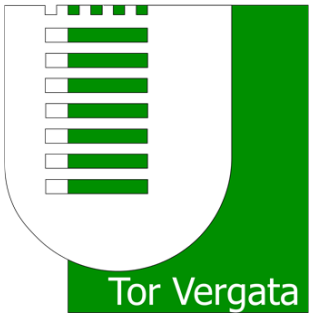


Università degli Studi di Roma "Tor Vergata"

Dipartimento di Ingegneria Civile e Ingegneria Informatica

Corso di

Performance Modeling of Computer Systems and Networks

Analisi delle Prestazioni e Capacity Planning di un Sistema di Videosorveglianza Biometrica Edge-Cloud

Aprile Carmine

0367475

Anno Accademico 2024/2025

Indice

1. Introduzione

Il caso di studio analizza la simulazione e la valutazione delle prestazioni di un sistema di videosorveglianza intelligente per la sicurezza aeroportuale. Il sistema effettua la scansione in tempo reale dei passeggeri in transito per l'identificazione in tempo reale di profili considerati sospetti o pericolosi, basandosi su un'architettura di calcolo distribuito Edge-Cloud.
Il sistema è ispirato alla struttura descritta nel testo “Performance Engineering” di G. Serazzi (Cap. 6).

Motivazioni e Obiettivi

L'obiettivo principale di questo studio è la progettazione e lo sviluppo di un modello di simulazione ad eventi discreti (DES) atto a valutare la scalabilità e l'affidabilità di un'infrastruttura di videosorveglianza biometrica. In un contesto critico come quello aeroportuale, la capacità di garantire tempi di risposta certi per l'identificazione di soggetti potenzialmente pericolosi è un requisito non funzionale imprescindibile.

Attraverso lo sviluppo di un simulatore sviluppato in Python, si intendono perseguire i seguenti obiettivi:

- **Validazione delle Prestazioni:** verificare che il sistema operi entro soglie di latenza accettabili in condizioni di traffico nominale
- **Stress Test:** identificare i punti di saturazione dell'architettura al crescere del flusso di passeggeri, quantificando l'impatto del carico sul tempo di risposta medio
- **Valutazione di Soluzioni Migliorative:** simulare scenari di upgrade hardware per determinare una configurazione ottima capace di mitigare eventuali fenomeni di congestione individuati nelle fasi di stress test

2. Descrizione del Sistema

Il sistema oggetto di studio è un’infrastruttura di videosorveglianza, progettata per il monitoraggio automatizzato al fine di garantire la sicurezza globale all'interno di un aeroporto. L'architettura è finalizzata all'identificazione in tempo reale di passeggeri appartenenti a diverse categorie di rischio (es. *suspect*, *dangerous*), monitorando i soggetti in ogni area della struttura: dai gate d'imbarco alle scale mobili, fino alle zone di attesa e le aree duty-free, mediante l’analisi biometrica dei volti catturati dalle telecamere disposte nell’intera struttura aeroportuale.

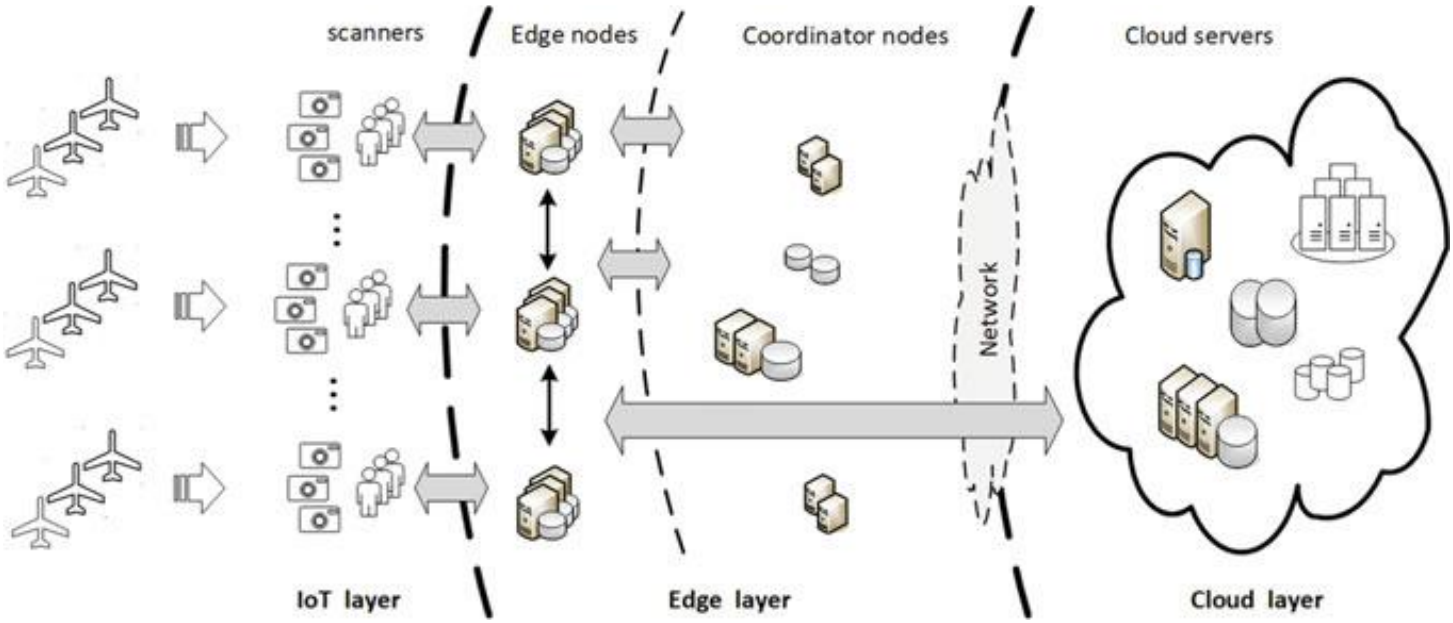
Architettura del Sistema

Il sistema si basa su una gerarchia di calcolo a tre livelli per gestire la mole di dati proveniente dalla scansione dei tratti biometrici:

- **Livello IoT :** rappresentato dalle telecamere disposte all’interno dell’aeroporto
- **Livello Edge:** l'aeroporto è suddiviso in zone, ognuna gestita da un Server Edge posizionati localmente all’infrastruttura aeroportuale . Ognuno dei Server Edge riceve le immagini dei volti catturati da un totale di 28 telecamere. Il compito del Server Edge è effettuare l’analisi dei volti e confrontarli con un database locale. Se il volto non presenta criticità ossia viene identificato con certezza come appartenente a categorie note (*poor-quality image, regular, suspect, dangerous*), l'operazione di analisi si conclude localmente.

Il livello Edge inoltre include i Nodi Coordinatori che ricevono l'esito dell'analisi dei dati biometrici di un passeggero, e nel caso venga classificato come individuo pericoloso si occupano di allertare il personale di sicurezza e di tracciare tramite l'ausilio delle telecamere i movimenti del passeggero.

- **Livello Cloud:** quando l'analisi locale non è sufficiente a classificare il soggetto, l'immagine analizzata ricade nella categoria **"unknown"**. In questo caso, i dati biometrici vengono inoltrati a un'infrastruttura Cloud remota dotata di grandi database e algoritmi di riconoscimento biometrico avanzati, capaci di eseguire operazioni di analisi più approfondite.



Flusso Operativo dei Dati

Ogni qualvolta una telecamera cattura l'immagine di un passeggero, il sistema attiva una procedura di riconoscimento che può seguire due percorsi distinti a seconda della complessità riscontrata nella fase di analisi del soggetto:

1. **Identificazione Locale (Edge Nodes):** la maggior parte dei passeggeri viene identificata tramite algoritmi di confronto rapido eseguiti direttamente sui nodi Edge. Se il volto non presenta criticità, l'elaborazione termina qui.

2. **Identificazione Remota (Cloud Server):** in una percentuale significativa di casi, l'analisi locale non produce un match univoco (categoria *unknown*). I dati biometrici vengono quindi inoltrati al livello Cloud per un'analisi approfondita. Il server remoto esegue algoritmi di riconoscimento avanzati.

3. **Sincronizzazione e Aggiornamento (Cloud-to-Edge):** una volta completata l'analisi remota, il Cloud invia il risultato al nodo Edge originario. Questo passaggio è critico: il nodo Edge provvede ad aggiornare il database locale con le nuove informazioni ottenute dal Cloud Layer. In questo modo, se il passeggero dovesse essere inquadrato da un'altra telecamera, il nodo Edge sarà in grado di identificarlo immediatamente senza interpellare nuovamente il Cloud.

4. **Feedback e Azione (Edge-to-Coordinators):** dopo aver aggiornato i registri locali, il nodo Edge invia l'esito finale ai Nodi Coordinatori. Se il soggetto è classificato come pericoloso, i Coordinatori attivano le procedure di sicurezza, allertano il personale e coordinano il tracciamento dinamico tramite il Livello IoT.

Requisiti di Quality of Service (QoS)

Per garantire l'efficacia della sicurezza aeroportuale, il sistema deve rispettare un requisito prestazionale stringente:

- **Analisi Locale in Tempo Reale:** la fase di analisi delle immagini presso i nodi Edge deve concludersi entro un tempo massimo di 3 secondi. Questo vincolo è dettato dalla dinamica del movimento dei passeggeri: un tempo superiore impedirebbe un intervento tempestivo del personale di sicurezza o il tracking continuo del soggetto tra diverse telecamere. Il soddisfacimento di questo requisito è il principale indicatore di qualità del servizio (*Quality of Service - QoS*).

3. Obiettivi dello Studio

L'attività di analisi è stata strutturata attorno a quattro obiettivi principali, volti a validare l'architettura attuale e a pianificare eventuali interventi migliorativi.

1. **Valutazione della QoS per i Job di Classe E**
   L'obiettivo consiste nel verificare la capacità del sistema di sostenere il carico di lavoro nominale ($\lambda = 1.4$ job/s). Nello specifico, si intende accertare che il Tempo di Risposta per le scansioni biometriche di Classe E (gestite esclusivamente dal nodo Edge) non superi il valore critico di **3 secondi**, limite oltre il quale l'identificazione perderebbe di efficacia operativa.

