Numerical Computation of Response Time Distributions Using Stochastic Reward Nets

Jogesh K. Muppala
Dept. of Computer Science
The Hong Kong University of Science and Technology
Clear Water Bay
Kowloon, Hong Kong

Kishor S. Trivedi*
Dept. of Electrical Engineering
Duke University
Durham, NC 27708, USA

Varsha Mainkar
Dept. of Computer Science
Duke University
Durham, NC 27708, USA

Vidyadhar G. Kulkarni
Curriculum in Operations Research
University of North Carolina
Chapel Hill, NC 27514, USA

Abstract

We consider the numerical computation of response time distributions for closed product form queueing networks using the *tagged customer* approach. We map this problem on to the computation of the time to absorption distribution of a finite-state continuous time Markov chain. The construction and solution of these Markov chains is carried out using a variation of stochastic Petri nets called stochastic reward nets (SRNs). We examine the effects of changing the service discipline and the service time distribution at a queueing center on the response time distribution. A multiserver queueing network example is also presented. While the tagged customer approach for computing the response time distribution is not new, this paper presents a new approach for computing the response time distributions using SRNs.

1 Introduction

Real-time systems are becoming increasingly popular in various areas like transaction processing (airline reservation systems, automatic teller systems), process control systems etc.

*This research was sponsored in part by the National Science Foundation under Grant CCR-9108114 and by the Naval Surface Warfare Center

These systems are typically characterized by stringent deadlines and high availability requirements. Thus, evaluation techniques for real-time systems should permit the computation of measures such as the probability of failing to meet a deadline.

Queueing networks have been successfully used in performance modeling of computer and communication systems (Lazowska et al. [9]). They are especially suited for representing resource contention and queueing for service. Most of the analysis techniques so far have concentrated on the evaluation of averages of various performance measures like throughput, utilization and response time using efficient algorithms such as convolution and mean value analysis (MVA) (Lavenberg [7]). For real-time systems, however, the knowledge of response time distributions is required in order to compute and/or minimize the probability of missing a deadline.

Closed-form solutions for response time distribution in queueing networks are available only in very few cases such as the $M/M/n$ FCFS queue. Methods for computing the Laplace transform of the response time distribution are available for queueing networks with special structure. For a recent survey of these methods, see Boxma and Daduna [3]. As mentioned in the survey, it is very difficult to obtain closed-form solutions for queueing networks with a general structure.

Numerical computation of the response time distribution is then the only alternative short of very expensive Monte-Carlo simulation. One such method is based on the *tagged customer* approach. In this method, an arbitrary customer is picked as the tagged customer and its passage through the network is tracked. By this method, the problem of computing the conditional response time distribution of the tagged customer is transformed into the time to absorption distribution of a finite-state, continuous time Markov chain (CTMC). Using the arrival theorem of Sevcik and Mitrani [16] (Lavenberg and Reiser [8]) we can establish the distribution of the other customers in the network at the instant of arrival of the tagged customer. This allows us to obtain the unconditional response time distribution.

Melamed and Yadin [10, 11] present a numerical method based on the tagged customer approach for evaluating the response time distribution in a discrete state Markovian queueing network. A recent paper by Conway and O'Brien [6] suggests the use of a hybrid analytic-simulation method for the computation of the response time distribution.

The problem in using the tagged customer approach is the difficulty of constructing and solving rather large Markov chains. The main contribution of this paper is the use of a variation of stochastic Petri nets called stochastic reward nets (SRN) for the compact specification, and automated generation and solution of these large Markov chains (Ciardo et al. [5]). This allows us to solve large and complex models.

In this paper, we present details on the use of SRNs for the computation of response time distribution using the tagged customer approach. We briefly review the tagged customer

approach using CTMCs in Section 2. We explain the method through a simple example in Section 3. In Section 4 we introduce stochastic reward nets (SRNs) and their solution techniques. In Section 5 we return to the example and illustrate the response time computation using SRNs. We also examine the effects of varying the service disciplines and the service time distributions at a queueing center on the response time distribution. These are discussed in Sections 6 and 7 respectively. In Section 8 we compute the response time distribution for transactions in an online transaction processing system. A few concluding remarks and some possible extensions of our technique are given in Section 9.

2 The Tagged Customer Approach using CTMC

In this section we review the tagged customer approach for the computation of response time distribution using CTMCs. Consider a closed queueing network with M nodes and N customers. To begin with assume that the nodes are single server queueing stations where service is performed according to a First Come First Served (FCFS) discipline. The service time of a customer at node j is an exponentially distributed random variable with parameter μ_j. We assume that the service times are independent of each other. When a customer completes its service at node i, it moves to node j with probability v_{ij}. The stochastic matrix $\mathbf{V} = [v_{ij}]$ is called the routing matrix. Note that there are no external customer arrivals to, nor departures from the network. Hence the number of customers in the network remains fixed at N.

Let $C_i(t)$ be the number of customers at node i at time t and $\mathbf{C}(t) = (C_1(t), \ldots, C_M(t))$. Then, $\mathcal{C} = \{\mathbf{C}(t), t \geq 0\}$ is a CTMC on the state space $\mathbf{S}(N) = \{(n_1, n_2, \ldots, n_M) : n_i \geq 0, \sum_i n_i = N\}$. The following result is well known:

Theorem 1 *Suppose that* $\mathbf{V}$ *is irreducible and let* $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \ldots, \lambda_n)$ *be a solution to* $\boldsymbol{\lambda} = \boldsymbol{\lambda}\mathbf{V}$. *Then the steady-state distribution of* $\{\mathbf{C}(t), t \geq 0\}$ *is given by*

$$\boldsymbol{\pi}_N(n) = \lim_{t \to \infty} P[\mathbf{C}(t) = \mathbf{n}] = G(N)\Pi_{i=1}^{M} \lambda_i^{n_i}, \mathbf{n} \in \mathbf{S}(N),$$

where $G(N)$ *is the normalization constant.*

In practice closed queueing networks arise as follows. A customer completing service at node i moves to node j with probability p'_{ij} or leaves the system altogether with probability $d_i = 1 - \sum_{j=1}^{M} p'_{ij}$. When a customer leaves the system it is instantaneously replaced by a new customer at node j with probability $d_{ij}, (\sum_{j=1}^{M} d_{ij} = 1)$, thus keeping the number of customers equal to N. The newly arrived customer is statistically identical to the departing customer. This can be modeled by a closed queueing network with routing matrix $v_{ij} = p'_{ij} + d_i d_{ij}$. Note that $\sum_{j=1}^{M} v_{ij} = 1$.

With this model we can define the response time of a customer as the amount of time that a customer spends in the network before departing. The following theorem from Sevcik and Mitrani [16] (Lavenberg and Reiser [8]) is useful in this context. Let Y_k be the state of the network just after the kth departure but just before the compensating customer is admitted to the system. Then Y_k is also the state of the system as seen by the $(k+1)$st entering customer (not including itself).

Theorem 2 $\lim_{k\to\infty} P[Y_k = \mathbf{n}] = \boldsymbol{\pi}_{N-1}(\mathbf{n}), \mathbf{n} \in \mathbf{S}(N-1)$.

This theorem forms the basis of the "tagged customer" approach for computing the response time distribution. Consider an arbitrary customer arriving at queueing station L and tag it. Theorem 2, also referred to as the Arrival Theorem, states that in a closed queueing network an arriving customer would see the network in equilibrium with one less customer. Thus if we are evaluating a system with N customers, the tagged customer sees the network in equilibrium with $N-1$ customers. The arrival theorem gives the probability distribution for the state of the system as seen by the arriving customer.

By tracking the movement of the tagged customer until it departs from the system, conditioned on the state of the system upon entry, we are able to obtain the conditional response time distribution. Construct a modified CTMC to keep track of the tagged customer as follows. Assume that the tagged customer enters the system at time 0. Let $K(\theta)$ represent the index of the node where the tagged customer is located at time θ and $J(\theta)$ be its position in the queue at node $K(\theta)$. If the tagged customer has left the network by time θ we set $K(\theta) = J(\theta) = 0$. Let $\mathcal{C} = \{\mathbf{C}(\theta), \theta \geq 0\}$ be the stochastic process describing the closed queueing network with $N-1$ customers in it. Then the modified CTMC is obtained as $\mathcal{C}' = \{(\mathbf{C}(\theta); (K(\theta), J(\theta))), \theta \geq 0\}$ where $\mathbf{C}(\theta) \in \mathbf{S}(N-1)$ describes the state of the $N-1$ untagged customers in the queueing network, while $(K(\theta), J(\theta))$ keep track of the tagged customer.

Let $R = \min\{\theta \geq 0 : (K(\theta), J(\theta)) = (0,0)\}$ be the first passage time in this modified CTMC. Then, R is the response time of the tagged customer, and hence the response time of an arbitrary customer. Thus the computation of the response time distribution is reduced to the computation of a first passage time (or time to absorption) for a CTMC.

To compute the unconditional response time distribution of the tagged customer, we use the arrival theorem (Theorem 2) to establish that the arriving customer sees the network in steady-state with one less customer. This is accomplished by setting the initial probability of those states in the modified CTMC $\mathcal{C}'$ in which the tagged customer has just arrived to the steady state probability of the corresponding state of the original CTMC ($\mathcal{C}$) with one less customer. More precisely:

$$P[(\mathbf{C}(0); (K(0), J(0))) = (\mathbf{n}; (L, n_L + 1))] = \boldsymbol{\pi}_{N-1}(\mathbf{n}), \quad 1 \leq L \leq M$$

where $\mathbf{n} = (n_1, n_2, \ldots, n_M) \in \mathbf{S}(N-1)$. All the other states in $\mathcal{C}'$ are assigned an initial probability zero. Let $\boldsymbol{\Lambda}$ be the set of absorbing states in the CTMC that is used to track the tagged customer. Then $\boldsymbol{\Lambda} = \{(\mathbf{n}; (0,0)) : \mathbf{n} \in \mathbf{S}(N-1)\}$ and,

$$P[R \leq \tau] = \sum_{j \in \boldsymbol{\Lambda}} P_j(\tau)$$

where $P_j(\tau)$ is the probability that the CTMC $\mathcal{C}'$ is in state j at time τ.

One major problem in implementing this approach is that the input to the numerical procedure that computes the distribution of R is the rate matrix of the CTMC $\mathcal{C}'$. However, building this rate matrix is a non-trivial task. We illustrate this by an example in the following section and later show how this task can be automated by using stochastic reward nets.

