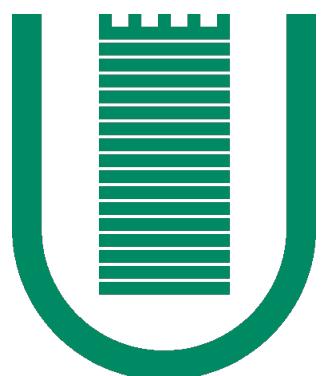


# *Performance Modeling of Computer Systems and Networks*

*Università degli studi di Roma “Tor Vergata”  
A.A. 2018/2019*



*Prof. V. de Nitto Persone  
Studentessa: Ricci Francesca*

*Relazione del Progetto*

# *Indice*

## *Algoritmo 1*

<i>Goal e Obiettivi</i> .....	3
<i>Modello Concettuale</i> .....	3
<i>Modello di Specifica</i> .....	5
<i>Modello Computazionale</i> .....	10
<i>Verifica e Validazione</i> .....	12
<i>Intervalli di Confidenza</i> .....	13
<i>Analisi Transiente</i> .....	14
<i>Statistiche Ottenute</i> .....	17
<i>Analisi Steady-State</i> .....	20
<i>Statistiche Ottenute</i> .....	25

## *Algoritmo 2*

<i>Goal e Obiettivi</i> .....	27
<i>Modello Concettuale</i> .....	27
<i>Modello di Specifica</i> .....	29
<i>Modello Computazionale</i> .....	35
<i>Verifica e Validazione</i> .....	36
<i>Intervalli di Confidenza</i> .....	37
<i>Analisi Transiente</i> .....	38
<i>Statistiche Ottenute</i> .....	41
<i>Analisi Steady-State</i> .....	44
<i>Statistiche Ottenute</i> .....	47
<i>Conclusioni</i> .....	51

# *Algoritmo 1*

## Goal e Obiettivi

I goal e gli obiettivi consistono nella realizzazione di un simulatore di tipo next-event per il sistema in cui il controllo degli accessi è gestito secondo il seguente algoritmo:

### **Algorithm 1**

- arrival:

**if**  $n_1 + n_2 = N$  → send on the Cloud

**else** accept the task on the Cloudlet

Si richiede di determinare se il sistema è stazionario o meno, di valutare il tempo medio di risposta, il throughput e la popolazione media globali e per classe, per il Cloudlet e per il Cloud. Si richiede inoltre di analizzare le statistiche del sistema transiente, ed eventualmente di quello steady-state. E' necessario poi definire un modello di code per descrivere il sistema. Tutti i risultati ottenuti tramite simulazione devono essere opportunamente validati e verificati.

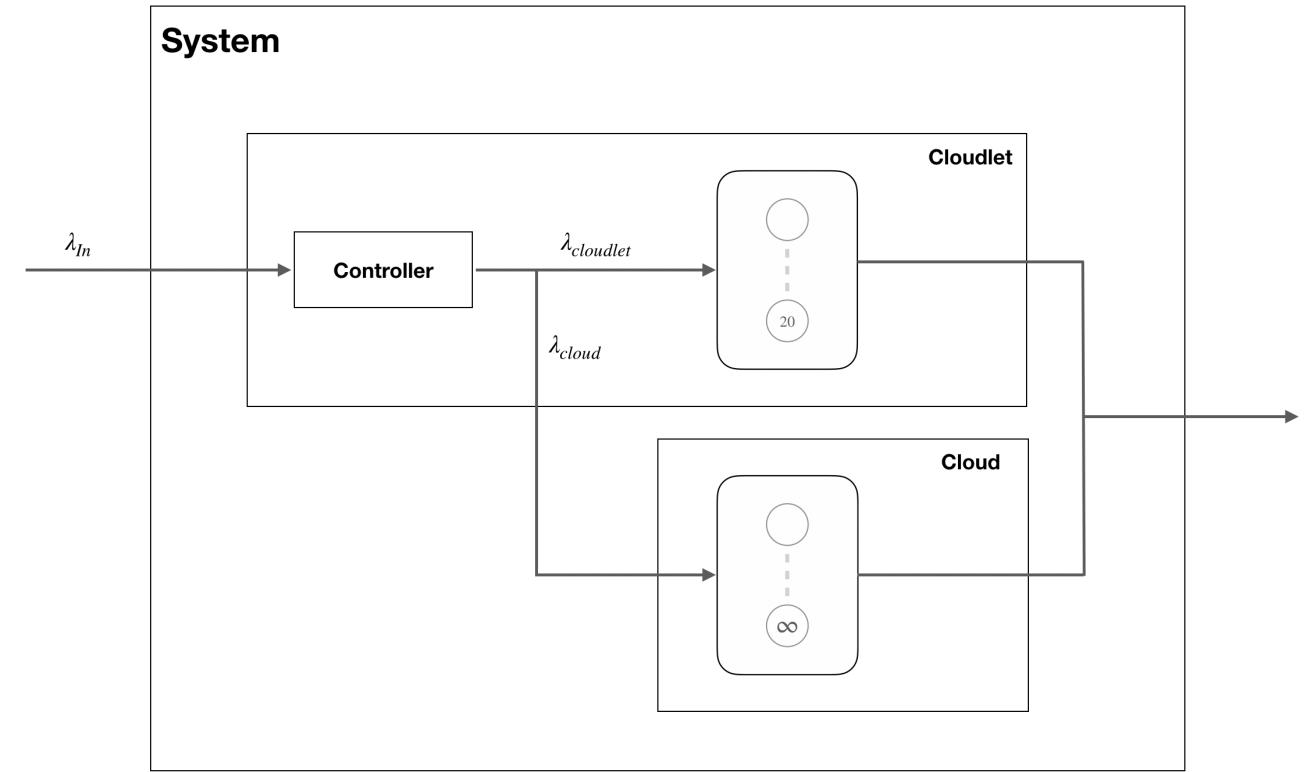
## Modello Concettuale

Il modello in analisi, con controllo degli accessi gestito secondo l'algoritmo numero 1, prevede che il flusso di job in arrivo al controller venga gestito in base al numero totale di job che sono presenti nel Cloudlet. Infatti, l'algoritmo prevede che finché la somma dei job di classe 1 ( $n_1$ ) e di classe 2 ( $n_2$ ) presenti nel Cloudlet è minore di  $N$  (numero dei server del Cloudlet), il Cloudlet possa accettare e gestire nuovi arrivi. Infatti, in tal caso il Cloudlet ha disponibili dei server in stato 'idle', e quindi è in grado di gestire ulteriori richieste. In caso contrario, ovvero quando  $n_1 + n_2 > N$ , cioè quanto tutti i server del Cloudlet sono in stato 'busy', il controller inoltrerà i nuovi job in arrivo al Cloud.

Per caratterizzare lo stato del sistema ad ogni istante di tempo sono state identificate le seguenti variabili di stato:

- $l(t)_{cllet}$  : numero totale di job presenti nel Cloudlet al tempo t
- $x_1(t)_{cllet}, x_2(t)_{cllet}, \dots, x_{20}(t)_{cllet}$  : stato dei server (0 o 1) nel Cloudlet al tempo t
- $x_1(t)_{cloud}, x_2(t)_{cloud}, x_3(t)_{cloud} \dots$  : stato dei server (0 o 1) nel Cloud al tempo t

Il modello che caratterizza il sistema in esame è mostrato nella figura di seguito. Il flusso in ingresso è stato rappresentato con la variabile  $\lambda_{In}$ , e rappresenta il flusso totale che arriva al sistema, comprendente il flusso dei job di classe 1 e il flusso dei job si classe 2. Tale flusso arriva al Controller, che è un componente del Cloudlet. Una parte di questo flusso verrà gestito dal Cloudlet, mentre un'altra parte sarà inoltrata al Cloud. Gli arrivi al Cloud dipendono dallo stato del Cloudlet; in particolare, al Cloud verranno inviati nuovi job solo quando tutti i server del Cloudlet sono occupati. Indichiamo il flusso in ingresso al sistema come somma dei flussi in ingresso dei job di classe 1 e di classe 2. Indichiamo inoltre il flusso in ingresso in ciascuno dei due server nel modo seguente:  $\lambda_{cloudlet}$  indica il flusso in ingresso totale (ovvero comprende quello dei job di classe 1 e quello dei job di classe 2) al Cloudlet, mentre  $\lambda_{cloud}$  indica il flusso totale in ingresso al Cloud.



## Modello di Specifica

Gli ingressi nel sistema, rappresentati con la variabile  $\lambda_{In}$ , costituiscono la quantità totale dei task che i dispositivi mobili richiedono di gestire a server esterni. Possiamo considerare tale flusso in ingresso come un flusso generato da eventi aleatori, in quanto la richiesta da parte dei dispositivi mobili di gestione dei task non segue regole precise. Di conseguenza si può assumere che gli arrivi al sistema possano essere modellati secondo distribuzioni di Poisson di parametri  $\lambda_1$  e  $\lambda_2$  per job di classe 1 e 2 rispettivamente. Sapendo che i tempi di interarrivo di un processo poissoniano si distribuiscono esponenzialmente, è possibile supporre che  $\frac{1}{\lambda_1}$  e  $\frac{1}{\lambda_2}$  rappresentino i tempi di interarrivo per job di classe 1 e 2 rispettivamente.

In particolare, è noto che la somma di due variabili aleatorie con distribuzione di Poisson di parametri  $\lambda_1$  e  $\lambda_2$  segue ancora una distribuzione di Poisson con parametro  $\lambda = \lambda_1 + \lambda_2$ . Si può affermare quindi che il flusso in ingresso al sistema è modellabile con una distribuzione di Poisson, e che il tempo di interarrivo dei job nel sistema è dato da

$$\frac{1}{\lambda_1 + \lambda_2}.$$

I flussi in ingresso nel server Cloudlet e nel server Cloud sono stati indicati con le variabili  $\lambda_{cloudlet}$  e  $\lambda_{cloud}$ . Poiché il Cloud riceve job solo quando il Cloudlet non è in grado di gestirli, è possibile definire il flusso in ingresso al Cloud in riferimento alla probabilità che tutti i server del Cloudlet sono occupati ( $\pi_{20}$ ). Il flusso in ingresso al Cloud è dato quindi dalla probabilità che un job arrivi al Cloud moltiplicata per il flusso totale in ingresso:

$$\lambda_{cloud} = \pi_{20} \cdot \lambda_{In}$$

In maniera analoga è possibile definire il flusso in ingresso al Cloudlet come il prodotto del flusso totale per la probabilità che un job possa essere gestito dal Cloudlet:

$$\lambda_{cloudlet} = (1 - \pi_{20}) \cdot \lambda_{In}$$

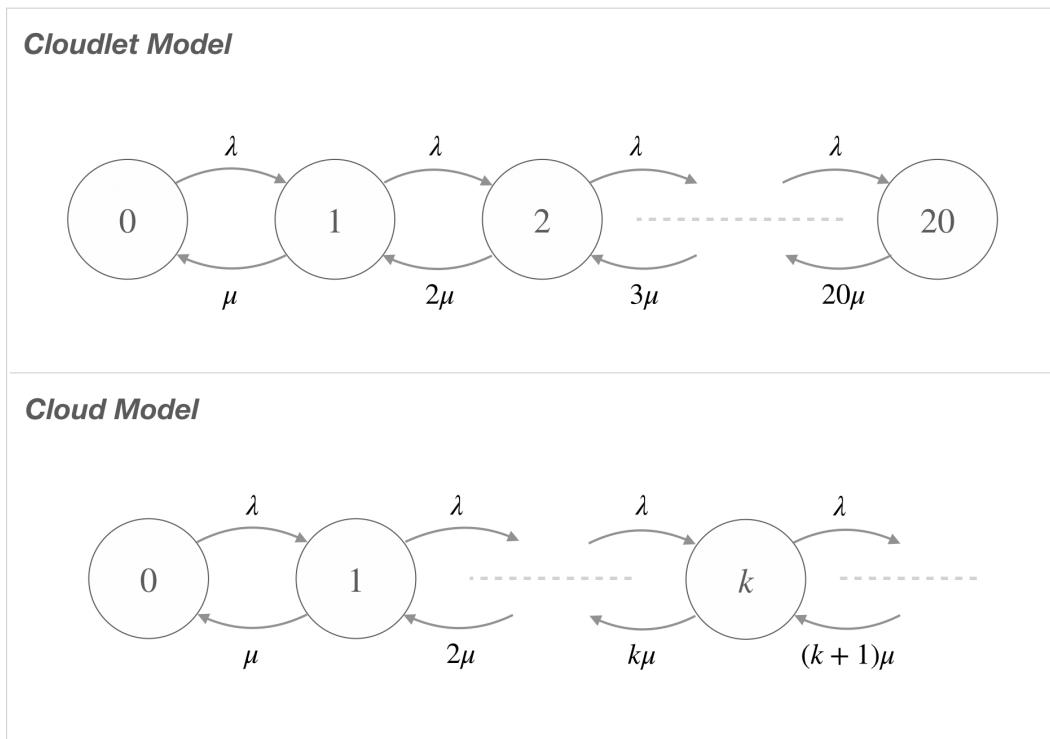
Per le successive analisi si ipotizza che sia gli arrivi al Cloudlet che gli arrivi al Cloud siano di Poisson, e che quindi i rispettivi tempi di interarrivo siano esponenziali. Assumendo che i tempi di interarrivo siano esponenziali, sapendo che i tempi di servizio sono anch'essi

esponenziali, che il Cloudlet possiede 20 risorse, e che il Cloud ha un numero illimitato di risorse, è possibile definire un modello di coda per il Cloudlet e per il Cloud:

- Cloudlet :  $M/M/20/20$
- Cloud :  $M/M/\infty$

Quindi, il Cloudlet è rappresentato come un multi-server con 20 serventi e capacità del sistema pari a 20, mentre il Cloud è rappresentato come un multi-server con  $\infty$  serventi e capacità infinita.

I sistemi di coda definiti precedentemente sono stati modellati con le seguenti catene di Markov:



Avendo definito per il Cloudlet un modello di tipo  $M/M/20/20$ , è possibile quantificare la probabilità di blocco, ovvero la probabilità che tutti i server siano busy tramite la formula Erlang-B:

$$\pi_k = \left(\frac{\lambda}{\mu}\right)^k \cdot \frac{\pi_0}{k!}$$

Dove la probabilità  $\pi_0$  è data da:  $\pi_0 = \frac{1}{\sum_{i=0}^k \left(\frac{\lambda}{\mu}\right)^i \cdot \frac{1}{i!}}$

Nel caso in esame ( $k = 20$ ) la probabilità di blocco è data da:

$$\pi_{20} = \left(\frac{\lambda_{In}}{\mu_{clet}}\right)^{20} \cdot \frac{\pi_0}{20!} \quad \text{Con} \quad \pi_0 = \frac{1}{\sum_{i=0}^{20} \left(\frac{\lambda_{In}}{\mu_{clet}}\right)^i \cdot \frac{1}{i!}}$$

Il parametro  $\lambda_{In}$  rappresenta il flusso totale in ingresso nel sistema, mentre il parametro  $\mu_{clet}$  rappresenta il service rate complessivo del Cloudlet. Tale parametro può essere calcolato a partire dal tempo medio di servizio nel Cloudlet, il quale può essere ottenuto calcolando la media ponderata del tempo medio di servizio per i job di classe 1 e del tempo medio di servizio per job di classe 2:

$$E[S]_{clet} = p_1 \cdot E[S]_{clet1} + p_2 \cdot E[S]_{clet2}$$

Dove  $E[S]_{clet1}$  e  $E[S]_{clet2}$  rappresentano i tempi medi di servizio per job di tipo 1 e di tipo 2 rispettivamente, mentre i valori  $p_1$  e  $p_2$  rappresentano la probabilità di avere job di tipo 1 e di tipo 2, e possono essere calcolate come segue:

$$p_1 = \frac{\lambda_1}{\lambda_{In}} \quad , \quad p_2 = \frac{\lambda_2}{\lambda_{In}}$$

A partire dalle considerazioni effettuate in precedenza, è possibile ricavare i valori teorici dei parametri in questione:

$$p_1 = \frac{\lambda_1}{\lambda_{In}} = \frac{4}{10.25} = 0.390244$$

$$p_2 = \frac{\lambda_2}{\lambda_{In}} = \frac{6.25}{10.25} = 0.609756$$

$$E[S]_{clet1} = \frac{1}{\mu_{clet1}} = \frac{1}{0.45} = 2.22222$$

$$E[S]_{clet2} = \frac{1}{\mu_{clet2}} = \frac{1}{0.27} = 3.703704$$

$$E[S]_{clet} = p_1 \cdot E[S]_{clet1} + p_2 \cdot E[S]_{clet2} = 3.125565$$

Quindi si ottiene:  $\mu_{clet} = \frac{1}{E[S]_{clet}} = \frac{1}{3.125565} = 0.319942$

$$\pi_{20} = \left( \frac{\lambda_{In}}{\mu_{clet}} \right)^{20} \cdot \frac{\pi_0}{20!} = 0.414549$$

(il calcolo di tale valore per la probabilità di blocco è stato effettuato utilizzando un codice ausiliario scritto in Java, e verificato risolvendo la catena di Markov con Matlab).

$$\lambda_{cloudlet} = (1 - \pi_{20}) \cdot \lambda_{In} = (1 - 0.414549) \cdot 10.25 = 6.000873$$

$$\lambda_{cloud} = \pi_{20} \cdot \lambda_{In} = 0.414549 \cdot 10.25 = 4.249127$$

Con un procedimento analogo è possibile ricavare i valori teorici relativi ai tempi di servizio del Cloud:

$$E[S]_{cloud1} = \frac{1}{\mu_{cloud1}} = \frac{1}{0.25} = 4 \quad E[S]_{cloud2} = \frac{1}{\mu_{cloud2}} = \frac{1}{0.22} = 4.545455$$

Supponendo che per il Cloud le probabilità di avere job di classe 1 e di classe 2 rimangano le stesse che si hanno per il controller, il tempo medio di servizio complessivo del Cloud è il seguente:

$$E[S]_{cloud} = p_1 \cdot E[S]_{cloud1} + p_2 \cdot E[S]_{cloud2} = 4.332594$$

$$\mu_{cloud} = \frac{1}{E[S]_{cloud}} = \frac{1}{4.332594} = 0.230809$$

Disponendo di tali dati, e conoscendo il valore della probabilità di blocco, è possibile ricavare il valore teorico del tempo di risposta complessivo del sistema.

Data l'assenza di code sia per il Cloudlet che per il Cloud, i tempi medi di risposta (per il Cloudlet, per il Cloud e complessivo) coincidono con i tempi medi di servizio:

$$E[S]_{clet} = E[t]_{clet} \quad E[S]_{cloud} = E[t]_{cloud}$$

$$E[t]_{system} = (1 - \pi_{20}) \cdot E[t]_{clet} + \pi_{20} \cdot E[t]_{cloud} = 3.625938$$

Inoltre, è possibile ricavare il valore del tempo di servizio del sistema per classe, ovvero relativo ai job di classe 1 e ai job di classe 2:

$$E[t_1]_{system} = (1 - \pi_{20}) \cdot E[S]_{clet1} + \pi_{20} \cdot E[S]_{cloud1} = 2.959198$$

$$E[t_2]_{system} = (1 - \pi_{20}) \cdot E[S]_{clet2} + \pi_{20} \cdot E[S]_{cloud2} = 4.052651$$

Avendo a disposizione i dati relativi al flusso in ingresso nei server, e conoscendo i tempi di risposta nei differenti casi, applicando la legge di Little è possibile ricavare i valori teorici riguardanti la popolazione media. Di seguito verranno calcolate la popolazione media nel Cloudlet e nel Cloud, totale e relativa a ciascuna classe:

$$E[N]_{clet} = \lambda_{clet} \cdot E[t]_{clet} = 18.756119$$

$$E[N_1]_{clet} = p_1 \cdot \lambda_{clet} \cdot E[t_1]_{clet} = 5.20401$$

$$E[N_2]_{clet} = p_2 \cdot \lambda_{clet} \cdot E[t_2]_{clet} = 13.552106$$

$$E[N]_{cloud} = \lambda_{cloud} \cdot E[t]_{cloud} = 18.409742$$

$$E[N_1]_{cloud} = p_1 \cdot \lambda_{cloud} \cdot E[t_1]_{cloud} = 6.632785$$

$$E[N_2]_{cloud} = p_2 \cdot \lambda_{cloud} \cdot E[t_2]_{cloud} = 11.776959$$

Per il calcolo dei valori teorici del throughput si considera la seguente relazione:

$$X = \min(\lambda, \mu)$$

Il sistema globale risulta essere stazionario, in quanto il server Cloud possiede una quantità infinita di risorse, in che rende il sistema in grado di gestire tutto il flusso in ingresso. Di conseguenza, il throughput è pari alla quantità di flusso in ingresso:

$$X = \lambda_{In} = 10.25$$

$$X_1 = \lambda_1 = 4$$

$$X_2 = \lambda_2 = 6.25$$

$$X_{clet} = \lambda_{clet} = 6.000873$$

$$X_{cloud} = \lambda_{cloud} = 4.249127$$

$$X_{clet1} = p_1 \cdot \lambda_{clet} = 2.341805$$

$$X_{clet2} = p_2 \cdot \lambda_{clet} = 3.659068$$

$$X_{cloud1} = p_1 \cdot \lambda_{cloud} = 1.658196$$

$$X_{cloud2} = p_2 \cdot \lambda_{cloud} = 2.590931$$

## Modello Computazionale

Il codice che implementa il simulatore next-event è stato realizzato in Java. Per la gestione degli eventi sono state utilizzate due strutture differenti: nel caso della gestione degli eventi del Cloudlet, data la dimensione fissa del numero di eventi gestibili, è stata utilizzata una variabile di tipo array, per cui ogni evento è associato ad un servente del Cloudlet; nel caso della gestione degli eventi del Cloud, invece, è stata inutilizzata una struttura dinamica, implementata come ‘heap’. In entrambi i casi, ogni evento tiene traccia di diverse informazioni:

- Un campo  $t$  in cui viene memorizzato il tempo di avvenimento dell’evento
- Un campo  $x$  che rappresenta lo stato del server (idle o busy)
- Un campo  $type$  che tiene traccia della tipologia di evento (classe 1 o classe 2)

Poiché un evento rappresenta qualcosa che altera lo stato del sistema, e che quindi comporta una modifica nei valori delle variabili di stato, sono stati considerati come eventi i seguenti:

- Arrivi di job di tipo 1 al controller
- Arrivi di job di tipo 2 al controller
- Partenze di job di tipo 1 dal Cloudlet
- Partenze di job di tipo 2 dal Cloudlet
- Partenze di job di tipo 1 dal Cloud
- Partenze di job di tipo 2 dal Cloud

Ciascuno di questi eventi comporta la modifica di alcune delle variabili di stato: del numero di job nel Cloudlet o dello stato dei server.

