Progetto di Performance Modeling of Computer Systems and Networks

a.a.2021/22

Gestione di un cinema

Appetito Giulio

Brinati Anastasia

Università degli Studi
di Roma Tor Vergata

SOMMARIO

MODELLO INIZIALE

0. INTRODUZIONE

0.1 Descrizione del sistema

Il sistema considerato per il nostro studio è un cinema multisala. La scelta è stata dettata da esperienze personali, essendo il cinema un luogo da noi frequentato solitamente: questo ci ha dato diversi spunti per lo sviluppo del progetto, avendo toccato con mano le problematiche che una attività di questo tipo può presentare. Il riferimento scelto per il nostro lavoro è il cinema Ariston di Colleferro (RM), grazie al quale abbiamo avuto modo di ottenere dei dati tramite un'intervista al titolare dell'attività.

Gli utenti che accedono al sistema possono essere di due tipologie: *senza biglietto*, o *con biglietto online*.

- Gli utenti del primo tipo dovranno acquistare il biglietto presso uno dei due sportelli della biglietteria, presente all'esterno del cinema, a partire dai 60 minuti che precedono l'inizio della proiezione;
- Gli utenti con biglietto online non avranno necessità di accodarsi presso gli sportelli della biglietteria, avendo accesso diretto alla struttura.

Una volta ottenuto l'accesso alla struttura, gli utenti devono sottoporsi al *controllo dei biglietti*, che viene effettuato - da uno o più addetti - immediatamente dopo l'ingresso del cinema.
Successivamente, gli utenti che hanno superato il controllo dei biglietti hanno a disposizione dei servizi facoltativi prima di entrare all'interno della sala per la proiezione del film. In particolare, la struttura offre:

- un'*area food*, in cui i clienti hanno modo di acquistare cibi e bevande da consumare durante la proiezione;
- un'*area gadgets*, in cui i clienti hanno la possibilità di acquistare merchandising a tema.

Il cinema in questione offre tre diversi orari per la visione di film:

- *16:00 (operatività cinema : 15.00 -> 17.00)*
- *20:00 (operatività cinema : 19.00 -> 21.00)*
- *23:00. (operatività cinema : 22.00 -> 00.00)*

Inoltre, i film proiettati sono principalmente di 3 tipologie:

- *film per bambini / cartoni animati*
- *supereroi / azione*
- *horror.*

0.2 Aspetti critici

Gli aspetti critici individuati relativi ai clienti sono i seguenti:

- attesa dei clienti in coda per acquistare il biglietto fisicamente presso gli sportelli della biglietteria;
- attesa dei clienti in coda per la convalida del proprio biglietto presso l'entrata del cinema;
- attesa dei clienti in coda per essere serviti presso l'area food, e conseguente attesa per il pagamento presso le rispettive casse;
- attesa dei clienti in coda per il pagamento presso le casse dell'area gadgets.

Inoltre, un arco di tempo eccessivamente elevato per accedere alla sala impatterebbe i guadagni derivanti dagli sponsor pubblicitari i quali, proiettando i propri spot prima dell'inizio del film, hanno concordato con il cinema un pagamento proporzionale al numero di spettatori presenti in sala durante la proiezione dei suddetti spot.

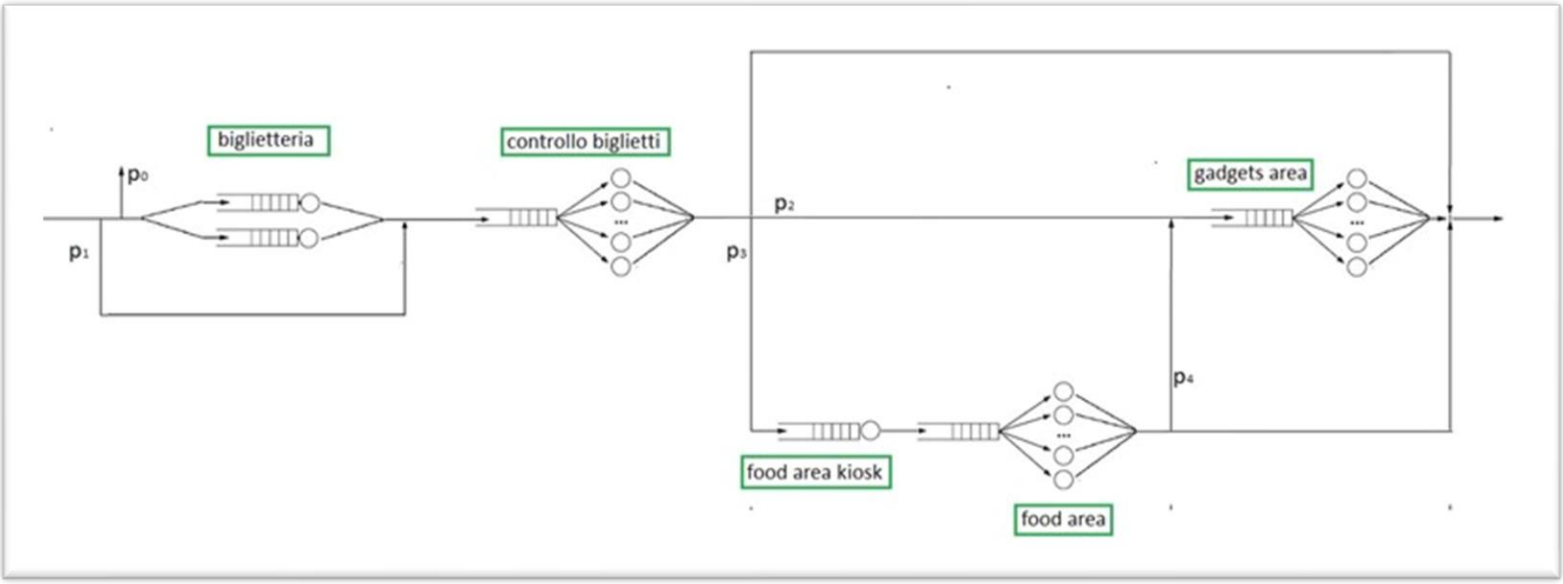
1. OBIETTIVI

L'obiettivo del presente studio è quello di massimizzare i profitti da parte del cinema.
Pertanto, si vuole determinare la configurazione ottima del sistema, individuando il numero ottimale di serventi per ognuno dei sottosistemi nelle varie fasce orarie considerate.

Inoltre si vogliono garantire i seguenti QoS (Quality of Service):

1. Il tempo di risposta complessivo del sistema deve essere mantenuto sotto i 5 minuti per la prima fascia oraria, sotto i 3 minuti per la seconda fascia oraria e 2 minuti per la terza fascia oraria (non considerando i tempi di percorrenza necessari per spostarsi da un centro all'altro);
2. Il 70% degli utenti deve assistere alla proiezione degli spot pubblicitari;

2. MODELLO CONCETTUALE



Il cinema è stato modellato con la rete riportata in figura. I sottosistemi di cui il sistema si compone sono i seguenti:

- *Biglietteria*: Il sottosistema relativo alla biglietteria è stato modellato (rispecchiando la realtà) mediante due **M/M/1** con code infinite, poichè l'esaurimento dei biglietti disponibili non è stato considerato un evento rilevante dal punto di vista probabilistico (relativamente al cinema considerato per il nostro studio);
- *Controllo biglietti:* il sottosistema relativo al controllo dei biglietti è stato modellato con una **M/M/K** con coda infinita, dal momento che è possibile avere più addetti al lavoro contemporaneamente;
- *Cassa Food area:* il sottosistema relativo alla cassa dell'area food è stata modellato, invece, con una **M/M/1** con coda infinita, rispettando la configurazione prevista dal cinema in esame;
- *Food area:* il sottosistema relativo all'area food è stato modellato mediante una **M/M/k** con coda infinita;
- *Gadgets area:* il sottosistema è stato modellato con una **M/M/K** con coda infinita.

2.1 Politiche di scheduling nelle code

Per le code in ciascuno dei sottosistemi considerati abbiamo ragionato seguendo il naturale andamento dei clienti, scegliendo di adottare per ciascuno una politica di scheduling **FIFO,** in quanto è ragionevole pensare che un utente presso uno qualsiasi dei centri venga servito in ordine di arrivo.
Inoltre, le politiche adottate sono **non-preemptive,** dal momento che, seguendo un ragionamento analogo al precedente, in nessuno dei sottosistemi ha senso interrompere il servizio di un utente per riprenderlo in un secondo momento, favorendo il servizio di un altro cliente.
Infine, si è prestata particolare attenzione all'instradamento che avviene in biglietteria: sono presenti due casse distinte, presso le quali gli utenti si indirizzano a seconda della meno affollata; abbiamo deciso di optare per una politica di routing 'least work left', di modo da assicurare che ogni cliente in arrivo si accodi presso il servente meno carico, seguendo la realtà dei fatti.

2.2 Stato del sistema

Ad ogni istante di tempo il sistema viene descritto dalle seguenti variabili di stato:

- *Numero di utenti (per centro)*
- *Numero di utenti in servizio (per centro)*
- *Numero di utenti in coda (per centro)*
- *Stato di ogni servente xi ∈ [busy, idle]*
- *Stato di ogni coda yi ∈ [empty, not_empty]*

2.3 Eventi

Un cambiamento nello stato da parte del sistema può verificarsi a seguito di una serie di eventi. Gli eventi considerati sono di diverse tipologie:

1) Completamento del servizio di un utente/job;
2) Arrivo di un nuovo utente ad un centro;
3) Inizio proiezione spot pubblicitari;
4) Inizio proiezione film;

2.3.1 Descrizione degli eventi

Ognuno dei precedenti eventi produce un cambiamento nello stato del sistema. Per ciascuno dei centri del sistema, si hanno situazioni in parte differenti.

