PAM—A Noniterative Approximate Solution Method for Closed Multichain Queueing Networks[1]

Ching-Tarng Hsieh[2] and Simon S. Lam

Department of Computer Sciences
The University of Texas at Austin
Austin, Texas 78712-1188

TR-87-28 July 1987
Revised, March 1988

Abstract

Approximate MVA algorithms for separable queueing networks are based upon an iterative solution of a set of modified MVA formulas. Although each iteration has a computational time requirement of $O(MK^2)$ or less, many iterations are typically needed for convergence to a solution. (M denotes the number of queues and K the number of closed chains or customer classes.) We present some faster approximate solution algorithms that are *noniterative*. They are suitable for the analysis and design of communication networks which may require tens to hundreds, perhaps thousands, of closed chains to model flow-controlled virtual channels. The basis of our method is the distribution of a chain's population proportional to loads to get initial estimates of mean queue lengths. This is the same basis used in the derivation of proportional upper bounds for single-chain networks; for a multichain network, such a proportional distribution leads to approximations rather than upper bounds of chain throughputs. Nevertheless, these approximate solutions provide chain throughputs, mean end-to-end delays, and server utilizations that are sufficiently accurate for the analysis and design of communication networks and possibly other distributed systems with a large number of customer classes. Three PAM algorithms of increasing accuracy are presented. Two of them have time and space requirements of $O(MK)$. The third algorithm has a time requirement of $O(MK^2)$ and a space requirement of $O(MK)$.

[1]This work was performed at the University of Texas at Austin under National Science Foundation grant no. ECS-8304734 and grant no. NCR-8613338. An abbreviated version of this paper appeared in *Proceedings ACM SIGMETRICS Conference*, May 1988, Santa Fe, New Mexico. Portions of this paper are reprinted by permission of the Association for Computing Machinery.

[2]Current address: ATT Bell Laboratories, Naperville, Illinois 60566.

1. Introduction

We are interested in fast solution algorithms for separable queueing networks with a large number of closed chains (also called customer classes), such as models of communication networks where each closed chain represents a flow-controlled virtual channel [9, 10, 12]. Tens to hundreds, perhaps thousands, of such flow-controlled virtual channels between source-destination node pairs may be active at a time in a practical network. Exact solution algorithms, such as the convolution and MVA algorithms, are obviously not applicable since their computational time and space requirements grow exponentially with the number of chains [11, 13]. In fact, they cannot be used for most networks with more than 6 or 7 chains. The tree convolution algorithm [8] as well as the tree MVA algorithms [5, 18] are more computationally efficient for the solution of networks with chains whose routes are sparse, by exploiting routing information. Nevertheless, the largest networks solved by the tree convolution algorithm have 32-50 chains, which are not sufficient for modeling many real networks. Furthermore, most algorithms for network design problems (e.g., optimal routing, topology optimization, etc.) involve a heuristic search for solutions that are optimum according to various network performance measures. A very fast algorithm is needed to evaluate these network performance measures at each of the numerous intermediate steps of such a heuristic search.

For the same reason, the approximate solution methods which have been shown empirically to have good accuracy [1, 2, 3, 12, 14, 15, 20] are deemed to be still too slow for communication network design problems. All of these methods are based upon an iterative solution of a set of modified MVA formulas. A single iteration of the Schweitzer algorithm has a computation time of $O(MK)$ while a single iteration of Linearizer or AQL has a computation time of $O(MK^2)$ where M denotes the number of queues and K the number of closed chains in the network [3, 14, 20]. (It was shown recently by de Souza e Silva and Muntz [16] that a single iteration of Linearizer can be implemented with a computation time of $O(MK^2)$ instead of $O(MK^3)$ as indicated by Chandy and Neuse [3].) Typically many iterations are needed for convergence to a solution, and it has been shown that for some networks convergence occurs extremely slowly [20]. Lastly, the accuracy of these methods has been examined only for networks with a small number of chains, i.e., those that can be solved exactly by the MVA algorithm.

Various methods for computing performance bounds are available [4, 6, 19]. However, for networks with a large number of chains, these bounds are generally too loose to be useful for communication network design.

The above considerations led us to investigate a class of faster approximation solution algorithms for closed multichain queueing networks. In particular, all of our algorithms are *noniterative*. We consider separable queueing networks of fixed-rate servers (also called queue-independent servers) and delay servers (also called infinite-server centers). Like all of the approximate solution methods referenced, our method is also based upon the MVA recursion formula. In our study of proportional throughput upper bounds for single-chain queueing networks, we found that they are very accurate for networks with small to medium population sizes [6]. These throughput upper bounds are obtained by distributing the population of a chain over the servers it visits proportional to server loads. In a multichain network, such a proportional distribution leads to approximations rather than upper bounds of chain throughputs. We found that these approximate solutions provide chain throughputs, mean end-to-end delays, and server utilizations that are sufficiently accurate for the analysis and design of communication networks [7, 9] and possibly some other distributed systems that have a large number of customer classes (see Sections 3 and 5); an iterative solution to improve accuracy is not necessary.

We shall refer to our method as the *Proportional Approximation Method (PAM)*. We present three algorithms of increasing accuracy that are based upon the distribution of a chain's population proportional to loads to get initial estimates of mean queue lengths: PAM_BASIC, PAM_IMPROVED, and PAM_TWO. The accuracy improvement of PAM_IMPROVED over PAM_BASIC is obtained by a simple scaling operation to ensure that the utilization of each server does not exceed one. The additional accuracy of PAM_TWO is obtained by executing the final two steps of the MVA recursion instead of just the last step. The computational time requirements are $O(MK)$ for PAM_BASIC and PAM_IMPROVED, and $O(MK^2)$ for PAM_TWO. Since PAM algorithms do not iterate, these are their total time requirements. All three PAM algorithms have space requirements of $O(MK)$.

The rest of this report is organized as follows. Algorithms PAM_BASIC and PAM_IMPROVED are presented in Section 2. In Section 3, we study their accuracy by comparing their predictions with exact solutions given by the tree convolution algorithm (TCA) for a network example and also for 100 randomly generated networks. These networks have fixed-rate servers only and possess characteristics of models of communication networks. We also show a correlation between the approximation errors for a chain and the utilization of any bottleneck server visited by the chain. (It is important to understand such a correlation because most network optimization algorithms are concerned with, in one way or another, eliminating bottlenecks). Algorithm PAM_TWO is presented in Section 4. In Section 5, we study the accuracy of PAM_TWO and PAM_IMPROVED by comparing their predictions with exact solutions given by the MVA algorithm for 500 networks generated randomly as specified by Zahorjan et al. [20]. These networks have both fixed-rate and delay servers and possess characteristics of models of computer systems.

2. Two PAM Algorithms with $O(MK)$ Computation Time

Consider a closed queueing network with K routing chains (also called customer classes by other authors). Let N_k be the customer population of the k^{th} chain. We refer to the vector $\underline{N}=(N_1,N_2,\ldots,N_K)$ as the *population vector* of the queueing network. If N_k in $\underline{N}$ is greater than or equal to 1, then $\underline{N}-\underline{1}_k$ refers to the population vector $\underline{N}$ with a chain k customer removed. Let τ_{mk} denote the load, or traffic intensity, of chain k customers at server m. (In models of communication networks, we also refer to τ_{mk} as the mean service time of chain k customers at server m with the assumption that visit ratios are the same for all servers visited by chain k customers.) We use $D_{mk}(\underline{n})$ to denote the approximate mean delay of chain k customers at server m, $q'_{mk}(\underline{n})$ to denote the approximate mean queue length of chain k customers at server m, and $T_k(\underline{n})$ to denote the approximate throughput of chain k customers, in a network whose population vector is $\underline{n}$.

PAM algorithms are based upon MVA formulas [13]. However, the iterations from population vector $\underline{0}$ to population vector $\underline{N}-\underline{1}_k$ in the MVA algorithm are skipped. Instead, mean queue length approximations $q'_{mh}(\underline{N}-\underline{1}_k)$ at population vector $\underline{N}-\underline{1}_k$ are obtained by distributing the population N_h of chain h over the servers it visits, proportional to τ_{mh}. This approach was the basis of proportional throughput upper bounds for single-chain networks in [6]. In multichain networks, this approach leads to approximations rather than upper bounds, as we shall see.

We present below two algorithms with time and space requirements of $O(MK)$. The first algorithm, PAM_BASIC, calculates throughputs of individual chains. The second algorithm, PAM_IMPROVED, calculates throughputs of chains and the utilizations of servers. The algorithm then checks server utilizations to see if any utilization exceeds 1. (This is possible because the chain throughputs are approximations.) The throughputs of those chains that visit a server whose utilization exceeds 1 are then scaled down.

