{Q_{TICKET_BUY}})$	77.57197	78.53709 ± 1.14106
$E(T_{Q_{DOC_PAT}})_{PRIO}$	0.40775	0.41379 ± 0.01132
$E(T_{Q_{DOC_PAT}})_{NO_PRIO}$	4.67751	4.54719 ± 0.13671
$E(T_{Q_{POLICE_CONTROL}})$	0.00000	0.00000 ± 0.00000
$E(T_{Q_{TURNSTILES}})_{PRIO}$	0.3787	0.39351 ± 0.02111
$E(T_{Q_{TURNSTILES}})_{NO_PRIO}$	3.32145	3.35981 ± 0.03857

11.11. Fascia 15:00 – 16:00

Statistica	Risultato Analitico	Risultato Sperimentale ($\alpha = 0.05$)
$E(T_{Q_{PARK_ENTRY}})$	1.86973	1.85882 ± 0.04519
$E(T_{Q_{TICKET_BUY}})$	12.7291	12.74679 ± 0.17830
$E(T_{Q_{DOC_PAT}})_{PRIO}$	0.74704	0.82044 ± 0.07356
$E(T_{Q_{DOC_PAT}})_{NO_PRIO}$	1.41822	1.32311 ± 0.09802
$E(T_{Q_{POLICE_CONTROL}})$	0.00000	0.00000 ± 0.00000
$E(T_{Q_{TURNSTILES}})_{PRIO}$	0.95233	0.95908 ± 0.22157
$E(T_{Q_{TURNSTILES}})_{NO_PRIO}$	1.74313	1.68677 ± 0.07972

11.12. Confronto Configurazioni

Le configurazioni prese in esame si differenziano per:

- **Stesso numero di serventi totali del modello precedente**, costo 1731€:

	Park Entry	Ticket Buy	Doc & Pat	Police Control	Turnstiles
1° Fascia	1	5	14	6	10
2° Fascia	1	11	27	13	20
3° Fascia	1	2	4	2	3

- **Minor numero di serventi rispetto al modello precedente**, costo 1631€:

	Park Entry	Ticket Buy	Doc & Pat	Police Control	Turnstiles
1° Fascia	1	5	12	6	9
2° Fascia	1	11	26	13	18
3° Fascia	1	2	3	2	2

11.13. Infinite Horizon Simulation

Come nel modello precedente, la simulazione ad orizzonte infinito è stata effettuata su un lasso di tempo molto più lungo rispetto al tempo reale, andando a prendere in considerazione una singola fascia per volta.

Si è considerato, quindi, un sistema statico, con un numero di serventi e flusso di arrivo che rimangono invariati al cambio di fascia oraria.

Per ogni fascia, si è cercato di raggiungere i seguenti obiettivi:

- Analizzare il comportamento del sistema allo stato stazionario;
- Individuare la configurazione migliore per rispettare i QoS e minimizzare i costi.

Per far ciò è stato utilizzato il metodo delle **Batch Means**. È stata considerata un'esecuzione suddivisa in $k = 256$ batches di dimensione $b = 8192$. Tali valori sono stati scelti cercando il valore di b per cui *l'autocorrelazione* del campione fosse minore di 0,2 per un lag $j = 1$. A questo punto, sono state calcolate le statistiche per ogni batch per un'ulteriore analisi attraverso i successivi grafici.

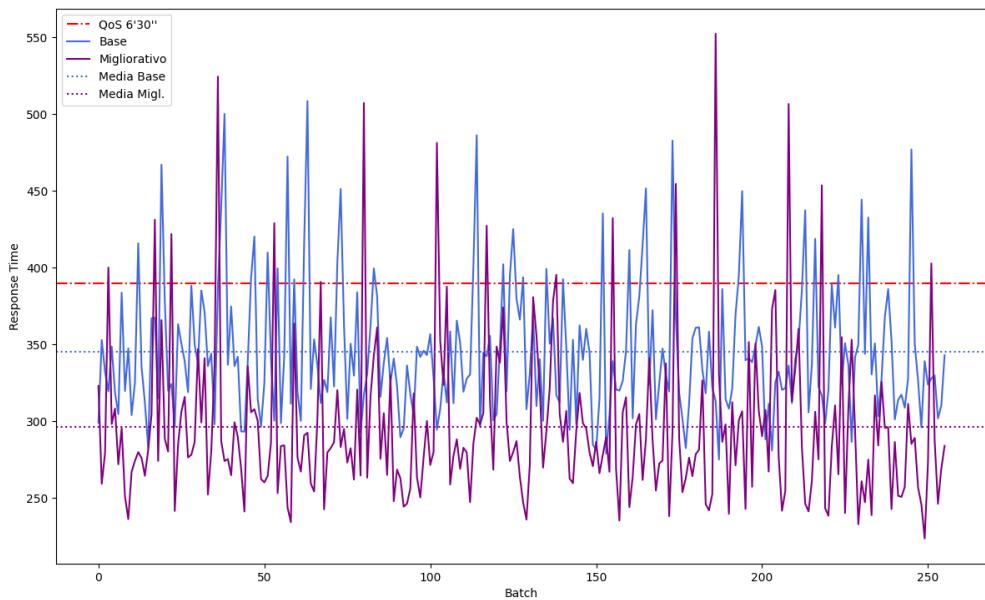
Nel prossimo paragrafo l'uso dei colori fa riferimento al tipo di configurazione specificata in '*Modello Migliorativo - Confronto Configurazioni*'.