- *Biglietteria*:
  - *Arrivo :* all'arrivo di un nuovo utente presso la biglietteria, questo deve accodarsi presso uno delle due code presenti, ciascuna per uno dei due serventi. Nel caso in cui dei due serventi uno solo sia attivo, la scelta risulta ovviamente obbligata; in caso contrario, la scelta avviene in maniera casuale con uguale probabilità. Nel caso in cui la coda scelta sia vuota (*yi == empty)*, l'utente viene servito presso lo sportello selezionato; in caso contrario (*yi == not_empty)* l'utente dovrà attendere il servizio degli utenti in coda, senza la possibilità di cambiare coda dopo averne selezionata una.
  - *Completamento :* dopo aver ottenuto il servizio presso uno dei due sportelli, l'utente può lasciare il centro e il servente in questione passa dallo stato *busy* allo stato *idle*, procedendo a servire un nuovo utente nella propria coda (se la coda è *not_empty)*.
- *Controllo biglietti:*
  - *Arrivo :* all'arrivo di un utente presso il centro *Controllo biglietti* questo deve accodarsi presso l'unica coda a disposizione (nel caso in cui, ovviamente, la coda sia *not_empty*). Nel caso in cui vi siano più operatori in servizio, l'utente di turno sceglierà il primo operatore libero.
  - *Completamento :* come nel caso precedente, dopo aver completato il servizio di un utente, il singolo operatore passa dallo stato *busy* allo stato *idle*, così da offrire di nuovo servizio agli eventuali altri utenti nella coda, mentre l'utente può scegliere se proseguire direttamente verso la sala cinematografica (concludendo di fatto la propria permanenza nel sistema), passare per l'area food o per l'area gadgets.
- *Cassa Area food:*
  - *Arrivo* : gli utenti che dopo aver completato il servizio scelgono di recarsi presso l'area food devono accodarsi dapprima presso l'unica coda a disposizione (nel caso in cui la coda sia *not_empty*) in attesa di ricevere servizio.

    - *Completamento* : una volta arrivato il proprio turno, l'utente riceve servizio presso la cassa (che passa dallo stato *busy* allo stato *idle*), così da avere accesso all'area food.
- *Area food:*
    - *Arrivo :* una volta completato il servizio presso la cassa, gli utenti che accedono all'area food devono accodarsi ancora presso l'unica coda disponibile, in attesa che uno dei serventi a disposizione (se più di uno) sia nello stato *idle*. La selezione del servente avviene selezionando il primo operatore *idle*.
    - *Completamento :* una volta completato il servizio presso uno degli operatori quest'ultimo passa dallo stato *busy* allo stato *idle*, mentre l'utente servito ha la possibilità di scegliere tra una sosta presso l'*area gadgets* e la prosecuzione diretta verso la sala cinematografica, così concludendo il proprio percorso all'interno del sistema.
- *Area gadgets:*
    - *Arrivo :* gli utenti che scelgono di sostare presso l'area gadgets, una volta selezionati i propri prodotti, deve recarsi presso l'unica coda a disposizione (nel caso la coda sia *not_empty*).
    - *Completamento :* Una volta giunto il proprio turno, l'utente riceve servizio presso uno degli operatori attivi (se più di uno). La selezione dell'operatore avviene, come prima, scegliendo il primo operatore *idle*. Completato il servizio, l'utente si dirige verso la sala cinematografica, concludendo così il proprio percorso.

2.4 Profitto

Si intende ora descrivere ricavi e costi che determinano il profitto del cinema.
Il cinema Ariston di Colleferro prevede, per ciascuna fascia oraria, un numero massimo di 13 dipendenti in totale, che possono ricoprire le varie posizioni presso le varie postazioni (es. controlloBiglietti, ..).
Ciascun dipendente lavora per turni della durata di 3h:

- 15:00-17:00
- 19:00-21:00
- 22:00-24:00

e percepisce in media 7 €/h.

Inoltre, il cinema offre il servizio di ristorazione che dispone di bibite (porzione media 33cl) e popcorn (porzione media 70g), vendute rispettivamente al prezzo medio di 2,5 € e 3,5 € (a porzione).

Queste informazioni saranno da calcolare in base al numero di spettatori che ad ogni fascia accederanno ai servizi, possiamo fare però delle stime teoriche in base ai valori numerici ricavati nel modello delle specifiche.
Tuttavia sappiamo che il cinema spende rispettivamente 160 € (->120litri) e 90 € (->25kg), per ogni fascia oraria, da cui ricaviamo che la spesa media del cinema per l'area food è circa: 250 €.

Infine, il biglietto per ciascuna proiezione ha il prezzo medio di 7€.

3. MODELLO DELLE SPECIFICHE

3.0 Dataset considerati

I dati tenuti in considerazione per lo sviluppo del modello sono stati ottenuti tramite un'intervista al titolare del *cinema multisale Ariston di Colleferro (RM)*.

[…]

	N° medio di spettatori 1°fascia oraria	**N° medio di spettatori 2°fascia oraria**	**N° medio di spettatori 3°fascia oraria**	**% utenti con biglietto online**	**% utenti che sosta presso l'area food**	**% utenti che sosta presso l'area gadget**
Cartoni animati	170	70	30	10%	70%	40%
Horror	10	20	80	15%	20%	5%
Supereroi	250	300	200	40%	80%	25%

3.1 Valori numerici

I dati di input al modello sono: il tasso medio di arrivo da parte degli utenti al cinema, i vari tassi medi di servizio - per ogni singolo servente - nei diversi nodi del sistema e le probabilità di routing tra i sottosistemi.

3.1.1 Tassi medi di arrivo

Il *tasso medio di arrivo* è stato ottenuto a partire dai dati relativi al numero medio di spettatori per spettacolo, tenendo in considerazione le tre diverse tipologie di film offerte dal cinema in questione.

Dal momento che l'arco temporale in cui la biglietteria eroga i biglietti è pari ai 60 minuti che precedono l'inizio del film, abbiamo ottenuto le seguenti statistiche:

- Fascia 1 (15:00 – 16:00): $\boldsymbol{\lambda}$ = (170 + 10 + 250) / 60 = 7.166 arrivi/min
- Fascia 2 (19:00 – 20:00): $\boldsymbol{\lambda}$ = (70 + 20 + 300) / 60 = 6.500 arrivi/min
- Fascia 3 (22:00 – 23:00): $\boldsymbol{\lambda}$ = (30 + 80 + 200) / 60 = 5.166 arrivi/min

3.1.2 Tassi medi di servizio

I *tassi medi di servizio* considerati nello sviluppo del sistema sono gli stessi riportati dall'intervistato.

- *Biglietteria*: il tempo medio di servizio (a utente) è pari a un tempo tra i 15 e i 20 secondi. Dunque, si ottiene un tasso $\boldsymbol{\mu biglietteria}$ = 1 utente /(7/24 min) = 3.428 utenti/min.
- *Controllo biglietti:* il tempo medio di servizio (per utente) per il controllo dei biglietti ha un valore compreso tra gli 8 e i 10 secondi (in media, 9 secondi). Dunque, si ottiene un tasso $\boldsymbol{\mu controllo}$ = 1 utente/ (9/60 minuti) = 6.667 utenti/min.
- *Food area kiosk:* A partire da statistiche raccolte durante l'intervista, il tempo medio di servizio presso la cassa dell'area food è pari a circa 10 secondi per utente. Ne deriva un tasso medio di servizio pari a $\boldsymbol{\mu kiosk}$ = 6 utenti/min.

- *Food area:* sempre seguendo le informazioni ottenute dal direttore del cinema, il tempo medio di servizio per utente è pari a circa 30 secondi, ottenendo un tasso ***µfoodArea*** = 2 utenti/min.
- *Gadgets area:* infine, sempre dalle statistiche ottenute dall'intervista, si ha un tempo medio di servizio pari a circa 45 secondi. Pertanto, si ottiene un tasso medio di servizio pari a ***µgadgetsArea*** = 1.333 utente/min.

3.1.3 Probabilità di routing

Per quanto riguarda le probabilità di routing, i valori seguono quanto riportato dall'intervistato. In particolare, dai dati raccolti si evince come le probabilità di routing variano a seconda della fascia oraria considerata. Questo è dovuto al fatto che – come spiegatoci dall'intervistato – ogni genere di film ha un proprio pubblico ben specifico, con caratteristiche differenti.
I valori percentuali delle persone che sostano presso l'area gadgets forniti dall'intervista erano relativi all'afflusso complessivo, motivo per il quale abbiamo scelto in seguito di suddividere il flusso in due porzioni, una relativa alle persone che sostano sia presso l'area food che l'area gadgets, ed una seconda relativa agli utenti che scelgono di sostare unicamente presso l'area gadgets. Dunque, a sostare presso l'area gadgets dopo aver concluso il servizio presso l'area food sono il 15% degli utenti per film di *supereroi*, il 20 degli utenti per i film *cartoni animati* e il 3% degli utenti per i film *horror;* conseguentemente, le restanti percentuali – rispettivamente, il 10%, 20% e 2% - degli utenti sosta presso l'area gadgets direttamente, senza passare per l'area food.

Riassumendo per le tre diverse fasce orarie, i valori individuati sono i seguenti:

Fascia 1 (15:00 – 16:00)

- P1 = (0.1*170 + 0.15*10 + 0.4*250) / (170 + 10 + 250) = 0.2755 = 27.55%
- P2 = (0.2*170 + 0.02*10 + 0.10*250) / (170 + 10 + 250) = 0.1376 = 13.76%
- P3 = (0.7*170 + 0.2*10 + 0.8*250) / (170 + 10 + 250) = 0.7465 = 74.65 %
- P4 = (0.2857*170 + 0.1*10 + 0.125*250) / (170 + 10 + 250) = 0.1879 = 18.79%

Fascia 2 (19:00 – 20:00)

- P1 = (0.1*70 + 0.15*20 + 0.4 *300) / (70 + 20 + 300) = 0.3333 = 33.33%
- P2 = (0. 2*70 + 0. 02*20 + 0. 10*300) / (70 + 20 + 300) = 0.1138 = 11.38%
- P3 = (0.7*70 + 0.2*20 + 0.8*300) / (70 + 20 + 300) = 0.7512 = 75.12%
- P4 = (0.2857*70 + 0. 1*20 + 0. 125*300) / (70 + 20 + 300) = 0.1525 = 15.25%

Fascia 3 (22:00 – 23:00)

- P1 = (0.1*30 + 0.15*80 + 0.4*200) / (30 + 80 + 200) = 0.3064 = 30.64%
- P2 = (0. 2*30 + 0.02*80 + 0. 10*200) / (30 + 80 + 200) = 0.0890 = 8.90%
- P3 = (0.7*30 + 0.2*80 + 0.8*200) / (30 + 80 + 200) = 0.6354 = 63.54 %
- P4 = (0.2857*30 + 0.1*80 + 0. 125*200) / (30 + 80 + 200) = 0.1341 = 13.41%

3.1.4 Costi proiezione spot pubblicitari

Inoltre, al fine di avere un'idea dei costi per la proiezione di spot pubblicitari in una sala cinematografica, abbiamo preso spunto dal sito web di un cinema di Pisa, in cui sono riportate le tariffe disponibili.

