

UNIVERSITÀ DI ROMA TOR VERGATA

# Right Sizing Spike Server

Relazione del progetto di PMCSN

Luca Falasca

lucafalasca08@gmail.com

luca.falasca@alumni.uniroma2.eu

Matricola: 0334722

A.A 2025/2026

# Contents

<b>1</b>	<b>Caso di Studio</b>	<b>2</b>
<b>2</b>	<b>Obiettivi dello studio</b>	<b>2</b>
<b>3</b>	<b>Modello concettuale</b>	<b>2</b>
3.1	Tempi di servizio . . . . .	3
3.2	Spike server . . . . .	3
3.3	Web Server . . . . .	3
<b>4</b>	<b>Modello delle specifiche</b>	<b>3</b>
4.1	Componenti fisici del sistema: . . . . .	3
4.2	Specifiche del carico di lavoro: . . . . .	3
4.3	Logica di controllo: . . . . .	4
4.4	Metrica di valutazione delle prestazioni: . . . . .	4
4.5	Gestione della fine della simulazione: . . . . .	4
<b>5</b>	<b>Modello Computazionale</b>	<b>4</b>
5.1	Stato del sistema . . . . .	4
5.2	Descrizione del job . . . . .	4
5.3	Identificazione degli eventi . . . . .	5
5.4	Logica di controllo e routing . . . . .	5
5.5	Configurazione dei parametri delle distribuzioni . . . . .	5
5.6	Struttura del simulatore . . . . .	6
5.6.1	Parallelismo . . . . .	6
5.7	Raccolta delle metriche . . . . .	7
5.8	Scheduler . . . . .	8
<b>6</b>	<b>Verifica</b>	<b>9</b>
6.1	Verifica del routing basato su soglia $SI_{max}$ . . . . .	9
6.1.1	Verifica con $SI_{max} = 0$ . . . . .	9
6.1.2	Verifica con $SI_{max} = \infty$ . . . . .	10
6.2	Verifica delle metriche prestazionali . . . . .	10
6.2.1	Verifica dell'utilizzazione al variare di $SI_{max}$ . . . . .	10
6.2.2	Verifica del throughput . . . . .	11
6.3	Verifica della distribuzione iperesponenziale . . . . .	12
<b>7</b>	<b>Validazione</b>	<b>13</b>
7.1	Confronto tempi di risposta . . . . .	13
7.2	Confronto Throughput . . . . .	16
<b>8</b>	<b>Risultati</b>	<b>17</b>
8.1	Obiettivo 1 . . . . .	18
8.2	Obiettivo 2 . . . . .	19
8.3	Obiettivo 3 . . . . .	21
<b>9</b>	<b>Conclusioni</b>	<b>23</b>
<b>A</b>	<b>Analisi Transitorio</b>	<b>24</b>
A.1	Transitorio Obiettivo 1 . . . . .	24
A.2	Obiettivo 2 . . . . .	26
A.3	Obiettivo 3 . . . . .	26

# 1 Caso di Studio

Il sistema oggetto di studio è un'architettura di data center per un Internet Service Provider, progettata per gestire dinamicamente le fluttuazioni di carico e garantire la Quality of Service (QoS) ottimizzando al contempo l'uso delle risorse.

Il problema principale affrontato è il "right-sizing", ovvero come evitare sia il sovradimensionamento (spreco di risorse) sia il sottodimensionamento (violazione degli SLA e degrado delle prestazioni), specialmente in presenza di fluttuazioni di carico a breve e lungo termine.

L'architettura proposta, come descritto nel caso di studio 6.2 del libro di testo "Performance Engineer", si basa su un livello di scaling verticale che gestisce i picchi di carico improvvisi e di breve durata. Questo livello introduce uno Spike Server dedicato. Un Load Controller monitora un indicatore di picco (Spike Indicator, SI), definito come il numero di richieste concorrenti totali presenti su un Web Server (tutte le richieste vengono gestite con un scheduler processor sharing).

Il comportamento del sistema seguirebbe quanto descritto:

- Quando l'indicatore SI supera una soglia di allarme  $SI_{max}$ , le nuove richieste in arrivo non vengono più inviate al Web Server congestionato, ma vengono reindirizzate allo Spike Server.
- Quando il carico sul Web Server diminuisce e SI scende al di sotto della soglia, il routing delle richieste torna alla normalità.

## 2 Obiettivi dello studio

Lo studio si pone l'obiettivo di analizzare e validare l'efficacia del modello di autoscaling basato su spike server attraverso la simulazione. Gli obiettivi specifici sono:

- Determinare il valore di  $SI_{max}$  più alto possibile (provando vari valori di SI) che mantenga comunque il tempo di risposta medio  $E[R] \leq 8$  secondi, con un tasso di arrivo iperesponenziale con media di 6.66 req/s (400 req/min).
- Analizzare come varia il tempo di risposta medio al variare del valore di  $SI_{max}$  per carichi di lavoro crescenti (da 1 req/s a 12 req/s).
- Verificare come cambia il contesto (sempre sotto carichi crescenti e al variare di  $SI_{max}$ ) se lo spike server ha un tasso di servizio doppio rispetto al webserver principale, invece che uguale come nel caso base.

## 3 Modello concettuale

Il modello descritto può essere schematizzato come in figura 1.

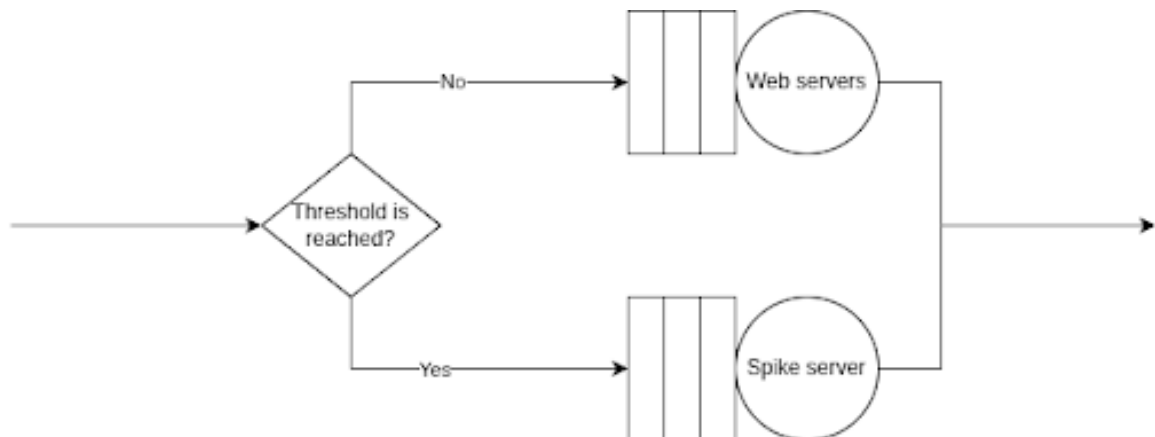


Figure 1: Modello concettuale del sistema di autoscaling gerarchico con Spike Server

I job arrivano e a seconda del livello di pienezza del webserver vengono instradati o al webserver principale o allo spike server. Nonostante il sistema possa sembrare troppo semplice per una analisi simulativa, il fatto che il routing non sia probabilistico lo rende molto complesso da analizzare in modo analitico. il routing dei job, dipende strettamente dallo stato del webserver nel momento del routing.

### 3.1 Tempi di servizio

Nel caso di studio affrontato nel libro viene utilizzata una distribuzione iperesponenziale nei tassi di servizio per modellare il fatto che ad un server arrivano job di dimensione molto variabile.

### 3.2 Spike server

Lo Spike Server avrà la stessa potenza rispetto al webserver nei primi 2 obiettivi. Nell'obiettivo 3, invece, avrà un tasso di servizio doppio rispetto a quello del webserver principale, in modo da poter smaltire rapidamente i picchi di carico. Per il calcolo di  $SI_{max}$  ottimale è importante sottolineare che lo spike server è bene usarlo il meno possibile, in quanto è una risorsa più costosa rispetto al webserver. Quindi anziché cercare di minimizzare il tempo di risposta medio del sistema, si cerca di rispettare lo SLA (8 secondi) utilizzando il meno possibile lo spike server.

### 3.3 Web Server

Il web server sarà solamente uno, per semplicità di modellazione. Per questo motivo non vengono gestite le fluttuazioni a lungo termine, in quanto queste richiederebbero un sistema con più webserver per implementare uno scaling orizzontale.

## 4 Modello delle specifiche

Queste variabili matematiche sono la rappresentazione dello stato del sistema:

- $SI(t) \in [0, SI_{max})$ : Spike indicator, indica il numero di job presenti nel web server al tempo  $t$ .
- $n_{spike}(t) \in [0, \infty)$ : numero di job presenti nello spike server al tempo  $t$ .

per cui lo stato del sistema è definito come la tupla  $S(t) = (SI(t), n_{spike}(t))$ .

### 4.1 Componenti fisici del sistema:

- Web server principale: server principale che gestisce le richieste in arrivo.
- Spike server: server secondario che viene attivato quando il carico sul web server principale supera una certa soglia ( $SI_{max}$ ).

Componenti logici del sistema:

- Load Controller: componente che monitora l'indicatore di picco (Spike Indicator, SI) e decide il routing delle richieste in base al valore di SI e alla soglia  $SI_{max}$ .

### 4.2 Specifiche del carico di lavoro:

- Arrivi:
  - Valore base: Processo di arrivo iperesponenziale con  $cv = 4$  e media 0.15 secondi (6.66 req/s)
  - Valore stress: Processo di arrivo iperesponenziale con carico variabile da 1 req/s a 12 req/s
- Servizio: Distribuzione iperesponenziale con  $cv = 4$  e media 0.16 secondi (web server e spike server), media 0.08 secondi (spike server nell'obiettivo 3)
- Scheduling: Processor Sharing

Vale la pena notare che, nel caso base del webserver, siccome  $\rho > 1$ , il sistema è instabile e senza spike server il tempo di risposta divergerebbe a infinito.

### 4.3 Logica di controllo:

- Arrivi nuovi job:
  - Se  $SI < SI_{max}$ : invia al webserver principale
  - Se  $SI = SI_{max}$ : invia allo spike server
- Completamento job:
  - Decrementa il contatore SI se il job era nel webserver principale
  - Decrementa il contatore n\_spike se il job era nello spike server

Nella logica originale del caso di studio, il valore di SI parte da  $SI_{max}$  e viene decrementato ad ogni completamento di un job. La logica adottata, nonostante sia inversa, è perfettamente equivalente.

