Z-Iteration: Efficient Estimation of Instantaneous Measures in Time-Dependent Multi-Class Systems

Ibrahim Matta
matta@ccs.neu.edu

College of Computer Science
Northeastern University
Boston, MA 02115

A. Udaya Shankar
shankar@cs.umd.edu

Department of Computer Science
University of Maryland
College Park, MD 20742

June 1997

Abstract

Multiple-class multiple-resource (MCMR) systems, where each class of customers requires a particular set of resources, are common. These systems are often analyzed under steady-state conditions. We describe a simple numerical-analytical method, referred to as *Z-iteration*, to estimate instantaneous (and steady-state) probability measures of time-dependent systems. The key idea is to approximate the relationship between certain instantaneous measures by the relationship between their steady-state counterparts, and use this approximation to solve dynamic flow equations. We show the generality of the Z-iteration by applying it to an integrated communication network, a parallel database server, and a distributed batch system. Validations against exact numerical solutions and discrete-event simulations show the accuracy and computational advantages of the Z-iteration.

Keywords: Multi-class systems, transient performance, dynamic flow models, queueing models, simulation.

Preliminary version appeared in *ACM SIGMETRICS/PERFORMANCE '95.*

1 Introduction

We consider a general multiple-class multiple-resource (MCMR) system. We have a set $\mathcal{R}$ of resources and a set $\mathcal{C}$ of customer classes. The nature of a resource depends on the system being modeled; for example, it may be computer memory, floor space, transmission capacity, etc. Each resource r has an attribute, denoted by $r.max$, which is a constant that indicates the maximum number of units in terms of which r is quantified.

Each class in $\mathcal{C}$ represents a class of customers that requires a particular set of resources. Depending on the system being modeled, customers can be user programs, manufactured products, network connections (calls), etc. Specifically, each class-c customer requires some subset $\mathcal{R}_c$ of resources, $\mathcal{R}_c \subseteq \mathcal{R}$. Furthermore, the class-c customer requires some number of units, denoted by $c.r.req$, of each resource $r \in \mathcal{R}_c$ (e.g. bandwidth, storage space, etc.). For example, a network connection would require some transmission and buffer capacity on each of the links of the path connecting its source to its destination.

Let

- $\lambda_c(t)$ denote the instantaneous arrival rate of class-c customers, and
- $1/\mu_c^r(t)$ denote the instantaneous service time of a class-c customer at r.

Thus we are interested not only in the steady-state behavior of the MCMR system, but also in its transient or non-stationary behavior. Transient conditions arise when the customer arrival rates or the service rates at the resources vary with time, due to externally time-varying factors or dynamic control decisions based on current or delayed system state information.

An arriving class-c customer is blocked at a resource $r \in \mathcal{R}_c$ iff $c.r.req$ exceeds the amount of the resource that is currently available (additional constraints can be incorporated too). An arriving class-c customer is blocked iff it is blocked at any $r \in \mathcal{R}_c$. A blocked customer is lost or retried later.

We want to obtain instantaneous probability measures of the different classes, such as the instantaneous blocking probabilities (or equivalently the throughputs). We introduce the following notation:

- $B_c(t)$, instantaneous blocking probability of class c.
- $B_c^r(t)$, instantaneous blocking probability of class c at resource $r \in \mathcal{R}_c$.
- $N_c^r(t)$, instantaneous average number of class-c customers waiting or in service at resource r.
- $U_c^r(t)$, instantaneous utilization of resource r by class-c customers (average number of class-c customers in service at resource r).

The generality of our model allows us to consider a variety of systems, including those with delayed feedback between changes in system state information and changes in control decisions. Examples of such systems include database locking systems, inventory systems, distributed batch systems, manufacturing systems, and communication networks. Because the class of a customer can be assigned when the customer arrives, it is straightforward to model state-dependent control policies such as assigning jobs to processors with the least workload.

MCMR systems have often been analyzed under steady-state conditions (e.g. [13, 15, 21, 5, 25, 3, 23, 11, 10]). Obtaining instantaneous measures for general time-dependent MCMR systems is another matter. Analytical solutions would seem to be intractable. Discrete-event simulation is a popular alternative, but it is often computationally very expensive, since it requires the averaging of a large number of independent simulation runs to obtain meaningful performance estimates. Straightforward numerical solution of equations such as the Chapman-Kolmogorov differential equations are also computationally very expensive, since it requires solving a set of coupled differential equations equal in number to the size of the state space (which can be very large).

In this paper, we formulate a *numerical-analytical* method, referred to as *Z-iteration*, that accurately estimates the time evolutions of instantaneous probability measures of general time-dependent MCMR systems at a fraction of the cost of discrete-event simulation or straightforward numerical solution. Furthermore, unlike simulation, it is easily parallelizable.

The MCMR system is decomposed into a set of multiple-class single-resource (MCSR) systems by invoking the resource independence assumption. This reduces the complexity of the problem, but a straightforward numerical solution of individual MCSR systems is still too expensive.

To solve an MCSR system, we use a standard flow equation (e.g. [6]) that expresses the time evolution of the instantaneous average number of customers $N_c^r(t)$ by a differential equation involving the instantaneous arrival and service rates (which are known), and the instantaneous blocking probabilities $B_c^r(t)$ and utilizations $U_c^r(t)$. The key idea is to express $B_c^r(t)$ and $U_c^r(t)$ in terms of $N_c^r(t)$, thereby resulting in a single differential equation in $N_c^r(t)$ that can be solved inexpensively.

We have discovered that the function of $B_c^r(t)$ in terms of $N_c^r(t)$ is *very well approximated* by the corresponding "steady-state" function of steady-state B_c^r in terms of steady-state N_c^r, i.e. assuming constant arrival and service rates. The same is true for $U_c^r(t)$, and indeed appears to hold for any instantaneous probability measure.

The desired steady-state functions are computed as the fixed-point of two steady-state expressions: (1) an expression for the steady-state B_c^r in terms of the steady-state offered loads λ_c/μ_c^r; and (2) an expression for the steady-state λ_c/μ_c^r in terms of the steady-state N_c^r and B_c^r. The λ_c/μ_c^r is an intermediate quantity for the fixed-point iteration.

These two steady-state expressions are available for a variety of MCSR systems, including self-service systems where the customer is also the server, and single- or multiple-server queueing systems [18, 6]. We point out that the expressions do not have to be closed form and can be implicit.

The rest of the paper is organized as follows. Section 2 describes the Z-iteration for single-class single-resource systems, and validates against exact solutions for $M/M/K/K$ and $M/M/1/K$ systems. Section 3 presents the method for the general MCMR model. In Sections 4, 5, 6, we apply the Z-iteration to three specific systems with time-varying inputs and dynamic control, namely, an integrated communication network, a parallel database server, and a distributed batch system. The first and third systems are modeled as systems with self-service resources, for which validations against discrete-event simulations are given in Section 7. The second system is modeled as a system with single-server resources, for which validations are given in Section 8. Section 9 concludes and describes related work.

2 The Z-Iteration for Single-Class Single-Resource Systems

To illustrate the basic ideas behind our solution method, let us consider a system with only one customer class and one resource. We will use the notation introduced in the Introduction but without any resource superscript r or class subscript c.

In general, obtaining the instantaneous probability measures is analytically intractable [28] and numerically very expensive due usually to the large state space [27]. For Markovian systems, the straightforward solution involves the well-known Chapman-Kolmogorov (C-K) differential equations, that is,

$$\frac{d\mathbf{P}(t)}{dt} = \mathbf{P}(t)\ \mathbf{Q}(\lambda(t),\ \mu(t))$$

where the row vector $\mathbf{P}(t)$ represents the instantaneous state probability vector and $\mathbf{Q}(.)$ is the generator matrix whose elements represent the transition rates between the states.

The time evolution of $N(t)$ is described by the flow equation

$$\frac{dN(t)}{dt} = \lambda(t)\ [1 - B(t)] - \mu(t)U(t)$$

or its corresponding difference equation (with δ being a small time step):

$$N(t+\delta) = N(t) - \mu(t)\ U(t)\ \delta + \lambda(t)\ \delta\ [1 - B(t)]$$

The second term in the right-hand side of the equation represents the average number of customers which leave during $[t,\ t+\delta)$. The third term represents the average number of new customers that are admitted during $[t,\ t+\delta)$.

Observe that if we could express $B(t)$ and $U(t)$ in terms of $N(t)$, then we could solve the equation inexpensively for the time evolution of these measures. Of course, obtaining such expressions exactly is also intractable. *However it turns out that the function of $B(t)$ in terms of $N(t)$ is very well approximated by the function of steady-state B in terms of steady-state N*, i.e. assuming that $\lambda(t)$ and $\mu(t)$ are constants. *The same holds for $U(t)$, and indeed appears to be true for any instantaneous state probability.*

The steady-state functions of B and U in terms of N are available in analytical closed-form only for very simple systems, e.g. $M/M/2/2$, $M/M/1/2$, and $M/M/1/\infty$. They are not available for blocking systems in general. We obtain them numerically using another approximation involving three steady-state expressions, namely:

$$U \text{ in terms of } N \textit{ assuming} \text{ a nonblocking system} \tag{1}$$

$$B \text{ in terms of steady-state offered load } \frac{\lambda}{\mu} \tag{2}$$

$$\frac{\lambda}{\mu} = \frac{U}{[1 - B]} \tag{3}$$

Expressions (1) and (2) are readily available in the literature. Equation (3) follows from equating the flow in, $\lambda[1-B]$, to the flow out, μU. We obtain U in terms of N from equation (1). We obtain B in terms of N as the fixed-point of equations (2) and (3), where N is fixed and U is replaced by the expression in (1).

For the self-service $M/M/K/K$ system, the equations corresponding to (1, 2, 3) are [18]:

$$
\begin{aligned}
U &= N \\
B &= \frac{\rho^K/K!}{\sum_{j=0}^{K}\rho^j/j!} \\
\rho &= \frac{N}{[1-B]}
\end{aligned}
$$

where $\rho = \lambda/\mu$. Note that the first equation, which was obtained assuming nonblocking ($K = \infty$), happens to be also valid with blocking.