Il codice è composto da due fasi principali: una fase di inizializzazione e una fase di elaborazione. Nella fase di inizializzazione vengono inizializzate tutte le variabili che verranno utilizzate in seguito, e si utilizza la libreria RNGS per la generazione di numeri pseudo-casuali. Viene inoltre generato il primo evento (che è un arrivo) e viene assegnato al Cloudlet. Successivamente, viene utilizzato un ciclo ‘while’ per tutta la durata della simulazione; la condizione di terminazione viene imposta da un “close the door time”, allo scadere del quale viene terminata la simulazione e vengono completati tutti i job presenti nel sistema. Ogni ripetizione del ciclo ‘while’ inizia con l’invocazione del metodo ‘*nextEvent*’ che si occupa di determinare quale sarà l’evento successivo. Tale funzione analizza le strutture dati che memorizzano gli eventi di Cloudlet e Cloud alla ricerca

dell'evento più imminente; gli eventi vengono inizialmente suddivisi in due categorie: arrivi e partenze.

### Arrivi

Nel caso l'evento più imminente sia un arrivo, viene controllato il numero di job di classe 1 e di classe 2 presenti nel Cloudlet: se la loro somma è minore del numero dei server, il job viene assegnato al Cloudlet. In tal caso, si cerca quale dei server del Cloudlet sia 'idle' da più tempo, e gli si assegna tale job (il server diventa conseguentemente 'busy'). Se invece tutti i server del Cloudlet sono 'busy', il job viene inoltrato al Cloud; in questo caso, dal momento che il Cloud possiede risorse infinite, il job viene assegnato al primo server disponibile (ciò viene fatto effettuando una 'insert' dell'evento della struttura dati 'heap').

In entrambi i casi, la gestione di un arrivo prevede la definizione del tipo di arrivo in questione; si utilizzano i metodi '*getArrival1*' e '*getArrival2*' per simulare i tempi di arrivo dei job di classe 1 e di classe 2 (si utilizza un'esponenziale il cui parametro dipende dalla classe del job). Si confrontano i tempi ottenuti: l'evento il cui tempo di arrivo è minore sarà quello che verrà gestito.

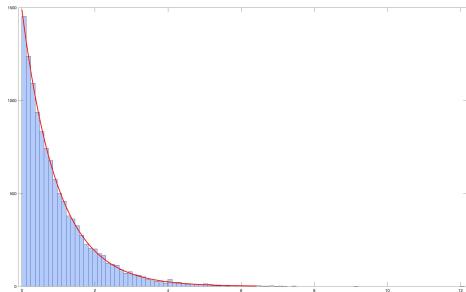
### Partenze

Un evento di completamento può verificarsi sul Cloud o sul Cloudlet. La gestione di tali eventi differisce leggermente nei due casi, date le diverse strutture dati di gestione. Nel caso di completamenti sul Cloudlet, il metodo '*nextEvent*' restituisce l'indice del server sul quale si verifica l'evento. Si controlla la classe del job che viene completato, e si utilizzano i metodi '*getService1\_clet*' o '*getService2\_clet*' per simulare i tempi di servizio corrispondenti (si utilizza un'esponenziale il cui parametro dipende dalla classe del job). Dopo aver aggiornato le variabili di interesse, si marca il server come 'idle'.

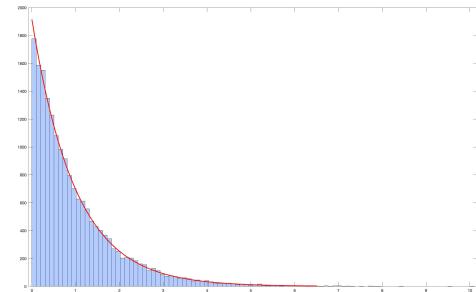
Nel caso di completamenti dal Cloud, la struttura dati heap permette di organizzare gli eventi in modo che siano automaticamente ordinati in base al loro istante di avvenimento. Di conseguenza, il completamento di un job viene gestito rimuovendo l'elemento 'root' della struttura. Anche in questo caso, in base alla classe di appartenenza del job completato, si utilizzano i metodi '*getService1\_Cloud*' o '*getService2\_Cloud*' per simulare i tempi di servizio corrispondenti.

## Verifica e Validazione

Per la verifica e la validazione del modello simulato sono stati tenuti in considerazione diversi aspetti. In primo luogo, sono state studiate le distribuzioni degli interarrivi dei job di classe 1 e di classe 2 e confrontati con la distribuzione esponenziale; come si può notare dai grafici ottenuti, la supposizione di arrivi di Poisson può essere ritenuta corretta:

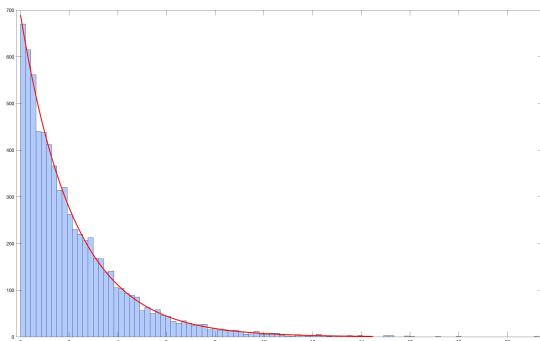


Distribuzione Interarrivi job di classe 1

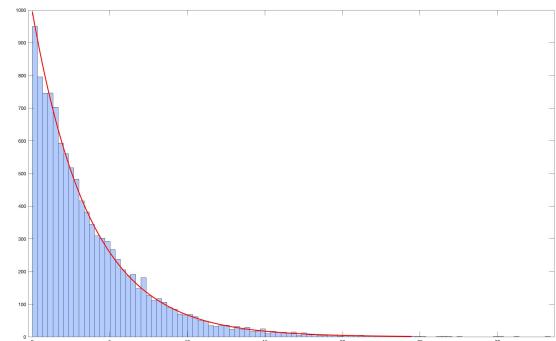


Distribuzione Interarrivi job di classe 2

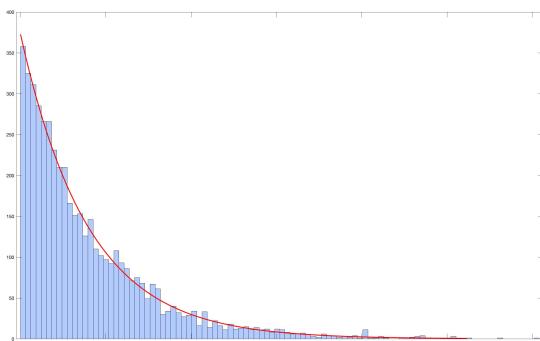
Come conferma del fatto che i tempi di servizio sia per il Cloudlet che per il Cloud (per job di classe 1 e di classe 2) siano esponenziali, sono state analizzate le distribuzioni dei tempi di servizio e confrontate con le distribuzioni esponenziali:



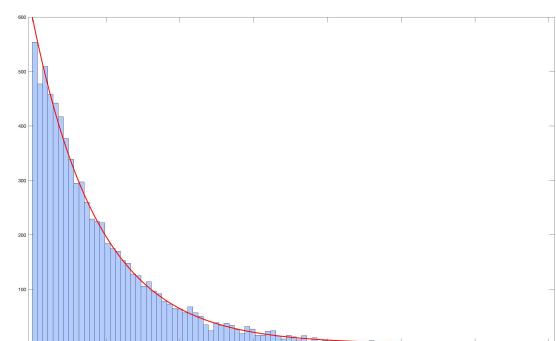
Distribuzione tempi di servizio per job di classe 1 sul Cloudlet



Distribuzione tempi di servizio per job di classe 2 sul Cloudlet



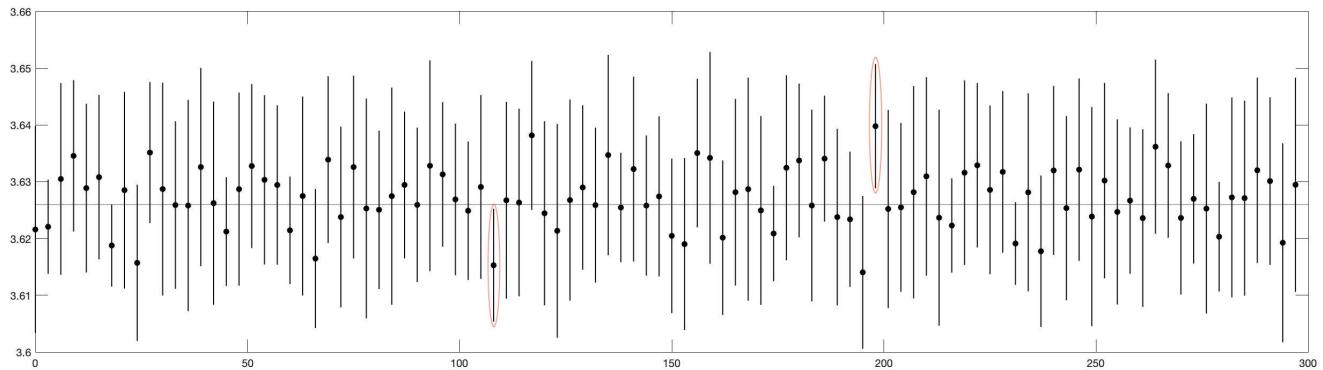
Distribuzione tempi di servizio per job di classe 1 sul Cloud



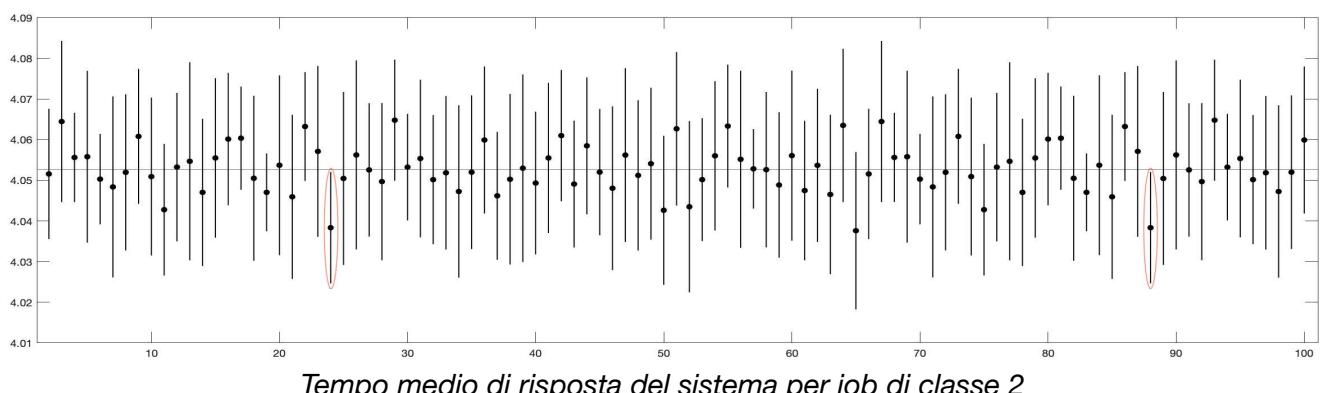
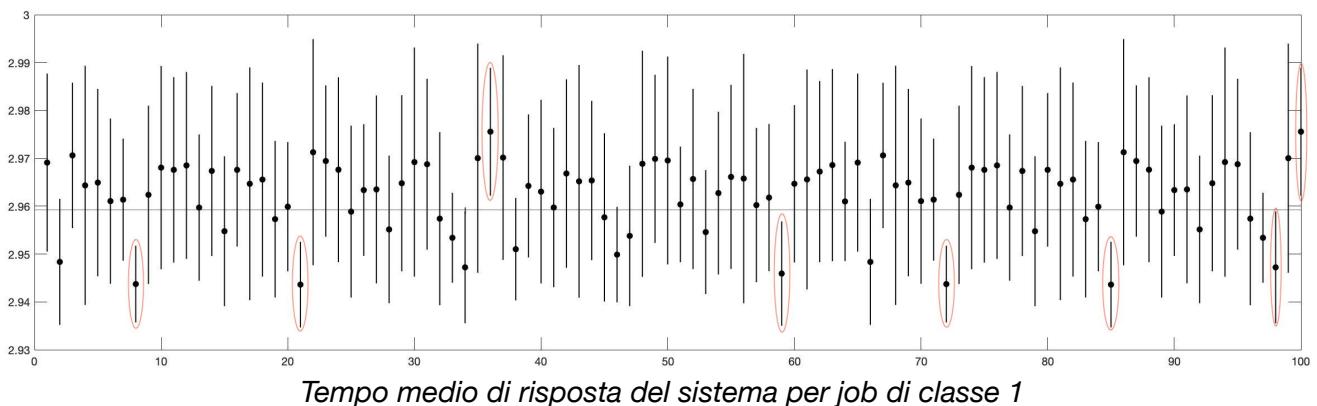
Distribuzione tempi di servizio per job di classe 2 sul Cloud

## Intervalli di Confidenza

I grafici seguenti mostrano cento intervalli di confidenza al 95% per alcune delle statistiche di interesse. Ogni intervallo di confidenza è relativo ad un campione di 10 elementi. Di seguito viene riportato un grafico che rappresenta la statistica riguardante il tempo di risposta medio del sistema:

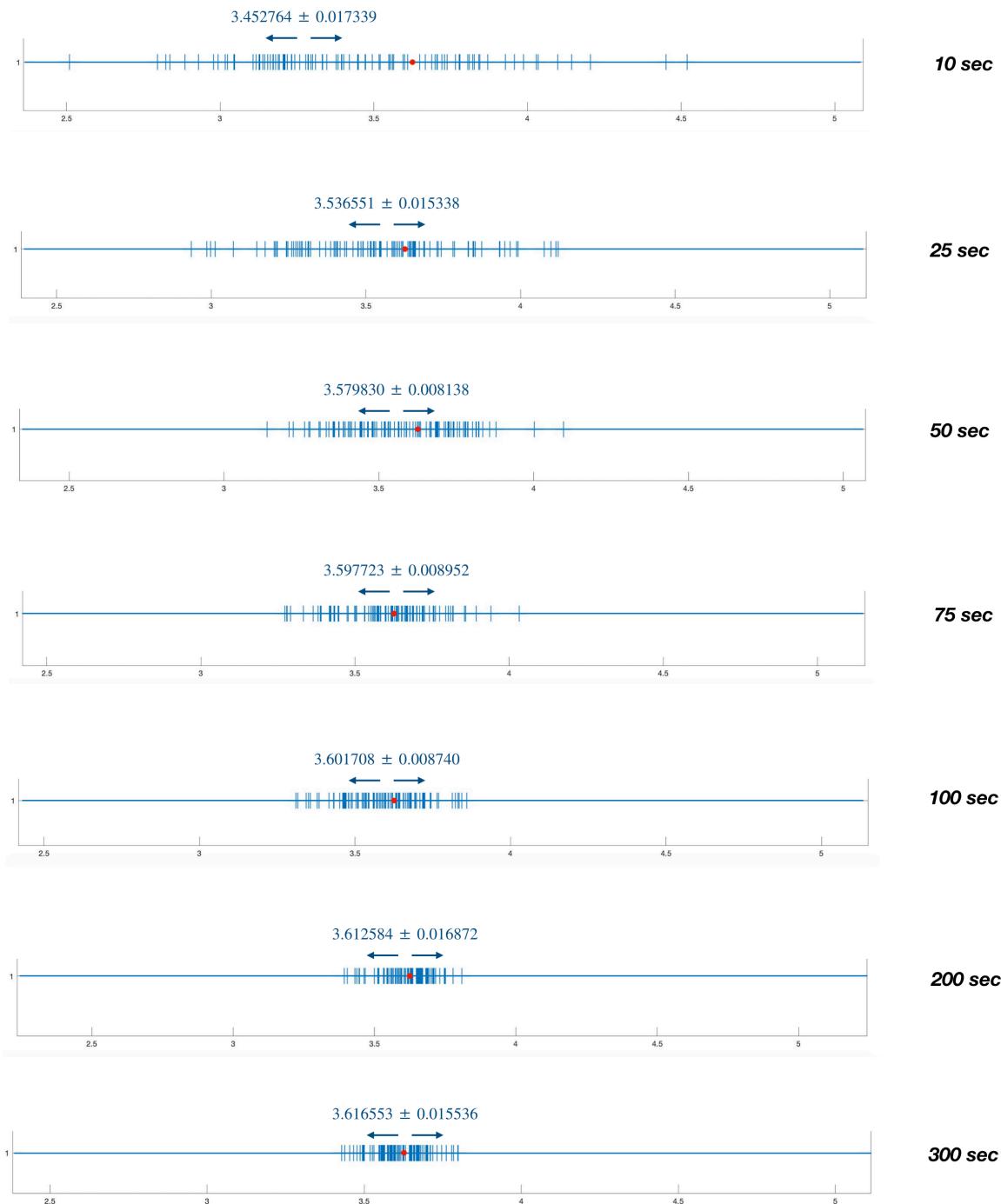


Si può notare che dei 100 intervalli di confidenza ottenuti, solo due non contengono il valore teorico. Vengono riportati di seguito i grafici relativi agli intervalli di confidenza per i tempi medi di risposta per classi:

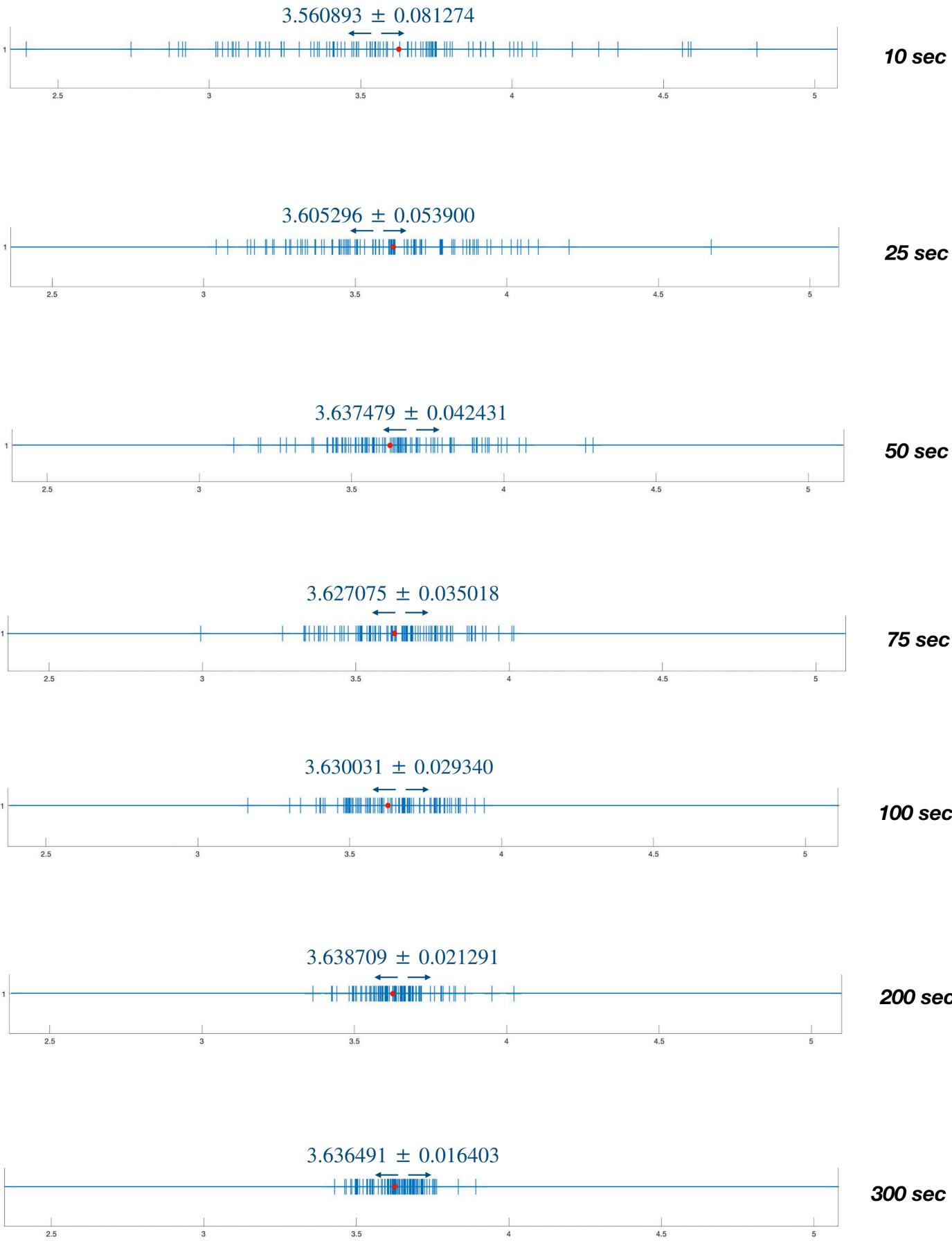


## Analisi Transiente

L'obiettivo di questo tipo di analisi è quello di definire il comportamento del sistema in un fissato intervallo di tempo di simulazione, per il quale vengono fissate le condizioni iniziali e di terminazione, e il sistema al termine della simulazione ritorna allo stato iniziale. Sono riportate le statistiche relative al tempo di risposta globale del sistema ipotizzando valori diversi per il “close the door time” e valori diversi per il seed iniziale. I valori simulati sono rappresentati in blu, mentre il puntino rosso rappresenta il valore steady-state del tempo medio di risposta. Si riportano i grafici relativi al seed iniziale 123456789:

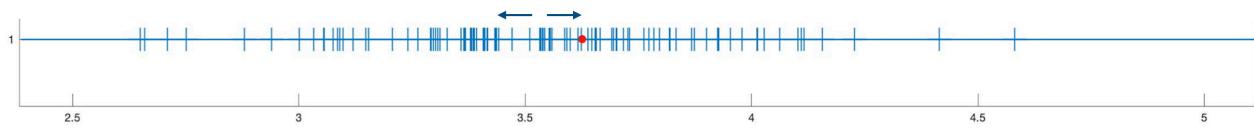


Seed = 987654321



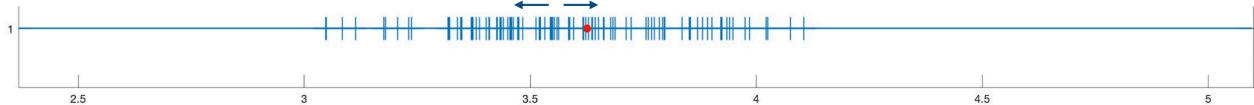
Seed = 5555555555

$3.532392 \pm 0.074126$



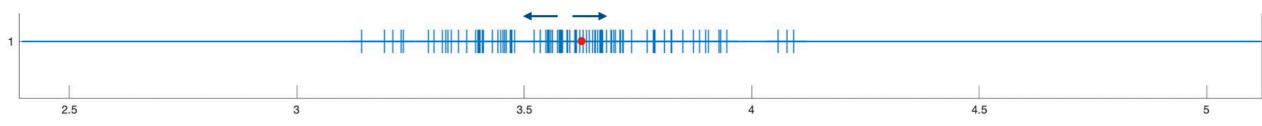
10 sec

$3.587379 \pm 0.048273$



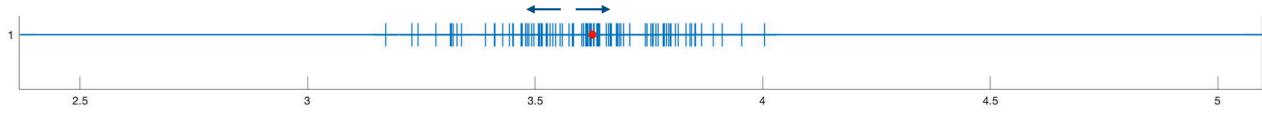
25 sec

$3.599408 \pm 0.039355$



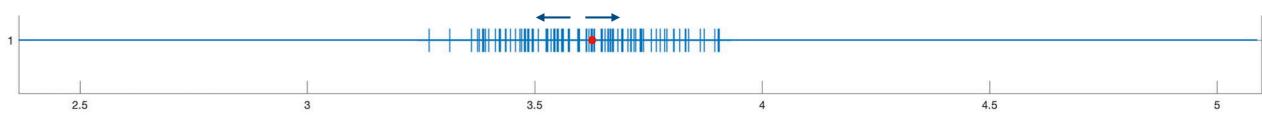
50 sec

$3.605655 \pm 0.034411$



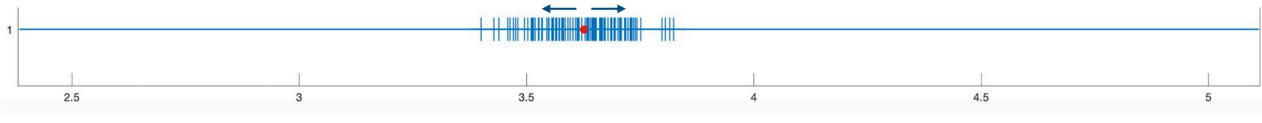
75 sec

$3.612042 \pm 0.028738$



100 sec

$3.623516 \pm 0.017977$



200 sec

$3.630410 \pm 0.014498$



300 sec

Dai grafici ottenuti è possibile notare che i valori simulati variano, a parità di istante di tempo e di condizioni iniziali, in base al seme scelto. In tutti e tre i casi si nota che all'aumentare del tempo di simulazione i valori di avvicinano sempre più al valore steady-state e gli intervalli di confidenza diventano più piccoli. Si nota quindi la presenza di un bias iniziale, dovuto al fatto che è stato scelto come stato iniziale del sistema lo stato idle.

Nel secondo e nel terzo caso a partire dal tempo simulato di 100 secondi si ottiene un intervallo di confidenza che contiene in valore teorico; ciò non avviene per il primo caso. In tutti e tre i casi, a partire dai 200s l'intervallo di confidenza contiene il valore teorico; di conseguenza, le successive valutazioni terranno conto di un tempo di simulazione di 200s.

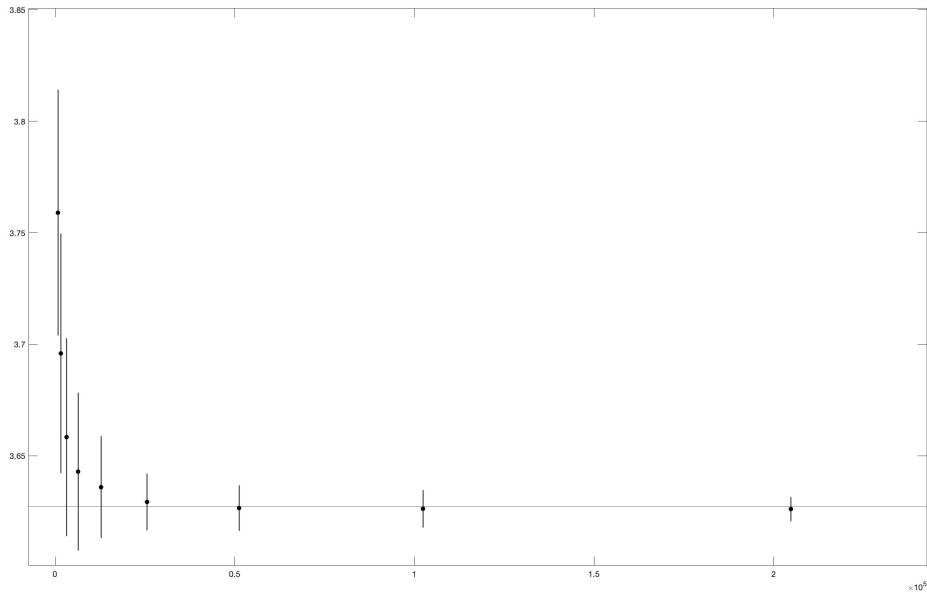
Sono state effettuate 100 simulazioni per il “close the door time” stabilito e sono state analizzate alcune delle statistiche di interesse considerando tre valori differenti del seed iniziale; per ognuna di esse è stato calcolato il relativo intervallo di confidenza al 95%.

Seed = 123456789	Valore simulato	Intervallo di Confidenza
$E[t]$	3.612584	[ 3.595712 , 3.629457 ]
$\lambda_{tot}$	9.651398	[ 9.589879 , 9.712918 ]
$E[N]_{clet}$	18.681686	[ 18.662785 , 18.700587 ]
$E[N]_{cloud}$	18.365273	[ 18.125785 , 18.604761 ]

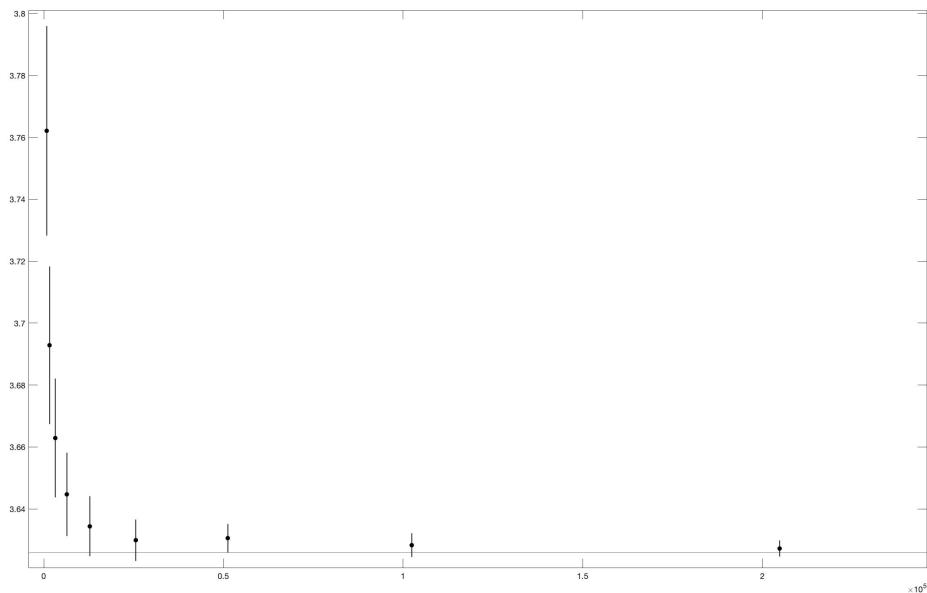
Seed = 987654321	Valore simulato	Intervallo di Confidenza
$E[t]$	3.638709	[ 3.617418 , 3.659999 ]
$\lambda_{tot}$	9.630537	[ 9.567377 , 9.693696 ]
$E[N]_{clet}$	18.688330	[ 18.667123 , 18.709538 ]
$E[N]_{cloud}$	18.666585	[ 18.392575 , 18.940595 ]

Seed = 555555555	Valore simulato	Intervallo di Confidenza
$E[t]$	3.623516	[ 3.605539 , 3.641493 ]
$\lambda_{tot}$	9.679277	[ 9.613478 , 9.745076 ]
$E[N]_{clet}$	18.698785	[ 18.678275 , 18.719294 ]
$E[N]_{cloud}$	18.532654	[ 18.275150 , 18.790158 ]

Per studiare il comportamento del sistema nella fase transitoria è stata utilizzata la libreria RNGS per creare 16 e 64 repliche indipendenti della simulazione, utilizzando per ognuna stream diversi per i tempi di servizio e di arrivo dei job. Sono state analizzate le statistiche riguardanti il tempo di risposta del sistema per 800, 1600, ...., 204800 job processati.



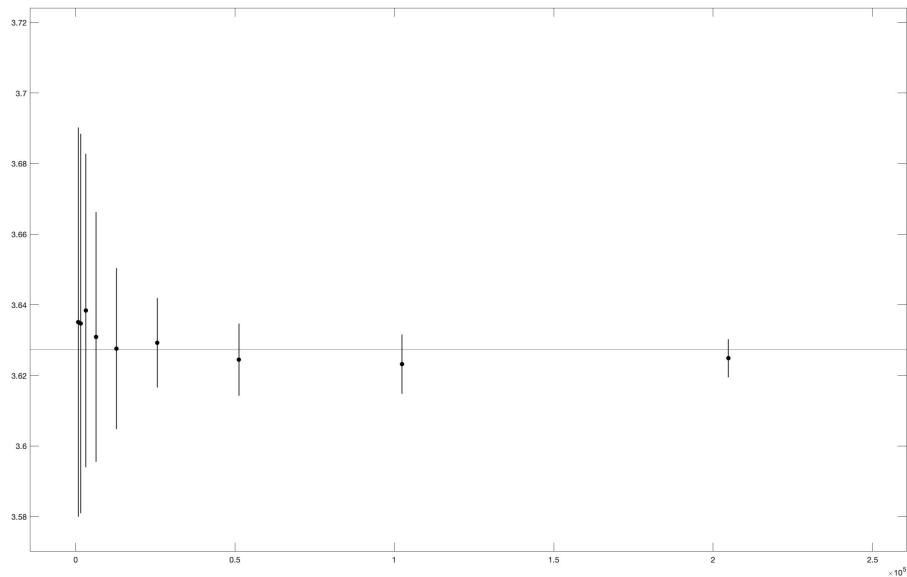
*Statistiche ottenute da 16 repliche*



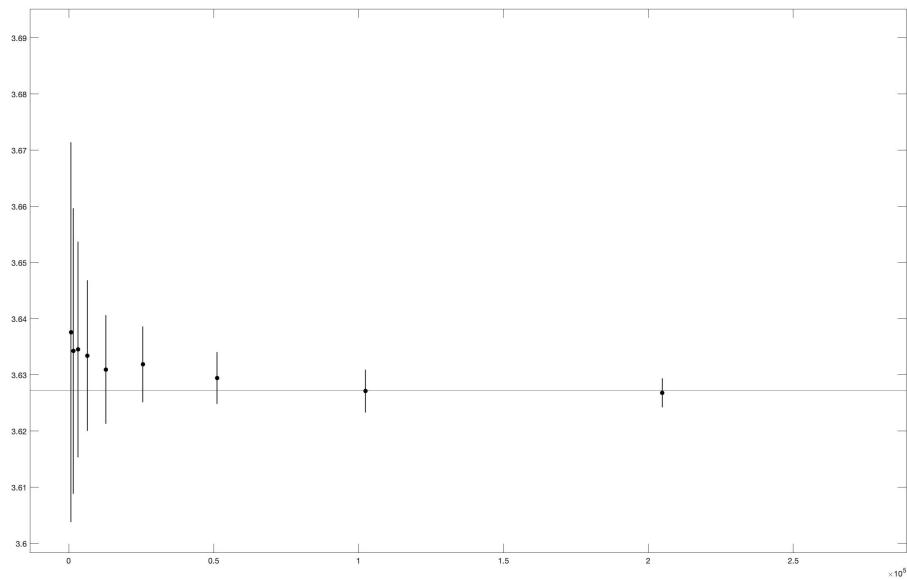
*Statistiche ottenute da 64 repliche*

Dai grafici emerge che all'aumentare del numero di job processati gli intervalli di confidenza diventano più piccoli e il valore medio di ogni intervallo si avvicina sempre di più al valore steady-state della statistica (rappresentato dalla retta orizzontale).

Si può notare che l'utilizzo di un numero maggiore di repliche implica una diminuzione generale dell'ampiezza degli intervalli di confidenza. In entrambi i casi vi è una conferma della presenza di un bias iniziale, in quanto lo stato iniziale del sistema è stato considerato idle per queste simulazioni. Tale bias iniziale può essere eliminato effettuando simulazioni che iniziano da uno stato del sistema non idle. Vengono riportati i grafici ottenuti:



*Statistiche ottenute da 16 repliche*



*Statistiche ottenute da 64 repliche*

## Analisi Steady-State

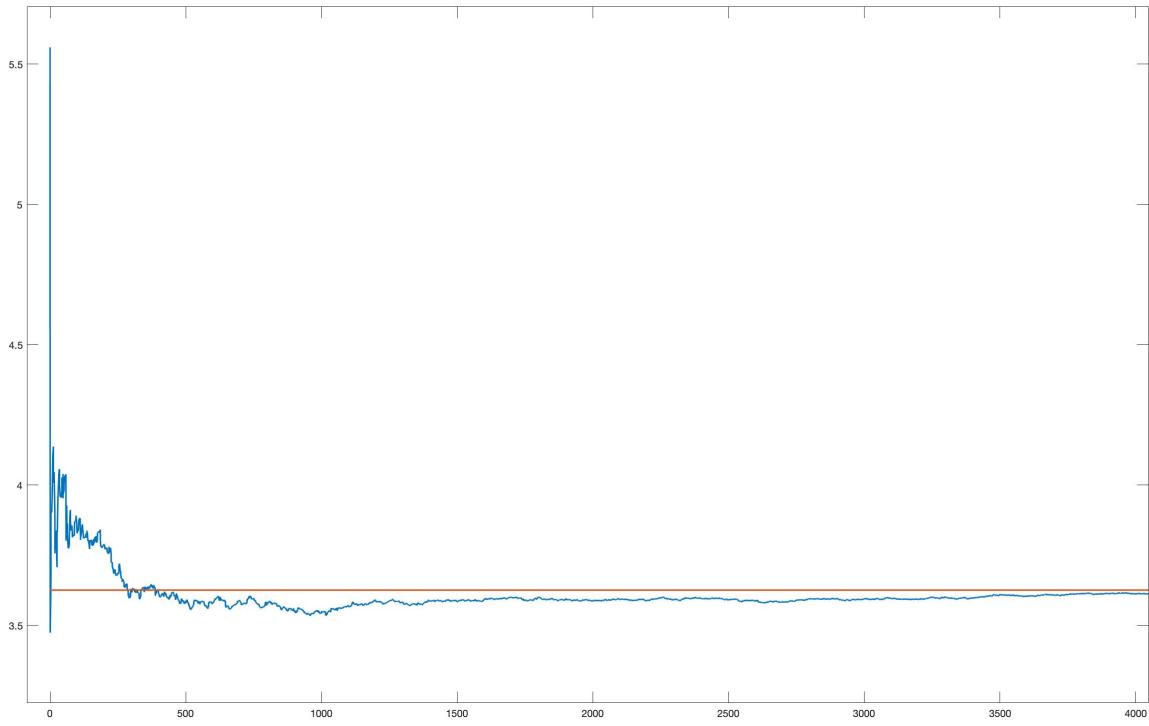
L'obiettivo di tale tipo di analisi è quello di definire il comportamento del sistema a regime. Per il seed iniziale 123456789, sono stati valutati in diversi istanti di tempo durante la simulazione alcuni indici rilevanti fino al raggiungimento di una fase stazionaria. Per tale studio si è scelto di utilizzare una situazione iniziale in cui il sistema è vuoto.

