A MOBILE EDGE COMPUTING ARCHITECTURE SYSTEM BASED ON THE VIDEO STREAMING CASE STUDY

Progetto di Modellistica, simulazione e valutazione delle prestazioni

MARTINA SALVATI (0292307), SIMONE BENEDETTI(0295385)

INDICE

1 INTRODUZIONE

Al giorno d'oggi, ci sono una serie di risultati di ricerca in architetture di mobile edge computing (MEC), algoritmi e risultati di implementazione.

Mobile Edge Computing (MEC) è un tipo di edge computing che estende le capacità del cloud computing portandolo ai margini della rete. MEC è nato dall'iniziativa ETSI (European Telecommunications Standards Institute) che originariamente si concentrava sul posizionamento di nodi edge sulla rete mobile, ma ora si è espanso per includere la rete fissa. Mentre il cloud computing tradizionale si verifica su server remoti situati lontano dall'utente e dal dispositivo, MEC consente ai processi di svolgersi in stazioni base, uffici centrali e altri punti di aggregazione sulla rete. Spostando il carico del cloud computing sui singoli server locali, MEC aiuta a ridurre la congestione sulle reti mobili e a ridurre la latenza, migliorando la qualità dell'esperienza (QoE) per gli utenti finali.

Questa è una prova evidente che il sistema MEC può consentire l'evoluzione delle tecnologie di rete.

Uno dei motivi principali per cui i sistemi basati su MEC stanno ricevendo molte attenzioni è il fatto che sia una delle soluzioni migliori in grado di supportare lo ***streaming video e la distribuzione di contenuti.***

La tecnica MEC consente di migliorare le prestazioni di elaborazione e l'efficienza energetica del dispositivo utente, scaricando l'attività sul server edge, avente prestazioni migliori rispetto ai dispositivi terminali.

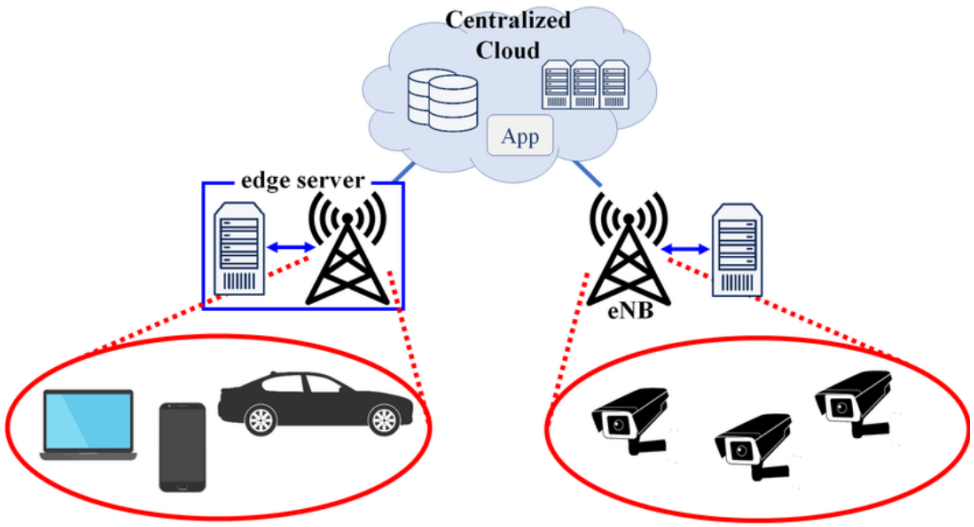

Fig. 1 Mobile edge computing (MEC) architecture

2 OBIETTIVI E CASE STUDY

Lo streaming video e i giochi per computer sono tra i media più popolari e con il più alto consumo di larghezza di banda in Internet. I contenuti video consumano oggi circa il 70% dell'utilizzo totale della larghezza di banda in Internet. I progressi negli strumenti di generazione multimediale, l'elevata potenza di elaborazione e la connettività ad alta velocità hanno consentito la generazione di contenuti multimediali live, interattivi e multi-view. L'edge computing ha lo scopo di fornire risorse di storage e computazionali vicino all'utente all'edge della rete, per ridurre al minimo la latenza e i tempi di risposta.

Lo scopo del progetto è implementare uno studio su un algoritmo di offload dei contenuti video che mira all'ottimizzazione in termini di ritardo.

Lo streaming video viene trasmesso in meno di 1 secondo, in genere 100-500 millisecondi.

Quando i servizi video in tempo reale vengono forniti a singoli utenti come lo streaming video è importante:

(i) ridurre al minimo il tempo di esecuzione totale. (inferiore ai 500 millisecondi)

3 ALGORITMO 1

L'algoritmo 1 rappresenta il modello base che racchiude l'idea del Mobile Edge Computing nello specifico caso d'uso del video streaming.

Quando si crea una video, vengono acquisite migliaia di singoli frame in sequenza. La frequenza dei frame influisce nel modo in cui viene percepito il movimento in un video. All'aumentare dei frame che vengono acquisiti, la percezione del video migliora, per questo è importante capire come far arrivare questi frame nel Web per soddisfare la richiesta del videostreaming.
Il sistema proposto nel modello, vede viaggiare i frame nella rete, coinvolti nella composizione finale dei video. La rete descrive il modo in cui viaggiano questi frame da i dispositivi locali fino al Cloud, passando per l'Edge.

3.1 MODELLO CONCETTUALE

La rete è stata modellata attraverso diversi sottosistemi:

- **Locale** : Il sottosistema locale rappresenta l'insieme dei blocchi Control Unit e Video Unit.
  - *Blocco Control Unit* : si occupa di ricevere ed elaborare le richieste dall'esterno e di instradarle verso Video Unit o Networking. Il blocco viene modellato come un $\mathrm{M}/\mathrm{M}/1$[1].
  - *Blocco Video Unit :* si occupa dell'elaborazione dei frame acquisiti dai dispositivi di video-acquisizione e rendering video. Nella pratica, i dispositivi di video-acquisizione non prevedono l'accodamento di richieste, per questo sono stati modellati un $\mathrm{M}/\mathrm{M}/2/2$[2].
- **Networking** : Il sottosistema networking rappresenta l'insieme dei blocchi Wlan Unit e eNode Unit.
  - *Blocco Wlan Unit* : si occupa dell'uploading/downloading nella rete dei frame video. E' stato modellato come un $\mathrm{M}/\mathrm{M}/2$[3] , in quanto l'AP per la rete WiFi ha solitamente 2 core (modem, ..).
  - *Blocco eNode Unit :* si occupa dell'uploading/downloading nella rete (LTE 4g) dei frame video. E' modellato come un $\mathrm{M}/\mathrm{M}/1$, in quanto l'AP per la rete 4g viene descritto solitamente un unico core computazionale.
- **Edge** : Il sottosistema Edge rappresenta il blocco Edge Unit.
  - *Blocco Edge Unit* : si occupa di *Optimization* (aumentando il bit Rate e diminuendo la latenza), Data Compression e Storage. E' modellato come un $\mathrm{M}/\mathrm{M}/4$, per il lavoro potenzialmente oneroso e multi-variato.

[1] Notazione Kendall : Un M/M/K si riferisce ad un sistema con k server, con coda infinita, e distribuzione di arrivi e di servizio esponenziale.
[2] Notazione Kendall : UN M/M/2/2 si riferisce ad un sistema multi-servente (2 core), senza coda, e distribuzione di arrivi e di servizio esponenziale.

- **Cloud** : Il sottosistema Cloud rappresenta il blocco Cloud Unit.
  - *Blocco Cloud Unit* : rappresenta il blocco finale, lontano dall'utente , punto cardinale per la distribuzione dei frame video nella rete e lo storage. Il Cloud Unit è modellato come un $M/M/\infty$. Il cloud, d'altra parte, data la sua capacità praticamente infinita può essere considerato come un server infinito, senza contesa tra utenti diversi.

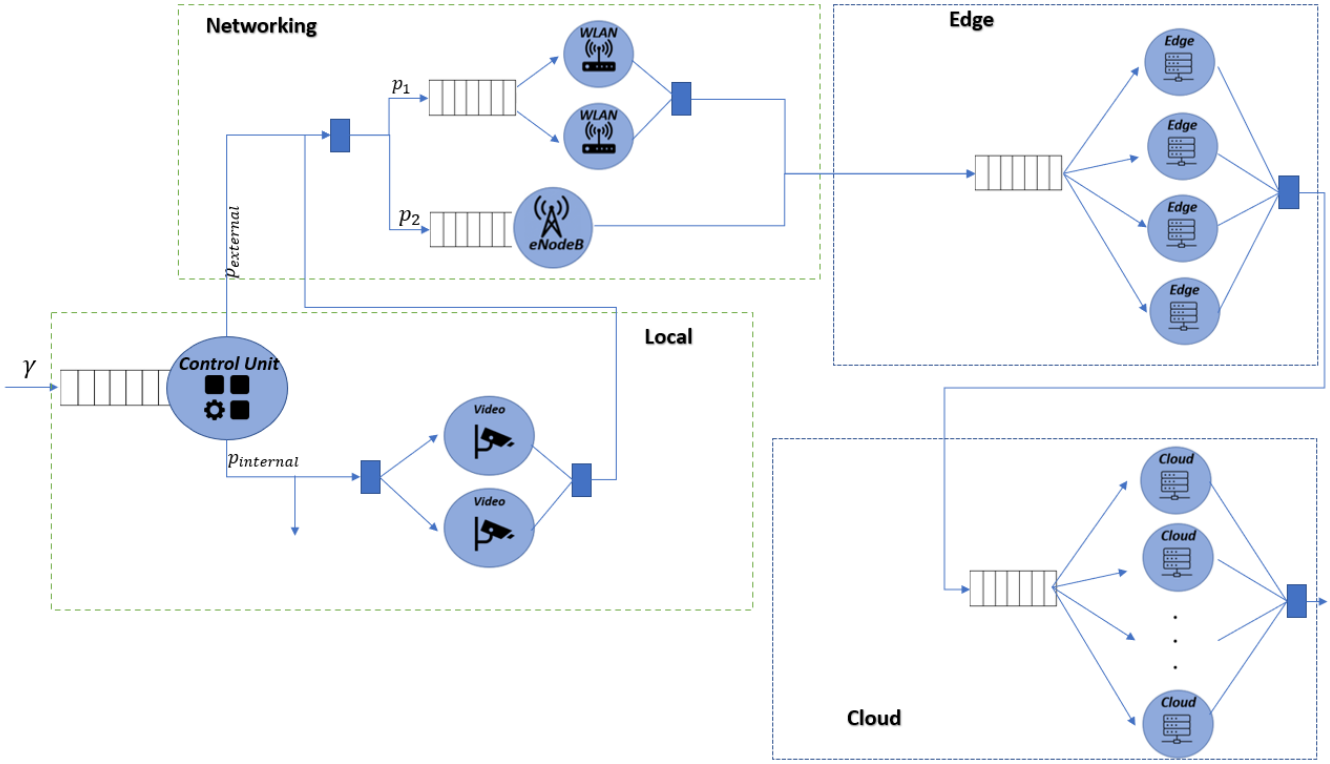

Fig. 2 Rete del sistema 2.

Le *variabili di stato* considerate sono :

- Stato di un servente (libero/occupato).

Gli *eventi* che possono esserci :

- Arrivo di una richiesta

3.2 MODELLO DELLE SPECIFICHE

I tempi di interarrivo ed i tempi di servizio sono modellati secondo una distribuzione Esponenziale, che rappresenta una buona scelta per il sistema in esame.

Periodo di osservazione :

- 1 h. (espressa in millisecondi $\mathbf{1 * 60 * 60 * 1000}\ \boldsymbol{ms}$)

Lo streaming video viene trasmesso in meno di 1 secondo, in genere 100-500 millisecondi, per questo 1h è ragionevole come periodo di osservazione date le unità di misura.

Tempi di servizio:

Blocco	Service time per Servente (ms)
CONTROL UNIT	15.0
VIDEO UNIT	15.0
WLAN FRAME UPLOAD TIME	30.0
ENODE FRAME UPLOAD TIME	40.0
EDGE PROCESSING TIME	35.0
CLOUD PROCESSING TIME	10.0

Probabilità di routing:

• $\boldsymbol{P_{internal} = 0,4}$	La probabilità che il blocco Control Unit invii la sua richiesta al Video Unit.
• $\boldsymbol{P_{external} = 1 - P_{internal} = 0,6}$	La probabilità che il blocco Control Unit invii la sua richieste verso il sottosistema Networking.
• $\boldsymbol{P_{wlan_{off}} = 0,11}$	La probabilità che l'AP wlan sia OFF , ovvero sia offline.
• $\boldsymbol{P_{wlan_{on}} = 1 - P_{wlan_{off}} = 0,89}$	La probabilità che l'AP sia ON.
• $\boldsymbol{P_{wlanchoice} = 0,88764}$	La probabilità di scegliere come AP di instradamento nella rete wlan. Essendo le prestazioni della rete WiFi più alte in media rispetto alla rete cellulare, viene giustificata la alta probabilità di scelta.
• $\boldsymbol{P_{wlanchoice} \cap P_{wlan_{on}} = 0,79 = p_1}$	La probabilità rappresenta in effetti la probabilità di instradamento delle richieste verso il blocco Wlan.
• $\boldsymbol{1 - P_{wlanchoice} \cap P_{wlan_{on}} = 0,21 = p_2}$	La probabilità di instradamento delle richieste verso il blocco eNode (LTE/4g).

3.3 MODELLO COMPUTAZIONALE

Per la simulazione della rete è stato utilizzato l'approccio di tipo Next-Event Simulation, in cui l'avanzamento del tempo si effettua tramite processamento dell'evento successivo. Il modello è sviluppato in linguaggio c, utilizzando la libreria *http://www.math.wm.edu/~leemis/simtext.code.c.*

3.3.1 Directory di progetto

La struttura delle directory di progetto :

- *./src/* : Contiene i sorgenti principali della simulazione, distinti da alg1 (sistema 1) e alg2 (sistema2).
- *./include* : Contiene gli header di progetto (config1.h/config2.h), file di configurazione utili al progetto e gli header per le librerie rngs,rvgs,rvms.
- *./lib* : Contiene i file di libreria.
- *./results* : Contiene i file .csv e .txt di output per le simulazioni.

