DISTRIBUTION ANALYSIS OF PRODUCT FORM QUEUEING NETWORKS

Edmundo de Souza e Silva

April 1987
CSD-870023

Distribution Analysis of Product Form Queueing Networks [1]

E. de Souza e Silva [2]
UCLA Computer Science Department

April 1987

[1]This research was supported by a grant from NSF INT-8514377 and CNPq-Brasil.

[2]This work was done while the author was on leave from PUC/RJ Electrical Engineering Department, visiting UCLA Computer Science Department.

Abstract

We develop a new computational algorithm which computes joint queue length distributions for product form queueing networks with single server fixed rate, infinite server and queue dependent server service centers. Joint distributions are essential to calculate availability measures using queueing network modeling. This new algorithm is obtained using the physical interpretation of the main equation in the recently proposed MVAC algorithm. However, besides obtaining joint distributions, the new recursion has much simpler characteristics than the MVAC recursion.

1 Introduction

In the past few years, several algorithms have been developed to solve product form queueing network models with multiple chains [LAVE83]. Recently, Conway and Georganas [CONW86a] developed a new computational algorithm called RECAL (Recursion by Chain Algorithm) which has some advantages over previous published algorithms (e.g., Convolution [REIS76]). Like Convolution, RECAL requires the computation of the normalisation constant and suffers from the same kind of underflow/overflow problems which may occur in the implementation of the Convolution algorithm. Subsequently, Conway, de Sousa e Silva and Lavenberg [CONW86b] developed a new recursion which is similar in form to that used in RECAL but does not require the computation of the normalisation constant. This new recursion was called MVAC (Mean Value Analysis by Chain Algorithm). The name was chosen since, like MVA (Mean Value Analysis, [REIS80]), for networks consisting only of single server fixed rate and infinite server service centers the recursion involves only the mean performance measures of interest (mean queue lengths, mean waiting times and mean throughput). Furthermore, as with MVA, the equations obtained have a physical interpretation. The MVAC algorithm can be extended for solving networks with queue length dependent service centers. This extension requires the computation of the marginal queue length distributions similarly to the MVA extension for these networks but, unlike MVA, the basic algorithm is numerically stable. (Reiser [REIS81] proposed an extension to the MVA algorithm which is numerically stable, but this extension is very costly.)

Both RECAL and MVAC algorithm have significantly less computational requirements than the Convolution and MVA algorithms when the number of service centers in the model being solved is small and the number of chains is large. One important application which may require the solution of networks of this kind is in availability modeling [GOYA85]. However, in availability modeling, not only mean measures or marginal queue length distributions are required to be computed. In general, joint queue length distributions are needed as well, which precludes the use of the above algorithms for this important application area.

Another (though less important) drawback of RECAL and MVAC is that both algorithms involve the use of modified networks which contain special chains which visit only one center in the network. The use of these chains obscures the description of the algorithm. Unlike Convolution or MVA algorithms, intermediate steps in their recursion produce results for the modified networks and thus these results are meaningless.

In this paper we develop a new recursion which computes the joint queue length distributions for product form queueing networks with single server fixed rate, infinite server and queue dependent server service centers. This new recursion is obtained using the

physical interpretation of the main equation in the MVAC algorithm. However, this new recursion has much simpler characteristics than the MVAC recursion. The recursion will be referred to as DAC (Distribution Analysis by Chain) since the joint queue length distributions are obtained. Unlike MVAC, at step k of the recursion, results are computed for a network which contains k single customer chains. No modifications are done in the original network. We will also show that this recursion is cheaper than the MVAC recursion and has the advantage of obtaining joint queue length distributions for the network, which are important in applications such as availability modeling.

The paper is organised as follows. Section 2 outlines the recursion and presents the main equations used. Section 3 presents the details of the DAC algorithm. An example is also given. In section 4 we consider the computational requirements for the algorithm. Section 5 presents extensions to the basic algorithm which may reduce computational costs substantially, when not all joint queue length distributions are required to be calculated. Finally, in section 6 we present our conclusions.

2 The New Recursion.

We consider closed queueing network models with multiple chains and queue dependent service centers. The following notation will be used throughout the paper and is summarised below:

J	=	number of service centers.
K	=	total number of chains in the network.
N_k	=	number of customers in chain k.
$\vec{N}$	=	population vector $= (N_1, \ldots, N_K)$.
$\vec{N}(k)$	=	population vector $= (N_1^k, \ldots, N_K^k)$ for a network where k customers only are present (from the total number of customers) and, in this case, there are N_1^k customers of chain 1, N_2^k customers of chain 2, etc.
$j(k)$	=	a specified service center visited by chain k.
θ_{jk}	=	visit ratio of a chain k customer to center j, scaled so that $\theta_{j(k)k} = 1$.
T_{jk}	=	mean service time of a chain k customer at center j.
a_{jk}	=	$\theta_{jk}.T_{jk}$ = relative utilisation of a chain k customer at center j.
λ_{jk}	=	$\theta_{jk}.\lambda_k$ = mean throughput of chain k customer at center j, where $\lambda_k = \lambda_{j(k)k}$.
L_{jk}	=	mean number of customers of chain k at center j.
W_{jk}	=	mean waiting time (queueing time + service time) of a chain k customer at center j.
$\mu_j(n)$	=	service rate of service center j when there are n customers present, $\mu_j(1) = 1$.
n_{jk}	=	number of chain k customers at center j.
n_j	=	$\sum_{k=1}^{K} n_{jk}$ = number of customers at service center j.
$\vec{n}_j$	=	$(n_{j1}, \ldots, n_{jK})$ = state of service center j.
$\underline{\vec{n}}$	=	$(\vec{n}_1, \ldots, \vec{n}_J)$ = state of the network.
$\vec{n}$	=	$(n_1, \ldots, n_J)$ = aggregate state of the network.
$\underline{S}(\vec{N})$	=	$\{\underline{\vec{n}} : \sum_{j=1}^{J} \vec{n}_j = \vec{N},\ n_{jk}$ is a nonnegative integer for all j and $k\ \}$ = state space of the network.
$P(\underline{\vec{n}}, \vec{N})$	=	steady state probability that the network is in state $\underline{\vec{n}} \in \underline{S}(\vec{N})$.
$S(\vec{N})$	=	$\{\vec{n} :\ \sum_{j=1}^{J} n_j = \sum_{k=1}^{K} N_k,\ n_j$ is a nonnegative integer for all $j\ \}$ = aggregate state space of the network.
$P(\vec{n}, \vec{N})$	=	steady state probability that the network is in state $\vec{n} \in S(\vec{N})$.
$P_j(n)$	=	steady state probability that there are n customers at service center j.
$\vec{1}_j$	=	J-dimensional vector whose j-th element is one and whose other elements are zero.

