Performance Modeling of Computer Systems and Networks

Pier Francesco Contino Alex Ponzo

Febbraio 2020

Indice

Capitolo 1

Introduzione

In questa relazione verrà discussa l'implementazione di un modello di simulazione discrete-event con un approccio di tipo event-driven. Il problema trattato, che verrà qui di seguito definito, è stato dapprima affrontato analiticamente ed i risultati ottenuti sono stati successivamente confrontati con quelli del modello di simulazione tramite un'analisi statistica. Al termine della trattazione, dopo le conclusioni, sarà inclusa una rapida panoramica sull'organizzazione del codice utilizzato per produrre i risultati.

1.1 Il problema

1.1.1 Primo modello

Un Service Provider ha necessità di gestire due diversi flussi di traffico, uno HTTP ed uno multimediale; sulla base dello studio sulla distribuzione dei due tipi di traffico durante la giornata, effettuato nei mesi precedenti, i seguenti valori medi sono stati ottenuti.

Orario	Traffico HTTP	Traffico multimediale
8:00-12:00	1150	550
12:00-15:00	500	800
15:00-20:00	1000	700
20:00-24:00	300	1100
24:00-8:00	100	200

Tabella 1.1: Distribuzione media del traffico (espressa in req/s)

L'azienda ha effettuato anche degli studi di settore per conoscere il tempo che i propri futuri clienti sono disposti ad attendere prima di ricevere una risposta dai server; in base a questi si vuole garantire uno slowdown medio non superiore a 3 per il traffico di tipo HTTP e non superiore a 2,5 per il traffico multimediale. Il Service Provider è interessato a conoscere il numero minimo di server che è

necessario acquistare per soddisfare la richiesta e quanti è necessario tenerne accesi in ogni intervallo, sapendo che:

- I server sono in grado di gestire richieste HTTP con un tempo medio di 5 millisecondi e richieste multimediali con un tempo medio di 8 millisecondi.
- Ogni server in una determinata fascia può lavorare solo su uno dei due flussi (può comunque essere spostato ad ogni cambio di fascia oraria se necessario).
- Entrambi i tipi di richiesta (HTTP e multimediale) sono non interrompibili.
- Il Service Provider dispone di capacità di memorizzazione molto elevate rispetto alle dimensioni del problema, pertanto il traffico in ingresso non può essere perso anche in caso di breve sovraccarico dei server.
- I tempi di smistamento del tipo di traffico in entrata verso i server incaricati di gestirlo sono trascurabili.
- I tempi di accensione, spegnimento o spostamento da un tipo di richiesta ad un'altra dei server sono trascurabili.

1.1.2 Modello migliorato

In un secondo momento, successivo alla messa in opera del modello sopra descritto, il Service Provider vuole aggiornare il sistema in modo da permettere l'abbonamento premium di una parte dei propri clienti in parallelo con la versione standard precedente.
Il Service Provider vuole conoscere la percentuale massima di richieste che è possibile spostare in abbonamenti di tipo premium in ogni fascia oraria mantenendo intatti i seguenti vincoli:

- Le richieste HTTP degli utenti premium avranno la priorità su quelle HTTP degli utenti standard.
- Le richieste multimediali degli utenti premium avranno la priorità su quelle multimediali degli utenti standard.
- Si vuole comunque garantire che tutti gli utenti (indipendentemente che siano abbonati o meno) ottengano un servizio con uno slowdown medio che rispetti i vincoli del sistema precedente (non superiore a 3 per il traffico di tipo HTTP e non superiore a 2,5 per il traffico multimediale).
- Il numero di server totali e la loro distribuzione nelle diverse fasce orarie, calcolati nella fase precedente, non vengono modificati.
- Gli abbonamenti premium sono validi per fascia oraria e non per l'intera giornata, si potranno quindi avere percentuali di utenti premium diverse in diversi orari.

Capitolo 2

Modello analitico - Maxima

Prima di procedere alla costruzione del modello di simulazione il problema è stato affrontato con un approccio analitico; questo ha permesso di ottenere dei valori teorici delle variabili di interesse, in particolare per quanto riguarda la fase stazionaria, che poi sono stati confrontati tramite un'analisi statistica con gli output del sistema di simulazione. In questa fase si è reso utile l'utilizzo di Maxima, un sistema di algebra computazionale che ha permesso di automatizzare un gran numero di calcoli.

2.1 Primo modello

La formula principale di cui tener conto per soddisfare la richiesta del problema è quella per il calcolo dello slowdown medio:

$$E[sd] = \frac{E[T_s]}{E[size]}.$$

In questa formula $E[size]$ rappresenta la grandezza media delle richieste, dimensionata rispetto alla velocità di calcolo dei server; tale valore è pertanto facile da ricavare come il reciproco del tempo di servizio medio del server per ogni tipo di richiesta. Il valore al numeratore rappresenta invece il tempo medio di sistema, calcolato come la somma del tempo medio di servizio più il tempo medio di coda; nel caso del nostro sistema a coda singola e servente multiplo (supponendo che sia il tasso di arrivo che quello di servizio siano markoviani):

$$E[T_s] = E[T_q] + E[S_i] = \frac{P_q \cdot \frac{E[S_i]}{m}}{1-\rho} + E[S_i].$$

Nella formula si indica con m il numero di server attivi e con ρ l'utilizzazione media dei server. Rimane dunque da calcolare il valore P_q che rappresenta la probabilità di trovare la coda occupata, questa può essere ricavata tramite:

$$P_q = \frac{(m \cdot \rho)^m}{m! \cdot (1 - \rho)} \cdot p(0).$$

Il valore utilizzato $p(0)$ rappresenta la probabilità di avere tutti i server liberi ed equivale a:

$$p(0) = \left[\sum_{i=0}^{m-1} \frac{(m \cdot \rho)^i}{i!} + \frac{(m \cdot \rho)^m}{m! \cdot (1 - \rho)} \right]^{-1}.$$

Applicando iterativamente le formule citate per valori di m crescenti (Maxima è stato molto utile nell'automatizzare questa funzione) è stato possibile ricavare il numero minimo di server per ogni fascia oraria necessari per soddisfare il vincolo sullo slowdown medio per ognuno dei due flussi di richieste. Il valore cercato quindi (il valore minimo di server totali da acquistare) è semplicemente esprimibile come il massimo sugli intervalli della somma dei server minimi per il traffico HTTP e di quelli minimi per il traffico multimediale.
Vengono di seguito riportati i valori ottenuti:

Orario	Server HTTP	Server multimediali
8:00-12:00	7	5
12:00-15:00	3	7
15:00-20:00	6	7
20:00-24:00	2	10
24:00-8:00	1	3

Tabella 2.1: Valori minimi di server necessari

Sulla base della tabella precedente è facile notare come la fascia oraria critica sia quella tra le 15:00 e le 20:00, in cui la somma dei carichi è maggiore (nonostante nessuno dei due flussi di richieste abbia il proprio picco in quell'ora); il numero minimo di server che il Service Provider deve acquistare è dunque 13 (6 per il traffico HTTP e 7 per quello multimediale nell'unica fascia oraria in cui verranno utilizzati tutti, nelle altre fasce potranno essere spostati tra i due flussi di richieste ed all'occorrenza spenti).

2.2 Modello migliorato

In questa seconda fase il modello precedente viene modificato; come prima i due flussi in entrata vengono separati, ma poi ognuno dei due viene gestito tramite un sistema a priorità con due code (quindi due classi) e serventi multipli. La prima conseguenza di questa modifica risulta nel calcolo di $E[T_{s_k}]$ ovvero il tempo medio di servizio della k-esima classe:

$$E[T_{s_k}] = E[T_{q_k}] + E[S_i] = \frac{P_q \cdot \frac{E[S_i]}{m}}{(1 - \sum_{j=1}^{k} \rho_j) \cdot (1 - \sum_{j=1}^{k-1} \rho_j)} + E[S_i].$$

Nel caso di studio il tempo di servizio di interesse è quello della seconda classe, quella standard. Questo perché il vincolo espresso sullo slowdown medio dovrà essere valido per entrambe le classi, ma la prima avrà un $E[T_s]$ inferiore; pertanto se il vincolo sullo slowdown medio sarà rispettato dalla classe con meno priorità lo sarà anche da quella a priorità maggiore. Sulla base di questo ragionamento si ricava la percentuale massima ammissibile per la classe premium a partire da:

$$E[sd] = \frac{P_q}{m \cdot (1-\rho) \cdot (1-\rho_1)} + 1;$$

tramite pochi passi è possibile dunque ricavare il valore cercato p_1 tenendo presente che $\rho \cdot p_1 = \rho_1$:

$$p_1 = \frac{1}{\rho} \cdot \left[1 - \frac{P_q}{m \cdot (1-\rho) \cdot (E[sd]-1)}\right].$$

Grazie alle formule sopra indicate è stato possibile ricavare i valori limite di p_1 che massimizzano il carico premium mantenendo inviolato il vincolo sullo slowdown della classe standard.
Per le richieste HTTP si ha:

Orario	p_1	$E[sd]$
8:00-12:00	0,958307984	3,0
12:00-15:00	0,35730337	3,0
15:00-20:00	0,847490129	3,0
20:00-24:00	0,476190476	3,0
24:00-8:00	0,999	2.99800199