11.14. Infinite Horizon Simulation – Fascia 13:00 – 14:00

Configurazione: {1,5,14,6,10}

Nella prima fascia oraria, utilizzando lo stesso numero di serventi totali della configurazione ottima per il modello base, si osserva che il sistema è stazionario ed inoltre ha un tempo di risposta inferiore ai 300 secondi, e che quindi rispetta il QoS.

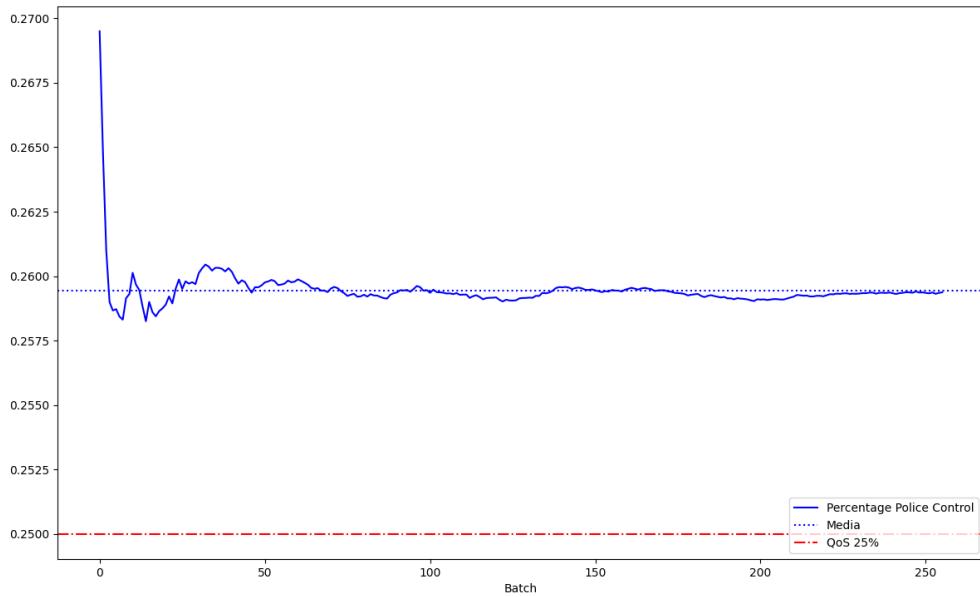
Inoltre, avendo introdotto code prioritarie, è possibile vedere come la media del tempo di risposta del modello migliorativo, risulti essere inferiore alla media del tempo di risposta del modello base di circa 50 secondi:



Con questa configurazione, viene soddisfatto anche il QoS relativo ai controlli della polizia.

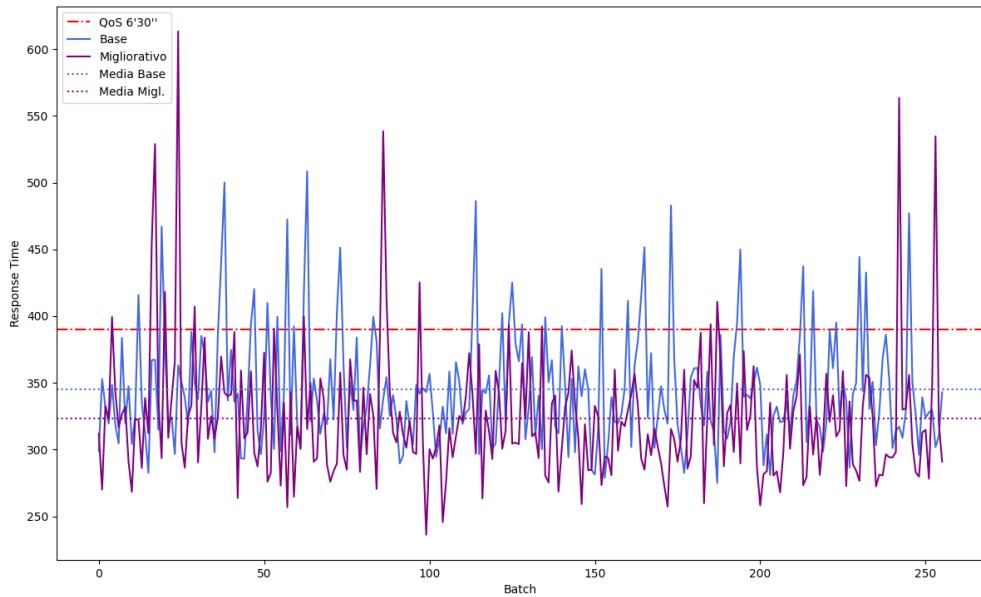
Si ottiene infatti, una percentuale di circa il 26% di controlli.