2. **Valutazione della QoS per i Job di Classe C**
   Si valuta se l'interazione con il livello Cloud garantisca tempi di risposta entro la soglia dei 5 secondi per i Job di Classe C. Sebbene questo parametro non sia un vincolo stringente come il precedente, esso rappresenta un indicatore fondamentale per la tempestività dei meccanismi di tracciamento e per l'allerta del personale di sicurezza in caso di soggetti pericolosi.

3. **Analisi di Scalabilità all'Aumentare del Carico**
   Si studia il comportamento del sistema a fronte di incrementi progressivi del tasso di arrivo (+10%, +20% e +30%), simulando scenari di picchi di traffico passeggeri. L'indagine mira a stabilire il "punto di rottura" dell'infrastruttura attuale, individuando se e quando

l'incremento del carico causi la violazione della soglia di 3s per la Classe E, rendendo necessario un potenziamento delle risorse.

4. **Analisi Comparativa dell'impatto di Edge Scaling**
   L'ultimo obiettivo valuta l'efficacia di un potenziamento delle risorse di calcolo locali come contromisura al degrado prestazionale rilevato nello scenario di massimo stress (+30% del tasso di arrivo). Nello specifico, la ricerca non si limita a un singolo intervento, ma analizza tre distinti scenari di upgrade hardware, che consentono una riduzione dei tempi di servizio del nodo Edge rispettivamente del 10%, 20% e 30% rispettivamente. L'indagine mira a individuare l'investimento hardware minimo necessario tra quelli proposti per riportare i QoS sui tempi di risposta sotto la soglia di sicurezza stabilita, garantendo la stabilità operativa del sistema anche a fronte di un incremento critico dei flussi passeggeri.

4. Modello Concettuale

Assunzioni Semplificative
Per rendere il modello facilmente trattabile, sono state adottate le seguenti assunzioni:

- **Omogeneità degli Arrivi:** si assume che il flusso totale dei passeggeri sia distribuito in modo uniforme tra tutti i nodi Edge dell'aeroporto. Tale ipotesi di simmetria permette di isolare e analizzare un singolo nodo Edge (e il relativo sottoinsieme di 28 telecamere) come campione statisticamente rappresentativo dell'intero sistema aeroportuale.

- **Tempi di Trasmissione Trascurabili:** si assume che il ritardo introdotto dall'infrastruttura di rete per il trasferimento dei dati tra i nodi Edge e il Cloud sia trascurabile rispetto ai tempi di elaborazione computazionale. Il tempo di risposta calcolato coinciderà quindi con il solo tempo di residenza nelle stazioni di calcolo.

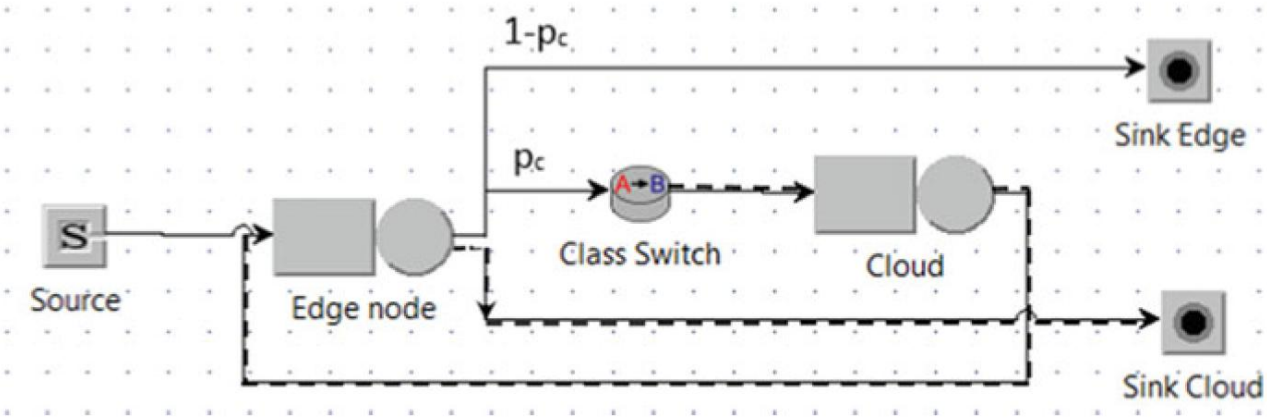
- **Trasparenza dei Nodi Coordinatori:** i nodi coordinatori, pur essendo fondamentali nell'architettura reale per la gestione degli allarmi, sono considerati "trasparenti" ai fini del calcolo delle prestazioni.

Struttura del Modello
Il sistema è rappresentato concettualmente come una rete di code aperta composta da due stazioni principali.

La struttura riflette la gerarchia Edge-Cloud ed è definita come segue:
- **Sorgente:** rappresenta il flusso in ingresso delle scansioni biometriche generate dal livello IoT
- **Stazione Edge:** riceve la totalità dei flussi provenienti dal livello IoT e l'intero traffico in uscita dalla stazione Cloud
- **Stazione Cloud:** riceve solo una frazione del traffico in uscita dalla stazione Edge

- **Class-Switch (Stazione Logica):** non esegue servizio, la sua unica attività consiste nel cambiare la classe di appartenenza dei Job che la attraversano e reindirizzarli verso la destinazione successiva (operazione istantanea)
- **Uscita (Sink)**: rappresenta il completamento delle attività e la fuoriuscita del job dal sistema, in particolare sono stati identificati due Sink differenti uno per i Job di classe E ed uno per quelli di classe C (rispettivamente Sink Edge e Sink Cloud)

Si tratta di un modello a "stadi", dove il completamento dell'analisi può avvenire dopo il primo stadio o richiedere l'attraversamento del secondo, a seconda della natura della richiesta.



Eventi del Sistema
All'interno del modello concettuale, sono stati identificati i seguenti eventi critici:

- **Arrivo di una nuova richiesta (Arrival):** questo evento rappresenta l'ingresso di un nuovo job nel modello. All'istante dell'arrivo, viene immediatamente immesso nel nodo Edge per iniziare la prima fase di elaborazione

- **Completamento del Servizio Edge (Departure Edge):** si verifica quando la risorsa Edge termina di elaborare un Job. È l'evento che conclude sia la fase di identificazione iniziale, sia l'eventuale fase di aggiornamento del database locale (per i job di ritorno dal Cloud).

- **Completamento del Servizio Cloud (Departure Cloud):** questo evento si verifica quando la risorsa Cloud termina l'elaborazione di un job di Classe C

- **Routing + Class Switch:** rappresenta il momento esatto in cui un job, dopo essere stato servito dal nodo Edge, determina la propria classe e di conseguenza il proprio percorso futuro all'interno del modello. Anche se concettualmente istantaneo, è il punto di snodo fondamentale per la rappresentazione del comportamento del sistema.

Variabili di Stato

Per definire univocamente la condizione del sistema in un istante t, è stato identificato un insieme di variabili che ne descrivono lo stato operativo. Tali variabili permettono di conoscere, in ogni momento, il carico di lavoro distribuito sulle risorse fisiche e la fase in cui si trovano le entità.
Il vettore di stato S(t) è composto da:

- **Popolazione delle Stazioni di Servizio** $\left(N_{Edge}(t), N_{Cloud}(t)\right)$**:** variabili intere che indicano il numero di Job presenti rispettivamente nel nodo Edge e nel nodo Cloud

- **Class-ID (E o C)** associato a ciascun Job presente nel sistema

- **Fase operativa del Job:** ogni job nel sistema può trovarsi ad eseguire una delle seguenti opzioni:
  - Identificazione Locale
  - Identificazione Cloud
  - Aggiornamento DB locale

- **Remaining Service Time (r_j):** per ogni job j in fase di servizio si definisce r_j come la quantità di tempo di calcolo ancora necessaria al completamento del compito

Caratterizzazione degli Eventi

L'evoluzione dinamica del sistema è governata da un insieme di eventi discreti. Ogni evento provoca una transizione di stato istantanea. Gli eventi identificati sono:

1. **Arrivo di un Job (Classe E):** un nuovo passeggero entra nell'area monitorata
   - *Transizione:* $n_E \rightarrow n_E + 1$

2. **Fine Servizio Edge (Job Classe E):** un job completa l'analisi locale. Possono verificarsi due sotto-eventi:
   - **Uscita (Successo):** il job viene identificato ed esce dal sistema
     - *Transizione:* $n_E \rightarrow n_E - 1$
   - **Inoltro Cloud (Unknown):** L'analisi fallisce e il job viene inviato al Cloud cambiando classe.
     - *Transizione:* $n_E \rightarrow n_E - 1$ e $n_C \rightarrow n_C + 1$

3. **Fine Servizio Cloud (Job Classe C):** il server remoto termina l'elaborazione e reinvia il job all'Edge per l'aggiornamento.
   - *Transizione:* $n_C \rightarrow n_C - 1$ *e* $n_E \rightarrow n_E + 1$

4. **Fine Servizio Edge (Job Classe C):** il server locale termina l'operazione di feedback o aggiornamento.
   - *Transizione:* $n_E \rightarrow n_E - 1$

5. Modello di Specifica

Adesso il modello concettuale viene trasformato in un modello di Specifica, andando a definire i parametri numerici e le distribuzioni stocastiche che definiscono il sistema.

In questo capitolo tutti i parametri numerici sono stati ricavati dal caso di studio del testo "*Performance Engineering*" di G. Serazzi.

5.1 Parametri del Modello

Tasso di Arrivo delle Scansioni Biometriche

Gli arrivi delle scansioni biometriche sono modellati come un Processo di Poisson. Ciò implica che gli inter-arrivi seguano una distribuzione Esponenziale, caratterizzata dalla totale assenza di memoria, scelta ideale per modellare sorgenti IoT indipendenti.

I tassi di arrivo che verrano usati durante l'approfondimento degli obiettivi dello studio sono indicati nella seguente tabella.