For the single-server $M/M/1/K$ system, the equations corresponding to (1, 2, 3) are [18]:

$$
\begin{aligned}
U &= \frac{N}{1+N} \\
B &= \frac{\rho^K}{\sum_{j=0}^{K}\rho^j} \\
\rho &= \frac{N}{(1+N)}\frac{1}{[1-B]}
\end{aligned}
$$

The first equation follows from $U = \rho$ and $N = \rho/(1-\rho)$. Note that it is *not* valid for the blocking case.

In the following subsections, we illustrate the accuracy of the Z-iteration and its approximations for the $M/M/K/K$ and the $M/M/1/K$ systems.

2.1 $M/M/2/2$

For this system, we can compute closed-form expressions. We have

$$B = \frac{\rho^2/2}{1+\rho+\rho^2/2} \tag{4}$$

$$N = \frac{\rho+\rho^2}{1+\rho+\rho^2/2} \tag{5}$$

Inverting equation (5) (and ignoring a root that is always negative), we get

$$\rho = \frac{N-1+\sqrt{1+2N-N^2}}{2-N} \tag{6}$$

Substituting (6) in (4), we get B in terms of N.

To illustrate the accuracy of approximating the instantaneous relationship between $B(t)$ and $N(t)$ by the steady-state relationship between B and N, we consider the evolution for $\lambda = \mu = 1$ and $t \in [0, 6]$ starting with an empty system at $t = 0$. Let $B_{\text{exact}}(t)$ and $N_{\text{exact}}(t)$ denote the exact solutions; they are derived in the Appendix. Figure 1 shows a plot comparing $B_{\text{exact}}(t)$ and $B\mid_{N=N_{\text{exact}}(t)}$ (i.e. steady-state expression for B with N replaced by the exact $N(t)$). Clearly, this shows that the approximation is quite good. In general, analyzing the errors is hard as it involves complex nonlinear equations in the average number of customers.

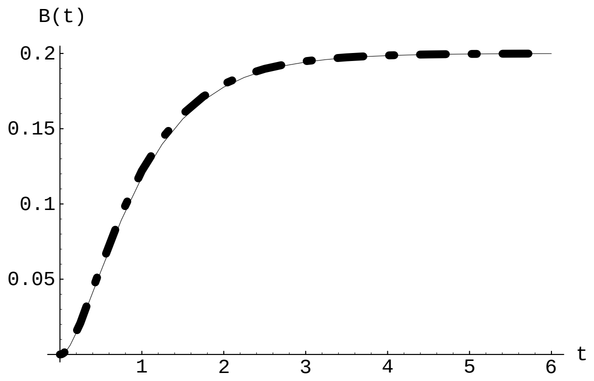

Figure 1: Accuracy of the approximation for the $M/M/2/2$ system. The dot-dashed line is $B_{\mathrm{exact}}(t)$, and the solid line is the approximation. $\lambda = \mu = 1$ for $t \geq 0$. System empty at $t = 0$.

For a general system, it is not possible to invert the equation corresponding to equation (5), which is why we are forced to resort to the fixed-point iteration. To illustrate the convergence of this iteration, we apply it to the $M/M/2/2$ case. In this case, the two equations are

$$\begin{aligned} B &= \frac{\rho^2/2}{1+\rho+\rho^2/2} \\ \rho &= \frac{N}{[1-B]} \end{aligned}$$

These two formulas define an equation of the form $B = F(B)$. Figure 2 shows a graphical example of the mapping F. It illustrates that F is a contractive mapping of $[0,\ 1)$ into $[0,\ 1)$ and hence it converges to a unique fixed point [16].

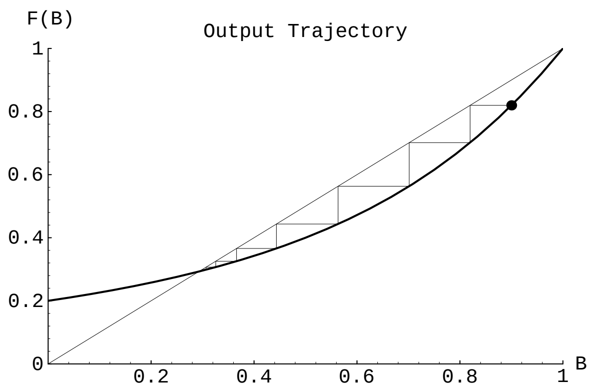

Figure 2: Convergence of the iteration for the $M/M/2/2$ system starting from $B = 0.9$ for $N = 1$.

2.2 $M/M/50/50$

Consider an $M/M/50/50$ system with varying λ and μ. Figure 3 shows the average number of customers in the system obtained by the Z-iteration and by the numerical solution of the C-K differential equations. Both solutions practically coincide illustrating the accuracy of our method. Both were coded in *Mathematica* using the differential equation solver `NDSolve` [29]. On a DEC OSF/1 Alpha workstation, the C-K solution required 1207 seconds of execution time, while our method required only 122 seconds. This is a significant savings in computation.

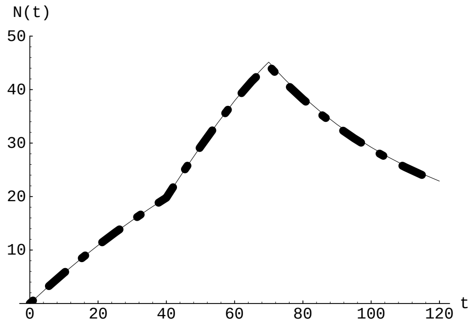

Figure 3: Comparison between the Z-iteration (solid) and the C-K solution (dot-dashed) for the $M/M/50/50$ system. $N(0) = 0$; $\lambda = 0.6$ for $t \in [0, 40]$, $\lambda = 1.2$ for $t \in (40, 70]$, and $\lambda = 0.2$ for $t \in (70, 120]$; $\mu = 0.01$ for $t \in [0, 70]$, and $\mu = 0.02$ for $t \in (70, 120]$.

2.3 $M/M/1/50$

Consider an $M/M/1/50$ system with varying λ and μ. Figure 4 shows the average number of customers in the system obtained by the Z-iteration and by the numerical solution of the C-K differential equations. Both solutions are very close. Both were coded in *Mathematica*. On a DEC OSF/1 Alpha workstation, the C-K solution required 1522 seconds of execution time, while our method required only 133 seconds. Again, this is a significant savings in computation.

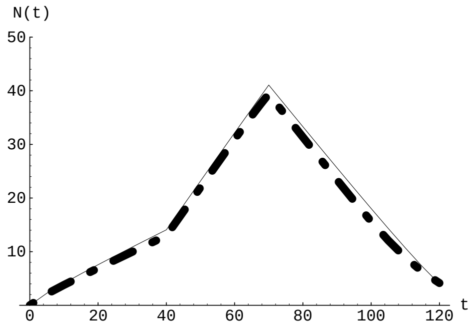

Figure 4: Comparison between the Z-iteration (solid) and the C-K solution (dot-dashed) for the $M/M/1/50$ system. $N(0) = 0$; $\lambda = 0.6$ for $t \in [0, 40]$, $\lambda = 1.2$ for $t \in (40, 70]$, and $\lambda = 0.2$ for $t \in (70, 120]$; $\mu = 0.3$ for $t \in [0, 70]$, and $\mu = 1$ for $t \in (70, 120]$.

3 The $\mathcal{Z}$-Iteration for Multiple-Class Multiple-Resource Systems

In this section, we present our solution method to the general MCMR model introduced in Section 1. As mentioned in Section 2, obtaining the instantaneous probability measures is analytically intractable and numerically very expensive. For Markovian systems, it involves the C-K differential equations for every resource r,

$$\frac{d\mathbf{P}^r(t)}{dt} = \mathbf{P}^r(t)\ \mathbf{Q}^r(\{\lambda_c(t),\ \mu_c^r(t) : c \in \mathcal{C}^r\})$$

where the row vector $\mathbf{P}^r(t)$ represents the instantaneous state probability vector of r, $\mathbf{Q}^r(.)$ is the generator matrix whose elements represent the transition rates between the states of r, and $\mathcal{C}^r$ denotes the set of classes requesting units of resource r.

We start with the following difference equations for $c \in \mathcal{C}^r$, where δ is the (small) time step:

$$N_c^r(t+\delta) = N_c^r(t) - \mu_c^r(t)\ U_c^r(t)\ \delta + \lambda_c(t)\ \delta \prod_{r' \in \mathcal{R}_c} [1 - B_c^{r'}(t)] \tag{7}$$

Note that the product term $\prod$ reflects the assumption made in Section 1 that a new class-c customer is admitted iff it is not blocked at any of the required $\mathcal{R}_c$ resources; this invokes the resource independence assumption.

By expressing $U_c^r(t)$ and $B_c^r(t)$ in terms of $\{N_{c'}^r(t)\ :\ c' \in \mathcal{C}^r\}$, we can solve equations (7) inexpensively. We approximate the functions of $U_c^r(t)$ in terms of $N_{c'}^r(t)$ and $B_c^r(t)$ in terms of $N_{c'}^r(t)$ by the corresponding functions assuming steady-state, i.e., assuming that the $\lambda_{c'}(t)$ and $\mu_{c'}^r(t)$ are constants.

The details of obtaining these steady-state functions are in the following subsections. We first obtain a steady-state relationship between U_c^r and the $N_{c'}^r$. We then obtain two steady-state expressions, one defining B_c^r in terms of the $\frac{\lambda_{c'}}{\mu_{c'}^r}$, and one defining $\frac{\lambda_c}{\mu_c^r}$ in terms of the $N_{c'}^r$ and B_c^r. By iterating to the fixed point of these two expressions with $N_{c'}^r$ fixed, we obtain the desired steady-state relationship between B_c^r and the $N_{c'}^r$.