Il numero di poliziotti, scelti in questa configurazione, risulta essere minimo poiché diminuendolo, si scenderebbe sotto il livello del QoS:



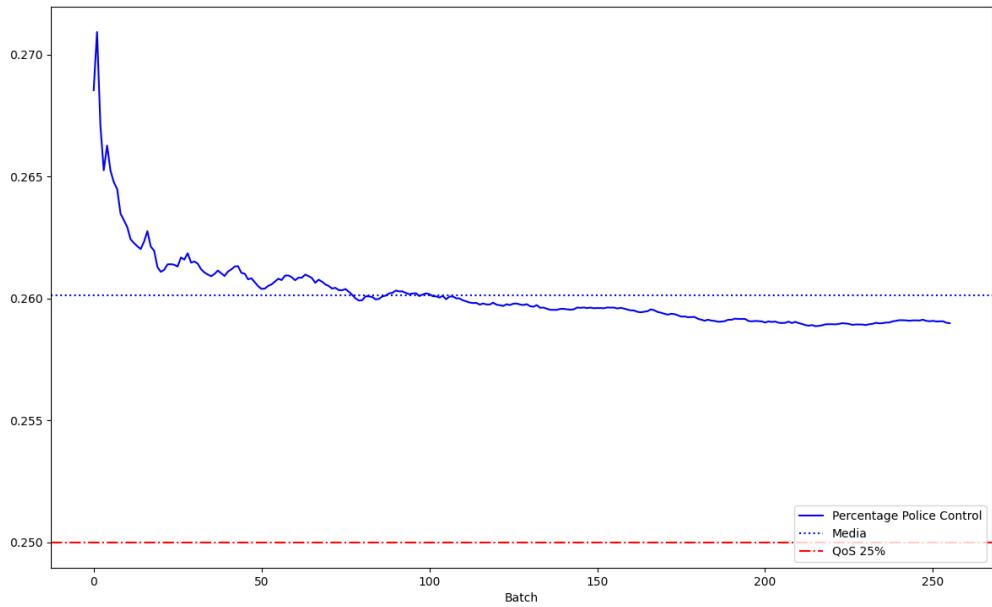
Configurazione: {1,5,12,6,9}

Andando a ridurre di tre il numero di serventi totali utilizzati, è possibile notare che il sistema è ancora stazionario e che il tempo di risposta medio rispetta il QoS (6 minuti e mezzo) ed è migliorato rispetto al modello base di circa 25 secondi:



Con questa configurazione viene rispettato anche il QoS dei controlli della polizia, ottenendo una percentuale media di controlli del 26%.

Anche in questo caso, quindi, il numero di poliziotti scelti risulta essere il minimo sufficiente per garantire il QoS desiderato:

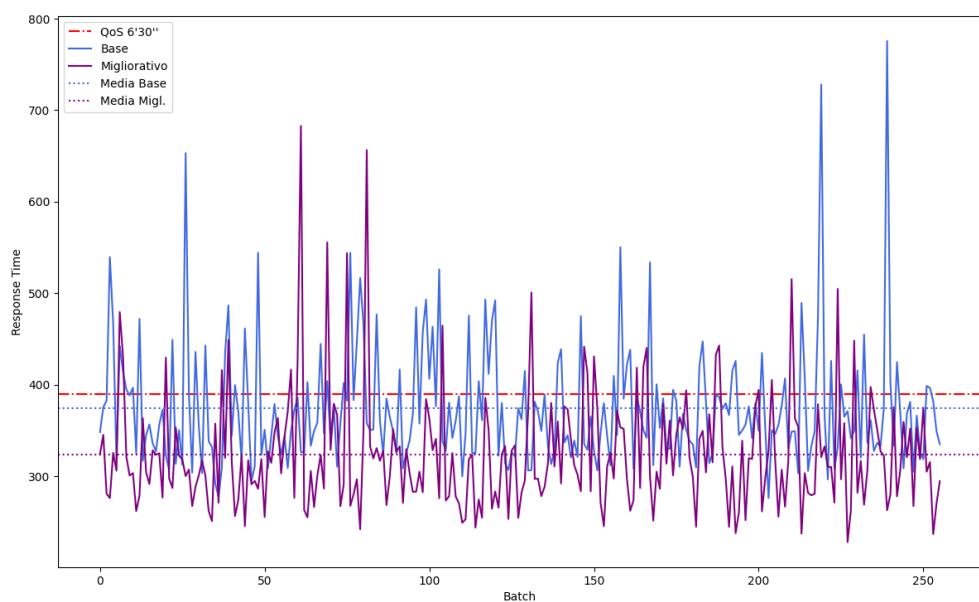


Utilizzando, quindi, quest'ultima configurazione $\{1,5,12,6,9\}$, si riesce a raggiungere gli obiettivi di servizio preposti e a ridurre i costi da sostenere.