Algorithm PAM_BASIC

Step 1: Calculate proportional approximations of mean queue lengths from

$$\gamma_{mk} = \tau_{mk} / \sum_{i=1}^{M} \tau_{ik}$$

for $m = 1,2,\ldots,M$, and $k = 1,2,\ldots,K$

$$q'_{mh}(\underline{N}) = \gamma_{mh} N_h$$

$$q'_{mh}(\underline{N}-\underline{1}_k) = \begin{cases} q'_{mh}(\underline{N}) & \text{if } h \neq k \\ q'_{mh}(\underline{N}) - \gamma_{mh} & \text{if } h = k \end{cases}$$

for $m = 1,2,\ldots,M$, $h = 1,2,\ldots,K$, and $k = 1,2,\ldots,K$.

Step 2: Calculate approximate mean delay of chain k at server m and approximate throughput of chain k from the following MVA formulas:

$$D_{mk}(\underline{N}) = \begin{cases} \tau_{mk}(1 + \sum_{h=1}^{K} q'_{mh}(\underline{N}-\underline{1}_k)) & \text{if } m \text{ is a fixed-rate server} \\ \tau_{mk} & \text{if } m \text{ is a delay server} \end{cases}$$

for $m = 1,2,\ldots,M$ and $k = 1,2,\ldots,K$, and

$$T_k(\underline{N}) = \frac{N_k}{\sum_{m=1}^{M} D_{mk}(\underline{N})} \qquad \text{for } k = 1,2,\ldots,K.$$

Total throughput of the network, if needed, is equal to the summation of the chain throughputs.

PAM_BASIC requires *(3M+1)K* multiplications and divisions. The number of additions and subtractions is also *O(MK)* if the sums $\sum_{i=1}^{M} \tau_{ik}$ in Step 1 and the sums $\sum_{h=1}^{K} q'_{mh}(\underline{N}-\underline{1}_k)$ in Step 2 are computed prior to looping over $m = 1,2,\ldots,M$ and $k = 1,2,\ldots,K$. The space requirement is *O(MK)*.

Algorithm PAM_IMPROVED

The first two steps of this algorithm are the same as those of Algorithm PAM_BASIC.

Step 3: Calculate server utilizations from the following formula:

$$U_m(\underline{N}) = \sum_{k=1}^{K} \tau_{mk} T_k(\underline{N})$$

for $m = 1,2,\ldots,M$, where $U_m(\underline{N})$ is the utilization of server m at population vector $\underline{N}$.

Step 4: Find the largest utilization S_k among the fixed-rate servers visited by chain k,

$$S_k = \max_{\substack{m \text{ in} \\ \text{chain } k \text{ and} \\ m \text{ is a fixed-rate server}}} U_m(\underline{N})$$

for $k=1,2,\ldots,K$.

Step 5: (Scale down throughputs of individual chains if necessary.)
If $S_k > 1$ then $T_k(\underline{N}) := T_k(\underline{N}) / S_k$.

Step 6: Calculate total throughput, and recalculate server utilizations if the throughput of any chain has been scaled down,

$$T(\underline{N}) = \sum_{k=1}^{K} T_k(\underline{N})$$

and

$$U_m(\underline{N}) = \sum_{k=1}^{K} \tau_{mk} T_k(\underline{N}), \qquad \text{for } m = 1,2,\ldots,M.$$

PAM_IMPROVED requires a maximum of *(2M+1)K* multiplications and divisions in Steps 3, 5, and 6 to calculate server utilizations and scale down chain throughputs, in addition to the time requirement of PAM_BASIC. Thus, PAM_IMPROVED requires a total of *(5M+2)K* multiplications and divisions. The number of additions and subtractions is *O(MK)*. The space requirement is *O(MK)*.

3. Experimental Results

We study in this section the accuracy of PAM_BASIC and PAM_IMPROVED using queueing networks that have characteristics of models of communication networks. Errors in the approximate chain throughputs, end-to-end delays, and server utilizations predicted by the PAM algorithms are obtained by comparing them with exact results calculated by TCA.

A communication network is specified by a set of nodes interconnected by full-duplex communication links. In our queueing network model, each link is modeled by two fixed-rate servers, one for each direction of the link; nodes are not modeled. Each flow-controlled virtual channel is modeled by a closed chain; the chain population corresponds to the flow-control window size [7, 9, 10, 12]. The route of a virtual channel is specified by the sequence of nodes it visits, which uniquely determines the sequence of fixed-rate servers visited by the corresponding chain. An additional fixed-rate server is inserted between the destination node and the source node of the route to form a closed chain. This is referred to as the *source server* and its service rate represents the packet generation rate of the user process that provides input to the virtual channel.

We examine the accuracy of the PAM algorithms by first studying a network example. We then present statistical results from a 100-network experiment. (The networks considered in this section have fixed-rate servers only.) We also examine carefully a correlation between the approximation errors for a chain and the maximum server utilization among those servers visited by the chain. Understanding such a correlation is important in applying PAM algorithms to communication network design problems because most network optimization algorithms are concerned with rerouting or reconfiguring a network to alleviate congestion at bottlenecks.

Network example

The network has 12 nodes, 15 links, and 20 chains. The set of links specified by node pairs and their capacities are shown in Table 1. The average packet length is 240 bits. The mean service time at the source server of each chain is 0.3 second. The populations and routes for all chains are shown in Table 2.

Link	Capacity (bits/second)
(1, 4)	1200
(8, 10)	9600
(1, 11)	9600
(8, 5)	4800
(1, 5)	2400
(2, 6)	2400
(3, 10)	1200
(7, 8)	4800
(9, 11)	2400
(12, 2)	4800
(1, 8)	1200
(3, 6)	9600
(4, 11)	9600
(7, 10)	1200
(9, 12)	1200

Table 1. Links and their capacities in the network example.

Chain	Population	Route (in node sequence)
1	2	5 8 7
2	3	6 2
3	2	6 3 10
4	3	6 2 12 9 11 4
5	4	9 12 2
6	3	8 5 1 11
7	2	6 3 10 8 1
8	2	9 11 1 8
9	2	7 8 5
10	2	10 8 1 11 4
11	3	12 2
12	3	12 2 6
13	2	1 11 9 12 2
14	2	6 3 10 8 7
15	3	8 5 1 11
16	3	9 11 1 5 8 10 3 6
17	2	8 10
18	2	2 6 3 10 8 7
19	2	8 10
20	3	6 2 12 9 11 4

Table 2. Populations and routes of chains in the network example.

The throughputs calculated by TCA, PAM_BASIC, and PAM_IMPROVED, and percentage errors of the two approximation algorithms, relative to TCA's exact results, are listed in Table 3. Note that the throughputs calculated by PAM_IMPROVED and PAM_BASIC are the same for those chains that do not visit a "bottleneck" queue, i.e., those that do not require execution of Step 5 in PAM_IMPROVED. For those chains that do visit bottleneck queues, PAM_IMPROVED has smaller approximation errors than PAM_BASIC. The throughputs of individual chains calculated by PAM_BASIC and PAM_IMPROVED are also plotted together with exact values in Figures 1 and 2 respectively. The server loads are highly unbalanced. The maximum server utilization, calculated by TCA, is 0.990. The average utilization over servers with nonzero utilizations is 0.398.

Chain	TCA	PAM_BASIC		PAM_IMPROVED	
	Value	Value	Error(%)	Value	Error(%)
1	3.0946	3.021	2	3.021	2
2	3.0995	3.069	1	3.069	1
3	1.3954	1.764	26	1.437	3
4	1.8674	1.923	3	1.923	3
5	2.7153	2.784	3	2.784	3
6	3.0215	2.974	2	2.974	2
7	1.0678	1.307	22	1.065	0
8	2.0149	2.07	3	2.070	3
9	3.0655	2.989	2	2.989	2
10	2.2768	2.148	6	2.148	6
11	3.2901	3.258	1	3.258	1
12	3.1181	3.058	2	3.058	2
13	1.6333	1.607	2	1.607	2
14	1.3055	1.63	25	1.327	2
15	3.0215	2.974	2	2.974	2
16	2.197	2.254	3	2.254	3
17	3.3054	3.279	1	3.279	1
18	1.1794	1.431	21	1.166	1
19	3.3054	3.279	1	3.279	1
20	1.8674	1.923	3	1.923	3
Total throughput	47.8417	48.744	2	47.606	0

Table 3. Exact throughputs, approximate throughputs, and approximation errors for the network example.