€ 360 /mese	€ 460 /mese	€ 560 /mese	€ 660 /mese
ABBONAMENTO ANNUALE	ABBONAMENTO PER 6 MESI	ABBONAMENTO PER 3 MESI	ABBONAMENTO MENSILE
3.000 Passaggi da 15" garantiti	1.500 Passaggi da 20" garantiti	750 Passaggi da 20" garantiti	270 Passaggi da 20" garantiti
9 passaggi giornalieri	9 passaggi giornalieri	9 passaggi giornalieri	9 passaggi giornalieri
Bacino di 180.000 spettatori	Bacino di 90.000 spettatori	Bacino di 45.000 spettatori	Bacino di 15.000 spettatori
Possibilità di modificare lo spot 2 volte	Possibilità di modificare lo spot 1 volta	~~Possibilità di modificare lo spot 2 volte~~	~~Possibilità di modificare lo spot 2 volte~~
Spot personalizzato al prezzo di 400€	Spot personalizzato al prezzo di 400€	Spot personalizzato al prezzo di 400€	Spot personalizzato al prezzo di 400€
Condivisione dello spot su Youtube	Condivisione dello spot su Youtube	Condivisione dello spot su Youtube	Condivisione dello spot su Youtube
Possibilità – a pagamento – di uno spazio di promozione all'ingresso del Cinema	Possibilità – a pagamento – di uno spazio di promozione all'ingresso del Cinema	Possibilità – a pagamento – di uno spazio di promozione all'ingresso del Cinema	Possibilità – a pagamento – di uno spazio di promozione all'ingresso del Cinema
ORDINA	ORDINA	ORDINA	ORDINA

"http://pubblicitacinemapisa.com/?page_id=3025"

Poiché il proprietario del cinema ci ha fornito come informazione al riguardo il fatto che offrono un'unica opzione con prezzo fisso (senza esporsi su di esso) per la proiezione di uno spot per due settimane, abbiamo deciso di utilizzare come riferimento il costo dell'abbonamento mensile riportato in figura.

$$costoProiezione = 330\,{}^{€}/_{2w} = 165\,{}^{€}/_{w} = 23.60\,{}^{€}/_{d} \cong 8\ {}^{€}/_{proiez}$$

Ogni proiezione ha la durata di 20 secondi, ed è previsto per il nostro sistema che gli spot pubblicitari vengano proiettati per 15 minuti in totale. Abbiamo calcolato quindi il numero di proiezioni in una fascia oraria:

$$numeroProiezioni = 15min/20s = 45proiez$$

Inoltre, il bacino di utenza considerato dal cinema in riferimento è 15000 spettatori al mese, ovvero 500 al giorno, per cui in una fascia oraria:

$$totaleSpettatori = \frac{15000\,{}^{spett}/_{m}}{30d * 3} \cong 167\ spett$$

Nel nostro caso, ogni spot viene riprodotto tre volte al giorno per 15 giorni, risultando in un ipotetico abbonamento bisettimanale del costo di:

$$abbonamento = 8€ * 3 * 15d = 360€$$

Infine abbiamo ricavato il guadagno proveniente da un singolo spettatore di modo da proporzionare successivamente il guadagno reale del cinema proveniente dalle proiezioni pubblicitarie rispettivamente all'effettivo totale degli arrivi di una fascia oraria.
Per ciascuna fascia oraria risulta che uno spettatore produce un income per il cinema di:

$$guadagnoASpettatore = \frac{costoProiezione * numeroProiezioni}{totaleSpettatori} = \frac{8€ * 45proiez}{167spett} = 2.155€$$

4. MODELLO COMPUTAZIONALE

4.1 Intro

Per lo sviluppo del modello computazionale è stato scelto di utilizzare un linguaggio general purpose quale *C,* con il quale è stato scritto il codice per il simulatore.
Abbiamo iniziato sviluppando un simulatore in grado di completare una singola esecuzione e restituire, stampando su schermo, le statistiche ottenute per ogni singolo centro e per il tempo di risposta del sistema.
Successivamente, il codice relativo al simulatore è stato inglobato in un programma in grado di effettuare più run consecutivamente, specificando il tempo di simulazione.

4.2 Strutture dati utilizzate

All'interno del codice del simulatore abbiamo fatto uso di alcune struct, utili per vari scopi.

```
struct{
  double current;          /* current time                */
  double next;            /* next (most imminent) event time    */
  double last;           /* last arrival time           */
} t;
```

t. Per tenere traccia del tempo simulato abbiamo usato questa struct *t* , che memorizza nei propri tre campi, rispettivamente, il tempo dell'evento corrente, del prossimo evento con il tempo più imminente e l'istante dell'ultimo evento. Ad ogni nuovo evento, le variabili di questa struct vengono man mano aggiornate. La simulazione, partendo dal valore *t.current* inizializzato con un valore specifico (0.0) arresta gli arrivi ad un certo istante di tempo (*close the door*), dopo il quale il programma prosegue fintanto che vi sono jobs da servire all'interno del sistema, ma arrestando i nuovi arrivi.

```
typedef struct center{
  double node;           /* time integrated number in the node  */
  double queue;          /* time integrated number in the queue */
  double service;        /* time integrated number in service   */
  double index;          /* used to count departed jobs */
  double number;           /* number in the node */
  double servers;
  char *name;
}center;
```

center. Per modellare a livello computazionale ognuno dei diversi sottosistemi (centri) che compongono il sistema abbiamo introdotto la struct *center*, la quale permette di descrivere, istante per istante, lo stato di un centro.
I campi *node, queue e service* mantengono rispettivamente il numero di jobs presenti nel centro,in coda e in servizio integrato nel tempo; il campo *index* viene utilizzato come contatore dei jobs serviti dal centro considerato; il campo number, invece, tiene traccia, istante per istante, dei jobs presenti all'interno del centro; infine, il campo *servers* memorizza il numero di serventi di cui il centro dispone (adeguatamente inizializzato in base ai dati di input al modello), mentre il campo *name* non ha un'utilità effettiva, ma viene utilizzata semplicemente per riferire un centro con il proprio nome quando è necessario visualizzare all'utente le statistiche.

Ad ogni occorrenza di un evento i campi di questa struct vengono ad essere modificati al fine di ottenere, una volta terminata la simulazione, le statistiche desiderate per ognuno dei vari centri. In particolare, ad ogni avanzamento del clock si va a sommare al valore corrente dei campi *node, queue e service* il valore, rispettivamente, delle persone nel centro,in coda e in servizio moltiplicati per la differenza di tempo tra l'istante precedente e l'istante corrente.

```
typedef struct serviceData{
  double mean;
  int stream;
}serviceData;
```

serviceData. Questa struct è stata introdotta al fine di mantenere i parametri relativi ad ogni singolo servente di un dato centro. In particolare, il campo *mean* memorizza il tasso medio di servizio di un singolo server, mentre il campo *stream* memorizza l'indice dello stream utilizzato per la generazione di numeri pseudocasuali, necessario a generare i tempi di servizio.

```
typedef struct event{          /* the next-event        */
  double t;                    /*   next event time     */
  int    x;                    /*   event status, 0 or 1 */
}event;
```

event. La struct *event* è stata utilizzata per tenere traccia dei vari tipi di eventi che possono modificare lo stato del sistema : durante l'esecuzione della simulazione viene allocato un array di *event* di lunghezza pari al numero di tipologie di eventi considerato (in questo caso, 12). Ognuno di questi *event* mantiene nel campo *t* l'istante di tempo del prossimo evento, mentre il campo *x* funge da "semaforo" per un certo tipo di evento : la variabile viene settata ad *1* nel caso in cui un evento di questa tipologia sia disponibile per essere gestito, mentre viene settata a *0* nel caso in cui non sia attivo nessun evento di questo tipo. Nello specifico, l'array utilizzato nella simulazione ordina nel seguente modo i tipi di evento possibili (ordinati per indice) :

0. Arrivo alla *biglietteria*
1. Completamento della *biglietteria* [0]
2. Completamento della *biglietteria* [1]
3. Arrivo al *controlloBiglietti* (proveniente dalla *biglietteria*[0])
4. Arrivo al *controlloBiglietti* (proveniente dalla *biglietteria*[1])
5. Completamento del *controlloBiglietti*
6. Arrivo alla *cassaFoodArea*
7. Completamento della *cassaFoodArea*
8. Arrivo alla *foodArea*
9. Completamento della *foodArea*
10. Arrivo alla *gadgetsArea*
11. Completamento della *gadgetsArea*

```
typedef struct multiserver{
   double service;
   int served;
   int occupied;
} multiserver;
```

multiserver. Questa struct è risultata utile a modellare l'insieme di server di un centro multiservente. Il campo *service* ha un ruolo analogo al campo *service* della struct center. Il campo *served* invece tiene il conto dei jobs serviti dallo specifico servente, mentre infine *occupied* funge da semaforo per segnalare se il servente in questione è busy o idle.

```
typedef struct outputStats{
  double avgInterarrivalTime;
  double avgArrivalRate;
  double avgWait;
  double avgDelay;
  double avgServiceTime;
  double avgNumNode;
  double avgNumQueue;
  double utilization;
  char *name;
  int jobs;
}outputStats;
```

outputStats. Questa struct è risultata utile nel momento in cui, una volta terminata la simulazione, bisognava memorizzare le statistiche di output di nostro interesse. I campi *avgInterarrivalTime* e *avgArrivalRate* memorizzano, rispettivamente, il tempo medio di interarrivo e il tasso medio di arrivo; il campo *wait* memorizza il tempo medio trascorso da un job all'interno del nodo considerato (tempo di risposta); *avgDelay,* invece, è il tempo medio in coda all'interno del nodo. Per concludere le statistiche relative al tempo troviamo *avgServiceTime,* che memorizza appunto il tempo medio di servizio presso il nodo considerato.
Passando alle statistiche *time averaged* troviamo i campi *avgNumNode* e *avgNumQueue* , che memorizzano rispettivamente il numero medio di job nel nodo e in coda.
Per finire, il campo *utilization* memorizza l'utilizzazione del nodo.