### 4.4 Metrica di valutazione delle prestazioni:

Lo SLA da rispettare è un tempo di risposta medio  $E[R] \leq 8$  secondi.

### 4.5 Gestione della fine della simulazione:

Dato che il sistema viene studiato a regime, il sistema viene simulato per un tempo fisso di 5000 secondi, scartando i primi 500 secondi di fase transitoria. Inoltre una volta raggiunto il tempo di stop, vengono completati tutti i job ancora in servizio, fermando gli arrivi di nuovi job allo scadere del tempo di stop. La fase transitoria viene scartata per evitare che i risultati vengano influenzati dallo stato iniziale del sistema e quindi ridurre il tempo di simulazione necessario per osservare il comportamento a regime del sistema.

## 5 Modello Computazionale

Il sistema di simulazione è implementato in Python e segue un modello ad eventi discreti. Sono state utilizzate alcuni file della libreria di Steve Park & Dave Geyer, tradotti in python da Philip Steele. In particolare sono state utilizzate le librerie per la generazione di numeri casuali per il multistream (rngs.py) e per le distribuzioni (rvgs.py).

### 5.1 Stato del sistema

Le variabili di programmazione che rappresentano lo stato del sistema sono:

- web\_jobs : list: è una lista che contiene i job in fase di processamento nel web server, la sua lunghezza rappresenta l'indicatore SI
- spike\_jobs : list: è una lista che contiene i job in fase di processamento nello spike server.

### 5.2 Descrizione del job

Ogni job è rappresentato da una classe Job:

```
class Job:
    def __init__(self,
                  arrival_time,
                  service_demand,
                  is_spike=False):
        self.arrival_time = arrival_time
        self.service_demand = service_demand
        self.remaining_work = service_demand
```

```
self.is_spike = is_spike
```

Dove:

- arrival\_time: tempo di arrivo del job
- service\_demand: tempo di servizio totale richiesto dal job
- remaining\_work: lavoro rimanente da completare (utile per lo scheduling processor sharing)
- is\_spike: booleano che indica se il job è stato assegnato allo spike server

### 5.3 Identificazione degli eventi

- arrivo di un nuovo job
- completamento di un job nel web server
- completamento di un job nello spike server

### 5.4 Logica di controllo e routing

Ogni volta che arriva un nuovo job:

- Se  $\text{len}(\text{web\_jobs}) < SI_{max}$ , il job viene aggiunto a web\_jobs.
- Se  $\text{len}(\text{web\_jobs}) == SI_{max}$ , il job viene aggiunto a spike\_jobs.

Ogni volta che un job completa rimuove dalla lista corrispondente il job (web\_jobs o spike\_jobs).

### 5.5 Configurazione dei parametri delle distribuzioni

- Stream 0: Inter-arrivi (Iperesponenziale, media 0.15s, cv=4).
- Stream 1: Servizio Web Server (Iperesponenziale, media 0.16s, cv=4).
- Stream 2: Servizio Spike Server (Iperesponenziale, media 0.16s, cv=4).

La distribuzione iperesponenziale è stata implementata utilizzando l'esponenziale e l'uniforme della libreria rvgs.py. Con l'uniforme calcoliamo la probabilità di scegliere una delle due esponenziali, e con l'esponenziale calcoliamo il servizio vero e proprio (Figura 2).

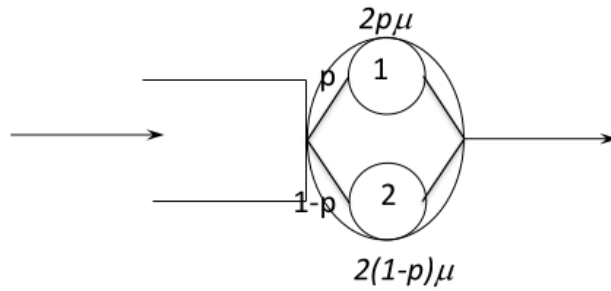


Figure 2: Generazione di una variabile casuale iperesponenziale

In particolare date in input la media e il coefficiente di variazione  $cv$ . Sapendo che  $g(p) = cv^2$ , possiamo calcolare  $p_1$ ,  $p_2$ ,  $m_1$  e  $m_2$  come:

$$p_1 = \frac{1 + \sqrt{\frac{cv^2 - 1}{cv^2 + 1}}}{2}$$

$$p_2 = 1 - p_1$$

$$m_1 = \frac{mean}{2 \cdot p_1}$$

$$m_2 = \frac{mean}{2 \cdot p_2}$$

e quindi utilizzare un semplice if per scegliere quale esponenziale utilizzare in base ad un campione uniforme:

```
u = Uniform(0,1)
if u < p1:
    return Exponential(m1)
else:
    return Exponential(m2)
```

È importante notare che per ogni campionamento dell'esponenziale si utilizzano, quindi, due estrazioni dallo stream del generatore di numeri casuali.

## 5.6 Struttura del simulatore

Il simulatore è strutturato come una classe Simulator che gestisce l'inizializzazione, l'esecuzione e la raccolta delle metriche della simulazione. La classe ha una serie di costanti di configurazione:

```
START          = 0.0
BIAS_PHASE     = 500.0
STOP           = 5000.0
INFINITY       = 1e15
SEED           = 8
REPLICAS       = 100
N_PROCESSES    = 40
```

Dove:

- START: tempo di inizio della simulazione
- BIAS\_PHASE: durata della fase transitoria (500 secondi)
- STOP: tempo di fine della simulazione (5000 secondi)
- INFINITY: un valore molto grande usato per inizializzare i tempi di evento (deve essere molto più grande di STOP)
- SEED: seme per il generatore di numeri casuali, che è stato utilizzato insieme alla libreria rngs.py per tutti i generatori di numeri casuali
- REPLICAS: numero di repliche della simulazione (100 repliche)
- N\_PROCESSES: numero di processi paralleli per l'esecuzione delle repliche (40 processi)

### 5.6.1 Parallelismo

Per velocizzare l'esecuzione delle repliche della simulazione, è stata implementata una logica di multiprocessing per sfruttare tutti i core della CPU disponibili. Tuttavia per farlo è stato necessario prendere alcune accortezze che permettessero di non minare né l'indipendenza delle repliche né la riproducibilità dei risultati. In particolare:

- ad ogni processo vengono assegnati 3 stream dei 256 disponibili, in modo che ogni processo abbia il proprio spazio di numeri casuali indipendenti dagli altri processi, ma in maniera tale che possano procedere in parallelo senza che la non deterministicità del parallelismo influisse sull'ordine di estrazione dei numeri casuali.

- Per il numero di repliche introdotte, ogni processo non si avvicinava nemmeno lontanamente al limite dello stream assegnato.

$$MODULUS/STREAM \approx 8388607$$

È stato comunque implementato un meccanismo in cui una volta sfiorato il limite dello stream, il processo riceva 3 nuovi stream esclusivi. Ed essendo il numero di processi 40, e ogni processo usa 3 stream, questo è tranquillamente possibile senza sfiorare il limite dei 256 stream e senza ulteriori meccanismi di controllo.

- Sempre per garantire la riproducibilità dei risultati, ad ogni processo vengono assegnate sempre lo stesso numero di repliche da eseguire, in modo tale che le ultime esecuzioni non vengano assegnate a processi diversi (e quindi stream diversi) in base all'ordine di esecuzione di essi.

## 5.7 Raccolta delle metriche

Per raccogliere le metriche è stata utilizzata una classe Track:

```
class Track:
    def __init__(self):
        self.area_node_web = 0.0
        self.area_node_spike = 0.0

        self.area_busy_web = 0.0
        self.area_busy_spike = 0.0

        self.completed_web = 0
        self.completed_spike = 0

        self.scaling_actions = 0
```

Dove:

- area\_node\_web: area sotto la curva del numero di job nel web server

$$\int_{BIAS\_PHASE}^{STOP} N_{web}(t) dt$$

- area\_node\_spike: area sotto la curva del numero di job nello spike server

$$\int_{BIAS\_PHASE}^{STOP} N_{spike}(t) dt$$

- area\_busy\_web: area sotto la curva del tempo di utilizzo del web server

$$\int_{BIAS\_PHASE}^{STOP} U_{web}(t) dt$$

- area\_busy\_spike: area sotto la curva del tempo di utilizzo dello spike server

$$\int_{BIAS\_PHASE}^{STOP} U_{spike}(t) dt$$

- completed\_web: numero di job completati nel web server
- completed\_spike: numero di job completati nello spike server
- scaling\_actions: numero di azioni di scaling (numero di volte in cui lo spike server è stato attivato)



Gli integrali sono calcolati ad ogni evento in maniera incrementale durante la simulazione.

Le metriche finali raccolte sono state:

- **web\_response\_time**: tempo di risposta medio nel web server calcolato come l'integrale del numero di job completati nel web server diviso per il numero di job completati nel web server

$$\frac{area\_node\_web}{completed\_web}$$

- **spike\_response\_time**: tempo di risposta medio nello spike server

$$\frac{area\_node\_spike}{completed\_spike}$$

- **total\_response\_time**: tempo di risposta medio totale

$$\frac{area\_node\_web + area\_node\_spike}{completed\_web + completed\_spike}$$

- **utilization\_web**: utilizzo medio del web server

$$\frac{area\_busy\_web}{TEMPO\_FINE\_SIMULAZIONE - BIAS\_PHASE}$$

- **utilization\_spike**: utilizzo medio dello spike server

$$\frac{area\_busy\_spike}{TEMPO\_FINE\_SIMULAZIONE - BIAS\_PHASE}$$

- **throughput\_web**: throughput medio del web server

$$\frac{completed\_web}{TEMPO\_FINE\_SIMULAZIONE - BIAS\_PHASE}$$

- **throughput\_spike**: throughput medio dello spike server

$$\frac{completed\_spike}{TEMPO\_FINE\_SIMULAZIONE - BIAS\_PHASE}$$

- **throughput\_total**: throughput medio totale

$$\frac{completed\_web + completed\_spike}{TEMPO\_FINE\_SIMULAZIONE - BIAS\_PHASE}$$

- **scaling\_actions**: numero di azioni di scaling

Per quanto riguarda le metriche finali su tutte le repliche, è stato utilizzato l'algoritmo di Welford per il calcolo delle medie e varianze e poi degli intervalli di confidenza al 95%.

