The Solution of Separable Queueing Network Models Using Mean Value Analysis

John Zahorjan
Eugene Wong

Department of Computer Science
University of Washington
Seattle, Washington 98195

ABSTRACT

Because it is more intuitively understandable than the previously existing convolution algorithms, Mean Value Analysis (MVA) has gained great popularity as an exact solution technique for separable queueing networks. However, the derivations of MVA presented to date apply only to closed queueing network models. Additionally, the problem of the storage requirement of MVA has not been dealt with satisfactorily. In this paper we address both these problems, presenting MVA solutions for open and mixed load independent networks, and a storage maintanence technique that we postulate is the minimum possible of any "reasonable" MVA technique.

1. INTRODUCTION

Separable queueing network models are a subset of the general class of queueing models distinguished by the fact that the equilibrium distributions of the states of these models have a particularly simple analytic form. This fact has been used to construct highly efficient exact solution techniques for them. Because these techniques are efficient, and the solutions of the more general class of network models are intractable (they require excessive time and space), separable queueing networks have become the most commonly used analytic models of computer systems.

Historically, the convolution algorithms [Buzen 73], [Chandy et al. 75], [Reiser & Kobayashi 76] were the first efficient, exact solution techniques for separable networks. The convolution algorithms exploit the form of the equilibrium state probabilities to save computational effort. These algorithms are complicated in the sense that it is often not clear intuitively how they work.

Reiser and Lavenberg [78,80] developed an alternative efficient solution technique, Mean Value Analysis (MVA), that was designed to overcome some numerical problems that can arise with the convolution algorithms. This algorithm has widespread appeal, primarily because it is much easier to understand and implement than the convolution algorithms. It should be noted, however, that the convolution algorithm is more general than MVA, and that there exist separable networks that can be solved satisfactorily by the former but not by the latter.

To date, most published applications of queueing networks to computer system modeling have involved closed models. Two other types of models, open and mixed, are becoming increasingly important, however, since they are natural representations of transaction processing and mixed batch/timesharing systems respectively. Although solutions for open and mixed models have recently appeared ([Chandy & Sauer 80], [Schwetman 80]), no simple, general derivation of them has been shown. The next section of this paper presents simple derivations of general results for load independent open, closed, and mixed networks, the purpose of which is to encourage the use of these more general models by providing simple explanations of their solutions.

The solution methods for both closed and mixed networks involves the calculation of solutions for all populations from zero customers in the network to the full customer population. The precise manner in which these solutions are computed has significant effect on the storage requirement of the technique. The latter part of this paper presents a new storage scheme that is more efficient than the previously published methods.

In section 2 we present a brief overview of the known MVA solution method for closed networks, and then extend this technique in a natural way to include open and mixed networks. In section 3 a storage efficient scheme for closed networks is presented. Section 4 concludes with an example utilizing the results in the previous two sections.

2. MEAN VALUE ANALYIS SOLUTION OF SEPARABLE NETWORKS

Mean value analysis [Reiser & Lavenberg 80] is based on applications of Little's theorem [Little 61] to separable queueing network models [Baskett et

This work was supported in part by the University of Washington Graduate School Research Fund and by the National Science Foundation under Grant No. MCS-8104879.

al. 75]. The first of these applies Little's theorem to the entire network to derive system throughput:

$$\lambda_r(\vec{N}) = \frac{N_r}{\sum_k V_{r,k} W_{r,k}(\vec{N})} \qquad (1)$$

Here $\lambda_r(\vec{N})$ is the system throughput of class* r with population $\vec{N} = (N_1, N_2, \ldots, N_R)$ in the network, N_r is the multiprogramming level of class r, $V_{r,k}$ is the average number of visits to device k made by a class r job, and $W_{r,k}(\vec{N})$ is the mean time spent in queue or in service (i.e., the response time) at device k by a class r customer when the network population is $\vec{N}$. Note that (1) could be computed from input parameter data if the values of the $W_{r,k}(\vec{N})$ were known.

The second application of Little's theorem derives the mean queue lengths at each device from device throughput and mean response time:

$$\bar{n}_{r,k}(\vec{N}) = V_{r,k}\lambda_r(\vec{N})W_{r,k}(\vec{N}) \qquad (2)$$

where $\bar{n}_{r,k}(\vec{N})$ is the mean class r queue length at device k with population $\vec{N}$ in the network.