	$E[t]$	$\lambda_{tot}$	$\lambda_1$	$\lambda_2$	$E[N]_{cler}$	$E[N]_{cloud}$
$t = 10$	2.690672	6.062588	3.119584	2.943004	16.165829	4.558252
$t = 20$	3.337147	6.984552	3.008198	3.976354	17.343750	7.945545
$t = 50$	3.651543	8.480281	3.472877	5.007404	18.425489	16.743809
$t = 100$	3.698930	9.235807	3.608693	5.627114	18.683985	19.711447
$t = 200$	3.721931	9.788025	3.825969	5.962056	18.788485	19.654461
$t = 300$	3.578763	9.935474	3.893851	6.041624	18.744329	18.287577
$t = 500$	3.558119	10.127909	4.002030	6.125879	18.733595	18.198162
$t = 1000$	3.534952	10.028000	3.924599	6.103401	18.722956	18.024239
$t = 2000$	3.582014	10.123790	3.959600	6.164190	18.773815	18.380644
$t = 3000$	3.588449	10.128462	3.948120	6.180342	18.787148	18.611860
$t = 5000$	3.595622	10.208835	3.984671	6.224164	18.776148	18.723505
$t = 10000$	3.609160	10.237112	4.008742	6.228470	18.783915	18.932980
$t = 25000$	3.607175	10.235441	4.016547	6.218894	18.795654	18.935095
$t = 50000$	3.618278	10.244147	3.990003	6.240316	18.803772	19.078245
$t = 100000$	3.619258	10.231806	3.995188	6.236618	18.805944	19.058563
$t = 300000$	3.625254	10.248344	3.996570	6.251775	18.808910	19.188370
$t = 500000$	3.625301	10.249372	3.998352	6.248339	18.809729	19.187216

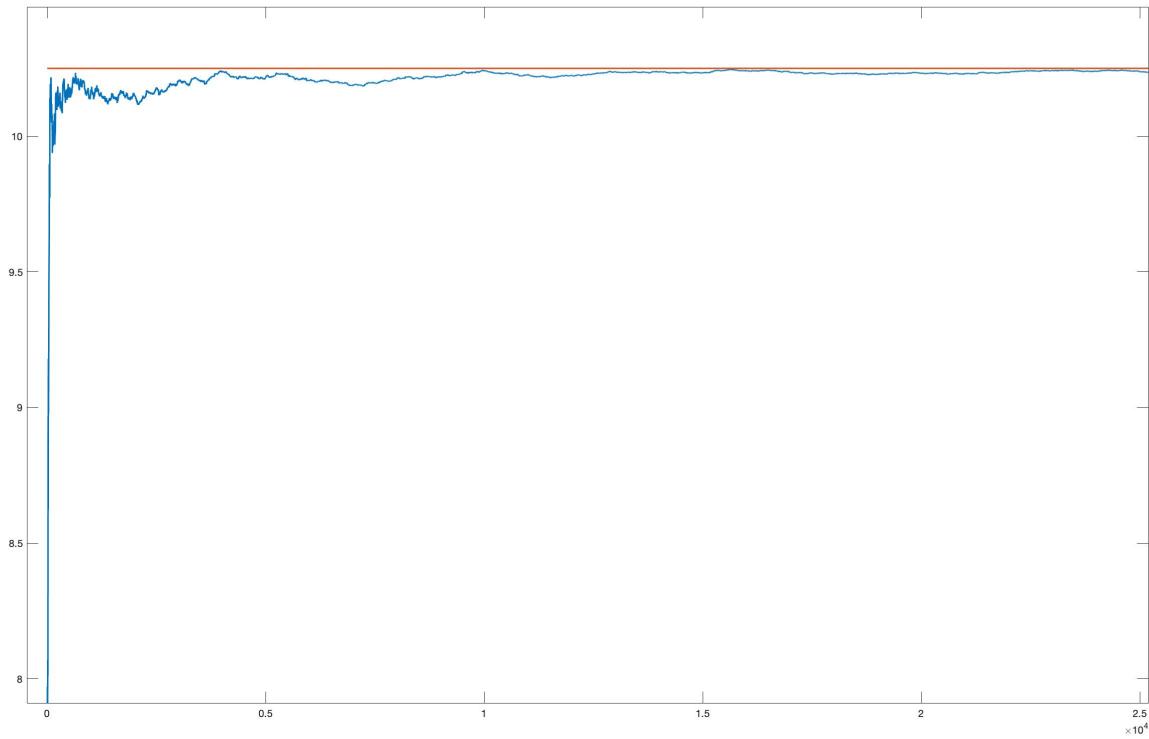
Dalla tabella precedente è possibile notare che è presente una fase di transitorio fino al tempo 50000s, dopo di che si può affermare che il sistema in generale raggiunge uno stato di regime.

In seguito verrà quindi considerato un tempo di simulazione pari a 50000s per l'analisi delle statistiche di interesse.

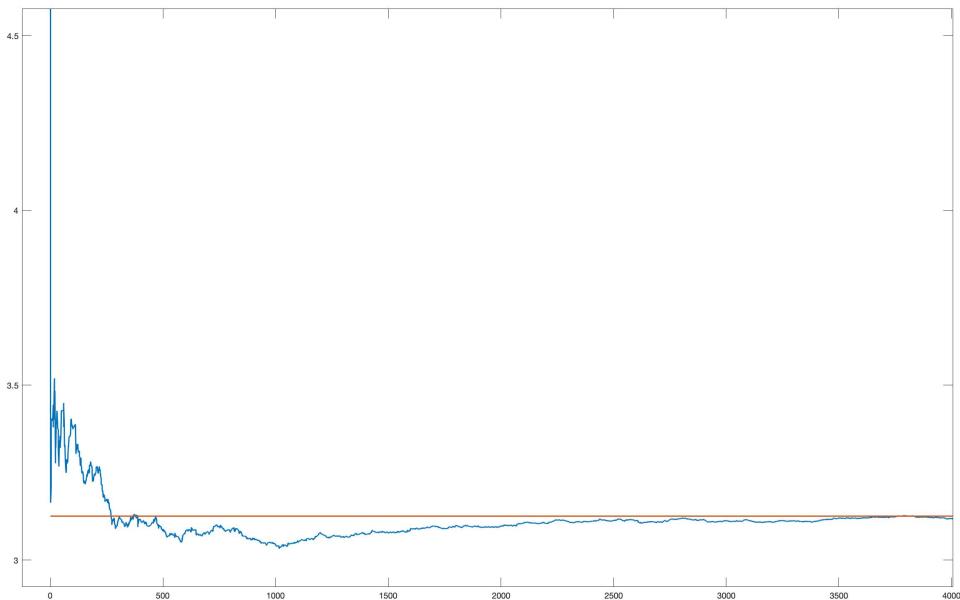
Sono state analizzate alcune statistiche per studiare la loro evoluzione durante la simulazione. Di seguito vengono riportati i grafici dei tempi di risposta globali e per classe (asse verticale) durante il tempo di simulazione (asse orizzontale).



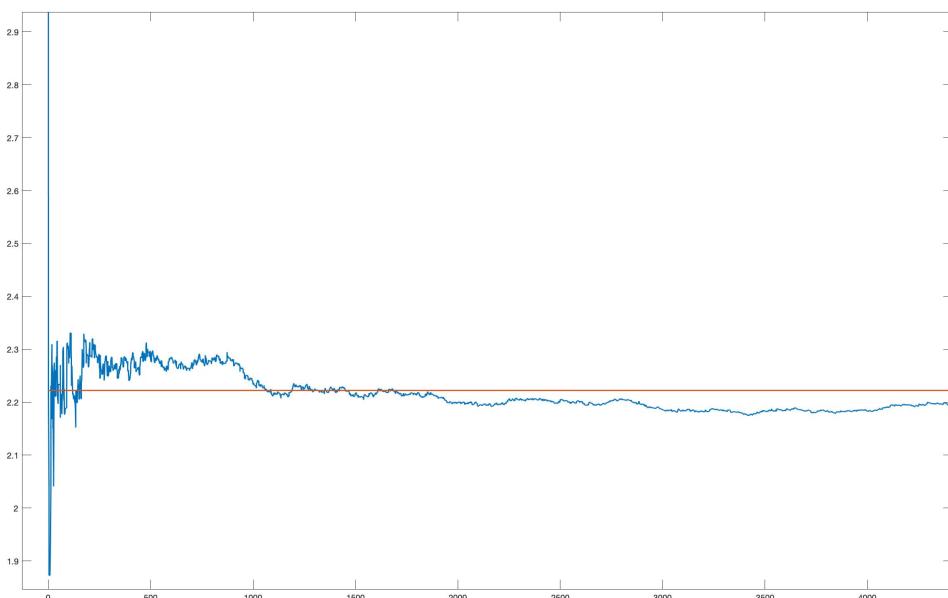
*Global System Response Time*



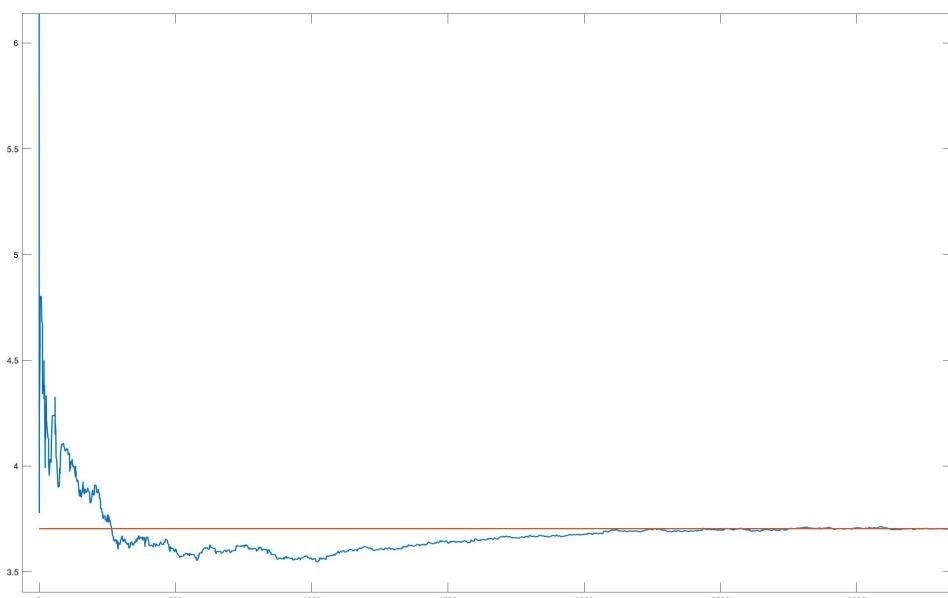
*Global System Throughput*



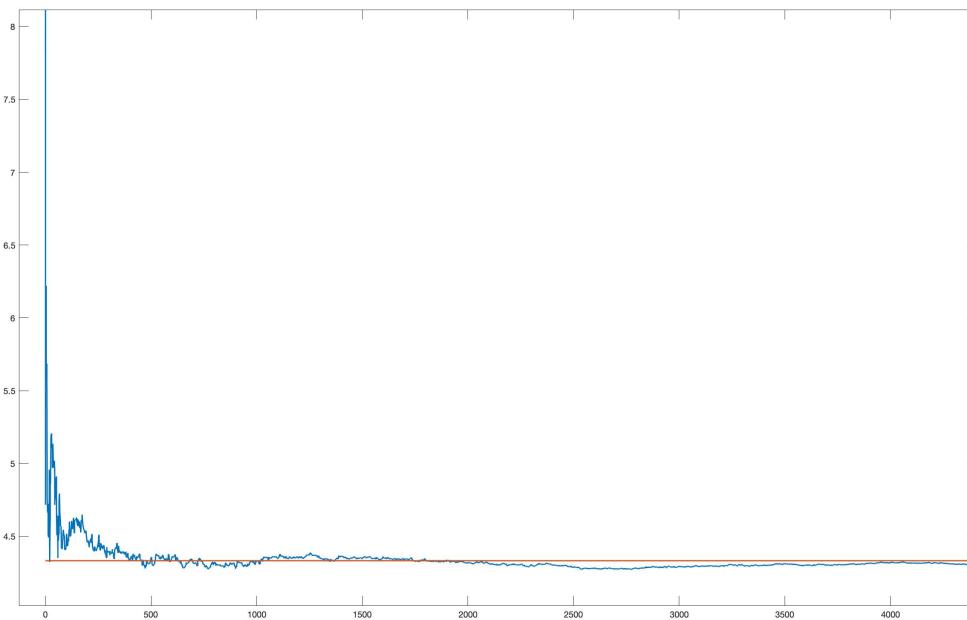
Cloudlet Response Time



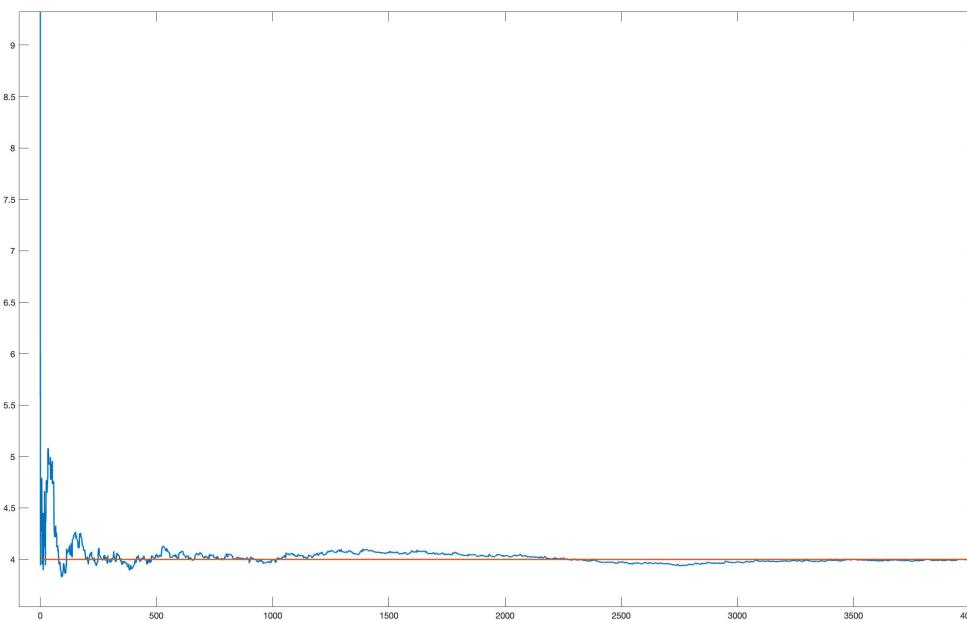
Cloudlet Response Time  
(class 1 jobs)



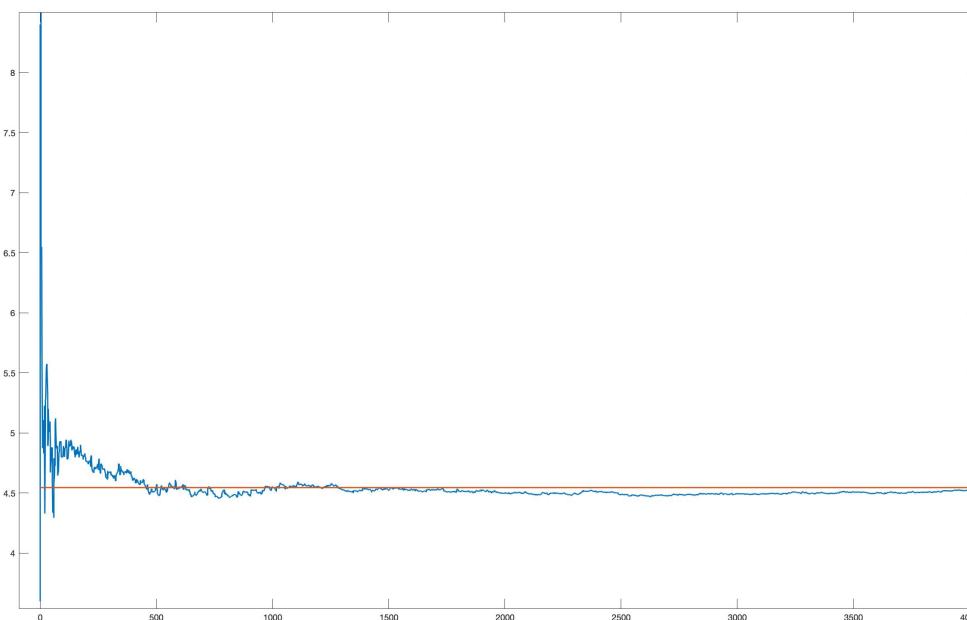
Cloudlet Response Time  
(class 2 jobs)



Cloud Response Time



Cloud Response Time  
(Class 1 jobs)



Cloud Response Time  
(Class 2 jobs)

## Statistiche ottenute

Il codice precedentemente descritto è stato testato per effettuare una simulazione “long run” per un intervallo di tempo pari a 50000 s utilizzando il seed iniziale 123456789; le statistiche ricavate sono riassunte nelle tabelle di seguito:

### **Statistiche generali**

<i>System global response time</i>	$E[t]$	3.627216
<i>System response time for class 1 jobs</i>	$E[t_1]$	2.962374
<i>System response time for class 2 jobs</i>	$E[t_2]$	4.052807
<i>System Global throughput</i>	$X$	10.223362
<i>System throughput for class 1 jobs</i>	$X_1$	3.990003
<i>System throughput for class 2 jobs</i>	$X_2$	6.233359

### **Statistiche relative al Cloudlet**

<i>Cloudlet global response time</i>	$E[t]_{cllet}$	3.126433
<i>Cloudlet response time for class 1 jobs</i>	$E[t_1]_{cllet}$	2.224573
<i>Cloudlet response time for class 2 jobs</i>	$E[t_2]_{cllet}$	3.704311
<i>Cloudlet Global throughput</i>	$X_{cllet}$	6.095100
<i>Cloudlet throughput for class 1 jobs</i>	$X_{cllet1}$	2.403797
<i>Cloudlet throughput for class 2 jobs</i>	$X_{cllet2}$	3.691303
<i>Effective class 1 jobs Cloudlet throughput</i>	$rate_{cllet1}$	2.34182
<i>Effective class 2 jobs Cloudlet throughput</i>	$rate_{cllet2}$	3.6607
<i>Mean (total) population on Cloudlet</i>	$E[N]_{cllet}$	18.775070
<i>Mean class 1 jobs population on Cloudlet</i>	$E[N_1]_{cllet}$	5.505145
<i>Mean class 2 jobs population on Cloudlet</i>	$E[N_2]_{cllet}$	13.269925

### Statistiche relative al Cloud

<i>Cloud global response time</i>	$E[t]_{cloud}$	4.335081
<i>Cloud response time for class 1 jobs</i>	$E[t_1]_{cloud}$	4.006651
<i>Cloud response time for class 2 jobs</i>	$E[t_2]_{cloud}$	4.545021
<i>Cloud Global throughput</i>	$X_{cloud}$	4.170596
<i>Cloud throughput for class 1 jobs</i>	$X_{cloud1}$	1.641389
<i>Cloud throughput for class 2 jobs</i>	$X_{cloud2}$	2.529208
<i>Mean (total) population on Cloud</i>	$E[N]_{cloud}$	18.577144
<i>Mean class 1 jobs population on Cloud</i>	$E[N_1]_{cloud}$	6.907517
<i>Mean class 2 jobs population on Cloud</i>	$E[N_2]_{cloud}$	11.671719

Come verifica dei risultati ottenuti la popolazione media simulata è stata confrontata con la popolazione media ottenuta come risultato dell'applicazione della legge di Little sui valori simulati del flusso in ingresso e del tempo medio di risposta:

$$E[N]_{cllet} = 18.775070$$

$$\lambda_{cllet} \cdot E[t]_{cllet} = 19.055922$$

$$E[N]_{cloud} = 18.577144$$

$$\lambda_{cloud} \cdot E[t]_{cloud} = 18.079871$$

$$E[N_1]_{cllet} = 5.505145$$

$$\lambda_{cllet1} \cdot E[t_1]_{cllet} = 5.347422$$

$$E[N_1]_{cloud} = 6.907517$$

$$\lambda_{cloud1} \cdot E[t_1]_{cloud} = 6.576473$$

$$E[N_2]_{cllet} = 13.269925$$

$$\lambda_{cllet2} \cdot E[t_2]_{cllet} = 13.673734$$

$$E[N_2]_{cloud} = 11.671719$$

$$\lambda_{cloud2} \cdot E[t_2]_{cloud} = 11.495303$$

Dai risultati ottenuti si può notare che i valori simulati della popolazione media si discostano leggermente da quelli ottenuti tramite la legge di Little, alcune volte per difetto altre volte per eccesso. Nel complesso, tuttavia, i valori simulati in entrambi i casi possono considerarsi in linea con i valori teorici, considerando un certo margine di errore.