Scenario	λ (scansioni/s)
Tasso di Arrivo Base	1.4
Tasso di Arrivo +10%	1.54
Tasso di Arrivo +20%	1.68
Tasso di Arrivo +30%	1.82

Tempi di Servizio

Tutti i nodi di servizio sono stati modellati come delle code M/M/1–PS, quindi i tempi di servizio sono esponenziali.

Le seguenti tabelle mostrano i Tempi di Servizio medio che i job richiedono per i nodi Edge e Cloud all'interno degli scenari considerati negli obiettivi studiati.

Scenario Server Edge Base

Resource (Station)	Class of requests	
	E	*C*
Edge Node (Base)	0.5	0.1
Cloud Server	-	0.8

Scenario potenziamento del 10% della capacità di calcolo del Nodo Edge

Resource (Station)	Class of requests	
	E	*C*
Edge Node (+10%)	0.45	0.09
Cloud Server	-	0.8

Scenario potenziamento del 20% della capacità di calcolo del Nodo Edge

Resource (Station)	Class of requests	
	E	*C*
Edge Node (+20%)	0.4	0.08
Cloud Server	-	0.8

Scenario potenziamento del 30% della capacità di calcolo del Nodo Edge

Resource (Station)	Class of requests	
	E	*C*
Edge Node (+30)	0.35	0.07
Cloud Server	-	0.8

Probabilità di Routing $(\boldsymbol{p_C})$

La probabilità che un job venga classificato come Classe C dopo il primo passaggio all'Edge viene considerata pari a $p_C = 0.4$

6. Modello Computazionale

Il modello di specifica è stato a questo punto trasformato in un modello computazionale tramite il quale è possibile effettuare una simulazione a eventi discreti basata sull'approccio Next-Event Simulation, in cui il tempo di simulazione non avanza in modo uniforme, ma progredisce saltando direttamente all'istante di occorrenza del prossimo evento cronologicamente più vicino nella lista degli eventi futuri.

Per l'implementazione del modello computazionale è stato utilizzato il linguaggio Python.

Strutture Dati

Descriviamo le principali strutture dati e classi utilizzate nel software:

- **Classe Job**: rappresenta la singola unità di lavoro nel sistema
  - current_class: identifica la classe attuale del Job ('E' o 'C')
  - birth: istante di ingresso nel sistema
  - finish: istante di uscita definitiva dal sistema
  - remaining: tempo di servizio residuo richiesto nel server attuale

- **Classe ps_server:** rappresenta un server con politica di servizio Processor-Sharing
  - name: nome che identifica il server
  - jobs: lista dei job attualmente in fase di elaborazione nel server
  - last_t: l'ultimo istante temporale in cui il server ha aggiornato il proprio stato
  - version: contatore incrementale usato per validare gli eventi di completamento ed evitare il processamento di eventi obsoleti causati da nuovi arrivi o partenze
  - busy_time_post: contatore usato per il calcolo dell'utilizzazione del server
  - area_n_post: contatore usato per il calcolo del numero medio di utenti nel server
  - I metodi principali sono:

        - process_arrival: gestisce l'ingresso di un nuovo job nel server
        - update_progress: aggiorna gli accumulatori di area e tempo in funzione del tempo trascorso dall'ultimo evento
        - next_departure_time: considerando l'attuale ripartizione della capacità di esecuzione del server, calcola l'istante in cui il prossimo job lascia il server
        - process_completion: gestisce l'uscita di un Job dal server (dopo averne verificato la validità dell'evento)

- **Classe Scheduler:** gestisce la coda degli eventi futuri. La struttura è basata su una heap binaria (heapq), che permette l'inserimento e l'estrazione dell'evento imminente con complessità O(log n), ottimizzando le prestazioni della simulazione.
  I metodi fondamentali per la gestione della Heap sono:
    - **schedule(event)**: inserisce un nuovo evento nella heap tramite heappush()
    - **next_event()**: estrae l'evento imminente dalla coda degli eventi tramite heappop()

Gestione degli Eventi
Le tipologie di evento gestite sono due:
- **Arrival**: arrivo di un nuovo job nel sistema
- **Departure**: completamento del servizio di un Job in un server

La logica generale prevede che il clock di simulazione avanzi all'istante del prossimo evento, tutti i server aggiornino il proprio stato tramite update_progress() e venga eseguita la funzione di gestione specifica.

Gestione degli Arrivi
L'evento di arrivo modella l'ingresso di un nuovo job nel sistema. Un arrivo viene gestito nel seguente modo:
1. **Pianificazione del prossimo arrivo esterno**: il sistema determina immediatamente l'istante in cui avverrà il prossimo ingresso campionando dalla distribuzione esponenziale. Un nuovo evento di arrivo viene quindi creato e inserito nella lista degli eventi (Heap binaria).
2. **Determinazione del carico di lavoro**: viene generata la quantità di servizio richiesta dal job per la sua permanenza nel primo nodo (Edge).
3. **Aggiornamento del nodo di calcolo**: il job viene inserito nel server di destinazione. Questa operazione modifica lo stato del server e incrementa un indice di versione interno. Tale indice è fondamentale per segnalare che, a causa del nuovo ingresso, la velocità di servizio per tutti i job presenti nel nodo deve essere ricalcolata secondo la logica del *Processor Sharing*.
4. **Ricalcolo della partenza imminente**: poiché la capacità del server è ora ripartita su un numero maggiore di utenti, i tempi di uscita previsti cambiano. Il sistema interroga il server per calcolare il nuovo istante di completamento più vicino tra tutti i job in corso e genera un corrispondente evento di partenza nella lista degli, associandovi la versione attuale del server.

Gestione dei Completamenti
L'evento di partenza gestisce l'uscita di un job da un server o dal sistema intero. La sua logica deve garantire la consistenza tra gli eventi programmati e lo stato attuale del sistema:

1. **Validazione tramite Versioning**: prima di processare il completamento, il sistema verifica se l'evento è ancora valido. Se dall'istante della sua programmazione fino all'istante corrente della simulazione è avvenuto un nuovo arrivo, la versione del server sarà cambiata e l'evento di partenza risulterà obsoleto. In questo caso, l'evento viene scartato senza produrre effetti. Questo meccanismo permette di gestire la dinamicità del *Processor Sharing* senza dover rimuovere fisicamente eventi dalla Heap, operazione computazionalmente inefficiente.

2. **Aggiornamento dello stato e rimozione**: se l'evento è invece valido, il job che ha terminato il servizio (quello con il tempo residuo minimo) viene rimosso dal server.

3. **Routing e Raccolta Dati**:
    - Se il job ha completato il suo intero percorso, il sistema registra il suo tempo di risposta totale negli accumulatori statistici.
    - Se il workflow prevede ulteriori passaggi (es. spostamento dall'Edge al Cloud o viceversa), il job viene inserito nel server successivo, ricominciando il ciclo di gestione arrivi per quel nodo.

4. **Pianificazione della partenza successiva**: la rimozione del job aumenta la capacità disponibile per quelli rimasti. Il sistema calcola quindi il nuovo prossimo istante di completamento e lo inserisce nel calendario degli eventi

7. Verifica

In questa fase viene analizzata la correttezza dell'implementazione del simulatore rispetto alle specifiche del modello concettuale. La verifica si pone l'obiettivo di dimostrare che il codice Python operi esattamente secondo le logiche definite nel modello concettuale, di specifica e computazionale garantendo l'assenza di inesattezze nel simulatore.

La fase di verifica è stata condotta su 128 repliche indipendenti, ciascuna con un orizzonte temporale della durata di T=3600s.

Durante la fase di verifica sono state analizzate i seguenti aspetti principali che caratterizzano il modello:

- Il processo di generazione degli arrivi
- La probabilità del routing dei Job in uscita dal nodo edge
- La generazione dei tempi di servizio (per ognuna delle possibili fasi di esecuzione di un Job)
- La correttezza del comportamento del Server Processor Sharing

Verifica del Tasso di Arrivo (λ)

Per accertare la correttezza del generatore di arrivi, è stata effettuata una simulazione a orizzonte finito, impostando un tempo di osservazione pari a T = 3600s (ossia un ora).

L'obiettivo è confermare che il processo di ingresso segua correttamente un Processo di Poisson con intensità pari al valore nominale definito come $\lambda = 1.4$ job/s.

Per effettuare la verifica del Tasso di Arrivo sono state eseguite 128 repliche della simulazione, ed è stato tracciato il numero totale di richieste effettivamente generate e immesse nel sistema, tramite l'uso della variabile total_arrivals.

Al termine di ogni replica:

1. Il valore di total_arrivals è stato utilizzato per calcolare il Tasso di Arrivo Osservato specifico di quella simulazione, calcolato come $\lambda_{obs} = \frac{total\_arrivals}{T}$
2. I 128 valori di λ_{obs} così raccolti sono stati elaborati statisticamente tramite la funzione get_estimate
3. È stata calcolata la media campionaria e l'intervallo di confidenza al 95%

Scenario	**Tasso di Arrivo Atteso (job/s)**	**Tasso di Arrivo Osservato (job/s)**
Carico Nominale	1.4	1.399307 +/- 0.003385

Verifica della Probabilità di Routing (p_C)

La corretta implementazione della logica decisionale all'uscita dal server Edge è stata verificata analizzando la probabilità di instradamento verso il Cloud (p_C). Nel modello concettuale, tale parametro definisce la frazione di job che, dopo la prima analisi sull'Edge, necessitano di un'elaborazione complessa remota.

Per ogni replica, è stato monitorato il rapporto tra il numero di volte in cui il generatore stocastico ha determinato il passaggio alla classe 'C' ed il numero totale di job entrati nel sistema.

Probabilità p_C attesa	**Probabilità p_C osservata**
0.4	0.399902 +/- 0.001198

Verifica delle Service Demands

L'obiettivo di questa fase è accertare che i carichi di lavoro generati per ogni fase del sistema siano coerenti con le distribuzioni di probabilità definite nel modello concettuale. Per ogni job che attraversa una fase di servizio (Analisi biometrica, Elaborazione Cloud, Update locale), il simulatore

campiona un valore dalla distribuzione esponenziale e lo assegna al job. Per verificare che tali estrazioni siano corrette, sono stati utilizzati due accumulatori nella classe Simulator:

1. **verify_sd_sum**: somma i valori di Service Demand non appena vengono generati
2. **verify_sd_count**: conta il numero di estrazioni effettuate

A questo punto la media delle Service Demands di una singola simulazione è ottenuta dal rapporto tra i due accumulatori della classe Simulator.