3.1 Utilization in terms of numbers of customers

Denoting the steady-state expression by $\mathcal{T}_c^r$, we have for $c \in \mathcal{C}^r$:

$$U_c^r = \mathcal{T}_c^r(\{N_{c'}^r\ :\ c' \in \mathcal{C}^r\}) \tag{8}$$

$\mathcal{T}_c^r$ is a function that reflects the load and service discipline of r. The exact form of $\mathcal{T}_c^r$ is application dependent. For a self-service facility as in an $M/G/K/K$ queueing system, $\mathcal{T}_c^r$ is clearly equal to N_c^r. The derivation of $\mathcal{T}_c^r$ is not always obvious. One approximation to obtain $\mathcal{T}_c^r$ in a systematic and easy way is to assume no blocking and then use steady-state queueing formulas expressing N_c^r in terms of the $\frac{\lambda_{c'}}{\mu_{c'}^r}$. Inverting these formulas, we obtain $\frac{\lambda_c}{\mu_c^r}$ in terms of the $N_{c'}^r$. Since we are assuming no blocking, we have $\mathcal{T}_c^r = \frac{\lambda_c}{\mu_c^r}$. Thus, we get $\mathcal{T}_c^r$ in terms of the $N_{c'}^r$. (cf. Sections 2 and 5.)

From equations (8), we can express $U_c^r(t)$ approximately in terms of $\{N_{c'}^r(t)\ :\ c' \in \mathcal{C}^r\}$ by replacing the steady-state measures U_c^r and N_c^r by their instantaneous counterparts $U_c^r(t)$ and $N_c^r(t)$. Doing this yields the following instantaneous equations for $c \in \mathcal{C}^r$:

$$U_c^r(t) = \mathcal{T}_c^r(\{N_{c'}^r(t)\ :\ c' \in \mathcal{C}^r\}) \tag{9}$$

3.2 Blocking probabilities in terms of numbers of customers

Denoting by $\mathcal{S}_c^r$ the steady-state expression defining B_c^r in terms of the $\frac{\lambda_{c'}}{\mu_{c'}^r}$, we have for $c \in \mathcal{C}^r$:

$$B_c^r = \mathcal{S}_c^r(\{\frac{\lambda_{c'}}{\mu_{c'}^r} : c' \in \mathcal{C}^r\}) \tag{10}$$

$\mathcal{S}_c^r$ can be obtained as follows. Define a *feasible state* of resource r by the number of customers of each class $c \in \mathcal{C}^r$ that r can simultaneously support, i.e. for which the total number of units requested does not exceed $r.max$. Let $\mathcal{F}^r$ denote the set of all feasible states of r. Assuming the $\lambda_{c'}(t)$ and $\mu_{c'}^r(t)$ are constants for all t, the steady-state transition rate between two states belonging to $\mathcal{F}^r$ is given by some function of $\lambda_{c'}$ and $\mu_{c'}^r$. A class-c customer is blocked in a state of $\mathcal{F}^r$ if its admittance would lead to a state outside $\mathcal{F}^r$. Refer to such states of $\mathcal{F}^r$ as class-c blocking states. Solving analytically for the probability of being in a class-c blocking state yields $\mathcal{S}_c^r$.

To illustrate, consider an $M/G/K/K$ resource used by one class of customers arriving according to a Poisson process of rate λ_c. Let each admitted customer be served by one of the K servers for an average duration of $1/\mu_c^r$. Then $\mathcal{S}_c^r$ is the Erlang-B formula, i.e. $\mathcal{S}_c^r = E(\frac{\lambda_c}{\mu_c^r}, K) = \frac{(\frac{\lambda_c}{\mu_c^r})^K/K!}{\sum_{j=0}^{K}(\frac{\lambda_c}{\mu_c^r})^j/j!}$ [18].

We obtain the steady-state expression defining $\frac{\lambda_c}{\mu_c^r}$ in terms of the $N_{c'}^r$ and B_c^r as follows. Equating the rates of departure and admission of class-c customers at resource r, we have $\mu_c^r\ U_c^r = \lambda_c\,[1 - B_c^r]$. From this and (8) we have for $c \in \mathcal{C}^r$:

$$\frac{\lambda_c}{\mu_c^r} = \frac{\mathcal{T}_c^r(\{N_{c'}^r : c' \in \mathcal{C}^r\})}{[1 - B_c^r]} \tag{11}$$

From equations (10) and (11), we can express $B_c^r(t)$ approximately in terms of $\{N_{c'}^r(t) : c' \in \mathcal{C}^r\}$ by replacing the steady-state measures B_c^r and N_c^r by their instantaneous counterparts $B_c^r(t)$ and $N_c^r(t)$, and replacing $\frac{\lambda_c}{\mu_c^r}$ by an instantaneous quantity $z_c^r(t)$ that we introduce. Doing this yields the following instantaneous equations for $c \in \mathcal{C}^r$:

$$B_c^r(t) = \mathcal{S}_c^r(\{z_{c'}^r(t) : c' \in \mathcal{C}^r\}) \tag{12}$$

$$z_c^r(t) = \frac{\mathcal{T}_c^r(\{N_{c'}^r(t) : c' \in \mathcal{C}^r\})}{[1 - B_c^r(t)]} \tag{13}$$

3.3 Overall calculation

Knowing $\{N_c^r(t)\colon c \in \mathcal{C}^r\}$ at some fixed t, we can solve equations (12) and (13) iteratively for $\{B_c^r(t)\colon c \in \mathcal{C}^r\}$. In particular, starting from an initial estimate $\{\hat{z}_c^r(t)\colon c \in \mathcal{C}^r\}$, we compute $\{B_c^r(t)\colon c \in \mathcal{C}^r\}$ from equations (12). Then, we use equations (13) to compute new values for $\{z_c^r(t)\colon c \in \mathcal{C}^r\}$. We repeat this process until the values of $\{z_c^r(t)\colon c \in \mathcal{C}^r\}$ stabilize.

Once we obtain $\{B_c^r(t) : r \in \mathcal{R},\ c \in \mathcal{C}^r\}$, we obtain $\{N_c^r(t+\delta) : r \in \mathcal{R},\ c \in \mathcal{C}^r\}$ using equations (7), and we repeat the process to obtain the time evolution of the performance measures for time instants $0,\ \delta,\ 2\delta,\ \cdots$.

Figure 5 outlines our solution method. In the outermost iteration, we obtain $\{N_c^r(t+\delta),\ B_c^r(t) : r \in \mathcal{R},\ c \in \mathcal{C}^r\}$ for $t = 0,\ \delta,\ 2\delta,\ \cdots$. The computation for each time t consists of two parts. The first part (steps 3-9) computes, for every $r \in \mathcal{R}$, $\{B_c^r(t)\colon c \in \mathcal{C}^r\}$ in terms of $\{N_c^r(t)\colon c \in \mathcal{C}^r\}$. The second part (step 10) computes, for every $r \in \mathcal{R}$ and $c \in \mathcal{C}^r$, $N_c^r(t+\delta)$ in terms of $\{N_{c'}^r(t)\colon c' \in \mathcal{C}^r\}$, $\lambda_c(t)$, $\mu_c^r(t)$, and $\{B_c^{r'}(t)\colon r' \in \mathcal{R}_c\}$. The first part involves the iterative procedure (steps 5-9) on equations (12) and (13) (steps 7 and 8).

1. Initialize $\{N_c^r(0) : r \in \mathcal{R},\ c \in \mathcal{C}^r\}$ /* 0 for initially empty system */
2. For $t = 0,\ \delta,\ 2\delta,\ \cdots$
   begin
3. For every $r \in \mathcal{R}$ /* Obtain $\{B_c^r(t) : c \in \mathcal{C}^r\}$ in terms of $\{N_c^r(t) : c \in \mathcal{C}^r\}$ */
   begin
4. Initialize $\{\hat{z}_c^r(t) : c \in \mathcal{C}^r\}$ /* arbitrary value if $t = 0$ */
   /* $\hat{z}_c^r(t - \delta)$ if $t > 0$ */
5. repeat
6. $z_c^r(t) \leftarrow \hat{z}_c^r(t)$, for every $c \in \mathcal{C}^r$
7. Obtain $\{B_c^r(t) : c \in \mathcal{C}^r\}$ in terms of $\{z_c^r(t) : c \in \mathcal{C}^r\}$
   using an instantaneous version of a steady-state formula (see (12))
8. Obtain $\{\hat{z}_c^r(t) : c \in \mathcal{C}^r\}$ in terms of $\{B_c^r(t), N_c^r(t) : c \in \mathcal{C}^r\}$
   using an instantaneous version of a steady-state formula (see (13))
9. until $\mid \hat{z}_c^r(t) - z_c^r(t) \mid < \epsilon$, for every $c \in \mathcal{C}^r$
   end
10. For every $r \in \mathcal{R}$ and $c \in \mathcal{C}^r$,
   obtain $N_c^r(t + \delta)$ in terms of $\{N_{c'}^r(t) : c' \in \mathcal{C}^r\}$, $\lambda_c(t)$, $\mu_c^r(t)$, and $\{B_c^{r'}(t) : r' \in \mathcal{R}_c\}$
   using a difference equation relating arrivals and departures (see (7))
   end

Figure 5: Evaluation method.

3.4 Comments

Assuming that L iterations are needed for convergence of the iterative procedure in steps 5-9 of Figure 5, the computational complexity for each time step is $O(|\mathcal{R}|\ |\mathcal{C}^r|\ (\ (|B_c^r| + |z_c^r|)L + |N_c^r|\ ))$, where $|B_c^r|$ is the cost of evaluating $B_c^r(.)$ via (12), $|z_c^r|$ that of evaluating $z_c^r(.)$ via (13), and $|N_c^r|$ that of evaluating $N_c^r(.)$ via (7). The Z-iteration requires storage of $O(V\ |\mathcal{R}|\ |\mathcal{C}^r|)$, where V is the number of instantaneous measures. From Figure 5, we have $V = 5$ since we have 5 instantaneous measures defined, namely, $B_c^r(.)$, $z_c^r(.)$, $N_c^r(.)$, $\lambda_c(.)$ and $\mu_c^r(.)$.