- *./statistiche* : Contiene i file acs/estimate/uvs, utili per calcolare gli intervalli di confidenza, autocorrelazioni e altri valori statistici.
- *./modelli_analitici* : Contiene i file xlsx con all'interno il modello analitico per ogni sistema.

3.3.2 Strutture dati

Server

La struttura server si riferisce ad ogni servente del sistema e contiene le informazioni necessarie per la gestione di ogni server (**Stato** (BUSY/IDLE) e **Statistiche varie, Blocco relativo)**

Block

La struttura contiene le informazioni di ogni blocco.

- Puntatori ai serventi del blocco
- Statistiche del blocco (numero totali arrivi, completamenti, ..)
- I job in coda implementati tramite una linked list
- Stream associato al blocco (Ogni blocco ha uno stream diversificato, per partizionare l'output proveniente da Lehmer RNG)

Completamento

E' la struttura che contiene il tempo di completamento riferito ad un servente di un specifico blocco. Ogni completamento viene mantenuto in una lista ordinata di completamenti.

Job

La struttura rappresenta la linked list dei job del sistema.

Clock

La struttura contiene il clock di simulazione, utile per la simulazione next-event.

Area

La struttura area viene utilizzata per il calcolo delle statistiche, somma dei tempi in coda.

3.3.3 Gestione Eventi

Gli eventi di completamento e gli arrivi vengono gestiti in maniera differente. In particolare viene verificato se il nextEvent è un evento di completamento o arrivo in base al tempo specificato.
Il valore del *clock.next* viene impostato al minimo tra *clock.arrival* (che rappresenta l'evento di arrivo di un job dall'esterno) e *clock.NextCompletition* (che rappresenta il prossimo completamento, estratto dalla lista ordinata dei completamenti. Se il minimo è *clock.arrival* viene gestito l'evento di arrivo, altrimenti viene gestito l'evento di completamento. Il *clock.current* viene impostato al *clock.next.*

3.3.3.1 Arrivi

Se l'evento è un arrivo viene chiamata la funzione *process_arrival()* che effettua il processamento di un arrivo. Gli arrivi dall'esterno vengono elaborati direttamente dal Control Unit. Se il servente del Control Unit risulta libero (in base allo stato) , l'arrivo viene servito e viene elaborato il completamento e aggiunto alla lista di completamenti. Altrimenti, se Control Unit risulta occupato, l'arrivo viene messo in coda seguendo la logica dell'accodamento FIFO. I completamenti vengono elaborati in base al routing delle probabilità : i job possono essere di tipo Internal (destinati a Video Unit) o External (destinati al Cloud). La logica di dispatching segue le probabilità previste dal sistema. Per ricavare le probabilità all'interno del sistema, viene usata la funzione $Uniform(0,1)$ che al suo interno utilizza $Random()$ (Lehmer Generator).

3.3.3.2 Completamenti

Se l'evento è un completamento viene chiamata la funzione *process_completion(comp c).* Il job presente in block risulta servito e viene eliminato dalla lista dei completamenti, la logica che segue dipende dalla disponibilità del server (o servers) del blocco. Se il blocco è libero allora viene effettuato un ulteriore processamento di un completamento e aggiunto alla lista dei completamenti con cambiamento dallo stato in BUSY. Altrimenti rimane disponibile. Il job che viene servito dal blocco viene indirizzato ad un blocco successivo e la destinazione viene decisa dalla funzione *getDestination(typeOfBlock, typeOfJob).*

```
112  // Ritorna il blocco destinazione di un job dopo il suo completamento
113  int getDestination(enum block_types from, int type) {
114      switch (from) {
115          case CONTROL_UNIT:
116              if(type==INTERNAL) {   //ROUTING TO VIDEO - INTERNAL JOB
117                  DEBUG_PRINT("JOB SERVITO  -> DIRECTED TO VIDEO UNIT\n");
118                  return VIDEO_UNIT;
119              }
120              if(type==EXTERNAL) {  //ROUTING TO CLOUD -EXTERNAL JOB
121                  DEBUG_PRINT("JOB  SERVITO  -> DIRECTED TO CLOUD\n");
122                  int ret = routing_to_cloud();
123                  DEBUG_PRINT("ROUTING FROM CONTROL UNIT TO %s\n", stringFromEnum(ret));
124                  return ret;
125              } else {
126                  return -1;
127              }
128          case VIDEO_UNIT:
129                int ret = routing_to_cloud();
130                DEBUG_PRINT("ROUTING FROM VIDEO UNIT TO %s\n", stringFromEnum(ret));
131                return ret;
132          case WLAN_UNIT:
133              return EDGE_UNIT;
134          case ENODE_UNIT:
135              return EDGE_UNIT;
136          case EDGE_UNIT:
137              return CLOUD_UNIT;
138          case CLOUD_UNIT:
139              return EXIT;
140              break;
141      }
142      return -1;
143  }
```

Fig. 3 Codice di getDestination

La funzione *routing_to_cloud()* rappresenta la logica di dispatching dei job verso il cloud, seguendo le probabilità previste dal sistema (Wlan/Enode).

```
159  //Fornisce il codice del blocco di destinazione partendo dal blocco di controllo iniziale
160  //logica del dispatcher
161  int routing_to_cloud() {
162  double random = Uniform(0, 1);
163         if(random<=P_WLAN_CHOICE) {
164            return WLAN_UNIT;
165         } else {
166            return ENODE_UNIT;
167         }
168      }
```

Fig. 4 Codice routing_to_cloud()

Una volta decisa la destinazione, in base alla tipologia di Blocco viene decisa la logica di servizio.

Se il blocco presenta server liberi, viene effettuato il processamento di un nuovo completamento altrimenti il job viene messo in coda. Se il blocco destinazione è di tipo Video unit la logica differisce in quanto non vi è presente una coda, quindi se i serventi di video unit vengono trovati occupati, allora il job lascia il sistema. Se il blocco destinazione è di tipo cloud unit il job viene immediatamente servito in quanto il cloud unit ha infiniti serventi.

3.3.3.3 Statistiche

Per ogni centro vengono calcolate le statistiche $\bar{r}, \bar{l}, \bar{q}, \bar{x}, \bar{w}, \bar{d}, \bar{s}$.

Tali statistiche vengono calcolate ai fini della verifica e validazione. Queste vengono computate anche per il Central System nella sua totalità, in quanto per l'obiettivo dello studio è necessario calcolare il suo tempo medio di risposta.

3.4 VERIFICA

L'obiettivo di questa fase è comprendere se il modello computazionale realizzato è consistente con il modello delle specifiche. Sono stati eseguiti dei test per verificare le relazioni matematiche e logiche definite nel modello delle specifiche e la correttezza delle funzionalità implementate.

Per fare questo abbiamo verificato:

A. Il tempo di risposta ($\bar{w}$) per un server è sempre uguale alla somma del tempo di attesa ($\bar{d}$) e del tempo di servizio ($\bar{s}$).
   - $\bar{w} = \bar{s} + \bar{d}$
B. Se le probabilità di routing dichiarate nelle specifiche si riflettono correttamente nei rapporti tra arrivi ad un blocco/completamenti blocco origine.
C. Se per il blocco a perdita Video Unit : $Arrivals * (1 - Ploss) = Completed$.

Per valutare queste condizioni sono state eseguite diverse run ad orizzonte sia finito che infinito.

Riportiamo qui i risultati di una run nell'arco di $3600000\ ms$ ad orizzonte finito.

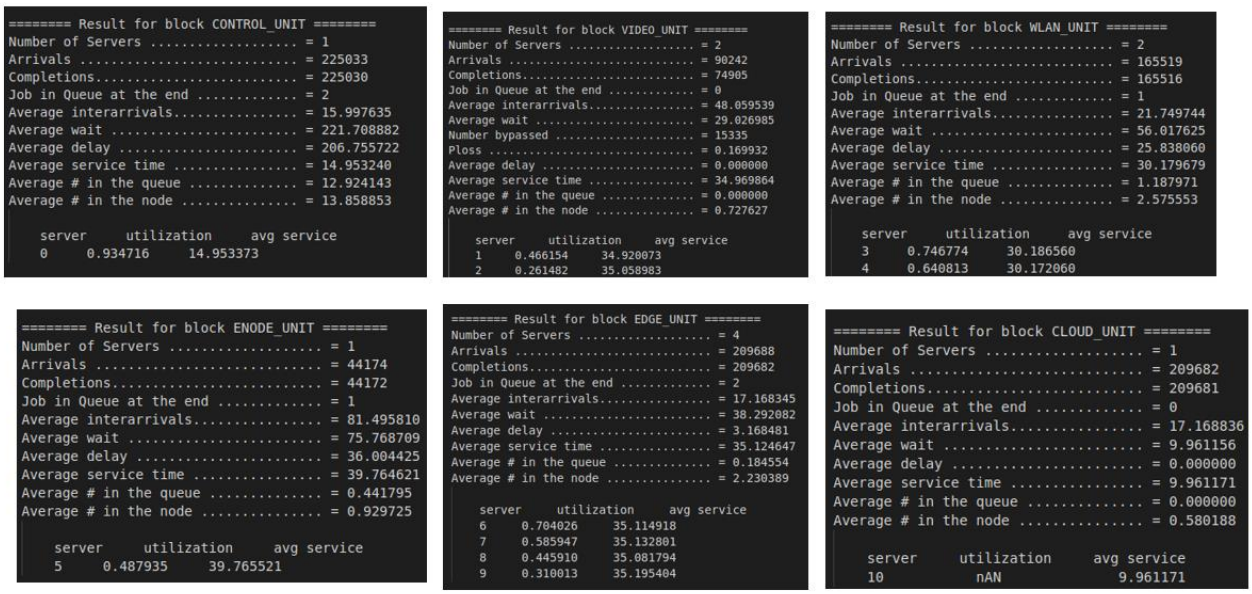

Fig. 5 Screen dei risultati

Verifica A : $\bar{w} = \bar{s} + \bar{d}$ (consistency check)

- ➔ **Control Unit** : $221.708882 = 14.953250 + 206.755722$
- ➔ **Video Unit** : $29.026985 = 0.0 + 34.969864 * (1 - 0.169932)$ (Sistema a perdita)
- ➔ **Wlan Unit** : $56.017625 = 25.838060 + 30.179679$
- ➔ **Enode Unit** : $75.768709 = 36.004425 + 39.764621$
- ➔ **Edge Unit** : $38.292082 = 3.168481 + 35.124647$
- ➔ **Cloud Unit** : $9.961156 = 9.961156 + 0.0$

Verifica B :

Vediamo che i job si distribuiscono secondo le probabilità di routing :

- ➔ **Video Unit** : $p_{internal} = 0.4 \rightarrow \frac{num.arrivi\ \ Video\ unit}{num.completamenti\ Control\ unit} = \frac{90244}{225030} = 0.4010$
- ➔ **Wlan:** $P_{wlanchoice} \cap P_{wlan_{on}} = 0{,}79 \rightarrow$

$$\frac{num.\ arrivi\ \ Wlan\ unit}{num.\ completamenti\ Control\ unit * 0.6 + num.\ completamenti\ Video\ Unit} = \frac{165520}{225030 * 0.6 + 74905} = 0.7884$$

- ➔ **Enode** : $1 - P_{wlanchoice} \cap P_{wlan_{on}} = 0{,}21 \rightarrow$

$$\frac{num.\ arrivi\ \ Enode\ unit}{num.\ completamenti\ Control\ unit * 0.6 + num.\ completamenti\ Video\ Unit} = \frac{44178}{225030 * 0.6 + 74905} = 0.210$$

Verifica C :

- ➔ **Video Unit** : $90242 * (1 - 0.169932) = 74906.99 \sim 74907$ (consinderando i job in fase di completamento)

Vediamo che il modello computazionale è effettivamente conforme al modello di specifica, quindi che l'implementazione del modello computazionale sia effettivamente corretta.

3.5 VALIDAZIONE

L'obiettivo di questa fase è comprendere se il modello computazionale realizzato sia consistente con il sistema che si deve analizzare, ovvero se esso sia un'approssimazione ragionevole del sistema.

Non avendo un dataset con dati reali a disposizione, si opera una validazione rispetto al modello analitico, quindi verificando che i risultati ottenuti rispettino le leggi teoriche. Per ottimizzare la fase di validazione è stato sviluppato un foglio Excel in cui sono espresse le leggi teoriche, in modo che modificando i parametri di input vengano automaticamente calcolati tutti i risultati attesi. Questo ha permesso di avere durante ogni fase di simulazione un immediato riscontro teorico, potendo quindi intervenire prontamente quando sono state rilevate delle imprecisioni.

