Analisi di una coda M/M/1 + G secondo la politica FIFO gated service

Andrea Di Ubaldo

April 2023

1 Introduzione

Il seguente progetto ha l'obiettivo di analizzare un sistema single-server dotato di una sorgente che gestisce gli arrivi degli utenti da servire, una o più code dove questi ultimi attendono di essere serviti. Una volta in servizio termina la loro permanenza all'interno del sistema. Il modello di rete di coda viene esaminato analizzando le più importanti misure di prestazione prodotte nelle sue esecuzioni al fine di confrontarlo con i modelli in letteratura.

2 Tool utilizzati

La piattaforma utilizzata per la modellazione, la configurazione e la simulazione del sistema è **OMNET++ versione 6.0** [1]. Per la fase di pulizia, organizzazione e analisi di dati sono state usate le librerie **SciPy**, **NumPy** e **Pandas** [3] per il linguaggio di programmazione **Python 3.8** [2], sviluppate sull'IDE **Visual Studio Code**.

3 Modello

3.1 Descrizione

Il modello da implementare ha l'obiettivo di gestire un numero variabile di utenti (anche detti Job), ed ha la seguente struttura:

- Sorgente che rappresenta l'ingresso dei Job all'interno del sistema e che istanzia gli utenti secondo una data distribuzione di probabilità. Ad ogni utente in arrivo viene assegnato un tempo di servizio e una deadline secondo delle distribuzione di probabilità che verranno descritte in seguito.
- Router smista i Job da servire in una delle code casualmente.
- Code all'ingresso, un numero variabile da una a quattro, di tipo FIFO (first-in first-out) che possono ospitare un numero illimitato di Job in attesa di essere serviti.

- Server, il cui ruolo è visitare le code in cerca di utenti da servire in ordine ciclico (dalla prima all'ultima) e secondo la politica 'gated-service': il server serve tutti gli utenti presenti nella coda al momento del suo arrivo, gli utenti arrivati saranno eventualmente serviti durante la visita successiva. Un job deve iniziare il servizio prima della deadline, altrimenti viene eliminato dalla coda. Se una coda risulta vuota la momento della visita, il server passa a visitare la coda successiva. Il tempo per passare da una coda all'altra si presume essere nullo. Una volta in servizio l'utente termina la sua elaborazione.
- Sink che colleziona le statistiche sui job che sono stati serviti.

3.2 Parametri

Le simulazioni sul modello sono state eseguite su diversi parametri:

- **n**: numero di code interessate nelle simulazioni
- $\boldsymbol{\lambda}$: media della distribuzione esponenziale dei tempi di interarrivo
- $\boldsymbol{\mu}$: media della distribuzione esponenziale negativa dei tempi di servizio
- $[\boldsymbol{a}, \boldsymbol{b}]$: intervallo di valori della distribuzione uniforme della deadline

I valori assegnati a questi parametri sono:

- **n**: 2, 3, 4, 5
- $\boldsymbol{\lambda}$: 2.0, 1.4, 1.2, 1.0
- $\boldsymbol{\mu}$: 3.0
- $[\boldsymbol{a}, \boldsymbol{b}]$: [0.5, 3.0], [0.5, 5.0]

Vi è un totale di 24 configurazioni, identificate nel file omnetpp.ini con {config}-n{n}. *Config* (First, Second, Third, Fourth, Fifth, Sixth, Seventh, Eighth) identifica una delle 8 configurazioni di valori principali composte dall'i-esimo valore per ogni parametro del tempo di interarrivo per ogni valore della distribuzione di probabilità del tempo di scadenza(deadline). In questo modo le prime quattro configurazioni osservano la distribuzione dei tempi di deadline nell'intervallo del primo valore di $[a, b]$, le successive osserveranno la distribuzione nell'intervallo del secondo. Mentre n il valore di n che viene variato per ogni configurazione (il numero di code utilizzate). E.g.: First-n1 è la configurazione composta dal valore 1 di ogni parametro e le code utilizzate sono solo una.

4 Implementazione

Per l'implementazione sono stati utilizzati i moduli implementati dalla libreria *queueinglib*, apportando le opportune modifiche ai nodi *Source*, *PassiveQueue*, *Server*, *Sink*. L'implementazione del nodo *Router* della libreria non è stata interessata da alcuna modifica.

4.1 Source

Il modulo Source si occupa di generare i job ad ogni **interArrivalTime**. Ad ogni job viene assegnato un tempo di servizio e una scadenza con i rispettivi metodi prima di essere inviato al Router.