11.15. Infinite Horizon Simulation – Fascia 14:00 – 15:00

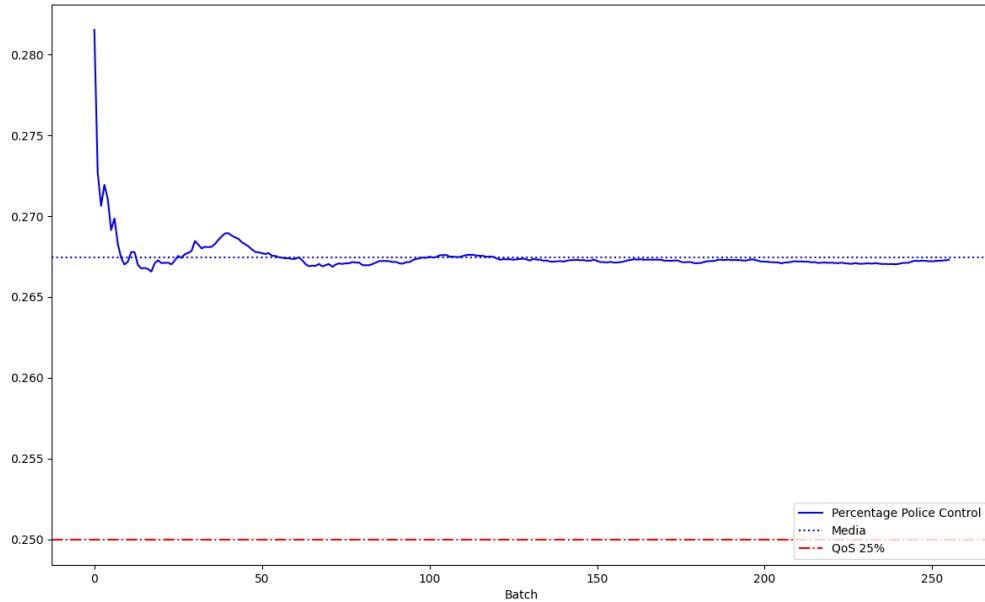
Configurazione: $\{1,11,27,13,20\}$

Per la seconda fascia oraria, con lo stesso numero complessivo di serventi utilizzati nel modello base, si ottiene che il sistema è ancora stazionario ed il tempo medio di risposta è migliorato rispetto al modello base di circa 50 secondi:



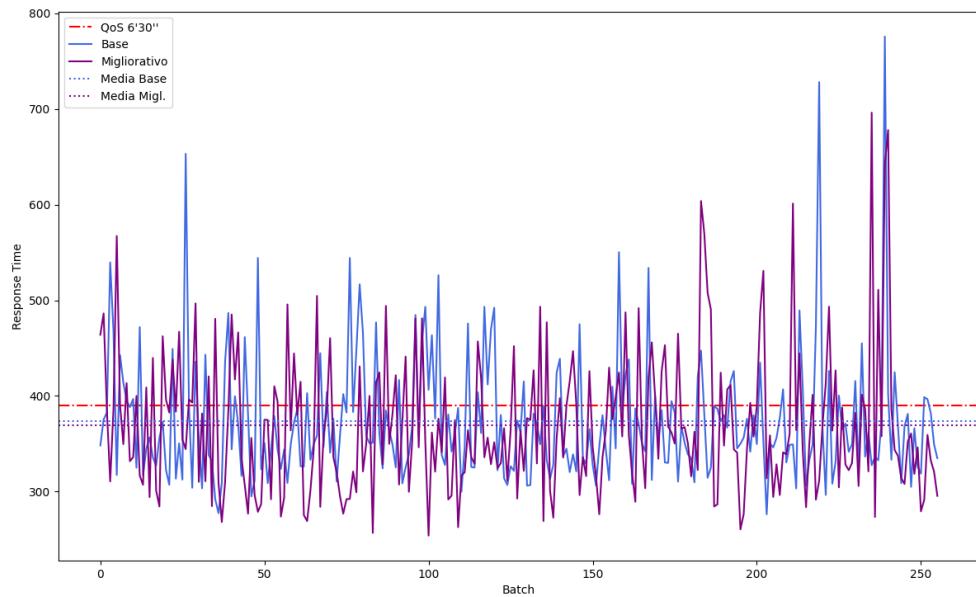
Anche il QoS dei controlli della polizia viene rispettato, raggiungendo una percentuale di controlli effettuati compresa tra il 26.5% e il 27%.

Dunque il numero di poliziotti scelto risulta essere il minimo necessario a garantire il QoS (25%):