## 5.8 Scheduler

Siccome lo scheduler è di tipo processor sharing, il tempo prima del prossimo evento da schedulare è calcolato come:

$$\min\{next\_arrival, time\_to\_complete\_web\_job, time\_to\_complete\_spike\_job\}$$

Dove:

- **next\_arrival**: tempo rimanente al prossimo arrivo

- `time_to_complete_web_job`: tempo rimanente al completamento del job con il minor lavoro rimanente nel web server
- `time_to_complete_spike_job`: tempo rimanente al completamento del job con il minor lavoro rimanente nello spike server

Per calcolare i tempi rimanenti al completamento dei job, non basta trovare il job con il minor lavoro rimanente, perché essendo in processor sharing il tempo di completamento dipende anche dal numero di job in servizio. Quindi il tempo di completamento del job con il minor lavoro rimanente è calcolato come:

$$time\_to\_complete\_web\_job = min\_remaining\_work \times len(web\_jobs)$$

$$time\_to\_complete\_spike\_job = min\_remaining\_work \times len(spike\_jobs)$$

Una volta trovato il tempo del prossimo evento, si avanza il tempo di simulazione di tale intervallo e si aggiorna il lavoro rimanente di tutti i job in servizio sottraendo il lavoro svolto in tale intervallo:

$$j.remaining\_work = j.remaining\_work - \frac{time\_advance}{len(web\_jobs)} \forall j \in web\_jobs$$

$$j.remaining\_work = j.remaining\_work - \frac{time\_advance}{len(spike\_jobs)} \forall j \in spike\_jobs$$

## 6 Verifica

Ci sono varie verifiche che si possono fare andando a controllare se tutta una serie di caratteristiche volute si riflettano veramente nel modello simulativo costruito e quindi nei risultati ottenuti. In particolare, verranno effettuate le seguenti verifiche:

- Verifica del corretto funzionamento di routing basato su soglia si  $SI_{max}$ .
  - Controllo che impostando  $SI_{max} = 0$  venga utilizzato solo lo spike server.
  - Controllo che impostando  $SI_{max} = \infty$  non venga mai utilizzato lo spike server.
- Verifica delle metriche prestazionali:
  - Verifico che aumentando  $SI_{max}$  l'utilizzazione del web server aumenti
  - Verifico che il throughput sia pari al tasso di arrivo quando il sistema è stabile e uguale al tasso di servizio quando il sistema è sovraccarico.
- Verifica della distribuzione
  - Verifica che media e cv della distribuzione iperesponenziale corrispondano ai valori impostati.

### 6.1 Verifica del routing basato su soglia $SI_{max}$

#### 6.1.1 Verifica con $SI_{max} = 0$

Impostando  $SI_{max} = 0$  si può verificare che tutti i job vengano instradati allo spike server. Infatti, come mostrato in figura 3, l'utilizzazione del web server è nulla, mentre tutto il carico viene gestito dallo spike server.

```

Simulazione completata con i seguenti parametri:
  SI_max: 0
  Arrival Mean: 0.15
  Web Mean: 0.16
  Spike Mean: 0.16
  Coefficiente di Variazione: 4.0
Statistiche raccolte con intervallo di confidenza al 95%:
spike_response_time      : 12.9032 +/- 1.1099 (Var: 32.069075)
total_response_time      : 12.9032 +/- 1.1099 (Var: 32.069075)
utilization_web          : 0.0000 +/- 0.0000 (Var: 0.000000)
utilization_spike        : 0.9910 +/- 0.0028 (Var: 0.000205)
throughput_web           : 0.0000 +/- 0.0000 (Var: 0.000000)
throughput_spike         : 6.5423 +/- 0.0339 (Var: 0.029877)
throughput_total         : 6.5423 +/- 0.0339 (Var: 0.029877)
scaling_actions          : 44.9200 +/- 7.3205 (Var: 1394.983434)

```

Figure 3: Verifica routing con  $SI_{max} = 0$

### 6.1.2 Verifica con $SI_{max} = \infty$

Impostando  $SI_{max} = \infty$  si può verificare che nessun job venga instradato allo spike server. Infatti, come mostrato in figura 4, l'utilizzazione dello spike server è nulla, mentre tutto il carico viene gestito dal web server.

```

Simulazione completata con i seguenti parametri:
  SI_max: 1000000000000000.0
  Arrival Mean: 0.15
  Web Mean: 0.16
  Spike Mean: 0.16
  Coefficiente di Variazione: 4.0
Statistiche raccolte con intervallo di confidenza al 95%:
web_response_time        : 13.1808 +/- 1.0418 (Var: 28.251530)
total_response_time      : 13.1808 +/- 1.0418 (Var: 28.251530)
utilization_web          : 0.9914 +/- 0.0028 (Var: 0.000202)
utilization_spike        : 0.0000 +/- 0.0000 (Var: 0.000000)
throughput_web           : 6.5389 +/- 0.0351 (Var: 0.032065)
throughput_spike         : 0.0000 +/- 0.0000 (Var: 0.000000)
throughput_total         : 6.5389 +/- 0.0351 (Var: 0.032065)
scaling_actions          : 0.0000 +/- 0.0000 (Var: 0.000000)

```

Figure 4: Verifica routing con  $SI_{max} = \infty$

## 6.2 Verifica delle metriche prestazionali

### 6.2.1 Verifica dell'utilizzazione al variare di $SI_{max}$

Impostando un carico di lavoro fisso e variando il valore di  $SI_{max}$ , si può verificare che l'utilizzazione del web server aumenti al crescere di  $SI_{max}$ , come mostrato in figura 5.

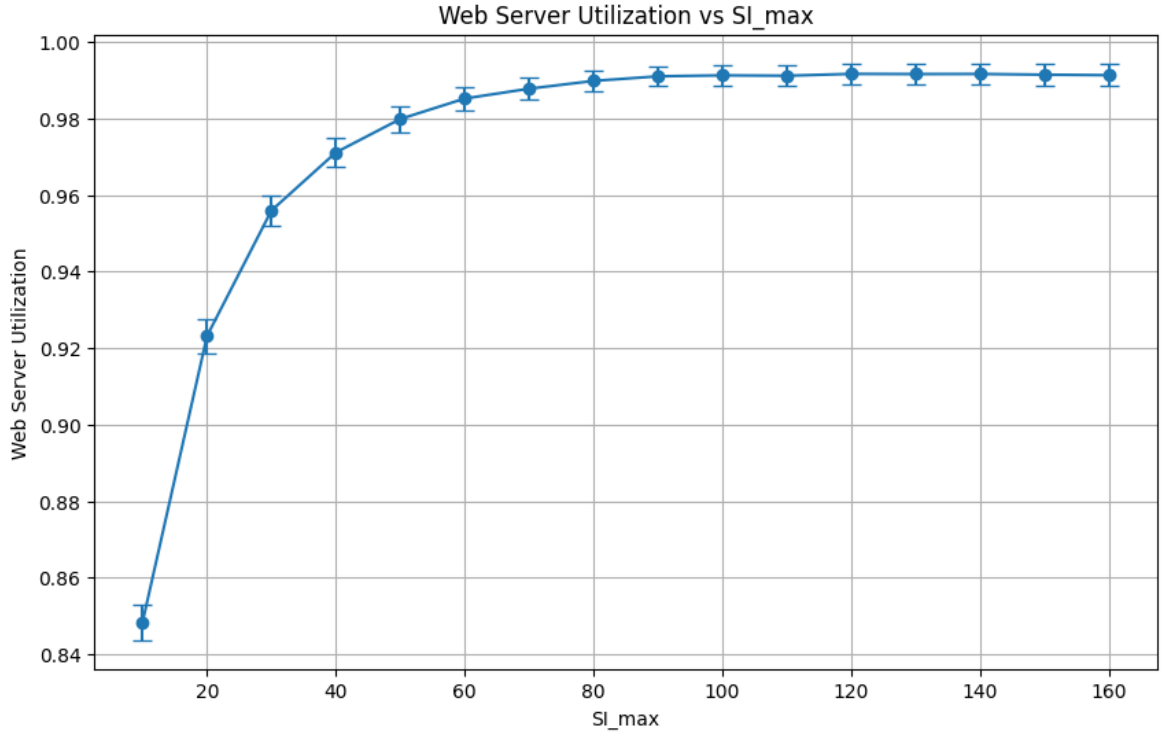


Figure 5: Verifica dell'utilizzazione del web server al variare di  $SI_{max}$

Table 1: Verifica dell'utilizzazione del web server al variare di  $SI_{max}$

$SI_{max}$	$U_{web}$	$U_{web}$ CI95
10	0.8197	0.0066
20	0.9058	0.0064
30	0.9427	0.0061
40	0.9619	0.0057
50	0.9728	0.0051
60	0.9788	0.0047
70	0.9818	0.0046
80	0.9832	0.0044
90	0.9838	0.0043
100	0.9844	0.0042
110	0.9847	0.0041
120	0.9847	0.0041
130	0.9849	0.0040
140	0.9851	0.0040
150	0.9850	0.0040
160	0.9848	0.0040

### 6.2.2 Verifica del throughput

Variando il carico di lavoro da 1 req/s a 12 req/s e impostando  $SI_{max} = \infty$ , in modo da misurare solo il web server, si può verificare che il throughput del sistema non superi mai il suo limite teorico, che nel caso considerato è uguale a  $\min(\mu_{web}, \lambda_{web})$ . Infatti, come evidenziato in Figura 6, il throughput misurato  $X_{measured} = N_{completed}/T_{sim}$  non supera il limite teorico e si mantiene sempre al di sotto di esso.