Server	Classe Job	Service Demands medio atteso (s)	Service Demands medio osservato (s)
Edge	E	0.5	0.499034 +/- 0.001193
Edge	C	0.1	0.099696 +/- 0.000371
Cloud	C	0.8	0.798467 +/- 0.002864

Verifica del Comportamento del Server Processor Sharing (PS)

La disciplina Processor Sharing governa la ripartizione della capacità di calcolo dei server Edge e Cloud. A differenza degli arrivi e delle domande di servizio, che seguono leggi stocastiche, il server PS opera secondo un algoritmo deterministico che deve garantire equità (*fairness*) e precisione nel tempo.

Per isolare il comportamento del server da interferenze statistiche, è stata progettata una suite di test unitari contenuta nel file test_ps_logic.py. Tali test sottopongono il server PS a 5 scenari controllati, confrontando i risultati osservati nel codice con i valori calcolati analiticamente.

In particolare, i 5 scenari su cui sono stati effettuati i test verificano la correttezza del comportamento del server PS a fronte di:

1) **Verifica della Velocità di Esecuzione di un Singolo Job**
   In presenza di un singolo job, è stato verificato che la velocità di esecuzione coincida con la capacità del server. Per un job con domanda S=10s, l'avanzamento della simulazione di 6s ha prodotto un tempo residuo di esattamente 4s, confermando la corretta linearità del servizio.

2) **Verifica della Ripartizione Equa**
   È stata validata la proprietà di *Fairness* in scenari di contesa sia simultanea che asincrona.
   - Contesa simultanea: con n=2 job arrivati all'istante t=0, si è verificato che la capacità di esecuzione del server venga dimezzata

- Contesa asincrona: inserendo un secondo job a simulazione avviata, il server PS ha dimostrato di saper ricalcolare correttamente l'accumulo del servizio, integrando i periodi a velocità piena con quelli a velocità ridotta

3) **Validazione del Meccanismo di Versioning**
   Il test ha confermato che:
   - Ogni variazione nel numero di job incrementa la versione del server
   - Gli eventi di partenza memorizzati nella *Future Event List* (FEL) basati su stati obsoleti vengono correttamente ignorati grazie al confronto delle versionif

   Questo test conferma che il meccanismo di Versioning garantisce che un job non possa completare il servizio prematuramente se il suo tempo di uscita è stato rallentato da nuovi arrivi.

4) **Verifica della Riassegnazione della Capacità**
   L'ultimo test ha dimostrato che, non appena un job (J1) completa il proprio servizio, il job rimanente (J2) recupera istantaneamente la piena capacità del server. Nello specifico, per un job J2 con residuo 5s, il tempo di partenza è stato ricalcolato correttamente da 20s a 15s a seguito della partenza di J1 all'istante t=10s.

8. Analisi del Transitorio

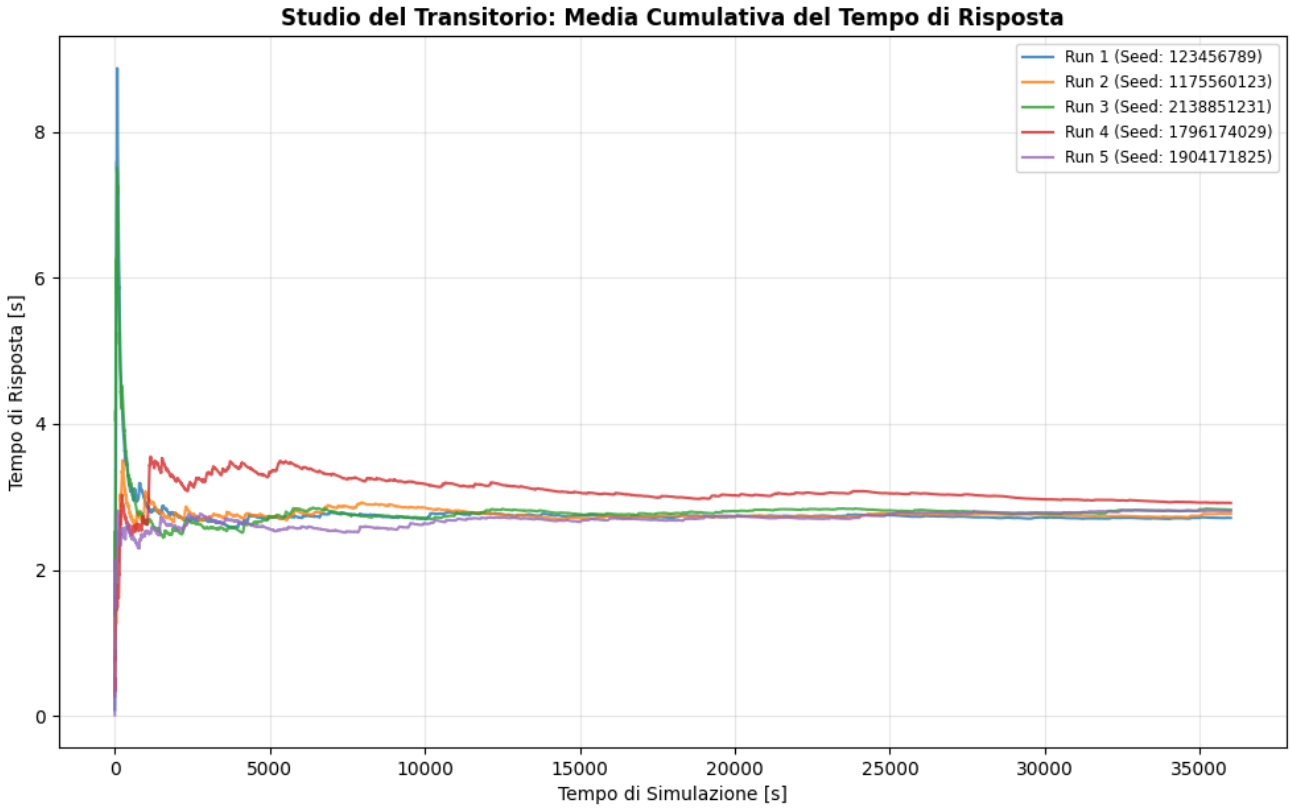
Il sistema aeroportuale è modellato su orizzonte infinito. Tuttavia, all'istante t=0 il simulatore parte da uno stato vuoto, una condizione che non riflette il reale carico operativo. I dati raccolti in questa fase iniziale (transitorio) devono essere esclusi per evitare di sottostimare i tempi di risposta medi. Per identificare la durata del transitorio, è stata monitorata l'evoluzione temporale del Tempo di Risposta medio del sistema (E[R]).

L'analisi osserva come la stima del tempo di risposta evolve al crescere del tempo di simulazione, cercando il punto in cui le oscillazioni iniziali si smorzano e la curva converge verso un valore stabile (stato stazionario).

Risultati dell'Osservazione

Dall'analisi dei grafici di convergenza (ottenuti con lo scenario "standard" a λ =1.4) è emerso quanto segue:

- Intorno ai 15000s le curve dei tempi di risposta di ognuna delle 5 run iniziano a diventare stabili



La finestra temporale della simulazione è stata impostata a 36000s (10 ore) in quanto ritenuta abbastanza grande da consentire di mostrare la convergenza del sistema e allo stesso tempo non eccessivamente grande da far perdere i dettagli dell'evoluzione iniziale del transitorio.

9. Validazione

La validazione rappresenta la fase di verifica della coerenza esterna del simulatore. Mentre la fase di verifica ha accertato il corretto funzionamento algoritmico del codice, la validazione mira a garantire che il modello implementato sia un'astrazione fedele della realtà modellata.

Data l'assenza di un data-set del sistema di sorveglianza aeroportuale modellato con il quale poter confrontare i valori generati dal simulatore, si è deciso di procedere con una validazione per confronto con il modello analitico.

Il sistema è stato analizzato come una rete aperta, caratterizzata da un flusso di arrivi poissoniano e nodi M/M/1 con disciplina Processor Sharing (PS).

L'elemento di complessità distintivo del modello risiede nel routing probabilistico con feedback. Ogni richiesta di tipo C genera un percorso multistadio che prevede un ritorno obbligatorio al server Edge per la sincronizzazione finale del database locale.

Il primo passo fondamentale è stato determinare la **Service Demand totale** (D_i) per ogni nodo.

Utilizzando i parametri di configurazione ($\lambda = 1.4\ job/s$ per il tasso di arrivo e $p_c = 0.4$ per la probabilità di instradamento al Cloud), abbiamo definito:

- **Domanda Totale all'Edge (D_E):** un job a prescindere dalla classe di appartenenza visita sempre l'Edge per la prima fase di analisi. Se il job è di tipo C (con probabilità p_C), tornerà nuovamente all'Edge per la fase di update dei database locali.

$$D_E = E[S_{Edge,\mathrm{E}}] + p_C \cdot E[S_{Edge,\mathrm{C}}] = 0.54s$$

- **Domanda Totale al Cloud (D_C):** solo i job di tipo C visitano il Cloud.

$$D_C = p_C \cdot E[S_{Cloud,\mathrm{C}}] = 0.32s$$

Una volta ricavate le domande totali, abbiamo applicato poi le leggi fondamentali della teoria delle code per ottenere i "Target" da confrontare con i risultati del simulatore:

1. **Utilizzazione (ρ):** rappresenta la frazione di tempo in cui il server è occupato

$$\rho_i = \lambda \cdot D_i$$

2. **Numero medio di utenti (E[N]):** rappresenta il numero medi di utenti in un server

$$E[N_i] = \frac{\rho_i}{1 - \rho_i}$$

3. **Tempo di Risposta Job di Classe E:**

$$R_E = \frac{E[S_{Edge,\mathrm{E}}]}{1 - \rho_E}$$

4. **Tempo di Risposta Job di Classe C:**

$$R_C = \frac{E[S_{Edge,\mathrm{E}}]}{1 - \rho_E} + \frac{E[S_{Cloud,\mathrm{C}}]}{1 - \rho_C} + \frac{E[S_{Edge,\mathrm{C}}]}{1 - \rho_E}$$

5. **Tempo di Risposta Totale:** è la somma dei tempi medi di risposta di ogni nodo visitato, pesati per il numero di visite:

$$E[R_{tot}] = \frac{D_E}{1 - \rho_E} + \frac{D_C}{1 - \rho_C}$$

Dopo aver calcolato i valori teorici attesi dal modello sono stati confrontati con i valori ottenuti dalla simulazione e sono stati ottenuti i risultati contenuti nella seguente tabella riassuntiva.