To compute equation (1), and hence equation (2), the values of the $W_{r,k}(\vec{N})$ must be known. It is the computation of these values that is the key step in the MVA technique. For Infinite Servers+ (IS) service centers, we have by definition that $W_{r,k}(\vec{N}) = S_{r,k}$, the mean service time of class r at center k. Thus, these service centers pose no problem in closed, open, or mixed networks. In light of this, the remainder of our discussion refers to FCFS service centers only.

2.1. Closed Networks

The required expression for the mean response times of closed separable models with R customer classes is given [Reiser & Lavenberg 80] by

$$W_{r,k}(\vec{N}) = S_{r,k}(1 + \sum_{j=1}^{R} \bar{n}_{j,k}(\vec{N} - \vec{e}_r)) \qquad \text{(3.closed)}$$

An intuitive explanation of (3.closed) is easily given for first-come-first-served (FCFS) and processor sharing (PS) service centers. In both cases the explanation relies on the arrival instant distribution theorem [Sevcik & Mitrani 79] that the mean number of customers seen in queue by an arriving customer is equal to the equilibrium number of customers there in the network with the same population less the arriving customer. Using this theorem, (3.closed) can be explained for FCFS centers as the time spent in service ($S_{r,k}$) plus the time spent waiting for all customers ahead of the arriving customer to complete service ($S_{r,k}\sum_j \bar{n}_{j,k}(\vec{N} - \vec{e}_r)$). For PS centers, (3.closed) is simply the basic service time requirement times a blow-up factor that is caused by competition with the other customers at the center for use of the server. It should be noted that although the arrival instant distribution theorem provides a simple explanation of (3.closed), it is not a fundamental part of the MVA technique. In fact, the original report [Reiser & Lavenberg 78] proved (3.closed) without proving the arrival instant theorem.

Combining (3.closed) with (1) and (2) above we get the following algorithm, which is the MVA technique for closed, load independent queueing network models. We note here that at each pass through the **FOR** loop, solutions for only some of the $\vec{n}$ can be computed. The particular order in which the solutions are computed determines the storage requirement of the technique. In section 3 this problem is considered in more detail.

FIGURE 1 - MVA ALGORITHM (Closed Networks)

Initialize: $\bar{n}_{r,k}(\vec{0}) = 0$

FOR all feasible $\vec{n} \leq \vec{N}$ **DO**

$$W_{r,k}(\vec{n}) = \begin{cases} V_{r,k}S_{r,k} & IS \\ V_{r,k}S_{r,k}(1 + \sum_{j=1}^{R} \bar{n}_{j,k}(\vec{n} - \vec{e}_r)) & Otherwise \end{cases}$$

$$\lambda_r(\vec{n}) = \frac{n_r}{\sum_k W_{r,k}(\vec{n})}$$

$$\bar{n}_{r,k}(\vec{n}) = \lambda_r(\vec{n})W_{r,k}(\vec{n})$$

2.2. Open Networks

The throughput rates at each center of an open network are input parameters, and thus equation (1) is replaced by these values. However, equation (2) must still be computed, and so there is a need to calculate the $W_{r,k}(\vec{N})$.

In order to derive an equation corresponding to (3.closed) for open networks, we appeal to the arrival instant theorem and our knowledge of separable networks. From the results of [Baskett et al. 75] we know that for any load independent, separable network QN we can construct an equivalent (possibly the same) separable network QN' consisting of only FCFS and IS service centers. QN' has the same $V_{r,k}S_{r,k}$ products and customer population $\vec{N}$ as QN. The state space of QN' isomorphic with that of QN, and corresponding states of each have the same equilibrium probabilities. Thus the two networks have identical high level performance measures. In light of this, we restrict our discussion to networks consisting entirely of FCFS and IS service centers. Doing so allows us to formulate equations corresponding to (3.closed) by appealing to the arrival instant theorem, thus avoiding much of the tedium of the proof of these equations from first principles.