Nella tabella seguente vengono riportate tutte le statistiche di interesse confrontate con i valori teorici determinati precedentemente. Per ogni statistica si riporta il relativo intervallo di confidenza al 95% calcolato con il metodo dei Batch Means (è stata effettuata una simulazione “long run” di 50000s e i dati raccolti sono stati suddivisi in 64 batches):

<b>Seed = 123456789</b>	<b>Valori Teorici</b>	<b>Valori Simulati</b>	<b>Intervallo di Confidenza</b>
$E[t]$	3.625938	3.627216	[ 3.626210 , 3.628222 ]
$E[t_1]$	2.959198	2.962374	[ 2.960904 , 2.963844 ]
$E[t_2]$	4.052651	4.052807	[ 4.051593 , 4.054020 ]
$X$	10.25	10.223362	[ 10.220294 , 10.226429 ]
$X_1$	4	3.990003	[ 3.988137 , 3.991870 ]
$X_2$	6.25	6.233359	[ 6.231200 , 6.235517 ]
$E[t]_{clct}$	3.125565	3.126433	[ 3.125284 , 3.127582 ]
$E[t_1]_{clct}$	2.222222	2.224573	[ 2.223602 , 2.225544 ]
$E[t_2]_{clct}$	3.703704	3.704311	[ 3.702672 , 3.705951 ]
$X_{clct}$	6.000873	6.095100	[ 6.085800 , 6.104401 ]
$X_{clct1}$	2.341805	2.403797	[ 2.392604 , 2.414989 ]
$X_{clct2}$	3.659068	3.691303	[ 3.689093 , 3.693514 ]
$E[N]_{clct}$	18.756119	18.775070	[ 18.769809 , 18.780331 ]
$E[N_1]_{clct}$	5.20401	5.505145	[ 5.486518 , 5.523772 ]
$E[N_2]_{clct}$	13.552106	13.269925	[ 13.246163 , 13.293686 ]
$E[t]_{cloud}$	4.332594	4.335081	[ 4.333391 , 4.336772 ]
$E[t_1]_{cloud}$	4	4.006651	[ 4.004299 , 4.009004 ]
$E[t_2]_{cloud}$	4.545455	4.545021	[ 4.542923 , 4.547119 ]
$X_{cloud}$	4.249127	4.170596	[ 4.150387 , 4.190806 ]
$X_{cloud1}$	1.658196	1.641389	[ 1.620060 , 1.662717 ]
$X_{cloud2}$	2.590931	2.529208	[ 2.527362 , 2.531053 ]
$E[N]_{cloud}$	18.409742	18.577144	[ 18.486292 , 18.667995 ]
$E[N_1]_{cloud}$	6.632785	6.907517	[ 6.803325 , 7.011708 ]
$E[N_2]_{cloud}$	11.776959	11.671719	[ 11.658438 , 11.685001 ]

# *Algoritmo 2*

## Goal e Obiettivi

I goal e gli obiettivi prevedono la realizzazione di un simulatore di tipo next-event per il sistema in cui il controllo degli accessi è gestito secondo l'algoritmo seguente:

### **Algorithm 2**

o class 1 arrival:

```
if  $n_1 = N \rightarrow$  send on the Cloud  
else if  $n_1 + n_2 < S \rightarrow$  accept  
else if  $n_2 > 0 \rightarrow$  accept the task on the Cloudlet and send a class 2 task on the Cloud  
else accept the task on the Cloudlet
```

o class 2 arrival:

```
if  $n_1 + n_2 \geq S \rightarrow$  send on the Cloud  
else accept the task on the Cloudlet
```

Si richiede di determinare se il sistema è stazionario o meno, di valutare il tempo medio di risposta, il throughput, la popolazione media globali e per classe, per il Cloudlet e per il Cloud. Si richiede inoltre di analizzare le statistiche del sistema transiente, ed eventualmente di quello steady-state. E' necessario poi definire un modello di code per descrivere il sistema. Le statistiche ottenute dalle simulazioni devono essere adeguatamente verificate e validate.

## Modello Concettuale

La logica di controllo degli arrivi per questo secondo algoritmo tiene traccia della classe del job in arrivo. L'algoritmo può essere interpretato come segue. Per arrivi di tipo 1:

- Nel caso in cui il Cloudlet disponga di server in stato ‘idle’, il job viene accettato sul Cloudlet

- Nel caso in cui tutti i server del Cloudlet siano ‘busy’, ma vi siano dei job di classe 2 in esecuzione, viene scelto ed interrotto un job di classe 2; il job di classe 1 eseguirà al posto di quello interrotto, mentre quest’ultimo verrà inviato al Cloud
- Nel caso in cui tutti i server del Cloudlet siano busy, e tutti stiano processando job di classe 1, il nuovo arrivo verrà inoltrato al Cloud.

Per gli arrivi di tipo 2:

- Se il Cloudlet dispone di server in stato ‘idle’, il job viene accettato e processato sul Cloudlet
- Se tutti i server del Cloudlet sono ‘busy’ (indipendentemente dalla classe dei job che sono in servizio), il job viene inviato al Cloud

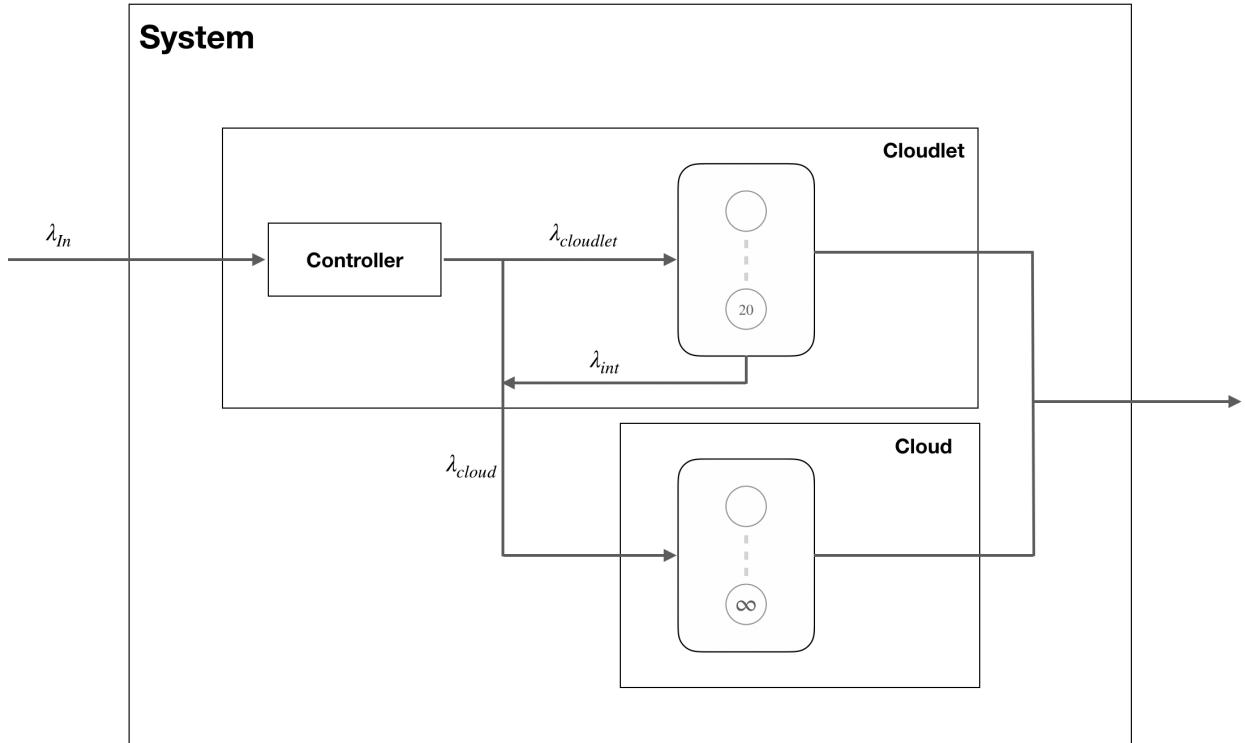
Per caratterizzare lo stato del sistema ad ogni istante di tempo sono state identificate le seguenti variabili di stato:

- $l_1(t)_{clet}$  : numero di job di classe 1 presenti nel Cloudlet al tempo t
- $l_2(t)_{clet}$  : numero di job di classe 2 presenti nel Cloudlet al tempo t
- $x_1(t)_{clet}, x_2(t)_{clet}, \dots, x_{20}(t)_{clet}$  : stato dei server (0 o 1) nel Cloudlet al tempo t
- $x_1(t)_{cloud}, x_2(t)_{cloud}, x_3(t)_{cloud} \dots$  : stato dei server (0 o 1) nel Cloud al tempo t

Il modello che caratterizza il sistema in esame è mostrato nella figura di seguito. Il flusso in ingresso è stato rappresentato con la variabile  $\lambda_{In}$  e rappresenta il flusso totale che arriva al sistema e che viene gestito dal controller; esso comprende il flusso di job di classe 1 e il flusso di job di classe 2.

Con la variabile  $\lambda_{cloudlet}$  è stato rappresentato il flusso totale in ingresso al Cloudlet, mentre con la variabile  $\lambda_{cloud}$  il flusso totale in ingresso al Cloud. Quest’ultimo comprende sia il flusso in ingresso al Cloud proveniente dal controller sia il flusso dei job provenienti dal Cloudlet, ovvero quelli di classe 2 che erano in esecuzione nel Cloudlet e che sono stati interrotti per permettere l’esecuzione di job di classe 1.

Il flusso dal Cloudlet al Cloud di tali job interrotti è stato rappresentato dalla variabile  $\lambda_{int}$ .



## Modello di Specifica

Come nel caso dell'algoritmo 1, gli ingressi nel sistema, rappresentati con la variabile  $\lambda_{In}$ , costituiscono la quantità totale dei task che i dispositivi mobili richiedono di gestire a server esterni. Anche in questo caso è possibile considerare il flusso in ingresso come un flusso generato da eventi aleatori. Di conseguenza gli arrivi al sistema possono essere modellati con distribuzioni di Poisson di parametri  $\lambda_1$  e  $\lambda_2$  per job di classe 1 e 2 rispettivamente.

I flussi in ingresso nel server Cloudlet e nel server Cloud sono stati indicati, come nel caso precedente, con le variabili  $\lambda_{cloudlet}$  e  $\lambda_{cloud}$ . La variabile  $\lambda_{cloudlet}$  indica la totalità dei flussi in ingresso al Cloudlet, comprendente i job di tipo 1 e quelli di tipo 2 ( $\lambda_{cloudlet} = \lambda_{cl1} + \lambda_{cl2}$ ). I job di tipo 1 vengono accettati dal Cloudlet quando ci sono server liberi e quando tutti i server sono occupati ma ce ne sono alcuni che stanno processando job di tipo 2. Quindi, considerando che l'unico caso in cui il Cloudlet non può gestire job di tipo 1 è quando tutti i server sono impegnati nell'elaborazione di job di tipo 1, il flusso in ingresso per i job di tipo 1 può essere calcolato come segue:

$$\lambda_{clet1} = \lambda_1 \cdot (1 - P(n_1 = N))$$

Dove con  $P(n_1 = N)$  si rappresenta la probabilità che tutti i server del Cloudlet siano occupati e che tutti i job in esecuzione nel Cloudlet siano di tipo 1.

I job di tipo 2 vengono accettati dal Cloudlet solo quando vi sono dei server del Cloudlet in stato ‘idle’. Quindi, il flusso in ingresso al Cloudlet di tipo 2 viene calcolato come:

$$\lambda_{clet2} = \lambda_2 \cdot (1 - \pi_{20})$$

Dove  $\pi_{20}$  rappresenta la probabilità che tutti i server del Cloudlet siano in stato ‘busy’, indipendentemente dal tipo di job che sono in servizio.

Per quanto riguarda il Cloud, anche in questo caso la variabile  $\lambda_{cloud}$  rappresenta la totalità dei flussi in ingresso al Cloud, comprendente quelli di tipo 1 e 2, provenienti dal controller e dal Cloudlet (ovvero i job interrotti).

I job di tipo 1 che arrivano al Cloud sono quelli che il Cloudlet non può gestire, ovvero quelli che arrivano al controller nel momento in cui il Cloudlet ha tutti i server in stato ‘busy’ e tutti i job in servizio sono di tipo 1. Quindi il flusso in ingresso al Cloud di tipo 1 è dato da:

$$\lambda_{cloud1} = \lambda_1 \cdot P(n_1 = N)$$

Al contrario, i job di classe 2 possono arrivare al Cloud in due modi differenti: possono provenire dal controller nel caso in cui tutti i server del Cloudlet siano ‘busy’, o possono provenire dal Cloudlet quando essi vengono prelazionati per favorire l’esecuzione di quelli di classe 1. Quindi, il flusso di tipo due in ingresso al Cloud è calcolato come segue:

$$\lambda_{cloud2} = \lambda_2 \cdot \pi_{20} + \lambda_{clet2} \cdot \pi_{int}$$

Dove con  $\pi_{int}$  si intende la probabilità che un job di classe due in esecuzione sul Cloudlet venga interrotto. Tale evento si verifica nel caso in cui sia in arrivo un job di classe 1, e che nel Cloudlet tutti i server siano occupati ( $n_1 + n_2 = N$ ) e che vi siano job di tipo 2 in esecuzione ( $n_2 > 0$ ). Quindi, tale probabilità viene calcolata nel seguente modo:

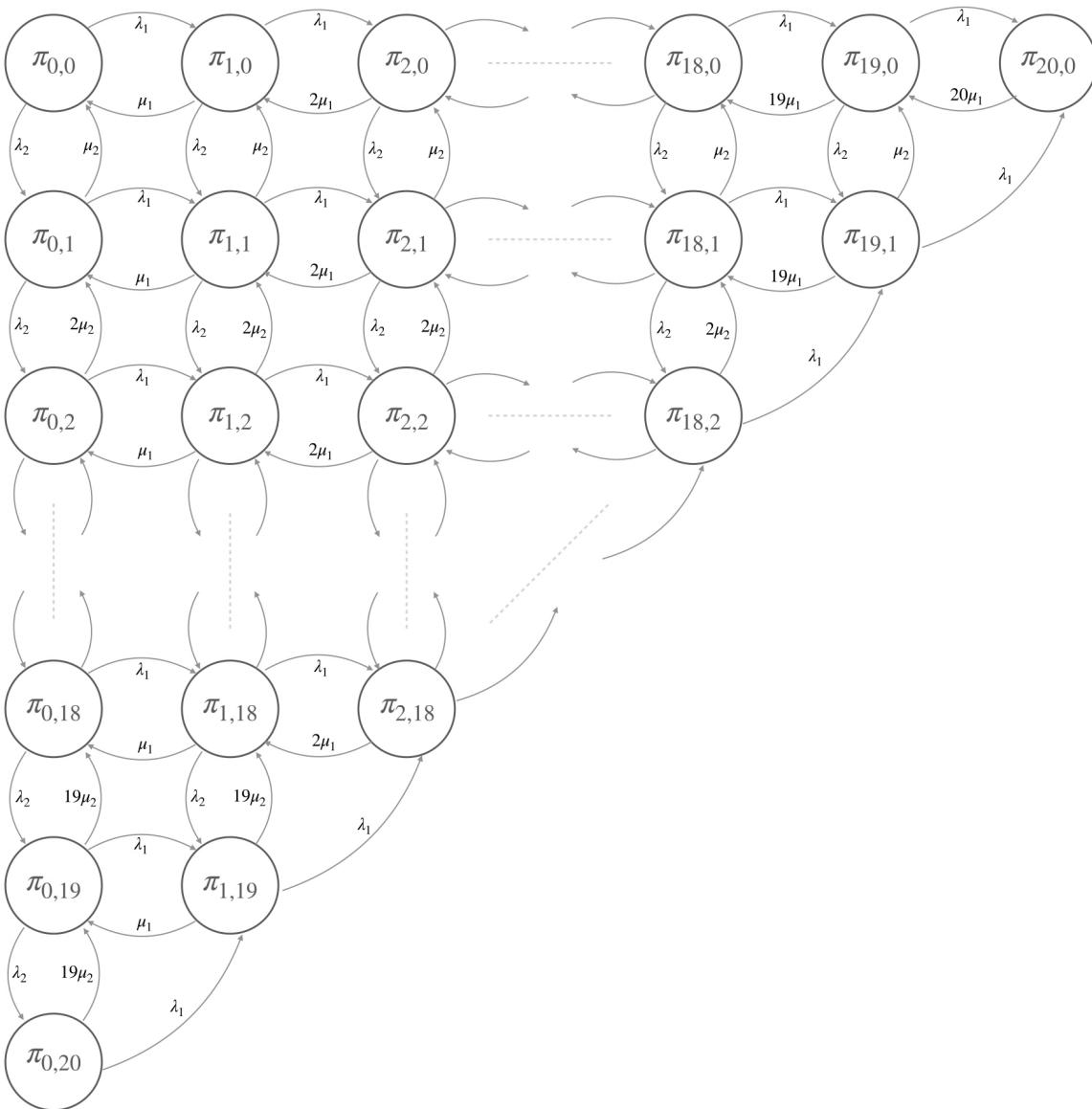
$$\pi_{int} = P(\{n_1 + n_2 = N\} \cap \{n_2 > 0\}) = P(\{n_1 + n_2 = N\} | \{n_2 > 0\}) * P(\{n_2 > 0\})$$

Per le successive analisi si ipotizza che sia gli arrivi al Cloudlet che gli arrivi al Cloud siano di Poisson, e che quindi i rispettivi tempi di interarrivo siano esponenziali. Assumendo che i tempi di arrivo siano esponenziali, sapendo che i tempi di servizio sono anch'essi esponenziali, che il Cloudlet possiede 20 risorse, e che il Cloud ha un numero illimitato di risorse, è possibile quindi definire un modello di code per il Cloudlet e per il Cloud:

- Cloudlet :  $M/M/20/20$
- Cloud :  $M/M/\infty$

Quindi, il Cloudlet è rappresentato come un multi-server con 20 serventi e capacità del sistema pari a 20, mentre il Cloud è rappresentato come un multi-server con  $\infty$  serventi e capacità infinita.

Nella figura seguente è stato rappresentato il modello di code relativo al Cloudlet:



Risolvendo la catena di Markov con Matlab, è stato possibile ottenere i valori delle probabilità di stato. Per calcolare la probabilità che un job di classe 2 sul Cloudlet venga interrotto sono state considerate le seguenti probabilità:

$$P(\{n_1 + n_2 = N\} | \{n_2 > 0\}) = P(\{n_1 + n_2 = 20\} | \{n_2 = 1\}) + \dots + P(\{n_1 + n_2 = 20\} | \{n_2 = 20\}) = \\ P(\{n_1 = 19\}, \{n_2 = 1\}) + \dots + P(\{n_1 = 0\}, \{n_2 = 20\}) = \sum_{i=0}^{19} \pi_{i,20-i} = 0.306091 \\ P(\{n_2 > 0\}) = \sum_{i=0}^{19} \sum_{j=1}^{20-i} \pi_{i,j} = 1 - \sum_{i=0}^{20} \pi_{i,0} = 0.995968$$

Da tali valori si ricava la probabilità che un job di tipo due venga interrotto nel Cloudlet:

$$\pi_{int} = P(\{n_1 + n_2 = N\} | \{n_2 > 0\}) \cdot P(\{n_2 > 0\}) = 0.304857$$

Con i dati a disposizione, è possibile determinare i valori dei flussi in ingresso al Cloudlet e al Cloud (si considera  $P(n_1 = N) = \pi_{20,0}$ ):

$$\lambda_{clet1} = \lambda_1 \cdot (1 - \pi_{20,0}) = 3.996355$$

$$\lambda_{clet2} = \lambda_2 \cdot (1 - \pi_{20,0}) = 4.331238$$

$$\lambda_{cloud1} = \lambda_1 \cdot \pi_{20,0} = 0.003645$$

$$\lambda_{cloud2} = \lambda_2 \cdot \pi_{20,0} + \lambda_{clet2} \cdot \pi_{int} = 1.918763 + 1.320408 = 3.239171$$

Il flusso totale in ingresso al Cloudlet è:

$$\lambda_{clet} = \lambda_{clet1} + \lambda_{clet2} = 8.327593$$

Mentre il flusso totale in ingresso al Cloud è:

$$\lambda_{cloud} = \lambda_{cloud1} + \lambda_{cloud2} = 3.242816$$

In particolare, il flusso in ingresso al Cloud proveniente solo dal Controller (escludendo quindi il flusso dei job interrotti) è:

$$\lambda'_{cloud} = \lambda_{cloud1} + \lambda'_{cloud2} = 1.922408$$

Il flusso di job di classe 2 in ingresso al Cloudlet e che verranno completati nel Cloudlet è:

$$\lambda_{clet2eff} = \lambda_{clet2} \cdot (1 - \pi_{int}) = 3.010829$$

Il tempo medio di servizio di Cloudlet e Cloud, per job di classe 1 e di classe 2 è lo stesso che era stato calcolato nel caso dell'algoritmo 1. Per il calcolo del tempo medio di risposta complessivo di Cloudlet e Cloud e dell'intero sistema occorre tener conto del tempo speso a causa dei job interrotti:

$$E[t]_{clet} = \frac{\lambda_{clet1}}{\lambda_{clet}} \cdot E[S_1]_{clet} + \frac{\lambda_{clet2eff}}{\lambda_{clet}} \cdot E[S_2]_{clet} + \frac{\lambda_{int}}{\lambda_{clet}} \cdot \alpha \cdot E[S_2]_{clet} = 2.405498 + \alpha \cdot 0.587253$$

$$E[t]_{cloud} = \frac{\lambda_{cloud1}}{\lambda_{cloud}} \cdot E[S_1]_{cloud} + \frac{\lambda'_{cloud2}}{\lambda_{cloud}} \cdot E[S_2]_{cloud} + \frac{\lambda_{int}}{\lambda_{cloud}} \cdot \left( E[S_2]_{cloud} + E[S_{setup}] \right) = 4.870584$$

Il parametro  $\alpha$  è stato utilizzato in quanto non si conosce la frazione di tempo per cui un job di tipo due è in servizio sul Cloudlet prima di essere interrotto.