3 A Motivating Example

Computing the response time distribution using the tagged customer approach is a two-step process.

1. Compute the steady-state probabilities for each of the states of the queueing network with one less customer, $\boldsymbol{\pi}_{N-1}(\mathbf{n})$.

2. Use these probabilities to compute the response time distribution, $P[R \leq t]$.

The following simple example illustrates each of these steps.

3.1 System Description

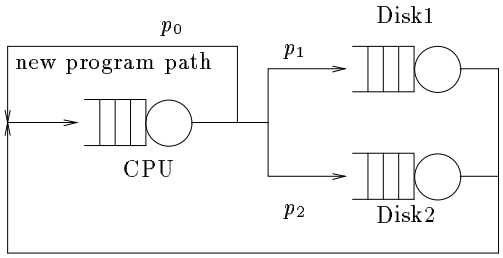
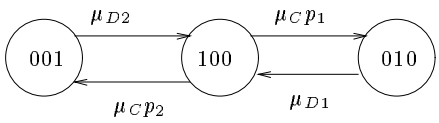


Figure 1: Central Server Model of a Computer System.

Let us consider a central server model (CSM) of a computing system as shown in Figure 1. We assume that the service discipline at all the queues is FCFS and the service time

distributions are exponential. The service rates of the CPU, Disk1 and Disk2 are μ_C, μ_{D1} and μ_{D2} respectively. When a customer finishes receiving a burst of service at the CPU, it will request access to Disk1 or Disk2 with probability p_1 and p_2 respectively, After completing the disk access, the customer rejoins the CPU queue for another burst of service. The customer will complete execution and exit the system with probability $p_0 = 1 - (p_1 + p_2)$. At the same time a statistically identical customer enters the system as indicated by the *new program path* in the figure. We assume that there are N customers in the system. For this model we define the response time as the amount of time elapsed from the instant at which the customer enters the CPU queue for its first service until the instant at which it emerges on the new program path.

3.2 Computing Initial State Probabilities



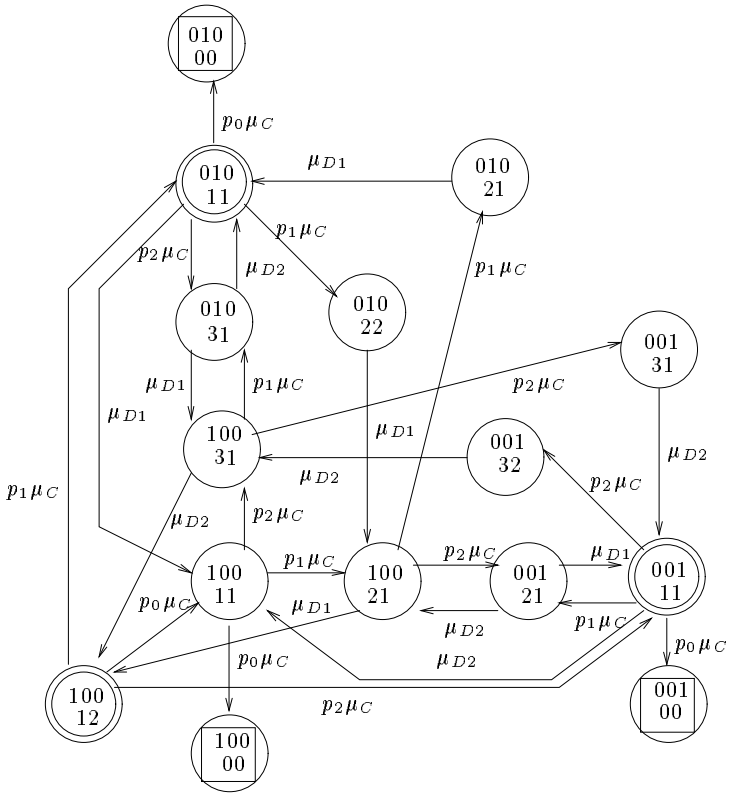
State label : $(i\ j\ k)$
i: No. of customers in CPU
j: No. of customers in Disk1
k: No. of customers in Disk2

Figure 2: CTMC for Computing the Steady-state Distribution of the Non-tagged Customer.

As mentioned earlier, the response time distribution computation can be formulated in terms of the absorption time distribution of a CTMC. Suppose we need to solve for the response time distribution of the CSM with two customers ($N = 2$). The first step in the process involves the solution of the queueing network with $N - 1$ customers, which in our case is 1. The corresponding CTMC ($\mathcal{C}$) is shown in Figure 2. This figure has three states corresponding to the non-tagged customer being present at the CPU, Disk1 and Disk2, respectively. The three components in the labels of the state correspond to the number of customers at the CPU, Disk1 and Disk2 respectively.

3.3 Computing the Response Time Distribution

The second step involves constructing the modified CTMC $\mathcal{C}'$ whose absorption time distribution yields the response time distribution. The corresponding CTMC is shown in Figure 3.



State label : $(i\ j\ k\ l\ m)$

i: No. of customers other than the tagged customer in CPU
j: No. of customers other than the tagged customer in Disk1
k: No. of customers other than the tagged customer in Disk2
l: Queue in which tagged customer is present
(1=CPU, 2=Disk1, 3=Disk2)
m: Position of tagged customer in queue

Figure 3: CTMC for Computing the Response Time Distribution.

In this figure, the first three components of the state label correspond to the number of customers in the CPU, Disk1 and Disk2 respectively, other than the tagged customer ($\mathbf{C}(\theta)$). The next two components give the position of the tagged customer; the first one is the index of the queue in which the tagged customer is residing ($K(\theta)$) and the second corresponds to the position of the tagged customer in the queue ($J(\theta)$). The queues are numbered as: 1 (CPU), 2 (Disk1) and 3 (Disk2). Here, $(K(\theta), J(\theta)) = 00$ indicates that the tagged customer has departed from the system. There are three absorbing states in the Markov chain, namely, (10000), (01000) and (00100). These states are explicitly identified in the figure by the squares enclosed within the circles. The tagged customer may arrive into the queueing system in states (10012), (01011) and (00111) which correspond to the other customer being at the CPU, Disk1 and Disk2 respectively. These three states are explicitly identified in the figure by double circles.

The important observation to be made here is that even a network with two customers and three queueing centers results in a fairly complex Markov chain (Figure 3). With increase in number of customers (or queueing stations), the Markov chain may even have more than 100,000 states, with a non-trivial rate matrix. Thus it is crucial that we have higher level descriptions from which such large rate matrices can be automatically generated and solved. Stochastic reward nets (SRNs) provide such a higher level description language. They are much simpler to describe than Markov models, and they can be mapped onto Markov reward models automatically. We will introduce SRN models and illustrate their use in computing response-time distributions in the following sections.

4 Stochastic Reward Nets

4.1 Definitions and Basic Terminology

A stochastic reward net (SRN) is an extension of a stochastic Petri net (SPN). The latter is in turn an extension of Petri nets. A rigorous mathematical description of stochastic reward nets may be found in Ciardo et al. [4].

A Petri net (PN) is a bipartite directed graph whose nodes are divided into two disjoint sets called *places* and *transitions* (Peterson [13]). Directed arcs in the graph connect places to transitions (called input arcs) and transitions to places (called output arcs). A *marked Petri net* is obtained by associating *tokens* with places. A *marking* of a PN is the distribution of tokens in the places of the PN.

In a graphical representation of a PN, places are represented by circles, transitions are represented by bars and the tokens are represented by dots or integers in the places. Input places of a transition are the set of places which are connected to the transition through input arcs. Similarly, output places of a transition are those places to which output arcs

are drawn from the transition.

A transition is considered *enabled* in the current marking if each of its input places contains at least one token. The *firing* of a transition is an atomic action in which one or more tokens are removed from each input place of the transition and one or more tokens are added to each output place of the transition, possibly resulting in a new marking of the PN.

A multiplicity may be associated with an arc of the PN, whereby a transition is enabled only if the number of tokens in each input place is at least equal to the the multiplicity of the input arc from that place. Upon firing the transition, the number of tokens deposited in each of its output places is equal to the multiplicity of the output arc. If the multiplicity of an arc is a function of the marking, then the arc is a *variable cardinality arc*.

If an *inhibitor* arc is drawn from a place to a transition then the transition cannot fire if the place contains at least one token. A multiple inhibitor arc implies that the place must contain at least as many tokens as the multiplicity (possibly marking-dependent) of the arc, to prevent the transition from firing.

A (boolean) *guard* $\mathcal{G}(.)$ can be associated with each transition (Ciardo et al. [5]). Whenever a transition satisfies all the input and inhibitor conditions in a marking M, the guard is evaluated. The transition is considered enabled only if the guard function $\mathcal{G}(M) = TRUE$.

A *Priority* relationship defines a partial order among transitions. Thus a priority relationship between two transitions t_1 and t_2 can be defined for example as $t_1 > t_2$ implying that t_1 has higher priority compared to t_2. In this case, whenever t_1 is enabled, then t_2 is automatically disabled, since t_1 has priority over t_2. This added flexibility provides a simple way to model the situation where $t_1 > t_2$, $t_3 > t_4$, but t_1 has no priority relation with respect t_3 or t_4.

Each distinct marking of the PN constitutes a separate state of the PN. A marking is *reachable* from another marking if there is a sequence of transition firings starting from the original marking which results in the new marking. The *reachability set (graph)* of a PN is the set (graph) of markings that are reachable from the other markings. In any marking of the PN, a number of transitions may be simultaneously enabled.

Associating exponentially distributed *firing times* with the transitions results in a *stochastic Petri net* (Molloy [12]). Allowing transitions to have either zero firing times (immediate transitions) or exponentially distributed firing times (timed transitions) gives rise to the generalized stochastic Petri net (GSPN) (Ajmone-Marsan et al. [1]). (Timed transitions are represented by hollow rectangles while immediate transitions are represented by thin bars.)

The markings of a GSPN are classified into two types. A marking is *vanishing* if at least one immediate transition is enabled in the marking and is *tangible* otherwise. Conflicts among immediate transitions in a vanishing marking are resolved using a *random switch* [1].

The firing rates of the timed transitions and firing probabilities of immediate transitions maybe marking dependent.