The arrival instant theorem for open networks states that the mean queue length seen at arrival to a queue by a customer is identical to the equilibrium mean queue length. This result coincides with our intuition. Since all product form networks are equivalent to some network with all service centers in series (i.e., no loops), and the $M \rightarrow M$ property [Muntz 73] holds in product form networks, arrival instants in open models occur at random. Thus the arrival instant distribution must be the equilibrium distribution, and so $W_{r,k} = S_{r,k}(1 + \sum_{j=1}^{R} \bar{n}_{j,k})$. This observation, combined with the fact that the throughput of class r at device k, $\lambda_{r,k}$, is an input parameter of an open network (or can be trivially derived from the inputs), gives

*Since all load independent networks with class changing are equivalent to networks without class changes ([Krzesinski & Teunissen 77], [Bruell 78]), we use the term "class" in the same manner that some authors have used "chain".

+IS service centers have sufficient servers that all customers can be served in parallel. Thus, no queueing delay occurs at these centers.

$$\bar{n}_{r,k} = \lambda_{r,k} W_{r,k}(\vec{N})$$
$$= \lambda_{r,k} S_{r,k}(1 + \sum_{j=1}^{R} \bar{n}_{j,k}))$$

We now note that the parenthesized expression above is independent of the class r. Thus we have

$$\bar{n}_{s,k} = \frac{\lambda_{s,k} S_{s,k}}{\lambda_{r,k} S_{r,k}} \bar{n}_{r,k}$$

for any two classes r and s. Using this information, and defining

$$\rho_{r,k} = \lambda_{r,k} S_{r,k}$$

which is the utilization of device k by class r, we have

$$\bar{n}_{r,k} = \rho_{r,k}(1 + \sum_{j=1}^{R} \frac{\rho_{j,k}}{\rho_{r,k}} \bar{n}_{r,k})$$

From this we get

$$\bar{n}_{r,k} = \frac{\rho_{r,k}}{1 - \sum_{j=1}^{R} \rho_{j,k}}$$

and, applying Little's theorem,

$$W_{r,k} = \frac{S_{r,k}}{1 - \sum_{j=1}^{R} \rho_{j,k}} \qquad \text{(3.open)}$$

Performance measures of interest for all classes can then be computed from these equations.

2.3. Mixed Networks

Mixed networks consist of some open and some closed classes. For these networks, the throughput rate of class r at device k is an input parameter for the open classes, but must be calculated as part of the solution for the closed classes.

The arrival instant theorem for mixed networks shows that customers in the open classes see the equilibrium queue length at arrival, while customers in the closed classes see the equilibrium queue length with a population diminished by one customer of their own class. Let O be the set of open classes, and C the set of closed classes. Then, for any class $r \in O$, we have

$$\bar{n}_{r,k}(\vec{n}) = \lambda_{r,k} S_{r,k}(1 + \sum_{c \in C} \bar{n}_{c,k}(\vec{n}) + \sum_{c \in O} \bar{n}_{c,k}(\vec{n}))$$
$$= \frac{\rho_{r,k}(1 + \sum_{c \in C} \bar{n}_{c,k}(\vec{n}))}{1 - \sum_{o \in O} \rho_{o,k}} \qquad \text{(3.mixed.open)}$$

where $\vec{n} = (n_1, n_2, \ldots, n_{|C|})$ represents the closed class populations.

The mean response time of any closed class r is given by

$$W_{r,k}(\vec{n}) = S_{r,k}\ (1 + \sum_{c \in C} \bar{n}_{c,k}(\vec{n} - \vec{e}_r) + \sum_{c \in O} \bar{n}_{c,k}(\vec{n} - \vec{e}_r))$$
$$= S_{r,k} \left(1 + \sum_{c \in C} \bar{n}_{c,k}(\vec{n} - \vec{e}_r) + \frac{(1 + \sum_{c \in C} \bar{n}_{c,k}(\vec{n} - \vec{e}_r)) \sum_{c \in O} \rho_{c,k}}{1 - \sum_{o \in O} \rho_{o,k}}\right)$$
$$= \frac{S_{r,k}(1 + \sum_{c \in C} \bar{n}_{c,k}(\vec{n} - \vec{e}_r))}{1 - \sum_{o \in O} \rho_{o,k}} \qquad \text{(3.mixed.closed)}$$

Equation (3.mixed.closed) shows that the performance measures of the closed classes in a mixed network are identical to those that would be observed in an equivalent closed network with a population consisting only of the closed classes, and with the service rates of all centers degraded by the total usage of all the open classes in the mixed network. This effect had been noted previously by [Krezinski & Teunnissen 77], although their proof was much more tedious, and did not give an expression for the performance measures of the open classes of the network.