We note that it might be required to make assumptions about the arrival or service distributions in order to obtain the $\mathcal{S}_c^r(.)$ and $\mathcal{T}_c^r(.)$ formulas.

Above we defined the feasible state of resource r by a multi-dimension vector representing the number of customers of each class $c \in \mathcal{C}^r$ that r can simultaneously support. In fact, we can define a feasible state differently as long as in this state, the total number of units requested does not exceed $r.max$. For example, we can define it by a single number representing the total number of units of r currently used by customers. Also, other criteria can used to further limit admission of customers.

The Z-iteration can also be used to directly solve for steady-state, if the $\lambda_c(t)$ and $\mu_c^r(t)$ are constants and a solution exists. We simply set $\frac{N_c^r(t+\delta)-N_c^r(t)}{\delta} = 0$ in equations (7) and use them in conjunction with equations (12) and (13) to iteratively solve for steady-state. There is no simple way to determine whether there exists a solution to such a nonlinear system. Even if the physical nature of the system suggests that a solution exists, the iteration may oscillate between different solutions, which can alert one to the instability of the system [3]. Obviously, such oscillations can also occur under transient conditions arising, for example, from dynamic control.

Observe that it is easy to realize parallel implementations of our method by mapping the computations for different resources onto different processors, and we would expect almost linear speedup.

The accuracy of our method depends on the approximation of the relationship between the $B_c^r(t)$ and the $N_c^r(t)$ by its steady-state counterpart, which is the fixed point of the iteration in steps 5-9 of Figure 5. Analyzing the errors and convergence of this iteration is hard in general because of the complex nature of the underlying nonlinear system. However, it can be shown in simple situations that the approximation is accurate when compared to the exact instantaneous solution, and that the iteration is a contractive mapping of $[0,\ 1)$ into $[0,\ 1)$ and hence it converges to a unique fixed point [16] (cf. Section 2). Furthermore, our experience indicates that our method yields accurate performance measures when compared to discrete-event simulation and that the iteration converges quickly (see Sections 7 and 8).

As we pointed out earlier, the exact form of $\mathcal{S}_c^r(.)$ and $\mathcal{T}_c^r(.)$ and the values of $\mu_c^r(.)$ and $\lambda_c(.)$ depend on the particular application. We next consider different applications, and show how they fit into the general model and solution procedure so far introduced.

4 Application: Integrated Network

Consider an integrated services network (e.g. ATM) that uses dynamic routing to support real-time communication (voice, video, etc.) between pairs of source and destination nodes. The connections of a service s arrive at the service's source node according to a Poisson process of rate λ_s, and have an end-to-end quality-of-service requirement D_s (e.g. delay). Each connection, once it is successfully setup, has a lifetime of average duration $\frac{1}{\mu_s}$. The source node uses its routing information to choose for the arriving connection a potential path/route to the service's destination node. The route and service together define the class of the connection. Note that because of the dynamic routing, class arrivals have time-varying statistics irrespective of whether the service arrivals have time-varying statistics. (See Figure 6.)

Resources in a network include link bandwidths, buffer spaces, etc. For this example, we assume link bandwidths are the main resources; thus $\mathcal{R}$ consists of link ids (where each id denotes the bandwidth component of the link). We assume a connection of service s requires the reservation of a certain amount of bandwidth on each link along its route that are enough to satisfy D_s. This reservation amount can be thought of as either the peak transmission rate of the connection or its "effective bandwidth" [12] varying between its peak and average transmission rates. The set $\mathcal{R}_c$ of a class-c connection would thus contain the links along the route of class c. The instantaneous arrival rate of class-c connections of service s, $\lambda_c(t)$, is a function of λ_s and the routing algorithm. Consider a routing scheme that regularly assigns probabilities to the candidate paths according to their measured loads. Arriving connections are routed independently according to these path probabilities. In this case, class-c connections of service s arrive according to a Poisson process of rate $\lambda_c(t) = \alpha_c(t)\ \lambda_s$, where $\alpha_c(t)$ is the load-dependent routing probability.

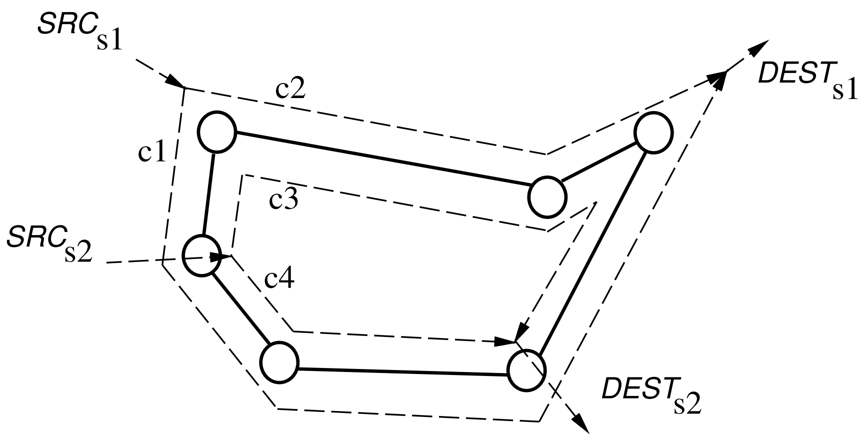

Figure 6: A network offering two services s1 and s2. Each service has two candidate routes for connection setup. Hence two classes are defined for each service: classes c1 and c2 for s1 connections, and classes c3 and c4 for s2 connections.

An arriving class-c connection of service s that finds insufficient bandwidth on any $r \in \mathcal{R}_c$ is blocked and lost. Otherwise, the connection is admitted and bandwidths are allocated to it on each $r \in \mathcal{R}_c$ for an average duration of $\frac{1}{\mu_c^r(t)} = \frac{1}{\mu_s}$. Note that this is a self-service system.

Thus, $r.max$ is the total link bandwidth of r, and $c.r.req$ is the amount of link bandwidth that must be allocated (reserved) for a class-c connection on $r \in \mathcal{R}_c$. Let's assume that the $c.r.req$ and $r.max$ are integers. Let the state of r indicate the amount of bandwidth allocated. Thus, $\mathcal{F}^r = \{0, 1, \cdots, r.max\}$. Let $P^r(j)$ denote the steady-state probability of r being in state j. Then the $P^r(.)$ satisfy the following recurrence relation [25]:

$$j \; P^r(j) = \sum_{c' \in \mathcal{C}^r} \frac{\lambda_{c'}}{\mu_{c'}^r} \, c'.r.req \; P^r(j - c'.r.req) \qquad j = 1, \ldots, r.max$$

where $\sum_{j=0}^{r.max} P^r(j) = 1$.

The steady-state blocking probability for class-c connections at r, B_c^r, is given by

$$B_c^r = \sum_{j=r.max-c.r.req+1}^{r.max} P^r(j)$$

This steady-state solution, which defines $\mathcal{S}_c^r(.)$ for this system, is valid for Poisson arrivals and general service times. It can be used in equations (12) after replacing the $\frac{\lambda_{c'}}{\mu_{c'}^r}$ by $z_{c'}^r(t)$.

Regarding the function $\mathcal{T}_c^r(.)$ used in equations (13), since r is self-service, we have

$$\mathcal{T}_c^r(.) = N_c^r(t)$$

Note that how and when the $\lambda_c(t)$ are varied with time allows one to model different routing algorithms and routing update synchronization at the network nodes. Also, the choice of blocking states in $\mathcal{F}^r$ can model various admission control schemes, in particular those which block connections even if their admission is feasible.

Systems with self-service resources are validated (against discrete-event simulations) in Section 7. There we consider systems equivalent to single-link network, and multi-link network. The multi-link network is used by several multi-hop connections representing main traffic, and several one-hop connections representing cross-traffic.

5 Application: Parallel Database Server

Consider a system of multiple disks on which data is partitioned according to some scheme, e.g. round-robin, range partitioning, etc. [4]. Each disk has a finite first-come-first-served (FCFS) queue where queries of different classes wait to be served. A query requests data retrieval from one or more disks in parallel. (See Figure 7.) This parallelism typically leads to reduction in data access time [4, 14]. The collection of disks needed by a query is defined by the query's class. We assume an arriving query requires one unit of space in the queue of each disk it needs to access.

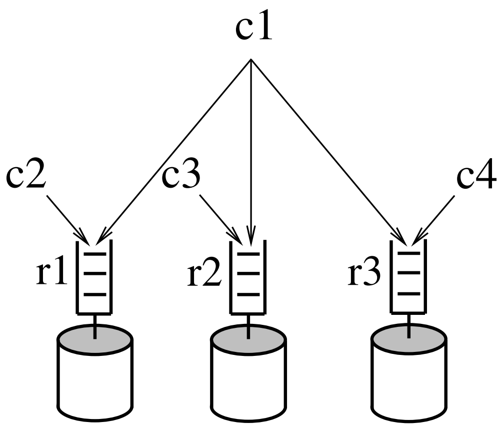

Figure 7: A 3-disk parallel database server. Class c1 requires data retrieval from all three disks. Other classes require data retrieval from only one disk.

Thus the resource set $\mathcal{R}_c$ of a class-c query contains the queues of disks that are needed by class c, and this is a function of the data partitioning scheme. $r.max$ is the total number of queries that r can accommodate, and $c.r.req = 1$ for $r \in \mathcal{R}$ and $c \in \mathcal{C}^r$. An arriving class-c query that finds no space in any $r \in \mathcal{R}_c$ is blocked and lost.