Innanzitutto, scriviamo le relazioni analitiche tra i componenti del sistema :

$$\begin{cases} \lambda_1 = \gamma \\ \lambda_2 = p_{internal} * \lambda_1 (1 - P_{loss}) \\ \lambda_3 = (p_{external} * \lambda_1 + \lambda_2) * p_{wlan_{choice}} * p_{wlan_{on}} \\ \lambda_4 = (p_{external} * \lambda_1 + \lambda_2)(1 - p_{wlan_{choice}} * p_{wlan_{on}}) \\ \lambda_5 = \lambda_3 + \lambda_4 \\ \lambda_6 = \lambda_5 \end{cases}$$

In seguito, le formule analitiche per i diversi componenti:

<table>
<tr>
<td>Componenti M/M/1
• $E\left[T_Q\right] = E[T] - \frac{1}{\mu} = \frac{\rho}{\mu - \lambda} = \frac{\rho E[S]}{1-\rho}$
• $E[T] = \frac{E[N]}{\lambda} = \frac{1}{\mu - \lambda} = \frac{E[S]}{1-\rho}$</td>
<td>Componenti M/M/K
• $E[S_i] = \frac{1}{\mu}$
• $E[S] = \frac{1}{m\mu}$
• $E\left[T_q\right] = \frac{1}{\lambda} P_q \frac{\rho}{1-\rho} = \frac{P_q E[S]}{1-\rho}$ $(Erlang\ C)$
• $E[T] = E\left[T_q\right] + kE[S]$</td>
</tr>
<tr>
<td>Componente M/M/k/k
• $P_{loss} = \pi_k = \frac{\left(\frac{\lambda}{\mu}\right)^k / k!}{\sum_{j=0}^{k} \left(\frac{\lambda}{\mu}\right)^j \frac{1}{j!}}$ $(Erlang\ B)$
• $\pi_0 = \frac{1}{\sum_{i=0}^{c} \left(\frac{\lambda}{\mu}\right)^i * \frac{1}{i!}}$
• $\lambda^i = \lambda(1 - P_{loss})$</td>
<td>Componente M/M/∞
• $E[T] = \frac{1}{\mu}$</td>
</tr>
<tr>
<td colspan="2">• $\rho = \begin{cases} \frac{\lambda}{\mu} = \lambda\, E[S] & se\ m = 1 \\ \frac{\lambda}{m\mu} = \frac{\lambda\, E[S]}{m} & se\ m > 1 \end{cases}$
• $R = \sum_{i=0}^{n} V_i R_i$ (Legge generale del tempo di risposta, usata per calcolare R)</td>
</tr>
</table>

La tabella dei dati è stata riprodotta estrapolando i dati delle statistiche dalla simulazione effettuata ad orizzonte finito durante $3600000\ ms$ (durata di simulazione), con numero di repliche pari a 128.
In particolari sono stati presi i vari csv creati dalla simulazione e utilizzate le funzione $estimate.c$.

Questo ci ha permesso di calcolare il valore dell'intervallo di confidenza del 95%.

VALORE TEORICO	**VALORE SIMULATIVO $\pm\alpha = 0,05$**
$\boldsymbol{E[Ts] = 352,1913698}$	$\boldsymbol{E[Ts] = 349,309809 \pm 3.518897}$

```
based upon 128 data points and with 95% confidence
the expected value is in the interval 349.309809 +/- 3.518897
```

Fig. 6 Stamp dell'esecuzione di estimate.c

```
for a sample of size 128
mean ................. = 349.310
standard deviation ... =  20.040
minimum .............. = 300.449
maximum .............. = 416.696
```

Fig. 7 stamp dell'esecuzione di uvs.c

Possiamo quindi vedere come il modello simulativo rispetti le leggi del modello analitico. Anche non avendo dati reali a disposizione questo ci conferma che il nostro modello rappresenta bene un modello reale, e possiamo quindi proseguire con la fase di analisi dei risultati.

3.6 Progettazione ed esecuzione degli Esperimenti

I risultati di ogni simulazione vengono scritti su un file csv differente, analizzabile poi tramite estimate.c e/o uvs.c per ottenere i valori di media, deviazione standard ed intervallo di confidenza.

Sono state pianificate delle simulazioni per:

- Analisi dello stato transiente attraverso il metodo delle repliche
- Analisi dello stato stazionario attraverso il metodo dei batch means

3.6.1 Simulazione ad orizzonte finito

Un'analisi dello stato transiente è permessa tramite la simulazione ad orizzonte finito.

Nella simulazione ad orizzonte finito viene effettuata una simulazione del sistema lungo $3600000\ ms$. Per effettuare un'analisi statistica dei valori ottenuti ad orizzonte finito, il procedimento di misurazione è stato replicato 128 volte. Ogni replica viene utilizzata per misurare le stesse statistiche, e fornisce quindi un punto del nostro campione. Per ottenere la media campionaria e l'intervallo di confidenza al 95% si è utilizzato il programma estimate.c, che internamente utilizza la distribuzione Student.

Come da linee guida il seed viene impostato tramite *PlantSeeds()* fuori dal ciclo di replicazione, mentre per le prove successive alla prima viene usato come stato iniziale di ogni stream rng lo stato finale degli stessi stream della replica precedente. In particolare, viene effettuata una GetSeed() e impiantato lo stato finale del seed della replica precedente, nello stato iniziale della replica successiva.

Facendo ciò si evitano possibili sovrapposizioni degli eventi generati dalle singole repliche.

Il seed iniziale per questa simulazione è 231232132.

Per analizzare le statistiche ottenute nel continuo, per ogni ripetizione, effettuiamo ogni $1000\ ms$ una misurazione del tempo di risposta, che viene salvato in file *csv*.

Nella nostra analisi, la simulazione ad orizzonte finito ha come obiettivi principali quello di analizzare il comportamento del sistema verificando che le statistiche individuate ad orizzonte finito coincidono effettivamente anche nel sistema reale.

3.6.2 Simulazione ad orizzonte infinito

Nella simulazione ad orizzonte infinito il sistema viene simulato per un tempo di simulazione "infinito", quindi di molto superiore al tempo reale. Questa simulazione ha permesso l'analisi dello stato stazionario, in questo modo si producono le statistiche a stato stazionario del sistema. Per ridurre il bias dello stato iniziale si effettua una run di simulazione più lunga. Per ricavare la media campionaria del tempo di risposta si utilizza il metodo delle *Batch Means*, suddividendo la run di simulazione in *k* batches di dimensione b. Da ogni batch si ricavano le statistiche tramite *calculate_statistics_inf().* Si resettano quindi le statistiche tramite *reset_statistics(),* ma senza azzerare lo stato del sistema, ovvero mantenendo tutti i job rimanenti dal precedente batch. In questo modo si genera un campione di *k* batches indipendenti, sul quale è possibile valutare la media campionaria. Le dimensioni di *b* e *k* utilizzate influenzano la *qualità* del campione; un valore di b grande permette di avere un campione con bassa autocorrelazione, mentre un maggior numero *k* di batch permette di avere un valore migliore in termini di intervallo di confidenza. Sono stati quindi utilizzati $k = 128\ batch$, e seguendo le linee guida proposte da Banks, Carson, Nelson, and Nicol si è cercato di individuare il valore di b per cui l'autocorrelazione del campione prodotto è inferiore a 0.2 per lag j = 1. E' stata definita la funzione *find_batch_b()* per generare un campione al variare di diversi valori di b. Su ogni campione è stata valutata quindi l'autocorrelazione tramite il programma *acs.c*. Si è visto che per $b = 120000$ si ha un'autocorrelazione di 0.146 per j=1, quindi si è scelto tale valore come dimensione del batch.

```
for 128 data points
the mean is ...    349.31
the stdev is ..     20.04

  j (lag)   r[j] (autocorrelation)
  1         0.146
  2         0.163
  3         0.310
  4         0.159
  5         0.410
  6         0.545
  7         0.976
  8         0.926
```

Fig. 8 stamp dell'esecuzione di acs.c

In totale, per ogni simulazione vengono quindi processati $b * k = 15.360.000\ job$.

L'obiettivo della simulazione ad orizzonte infinito ai fini del nostro studio è l'analisi dei tempi di risposta del sistema a steady state e verifica del soddisfacimento dei *QoS*.

3.7 ANALISI RISULTATI

I csv per effettuare le analisi sottostanti si trovano nella directory di progetto /results/alg1/.

Per la riproduzione dei grafici sono stati usati degli script in python.

3.7.1 Analisi ad orizzonte finito

L'analisi ad orizzonte finito ci ha permesso di effettuare l'analisi transiente.

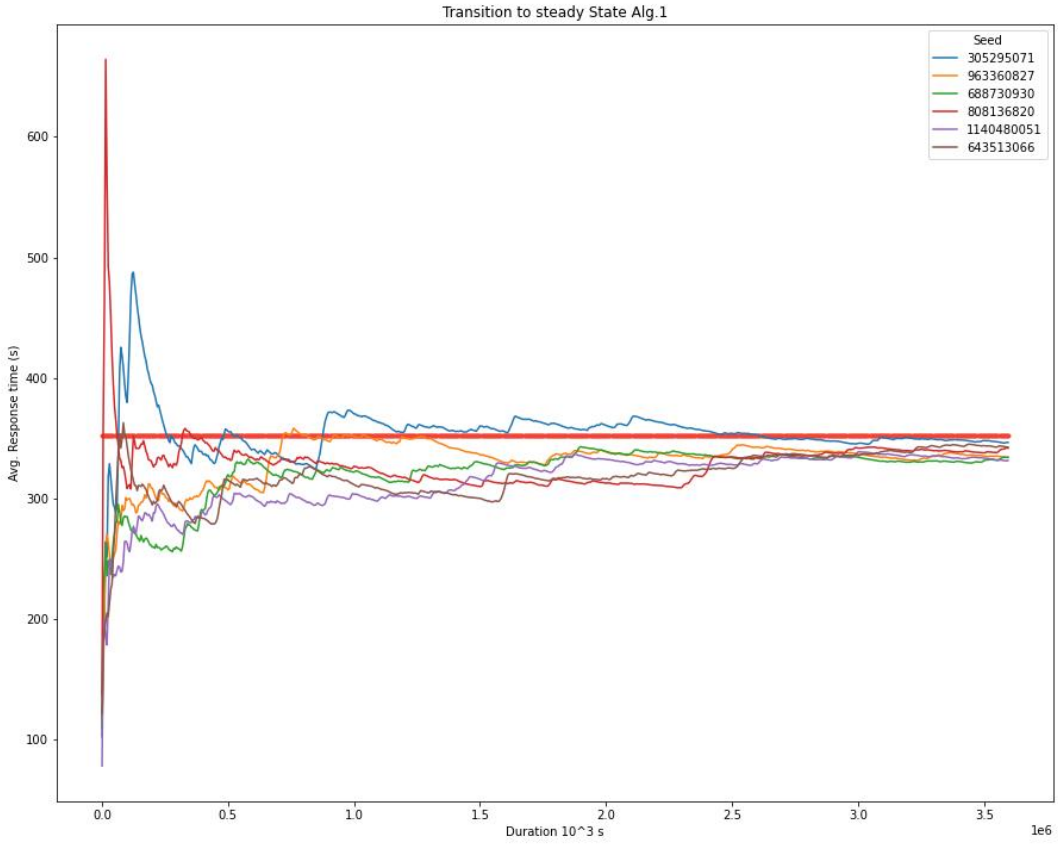

Grafico 1 ; Analisi della transizione del sistema verso lo stato stazionario (alg.1)

Nel grafico 1, sulle ascisse è riportata la durata della simulazione, sulle ordinate il tempo medio di risposta e la retta orizzontale rossa rappresenta il valore teorico del tempo di risposta medio.
Sono rappresentate 6 “curve” relative a 6 seed iniziali differenti. Dal grafico si osserva che, come ci si aspettava, per periodi di simulazione brevi i valori ottenuti per seed diversi si discostano tra di loro e dal valore teorico (il sistema si trova nello stato transiente). A partire da periodi di circa 3000000 ms, le “curve” tendono a convergere verso il valore teorico (il sistema passa allo stato stazionario).

Dall’analisi precedente, si è deciso pertanto di utilizzare 3000000ms come durata massima delle simulazioni per l’analisi transiente.

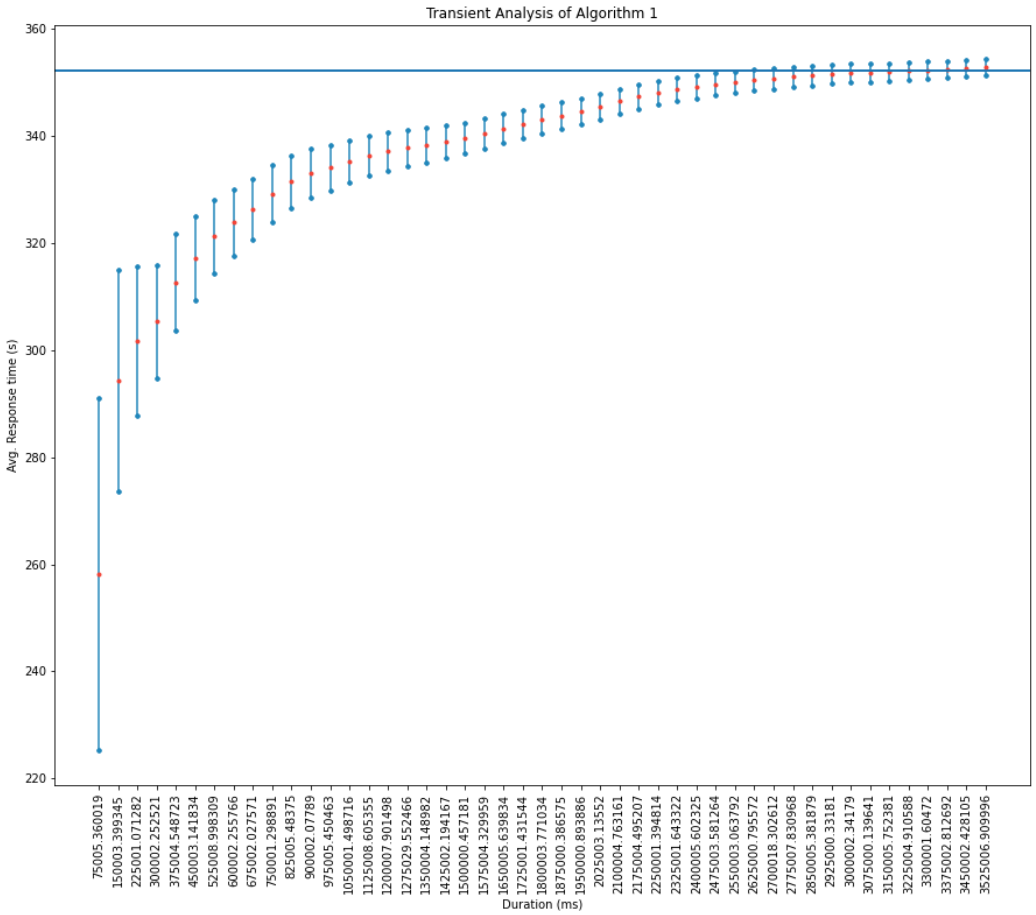

Grafico 2 ; Transient Analysis of Alg.1

Nel grafico 2, sulle ascisse viene riportata la durata delle simulazioni, sulle ordinate il tempo medio di risposta del e la linea orizzontale rappresenta il valore teorico. Fissato un valore per la durata delle simulazioni 75000ms, i dati ottenuti dalle repliche eseguite vengono utilizzati per determinare gli intervalli

di confidenza. Si osserva che a seguito dei primi ensemble di simulazioni (aventi durata di simulazione pari da 75000ms fino a 2400000ms), gli altri intervalli di confidenza contengono il valore teorico.