By associating reward rates with the markings of the SPN we obtain stochastic reward nets (SRN) (Ciardo et al. [4]). An SRN can be automatically converted into a Markov reward model thus permitting the evaluation of not only performance and availability but also their combination. Putting all this together, we define the SRN as (Ciardo et al. [4]):

Definition 1 *A (marked) SRN is a tuple*

$$A = (P, T, DI, DO, DH, \mathcal{G}, >, \lambda, PS, M_0, \mathbf{r})$$

where

- $P = \{p_1, \ldots, p_N\}$ *is a finite set of places.*
- $T = \{t_1, \ldots, t_M\}$ *is a finite set of transitions.*
- $\forall p_i \in P, \forall t_j \in T, DI_{i,j} : \mathbb{N}^N \rightarrow \mathbb{N}$ *is the marking dependent multiplicity of the input arc from place p_i to transition t_j; if the multiplicity is zero, the input arc is absent.*
- $\forall p_i \in P, \forall t_j \in T, DO_{i,j} : \mathbb{N}^N \rightarrow \mathbb{N}$ *is the marking dependent multiplicity of the output arc from transition t_j to place p_i; if the multiplicity is zero, the output arc is absent.*
- $\forall p_i \in P, \forall t_j \in T, DH_{i,j} : \mathbb{N}^N \rightarrow \mathbb{N}$ *is the marking dependent multiplicity of the inhibitor arc from place p_i to transition t_j; if the multiplicity is zero, the inhibitor arc is absent.*
- $\forall t_j \in T, \mathcal{G}_j : \mathbb{N}^N \rightarrow \{0, 1\}$ *is the marking dependent guard of transition t_j.*
- $>$ *is a transitive and irreflexive relation imposing a priority among transitions. In a marking M_j, t_1 is enabled iff it satisfies its input and inhibitor conditions, its guard evaluates to 1, and no other transition t_2 exists such that $t_2 > t_1$, and t_2 satisfies all other conditions for enabling.*
- $\forall t_j \in T$, *such that t_j is a timed transition,* $\lambda_j : \mathbb{N}^N \rightarrow \mathbb{R}^+$ *is the marking dependent firing rate of transition t_j, and $\lambda = [\lambda_j]$.*
- $\forall t_j \in T$, *such that t_j is an immediate transition,* $PS_{t_j} : \mathbb{N}^N \rightarrow [0, 1]$ *is the marking dependent firing probability for transition t_j, given that the transition is enabled.*
- $M_0 \in \mathbb{N}^N$ *is the initial marking.*
- $r_j \in \mathbb{R}$ *is a reward rate associated with each tangible marking M_j that is reachable from the initial marking M_0, and $\mathbf{r} = [r_j]$.*

4.2 Solution Techniques for SRN model

The behavior of a SRN can be entirely described by generating its extended reachability graph (ERG) (Ajmone Marsan et al. [1]). The ERG corresponding to a SRN can be constructed as follows. Each node in this graph corresponds to a marking of the SRN. If the firing of a transition t_i changes the marking of the SRN from M_j to M_k, then an arc is drawn from the node corresponding to marking M_j to the node corresponding to marking M_k. Depending on whether the transition is immediate or timed, the arcs are labeled with probabilities or rates. The time complexity for the generation of ERG from the SRN is $O(MNn_M + eN\log n_M)$ where n_M is the number of markings in the ERG, e is the number of arcs in the ERG, M is the number of transitions and N is the number of places in the SRN.

The extended reachability graph can be converted into a CTMC by eliminating the arcs labeled with probabilities. Let $\mathcal{V}$ represent the set of vanishing markings and $\mathcal{T}$ represent the set of tangible markings. The matrix $\mathbf{P}^{\mathcal{V}} = [\mathbf{P}^{\mathcal{VV}}|\mathbf{P}^{\mathcal{VT}}]$ describes the probability of transition from vanishing to vanishing ($\mathbf{P}^{\mathcal{VV}}$) or tangible ($\mathbf{P}^{\mathcal{VT}}$) markings, and $\mathbf{U}^{\mathcal{T}} = [\mathbf{U}^{\mathcal{TV}}|\mathbf{U}^{\mathcal{TT}}]$ gives the rates of transitions from the tangible markings to vanishing ($\mathbf{U}^{\mathcal{TV}}$) or tangible ($\mathbf{U}^{\mathcal{TT}}$) markings. The generator matrix $\mathbf{Q}$ of the underlying CTMC is obtained from the matrix description of the ERG using the following equation (Ajmone Marsan et al. [1]):

$$\mathbf{Q} = \mathbf{U}^{\mathcal{TT}} + \mathbf{U}^{\mathcal{TV}}W \tag{1}$$

where $[I - \mathbf{P}^{\mathcal{VV}}]W = \mathbf{P}^{\mathcal{VT}}$. Let $n_{\mathcal{T}} = |\mathcal{T}|$, $n_{\mathcal{V}} = |\mathcal{V}|$ and $\eta_{\mathcal{VV}}$ and $\eta_{\mathcal{TV}}$ be the number of nonzero entries in $(I - \mathbf{P}^{\mathcal{VV}})$ and $\mathbf{U}^{\mathcal{TV}}$, respectively. If $\eta_{\mathcal{VV}}$ is $O(n_{\mathcal{V}})$ and $\eta_{\mathcal{TV}}$ is $O(n_{\mathcal{V}} + n_{\mathcal{T}})$ then a bound on the number of floating point operations required by this transformation of the ERG to CTMC is $O(n_{\mathcal{T}}(\eta_{\mathcal{VV}} + \eta_{\mathcal{TV}}))$.

The transient behavior of a CTMC can be described by the Kolmogorov differential equation (Trivedi [17]):

$$\frac{d}{dt}\mathbf{P}(t) = \mathbf{P}(t)\mathbf{Q}, \qquad \mathbf{P}(0) = \mathbf{p}_0. \tag{2}$$

Here $\mathbf{P}(t)$ is the state probability vector at time t and $\mathbf{p}_0$ represents the initial probability vector of the CTMC.

By taking limits of both sides of the above equation, we get the equation for the steady-state probability vector $\boldsymbol{\pi}$ of the CTMC:

$$\boldsymbol{\pi}\mathbf{Q} = 0, \qquad \sum \pi_i = 1. \tag{3}$$

Numerical methods for obtaining the steady-state and transient solutions of SRN models are described in detail in Ciardo et al. [4].

4.2.1 Computation of Measures at the SRN Level

Suppose X represents the random variable corresponding to the reward rate in steady-state, then the expected reward rate $E[X]$ can be computed as,

$$E[X] = \sum_{M_i \in \mathcal{T}} r_i \pi_i$$

where π_i is the steady-state probability of tangible marking M_i (the probability of a vanishing marking is zero).

Besides the steady-state analysis of SRNs we also carry out transient and cumulative transient analysis. Thus, the expected value of the reward rate as a function of time, $E[X(t)]$, can be computed as,

$$E[X(t)] = \sum_{M_i \in \mathcal{T}} r_i P_i(t)$$

where $P_i(t)$ is the probability of tangible marking M_i at time t.

Alternatively, we might be interested in the accumulated reward, $Y(t)$, over the interval $[0, t)$. The expected value of the accumulated reward, $E[Y(t)]$, can be computed as,

$$E[Y(t)] = \sum_{M_i \in \mathcal{T}} r_i \int_0^t P_i(\tau) d\tau.$$

The expected accumulated reward until absorption $E[Y(\infty)]$ can also be computed as,

$$E[Y(\infty)] = \sum_{M_i \in \mathcal{T}} r_i \int_0^\infty P_i(\tau) d\tau.$$

It must be noted that the definition of reward rates is orthogonal to the analysis type that is used. Thus with the same reward definition we can compute the steady-state expected reward rate as well as instantaneous reward rate at time t, expected accumulated reward and expected time-averaged reward over the interval [0,t).

In the following sections we will discuss the use of SRNs for modeling the behavior of queueing networks and illustrate how the response time distribution can be computed by associating an appropriate reward rate to the SRN model.

5 The Tagged Customer Method for the CSM using SRNs

The response time distribution computation using the SRN proceeds in two steps as illustrated in Figure 4. We will illustrate this procedure for the central server queuing network model shown in Figure 1.

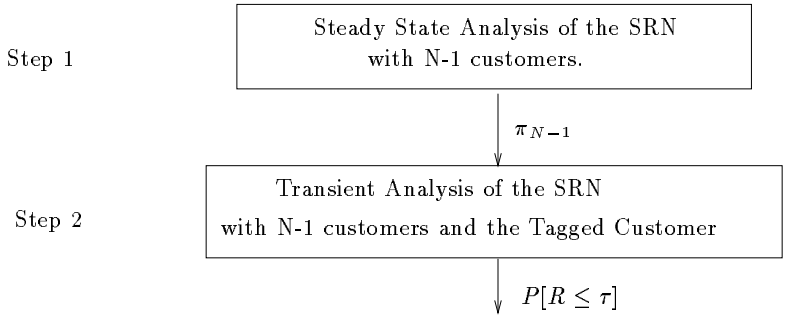


Figure 4: Steps in Computing Response Time Distribution for a Closed Queueing Network.

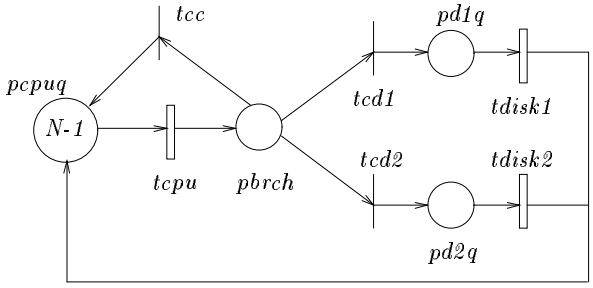


Figure 5: SRN Model of the Central Server System.

5.1 Computing Initial State Probabilities using SRNs

Given that there are N customers in the system, the first step involves solving for the steady-state probabilities $\boldsymbol{\pi}_{N-1}$ with one less customer. To evaluate these probabilities, we construct an SRN model corresponding to the queueing network as shown in Figure 5. In this figure, places *pcpuq*, *pd1q* and *pd2q* represent the CPU, Disk1 and Disk2 queues. Timed transitions *tcpu*, *tdisk1* and *tdisk2* represent the service of customers at the CPU, Disk1 and Disk2 respectively. Immediate transitions *tcc*, *tcd1* and *tcd2* represent the branching of customers after a CPU burst. The corresponding probabilities for these immediate transitions are given by p_0, p_1 and p_2 respectively.

