Z-Iteration: A Simple Method for Throughput Estimation in Time-Dependent Multi-Class Systems*

Ibrahim Matta
imm@cs.umd.edu

A. Udaya Shankar
shankar@cs.umd.edu

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

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 method, referred to as *Z-iteration*, to estimate both transient and steady-state performances of such systems. The method makes use of results and techniques available from queueing theory, network analysis, dynamic flow theory, and numerical analysis. We show the generality of the Z-iteration by applying it to an ATM network, a parallel disk system, and a distributed batch system. Validations against discrete-event simulations show the accuracy and computational advantages of the Z-iteration.

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 requests, and $1/\mu_c^r(t)$ denote the instantaneous service (or processing) time of a class-c request 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 statistics of the customer arrival processes or the service rates of 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. Among the main performance measures of interest are the instantaneous blocking probabilities (or equivalently the throughputs) of the different classes, instantaneous average number of customers at resources, etc.

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. [12, 14, 20, 5, 25, 3, 22, 10]). In this paper, we formulate a dynamic flow model [6] to account for transient conditions as well. We solve our model by an iteration that differs from iterations used in steady-state analysis, which only solve for steady-state measures.

*This work is supported in part by ARPA and Philips Labs under contract DASG60-92-0055 to Department of Computer Science at the University of Maryland, and by National Science Foundation Grant No. NCR 89-04590. The views, opinions, and/or findings contained in this paper are those of the author(s) and should not be interpreted as representing the official policies, either expressed or implied, of the Advanced Research Projects Agency, PL, the National Science Foundation, or the U.S. Government.

Our solution method

The generality and time-dependency of our model seem to preclude analytical closed-form solutions. Our solution method, referred to as *Z-iteration* is, however, numerical. This has significant computational advantages over the (straightforward) discrete-event simulation approach, which requires the averaging of a large number of independent simulation runs to obtain meaningful performance estimates.

The instantaneous blocking probability of a class c, denoted by $B_c(t)$, is defined as the instantaneous probability that a class-c request is blocked at any of the required $\mathcal{R}_c$ resources. For this, we decompose our MCMR system into a set of multiple-class single-resource (MCSR) systems by invoking the resource independence assumption. Denoting by $B_c^r(t)$ the instantaneous blocking probability of class c at resource r, we have $B_c(t) = 1 - \prod_{r\in\mathcal{R}_c}(1 - B_c^r(t))$.

The Z-iteration computes $B_c^r(t)$ together with $N_c^r(t)$, the instantaneous average number of class-c requests waiting or in service at resource r, and $U_c^r(t)$, the instantaneous average number of class-c requests in service at resource r.

Let the index c' range over the set of classes requiring resource r. The Z-iteration depends upon the availability of two steady-state results about each MCSR system $r \in \mathcal{R}$ assuming that the $\lambda_{c'}(t)$ and $\mu_{c'}^r(t)$ are constants: (1) an expression for the steady-state blocking probability B_c^r in terms of the steady-state actual offered loads $\lambda_{c'}/\mu_{c'}^r$; and (2) an expression for the steady-state utilization U_c^r in terms of the steady-state average numbers of customers $N_{c'}^r$, from which we readily obtain an expression for λ_c/μ_c^r in terms of the $N_{c'}^r$ and B_c^r.

These two steady-state results 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 [17, 6]. We point out that the expressions do not have to be closed form and can be implicit.

We make use of the concept of *instantaneous fictitious offered load*, originally introduced in [7], to obtain instantaneous versions of the above expressions. Specifically, we replace B_c^r by $B_c^r(t)$, the $N_{c'}^r$ by $N_{c'}^r(t)$, and the $\lambda_{c'}/\mu_{c'}^r$ by instantaneous fictitious offered loads $z_{c'}^r(t)$. (Note that λ_c/μ_c^r is not replaced by $\lambda_c(t)/\mu_c^r(t)$.)

