Università degli Studi di Ferrara
Corso di Laurea in Ingegneria Elettronica

Simulatore di Sistemi a Coda di tipo M/M/1, M/M/1/Y, M/D/1, M/D/1/Y

Tesina di Reti di Telecomunicazioni
di
Tarin Gamberini

Corso di Reti di Telecomunicazioni (ante riforma 3+2)
Anno Accademico 2002/2003
Docente *G. Mazzini*

taringamberini [at] taringamberini [dot] com
www.taringamberini.com

Indice

Elenco delle tabelle

Elenco delle figure

Introduzione

La commutazione di pacchetto è una metodologia di consegna dell'informazione che presuppone una visione del cammino fra trasmettitore e ricevitore formata da una rete di nodi. Ogni nodo, a seconda delle situazioni, può comportarsi da trasmettitore, da ricevitore o da ripetitore. Per svolgere quest'ultima funzione il nodo deve essere dotato di un sistema di code, atto a memorizzare i pacchetti ad esso inviati, prima di instradarli verso le opportune destinazioni (store and forward).

L'analisi di sistemi a coda è basata sulla creazione di un modello teorico che, sulla base di opportune ipotesi esemplificative, permette di determinare analiticamente le grandezze da osservare.

Un simulatore del sistema, programmato al calcolatore, diviene uno strumento fondamentale per verificare quanto le ipotesi esemplificative abbiano portato a soluzioni teoriche accettabili, intese come *confrontabili* con la realtà riprodotta dal simulatore.

Il linguaggio C offre primitive per scopi statistici troppo semplici, per cui la generazione di una variabile casuale distribuita secondo Poisson è stata calcolata adottando la tecnica di trasformazione inversa, riportata in appendice A.

Il simulatore è stato scritto in linguaggio C all'interno dell'ambiente di sviluppo Borland C++ Builder. Il codice sorgente è disponibile in appendice B.

Le simulazioni sono state condotte su di un Pentium 4 a 1.5 GHz che in circa 20 sec. ha permesso di valutare l'evoluzione dinamica della coda all'arrivo di 10^6 utenti.

Capitolo 1

Diagramma di flusso del simulatore

Il principio di funzionamento del simulatore è schematizzato in figura 1.1.

Il programma inizia generando l'arrivo di un utente il quale trovando il sistema vuoto viene immediatamente servito.

Inizia ora il ciclo più esterno che verifica se sono stati generati tutti gli arrivi definiti nella costante NUTENTI . In caso affermativo la simulazione termina, diversamente si procede.

Ci si chiede se l'utente precedentemente generato P è già arrivato. In caso affermativo se ne genera uno nuovo N, altrimenti si procede avvicinandosi all'istante in cui l'utente P arriverà.

Ora l'utente precedentemente generato P, od il nuovo appena generato N, se trovano il sistema vuoto vengono immediatamente serviti, in caso contrario il sistema è occupato, nel senso che l'utente S sta ancora fruendo del sevizio. In quest'ultimo caso se l'arrivo di P, o di N, precede l'istante di uscita dal sevizio di S occorrerà accodare l'utente arrivato.

Se l'arrivo di P, o di N, segue l'istante di uscita dal sevizio di S, si serve l'eventuale utente C che sta attendendo in coda. Il destino di P, o di N, verrà deciso nei cicli successivi in funzione del loro istante di arrivo e di ciò che accadrà all'utente C entrato in servizio.

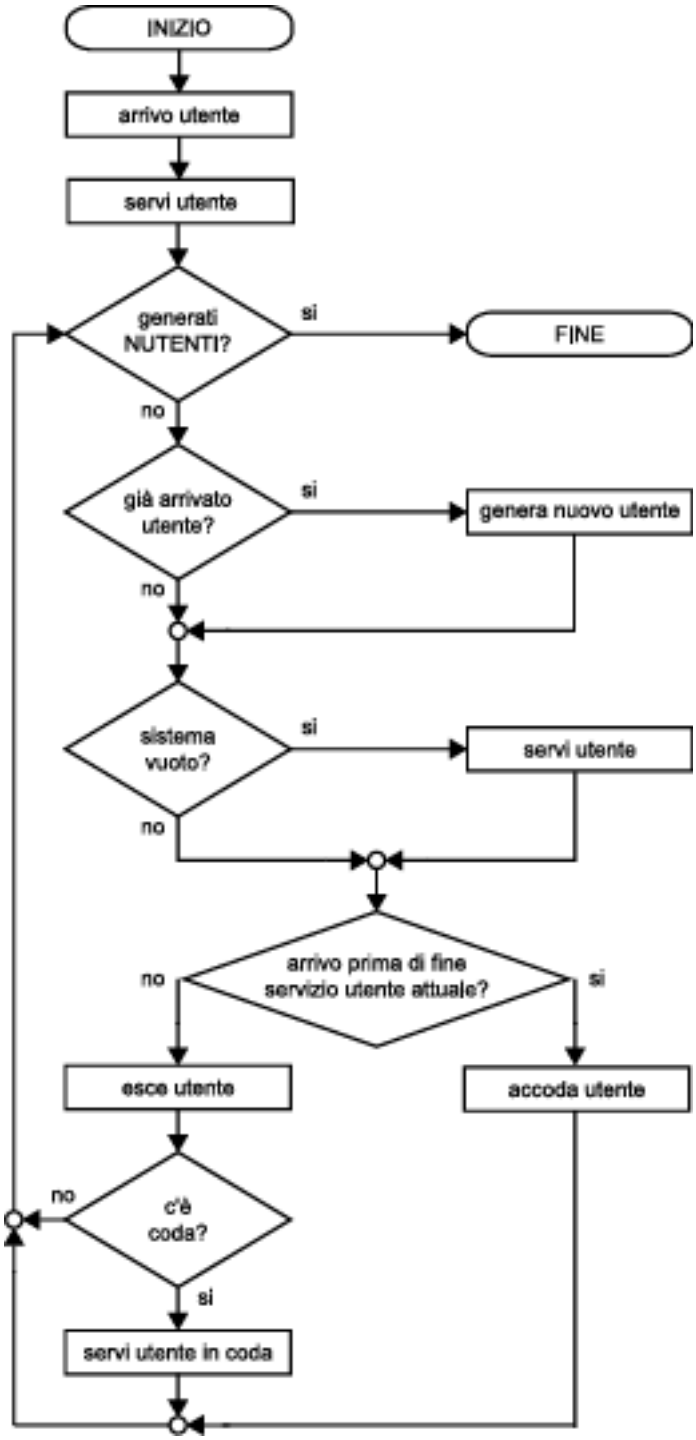

Figura 1.1: Diagramma di flusso del simulatore

Capitolo 2

Errori fra teoria e simulazioni

Durante il corso abbiamo sviluppato un modello matematico di carattere generale che, attraverso alcune ipotesi esemplificative, è stato possibile risolvere analiticamente.

Il simulatore conferma i risultati teorici con una certa approssimazione, strettamente legata al numero di utenti NUTENTI impiegati per determinare i valori medi delle grandezze di interesse.

Relativamente al modello M/M/1 abbiamo voluto renderci conto del livello di approssimazione del simulatore per verificare quanto le ipotesi esemplificative introdotte nel modello teorico fossero accettabili.

Con un numero di utenti NUTENTI $= 10^4$ abbiamo ottenuto gli andamenti delle figure 2.1, 2.2, 2.3, 2.4 e 2.5. relativi rispettivamente al tempo medio di attesa in coda $E[Tq]$, al tempo medio di permanenza nel sistema $E[T]$, al tempo medio di servizio $E[x]$, al numero medio di utenti in coda $E[q]$ ed infine al numero medio di utenti nel sistema $E[k]$.

Le discrepanze fra gli andamenti simulati e quelli teorici diventano sempre meno evidenti al crescere del numero di utenti come mostrato nelle figure 2.6, 2.7, 2.8, 2.9, 2.10.

Anche una valutazione degli errori relativi percentuali, calcolati come:

$$Er_{rel} = \frac{|valoreteorico - valorepratico|}{valoreteorico} * 100$$

conferma l'aumento della precisione raggiunta al crescere del numero di utenti come appare dalle figure 2.11, 2.12, 2.13, 2.14, 2.15, 2.16.