Si nota inoltre che a mano a mano che aumenta la durata delle simulazioni, l'ampiezza dell'intervallo di confidenza diminuisce, a causa della riduzione della varianza dei dati.

3.7.2 Analisi ad orizzonte infinito

Si analizza l'output sull'analisi dello stato stazionario.
Nel grafico 3 vengono riportati i dati ottenuti utilizzando il metodo dei batch means, con $B = 120.000, K = 128$. Sulle ascisse sono riportati i seed iniziali utilizzati, sulle ordinate il tempo medio di risposta e la linea orizzontale rappresenta il valore teorico. Si osserva che si ha una copertura alta, l'intervallo di confidenza relativo contiene il valore teorico. Il secondo e terzo ensemble sembrano discostarsi maggiormente dal valore teorico.

Tale grafico è stato realizzato per verificare la convergenza del sistema, indipendentemente dal seed iniziale utilizzato. Va inoltre sottolineato che l'ampiezza degli intervalli di confidenza rappresentati è abbastanza piccola.

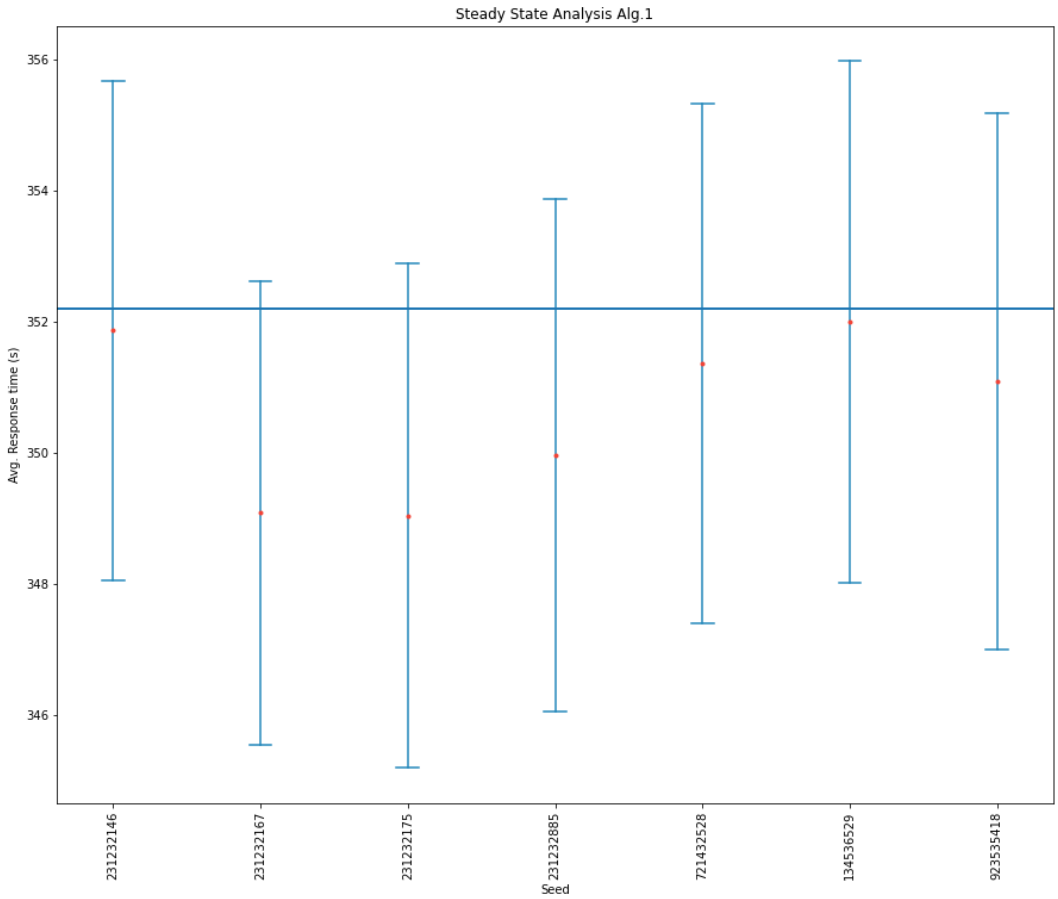

Grafico 3 Steady State Analysis Alg.1

3.8 CONSIDERAZIONI FINALI

Il blocco Control Unit rappresenta il collo di bottiglia del sistema in analisi.

Si evince dal grafico che il tempo di risposta relativo al blocco control unit rappresenta più del 50% del tempo totale speso da un job all'interno del sistema.

E' considerevole quindi procedere con il miglioramento abbattendo questi tempi spesi sul control unit.

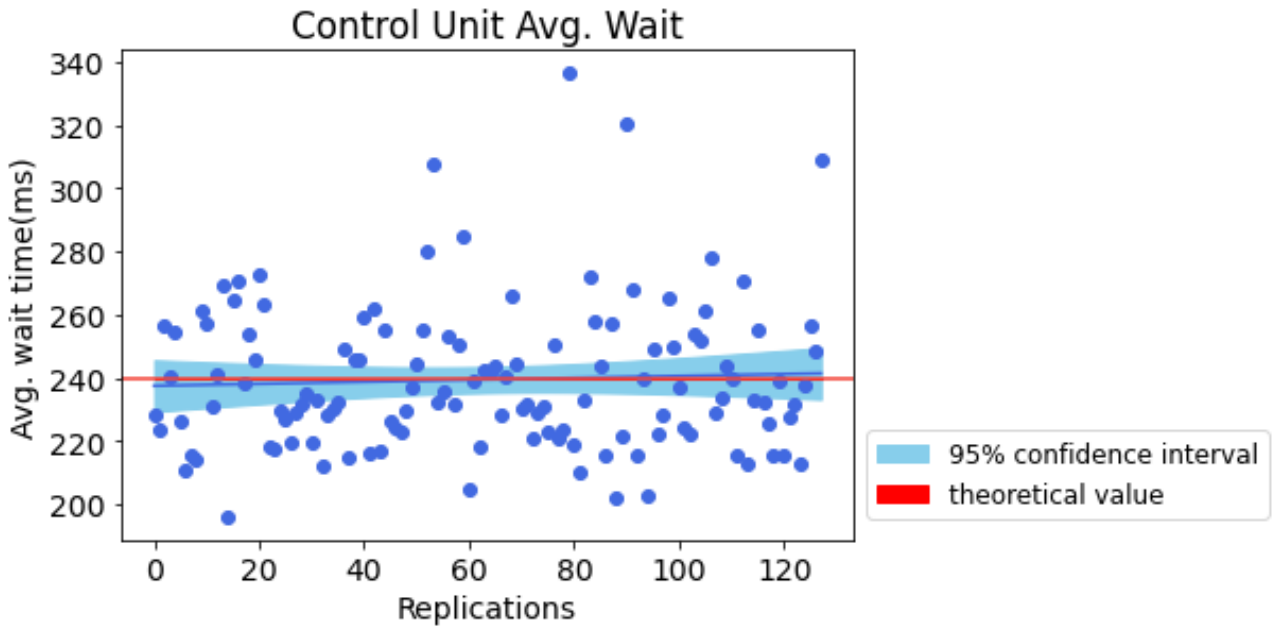

Fig. 9 Control Unit Average wait time – 128 replications

Dati i risultati ottenuti, si è considerato di eliminare il collo di bottiglia modificando Control Unit da single core e dual core.

Un miglioramento ulteriore si potrebbe avere effettuando il bilanciamento tra Enode e Wlan Unit.

4 ALGORITMO MIGLIORATIVO

Obiettivo dell'algoritmo migliorativo è eliminare il collo di bottiglia su Control Unit ed effettuare un bilanciamento tra Enode e Wlan Unit per migliorare i tempi di risposta complessivi.

Ci aspettiamo che il tempo di risposta totale medio non sia minore dei 100 ms, limite inferiore dei tempi di risposta medi misurati per lo streaming video.

4.1 MODELLO CONCETTUALE

La differenza principale del modello rispetto all'algoritmo 1 sono i sottosistemi *local* e *networking*.

- *Local* : Il blocco control Unit, che risulta essere collo di bottiglia nell'algoritmo 1, diventa un blocco dual core M/M/2.
- *Networking* : Il sottosistema networking risulta ora essere bilanciato. Il blocco eNodeB diventa un blocco dual core M/M/2.

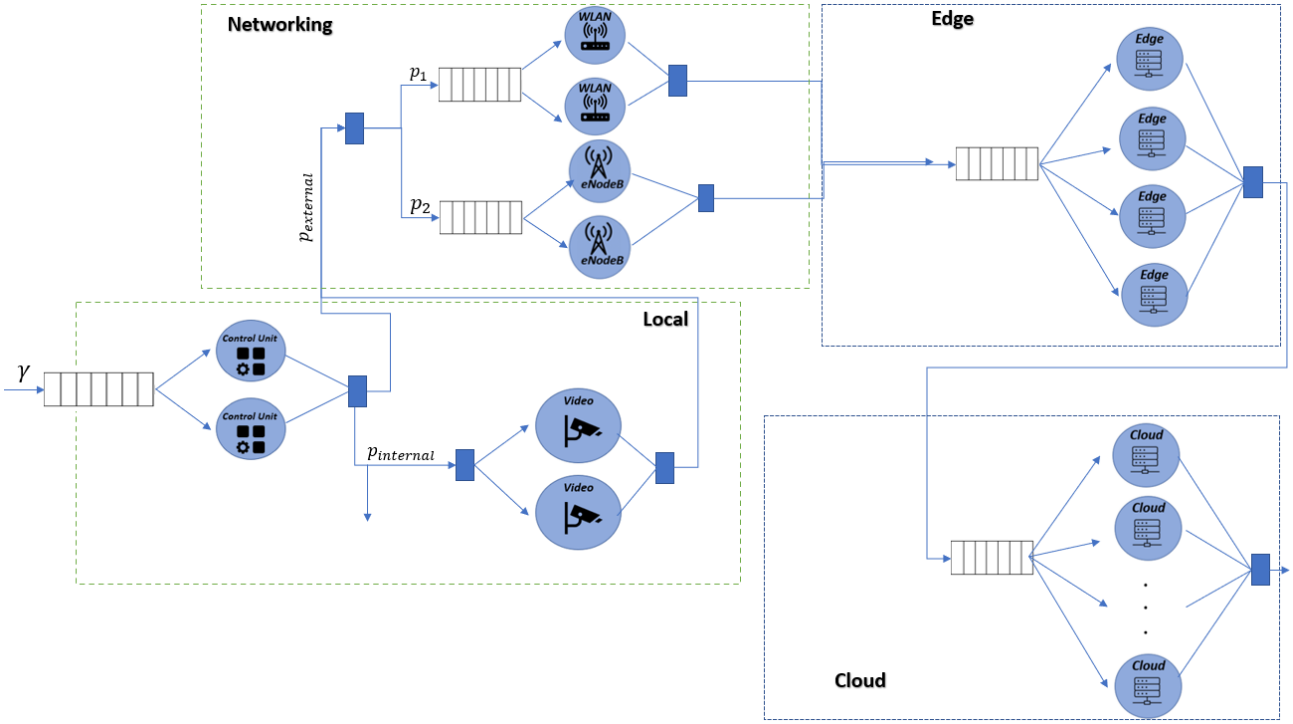

Fig. 10 Modello concettuale dell'algoritmo migliorativo

Le *variabili di stato* considerate sono :

- Stato di un servente (libero/occupato).

Gli *eventi* che possono esserci :

- Arrivo di una richiesta

4.2 MODELLO DELLE SPECIFICHE

I tempi di interarrivo ed i tempi di servizio sono modellati secondo una distribuzione Esponenziale, che rappresenta una buona scelta per il sistema in esame. Il periodo di osservazione è lo stesso utilizzo dall'algoritmo 1, anche i tempi di servizio dei server dei blocchi rimane invariato.

Innanzitutto, per effettuare il miglioramento del bilanciamento tra Enode e Wlan sono state fatte delle considerazioni per calcolare la nuova probabilità di routing .

$$\begin{cases} \lambda_1 = \gamma \\ \lambda_2 = p_{internal} * \lambda_1(1 - P_{loss}) \\ \lambda_3 = (p_{external} * \lambda_1 + \lambda_2) * p_{wlan_{choice}} * p_{wlan_{on}} \\ \lambda_4 = (p_{external} * \lambda_1 + \lambda_2)(1 - p_{wlan_{choice}} * p_{wlan_{on}}) \\ \lambda_5 = \lambda_3 + \lambda_4 \\ \lambda_6 = \lambda_5 \\ \rho_3 = \rho_4 \end{cases}$$

$$\rightarrow p_{wlan_{choice}} * p_{wlan_{on}} = p_1 = 1 - 0.428571 = 0{,}571429$$

Probabilità di routing:

• $\boldsymbol{P_{internal} = 0,4}$	La probabilità che il blocco Control Unit invii la sua richiesta al Video Unit.
• $\boldsymbol{P_{wlanchoice} \cap P_{wlan_{on}} = 0,571429 = p_1}$	La probabilità rappresenta in effetti la probabilità di instradamento delle richieste verso il blocco Wlan.