4.3 Generazione di numeri pseudocasuali

Per la generazione di numeri pseudocasuali abbiamo fatto uso della librera *rngs.h* , la quale ci è stata utile in particolare delle funzioni *PlantSeeds(SEED),* necessaria ad inizializzare tutti gli streams, *SelectStream(stream)* per selezionare un particolare stream per un dato processo stocastico.
Dal momento che sia i tempi di interarrivo che di servizio sono stati modellati con la distribuzione Esponenziale (ognuna con la rispettiva media) per la generazione dei suddetti tempi è stata utilizzata la funzione – sempre offerta dalla libreria *rngs.h* – *GetArrival()*, generando di volta in volta valori grazie alla funzione *Exponential()*.
Per quanto riguarda le probabilità di routing, invece, abbiamo utilizzato delle variabili con distribuzione uniforme, sfruttando dunque a tal proposito la funzione *Uniform(a,b)* (nel nostro caso scegliendo l'intervallo [0,1]).
Una volta individuati i vari processi stocastici in gioco nel sistema abbiamo proceduto a selezionare uno stream per ognuno di essi.

- STREAM_ARRIVALS 0 : utilizzato per la generazione dei tempi di arrivo al sistema;
- STREAM_BIGLIETTERIA0 1 : utilizzato per la generazione dei tempi di servizio della *biglietteria[0];*
- STREAM_BIGLIETTERIA1 2 : utilizzato per la generazione dei tempi di servizio della *biglietteria[1];*

- STREAM_CONTROLLOBIGLIETTI 3 : utilizzato per la generazione dei tempi di servizio del centro *controlloBiglietti;*
- STREAM_CASSA_FOOD_AREA 4 : utilizzato per la generazione dei tempi di servizio del centro *cassaFoodArea;*
- STREAM_FOOD_AREA 5 : utilizzato per la generazione dei tempi di servizio del centro *foodArea;*
- STREAM_GADGETS_AREA 6 : utilizzato per la generazione dei tempi di servizio del centro *gadgetsArea;*
- STREAM_ROUTING_TICKETMODE 7 : utilizzato per la generazione della probabilità di routing relativa all'arrivo di utenti con biglietto *online* o *fisico;*
- STREAM_ROUTING_CONTROLLO 8 : utilizzato per la generazione della probabilità di routing relativa alla distinzione tra utenti che decidono di sostare presso la *foodArea,* presso la *gadgetsArea* o che decidono di proseguire direttamente, uscendo dal sistema*;*
- STREAM_ROUTING_GADGETS 9 : utilizzato per la generazione della probabilità di routing relativa alla distinzione tra utenti che decidono di sostare presso la *gadgetsArea* dopo la *foodArea* e quelli che decidono di proseguire, uscendo dal sistema.

5. VERIFICA

Giunti a questo punto, è risultato necessario andare ad effettuare una verifica del simulatore sviluppato. Questa pratica consiste nell'attestare che il modello computazionale realizzato produca risultati corretti per il modello in esame: ci siamo preoccupati di confrontare i valori delle statistiche ottenuti dalle varie run di simulazione con i valori teorici provenienti dal modello analitico.
Per valutare il simulatore abbiamo scelto di considerare le statistiche dello stato stazionario, effettuando una simulazione ad orizzonte infinito con k=128 batch di dimensione b=1024 job, avendo come configurazione di serventi

{biglietteria, controllo biglietti, cassa area food, area food, gadgets area}={2, 2, 1, 3, 2}.

Tale scelta è legata al fatto per cui i valori teorici delle varie statistiche sono relative ad un sistema stazionario. L' initial SEED scelto per effettuare la verifica è stato 123456789.
Per il nostro studio abbiamo considerato i seguenti punti per svolgere la verifica:

- Il tempo medio di risposta per ogni centro deve essere sempre uguale alla somma tra tempo medio d'attesa e tempo medio di servizio;
- Il numero medio di utenti in ogni centro deve essere sempre uguale alla somma tra numero medio di utenti in coda e numero medio di utenti in servizio;
- Confronto dei valori ottenuti dalla simulazione e valori teorici per ogni statistica ed ogni centro.

4.4.1 Biglietteria

```
biglietteria
Statistics based upon 128 data points.
With 95% confidence,the expected values are in the intervals:

avgInterarrivalTime
0.193 +/- 0.001

avgArrivalRate
5.195 +/- 0.029

avgWait
1.177 +/- 0.043

avgDelay
0.886 +/- 0.042

avgServiceTime
0.290 +/- 0.002

avgNumNode
6.124 +/- 0.241

avgNumQueue
4.617 +/- 0.232

utilization
0.753 +/- 0.006
```

$\lambda_{biglietteria} = \lambda * (1 - p_{online_1}) = 7.166 * (1 - 0.2755) = 5.191$ job/min
$m = 2$

$\mu_i = 3.428\ job/min$

$\rho = \lambda_{biglietteria} * \frac{E[S_i]}{m} = 0.757$

$E[T_q] = \frac{\rho * E[S_i]}{1-\rho} = 0.908$ min

$E[T_s] = E[T_q] + E[S_i] = 1.200$ min

Consistency check:

- $E[T_s] = 1.177$ min

  $E[T_q] + E[S] = 0.886 + 0.290 = 1.176\ min$ ✓

- $E[N_q] + m * \rho = 4.617 + 2 * 0.753 = 6.123$

  $E[N] = 6.123$ ✓

4.4.2 ControlloBiglietti

```
controlloBiglietti
Statistics based upon 128 data points.
With 95% confidence,the expected values are in the intervals:

avgInterarrivalTime
0.140 +/- 0.001

avgArrivalRate
7.173 +/- 0.039

avgWait
0.211 +/- 0.003

avgDelay
0.061 +/- 0.002

avgServiceTime
0.075 +/- 0.000

avgNumNode
1.513 +/- 0.025

avgNumQueue
0.437 +/- 0.019

utilization
0.538 +/- 0.004
```

$\lambda = \ \lambda = 7.166\ job/min$

$\mu = 6.667\ job/min$

$E[S_i] = \frac{1}{\mu} = 0.149\ min$

$m = 2$

$E[S] = \frac{E[S_i]}{\mathrm{m}} = \frac{1}{m*\mu} = 0.0749\ min$

$\rho = \lambda * E[S] = 0.536$

$$E[T_q] = \frac{P_q * E[S]}{1-\rho} = 0.0602\ min$$

$$P_q = \frac{(m\rho)^m}{m!\,(1-\rho)} * \left(\sum_{i=0}^{m-1} \frac{(m\rho)^i}{i!} + \frac{(m\rho)^m}{m!\,(1-\rho)}\right)^{-1} = \frac{(2*\rho)^2}{2!\,(1-\rho)} * \left(\sum_{i=0}^{1} \frac{(2\rho)^i}{i!} + \frac{(2\rho)^2}{2!\,(1-\rho)}\right)^{-1}$$

$$= 0.446$$

$$E[T_s] = E[T_q] + E[S_i] = 0.258\ min$$

$$E[N_q] = \lambda * E[T_q] = 0.659$$

$$E[N] = \lambda * E[T_s] = 1.848$$

Consistency check:

- $E[T_s] = 0.211\ min$

  $E[T_q] + E[S_i] = 0.061 + 0.075 * 2 = 0.211\ min$ ✓

- $E[N_q] + m * \rho = 0.437 + 2 * 0.538 = 1.513$

  $E[N] = 1.513$ ✓

4.4.3 CassaFoodArea

```
cassaFoodArea
Statistics based upon 128 data points.
With 95% confidence,the expected values are in the intervals:

avgInterarrivalTime
0.187 +/- 0.001

avgArrivalRate
5.349 +/- 0.030

avgWait
1.518 +/- 0.144

avgDelay
1.352 +/- 0.144

avgServiceTime
0.166 +/- 0.001

avgNumNode
8.182 +/- 0.815

avgNumQueue
7.293 +/- 0.811

utilization
0.890 +/- 0.006
```

$\lambda_{cassaFoodArea} = \lambda * p_{foodArea_1} = 7.166 * 0.7465 = 5.3494\ job/min$

$m = 1$

$\mu = 6.0\ job/min$

$$E[S] = \frac{1}{\mu} = \frac{1}{6.0} = 0.166\ min$$

$$\rho = \frac{\lambda_{cassaFoodArea}}{\mu} = \frac{5.3494}{6.0} = 0.891$$

$$E[T_q] = \frac{\rho * E[S]}{1-\rho} = \frac{0.891 * 0.166}{1-0.891} = 1.356\ min$$

$$E[T_s] = E[T_q] + E[S] = 1.522\ min$$

$$E[N_q] = \lambda_{cassaFoodArea} * E[T_q] = 7.253$$

$$E[N] = \lambda_{cassaFoodArea} * E[T_s] = 8.141$$

Consistency check:

- $E[T_s] = 1.518$

  $E[T_q] + E[S] = 1.352 + 0.166 = 1.518$ ✓
- $E[N_q] + \rho = 7.293 + 0.890 = 8.183$

  $E[N] = 8.182$ ✓

4.4.4 FoodArea

```
foodArea
Statistics based upon 128 data points.
With 95% confidence,the expected values are in the intervals:

avgInterarrivalTime
0.187 +/- 0.001

avgArrivalRate

avgWait
1.634 +/- 0.151

avgDelay
1.136 +/- 0.149

avgServiceTime
0.166 +/- 0.001

avgNumNode
8.794 +/- 0.858

avgNumQueue
6.137 +/- 0.845

utilization
0.886 +/- 0.007
```

$$\lambda_{foodArea} = \lambda * p_{foodArea_1} = 7.166 * 0.7465 = 5.3494\ job/min$$

$$\mu = 2.0\ job/min$$

$$E[S_i] = \frac{1}{\mu} = 0.500\ min$$

$$N = 3$$

$$E[S] = \frac{E[S_i]}{m} = \frac{1}{m*\mu} = 0.166\ min$$

$\rho = \lambda_{foodArea} * E[S] = 0.888$

$E[T_q] = \frac{P_q * E[S]}{1-\rho} = 1.233\ min$

$P_q = \frac{(m\rho)^m}{m!\,(1-\rho)} * \left(\sum_{i=0}^{m-1} \frac{(m\rho)^i}{i!} + \frac{(m\rho)^m}{m!\,(1-\rho)}\right)^{-1} = \frac{(2*\rho)^2}{2!\,(1-\rho)} * \left(\sum_{i=0}^{1} \frac{(2\rho)^i}{i!} + \frac{(2\rho)^2}{2!\,(1-\rho)}\right)^{-1}$

$= 0.8022$

$E[T_s] = E[T_q] + E[S_i] = 1.733\ min$

$E[N_q] = \lambda_{foodArea} * E[T_q] = 6.596$

$E[N] = \lambda_{foodArea} * E[T_s] = 7.488$

Consistency check:

- $E[T_s] = 1.713$

  $E[T_q] + E[S_i] = 1.213 + 0.167 * 3 = 1.714$ ✓

- $E[N_q] + \rho * m = 6.497 + 0.891 * 3 = 9.170$

  $E[N] = 9.170$ ✓

4.4.5 GadgetsArea

```
gadgetsArea
Statistics based upon 128 data points.
With 95% confidence,the expected values are in the intervals:

avgInterarrivalTime
0.511 +/- 0.003

avgArrivalRate
1.960 +/- 0.010

avgWait
1.617 +/- 0.043

avgDelay
0.867 +/- 0.041

avgServiceTime
0.375 +/- 0.002

avgNumNode
3.172 +/- 0.092

avgNumQueue
1.703 +/- 0.085

utilization
0.735 +/- 0.005
```

$\lambda_{gadgetsArea} = \lambda * (p_{gadgetsArea_1} + p_{foodArea_1} * p_{gadgetsAfterFood\ _1}) = 7.166 * (0.1376 + 0.7465 * 0.1879) = 1.991\ job/min$

$\mu = 1.333\ job/min$

$$E[S_i] = \frac{1}{\mu} = 0.750\ min$$

$$m = 2$$

$$E[S] = \frac{E[S_i]}{\mathrm{m}} = \frac{1}{m*\mu} = 0.375\ min$$

$$\rho = \lambda_{gadgetsArea} * E[S] = 0.746$$

$$E[T_q] = \frac{P_q * E[S]}{1-\rho} = 0.946\ min$$

$$P_q = \frac{(m\rho)^m}{m!\,(1-\rho)} * \left(\sum_{i=0}^{m-1} \frac{(m\rho)^i}{i!} + \frac{(m\rho)^m}{m!\,(1-\rho)} \right)^{-1} = 0.638$$

$$E[T_s] = E[T_q] + E[S_i] = 1.697\ min$$

$$E[N_q] = \lambda_{gadgetsArea} * E[T_q] = 1.885$$

$$E[N] = \lambda_{gadgetsArea} * E[T_s] = 2.631$$

Consistency check:

- $E[T_s] = 1.644\ min$

  $E[T_q] + E[S_i] = 0.894 + 0.375 * 2 = 1.644\ min$ ✓

- $E[N_q] + \rho * m = 1.753 + 0.735 * 2 = 3.223$

  $E[N] = 3.224$ ✓

6. VALIDAZIONE

Effettuare una buona validazione necessiterebbe avere dei dati sufficientemente realistici: dal momento che non ci è stato possibile ottenere dati reali riguardo i vari tempi che si sperimentano all'interno del sistema reale, abbiamo deciso di svolgere il processo di validazione creando dei casi di test (consistency checks), utili al fine di valutare se l'andamento del sistema sia affine alle aspettative.
Questo step permette di valutare se il modello computazionale verificato è una buona approssimazione del sistema reale.
Fissate tre configurazioni (una per fascia oraria) e la coppia (b, k) = (1024, 128), ci aspettiamo che i tempi in coda in ogni centro diminuiscano al decresccere del tasso medio di arrivo, una situazione che verosimilmente si verificherebbe in un cinema reale.

6.2 Confronto tra le fasce orarie

Le tre tabelle seguenti riportano, rispettivamente, i tempi in coda per ciascun centro relativamente al tasso medio di arrivo considerato.
1° fascia: {2, 2, 1, 3, 2}

lambda	biglietteria	controlloBiglietti	cassaFoodArea	foodArea	gadgetsArea
7,166	0.886 +/- 0.042	0.061 +/- 0.002	1.352 +/- 0.144	1.136 +/- 0.149	0.867 +/- 0.041
6,166	0.538 +/- 0.021	0.041 +/- 0.002	0.524 +/- 0.028	0.406 +/- 0.026	0.475 +/- 0.021
5,166	0.345 +/- 0.011	0.027 +/- 0.001	0.294 +/- 0.010	0.182 +/- 0.010	0.266 +/- 0.011

2° fascia: {2, 1, 1, 3, 2}

lambda	biglietteria	controlloBiglietti	cassaFoodArea	foodArea	gadgetsArea
6,500	0.486 +/- 0.018	0.046 +/- 0.002	0.723 +/- 0.058	0.543 +/- 0.030	0.309 +/- 0.011
5,500	0.323 +/- 0.011	0.031 +/- 0.001	0.363 +/- 0.014	0.245 +/- 0.013	0.194 +/- 0.007
4,500	0.219 +/- 0.006	0.019 +/- 0.001	0.213 +/- 0.007	0.115 +/- 0.006	0.111 +/- 0.004

3° fascia: {2, 1, 1, 2, 1}

lambda	biglietteria	controlloBiglietti	cassaFoodArea	foodArea	gadgetsArea
5,166	0.314 +/- 0.010	0.027 +/- 0.001	0.198 +/- 0.005	0.105 +/- 0.005	0.083 +/- 0.003
4,166	0.206 +/- 0.005	0.017 +/- 0.001	0.132 +/- 0.004	0.053 +/- 0.003	0.049 +/- 0.002
3,166	0.135 +/- 0.003	0.009 +/- 0.000	0.083 +/- 0.002	0.022 +/- 0.001	0.025 +/- 0.001

Come si vede, i tempi in coda di ciascun centro diminuiscono al diminuire del tasso medio di arrivo, come ci si aspettava.

7. PROGETTAZIONE DEGLI ESPERIMENTI

La progettazione degli esperimenti si articola di 3 passi principali:ù

1. Calcolo delle configurazioni minime del sistema;
2. Simulazione ad orizzonte infinito;
3. Simulazione ad orizzonte finito.

7.1 Calcolo configurazioni minime

Il calcolo del numero minimo di serventi per ogni centro è necessario per garantire la condizione di stazionarietà del sistema. Avendo a disposizione i dati relativi al tasso medio di arrivo e di servizio di ogni centro, oltre che le varie probabilità di routing, abbiamo calcolato il numero necessario minimo di serventi per ogni centro seguendo la formula

$$\frac{\lambda}{m*\mu} < 1 \quad .$$

Questo calcolo è stato iterato per ognuna delle tre fasce orarie: la tabella seguente riporta il numero minimo di serventi per ogni coppia (centro, fascia oraria), utilizzando i valori di λ e μ dei rispettivi centri.

	biglietteria	**controlloBiglietti**	**cassaFoodArea**	**foodArea**	**gadgetsArea**
Fascia oraria 1	2	2	1	3	2
Fascia oraria 2	2	1	1	3	2
Fascia oraria 3	2	1	1	2	1

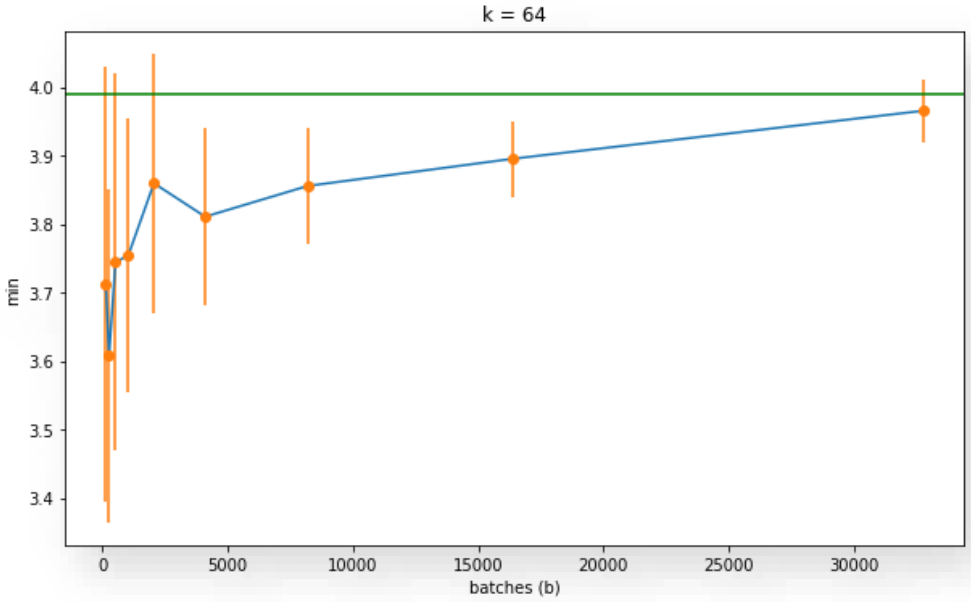
7.2 Simulazione ad orizzonte infinito

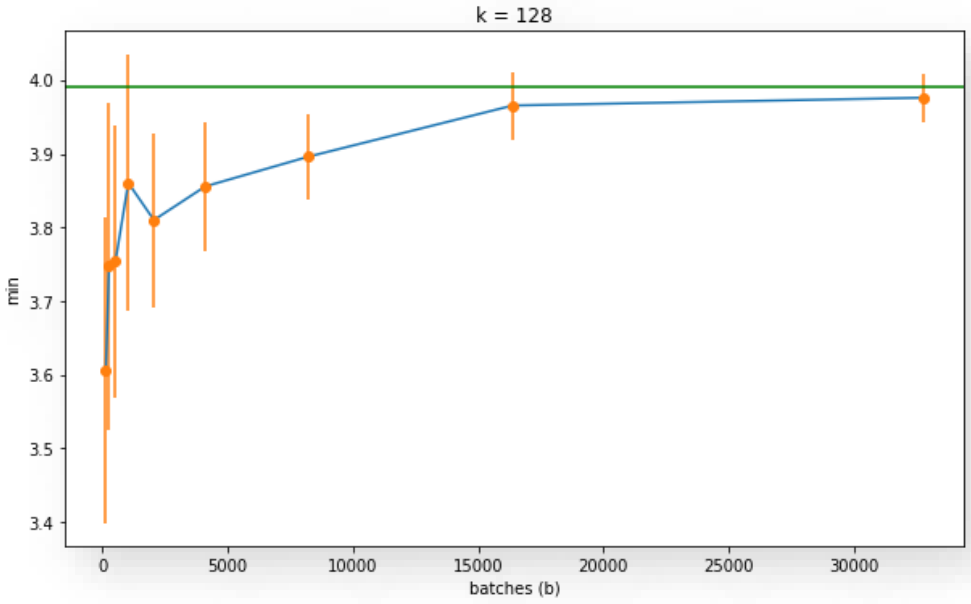
Nella simulazione ad orizzonte infinito il sistema viene simulato per un tempo 'infinito', ovvero considerando un arco temporale di gran lunga superiore alle tempistiche di una qualsiasi situazione reale in cui il sistema può operare.
Le statistiche ottenute da questo tipo di simulazione sono quelle dello stato stazionario del sistema. Effettuando una simulazione per un tempo molto più lungo di quello 'realistico' di una fascia oraria (circa un'ora), si riduce il bias dello stato iniziale, la cui scelta diventa appunto sempre meno rilevante al crescere del tempo di simulazione : 'il sistema perde memoria del proprio stato iniziale'.