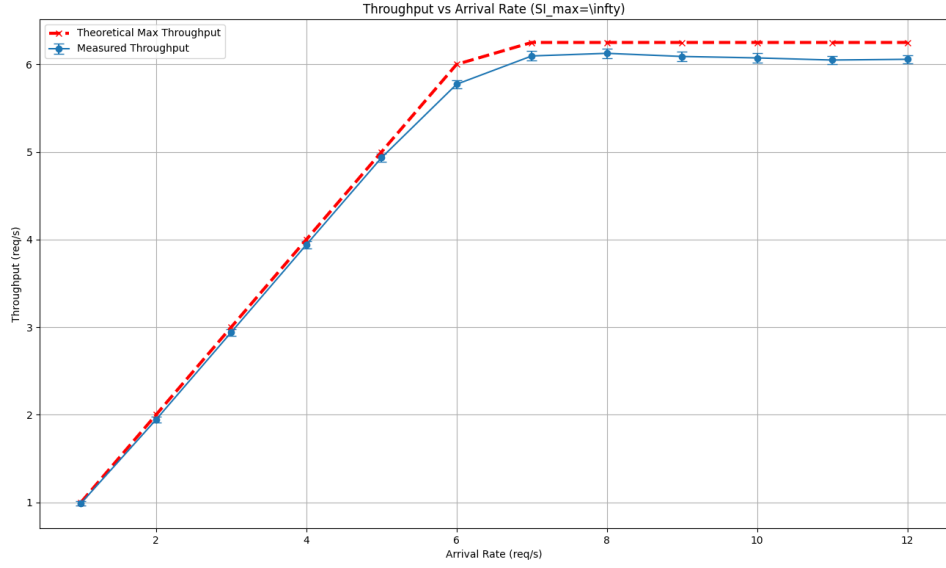


Figure 6: Throughput misurato che supera il valore teorico

Table 2: Verifica del Throughput

$\lambda$	$X_{measured}$	$X_{measured}$ CI95	$X_{theoretical}$
1	0.9889	0.0214	1.0000
2	1.9442	0.0304	2.0000
3	2.9411	0.0367	3.0000
4	3.9406	0.0442	4.0000
5	4.9337	0.0469	5.0000
6	5.7728	0.0475	6.0000
7	6.0958	0.0552	6.2500
8	6.1260	0.0553	6.2500
9	6.0902	0.0522	6.2500
10	6.0736	0.0512	6.2500
11	6.0492	0.0474	6.2500
12	6.0581	0.0463	6.2500

### 6.3 Verifica della distribuzione iperesponenziale

Per verificare che la distribuzione iperesponenziale implementata nel simulatore corrisponda a quella voluta, sono state effettuate delle misurazioni della media e del coefficiente di variazione dei tempi di servizio generati, confrontandoli con i valori teorici impostati. In particolare sono state effettuate sempre 100 repliche di simulazione con  $SI_{max} = 80$ . Effettivamente, come mostrato in figura 7, i valori teorici rientrano perfettamente negli intervalli di confidenza al 95% delle misurazioni effettuate.

```

meas_arrival_mean      : 0.1499 +/- 0.0007 (Var: 0.000014)
meas_arrival_cv        : 3.9961 +/- 0.0168 (Var: 0.007359)
meas_web_service_mean  : 0.1606 +/- 0.0006 (Var: 0.000011)
meas_web_service_cv    : 4.0032 +/- 0.0177 (Var: 0.008145)
meas_spike_service_mean : 0.1592 +/- 0.0030 (Var: 0.000230)
meas_spike_service_cv  : 3.9284 +/- 0.0819 (Var: 0.174522)

```

Figure 7: Verifica della distribuzione iperesponenziale dei tempi di servizio

## 7 Validazione

Dato che in questo studio non sono disponibili dati reali con cui confrontare i risultati ottenuti dalla simulazione, la validazione del modello simulativo confronterà i risultati ottenuti con quelli ottenuti dallo studio di riferimento nella prossima sezione del report.

### 7.1 Confronto tempi di risposta

Nello studio di riferimento vengono effettuati dei test sui tempi di risposta sul webserver e sullo spike server.

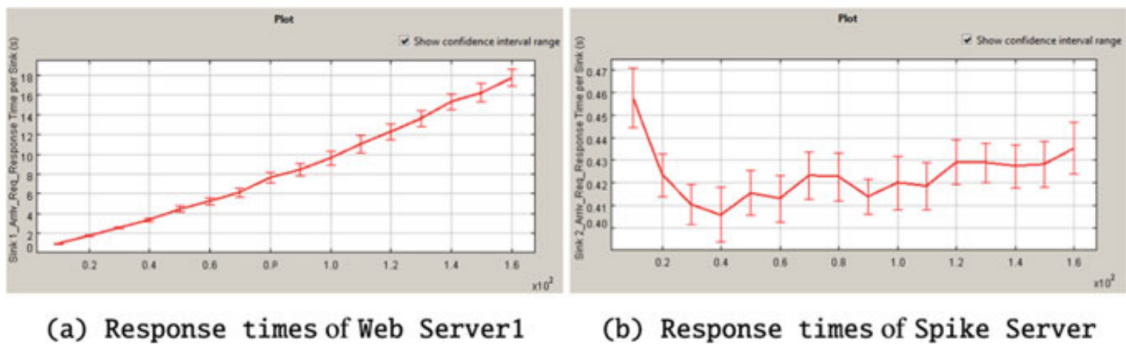


Figure 8: Tempi di risposta del webserver e dello spike server in funzione del carico nel caso di studio di Serazzi

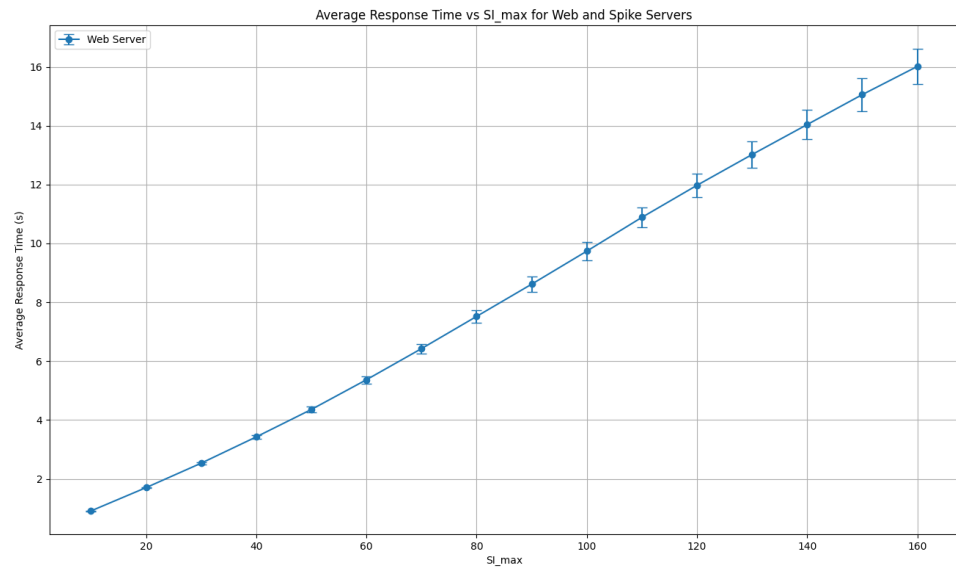


Figure 9: Tempi di risposta del webserver in funzione del SI\_max ottenuti dalla simulazione.

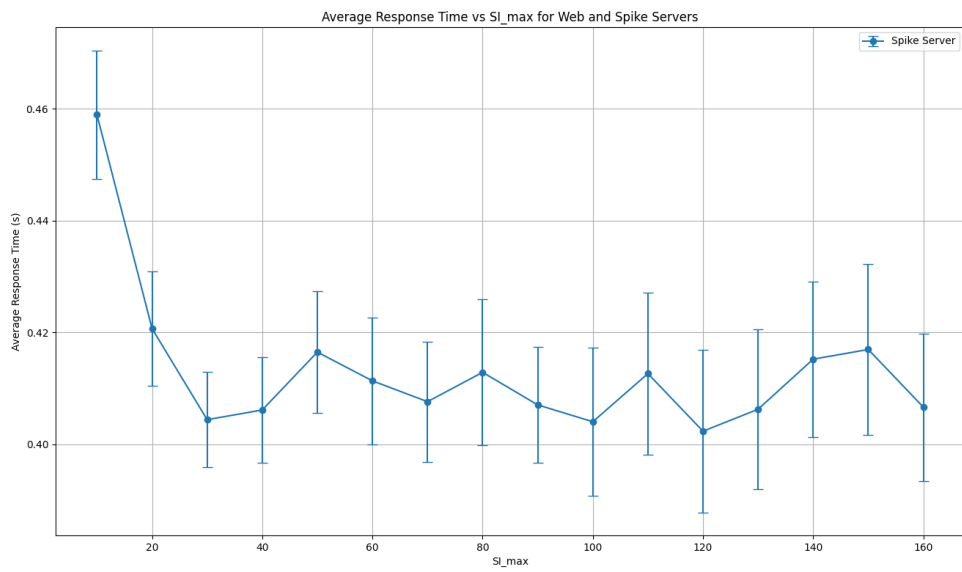


Figure 10: Tempi di risposta dello spike server in funzione del SI\_max ottenuti dalla simulazione.