4.3 MODELLO COMPUTAZIONALE

Il modello computazionale dell'algoritmo migliorativo non differisce di molto rispetto a quello utilizzato per implementare il sistema originale. Per come è stato impostato il codice di base, è stato facile aggiungere un core al control Unit e modificare la probabilità per consentire il bilanciamento tra Enode e Wlan.

4.4 VERIFICA

Le verifiche di questa fase sono le stesse di quelle del precedente algoritmo 1. E' stata aggiunta una verifica aggiuntiva per accertarsi del bilanciamento.

In particolare:

A. Il tempo di risposta ($\bar{w}$) per un server è sempre uguale alla somma del tempo di attesa ($\bar{d}$) e del tempo di servizio ($\bar{s}$). $\bar{w} = \bar{s} + \bar{d}$
B. Se le probabilità di routing dichiarate nelle specifiche ritornano nei rapporti tra ingressi e uscite.
C. Se per il blocco a perdita Video Unit : Arrivals *(1-Ploss)= Completed.
D. Bilanciamento Enode e Wlan

Per valutare queste condizioni sono state eseguite diverse run ad orizzonte sia finito che infinito. Riportiamo qui i risultati di una run ad orizzonte finito nell'arco di $3600000\ ms$.

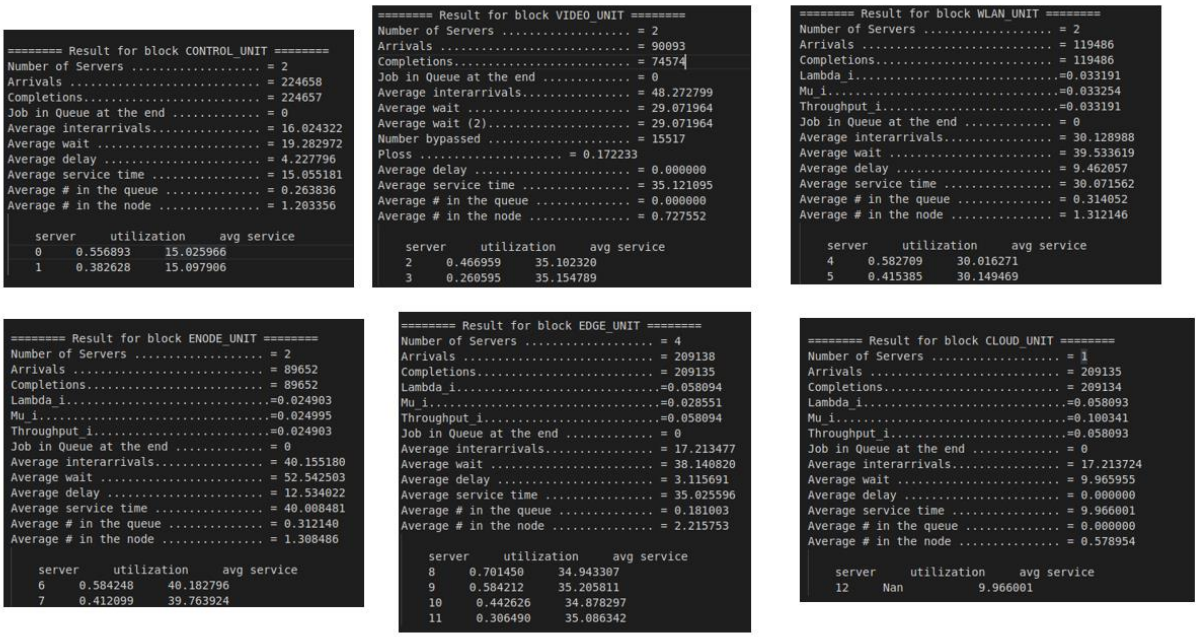

Fig. 11 Output

Verifica A : $\overline{w} = \bar{s} + \bar{d}$ (consistency check)

- ➔ **Control Unit** : $19.2829 = 16.024 + 4.227$
- ➔ **Video Unit** : $29.026985 = 0.0 + 35.121095 * (1 - 0.172233)$ (Sistema a perdita)
- ➔ **Wlan Unit** : $39.5336 = 30.1289 + 9.46$
- ➔ **Enode Unit** : $52.5425 = 40.115 + 12.5340$
- ➔ **Edge Unit** : $38.1408 = 3.11156 + 35.0255$
- ➔ **Cloud Unit** : $9.9659 = 9.9659 + 0.0$

Verifica B :

Verichiamo le probabilità di routing:

- ➔ **Video Unit** : $p_{internal} = 0.4 \rightarrow \frac{num.arrivi\ \ Video\ unit}{num.completamenti\ Control\ unit} = 0.4010$
- ➔ **Wlan:** $P_{wlanchoice} \cap P_{wlan_{on}} = 0{,}5714 \rightarrow$

$$\frac{num.\ arrivi\ \ Wlan\ unit}{num.\ completamenti\ Control\ unit * 0.6 + num.\ completamenti\ Video\ Unit} = 0{,}5714$$

Verifica C :

- ➔ **Video Unit** : $90093 * (1 - 0.1722) = 74575.25$ (considerando i job in fase di completamento)

Verifica D:

- ➔ Si evince che l'utilizzazione del blocco wlan risulta essere uguale a quella del blocco enode

4.5 VALIDAZIONE

Nella fase di validazione si confrontano i risultati della simulazione rispetto al modello analitico.

La tabella dei dati è stata riprodotta estrapolando i dati delle statistiche dalla simulazione effettuata ad orizzonte finito durante $3600000\ ms$ (durata di simulazione), con numero di repliche pari a 128.
In particolari sono stati presi i vari csv creati dalla simulazione e utilizzate le funzione $estimate.c$. Questo ci ha permesso di calcolare il valore dell'intervallo di confidenza del 95%.

VALORE TEORICO	**VALORE SIMULATIVO $\pm\alpha = 0,05$**
$E[Ts] = 116{,}4626$	$E[Ts] = 116.014692 \pm 0.698782$

Possiamo quindi vedere come il modello simulativo rispetti le leggi del modello analitico. Anche non avendo dati reali a disposizione questo ci conferma che il nostro modello rappresenta bene un modello reale, e possiamo quindi proseguire con la fase di analisi dei risultati.

4.5.1 Simulazione ad orizzonte finito e infinito

Nella simulazione ad orizzonte finito viene effettuata una simulazione del sistema lungo $3600000\ ms$.
Per effettuare un'analisi statistica dei valori ottenuti ad orizzonte finito, il procedimento di misurazione è stato replicato 128 volte, ottenendo quindi un ensemble di dimensione pari a 128. Ogni replica viene utilizzata per misurare le stesse statistiche, e fornisce quindi un punto del nostro campione. Per ottenere la media campionaria e l'intervallo di confidenza al 95% si è utilizzato il programma estimate.c, che internamente utilizza la distribuzione Student.

```
based upon 128 data points and with 95% confidence
the expected value is in the interval 116.014692 +/- 0.698782
```

Fig. 12 Stamp output estimate.c

```
for a sample of size 128
mean ................. = 116.015
standard deviation ... =   3.980
minimum .............. = 105.203
maximum .............. = 128.420
```

Fig. 13 Stamp output uvs.c

Per la simulazione ad orizzonte infinito si è utilizzata la stessa tecnica di simulazione utilizzata per l'algoritmo 1. Si è visto che per $b = 1300$, $k = 128$ si ha un'autocorrelazione di 0.016 per j=1, quindi si è scelto tale valore come dimensione del batch.

```
for 128 data points
the mean is ...    116.01
the stdev is ..      3.98

  j (lag)   r[j] (autocorrelation)
  1        0.016
  2        0.125
  3        0.607
  4        1.233
  5        1.167
  6        1.186
  7        1.350
  8        1.312
```

Fig. 14 Stamp output acs.c

In totale, per ogni simulazione vengono quindi processati $b * k = 166400\ job$.

4.6 ANALISI RISULTATI

I csv per effettuare le analisi sottostanti si trovano nella directory di progetto /results/alg1_migliorativo/. Per la riproduzione dei grafici sono stati usati degli script in python.

4.6.1 Analisi ad orizzonte finito

L'analisi ad orizzonte finito ci ha permesso di effettuare l'analisi transiente.

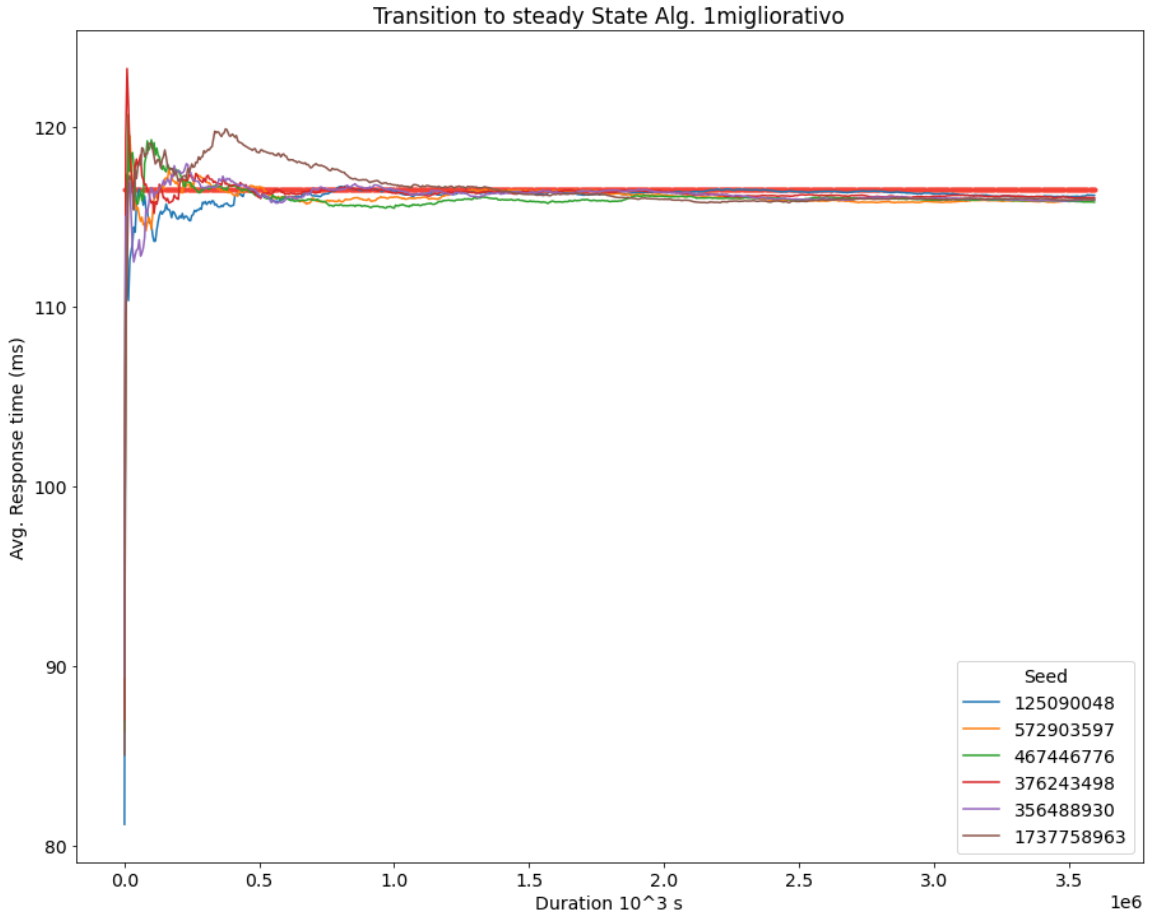

Grafico 4 Transition to steady State alg1

Nel grafico 4, sulle ascisse è riportata la durata della simulazione, sulle ordinate il tempo medio di risposta e la retta orizzontale rossa rappresenta il valore teorico del tempo di risposta medio.
Sono rappresentate 6 “curve” relative a 6 seed iniziali differenti. Dal grafico si osserva che, come ci si aspettava, per periodi di simulazione brevi i valori ottenuti per seed diversi si discostano tra di loro e dal valore teorico (il sistema si trova nello stato transiente). A partire da periodi di circa 1500000ms, le “curve” tendono a convergere verso il valore teorico (il sistema passa allo stato stazionario). Dall’analisi precedente, si è deciso pertanto di utilizzare 1500000ms come durata massima delle simulazioni per l’analisi transiente.