4.2 PassiveQueue

Il modulo *PassiveQueue* si occupa di mantenere in coda i Job generati dal Source. Il job in ingresso della coda viene mandato direttamente al Server se disponibile e se la coda risulta vuota (caso iniziale e ogni situazione in cui non ci sono Job in nessuna coda e il Server è libero), altrimenti viene messo in coda. Quando il Server ha terminato di servire il Job invoca il metodo *request* per richiedere uno dei Job oggetti della visita. Se questi viene richiesto dopo la scadenza viene eliminato dalla coda, altrimenti viene inviato al Server. Qualora non ci siano più Job da servire durante la visita, la coda invia un messaggio chiamato **noJobsLeft**. In tal modo il Server è certo di aver terminato i Job da visitare e dovrà effettuare una nuova visita in una delle code disponibili. Questa componente raccoglie anche le statistiche riguardo i Job scaduti prima di essere processati, utili per l'analisi del modello.

4.3 Server

Il Server si occupa di simulare un tempo di permanenza al proprio interno definito dal parametro **serviceTime**. Tale componente si occupa di effettuare una visita presso una delle code che ospitano Job in attesa al proprio interno. In seguito alla visita, la coda visitata invierà il primo Job non scaduto da servire. Il Server continuerà a servire quella coda finchè non avrà servito tutti gli utenti o non saranno scaduti in precedenza. Quando riceverà il messaggio **noJobsLeft** capisce che la visita è terminata e passerà a visitare una delle code, se disponibili. Il Server raccoglie anche le statistiche sul tempo di risposta del sistema relativo ad ogni Job servito, in modo da poter valutare il modello sotto questo aspetto.

4.4 Sink

Il Sink si occupa di distruggere i Job usciti dal sistema e raccogliere le statistiche sulla loro permanenza nel sistema necessari per effettuare l'analisi del modello.

5 Statistica

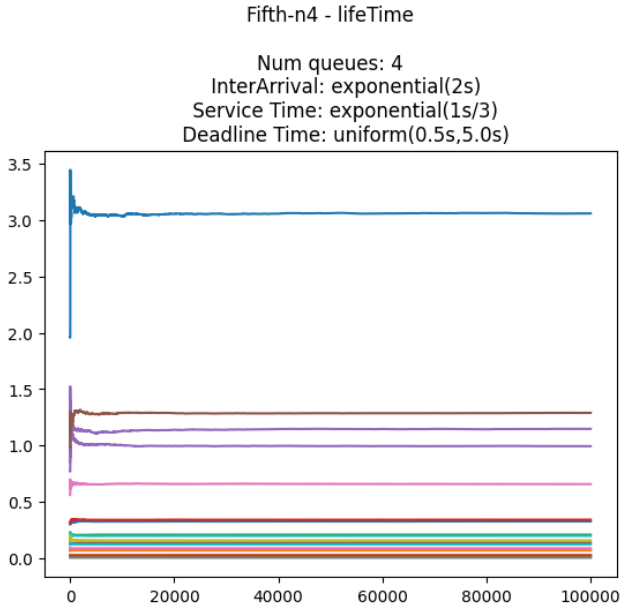
L'obiettivo era quello di simulare il modello, raccoglierne e analizzarne le statistiche al fine di convalidarne il funzionamento, in modo da poterlo allineare agli altri modelli analitici di reti a coda. Per ogni combinazione di configurazioni e valori di n effettuata sono stati calcolati le stime puntuali e relativi intervalli di confidenza:

- mediana della distribuzione del tempo di risposta del sistema
- tempo medio di permanenza nel sistema dei Job
- tempo massimo e minimo di permanenza nel sistema dei Job
- numero medio di utenti non serviti

Ogni combinazione (24) è stata ripetuta 20 volte, per un tempo complessivo di 100000 secondi. Per mostrare le statistiche richieste e i risultati del progetto si esaminerà la configurazione **Fourth**, poichè presenta l'intervallo della distribuzione uniforme della scadenza più ristretto e il tempo di interarrivo con media più alta, ma le stesse informazioni sono disponibili per ciascuna configurazione del modello.