Table 3: Validazione tempi di risposta del webserver

$SI_{max}$	$E[T_s]_{web}$	$E[T_s]_{web}$ CI95
10	0.9096	0.0075
20	1.7015	0.0197
30	2.5320	0.0375
40	3.4252	0.0605
50	4.3567	0.0912
60	5.3675	0.1252
70	6.4253	0.1667
80	7.5198	0.2103
90	8.6157	0.2536
100	9.7376	0.3011
110	10.8851	0.3462
120	11.9747	0.3967
130	13.0199	0.4450
140	14.0386	0.5004
150	15.0560	0.5574
160	16.0146	0.6099

Table 4: Validazione tempi di risposta dello spike server

$SI_{max}$	$E[T_s]_{spike}$	$E[T_s]_{spike}$ CI95
10	0.4589	0.0115
20	0.4207	0.0102
30	0.4044	0.0085
40	0.4061	0.0095
50	0.4165	0.0108
60	0.4113	0.0113
70	0.4076	0.0107
80	0.4129	0.0130
90	0.4070	0.0103
100	0.4040	0.0133
110	0.4126	0.0145
120	0.4023	0.0145
130	0.4063	0.0143
140	0.4152	0.0139
150	0.4169	0.0153
160	0.4066	0.0132

Controntandoli con i risultati ottenuti dalla simulazione progettata, si nota come l'andamento sia essenzialmente lo stesso, tuttavia i valori dei tempi differiscono. Questo è dovuto dal tempo di simulazione che il caso di studio originale non esplicita direttamente e quindi probabilmente differente dai 5000 secondi utilizzati nelle simulazioni di test. Tuttavia per non rendere troppo lunghi gli esperimenti, è stata eseguita una simulazione con  $SI_{max} = 160$  e aumentato il tempo di simulazione a 20000 secondi, per dimostrare che il motivo della differenza nei tempi di risposta è dovuto proprio al tempo di simulazione.



```

SI_max: 160
Arrival Mean: 0.15
Web Mean: 0.16
Spike Mean: 0.16
Coefficiente di Variazione: 4.0
Statistiche raccolte con intervallo di confidenza al 95%:
web_response_time      : 17.8339 +/- 0.2711 (Var: 1.913843)
spike_response_time    : 0.4290 +/- 0.0071 (Var: 0.001299)
total_response_time    : 16.6907 +/- 0.2125 (Var: 1.175528)
utilization_web        : 0.9991 +/- 0.0003 (Var: 0.000003)
utilization_spike      : 0.0688 +/- 0.0033 (Var: 0.000279)
throughput_web         : 6.1952 +/- 0.0146 (Var: 0.005573)
throughput_spike       : 0.4294 +/- 0.0198 (Var: 0.010167)
throughput_total       : 6.6247 +/- 0.0158 (Var: 0.006467)
scaling_actions        : 2282.1700 +/- 96.0887 (Var: 240343.314242)

```

Figure 11: Tempi di risposta del webserver con  $SI_{max} = 160$  ottenuti dalla simulazione con 20000 secondi di simulazione.

## 7.2 Confronto Throughput

Nello studio di riferimento vengono effettuati dei test sul throughput del webserver.

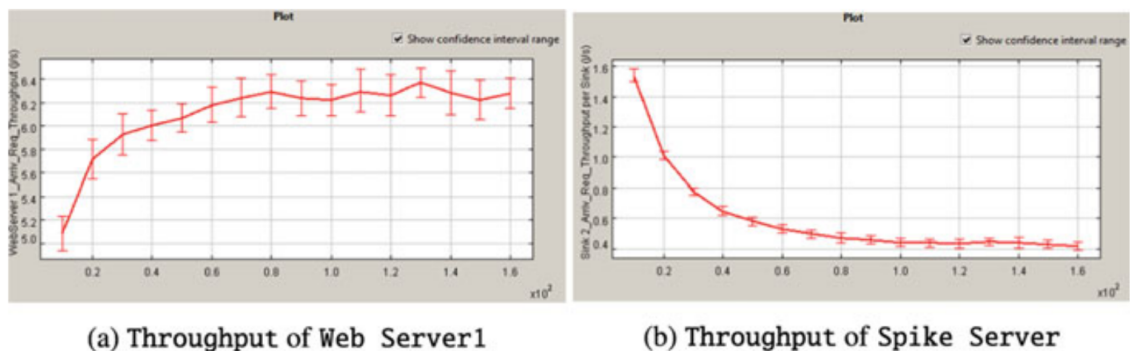


Figure 12: Throughput del webserver e dello spike server in funzione del carico nel caso di studio di Serazzi

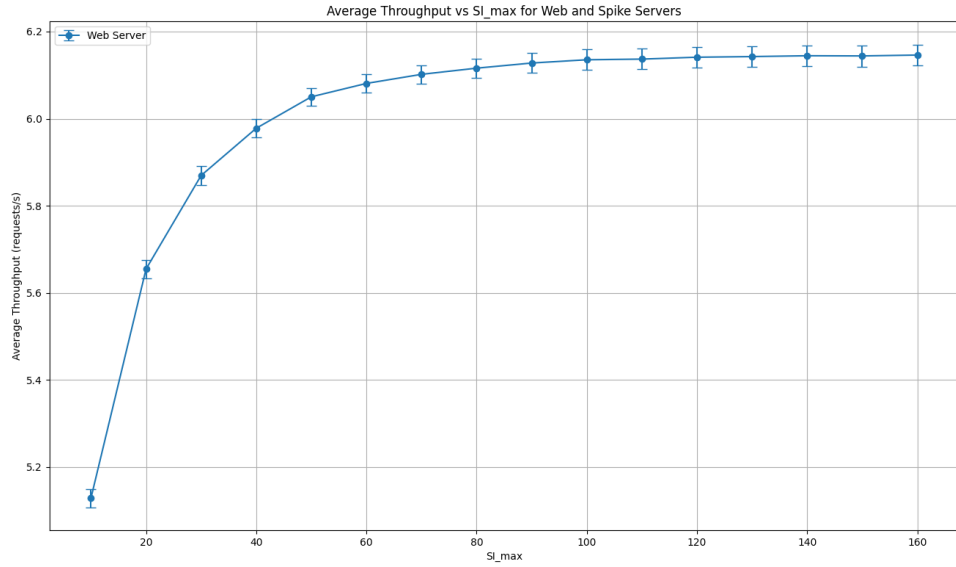


Figure 13: Throughput del webserver in funzione del SI\_max ottenuti dalla simulazione.

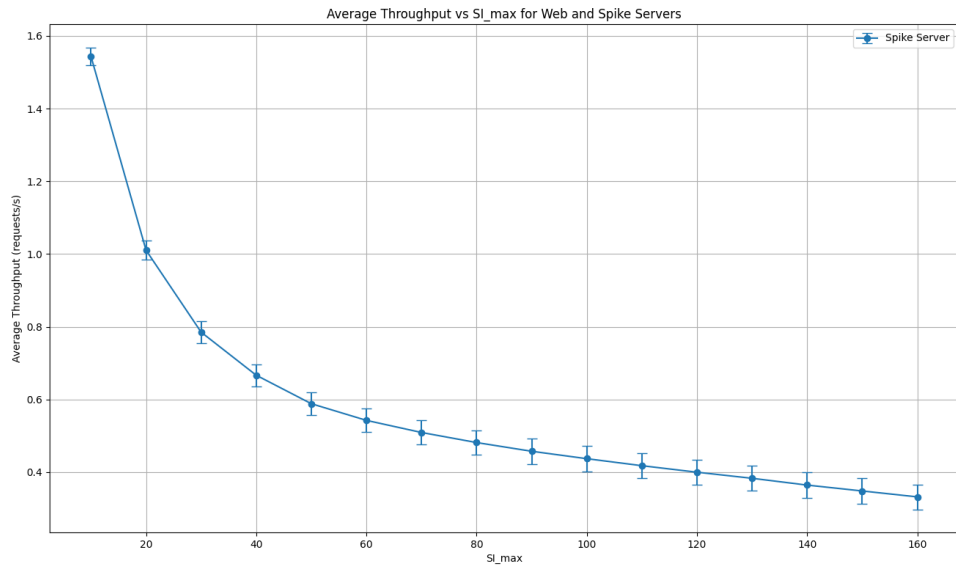


Figure 14: Throughput dello spike server in funzione del SI\_max ottenuti dalla simulazione.

L'andamento e i valori qui sono essenzialmente gli stessi. Purtroppo il caso di studio originale non fornisce i valori numerici precisi su cui ha costruito i grafici per confrontarli direttamente.

## 8 Risultati

Gli esperimenti condotti mirano a raggiungere gli obiettivi delineati nella sezione 2.

## 8.1 Obiettivo 1

Come si può evincere dalla figura 15, il tempo di risposta medio  $E[R]$  rimane al di sotto della soglia di 8 secondi per valori di  $SI_{max}$  fino a circa 90. Esattamente come suggerito nello studio di riferimento, il valore ottimale di  $SI_{max}$  che massimizza l'utilizzo del web server mantenendo il rispetto dello SLA si attesta intorno a 80-90. Considerando che con l'intervallo di confidenza con 90 si sfora, il valore più prudente da adottare per  $SI_{max}$  tra i due risulta essere 80.

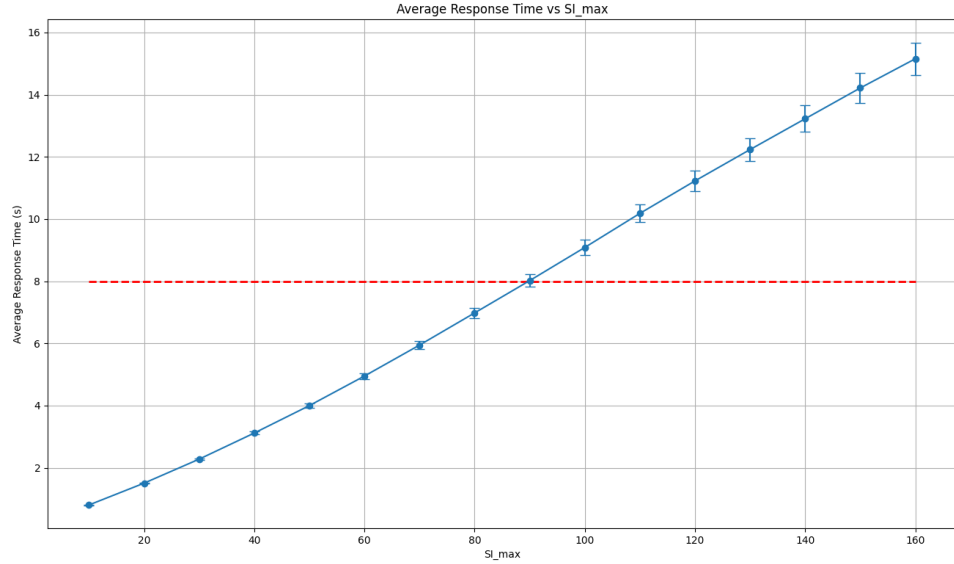


Figure 15: Tempo di risposta medio in funzione di  $SI_{max}$

Table 5: Tempo di risposta medio in funzione di  $SI_{max}$

$SI_{max}$	$E[T_s]$	$E[T_s]$ CI95
10	0.8053	0.0060
20	1.5063	0.0132
30	2.2786	0.0258
40	3.1180	0.0439
50	4.0000	0.0684
60	4.9507	0.0962
70	5.9467	0.1309
80	6.9813	0.1671
90	8.0204	0.2033
100	9.0866	0.2438
110	10.1823	0.2833
120	11.2269	0.3283
130	12.2347	0.3722
140	13.2254	0.4227
150	14.2151	0.4754
160	15.1546	0.5239

## 8.2 Obiettivo 2

Come mostrato in figura 16, accade lo stesso fenomeno descritto nello studio di riferimento: al crescere del tasso di arrivo, il tempo di risposta medio aumenta all'aumentare del tasso degli arrivi, fino a che non si arriva a saturare il sistema ( $\rho > 1$ ) e quindi a superare la soglia  $SI_{max}$ . A questo punto, il tempo di risposta comincia a calare fino a che anche lo spike server si satura e il tempo di risposta ricomincia a salire per valori di  $\lambda \geq 11$  nella maggior parte dei casi. Per valori molto bassi di  $SI_{max}$  il tempo di risposta continua a salire quasi sempre, questo accade perché essenzialmente è come se il webserver non venisse utilizzato e tutto il carico finisse sullo spike server, che quindi tende a saturarsi già per valori più bassi di  $\lambda$ . Nelle tabelle sottostanti (Tabella 6 e Tabella 9) sono riportati i valori numerici però mostrando solo i valori di  $SI_{max}$  a multipli di 20, questa scelta è dovuta alla grande quantità di dati generata. In ogni caso i dati sono consultabili nei file csv all'indirizzo <https://github.com/LucaFalasca/RightSizing-SpikeServer>.