Componente	Metrica	Risultato Teorico Atteso	Risultato Sperimentale
E	ρ_E	0.756	0.755764 +/- 0.000525
	R_E	2.049	2.047516 +/- 0.0067
	$E(N_E)$	3.098	3.095894 +/- 0.01068
C	ρ_C	0.448	0.447824 +/- 0.000534
	R_C	3.908	3.907015 +/- 0.00914
	$E(N_C)$	0.811	0.811727 +/- 0.00203
Sistema Totale	$E[R_{tot}]$	2.792	2.791316 +/- 0.007419

Dunque, possiamo concludere che anche in assenza di un data-set reale, il modello rappresenta correttamente il sistema di riferimento ed è dunque possibile proseguire con la fase di studio degli obiettivi.

10. Preparazione delle Simulazioni

L'attività di sperimentazione è stata progettata per valutare le prestazioni del sistema di sorveglianza aeroportuale basato su dati biometrica in diverse condizioni operative. Lo studio si sviluppa in diverse prove sperimentali distinte, ciascuna mirata a rispondere a specifici quesiti relativi alle prestazioni, scalabilità e all'efficienza dell'infrastruttura.

I parametri di input e le configurazioni di simulazione sono gestiti all'interno del file configurazione_sistema.py, garantendo la riproducibilità e la consistenza dei risultati.

Per ogni scenario analizzato, è stata adottata una tecnica di simulazione ad "orizzonte temporale infinito" identificato da una finestra temporale dell'esecuzione di durata 100.000 secondi con l'obiettivo di studiare il comportamento del sistema a regime stazionario.

Al fine di eliminare il bias indotto dalle condizioni iniziali (sistema inizialmente vuoto), è stato introdotto un periodo di warmup pari a 15.000 secondi; i dati raccolti in questo intervallo vengono sistematicamente esclusi dal calcolo delle medie finali per non influenzare la precisione delle stime.

La precisione statistica è garantita dall'esecuzione di 128 repliche indipendenti per ogni scenario. I risultati finali, comprensivi di medie campionarie e intervalli di confidenza al 95%, vengono salvati nella cartella sim_results e elaborati mediante la distribuzione t-Student attraverso lo script stima_confidenza.py

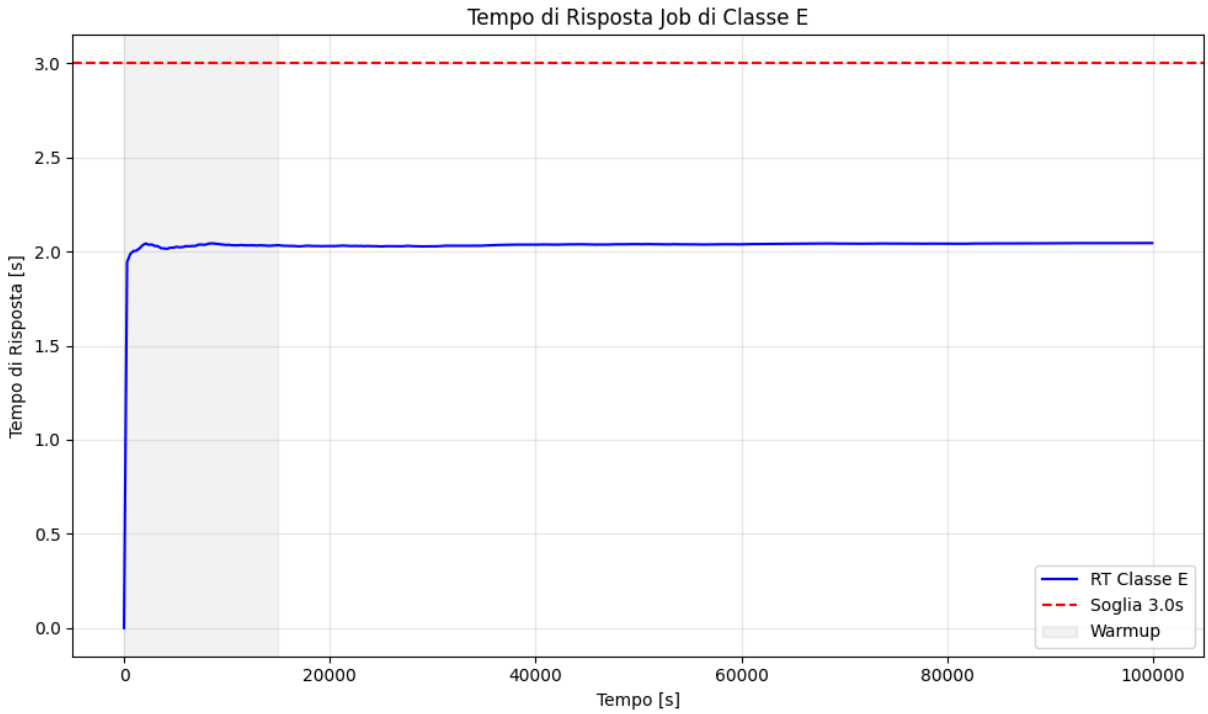
Il simulatore implementato adotta strategie rigorose per la gestione della stocasticità:

- Per garantire l'indipendenza statistica totale tra le repliche, è stata implementata una logica di concatenazione. Ogni replica di una simulazione dello stesso scenario inizia il proprio ciclo di generazione partendo esattamente dallo stato finale (seed) restituito dalla replica precedente tramite il metodo getSeed(). Questo approccio assicura che l'intera fase sperimentale utilizzi porzioni non sovrapposte della sequenza del generatore di Lehmer.
- **Stream Multipli:** per isolare i diversi fenomeni casuali, sono stati definiti 5 flussi indipendenti:
    - *Stream 0:* Inter-arrivi dei dati biometrici
    - *Stream 1:* Scelte di routing verso il Cloud ($p_C = 0.4$)
    - *Stream 2:* Tempi di servizio per l'analisi biometrica (Edge Classe E)
    - *Stream 3:* Tempi di servizio per l'aggiornamento database (Edge Classe C)
    - *Stream 4:* Tempi di servizio del server Cloud

11. Analisi dei Risultati

Obiettivo 1: Analisi QoS sul Tempo di Risposta per i Job di classe E

Il primo obiettivo della sperimentazione è la verifica del requisito fondamentale di reattività del sistema: la scansione biometrica iniziale (Classe E) deve essere completata entro una soglia massima di 3 secondi. Dall'osservazione dei risultati ottenuti nelle 128 repliche in condizioni nominali ($\lambda = 1.4$), si evince che il sistema rispetta ampiamente le specifiche dell'obiettivo. Il tempo di risposta

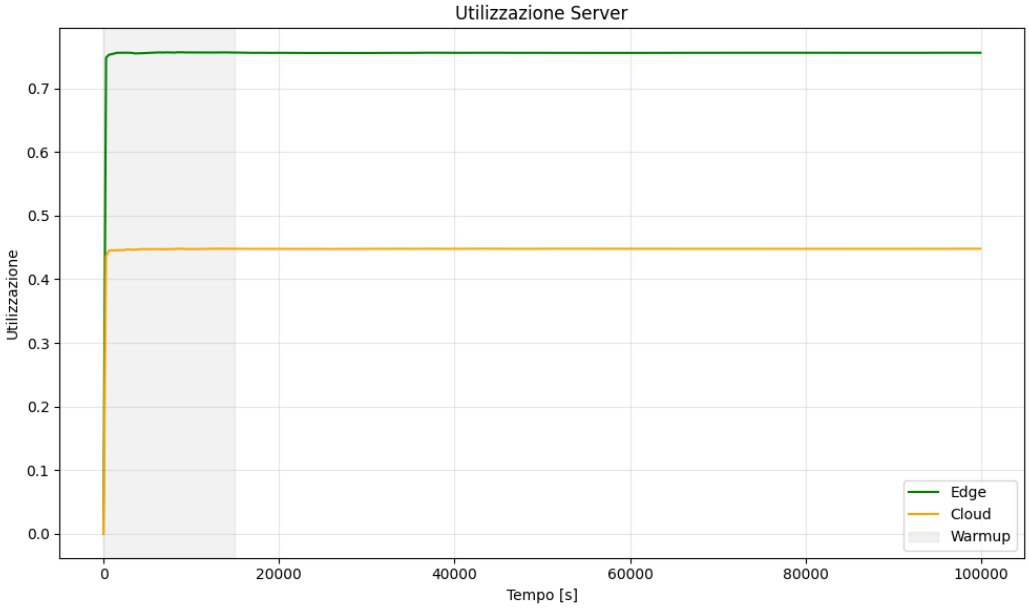


medio per i job di Classe E si attesta su un valore di 2.0377 +/- 0.0093s, con un intervallo di confidenza del 95%.

Il fatto che il tempo di risposta sia significativamente inferiore (circa il 30%) rispetto alla soglia dei 3 secondi indica che l'Edge Node, pur essendo il nodo più sollecitato come evidenziato dal prossimo grafico, riesce a smaltire le richieste di analisi iniziale in modo rapido.

L'ampio margine di sicurezza tra il tempo di risposta ed il QoS consente di avere un "cuscinetto" prestazionale che consente al sistema di assorbire piccole fluttuazioni temporanee nel tasso di arrivo senza violare immediatamente il rispetto del QoS.