A solution technique for mixed networks can be constructed using equation (3.mixed.closed) above. First, the total utilizations of all devices by the open classes are computed using the utilization law $\rho_{r,k} = \lambda_r V_{r,k} S_{r,k}$. A closed model containing only the closed class customers is then constructed. The loadings of this model are the $V_{r,k} S_{r,k}$ of the original model, divided by the utilizations of the open classes at each center. The solution of this model yields throughputs, response times, and queue lengths for the closed classes (utilizations are obtained by multiplying these throughputs with the loadings of the original mixed network). Performance measures for the open classes can then be computed for any populations of interest using equation (3.mixed.open) above.

In section 4 we will illustrate the use of these formulae with an example. In the next section we first discuss the efficient implementation of the basic MVA recursion required for closed and mixed networks.

3. STORAGE CONSIDERATIONS WITH MVA

Because the solution of a network with closed class population $\vec{n}$ depends on the R sets of mean queue lengths for the network with populations $\vec{n} - \vec{e}_r$, $r = 1, \ldots, R$, it is apparent that at any point in the MVA recursion solutions of the network for a number of intermediate populations must be maintained. In reference to figure 1, the iteration defined by the **FOR** statement must occur over all $\vec{n} \leq \vec{N}$, but at each step of the iteration, solutions can be computed for only those $\vec{n}$ for which the R solutions for populations $\vec{n} - \vec{e}_r$ are already known. In this section we present a new scheme for determining the order and manner of storage of these intermediate solutions that is more efficient than those previously demonstrated.

The most straightforward way to implement the **FOR** loop of figure 1 is shown in figure 2. At each pass through the body of the loop, it is certain that the mean queue lengths of all necessary predecessor populations have been computed. By simply storing the mean queue lengths for all populations from $\vec{0}$ to $\vec{N}$, all required values are available when needed in an iteration step. This was the scheme suggested by Reiser and Lavenberg [78] in the original presentation of MVA.

The above scheme requires that queue lengths for $\prod_{r=1}^{R}(N_r + 1)$ populations be stored simultaneously. Even for moderately sized networks, this requirement can be excessive. A simple modification of

FIGURE 2 - SIMPLE MVA RECURSION

```
FOR n_1 = 0 UNTIL N_1 DO
  FOR n_2 = 0 UNTIL N_2 DO

    FOR n_R = 0 UNTIL N_R DO

      <body of loop>

    END

  END;
END;
```

this basic scheme [Myhre 79] results in significant savings, however. Instead of storing all intermediate populations, only those populations for which $n_g = c$, $1 \leq c \leq N_g$, are stored, where g is the class with the largest population. Using this scheme, the loop for class g must be the outermost, and values for queue lengths with population $(n_1,\ldots,n_g,\ldots,n_R)$ can be overlayed directly onto the storage used for population $(n_1,\ldots,n_g-1,\ldots,n_R)$. The number of solutions that must be stored simultaneously is therefore $\prod_{r=1,r\neq g}^{R} (N_r+1)$.

An alternative iteration order [Zahorjan 80] is to compute at each step the solutions of all populations $\vec{n} = (n_1,\ldots,n_R)$ such that $\sum_{r=1}^{R} n_r \equiv n$ is a constant. In what follows we assume that the classes are numbered in order of increasing population size. Define $P_n(\vec{N})$ to be this set of populations. For example, figure 3 shows the $P_n(\vec{N})$ for a network with three closed classes and $\vec{N} = (1,2,2)$. The solution of the network is calculated by first computing the queue lengths for all populations $P_1(\vec{N})$, then using these to compute queue lengths for populations $P_2(\vec{N})$, and so forth, up to the full population.

Let $\omega(\vec{N})$ be the maximum cardinality (over n) of the sets of populations $P_n(\vec{N})$. $\omega(\vec{N})$ is a measure of the storage requirement of this scheme. It seems intuitively clear that $\omega(\vec{N})$ is less than the comparable measure for the previously discussed method, $\prod_{r=1,r\neq g}^{R} (N_r+1)$. Unfortunately, it appears to be extremely difficult to derive a closed form expression for the actual storage requirement. Brumfeld [81] has demonstrated that a special case of Buzen's convolution algorithm [Buzen 73] can be used to numerically calculate $\omega(\vec{N})$. This procedure is shown in figure 4. The computation requires time $O(RN^2)$, and results in a matrix H of size $R \times (N+1)$, where $N = \sum_{r=1}^{R} N_r$. Each element $[n,r]$ of the matrix represents the number of distinct valid populations that can be constructed by allocating n customers to the first r classes such that multiprogramming constraints are observed. Thus, $\omega(\vec{N})$ is given by $\max_n \{H[R,n]\}$. Figure 5 shows the matrix H resulting from the example given in figure 3.