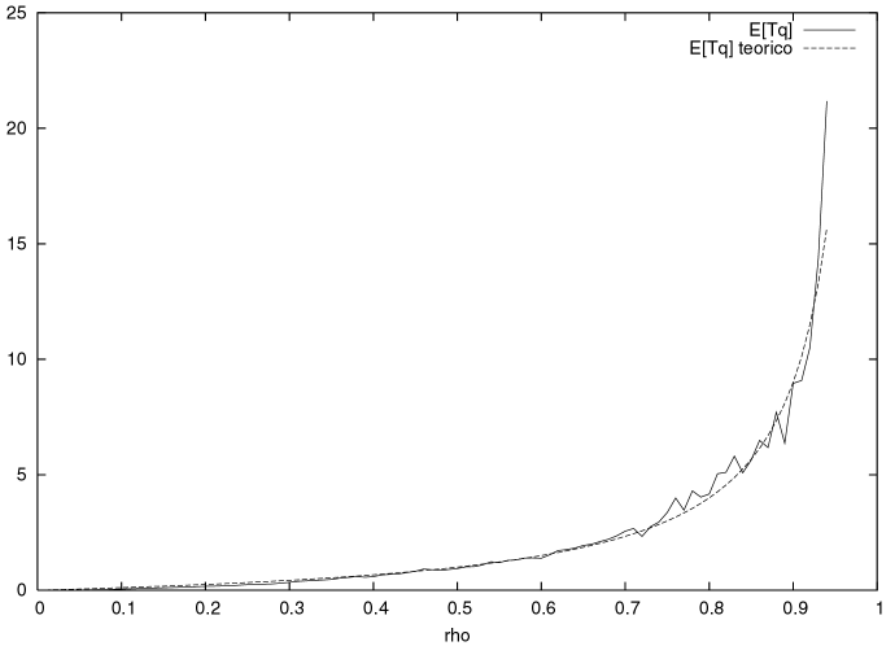

Figura 2.1: Tempo medio di attesa in coda $E[Tq]$ con NUTENTI $= 10^4$

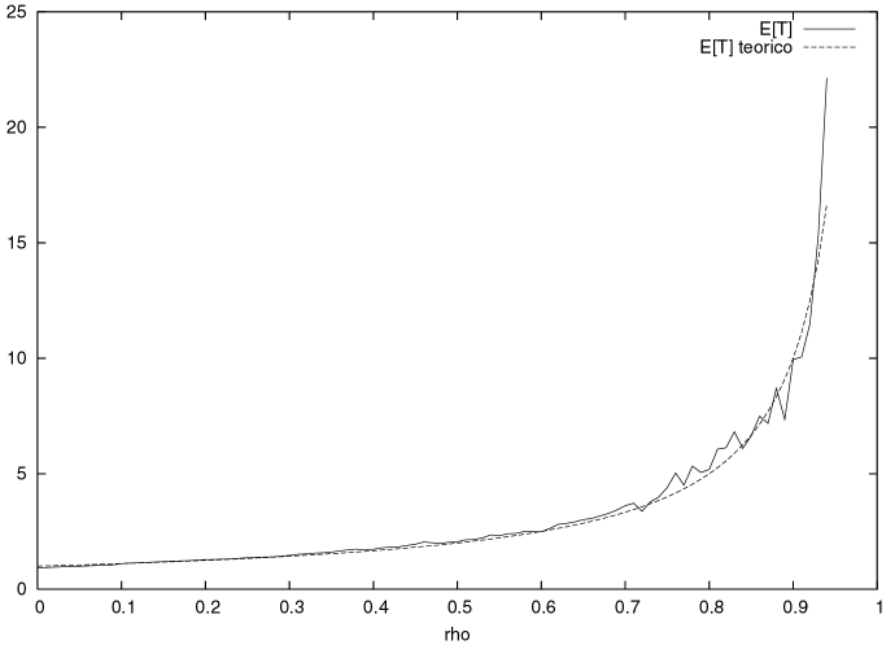

Figura 2.2: Tempo medio trascorso nel sistema $E[T]$ con NUTENTI $= 10^4$

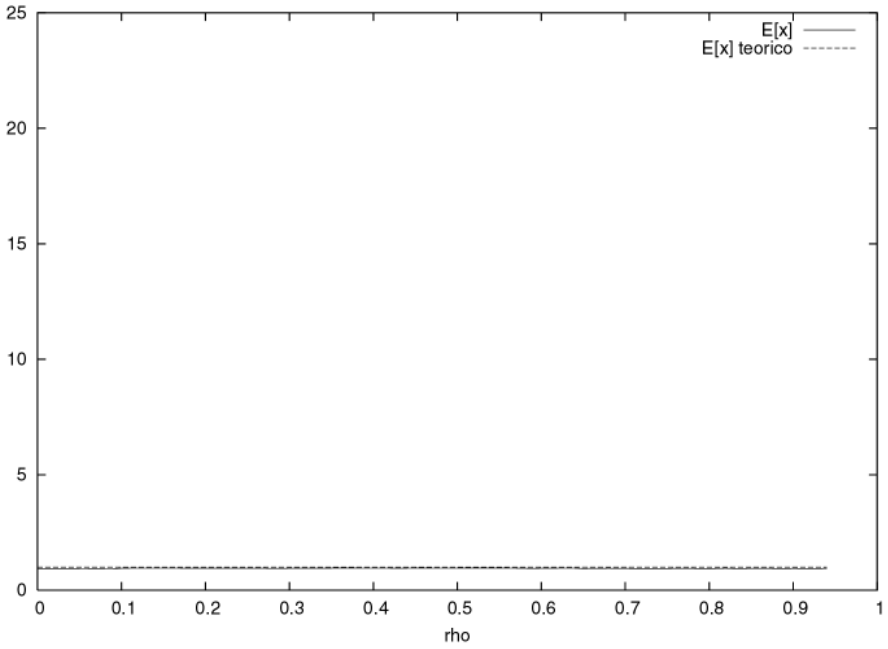

Figura 2.3: Tempo medio di servizio $E[x]$ con NUTENTI $= 10^4$

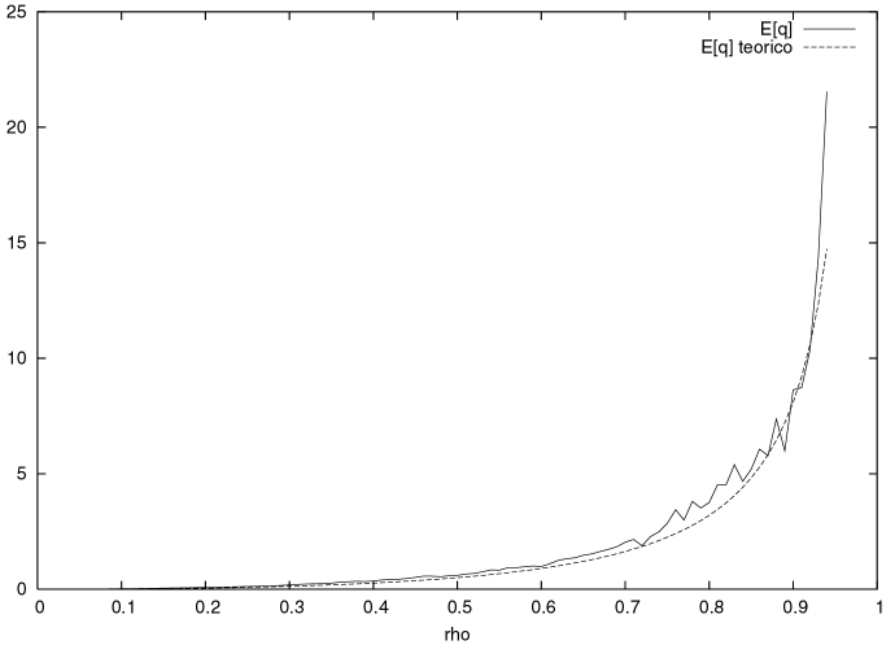

Figura 2.4: Numero medio di utenti in coda $E[q]$ con NUTENTI $= 10^4$

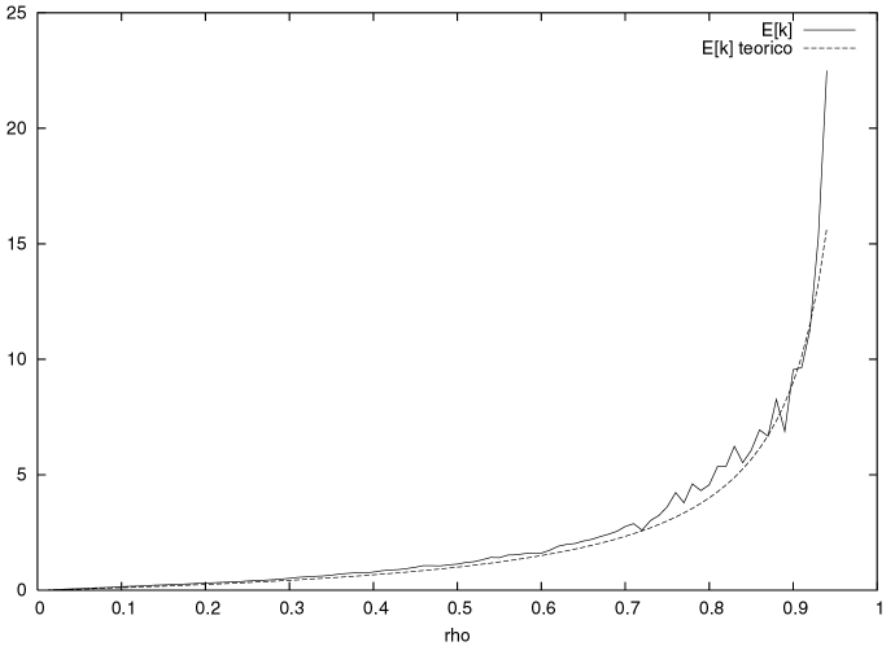

Figura 2.5: Numero medio di utenti nel sistema $E[k]$ con NUTENTI $= 10^4$

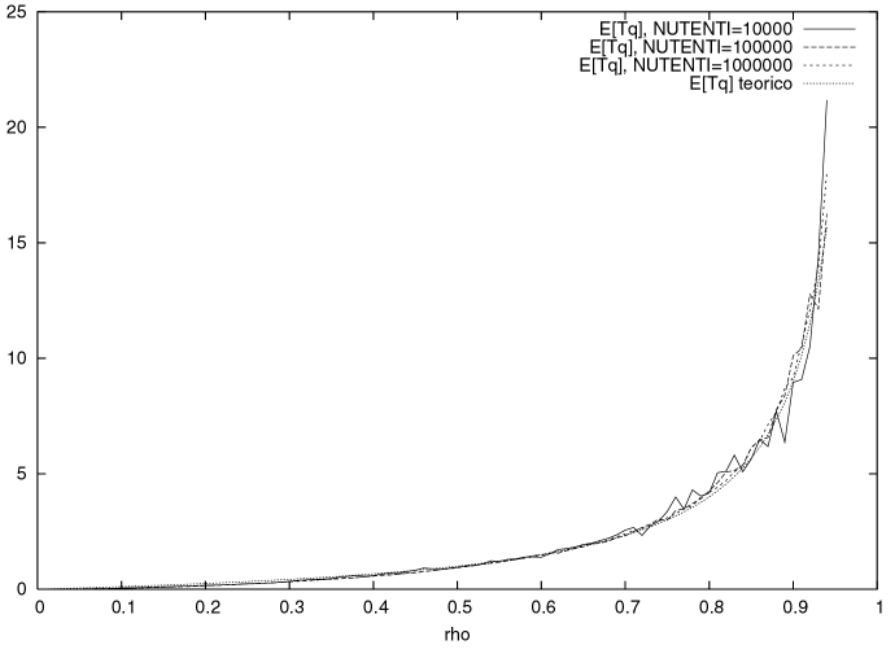

Figura 2.6: $E[Tq]$ con NUTENTI $= 10^4, 10^5, 10^6$

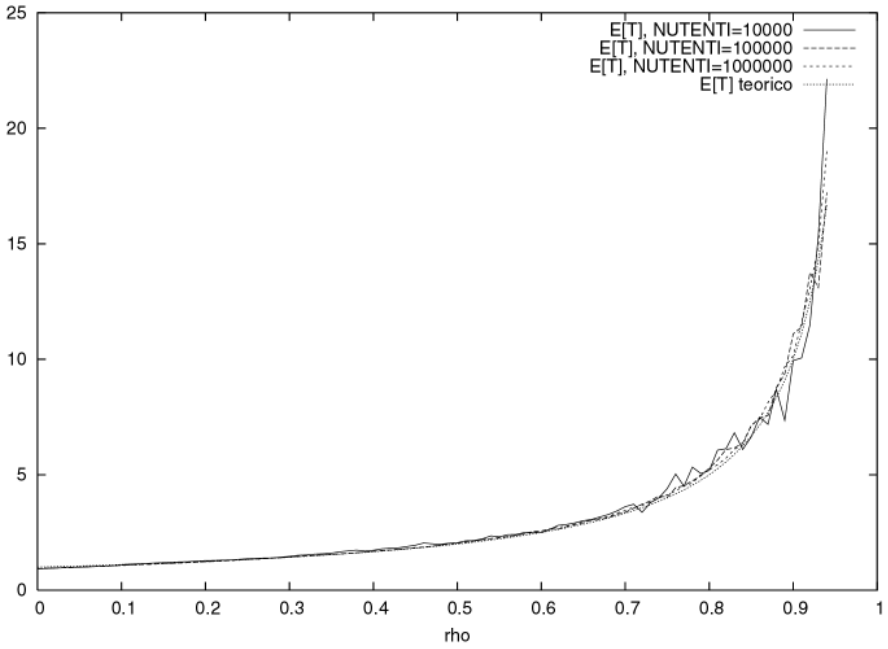

Figura 2.7: $E[T]$ con NUTENTI $= 10^4, 10^5, 10^6$

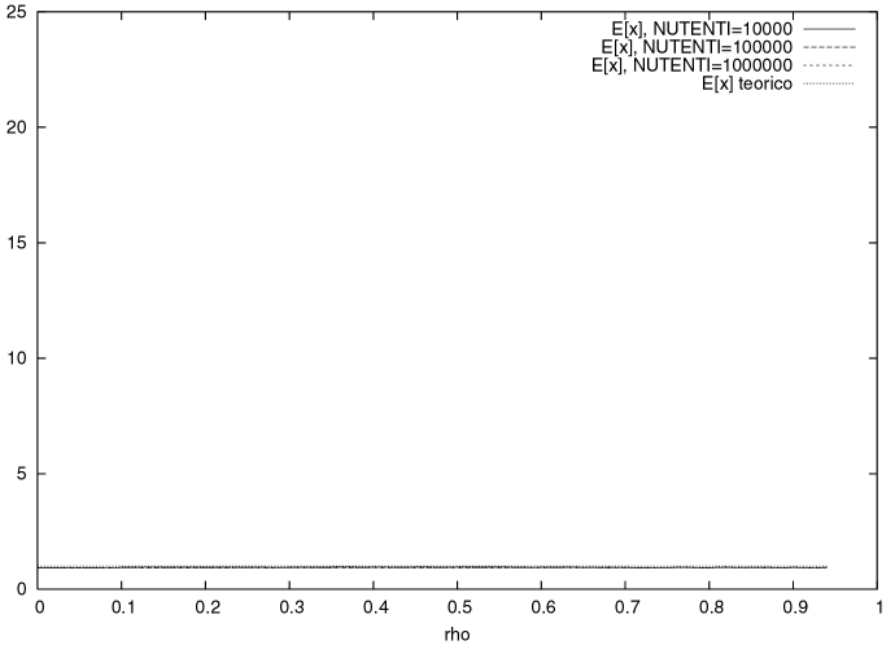

Figura 2.8: $E[x]$ con NUTENTI $= 10^4, 10^5, 10^6$

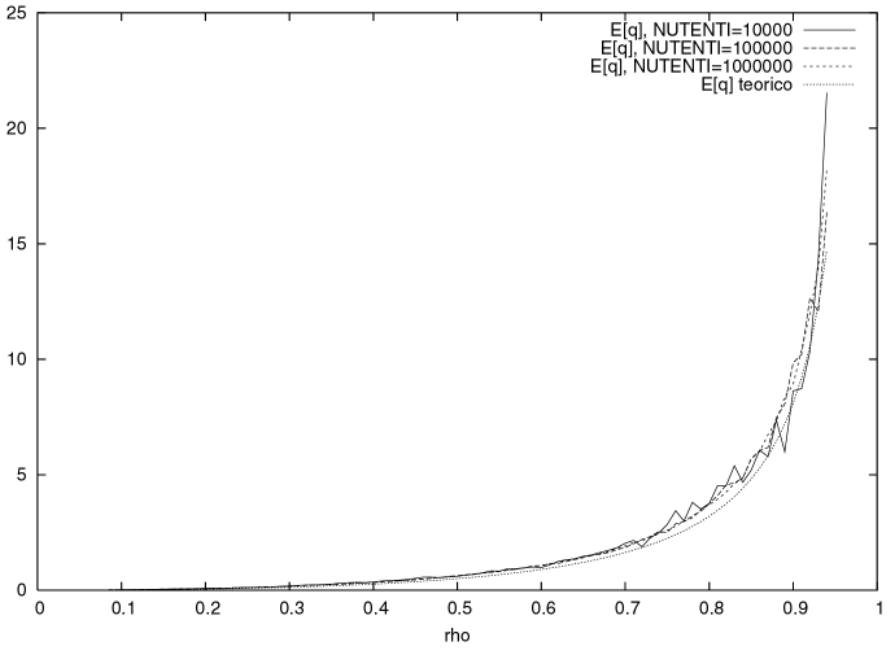

Figura 2.9: $E[q]$ con NUTENTI $= 10^4, 10^5, 10^6$

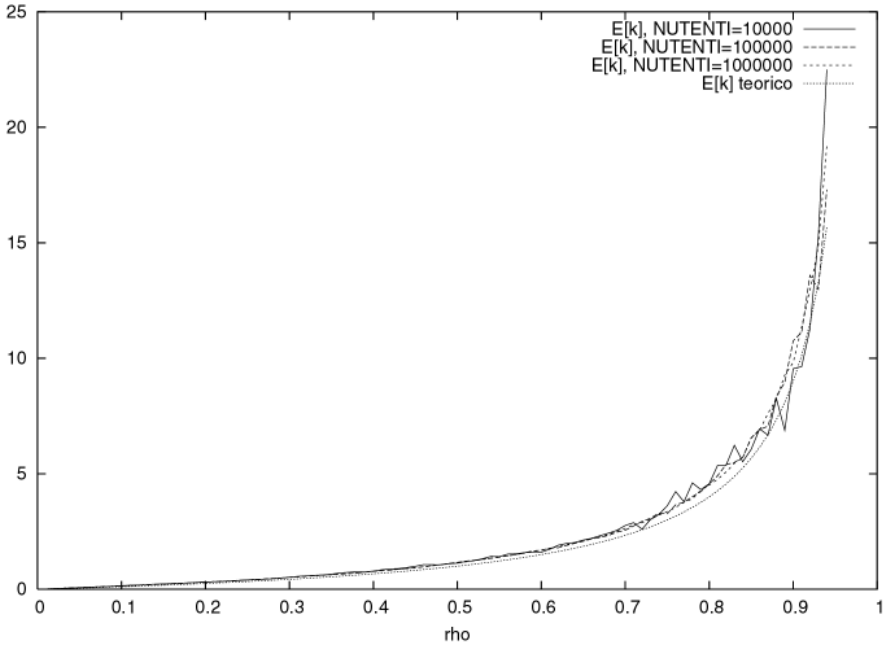

Figura 2.10: $E[k]$ con NUTENTI $= 10^4, 10^5, 10^6$

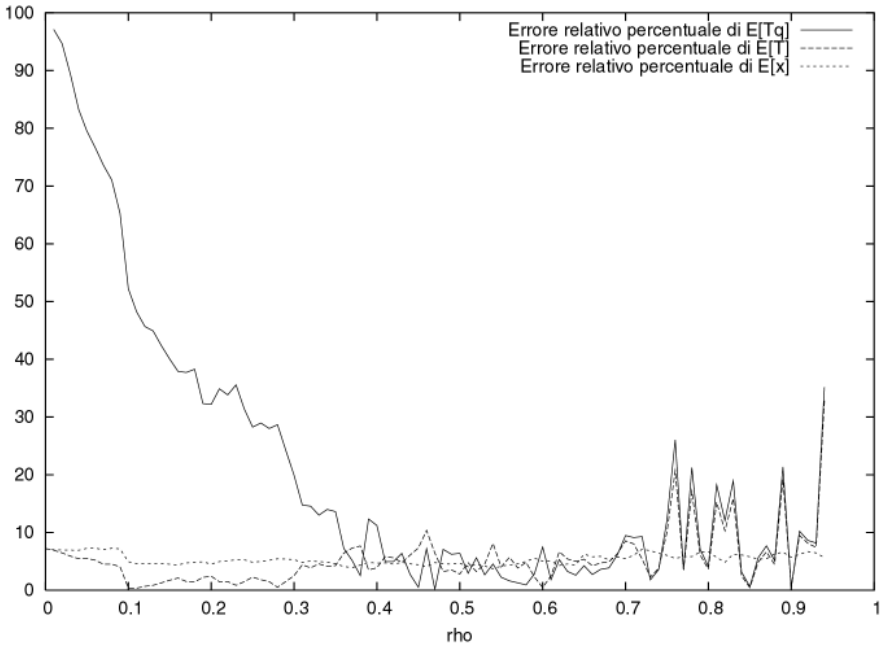