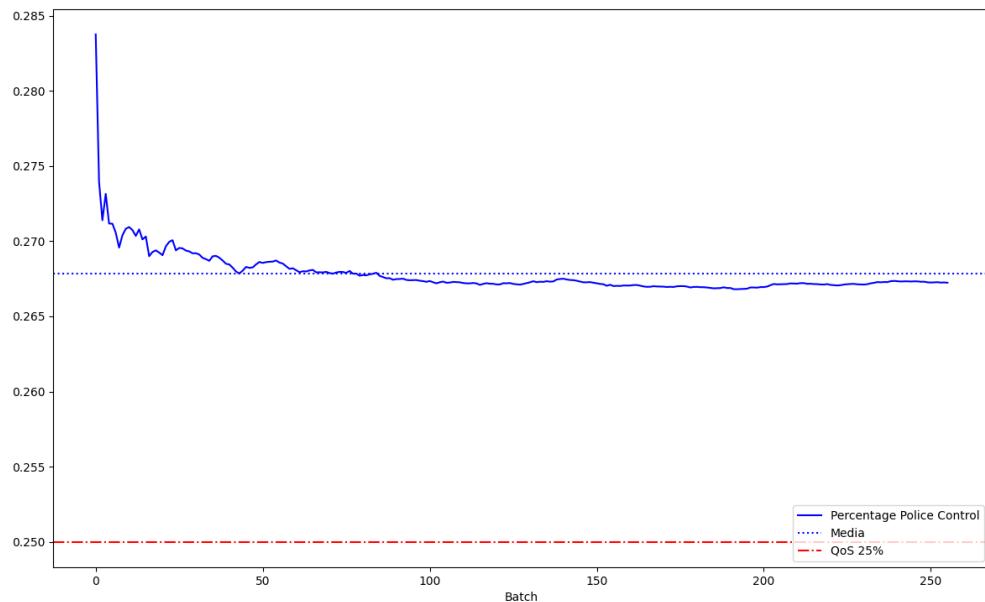
Configurazione: {1,11,26,13,18}

Andando a rimuovere tre serventi dal sistema, si ottiene che il sistema è ancora stazionario, che il tempo di risposta medio del sistema rispetta il QoS (6 minuti e mezzo) ed è più basso del tempo di risposta medio del modello base:



Con questa configurazione viene rispetta anche il QoS dei controlli della polizia in quanto si ottiene una percentuale di controlli compresa tra il 26.5% e

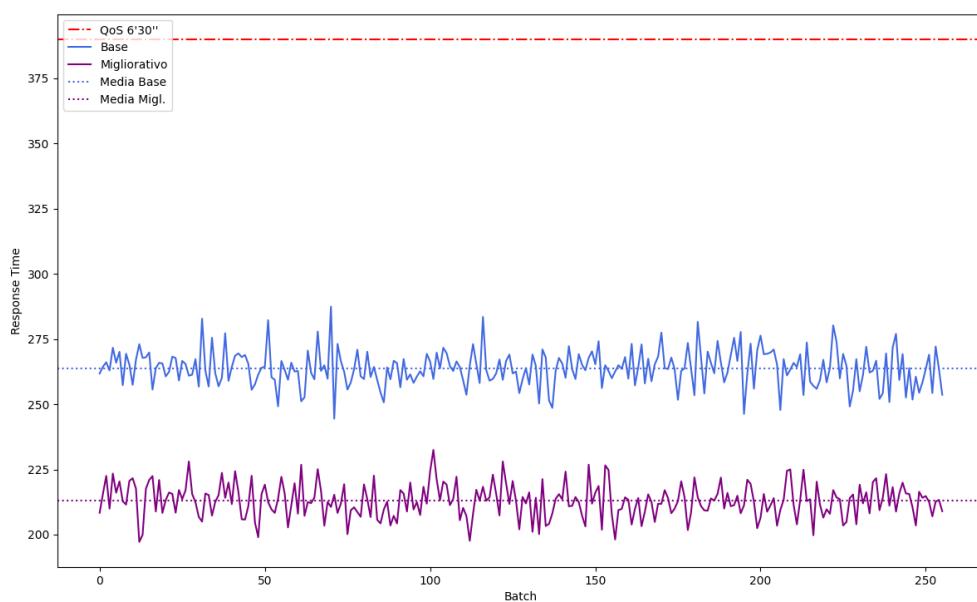
il 27%. Quindi il numero di poliziotti, essendo lo stesso della configurazione precedente, risulta essere minimo per soddisfare il QoS (25%):