The basic MVA solution technique requires that solutions of populations $P_{n-1}(\vec{N})$ be calculated, and

FIGURE 3

$\vec{N}$ = (1,2,2) (1 class A, 2 class B, and 2 class C customers)

n	Populations					
5:	(1,2,2)					
4:	(1,2,1)	(1,1,2)	(0,2,2)			
3:	(1,2,0)	(1,1,1)	(0,2,1)	(1,0,2)	(0,1,2)	
2:		(1,1,0)	(0,2,0)	(1,0,1)	(0,1,1)	(0,0,2)
1:				(1,0,0)	(0,1,0)	(0,0,1)

that information about mean queue lengths for these populations be maintained. This information is then used to calculate solutions for populations $P_n(\vec{N})$, and so forth, until the solution of the network with the full population is derived. A significant problem in implementing this scheme in an MVA solution package is how to overlay the queue lengths of populations $P_n(\vec{N})$ over the statistics for populations $P_{n-1}(\vec{N})$. If this overlaying were not done, the storage requirement for the technique would be doubled.

FIGURE 4

```
H[*,*] := 0;
TOTAL-POP := 0;
FOR n:=0 UNTIL N_1 DO
  H[1,n] := 1;
FOR r:=2 to R DO
    TOTAL-POP := TOTAL-POP + N_{r-1};
    FOR n:=0 UNTIL TOTAL-POP DO
      sum := 0;
      FOR j:=n DOWNTO MAX {0,n-N_r} DO
        sum := sum + H[r-1,j];
      H[r,n] := sum;
      end;
    end;
```

A simple and efficient solution to this problem is provided by the following. The solutions for populations $P_n(\vec{N})$ are partitioned into subsets $P_{n,c}(\vec{N})$ such that for all populations in the subset $n_R = c$. Then, the storage for $P_n(\vec{N})$ is organized as shown below:

$$P_{n,0}(\vec{N}) \mid P_{n,1}(\vec{N}) \mid \cdots \mid P_{n,m(n)}(\vec{N})$$

where $m(n) = \text{MIN}\{n, N_R\}$. The overlay of $P_{n+1}(\vec{N})$ on $P_n(\vec{N})$ can be illustrated as

$$\begin{array}{c|c|c|c|c} P_{n+1,0}(\vec{N}) & P_{n+1,1}(\vec{N}) & P_{n+1,2}(\vec{N}) & \cdots & P_{n+1,m(n+1)}(\vec{N}) \\ & P_{n,0}(\vec{N}) & P_{n,1}(\vec{N}) & \cdots & P_{n,m(n)}(\vec{N}) \end{array}$$

It is clear that $|P_{n+1,j+1}(\vec{N})| = |P_{n,j}(\vec{N})|$, since in each case the number of populations is equal to the number of ways of distributing $n-j$ customers over classes $1,2,\ldots,R-1$. Thus, there is sufficient room to perform the substitution.

FIGURE 5

	CLASS		
	A	B	C
0	1	1	1
1	1	2	3
2		2	5
3		1	5
4			3
5			1

By ordering the storage of the populations within $P_{n+1,j+1}(\vec{N})$ and $P_{n,j}(\vec{N})$ in the same way, each population of $P_{n,j}(\vec{N})$ is overwritten only when it is no longer needed. A particular population $(n_1,\ldots,n_{R-1},j)$ of $P_{n,j}(\vec{N})$ is needed to compute solutions for the (at most) R populations that are 1 greater in any one of the R indices. $R-1$ of these descendant populations are obtained by incrementing some class other than class R, and so fall in $P_{n+1,j}(\vec{N})$, which is computed before the storage for $P_{n,j}(\vec{N})$ is overlayed. The remaining population $(n_1,n_2,\ldots,j+1)$ replaces the original population when it is computed, but at that time the original population is no longer needed. Thus, every population is replaced immediately when it is no longer needed, and so no unnecessary storage is preserved. In light of this fact, we conjecture that this scheme requires the simultaneous storage of solutions of the network for the fewest populations of all those for which the iteration is done according to the scheme illustrated in figure 3.