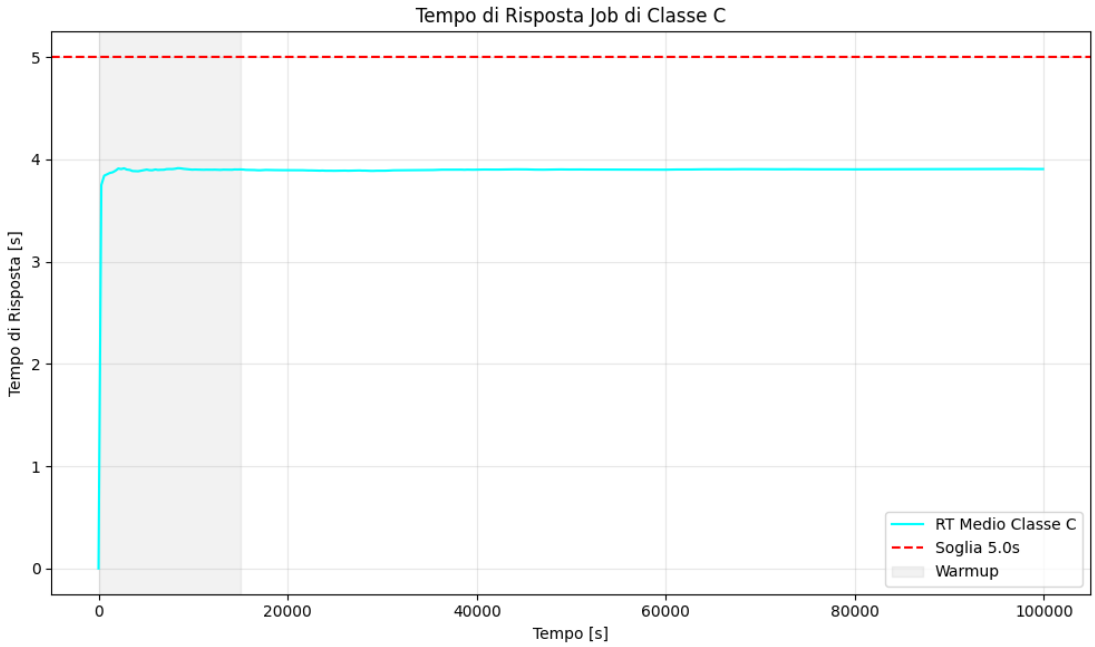
La convergenza del grafico verso un valore asintotico stabile conferma che il rispetto del requisito dei 3s non è un evento sporadico, ma una caratteristica strutturale del sistema a regime stazionario.



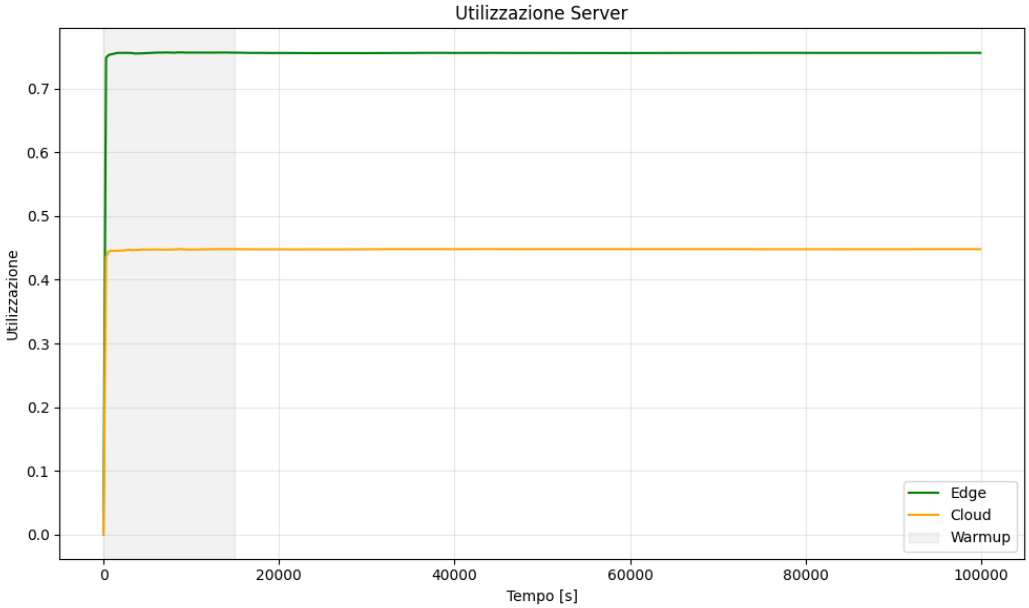
Obiettivo 2: Analisi QoS sul Tempo di Risposta dei Job di classe C

Il secondo obiettivo mira a verificare se l'infrastruttura sia in grado di supportare il flusso di operazioni più oneroso. Infatti, un job di classe C è soggetto a una "tripla latenza": deve attendere la propria elaborazione sul nodo edge poi sul nodo cloud ed infine un secondo ciclo di elaborazione sull'edge per eseguire l'update del DB locale.

Dall'analisi delle simulazioni effettuate nello scenario nominale ($\lambda = 1.4\ job/s$), il tempo di risposta medio per la Classe C risulta di 3.9 secondi, dunque anche questo QoS è soddisfatto.



Abbiamo a questo punto confermato che il sistema così come è strutturato è in grado di gestire il carico di lavoro in modo efficace rispettando tutti i QoS.

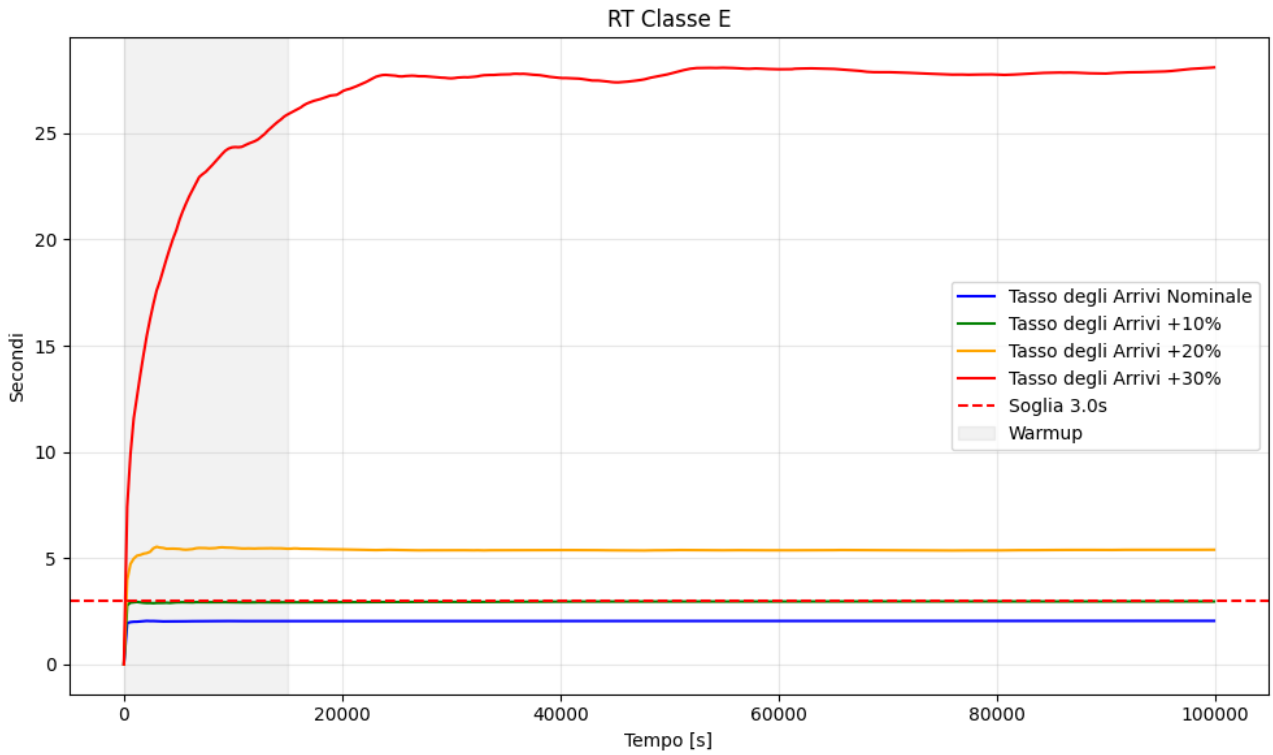


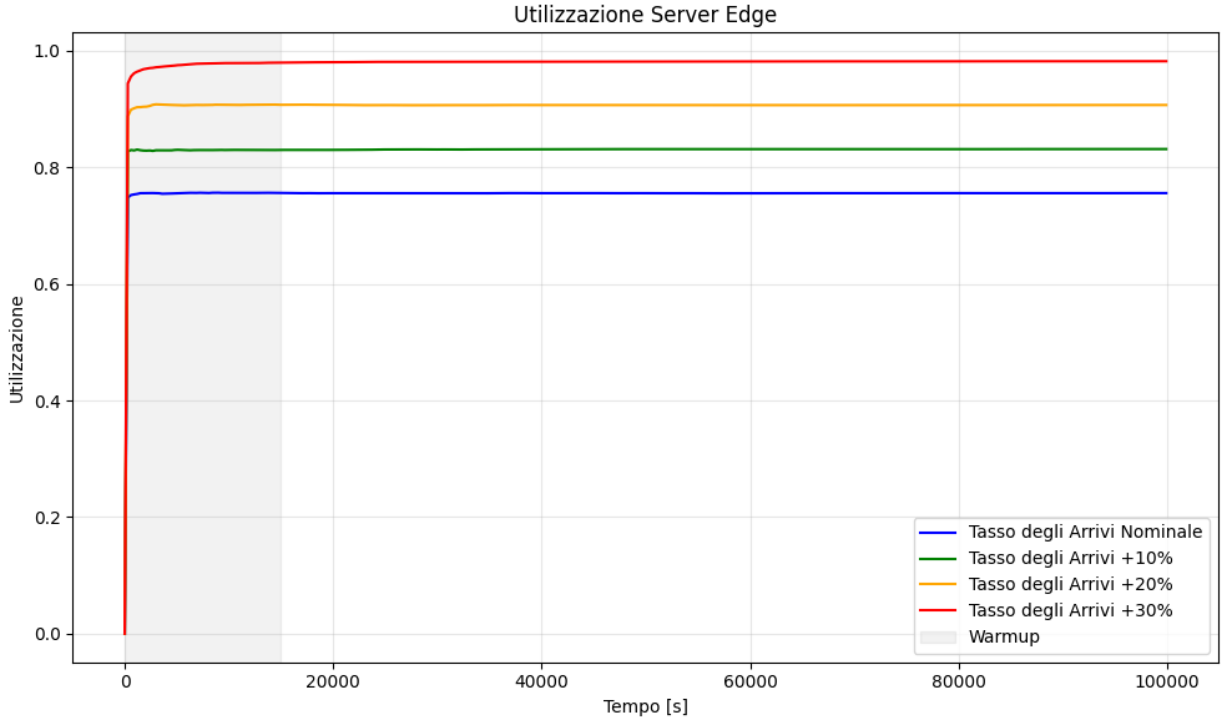
Come facilmente prevedibile il grafico delle utilizzazioni dei server mostra che il nodo edge è quello più utilizzato. Questa è una diretta conseguenza del fatto che gestisce sia tutti gli arrivi che una frazione dei job che smista verso il nodo Cloud. La conseguenza di questo risultato è che il tempo di Risposta dei job di classe C viene frenato per ben due volte dal server Edge.