Figura 2.11: Err. rel. perc. di $E[Tq]$, $E[T]$ ed $E[x]$ con NUTENTI $= 10^4$

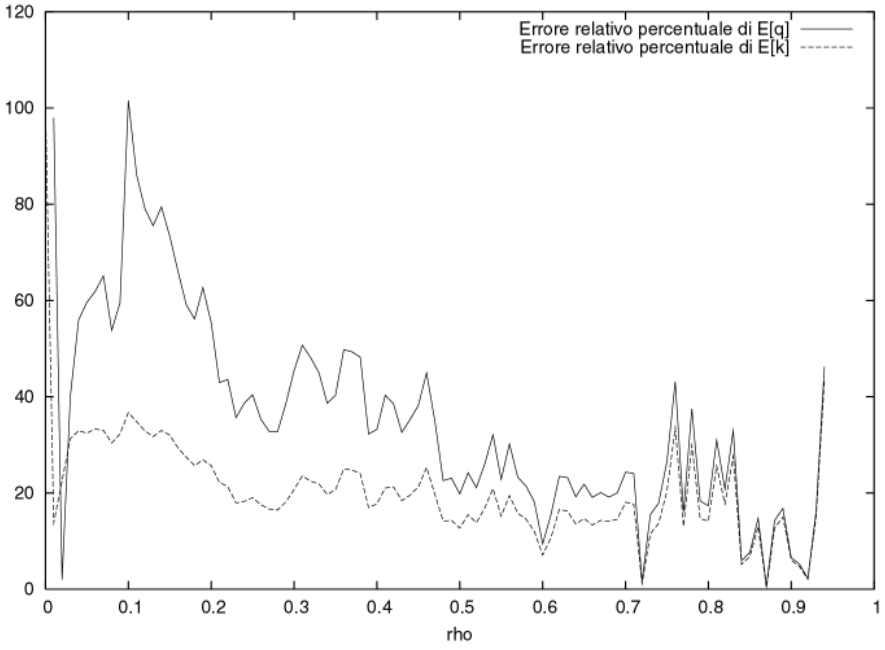

Figura 2.12: Err. rel. perc. di $E[q]$ ed $E[k]$ con NUTENTI $= 10^4$

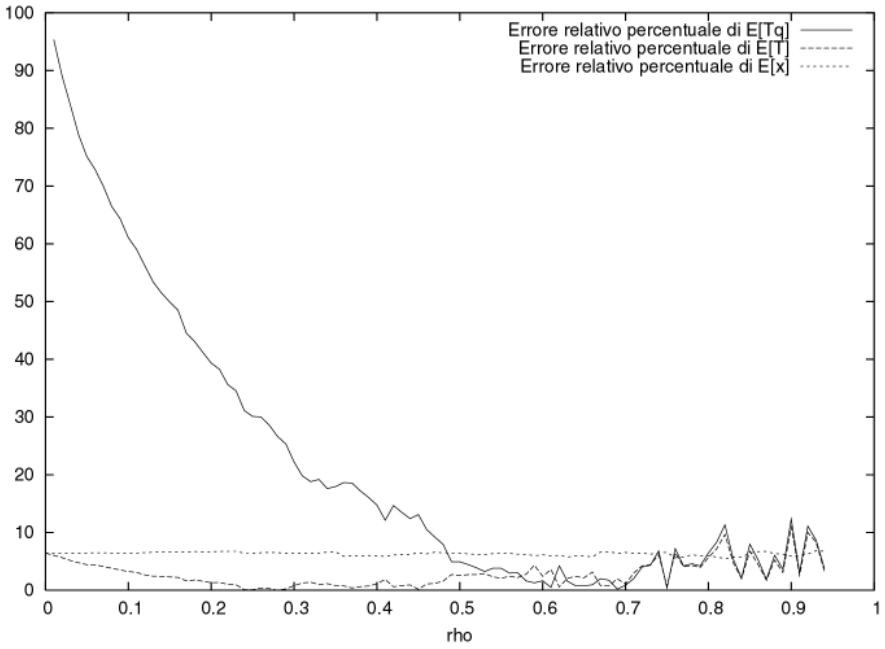

Figura 2.13: Err. rel. perc. di $E[Tq]$, $E[T]$ ed $E[x]$ con NUTENTI $= 10^5$

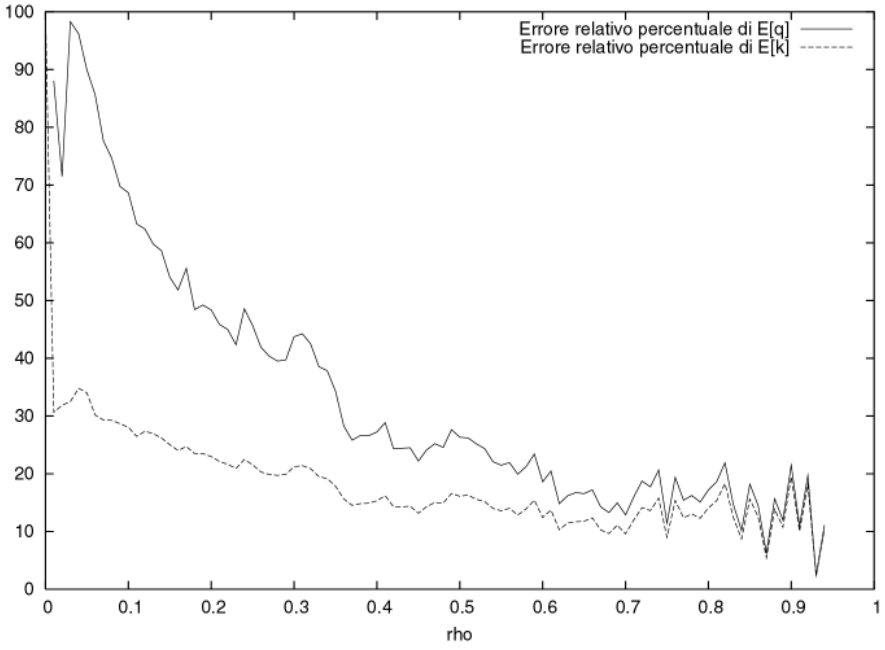

Figura 2.14: Err. rel. perc. di $E[q]$ ed $E[k]$ con NUTENTI $= 10^5$

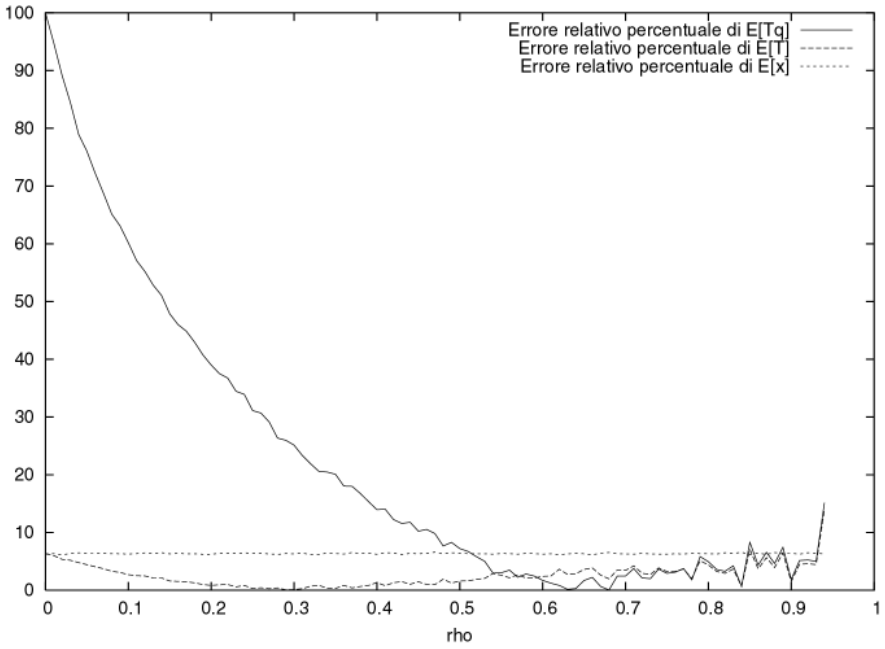

Figura 2.15: Err. rel. perc. di $E[Tq]$, $E[T]$ ed $E[x]$ con NUTENTI $= 10^6$

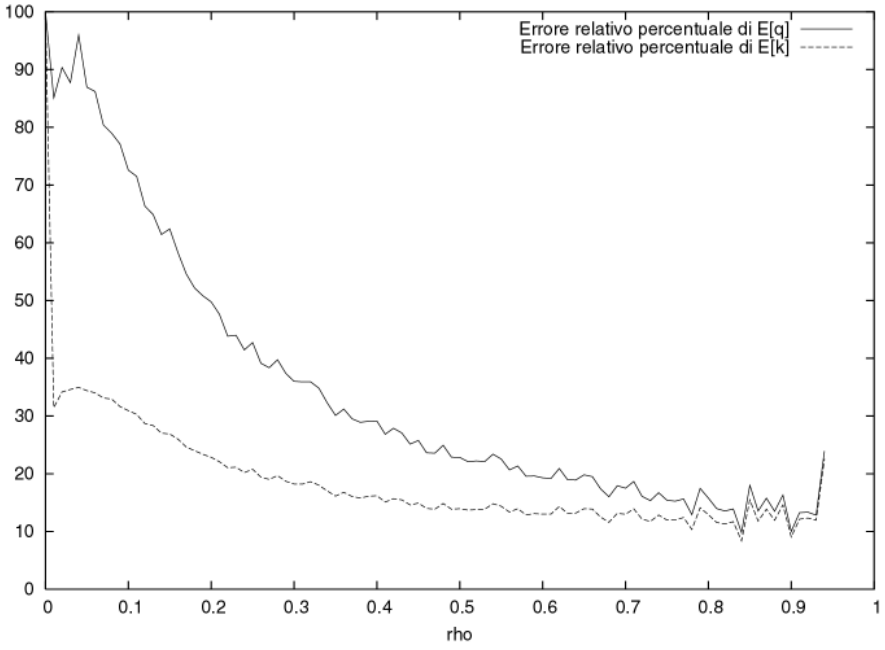

Figura 2.16: Err. rel. perc. di $E[q]$ ed $E[k]$ con NUTENTI $= 10^6$

Per concludere questa breve ma dovuta discussione sulla validità della simulazione, facciamo alcune osservazioni.

Gli errori relativi percentuali iniziali risultano molto elevati perché le grandezze teoriche assumono inizialmente valori prossimi allo zero, per cui assistiamo ad un fenomeno di *amplificazione delle fluttuazioni statistiche* sui dati iniziali. Successivamente gli errori decrescono.