It is fairly easy to show that the scheme discussed above requires less storage than Myhre's method. The key to this is to note that the storage requriement of Myhre's technique is $\sum_n H[R-1,n]$, that is, the sum of all elements of column $R-1$ of H. The storage requirement of the new scheme at step n of the iteration is

$$|P_{n,0}(\vec{N})| + |P_{n-1}(\vec{N})| = H[R-1,n] + H[R,n] = \sum_{0\le j=n-N_R-1}^{n} H[R-1,j]$$

Thus, the storage requirement for any n is given by the sum of a portion of the elements of column $R-1$ of H, and so must be less than that of Myhre's scheme, the most efficient previously published technique.

In addition to the problem of storing the solutions of the intermediate populations, a solution package implementing this scheme must deal with the problem of enumerating the populations of $P_n(\vec{N})$. In general this is a difficult problem, but given an enumeration of $P_{n-1}(\vec{N})$, efficient schemes exist. These schemes, and the trivial enumeration of $P_0(\vec{N})$ provide the necessary function.

A particularly simple enumeration scheme is suggested by examing figure 3. All populations $P_{n+1,j}(\vec{N})$ can be obtained for $j\ge 1$ by increasing each population of $P_n(\vec{N})$ by one class R customer (and eliminating those populations that violate the multiprogramming level constraints). Thus only the enumeration of $P_{n+1,0}(\vec{N})$ presents a problem.

Let $P^j_{n,0}(\vec{N})$ be the subset of populations of $P_{n,0}(\vec{N})$ for which classes $R, R-1, \ldots, R-j$ have population level zero. $P_{n+1,0}(\vec{N})$ can be generated by adding one customer of class $R-j-1$ to each population $P^j_{n,0}(\vec{N})$ for $j=0,\ldots,R-2$. This enumeration of populations can then be used to determine which solutions are necessary in the next step of the MVA iteration.

In the next section we consider a simple example that illustrates the results of sections 2 and 3. This example should help to clarify the actual replacement scheme used.

4. AN EXAMPLE

In this section an example is used to illustrate our results. The network to be solved consists of three closed classes, and two open classes. Such a model might be constructed for a system processing both batch jobs (the closed classes) and data base transactions (the open classes). The populations of closed classes A,B, and C are 1, 2, and 2 respectively. The arrival rate of open classes D and E are 2 transactions/second and 3 transactions/second respectively. There are three service centers in the network. Center 1 has an IS scheduling discipline, while centers 2 and 3 have PS and FCFS respectively. The service demands ($V_{r,k}S_{r,k}$ products) of the classes at each device are shown in figure 6.

FIGURE 6

		$V_{r,k}S_{r,k}$		
Class	N_r	C1	C2	C3
A	1	10	2	6
B	2	10	3	2
C	2	10	1	4
D	2/sec.	0	1/8	1/8
E	3/sec.	0	1/6	1/12

The solution of this model begins by first computing the total utilization of each non-IS service center by the open classes. This is done using the utilization law [Denning & Buzen 78] that utilization equals the product of throughput and service time. Thus, $\rho_{D,2} = .25$, $\rho_{E,2} = .5$, $\rho_{D,3} = .25$, and $\rho_{E,3} = .25$. Next the solution of the three closed classes is computed by solving the closed network with the $V_{r,k}S_{r,k}$ loadings for each multiplied by $\frac{1}{.25}$ for center 2, and $\frac{1}{.5}$ for center 3. This inflation of the basic loading accounts for the interference of the open classes.

The closed model solution is computed using the MVA algorithm of figure 1. The storage management that takes place is shown in figure 3. The results of this solution yield the following mean queue lengths for the closed classes:

$$\bar{n}_{A,1} = .172 \quad \bar{n}_{A,2} = .372 \quad \bar{n}_{A,3} = .456$$
$$\bar{n}_{B,1} = .416 \quad \bar{n}_{B,2} = 1.136 \quad \bar{n}_{B,3} = .448$$
$$\bar{n}_{C,1} = .514 \quad \bar{n}_{C,2} = .584 \quad \bar{n}_{C,3} = .903$$

These can then be used in an application of equation (3.mixed.open) to compute the following open class queue lengths:

$$\bar{n}_{D,2} = 3.0916 \quad \bar{n}_{D,3} = 1.404$$
$$\bar{n}_{E,2} = 6.183 \quad \bar{n}_{E,3} = 1.404$$

Additional performance measures, such as device utilizations, can be computed straightforwardly using well known relationships with mean response time and mean queue length (see [Denning & Buzen 78] for a survey of these formulae).