Obiettivo 3: Stress Test e Sensibilità al Carico di Lavoro

Il terzo obiettivo dello studio consiste nel valutare la scalabilità dell'infrastruttura e la tenuta dei requisiti di QoS a fronte di un incremento progressivo del tasso di arrivo dei passeggeri. Vengono

analizzati i possibili scenari tipici dei periodi di alta affluenza aeroportuale, applicando al sistema carichi incrementali del +10%, +20% e +30% rispetto al valore nominale ($\lambda = 1.4$).

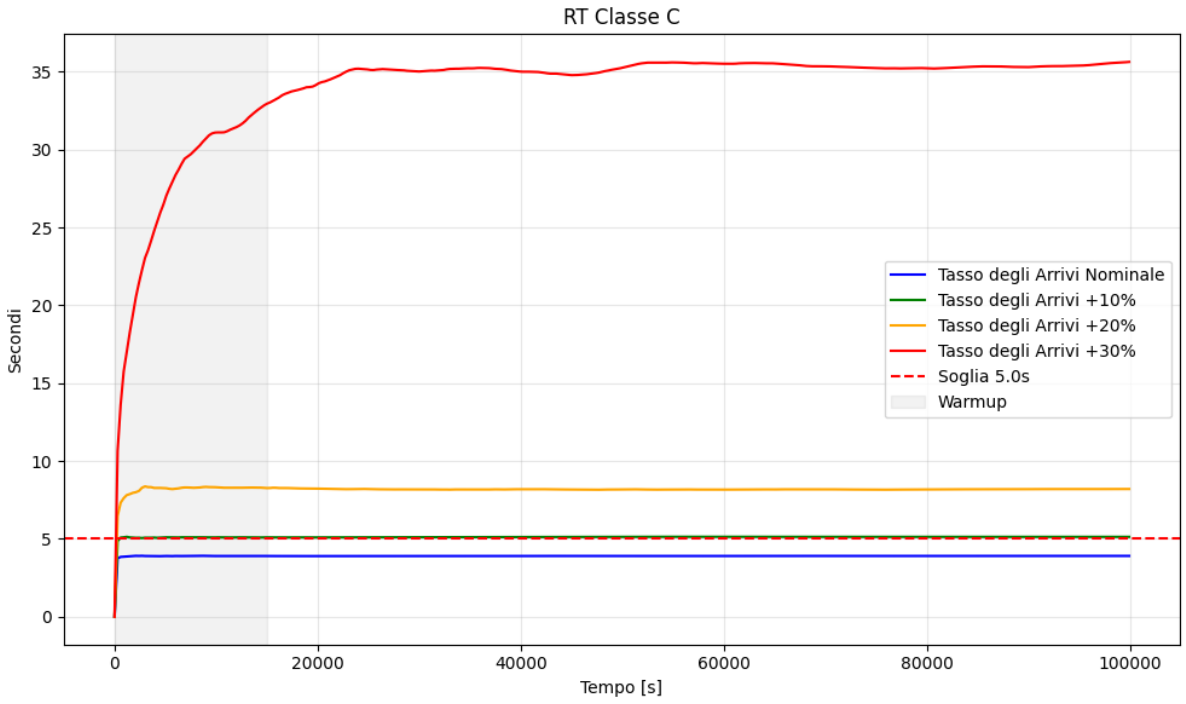




L'osservazione dei risultati evidenzia una degradazione esponenziale delle prestazioni. Infatti nonostante l'incremento del traffico sia lineare, i risultati mostrano un degradamento delle

prestazioni che segue un andamento esponenziale, specialmente nel passaggio dallo scenario +20% a quello +30%.

Tale fenomeno è riconducibile alla saturazione dell'Edge Server, il cui valore di utilizzazione ρ tende pericolosamente al valore 1. Quando il sistema opera in prossimità del limite fisico di capacità $\rho \to 1$, il fattore di rallentamento indotto dalla concorrenza delle risorse $\frac{1}{1-\rho}$ amplifica notevolmente ogni minima variazione del carico, trasformando il sistema da stabile a congestionato.



L'analisi più critica riguarda il Tempo di Risposta dei job di Classe C, i quali manifestano un'esplosione dei tempi di attesa che supera di oltre 10 secondi i valori registrati per la Classe E nello scenario di massimo stress. Questa discrepanza non è casuale, ma è l'effetto diretto del workflow "Edge-Cloud-Edge". Sotto carico elevato, il job di Classe C subisce una "doppia penalità" sul server Edge: una prima volta durante la scansione iniziale e una seconda volta, durante la fase di update post-Cloud.

Un ulteriore conferma di quanto analizzato fino ad ora ci viene data dall'andamento del numero di utenti nel nodo Edge, il quale cresce in modo esponenziale a conferma dell'incapacità del nodo di servire in modo adeguato tutte le richieste che lo popolano.

I dati raccolti decretano il fallimento del rispetto dei QoS:

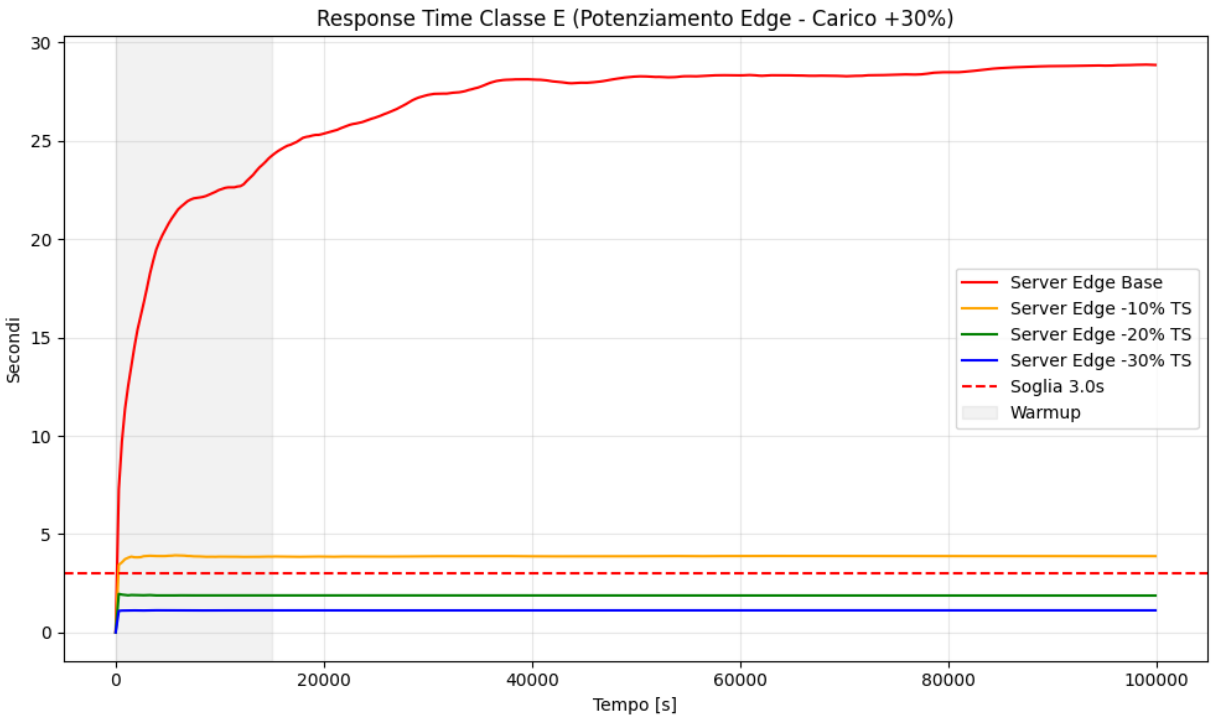
- **Classe E:** la soglia dei 3 secondi viene violata non appena il carico supera il 10% scenario in cui invece il tempo di risposta registrato è di 2.9507 +/- 0.0185 e quindi conforme al QoS
- **Classe C:** il requisito dei 5 secondi viene infranto già nello scenario +10%, con picchi di latenza che rendono vana la logica di identificazione in tempo reale.

In conclusione, lo Stress Test dimostra che l'attuale architettura, seppur valida in condizioni nominali, è totalmente priva di margini di scalabilità, rendendo necessario un intervento correttivo per gestire anche minimi scostamenti dal traffico previsto.