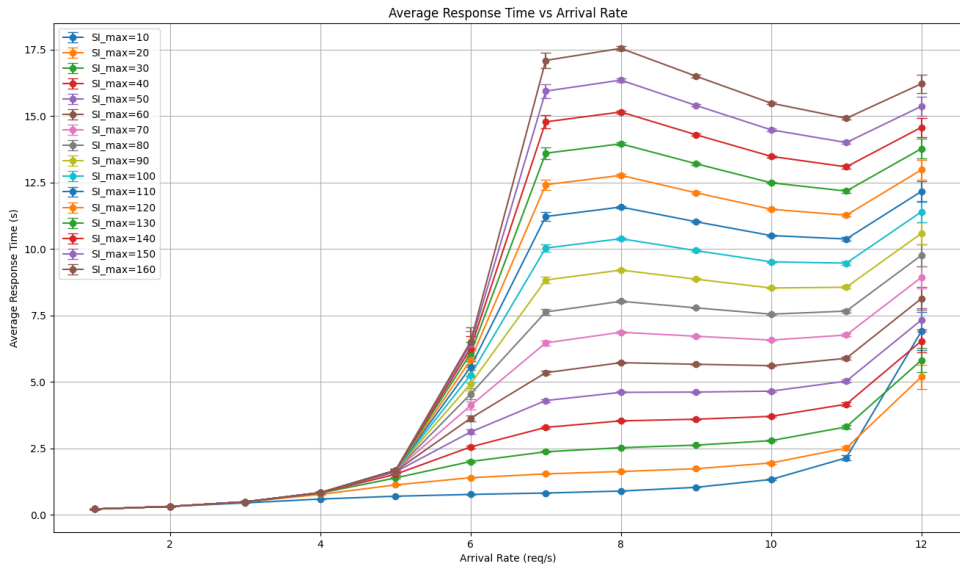


Figure 16: Tempo di risposta medio in funzione del tasso di arrivo e di  $SI_{max}$

Table 6: Tempo di risposta medio in funzione del tasso di arrivo e di  $SI_{max}$ 

SI max	$\lambda$	$E[T_s]$	$E[T_s]$ CI95	SI max	$\lambda$	$E[T_s]$	$E[T_s]$ CI95
20	1	0.2195	0.0038	100	1	0.2195	0.0038
20	2	0.3176	0.0053	100	2	0.3177	0.0053
20	3	0.4876	0.0099	100	3	0.4899	0.0111
20	4	0.7702	0.0126	100	4	0.8363	0.0197
20	5	1.1277	0.0163	100	5	1.6729	0.0580
20	6	1.3977	0.0158	100	6	5.2647	0.2931
20	7	1.5423	0.0123	100	7	10.0359	0.1499
20	8	1.6321	0.0111	100	8	10.3884	0.0433
20	9	1.7357	0.0140	100	9	9.9396	0.0280
20	10	1.9532	0.0320	100	10	9.5185	0.0286
20	11	2.5122	0.0685	100	11	9.4683	0.0585
20	12	5.1907	0.4602	100	12	11.3979	0.3965
40	1	0.2195	0.0038	120	1	0.2195	0.0038
40	2	0.3177	0.0053	120	2	0.3177	0.0053
40	3	0.4899	0.0111	120	3	0.4899	0.0111
40	4	0.8327	0.0185	120	4	0.8363	0.0197
40	5	1.5277	0.0391	120	5	1.6729	0.0580
40	6	2.5559	0.0532	120	6	5.8265	0.3824
40	7	3.2913	0.0396	120	7	12.4191	0.1921
40	8	3.5365	0.0226	120	8	12.7711	0.0496
40	9	3.5982	0.0174	120	9	12.1152	0.0353
40	10	3.7059	0.0273	120	10	11.4965	0.0340
40	11	4.1629	0.0642	120	11	11.2706	0.0602
40	12	6.5401	0.4375	120	12	12.9782	0.3712
60	1	0.2195	0.0038	140	1	0.2195	0.0038
60	2	0.3177	0.0053	140	2	0.3177	0.0053
60	3	0.4899	0.0111	140	3	0.4899	0.0111
60	4	0.8358	0.0194	140	4	0.8363	0.0197
60	5	1.6461	0.0525	140	5	1.6729	0.0580
60	6	3.6343	0.1164	140	6	6.2580	0.4695
60	7	5.3484	0.0756	140	7	14.7827	0.2410
60	8	5.7243	0.0319	140	8	15.1544	0.0592
60	9	5.6653	0.0209	140	9	14.2970	0.0419
60	10	5.6126	0.0262	140	10	13.4849	0.0394
60	11	5.8901	0.0594	140	11	13.0898	0.0575
60	12	8.1362	0.4258	140	12	14.5766	0.3567
80	1	0.2195	0.0038	160	1	0.2195	0.0038
80	2	0.3177	0.0053	160	2	0.3177	0.0053
80	3	0.4899	0.0111	160	3	0.4899	0.0111
80	4	0.8363	0.0197	160	4	0.8363	0.0197
80	5	1.6704	0.0576	160	5	1.6729	0.0580
80	6	4.5467	0.2029	160	6	6.5128	0.5297
80	7	7.6317	0.1148	160	7	17.0925	0.2905
80	8	8.0334	0.0378	160	8	17.5471	0.0714
80	9	7.7872	0.0240	160	9	16.4999	0.0509
80	10	7.5497	0.0255	160	10	15.4854	0.0454
80	11	7.6654	0.0564	160	11	14.9160	0.0544
80	12	9.7624	0.4081	160	12	16.2153	0.3499

### 8.3 Obiettivo 3

Nell'obiettivo 3 si va ad duplicare la potenza dello spike server, per verificare se questo evita il saturamento dello spike server e quindi il successivo aumento del tempo di risposta medio. Come mostrato in figura 17, lo spike server non arriva più a saturarsi e il tempo di risposta continua a diminuire all'aumentare del tasso di arrivo. I dati tabulari sono riportati nella Tabella 9.

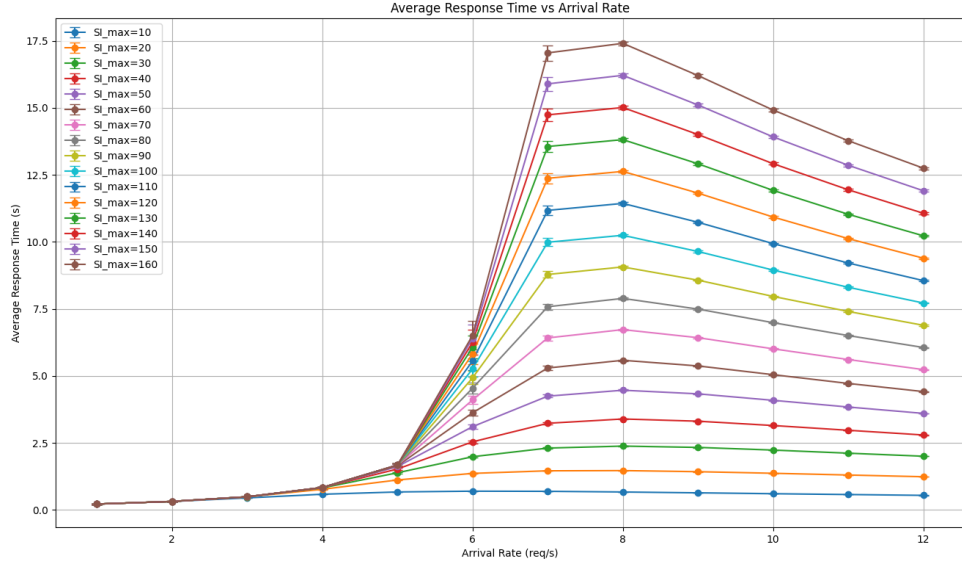


Figure 17: Tempo di risposta medio in funzione del tasso di arrivo e di  $SI_{max}$  con spike server potenziato

Table 7: Tempo di risposta medio in funzione del tasso di arrivo e di  $SI_{max}$  con spike server potenziato