This yields for every $r \in \mathcal{R}$ two "instantaneous" expressions, one for $B_c^r(t)$ in terms of the $z_{c'}^r(t)$, and one for $z_c^r(t)$ in terms of the $N_{c'}^r(t)$ and $B_c^r(t)$. A third instantaneous expression is obtained from standard flow balance, defining $N_c^r(t+\delta)$ in terms of the $N_{c'}^r(t)$, $\lambda_c(t)$, $\mu_c^r(t)$, and $B_c^{r'}(t)$ for $r' \in \mathcal{R}_c$, where δ is the time step for computing the instantaneous measures. With these three instantaneous expressions we compute the $B_{c'}^r(t)$, $z_{c'}^r(t)$, and $N_{c'}^r(t+\delta)$ in terms of the $N_{c'}^r(t)$, $\lambda_{c'}(t)$ and $\mu_{c'}^r(t)$ for $t = 0,\ \delta,\ 2\delta,\ \cdots$. Specifically, given the $N_{c'}^r(t)$, we iterate over the first two expressions until the $B_{c'}^r(t)$ and $z_{c'}^r(t)$ converge. Then we use the third expression to compute the $N_{c'}^r(t+\delta)$.

Organization of the paper

The rest of the paper is organized as follows. Section 2 presents the Z-iteration method for the general MCMR model. In Sections 3, 4, 5, we apply the Z-iteration to three specific systems with time-varying inputs and dynamic control, namely, an ATM network, a parallel disk system, 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 6. The second system is modeled as a system with single-server resources, for which validations are given in Section 7. We discuss related work in Section 8. Section 9 concludes the paper.

2 The Z-Iteration

Figure 1 outlines our solution method to the general MCMR model introduced in Section 1. Recall that the following measures have been introduced:

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

Let $\mathcal{C}^r$ denote the set of classes requesting units of resource r. 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 an iterative procedure (steps 5-9) on instantaneous versions of two steady-state formulas (steps 7 and 8). We describe these in detail below.

We define a *feasible state* of resource r by the number of requests from 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.

Details of step 7

The first steady-state formula expresses the steady-state blocking probability B_c^r of class c at resource r in terms of the steady-state actual offered loads $\{\lambda_{c'}/\mu_{c'}^r :\ c' \in \mathcal{C}^r\}$. That is, 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 request 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. We solve analytically for the probability of being in a class-c blocking state, yielding a formula $\mathcal{S}_c^r$ in terms of the $\frac{\lambda_{c'}}{\mu_{c'}^r}$:

$$B_c^r = \mathcal{S}_c^r(\{\frac{\lambda_{c'}}{\mu_{c'}^r} :\ c' \in \mathcal{C}^r\}) \quad \text{for } c \in \mathcal{C}^r \tag{1}$$

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 $\{\hat{z}_c^r(t) : r \in \mathcal{R},\ c \in \mathcal{C}^r\}$ /* 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 (2))
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 (4))
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 (5))
end

Figure 1: Evaluation method.

To illustrate, consider an $M/G/m/m$ 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 m servers for an average duration of $1/\mu_c^r$. Then $\mathcal{S}_c^r$ is the Erlang-B formula, i.e. $\mathcal{S}_c^r = E(\frac{\lambda_c}{\mu_c^r}, m) = \frac{(\frac{\lambda_c}{\mu_c^r})^m/m!}{\sum_{j=0}^{m}(\frac{\lambda_c}{\mu_c^r})^j/j!}$ [17].

The instantaneous version of (1) is obtained by replacing B_c^r by $B_c^r(t)$ and $\frac{\lambda_{c'}}{\mu_{c'}^r}$ by $z_{c'}^r(t)$, yielding

$$B_c^r(t) = \mathcal{S}_c^r(\{z_{c'}^r(t) : c' \in \mathcal{C}^r\}) \quad \text{for } c \in \mathcal{C}^r \tag{2}$$

Details of step 8

The second steady-state formula, which we refer to as $\mathcal{T}_c^r$, expresses U_c^r, the steady-state utilization of resource r by class-c customers, in terms of $\{N_{c'}^r : c' \in \mathcal{C}^r\}$, the steady-state average numbers of customers at resource r:

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

From this and $\mu_c^r\, U_c^r = \lambda_c\,[1 - B_c^r]$, obtained by equating the departure rate to the admission rate, we have

$$\frac{\lambda_c}{\mu_c^r} = \frac{\mathcal{T}_c^r(\{N_{c'}^r : c' \in \mathcal{C}^r\})}{[1 - B_c^r]} \quad \text{for } c \in \mathcal{C}^r \tag{3}$$

$\mathcal{T}_c^r$ is a function that reflects the load and service discipline of r. The exact form of $\mathcal{T}_c^r$ is application dependent. For example, consider a self-service facility where the customer is also the server, as in an $M/G/m/m$ queueing system; here $\mathcal{T}_c^r$ 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 way is to use steady-state queueing formulas expressing N_c^r in terms of the $\frac{\lambda_{c'}}{\mu_{c'}^r}$ assuming no blocking. Inverting these formulas, we obtain $\frac{\lambda_c}{\mu_c^r}$ in terms of the $N_{c'}^r$. Since we are assuming no blocking, from equation (3), we have $\mathcal{T}_c^r = \frac{\lambda_c}{\mu_c^r}$. Thus, we get $\mathcal{T}_c^r$ in terms of the $N_{c'}^r$. (See Section 4.)

The instantaneous version of (3) yields

$$z_c^r(t) = \frac{\mathcal{T}_c^r(\{N_{c'}^r(t) : c' \in \mathcal{C}^r\})}{[1 - B_c^r(t)]} \quad \text{for } c \in \mathcal{C}^r \tag{4}$$

Knowing $\{N_c^r(t)\!: c \in \mathcal{C}^r\}$ at some fixed t, we can solve equations (2) and (4) iteratively for $\{B_c^r(t)\!: c \in \mathcal{C}^r\}$. In particular, starting from an initial estimate $\{\hat{z}_c^r(t)\!: c \in \mathcal{C}^r\}$, we compute $\{B_c^r(t)\!: c \in \mathcal{C}^r\}$ from equations (2). Then, we use equations (4) to compute new values for $\{z_c^r(t)\!: c \in \mathcal{C}^r\}$. We repeat this process until the values of $\{z_c^r(t)\!: c \in \mathcal{C}^r\}$ stabilize as illustrated in steps 5-9 of Figure 1.

Details of step 10

At a fixed time t, once we obtain $\{B_c^r(t) : r \in \mathcal{R},\ c \in \mathcal{C}^r\}$, we obtain $\{N_c^r(t+\delta) : r \in \mathcal{R},\ c \in \mathcal{C}^r\}$, where δ is the discrete-time step, using the following difference equation:

$$N_c^r(t+\delta) = N_c^r(t) \;-\; \mu_c^r(t)\, U_c^r(t)\,\delta \;+\; \lambda_c(t)\,\delta \prod_{r' \in \mathcal{R}_c} [1 - B_c^{r'}(t)] \tag{5}$$

The second term in the right-hand side of equation (5) represents the average number of class-c requests which finish using (and depart from) resource r during $[t,\ t+\delta)$; the quantity $U_c^r(t)$ is computed from $\mathcal{T}_c^r(\{N_{c'}^r(t) : c' \in \mathcal{C}^r\})$. The third term represents the average number of new class-c requests that are admitted to resource r during $[t, t+\delta)$. Note that the product term $\prod$ reflects the assumption made in Section 1 that a new class-c request is admitted iff it is not blocked at any of the required $\mathcal{R}_c$ resources (this invokes the resource independence assumption).

Comments

Assuming that K iterations are needed for convergence of the iterative procedure in steps 5-9 of Figure 1, the computational complexity for each time step is $O(|\mathcal{R}|\ |\mathcal{C}^r|\ (\ (|B_c^r| + |z_c^r|)K + |N_c^r|\ ))$, where $|B_c^r|$ is the cost of evaluating $B_c^r(.)$ via (2), $|z_c^r|$ that of evaluating $z_c^r(.)$ via (4), and $|N_c^r|$ that of evaluating $N_c^r(.)$ via (5). The Z-iteration requires storage of $O(V\ |\mathcal{R}|\ |\mathcal{C}^r|)$, where V is the number of instantaneous measures. From Figure 1, we have $V=5$ since we have 5 instantaneous measures defined, namely, $B_c^r(.)$, $z_c^r(.)$, $N_c^r(.)$, $\lambda_c(.)$ and $\mu_c^r(.)$.