In this section we compute the steady state probabilities $P(\vec{n}, \vec{N})$ and the mean performance measures λ_k for all chains k in the network. Similar to the MVAC algorithm, we consider only networks such that each chain contains only one customer. We emphasize that any network can be converted to a related single customer per chain network and the performance measures of the original network can be easily derived from the performance measures of the related network.

Recently Conway, de Sousa e Silva and Lavenberg [CONW86b] developed a new recursive algorithm called MVAC. For networks which include queue dependent service centers, the algorithm computes $P_j(n)$, the equilibrium probability that there are n customers in service center j, $\forall j$, $\forall n$, and the average measures L_{jk} and λ_k, $\forall j$, $\forall k$. In that algorithm, the network is modified such that special chains called single customer self loop (SCSL) are added to the network. This modification to the original network obscures the description of the algorithm. However, one of the most interesting aspects of the MVAC algorithm is the physical interpretation of the main equations used.

In this paper we use this physical interpretation to develop a new algorithm with simpler characteristics. The new recursion presented in this paper is done in a chain by chain basis, similar to MVAC. However, no SCSL chains are used. We will show that if we are given the equilibrium state probabilities of a network with k chains $P^k(\vec{n}, \vec{N}(k))$ (or $P^k(\vec{n}, \vec{N}(k))$ using the aggregate state space), the equilibrium state probabilities of a new network which contains one more chain, i.e., $P^{k+1}(\vec{n}, \vec{N}(k+1))$ (or $P^{k+1}(\vec{n}, \vec{N}(k+1))$) can be easily calculated. We recall that we are considering chains with a single customer only.

The outline of the algorithm is presented below:

(0) Initial step, $k = 1$.
Take the original network and remove all but one of its chain. Find the performance measures of this single chain network, including marginal queue length distributions. This measures can be easily calculated by using known equations, say MVA equations.

(1) Add one more chain to the resulting network. This network has now $k+1$ chains. The current performance measures can be calculated from the performance measures of the network with k chains.

(2) $k \leftarrow k+1$.
If $k = K$, i.e., if all chains of the original network were already added, then stop. Otherwise go to step (1).

We describe the equations used in step 1 of the algorithm. Assume that the network currently contains $k-1$ chains, and we are given the equilibrium state probability that the network is in state $\vec{n} = (n_1, n_2, \ldots, n_J)$, i.e., $P^{k-1}(\vec{n}, \vec{N}(k-1))$. For notational convenience we drop the notation $\vec{N}(k-1)$, i.e., $P^{k-1}(\vec{n}) = P^{k-1}(\vec{n}, \vec{N}(k-1))$. From $P^{k-1}(\vec{n})$, $P_j^{k-1}(n)$ can be easily obtained:

$$P_j^{k-1}(n) = \sum_{\substack{i=1 \\ i \neq j}}^{J} \sum_{n_i=0}^{k-1} P^{k-1}(n_1, n_2, \ldots, n_J) \tag{1}$$

Using MVA equations for networks with load dependent service centers [LAVE83] we can calculate λ_k^k and L_{jk}^k, i.e., the throughput of the chain being added to the network at current step k (chain k) and the average queue length of the single customer of this chain at service center j. This is true since these performance measures depend on the performance measures of a network with the customer of chain k removed. Therefore, we have:

$$\lambda_k^k = \frac{1}{\sum_{j=1}^{J} a_{jk} \sum_{n=1}^{k} \frac{n}{\mu_j(n)} P_j^{k-1}(n-1)} \tag{2}$$

and

$$L_{jk}^k = \lambda_k^k a_{jk} \sum_{n=1}^{k} \frac{n}{\mu_j(n)} P_j^{k-1}(n-1) \tag{3}$$

Note that since all chains are assumed to have only one customer, L_{jk}^k is the probability that the customer of chain k is at service center j. Using this observation, let $P^k(\vec{n} \mid n_{jk} = 1)$ be the steady state conditional probability that the network (with k single customer chains) is in state $\vec{n} \in S(\vec{N}(k))$ given that the single chain k customer is in service center j. Unconditioning we obtain $P^k(\vec{n})$.

$$P^k(\vec{n}) = \sum_{j=1}^{J} L_{jk}^k P^k(\vec{n} \mid n_{jk} = 1) \tag{4}$$

We now invoke a lemma used to establish the main results for the MVAC algorithm. The lemma states that the conditional steady state distribution given that the single

chain k customer is in a particular service center (say service center j) is equal to the unconditional steady state distribution of a network in which chain k is replaced by a single chain which visits only service center j. Recall that the relative utilisation of this chain can always be scaled to one. Formally, we have:

$$P^k(\vec{n} \mid n_{jk} = 1) = P^{k-1}(\vec{n} - \vec{1}_j, 1_j) \tag{5}$$

where $P^{k-1}(\vec{n} - \vec{1}_j, 1_j)$ is, by definition, the steady state probability distribution of a network with $k-1$ chains plus a single customer chain which visits service center j only. Using (5), (4) can be rewritten:

$$P^k(\vec{n}) = \sum_{j=1}^{J} L_{jk}^k P^{k-1}(\vec{n} - \vec{1}_j, 1_j) \tag{6}$$

Observe that if we can calculate $P^{k-1}(\vec{n} - \vec{1}_j, 1_j)$ from the performance measures of a network with $k-1$ chains, the algorithm will be complete. This is the subject of the following theorem.

Theorem 1 *If service center j is infinite server (IS)*

$$P^k(\vec{n}, 1_j) = P^k(\vec{n}) \tag{7}$$

Otherwise:

$$P^k(\vec{n}, 1_j) = \frac{n_j + 1}{\mu_j(n_j + 1)} \tau_j^{k+1} P^k(\vec{n}) \tag{8}$$

where

$$\tau_j^k = \begin{cases} \dfrac{1}{1 + L_j^{k-1}} & \text{if center } j \text{ is SSFR} \\ \dfrac{1}{\sum_{n=1}^{k} \dfrac{n}{\mu_j(n)} P_j^{k-1}(n-1)} & \text{if center } j \text{ is queue length dependent} \end{cases} \tag{9}$$

Proof:

Equation (7) is trivially obtained since, if j is IS service center, the chain visiting only j does not affect the performance measures of the network.