Obiettivo 4: Valutazione dell'Efficacia delle Soluzioni Migliorative sul Nodo Critico

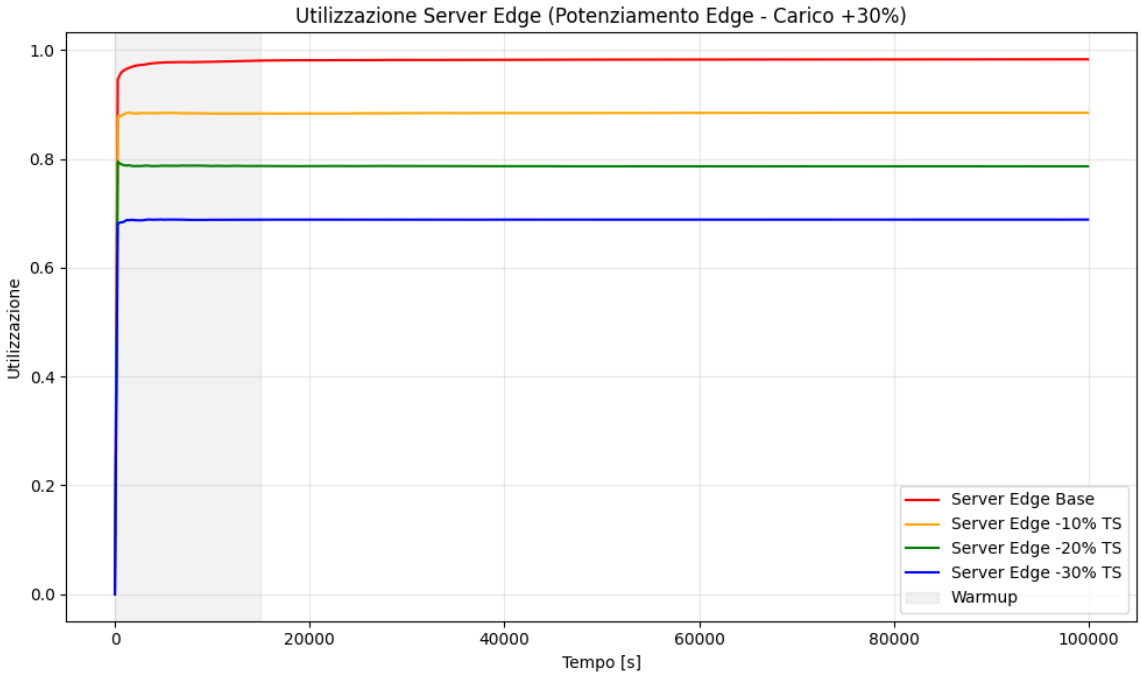
L'ultima fase della sperimentazione si configura come un'analisi "What-If" volta a individuare la strategia di intervento ottimale per ristabilire i requisiti di QoS compromessi nello scenario di stress massimo (+30% del tasso di arrivo). Avendo identificato l'Edge Server come l'unico collo di bottiglia del sistema, la strategia correttiva si focalizza esclusivamente sul potenziamento della sua capacità di calcolo, simulando riduzioni graduali dei tempi di servizio del 10%, 20% e 30%.

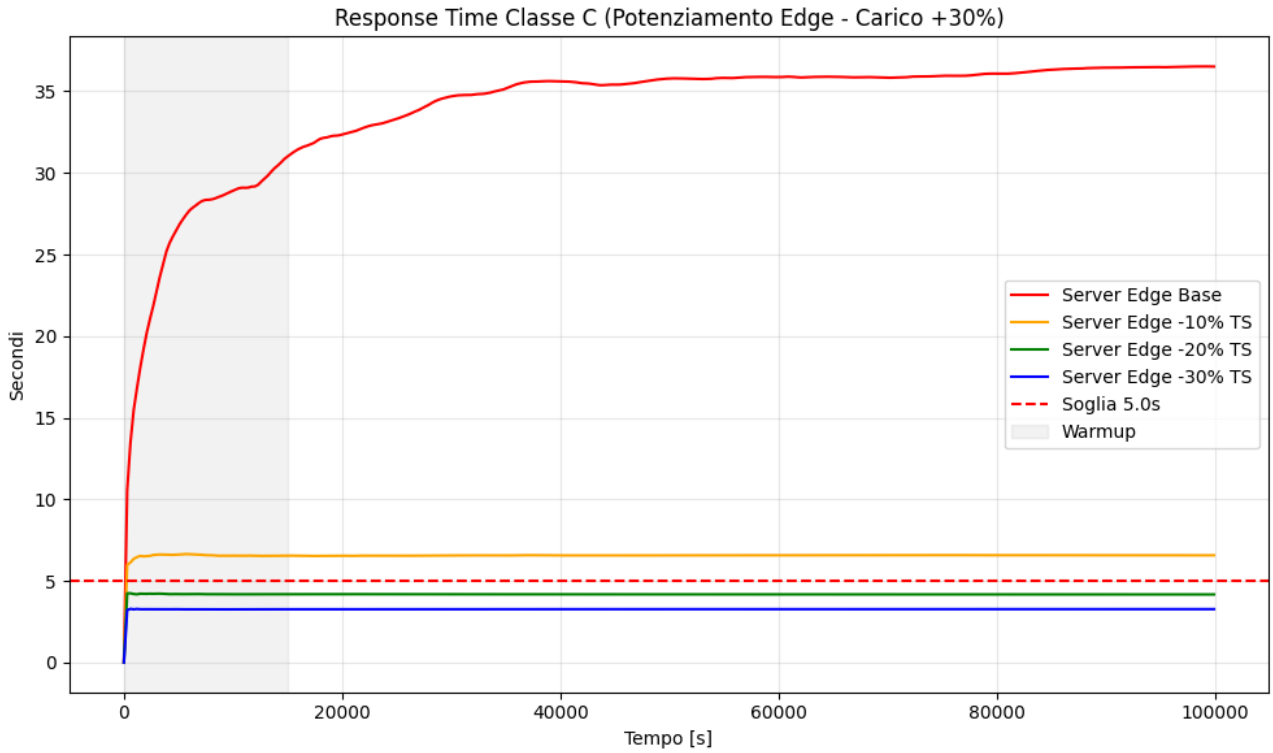
L'intervento hardware agisce direttamente sul parametro critico del sistema: l'utilizzazione dell'Edge Server. Poiché $\rho = \lambda \cdot E[S]$, ridurre il tempo di servizio E[S] permette di abbassare l'utilizzazione anche a fronte di un tasso di arrivo λ elevato.



Si osserva che un upgrade del 10% è insufficiente, mentre l'incremento prestazionale del 20% si rivela essere la soglia ottima del miglioramentp. In questa configurazione, l'utilizzazione dell'Edge scende a valori accettabili (circa l'80%), permettendo alla politica *Processor Sharing* di gestire i job concorrenti senza generare le code infinite osservate durante lo Stress Test precedente.

L'effetto più evidente dell'upgrade si registra nella Classe C, che nello scenario di stress era la più penalizzata. Con un potenziamento del 20%, il Response Time della Classe E torna stabilmente sotto i 3s e, quello della Classe C rientra sotto la soglia critica dei 5s.





L'analisi dell'obiettivo 4 mostra come il potenziamento mirato dell'Edge (risorsa collo di bottiglia) ha reso possibile rendere l'intera infrastruttura resiliente a un carico del +30% con un investimento hardware relativamente contenuto (+20%). Anche se la classe C era la più penalizzata per il tempo di risposta un eventuale potenziamento del nodo Cloud non avrebbe migliorato la situazione poiché il collo di bottiglia dell'intero sistema era rappresentato dal nodo edge

12. Conclusioni

Lo studio condotto ha permesso di modellare e analizzare con rigore statistico un'infrastruttura Edge-Cloud per la videosorveglianza aeroportuale. Grazie all'impiego di una simulazione a eventi discreti (DES) e all'applicazione di metodologie di *Capacity Planning*, è stato possibile mappare con precisione i limiti di scalabilità del sistema attuale.

Dall'indagine sperimentale emergono tre evidenze fondamentali:

- Identificazione del Bottleneck strutturale: il nodo Edge è stato identificato come l'anello debole della catena. Sebbene il Cloud gestisca le analisi più complesse, l'Edge subisce la "doppia penalità" di dover gestire sia i flussi biometrici in ingresso che i feedback di aggiornamento del database.
- Fragilità della QoS nominale: il sistema è dimensionato correttamente per il carico standard ($\lambda = 1.4$), ma opera con margini di sicurezza minimi. Già con un incremento del 10% del traffico, i tempi di risposta della Classe C sfiorano la soglia critica, indicando un'imminente saturazione.

- Efficacia della mitigazione locale: l'analisi "What-If" ha dimostrato che non è necessario potenziare l'intera infrastruttura. Un upgrade mirato del 20% della capacità di calcolo del solo nodo Edge è sufficiente a rendere il sistema resiliente anche a picchi di traffico del +30%, riportando tutte le metriche dei tempi di risposta (Classe E < 3s, Classe C < 5s) entro le specifiche di sicurezza aeroportuale.

13. Limitazioni del Modello

Nonostante la validazione analitica abbia confermato l'accuratezza del simulatore, il modello presenta alcune semplificazioni che ne circoscrivono l'aderenza alla realtà operativa aeroportuale:

- L'uso di un processo di Poisson assume che i passeggeri arrivino in modo indipendente e costante. In aeroporto, tuttavia, gli arrivi sono spesso caratterizzati da fenomeni di burstiness (es. l'apertura simultanea di più gate o lo sbarco di un intero volo), che potrebbero mettere sotto stress il Server Edge molto più di quanto previsto da un flusso medio uniforme.
- Trasparenza dell'infrastruttura di rete: è stato assunto che i ritardi di trasmissione tra Edge e Cloud siano trascurabili. In uno scenario reale, la latenza della rete potrebbe sommare secondi preziosi al tempo di risposta della Classe C, rendendo il QoS di 5 secondi ancora più difficile da garantire.
- Assenza di politiche di prioritizzazione: il server PS attuale tratta tutti i job allo stesso modo. Nella realtà, i job di Classe C (aggiornamento per soggetti pericolosi) dovrebbero avere la priorità assoluta per allertare il personale, ma il modello attuale non distingue l'urgenza dell'elaborazione.