I tempi di risposta per classe relativi a Cloudlet e Cloud sono stati calcolati come segue:

$$E[t_1]_{clet} = E[S_1]_{clet} = 2.222222 \quad E[t_1]_{cloud} = E[S_1]_{cloud} = 4$$

$$E[t_2]_{clet} = \frac{\lambda_{clet2eff}}{\lambda_{clet2}} \cdot E[S_2]_{clet} + \frac{\lambda_{int}}{\lambda_{clet2}} \cdot \alpha \cdot E[S_2]_{clet} = 2.574603 + \alpha \cdot 1.129101$$

$$E[t_2]_{cloud} = \frac{\lambda'_{cloud2}}{\lambda_{cloud2}} \cdot E[S_2]_{cloud} + \frac{\lambda_{int}}{\lambda_{cloud2}} \cdot \left( E[S_2]_{cloud} + E[S_{setup}] \right) = 4.871565$$

Con un procedimento analogo sono stati calcolati i tempi di risposta relativi all'intero sistema, ovvero il tempo di risposta globale e i tempi di risposta per classi:

$$E[t_1] = \frac{\lambda_{clet1}}{\lambda_1} \cdot E[S_1]_{clet} + \frac{\lambda_{cloud1}}{\lambda_1} \cdot E[S_1]_{cloud} = 2.223842$$

$$E[t_2] = \frac{\lambda_{clet2eff}}{\lambda_2} \cdot E[S_2]_{clet} + \frac{\lambda'_{cloud2}}{\lambda_2} \cdot E[S_2]_{cloud} + \frac{\lambda_{int}}{\lambda_2} \cdot \left( E[S_2]_{cloud} + E[S_{setup}] + \alpha \cdot E[S_2]_{clet} \right) = 4.308967 + \alpha \cdot 0.782463$$

$$\begin{aligned}
E[t]_{system} = & \frac{\lambda_{clet1}}{\lambda_{tot}} \cdot E[S_1]_{clet} + \frac{\lambda_{cloud1}}{\lambda_{tot}} \cdot E[S_1]_{cloud} + \\
& \frac{\lambda_{clet2eff}}{\lambda_{tot}} \cdot E[S_2]_{clet} + \frac{\lambda'_{cloud2}}{\lambda_{tot}} \cdot E[S_2]_{cloud} + \\
& \frac{\lambda_{int}}{\lambda_{tot}} \cdot \left( E[S_2]_{cloud} + E[S_{setup}] + \alpha \cdot E[S_2]_{clet} \right) = 3.495259 + \alpha \cdot 0.477111
\end{aligned}$$

Applicando la legge di Little è stato possibile ricavare i valori teorici riguardanti la popolazione media. Di seguito verranno calcolate la popolazione media nel Cloudlet e nel Cloud, totale e relativa a ciascuna classe:

$$E[N]_{clet} = 16.855767 + \alpha \cdot 4.114989$$

$$E[N]_{cloud} = 15.794413$$

$$E[N_1]_{clet} = 8.880788$$

$$E[N_1]_{cloud} = 0.014580$$

$$E[N_2]_{clet} = 7.751689 + \alpha \cdot 3.399530$$

$$E[N_2]_{cloud} = 15.779837$$

Il valore della popolazione media nel Cloudlet è stato calcolato anche a partire dai valori delle probabilità di stato della catena di Markov del Cloudlet; il valore trovato è il seguente:

$$E[N]_{clet} = \sum_{i+j=1}^{20} \pi_{i,j} \cdot (i + j) = 17.821449$$

Per il calcolo dei valori teorici del throughput si considera la seguente relazione:

$$X = \min(\lambda, \mu)$$

Il sistema globale risulta essere stazionario, in quanto il server Cloud possiede una quantità infinita di risorse, e ciò fa sì che il sistema sia in grado di gestire tutto il flusso in ingresso. Di conseguenza, il throughput è pari alla quantità di flusso in ingresso:

$$X = \lambda_{In} = 10.25$$

$$X_1 = \lambda_1 = 4$$

$$X_2 = \lambda_2 = 6.25$$

$$X_{clet} = \lambda_{clet1} + \lambda_{clet2eff} = 7.007184$$

$$X_{cloud} = \lambda_{cloud1} + \lambda_{cloud2} = 3.242817$$

$$X_{clet1} = \lambda_{clet1} = 3.996355$$

$$X_{clet2} = \lambda_{clet2eff} = 3.010829$$

$$X_{cloud1} = \lambda_{cloud1} = 0.003645$$

$$X_{cloud2} = \lambda_{cloud2} = 3.239172$$

Per quanto riguarda le statistiche relative ai job interrotti, sapendo che  $\pi_{int} = 0.304857$ , è possibile affermare che circa il 30.4857% dei job di classe 2 in arrivo al Cloudlet viene interrotto. Per calcolare il tempo di risposta di tale job si è ragionato nel seguente modo:

$$E[t_{int}] = \alpha \cdot E[S_2]_{clet} + E[S_{setup}] + E[S_2]_{cloud} = \alpha \cdot 3.703704 + 5.345455$$

(si ipotizza che quando un job di classe 2 viene interrotto, esso debba riprendere la propria esecuzione da zero sul Cloud; quindi il tempo totale di risposta di un job interrotto tiene conto del tempo speso, ovvero “perso”, sul Cloudlet, del tempo di Setup necessario per la ripresa del servizio sul Cloud e del tempo di servizio sul Cloud).

## Modello Computazionale

Il codice che realizza il simulatore è stato implementato in Java. La struttura del codice è simile a quella presentata per l'algoritmo 1, ma in questo caso cambia la logica di gestione degli arrivi. Come nel caso precedente, si utilizza il metodo '*nextEvent*' per selezionare l'evento più imminente (che può essere un arrivo al controller, una partenza dal Cloudlet o una partenza dal Cloud). Nel caso in cui l'evento sia un arrivo, vengono simulati il tempo di arrivo di un job di tipo uno e il tempo di arrivo di un job di tipo due: l'evento che sarà preso in gestione sarà quello con il cui tempo di next-event è minore.

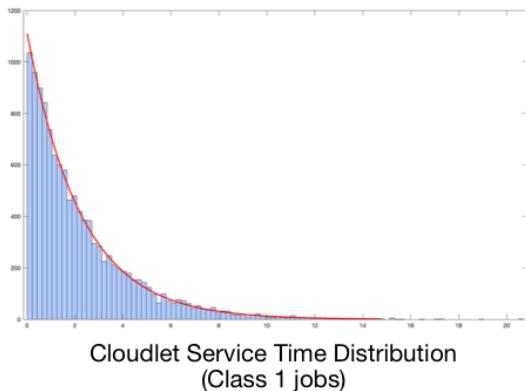
Dopo aver stabilito la tipologia di arrivo, viene fatto un controllo sul valore  $n_1 + n_2$ : se tale valore è minore di N, indipendentemente dalla classe del job, esso si accetta sul Cloudlet. Altrimenti, se il job è di classe 2 oppure se il job è di classe 1 ma si ha  $n_1 = N$ , il nuovo arrivo viene mandato al Cloud.

Altrimenti, se il nuovo arrivo è di classe 1, tutti i server sono occupati ma vi sono in esecuzione job di classe 2, uno di tali job viene interrotto e mandato sul Cloud. Si tiene traccia del tempo speso nel Cloudlet dal job interrotto, così da poter determinare il tempo totale di permanenza di tale job nel sistema (ottenuto come somma del tempo speso nel Cloudlet, del tempo di setup, e del tempo di servizio nel Cloud).

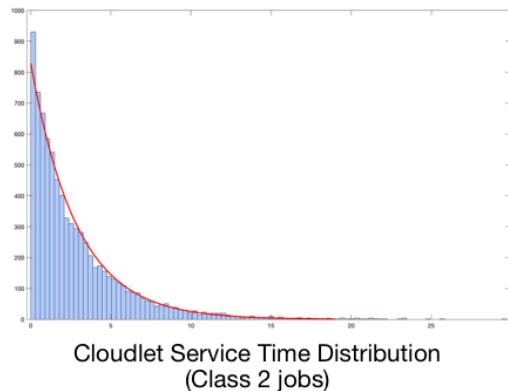
Se vi sono più job di tipo due in esecuzione nel Cloudlet, sono state sperimentate due logiche di interruzione: la scelta di uno di essi in modo random, o la scelta basata su tempo di next-event: viene interrotto il job il cui next-event time è maggiore (ovvero quello che sarebbe dovuto rimanere nel Cloudlet più a lungo). I risultati di tali sperimenti sono riportati in seguito.

# Verifica e Validazione

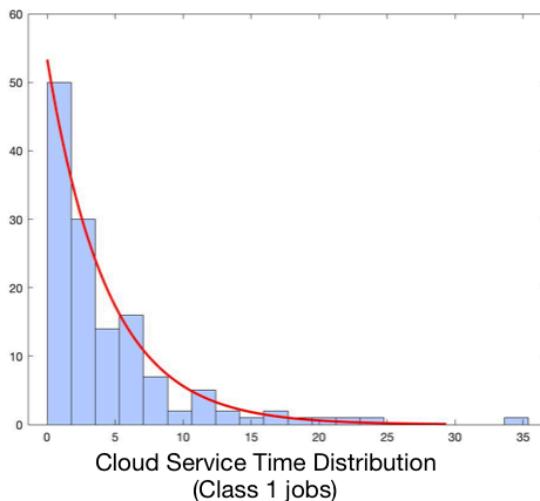
Sono state studiate le distribuzioni dei tempi di servizio dei job di classe 1 e di classe 2 nel Cloudlet e nel Cloud e confrontati con la distribuzione esponenziale; i grafici ottenuti forniscono una conferma del fatto che i tempi di servizio sono esponenziali.



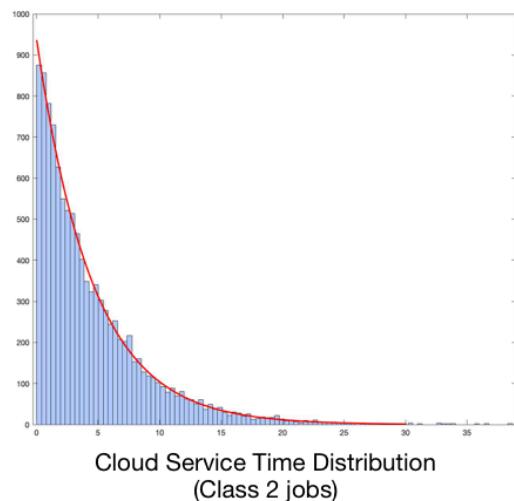
Cloudlet Service Time Distribution  
(Class 1 jobs)



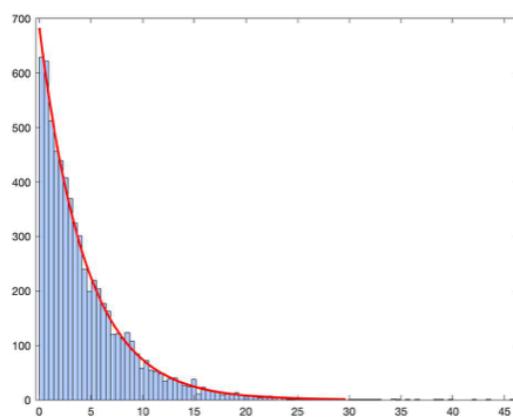
Cloudlet Service Time Distribution  
(Class 2 jobs)



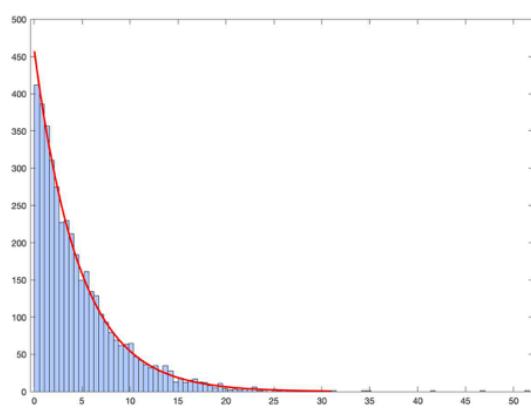
Cloud Service Time Distribution  
(Class 1 jobs)



Cloud Service Time Distribution  
(Class 2 jobs)



Cloud Service Time Distribution  
(Class 2 jobs from Controller)



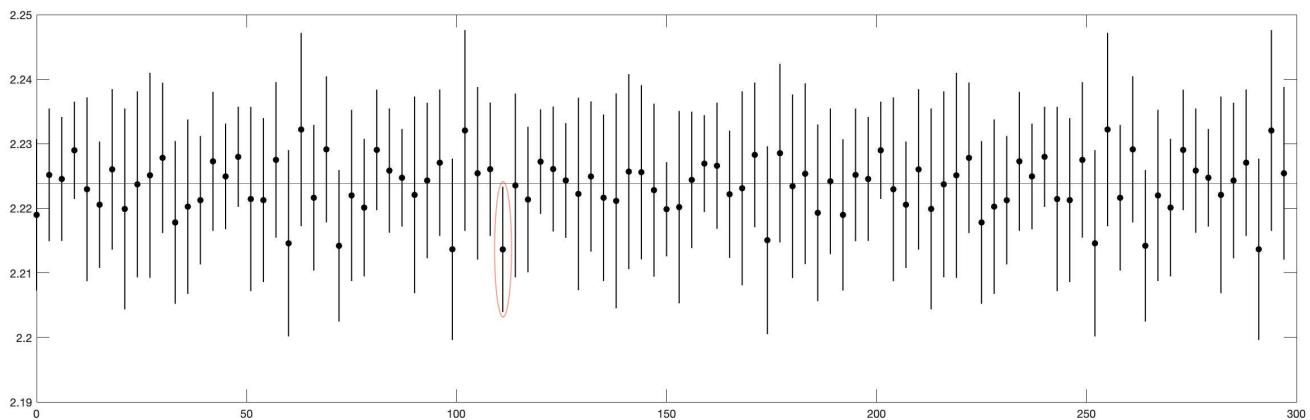
Cloud Service Time Distribution  
(Class 2 interrupted)

## Intervalli di Confidenza

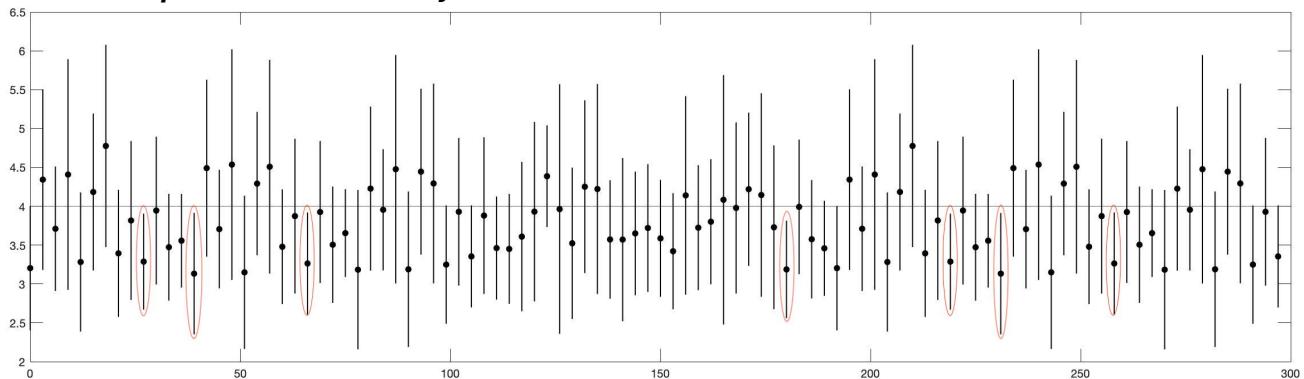
Si riportano di seguito i grafici relativi agli intervalli di confidenza riguardanti alcune delle statistiche di interesse. Ogni grafico riporta cento intervalli di confidenza; la retta orizzontale rappresenta il valore steady-state della statistica. Ogni intervallo di confidenza è stato ottenuto a partire da un campione di 10 valori, ottenuti dalla ripetizione di 10 simulazioni di 5000s ciascuna.

Si riportano di seguito i grafici relativi ai tempi medi di risposta per job di classe uno sul Cloudlet e sul Cloud:

**System Response Time Class 1 jobs**

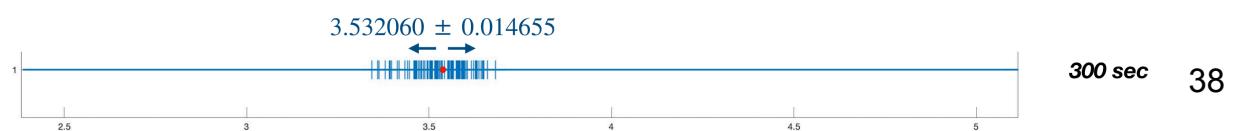
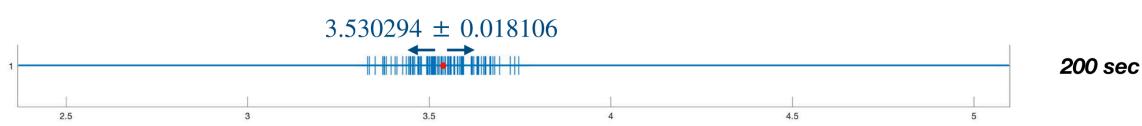
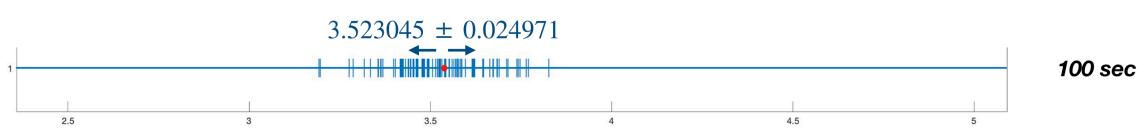
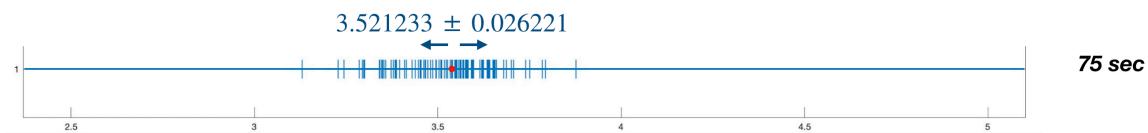
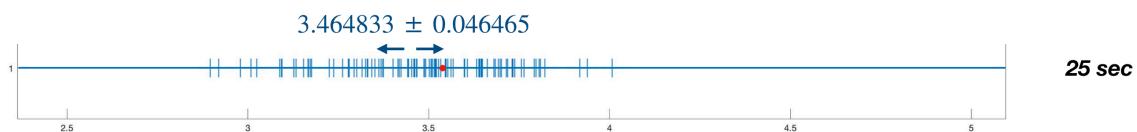
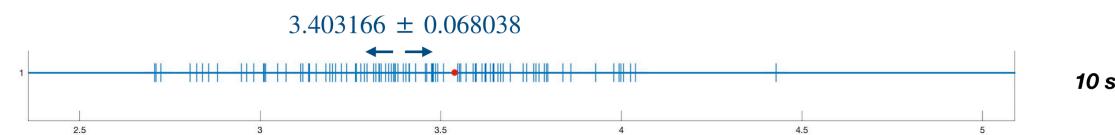


**Cloud Response Time Class 1 jobs**



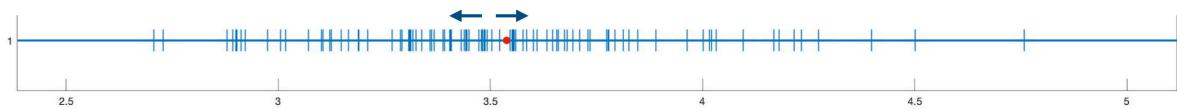
## Analisi Transiente

L'obiettivo di questo tipo di analisi è quello di definire il comportamento del sistema in un certo intervallo di tempo di simulazione in cui le condizioni iniziali e di terminazione sono fissate e il sistema al termine della simulazione ritorna allo stato iniziale. Sono riportate le statistiche relative al tempo di risposta globale del sistema ipotizzando valori diversi per il "close the door time"; sono state effettuate simulazioni con tre diversi seed iniziali. I valori simulati sono rappresentati in blu, mentre il puntino rosso rappresenta il valore steady-state del tempo medio di risposta. Il primo grafico è relativo al seed 123456789.