To prove equation (8), recall that $\vec{n} = (\vec{n}_1, \vec{n}_2, \ldots, \vec{n}_J)$. For product form networks, the equilibrium state probabilities are given by [LAVE83] :

$$P^k(\vec{n}, 1_j) = \frac{1}{G^k(1_j)} \prod_{i=1}^{J} \frac{n_i!}{\prod_{v=1}^{n_i} \mu_i(v)} \prod_{i=1}^{k+1} \frac{a_{ii}^{n_{ii}}}{n_{ii}!} \tag{10}$$

where $G^k(1_j)$ is the normalisation constant of a network with k single customer chains and a single chain visiting center j only. In (10) this chain has index $(k+1)$.

Extracting center j from the product above:

$$P^k(\vec{n}, 1_j) = \frac{1}{G^k(1_j)} \left[\prod_{\substack{i=1 \\ i \neq j}}^{J} \frac{n_i!}{\prod_{v=1}^{n_i} \mu_i(v)} \prod_{i=1}^{k} \frac{a_{ii}^{n_{ii}}}{n_{ii}!} \cdot \frac{(n_j+1)!}{\prod_{v=1}^{n_j+1} \mu_j(v)} \prod_{i=1}^{k} \frac{a_{ji}^{n_{ji}}}{n_{ji}!} a_{jk+1} \right]$$

where the second product above was obtained by: noting that center j has n_j+1 customer due to the chain that visits only this center (chain $k+1$); $a_{ik+1} = 0 \; \forall i \neq j$. Since $a_{jk+1} = 1$,

$$P^k(\vec{n}, 1_j) = \frac{1}{G^k(1_j)} \left[\frac{n_j+1}{\mu_j(n_j+1)} \prod_{i=1}^{J} \frac{n_i!}{\prod_{v=1}^{n_i} \mu_i(v)} \cdot \prod_{i=1}^{k} \frac{a_{ii}^{n_{ii}}}{n_{ii}!} \right]$$

Multiplying and dividing by $G^k(0)$, i.e., the normalisation constant of the network with k chains with the chain visiting only center j removed:

$$P^k(\vec{n}, 1_j) = \frac{G^k(0)}{G^k(1_j)} \frac{(n_j+1)}{\mu_j(n_j+1)} \left[\frac{1}{G^k(0)} \prod_{i=1}^{J} \frac{n_i!}{\prod_{v=1}^{n_i} \mu_i(v)} \cdot \prod_{i=1}^{k} \frac{a_{ii}^{n_{ii}}}{n_{ii}!} \right]$$

The term in brackets in the above equation is, $P^k(\vec{n})$. Therefore:

$$P^k(\vec{n}, 1_j) = \frac{G^k(0)}{G^k(1_j)} \frac{(n_j+1)}{\mu_j(n_j+1)} P^k(\vec{n}) \tag{11}$$

But $G^k(0)/G^k(1_j)$ is the throughput of the chain which visits only center j, τ_j^{k+1}

$$\frac{G^k(0)}{G^k(1_j)} = \tau_j^{k+1} = \begin{cases} \dfrac{1}{1+L_j^k} & \text{if center } j \text{ is SSFR} \\ \dfrac{1}{\sum_{n=1}^{k+1} \dfrac{n}{\mu_j(n)} P_j^k(n-1)} & \text{if center } j \text{ is queue length dependent} \end{cases} \quad (12)$$

where the last equality uses equation (2) for the chain which visits center j only and so $a_{lk+1} = 0$ for $l \neq j$ and $a_{jk+1} = 1$.

Substituting (12) into (11) we get:

$$P^k(\vec{n}, 1_j) = \left[\frac{n_j + 1}{\mu_j(n_j+1)} \tau_j^{k+1}\right] P^k(\vec{n}) \quad (13)$$

Finally, we note that

$$P^k(\vec{n}, 1_j) = \sum_{\substack{|\vec{n}_1| = n_1 \\ \vdots \\ |\vec{n}_k| = n_k}} P^k(\vec{n}, 1_j) =$$

where $|\vec{n}_l| = \sum_{i=1}^{k} n_{li}$

Since the term in brackets in (13) is constant with the sum, (8) is proven. □

3 The DAC Algorithm.

Our main goal is to compute the equilibrium state probability that the network is in state $\vec{n} \in S(\vec{N})$. As we will see in the example presented later this measurement is important for availability modeling. The first step of the algorithm requires the computation of the equilibrium state probabilities of a network with a single chain with one customer. This can easily be obtained by using MVA equations (2) and (3), which simplifies to:

$$L_{j1}^1 = \frac{a_{j1}}{\sum_{j=1}^{J} a_{j1}} \quad \forall j \tag{14}$$

Since there is only one customer at chain 1, $P^1(\vec{1}_j) = L_{j1}^1 \; \forall j$.

The remaining steps require equations (6) and (8) which can be combined into a single equation (since equation (7) is a particular case of (8) we use only (8) in the remaining of the paper):

$$P^k(\vec{n}) = \sum_{j=1}^{J} L_{jk}^k \frac{n_j}{\mu_j(n_j)} \tau_j^k P^{k-1}(\vec{n} - \vec{1}_j) \tag{15}$$

Below we present the details of the DAC algorithm.

Step 1:

Set $k = 1$.
Use, equation (14) to calculate L_{j1}^1 (and so $P^1(\vec{1}_j)$) for all centers j in the network.

Step 2:

For $k = 2$, $k \leq K$:

(a) Use equation (2) to calculate λ_k^k.

(b) Use equation (3) to calculate $L_{jk}^k \; \forall j \; 1 \leq j \leq J$.

(c) Use equation (15) to calculate $P^k(\vec{n}) \; \vec{n} \in S(\vec{N}(k))$.

(d) From $P^k(\vec{n})$ and equation (1), $P_j^k(n)$ can be easily obtained.

The DAC algorithm as stated above yields not only the equilibrium state probability that the network is in state $\vec{n} \in S(\vec{N})$, but also the average performance measures for chain K (i.e., λ_k^k and $L_{jk}^k \; \forall j$). It does not yields average performance measures of other chains in the network. As we will show below, other performance measures can be calculated, if needed. Furthermore, one can easily verify that if a center in the network (say center i) is visited only by customers of a single chain, say chain t, then the throughput for this chain can be calculated by:

$$\lambda_t^k = \frac{\sum_{n=1}^{N_i^k} \mu_i(n) P_i^k(n)}{a_{it}} \tag{16}$$

Similarly to the MVAC algorithm, the average performance measures of the individual chains can be calculated if we reexecute the basic step. Assume that, from the K chains only D are distinct. We order the K chains such that the last D ones added to the network are distinct. After executing steps 1 and 2 for $1 \leq k \leq K$, we exchange labels of the last chain with any other chain belonging to the last D chains added (say chain $k-l$). Step 2 is executed for $k-l \leq k \leq K-1$ and steps 2a and 2b are executed for $k = K$. However, we emphasize that without the relabeling mechanism the whole set of equilibrium states probabilities are calculated, unlike MVAC. This may be sufficient for a particular application such as availability modeling.