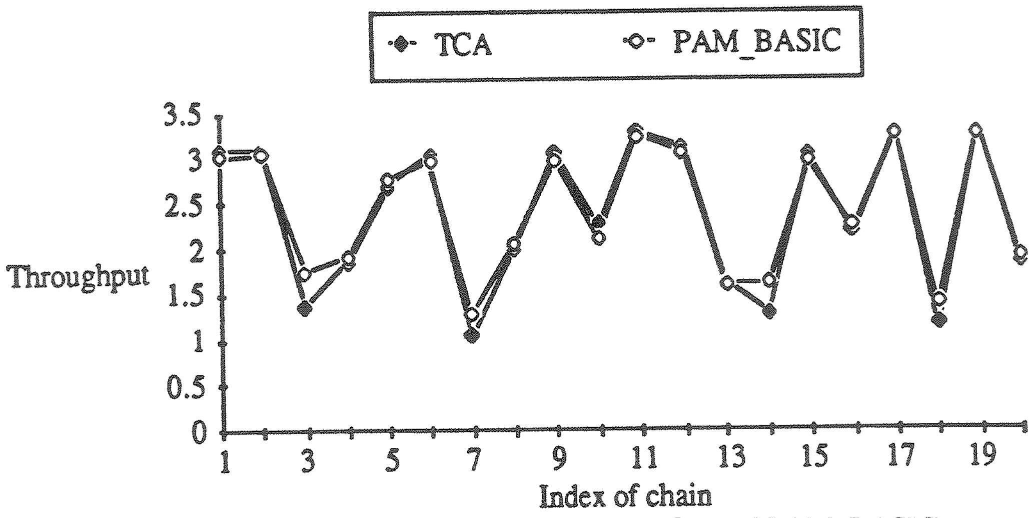

Figure 1. Throughputs of chains calculated by TCA and PAM_BASIC.

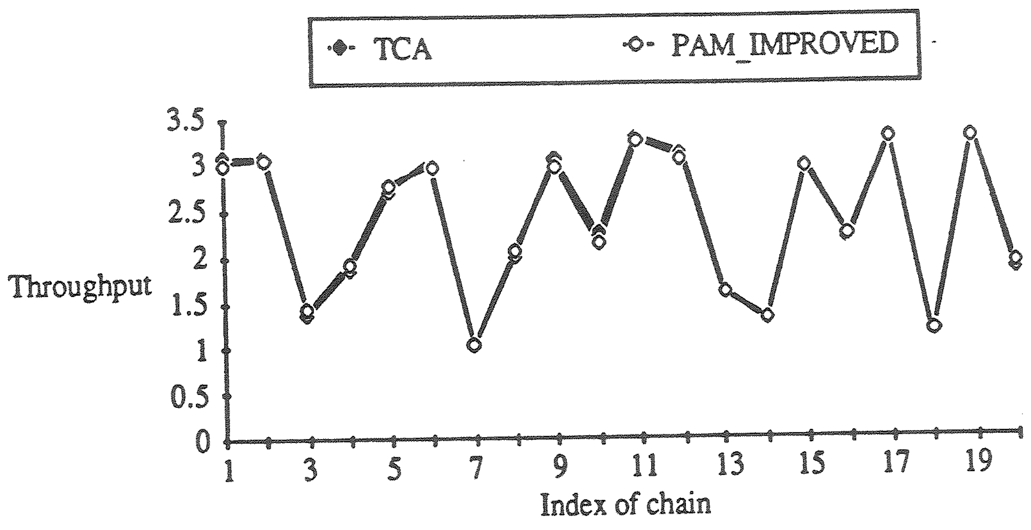

Figure 2. Throughputs of chains calculated by TCA and PAM_IMPROVED.

Both PAM and methods for calculating throughput bounds [4, 6] are based upon estimating mean queue lengths. To get throughput bounds, however, it is necessary to assume the best and the worst distributions of mean queue lengths in order to get upper bounds and lower bounds. Figure 3 shows that chain throughputs calculated by PAM_IMPROVED are much closer to exact values than the upper and lower bounds presented in [6]. Notice that the exact throughputs, calculated by TCA, of chains 3, 7, 14, and 18 are closer to their lower bounds than their upper bounds because bottlenecks are present in their routes (as was previously observed in [6]).

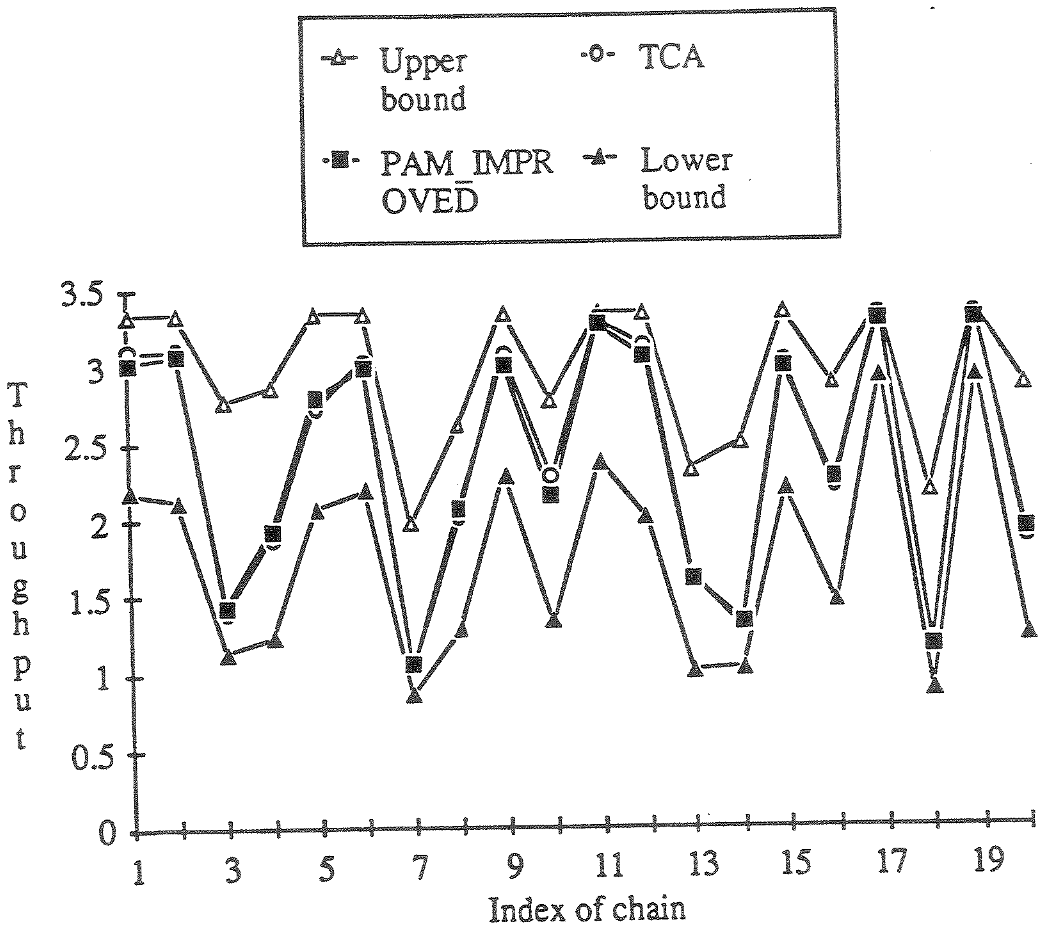

Figure 3. Approximate values, exact values, and upper and lower bounds of chain throughputs.

The approximation errors of PAM_BASIC for chains 3, 7, 14, and 18 are significantly larger than errors for other chains. Table 2 shows that those four chains are the ones that visit the server (communication channel) from node 3 to node 10. This server has the maximum utilization, calculated by TCA to be 0.990, in the network. An explanation of these large errors is the following: Proportional approximation, as the basis for throughput upper bounds in [6] and PAM algorithms herein, under-estimates the mean queue lengths of a chain at servers that are highly utilized relative to other servers visited by the chain. While PAM_BASIC has large errors for such chains, Table 3 and Figure 2 show that PAM_IMPROVED do not have large errors in its throughputs for chains 3, 7, 14, and 18, suggesting that its throughput scaling operation in Step 5 is quite effective.

Despite the relatively large errors of PAM_BASIC in predicting the throughputs of chains visiting a bottleneck sever, it is still quite useful in network design algorithms. For example, it is used in the optimal routing algorithm presented in [9] because it is the fastest of the PAM algorithms. Also, in choosing the best route for a chain among various candidates, a route that visits a highly utilized server will not likely be chosen; thus the accuracy of PAM_BASIC's throughput prediction for such a route is irrelevant.