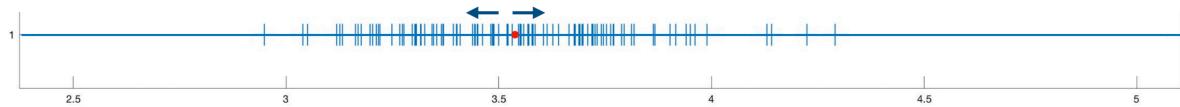
**Seed : 987654321**

$3.503459 \pm 0.081127$



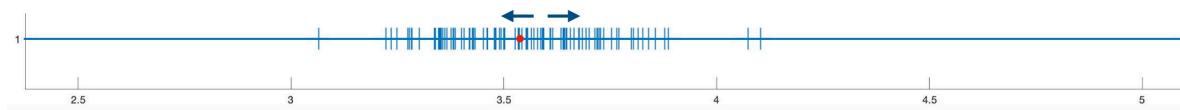
**10 sec**

$3.529085 \pm 0.054141$



**25 sec**

$3.540616 \pm 0.038178$



**50 sec**

$3.550482 \pm 0.031194$



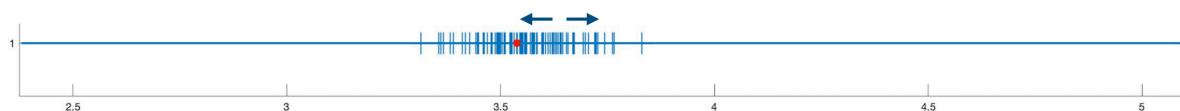
**75 sec**

$3.550504 \pm 0.029125$



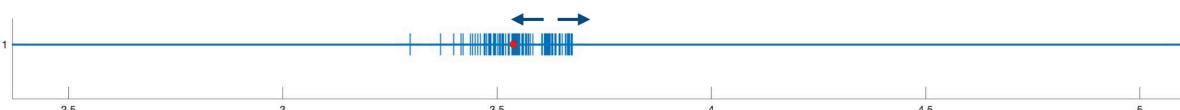
**100 sec**

$3.555364 \pm 0.019451$



**200 sec**

$3.547738 \pm 0.014679$



**300 sec**

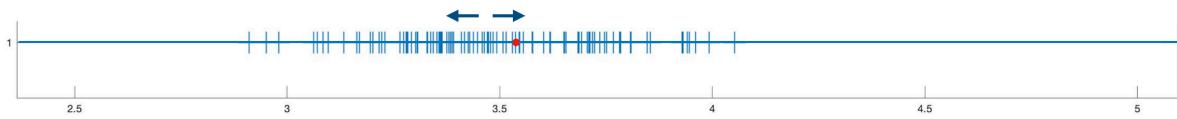
**Seed : 555555555**

$3.468040 \pm 0.073139$



**10 sec**

$3.490876 \pm 0.049909$



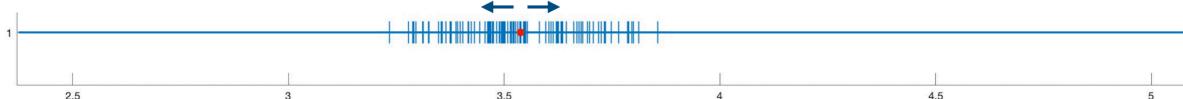
**25 sec**

$3.517026 \pm 0.035314$



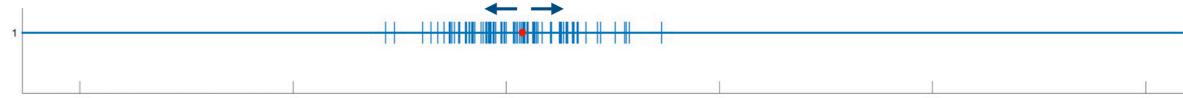
**50 sec**

$3.532157 \pm 0.027496$



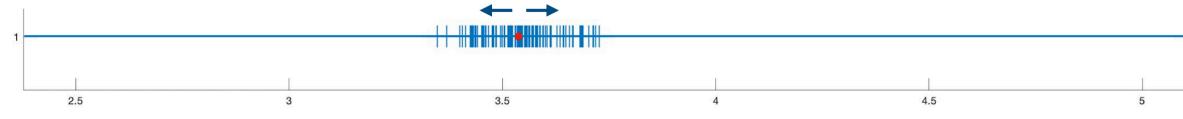
**75 sec**

$3.531567 \pm 0.024141$



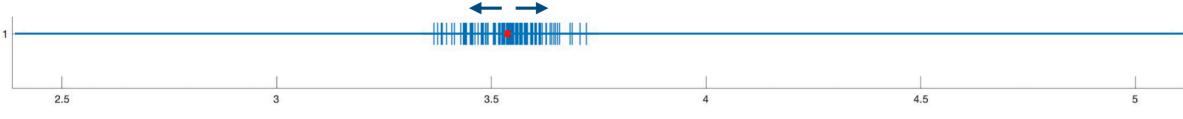
**100 sec**

$3.542879 \pm 0.016771$



**200 sec**

$3.538610 \pm 0.015004$



**300 sec**

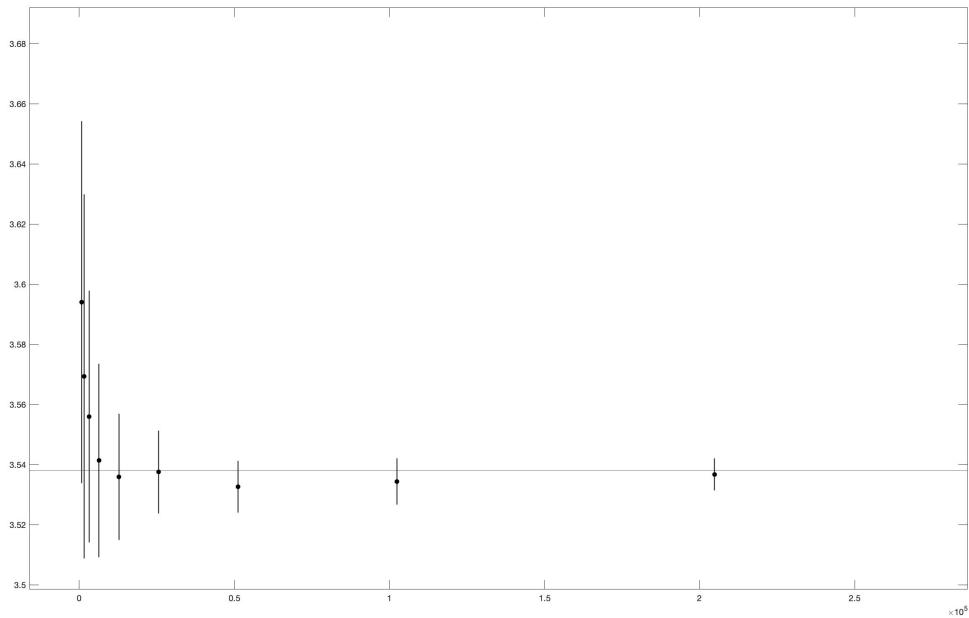
Dai grafici emerge che i valori simulati variano in base al seme scelto. In tutti e tre i casi si nota che all'aumentare del tempo di simulazione i valori di avvicinano sempre più al valore steady-state e gli intervalli di confidenza diventano più piccoli. Proprio a causa delle scelta delle condizioni iniziali (stato idle del sistema) è presente un bias iniziale. In tutti e tre i casi, a partire dai 100s l'intervallo di confidenza contiene il valore steady-state della statistica; di conseguenza, le successive valutazioni terranno conto di un tempo di simulazione di 100s. Sono state effettuate 100 simulazioni per il “close the door time” stabilito e sono state analizzate alcune delle statistiche di interesse considerando tre valori differenti del seed iniziale; per ognuna di esse è stato calcolato il relativo intervallo di confidenza al 95%.

<b>Seed = 123456789</b>	Valore simulato	Intervallo di Confidenza
$E[t]$	3.523045	[ 3.498074 , 3.548015 ]
$\lambda_{tot}$	9.198118	[ 9.103799 , 9.292437 ]
$E[N]_{clct}$	18.328547	[ 18.299937 , 18.357157 ]
$E[N]_{cloud}$	17.022218	[ 16.697775 , 17.346661 ]
$\%_{int}$	33.659832	[ 33.026365 , 34.293299 ]
$E[t_{init}]$	6.834868	[ 6.756412 , 6.913323 ]

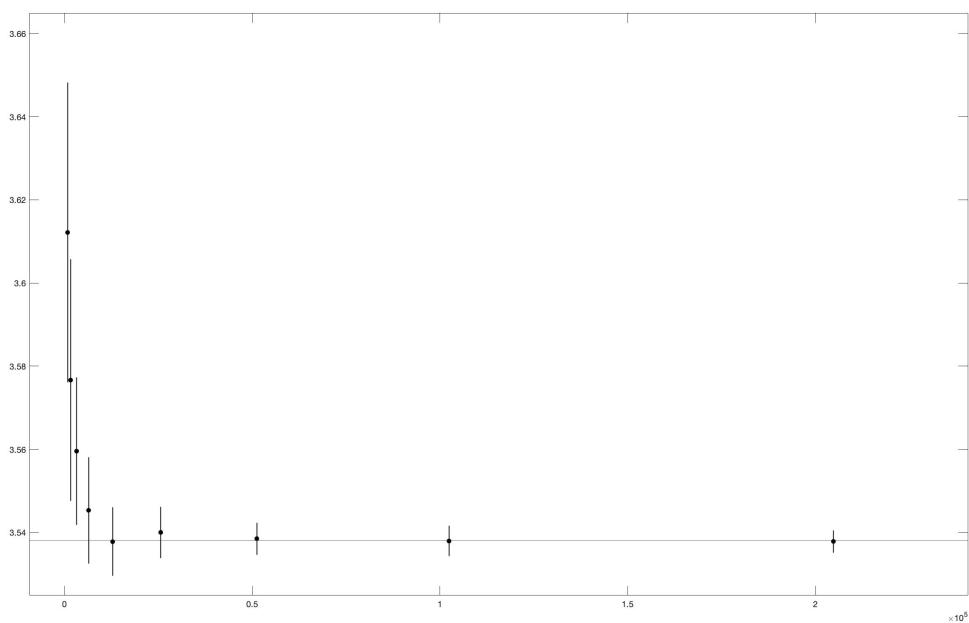
<b>Seed = 987654321</b>	Valore simulato	Intervallo di Confidenza
$E[t]$	3.550504	[ 3.521379 , 3.579629 ]
$\lambda_{tot}$	9.194155	[ 9.108211 , 9.280099 ]
$E[N]_{clct}$	18.323581	[ 18.291508 , 18.355653 ]
$E[N]_{cloud}$	17.357326	[ 17.004743 , 17.709909 ]
$\%_{int}$	34.699235	[ 33.989268 , 35.409202 ]
$E[t_{init}]$	6.951061	[ 6.868139 , 7.033983 ]

<b>Seed = 555555555</b>	Valore simulato	Intervallo di Confidenza
$E[t]$	3.531567	[ 3.507426 , 3.555708 ]
$\lambda_{tot}$	9.279121	[ 9.195245 , 9.362998 ]
$E[N]_{clct}$	18.332334	[ 18.300150 , 18.364519 ]
$E[N]_{cloud}$	17.344847	[ 17.050534 , 17.639161 ]
$\%_{int}$	34.535740	[ 33.799563 , 35.271918 ]
$E[t_{init}]$	6.944704	[ 6.859437 , 7.029971 ]

Per comprendere il comportamento del sistema nella fase transitoria è stata utilizzata la libreria RNGS per creare 16 e 64 repliche indipendenti della simulazione, utilizzando per ognuna stream diversi per i tempi di servizio e di arrivo dei job. Le statistiche analizzate riguardano il tempo medio di risposta del sistema per 800, 1600,..., 204800 job processati.

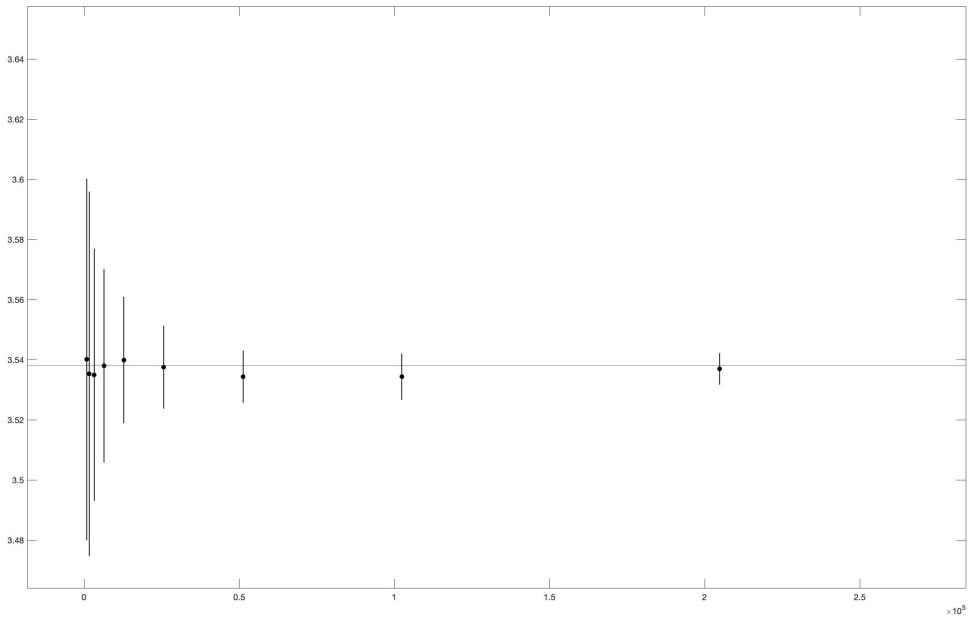


*Statistiche ottenute da 16 repliche*

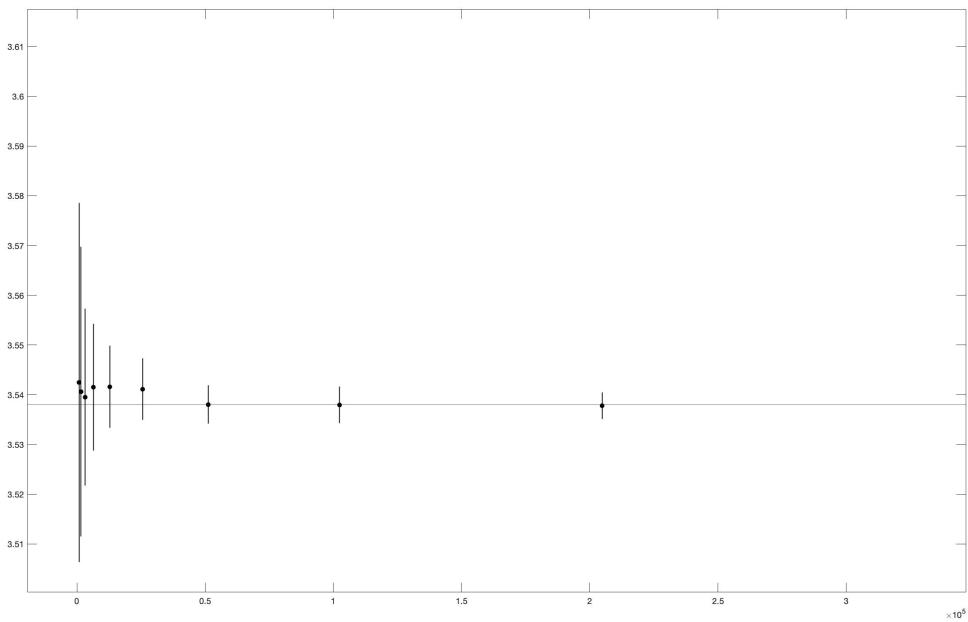


*Statistiche ottenute da 64 repliche*

Si può notare che l'utilizzo di un numero maggiore di repliche implica una diminuzione dell'ampiezza degli intervalli di confidenza. In entrambi i casi tuttavia è presente un bias iniziale, in quanto la condizione lo stato iniziale del sistema è stato considerato idle per queste simulazioni. Il Bias iniziale può essere eliminato effettuando simulazioni che partono da uno stato del sistema non idle.



*Statistiche ottenute da 16 repliche*



*Statistiche ottenute da 64 repliche*

## Analisi Steady-State

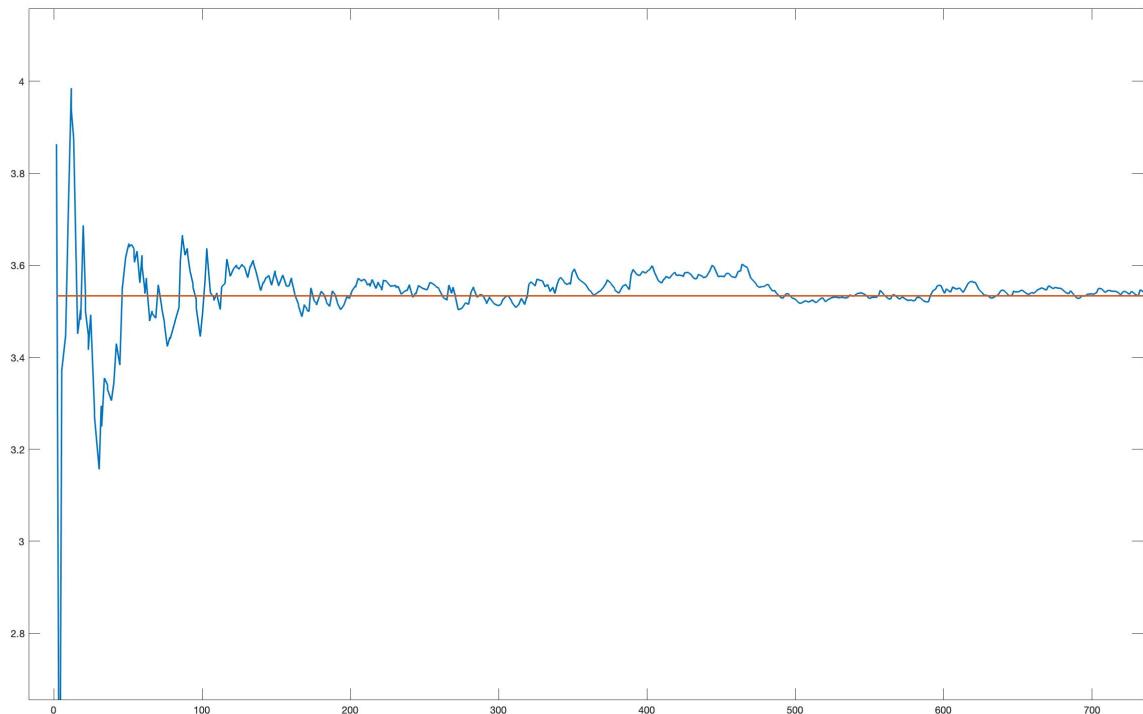
L'obiettivo di tale tipo di analisi è quello di definire il comportamento del sistema a regime. Sono state valutati alcuni indici rilevanti in diversi istanti di tempo durante la simulazione, fino al raggiungimento di una fase stazionaria (il seed iniziale utilizzato è 123456789). Per tale studio si è scelto di utilizzare una situazione iniziale in cui il sistema è vuoto.

	$E[t]$	$\lambda_{tot}$	$\lambda_1$	$\lambda_2$	$E[N]_{cler}$	$E[N]_{cloud}$
$t = 10$	2.925804	4.642409	2.388812	2.253597	15.654639	4.804124
$t = 20$	3.307753	7.097948	3.057037	4.040911	17.104712	8.636126
$t = 50$	3.525347	9.108678	3.730220	5.378457	18.229650	16.824612
$t = 100$	3.736158	9.468953	3.699790	5.769163	18.398277	19.708473
$t = 200$	3.636200	9.795562	3.828915	5.966647	18.562134	18.879873
$t = 300$	3.565355	9.635728	3.776376	5.859352	18.554811	18.051626
$t = 500$	3.485645	10.009463	3.955226	6.054237	18.492805	17.511011
$t = 1000$	3.462630	10.131654	3.965165	6.166488	18.469545	17.270458
$t = 2000$	3.476265	10.128613	3.961486	6.167127	18.508831	17.515448
$t = 3000$	3.509866	10.193808	3.973592	6.220215	18.514357	18.061195
$t = 5000$	3.497070	10.205850	3.983506	6.222344	18.516216	18.043194
$t = 10000$	3.509274	10.225508	4.004159	6.221349	18.540146	18.191513
$t = 25000$	3.513608	10.234055	4.016003	6.218052	18.542419	18.231838
$t = 50000$	3.523577	10.244403	4.003931	6.240472	18.550995	18.352444
$t = 100000$	3.528718	10.230343	3.994617	6.235726	18.548771	18.362995
$t = 300000$	3.534500	10.248379	3.996583	6.251795	18.553223	18.476212
$t = 500000$	3.535633	10.249247	4.000984	6.248262	18.552517	18.491827

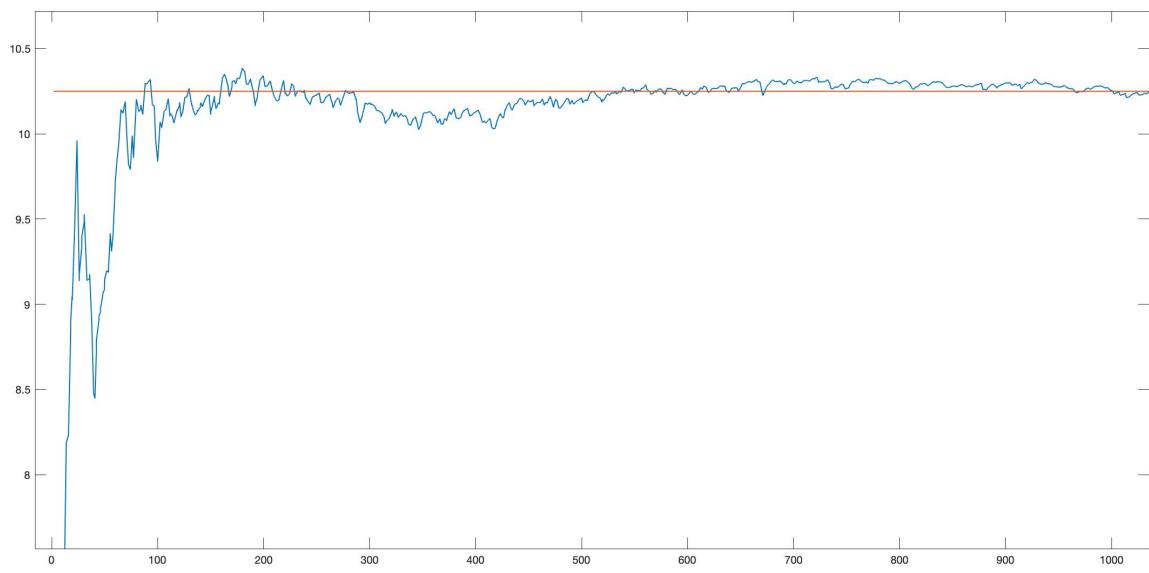
Dalla tabella precedente è possibile notare che è presente una fase di transitorio fino al tempo 50000s , dopo di che il sistema raggiunge lo stato di regime generale.