Regarding the convergence of the above iterative procedure, it is unfortunately virtually impossible to demonstrate convergence analytically in most practical situations due to the complex nature of the underlying nonlinear system. There is no simple way to determine whether there exist solutions to such a system and, if so, under what conditions it is unique. Even if the system has a unique solution, the iteration may not converge to it and rather oscillate between different values. Although sufficient conditions for convergence can be obtained with techniques such as the control-theoretic Liapunov method [24], they are very difficult to obtain even for moderately complex systems. Despite the lack of analytical results, our numerical computations indicate that the iteration converges quickly in many practical situations (see Sections 6 and 7).

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 requests from 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 requests.

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 (5) and use them in conjunction with equations (2) and (4) to iteratively solve for steady-state.

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.

3 Application: ATM Network

Consider an ATM network that uses dynamic routing to support real-time communication (voice, video, etc.) between pairs of source and destination nodes. The connections of a service i arrive at the service's source node according to a Poisson process of rate λ_i, and have an end-to-end quality-of-service requirement D_i (e.g. delay). Each connection, once it is successfully setup, has a lifetime of average duration $\frac{1}{\mu_i}$. 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.

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 i requires the reservation of a certain amount of bandwidth on each link along its route that are enough to satisfy D_i. This reservation amount can be thought of as either the peak transmission rate of the connection or its "effective bandwidth" [11] 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 i, $\lambda_c(t)$, is a function of λ_i 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 i arrive according to a Poisson process of rate $\lambda_c(t) = \alpha_c(t)\ \lambda_i$, where $\alpha_c(t)$ is the load-dependent routing probability.

An arriving class-c connection of service i 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 $\frac{1}{\mu_c^r(t)} = \frac{1}{\mu_i}$. 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 $Q^r(j)$ denote the steady-state probability of r being in state j. Then $Q^r(.)$ satisfies the following recurrence relation [25]:

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

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

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

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

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 (2) after replacing the $\frac{\lambda_{c'}}{\mu_{c'}^r}$ by fictitious offered loads $z_{c'}^r(t)$.

Regarding the function $\mathcal{T}_c^r(.)$ used in equations (4), 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 6. There we consider systems equivalent to single-link network, and tandem multi-hop network. The tandem network is used by several multi-hop connections representing main traffic, and several one-hop connections representing cross-traffic.

4 Application: Parallel Disk System

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. This parallelism typically leads to reduction in data access time [4, 13]. 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.

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 requests 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 $c \in \mathcal{C}^r$:

$$B_c^r = \frac{\left(\frac{\sum_{c' \in \mathcal{C}^r} \lambda_{c'}}{\mu^r}\right)^{r.max}}{\sum_{j=0}^{r.max}\left(\frac{\sum_{c' \in \mathcal{C}^r} \lambda_{c'}}{\mu^r}\right)^j} \quad [17]$$

This steady-state solution can be used in equations (2) 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 introduced in Section 2 to derive the function $\mathcal{T}_c^r(.)$ used in equations (4). 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 [17]

$$N_c^r = \frac{\lambda_c}{\mu^r - \sum_{c' \in \mathcal{C}^r} \lambda_{c'}} \qquad (6)$$

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 7.

5 Application: Distributed Batch System

Consider a distributed batch system such as Condor [19]. 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 [18]. 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 (6), 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,

$$\lambda_c(t) = [\lambda_i + \sum_{\substack{\text{classes } c' \text{ of type } i \\ r' \in \mathcal{R}_{c'}}} \lambda_{c'}(t-\delta)\ B_{c'}^{r'}(t-\delta)]\ \alpha_c(t) \quad (7)$$

The $\sum$ term in equation (7) 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 ATM system discussed in Section 3, 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 [22, 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 $\alpha_c(t)$.