Assume class-c queries arrive according to a Poisson process of rate $\lambda_c(t)$. Also, assume that the service time of any query in r is exponentially distributed with mean $\frac{1}{\mu^r}$; thus $\frac{1}{\mu_c^r(t)} = \frac{1}{\mu^r}$ for all $c \in \mathcal{C}^r$.

Let the state of r denote the total number of queries waiting or in service in r. Thus, $\mathcal{F}^r = \{0, 1, \ldots, r.max\}$. The steady-state blocking probability for class-c queries at r is the steady-state probability of r being in state $r.max$. This steady-state solution is well-known for the $M/M/1/r.max$ queueing system, in particular, for $c \in \mathcal{C}^r$:

$$B_c^r = \frac{\left(\frac{\sum_{c' \in \mathcal{C}^r} \lambda_{c'}}{\mu^r}\right)^{r.max}}{\sum_{j=0}^{r.max} \left(\frac{\sum_{c' \in \mathcal{C}^r} \lambda_{c'}}{\mu^r}\right)^j} \quad [18]$$

This steady-state solution can be used in equations (12) after replacing $\frac{\sum_{c' \in \mathcal{C}^r} \lambda_{c'}}{\mu^r}$ by $\sum_{c' \in \mathcal{C}^r} z_{c'}^r(t)$.

We employ the technique explained in Sections 2 and 3 to derive the function $\mathcal{T}_c^r(.)$ used in equations (13). Assuming steady-state and no blocking, we can treat the $M/M/1/r.max$ system of r as an $M/M/1/\infty$ system. At steady-state, we know that [18]

$$N_c^r = \frac{\lambda_c}{\mu^r - \sum_{c' \in \mathcal{C}^r} \lambda_{c'}} \tag{14}$$

From this and $\mathcal{T}_c^r(.) = \frac{\lambda_c}{\mu^r}$, which holds assuming no blocking, we have[1]

$$\mathcal{T}_c^r(.) = \frac{N_c^r}{1 + \sum_{c' \in \mathcal{C}^r} N_{c'}^r}$$

Therefore, in the transient regime, we have

$$\mathcal{T}_c^r(.) = \frac{N_c^r(t)}{1 + \sum_{c' \in \mathcal{C}^r} N_{c'}^r(t)}$$

The above model can be used to study various data partitioning schemes for high-performance indexing [4]. Systems with single-server resources are validated (against discrete-event simulations) in Section 8.

6 Application: Distributed Batch System

Consider a distributed batch system such as Condor [20]. Batch jobs (user programs) are submitted to a central manager (CM). Assume batch jobs of type i arrive to the CM according to a Poisson process of rate λ_i. The CM uses its information about the load on the various workstations to choose for the arriving batch job a potential workstation for its execution. The class of the batch job is defined by the workstation it is routed to by the CM and the job type.

Each batch job would typically require resources such as memory, disk space, and CPU processing power to execute on a workstation. For this example, we assume all required resources other than the CPU are always available. The set $\mathcal{R}_c$ of a class-c batch job would thus contain the CPU of the workstation to which the job is routed.

We assume only one job can be running on each workstation at a time. Thus, if the owner of the workstation executes a job of his/her own, then the batch job currently executing on his/her workstation, if any, is suspended and its execution resumed later when the owner job finishes execution. An arriving class-c batch job that finds another batch job running or suspended on $r \in \mathcal{R}_c$ is blocked and returned to the CM. Otherwise, it is admitted for processing with mean processing time of $1/\mu_c^r(t)$. This processing time includes the time during which the batch job is suspended due to owner processes [19]. Note that in this application, we do not assume that blocked jobs are lost, rather they are returned to the CM for retry.

[1] From (14), we have (i) $N_c^r = \frac{\lambda_c/\mu^r}{1 - \sum_{c' \in \mathcal{C}^r} \lambda_{c'}/\mu^r}$, and thus (ii) $\sum_{c' \in \mathcal{C}^r} N_{c'}^r = \frac{\sum_{c' \in \mathcal{C}^r} \lambda_{c'}/\mu^r}{1 - \sum_{c' \in \mathcal{C}^r} \lambda_{c'}/\mu^r}$. Rearranging the last equation, we have (iii) $\sum_{c' \in \mathcal{C}^r} \lambda_{c'}/\mu^r = \frac{\sum_{c' \in \mathcal{C}^r} N_{c'}^r}{1 + \sum_{c' \in \mathcal{C}^r} N_{c'}^r}$. Substituting (iii) in (i), we get an expression for $\frac{\lambda_c}{\mu^r}$, which together with $\mathcal{T}_c^r(.) = \frac{\lambda_c}{\mu^r}$ yields the desired result.

The instantaneous arrival rate of class-c batch jobs of type i, $\lambda_c(t)$, is a function of λ_i, the load balancing algorithm used by the CM, and the rate of retrials of type i batch jobs. Assume the load balancing algorithm regularly assigns to the candidate workstations probabilities according to their measured loads. Arriving batch jobs are routed independently according to these probabilities. Let $\alpha_c(t)$ denote the load-dependent probability that the type i batch job belongs to class c, i.e. is routed to $r \in \mathcal{R}_c$. Then,

$$\lambda_c(t) \;=\; [\lambda_i + \sum_{\substack{\text{classes } c' \text{ of type } i \\ r' \in \mathcal{R}_{c'}}} \lambda_{c'}(t-\delta)\; B_{c'}^{r'}(t-\delta)]\; \alpha_c(t) \tag{15}$$

The $\sum$ term in equation (15) represents the total rate of retrials of type i batch jobs, which is a function of their blocking probabilities. In this model, $r.max$ is the maximum number of batch jobs that r handles. $r.max = 1$ and $c.r.req = 1$ for $r \in \mathcal{R}$ and $c \in \mathcal{C}^r$. Let the state of r denote the total number of batch jobs running or suspended on r. Then, $\mathcal{F}^r = \{0, 1\}$. This system is similar to the self-service integrated network discussed in Section 4, and hence we can use the $\mathcal{S}_c^r(.)$ and $\mathcal{T}_c^r(.)$ formulas presented there.

Indeed, we are assuming here the arrival processes are Poisson. This is not, in general, true since the composite traffic contains blocked batch jobs returned immediately at the next time step to the system for retry. This assumption is less restrictive if blocked batch jobs are returned to the system after waiting an independent random period [23, 8]. This waiting effect can be easily incorporated into the above model. This model can be used to study the interactions between owner jobs and batch jobs, and examine various load balancing schemes through the $1/\mu_c^r(t)$ and $\alpha_c(t)$.

7 Validation of Systems with Self-Service Resources

In this section, we compare the results obtained using our method with those obtained using discrete-event simulation for systems with self-service resources. Both methods were coded in C. In our method, we obtain instantaneous performance measures through equations (7), (9), (12), and (13), substituting with the appropriate application-dependent parameters and formulas. We take the discrete-time step δ to be 0.1.

The simulation model differs from our analytical model in that the actual events of arrival and processing of customer requests are simulated according to the specified probability distributions and system characteristics (i.e. service disciplines, admission policy, etc.). To obtain reliable performance estimates, a number of independent replications (i.e. simulation runs) must be carried out and averaged. In particular, let $X^{(i)}(t)$ denote a generic measure computed at time instant t in replication i, where t takes on the successive values $t_1, t_2, \cdots, t_k, \cdots$. Then, the mean value of this measure at particular time instant t_k is estimated as $\sum_{i=1}^{N} X^{(i)}(t_k)/N$, where N is the total number of replications. The larger N is, the more accurate the simulation estimates are [22]. In our simulations, the performance measures are computed for $t = 1, 2, 3, \cdots$.

The measures considered are precisely defined as they are introduced below. In all experiments, we start with empty systems. For the cases with $N = 50$, the observed mean of the simulation measures at various time instants typically show 95% confidence interval for a $\pm$ 10% range. For the cases with higher N, 95% confidence interval is obtained for a $\pm$ 3% range.

Class c	$\mathcal{R}_c$	$c.r.req$	λ_c	$1/\mu_c$
c1	{r1}	30	0.125	5
c2	{r1}	15	0.5	1
c3	{r1}	50	0.2	2
c4	{r1}	10	0.1	2
c5	{r1}	40	0.125	1
c6	{r1}	25	0.5	0.5
c7	{r1}	30	1.0	0.5
c8	{r1}	10	0.0625	10
c9	{r1}	5	1.0	0.2
c10	{r1}	50	0.25	2

Figure 8: Parameters of 10 classes using r1 with r1.max = 200.

We first consider a MCSR system with a single resource r1 used by 10 customer classes whose parameters are shown in Figure 8.

Class-c customers arrive at r1 according to a Poisson process of rate λ_c. The system is self-service. In particular, an admitted class-c customer holds the acquired c.r1.req resource units for an exponential duration with mean $1/\mu_c$ before releasing them. This system is similar to a single-link integrated network modeled as in Section 4, and hence we use the $\mathcal{T}_c^r(.)$ and $\mathcal{S}_c^r(.)$ formulas presented there to obtain the performance measures by our method.

Figures 13, 14, and 15 show the time behavior of the total number of in-service customers, the fraction of resource units allocated, and the total throughput, respectively. The first measure denotes the total number of customers currently holding resource units, which is equal to $\sum_{c' \in \mathcal{C}^{\mathrm{r1}}} N_{c'}^{\mathrm{r1}}(t)$ in our method. The second measure denotes the fraction of r1.max currently being held by customers, which is equal to $(\sum_{c' \in \mathcal{C}^{\mathrm{r1}}} N_{c'}^{\mathrm{r1}}(t) \times c'.\mathrm{r1}.req)/\mathrm{r1}.max$ in our method. The third measure denotes the total current admission rate, which is equal to $\sum_{c' \in \mathcal{C}^{\mathrm{r1}}} \lambda_{c'}[1 - B_{c'}^{\mathrm{r1}}(t)]$ in our method. Generally, it is equal to $\sum_{c' \in \mathcal{C}} \lambda_{c'} \prod_{r' \in \mathcal{R}_{c'}} [1 - B_{c'}^{r'}(t)]$ for MCMR systems.