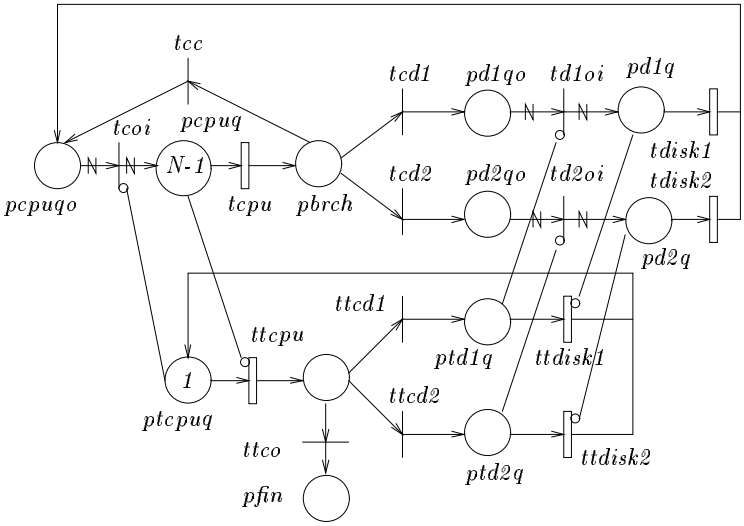
We compute the steady-state probabilities of the system being in state $(i, j, N-(i+j+1))$ with i customers at the CPU, j customers in Disk1 and the remaining $N-(i+j+1)$ customers in Disk2. In the SRN, the state $(i, j, N-(i+j+1))$ corresponds to having i tokens in place *pcpuq*, j tokens in place *pd1q* and $N-(i+j+1)$ tokens in place *pd2q* respectively.

5.2 SRN Model for Computing the Response Time Distribution

In the second step, we consider the motion of the tagged customer through the network. This is obtained by the SRN shown in Figure 6. In the figures #(p) represents the number of tokens in place p and a # associated with a transition means that its firing rate is marking dependent. An inhibitor arc is represented by a circle (instead of an arrow) at its head. A zigzag line on an arc indicates that the multiplicity of that arc is marking dependent.

In the SRN of Figure 6, places *pcpuqo, pcpuq, ptcpuq* and transitions *tcoi, tcpu, ttcpu* together implement the FCFS queue corresponding to the CPU. When the tagged customer is in the queue, all customers ahead of the tagged customer are represented by the tokens in place *pcpuq* (for convenience we shall call this place the inner queue) and customers behind the tagged customer are in place *pcpuqo* (we shall also refer to this place as the catchment area). The tagged customer will be scheduled for service only when all the customers ahead of it receive service and leave the queue. This is implemented by the inhibitor arc from place *pcpuq* to transition *ttcpu*. All customers arriving after the tagged customer are deposited in place *pcpuqo*. These customers cannot move into the inner queue while the tagged customer is still in the queue. This is implemented by the inhibitor arc from place *ptcpuq* to transition *tcoi*. These customers are moved into the inner queue when the tagged customer departs from the queue.

If the tagged customer is not in the queue, an arriving customer can directly proceed from the catchment area into the inner queue. This is implemented using the variable arc function defined in Figure 6. The variable arc function defines the multiplicity of the input arc from *pcpuqo* to *tcoi* and the output arc from *tcoi* to *pcpuq* to be equal to the number of



Transition	Guard
All	$(\#(pfin) = 0)$

Transition Priority Relationship
(tcd1, tcd2, tcc) > (tcoi, td1oi, td2oi)
(ttcd1, ttcd2, ttco) > (tcoi, td1oi, td2oi)

Arcs	Variable Arc Function
pcpuqo → tcoi and tcoi → pcpuq	$\max(1, \#(pcpuqo))$
pd1qo → td1oi and td1oi → pd1q	$\max(1, \#(pd1qo))$
pd2qo → td2oi and td2oi → pd2q	$\max(1, \#(pd2qo))$

Figure 6: SRN Model of the CSM to Compute Response Time Distribution.

tokens in *pcpuqo*, if it is not empty and 1 if *pcpuqo* is empty. Whenever, the transition *tcoi* is enabled, it will move all the tokens from *pcpuqo* to *pcpuq*. A similar mechanism is used at the disk queues. The guard is used to stop the firing of all the transitions once the token corresponding to the tagged customer reaches *pfin*.

5.3 Computing the Response Time Distribution

As we mentioned earlier, the probability vector $\boldsymbol{\pi}_{N-1}$ is used to define the initial probability distribution of the CTMC that is used in computing the response time distribution. A similar process can be adopted in using SRNs. We used the software package SPNP (Ciardo et al. [5]) to solve our SRN models. SPNP provides a mechanism for defining the initial probability vector over the set of states in the underlying Markov chain, in terms of the net level entities like the number of tokens in a place. For example, the initial probability for a state with i customers at the CPU, j customers at Disk1 and k customers in Disk2 at the instant of arrival of the tagged customer, in the SRN model shown in Figure 6 can be specified by assigning this probability to the marking in which there are i tokens in place *pcpuq*, j tokens in *pd1q* and k tokens in *pd2q*, the token corresponding to the tagged customer is in place *ptcpuq* and all the remaining places are empty. These initial probabilities are obtained by solving for the steady-state probabilities of the SRN shown in Figure 5.

The probability of place *pfin* being non-empty at time τ gives the probability that the tagged customer has completed by time τ. This can be computed as the expected value of the instantaneous reward rate $E[X(\tau)]$ by attaching a reward rate r_i with tangible marking M_i as follows:

$$r_i = \#(ptfin, M_i)$$

In this paper, $\#(p, M_i)$ represents the number of tokens in place p in marking M_i. The above reward rate specification assigns a reward rate $r_i = 1$ to all the markings in which place *pfin* is non-empty and reward rate $r_i = 0$ to all the other markings. The distribution of the response time is evaluated by computing $E[X(\tau)]$ for different values of τ. The mean time to absorption (token appearing in place *pfin*) gives the mean response time for the queueing network. This was used in validating the SRN model since the mean response time can be computed using product form solution techniques like mean value analysis and convolution.

5.4 Numerical Results

The response time distributions of the central server model for different numbers of customers (5, 10 and 15) are shown in Figure 7. In this example, we assume that $\mu_C = 50.0$, $\mu_{D1} = 30.0$, $\mu_{D2} = 20.0$, $p_1 = 0.45$ and $p_2 = 0.3$.

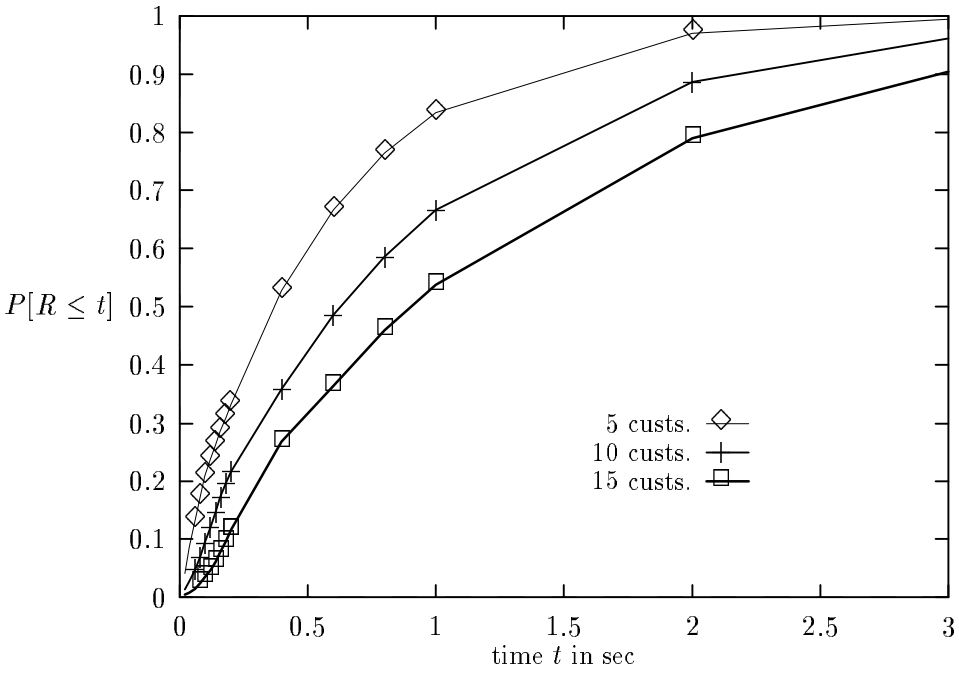


Figure 7: Response Time Distribution for Different Number of Customers.

Given that the response time distribution can be computed, the next important question is the size of the problems that can be handled by this method. Table 1 gives the number of states and the number of arcs in the Markov chain as a function of the number of customers in the CSM model with one CPU and two disks. From Table 1, one can easily observe that our method can handle queueing networks of moderate population sizes. The other important factor causing an increase in the state space is the number of queues in the model. The CSM model lends itself easily to variation in the number of queues, in that we can vary the number of disks in the model.

No. of Customers	No. of States	No. of Arcs
5	120	335
10	715	2420
15	2160	7880
20	4830	18340
25	9100	35425
30	15345	60760
35	23940	95970

Table 1: Markov Chain Sizes For Different Number of Customers.

No. of Disks	No. of Customers	No. of States	No. of Arcs
2	5	120	335
3	5	315	1120
4	5	700	2926
5	5	1386	6510
2	10	715	2420
3	10	3080	13915
4	10	10725	59202
5	10	32032	205205
2	15	2160	7880
3	15	12920	64260
4	15	61200	347920

Table 2: Markov Chain Sizes vs the Number of Disks.

Table 2 summarizes the sizes of the Markov chains as a function of both the number of disks in the CSM model as well as the number of customers. It is easy to see from the table

that with an increase in the number of queues there is a dramatic increase in the size of the state space. However, we have successfully generated and solved CTMCs with nearly half a million states.