6 Validation of Systems with Self-Service Resources

In this section, we compare the results obtained using our approach with those obtained using discrete-event simulation for systems with self-service resources. In our approach, we obtain instantaneous performance measures through equations (2), (4), and (5), 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 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 [21]. In our simulations, the performance measures are periodically computed at $t = 1, 2, 3, \cdots$.

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, the same confidence interval is obtained for a $\pm$ 3% range.

We first consider a MCSR system with a single resource r1 used by 10 customer classes whose parameters are shown in Figure 2.

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

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 ATM network modeled as in Section 3, and hence we use the $\mathcal{T}_c^r(.)$ and $\mathcal{S}_c^r(.)$ formulas presented there to obtain the performance measures by our approach.

Figures 3, 4, and 5 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 $\sum_{c' \in \mathcal{C}^{r1}} N_{c'}^{r1}(t)$ in our approach. The second measure denotes the fraction of r1.max currently being held by customers, which is equal to $(\sum_{c' \in \mathcal{C}^{r1}} N_{c'}^{r1}(t) \times c'.\text{r1}.req)/\text{r1}.max$ in our approach. The third measure denotes the total current admission rate, which is equal to $\sum_{c' \in \mathcal{C}^{r1}} \lambda_{c'}(1 - B_{c'}^{r1}(t))$ in our approach. 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 approach yields results very close to the exact values. In addition, we found our approach 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 approach 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 1 is less than 6 iterations for $\epsilon = 10^{-5}$ and $\hat{z}_c^r(0) = \lambda_c/\mu_c^r$.

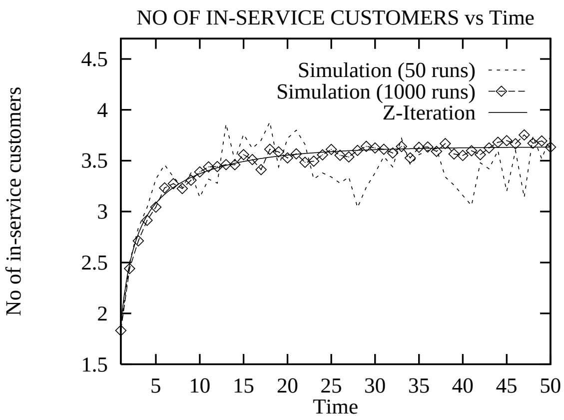

Figure 3: Total number of in-service customers versus time. MCSR self-service system.

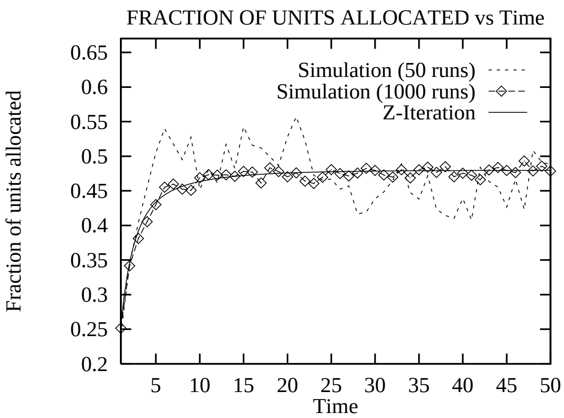

Figure 4: Fraction of resource units allocated versus time. MCSR self-service system.

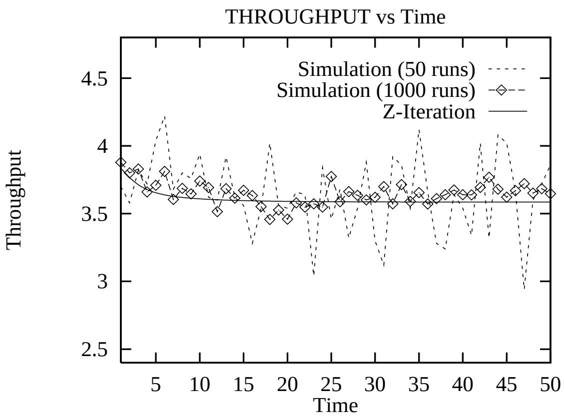

Figure 5: Total throughput versus time. MCSR self-service system.