Al crescere del numero di utenti aumenta la precisione del simulatore ma cresce altresì il tempo reale di simulazione al calcolatore, come riportato in tabella 2.1. Negli esperimenti successivi, cercando di coniugare dati at-

Numero di utenti	**Tempo di simulazione**
10^4	1 sec.
10^5	3 sec.
10^6	20 sec.
10^7	4 min.

Tabella 2.1: Tempi di simulazione su di un Pentium 4 a 1.5 GHz

tendibili con brevi tempi di simulazione, abbiamo configurato il simulatore per generare un numero di utenti NUTENTI $= 10^6$.

Capitolo 3

Sistemi M/M/1

La simulazione di un sistema con arrivi e servizi distribuiti secondo Poisson e con un unico servitore è stata eseguita configurando il simulatore con i valori riportati in tabella 3.1. La dimensione della coda è di fatto infinita, poiché

Parametro	**Valore**	**Descrizione**
NUTENTI	1000000	Numero di utenti del sistema.
Y	1000000	Massima dimensione della coda.
MU	1 sec^{-1}	Tasso di morte degli utenti.
LAMBDA	[0.000001 .. 0.95MU] sec^{-1}	Tasso di nascita degli utenti.
INCR_LAMBDA	0.01 sec^{-1}	Incremento sul tasso di nascita degli utenti.

Tabella 3.1: Parametri di simulazione del sistema M/M/1

essendo uguale al numero di utenti è in grado di contenerli tutti.

In figura 3.1 osserviamo l'andamento del numero medio di utenti in coda $E[q]$ ed il numero medio di utenti nel sistema $E[k]$. A mano a mano che gli arrivi diventano più frequenti, fino a raggiungere la velocità di smaltimento del sistema, assistiamo ad un aumento del numero di utenti in coda e nel sistema.

In figura 3.2 osserviamo gli andamenti del tempo medio di attesa in coda $E[Tq]$, del tempo medio di permanenza nel sistema $E[T]$ e del tempo medio di servizio $E[x]$. Al crescere del tasso di arrivo aumentano i tempi di permanenza in coda e nel sistema.

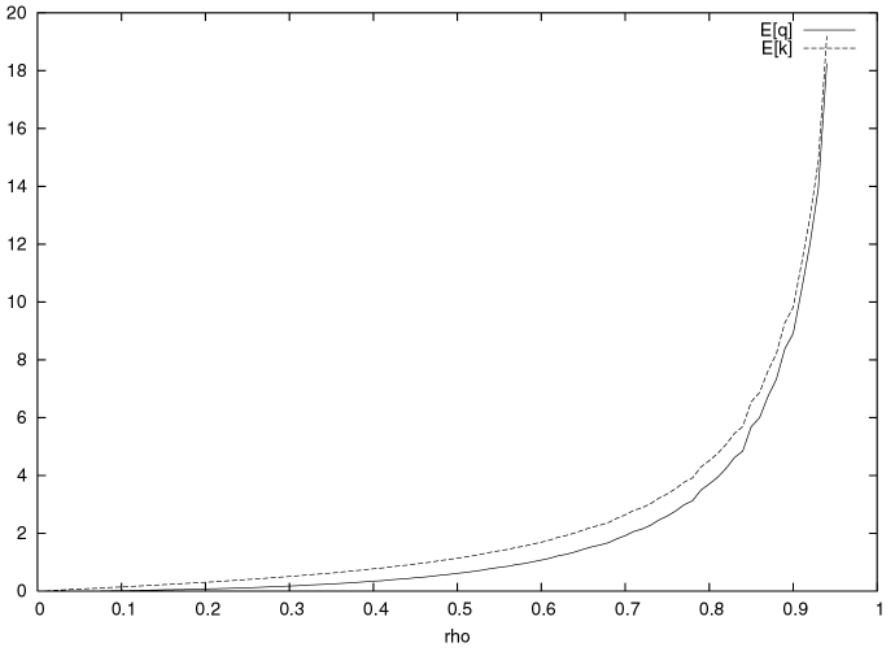

Figura 3.1: $E[q]$ ed $E[k]$ per un sistema M/M/1

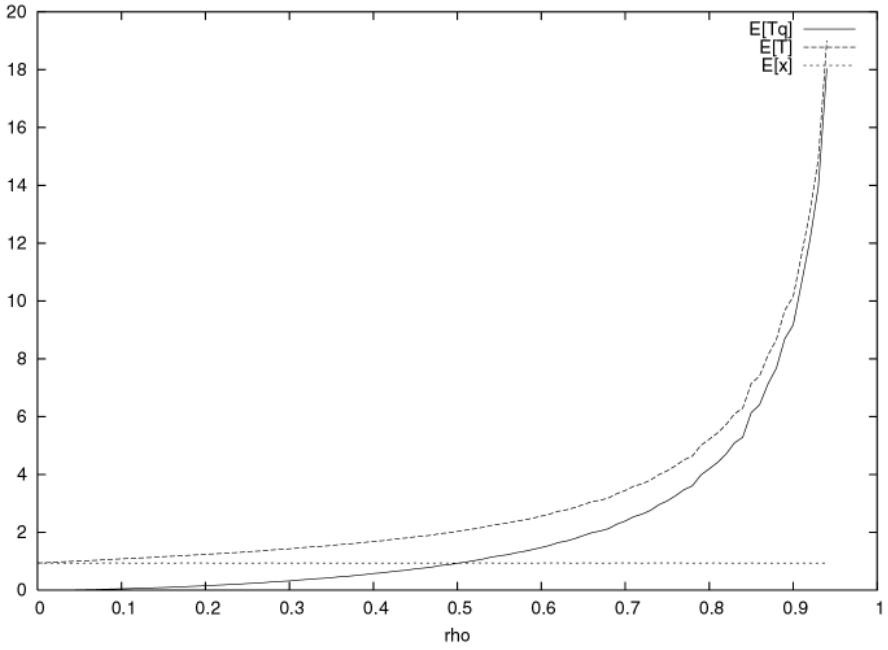

Figura 3.2: $E[Tq]$, $E[T]$ ed $E[x]$ per un sistema M/M/1

In figura 3.3 osserviamo i tempi medi, rispetto al tempo totale di simulazione, in cui il servitore risulta libero $E[Tlib]$ ed occupato $E[Tocc]$. Al

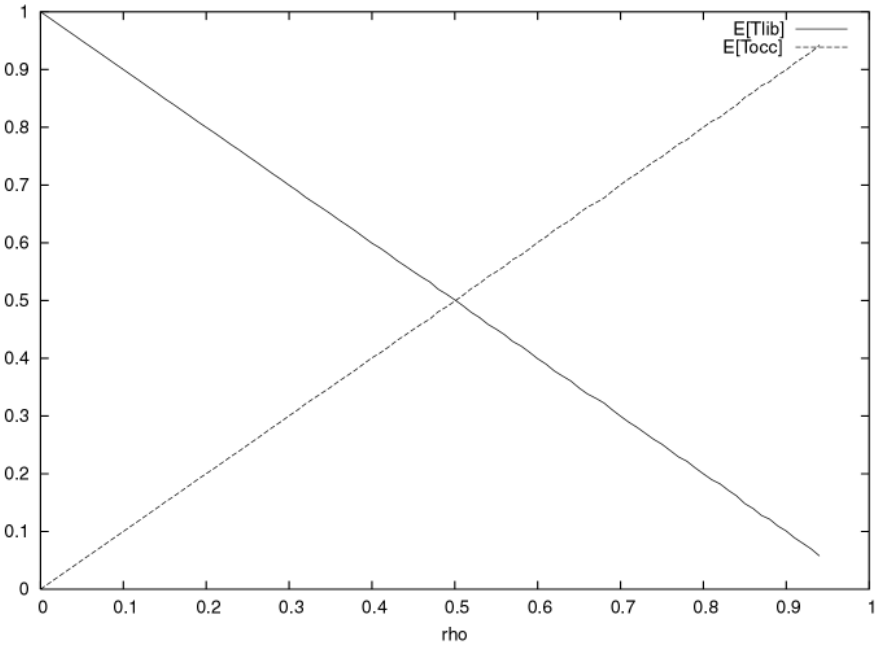

Figura 3.3: $E[Tlib]$ ed $E[Tocc]$ per un sistema M/M/1

crescere del tasso di arrivo il servitore è sempre meno libero, ed è sempre più occupato nel servire gli utenti.

Capitolo 4

Sistemi M/M/1/Y

La simulazione di un sistema con arrivi e servizi distribuiti secondo Poisson e con un unico servitore con coda di dimensione Y è stata eseguita configurando il simulatore con i valori riportati in tabella 4.1. Abbiamo fissato la

Parametro	Valore	Descrizione
NUTENTI	1000000	Numero di utenti del sistema.
Y	10	Massima dimensione della coda.
MU	1 sec^{-1}	Tasso di morte degli utenti.
LAMBDA	[0.000001 .. 0.95MU] sec^{-1}	Tasso di nascita degli utenti.
INCR_LAMBDA	0.01 sec^{-1}	Incremento sul tasso di nascita degli utenti.

Tabella 4.1: Parametri di simulazione del sistema M/M/1/Y

dimensione della coda a 10 utenti, in modo da ben evidenziare le differenze di comportamento rispetto ai sistemi M/M/1.

In figura 4.1 osserviamo l'andamento del numero medio di utenti in coda $E[q]$ ed il numero medio di utenti nel sistema $E[k]$. L'andamento è identico a quello osservato per i sistemi M/M/1 con la differenza che ora il numero medio di persone è diminuito notevolmente, essendo vincolato dalla coda finita.

In figura 4.2 osserviamo gli andamenti del tempo medio di attesa in coda $E[Tq]$, del tempo medio di permanenza nel sistema $E[T]$, e del tempo medio di servizio $E[x]$. Ancora stesso andamento dei sistemi M/M/1 ma con tempi

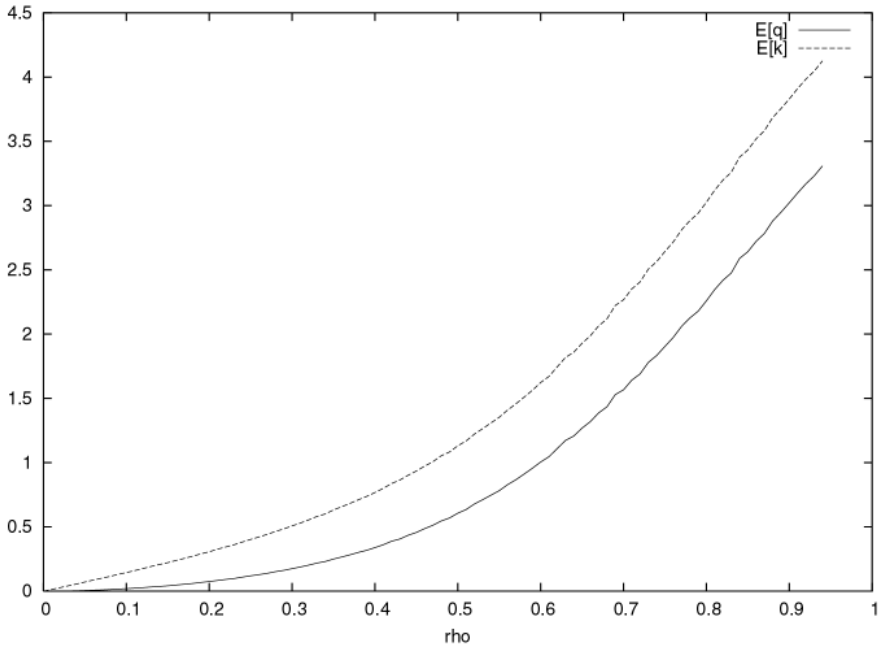

Figura 4.1: $E[q]$ ed $E[k]$ per un sistema M/M/1/Y con $\mathbf{Y} = 10$

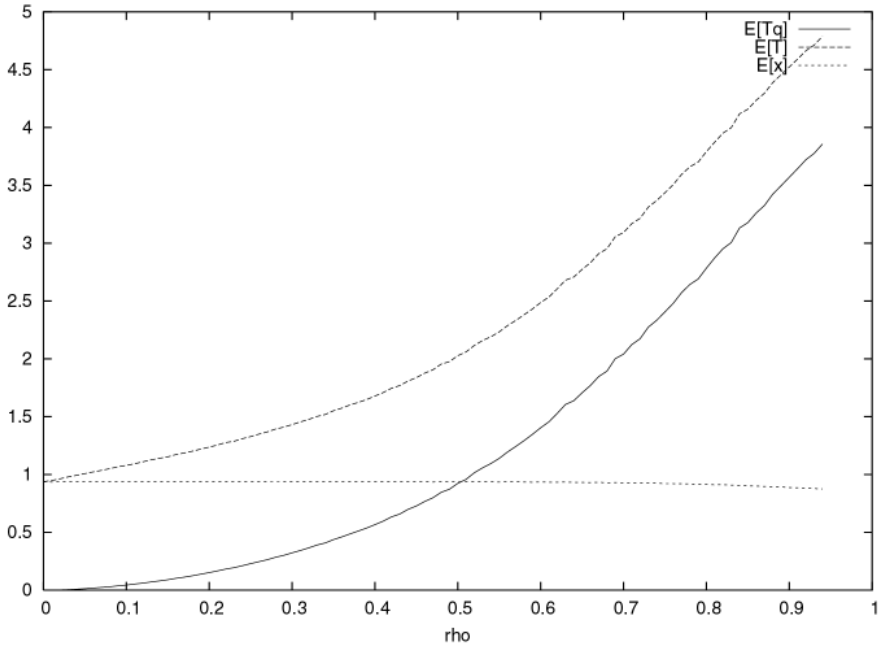

Figura 4.2: $E[Tq]$, $E[T]$ ed $E[x]$ per un sistema M/M/1/Y con $\mathbf{Y} = 10$

più bassi, vincolati dalla coda finita.