In our simulations, the first two measures displayed at time instant t ($t = 1, 2, 3, \cdots$) are simply the values of these measures as observed at t. The last measure, namely the total throughput, displayed at time instant t is defined to be the total number of customers admitted in the interval $[t-1,\ t)$.

Our method yields results very close to the exact values. In addition, we found our method much less time-consuming than simulation. This is especially because the latter requires the averaging of a large number of independent simulation runs. To give an idea of the computational savings, for this experiment, on a DECstation 5000/133, our method required around 6 seconds of execution time while the 50-run and 1000-run simulations required around 25 seconds and 8 minutes, respectively. The number of iterations required at each time step for convergence of the iterative procedure in steps 5-9 of Figure 5 is less than 6 iterations for $\epsilon = 10^{-5}$ and $\hat{z}_c^r(0) = \lambda_c/\mu_c^r$.

We next validate our resource independence assumption manifested in equation (7) by the product term $\prod$. We consider a similar self-service system but with 3 resources and 20 customer classes. Out of the 20 classes, 10 classes require all 3 resources. A class-c customer requires the same number of units of each $r \in \mathcal{R}_c$. Figure 9 shows the system parameters. Note that this system can be regarded as a multi-link integrated network modeled as in Section 4. See Figure 10. Here, classes 1 to 10 represent multi-hop connections modeling main traffic, while other classes represent one-hop connections modeling cross-traffic.

Class c	$\mathcal{R}_c$	$c.r.req$	λ_c	$1/\mu_c$
c1	{r1, r2, r3}	30	0.125	5
c2	{r1, r2, r3}	15	0.5	1
c3	{r1, r2, r3}	50	0.2	2
c4	{r1, r2, r3}	10	0.1	2
c5	{r1, r2, r3}	40	0.125	1
c6	{r1, r2, r3}	25	0.5	0.5
c7	{r1, r2, r3}	30	1.0	0.5
c8	{r1, r2, r3}	10	0.0625	10
c9	{r1, r2, r3}	5	1.0	0.2
c10	{r1, r2, r3}	50	0.25	2
c11	{r1}	30	0.125	5
c12	{r1}	15	0.5	1
c13	{r1}	50	0.2	2
c14	{r2}	10	0.1	2
c15	{r2}	40	0.125	1
c16	{r2}	25	0.5	0.5
c17	{r3}	30	1.0	0.5
c18	{r3}	10	0.0625	10
c19	{r3}	5	1.0	0.2
c20	{r3}	50	0.25	2

Figure 9: Parameters of 20 classes using 3 resources r1, r2, and r3 with r1.max = 150, r2.max = 200, and r3.max = 250.

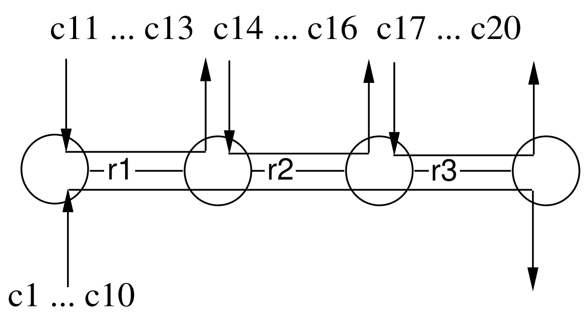

Figure 10: Multi-link network.

Figure 16 shows the instantaneous total throughput. Simulation results, denoted by **Exp**, are for Poisson arrivals and exponential holding times. Simulation results, denoted by **Det**, are for Poisson arrivals and deterministic holding times. The results show the accuracy of our method in both cases as they satisfy the assumptions required to obtain the $\mathcal{T}_c^r(.)$ and $\mathcal{S}_c^r(.)$ formulas used here. (Our experiments with deterministic arrivals show large errors as expected.)

Next, we consider a similar self-service system whose parameters are given in Figure 11. Here, λ_{c1} varies with time. This mimics the effect of traffic control policies such as flow control and routing. We assume λ_{c1} alternates every 20 time units between zero and 0.125, starting with zero. Figures 17 and 18 show the instantaneous total throughput and blocking probability, respectively. Our method accurately reproduces the behavior obtained by simulation. We compute the instantaneous blocking probability $B(t)$ from the throughput $\gamma(t)$ using the relation $B(t) = 1 - \gamma(t)/\lambda(t)$, where $\lambda(t)$ is the instantaneous total arrival rate of customers. We do this rather than compute $B(t)$ directly from the simulations because doing that would require averaging over a very large number of replications, because $B(t)$ typically has a very low value and thus a high sample variance.

Class c	$\mathcal{R}_c$	$c.r.req$	λ_c	$1/\mu_c$
c1	{r1, r2, r3}	30	$0 \leftrightarrow 0.125$	5
c2	{r1}	30	0.125	5
c3	{r2}	10	0.1	2
c4	{r3}	50	0.25	2

Figure 11: Parameters of 4 classes using 3 resources r1, r2, and r3 with r1.max = 50, r2.max = 100, and r3.max = 150.

8 Validation of Systems with Single-Server Resources

In this section, we compare the results obtained using our method with those obtained using discrete-event simulation for systems with single-server resources. The performance measures are computed as described in Section 7. Similar confidence intervals are also observed for the measures obtained by simulation.

We consider a MCMR system with 3 resources and 4 customer classes. Out of the 4 classes, class c1 requires all 3 resources. A class-c1 customer requires one unit of each resource. Figure 12 shows the system parameters.

Class c	$\mathcal{R}_c$	$c.r.req$	λ_c	$1/\mu_c$
c1	{r1, r2, r3}	1	0.2	1
c2	{r1}	1	0.5	1
c3	{r2}	1	0.8	1
c4	{r3}	1	0.4	1

Figure 12: Parameters of 4 classes using 3 resources with $r.max$ = 5 each.

Class-c customers arrive according to a Poisson process of rate λ_c. Each resource consists of a single-server with a finite waiting room and a FCFS scheduling discipline. An admitted class-c customer occupies one unit of space, and requires an exponential service time with unit mean. This system is similar to the parallel database server discussed in Section 5, and hence we use the $\mathcal{T}_c^r(.)$ and $\mathcal{S}_c^r(.)$ formulas presented there to obtain the performance measures by our method. Figure 19 shows the instantaneous total throughput. The results obtained by our method agree with those obtained by simulation.

We next consider the same system but with λ_{c1} varying with time. We assume λ_{c1} alternates every 20 time units between zero and 0.2, starting with zero. Figures 20 and 21 show the instantaneous total throughput and blocking probability, respectively. Our method accurately reproduces the behavior obtained by simulation.

9 Conclusions

The Z-iteration computes instantaneous probability measures of general time-dependent MCMR systems. It integrates techniques from several areas, including standard queueing theory techniques [18]; the resource decomposition technique [17, 9]; the dynamic flow technique [7, 6, 27, 8]; and the technique of repeated substitutions used in numerical analysis to solve nonlinear equations [16].

MCMR systems have often been analyzed under steady-state conditions (e.g. [13, 15, 21, 5, 25, 3, 23, 11]). The Z-iteration differs from (fixed-point) iterations commonly used in steady-state analysis, which only solve for steady-state measures.

Our model yields the time-varying behavior of a general MCMR system. We use the well-known decomposition technique [17, 15] to approximate the system as a collection of MCSR systems. For each MCSR system, we describe the evolution of the instantaneous average number of customers of each class by relating its instantaneous admission rate to its instantaneous departure rate. The computation of these instantaneous rates uses a basic concept, that of approximating instantaneous relationships by their steady-state counterparts.

To obtain the instantaneous admission rates, we adapt steady-state queueing formulas to yield the instantaneous blocking probability of each class in terms of the instantaneous average numbers of customers waiting or in service. To obtain the instantaneous departure rates, we again adapt steady-state queueing formulas to yield the instantaneous utilization of each class in terms of the instantaneous average numbers of customers waiting or in service. Unlike other transient analysis studies (e.g. [27]), our model considers general multi-class systems, where, for example, each class has different resource and service needs, and resources have different scheduling disciplines. Also, our model can be applied to describe general dynamic control schemes with the arrival and/or service rate of a class changing as a function of the instantaneous system state. Though we do not consider here sequential resource needs by one customer (a customer requests all needed resources simultaneously), our model can be easily extended to capture this situation.

Our dynamic flow model is quite general, and can be used to study both transient and steady-state performances of various MCMR blocking and non-blocking systems. Our method has advantages over other methods that might be used to analyze transient behaviors. One such method is that of time-dependent queueing models, which involve probability distributions for all events. However, such models are extremely difficult to solve analytically [28], and computationally expensive to solve numerically [27]; A second method is that of diffusion models, which utilize averages and variances [2, 24]. Such models involve partial differential equations and are usually intractable. A third method is that of fluid models, which utilize average quantities only [1]. Such models involve ordinary differential equations and are usually tractable. However, dynamic flow models appear more accurate since they include detailed probabilistic descriptions manifested in our model in the computation of both the instantaneous blocking probabilities and the instantaneous utilizations.