Example:

Consider an availability model of a computer system with two types of components, say memory and CPU. There are three memory modules, one of them is a spare module and two are active modules. There is only one CPU. We assume that the spare memory can not fail if it is not being used. Furthermore, when a memory unit fails the spare unit is immediately switched to full operation. We assume independence of failure rates. Once a component fails it goes to a repair facility. There is only one repair man, he chooses a component to repair at random from the repair queue and a new failure preempts the repair of an old one. Figure 1 shows the queueing network model for this system.

In that Figure chain 2 represents the behavior of the memory modules and chain 1 represents the behavior of the CPU. Center 2 models the failure behavior of the memory modules. Note that the queue length dependent center 2 is equivalent to a two server service center which models the fact that only 2 memory modules can fail if 3 modules are in good condition. Center 1 models the failure behavior of the CPU and center 3 models the repair facility. The appendix presents the parameter values for this network and the

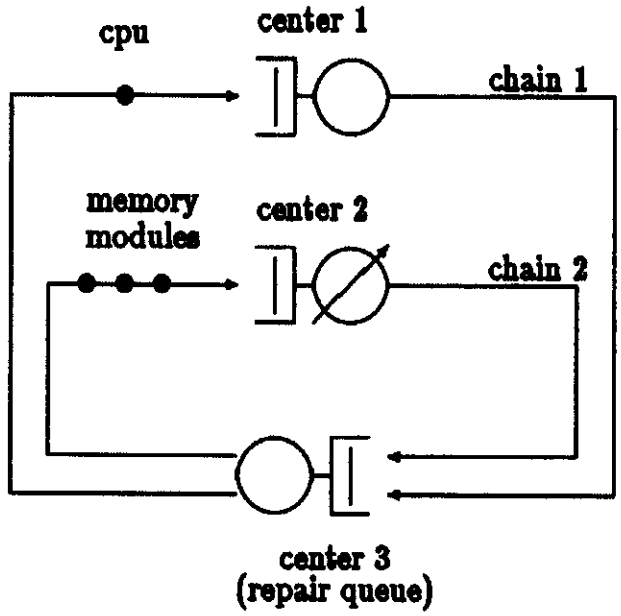

Figure 1: An availability model of a computer system.

calculations at each step of the algorithm.

Now we assume that the spare memory is being powered even if it is not in use and thus can also fail (hot stand by). This behavior can be modeled by the same network of Figure 1, if we alter the service rate of center 2 to include the failure rate of the spare unit. In other words, $\mu_2(3)/a_{22} = 2\lambda_{mem} + \lambda_{spare}$ where λ_{mem} and λ_{spare} are the failure rates of the working units and the spare units, respectively. The appendix also presents the final results when a hot stand by unit is used. Note that only step $k = 4$ has to be recomputed in this case.

From the example we verify the importance of the calculated measures. For instance, the probability that all four modules are working simultaneously is simply $P^4(1,3,0)$. If we assume that the system is operational when at least one CPU and one memory module is operational, then the availability of the system is $AV = P^4(1,3,0)+P^4(1,2,1)+P^4(1,1,2)$.

Since the iteration requires the calculation of the performance measures with one less "customer", using the example we can verify, for instance, the degradation of availability when no spare memory is used. This can be done by comparing the results of the original network with a new network with one less memory module (which correspond to the results of step $k = 3$ in the appendix). This contrasts with MVAC for queue dependent centers or RECAL, in which only the last step gives meaningful results. Previous steps produce results for networks with the so called self looping chains.

4 Computational Requirements.

In this section we derive the computational cost of the DAC algorithm. We first determine the cost when the goal is to compute the full set of joint equilibrium probability distributions. Next we derive the cost when average measures for all distinct chains needed to be computed. Comparisons with MVAC will also be made.

We point out that significant savings can be obtained if only a subset of these probabilities needs to be calculated, as we will demonstrate in section 5. In the expressions for computational costs, we count only the operations of multiplication and division, each having a unitary cost. The cost of additions will be ignored. We also assume that all centers in the network are queue dependent. Savings can be obtained if we have SSFR and/or IS centers.

Step 1 of the algorithm requires J divisions. For a subsequent step k we need to compute:

(a) λ_k^k and L_{jk}^k $\forall j$, using equation (2) and (3) respectively. Note that these equations have the common factor.

$$\frac{1}{\tau_j^k} = \sum_{n=1}^{k} \frac{n}{\mu_j(n)} P_j^{k-1}(n-1)$$

which needs to be computed only once for all service centers j. The above expressions require $2k$ operations for a single value of j. Therefore, for calculating (2) and (3) we need a total of $2J(k+1)+1$ operations.

(b) $P^k(\vec{n})$ for $\vec{n} \in S(\vec{N})$. From equation (15) it is easy to see that we require $4J$ operations at most, to compute $P^k(\vec{n})$ for a given value of $\vec{n}$. Since there are $\binom{J-1+k}{J-1}$ values of $\vec{n}$, the total cost is:

$$4J\binom{J-1+k}{J-1}$$

Finally, the total cost σ_{DAC}^1 of the algorithm is:

$$\begin{aligned}\sigma_{DAC}^1 &= J + \sum_{k=2}^{K}\left[2J(k+1)+1+4J\binom{J-1+k}{J-1}\right] \\ &= 4J\binom{J+K}{J} + JK(K+3) + K - 4J^2 - 7J - 1\end{aligned}$$

The first term is the dominant factor, therefore,

$$\sigma^1_{DAC} \approx 4J\binom{J+K}{J}$$

Note that this computational cost assumes that all centers are queue dependent. The computation may be reduced if we have SSFR and/or IS centers.

The basic step of the MVAC algorithm has cost σ^{BAS}_{MVAC}:

$$\sigma^{BAS}_{MVAC} = J(J+2)\frac{K}{J+1}\binom{J+K}{J} \geq JK\binom{J+K}{J}$$

which is comparable to the DAC algorithm. However, the basic step of the MVAC algorithm only computes the average measures for chain K. The DAC algorithm besides computing average measures for chain K computes all equilibrium joint probability distributions.