In figura 4.3 osserviamo i tempi medi in cui il servitore risulta libero $E[Tlib]$ ed occupato $E[Tocc]$. Simili a quelli degli M/M/1 ma con incurva-

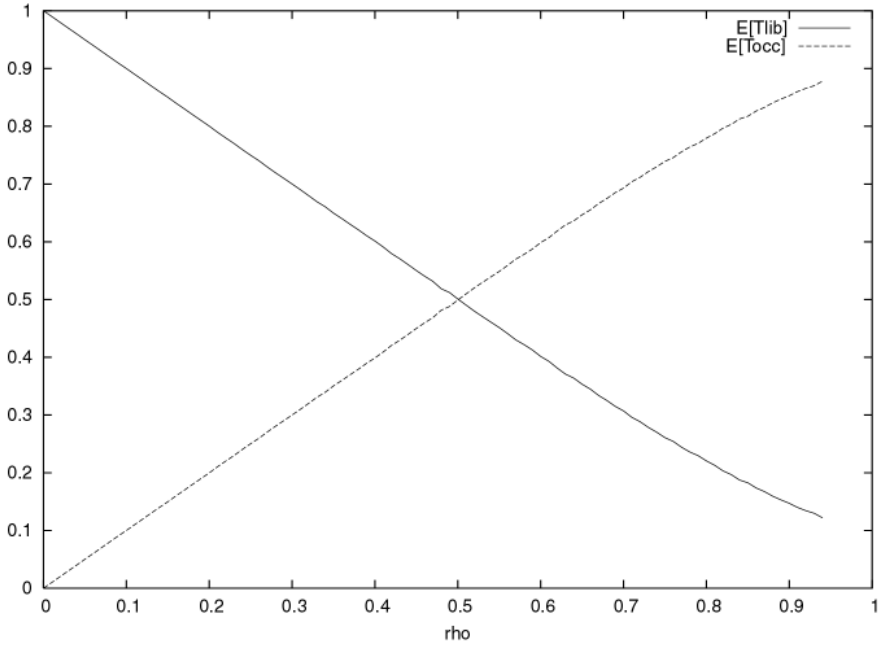

Figura 4.3: $E[Tlib]$ ed $E[Tocc]$ per un sistema M/M/1/Y con $\mathbf{Y} = 10$

mento verso il centro per ρ tendente ad 1. In effetti la dimensione finita della coda rende il servitore mediamente meno occupato e più libero poiché alcuni utenti non accederanno al sistema, trovando la coda piena.

Abbiamo ritenuto interessante osservare anche gli andamenti di figura 4.4 del numero medio di utenti persi per coda piena $E[l]$ ed il numero medio di occorrenze di coda piena $E[f]$. Per bassi tassi di arrivo i due fenomeni sono inesistenti. Per $\rho \in [0.4, 0.9]$ osserviamo che quasi per ogni volta che la coda è piena si perde un utente. Per $\rho \in [0.9, 1]$ la tendenza si inverte leggermente e per ogni volta che la coda è piena si perde quasi più di un utente. Mediamente si perde un massimo del 7% degli utenti generati.

Limitando esageratamente la coda ad $\mathbf{Y} = 5$ utenti, gli andamenti dei tempi medi in cui il servitore risulta libero $E[Tlib]$ ed occupato $E[Tocc]$ divengono come in figura 4.5, in cui il servitore è sempre più libero e meno occupato rispetto al caso con $\mathbf{Y}$ =10.

In figura 4.6 riportiamo gli andamenti relativi al numero medio di utenti persi per coda piena $E[l]$ ed al numero medio di occorrenze di coda piena $E[f]$. Il fenomeno di inversione si manifesta sempre intorno ad un $\rho \approx$

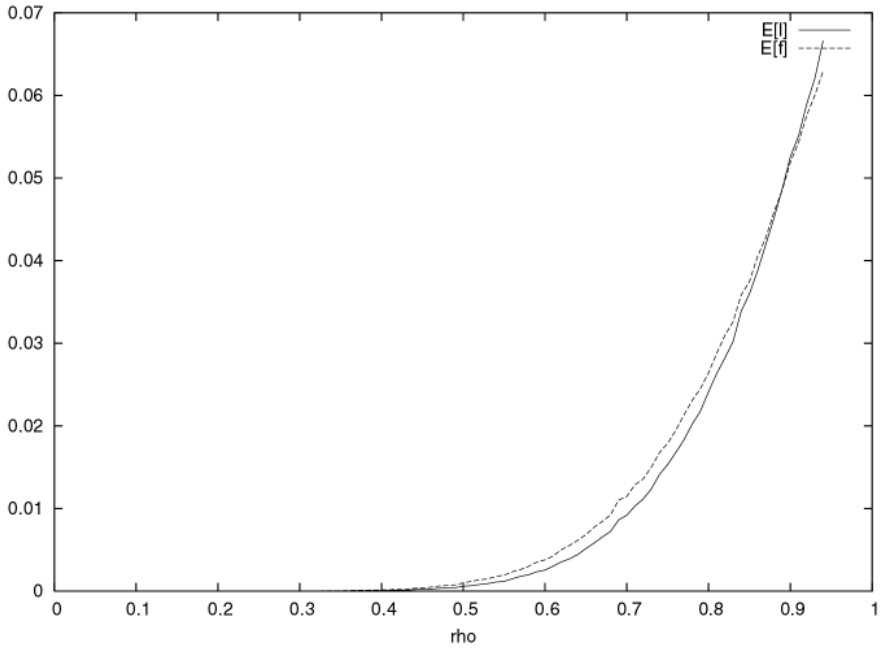

Figura 4.4: $E[l]$ ed $E[f]$ per un sistema M/M/1/Y con $\mathbf{Y} = 10$

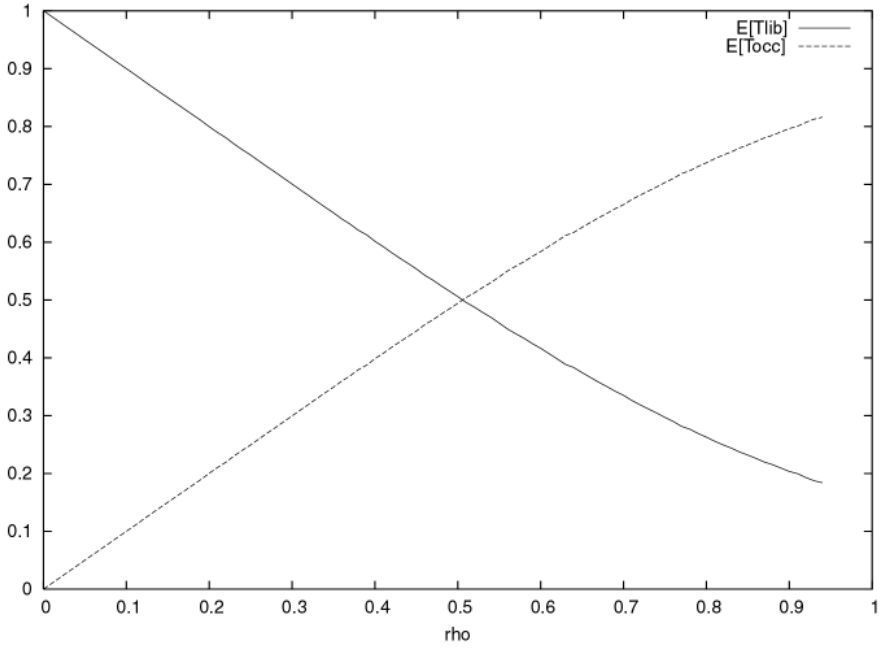

Figura 4.5: $E[Tlib]$ ed $E[Tocc]$ per un sistema M/M/1/Y con $\mathbf{Y} = 5$

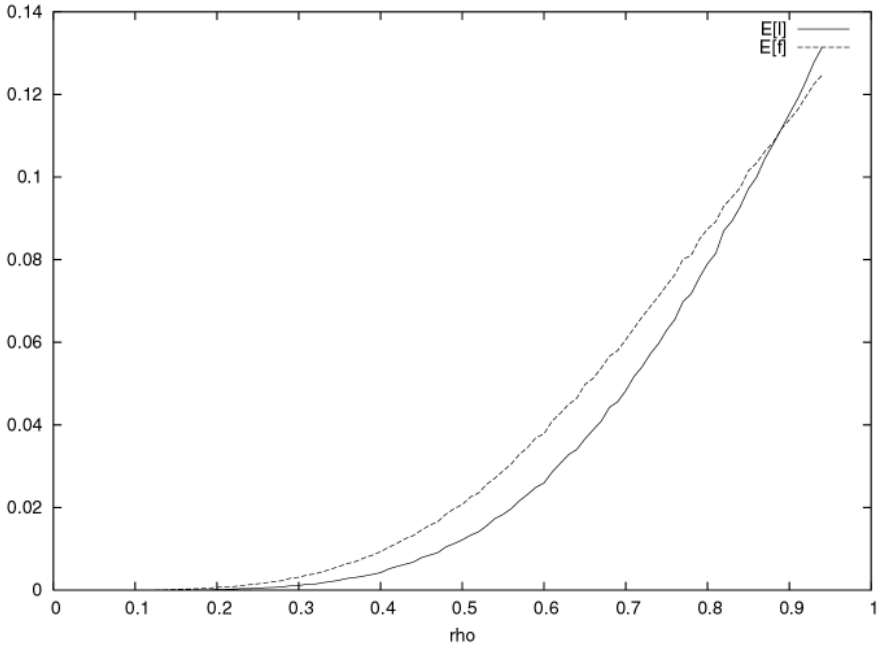

Figura 4.6: $E[l]$ ed $E[f]$ per un sistema M/M/1/Y con $\mathtt{Y} = 5$

0.9. Mediamente si perde un massimo del 13% degli utenti generati, ossia dimezzando la coda sono raddoppiate le perdite.

Capitolo 5

Sistemi M/D/1

La simulazione di un sistema con arrivi distribuiti secondo Poisson, servizio deterministico fissato a 0.9 sec e con un unico servitore è stata eseguita configurando il simulatore con i valori riportati in tabella 5.1. La dimensione

Parametro	**Valore**	**Descrizione**
NUTENTI	1000000	Numero di utenti del sistema.
Y	1000000	Massima dimensione della coda.
MU	1 sec^{-1}	Tasso di morte degli utenti.
LAMBDA	[0.000001 .. 0.95MU] sec^{-1}	Tasso di nascita degli utenti.
INCR_LAMBDA	0.01 sec^{-1}	Incremento sul tasso di nascita degli utenti.
$E[x] = x$	0.9 sec	Tempo di servizio.

Tabella 5.1: Parametri di simulazione del sistema M/D/1

della coda è di fatto infinita, poiché essendo uguale al numero di utenti è in grado di contenerli tutti.

La simulazione evidenzia in figura 5.1 l'andamento del numero medio di utenti in coda $E[q]$ ed il numero medio di utenti nel sistema $E[k]$. Il numero di utenti in coda e nel sistema è più basso rispetto ai sistemi M/M/1 poiché è stato scelto un tempo di servizio fisso a 0.9 sec. più breve rispetto a quello medio del M/M/1.

Ciò appare chiaramente dagli andamenti di $E[Tq]$, di $E[T]$ e di $E[x] = x$ in figura 5.2. Al crescere del tasso di arrivo aumentano i tempi di permanenza

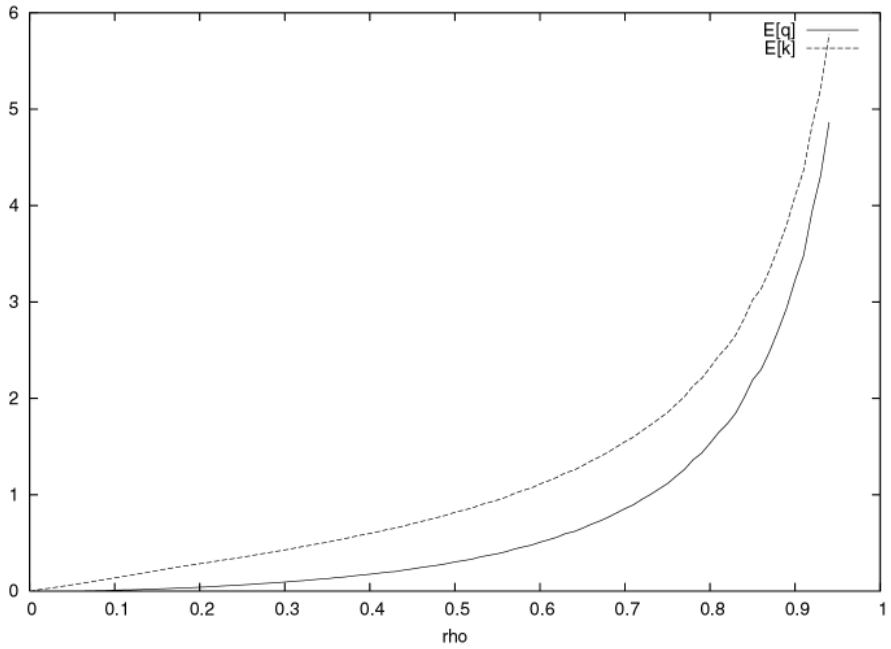

Figura 5.1: $E[q]$ ed $E[k]$ per un sistema M/D/1

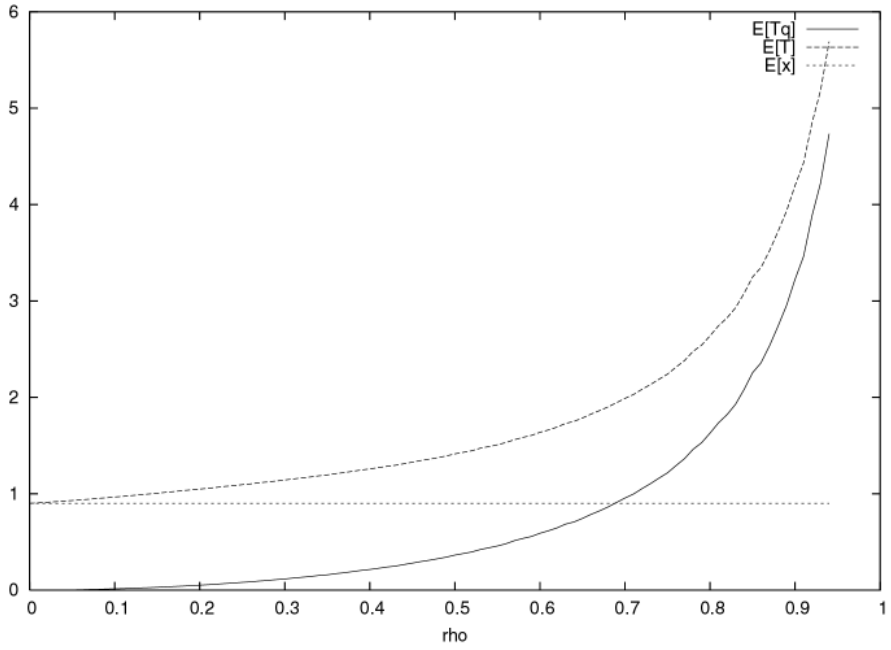

Figura 5.2: $E[Tq]$, $E[T]$ ed $E[x]$ per un sistema M/D/1

in coda e nel sistema.