6 Effect of Different Service Disciplines

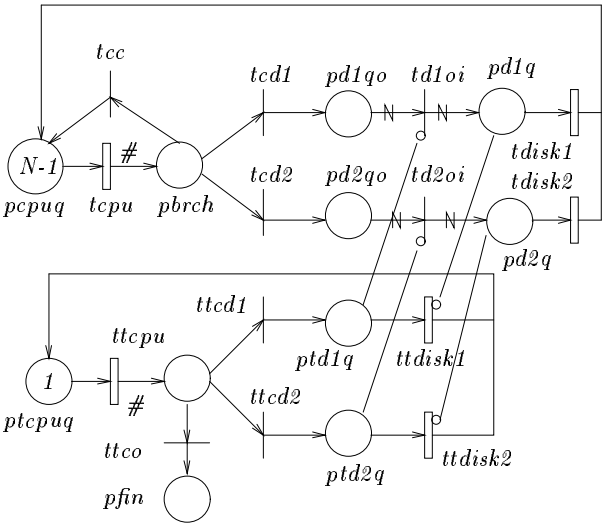
From the BCMP theorem (Baskett et al. [2]), we know that the mean response time for a customer in a queueing network is independent of the service disciplines at queueing centers as long as the service rates remain the same and the service discipline is either first come first served (FCFS), processor sharing (PS) or last come first served preemptive resume (LCFSPR). However, the response time distribution is sensitive to the service discipline. In this section, we will vary the service discipline at the CPU for the central server model and compare the response time distribution for each of the cases. In each of these cases (FCFS, PS and LCFSPR) the SRN model used to compute the steady-state probabilities remains the same and is shown in Figure 5.

6.1 SRN with Tagged Customer for the PS Service Discipline

Figure 8 shows the SRN model used in computing the response time distribution of the CSM with PS discipline at the CPU. We notice that modeling PS is easier than FCFS since we need not keep track of the position of the tagged customer in the CPU queue. The firing rates of transitions *tcpu* and *ttcpu* are now marking dependent since each customer receives a fraction of the CPU service based on the number of customers in the queue.

6.2 SRN with Tagged Customer for the LCFSPR Service Discipline

Figure 9 shows the SRN model used in computing the response time distribution of the CSM with LCFSPR service discipline at the CPU. In this figure we notice that whenever the tagged customer arrives into the CPU queue (*ptcpuqo*), all the customers waiting at the CPU, in place *pcpuq*, are moved into the outer queue *pcpuqo* since the incoming tagged customer preempts the customer in service. This is implemented by adding a place *ptcpuqo* (to signal the arrival of the tagged customer into the CPU queue), the immediate transition *tcio* together with the variable multiplicity arcs from *pcpuq* to *tcio* and *tcio* to *pcpuqo*. Similarly, any customer coming to the CPU after the tagged customer will preempt the tagged customer. This is handled by the inhibitor arc from place *pcpuq* to the transition *ttcpu*. When the tagged customer departs from the CPU queue, all the previously preempted customers may resume execution. This is handled by the immediate transition *tcoi* together with its variable cardinality input and output arcs.



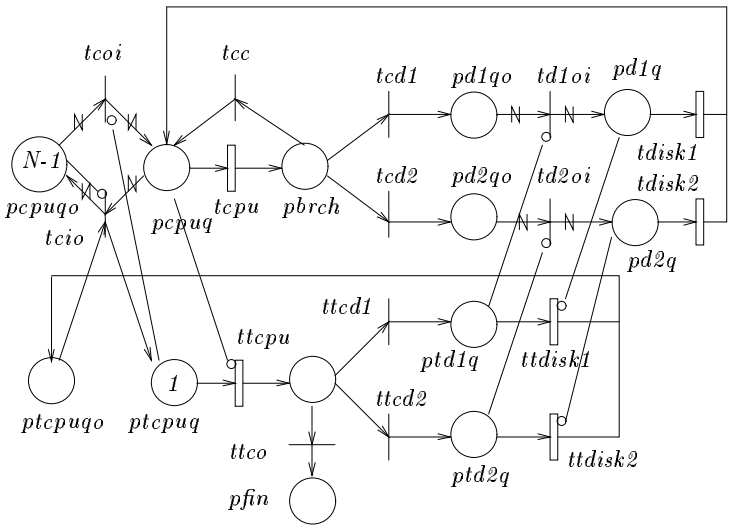
Transition	Guard
All	$(\#(pfin) = 0)$

Transition	Rate Function
tcpu	$\mu_C \#(pcpuq)/(\#(pcpuq) + \#(ptcpuq))$
ttcpu	$\mu_C/(\#(pcpuq) + \#(ptcpuq))$

Transition Priority Relationship
(tcd1, tcd2, tcc) > (td1oi, td2oi)
(ttcd1, ttcd2, ttco) > (td1oi, td2oi)

Arcs	Variable Arc Function
pd1qo → td1oi and td1oi → pd1q	$\max(1, \#(pd1qo))$
pd2qo → td2oi and td2oi → pd2q	$\max(1, \#(pd2qo))$

Figure 8: SRN for Computing the Response Time Distribution with PS Discipline.



Transition	Guard
All	$(\#(pfin) = 0)$

Transition Priority Relationship
(tcd1, tcd2, tcc) > (tcoi, td1oi, td2oi) > tcio
(ttcd1, ttcd2, ttco) > (tcoi, td1oi, td2oi)

Arcs	Variable Arc Function
pcpuq → tcio and tcio → pcpuqo	$\#(pcpuq)$
pcpuqo → tcoi and tcoi → pcpuq	$\max(1, \#(pcpuqo))$
pd1qo → td1oi and td1oi → pd1q	$\max(1, \#(pd1qo))$
pd2qo → td2oi and td2oi → pd2q	$\max(1, \#(pd2qo))$

Figure 9: SRN for Computing the Response Time Distribution with LCFSPR Discipline.

6.3 Numerical Results

Figure 10 plots the response time distribution for the three different service disciplines. In all the cases, we assume that the number of customers is 10. We can see that both PS and LCFSPR perform better than FCFS for smaller time values. Both PS and LCFSPR disciplines tend to favor short customers at the cost of additional delays for long customers. In the LCFSPR environment a long customer has a much higher probability of being preempted than a short customer. All the three curves have crossover points because the mean response time is the same in all the three cases.

7 Effect of Different Service Time Distributions

The BCMP theorem (Baskett et al. [2]) also states that the mean response time is independent of the nature of the service time distribution at a queueing center as long as the service discipline is either PS or LCFSPR, the mean service time remains the same, and the service time distribution has a rational Laplace transform. In this section we study the effect of changing the service time distribution at the CPU on the response time distribution of a customer. We assume that the service discipline at the CPU is PS. We know that the coefficient of variation C_v for exponential distribution is 1. It would be interesting to study the queueing system for $C_v < 1$ and $C_v > 1$. We know that C_v is greater than 1 for hyper-exponential distributions and C_v is less than 1 for hypo-exponential and Erlang distributions. For the same number of stages, the Erlang distribution gives a smaller C_v compared to a hypo-exponential distribution. Thus we choose to compare the response time distribution for the exponential, 4-stage Erlang and 3-stage hyper-exponential service time distributions. The parameters of these distributions ($\mu_{C1}, \ldots, \mu_{C4}$ for Erlang and $\mu_{C1}, \ldots, \mu_{C3}$ for hyper-exponential) are chosen in such a way that the mean remains the same in all the cases.

7.1 SRNs for 4-stage Erlang Distribution

The SRN model used to compute the steady-state probabilities for the CSM with 4-stage Erlang service distribution at the CPU is shown in Figure 11. The SRN model used to compute the response time distribution in this case is shown in Figure 12. In these figures we notice how the four stages of the Erlang distribution are modeled using four places *pcpuq1* through *pcpuq4* and four transitions *tcpu1* through *tcpu4* with exponentially distributed firing times. The marking dependent rates of each of these transitions are also shown. Thus a customer could be in any of the four stages of the Erlangian service.

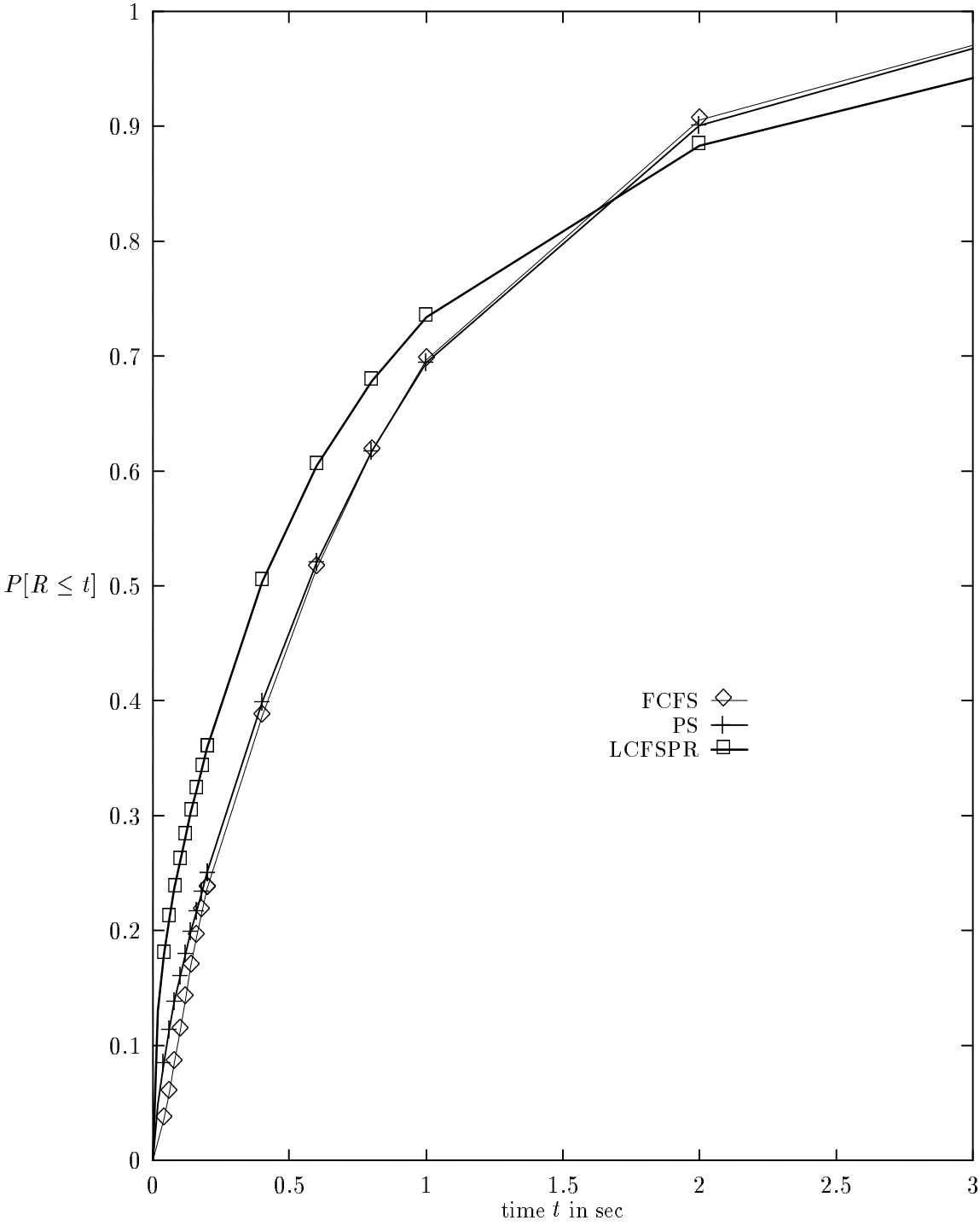
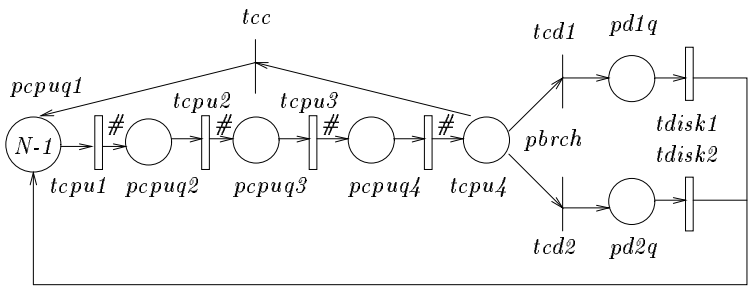


Figure 10: Response Time Distribution for Different Service Disciplines.