Next, assume that we use the DAC algorithm to compute the average measures of all chains plus the joint probability distributions. Assume also that there are D distinct chains in the network. The subsequent steps that need to be performed have cost σ^2_{DAC}:

$$\sigma^2_{DAC} \cong \sum_{i=1}^{D-1}\sum_{k=K-D+i}^{K-1} 4J\binom{J-1+k}{J-1}$$

where the approximation comes from the fact that we are using only the combinatorial term for the cost. Therefore:

$$\begin{aligned}\sigma^2_{DAC} &\cong \sum_{i=1}^{D-1} 4J\left[\binom{J+K-1}{J} - \binom{J+K-D+i-1}{J}\right] \\ &= 4J(D-1)\binom{J+K-1}{J} + 4J\binom{J+K-1}{J+1} - 4J\binom{J+K-D}{J+1}\end{aligned}$$

and so, the total cost will be $\sigma_{DAC} = \sigma^1_{DAC} + \sigma^2_{DAC}$.

Suppose that we have D distinct chains with N customers each, so that $K = ND$. If we fix J and N it follows that as $D \to \infty$ $\sigma_{DAC} = O(D^{J+1})$ which is in the same order of the MVAC algorithm for SSFR and IS only with minimum cost, and is cheaper than

the version for networks with queue dependent centers. We emphasize again that, unlike MVAC, the set of joint probability distributions are calculated. If we are only interested in the set of joint probability distributions, then as $D \rightarrow \infty$ the cost is $O(D^J)$.

For availability modeling the number of centers may grow if several components of the same type have spare units. If each set of components of the same type has at least one spare unit, then the number of centers in the network will be identical to the number of distinct chains. (From the example in section 3 it is easy to see why this is true.) From the expressions obtained above, it seems that the computational costs would be prohibitive. However, since for this kind of models the chains do not visit all centers, substantial savings can be obtained. Assume that we have D distinct component types and one of them (say component type t) has N_t units. Some of the N_t units will be spare ones. Also, for simplicity, assume that $N_t = N_v = N$, $\forall t, v$, and that we have only one repair center. Therefore, $J = D + 1$. At step k such that $k/N = M + \nu$ we have already added M distinct chains and ν units of distinct chain $M + 1$. Therefore, for a step $2 \leq k \leq K$ we need $8(N+1)^M(\nu+1)$ operations. (Note that $L^k_{j_k}$ is different than zero for two centers only since each chain in this model visits two centers.) Finally, the total cost is:

$$\sigma^s_{DAC} = 8\left(\frac{N+2}{2}\right)\sum_{v=1}^{D}(N+1)^v$$

5 Extensions to the Algorithm.

In this section we make a few observations concerning the application of the algorithm when the goal is to calculate joint distributions considering the detailed state space $\underline{S}(\vec{N})$, i.e., when we discriminate customers from different chains at different service centers. Next we extend the DAC algorithm to be used when not all the joint distributions need to be calculated and show the potential savings.

Consider the situation where $P(\vec{n})$ are required to be determined. We invoke again the lemma used to establish the main results for the MVAC algorithm. Equation (6) can be rewritten to include the detailed state space:

$$P^k(\vec{n}) = \sum_{j=1}^{J} L_{jk}^k P^{k-1}(\vec{n} - \vec{1}_{jk}, 1_j) \tag{17}$$

where $\vec{1}_{jk}$ is the JK-dimensional vector in which all elements are zero except one element at position jk which is one. Using equation (13) obtained in the proof of Theorem 1, in (17) we obtain:

$$P^k(\vec{n}) = \sum_{j=1}^{J} L_{jk}^k \frac{n_j}{\mu_j(n_j)} \tau_j^k P^{k-1}(\vec{n} - \vec{1}_{jk}) \tag{18}$$

The other equations used in the algorithm (equations (2) and (3)) remain the same. The two steps in the DAC algorithm remain basically the same. Since the joint probabilities of the detailed state space are obtained when the two steps are executed, there is no need to reexecute step 2 again to obtain average measures for individual chains.

The computational cost increases since the state space that needs to be searched increases. In general, at a step k, the computation cost depends on the number of customers of each distinct chain up to that step, i.e.:

$$\sigma(k) \propto \prod_{i=1}^{D} \binom{J-1+n_i^k}{J-1}$$

where n_i^k is the total number of customers of "distinct" chain i at step k. Note that the product is the number of states in $\underline{S}(\vec{N}(k))$.

Depending on the particular problem, the cost could be much larger than or even identical to the cost of the original DAC algorithm. For instance, in the example of section

3, it is easy to verify that the detailed state space $\underline{S}(\vec{N})$ has the same number of states than the aggregate $S(\vec{N})$ at each step k of the algorithm.

Example:

Consider a distributed architecture system as illustrated in Figure 2.

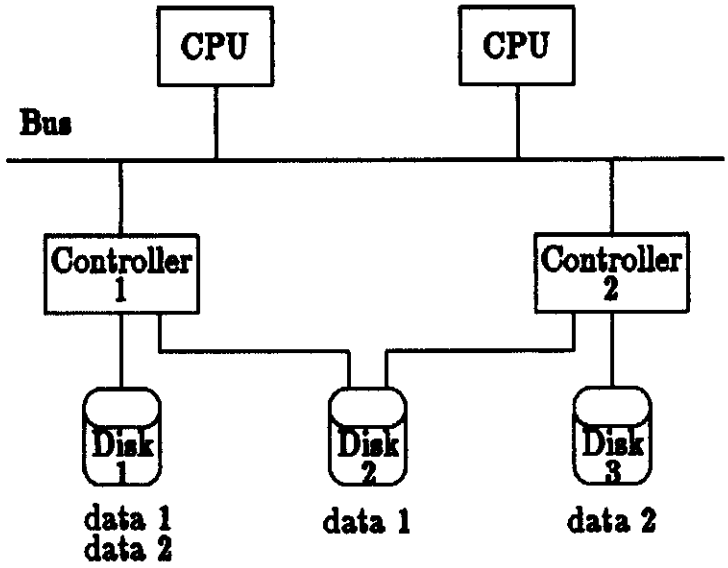

Figure 2: A distributed architecture system.

In this system, one of the CPU's is spare and critical data (d_1 and d_2) is replicated. The queueing model for this system is shown in Figure 3, where the chains visiting the IS center represent the bus, controller 1, controller 2 and disks 1, 2 and 3.

We assume that all data is available if there is a path between the operational CPU and both data items d_1 and d_2. Therefore, to determine the probability that the data is available we need to calculate $\{P^s(n_1, \vec{n}_2, n_3)\}$, which requires equation (18).