Tabella 2.2: Percentuale massima di carico premium per le richieste HTTP

Per le richieste multimediali si ottengono invece i seguenti valori:

Orario	p_1	$E[sd]$
8:00-12:00	0,229307606	2,5
12:00-15:00	0,173507331	2,5
15:00-20:00	0,960751422	2,5
20:00-24:00	0,750626056	2,5
24:00-8:00	0,999	1,41859832

Tabella 2.3: Percentuale massima di carico premium per le richieste multimediali

Come è evidenziato dai valori dello slowdown che coincidono i vincoli, i valori di p_1 rappresentano il limite massimo, aumentarli significherebbe comportare un ritardo per gli utenti standard superiore a quello voluto.
Le uniche due eccezioni a questo criterio sono i due flussi di richieste nella fascia oraria "meno impegnativa" dal punto di vista del carico, ovvero 24:00-8:00; in

questo intervallo di tempo il valore dello slowdown è inferiore al vincolo, ad indicare che in realtà il valore di p_1 potrebbe essere aumentato ancora. Tale comportamento è dovuto al fatto che i server supporterebbero un carico premium superiore a quello totale attualmente esercitato senza infrangere i vincoli sulla classe standard (in questo caso la formula andrebbe modificata); benché questo sia ammissibile (l'azienda può scegliere di utilizzare i server anche per altre operazioni in questo intervallo), sia in linea teorica che per la correttezza della successiva valutazione si è scelto di impostare un limite massimo di p_1 pari a 0,999, in modo da lasciare invariato il carico di richieste ricevuto dai server.
Ora, tenendo conto delle semplificazioni secondo cui gli utenti che esercitano i due tipi di richieste sono gli stessi e che la proporzione tra di esse di ogni utente sia anch'essa costante, si può procedere all'identificazione dei valori massimi di utenti premium supportati dal sistema considerando il minimo tra i due valori di p_1 appena ricavati:

Orario	p_1
8:00-12:00	0,229307606
12:00-15:00	0,173507331
15:00-20:00	0,847490129
20:00-24:00	0,476190476
24:00-8:00	0,999

Tabella 2.4: Percentuale massima di utenti premium

Capitolo 3

Implementazione del modello di simulazione

L'implementazione del modello di simulazione è stata effettuata seguendo i passi del seguente algoritmo:

1. Identificazione dello scopo e degli obiettivi dell'analisi.
2. Costruzione di un modello concettuale ed identificazione delle variabili di stato del sistema.
3. Conversione del modello concettuale in un modello delle specifiche, con conseguente identificazione dei modelli stocastici di interesse.
4. Realizzazione di un modello computazionale basato sul modello delle specifiche.
5. Verifica del sistema, per valutare la consistenza del modello computazionale rispetto a quello delle specifiche.
6. Validazione del sistema, per valutare la consistenza del modello computazionale rispetto al sistema che si intende realizzare.

Tramite questi passi sono stati costruiti i modelli di entrambe le fasi del problema iniziale, in questo capitolo ognuno di essi verrà quindi descritto.

3.1 Primo modello

3.1.1 Scopo ed obiettivi

L'obiettivo dello studio è quello di minimizzare il numero di server totali acquistati dal Service Provider; per far ciò è necessaria un'analisi che determini di quanti server c'è bisogno in ogni fascia oraria per gestire i due flussi di richieste.

Il numero minimo di server che è necessario avere accesi in ogni intervallo di tempo dovrà essere sufficiente anche a mantenere inviolati i vincoli sullo slowdown delle richieste.

3.1.2 Modello Concettuale

Per lo sviluppo di un modello concettuale è importante identificare le variabili di stato che possano rappresentare in maniera fedele una descrizione del sistema in ogni istante. Per far questo è sufficiente tenere traccia in ogni istante di:

- Numero di server attualmente occupati da richieste di tipo HTTP.
- Numero di server attualmente occupati da richieste di tipo multimediale.
- Numero di richieste HTTP in coda in attesa di essere servite.
- Numero di richieste multimediali in coda in attesa di essere servite.

Questo viene naturalmente fatto mantenendo fissato il numero di server attivi all'interno di ogni fascia oraria.
Il diagramma seguente permette di avere una visione d'insieme del sistema.

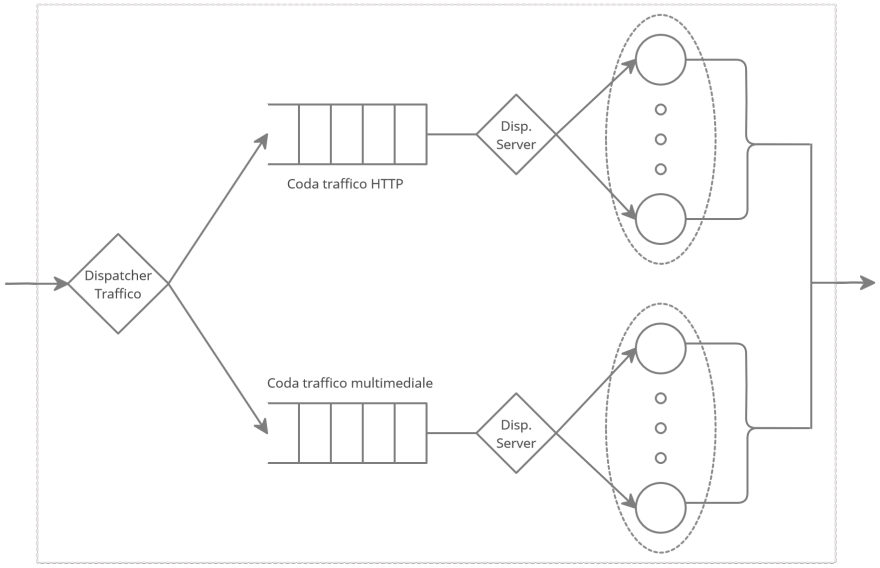


Figura 3.1: Diagramma del sistema

Come da descrizione del problema si possono notare due diversi flussi di traffico, il componente "Dispatcher di traffico" sta ad indicare questa suddivisione iniziale del traffico che raggiunge il Service Provider; ogni flusso viene dunque

immesso in un sistema a coda singola e multiserver (per quasi la totalità tempo, ma in fasce orarie particolari sarà anche possibile gestire le richieste con un solo server, in tal caso l'architettura di gestione di ogni flusso sarà naturalmente a coda singola e servente singolo). I tre dispatcher utilizzati (quello per il traffico iniziale ed i due per le assegnazioni delle richieste ai vari server) sono da considerarsi come componenti ideali e siccome i loro tempi di servizio sono trascurabili (come da specifiche iniziali) non si rende necessario tenere in considerazione eventuali richieste all'interno di tali dispositivi come variabili di stato. Si fa notare anche la scelta di adottare code infinite nell'architettura del sistema, questa è naturalmente un'astrazione della realtà, ma è conforme con la descrizione del problema, secondo cui il Service Provider "dispone di capacità di memorizzazione molto elevate rispetto alle dimensioni del problema". Le richieste in uscita dai server lasciano infine il sistema e raggiungono nuovamente i clienti una volta esaudite.

3.1.3 Modello delle specifiche

Siccome non si hanno dati storici da analizzare (questo sistema deve essere ancora realizzato) si andrà a scegliere il modello stocastico ipotizzato come più adatto a questo scenario. Innanzitutto procedendo ad un'analisi degli arrivi e delle esecuzioni delle richieste, queste verranno supposte come markoviane; di conseguenza il sistema più indicato allo scopo è dato dal modello M/M/m.

3.1.4 Modello computazionale

In questa sezione verranno specificati i principi chiave che hanno portato alla scrittura del programma di simulazione. Per prima cosa c'è stata la scelta di un generatore di numeri pseudo-casuali, per questo si è scelto un generatore di Lehmer:

$$x_{i+1} = g(x_i) \tag{3.1}$$

$$g(x) = ax \mod m \tag{3.2}$$

Come si vede dalle equazioni che compongono un generatore di Lehmer particolare importanza è attribuita ai parametri a ed m; questi parametri influiscono sulla lunghezza del periodo del generatore, cioè dopo quanto quest'ultimo inizia a ripetere i numeri generati. La lunghezza massima del periodo è m - 1 e in quel caso la sequenza si definisce *full-period*. Ma questa non è l'unica caratteristica importante, bisogna assicurarsi di scegliere dei parametri che non causino l'overflow durante l'operazione di moltiplicazione (definiti moltiplicatori modulo-compatibili). Infine bisogna considerare quanto le sequenze generate siano vicine alla casualità, facendo attenzione a non utilizzare lo stesso flusso per generare eventi che dovrebbero essere indipendenti; per garantire questo è stato scelto un generatore multi-flusso. In questo primo scenario gli eventi possibili sono arrivi e completamenti, verranno quindi utilizzati due flussi distinti.

a	m
48271	2147483647

Tabella 3.1: I valori utilizzati nella simulazione

Un generatore di Lehmer può produrre dei valori decimali con precisione $1/m$ che appartengono all'intervallo [$1/m$, 1 - $1/m$]; per questa simulazione ci sarà bisogno di generare una distribuzione esponenziale, pertanto verrà utilizzata una funzione che possa adattare l'intervallo di partenza.