I tempi medi, rispetto al tempo totale di simulazione, in cui il servitore risulta libero $E[Tlib]$ ed occupato $E[Tocc]$ hanno gli andamenti di figura 5.3. Come nell' M/M/1 non abbiamo incurvamento delle caratteristiche nel tratto

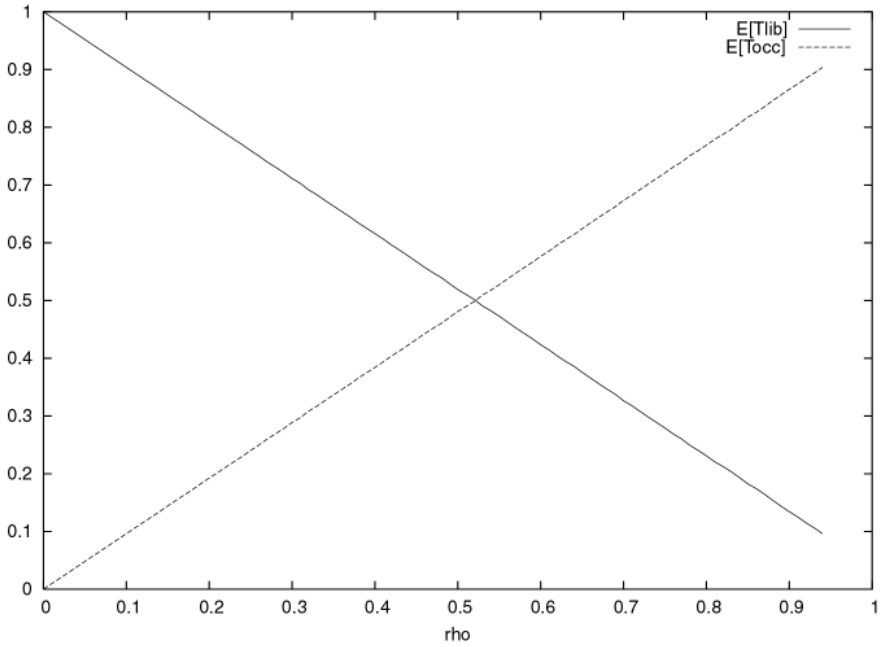

Figura 5.3: $E[Tlib]$ ed $E[Tocc]$ per un sistema M/D/1

prossimo a ρ unitario, ma osserviamo un calo di pendenza in $E[Tocc]$ ed un aumento in $E[Tlib]$, sempre giustificato al fatto che ora abbiamo un servitore sempre (non solo mediamente) più veloce.

Capitolo 6

Sistemi M/D/1/Y

La simulazione di un sistema con arrivi distribuiti secondo Poisson, servizio deterministico fissato a 0.9 sec e con un unico servitore con coda di dimensione Y è stata eseguita configurando il simulatore con i valori riportati in tabella 6.1. La simulazione evidenzia in figura 6.1 l'andamento del numero

Parametro	**Valore**	**Descrizione**
NUTENTI	1000000	Numero di utenti del sistema.
Y	10	Massima dimensione della coda.
MU	1 sec^{-1}	Tasso di morte degli utenti.
LAMBDA	[0.000001 .. 0.95MU] sec^{-1}	Tasso di nascita degli utenti.
INCR_LAMBDA	0.01 sec^{-1}	Incremento sul tasso di nascita degli utenti.
$E[x] = x$	1 sec	Tempo di servizio.

Tabella 6.1: Parametri di simulazione del sistema M/D/1/Y

medio di utenti in coda $E[q]$ ed il numero medio di utenti nel sistema $E[k]$. L'andamento è identico a quello osservato per i sistemi M/D/1 con la differenza che ora il numero medio di persone è diminuito, essendo vincolato dalla coda finita.

In figura 6.2 osserviamo gli andamenti del tempo medio di attesa in coda $E[Tq]$, del tempo medio di permanenza nel sistema $E[T]$, e del tempo medio di servizio $E[x]$. Ancora stesso andamento dei sistemi M/D/1 ma con tempi più bassi, vincolati dalla coda finita.

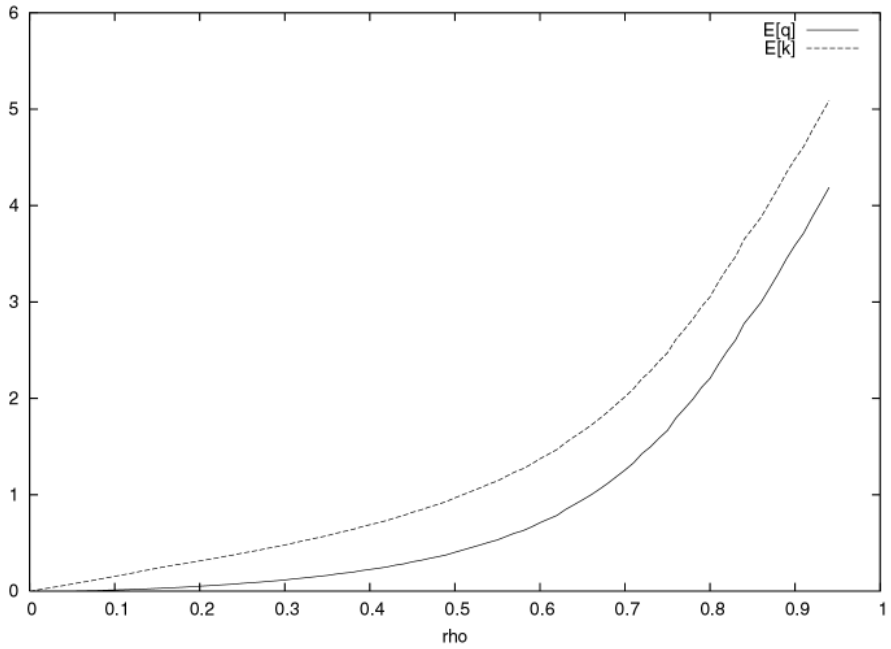

Figura 6.1: $E[q]$ ed $E[k]$ per un sistema M/D/1/Y con $\mathbf{Y} = 10$

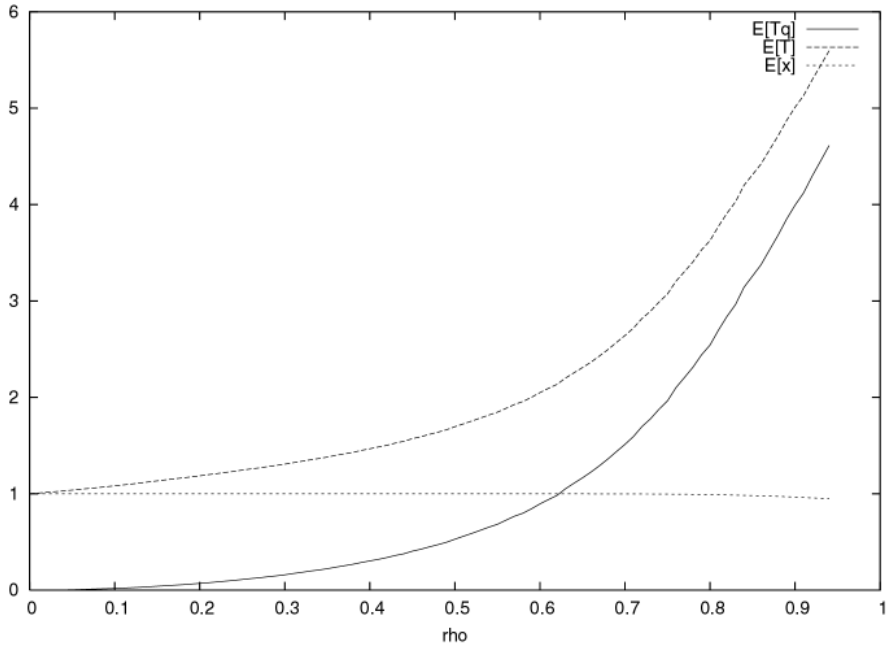

Figura 6.2: $E[Tq]$, $E[T]$ ed $E[x]$ per un sistema M/D/1/Y con $\mathbf{Y} = 10$

I tempi in cui il servitore risulta libero $E[Tlib]$ ed occupato $E[Tocc]$ hanno gli andamenti di figura 6.3. Simili a quelli degli M/D/1 ma con incurvamento

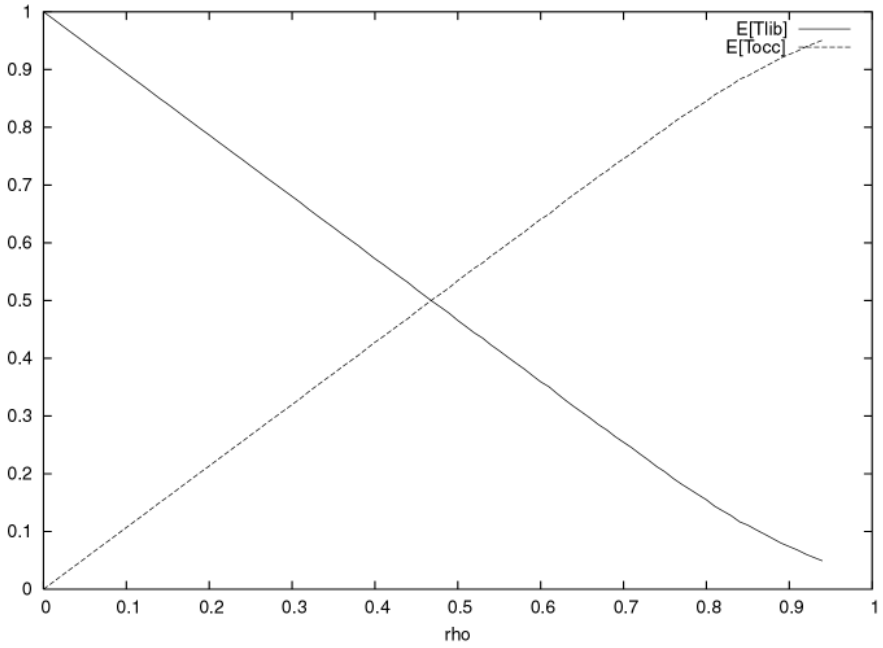

Figura 6.3: $E[Tlib]$ ed $E[Tocc]$ per un sistema M/D/1/Y con $\mathbf{Y} = 10$

verso il centro appena percepibile per ρ tendente ad 1. In effetti la dimensione finita della coda rende il servitore mediamente meno occupato e più libero poiché alcuni utenti non accederanno al sistema, trovando la coda piena.

Abbiamo ritenuto interessante osservare in figura 6.4 anche l'andamento del numero medio di utenti persi per coda piena $E[l]$ ed il numero medio di occorrenze di coda piena $E[f]$. Per bassi tassi di arrivo i due fenomeni sono inesistenti. Per $\rho > 0.5$ il fenomeno di perdita di utenti è decisamente inferiore a quello di occorrenze di coda piena, sicuramente anche per via del servitore più veloce. Mediamente si perde un massimo del 4% degli utenti generati.