5.1 Transiente iniziale

Nel momento in cui un sistema inizia da poco ad essere operativo, tale sistema è fortemente influenzato dallo stato iniziale e dal tempo trascorso prima dell'attivazione. Questa condizione del sistema è detta transitoria. Trascorso un lasso di tempo iniziale il sistema si stabilizza e raggiunge condizioni stazionarie. Durante gli esperimenti visualizzando i risultati ottenuti è stato notato un comportamento anomalo nella fase iniziale. Viene mostrata una configurazione d'esempio di questo comportamento:



È evidente che nelle prime fasi di simulazione il sistema è stabile. Valutando tutte le configurazioni si è deciso di eliminare questa fase e analizzare il sistema

dal momento in cui è a pieno regime. La fase transiente iniziale viene rimossa grazie alla funzionalità *wake-up time* di OMNeT++. La fase transiente individuata per questo sistema è di 10000 unità di tempo simulato, su un totale di 100000 unità di tempo della simulazione. Grafici relativi al comportamento del sistema con la fase di transiente iniziale possono essere trovate nelle cartelle:

- charts/charts_vector_results/lifetime

5.2 Risultati

Per verificare la non stabilità del sistema è stato applicato il Teorema di Little: $L = 1/\mu * W$ (dove μ è il tempo di interarrivo e W è il tempo medio di permanenza all'interno del sistema). Per un sistema stabile questo valore L deve essere tale che L ¡ 1. Possiamo notare come il sistema risulti stabile in tutte le configurazioni della simulazione.

Configurazione	μ	W	L
First-n1	2.0	0.433	0.216
First-n2	2.0	0.425	0.212
First-n4	2.0	0.421	0.210
Second-n1	1.4	0.474	0.338
Second-n2	1.4	0.457	0.326
Second-n4	1.4	0.45	0.321
Third-n1	1.2	0.498	0.415
Third-n2	1.2	0.475	0.395
Third-n4	1.2	0.465	0.387
Fourth-n1	1.0	0.53	0.53
Fourth-n2	1.0	0.496	0.496
Fourth-n4	1.0	0.483	0.483
Fifth-n1	2.0	0.475	0.237
Fifth-n2	2.0	0.456	0.228
Fifth-n4	2.0	0.448	0.224
Sixth-n1	1.4	0.545	0.389
Sixth-n2	1.4	0.507	0.362
Sixth-n4	1.4	0.491	0.35
Seventh-n1	1.2	0.586	0.488
Seventh-n2	1.4	0.535	0.445
Seventh-n4	1.4	0.513	0.427
Eighth-n1	1.0	0.647	0.647
Eighth-n2	1.0	0.572	0.572
Eighth-n4	1.0	0.542	0.542

Il modello oggetto della simulazione è un'implementazione della rete di coda M/M/1 con code FIFO e controllo di ammissione dei Job all'inizio del servizio. Il controllo accerta che Job inizia il servizio prima del suo tempo di scadenza. È possibile definire il loss ratio α come la limitata probabilità che un Job venga

rigettato al momento del controllo di ammissione. Numerosi studi indicano che il loss ratio si minimizza quando la distribuzione di probabilità dei tempi di scadenza è degenere [5]. In tutte le configurazioni del modello la distribuzione del tempo di scadenza è uniforme, qualora si utilizzasse una distribuzione degenere si dovrebbe riscontrare una diminuzione di Job eliminati a causa della scadenza.

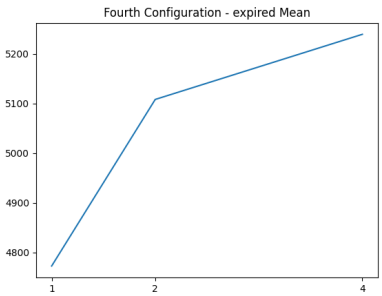


Figure 1: Job scaduti con distrib. uniforme

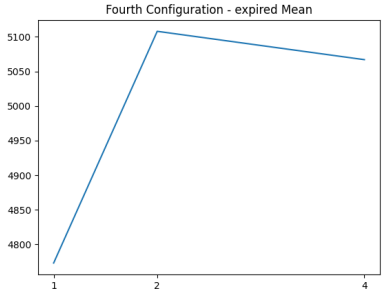


Figure 2: Job scaduti con distrib. degenere

Confrontando il numero di pacchetti persi di una simulazione con configurazione *Fourth* (una di quelle con più pacchetti persi) con la distribuzione uniforme $[a, b] = [0.5, 3.0]$ e quelli della stessa configurazione ma con distribuzione deterministica (ad esempio con costante 1.75, che risulta essere il valore della media della distribuzione in quell'intervallo), è possibile notare come il numero di pacchetti persi diminuisca.