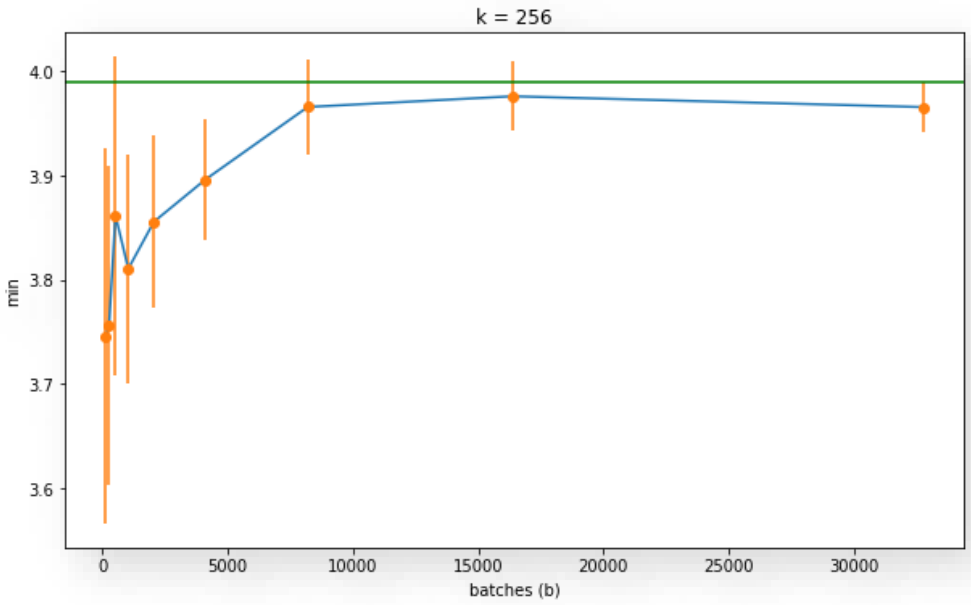
Per ottenere la media campionaria del tempo di risposta del sistema e di ogni singolo centro abbiamo utilizzato il metodo *Batch Means*, suddividendo la run di simulazione in *k* batches di size *b* (numero di job per ciascuna batch). È previsto che ogni centro del sistema raggiunga k batches, e la simulazione terminerà solamente quando questo sarà vero per tutti i centri.
Per ogni metrica di interesse viene calcolata una media per ciascuna batch e per ciascun centro; una volta raggiunto il numero indicato di batches, si termina la raccolta delle stime di un determinato centro e viene calcolato l'intervallo di confidenza proprio a partire dalle batch means computate.
In questo caso il bias dello stato iniziale viene eliminato, poichè le statistiche per ogni batch sono calcolate di volta in volta a partire da uno stato iniziale differente, ovvero come è stato 'lasciato' il sistema dal precedente batch; al termine di ciascuna batch le statistiche di ogni centro vengono salvate in strutture dati apposite, quali nel nostro caso una matrice di struct *output* che tiene conto del centro osservato e

della batch in questione, per essere poi resettate di modo da prepararle al calcolo delle statistiche della batch successiva.
Questo metodo riduce inoltre l'autocorrelazione fra i dati, dato che l'unico punto di contatto fra le varie osservazioni è che lo stato finale in cui si trova il sistema alla fine di una singola batch, è lo stato iniziale del sistema per la successiva.
La scelta dei parametri (b, k) è avvenuta seguendo quanto suggerito dal libro di testo, facendo riferimento a Batch Means come un algoritmo "black box": abbiamo fissato di volta in volta il numero di batch k = 64, 128, 256, e abbiamo testato valori di b duplicati di volta in volta fintanto che veniva raggiunta una "convergenza" accettabile.
Abbiamo inoltre tenuto in considerazione come limite b=32768, al fine di evitare tempi di esecuzione eccessivamente elevati dal punto di vista dell'utente.

Di seguito sono riportati i grafici che mostrano l'andamento del tempo di risposta dell'intero sistema al variare dei parametri (b,k). Abbiamo deciso di valutare la bontà della scelta di questi due parametri in base a questa metrica poiché è tra gli obiettivi principali dello studio quello di ridurre il tempo di risposta complessivo, pertanto una scelta dei parametri che ci garantisse un valore vicino a quello teorico ci è sembrata la miglior scelta.
In ordine, compaiono i grafici relativi ad una scelta di k (numero di batch) pari a 64, 128 e 256. Sull'asse delle ascisse compare la taglia delle batch (b), mentre la linea in verde riporta il valore teorico calcolato per il tempo di risposta del sistema (nel caso della prima fascia oraria, con configurazione {2,2,1,3,2}) : E(Ts) = 3.99 min.







La scelta finale è (b, k) = (8192, 256), poiché garantisce una buona acccuratezza sul valore del tempo di risposta del sistema, e allo stesso tempo mantiene i tempi di esecuzione non eccessivamente elevati.

7.3.1 Esperimenti

A partire dalle *configurazioni minime* del sistema per ogni fascia oraria, abbiamo utilizzato la simulazione ad orizzonte infinito al fine di determinare la configurazione ottima del sistema: in questo ambito, per configurazione ottima intendiamo quella che garantisce il soddisfacimento del QOS relativo al tempo di riposta.
La scelta della simulazione ad orizzonte infinito per questo fine deriva dal fatto che, sebbene in una situazione reale la finestra di tempo di operatività del sistema risulti relativamente breve, è in questo modo è possibile eliminare il bias dello stato iniziale del sistema. Realisticamente questo è legato al fatto per cui, nel momento in cui gli sportelli della biglietteria aprono, non è sempre vero che il sistema sia vuoto.
Dunque, gli esperimenti in questo ambito hanno l'obiettivo di testare varie configurazioni del sistema (modificando il numero di serventi presso i centri multiservente) così da determinarne la migliore.

7.3 Simulazione ad orizzonte finito

Per la simulazione ad orizzonte finito è stato considerato un tempo di simulazione limitato a 60 minuti (*close the door*), rispettando l'intervallo di tempo reale in cui c'è un flusso di arrivi al sistema.
Poiché le tre fasce orarie considerate non sono consecutive nel tempo, abbiamo deciso di effettuare ogni simulazione in riferimento ad una fascia oraria solamente (il sistema simulato permette di scegliere in quale fascia oraria svolgere la simulazione in quanto cambieranno diversi parametri, tra cui il tasso medio di arrivo e le probabilità di routing).
Per la simulazione ad orizzonte finito la scelta dello stato iniziale del sistema ha un impatto importante sulle statistiche: nel nostro caso abbiamo fatto in modo che il sistema fosse *idle* sia all'inizio che alla fine della simulazione. Allo scopo di effettuare un'analisi statistica dei valori ottenuti abbiamo operato replicando la simulazione per un numero pari a 128 volte, cambiando soltanto lo stato iniziale del generatore di numeri pseudocasuali. Ogni *replicazione* è volta a misurare indipendentemente le stesse metriche, fornendo un punto di stima (*datapoint*) utile al fine di calcolare gli intervalli di confidenza per ognuna di esse. Dunque, *l'INITIAL SEED deve* essere scelto in modo tale da non avere nessuna sovrapposizione fra repliche nella sequenza di numeri random utilizzata: a tale scopo, abbiamo utilizzato la tecnica standard che prevede di utilizzare lo stato finale di ogni stream del generatore per una replica come stato iniziale per la prossima. Questo è ottenuto effettuando la chiamata alla funzione *PlantSeeds(SEED)* una sola volta al di fuori del ciclo di replicazione, scegliendo come *SEED* il valore 123456789. Per memorizzare i dati relativi alle varie repliche abbiamo utilizzato la struct *outputStats:* nello specifico, abbiamo creato una struct di questo tipo per ogni centro del sistema e per ogni replica, andando successivamente a memorizzarla all'interno di una matrice di *outputStats* (*matrix*), in cui la riga i-ima corrisponde alla replicazione i-esima e la colonna j-ima corrisponde al centro j-imo.
Dopo aver eseguito la totalità delle repliche, la matrice è stata utilizzata – con l'ausilio del programma *estimate.c* – per calcolare gli intervalli di confidenza per ogni centro e per ogni statistica di interesse, oltre che per il tempo di risposta del sistema.

7.4.1 Esperimenti

Poiché le configurazioni ottenute dalla simulazione precedente prendono in considerazione solamente il QoS relativo al tempo di risposta, non è detto che siano quelle che forniscono il maggior profitto da parte del cinema. Dunque, considerando il reale tempo di operatività della biglietteria pari a 60 minuti, consideriamo la variabile legata al profitto del cinema: in particolare, dal momento che i guadagni relativi alla proiezione di spot cinematografici sono direttamente proporzionali al numero di persone che assistono alla proiezione, potrebbe essere vantaggioso modificare la configurazione del sistema in questo senso.

8. ESECUZIONE DELLE SIMULAZIONI

8.1 Esperimenti per QoS1

Una volta terminata la progettazione degli esperimenti siamo passati all'esecuzione delle simulazioni, in modo da svolgere gli esperimenti che ci siamo prefissati di fare. Nello specifico, abbiamo iniziato svolgendo gli esperimenti necessari a soddisfare il QoS1, analizzando ogni fascia oraria di volta in volta.

8.1.1 Fascia oraria 1

A partire dalla configurazione minima del sistema – ovvero {2,2,1,3,2} – effettuando una run di simulazione ad orizzonte infinito otteniamo un valore del tempo di risposta complessivo pari a

$$E[T_s] = 3.965638 \pm 0.045763\ min.$$

Poiché questa configurazione rispetta immediatamente il primo QoS possiamo considerarla ottimale.