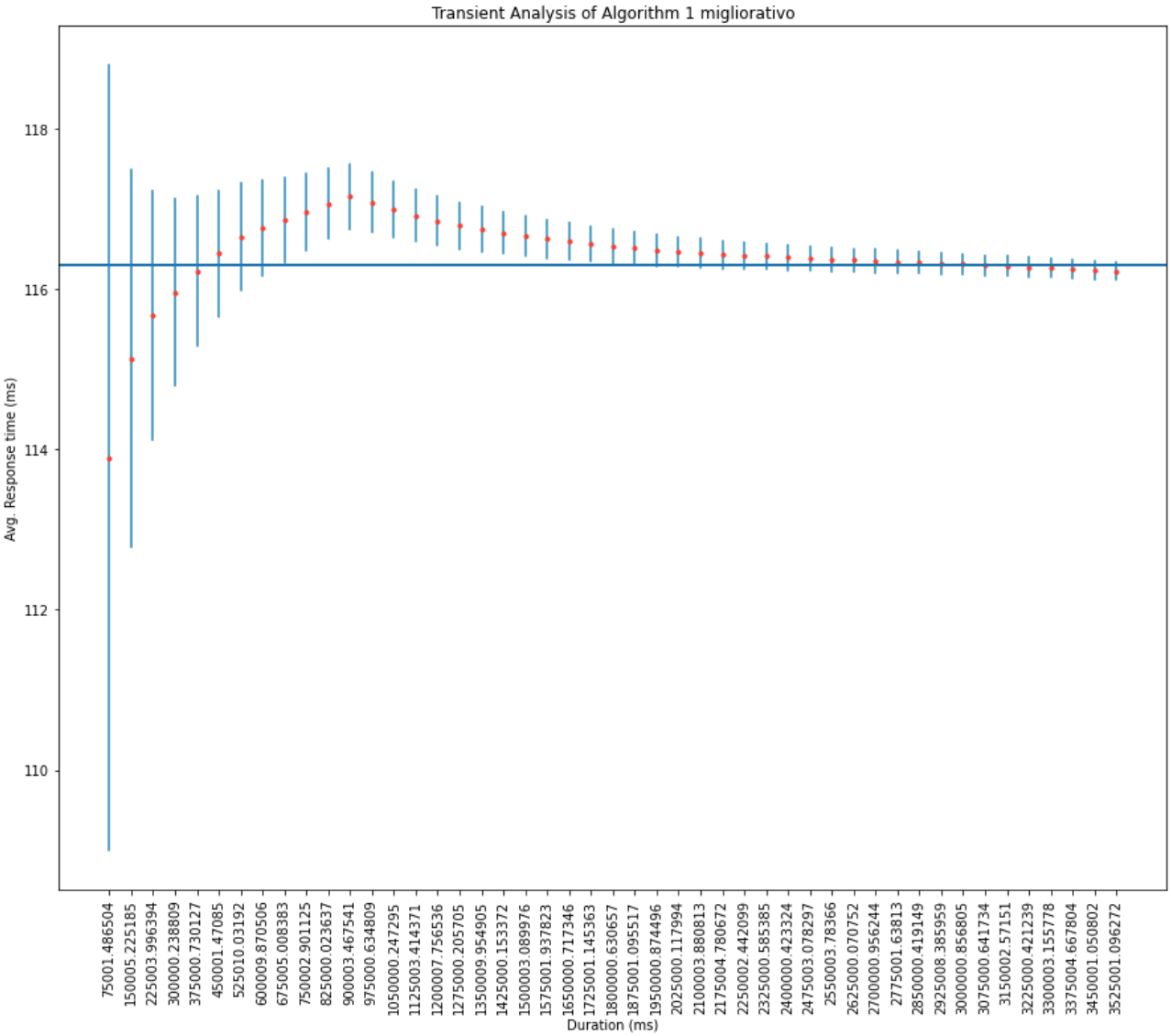

Grafico 5 Transient Analysis of algorithm 1 migliorativo

Nel grafico 5, sulle ascisse viene riportata la durata delle simulazioni, sulle ordinate il tempo medio di risposta del e la linea orizzontale rappresenta il valore teorico. Fissato un valore per la durata delle simulazioni 75000ms, i dati ottenuti dalle repliche eseguite vengono utilizzati per determinare gli intervalli

di confidenza. Si osserva che a seguito dei primi ensemble di simulazioni (aventi durata di simulazione pari da 75000ms fino a 1500000ms), gli altri intervalli di confidenza contengono il valore teorico.

Si nota inoltre che a mano a mano che aumenta la durata delle simulazioni, l'ampiezza dell'intervallo di confidenza diminuisce, a causa della riduzione della varianza dei dati.

4.6.2 Analisi ad orizzonte infinito

Si analizza l'output sull'analisi dello stato stazionario.
Nel grafico successivo vengono riportati i dati ottenuti utilizzando il metodo dei batch means, con $B = 1300, K = 128$. Sulle ascisse sono riportati i seed iniziali utilizzati, sulle ordinate il tempo medio di risposta e la linea orizzontale rappresenta il valore teorico. Si osserva che si ha una copertura di circa 80%

Tale grafico è stato realizzato per verificare la convergenza del sistema, indipendentemente dal seed iniziale utilizzato.

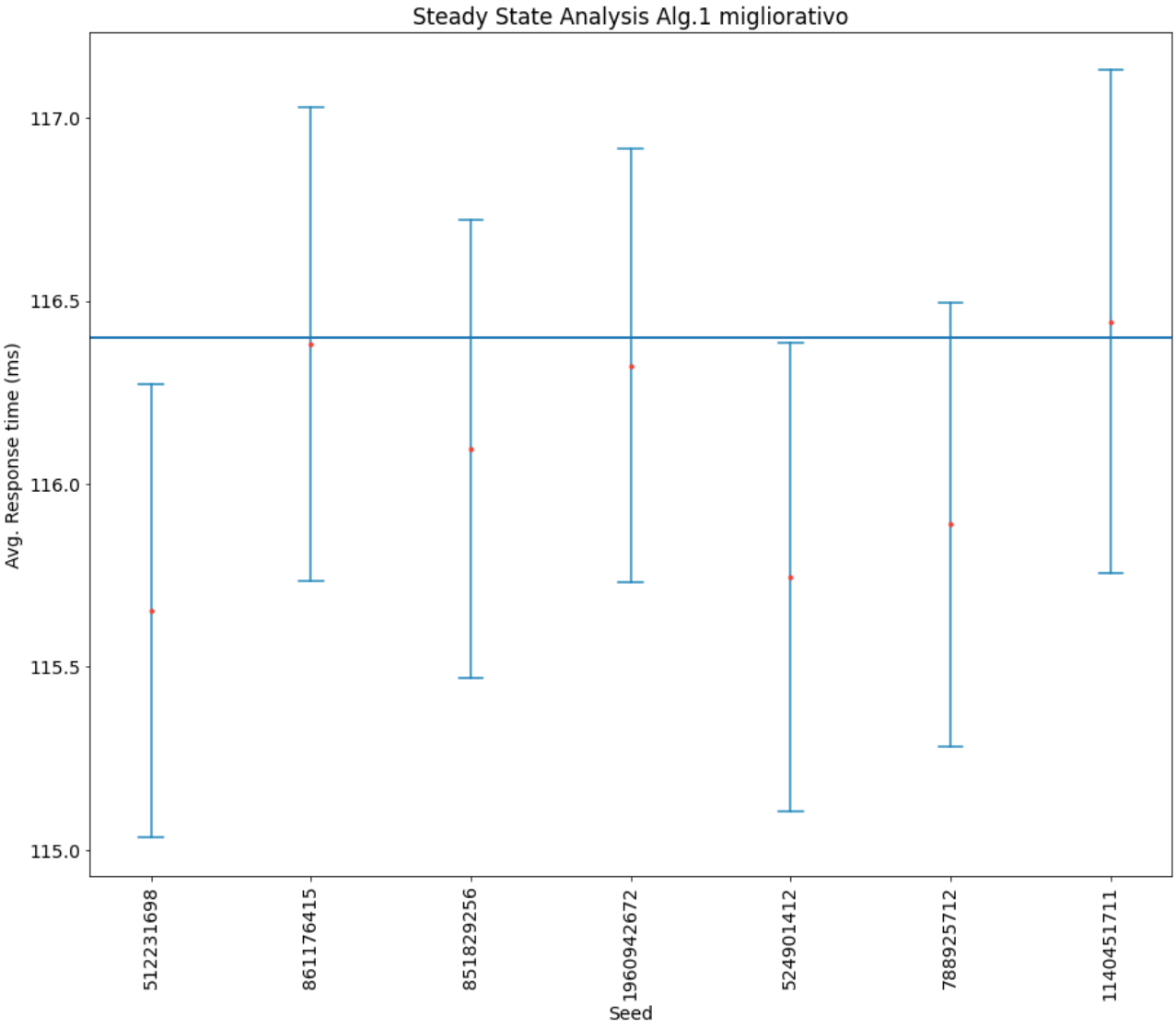

Grafico 6 Steady state analysis of alg1 migliorativo

4.7 CONSIDERAZIONI FINALI

Come si evince dal grafico 7 sono stati abbattuti i tempi di risposta spesi dai job sul control unit.

In particolare, ora il tempo speso di un job in control unit è circa il 15% del tempo totale di un job nel sistema. Confrontando invece i tempi di risposta con l'algoritmo 1, si è passati da una media circa di 240ms a 19ms.

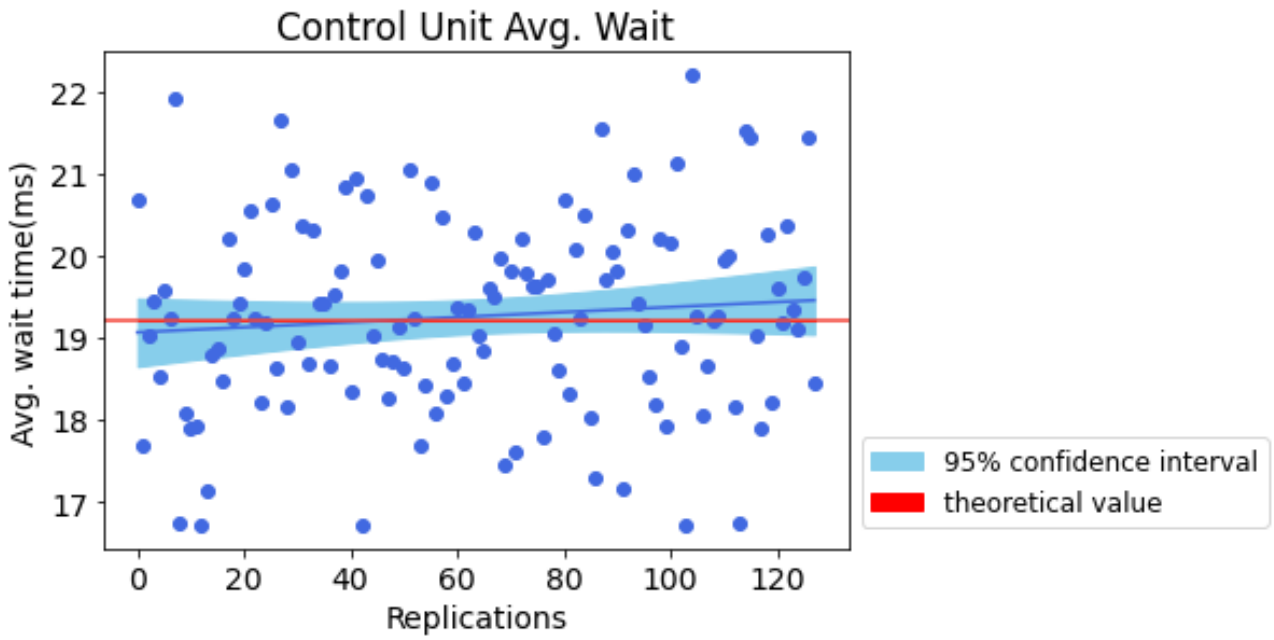

Grafico 7 Avg Wait time Control Unit

L'obiettivo dell'algoritmo migliorativo è stato raggiunto.

5 ALGORITMO 2

5.1 5G

5G Mobile Edge Computing (MEC), noto anche come Multi-access edge computing, è un'architettura di rete che combina elementi di tecnologia IT e cloud computing per fornire un framework di rete software ai margini di una rete cellulare 5G.

L'implementazione di MEC nei nodi perimetrali di una rete 5G può fornire un utilizzo efficiente della rete delle risorse di elaborazione e comunicazione. Ciò porterà a processi di rete off-load e ridurrà la congestione complessiva della rete.

5G MEC può aumentare l'esperienza dell'utente eseguendo applicazioni più vicine all'utente di celle 5G.

MEC può, quindi, aiutare gli operatori mobili a sviluppare nuove applicazioni 5G e nuovi servizi 5G in modo più flessibile e scalabile.

5.2 OBIETTIVO DELL'ALGORITMO 2

Obiettivo dell'algoritmo 2 è studiare il comportamento del sistema cambiando il sottosistema networking. La rete 5G viene utilizzata come unico strumento di accesso all'Edge, togliendo il dualismo rete Wifi e rete

4G. Il motivo di questo ulteriore algoritmo è dare un'impronta più moderna e con maggior risparmio energetico, mantenendo prestazioni elevate del sistema. Lo sviluppo di questa nuova rete parte dall'interesse reciproco dello sviluppo delle nuove reti moderne. Ci è sembrato interessante cercare di capire come potrebbe essere strutturata una rete con questo nuovo componente e capire in base ai nostri obiettivi i vari vantaggi e svantaggi.

5.3 MODELLO CONCETTUALE

Il sottosistema Networking varia considerevolmente rispetto agli algoritmi precedenti, mentre la configurazione dei sottosistema Local, Edge e Cloud Computation risulta uguale alle configurazioni dell'algoritmo migliorativo.

- *Networking*: il blocco networking si occupa dell'uploading/downloading nella rete dei frame video tramite tecnologia 5g. E' rappresentato da una coda M/M/2.