Transition	Rate Function
tcpu1	$\mu_{C1}\#(pcpuq1)/(\#(pcpuq1)+\#(pcpuq2)+\#(pcpuq3)+\#(pcpuq4))$
tcpu2	$\mu_{C2}\#(pcpuq2)/(\#(pcpuq1)+\#(pcpuq2)+\#(pcpuq3)+\#(pcpuq4))$
tcpu3	$\mu_{C3}\#(pcpuq3)/(\#(pcpuq1)+\#(pcpuq2)+\#(pcpuq3)+\#(pcpuq4))$
tcpu4	$\mu_{C4}\#(pcpuq4)/(\#(pcpuq1)+\#(pcpuq2)+\#(pcpuq3)+\#(pcpuq4))$

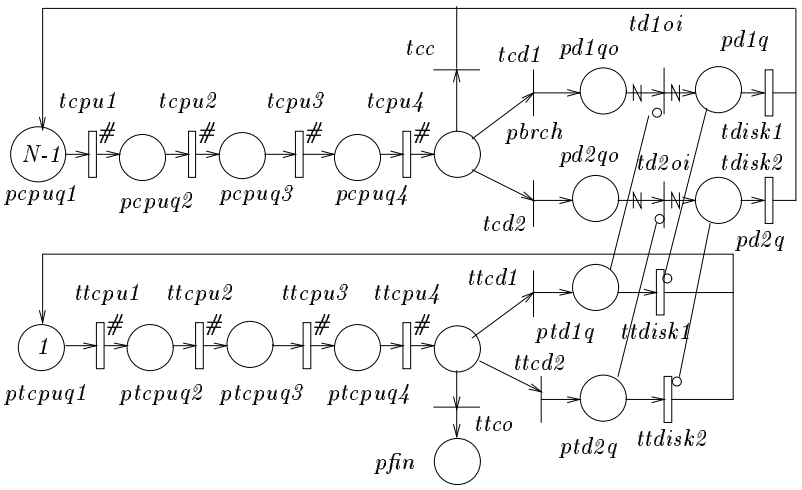
Figure 11: CSM with Erlang Service Time Distribution.

7.2 SRNs for 3-stage Hyper-Exponential Distribution

The SRN model used to compute the steady-state probabilities for the CSM with the three stage hyper-exponential service time distribution at the CPU is shown in Figure 13. The SRN model used to compute the response time distribution in this case is shown in Figure 14. Each of the three stages of the hyper-exponential distribution are again modeled using three places *pcpuq1* through *pcpuq3* and three transitions *tcpu1* through *tcpu3*. A customer arriving at the CPU may enter any of the three stages with probabilities α_1, α_2 and α_3 respectively. This is modeled by the three immediate transitions *tc1* through *tc3*.

7.3 Numerical Results

Figure 15 plots the response time distribution for the three different service time distributions. For the 4-stage Erlang case, we use $\mu_{C1} = \mu_{C2} = \mu_{C3} = \mu_{C4} = 200.0$. The other parameters remain the same as in the exponential case. For the 3-stage hyper-exponential case, we use $\mu_{C1} = 120.0$, $\alpha_1 = 0.8$, $\mu_{C2} = 22.5$, $\alpha_2 = 0.15$, $\mu_{C3} = 7.5$ and $\alpha_3 = 0.05$ respectively. For the exponential case, $\mu_C = 50.0$. We note that for small time values, the hyper-exponential service times provide a lower response time than the exponential case which in turn provides a smaller response time than the Erlang case. As the mean response time is the same in each case, the three curves cross over and the order reverses after the intersection.



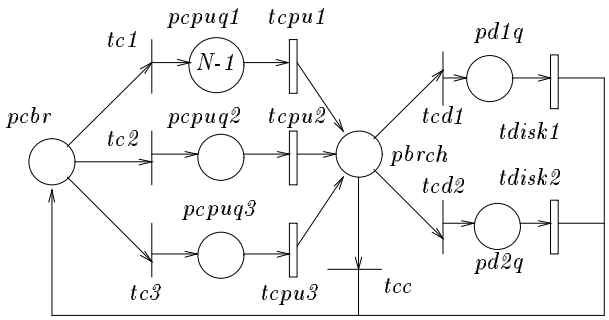
Transition	Guard
All	$(\#(pfin) = 0)$

Transition Priority Relationship
(tcd1, tcd2, tcc) > (td1oi, td2oi)
(ttcd1, ttcd2, ttco) > (td1oi, td2oi)

Arcs	Variable Arc Function
pd1qo → td1oi and td1oi → pd1q	$\max(1, \#(pd1qo))$
pd2qo → td2oi and td2oi → pd2q	$\max(1, \#(pd2qo))$

Transition	Rate Function
tcpuj (j = 1,..., 4)	$\mu_{Cj}\#(pcpuqj)/(\#(pcpuq1) + \#(pcpuq2) + \#(pcpuq3) + \#(pcpuq4)$ $+\#(ptcpuq1) + \#(ptcpuq2) + \#(ptcpuq3) + \#(ptcpuq4))$
ttcpuj (j = 1,..., 4)	$\mu_{Cj}/(\#(pcpuq1) + \#(pcpuq2) + \#(pcpuq3) + \#(pcpuq4)$ $+\#(ptcpuq1) + \#(ptcpuq2) + \#(ptcpuq3) + \#(ptcpuq4))$

Figure 12: SRN for Computing the Response Time Distribution with Erlang Service Times.



Transition	Rate Function
tcpu1	$\mu_{C1}\#(pcpuq1)/(\#(pcpuq1)+\#(pcpuq2)+\#(pcpuq3))$
tcpu2	$\mu_{C2}\#(pcpuq2)/(\#(pcpuq1)+\#(pcpuq2)+\#(pcpuq3))$
tcpu3	$\mu_{C3}\#(pcpuq3)/(\#(pcpuq1)+\#(pcpuq2)+\#(pcpuq3))$

Figure 13: CSM with Hyper-exponential Service Time Distribution.

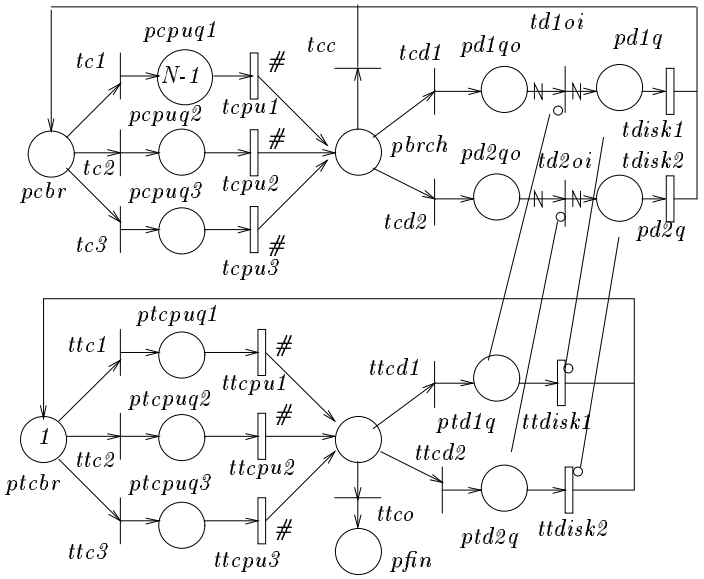
8 Multiserver and Infinite Server Models

In this section we model a practical example based on an online transaction processing system (OLTP). An OLTP system is needed when many users require fast access to information such as records in large databases. Examples of such systems include airline reservation systems and automated bank-teller systems. These systems are characterized by high throughput and high availability requirements.

8.1 The OLTP System Description

A typical architecture of an OLTP system is shown in Figure 16. The front-end of the system is composed of a transaction generator, e.g., a terminal, barcode reader etc. and transaction processors (TP) which analyze the submitted transaction to determine the information needed from the databases and also provide error recovery capabilities. The back-end of the system consists of a set of database processors (DBP) which handle reading and updating the records in the databases according to the requests submitted by the transaction processors. The transactions visit the TPs and DBPs in succession until the necessary processing is completed. Thus a good measure of performance for an OLTP system is the response time of a transaction. The OLTP can best be modeled by a closed queueing network with terminals. A queueing network model for an OLTP based on these assumptions is shown in Figure 17.

In this model it is assumed that the TPs have a single queue from which transactions are selected for processing on a first come first served (FCFS) basis. The TPs are modeled



Transition	Guard
All	$(\#(pfin) = 0)$

Transition Priority Relationship
(tc1, tc2, tc3) > (tcd1, tcd2, tcc) > (td1oi, td2oi)
(ttcd1, ttcd2, ttco) > (td1oi, td2oi)
(ttc1, ttc2, ttc3) > (tcd1, tcd2, tcc) > (td1oi, td2oi)

Arcs	Variable Arc Function
pd1qo → td1oi and td1oi → pd1q	$\max(1, \#(pd1qo))$
pd2qo → td2oi and td2oi → pd2q	$\max(1, \#(pd2qo))$

Transition	Rate Function
tcpuj (j = 1,..., 3)	$\mu_{Cj}\#(pcpuqj)/(\#(pcpuq1) + \#(pcpuq2) + \#(pcpuq3)$ $+\#(ptcpuq1) + \#(ptcpuq2) + \#(ptcpuq3))$
ttcpuj (j = 1,..., 3)	$\mu_{Cj}/(\#(pcpuqj) + \#(pcpuq2) + \#(pcpuq3)$ $+\#(ptcpuq1) + \#(ptcpuq2) + \#(ptcpuq3))$

Figure 14: SRN for Computing the Response Time Distribution with Hyper-exponential Service Times.