Most network optimization algorithms are concerned, in one way or another, with the alleviation of traffic at highly utilized servers. In such applications, the accuracy of PAM_BASIC at an intermediate step of the optimization algorithm is not very important. For example, in designing the link capacities of a network, identifying the communication channel from node 3 to node 10 to be a bottleneck will probably lead to an increase in its capacity. Suppose the communication channel capacity is increased from 1200 bits/second to 4800 bits/second and PAM_BASIC is again applied to calculate chain throughputs. Figure 4 shows the approximation errors of PAM_BASIC for the two cases of the network example, where "No bottleneck" denotes the network with a 4800 bits/second communication channel from node 3 to node 10. Note that after such a capacity increase, approximation errors of PAM_BASIC for all chains fall below 5%. The largest server utilization in the so-called "No bottleneck" network is 0.870, which is still quite high. The correlation between errors of PAM_BASIC and maximum server utilizations is illustrated again in Figure 11 for the 100 randomly generated networks to be presented below.

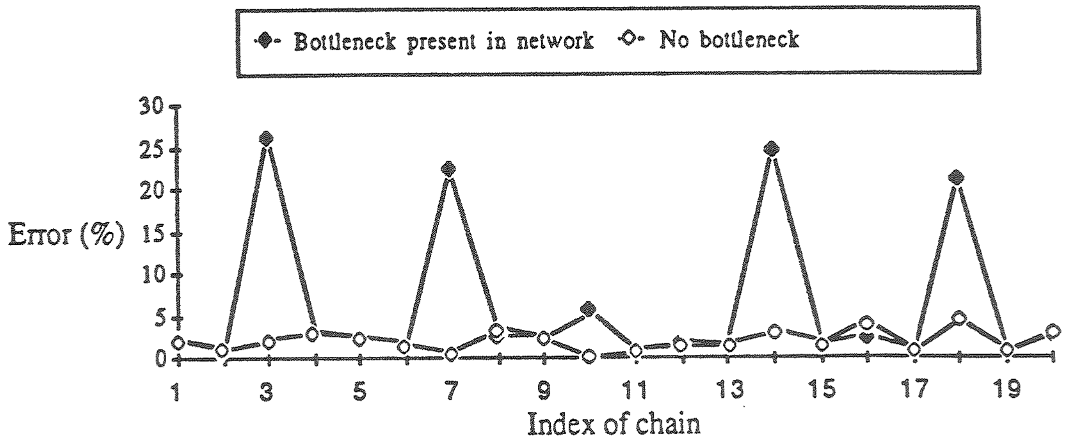

Figure 4. Approximation errors of PAM_BASIC with and without a bottleneck in the network example.

Figure 5 shows the approximation errors of PAM_IMPROVED for the same two cases of the network example. There appear to be no significant differences between the two cases.

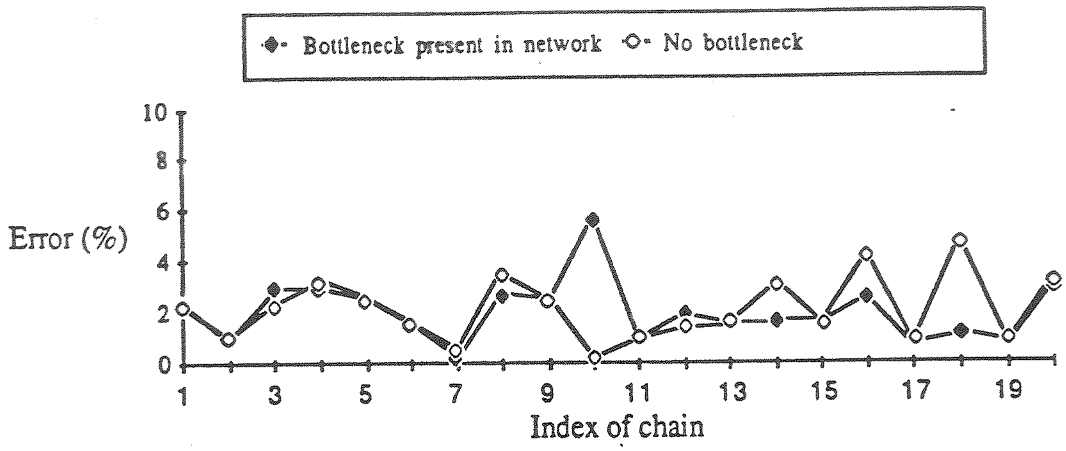

Figure 5. Approximation errors of PAM_IMPROVED with and without a bottleneck in the network example.

Source servers in the network are visited by only one chain. Figure 6 illustrates that if the mean service time at each source server in the network is increased from 0.3 to 0.5 second, the errors of PAM_BASIC become smaller. This behavior suggests that the accuracy of PAM_BASIC is less affected by a high utilization at source servers where there is no interaction between chains. In the context of a communication network, there is a more intuitive explanation. I.e., source servers model the generation of packets for input to a virtual channel. Increasing the mean service time from 0.3 to 0.5 second models a decrease in the packet generation rate.

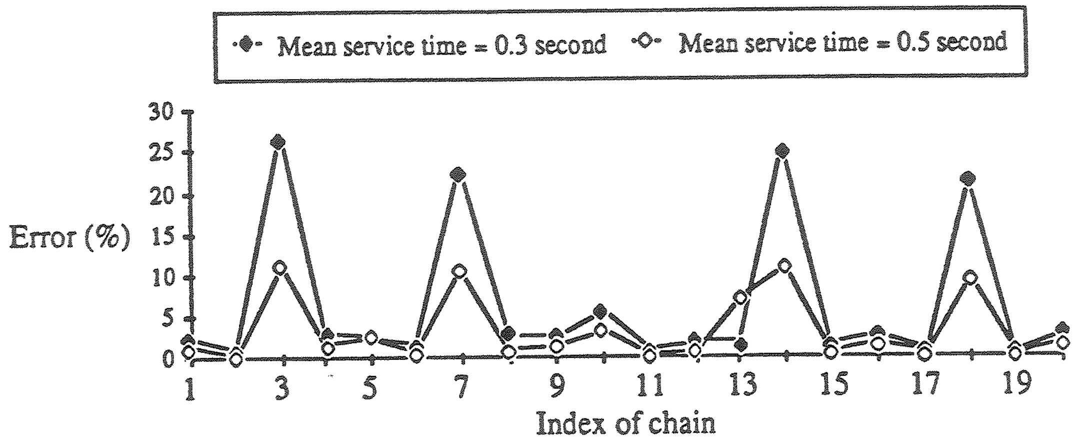

Figure 6. Approximation errors of PAM_BASIC with different mean service times at source servers in the network example.

Figure 7 shows the approximation errors of PAM_IMPROVED for the same two cases. The accuracy of PAM_IMPROVED appears to be insensitive to the change in the mean service time of source servers.

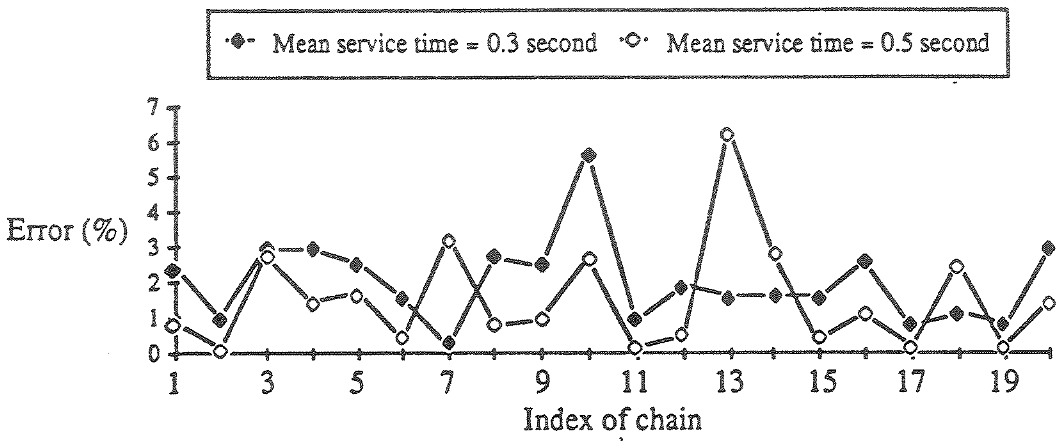

Figure 7. Approximation errors of PAM_IMPROVED with different mean service times at source servers in the network example.

Statistical results from 100 networks

We next present some statistical results from applying PAM_IMPROVED and TCA to 100 models of communication networks generated randomly as described in Appendix A. Statistics of parameters of the 100 networks generated are summarized in Table 4, where #packets is the summation of all chain populations in a network, and the average queue utilization is computed over those servers with nonzero utilizations.

	Max.	Min.	Ave.
1. #nodes	25	7	15
2. #links	37	9	20
3. #queues	74	18	40
4. #chains	43	8	23
5. #packets	107	19	57
6. Ave. queue util.	0.546	0.291	0.419

Table 4. Statistics of 100 test networks.