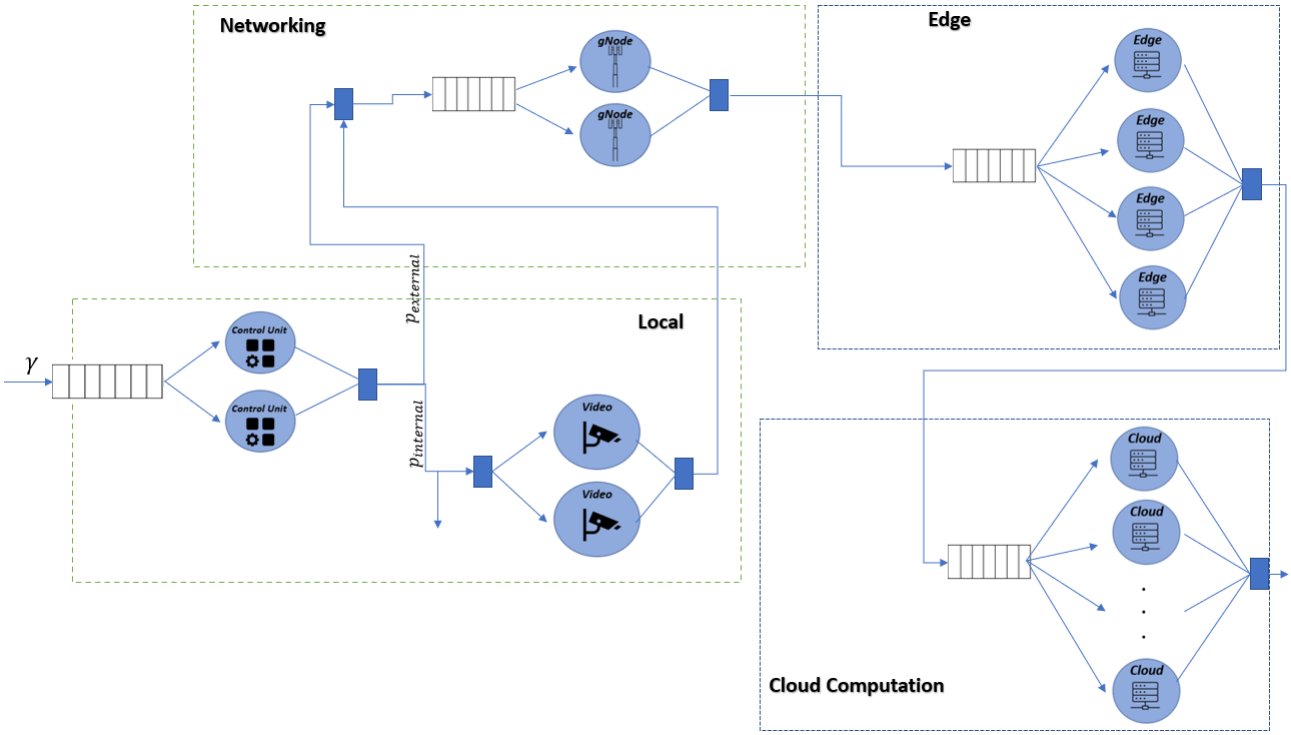

Fig. 15 Modello concettuale algoritmo 2

5.4 MODELLO DELLE SPECIFICHE

Il periodo di osservazione per questo modello è lo stesso dell'algoritmo 1.

Il tempo di servizio associato al gNode rappresentante un servente della rete 5G è $30.0\ ms$.

Il resto dei tempi di servizio è uguale alle specifiche dell'algoritmo 1.

Le probabilità di routing, in particolare $\boldsymbol{P_{internal}}$ **e** $\boldsymbol{P_{external}}$ rimangono uguali, spariscono le probabilità legate al Wlan Unit e a eNode Unit. Tutto ciò che esce dal Control Unit per la sua probabilità di routing e quello che esce da Video Unit sarà elaborato completamente dal blocco gNode.

5.5 MODELLO COMPUTAZIONALE

Il modello computazionale dell'algoritmo migliorativo non differisce di molto rispetto a quello utilizzato per implementare il sistema originale. Per come è stato impostato il codice di base, è stato facile aggiungere un core al control Unit e togliere entrambi i blocchi di rete Wlan e eNode e inserire il blocco gNode.

La logica di routing risulta essere semplificata, questo in quanto non vi più la possibilità di instradare i frame video verso eNode/Wlan Unit.

```
454   int routing_to_cloud() {
455       return GNODE_UNIT;
456     }
```

Fig.14 Routing to cloud viene semplificata

5.6 VERIFICA

Le verifiche effettuate riprendono quelle dell'algoritmo 1.

Per fare questo abbiamo verificato:

A. Il tempo di risposta ($\bar{w}$) per un server è sempre uguale alla somma del tempo di attesa ($\bar{d}$) e del tempo di servizio ($\bar{s}$). $\bar{w} = \bar{s} + \bar{d}$
B. Se le probabilità di routing dichiarate nelle specifiche ritornano nei rapporti tra ingressi e uscite.
C. Se per il blocco a perdita Video Unit : $Arrivals * (1 - Ploss) = Completed$.

Per valutare queste condizioni sono state eseguite diverse run ad orizzonte sia finito che infinito. Riportiamo qui i risultati di una run ad orizzonte finito nell'arco di $3600000\ ms$.

```
======== Result for block CONTROL_UNIT ========
Number of Servers ..... = 2
Arrivals ..... = 225175
Completions..... = 225172
Job in Queue at the end ..... = 1
Average interarrivals..... = 15.987513
Average wait ..... = 19.180846
Average delay ..... = 4.227654
Average service time ..... = 14.953474
Average # in the queue ..... = 0.264435
Average # in the node ..... = 1.199739

    server     utilization     avg service
    0     0.555260     14.955886
    1     0.380062     14.950117
```

```
======== Result for block VIDEO_UNIT ========
Number of Servers ..... = 2
Arrivals ..... = 89947
Completions..... = 74715
Job in Queue at the end ..... = 0
Average interarrivals..... = 48.182937
Average wait ..... = 29.161719
Average wait (2)..... = 29.161719
Number bypassed ..... = 15232
Ploss ..... = 0.169344
Average delay ..... = 0.000000
Average service time ..... = 35.106861
Average # in the queue ..... = 0.000000
Average # in the node ..... = 0.728616

    server     utilization     avg service
    2     0.468265     35.100021
    3     0.260351     35.119169
```

```
======== Result for block GNODE_UNIT ========
Number of Servers ..... = 2
Arrivals ..... = 209940
Completions..... = 209931
Lambda_i.....=0.058317
Mu_i.....=0.033354
Throughput_i.....=0.058317
Job in Queue at the end ..... = 7
Average interarrivals..... = 17.147700
Average wait ..... = 122.406363
Average delay ..... = 92.425210
Average service time ..... = 29.981321
Average # in the queue ..... = 5.389948
Average # in the node ..... = 7.138355

    server     utilization     avg service
    4     0.895747     30.178642
    5     0.852669     29.778813
```

```
======== Result for block EDGE_UNIT ========
Number of Servers ..... = 4
Arrivals ..... = 209931
Completions..... = 209929
Lambda_i.....=0.058314
Mu_i.....=0.028543
Throughput_i.....=0.058314
Job in Queue at the end ..... = 0
Average interarrivals..... = 17.148435
Average wait ..... = 38.395372
Average delay ..... = 3.360804
Average service time ..... = 35.035059
Average # in the queue ..... = 0.195983
Average # in the node ..... = 2.239001

    server     utilization     avg service
    6     0.703102     35.158912
    7     0.585116     35.115059
    8     0.445178     34.900564
    9     0.309651     34.799697
```

```
======== Result for block CLOUD_UNIT ========
Number of Servers ..... = 1
Arrivals ..... = 209929
Completions..... = 209928
Lambda_i.....=0.058314
Mu_i.....=0.099869
Throughput_i.....=0.058314
Job in Queue at the end ..... = 0
Average interarrivals..... = 17.148599
Average wait ..... = 10.013149
Average delay ..... = 0.000000
Average service time ..... = 10.013160
Average # in the queue ..... = 0.000000
Average # in the node ..... = 0.583905

    server     utilization     avg service
    10     Nan     10.013160
```

Figura 15 - Output

Verifica A : $\bar{w} = \bar{s} + \bar{d}$ (consistency check)

- ➔ **Control Unit** : $19.1808 = 14.953 + 4.227$
- ➔ **Video Unit** : $29.161719 = 0.0 + 35.106861 * (1 - 0.169344)$ (Sistema a perdita)
- ➔ **Gnode Unit** : $122.406363 = 92.425210 + 29.981321$
- ➔ **Edge Unit** : $38.395372 = 3.360804 + 35.035059$
- ➔ **Cloud Unit** : $10.013149 = 10.013160 + 0.0$

Verifica B :

Verichiamo le probabilità di routing:

- ➔ **Video Unit** : $p_{internal} = 0.4 \rightarrow \frac{num.arrivi\ \ Video\ unit}{num.completamenti\ Control\ unit} = 0.4010$

Verifica C :

Video Unit : $89947 * (1 - 0.169344) = 74715.015$ (considerando i job in fase di completamento)

5.7 VALIDAZIONE

Nella fase di validazione si confrontano i risultati della simulazione rispetto al modello analitico.

La tabella dei dati è stata riprodotta estrapolando i dati delle statistiche dalla simulazione effettuata ad orizzonte finito durante $3600000\ ms$ (durata di simulazione), con numero di repliche pari a 128.
In particolari sono stati presi i vari csv creati dalla simulazione e utilizzate le funzione $estimate.c$.

Questo ci ha permesso di calcolare il valore dell'intervallo di confidenza del 95%.

VALORE TEORICO	**VALORE SIMULATIVO** $\pm\alpha = 0,05$
$E[Ts] = 192,25$	$E[Ts] = 189.99 \pm 1,23445$

Possiamo quindi vedere come il modello simulativo rispetti le leggi del modello analitico. Anche non avendo dati reali a disposizione questo ci conferma che il nostro modello rappresenta bene un modello reale, e possiamo quindi proseguire con la fase di analisi dei risultati.

5.7.1 Simulazione ad orizzonte infinito e finito

Nella simulazione ad orizzonte finito viene effettuata una simulazione del sistema lungo $3600000\ ms$.
Per effettuare un'analisi statistica dei valori ottenuti ad orizzonte finito, il procedimento di misurazione è stato replicato 128 volte, ottenendo quindi un ensemble di dimensione pari a 128. Ogni replica viene utilizzata per misurare le stesse statistiche, e fornisce quindi un punto del nostro campione. Per ottenere la media campionaria e l'intervallo di confidenza al 95% si è utilizzato il programma estimate.c, che internamente utilizza la distribuzione Student.

```
based upon 128 data points and with 95% confidence
the expected value is in the interval 189.196572 +/- 3.819992
```

Fig. 16 Stampt output estimate.c

```
for a sample of size 128
mean ................ = 189.197
standard deviation ... =  21.755
minimum ............. = 145.100
maximum ............. = 246.568
```

Fig. 17 Stamp output uvs.c

Per la simulazione ad orizzonte infinito si è utilizzata la stessa tecnica di simulazione utilizzata per l'algoritmo 1. Si è visto che per $b = 5000$, $k = 128$ si ha un'autocorrelazione di 0.042 per j=1, quindi si è scelto tale valore come dimensione del batch.

In totale, per ogni simulazione vengono quindi processati $b * k = 640.000\ job$.

```
for 128 data points
the mean is ...    189.20
the stdev is ..     21.75

  j (lag)   r[j] (autocorrelation)
  1         0.042
  2         0.083
  3         0.007
  4         0.075
  5        -0.090
  6        -0.020
  7        -0.293
  8        -0.323
```

Fig. 18 Stampt output acs.c

5.8 ANALISI RISULTATI

I csv per effettuare le analisi sottostanti si trovano nella directory di progetto /results/alg2/.
Per la riproduzione dei grafici sono stati usati degli script in python.

5.8.1 Analisi ad orizzonte finito

L'analisi ad orizzonte finito ci ha permesso di effettuare l'analisi transiente.

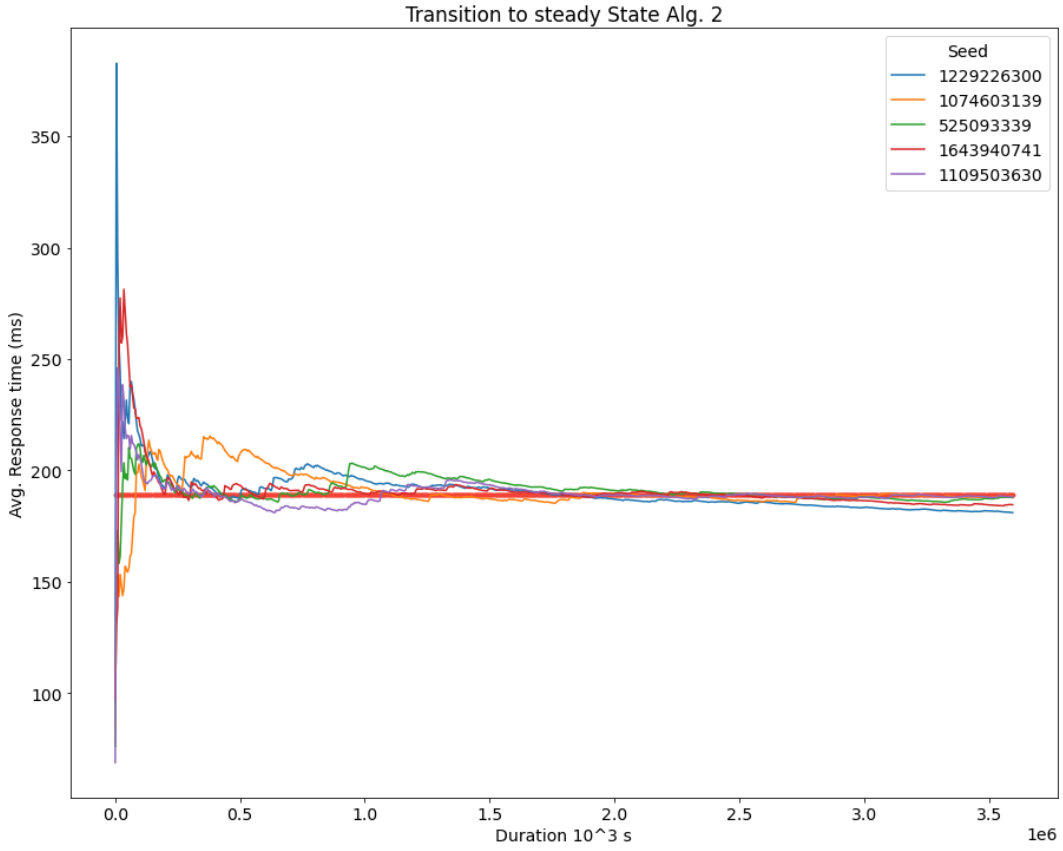

Grafico 8 Transition to steady state alg2

Nel grafico in alto, sulle ascisse è riportata la durata della simulazione, sulle ordinate il tempo medio di risposta e la retta orizzontale rossa rappresenta il valore teorico del tempo di risposta medio.
Sono rappresentate 5 “curve” relative a 5 seed iniziali differenti. Dal grafico si osserva che, come ci si aspettava, per periodi di simulazione brevi i valori ottenuti per seed diversi si discostano tra di loro e dal valore teorico (il sistema si trova nello stato transiente). A partire da periodi di circa 1600000ms ms, le “curve” tendono a convergere verso il valore teorico (il sistema passa allo stato stazionario). Dall’analisi precedente, si è deciso pertanto di utilizzare 1600000ms come durata massima delle simulazioni per l’analisi transiente.