$$U = uniforme(0, 1) \tag{3.3}$$

$$Z = F^{-1}(U) \tag{3.4}$$

$$F(x) = 1 - e^{-\lambda x} \tag{3.5}$$

$$Z = F^{-1}(x) = -\frac{1}{\lambda}\ln(1 - x) \tag{3.6}$$

Una volta definiti questi strumenti base si può parlare della funzione di simulazione, in questa si hanno come eventi gli arrivi ed i completamenti, e di volta in volta bisogna trovare l'evento più prossimo tra essi; per fare questo c'è bisogno di una struttura dati dove vengono salvati questi tempi, si è scelto di utilizzare una lista ordinata per ottimizzare la ricerca dell'evento più imminente. Quando si presenta un evento i contatori delle metriche di interesse vengono aggiornati: il numero di job in servizio, il numero di job in coda, il numero di job completati, insieme a delle metriche ricavate da essi tramite le seguenti formule:

$$node_{area}(i) = node_{area}(i-1) + (n_{queue} + n_{service})(t(i) - t(i-1)) \tag{3.7}$$

$$queue_{area}(i) = queue_{area}(i-1) + (n_{queue})(t(i) - t(i-1)) \tag{3.8}$$

$$service_{area}(i) = service_{area}(i-1) + (n_{service})(t(i) - t(i-1)) \tag{3.9}$$

La condizione iniziale dei contatori nel caso di simulazioni stazionarie è 0 in caso di simulazioni transitorie verrà settato a valori congrui (questa parte verrà spiegata in dettaglio nella sezione apposita). È facile notare che se la condizione iniziale è zero a fine simulazione si avranno le formule:

$$node_{area} = \sum_{i=1}^{n}(n_{queue} + n_{service}) \cdot \Delta t_i \tag{3.10}$$

$$queue_{area} = \sum_{i=1}^{n} n_{queue} \cdot \Delta t_i \tag{3.11}$$

$$service_{area} = \sum_{i=1}^{n} n_{service} \cdot \Delta t_i \tag{3.12}$$

Dove Δt_i rappresenta il tempo tra i due eventi i e i - 1. I risultati della simulazione sono i valori medi del tempo di risposta, del tempo di coda, dell'utilizzazione dei server, del tempo di servizio; essi vengono ricavati dai vari contatori a fine simulazione tramite le formule:

$$E[ts] = \frac{node_{area}}{n_{jobs}} \tag{3.13}$$

$$E[tq] = \frac{queue_{area}}{n_{jobs}} \tag{3.14}$$

$$E[s] = \frac{service_{area}}{n_{jobs}} \tag{3.15}$$

$$\rho = \frac{service_{area}}{T \cdot n_{jobs}} \tag{3.16}$$

Dove T rappresenta il tempo totale della simulazione e n_{jobs} è il numero totale dei job eseguiti durante la simulazione.
Per avere dei risultati attendibili dalle simulazioni c'è bisogno di replicare la simulazione con semi diversi per i generatori random ed aggregare i dati così fornendo degli ensemble. Avendo un numero sufficiente di dati si possono elaborare delle statistiche; per l'analisi stazionaria vengono calcolato gli intervalli di confidenza basandosi sulla media e la varianza tramite l'algoritmo qui esposto, che permette di scandire i risultati una sola volta rispetto all'utilizzo di un approccio naif.

Algorithm 3.1: Algoritmo di Welford

```
1  begin
2      int n = 0
3      float mean = 0.0
4      float v = 0.0
5          foreach x in data
6              n++
7              d ← x − mean
8              v = v + d * d * (n−1)/n
9              mean = mean + d / n
10         end
11     stdev = sqrt(v / n − 1)
12     return mean, stdev
13 end
```

3.1.5 Verifica e Validazione

Per quanto riguarda la verifica sono stati effettuati dei test tramite il controllo di alcune condizioni di consistenza; innanzitutto si vuole che la coda sia vuota se i server sono vuoti, in secondo luogo che il numero di job serviti non sia mai maggiore del numero di server. Per verificare queste funzioni sono stati utilizzati degli assert, costrutti del linguaggio c che possono poi essere disabilitati agilmente una volta finita la fase di test.

Per quanto riguarda la validazione innanzitutto bisogna verificare che i risultati della simulazione combacino con i risultati analitici prodotti. In aggiunta è

utile fare un confronto tra scenari simili in cui c'è un cambio di una sola variabile tra le simulazioni, per osservare che il sistema abbia l'evoluzione attesa.

λ	μ	m	$E[t_s]$	$E[t_q]$	$E[t_s]$ - $E[t_q]$
800	7	125	0.018065141	0.010065429	0.007999712
700	7	125	0.010770313	0.002770767	0.007999546

Tabella 3.2: Dati delle simulazioni

Si può notare dalla tabella che le simulazioni hanno un comportamento congruo, all'aumentare del traffico si ha un aumento dei valori di $E[t_s]$ e $E[t_q]$, inoltre si può vedere che la differenza tra di essi rimane costante (tralasciando un piccolo errore) e combacia con il valore $E[s]$ ricavato da $1/\mu$.
Per verificare il modello transitorio non è possibile utilizzare l'analisi, ma si può effettuare un confronto tra i valori transitori a lungo termine e quelli stazionari.

Metrica	Stazionario	Transitorio
$E[t_s]$	0.018065141	0.018101026
$E[t_q]$	0.010065429	0.010100661
ρ	0.914144219	0.914094984

Tabella 3.3: Confronto transitorio stazionario multi12:15 tempo transitorio 3 ore

Si può vedere che quando il tempo aumenta anche la simulazione transitoria, che inizia con la coda non vuota, converge verso i valori stazionari.

3.2 Modello migliorato

3.2.1 Scopo ed obiettivi

L'obiettivo dello studio in questo secondo modello del sistema è quello di ricavare il numero massimo di richieste premium che è possibile supportare in ogni fascia oraria; nel far ciò è necessario tenere conto che questo tipo di richieste ha la priorità su quelle standard, comportando quindi un ritardo nella gestione media di queste ultime. Mantenendo fissato il numero di server in ogni fascia calcolato nella fase precedente si deve quindi valutare quale percentuale consente di non infrangere il vincolo sullo slowdown che deve rimanere valido. Siccome da specifiche del problema gli utenti premium possono fare sia richieste HTTP che multimediali, sarà necessario calcolare alla fine il minimo tra i due valori soglia percentuali dei due flussi calcolati precedentemente in ogni fascia oraria.

3.2.2 Modello Concettuale

Come per il modello precedente si procede all'identificazione delle variabili di stato necessarie a rappresentare il sistema in ogni istante; in questa evoluzione i valori di cui tenere traccia sono:

- Numero di server attualmente occupati da richieste di tipo HTTP premium.
- Numero di server attualmente occupati da richieste di tipo HTTP standard.
- Numero di richieste nella coda HTTP premium.
- Numero di richieste nella coda HTTP standard.
- Numero di server attualmente occupati da richieste di tipo multimediale premium.
- Numero di server attualmente occupati da richieste di tipo multimediale standard.
- Numero di richieste nella coda multimediale premium.
- Numero di richieste nella coda multimediale standard.

Va fatto presente che idealmente sarebbe possibile ridurre il numero di variabili di sistema accorpando i valori dei server occupati da richieste premium e standard, riducendone quindi il numero di due (una variabile in meno per il traffico HTTP ed una in meno per quello multimediale). Questo è possibile poiché le richieste sono non interrompibili, pertanto il sistema risultante sarà senza preemption, ciò in aggiunta al fatto che le distribuzioni dei tempi di esecuzione del traffico premium e standard sono identiche fa sì che sia indifferente per il traffico in ingresso conoscere quale tipo di richiesta sta occupando un server.

Nonostante tale riduzione sia possibile comunque, per mantenere una semplicità di rappresentazione tali variabili saranno presenti. Questa scelta consentirà anche di mantenere un certo livello di fedeltà rispetto alle scelte fatte nella fase di implementazione del modello computazionale, in cui tali valori saranno importanti per la raccolta di alcune delle statistiche.
Il diagramma seguente rappresenta le caratteristiche del sistema modificato.

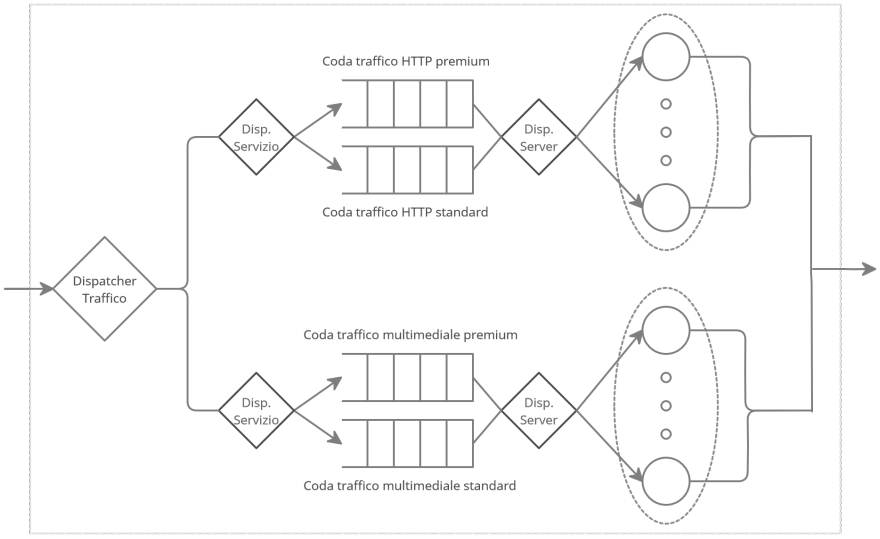


Figura 3.2: Diagramma del sistema migliorato

Come nel modello precedente il traffico in ingresso nel sistema viene suddiviso in base al tipo dal primo dispatcher; a questo punto però avviene un secondo smistamento, in cui in ognuno dei due flussi viene separato il traffico premium da quello standard, memorizzandolo in code separate tramite un secondo livello di dispatching. Un terzo ed ultimo dispatcher in cascata permette quindi di distribuire le richieste ai server disponibili, avendo cura di assegnare prima quelle nella coda del traffico premium e passando ad assegnare quelle nella coda del traffico standard esclusivamente se la coda precedente risulta vuota. Con questa architettura dunque ognuno dei due flussi di traffico sarà gestito tramite un sistema a coda multipla e servente multiplo (tranne il caso particolare in cui basta un server solo a soddisfare le richieste nella fascia oraria). Le altre caratteristiche del modello migliorato restano analoghe a quelle precedenti, con tutte le code illimitate ed i dispatcher con tempi di servizio trascurabili.