Next we present results that may be used to reduce the cost of the DAC algorithm when the goal is to calculate a subset of the joint queue length distributions. Assume that in step k of the algorithm we need to calculate $P^k(\vec{n}_{k_1}, \ldots, \vec{n}_{k_r})$ where the set $\{k_i\}$ represent indexes of the service centers in the network. It is easy to show that, similarly to equation (17), we have:

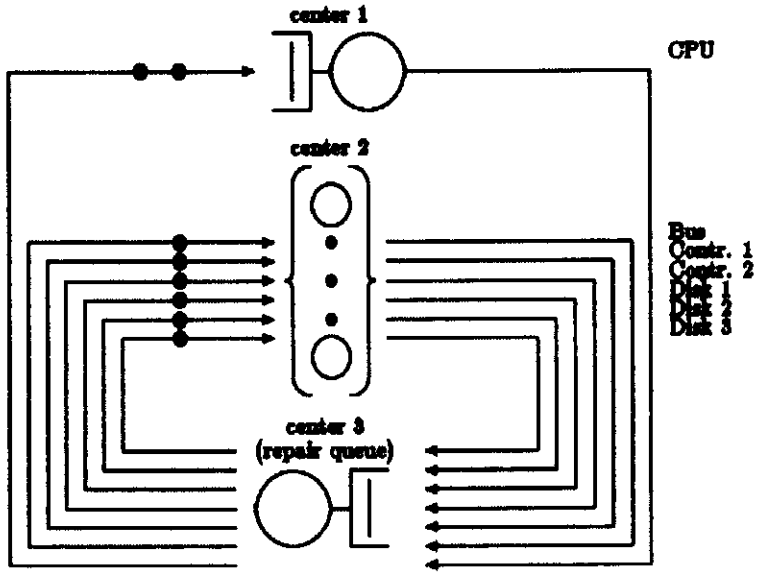

Figure 3: The queueing model of the system of Figure 2.

$$P^k(\vec{n}_{k_1},\ldots,\vec{n}_{k_v}) = \sum_{\substack{j=1 \\ j\in\{k_i\}}}^{J} L^k_{jk} P^{k-1}(\vec{n}_{k_1},\ldots,\vec{n}_j-\vec{1}_k,\ldots,\vec{n}_{k_v},1_j) + \sum_{\substack{j=1 \\ j\notin\{k_i\}}}^{J} L^k_{jk} P^{k-1}(\vec{n}_{k_1},\ldots,\vec{n}_{k_v},1_j) \quad (19)$$

where $\vec{1}_k$ is the K-dimensional vector in which all elements are zero except one element at position k which is one. It remains to be computed $P^{k-1}(\vec{n}_{k_1},\ldots,\vec{n}_{k_v},1_j)\ \forall k$. For that we need the following corollary.

Corollary 1 *If service center j is infinite server*

$$P^k(\vec{n}_{k_1},\ldots,\vec{n}_{k_v},1_j) = P^k(\vec{n}_{k_1},\ldots,\vec{n}_{k_v}) \quad (20)$$

Otherwise:

$$P^k(\vec{n}_{k_1},\ldots,\vec{n}_{k_v},1_j) =$$

$$= \frac{n_j+1}{\mu_j(n_j+1)}\tau_j^{k+1}P^k(\vec{n}_{k_1},\ldots,\vec{n}_{k_v}) \qquad j\in\{k_i\}1\le i\le v \tag{21}$$

$$= \sum_{n_j=0}^{k}\frac{n_j+1}{\mu_j(n_j+1)}\tau_j^{k+1}P^k(\vec{n}_{k_1},\ldots,\vec{n}_{k_v},n_j) \quad j\notin\{k_i\}1\le i\le v \tag{22}$$

Proof:

The proof follows directly from Theorem 1. First we note that:

$$P^k(\vec{n}_{k_1},\ldots,\vec{n}_{k_v},1_j) = \sum_{\substack{\vec{n}_l=\vec{0}\\ \forall l\notin\{k_i\}}}^{\vec{N}(k)} P^k(\vec{n},1_j)$$

where $\vec{0}$ is the K-dimensional vector in which all elements are zero, $\sum_{\substack{\vec{n}_l=\vec{0}\\ l\notin\{k_i\}}}^{\vec{N}(k)} = \sum_{n_{u_1}}\sum_{n_{u_2}}\ldots\sum_{n_{u_m}}$ and the set $\{u_r\}$ has no intersection with the set $\{k_i\}$, $1\le r\le m$ $1\le i\le v$. Therefore, the sum is over all possible combinations of customers of different chains at center l, $\forall l\notin\{k_i\}$. Using equation (13):

$$P^k(\vec{n}_{k_1},\ldots,\vec{n}_{k_v},1_j) = \sum_{\substack{\vec{n}_l=\vec{0}\\ \forall l\notin\{k_i\}}}^{\vec{N}(k)}\left[\frac{n_j+1}{\mu_j(n_j+1)}\tau_j^{k+1}\right]P^k(\vec{n}) \tag{23}$$

We have to consider two cases:

(a) $j\in\{k_i\}$ $1\le i\le v$.

In this case the term is brackets in the equation (23) can be shifted outside the two sums:

$$P^k(\vec{n}_{k_1},\ldots,\vec{n}_{k_v},1_j) = \frac{n_j+1}{\mu_j(n_j+1)}\tau_j^{k+1}\sum_{\substack{\vec{n}_l\\ \forall l\notin\{k_i\}}} P^k(\vec{n})$$

and, since the sum above is for all states $\vec{n}_l$ such that $l\notin\{k_i\}$, equation (21) is obtained.

(b) $j \notin \{k_i\}\ 1 \le i \le v$.

In this case, the term in brackets in equation (23) can be shifted outside the sum when $l \neq j$ only. Therefore:

$$\begin{aligned} P^k(\vec{n}_{k_1}, \ldots, \vec{n}_{k_v}, 1_j) &= \\ &= \sum_{\vec{n}_j} \frac{n_j+1}{\mu_j(n_j+1)} \tau_j^{k+1} \sum_{\substack{\vec{n}_l \\ \forall l \in \{k_i\},\ l \neq j}} P^k(\vec{\underline{n}}) = \\ &= \tau_j^{k+1} \sum_{n_j=0}^{k} \frac{n_j+1}{\mu_j(n_j+1)} \sum_{|\vec{n}_j|=n_j} P^k(\vec{n}_{k_1}, \ldots, \vec{n}_{k_v}, \vec{n}_j) \end{aligned}$$

where we broke $\sum_{\vec{n}_j}$ into two sums: $\sum_{n_j}^{k}$ and $\sum_{|\vec{n}_j|=n_j}$. Equation (22) follows from the equation above. □

It is easy to see that equations (20), (21) and (22) can be simplified in the case we are using the aggregate state space where only the number of customers at each center is recorded. In this case:

- For IS centers:

$$P^k(n_{k_1}, \ldots, n_{k_v}, 1_j) = P^k(n_{k_1}, \ldots, n_{k_v}) \tag{24}$$

- Otherwise:

$$\begin{aligned} & P^k(n_{k_1}, \ldots, n_{k_v}, 1_j) = \\ = & \quad \frac{n_j+1}{\mu_j(n_j+1)} \tau_j^{k+1} P^k(n_{k_1}, \ldots, n_{k_v}) \qquad j \in \{k_i\} \qquad (25) \\ = & \quad \tau_j^{k+1} \sum_{n_j=0}^{k} \frac{n_j+1}{\mu_j(n_j+1)} P^k(n_{k_1}, \ldots, n_{k_v}, n_j) \qquad j \notin \{k_i\} \qquad (26) \end{aligned}$$

Equations (21), (22) (or (25), (26)) can be combined with equation (19) to obtain a single equation equivalent to (15). In case of using the aggregate state space we have:

$$P^k(\vec{n}_{k_1}, \ldots, \vec{n}_{k_v}) =$$

$$= \sum_{\substack{j=1 \\ j \notin \{k_i\}}}^{J} L_{jk}^{k} \frac{n_j}{\mu_j(n_j)} \tau_j^k P^{k-1}(n_{k_1}, \ldots, n_j - 1, \ldots, n_{k_v})$$

$$+ \sum_{\substack{j=1 \\ j \notin \{k_i\}}}^{J} L_{jk}^{k} \tau_j^k \sum_{n_j=1}^{k-(n_{k_1}+\ldots+n_{k_v})} \frac{n_j}{\mu_j(n_j)} P^{k-1}(n_{k_1}, \ldots, n_{k_v}, n_j - 1) \qquad (27)$$

Example:

Consider the example of section 3 and assume that the only measure that needs to be calculated is the probability that the CPU is working, i.e., $P_1^4(1)$. Since this example is used to demonstrate the use of equation (27), we assume that the steps for $k = 1, 2, 3$ remain the same. For step 4 we have:

λ_4^4, L_{j4}^4, τ_{j4}^4 $1 \leq j \leq 3$, are calculated as previously shown. Using (27) to calculate $P_1^4(1)$:

$$\begin{aligned} P_1^4(1) = \quad & L_{14}^4 \tau_1^4 P_1^3(0) \\ & + L_{24}^4 \tau_2^4 \sum_{n_2=1}^{3} \frac{n_2}{\mu_2(n_2)} P^3(1, n_2 - 1) \\ & + L_{34}^4 \tau_3^4 \sum_{n_3=1}^{3} n_3 P^3(1, n_3 - 1) \\ = \quad & 0.6694 \end{aligned}$$

The example above illustrates the use of equation (27) but does not illustrate the savings that may be obtained. To illustrate the computational savings, let us assume that the goal is to calculate $P_j(n_j)$ $1 \leq j \leq J$. In this case, the computation remains the same up to step $k = K - J + 1$, i.e., the marginal joint queue length distributions need to be calculated ($P^{K-J+1}(n_1, \ldots, n_J)$). However, from this step up to $k = K$, not all the joint distributions need to be calculated. At step $k = K - J + 1 + l$, $1 \leq l \leq J - 1$ the computations are:

$$\sigma^L(k) \approx 4J \sum_{i=0}^{k} \binom{J-l-1+i}{J-l-1} \binom{J}{J-l}$$

The combinatorial term $\binom{J}{J-l}$ gives all the combinations necessary for determining

$P^k(n_{k_1}, \ldots, n_{k_v})$ where $v = J - l$, and the first combinatorial term assumes that the total number of customers in centers $k_1, \ldots, k_v$ is i, $0 \leq i \leq k$. Therefore:

$$\sigma^L(k) \approx 4J \binom{J-l+k}{J-l} \binom{J}{J-l}$$

Comparing the above cost with the cost obtained in section 4 for step k (the cost is for calculating, at step k, the total joint queue length distributions), we see that we use the previous algorithm up to step $k_r - 1$ such that, for step k_r,

$$\binom{J-1+k_r}{J-1} \geq \binom{J-l+k_r}{J-l} \binom{J}{J-l}$$

Then, for steps $k_r \leq k \leq K$ we use the steps outlined above, i.e., equations (27). For example, if $k = 12$ and $J = 8$, $k_r = 10$. Therefore in the last three steps we calculate the performance measures using equation (27).

Similar savings can be obtained when we consider the detailed state space (i.e., distinguish customers in their chains), and we leave out the details for conciseness.

6 Conclusions.

We developed a new recursion for solving queueing network models with single server fixed rate, infinite servers and queue dependent server service centers, which we call DAC. The asymptotic growth of the computational cost of DAC is the same as MVAC for networks with SSFR and IS centers only and is cheaper than MVAC for (a) networks with queue length dependent centers and (b) if the measures of interest are joint marginal queue length distributions. Furthermore, unlike MVA or MVAC, the algorithm calculates the set of joint marginal queue length distributions, which is essential for availability modeling. The recursion of this new algorithm is much simpler than the recursion of MVAC. At each step, measures are obtained for a network with one more customer. No fictitious chains are introduced. Like MVAC, DAC is numerically stable.

We introduced extensions to the basic algorithm. These extensions include the calculations of joint probabilities when customers are discriminated according to the chains they belong. We also obtain expressions which significantly reduce computational costs when only a subset of the queue length distributions are needed. Examples which illustrate the applicability of the algorithm were presented. Finally, we point out that: (a) average performability measures [MEYE80] can be calculated from the joint distributions of queue length by assigning a reward (which represents a measure of performance) to each state of the availability model, and (b) sensitivity analysis (which is important for availability modeling [GOYA86]) can be done, if we use the results presented in [DeSO84] and the results of this paper.

Acknowledgement

The author wishes to thank S. S. Lavenberg for his helpful comments and suggestions.

Appendix.

The parameter values for the network of Figure 1 are (assuming the spare memory unit cannot fail): $J = 3$, $a_{11} = 5$, $a_{12} = 0$, $a_{21} = 0$, $a_{22} = 10$, $a_{31} = 2$, $a_{32} = 1$. Since we are assuming chains with single customers, we subdivide chain 2 into 3 chains: 2, 3, and 4. Therefore, $K = 4$.

- Computations for $k = 1$ (chain 1 is added).

  $P^1(1,0,0) = L^1_{11} = \frac{a_{11}}{a_{11}+a_{21}} = 0.7143$
  $P^1(0,1,0) = L^1_{21} = 0$
  $P^1(0,0,1) = L^1_{31} = 0.2857$

  Therefore:

  $P^1_1(0) = P^1(0,0,1)$; $\quad P^1_1(1) = P^1(1,0,0)$
  $P^1_2(0) = P^1(1,0,0) + P^1(0,0,1) = 1$; $\quad P^1_2(1) = 0$
  $P^1_3(0) = P^1(1,0,0)$; $\quad P^1_3(1) = P^1(0,0,1)$.