8.1.1 Fascia oraria 2

A partire dalla configurazione minima {2,1,1,3,2} – effettuando una run di simulazione ad orizzonte infinito otteniamo un valore del tempo di risposta complessivo pari a

$$E[T_s] = 8.538984 \pm 0.474472\ min.$$

Poiché questa configurazione non rispetta il primo QoS, superando di gran lunga il valore desiderato per il tempo di risposta, aumentiamo il numero di serventi presso i centri multiservente in maniera incrementale, partendo dal centro *controlloBiglietti*, poiché presenta l'utilizzazione maggiore.

Configurazione	$\boldsymbol{E[T_s]}$
{2,2,1,3,2}	$2.487101 \pm 0.15123\ min$

La configurazione appena trovata ci permette di mantenere accettabilmente il tempo di risposta del sistema sotto i 3 minuti, soddisfacendo dunque il QoS1.

8.1.1 Fascia oraria 3

A partire dalla configurazione minima {2,1,1,2,1} , effettuando una run di simulazione ad orizzonte infinito otteniamo un valore del tempo di risposta complessivo pari a

$$E[T_s] = 1.487106 \pm 0.002452\ min.$$

Poiché questa configurazione rispetta anch'essa il primo QoS, non è necessario incrementare il numero di serventi presso i centri multiservente.

Una volta ottenute le configurazioni ottimali per ciascuna fascia, queste saranno usate come punto di partenza per gli esperimenti successivi.

8.2 Esperimenti per QoS2 e profitto

Ricaviamo il profitto del cinema con la seguente equazione:

$$profitto = income - expense = (biglietti + food + pubblicità) - (stipendi + spesaFood)$$

8.2.1 Fascia oraria 1

A partire dalla configurazione ottenuta in precedenza {2,2,1,3,2}, effettuando una run di simulazione ad orizzonte *finito* otteniamo un valore percentuale del numero complessivo di utenti che sono entrati in sala (hanno terminato il percorso nel sistema) entro 15 minuti dall'inizio del film:

$$\% = 69.152 \pm 0.388\ min.$$

Poiché questa configurazione non rispetta il secondo QoS, incrementiamo il numero di serventi presso i centri multiservente.
Aggiungendo un solo servente per volta otteniamo le tre diverse configurazioni {2,3,1,3,2}, {2,2,1,4,2}, {2,2,1,3,3}. I valori ottenuti sono i seguenti:

Configurazione	% spettatori pubblicità
{2,3,1,3,2}	$68.889 \pm 0.346\ min$
{2,2,1,4,2}	$70.385 \pm 0.385\ min$
{2,2,1,3,3}	$69.533 \pm 0.394\ min$

Risulta che la seconda configurazione, che va ad incrementare i serventi del centro con maggior utilizzazione in questo caso (foodArea), riporta una percentuale sufficiente a soddisfare il QoS 2.
Per la prima fascia oraria è dunque previsto che il cinema abbia 11 dipendenti in servizio, distribuiti nei vari centri secondo {2,2,1,4,2}; la spesa degli stipendi risulta quindi:

$$\boldsymbol{stipendi} = 3h * 7\,{}^{\$}/_{h} * 11dip = 231\ €$$

Il guadagno derivante dalle pubblicità è invece:

$$\boldsymbol{pubblicità} = guadagnoASpettatore^{1} * (totalArrivals^{2} * \%spettatoriPubblicità)$$
$$= 2.155€ * (407 * 0.70385) \cong 617.33€$$

.[1] $guadagnoASpettatore$ = 2.155€ , come ricavato a pagina 13;
.[2] $totalArrivals \cong 407$,

```
Total arrivals number based upon 256 data points and with 95% confidence
the expected value is in the interval 406.746 +/- 2.726
```

Proseguiamo sperimentando se un incremento nel numero di serventi nei centri multiservente può portare ad un ulteriore aumento nella percentale di spettatori delle pubblicità conveniente per il profitto.

Configurazione	% spettatori pubblicità
{2,3,1,4,2}	$70.311 \pm 0.343\ min$
{2,2,1,5,2}	$70.681 \pm 0.385\ min$
{2,2,1,4,3}	$70.958 \pm 0.353\ min$

Con la terza configurazione abbiamo un guadagno derivante dalla pubblicità di:

$$\boldsymbol{pubblicit}\text{à} = 2.155€ * (407 * 0.70958) \cong 622.36€$$

Dunque un aumento di solamente $622.36€ - 617.33€ = 5.03€$ a fronte dei 21€ spesi per lo stipendio del nuovo dipendente.

In conclusione risulta conveniente che per la prima fascia si adotti la configurazione {2,2,1,4,2}, che ci garantisce il tempo medio di risposta del sistema nel transiente pari a

$$E[T_s] = 2.581267 \text{ +/- } 0.077164.$$

Il profitto finale che si ottiene è dunque di:

$$\begin{aligned} profitto &= (biglietti + food + pubblicità) - (stipendi + spesaFood^{3}) \\ &= (7€ * totalArrivals + (2.5€ + 3.5€) * (totalArrivals * 0.7465) + 617.33€) \\ &- (231€ + 250€) = 4808.28€ \end{aligned}$$

.[3] $spesaFood$ = 250 € , come ricavato a pagina 9;

8.2.2 Fascia oraria 2

A partire dalla configurazione ottenuta in precedenza {2,2,1,3,2}, effettuando una run di simulazione ad orizzonte *finito* otteniamo un valore percentuale del numero complessivo di utenti che sono entrati in sala (hanno terminato il percorso nel sistema) entro 15 minuti dall'inizio del film: $\% = 70.921 \pm 0.380\ min$. Pertanto, il QoS 2 risulta così soddisfatto.

Per la seconda fascia oraria è dunque previsto che il cinema abbia 10 dipendenti in servizio, distribuiti nei vari centri secondo {2,2,1,3,2}; la spesa degli stipendi risulta quindi:

$$\boldsymbol{stipendi} = 3h * 7\,{}^{\$}/_{h} * 10dip = 210\ €$$

Il guadagno derivante dalle pubblicità è invece:

$$\begin{aligned} \boldsymbol{pubblicit}\text{à} &= guadagnoASpettatore^{1} * (totalArrivals^{2} * \%spettatoriPubblicità) \\ &= 2.155€ * (370 * 0.70921) \cong 565.48€ \end{aligned}$$

.[1] $guadagnoASpettatore$ = 2.155€ , come ricavato a pagina 13;
.[2] $totalArrivals \cong 370$,

```
Total arrivals number based upon 256 data points and with 95% confidence
the expected value is in the interval 369.449 +/- 2.860
```

Proseguiamo sperimentando se un incremento nel numero di serventi nei centri multiservente può portare ad un ulteriore aumento nella percentale di spettatori delle pubblicità conveniente per il profitto.

Configurazione	% spettatori pubblicità
{2,3,1,3,2}	$70.989 \pm 0.398\ min$
{2,2,1,4,2}	$71.631 \pm 0.371\ min$
{2,2,1,3,3}	$70.985 \pm 0.353\ min$

Con la seconda configurazione abbiamo un guadagno derivante dalla pubblicità di:

$$\boldsymbol{pubblicit}\text{à} = 2.155€ * (370 * 0.71631) \cong 571.14€$$

Dunque un aumento di solamente $571.14€ - 565.48€ = 5.66€$ a fronte dei 21€ spesi per lo stipendio del nuovo dipendente.

In conclusione risulta conveniente che per la seconda fascia si adotti la configurazione {2,2,1,3,2}, che ci garantisce il tempo medio di risposta del sistema nel transiente pari a

$$E[T_s] = 2.299371 \text{ +/- } 0.063107.$$

Il profitto finale è di:

$$\begin{aligned} profitto &= (biglietti + food + pubblicità) - (stipendi + spesaFood^{3}) \\ &= (7€ * totalArrivals + (2.5€ + 3.5€) * (totalArrivals * 0.7512) + 565.48€) \\ &- (231€ + 250€) = 4342.14€ \end{aligned}$$

.[3] $spesaFood$ = 250 € , come ricavato a pagina 9;

8.2.3 Fascia oraria 3

A partire dalla configurazione ottenuta in precedenza {2,1,1,2,1}, effettuando una run di simulazione ad orizzonte *finito* otteniamo un valore percentuale del numero complessivo di utenti che sono entrati in sala (hanno terminato il percorso nel sistema) entro 15 minuti dall'inizio del film: $\% = 73.212 \pm 0.532\ min.$
Pertanto, il QoS 2 risulta così soddisfatto.

Per la terza fascia oraria è dunque previsto che il cinema abbia 7 dipendenti in servizio, distribuiti nei vari centri secondo {2,1,1,2,1}; la spesa degli stipendi risulta quindi:

$$\boldsymbol{stipendi} = 3h * 7\,{}^{\$}/_{h} * 7dip = 147\ €$$

Il guadagno derivante dalle pubblicità è invece:

$$\begin{aligned} \boldsymbol{pubblicit}\text{à} &= guadagnoASpettatore^{1} * (totalArrivals^{2} * \%spettatoriPubblicità) \\ &= 2.155€ * (148 * 0.73212) \cong 233.50\ € \end{aligned}$$

.[1] $guadagnoASpettatore$ = 2.155€ , come ricavato a pagina 13;
.[2] $totalArrivals \cong 148$,

```
Total arrivals number based upon 256 data points and with 95% confidence
the expected value is in the interval 147.434 +/- 1.910
```

Proseguiamo sperimentando se un incremento nel numero di serventi nei centri multiservente può portare ad un ulteriore aumento nella percentale di spettatori delle pubblicità conveniente per il profitto.

Configurazione	% spettatori pubblicità
{2,2,1,2,1}	$73.353 \pm 0.545\ min$
{2,1,1,3,1}	$73.305 \pm 0.536\ min$
{2,1,1,2,2}	$73.223 \pm 0.571\ min$

Con la seconda configurazione abbiamo un guadagno derivante dalla pubblicità di:

$$\boldsymbol{pubblicit}\text{à} = 2.155€ * (148 * 0.73353) \cong 233.95€$$

Dunque un aumento di appena $233.95€ - 233.50\ € = 0.45\ €$ a fronte dei 21€ spesi per lo stipendio del nuovo dipendente.