A Derivations for $M/M/2/2$

From [26], we have the following exact instantaneous solution. Consider the $M/M/2/2$ system initially empty with customer arrival rate $\lambda(t) = \lambda$ and service rate $\mu(t) = \mu$ for all $t \geq 0$. The Laplace transform $P_j(s)$ of the state probability $P_j(t)$, where j denotes the number of customers in the system ($j = 0, 1, 2$), is given by

$$\begin{aligned} P_j(s) &= \int_0^\infty e^{-st} P_j(t)dt \\ &= \sum_{i=j}^{2} (-1)^{i-j} \begin{pmatrix} i \\ j \end{pmatrix} \beta_i(s) \end{aligned}$$

where

$$\beta_i(s) = \frac{\frac{1}{(s+i\mu)} \sum_{r=i}^{2} \begin{pmatrix} 2 \\ r \end{pmatrix} \frac{(s+i\mu)\cdots(s+r\mu)}{\lambda^{r+1-i}}}{\sum_{r=0}^{2} \begin{pmatrix} 2 \\ r \end{pmatrix} \frac{s(s+\mu)\cdots(s+r\mu)}{\lambda^{r+1}}}$$

From the above result, we can directly obtain the exact instantaneous expressions for the blocking probability $P_2(t)$, denoted by $B_{\text{exact}}(t)$, and for the average number of customers in the system, denoted by $N_{\text{exact}}(t)$. In particular, we use *Mathematica* [29] to obtain the inverse Laplace transforms of the following:

$$\begin{aligned} B_{\text{exact}}(s) &= P_2(s) = \beta_2(s) = \frac{\lambda^2}{\lambda^2 s + 2\lambda s(s+\mu) + s(s+\mu)(s+2\mu)} \\ N_{\text{exact}}(s) &= P_1(s) + 2P_2(s) = \beta_2(s)\left(\frac{s + 2\lambda + 2\mu}{\lambda}\right) \end{aligned}$$

References

[1] J-C. Bolot and A.U. Shankar. Analysis of a Fluid Approximation to Flow Control Dynamics. In Proc. *IEEE INFOCOM '92*, Florence, Italy, May 1992.

[2] H. Chen and A. Mandelbaum. Discrete Flow Networks: Diffusion Approximations and Bottlenecks. *Annals of Probability*, 19(4):1463–1519, October 1991.

[3] S-P. Chung and K. Ross. Reduced Load Approximations for Multirate Loss Networks. *IEEE Transactions on Communications*, 41(8):1222–1231, August 1993.

[4] D. DeWitt and J. Gray. Parallel Database Systems: The Future of High Performance Database Systems. *CACM*, 35(6):85–98, June 1992.

[5] Z. Dziong and J. Roberts. Congestion Probabilities in a Circuit-Switched Integrated Services Network. *Performance Evaluation*, 7:267–284, 1987.

[6] J. Filipiak. *Modeling and Control of Dynamic Flows in Communication Networks*. New York: Springer-Verlag, 1988.

[7] F. Le Gall and J. Bernussou. An Analytical Formulation for Grade of Service Determination in Telephone Networks. *IEEE Transactions on Communications*, COM-31(3):420–424, March 1983.

[8] M.R. Garzia and C.M. Lockhart. Nonhierarchical Communications Networks: An Application of Compartmental Modeling. *IEEE Transactions on Communications*, 37:555–564, June 1989.

[9] A. Girard. *Routing and Dimensioning in Circuit-Switched Networks.* Addison-Wesley Publishing Company, 1990.

[10] A. Girard and M. Bell. Blocking Evaluation for Networks with Residual Capacity Adaptive Routing. *IEEE Transactions on Communications*, COM-37:1372–1380, 1989.

[11] A. Greenberg and P. Wright. Design and Analysis of Master/Slave Multiprocessors. *IEEE Transactions on Computers*, 40(8):963–976, August 1991.

[12] R. Guerin, H. Ahmadi, and M. Naghshineh. Equivalent Capacity and its Application to Bandwidth Allocation in High-Speed Networks. *IEEE J. Select. Areas Commun.*, SAC-9(7):968–981, September 1991.

[13] S. Jordan and P. Varaiya. Throughput in Multiple Service, Multiple Resource Communication Networks. *IEEE Transactions on Communications*, 39(8):1216–1222, August 1991.

[14] I. Kamel and C. Faloutsos. Parallel R-Trees. In Proc. *ACM SIGMOD*, pages 195–204, San Diego, CA, June 1992.

[15] F.P. Kelly. Blocking Probabilities in Large Circuit-Switched Networks. *Adv. Appl. Prob.*, 18:473–505, 1986.

[16] D. Kincaid and W. Cheney. *Numerical Analysis: Mathematics of Scientific Computing.* Brooks/Cole Publishing Company, 1991.

[17] L. Kleinrock. *Communication Nets: Stochastic Message Flow and Delay.* New York: McGraw Hill, 1964.

[18] L. Kleinrock. *Queueing Systems*, volume I and II. New York: Wiley, 1976.

[19] S. Leutenegger and X-H. Sun. Distributed Computing Feasibility in a Non-Dedicated Homogeneous Distributed System. In Proc. *Supercomputing '93*, November 1993.

[20] M. Litzkow, M. Livny, and M. Mutka. Condor – A Hunter of Idle Workstations. In Proc. *8th International Conference on Distributed Computing Systems*, San Jose, CA, June 1988.

[21] G. Louth, M. Mitzenmacher, and F.P. Kelly. Computational Complexity of Loss Networks. *Theoretical Computer Science*, 125:45–59, 1994.

[22] W. Lovegrove, J. Hammond, and D. Tipper. Simulation Methods for Studying Nonstationary Behavior of Computer Networks. *IEEE J. Select. Areas Commun.*, 8(9):1696–1708, December 1990.

[23] D. Mitra and P. Weinberger. Probabilistic Models of Database Locking: Solutions, Computational Algorithms, and Asymptotics. *J. ACM*, 31(4):855–878, October 1984.

[24] A. Mukherjee and J.C. Strikwerda. Analysis of Dynamic Congestion Control Protocols: A Fokker-Plank Approach. In Proc. *ACM SIGCOMM '91*, Zurich, Switzerland, September 1991.

[25] J. Roberts. A Service System with Heterogeneous User Requirements - Application to Multi-Services Telecommunications Systems. In *Performance of Data Communications Systems and Their Applications*, pages 423–431. G. Pujolle (Editor). North-Holland Publishing Company, 1981.

[26] L. Takács. *Introduction to the Theory of Queues*, pages 174–187. Greenwood Press, Westport, Connecticut, 1961.

[27] D. Tipper and M.K. Sundareshan. Numerical Methods for Modeling Computer Networks Under Nonstationary Conditions. *IEEE J. Select. Areas Commun.*, 8(9):1682–1695, December 1990.

[28] S. Tripathi and A. Duda. Time-Dependent Analysis of Queueing Systems. *INFOR*, 24(3):199–219, 1986.

[29] S. Wolfram. *Mathematica: A System for Doing Mathematics by Computer.* Addison-Wesley, 1991.

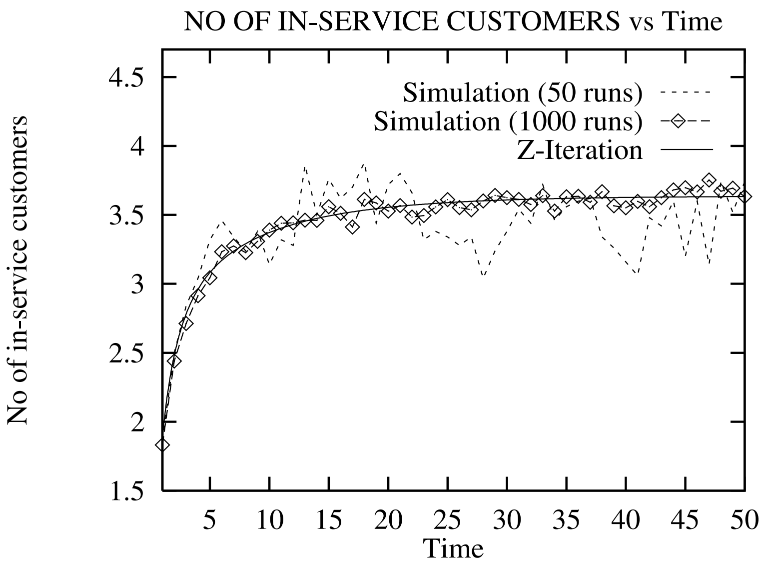

Figure 13: Total number of in-service customers versus time. MCSR self-service system.

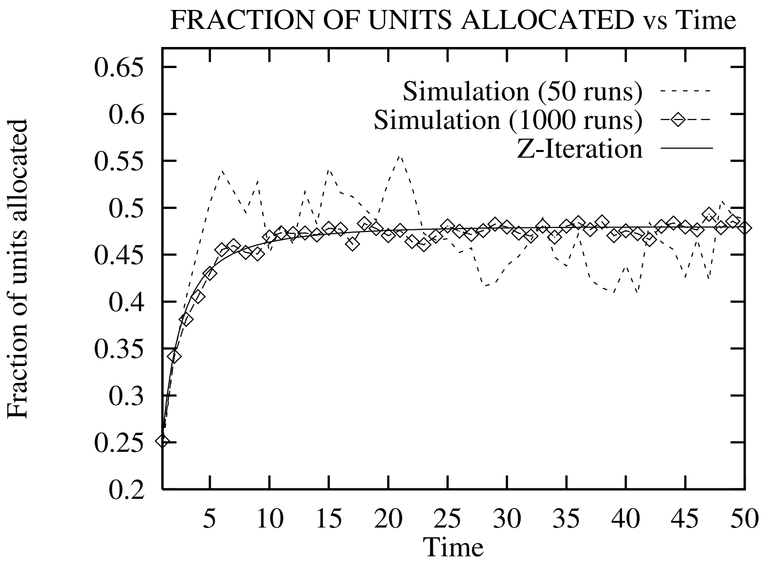

Figure 14: Fraction of resource units allocated versus time. MCSR self-service system.