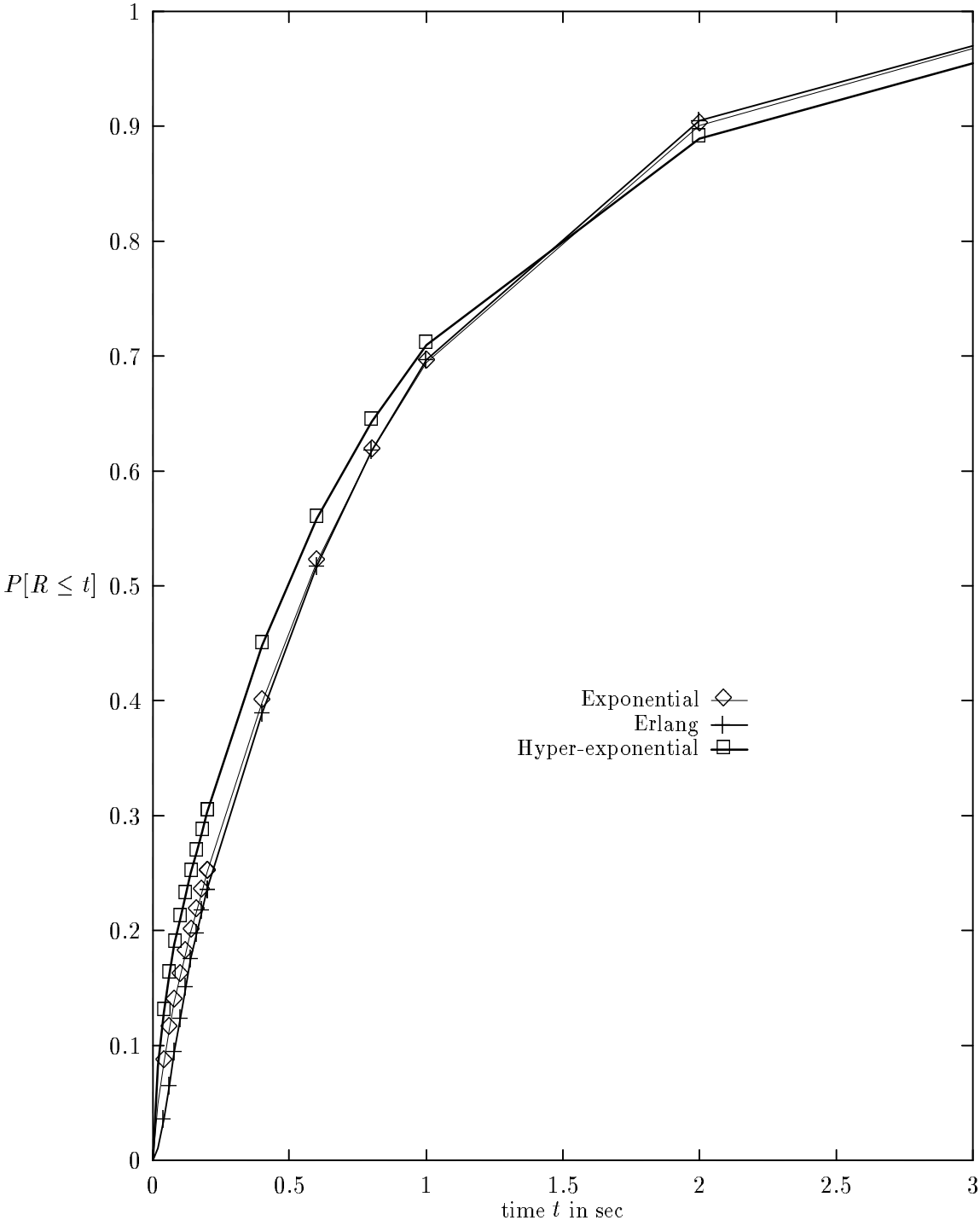


Figure 15: Response Time Distribution for Different Service Time Distributions.

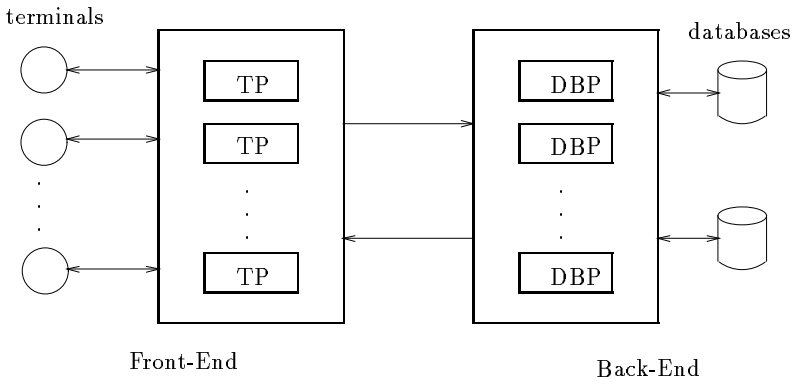


Figure 16: Architecture of an OLTP System.

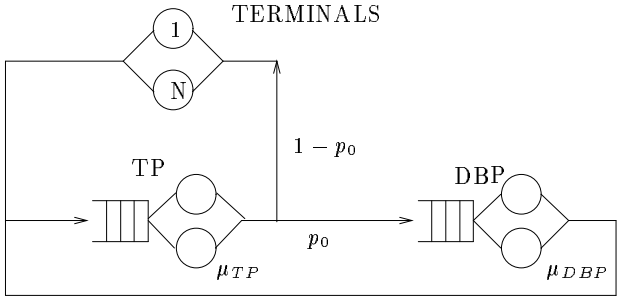
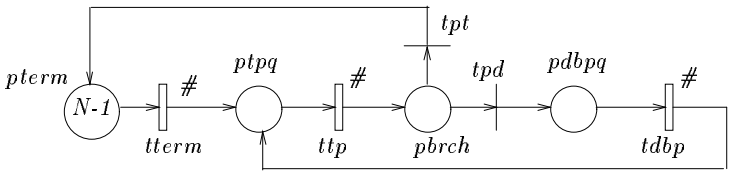


Figure 17: Closed Queueing Network Model of the OLTP system.

using a multi-server queue with the number of servers in the queue equal to the number of TPs. The DBPs are similarly configured. The service times of the TPs are exponentially distributed with rate μ_{TP} and the service times of the DBPs are exponentially distributed with rates μ_{DBP}. The average time between completion of a transaction and submission of the next transaction at a terminal, which is equivalent to the think time at the terminal, is exponentially distributed with rate λ. The number of terminals available in the system is assumed to be N. The number of TPs and the number of DBPs are assumed to be N_{TP} and N_{DBP} respectively.

8.2 The SRN Models for Computing Response Time Distribution

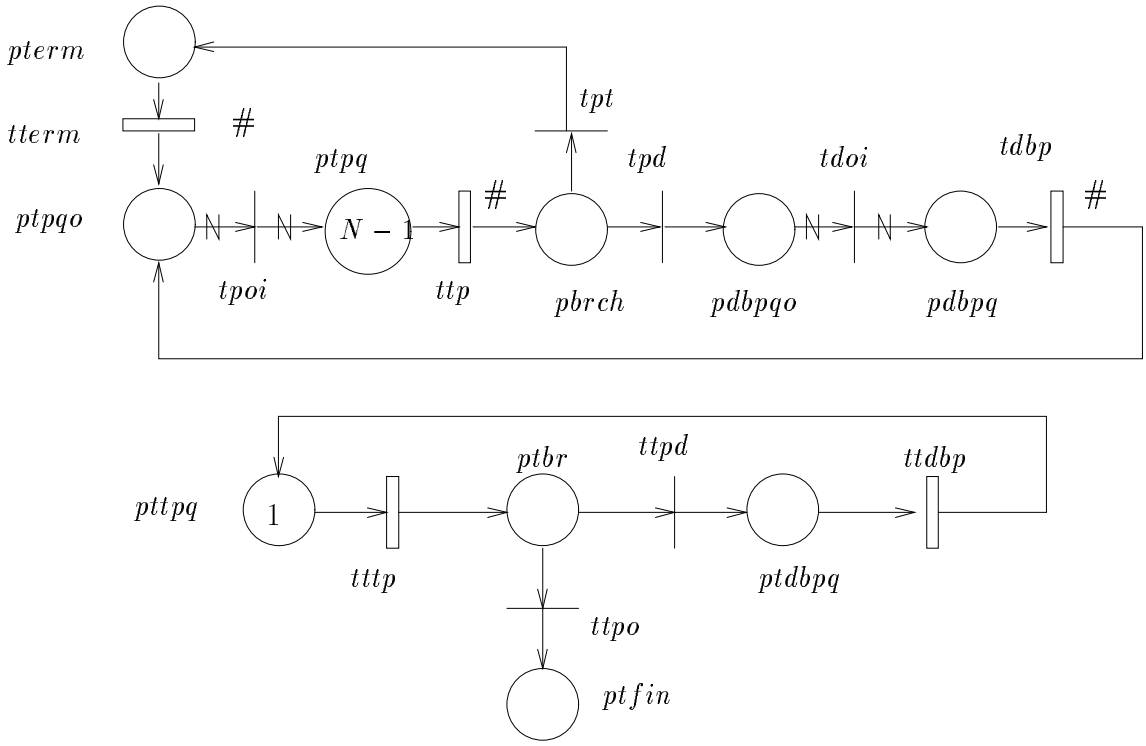


Transition	Rate Function
tterm	$\#(pterm)\lambda$
ttp	$\min(N_{TP}, \#(ptpq))\mu_{TP}$
tdbp	$\min(N_{DBP}, \#(pdbpq))\mu_{DBP}$

Figure 18: SRN of the OLTP System.