In conclusione risulta conveniente che per la terza fascia si adotti la configurazione {2,2,1,2,1}, che ci garantisce il tempo medio di risposta del sistema nel transiente pari a

$$E[T_s] = 1.353185 \text{ +/- } 0.018130.$$

Il profitto finale è di:

$$\begin{aligned} profitto &= (biglietti + food + pubblicit\text{à}) - (stipendi + spesaFood^{3}) \\ &= (7€ * totalArrivals + (2.5€ + 3.5€) * (totalArrivals * 0.6354) + 233.50€) \\ &\quad - (231€ + 250€) = 1352.73€ \end{aligned}$$

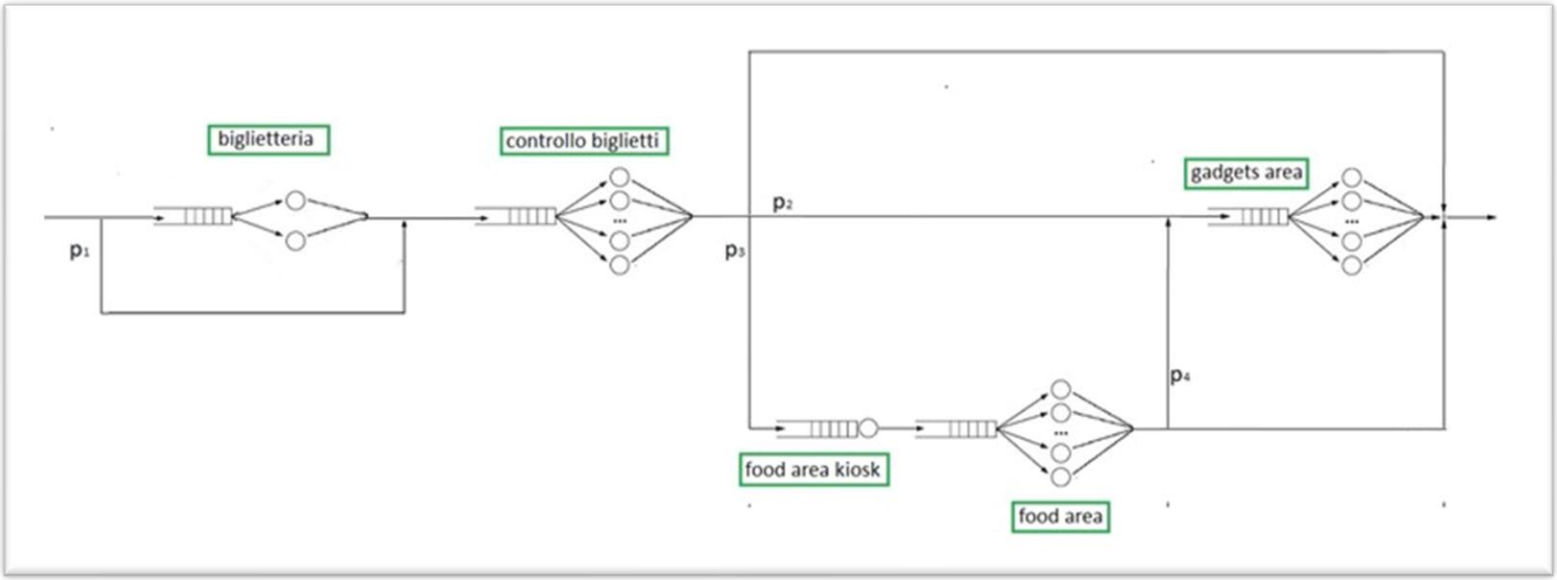
.[3] $spesaFood$ = 250 € , come ricavato a pagina 9;

9. MODELLO MIGLIORATIVO

9.1 Descrizione

Il modello migliorativo ideato lascia il sistema originario pressochè invariato, fatta eccezione per il centro *biglietteria* : il miglioramento che abbiamo pensato è stato quello di modificare la configurazione di questo centro, eliminando la divisione in due code e due serventi, rendendo di fatto la biglietteria un centro multiservente con coda singola e due serventi.
In questo caso, l'obiettivo è quello di valutare se l'introduzione di questo cambiamento porta ad avere un miglioramento nelle prestazioni del sistema. In particolare, ci curiamo di valutare se i due QoS vengono soddisfatti in maniera più consistente, così da assicurare un profitto maggiore da parte del cinema.

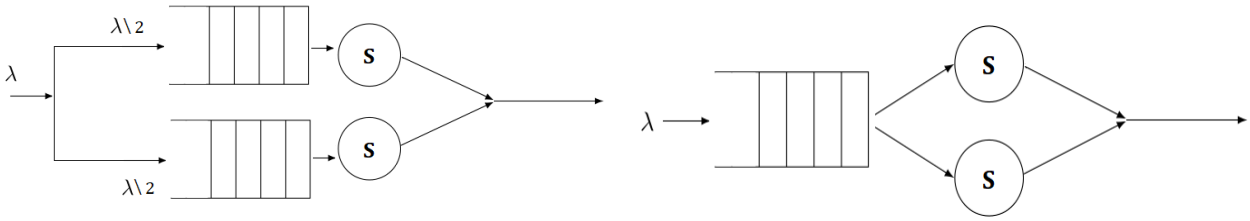
Per quanto riguarda il modello concettuale il simulatore rimane molto simile al modello precedentemente sviluppato. La rete di code che modella il nuovo sistema è il seguente riportato in figura:



Per quanto riguarda, invece, il modello delle specifiche il sistema rimane completamente invariato rispetto al precedente; a livello computazionale abbiamo trasformato il centro biglietteria in un multiserver dual-core, con la stessa gestione implementata per gli altri centri multiservente.

9.2 Verifica

Una volta finito di sviluppare il modello computazionale relativo a questo nuovo sistema è risultato necessario effettuare di nuovo una verifica per il centro *biglietteria*, dal momento che i rimanenti centri restano invariati. I seguenti valori derivano dalla prima fascia oraria.



A) B)

$$\rho = \frac{\lambda}{2} * S \qquad \rho = \frac{\lambda}{2} * S$$

$$E[T_s] = E[S] + E[T_q] = S + \frac{\rho * S}{1-\rho} \quad > \quad E[T_s] = E[S_i] + E[T_q] = S + \frac{Pq * S}{1-\rho}$$

$$\frac{\rho * S}{1-\rho} > \frac{Pq * S}{1-\rho}$$

$$\rho > Pq$$

```
biglietteria
Statistics based upon 256 data points.
With 95% confidence,the expected values are in the intervals:

avgInterarrivalTime
0.193 +/- 0.000

avgArrivalRate
5.193 +/- 0.007

avgWait
1.202 +/- 0.012

avgDelay
0.910 +/- 0.012

avgServiceTime
0.292 +/- 0.000

avgNumNode
6.244 +/- 0.067

avgNumQueue
4.729 +/- 0.065

utilization
0.757 +/- 0.001
```

```
Cinema response time based upon 256 data points and with 95% confidence
the expected value is in the interval 3.172095 +/- 0.027106
```

```
biglietteria
Statistics based upon 256 data points.
With 95% confidence,the expected values are in the intervals:

avgInterarrivalTime
0.193 +/- 0.000

avgArrivalRate
5.193 +/- 0.007

avgWait
0.684 +/- 0.006

avgDelay
0.392 +/- 0.006

avgServiceTime
0.146 +/- 0.000

avgNumNode
3.554 +/- 0.034

avgNumQueue
2.039 +/- 0.031

utilization
0.758 +/- 0.002
```

```
Cinema response time based upon 256 data points and with 95% confidence
the expected value is in the interval 1.721858 +/- 0.005163
```

E' infatti possibile notare come l'utilizzazione sia invariata fra i due casi, stesso vale ovviamente per l'average arrival rate, mentre l'average delay, e dunque anche l'average wait, risulta quasi dimezzato.
Notiamo anche che l'average service time E[S]=S nel caso A), mentre E[S]=E[Si]/N nel caso B).

$$\lambda_{biglietteria} = \lambda * (1 - p_{online}) = 7.166 * 0.7245 = 5.191\ job/min$$

$$\mu = 3.428\ job/min$$

$$E[S_i] = \frac{1}{\mu} = 0.291\ min$$

$$N = 2$$

$$E[S] = \frac{E[S_i]}{\mathrm{m}} = \frac{1}{m*\mu} = 0.145\ min$$

$$\rho = \lambda_{biglietteria} * E[S] = 0.752$$

$$E[T_q] = \frac{P_q * E[S]}{1 - \rho} = 0.381\ min$$

$$P_q = \frac{(m\rho)^m}{m!\,(1-\rho)} * \left(\sum_{i=0}^{m-1} \frac{(m\rho)^i}{i!} + \frac{(m\rho)^m}{m!\,(1-\rho)}\right)^{-1} = \frac{(2*\rho)^2}{2!\,(1-\rho)} * \left(\sum_{i=0}^{1} \frac{(2\rho)^i}{i!} + \frac{(2\rho)^2}{2!\,(1-\rho)}\right)^{-1}$$
$$= 0.6526$$

$$E[T_s] = E[T_q] + E[S_i] = 0.672\ min$$

$$E[N_q] = \lambda_{biglietteria} * E[T_q] = 1.977$$

$$E[N] = \lambda_{biglietteria} * E[T_s] = 3.488$$

Consistency check:

- $E[T_s] = 0.684$

  $E[T_q] + E[S_i] = 0.392 + 0.146 * 2 = 0.684$ ✓

- $E[N_q] + \rho * m = 2.039 + 0.758 * 2 = 3.555$

  $E[N] = 3.554$ ✓

9.3 Risultati

A partire dalla stessa configurazione iniziale considerata per la prima fascia, {2,2,1,3,2}, è possibile soddisfare largamente il primo QoS, e più sufficientemente il secondo, ottenendo il seguente profitto del cinema:

$$profitto = (biglietti + food + pubblicità) - (stipendi + spesaFood^{3})$$
$$= (7€ * totalArrivals + (2.5€ + 3.5€) * (totalArrivals * 0.7465) + 627.54€)$$
$$- (210€ + 250€) = 4839.49€$$

$$pubblicità = guadagnoASpettatore^{1} * (totalArrivals^{2} * \%SpettatoriPubblicità^{4})$$
$$= 2.155€ * (407 * 0.71549) \cong 627.54\ €$$
$$stipendi = 3h * 7\,{}^{\$}/_{h} * 10\ dip = 210\ €$$

.[1] $guadagnoASpettatore$ = 2.155€ , come ricavato a pagina 13;

.[2] $totalArrivals \cong 407$;

.[3] $spesaFood$ = 250 € , come ricavato a pagina 9;

.[4] $\%SpettatoriPubblicità$, `Percentage of people inside for pubblicity based upon 256 data points and with 95% confidence the expected value is in the interval 71.549 +/- 0.506`;

Link repository GitHub :
GiulioAppetito/Progetto-PMCSN: Progetto PMCSN a.a.2021/22 (github.com)