11.16. Infinite Horizon Simulation – Fascia 15:00 – 16:00

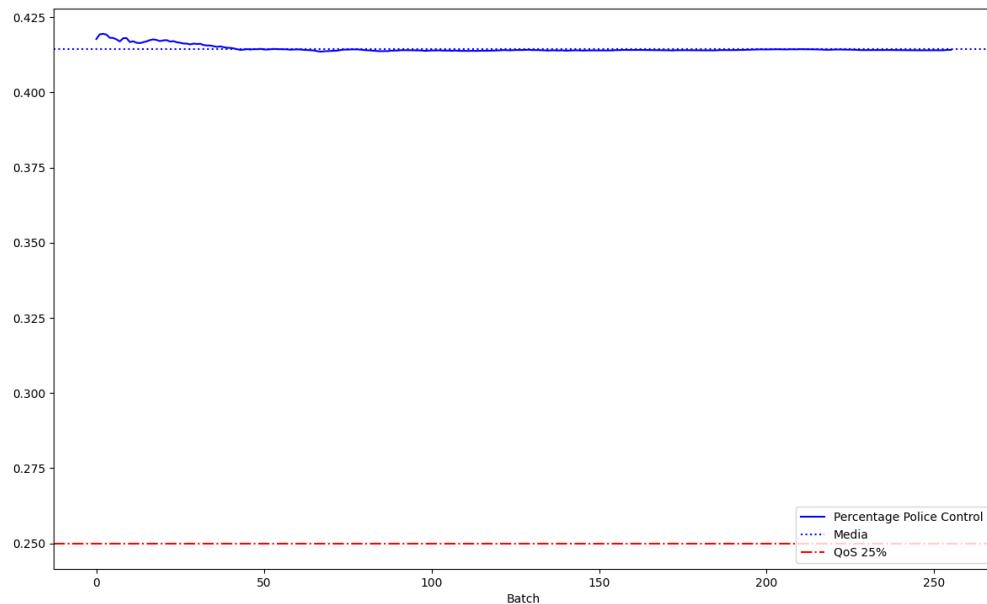
Configurazione: {1,2,4,2,3}

Nella terza fascia oraria, con lo stesso numero complessivo di serventi utilizzati nel modello base, si ottiene che il sistema è ancora stazionario ed il tempo di risposta medio del sistema è migliorato rispetto al modello base di circa 50 secondi:



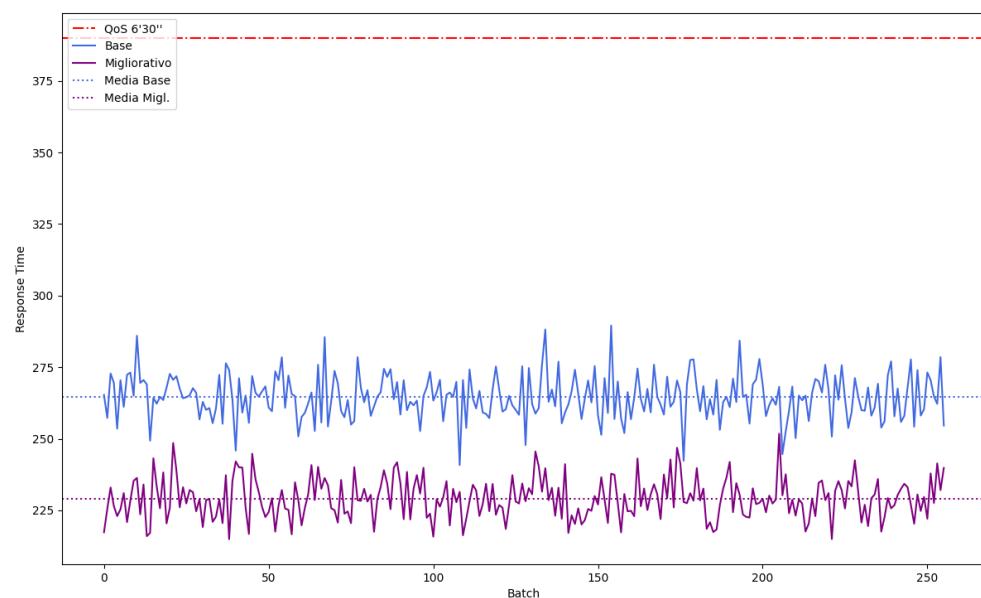
Con questa configurazione, anche il QoS dei controlli della polizia viene rispettato, in quanto, si ottiene una percentuale di controlli effettuati compresa tra il 40% e il 42.5%.

Anche in questo caso il numero di poliziotti impiegati risulta il minimo per rispettare il QoS (25%) e per non variare il modello:



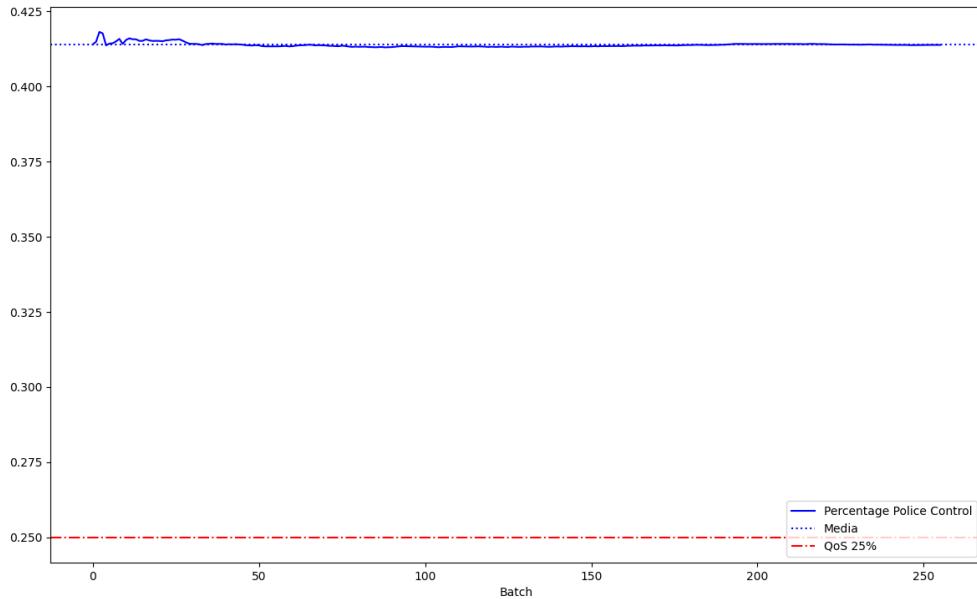
Configurazione: {1,2,3,2,2}

Andando a rimuovere due serventi dal numero totale rispetto al modello base, si ottiene ancora che il sistema è stazionario, che il tempo di risposta medio continua a rispettare il QoS e ad essere inferiore del tempo di risposta del modello base:



Con questa configurazione viene rispettato anche il QoS dei controlli effettuati dalla polizia, in quanto si ottiene una percentuale ancora compresa tra il 40% e il 42.5%.