The SRN model for the steady-state computation is shown in Figure 18. The SRN model for computing the response time distribution is shown in Figure 19. We have already seen how the position of the tagged customer is tracked through a single server queue in an earlier example. In this case, we need to see how multiple servers are handled while keeping track of the tagged customer. In the SRN shown in Figure 19, the places *ptpqo, ptpq, pttpq* and the transitions *tpoi, ttp, tttp* together implement the multi-server FCFS queue corresponding to the TPs. The tagged customer will be scheduled for service only when the number of customers ahead of it becomes less than the number of TPs. This is implemented by the guard controlling the firing of the transition *tttp*. Whenever a TP becomes available and the tagged customer is already in service, another customer can be admitted into the inner queue from the catchment area and scheduled for service. A similar mechanism is used for the DBP queue.

8.3 Numerical Results



Transition	Guard
tpoi	$(\#(pttpq) = 0 \vee \#(ptpq) < N_{TP} - 1) \wedge (\#(ptbr) = 0) \wedge (\#(ptfin) = 0)$
tdoi	$(\#(ptdbpq) = 0 \vee \#(pdbpq) < N_{DBP} - 1) \wedge (\#(ptfin) = 0)$
tttp	$(\#(ptpq) < N_{TP}) \wedge (\#(ptfin) = 0)$
ttdbp	$(\#(pdbpq) < N_{DBP}) \wedge (\#(ptfin) = 0)$
others	$\#(ptfin) = 0$

Transition	Rate Function
tterm	$\#(pterm)\lambda$
ttp	$\min(N_{TP}, \#(ptpq))\mu_{TP}$
tdbp	$\min(N_{DBP}, \#(pdbpq))\mu_{DBP}$

Transition Priority Relationship
(ttpd, ttpo, tpd, tpt) > (tpoi, tdoi)

Arcs	Variable Arc Function
ptpqo → tpoi and tpoi → ptpq	1 if $(\#(pttpq) > 0)$ $\max(1, \#(ptpqo))$ otherwise
pdbpqo → tdoi and tdoi → pdbpq	1 if $(\#(ptdbpq) > 0)$ $\max(1, \#(pdbpqo))$ otherwise

Figure 19: SRN for Computing the Response Time Distribution of the OLTP System.

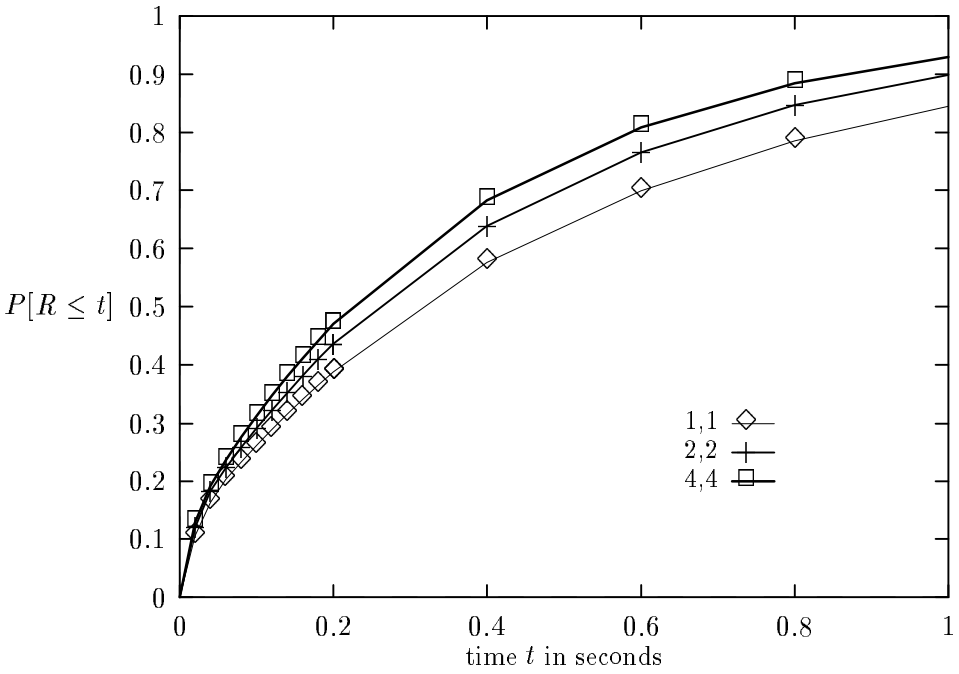


Figure 20: Response Time Distribution of the OLTP System.

The response time distributions for various configurations of the OLTP system are shown in Figure 20. We assume that the number of TPs is equal to the number of DBPs. We also assume that with every additional processor, the number of terminals is increased by 5. We set $\lambda = 1.0$, $\mu_{TP} = 50.0$, $\mu_{DBP} = 20.0$ and $p_0 = 0.8$. The plots indicate the response times for different number of TPs and DBPs ((1,1), (2,2) and (4,4)). It is interesting to note that the probability of completion for a customer at any time t increases with increase in the number of processors available even though the number of terminals is correspondingly increased. The size of the Markov chains and the number of arcs in the Markov chain for each case are shown in Table 3.

No. of TPs/DBPs	No. of Terminals	No. of States	No. of Arcs
1	5	85	205
2	10	405	1305
3	15	1088	3722
4	20	2262	7998
8	40	14428	53932
10	50	28660	108601

Table 3: Sizes of the Markov Chains for Different Configurations.

9 Conclusions and Future Extensions

We presented a method of numerically computing the response time distribution for closed product form queueing networks. The ability to generate and solve for the transient probabilities of large Markov chains using SRNs is the basis for our technique. We considered the effects of variation of the service disciplines and the service time distributions on the response time distribution. A realistic model of an OLTP system was also presented. The response time distribution is very useful in studying the performance of such real-time systems. This method of numerically computing the response time distribution can easily be applied to any moderate-sized closed BCMP network.

As we have seen earlier in this paper, our method is applicable to moderate sized queueing networks. It must be noted that the size of the queueing networks that can be handled is limited by the size of the memory available on the computer and is not a limitation of the method. In order to handle large queueing networks, we may either consider the use of a supercomputer with a large main memory or the use of approximation techniques. Several approximation methods have been suggested in the past by Raatikainen [14], Salza and

Lavenberg [15]. We are in the process of adapting these approximation techniques in the context of SRNs.

We have restricted ourselves to product form queueing networks in this paper. The primary reason is that the distribution of remaining customers at the instant of arrival of the tagged customer is easily computable in this case. Consequently, computing the unconditional response time distribution is also easier. It is of interest to see how we could extend our technique to general Markovian queueing networks which do not necessarily satisfy the product form assumptions.

The method of response time distribution computation in this paper is based on the assumption that the queueing network is in steady state when the tagged customer arrives at the queue. Transient response time distribution where we consider the response time distribution for the $n\,th$ arriving customer is also of interest.

Acknowledgements

The authors wish to thank Gianfranco Ciardo, Lorrie Tomek, Steve Woolet for their helpful comments.

References

[1] M. Ajmone Marsan, G. Conte, and G. Balbo. A class of generalized stochastic Petri nets for the performance evaluation of multiprocessor systems. *ACM Trans. Comput. Syst.*, 2(2):93–122, May 1984.

[2] F. Baskett, K. M. Chandy, R. R. Muntz, and F. G. Palacios. Open, Closed and Mixed Networks of Queues with Different Classes of Customers. *J. ACM.*, 22(2):248–260, Apr. 1975.

[3] O. J. Boxma and H. Daduna. Sojourn times in queueing networks. In H. Takagi, editor, *Stochastic Analysis of Computer and Communication Systems*, pages 401–450. Elsevier Science Publishers, B. V. (North-Holland), Amsterdam, The Netherlands, 1990.

[4] G. Ciardo, A. Blakemore, P. F. Chimento, J. K. Muppala, and K. S. Trivedi. Automated generation and analysis of Markov reward models using Stochastic Reward Nets. In C. Meyer and R. J. Plemmons, editors, *Linear Algebra, Markov Chains, and Queueing Models, IMA Volumes in Mathematics and its Applications*, volume 48. Springer-Verlag, Heidelberg, Germany, 1992.

[5] G. Ciardo, J. Muppala, and K. Trivedi. SPNP: Stochastic Petri net package. In *Proc. Int. Workshop on Petri Nets and Performance Models*, pages 142–150, Los Alamitos, CA, Dec. 1989. IEEE Computer Society Press.

[6] A. E. Conway and D. E. O'Brien. Estimating response time distributions in queueing networks. In *Performance of Distributed Systems and Integrated Communication Networks*, pages 225–244. Elsevier Science Publishers, B. V. (North-Holland), Amsterdam, The Netherlands, 1992.

[7] S. S. Lavenberg. *Computer Performance Modeling Handbook.* Academic Press, New York, 1983.

[8] S. S. Lavenberg and M. Reiser. Stationary state probabilities of arrival instants for closed queueing networks with multiple types of customers. *J. Appl. Prob.*, pages 1048–1061, Dec. 1980.

[9] E. D. Lazowska, J. Zahorjan, G. S. Graham, and K. C. Sevcik. *Quantitative System Performance.* Prentice-Hall, Englewood Cliffs, NJ, USA, 1984.

[10] B. Melamed and M. Yadin. Numerical computation of sojourn-time distributions in queueing networks. *J. ACM.*, 31(4):839–854, Oct. 1984.

[11] B. Melamed and M. Yadin. Randomization procedures in the computation of cumulative-time distributions over discrete state Markov processes. *Oper. Res.*, 32(4):926–944, July-Aug. 1984.

[12] M. K. Molloy. Performance analysis using stochastic Petri nets. *IEEE Trans. Comput.*, C-31(9):913–917, Sept. 1982.

[13] J. L. Peterson. *Petri Net Theory and the Modeling of Systems.* Prentice-Hall, Englewood Cliffs, NJ, USA, 1981.

[14] K. E. E. Raatikainen. Approximating response time distributions. In *Proc. 1989 ACM SIGMETRICS Int. Conf. on Measurement and Modeling of Computer Systems*, pages 190–199, Berkeley, CA, 1989.

[15] S. Salza and S. S. Lavenberg. Approximating response time distributions in closed queueing network models of computer performance. In F. J. Klystra, editor, *Performance'81*, pages 659–666, B.V. (North-Holland), Amsterdam, 1981.

[16] K. C. Sevcik and I. Mitrani. The distribution of queueing network states at input and output instants. *J. ACM.*, 28(2):358–371, Apr. 1981.

[17] K. S. Trivedi. *Probability & Statistics with Reliability, Queueing, and Computer Science Applications*. Prentice-Hall, Englewood Cliffs, NJ, USA, 1982.