We next validate our resource independence assumption manifested in equation (5) 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 6 shows the system parameters. Note that this system can be regarded as a tandem multi-hop ATM network modeled as in Section 3. See Figure 7. 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 6: Parameters of 20 classes using 3 resources r1, r2, and r3 with r1.max = 150, r2.max = 200, and r3.max = 250.

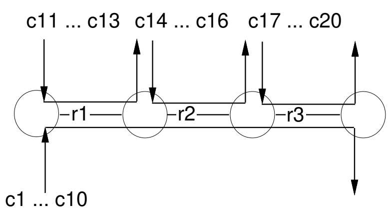

Figure 7: Tandem network.

Figure 8 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 approach 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.)

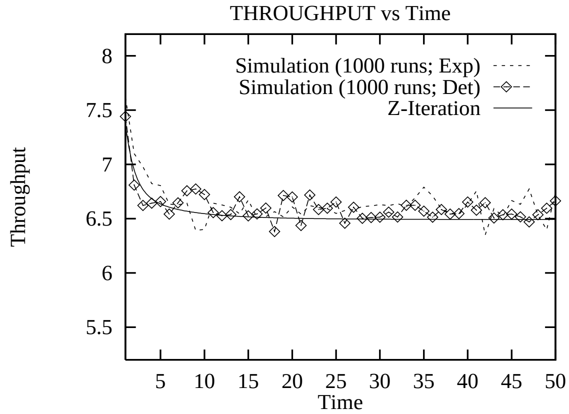

Figure 8: Total throughput versus time. MCMR system with self-service resources.

Next, we consider a similar self-service system whose parameters are given in Figure 9. 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 10 and 11 show the instantaneous total throughput and blocking probability, respectively. Our approach 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 requests. 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 9: Parameters of 4 classes using 3 resources r1, r2, and r3 with r1.$max = 50$, r2.$max = 100$, and r3.$max = 150$.

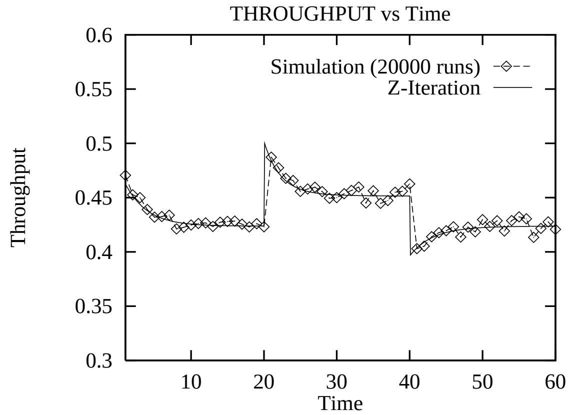

Figure 10: Total throughput versus time. MCMR system with self-service resources. Time-varying arrivals.

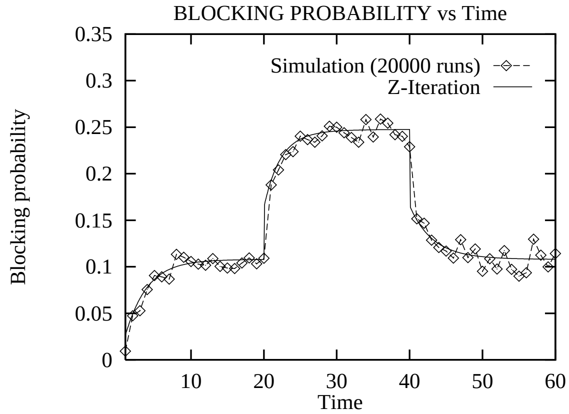

Figure 11: Blocking probability versus time. MCMR system with self-service resources. Time-varying arrivals.

7 Validation of Systems with Single-Server Resources

In this section, we compare the results obtained using our approach with those obtained using discrete-event simulation for systems with single-server resources. The performance measures are computed as described in Section 6. 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 disk system discussed 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 approach. Figure 13 shows the instantaneous total throughput. The results obtained by our approach 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 14 and 15 show the instantaneous total throughput and blocking probability, respectively. Our approach accurately reproduces the behavior obtained by simulation.