SI max	$\lambda$	$E[T_s]$	$E[T_s]$ CI95	SI max	$\lambda$	$E[T_s]$	$E[T_s]$ CI95
20	1	0.2195	0.0038	100	1	0.2195	0.0038
20	2	0.3176	0.0053	100	2	0.3177	0.0053
20	3	0.4875	0.0099	100	3	0.4899	0.0111
20	4	0.7690	0.0126	100	4	0.8363	0.0197
20	5	1.1181	0.0158	100	5	1.6729	0.0580
20	6	1.3633	0.0146	100	6	5.2617	0.2926
20	7	1.4595	0.0104	100	7	9.9856	0.1477
20	8	1.4677	0.0074	100	8	10.2436	0.0414
20	9	1.4267	0.0058	100	9	9.6399	0.0287
20	10	1.3677	0.0045	100	10	8.9465	0.0284
20	11	1.3027	0.0035	100	11	8.3056	0.0251
20	12	1.2382	0.0032	100	12	7.7184	0.0227
40	1	0.2195	0.0038	120	1	0.2195	0.0038
40	2	0.3177	0.0053	120	2	0.3177	0.0053
40	3	0.4899	0.0111	120	3	0.4899	0.0111
40	4	0.8327	0.0185	120	4	0.8363	0.0197
40	5	1.5265	0.0390	120	5	1.6729	0.0580
40	6	2.5407	0.0523	120	6	5.8247	0.3820
40	7	3.2317	0.0374	120	7	12.3706	0.1900
40	8	3.3925	0.0201	120	8	12.6261	0.0482
40	9	3.3098	0.0130	120	9	11.8170	0.0362
40	10	3.1488	0.0104	120	10	10.9252	0.0350
40	11	2.9712	0.0078	120	11	10.1203	0.0312
40	12	2.7961	0.0075	120	12	9.3900	0.0282
60	1	0.2195	0.0038	140	1	0.2195	0.0038
60	2	0.3177	0.0053	140	2	0.3177	0.0053
60	3	0.4899	0.0111	140	3	0.4899	0.0111
60	4	0.8358	0.0194	140	4	0.8363	0.0197
60	5	1.6459	0.0525	140	5	1.6729	0.0580
60	6	3.6257	0.1156	140	6	6.2569	0.4693
60	7	5.2940	0.0735	140	7	14.7358	0.2390
60	8	5.5813	0.0292	140	8	15.0097	0.0583
60	9	5.3716	0.0177	140	9	14.0020	0.0434
60	10	5.0454	0.0153	140	10	12.9146	0.0416
60	11	4.7184	0.0131	140	11	11.9422	0.0379
60	12	4.4150	0.0121	140	12	11.0653	0.0341
80	1	0.2195	0.0038	160	1	0.2195	0.0038
80	2	0.3177	0.0053	160	2	0.3177	0.0053
80	3	0.4899	0.0111	160	3	0.4899	0.0111
80	4	0.8363	0.0197	160	4	0.8363	0.0197
80	5	1.6704	0.0576	160	5	1.6729	0.0580
80	6	4.5417	0.2023	160	6	6.5122	0.5296
80	7	7.5794	0.1126	160	7	17.0474	0.2883
80	8	7.8899	0.0357	160	8	17.4033	0.0705
80	9	7.4914	0.0227	160	9	16.2019	0.0534
80	10	6.9846	0.0218	160	10	14.9150	0.0497
80	11	6.5039	0.0192	160	11	13.7714	0.0447
80	12	6.0600	0.0175	160	12	12.7474	0.0402

## 9 Conclusioni

Da questo lavoro si ottiene conferma di quanto già evidenziato nello studio di Serazzi, il valore ottimale per  $SI_{max}$  si attesta intorno a 80-90 per il carico di lavoro di 6.66 req/s, che massimizza l'utilizzo del web server mantenendo il rispetto dello SLA sul tempo di risposta medio. Inoltre lo studio evidenzia che la capacità dello spike server va tarata in base al carico di lavoro atteso, perché altrimenti si rischia di saturare anche lui e vanificare l'efficacia del sistema.

Il codice del simulatore e tutti gli esperimenti sono disponibili al seguente link: <https://github.com/LucaFalasca/RightSizing-SpikeServer>



## A Analisi Transitorio

L'analisi del transitorio è stata effettuata per osservare l'andamento del tempo di risposta nel tempo, in modo da analizzare dopo quanto tempo il sistema raggiunge lo stato stazionario. Per fare ciò come negli esperimenti precedenti è stato utilizzato un unico seed = 8, un tempo di simulazione di 5000 secondi ma stavolta ovviamente senza eliminare la fase transitoria, che è proprio quella che ci interessa studiare. Il campionamento del tempo di risposta è stato effettuato ogni 100 secondi.

### A.1 Transitorio Obiettivo 1

Qui si mostra l'analisi del transitorio nel caso dell'Obiettivo 1, quindi con  $\lambda = 6.66req/s$  e  $SI_{max} = 80$ . Si vede che il tempo di risposta tende a stabilizzarsi intorno al valore di 1500 secondi, anche se continua a salire lentamente. Inoltre come si vede dai valori tabellari, l'intervallo di confidenza si stringe sempre di più, segno che il sistema sta raggiungendo uno stato stazionario.

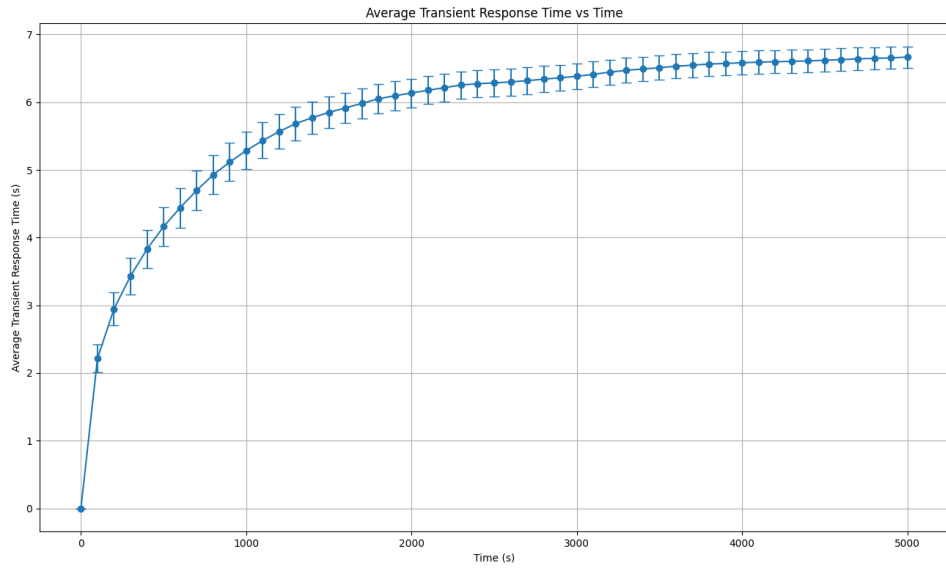


Figure 18: Andamento del tempo di risposta nel tempo per l'Obiettivo 1.

Table 8: Tempo di risposta medio in funzione del tempo per  $SI_{max} = 80$  e  $\lambda = 6.66req/s$ .

$t$	$E[T_s]$	$E[T_s]$ CI95
0	0.0000	0.0000
100	2.2141	0.2092
200	2.9470	0.2432
300	3.4307	0.2688
400	3.8307	0.2857
500	4.1619	0.2902
600	4.4418	0.2918
700	4.6968	0.2928
800	4.9273	0.2889
900	5.1153	0.2817
1000	5.2862	0.2729
1100	5.4343	0.2655
1200	5.5673	0.2574
1300	5.6856	0.2502
1400	5.7722	0.2401
1500	5.8500	0.2292
1600	5.9156	0.2227
1700	5.9836	0.2200
1800	6.0506	0.2174
1900	6.0957	0.2139
2000	6.1368	0.2102
2100	6.1771	0.2059
2200	6.2151	0.2028
2300	6.2549	0.2002
2400	6.2738	0.1998
2500	6.2850	0.1998
2600	6.3008	0.2005
2700	6.3189	0.1990
2800	6.3414	0.1964
2900	6.3609	0.1940
3000	6.3834	0.1916
3100	6.4128	0.1877
3200	6.4447	0.1836
3300	6.4720	0.1825
3400	6.4909	0.1818
3500	6.5107	0.1813
3600	6.5313	0.1801
3700	6.5483	0.1780
3800	6.5642	0.1761
3900	6.5733	0.1740
4000	6.5832	0.1729
4100	6.5922	0.1716
4200	6.5985	0.1703
4300	6.6043	0.1690
4400	6.6104	0.1672
4500	6.6205	0.1664
4600	6.6303	0.1661
4700	6.6412	0.1647
4800	6.6482	0.1632
4900	6.6552	0.1606
5000	6.6678	0.1587

## A.2 Obiettivo 2

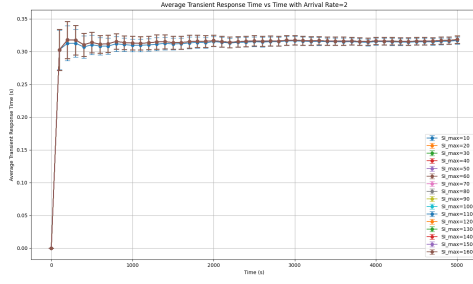
Sono state eseguite le analisi del transitorio per tutte le combinazioni di  $SI_{max}$  da 10 a 160 e tassi di arrivo da 1 req/s a 12 req/s. Qui esporrò solo le combinazioni più significative e rappresentative. In ogni caso, anche se sul report ne esporrò solo un sottoinsieme, tutte le immagini e tutte le tabelle con i dati sono disponibili nella cartella `docs/images` e `docs/data_tables` del progetto.

Dagli esperimenti si possono fare una serie di considerazioni:

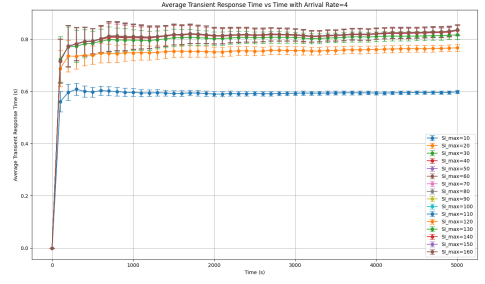
- Per i casi con arrival rate basso e alto la convergenza sembra arrivare molto velocemente (Figure 19a, 19b, 19e, 19f). Questo perché nel primo caso il sistema è poco carico e quindi riesce a smaltire velocemente le richieste, mentre nel secondo caso il sistema è sovraccarico e quindi il tempo di risposta cresce rapidamente fino a stabilizzarsi su un valore alto.
- I casi più interessanti sono quelli con arrival rate medio vicino alla saturazione del sistema (Figure 19c, 19d). In questi casi, in cui il sistema è carico ma con una utilizzazione tale per cui il web server è quasi sempre pieno e lo spike server viene attivato e disattivato frequentemente, il tempo di risposta impiega più tempo a stabilizzarsi, soprattutto per valori di  $SI_{max}$  alti. Questo comportamento è osservabile anche nelle figure 20e, 20f che mostrano lo stesso fenomeno da un'altra prospettiva.