3.2.3 Modello delle specifiche

Come nella parte precedente anche questo sistema è assimilabile ad un M/M/m ma qui in più si hanno delle classi di priorità indipendenti dalle dimensioni dei

job, inoltre non è possibile la preemption data la non interrompibilità dei job.

3.2.4 Modello computazionale

Molte cose dette nel punto 3.1.4 riguardano anche questo punto e qui non saranno ripetute ma ci si concentrerà solo sulle differenze.
L'aggiunta delle classi di priorità e il voler sapere le metriche relative ad esse richiede l'aggiunta di contatori e metriche replicati per ognuna delle due classi, si avranno quindi:

- $node1_{area}$, $service1_{area1}$, $queue1_{area}$
- $node2_{area}$, $service2_{area2}$, $queue2_{area}$
- $n1_{queue}$, $n1_{service}$, $n1_{njobs}$
- $n2_{queue}$, $n2_{service}$, $n2_{njobs}$

che seguono le medesime formule di quelli precedenti. Ora però bisogna discriminare i job terminati in base alla classe, questo richiede che la lista dei job in esecuzione abbia un campo dedicato; un'altra differenza è che ora c'è un flusso casuale aggiuntivo per determinare la classe di appartenenza delle richieste (standard o premium) ed una funzione per ricavare una distribuzione di Bernoulli.

$$Bernoulli(x) = \begin{cases} 1 & se \quad x \geq p \\ 0 & se \quad x < p \end{cases} \tag{3.17}$$

Infine bisogna dare la priorità ai job della classe uno, questo viene fatto verificando che $n1_{queue}$ venga decrementato per primo se è maggiore di zero; si può vedere come questo approccio non vada a modificare in alcun modo le prestazioni globali così come ci si aspetterebbe dall'analisi teorica.

Algorithm 3.2: Algoritmo priorità

```
1  begin
2      if (n1Queue > 0 )
3          n1Queue--;
4          n1Service++;
5          lista.aggiungiJob(tempo = t + servizio(mu), priorita' = 1)
6      end
7      else
8          n2Queue--;
9          n2Service++;
10         lista.aggiungiJob(tempo = t + servizio(mu), priorita' = 2)
11     end
12 end
```

3.2.5 Verifica e validazione

Per quanto riguarda la verifica data la divisione in due classi si possono aggiungere ulteriori controlli di consistenza dove viene verificato che la somma di $n1_{queue}$ e $n2_{queue}$ sia uguale a n_{queue} e lo stesso per le metriche di servizio ($n_{service}$).
La validazione del modello migliorativo passa dalla conferma che il tempo di risposta medio totale del sistema a priorità astratta è lo stesso di un sistema senza classi di priorità.

Inoltre si può vedere che $E[t_{s_1}] < E[t_s] < E[t_{s_2}]$ così come ci si aspetterebbe dall'analisi.

Orario	$E[t_s]$	$E[t_{s_1}]$	$E[t_{s_2}]$
8:00-12:00	0.007123475	0.006782185	0.014960757
12:00-15:00	0.012004826	0.006664554	0.014972069
15:00-20:00	0.007932661	0.006664686	0.014977686
20:00-24:00	0.011417237	0.007497781	0.014979459
24:00-8:00	0.009994693	0.009989710	0.014794386

Tabella 3.4: Confronto $E[t_s]$ HTTP

Orario	$E[t_s]$	$E[t_{s_1}]$	$E[t_{s_2}]$
8:00-12:00	0.017547786	0.009438330	0.019959890
12:00-15:00	0.018065141	0.009025543	0.019962035
15:00-20:00	0.010770313	0.010394869	0.019949419
20:00-24:00	0.012064908	0.009438043	0.019969909
24:00-8:00	0.009562307	0.009560559	0.011236990

Tabella 3.5: Confronto $E[t_s]$ multimediale

Capitolo 4

Simulazione ed analisi dei risultati

Conseguentemente alla realizzazione del modello computazionale ottenuto seguendo l'algoritmo precedente l'impiego di quest'ultimo è stato guidato dai seguenti passi aggiuntivi:

7. Design degli esperimenti di simulazione.
8. Esecuzione degli esperimenti.
9. Analisi dei risultati delle simulazioni.

Escludendo, per ovvi motivi, la fase di esecuzione, gli altri due passi verranno qui di seguito descritti.

4.1 Design degli esperimenti di simulazione

Particolare risalto nella simulazione sarà dato soprattutto all'analisi degli stati stazionari, in modo da garantire il corretto funzionamento del sistema a regime in ogni orario; per ogni fascia oraria la simulazione si protrarrà quindi per un tempo maggiore dell'intervallo stesso, questo permetterà anche di perdere quasi completamente la dipendenza dai parametri iniziali della simulazione stessa. Grazie a questa proprietà è stato possibile scegliere a piacere il valore iniziale delle variabili di stato (sono state tutte poste a zero per semplicità). Allo stesso modo, grazie alla dimensione notevole dell'intervallo di simulazione è possibile trascurare anche l'effetto dei pochi valori al termine della stessa, pertanto anche le modalità di terminazione della simulazione possono essere scelte a piacere (si è scelto di interrompere semplicemente la raccolta delle statistiche al raggiungimento del tempo limite, senza attendere lo svuotamento dei server e delle code). Ci si occuperà poi anche dell'analisi delle fasi transienti; anche di queste ve ne sarà una per ogni cambio di fascia oraria, momento in cui viene modificato il

numero di server attivi e la frequenza delle richieste in arrivo al sistema. In questa parte dello studio del comportamento del sistema è stato necessario suddividere la simulazione in piccolissimi intervalli per comprendere l'andamento temporale delle variabili fondamentali (in particolare l'attenzione è stata posta sul tempo medio di sistema ed il suo avvicinamento al valore stazionario). Ad ogni fine di micro-simulazione i valori medi voluti sono stati estratti e lo stato finale del sistema è stato impostato come stato iniziale della micro-simulazione successiva; stesso metodo è stato adottato con il cambio di fascia oraria, in cui lo stato iniziale di ogni fascia è stato fatto coincidere con lo stato finale di quella precedente (in questo caso, a differenza dello studio del valore stazionario, la grandezza dell'intervallo di simulazione di ogni fascia oraria è dello stesso ordine di grandezza della durata di quest'ultima). Questo tipo di analisi è comunque meno rilevante rispetto a quella degli stati stazionari allo scopo di risolvere il problema iniziale, vengono dunque trascurati piccoli errori di approssimazione dei valori (qui le condizioni iniziali e finali contano) poiché l'enfasi non è sull'ottenimento di valori esatti ma sullo studio della velocità iniziale di convergenza verso i valori stazionari.

4.2 Analisi dei risultati

Per entrambe le fasi della simulazione e per ogni fascia oraria sono state lanciate più esecuzioni (20 per la precisione) con diversi semi assegnati ai generatori random, ciò ha permesso poi di ricavare delle statistiche aggregate di maggiore interesse per il caso di studio; oltre alle medie delle variabili cercate, utili per entrambe le fasi, sono stati ricavati degli intervalli di confidenza di queste ultime, utilizzati per mettere a confronto i valori stazionari ricavati dalla simulazione con quelli analitici.

4.2.1 Analisi dello stato stazionario

Grazie alla grande mole di dati raccolti è possibile osservare il comportamento del sistema simulato e controllare la similarità dei dati ottenuti in output con quelli ottenuti nella fase analitica, concentrando l'attenzione sul valore del tempo medio di sistema (che come abbiamo visto è, oltre al tempo medio di servizio che è stato fissato, l'unica variabile necessaria per il calcolo dello slowdown medio, che rappresenta il vincolo cercato). Per far questo però risulta particolarmente importante utilizzare dei modelli statistici per poter valutare il comportamento generale aggregando le varie simulazioni; viene dunque ricavato un intervallo di confidenza di livello $1 - \alpha$ per $E[t_s]$, pari a:

$$\left(\bar{x} - \frac{s \cdot t_{n-1,\alpha/2}}{\sqrt{n}}, \bar{x} + \frac{s \cdot t_{n-1,\alpha/2}}{\sqrt{n}}\right) \tag{4.1}$$

in cui i valori $\bar{x}$ ed s sono rispettivamente la media e la varianza campionarie (ricavate tramite l'algoritmo di Welford), n è il numero di simulazioni con

diversi semi effettuate (quindi 20) e $t_{n-1,\alpha/2}$ rappresenta il valore del quantile di ordine $1 - \alpha/2$ di una funzione di Student con 19 gradi di libertà.
In seguito alla scelta di un valore di α pari a 0,02 (quindi di un livello di confidenza pari al 98%) il valore analitico della media del tempo di sistema calcolato a monte della simulazione è stato confrontato con l'intervallo risultante.