Sono state analizzate alcune statistiche per capire il loro evolvere durante la simulazione.

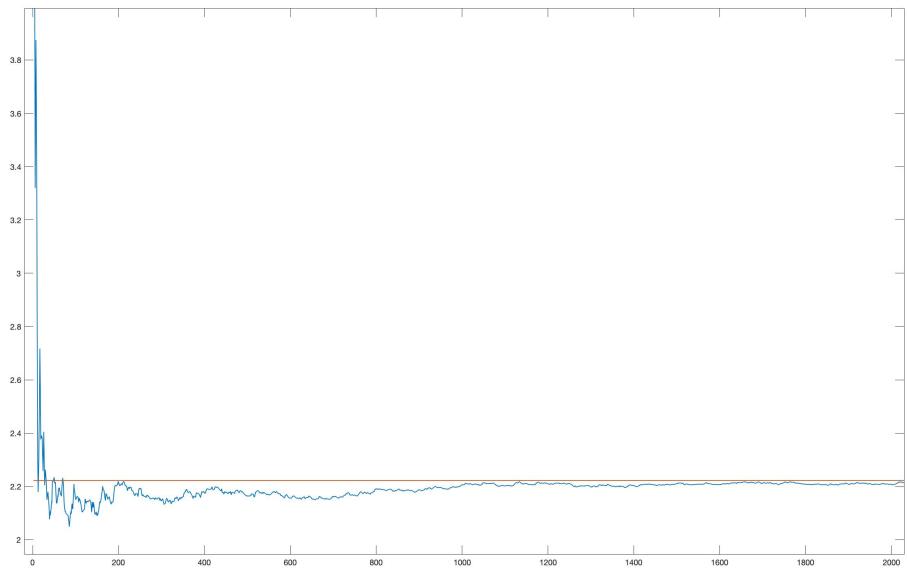
Di seguito vengono riportati i grafici riguardanti i tempi di risposta globali e per classe (riportati sull'asse verticale) durante il tempo di simulazione (sull'asse orizzontale).



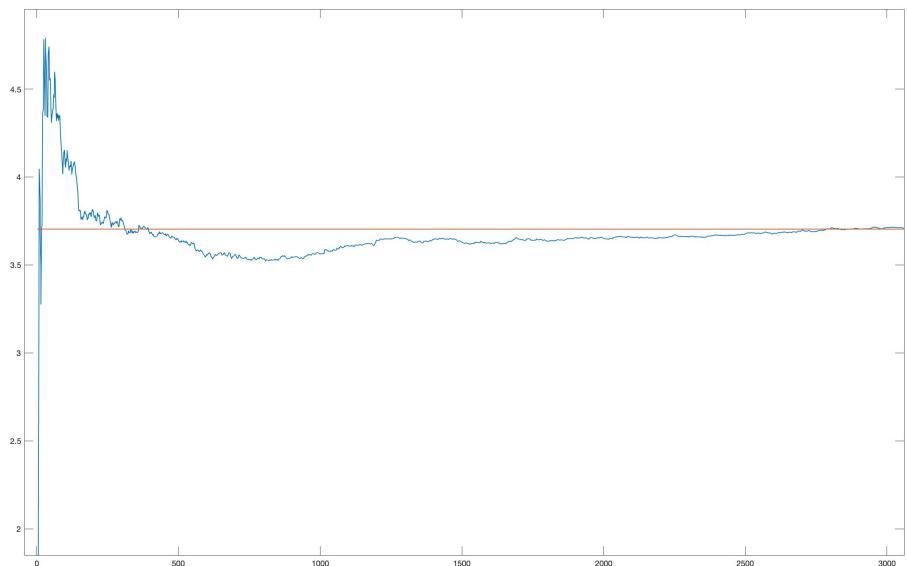
Global System Response time



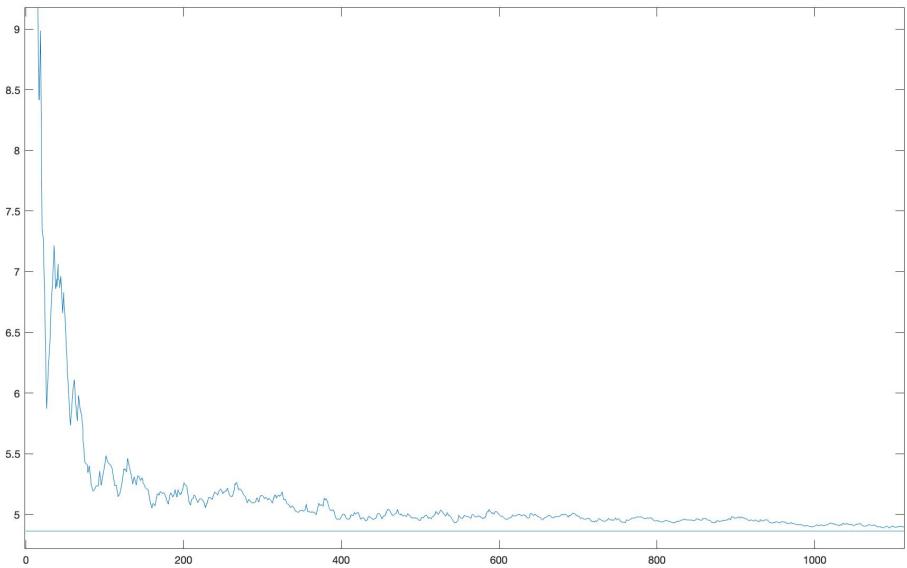
Global Throughput



*Cloudlet Response time  
(class 1 jobs)*



*Cloudlet Response time  
(class 2 jobs)*



*Cloud Response time  
(class 2 jobs)*

## Statistiche ottenute

Il codice descritto precedentemente è stato testato per effettuare una simulazione per un intervallo di tempo tale che il sistema risultasse stazionario, utilizzando il seed iniziale 123456789; le statistiche ricavate sono riassunte nelle tabelle di seguito. Sono state riportate due colonne che presentano i valori relativi alla simulazione con interruzione dei job di tipo 2 random e con interruzione basata sul tempo di next-event:

<b>Statistiche generali</b>		<b>Interruzione random</b>	<b>Interruzione longest time</b>
System global response time	$E[t]$	3.538013	3.089458
System response time for class 1 jobs	$E[t_1]$	2.223017	2.224748
System response time for class 2 jobs	$E[t_2]$	4.379736	3.642241
System Global throughput	$X$	10.248359	10.268587
System throughput for class 1 jobs	$X_1$	3.999713	4.0832033
System throughput for class 2 jobs	$X_2$	6.248645	6.195776
Percentage class 2 interrupted jobs	%	35.465875 %	24.183%
Response Time class 2 interrupted jobs	$E[t_{int}]$	6.844380	6.607023

<b>Statistiche relative al Cloudlet</b>		<b>Interruzione random</b>	<b>Interruzione longest time</b>
Cloudlet global response time	$E[t]_{cl}$	2.805445	2.416639
Cloudlet response time for class 1 jobs	$E[t_1]_{cl}$	2.222069	2.223761
Cloudlet response time for class 2 jobs	$E[t_2]_{cl}$	3.704803	2.640646
Cloudlet Global throughput	$X_{cl}$	6.590556	7.472945
Cloudlet throughput for class 1 jobs	$X_{cl1}$	3.997502	4.081138
Cloudlet throughput for class 2 jobs	$X_{cl2}$	2.593054	3.402175
Effective class 1 jobs Cloudlet throughput	$rate_{cl1}$	3.993628	3.994099
Effective class 2 jobs Cloudlet throughput	$rate_{cl2}$	2.594595	3.439079
Mean (total) population on Cloudlet	$E[N]_{cl}$	18.553458	18.059412
Mean class 1 jobs population on Cloudlet	$E[N_1]_{cl}$	9.050082	9.075473
Mean class 2 jobs population on Cloudlet	$E[N_2]_{cl}$	9.503376	8.983941

<b>Statistiche relative al Cloud</b>		<b>Interruzione random</b>	<b>Interruzione longest time</b>
<i>Cloud global response time</i>	$E[t]_{cloud}$	4.857952	4.866993
<i>Cloud response time for class 1 jobs</i>	$E[t_1]_{cloud}$	3.933928	4.368122
<i>Cloud response time for class 2 jobs</i>	$E[t_2]_{cloud}$	4.858509	4.867319
<i>Cloud Global throughput</i>	$X_{cloud}$	3.657802	2.795642
<i>Cloud throughput for class 1 jobs</i>	$X_{cloud\ 1}$	0.002211	0.002066
<i>Cloud throughput for class 2 jobs</i>	$X_{cloud\ 2}$	3.655591	2.793601
<i>Mean (total) population on Cloud</i>	$E[N]_{cloud}$	18.527233	13.606372
<i>Mean class 1 jobs population on Cloud</i>	$E[N_1]_{cloud}$	0.009737	0.009023
<i>Mean class 2 jobs population on Cloud</i>	$E[N_2]_{cloud}$	18.517496	13.597348

La prima colonna riporta i valori ottenuti quando il criterio di prelazione è di tipo random, ovvero quando viene scelto il primo server del Cloudlet che possiede un job di tipo due in esecuzione. La seconda colonna, invece, riporta i valori ottenuti quando il job prelazionato è quello che possiede il tempo di next-event maggiore.

Si può notare che, nel caso del secondo metodo di prelazione, si ha una diminuzione del tempo di risposta medio del sistema, una diminuzione dei tempi di risposta per job di classe 2 nel Cloudlet, una netta diminuzione del numero medio di job nel Cloud, oltre che una riduzione nella percentuale di job interrotti.

Queste differenze possono essere attribuite al fatto che, i job con tempo di next-event maggiore sono generalmente quelli arrivati per ultimi nel Cloudlet, ovvero quelli che risiedono nel Cloudlet da meno tempo. Di conseguenza, interrompendo tali job non si rischia di interrompere un job già in servizio da molto tempo, risparmiando così tempo di esecuzione sul Cloudlet.

Inoltre, tale criterio fa sì che vengano completati più job di tipo due sul Cloudlet, e di conseguenza esso avrà più posti liberi a disposizione; questo spiega la notevole diminuzione di popolazione media sul Cloud e la diminuzione della percentuale di job interrotti. Infatti, avendo più posti liberi a disposizione, il Cloudlet potrà accettare più job di classe 1 in arrivo senza dover interrompere job di classe 2 in esecuzione.

Come verifica dei risultati ottenuti è stata applicata la legge di Little per capire se i valori simulati possano essere considerati accettabili:

$$E[N]_{clet} = 18.553458$$

$$\lambda_{clet} \cdot E[t]_{clet} = 18.489442$$

$$E[N_1]_{clet} = 9.050082$$

$$\lambda_{clet1} \cdot E[t_1]_{clet} = 8.882725$$

$$E[N_2]_{clet} = 9.503376$$

$$\lambda_{clet2} \cdot E[t_2]_{clet} = 9.606754$$

$$E[N]_{cloud} = 18.527233$$

$$\lambda_{cloud} \cdot E[t]_{cloud} = 17.769427$$

$$E[N_1]_{cloud} = 0.009737$$

$$\lambda_{cloud1} \cdot E[t_1]_{cloud} = 0.008698$$

$$E[N_2]_{cloud} = 18.517496$$

$$\lambda_{cloud2} \cdot E[t_2]_{cloud} = 17.760722$$

Dai risultati ottenuti si può notare che, nella maggior parte dei casi, i valori simulati della popolazione media sono leggermente maggiori di quelli provenienti dall'applicazione della legge di Little, e quest'ultimi in generale si avvicinano di più ai valori teorici. Nel complesso, tuttavia, considerando un certo margine di errore, i valori simulati possono essere considerati nel complesso accettabili.

Nella tabella seguente vengono riportate tutte le statistiche di interesse confrontate con i valori teorici determinati precedentemente (i valori simulati fanno riferimento alle statistiche ottenute con il metodo di prelazione random). Per ogni statistica è stato calcolato il relativo intervallo di confidenza al 95% utilizzando il metodo dei Batch Means (le statistiche raccolte durante una simulazione “long-run” di 50000s sono state suddivise in 64 batches).

	<b>Valori Teorici</b>	<b>Valori Simulati</b>	<b>Intervallo di Confidenza</b>
$E[t]$	3.686103 ( $\alpha = 0.4$ )	3.538013	[ 3.537680 , 3.538346 ]
$E[t_1]$	2.223842	2.223017	[ 2.222636 , 2.223397 ]
$E[t_2]$	4.621952 ( $\alpha = 0.4$ )	4.379736	[ 4.379268 , 4.380203 ]
$X$	10.25	10.248359	[ 10.247110 , 10.249607 ]
$X_1$	4	3.999713	[ 3.998961 , 4.000466 ]
$X_2$	6.25	6.248645	[ 6.247957 , 6.249334 ]
$\%_{int}$	30.4857 %	35.465875 %	[ 35.454041 , 35.477708 ]
$E[t_{int}]$	6.826936 ( $\alpha = 0.4$ )	6.844380	[ 6.843191 , 6.845569 ]
$E[t]_{clct}$	2.640399 ( $\alpha = 0.4$ )	2.805445	[ 2.805181 , 2.805710 ]
$E[t_1]_{clct}$	2.222222	2.222069	[ 2.221690 , 2.222447 ]
$E[t_2]_{clct}$	3.026243 ( $\alpha = 0.4$ )	3.704803	[ 3.704320 , 3.705285 ]
$X_{clct}$	7.007185	6.590556	[ 6.589943 , 6.591170 ]
$X_{clct1}$	3.996355	3.997502	[ 3.996755 , 3.998249 ]
$X_{clct2}$	3.01083	2.593054	[ 2.592417 , 2.593691 ]
$E[N]_{clct}$	17.821449	18.553458	[ 18.552999 , 18.553918 ]
$E[N_1]_{clct}$	8.880788	9.050082	[ 9.047914 , 9.052250 ]
$E[N_2]_{clct}$	9.111501 ( $\alpha = 0.4$ )	9.503376	[ 9.501558 , 9.505194 ]
$E[t]_{cloud}$	4.870584	4.857952	[ 4.857469 , 4.858435 ]
$E[t_1]_{cloud}$	4	3.933928	[ 3.909525 , 3.958331 ]
$E[t_2]_{cloud}$	4.871565	4.858509	[ 4.858023 , 4.858995 ]
$X_{cloud}$	3.242816	3.657802	[ 3.656672 , 3.658933 ]
$X_{cloud1}$	0.003645	0.002211	[ 0.002187 , 0.002235 ]
$X_{cloud2}$	3.239171	3.655591	[ 3.654466 , 3.656716 ]
$E[N]_{cloud}$	15.794413	18.527233	[ 18.521312 , 18.533154 ]
$E[N_1]_{cloud}$	0.014580	0.009737	[ 0.009598 , 0.009876 ]
$E[N_2]_{cloud}$	15.779837	18.517496	[ 18.511575 , 18.523417 ]

## Conclusioni

Analizzando le statistiche ottenute applicando i due diversi algoritmi ciò che emerge è che i risultati simulati sono in linea con le considerazioni teoriche. Dal punto di vista delle statistiche globali si nota che il tempo medio di risposta del sistema per job di classe 1 diminuisce nel secondo algoritmo, mentre quello relativo ai job di classe 2 aumenta:

Statistica Transiente (200s)	Algoritmo 1	Algoritmo 2
$E[t_1]$	2.938245	2.225911
$E[t_2]$	4.043463	4.363726

Statistica Steady-State (50000s)	Algoritmo 1	Algoritmo 2
$E[t_1]$	2.962374	2.223017
$E[t_2]$	4.052807	4.379736

Questa variazione è dovuta al fatto che nel secondo algoritmo sono favoriti i job di classe 1 rispetto a quelli di classe 2; i job di classe 1 infatti vengono serviti prevalentemente sul Cloudlet (che possiede un tasso di servizio maggiore del Cloud), e quindi la loro esecuzione richiede un tempo in media minore.

Al contrario, i job di classe 2 nel secondo algoritmo subiscono prelazione, e questo fa sì che il loro tempo medio di risposta aumenti rispetto all'algoritmo 1 (infatti il tempo medio di risposta per i job prelazionati tiene conto del tempo speso nel Cloudlet, del tempo di servizio sul Cloud e del tempo di setup necessario per la ripresa dell'esecuzione sul Cloud).

Per quanto riguarda il throughput, sia il valore globale che quello per classi non subisce grandi variazioni tra i due algoritmi, e in entrambi i casi i valori si mantengono vicini a quelli teorici. Il throughput relativo ai singoli service node, invece, varia notevolmente:

Statistica Transiente (200s)	Algoritmo 1	Algoritmo 2
$X_{cl}$	5.724571	6.272195
$X_{cl1}$	2.234487	3.771617
$X_{cl2}$	3.490084	2.500578

Statistica Transiente (200s)	Algoritmo 1	Algoritmo 2
$X_{cloud}$	3.926827	3.410671
$X_{cloud1}$	1.526829	0.001421
$X_{cloud2}$	2.399999	3.409250

Statistica Steady-State (50000s)	Algoritmo 1	Algoritmo 2
$X_{clet}$	6.095100	6.590556
$X_{clet1}$	2.403797	3.997502
$X_{clet2}$	3.691303	2.593054
$X_{cloud}$	4.170596	3.657802
$X_{cloud1}$	1.641389	0.002211
$X_{cloud2}$	2.529208	3.655591

Come nel caso dei tempi di risposta, anche per il throughput le variazioni sono dovute al differente modo in cui sono favoriti i job di classe uno nel secondo algoritmo: a causa della prelazione, il throughput per i job di classe 1 aumenta nel Cloudlet e diminuisce notevolmente nel Cloud; al contrario, con il secondo algoritmo si ha una diminuzione del throughput per i job di classe 2 nel Cloudlet e un aumento nel Cloud.

Si nota un aumento del tempo medio di risposta per job di classe 2 sul Cloud nel caso del secondo algoritmo: l'aumento è dovuto al fatto che per i job prelazionati che arrivano al Cloud è necessario un tempo di setup per la ripresa dell'esecuzione; per tali job infatti il tempo di risposta viene calcolato come somma del tempo di setup e del tempo di servizio sul Cloud.

Vi sono delle variazioni anche nel numero medio di job nei due service node:

Statistica Transiente (200s)	Algoritmo 1	Algoritmo 2
$E[N]_{clet}$	18.681686	18.444598
$E[N_1]_{clet}$	5.220047	8.927762
$E[N_2]_{clet}$	13.461639	9.516836
$E[N]_{cloud}$	18.365273	17.788678
$E[N_1]_{cloud}$	6.611515	0.005753
$E[N_2]_{cloud}$	11.753758	17.782925

Statistica Steady-State (50000s)	Algoritmo 1	Algoritmo 2
$E[N]_{cllet}$	18.775070	18.553458
$E[N_1]_{cllet}$	5.505145	9.050082
$E[N_2]_{cllet}$	13.269925	9.503376
$E[N]_{cloud}$	18.577144	18.527233
$E[N_1]_{cloud}$	6.907517	0.009737
$E[N_2]_{cloud}$	11.671719	18.517496

Dal confronto emerge che con il secondo algoritmo il numero medio di job di classe 1 aumenta nel Cloudlet, mentre subisce una netta diminuzione nel Cloud. Per i job di classe 2 avviene invece l'opposto. La prelazione dei job della classe 2 dal Cloudlet in favore di quelli della classe 1 giustifica l'aumento del numero medio di job della classe 1 nel Cloudlet e l'aumento del numero medio di job della classe 2 nel Cloud.