- $k = 2$ (one customer of chain 2 is added).

  $\forall j$, calculate τ^k_j.

  $\tau^2_1 = 0.5833$, $\tau^2_2 = 1$, $\tau^2_3 = 0.7777$

  $\lambda^2_2 = = \ 0.08861$

  $L^2_{12} = 0$; $L^2_{22} = 0.8861$; $L^2_{32} = 0.1139$

  $P^2(2,0,0) = 0$; $\quad P^2(1,1,0) = 0.6329$; $\quad P^2(1,0,1) = 0.06329$
  $P^2(0,2,0) = 0$; $\quad P^2(0,1,1) = 0.2532$; $\quad P^2(0,0,2) = 0.05063$

  $P^2_1(0) = 0.3038$, $\quad P^2_1(1) = 0.6962$, $\quad P^2_1(2) = 0$
  $P^2_2(0) = 0.1139$, $\quad P^2_2(1) = 0.8861$, $\quad P^2_2(2) = 0$
  $P^2_3(0) = 0.6329$, $\quad P^2_3(1) = 0.3165$, $\quad P^2_3(2) = 0.05063$

- $k = 3$ (another chain 2 customer is added)

  $\forall j$, calculate τ^k_j.

  $\tau^3_1 = 0.5896$, $\tau^3_2 = 1$, $\tau^3_3 = 0.7054$

  $\lambda^3_3 = 0.08758$

  $L^3_{13} = 0$, $L^3_{23} = 0.8758$, $L^3_{33} = 0.1242$

  $P^3(3,0,0) = P^3(2,1,0) = P^3(2,0,1) = P^3(0,3,0) = 0$
  $P^3(1,2,0) = 0.5543$, $\quad P^3(1,1,1) = 0.1109$, $\quad P^3(1,0,2) = 0.01109$
  $P^3(0,2,1) = 0.2217$, $\quad P^3(0,1,2) = 0.08869$, $\quad P^3(0,0,3) = 0.0133$

$P_1^3(0) = 0.3237$, $P_1^3(1) = 0.6763$, $P_1^3(2) = 0$, $P_1^3(3) = 0$
$P_2^3(0) = 0.02439$, $P_2^3(1) = 0.1996$, $P_2^3(2) = 0.7761$, $P_2^3(3) = 0$
$P_3^3(0) = 0.5543$, $P_3^3(1) = 0.3326$, $P_3^3(2) = 0.09978$, $P_3^3(3) = 0.0133$

- $k = 4$ (last chain 2 customer is added).

  $\forall j$, calculate τ_j^4.

  $\tau_1^4 = 0.5966$, $\tau_2^4 = 0.7205$, $\tau_3^4 = 0.6361$

  $\lambda_4^4 = 0.08758$

  $L_{14}^4 = 0$, $L_{24}^4 = 0.8983$, $L_{34}^4 = 0.1017$

  $P^4(4,0,0) = P^4(3,1,0) = P^4(3,0,1) = P^4(2,2,0) = P^4(2,1,1) = P^4(2,0,2) = P^4(0,4,0) = 0$
  $P^4(1,3,0) = 0.5381$, $P^4(1,2,1) = 0.1076$, $P^4(1,1,2) = 0.02152$
  $P^4(1,0,3) = 0.002152$, $P^4(0,3,1) = 0.2153$, $P^4(0,2,2) = 0.08609$
  $P^4(0,1,3) = 0.02582$, $P^4(0,0,4) = 0.003442$

The final results when the spare memory unit can fail (assume $\lambda_{spare} = 1/20$) are:

- $k = 4$

  $\forall j$, calculate τ_j^4.

  $\tau_1^4 = 0.5966$, $\tau_2^4 = 0.8657$, $\tau_3^4 = 0.6361$

  $\lambda_4^4 = 0.0762$

  $L_{14}^4 = 0$, $L_{24}^4 = 0.8802$, $L_{34}^4 = 0.1198$

  $P^4(4,0,0) = P^4(3,1,0) = P^4(3,0,1) = P^4(2,2,0) = P^4(2,1,1) = P^4(2,0,2) = P^4(0,4,0) = 0$
  $P^4(1,3,0) = 0.5068$, $P^4(1,2,1) = 0.1267$, $P^4(1,1,2) = 0.02535$
  $P^4(1,0,3) = 0.002535$, $P^4(0,3,1) = 0.2027$, $P^4(0,2,2) = 0.1014$
  $P^4(0,1,3) = 0.03041$, $P^4(0,0,4) = 0.004054$

References

[CONW86a] A.E. Conway & N.D. Georganas, "RECAL - A New Efficient Algorithm for the Exact Analysis of Multiple-Chain Closed Queueing Networks" *JACM*, vol. 33, no. 4 pp. 768-791, October 1986.

[CONW86b] A.E. Conway, E. de Souza e Silva & S.S. Lavenberg "Mean Value Analysis by Chain of Product Form Queueing Networks", IBM Research Report, RC 11641, April 1986, to appear in IEEE Transactions on Computers.

[DeSO84] E. de Souza e Silva & R.R. Muntz "Simple Relationships Among Moments in Product Form Queueing Network Models" UCLA Computer Science Department, 1984, to appear in IEEE Transactions on Computers

[GOYA85] A. Goyal, S.S. Lavenberg & K.S. Trivedi "Probabilistic Modeling of Computer Systems Availability" IBM Research Report, RC 11076, March 1985, to appear in Annals of Oper. Res.

[GOYA86] A. Goyal, W.C. Carter, E. de Souza e Silva, S.S. Lavenberg & "The System Availability Estimator", *Proceedings of FTCS-16*, pp. 84-89, July 1986

[LAVE83] S.S. Lavenberg (Editor), "Computer Performance Modeling Handbook", *Academic Press, New York*, 1983.

[MEYE80] J.F. Meyer, "On Evaluating Performability of Degradable Computer Systems", *IEEE Transactions on Computers*, vol. C-29, pp. 720-731, 1980.

[REIS76] M. Reiser & H. Kobayashi "On the Convolution Algorithm for Separable Queueing Networks", *International Symposium on Computer Performance* pp. 109-117, 1976.

[REIS80] M. Reiser & S.S. Lavenberg, "Mean Value Analysis of Closed Multichain Queueing Networks" *JACM*, no. 27, pp. 313-322, 1980.

[REIS81] M. Reiser, "Mean-Value Analysis and Convolution Method for Queue-Dependent Servers in Closed Queueing Networks", *Performance Evaluation*, no. 1, pp. 7-18, 1981.