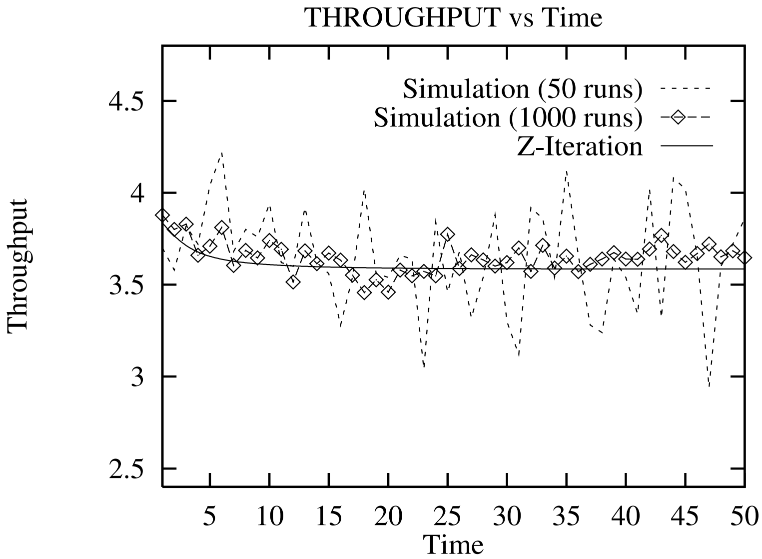

Figure 15: Total throughput versus time. MCSR self-service system.

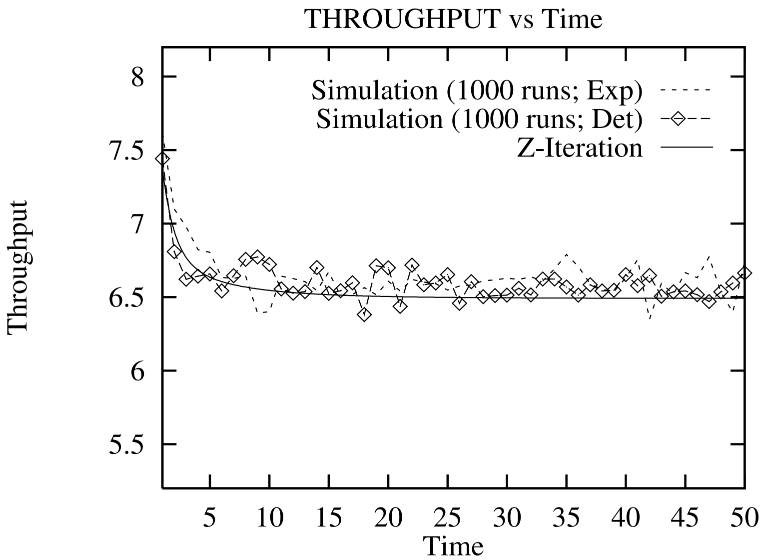

Figure 16: Total throughput versus time. MCMR system with self-service resources.

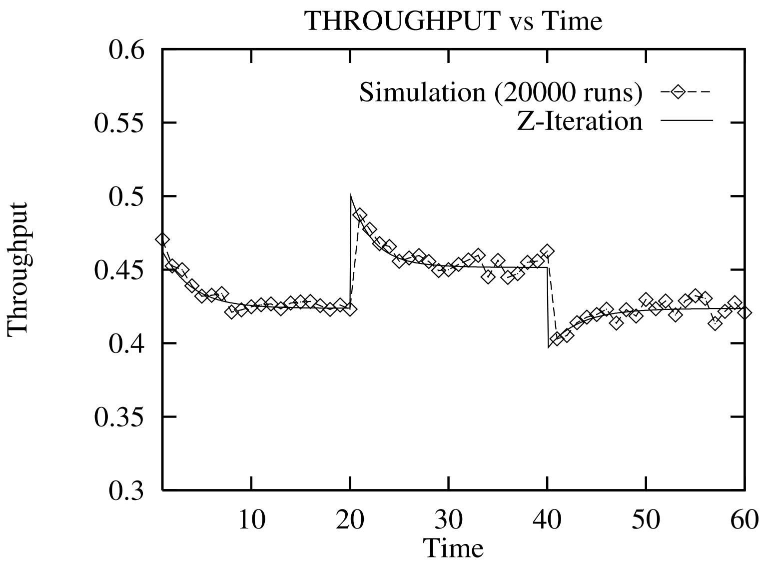

Figure 17: Total throughput versus time. MCMR system with self-service resources. Time-varying arrivals.

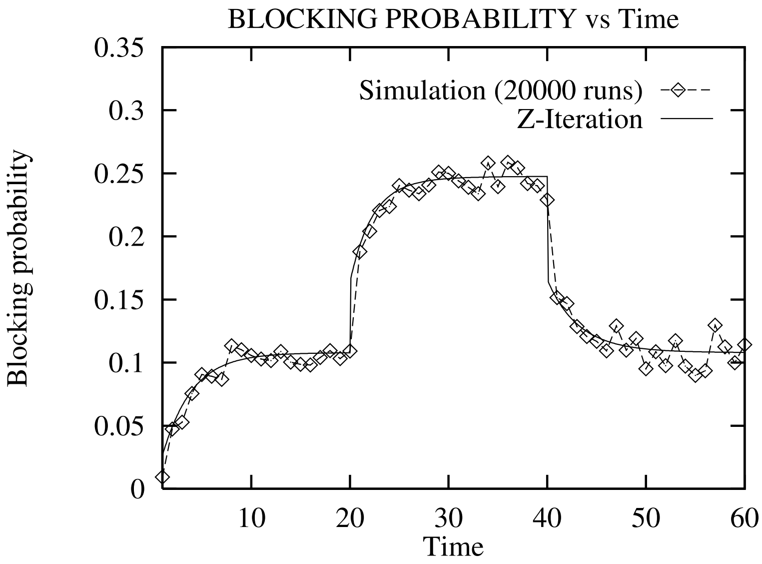

Figure 18: Blocking probability versus time. MCMR system with self-service resources. Time-varying arrivals.

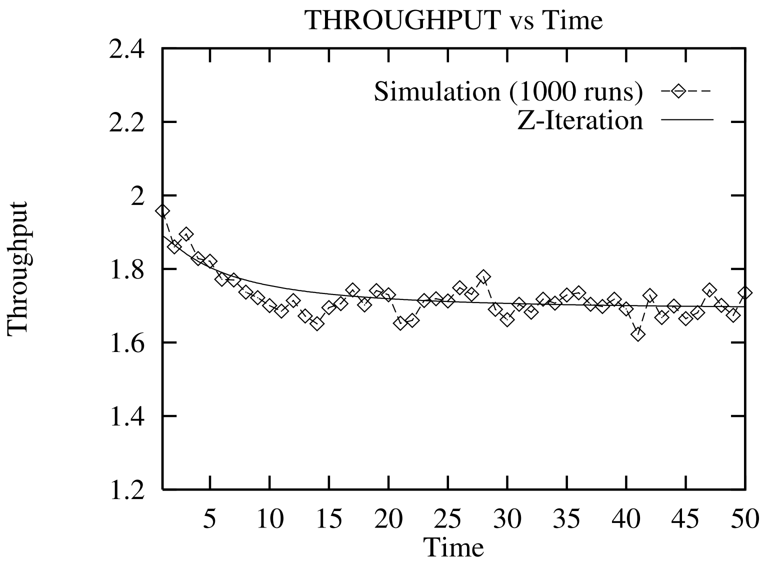

Figure 19: Total throughput versus time. MCMR system with single-server resources.

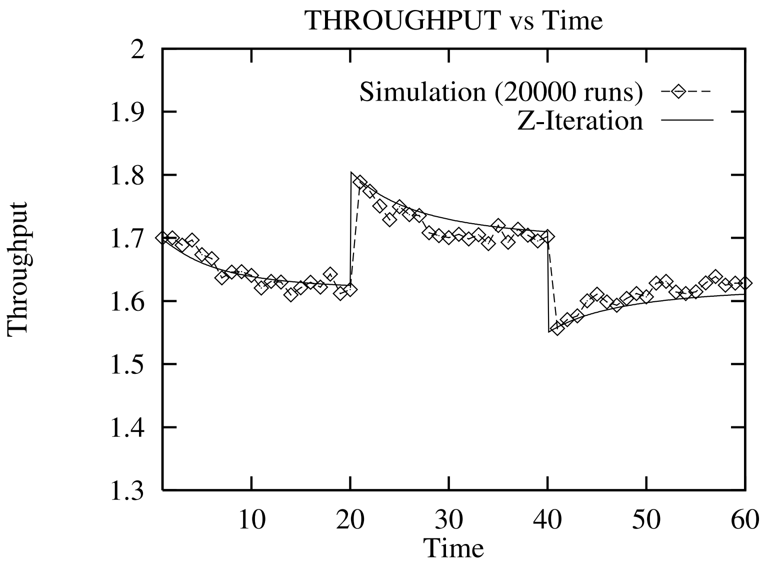

Figure 20: Total throughput versus time. MCMR system with single-server resources. Time-varying arrivals.

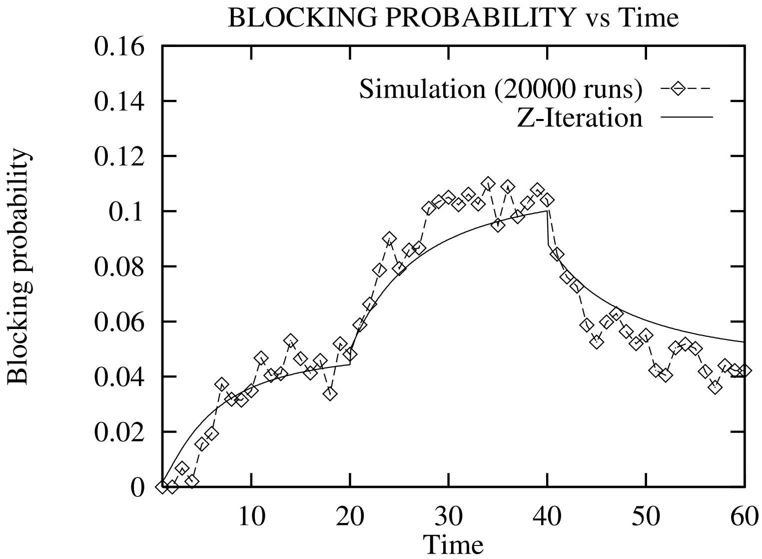

Figure 21: Blocking probability versus time. MCMR system with single-server resources. Time-varying arrivals.