Percentage errors of total throughputs, chain throughputs, and server utilizations of PAM_IMPROVED, relative to TCA exact results, are shown in Figures 8, 9, and 10. The maximum percentage error in the total throughput calculated by PAM_IMPROVED is 3.55% (see Figure 8). Figures 9 and 10 show that the chain throughputs and server utilizations calculated by PAM_IMPROVED are also very accurate with only a small number of exceptions. The maxima, means, and variances of the percentage errors of PAM_IMPROVED for the three performance measures are shown in Table 5.

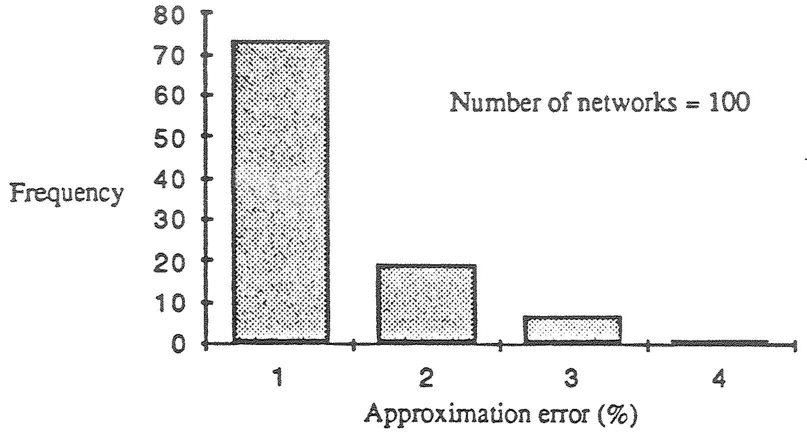

Figure 8. Distribution of approximation errors of total throughputs calculated by PAM_IMPROVED for the 100 networks.

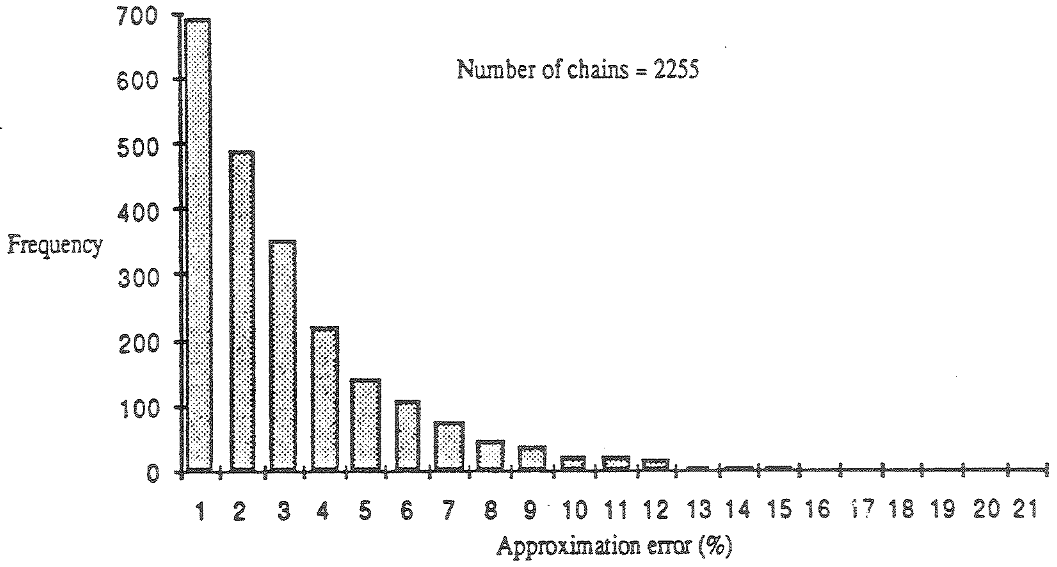

Figure 9. Distribution of approximation errors of throughputs of individual chains calculated by PAM_IMPROVED for the 100 networks.

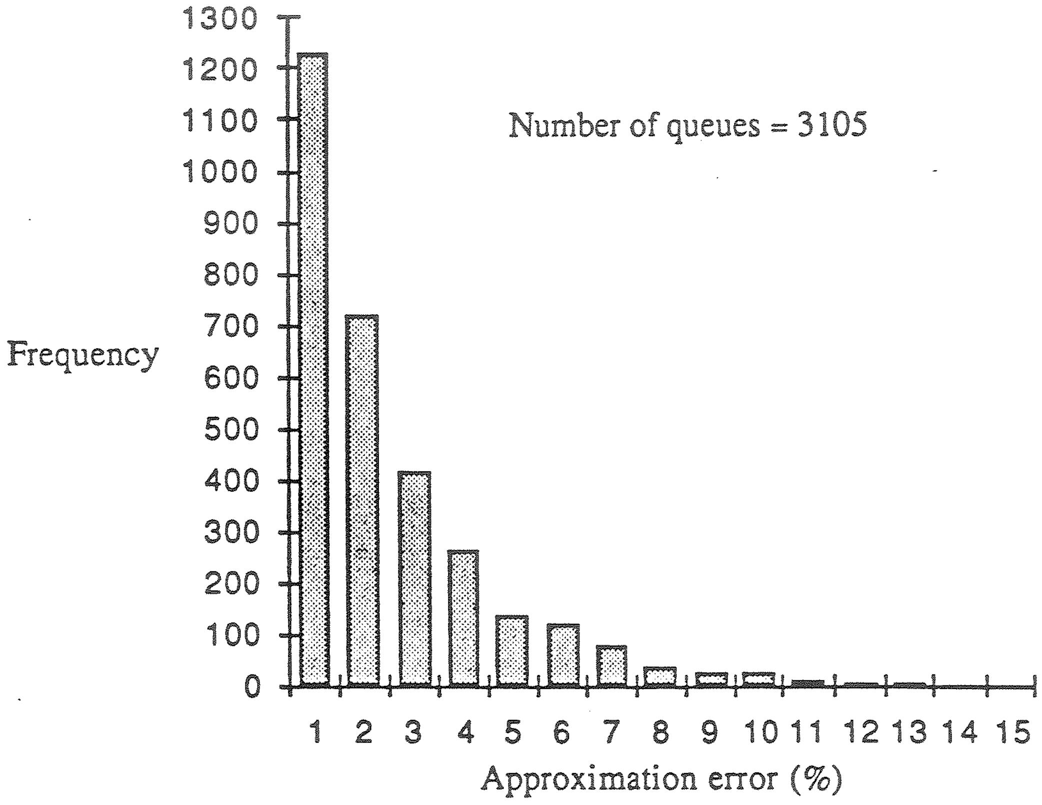

Figure 10. Distribution of approximation errors of server utilizations calculated by PAM_IMPROVED for the 100 networks.

	Statistics of percentage errors				
	#samples	**Minimum**	**Maximum**	**Mean**	**Variance**
Total throughput	100	0.009	3.554	0.824	0.002
Throughput of chain	2255	0.000	20.503	2.667	7.167
Utilization of queue	3104	0.000	14.769	2.064	4.263

Table 5. Statistics of approximation errors of total throughput, chain throughputs, and server utilizations calculated by PAM_IMPROVED for the 100 networks.

Figure 11 shows a strong correlation between the approximation error of PAM_BASIC in predicting a chain's throughput and the maximum utilization among servers visited by the chain. The points plotted in Figure 11 were obtained by selecting one chain from each of the 100 randomly generated networks and applying PAM_BASIC and TCA to calculate its approximate and exact throughputs. Note that

PAM_BASIC is quite accurate if the maximum utilization is less than about 0.9.