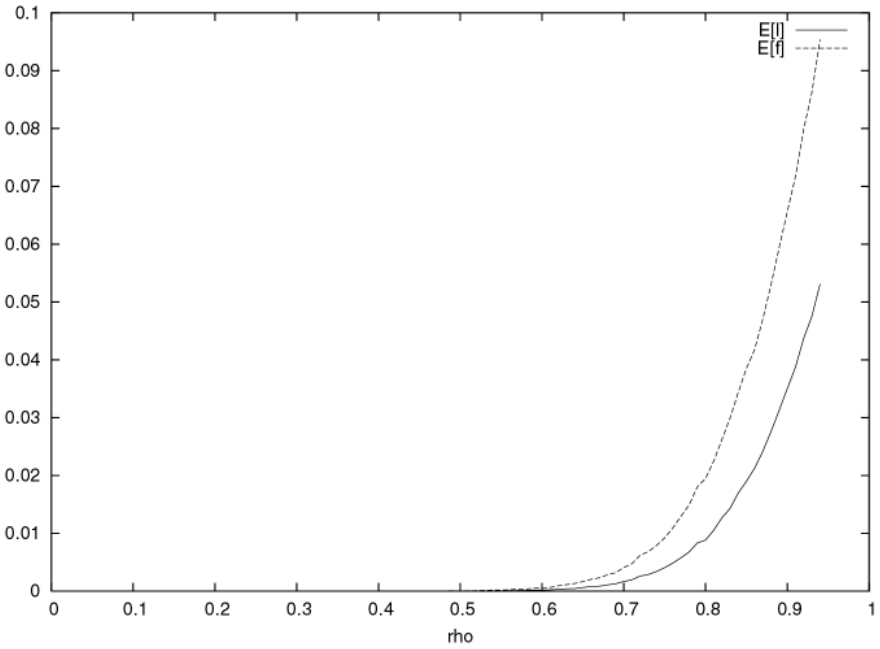

Figura 6.4: $E[l]$ ed $E[f]$ per un sistema M/D/1/Y con $\mathbf{Y} = 10$

Conclusioni

I risultati simulati confermano quelli teorici con un discreto grado di precisione.

Miglioramenti sarebbero ottenibili aumentando il numero di utenti generati per collezionare dati statistici, con conseguente aumento del tempo reale di simulazione.

Esiste tuttavia un limite inferiore oltre il quale gli errori relativi non potranno mai scendere, limite imposto dalle ipotesi esemplificative introdotte nel modello teorico.

Interessanti conclusioni ulteriori sarebbero state possibili confrontando fra loro le dinamiche delle grandezze di interesse al variare dei parametri caratteristici, ma questo avrebbe richiesto una notevole quantità di tempo per collezionare e graficare decine di simulazioni.

Il simulatore è stato costruito per analizzare i sistemi M/M/1 ed M/M/1/Y. Con una piccola modifica è stato possibile modificarlo per analizzare anche i sistemi M/D/1 e M/D/1/Y. Ulteriori piccole modifiche sono possibili per analizzare anche i sistemi D/M/1, D/M/1/Y, D/D/1 e D/D/1/Y.

Si potrebbe prevedere un sistema più raffinato, del semplice random offerto dal linguaggio C, per la generazione di variabili casuali distribuite uniformemente.

Infine il simulatore, in questa sua prima versione, è stato scritto mirando alla leggibilità del codice, miglioramenti futuri potrebbero prendere in considerazione soluzioni più efficienti in termini di tempo, più eleganti dal punto di vista informatico e più amichevoli "sviluppando" un' interfaccia utente.

Appendice A

Generazione di variabile distribuita secondo Poisson

La variabile aleatoria X con funzione di densità di probabilità $f_X(x)$ è stata calcolata adottando la tecnica di trasformazione inversa.

1. Si calcola la funzione di distribuzione di probabilità di $f_X(x)$. Tale funzione (qualora sia possibile calcolarla in forma chiusa) è continua, monotona crescente ed è sempre compresa tra 0 e 1 (per definizione $F_X(x) = P[X = x]$).

2. Si pone $u = F_X(x)$, con u numero casuale distribuito uniformemente nell'intervallo $[0, 1]$, ossia $u \sim U(0, 1)$.

3. Si risolve $X = F_X^{-1}(u)$ ottenendo la variabile aleatoria X distribuita secondo la desiderata $f_X(x)$, ossia $X \sim f_X(x)$.

Nel nostro caso volendo una variabile casuale distribuita secondo Poisson, abbiamo considerato la funzione di distribuzione di probabilità:

$$F(t) = 1 - e^{-\lambda t}$$

quindi ponendo:

$$u = 1 - e^{-\lambda t}$$

che invertita:

$$t = -\frac{1}{\lambda} ln(1 - u)$$

Per generare u abbiamo utilizzato la funzione *random(n)* che genera un numero compreso fra 1 ed n distribuito in modo uniforme. Abbiamo scelto una $RISOLUZIONE = 1000000000$ in modo da avere pressoché una

varietà continua di valori di $u \in [0,1]$, o meglio $u \in [1/n,1]$, tramite la normalizzazione:

$$u = \frac{random(RISOLUZIONE)}{RISOLUZIONE}$$

Infine abbiamo calcolato t:

$$t = -\frac{1}{\lambda} ln(1-u)$$

Come si può osservare dalla funzione:

```
double poisson()
// genera un tempo di interarrivo distribuito secondo Poisson
{
  return (-logl(1 - (random((double)RISOLUZIONE) / (double)RISOLUZIONE))
          / (double)lambda);
}
```

Appendice B

Codice sorgente del simulatore

Riportiamo il codice per la simulazione di sistemi M/M/1 ed M/M/1/Y. Il simulatore ad eventi è stato scritto in linguaggio C all'interno dell'ambiente di sviluppo Borland C++ Builder.