## A.3 Obiettivo 3

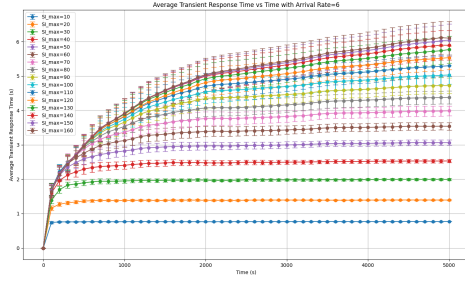
Per l'obiettivo 3 non sono state effettuate analisi del transitorio, in quanto ci si aspetta lo stesso comportamento dell'obiettivo 2, dato che l'unica differenza è la potenza dello spike server.



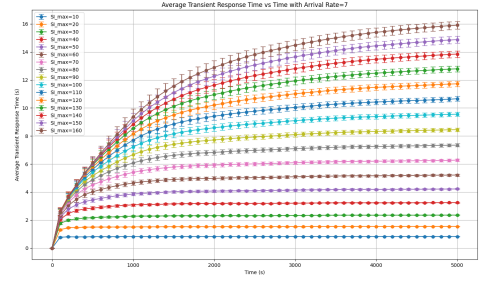
(a)  $\lambda = 2$



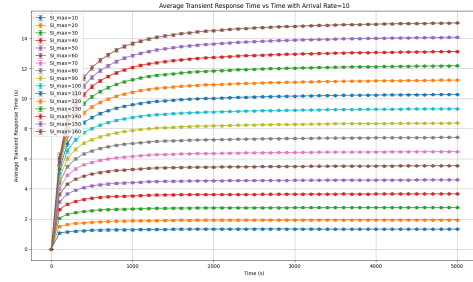
(b)  $\lambda = 4$



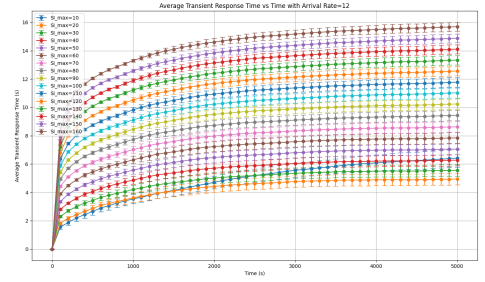
(c)  $\lambda = 6$



(d)  $\lambda = 7$

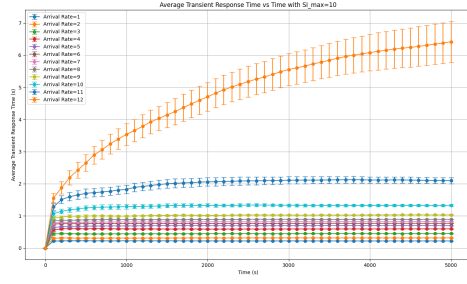


(e)  $\lambda = 10$

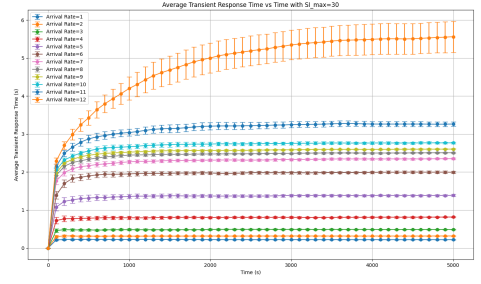


(f)  $\lambda = 12$

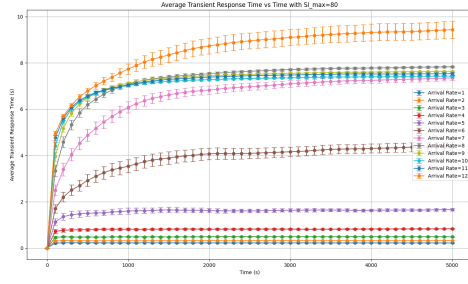
Figure 19: Andamento del tempo di risposta nel tempo per diversi tassi di arrivo  $\lambda$ .



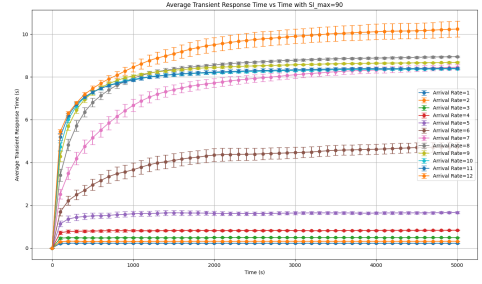
(a)  $SI_{max} = 10$



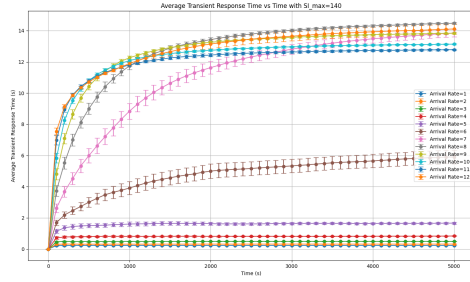
(b)  $SI_{max} = 30$



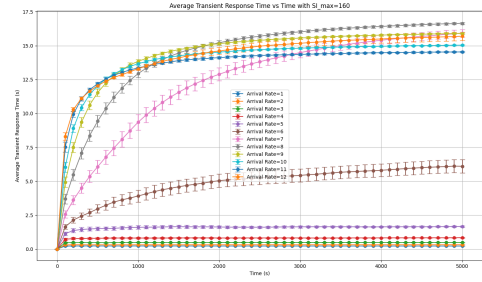
(c)  $SI_{max} = 80$



(d)  $SI_{max} = 90$



(e)  $SI_{max} = 140$



(f)  $SI_{max} = 160$

Figure 20: Andamento del tempo di risposta nel tempo per diverse soglie di  $SI_{max}$ .

Table 9: Tempo di risposta medio in funzione del tempo per alcune combinazioni di  $SI_{max}$  e  $\lambda$ .

$SI_{max} = 10 \wedge \lambda = 2$			$SI_{max} = 130 \wedge \lambda = 8$		$SI_{max} = 160 \wedge \lambda = 12$	
$t$	$E[T_s]$	$E[T_s]$ CI95	$E[T_s]$	$E[T_s]$ CI95	$E[T_s]$	$E[T_s]$ CI95
0	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
100	0.3026	0.0303	3.6561	0.3495	8.3071	0.2852
200	0.3126	0.0268	5.4231	0.3631	10.2838	0.1593
300	0.3126	0.0211	6.7939	0.3598	11.1085	0.1069
400	0.3069	0.0169	7.8612	0.3381	11.6463	0.0909
500	0.3103	0.0149	8.6487	0.3232	12.0472	0.0849
600	0.3083	0.0132	9.3069	0.3022	12.3772	0.0831
700	0.3082	0.0124	9.8843	0.2797	12.6542	0.0840
800	0.3118	0.0112	10.3280	0.2524	12.8572	0.0886
900	0.3105	0.0099	10.7088	0.2298	13.0786	0.0981
1000	0.3095	0.0097	11.0393	0.2119	13.3010	0.1118
1100	0.3096	0.0097	11.3066	0.1950	13.4913	0.1324
1200	0.3100	0.0094	11.5190	0.1800	13.6705	0.1559
1300	0.3111	0.0099	11.7050	0.1685	13.8374	0.1834
1400	0.3124	0.0096	11.8616	0.1591	13.9812	0.2031
1500	0.3114	0.0091	12.0102	0.1495	14.0942	0.2187
1600	0.3119	0.0089	12.1282	0.1421	14.2290	0.2315
1700	0.3131	0.0090	12.2334	0.1341	14.3329	0.2421
1800	0.3138	0.0084	12.3115	0.1274	14.4341	0.2513
1900	0.3133	0.0080	12.3909	0.1212	14.5298	0.2591
2000	0.3151	0.0077	12.4727	0.1155	14.6071	0.2676
2100	0.3140	0.0076	12.5484	0.1118	14.6846	0.2738
2200	0.3129	0.0073	12.6149	0.1070	14.7631	0.2784
2300	0.3138	0.0075	12.6808	0.1029	14.8281	0.2823
2400	0.3142	0.0074	12.7322	0.1001	14.8955	0.2858
2500	0.3150	0.0074	12.7762	0.0973	14.9600	0.2899
2600	0.3144	0.0072	12.8256	0.0947	15.0133	0.2914
2700	0.3150	0.0071	12.8696	0.0914	15.0522	0.2919
2800	0.3150	0.0068	12.9111	0.0891	15.0922	0.2929
2900	0.3161	0.0069	12.9474	0.0880	15.1340	0.2937
3000	0.3161	0.0067	12.9801	0.0865	15.1746	0.2945
3100	0.3157	0.0065	13.0096	0.0854	15.2182	0.2931
3200	0.3153	0.0062	13.0437	0.0836	15.2589	0.2917
3300	0.3155	0.0061	13.0709	0.0816	15.2945	0.2921
3400	0.3150	0.0059	13.0947	0.0803	15.3335	0.2934
3500	0.3150	0.0058	13.1258	0.0788	15.3706	0.2955
3600	0.3147	0.0058	13.1462	0.0768	15.3968	0.2977
3700	0.3150	0.0056	13.1683	0.0748	15.4177	0.2982
3800	0.3144	0.0054	13.1906	0.0733	15.4425	0.2998
3900	0.3140	0.0052	13.2047	0.0730	15.4678	0.3027
4000	0.3152	0.0053	13.2205	0.0723	15.4936	0.3054
4100	0.3150	0.0053	13.2398	0.0715	15.5212	0.3084
4200	0.3148	0.0054	13.2537	0.0701	15.5469	0.3104
4300	0.3145	0.0053	13.2715	0.0685	15.5684	0.3104
4400	0.3142	0.0053	13.2889	0.0683	15.5794	0.3107
4500	0.3147	0.0053	13.3060	0.0680	15.5934	0.3118
4600	0.3148	0.0053	13.3232	0.0671	15.6107	0.3123
4700	0.3147	0.0051	13.3364	0.0657	15.6345	0.3128
4800	0.3155	0.0051	13.3497	0.0642	15.6525	0.3138
4900	0.3158	0.0050	13.3568	0.0628	15.6718	0.3139
5000	0.3167	0.0054	13.3693	0.0621	15.6931	0.3142