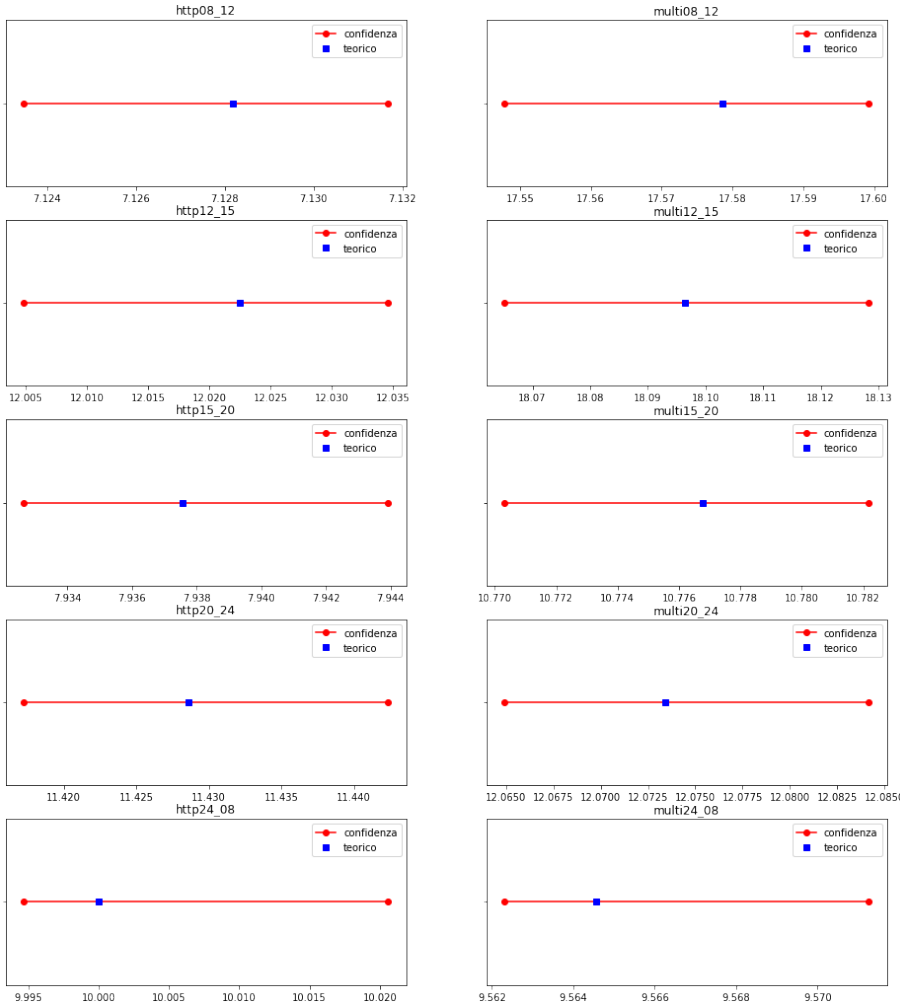


Figura 4.1: Intervalli di confidenza al 98% e medie teoriche di $E[T_s]$

Come è chiaramente visibile dalla figura, in cui le grandezze sono espresse in millisecondi, l'intervallo di confidenza contiene la media teorica per tutte le fasce orarie e per entrambi i tipi di traffico; questo conferma che il sistema simulato si comporta come atteso con il numero di server fissato nella sezione analitica.
Spostando l'attenzione sul modello del sistema modificato con due code per

gestire il traffico premium, si può seguire un ragionamento analogo; naturalmente come già specificato il valore di $E[T_s]$ è in linea con il precedente, ma il vincolo sullo slowdown dovrà essere rispettato anche dalla seconda classe (quella standard con priorità inferiore), pertanto i valori pertinenti riguardano esclusivamente $E[T_{s_2}]$.

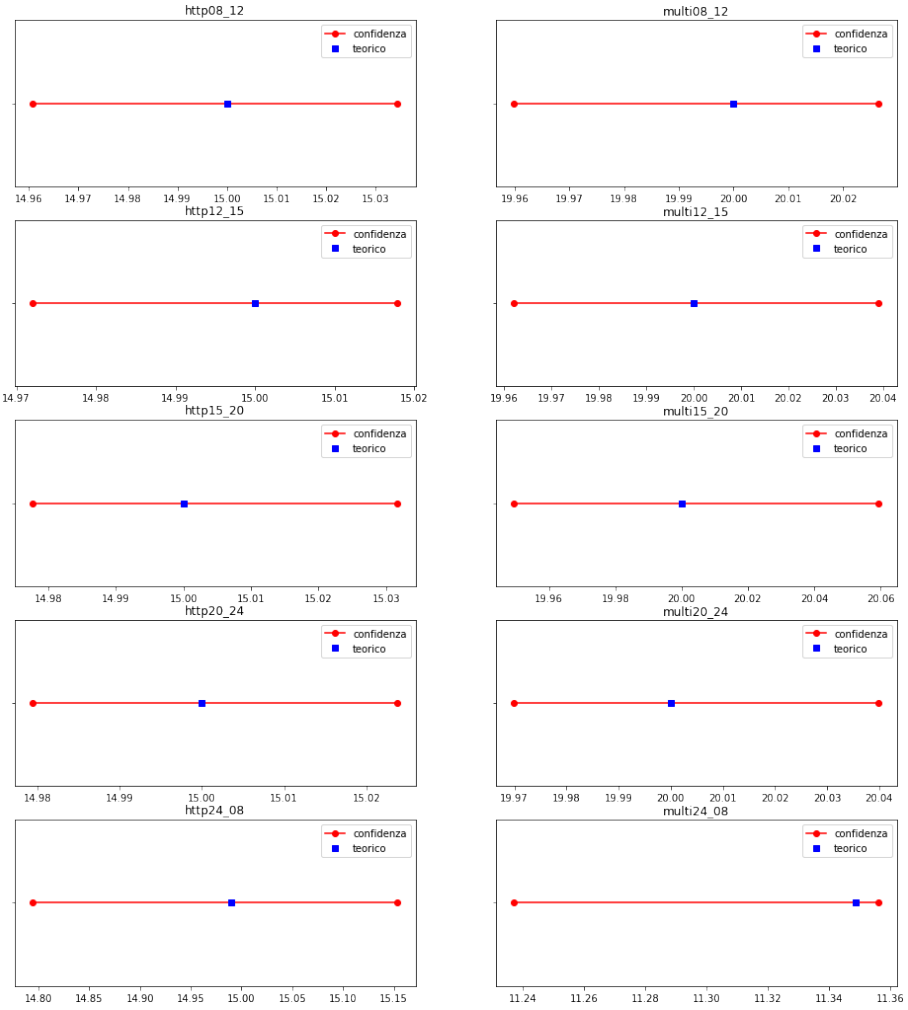


Figura 4.2: Intervalli di confidenza al 98% e medie teoriche di $E[T_{s_2}]$

Anche qui, come nel caso precedente, il sistema si comporta nel modo atteso con i parametri calcolati nella fase analitica, per entrambi i flussi di richieste ed in tutte le fasce orarie.

Si riportano qui di seguito per completezza i valori degli intervalli di confidenza di $E[T_s]$ e ρ per quanto riguarda il primo modello e di $E[T_{s_2}]$ e ρ_2 per quanto riguarda il modello:

Orario	$E[T_s]^-$	$E[T_s]^+$	ρ^-	ρ^+
8:00-12:00	0,007123475	0,007131677	0,821330748	0,821452983
12:00-15:00	0,012004826	0,012034616	0,833106697	0,833295467
15:00-20:00	0,007932661	0,007943914	0,833235540	0,833381515
20:00-24:00	0,011417237	0,011442348	0,749701462	0,750062340
24:00-8:00	0,009994693	0,010020558	0,499714736	0,500353650

Tabella 4.1: Intervalli di confidenza per richieste HTTP

Orario	$E[T_s]^-$	$E[T_s]^+$	ρ^-	ρ^+
8:00-12:00	0,017547786	0,017599215	0,879774952	0,879996993
12:00-15:00	0,018065141	0,018128292	0,914094984	0,914361674
15:00-20:00	0,010770313	0,010782175	0,799824723	0,800042568
20:00-24:00	0,012064908	0,012084167	0,879908713	0,880044200
24:00-8:00	0,009562307	0,009571250	0,533088863	0,533530555

Tabella 4.2: Intervalli di confidenza per richieste multimediali

Orario	$E[T_{s_2}]^-$	$E[T_{s_2}]^+$	ρ_2^-	ρ_2^+
8:00-12:00	0,014960757	0,015034439	0,034242915	0,034280617
12:00-15:00	0,014972069	0,015017861	0,535421402	0,535617445
15:00-20:00	0,014977686	0,015031673	0,127051713	0,127139753
20:00-24:00	0,014979459	0,015023757	0,392603348	0,392918032
24:00-8:00	0,014794386	0,015153381	0,000487268	0,000503343