Non essendo cambiato il numero di poliziotti impiegati, questo risulta ancora minimo per garantire il QoS desiderato (25%) e per non variare il modello:



11.17. Finite Horizon Simulation

Come nel modello precedente, la simulazione ad orizzonte finito è stata effettuata sulle 3 fasce orarie, e i risultati sono stati mediati su 256 ripetizioni in cui:

- Il sistema è vuoto ad inizio e fine simulazione;
- Ogni ripetizione fornisce statistiche rappresentanti lo studio del transiente del sistema;
- L'insieme delle statistiche rappresenta un punto del campione.

Il sistema viene assunto dinamico al cambio di fascia oraria, ovvero variano sia il numero di serventi attivi per ogni centro che il flusso degli arrivi nel sistema.

Per ogni ripetizione, sono state effettuate le misurazioni del tempo di risposta del sistema *ogni 10 secondi* (come somma dei tempi di risposta dei centri), per un totale di *1080 misurazioni per ripetizione*.

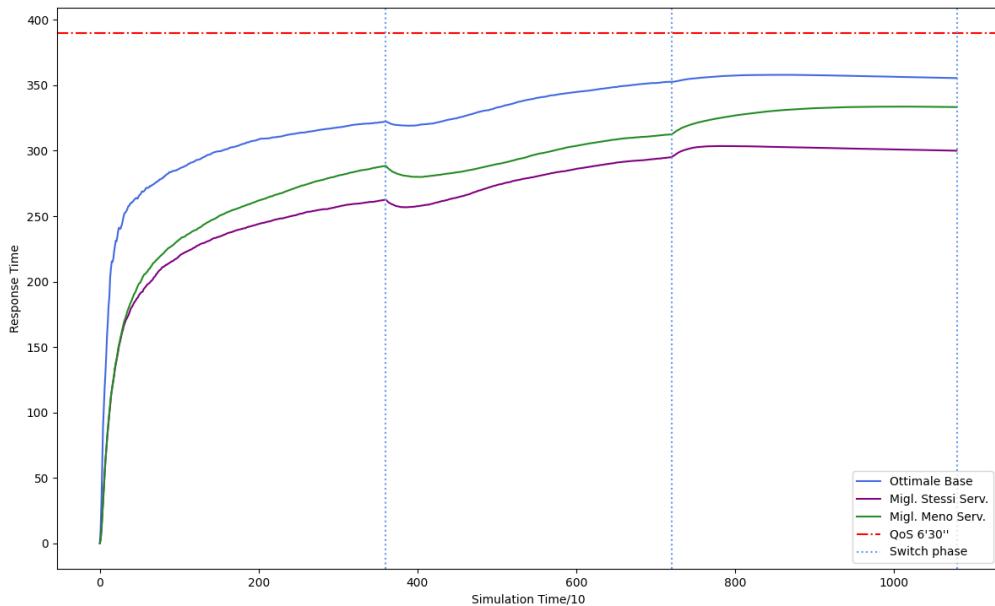
Gli obiettivi principali della simulazione ad orizzonte finito sono:

- analizzare il comportamento del sistema al cambio fascia;
- verificare che le configurazioni individuate siano valide;
- individuare la configurazione migliore per rispettare i QoS e minimizzare i costi;
- analizzare il comportamento del sistema nel transiente.