5. SUMMARY

In this paper we have addressed the solution of separable queueing network models using the Mean

Value Analysis [Reiser & Lavenberg 80] solution technique. The basic MVA derivation has been extended to include both open and mixed load independent networks. In addition, a new scheme that reduces the storage requirement of the closed and mixed solution algorithms was developed. Together, these results represent methods for broadening the applicability of the MVA technique to the important class of open and mixed models, and to include larger closed models than were previously feasible.

REFERENCES

[Baskett et al. 75]
Baskett, F., Chandy, K.M., Muntz, R.R., and Palacios, F., "Open, closed and mixed networks of queues with different classes of customers", J. ACM 22,2 (April 1975), 248-260.

[Bruell 78]
Bruell, S.C., "On single and multiple job class queueing network models of computer systems", Ph.D. Thesis, Purdue University, December 1978.

[Brumfeld 81]
Brumfeld, J., Technical Report, Purdue University, to appear.

[Buzen 73]
Buzen, J.P., "Computational algorithms for closed queueing networks with exponential servers", Comm. ACM 16,9 (Sept. 1973), 527-531.

[Chandy et al. 75]
Chandy, K.M., Herzog, U., and Woo, L.S., "Parametric analysis of queueing networks", IBM J. Res. Dev. 19 (Jan. 1975), 36-42.

[Chandy and Sauer 80]
Chandy, K.M., and Sauer, C.H., "Computational algorithms for product form queueing networks", Commun. ACM 20,10 (Oct. 1980), 573-583.

[Denning & Buzen 78]
Denning, P.J., and Buzen, J.P., "The operational analysis of queueing network models", ACM Comp. Surveys 10,3 (Sept. 1978), 225-262.

[Krzesinski & Teunnisen 77]
Krzesinski, A.E., and Teunissen, P., "Efficient computational forms for the normalising constant and the statistical measures of mixed, multiclass queueing networks", Rpt. RW77-04, Dept. of Computer Science, Univ. of Stellenbosch, 1977.

[Little 61]
Little, J.D.C., "A proof of the queueing formula $L=\lambda W$", Oper. Res. 9,3 (May 1961), 383-387.

[Muntz 63]
Muntz, R.R., "Poisson departure processes and queueing networks", 7th Annual Princeton Conf. on Inf. Sci. and Sys., March 1973.

[Myhre 79]
Myhre, S., "Performance modelling of computer systems", M.Sc. Thesis, Univ. of Washington, 1979.

[Reiser & Kobayashi 76]
Reiser, M., and Kobayashi, H., "On the convolution algorithm for separable queueing networks", Proc. Int. Symp. CPMME 1976, 109-117.

[Reiser & Lavenberg 78]
Reiser, M., and Lavenberg, S.S., "Mean value analysis of closed multichain queueing networks", IBM Res. Rpt. RC 7023, 1978.

[Reiser & Lavenberg 80]
Reiser, M., and Lavenberg, S.S., "Mean value analysis of closed multichain queueing networks", J. ACM 27,2 (April 1980), 313-322.

[Schwetman 80]
Schwetman, H., "Implementing the mean value algorithm for the solution of queueing network models", Tech. Rpt. CSD-TR 355, Dept. of Computer Sciences, Purdue Univ., Dec. 1980.

[Sevcik & Mitrani 79]
Sevcik, K.C., and Mitrani, I., "The distribution of queueing network states at input and output instants", J. ACM 28,2 (April 1981), 358-371.

[Zahorjan 80]
Zahorjan, J., "The approximate solution of large queueing network models", Tech. Rpt. CSRG-122, Comp. Sys. Res. Grp., Univ. of Toronto, 1980.

Acknowledgements

The authors gratefully acknowlege the assistance provided by discussions with Jeff Brumfeld and Ken Sevcik in the prepartion of this work.