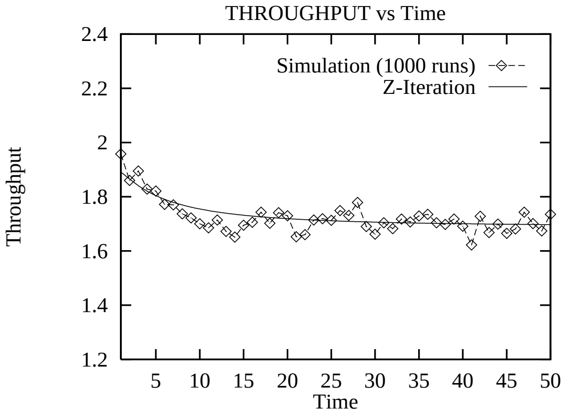

Figure 13: Total throughput versus time. MCMR system with single-server resources.

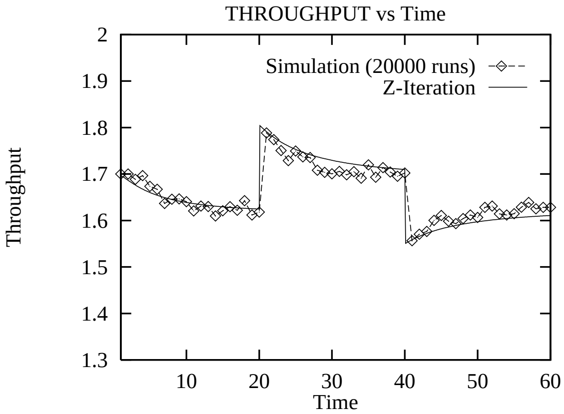

Figure 14: Total throughput versus time. MCMR system with single-server resources. Time-varying arrivals.

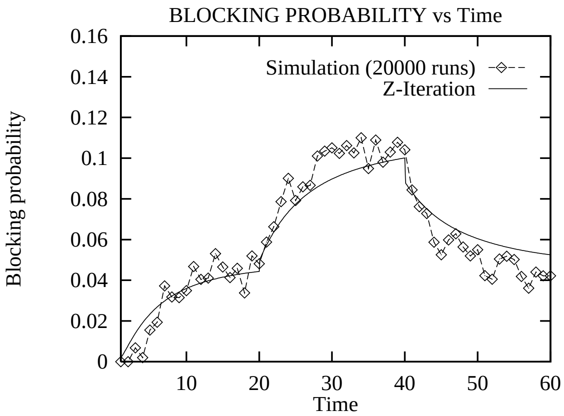

Figure 15: Blocking probability versus time. MCMR system with single-server resources. Time-varying arrivals.

8 Related Work

MCMR systems have often been analyzed under steady-state conditions (e.g. [12, 14, 20, 5, 25, 3, 22, 10]). In this paper, we formulated a dynamic flow model [6] to account for transient conditions as well. We solved our model by an iteration that differs from iterations used in steady-state analysis, which only solve for steady-state measures.

Our model yields the time-varying average behavior of a general MCMR system. We use the well-known decomposition technique [16, 14] 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.

To obtain the instantaneous admission rates, we adapt steady-state queueing formulas to yield the instantaneous blocking probability of each class in terms of instantaneous fictitious offered loads. The concept of fictitious offered load was originally introduced in [7], where it was used to obtain steady-state blocking probability and carried load for a specific call routing and network topology.

Reference [7] considered a network of source nodes, destination nodes, and intermediate nodes, with a link from every source node to every intermediate node, and a link from every intermediate node to every destination node. Each link can carry a fixed total number of calls. The call arrival process from a source to a destination is Poisson with fixed rate. The call routing is not dynamic; a fixed fraction of the call arrivals is routed through every intermediate node. In addition, overflow traffic (due to blocking links) is routed through alternate available routes (from equation (15) in [7], it appears that overflow from a blocking link is duplicated on all alternate available routes). Each call, once admitted, has an exponential holding time of fixed mean that is the same for all calls. The blocking probability of a link is given by the Erlang-B formula expressed in terms of fictitious combined offered load. The system is solved for steady-state average number of calls on each link by equating the call departure rate to the call admission rate.