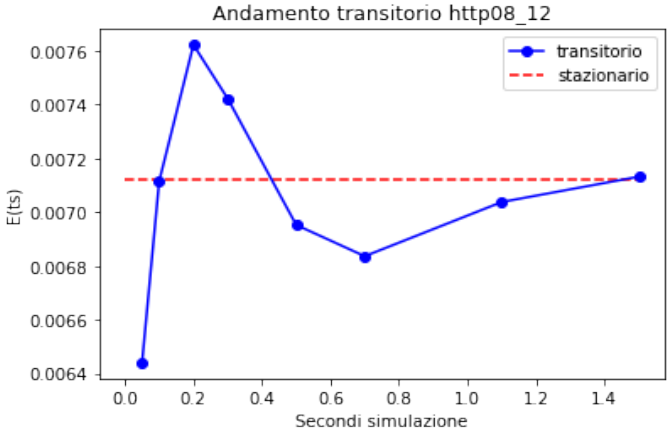
Tabella 4.3: Intervalli di confidenza per richieste HTTP standard

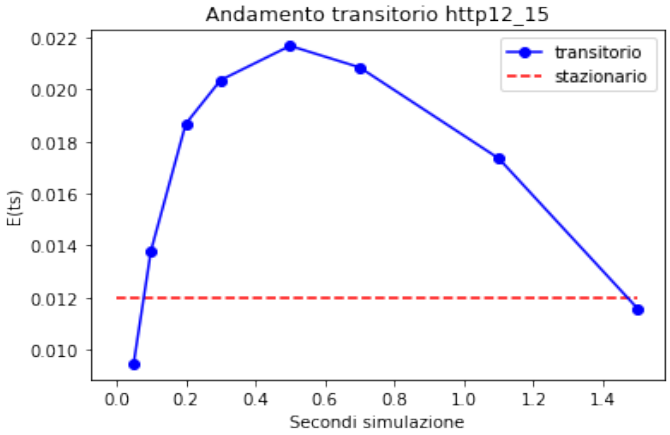
Orario	$E[T_{s_2}]^-$	$E[T_{s_2}]^+$	ρ_2^-	ρ_2^+
8:00-12:00	0,019959890	0,020026449	0,677993258	0,678207051
12:00-15:00	0,019962035	0,020039011	0,755477440	0,755747908
15:00-20:00	0,019949419	0,020059373	0,031388732	0,031442564
20:00-24:00	0,019969909	0,020039842	0,219364553	0,219465421
24:00-8:00	0,011236990	0,011356196	0,000527007	0,000537618

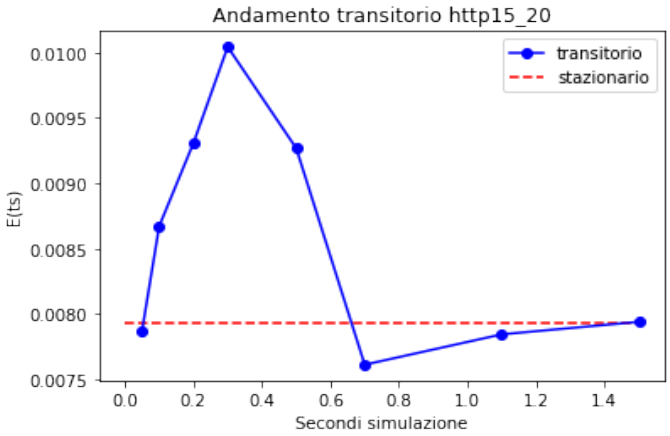
Tabella 4.4: Intervalli di confidenza per richieste multimediali standard

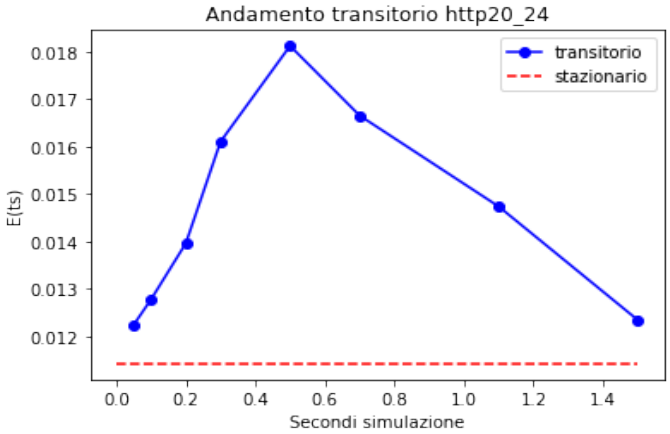
4.2.2 Analisi della fase transiente

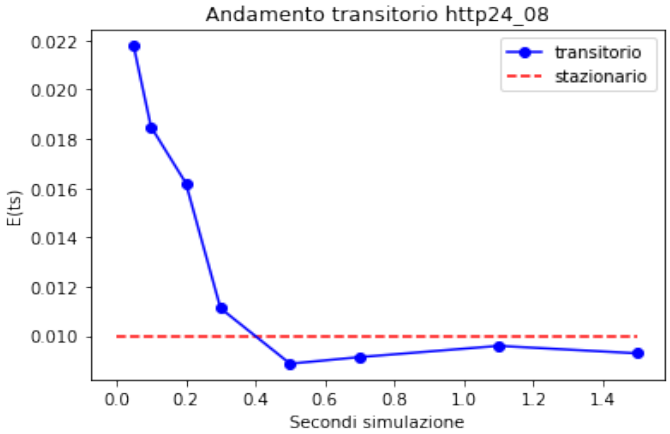
Ad ogni cambio di fascia oraria risultano modificati sia il numero di server attivi che i flussi di richieste in ingresso, ciò comporta naturalmente dei tempi di assestamento del sistema (supponendo comunque come istantanea la variazione del numero di server). Ciò che emerge dall'analisi comunque è che questi tempi di assestamento hanno una durata notevolmente ridotta (dell'ordine dei 2-3 secondi) rispetto alle dimensioni delle fasce orarie; la conseguenza diretta di questo consiste nella trascurabilità di tale periodo nello studio del problema, dato che il Service Provider è maggiormente interessato al comportamento del sistema a regime. Vengono quì di seguito riportati i grafici rappresentativi delle evoluzioni di $E[T_s]$ e di $E[T_{s_2}]$ (per il modello migliorato) da cui si evince come tali valori, dopo rapide oscillazioni, si avvicinano molto al valore stazionario. Questi valori sono stati ricavati come la media delle stesse grandezze nelle 20 simulazioni con semi differenti.

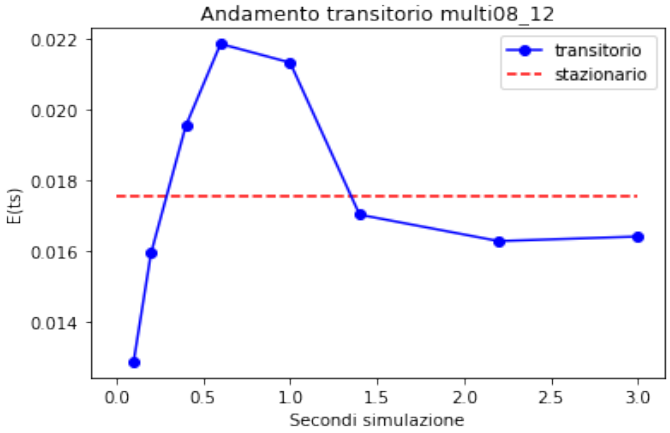


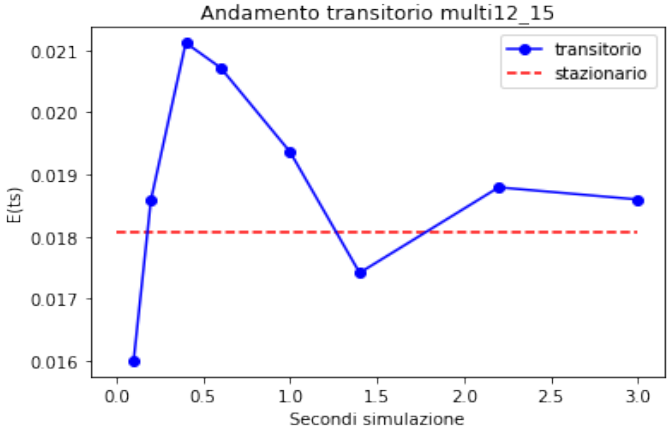


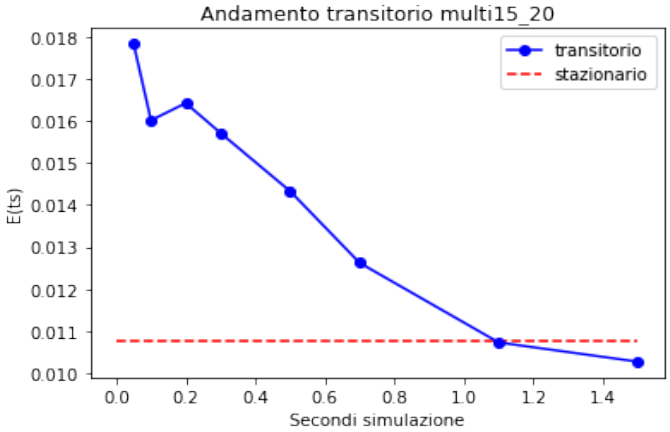

Andamento transitorio http15_20
transitorio
stazionario
E(ts)
Secondi simulazione
0.0100
0.0095
0.0090
0.0085
0.0080
0.0075
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4