Per ottenere il codice per simulare i sistemi M/D/1 ed M/D/1/Y e sufficiente sostituire la function *txUtente* come indicato nel codice.

```
// Simulatore ad eventi di Sistemi a Coda di tipo MM1, MM1Y, MD1, MD1Y - Ver. 1.0
// Copyright (C) 2003 Tarin Gamberini
//
// This program is free software; you can redistribute it and/or
// modify it under the terms of the GNU General Public License
// as published by the Free Software Foundation; either version 2
// of the License, or (at your option) any later version.
//
// This program is distributed in the hope that it will be useful,
// but WITHOUT ANY WARRANTY; without even the implied warranty of
// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
// GNU General Public License for more details.
//
// You should have received a copy of the GNU General Public
// License along with this program; if not, write to the Free
// Software Foundation, Inc., 59 Temple Place - Suite 330,
// Boston, MA 02111-1307, USA.


// Simulatore ad eventi di Sistemi a Coda di tipo MM1, MM1Y, MD1, MD1Y - Ver. 1.0
// Copyright (C) 2003 Tarin Gamberini
//
// Questo programma è software libero; è lecito redistribuirlo o
// modificarlo secondo i termini della Licenza Pubblica Generica
// GNU come è pubblicata dalla Free Software Foundation; o la
// versione 2 della licenza o (a propria scelta) una versione
// successiva.
//
// Questo programma è distribuito nella speranza che sia utile, ma
// SENZA ALCUNA GARANZIA; senza neppure la garanzia implicita di
// NEGOZIABILITÀ o di APPLICABILITÀ PER UN PARTICOLARE SCOPO. Si
```

```
// veda la Licenza Pubblica Generica GNU per avere maggiori
// dettagli.
//
// Questo programma deve essere distribuito assieme ad una copia
// della Licenza Pubblica Generica GNU; in caso contrario, se ne
// può ottenere una scrivendo alla Free Software Foundation,
// Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA.



//---------------------------------------------------------------------------
#include <vcl\condefs.h>
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <math.h>
#include <time.h>



#pragma hdrstop
//---------------------------------------------------------------------------
USERES("SimuMM1Y.res");
//---------------------------------------------------------------------------
double  NUTENTI     = 1000000;  // numero di utenti del sistema
double  Y           = 1000000;  // massima dimensione della coda
double  lambda;                 // tasso di nascita degli utenti
double  MU          = 1;        // tasso di morte degli utenti
double  INCR_LAMBDA = 0.01;     // incremento sul tasso di nascita degli utenti

unsigned int  x;  // contatore del numero di utenti attualmente in servizio
unsigned int  q;  // contatore del numero di utenti attualmente in coda
unsigned int  k;  // contatore del numero di utenti attualmente nel sistema

double  accumula_tx; // accumula il tempo trascorso in servizio
double  accumula_tq; // accumula il tempo trascorso in coda
double  accumula_tk; // accumula il tempo trascorso nel sistema

double  accumula_x;  // accumula il numero di utenti attualmente in servizio
double  accumula_q;  // accumula il numero di utenti attualmente in coda
double  accumula_k;  // accumula il numero di utenti attualmente nel sistema

double  nUtEntratiServ;  // contatore del numero di utenti entrati in servizio
double  nUtEntratiCoda;  // accumula il numero di utenti entrati in coda
double  nUtEntratiSist;  // accumula il numero di utenti entrati nel sistema

double  nUtUscitiServ;   // contatore del numero di utenti usciti dal servizio
double  nUtUscitiCoda;   // accumula il numero di utenti usciti in coda
double  nUtUscitiSist;   // accumula il numero di utenti usciti nel sistema

double  nUtentiGenerati; // contatore del numero di utenti generati
double  nUtentiPersi;    // contatore del numero di utenti persi per coda piena
double  occorrenzeCodaPiena;  // numero di volte in cui la coda è piena

double  tInterarrivo;  // tempo trascorso fra due arrivi consecutivi di utenti
double  tSimulazione;  // accumula il tempo di simulazione
double  tServitoreLibero;  // accumula il tempo in cui il servitore è libero
double  tServitoreOccupato;  // accumula il tempo in cui il servitore è occupato



struct tipoUtente
```

```
{
  double  tx;           // tempo di servizio assegnato
  double  tq;           // tempo trascorso in coda
  double  tk;           // tempo trascorso nel sistema

  // tempo rimanente all'esaurimento del tempo di servizio assegnato
  double  txRimanente;
}  utente;              // variabile che identifica un utente



struct tipoInCoda
{
  tipoUtente      utente;    // tempo corrente di attesa in coda
  struct tipoInCoda  *succ;    // puntatore al successivo utente in coda
}  *pTesta, *pCoda;        // puntatori di testa e di coda della lista FIFO

FILE      *fileDati;      // file di dati

// cardinalità dell'insieme dei possibili tempi interarrivo
double RISOLUZIONE = 1000000000;



double poisson()
// genera un tempo di interarrivo distribuito secondo Poisson
{
  return (-logl(1 - (random((double)RISOLUZIONE) / (double)RISOLUZIONE))
          / (double)lambda);
}



double txUtente()
// genera un tempo di interservizio distribuito secondo Poisson
{
  return (-logl(1 - (random((double)RISOLUZIONE) / (double)RISOLUZIONE))
          / (double)MU);
}
/*double txUtente()
// genera un tempo di interservizio deterministico
{
  return (0.9);
} */



void inizializzaSimulazione()
{
  randomize();

  x = 0;            // contatore del numero di utenti attualmente in servizio
  q = 0;            // contatore del numero di utenti attualmente in coda
  k = 0;            // contatore del numero di utenti attualmente nel sistema

  accumula_tx = 0;  // accumula il tempo trascorso in servizio
  accumula_tq = 0;  // accumula il tempo trascorso in coda
  accumula_tk = 0;  // accumula il tempo trascorso nel sistema

  accumula_x = 0;   // accumula il numero di utenti attualmente in servizio
  accumula_q = 0;   // accumula il numero di utenti attualmente in coda
```

```
  accumula_k = 0;   // accumula il numero di utenti attualmente nel sistema

  nUtEntratiServ = 0;  // contatore del numero di utenti entrati in servizio
  nUtEntratiCoda = 0;  // accumula il numero di utenti entrati in coda
  nUtEntratiSist = 0;  // accumula il numero di utenti entrati nel sistema

  nUtUscitiServ = 0;   // contatore del numero di utenti usciti dal servizio
  nUtUscitiCoda = 0;   // accumula il numero di utenti usciti in coda
  nUtUscitiSist = 0;   // accumula il numero di utenti usciti nel sistema

  nUtentiGenerati = 0; // contatore del numero di utenti generati
  nUtentiPersi = 0;    // contatore del numero di utenti persi per coda piena
  occorrenzeCodaPiena = 0;  // numero di volte in cui la coda è piena

  tInterarrivo;        // tempo trascorso fra due arrivi consecutivi di utenti
  tSimulazione = 0;      // accumula il tempo di simulazione
  tServitoreLibero = 0;    // accumula il tempo in cui il servitore è libero
  tServitoreOccupato = 0;    // accumula il tempo in cui il servitore è occupato

  // creo la coda vuota
  pTesta = NULL;
  pCoda = NULL;
}



void inserisciInCoda(tipoUtente utente)
// gestisce l'accodamento di un utente in coda alla lista
{
  tipoInCoda  *pTmp;

    // creo un utente
    pTmp = (tipoInCoda *)malloc(sizeof(tipoInCoda));
    pTmp->utente = utente;
    pTmp->succ = NULL;
    // lo inserisco in coda
    if (pTesta == NULL)
      pTesta = pTmp;
    else
      pCoda->succ = pTmp;
    pCoda = pTmp;
}



tipoUtente prelevaInTesta()
// ritorna l'elemento di testa della lista, cancellandolo.
// il chiamante si è già assicurato che la coda NON sia vuota
{
    tipoInCoda  *pTmp;
    tipoUtente  utente;

    if (pTesta != NULL)
    {
         utente = pTesta->utente;
        pTmp = pTesta;
        pTesta = pTesta->succ;
        if (pTmp->succ == NULL)
          pCoda=NULL;
        free(pTmp);
    }
    return (utente);
```

```
}



void aggiorna_tq_tkCoda(double time)
// aggiunge agli utenti in coda il tempo time
{
  tipoInCoda  *pTmp;

    pTmp = pTesta;
    //printf("I");
    while (pTmp != NULL)
    {
      //printf(".");
        pTmp->utente.tq = pTmp->utente.tq + time;
        pTmp->utente.tk = pTmp->utente.tk + time;
        pTmp = pTmp->succ;
    }
    //printf("F\n");
}



char serviUtente(tipoUtente *utente, unsigned int *x, double *nUtEntratiServ)
// gestisce l'entrata in servizio dell'utente
{
    // incremento il contatore degli utenti entrati in servizio
    (*nUtEntratiServ)++;

  (*x)++;  // l'utente entra in servizio
  utente->tx = txUtente();  // starà in servizio per un certo tempo txUtente
  utente->txRimanente = utente->tx;  // setto il tempo di servizio rimanente

    return (1);                // l'utente arrivato è stato gestito
}



char gestisciArrivoUtenteConSistemaVuoto(tipoUtente *utente, unsigned int *x,
     unsigned int *k, double *nUtEntratiServ, double *nUtEntratiSist)
// gestisce l'entrata in servizio di un utente quando il sistema è vuoto
{
    // incremento il contatore degli utenti entrati nel sistema
    (*nUtEntratiSist)++;
    // accumulo il numero di utenti attualmente nel sistema
  accumula_k = accumula_k + *k;
  // accumulo il numero di utenti attualmente in coda
  accumula_q = accumula_q + q;

  (*k)++;                  // arriva l'utente che entra nel sistema
    utente->tx = 0;        // non è ancora stato messo in servizio
    utente->tq = 0;        // non ha atteso in coda perchè il sistema è vuoto
    utente->tk = 0;        // è appena entrato nel sistema
    utente->txRimanente = 0; // non è ancora stato messo in servizio

    // gestisce l'entrata in servizio dell'utente
    return (serviUtente(utente, x, nUtEntratiServ));
}



char gestisciAccodamentoUtenteMM1Y(unsigned int *q, unsigned int *k,
```

```
     double *nUtEntratiCoda, double *nUtEntratiSist, double *nUtentiPersi)
// gestisce l'eventuale (MM1Y) accodamento dell'utente
{
  tipoUtente utDaAccodare;

  // se la coda NON è piena
     if (*q < Y)
    {
      // incremento il contatore del numero totale di utenti entrati nel sistema
      (*nUtEntratiSist)++;
      // incremento il contatore del numero totale di utenti entrati in coda
      (*nUtEntratiCoda)++;
        // accumulo il numero di utenti attualmente nel sistema
    accumula_k = accumula_k + *k;
    // accumulo il numero di utenti attualmente in coda
    accumula_q = accumula_q + *q;

        (*k)++;                // arriva l'utente che entra nel sistema
    (*q)++;                // incremento il numero di utenti in coda
        utDaAccodare.tx = 0;        // non è ancora stato messo in servizio
      utDaAccodare.tq = 0;        // entrerà ora in coda
      utDaAccodare.tk = 0;        // è appena entrato nel sistema
        utDaAccodare.txRimanente = 0;    // non è ancora stato messo in servizio
        inserisciInCoda(utDaAccodare);    // gestisce l'accodamento

        // se la coda è piena aggiorno il contatore di coda piena
        if (*q == Y) occorrenzeCodaPiena++;
    }
    // se la coda è piena
    else
        (*nUtentiPersi)++;// arriva l'utente che trovando la coda piena sarà perso
    return (1);                // l'utente arrivato è stato gestito
}



void gestisciUscitaUtente(tipoUtente utente, unsigned int *x, unsigned int *k,
     double *nUtUscitiServ, double *nUtUscitiSist)
// gestisce l'uscita dell'utente attualmente in servizio
{
    // incremento il contatore del numero totale di utenti usciti dal servizio
    (*nUtUscitiServ)++;
    // incremento il contatore del numero totale di utenti usciti dal sistema
    (*nUtUscitiSist)++;
    // accumulo i tempi di servizio
    accumula_tx = accumula_tx + utente.tx;
    // accumulo i tempi di permanenza nel sistema
    accumula_tk = accumula_tk + utente.tk;

    (*x)--;                 // l'utente esce dal servizio
    (*k)--;                 // ed esce dal sistema
}



void gestisciServiUtenteInCoda(tipoUtente *utente, unsigned int *x,
     unsigned int *q, double *nUtEntratiServ, double *nUtUscitiCoda)
// gestisce l'entrata in servizio del primo utente in coda (FIFO)
{
   // incremento il contatore del numero totale di utenti usciti dalla coda
    (*nUtUscitiCoda)++;
    // accumulo i tempi di attesa in coda
```

```
    accumula_tq = accumula_tq + utente->tq;

    (*q)--;                  // l'utente esce dalla coda
    *utente = prelevaInTesta();        // prelevo il primo utente dalla coda
    // gestisco l'entrata in servizio dell'utente
    serviUtente(utente, x, nUtEntratiServ);
}



void simulazione()
{
  char  arrivato;  // 1 se è arrivato un utente, 0 se NON è arrivato alcun utente

  // arriva il primo utente che trova il sistema vuoto
    arrivato = gestisciArrivoUtenteConSistemaVuoto(&utente, &x, &k,
               &nUtEntratiServ, &nUtEntratiSist);
    nUtentiGenerati = 0;
    while (nUtentiGenerati < NUTENTI)
    {
      // se l'utente è già arrivato
        if (arrivato)
        {
          nUtentiGenerati++;      // genero un arrivo
          // il prossimo utente arriverà fra un tempo tInterarrivo
          tInterarrivo = poisson();
            arrivato = 0;        // poichè NON è ancora arrivato
        }
    // se il sistema è vuoto
        if (k == 0)
        {
          // la simulazione avanza temporalmente fino all'istante di arrivo
      tSimulazione = tSimulazione + tInterarrivo;
            // accumulo il tempo di servitore libero
            tServitoreLibero = tServitoreLibero + tInterarrivo;
            // arriva l'utente che trova il sistema vuoto
      arrivato = gestisciArrivoUtenteConSistemaVuoto(&utente, &x, &k,
                 &nUtEntratiServ, &nUtEntratiSist);
        }
        // se il sistema NON è vuoto
        else
        {
          // se l'utente arriverà prima che termini il servizio di quello
          // attualmente in servizio
          if (tInterarrivo < utente.txRimanente)
          {
              // la simulazione avanza temporalmente fino all'istante di arrivo
                tSimulazione = tSimulazione + tInterarrivo;
                // trascorre il tempo tInterarrivo per l'utente in servizio
                utente.tk = utente.tk + tInterarrivo;
                // aggiorno il tempo di servizio rimanente
      utente.txRimanente = utente.txRimanente - tInterarrivo;
           // trascorre il tempo tInterarrivo per gli EVENTUALI utenti in coda
      aggiorna_tq_tkCoda(tInterarrivo);
                // gestisco l'eventuale (MM1Y) accodamento dell'utente
                arrivato = gestisciAccodamentoUtenteMM1Y(&q, &k, &nUtEntratiCoda,
                           &nUtEntratiSist, &nUtentiPersi);
          }
            // se l'utente arriverà dopo che termini il servizio di quello
            // attualmente in servizio
          else
          {
```

```
               // la simulazione avanza temporalmente fino all'istante di uscita
                tSimulazione = tSimulazione + utente.txRimanente;
                // trascorre il tempo txRimanente per l'utente in servizio
        utente.tk = utente.tk + utente.txRimanente;
        // trascorre il tempo txRimanente per gli EVENTUALI utenti in coda
          aggiorna_tq_tkCoda(utente.txRimanente);
                // trascorre il tempo txRimanente per l'utente in arrivo
                tInterarrivo = tInterarrivo - utente.txRimanente;
                // gestisce l'uscita dell'utente attualmente in servizio
                gestisciUscitaUtente(utente, &x, &k, &nUtUscitiServ,
                                     &nUtUscitiSist);
        // se c'è coda
              if (q > 0)
              {
                 // gestisce l'entrata in servizio del primo utente in coda (FIFO)
                    gestisciServiUtenteInCoda(&utente, &x, &q, &nUtEntratiServ,
                     &nUtUscitiCoda);
              }
          }// se l'utente arriverà dopo che termini il servizio di quello
           // attualmente in servizio
    }// se il sistema NON è vuoto
    }// while (nUtentiGenerati < NUTENTI)
}



void stampaRisultati()
{
  //printf("Fattore di utilizzazione                          rho   = %g
  //\n", lambda / (double)MU);
    fprintf(fileDati, "%g ", lambda / (double)MU);

    if (nUtUscitiCoda != 0)
    {
       //printf("Tempo di attesa in coda medio per utente         E[Tq] = %g
       //\n", MU * accumula_tq / (double)nUtentiGenerati);
    fprintf(fileDati, "%g ", MU * accumula_tq / (double)nUtentiGenerati);
  }
    else
    {
    //printf("Tempo di attesa in coda medio per utente         E[Tq] = %g\n", 0);
    fprintf(fileDati, "%g ", 0);
    }
    //printf("Tempo di permanenza nel sistema medio per utente E[T]  = %g\n", MU *
     accumula_tk / (double)nUtentiGenerati);
    fprintf(fileDati, "%g ", MU * accumula_tk / (double)nUtentiGenerati);
    //printf("Tempo di servizio medio per utente               E[x]  = %g\n", MU *
     accumula_tx / (double)nUtentiGenerati);
    fprintf(fileDati, "%g ", MU * accumula_tx / (double)nUtentiGenerati);

    if (nUtUscitiCoda != 0)
    {
       //printf("Numero medio di utenti in coda                   E[q]  = %g\n",
       // accumula_q / (double)nUtentiGenerati);
      fprintf(fileDati, "%g ", accumula_q / (double)nUtentiGenerati);
  }
    else
    {
    //printf("Numero medio di utenti in coda                   E[q]  = %g\n", 0);
      fprintf(fileDati, "%g ", 0);
    }
    //printf("Numero medio di utenti nel sistema               E[k]  = %g\n",
```

```
    // accumula_k / (double)nUtentiGenerati);
    fprintf(fileDati, "%g ", accumula_k / (double)nUtentiGenerati);

    //printf("Numero medio di utenti persi                    E[l]  = %g\n",
    // nUtentiPersi / (double)nUtentiGenerati);
    fprintf(fileDati, "%g ", nUtentiPersi / (double)nUtentiGenerati);
    //printf("Numero medio di coda piena                      E[f]  = %g\n",
     occorrenzeCodaPiena / (double)nUtentiGenerati);
    fprintf(fileDati, "%g ", occorrenzeCodaPiena / (double)nUtentiGenerati);

    //printf("Tempo medio di servitore libero                 E[Tlib]  = %g\n",
     tServitoreLibero / (double)tSimulazione);
    fprintf(fileDati, "%g ", tServitoreLibero / (double)tSimulazione);
    //printf("Tempo medio di servitore occupato               E[Tocc]  = %g\n",
     (tSimulazione-tServitoreLibero) / (double)tSimulazione);
    fprintf(fileDati, "%g ", (tSimulazione-tServitoreLibero) /
                             (double)tSimulazione);


    // gli andamenti teorici per il solo MM1
    if (nUtUscitiCoda != 0)
    {
       //printf("Tempo di attesa in coda medio per utente teorico        E[Tq] =
       // %g\n", (lambda / (double)MU) / (MU*(1-(lambda / (double)MU))) );
    fprintf(fileDati, "%g ", (lambda / (double)MU) / (MU*(1-(lambda / (double)MU))
    ) );
  }
    else
    {
    //printf("Tempo di attesa in coda medio per utente        E[Tq] = %g\n", 0);
    fprintf(fileDati, "%g ", 0);
    }
    //printf("Tempo di permanenza nel sistema medio per utente E[T]  = %g\n", (1/
    //MU) / (1-(lambda / (double)MU)));
    fprintf(fileDati, "%g ", (1/MU) / (1-(lambda / (double)MU)));
    //printf("Tempo di servizio medio per utente              E[x]  = %g\n", 1/
    //MU);
    fprintf(fileDati, "%g ", 1/MU);

    if (nUtUscitiCoda != 0)
    {
       //printf("Numero medio di utenti in coda                   E[q]  = %g\n",
       // (lambda / (double)MU)*(lambda / (double)MU)/(1-(lambda / (double)MU)));
      fprintf(fileDati, "%g ", (lambda / (double)MU)*(lambda / (double)MU)/(1-
                               (lambda / (double)MU)));
  }
    else
    {
    //printf("Numero medio di utenti in coda                  E[q]  = %g\n", 0);
      fprintf(fileDati, "%g ", 0);
    }
    //printf("Numero medio di utenti nel sistema              E[k]  = %g\n",
     (lambda / (double)MU) / (1-lambda / (double)MU));
    fprintf(fileDati, "%g ", (lambda / (double)MU) / (1-lambda / (double)MU));

    fprintf(fileDati, "\n");
}



main()
{
```

```
  if ( (fileDati = fopen("D:\\TARIN\\MyPrg\\Tesina\\sim.txt", "wt")) == NULL )
    {fprintf(stderr, "Impossibile creare il file dati");exit (-1);}

  for(lambda=0.000001; lambda<0.95*MU; lambda+=INCR_LAMBDA)
  {
    inizializzaSimulazione();
      simulazione();
      stampaRisultati();
  }

  fclose(fileDati);
}
```

Appendice C

GNU Free Documentation License

Preamble

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The **"Document"**, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as

"you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A **"Modified Version"** of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A **"Secondary Section"** is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The **"Invariant Sections"** are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The **"Cover Texts"** are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A **"Transparent"** copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called **"Opaque"**.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The **"Title Page"** means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section **"Entitled XYZ"** means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as **"Acknowledgements"**, **"Dedications"**, **"Endorsements"**, or **"History"**.) To

"Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties–for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright ©YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts." line with this:

> with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.