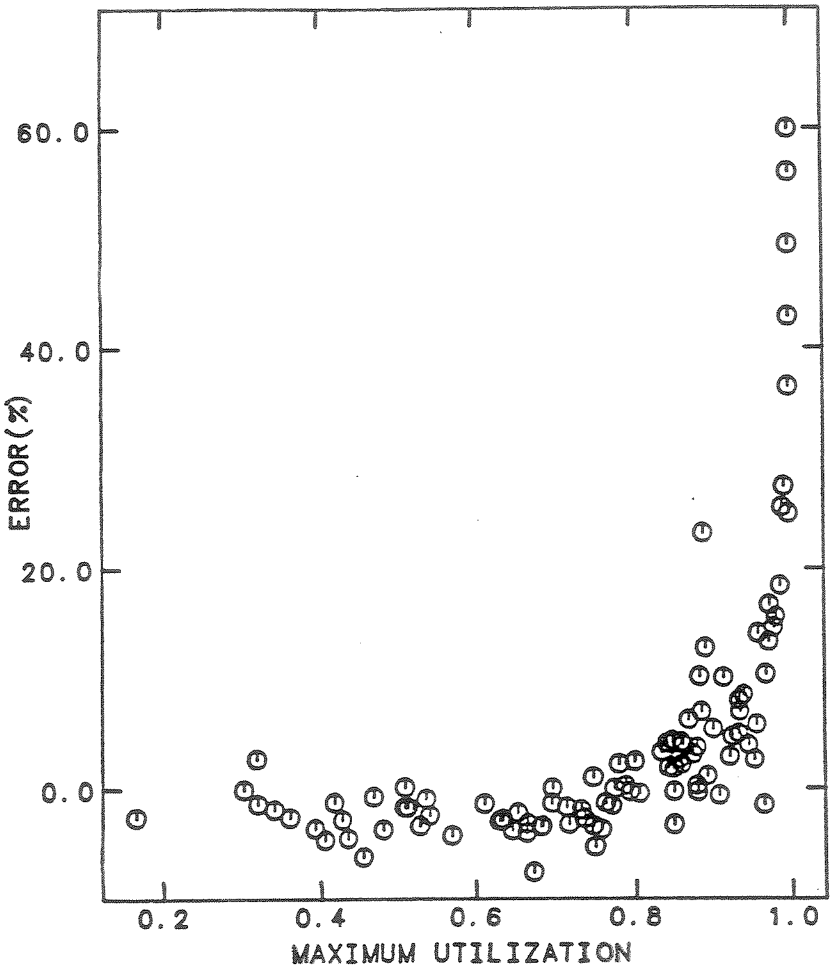

Figure 11. Correlation between percentage error of chain throughput calculated by PAM_BASIC and the maximum utilization of the servers visited by the chain.

4. PAM Algorithm with $O(MK^2)$ Computation Time

A simple approach to improve the accuracy of the PAM algorithms presented in the previous section, without resorting to an iterative solution, is to execute the last two steps of the MVA recursion instead of just the last step, again using the proportional approximation to get initial mean queue length estimates. Our algorithm, presented below, will be referred to as PAM_TWO. Such an approach to trade computation time for accuracy is not unlike the approach of Linearizer which improves the accuracy of Schweitzer's algorithm [3, 14] and the approach of bound hierarchies [4, 6].

Algorithm PAM_TWO

Step 1: Calculate proportional approximations of mean queue lengths from

$$\gamma_{mk} = \tau_{mk} / \sum_{i=1}^{M} \tau_{ik} \qquad \text{for } m = 1,2,\ldots,M, \text{ and } k = 1,2,\ldots,K$$

$$q'_{ml}(\underline{N}) = \gamma_{ml} N_l$$

$$q'_{ml}(\underline{N}-\underline{1}_k-\underline{1}_h)= \begin{cases} q'_{ml}(\underline{N}) & \text{if } l \neq k \text{ and } l \neq h \\ q'_{ml}(\underline{N})-\gamma_{ml} & \text{if } (l=k \text{ or } l=h) \text{ and } h \neq k \\ q'_{ml}(\underline{N})-2\gamma_{ml} & \text{if } l=k=h \end{cases}$$

for $m = 1,2,\ldots,M$, $l = 1,2,\ldots,K$, $h = 1,2,\ldots,K$, and $k = 1,2,\ldots,K$. Note that $q'_{ml}(\underline{N}-\underline{1}_k-\underline{1}_h)$ may have a negative value if $N_l<2$. In this case, $q'_{ml}(\underline{N}-\underline{1}_k-\underline{1}_h)$ is assigned the value of zero.

Step 2: For $k = 1,2,\ldots,K$, repeat Step 2.1 and Step 2.2.

Step 2.1: For $h=1,2,\ldots,K$, repeat the following calculations:

$$D_{mh}(\underline{N}-\underline{1}_k)= \begin{cases} \tau_{mh}(1 + \sum_{l=1}^{K} q'_{ml}(\underline{N}-\underline{1}_k-\underline{1}_h)) & \text{if } m \text{ is a fixed-rate server} \\ \tau_{mh} & \text{if } m \text{ is a delay server} \end{cases}$$

for $m = 1,2,\ldots,M$,

$$T_h(\underline{N}-\underline{1}_k)= \begin{cases} \dfrac{N_h}{\sum_{m=1}^{M} D_{mh}(\underline{N}-\underline{1}_k)} & \text{if } h \neq k \\ \dfrac{N_h - 1}{\sum_{m=1}^{M} D_{mh}(\underline{N}-\underline{1}_k)} & \text{if } h = k, \end{cases}$$

$$q'_{mh}(\underline{N}-\underline{1}_k) = D_{mh}(\underline{N}-\underline{1}_k) \times T_h(\underline{N}-\underline{1}_k)$$

for $m = 1,2,\ldots,M$.

Step 2.2: Calculate the following,

$$D_{mk}(\underline{N}) = \begin{cases} \tau_{mk}(1 + \sum_{h=1}^{K} q'_{mh}(\underline{N}-\underline{1}_k)) & \text{if } m \text{ is a fixed-rate server} \\ \\ \tau_{mk} & \text{if } m \text{ is a delay server} \end{cases}$$

for $m = 1,2,\ldots,M$, and

$$T_k(\underline{N}) = \frac{N_k}{\sum_{m=1}^{M} D_{mk}(\underline{N})}.$$

Step 3: Execute Steps 3-6 of PAM_IMPROVED.

Step 1 of PAM_TWO requires $2MK$ multiplications and divisions. Step 2 of PAM_TWO requires $2MK^2 + (M+3)K$ multiplications and divisions. Step 3 of PAM_TWO requires $(2M+1)K$ multiplications and divisions. Thus, PAM_TWO requires a total of $2MK^2 + (5M+4)K$ multiplications and divisions. Note that the number of additions and subtractions is $O(MK^2)$ if the sums $\sum_{l=1}^{K} q'_{ml}(\underline{N}-\underline{1}_k-\underline{1}_h)$ needed in Step 2.1 are computed from

$$\sum_{l=1}^{K} q'_{ml}(\underline{N}-\underline{1}_k-\underline{1}_h)=\sum_{l=1}^{K} q'_{ml}(\underline{N})-\gamma_h-\gamma_k$$

where the sums $\sum_{l=1}^{K} q'_{ml}(\underline{N})$ are computed prior to looping over $k = 1,2,\ldots,K$ and $h = 1,2,\ldots,K$ in Step 2.

The space requirement of PAM_TWO is $O(MK)$. Note that arrays of size MK are adequate for the execution of Step 2; they are used to hold values of $\{\tau_{mk}\}$ and to hold values of mean queue lengths in Steps 2.1 and 2.2.

5. More Experimental Results

The 100 networks used in Section 3 have characteristics of communication networks: large number of chains with sparse routes and small chain populations. In this section, we examine the accuracy of PAM_TWO and PAM_IMPROVED using 500 networks generated randomly according to the specification used in the study of iterative AMVA algorithms by Zahorjan et al. [20, Table 2]. (There is one difference: We set the minimum number of servers at 5 instead of 2.) The network generation parameters are shown in Table 6, where U indicates the uniform distribution. Networks generated by these parameters possess characteristics of computer systems rather than communication networks. Note that every network generated has four chains and each chain visits every server in the network. Compared to the 100 networks in Section 3, these 500 networks have larger chain populations and much larger service time variations; there are also delay servers in addition to fixed-rate servers.

Scheduling Discipline	Prob[Infinite Server] = 0.05 Prob[Load Independent] = 0.95
Population Size	Class 1 : U(1, 10) Class 2 : U(1, 5) Class 3 : U(1, 5) Class 4 : U(1, 5)
Loadings	U(0.1, 50.0)
Number of Centers	U(5, 50)

Table 6. Network generation parameters.

We used PAM_TWO and PAM_IMPROVED to calculate approximate chain throughputs and the MVA algorithm to calculate exact chain throughputs for the 500 networks generated. The average and maximum percentage errors in the approximate chain throughputs, relative to exact MVA solutions, are shown in Table 7 for PAM_IMPROVED and PAM_TWO. Both PAM algorithms have small average errors but fairly large maximum errors. Table 7 also shows that PAM_TWO is more accurate than PAM_IMPROVED.

Technique	Measure	Error(%)	#Networks
PAM_IMPROVED	Average Maximum	2.3 40.3	500
PAM_TWO	Average Maximum	0.8 30.8	500

Table 7. Relative errors in chain throughputs calculated by PAM_IMPROVED and PAM_TWO for 500 networks.

Although the maximum percentage errors are large, Figure 12 shows that only a very small fraction of the chains have percentage errors larger than 10% while more than three-fourths of the chains have less than 1% error for PAM_TWO and less than 3% error for PAM_IMPROVED.