Inoltre confrontando le distribuzioni del tempo di attesa dei Job in media sui 20 esperimenti della simulazione con differenti distribuzioni per il tempo di scadenza, è possibile notare che passando dalla distribuzione uniforme a quella degenere non ci siano rilevanti differenze. Si è scelto questo confronto dato che il tempo di attesa dei Job è una delle metriche più importanti da considerare in quanto a prestazioni delle reti di code FIFO con controllo di ammissione all'inizio del servizio [4]

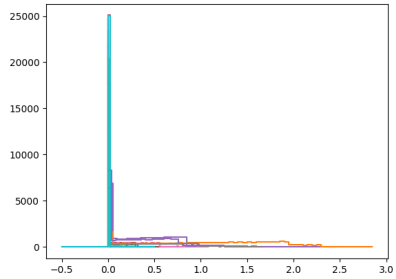


Figure 3: Coda 1 con distrib. uniforme

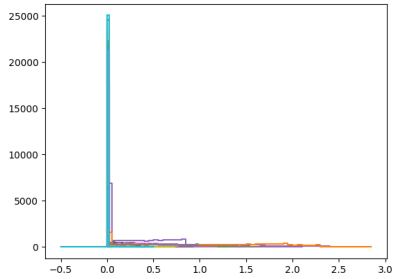


Figure 4: Coda 2 con distrib. uniforme

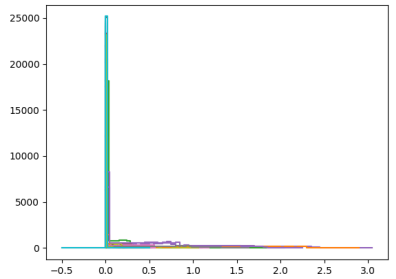


Figure 5: Coda 3 con distrib. uniforme

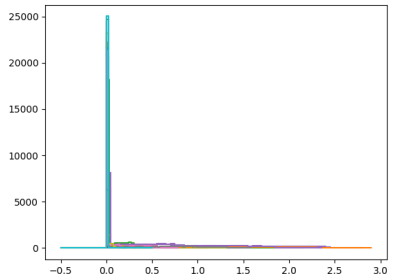


Figure 6: Coda 4 con distrib. uniforme

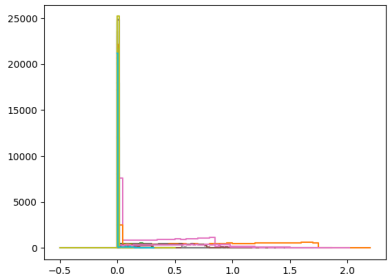


Figure 7: Coda 1 con distrib. degenere

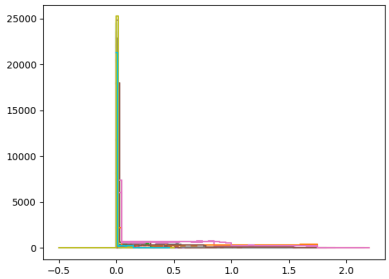


Figure 8: Coda 2 con distrib. degenere

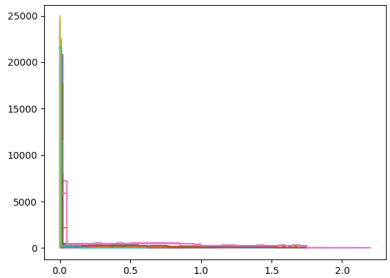


Figure 9: Coda 3 con distrib. degenere

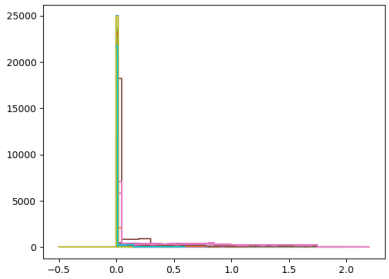


Figure 10: Coda 4 con distrib. degenere

5.3 Intervalli di confidenza

Partendo dai dati scalari di ogni configurazione raccolti ed estratti dalle simulazioni, sono stati calcolati gli intervalli di confidenza per ogni misura di prestazione richiesta. Per calcolare gli intervalli di confidenza, sono stati calcolati i seguenti:

- media del tempo tempo di permanenza medio per ogni Job di 20 run
- varianza (**var**)
- deviazione standard (**std_dev**)
- errore standard della media

Successivamente, con 19 gradi di libertà ($20 - 1 = 19$), i livelli di confidenza (90% e 95%), e i valori di α tali che:

- $a = 1 - 0.90 = 0.10 => Z(a/2) = 0.050$ per 90%
- $a = 1 - 0.95 = 0.05 => Z(a/2) = 0.025$ per 95%

I valori t per ciascun livello di confidenza risultano essere:

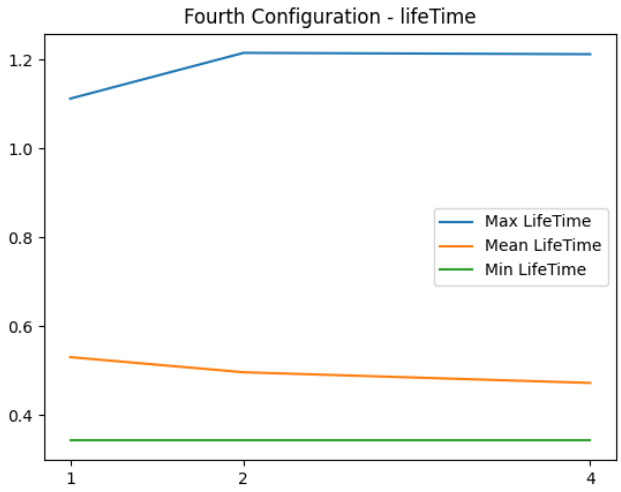
- $t90 = 1.729$
- $t95 = 2.093$

Infine per il calcolo degli intervalli di confidenza (limite superiore e inferiore) per t90 e t95:

- $Min\_95 = media - (t95 * \sqrt{var}/\sqrt{n})$
- $Max\_95 = media + (t95 * \sqrt{var}/\sqrt{n})$
- $Min\_90 = media - (t90 * \sqrt{var}/\sqrt{n})$
- $Max\_90 = media + (t90 * \sqrt{var}/\sqrt{n})$

5.4 Tempo di permanenza

Possiamo vedere il tempo medio, massimo e minimo di permanenza dei Job generati in ciascuna delle 20 simulazioni della configurazione **Fourth** (tempi di interarrivo esponenziali con media 1, quelli di servizio esponenziali negativi con media 3, e scadenza distribuita uniformemente nell'intervallo $[0.5s, 3.0s]$), al variare del numero di code utilizzate.



5.4.1 Intervalli di confidenza

Questi sono gli intervalli di confidenza del tempo medio, massimo e minimo di permanenza dei Job generati in ciascuna delle 20 run della configurazione **Fourth** (tempi di interarrivo esponenziali con media 1, quelli di servizio esponenziali negativi con media 3, e scadenza distribuita uniformemente nell'intervallo

$[0.5s, 3.0s]$) secondo i livelli di confidenza 95 e 90, al variare del numero di code utilizzate.

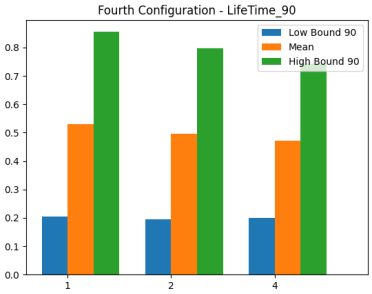


Figure 11: Confidence interval 90

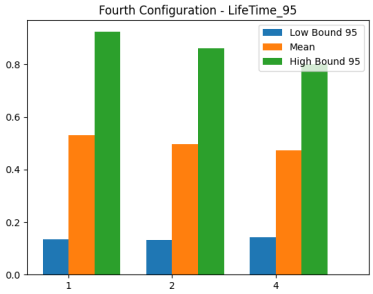
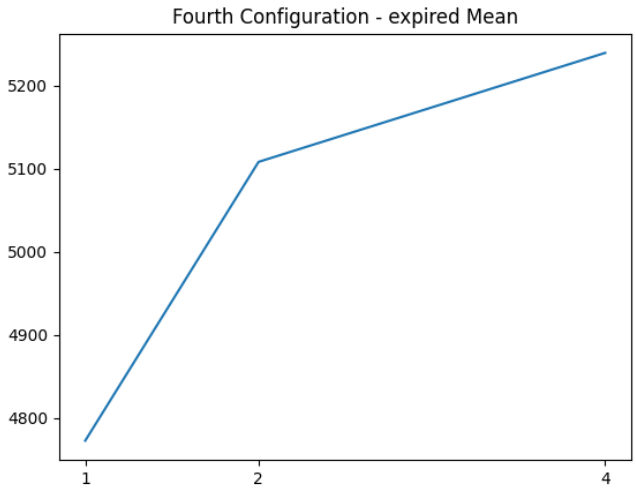


Figure 12: Confidence interval 95

5.5 Numero di Job scaduti

Di seguito la stima del numero di Job eliminati poichè scaduti prima di iniziare il servizio in ciascuna delle 20 simulazioni della configurazione **Fourth** (tempi di interarrivo esponenziali con media 1, quelli di servizio esponenziali negativi con media 3, e scadenza distribuita uniformemente nell'intervallo $[0.5s, 3.0s]$), al variare del numero di code utilizzate.