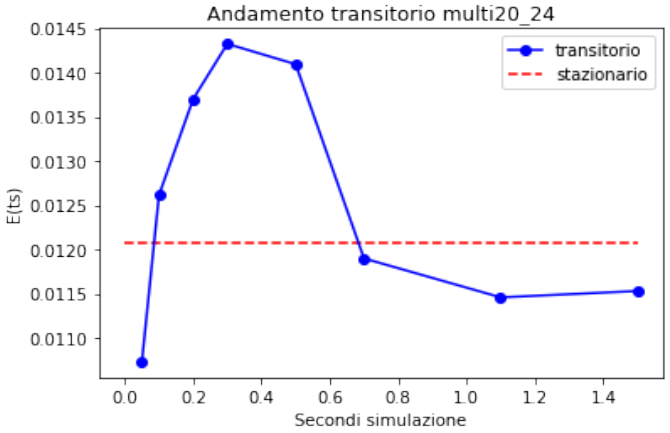

Andamento transitorio http20_24
transitorio
stazionario
E(ts)
Secondi simulazione
0.018
0.017
0.016
0.015
0.014
0.013
0.012
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4

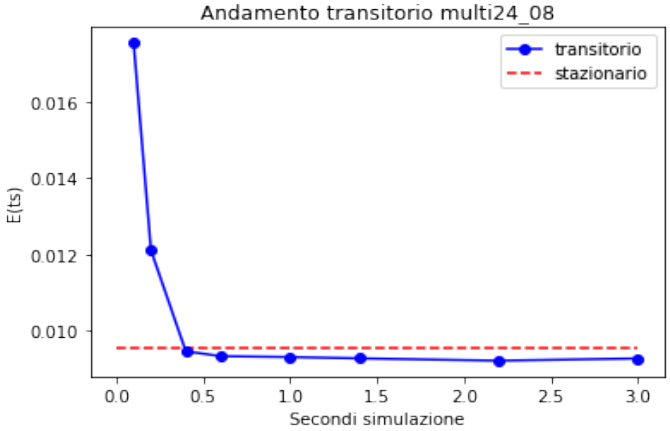

Andamento transitorio http24_08
transitorio
stazionario
E(ts)
Secondi simulazione
0.022
0.020
0.018
0.016
0.014
0.012
0.010
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4

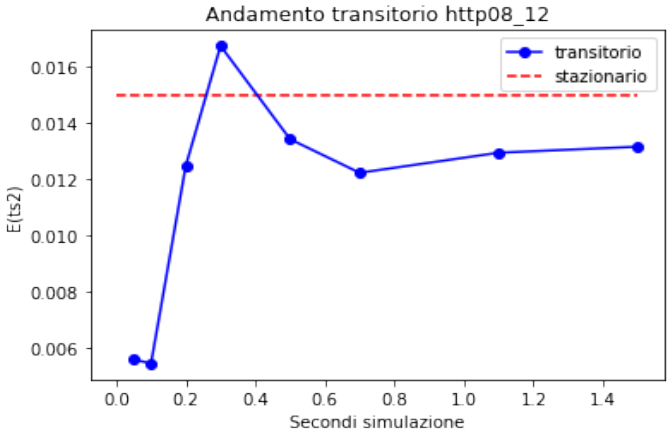

Andamento transitorio multi08_12
transitorio
stazionario
E(ts)
Secondi simulazione
0.022
0.020
0.018
0.016
0.014
0.0
0.5
1.0
1.5
2.0
2.5
3.0

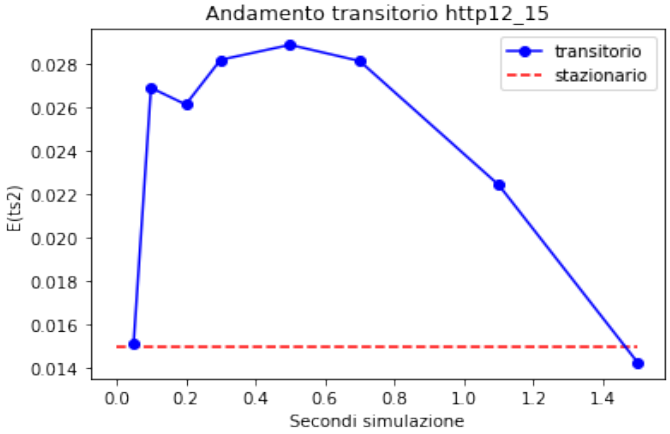

Andamento transitorio multi12_15
transitorio
stazionario
E(ts)
Secondi simulazione
0.021
0.020
0.019
0.018
0.017
0.016
0.0
0.5
1.0
1.5
2.0
2.5
3.0

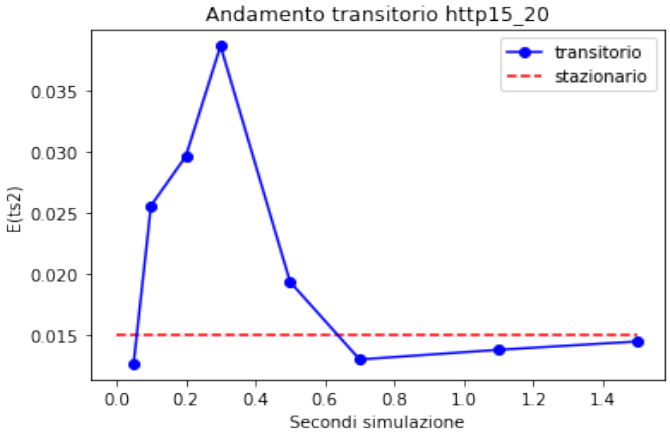

Andamento transitorio multi15_20
transitorio
stazionario
E(ts)
Secondi simulazione
0.018
0.017
0.016
0.015
0.014
0.013
0.012
0.011
0.010
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4

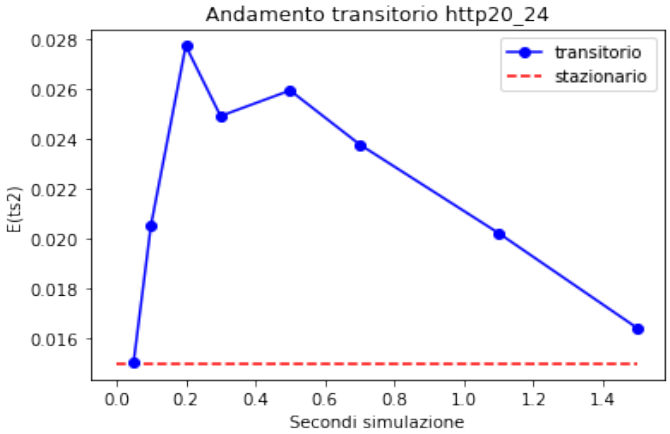

Andamento transitorio multi20_24
transitorio
stazionario
E(ts)
Secondi simulazione
0.0145
0.0140
0.0135
0.0130
0.0125
0.0120
0.0115
0.0110
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4

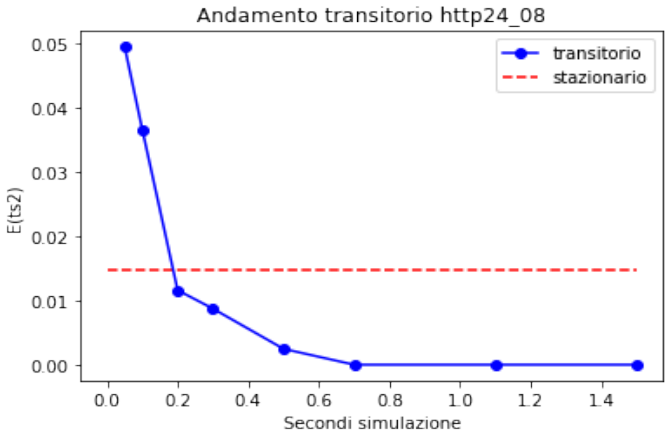

Andamento transitorio multi24_08
transitorio
stazionario
E(ts)
Secondi simulazione
0.016
0.014
0.012
0.010
0.0
0.5
1.0
1.5
2.0
2.5
3.0

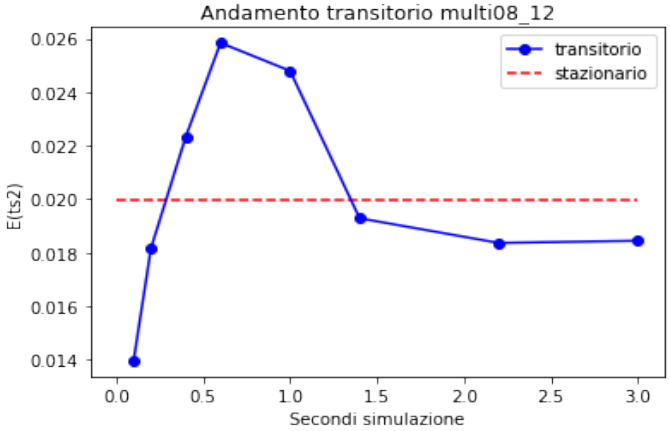

Andamento transitorio http08_12
transitorio
stazionario
E(ts2)
Secondi simulazione
0.016
0.014
0.012
0.010
0.008
0.006
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4