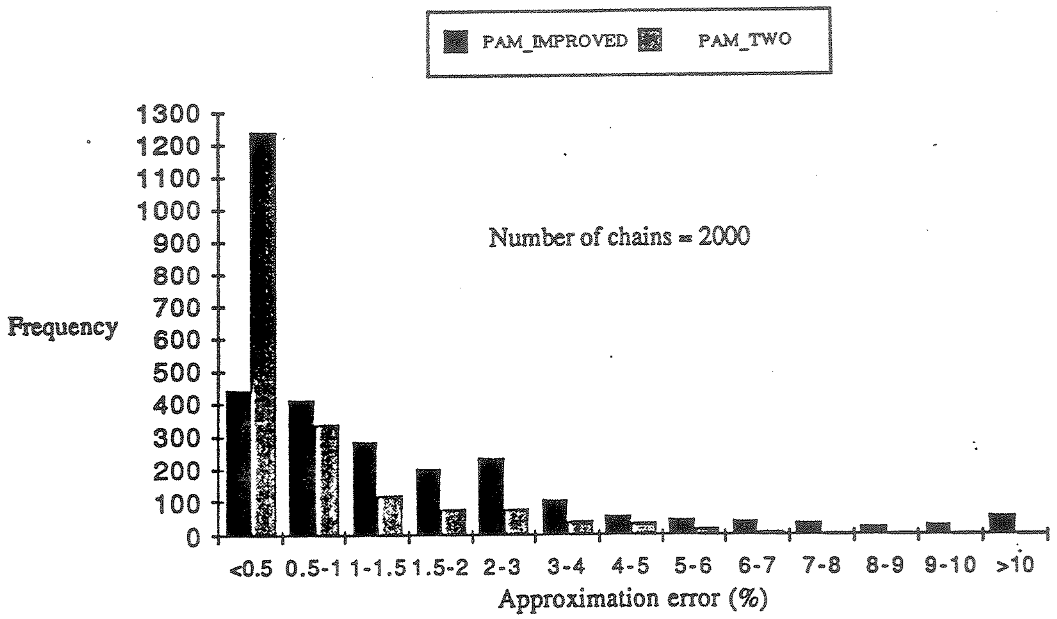

Figure 12. Distribution of approximation errors of throughputs of individual chains calculated by PAM_IMPROVED and PAM_TWO for 500 networks.

In Section 3, we found a strong correlation between the approximation error of a chain's throughput calculated by PAM_BASIC and the maximum utilization among servers visited by the chain. Table 8 shows that such a correlation also exists for PAM_IMPROVED and PAM_TWO. For those chains which do not visit servers with utilizations between 0.95 and 1, both the average and maximum approximation errors become substantially smaller for both PAM algorithms (see row 2 of Table 8).

Max. util. ≤	Approximation errors of chain throughputs					
	PAM_IMPROVED			PAM_TWO		
	Max.	Ave.	#Chains	Max.	Ave.	#Chains
1.0	40.3	2.3	2000	30.8	0.8	2000
0.95	11.4	1.6	1784	9.6	0.5	1860
0.9	9.4	1.5	1732	5.4	0.5	1784
0.8	8.4	1.2	1556	3.1	0.3	1612
0.7	4.8	1.0	1392	1.7	0.2	1416
0.6	3.8	0.8	1136	1.4	0.2	1164
0.5	2.1	0.6	744	1.4	0.1	788

Table 8. Correlation between percentage errors in approximate chain throughputs and maximum server utilizations for PAM_IMPROVED and PAM_TWO.

Although we generated the 500 networks similar to what Zahorjan et al. did in their study of three iterative AMVA algorithms, a direct comparison of the accuracy of PAM and the iterative AMVA algorithms cannot be made. There are two reasons. First, we evaluated approximation errors in chain throughputs and server utilizations (we are also interested in mean end-to-end delays of virtual channels as a performance measure but these can be obtained from chain throughputs and Little's formula). The iterative AMVA algorithms were evaluated by the maximum approximation error in mean queue lengths $q_{mk}(\underline{N})$ in [20] and also mean delays $D_{mk}(\underline{N})$ and server utilizations $U_{mk}(\underline{N})$ in [3] for all m and k. The performance measures of AMVA have finer granularity than ours. On the other hand, the AMVA studies used a specially defined measure of error called "tolerance error" instead of the usual relative error which we use. The tolerance error used by Zahorjan et al. [20] is the following:

$$\max_{m,k} \; [q_{mk}(\underline{N}) - q_{mk}{}^{*}(\underline{N})] \, / \, N_k$$

where $q_{mk}(\underline{N})$ is the approximate value and $q_{mk}{}^{*}(\underline{N})$ is the exact value. Note that N_k is used in the denominator instead of $q_{mk}{}^{*}(\underline{N})$. It was argued that tolerance error was used in place of relative error because the latter measure is very sensitive to small (absolute) errors in small values. (That is exactly how we got most of the large percentage errors for PAM algorithms since we use relative error as our measure.)

The tolerance errors in mean queue lengths of the three iterative AMVA algorithms from the last column of Table 4 in [20] are reproduced in Table 9. However, it is not possible to make a direct comparison between the relative errors in Table 7 for PAM and the tolerance errors in Table 9 for AMVA. On the one hand, for the same numerical values of approximate and exact solutions, tolerance errors are much smaller than relative errors. On the other hand, $q_{mk}(\underline{N})$ is a measure having finer granularity than chain throughput (which is obtained from a summation of mean delays) and is expected to have larger approximation errors.

Technique	Measure	Tolerance Error (%)	#Networks
Schweitzer's method	Average Maximum	2.5 30.5	2000
Linearizer	Average Maximum	0.5 2.4	2000
AQL	Average Maximum	0.3 3.3	2000

Table 9. Tolerance errors in mean queue lengths calculated by 3 iterative AMVA algorithms from Zahorjan et al. [20].

To carry out a direct comparison of AMVA and PAM, we will have to implement the AMVA algorithms (which we have not done). Moreover, the comparison can only be made for networks with a small number of closed chains, which is the current domain of applications of iterative AMVA algorithms. For such networks, we conjecture that Linearizer and AQL are more accurate than PAM algorithms (at the expense of more computation time for both Linearizer and AQL and more space for Linearizer).

For networks with a large number of closed chains, however, the accuracy and convergence behavior of iterative AMVA algorithms are still unknown. For this class of networks, PAM is the only approximate solution method whose accuracy has been studied.

A summary of the computational time and space requirements of PAM algorithms and the three iterative AMVA algorithms is shown in Table 10. For Linearizer, we have indicated the time requirement of the new implementation described by de Souza e Silva and Muntz [16] instead of that of Chandy and Neuse [3]. The other time and space requirements of the iterative AMVA algorithms are according to Zahorjan et al. [20]. Note that the AMVA algorithms' time requirements are usually stated for a single iteration, which must be multiplied by the number of iterations that are needed for convergence to a solution.

Algorithm	**Time Requirement**	**Space Requirement**
Schweitzer	$O(MK)$ [# iterations]	$O(MK)$
Linearizer	$O(MK^2)$ [# iterations]	$O(MK^2)$
AQL	$O(MK^2)$ [# iterations]	$O(MK)$
PAM_BASIC	$O(MK)$	$O(MK)$
PAM_IMPROVED	$O(MK)$	$O(MK)$
PAM_TWO	$O(MK^2)$	$O(MK)$

Table 10. Summary of time and space requirements of iterative AMVA algorithms and PAM algorithms.

6. Conclusions

PAM algorithms have been designed for the approximate solution of queueing networks with a large number of closed chains and relatively small chain population sizes. Because they are noniterative, they are suitable for many communication network design and optimization problems that are typically based upon a heuristic search for an optimum; a very fast evaluation of network performance is needed at each of step of such a heuristic search. (When a network is slightly modified, the array of approximate mean queue lengths in a PAM implementation does not have to be completely recalculated; only those elements of the array corresponding to the network change have to be recomputed.) PAM algorithms provide chain throughputs, mean end-to-end delays, and server utilizations that have adequate accuracy for such purposes. For example, PAM_BASIC, the fastest of the PAM algorithms was used in [9] for optimal routing. In choosing a route for a new virtual channel, routes that visit a congested communication channel are rarely chosen. Thus, the relatively large approximation errors of PAM_BASIC for such routes do not affect its effectiveness in the optimal routing algorithm. We also found in one example that ranking candidate routes by approximate chain throughputs, calculated by PAM_BASIC, and by exact chain throughputs, calculated by TCA, had the same result for the top several candidates.

PAM algorithms can be applied to the analysis and design of other distributed systems with a large number of closed chains if the network performance measures needed to evaluate these systems are the ones that have been tested for the PAM algorithms. The iterative AMVA algorithms provide performance measures of finer granularity, i.e., mean queue lengths, mean delays and utilizations for server *m* and class *k*, which are not calculated by the three PAM algorithms in this paper.