Configurazione 1: $\{1,5,14,6,10\} — \{1,11,27,13,20\} — \{1,2,4,2,3\}$

Configurazione 2: $\{1,5,12,6,9\} — \{1,11,26,13,18\} — \{1,2,3,2,2\}$

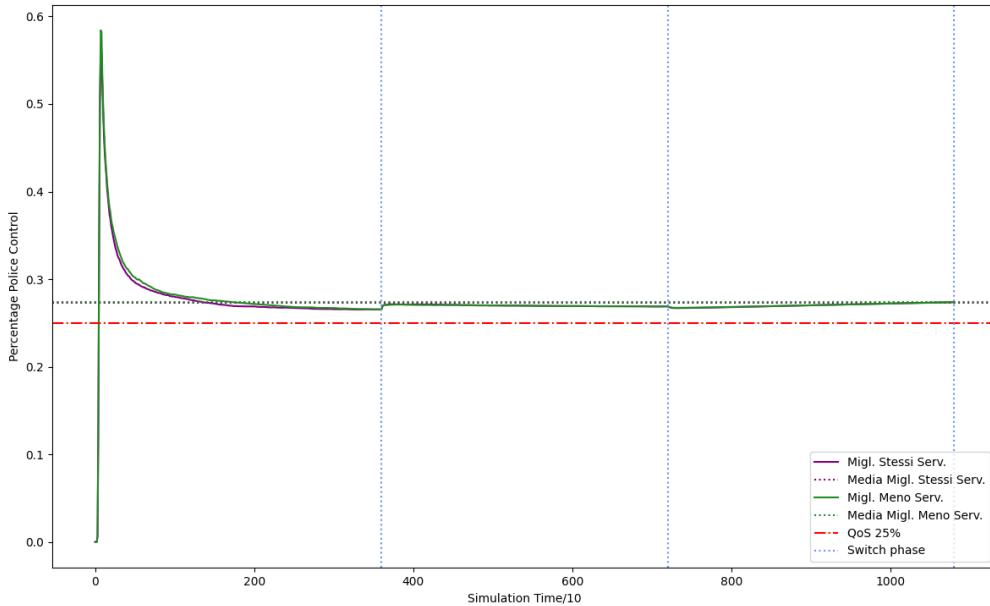
Entrambe le configurazioni utilizzate nel modello migliorativo rispettano il QoS del tempo massimo di risposta del sistema:



In particolare, il *modello migliorativo con stesso numero di serventi* rispetto al precedente (*Base*), è quello con tempo di risposta totale minore, mentre, il *modello migliorativo con minor numero di serventi* rispetto al precedente (*Base*) è quello con tempo di risposta totale compreso tra gli altri due e con un risparmio di 100€.

Quest'ultimo quindi, risulta essere un *trade-off* tra prestazioni e prezzo.

Entrambe le configurazioni rispettano anche il QoS dei controlli della polizia che è di almeno il 25%:



Risulta interessante notare che in prima fascia, la percentuale di controlli da parte della polizia è maggiore nella configurazione con meno serventi.

Come già visto in fase di validazione nel modello base, diminuendo i serventi in ‘Doc & Pat’ la percentuale di controllo aumenta.

12. Conclusioni

In entrambi i modelli presentati vengono rispettati i QoS prestabiliti.

Il modello base corrisponde al caso reale in cui i disabili hanno percorsi completamente separati.

Nel modello migliorativo, tutti gli utenti attraversano gli stessi centri ma i disabili hanno code prioritarie e percorsi dedicati.

Il modello migliorativo proposto con lo stesso numero di serventi, e quindi con le stesse spese, permette di avere una notevole riduzione dei tempi di risposta.

Il modello migliorativo proposto con minor numero di serventi permette di avere anch’esso una riduzione dei tempi di risposta rispetto al modello base ma prevede anche un risparmio di 100€.

13. Implementazione

Nel sistema, inizialmente vuoto, cominciano a verificarsi arrivi dall'esterno che vengono instradati dalla funzione *Initial_routing_func*, che si occupa di gestirne il routing.

Nel momento in cui si verifica il prossimo evento nel sistema, trovato con la *NextEventNodes*, se ne controlla la tipologia (arrivo o completamento), andando ad invocare le funzioni *NewArrivalManage* o *NewCompletionManage*.

La prima si occupa di gestire un nuovo arrivo in un centro. Se un job non trova coda, e almeno un servente è libero, viene mandato in servizio invocando la funzione *FindOne*, che applicherà la politica *Equity* per trovare il servente libero (*IDLE*) da più tempo. Successivamente vengono aggiornate le statistiche del centro. In caso non sia possibile andare subito in servizio, il job viene messo in coda.

La funzione *NewCompletionManage*, invece, gestisce i completamenti di un nodo, andando ad instradarli verso i corretti centri successivi, tramite apposite funzioni di routing. La funzione *EntryNextNode*, per gestire il corretto passaggio di un job da un nodo ad un altro, usa gli stessi meccanismi della *NewArrivalManage*. In questo modo il job appena processato, risulterà essere un nuovo arrivo per l'altro centro. Una volta instradato, se si ha coda, si processerà un nuovo job, altrimenti il servente verrà messo *IDLE*.

Il sistema prevede tre fasce orarie con diversi flussi di arrivo e diverso dimensionamento. Nel momento in cui avviene un cambio di fascia, tramite la funzione *Switch_phase* viene gestita l'allocazione/deallocazione dei serventi.

Nei centri in cui avviene l'allocazione, in base all'eventuale coda presente, i nuovi serventi, oltre ad essere attivati, vengono resi disponibili per prendere subito un job in lavorazione.

Nei centri in cui viene diminuito il numero di serventi, se qualcuno di essi non ha ancora terminato l'elaborazione del suo job, viene disattivato ma rimane comunque in stato *BUSY*. Verrà quindi trovato dalla *NextEventNodes* e potrà essere gestito il relativo completamento.

Il sistema, una volta raggiunto il tempo di *STOP* (per le simulazioni ad orizzonte finito) oppure l'arrivo di $k \cdot b$ jobs (per le simulazioni ad orizzonte infinito), non processerà più nuovi arrivi esterni e si occuperà di ultimare i jobs restanti nel sistema.