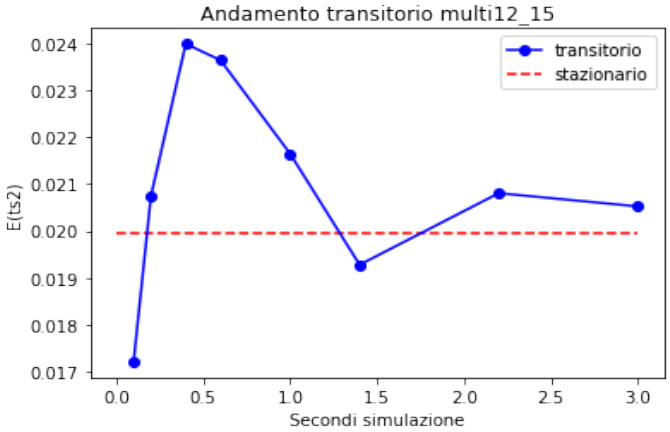

Andamento transitorio http12_15
transitorio
stazionario
E(ts2)
Secondi simulazione

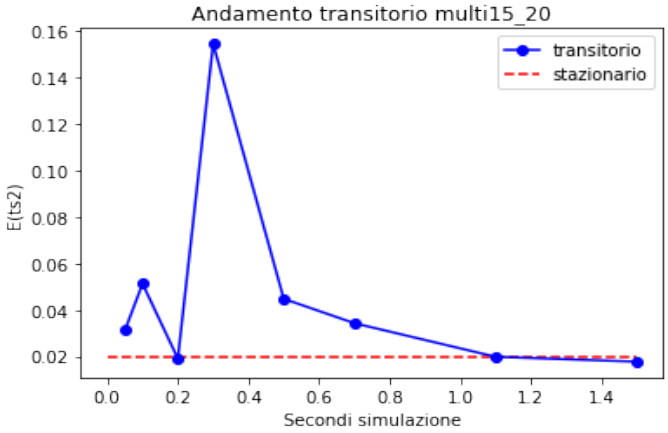

Andamento transitorio http15_20
transitorio
stazionario
E(ts2)
Secondi simulazione

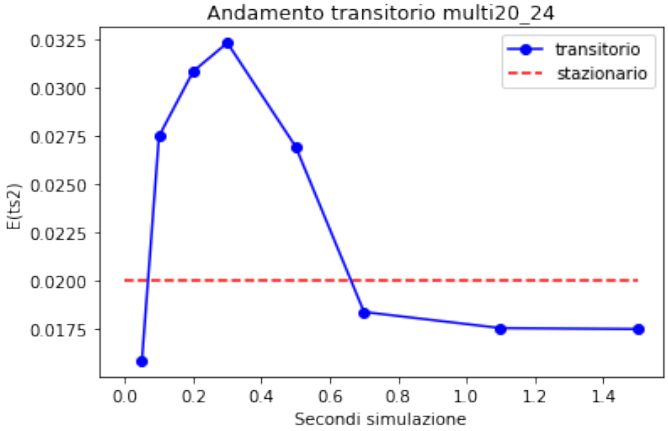

Andamento transitorio http20_24
transitorio
stazionario
E(ts2)
Secondi simulazione

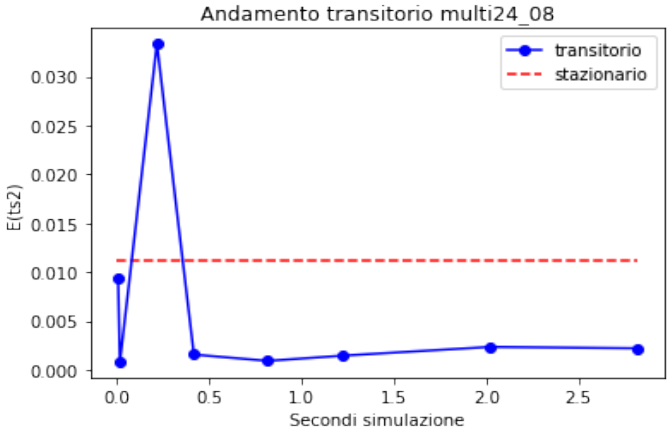

Andamento transitorio http24_08
transitorio
stazionario
E(ts2)
Secondi simulazione


Andamento transitorio multi08_12
transitorio
stazionario
E(ts2)
Secondi simulazione


Andamento transitorio multi12_15
transitorio
stazionario
E(ts2)
Secondi simulazione


Andamento transitorio multi15_20
transitorio
stazionario
E(ts2)
Secondi simulazione
0.16
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4


Andamento transitorio multi20_24
transitorio
stazionario
E(ts2)
Secondi simulazione
0.0325
0.0300
0.0275
0.0250
0.0225
0.0200
0.0175
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4


Andamento transitorio multi24_08
transitorio
stazionario
E(ts2)
Secondi simulazione
0.030
0.025
0.020
0.015
0.010
0.005
0.000
0.0
0.5
1.0
1.5
2.0
2.5

Capitolo 5

Conclusioni

La simulazione effettuata consente di convalidare i risultati ottenuti nella fase di analisi preliminare; pertanto per la soluzione del primo problema posto, si consiglia di acquistare un minimo di 13 server, da distribuirsi nelle varie fasce orarie secondo quanto descritto nella tabella 2.1; in questo modo è possibile minimizzare i costi per il Service Provider mantenendo un livello di servizio adeguato che rispetti lo slowdown richiesto.
Per quanto riguarda la gestione del traffico partizionato tra utenti standard e premium, la simulazione conferma che i valori della tabella 2.4 sono validi e consentono di massimizzare il numero delle sottoscrizioni premium senza degradare eccessivamente le prestazioni per gli utenti standard, questi ultimi vedranno ancora rispettato il precedente vincolo sullo slowdown.
In ultimo si fa notare come, in assenza di ritardi dovuti al riavvio dei server e considerando il valore del flusso come una funzione a gradini, le fasi di transizione tra diverse fasce di orario sono estremamente rapide e non inficiano sul risultato voluto.

Capitolo 6

Organizzazione del codice e dei risultati

Per la parte di simulazione si è scelto di utilizzare il linguaggio C in quanto è un linguaggio molto versatile e permette di creare codice efficiente. Il codice è stato diviso in numerose funzioni e file in modo da renderlo più ordinato (il c prevede dei file header che contengono le definizioni delle funzioni o variabili). Qui di seguito verrà data una breve descrizione dei file e delle funzionalità associate insieme alle istruzioni per eseguire la simulazione.

File	Descrizione
msq.c	punto di lancio delle simulazioni
stat.c	punto di lancio delle analisi statistiche dei risultati
steady.c	simulazione dello stato stazionario
transient.c	simulazione dello stato transiente
rngs.c	generatore di lehmer e definizione delle distribuzioni statistiche usate
utils/csv.c	esportatore dei risultati e importatore dei parametri dal formato CSV
utils/list.c	una lista con inserimento ordinato e un campo per riconoscere la classe dei job
utils/utils.c	delle semplici funzioni helper

I parametri csv di input sono divisi per ogni tipologia di simulazione e sono nelle directory:

- input/transient/
- input/steady/
- input/

Le simulazioni standard supportano csv con i seguenti campi (nel seguente ordine): *label*: una stringa per identificare la sessione di simulazione nei risultati, *seconds* i secondi che costituiscono la durata delle simulazioni transitorie (tipo decimale), m il numero di server nella simulazione (tipo intero), *lambda* il numero di arrivi al secondo (tipo intero), *mu* il numero di job eseguiti al secondo (tipo intero).
Le simulazioni con due classi di priorità richiedono un ulteriore parametro: *p1* la frazione di job nella prima classe o anche la probabilità che un job sia nella classe con priorità superiore (tipo decimale ≤ 1).
I semi per generare i numeri random vengono prelevati dal file /input/seed.csv con un formato di uno per riga.
I risultati della simulazione vengono caricati nei path relativi:

- otuput/transient/priority/
- otuput/transient/standard/
- otuput/transient/stat/
- otuput/steady/standard/
- otuput/steady/priority/
- otuput/steady/stat/

Il file esportato anch'esso csv contiene i campi aggiuntivi: $E[t_s]$ tempo di risposta medio, $E[t_q]$ tempo di coda medio, *util* utilizzazione media dei server.

Il programma è stato sviluppato, compilato ed eseguito su una piattaforma Linux per la compilazione fa uso di un *Makefile* e sul compilatore *gcc* quindi si garantisce il funzionamento su tale piattaforma.
Dopo avere compilato eseguire il file ottenuto msq.o, questa operazione richiede un po' di tempo con le configurazioni originali, si possono modificare i file di input senza modificarne nome/formato a meno di non modificare il sorgente.
I dati prodotti possono essere analizzati dal file stat.o che produrrà nelle directory stat delle statistiche: nel caso dell'analisi stazionaria degli intervalli di confidenza, nel caso dell'analisi transitoria una media campionaria.