Although we have tested the accuracy of PAM algorithms extensively, it is not possible to conclude that they have been tested for all possible combinations of network parameters and all interesting regions of the parameter space. We did test them for networks with a large number of chains, i.e., models of communication networks that can be solved exactly by TCA. Iterative AMVA algorithms were tested on networks with just a few chains, i.e., ones that can be solved exactly by MVA. It is also an impossibility to conclude that the accuracy of PAM algorithms demonstrated in this report will scale up to networks with hundreds or thousands of closed chains (which are required to model communication networks of the present and future). Both exact solutions and discrete-event simulations of such large networks are not currently feasible. We conjecture that the accuracy, in terms of average errors, will scale up. In fact, average errors will likely be smaller due to averaging of estimates that are too high with ones that are too low.

We also studied a few networks which have many delay servers where mean service times are much larger than the mean service times of fixed-rate servers. For such networks, PAM algorithms are extremely accurate. (Note that the distribution of a chain's population proportional to loads is an exact operation in a network consisting of delay servers only.) We also showed in Section 3 that PAM algorithms are accurate for networks with fixed-rate servers only. We found, from a few examples, that the accuracy of PAM algorithms is slightly worse (than what is shown in this report) for networks in which the delay servers and the fixed-rate servers visited by each chain have about the same aggregate load. We added such a delay server to each chain in the network example of Table 3. The maximum error of PAM_IMPROVED went up to 8.5% from 6% in Table 3.

Appendix A. Network Generator [7]

Of the 100 networks generated, 70 are SMALL networks and the rest are LARGE networks. End-to-end acknowledgment delays are not modeled. The average packet length is 240 bits.

Nodes

The network generator creates two types of networks (SMALL and LARGE), distinguished by the number of nodes and the number of chains. The number of nodes in a SMALL network ranges from 6 to 25. The probability distribution has the shape of a triangle with a mean of 16. The number of nodes in a LARGE network ranges from 11 to 30 with a mean of 21. Each node is characterized by two parameters:

(i) Location, in x and y coordinates, to be used for calculating the distance between two nodes.

(ii) Minimum number of links connected to the node, designated as $L[i]$ for node i. This number is randomly selected from 2, 3, and 4 with a mean of 2.5.

Links

The number of links in the network is not directly sampled from a random variable and is not known until the network is created. Factors that determine the number of links in a network are the number of nodes and $L[i]$, for all i. The topology of each network is determined in a manner similar to the method given by Steiglitz et al. [17]. The procedure is briefly described in the following:

Step 1: Select a node i such that $L[i] = \max_j L[j]$.
If $L[i] \leq 0$ then stop.

Step 2: Put a link between node i and node n where node n is the nearest neighbor of node i that satisfies the following conditions:
(i) node n is not connected to node i.
(ii) $L[n] > 0$.
Condition (ii) is ignored if none of the nodes satisfy this condition. In this case, the node nearest to node i satisfying (i) is selected. Let $L[i] := i - 1$ and $L[n] := L[n] - 1$. Go to Step 1.

The capacity of a link is selected from 1200, 2400, 4800, and 9600 bits per second with equal probabilities.

Chains

The number of chains is selected to be 1 or 2 times the number of nodes in the network with equal probabilities. The source node of each chain is selected from all nodes with equal probabilities. The length (number of servers) of the chain is selected with the following probabilities: p_1=0.1, p_2=0.2, p_3=0.3, p_4=0.2, p_5=0.1, p_6=0.05, p_7=0.03, p_8=0.01, and p_9=0.01, where p_i is the probability of having i servers in the chain. The length ranges from 1 to 9 with a mean of 3.38. To establish a route, the chain extends iteratively from the last node (in the partial route) to one of its adjacent nodes until the designated length is reached or the chain cannot be further extended without forming a loop. In each iteration, all adjacent nodes that are not already in the route are selected with equal probabilities. The mean service time at the source server is selected from 0.1, 0.2, and 0.3 second with equal probabilities. The population of the chain is selected from 2, 3, and 4 with probabilities 0.6, 0.3, and 0.1 respectively.

Connectivity

Each network generated is manually checked for connectivity. Networks with disconnected components are discarded. Others are accepted as they are, or accepted after a minor modification (e.g., add a link).

Lastly, those networks that cannot be (effectively) handled by the tree convolution algorithm are also discarded.

REFERENCES

[1] Y. Bard, "Some Extensions to Multiclass Queueing Network Analysis," *Performance of Computer Systems*, M. Arato et al. (ed.), North-Holland, Amsterdam, 1979.

[2] W.-M. Chow, "Approximations for Large Scale Closed Queueing Networks," *Performance Evaluation*, Vol. 3, No. 1, Feb. 1983, pp. 1-12.

[3] K. M. Chandy and D. Neuse, "Linearizer: A Heuristic Algorithm for Queueing Network Models of Computer Systems," *Comm. ACM*, Vol. 25, No. 2, Feb. 1982, pp. 126-134.

[4] D. L. Eager and K. C. Sevcik, "Bound Hierarchies for Multiple-Class Queueing Networks," *Journal of ACM*, Vol. 33, No. 1, Jan. 1986, pp. 179-206.

[5] K. P. Hoyme, S. C. Bruell, P. V. Afshari, and R. Y. Kain, "A Tree-Structured Mean Value Analysis Algorithm," *ACM Trans. on Computer Systems*, Vol. 4, No. 2, May 1986, pp. 178-185.

[6] C.-T. Hsieh and S. S. Lam, "Two Classes of Performance Bounds for Closed Queueing Networks," *Performance Evaluation*, Vol. 7, No. 1, 1987, pp. 3-30.

[7] C.-T. Hsieh, "Models and Algorithms for the Design of Store-and-Forward Communication Networks," Ph.D. Thesis, Department of Computer Sciences, University of Texas at Austin, 1987.

[8] S. S. Lam and Y. L. Lien, "A Tree Convolution Algorithm for the Solution of Queueing Networks," *Comm. ACM*, Vol. 26, No. 3, March 1983, pp. 203-215.

[9] S. S. Lam and Ching-Tarng Hsieh, "Modeling, Analysis, and Optimal Routing of Flow-Controlled Communication Networks," Technical Report TR-87-24, Dept. of Computer Sciences, Univ. of Texas at Austin, June 1987.

[10] S. S. Lam and Y. L. Lien, "Modeling and Analysis of Flow-Controlled Packet Switching Networks," *Proc. 7th Data Communications Symposium*, Mexico City, October 1981.

[11] M. Reiser and H. Kobayashi, "Queueing Networks with Multiple Closed Chains: Theory and Computational Algorithms," *IBM J. Res. Develop.*, Vol. 21, 1975, pp. 283-294.

[12] M. Reiser, "A Queueing Network Analysis of Computer Communication Networks with Window Flow Control," *IEEE Trans. on Communication*, Vol. COM-27, 1979, pp. 1199-1209.

[13] M. Reiser and S. Lavenberg, "Mean Value Analysis of Closed Multichain Queueing Networks," *Journal of ACM*, Vol. 27, No. 2, April 1980, pp. 313-322.

[14] P. Schweitzer, "Approximate Analysis of Multiclass Closed Networks of Queues," *Proc. Int. Conf. Stochastic Control and Optimization*, Amsterdam, 1979.

[15] E. de Souza e Silva, S. S. Lavenberg, and R. R. Muntz, "A Clustering Approximation Technique for Queueing Network Models with a Large Number of Chains," *IEEE Trans. on Computers*, Vol. C-35, No. 5, May 1986, pp. 419-430.

[16] E. de Souza e Silva and R. R. Muntz, "A Note on the Computational Cost of the Linearizer Algorithm for Queueing Networks," Technical Report CSD-870025, Dept. of Computer Science, UCLA, June 1987.

[17] K. Steiglitz, P. Weiner, and D. J. Kleitman, "The Design of Minimum Cost Survivable Networks," *IEEE Transactions on Circuit Theory*, Vol. CT-16, 1969, pp. 455-460.

[18] S. Tucci and C. H. Sauer, "The Tree MVA Algorithm," *Performance Evaluation*, Vol. 5, No. 3, August 1985, pp. 187-196.

[19] J. Zahorjan, K. C. Sevcik, D. L. Eager, and B. Galler, "Balanced Job Bound Analysis of Queueing Networks," *Comm. ACM*, Vol. 25, No. 2, 1982, pp. 132-141.

[20] J. Zahorjan, D. L. Eager, and H. Sweillam, "Accuracy, Speed, and Convergence of Approximate Mean Value Analysis," Technical Report, Dept. of Computer Science, Univ. of Washington, August 1986; to appear in *Performance Evaluation.*