Our model extends this fictitious offered load concept to *any* system where the steady-state blocking probabilities can be expressed in terms of offered loads. This allows us to consider 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 routing schemes with the arrival rate of a class changing as a function of the instantaneous system state.

To obtain the instantaneous departure rates, we again adapt steady-state queueing formulas to yield the instantaneous average number of in-service customers of each class in terms of the instantaneous average numbers of customers waiting and in service. The same technique was used in [26], where feedforward queueing networks were considered. Each service center is an $M/M/1$ infinite FCFS queue with the same average service time for all classes. The routing of each class is a time-dependent Bernoulli process. Compared to our model, this does not model blocking resources, or service centers with complicated structure (e.g. service centers consisting of multiple resources with different scheduling disciplines serving customers with different needs). Though we do not consider here sequential resource needs by one customer (a customer requests all needed resources simultaneously), our model is 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 approach has advantages over other approaches that might be used to analyze transient behaviors. One such approach is that of time-dependent queueing models, which involve probability distributions for all events. However, such models are extremely difficult to solve analytically [27], and computationally expensive to solve numerically [26]; A second approach is that of diffusion models, which utilize averages and variances [2, 23]. Such models involve partial differential equations and are usually intractable. A third approach 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 average number of in-service customers.

9 Conclusions

We described the Z-iteration, a simple method to estimate transient and steady-state performances of MCMR systems. The Z-iteration integrates techniques from several areas, including standard queueing theory techniques [17]; the resource decomposition technique [16, 9]; the dynamic flow technique used for approximating system dynamics and nonlinearity [7, 6, 26, 8]; and the technique of repeated substitutions used in numerical analysis to solve nonlinear equations [15].

We have shown the generality and validity of the Z-iteration by applying it to three systems, namely, an ATM network, a parallel disk system, and a distributed batch system. Future work remains to carry out further validations and detailed evaluation of such systems. We are currently using the Z-iteration to analyze traffic control schemes for ATM networks.

References

[1] J-C. Bolot and A.U. Shankar. Analysis of a fluid approximation to flow control dynamics. In *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. Greenberg and P. Wright. Design and analysis of master/slave multiprocessors. *IEEE Transactions on Computers*, 40(8):963–976, August 1991.

[11] 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.

[12] S. Jordan and P. Varaiya. Throughput in multiple service, multiple resource communication networks. *IEEE Transactions on Communications*, 39(8):1216–1222, August 1991.

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

[14] F.P. Kelly. Blocking probabilities in large circuit-switched networks. *Adv. Appl. Prob.*, 18:473–505, 1986.

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

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

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

[18] S. Leutenegger and X-H. Sun. Distributed computing feasibility in a non-dedicated homogeneous distributed system. In *Supercomputing '93*, November 1993.

[19] M. Litzkow, M. Livny, and M. Mutka. Condor – a hunter of idle workstations. In *8th International Conference on Distributed Computing Systems*, San Jose, CA, June 1988.

[20] G. Louth, M. Mitzenmacher, and F.P. Kelly. Computational complexity of loss networks. *Theoretical Computer Science*, 125:45–59, 1994.

[21] 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.

[22] D. Mitra and P. Weinberger. Probabilistic models of database locking: Solutions, computational algorithms, and asymptotics. *J. ACM*, 31(4):855–878, October 1984.

[23] A. Mukherjee and J.C. Strikwerda. Analysis of dynamic congestion control protocols: A fokker-plank approach. In *ACM SIGCOMM '91*, Zurich, Switzerland, September 1991.

[24] K. Ogata. *Discrete-Time Control Systems*. Prentice-Hall, Inc., 1987.

[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] 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.

[27] S. Tripathi and A. Duda. Time-dependent analysis of queueing systems. *INFOR*, 24(3):199–219, 1986.