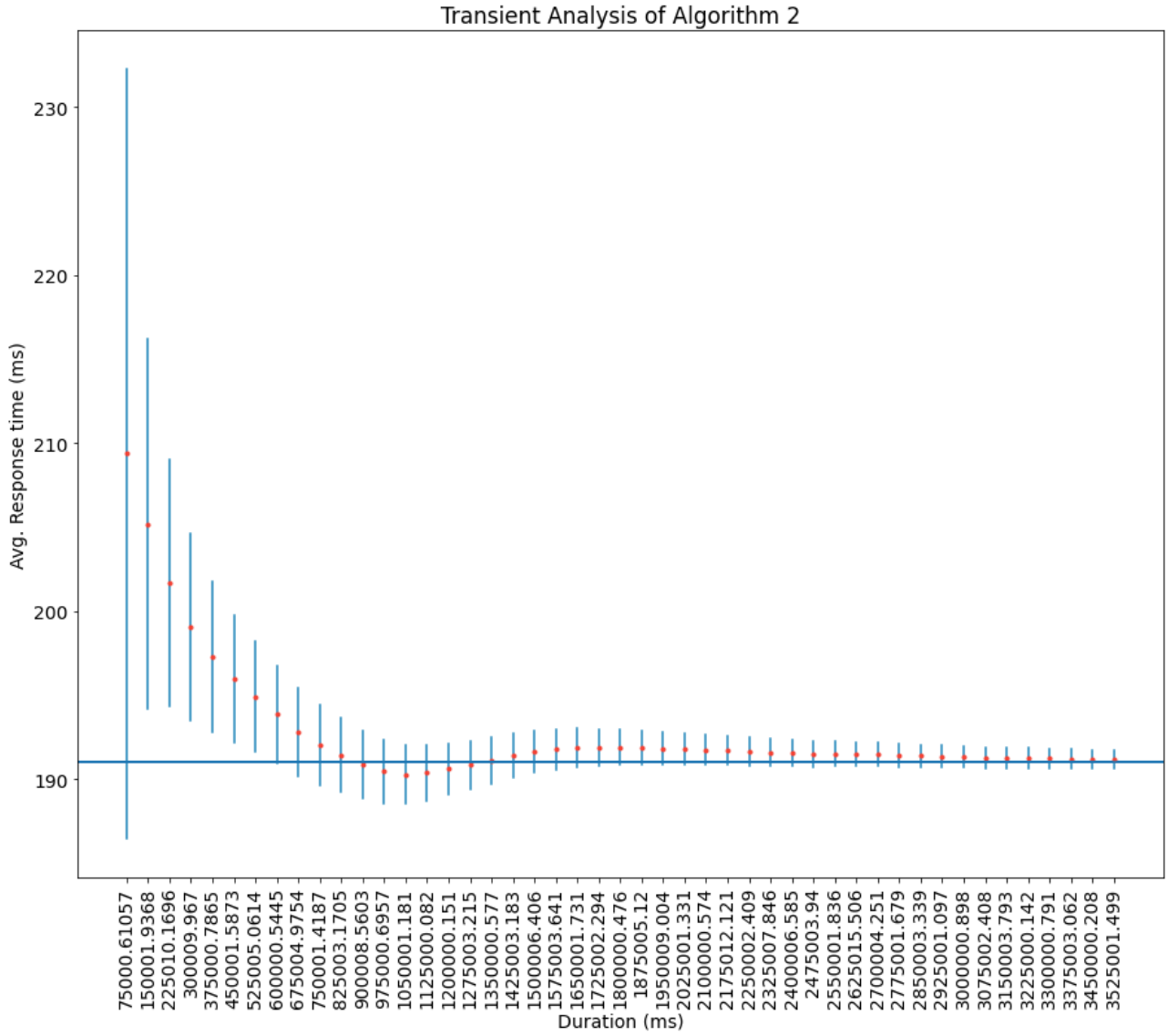

Grafico 9 Transient Analysis of algorithm 2

Nel grafico in alto, sulle ascisse viene riportata la durata delle simulazioni, sulle ordinate il tempo medio di risposta del e la linea orizzontale rappresenta il valore teorico. Fissato un valore per la durata delle simulazioni 75000ms, i dati ottenuti dalle repliche eseguite vengono utilizzati per determinare gli intervalli

di confidenza. Si osserva che a seguito dei primi ensemble di simulazioni (aventi durata di simulazione pari da 75000ms fino a 500000ms), gli altri intervalli di confidenza contengono il valore teorico.

Si nota inoltre che a mano a mano che aumenta la durata delle simulazioni, l'ampiezza dell'intervallo di confidenza diminuisce, a causa della riduzione della varianza dei dati.

5.8.2 Analisi ad orizzonte infinito

Si analizza l'output sull'analisi dello stato stazionario.
Nel grafico successivo vengono riportati i dati ottenuti utilizzando il metodo dei batch means, con $B = 5000$, $K = 128$. Sulle ascisse sono riportati i seed iniziali utilizzati, sulle ordinate il tempo medio di risposta e la linea orizzontale nera rappresenta il valore teorico. Si osserva che si ha una copertura di circa 80%. Tale grafico è stato realizzato per verificare la convergenza del sistema, indipendentemente dal seed iniziale utilizzato.

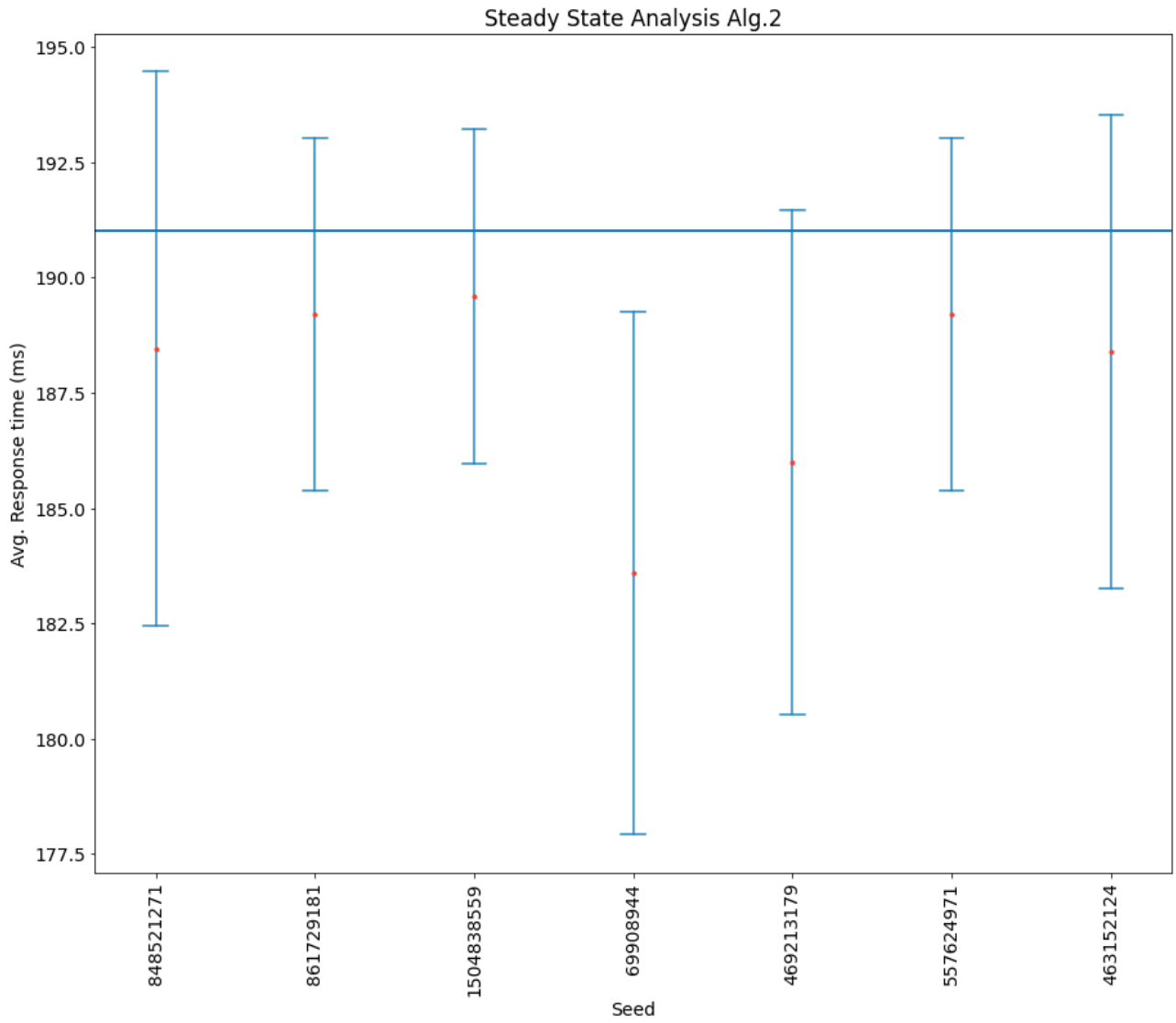

Grafico 10 Steady state analysis of alg 2

5.9 CONSIDERAZIONI FINALI

L'algoritmo 2 rappresenta una valida alternativa al soddisfacimento dei QoS richiesti dal case study.

L'algoritmo permette di avere una visione della struttura di un sistema con possibilità di aggancio alla rete tramite tecnologia 5g.

6 CONCLUSIONI

In conclusione, confrontando gli algoritmi proposti, si evince che tutti gli algoritmi rientrano nel range [100,500] ms di tempo di risposta medio per l'esecuzione di un job.

Il grafico sottostante rappresenta degli istogrammi posti in parallelo. Gli istogrammi rappresentano la distribuzione dei tempi di risposta medi totali.

I dati utilizzati per i grafici sono l'insieme dei tempi di risposta calcolati tramite simulazione infinita, usando i parametri utilizzati nelle simulazioni sopra illustrate.

- In particolare, l'algoritmo 1 risulta essere il peggiore dei tre, accostandosi molto vicino al limite imposto dai QoS.
- L'algoritmo alternativo 2 si pone esattamente nel mezzo tra l'algoritmo 1 e algoritmo 1 migliorativo per i suoi tempi di risposta.
- Mentre l'algoritmo 1 migliorativo risulta essere il migliore in tempi di risposta medi, rimanendo nel limite minimo consentito dei tempi di risposta realistici dei sistemi di video streaming.

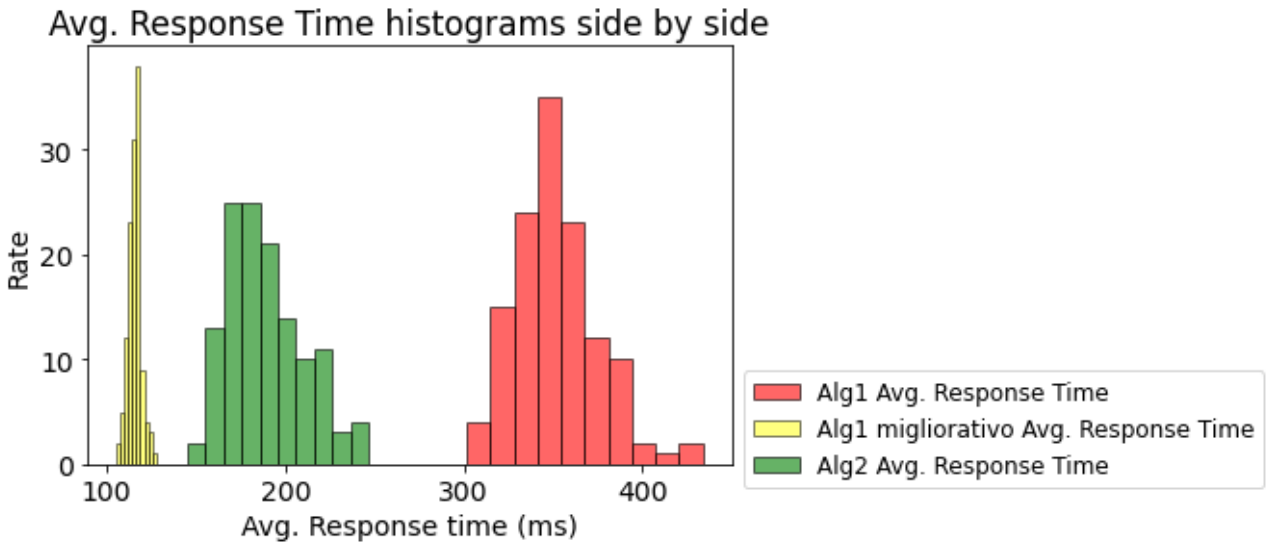

Fig. 19 Grafico di confronto dei tempi di risposta dei tre algoritmi

6.1 SVILUPPI FUTURI

Uno sviluppo futuro del progetto potrebbe essere quello di studiare il consumo energetico in relazione ai sistemi illustrati. Si può mettere in relazione il tempo di risposta medio per ogni blocco con le relative potenze espresse in Watt, e quindi calcolare il consumo energetico in kWh.

7 BIBLIOGRAFIA

- link alla directory github
- Analisi schede video: oltre gli FPS per entrare nel secondo | Pagina 1: Introduzione | Hardware Upgrade (hwupgrade.it)
- (PDF) Tradeoff Analysis for Mobile Cloud Offloading Based on an Additive Energy-Performance Metric (researchgate.net)
- (PDF) Mobile Edge Computing: Survey and Research Outlook (researchgate.net)
- https://ieeexplore.ieee.org/document/7946410