5.5.1 Intervalli di confidenza

Questi sono gli intervalli di confidenza dei Job scaduti in ciascuna delle 20 run della configurazione **Fourth** (tempi di interarrivo esponenziali con media 1, quelli di servizio esponenziali negativi con media 3, e scadenza distribuita uniformemente nell'intervallo $[0.5s, 3.0s]$) secondo i livelli di confidenza 95 e 90, al variare del numero di code utilizzate.

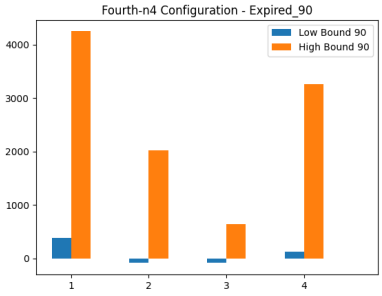


Figure 13: Confidence interval 90

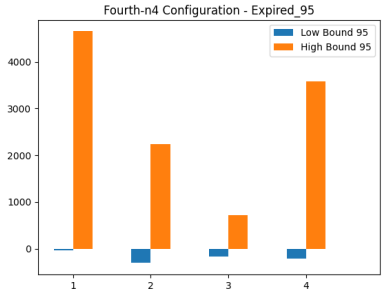


Figure 14: Confidence interval 95

Possiamo notare come il limite inferiore di quasi tutti gli intervalli di confidenza sia negativo. Questo è dovuto all'eccessiva varianza del numero di Job scaduti nelle 20 run di ciascuna configurazione.

5.6 Mediana della distribuzione del tempo di risposta e relativi intervalli di confidenza

Infine la mediana della distribuzione del tempo di risposta del sistema in ciascuna delle 20 simulazioni della configurazione **Fourth** (tempi di interarrivo esponenziali con media 1, quelli di servizio esponenziali negativi con media 3, e scadenza distribuita uniformemente nell'intervallo $[0.5s, 3.0s]$), al variare del numero di code utilizzate (con i relativi intervalli di confidenza).

6 Plotting e processing dei Dati

Tutti i dati prodotti da OMNeT++ sono stati salvati su file di tipo *.vec, *.sca e *.vci. Tramite *opp_scavetool* sono stati estratti sia i vettori di tutte le osservazioni simulate, sia i valori scalari generati dal sistema. Gli script utilizzati per l'estrazione si trovano al path *./StatisticalAnalysis/create_*.sh*.

I file generati sono stati analizzati successivamente tramite degli script in Python presenti nella cartella *./StatisticalAnalysis/plot_*.py*. Gli script si occupano di calcolare e generare i grafici per mostrare le statistiche richieste.

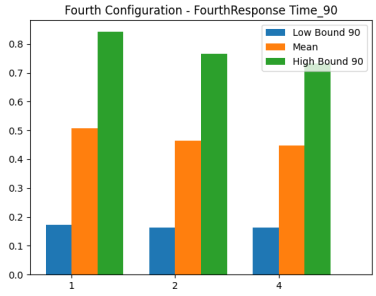


Figure 15: Confidence interval 90

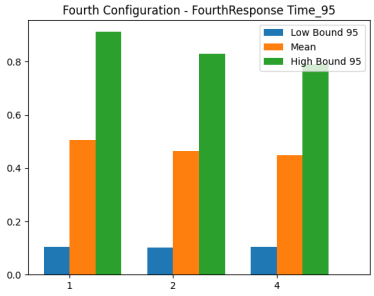


Figure 16: Confidence interval 95

References

[1] *OMNeT++ Simulation Manual v6.0.*

[2] *Python.*

[3] *Result analysis with Python.*

[4] J. W. Cohen. Single server queues with restricted accessibility. *Journal of Engineering Mathematics*, 1969.

[5] Debasis Sengupta Sudipta Das, Lawrence Jenkins. Analysis of an m/m/1 + g queue operated under the fcfs policy with exact admission control. *Queueing Systems*, 2013.