Analysis of Queueing Network Models with Population Size Constraints and Delayed Blocked Customers

Alexander Thomasian
Burroughs Corporate Center
for Performance Modeling
3519 West Warner Ave.
Santa Ana, CA 92704

Paul Bay
T.R.W. Defense Systems Group
One Space Park
Redondo Beach, CA 90278

Abstract

Queueing Network Models - QNM's with population size constraints and delayed blocked customers occur due to MultiProgramming Level - MPL constraints in computer systems and window flow-control mechanisms in Computer Communication Networks - CCN's. The computational cost of existing algorithms is unacceptable for large numbers of chains and high population sizes. A fast approximate solution technique based on load concealment is presented to solve such QNM's. The solution procedure is non-iterative in the case of fixed rate Poisson arrivals, while iteration is required in the case of quasi-random arrivals. Each iteration requires the solution of a single chain network of queues comprised of stations visited by each chain. We then present an algorithm to detect saturated chains and determine their maximum throughput. A fast solution algorithm due to Reiser for closed chains is also extended to the case of quasi-random arrivals. The accuracy of the proposed solution techniques is compared to previous techniques by applying it to a test case, reported in the literature, and a set of randomly generated examples.

1. Introduction

Queueing network models - QNM's have been used extensively in the modeling and analysis of computer systems and networks. The very low computational cost and adequate accuracy of QNM's in predicting the performance of multiprogrammed computer systems has been generally established.

A class of QNM's known as product-form or separable QNM's [2], has been of particular interest due to their low solution cost made possible by the convolution and Mean-Value Analysis (MVA) algorithms and their variations [15] (referred to hereafter as exact methods). Solution time and space requirements can be reduced substantially in QNM's when chains visit a small subset of the stations in the network (sparseness property) and/or chains are clustered in certain parts of the network (locality property) by using the Tree Convolution Algorithm - TCA [13]. In addition, fast approximate solutions have been developed for product-form networks [1, 5, 6, 15, 21, 25], when the cost of exact solutions is excessive, but the QNM does not have any special properties. A specialized iterative method was developed in [7] to analyze QNM's with large numbers of remote and local chains in the context of a distributed system.

Assumptions resulting in analytical tractability of QNM's, do not allow for representation of certain important characteristics typical of many systems [4]. In this paper we are interested in developing efficient solution methods to take into account blocking constraints representative of fixed memory size in multiprogrammed computer systems and fixed population constraints due to window flow control mechanisms in Computer Communication Networks - CCN's. In a multiprogrammed computer system (window flow controlled network) when the maximum MPL (window size) is reached, jobs (messages) are queued pending activation. This situation differs from the case with population size constraints, where blocked customers are lost [10]. Generally, solutions taking into account customer blocking are necessary when one is interested in user level performance measures (rather than the performance of a subsystem, e.g., the communication subnet).

Approximate solutions of Single Chain Population Constrained Queueing Networks - SCPCQN's are readily obtained using a combination of hierarchical decomposition and flow-equivalence techniques [15, 17]. The queueing network is substituted by a Flow-Equivalent Service Center - FESC (state-dependent exponential server). The resulting system can be represented by a single-dimensional birth-death process when the arrival process is random or quasi-random and solved accordingly. However, the application of this technique to Multiple Chain Population Constrained Queueing Networks - MCPCQN's is not as straightforward. For the MCPCQN problem the solution can be obtained by first solving the network consisting of a set of **K** chains and **M** stations (K = |**K**| and M = |**M**| are the number of chains and stations) for all possible population combinations $n = [n_1,...,n_K]$ up to the maximum population size vector $W = [W_1,...,W_K]$ using an exact method. The resulting composite queue with throughputs $T_k(n)$, $0 \leq n \leq W$ and $\forall\ k \in \mathbf{K}$, is then substituted for the subnetwork. A multi-dimensional Markov chain incorporating the arrival process to the network then need be constructed and solved for steady-state probabilities (queue-length distribution) to compute the performance measures for the system.

The drawback of the above approach is that the time and space requirements for the solution of the Markov chain (for the higher level model) grow rapidly with the number of chains in the QNM (for a detailed discussion the reader is referred to [22]), making the approach infeasible except for very small models. For this reason several approximate solution methods have been proposed for efficiently solving such models.

Brandwajn [3] and Menasce and Almeida [18] (B-MA, for short) independently developed a non-iterative approximate solution technique for solving MCPCQN's. Their approach is based on a "manifold application of equivalence and decomposition" [3]. The approximate solution technique requires the solution for throughputs $T_k(n)$ for all $0 \leq n \leq W$ and $\forall\ k \in \mathbf{K}$. This requirement alone incurs a computational cost proportional to $O(KM \prod_{k=1}^{K} [W_k+1])$, when the throughputs are obtained using a general purpose exact method.

A second approximate solution technique based on iteration was independently developed by Brandwajn [3] and Lazowska and Zahorjan [16] (B-LZ for short). The technique is similar in spirit to [3, 18] in that the multi-dimensional Markov chain representing the MCPCQN is solved "one-dimension-at-a-time". The approach differs from the first approach primarily in that it doesn't require *complete* decomposition (i.e., computation of throughputs for all possible states). Specifically, in computing the queue-length distribution for chain j, we use throughputs obtained by solving a multi-chain QNM where the effect of the other chains is represented by their mean active population, $T_j(\bar{n}_1,\bar{n}_2,...,n_j,...\bar{n}_K)$, $1 \leq n_j \leq W_j$. Note that for $n_j > W_j$ the throughput remains constant at the value for W_j. Since the mean queue-lengths for other chains are unknown a priori, an iterative technique is suggested. The iteration is started by setting all other chain populations at a reasonable value, e.g., $\min(W_k, \bar{n}_k)$, $\forall\ k \in \mathbf{K}$, where $\bar{n}_k$ is obtained ignoring population size constraints.

The computational cost of the above technique is largely dependent on the method used for obtaining the throughputs $T_j(\bar{n}_1,\bar{n}_2,...,n_j,...\bar{n}_K)$. Exact solution techniques would incur a *per-iteration* cost of $O(KM \prod_{k=1}^{K} [W_k+1])$. Variations of exact techniques have to be used when other population sizes are non-integral, resulting in increased cost [8]. Computational cost is reduced significantly by using either of two approximate techniques based on MVA. The first technique is the Bard-Schweitzer algorithm [1, 25] which has computational complexity of $O(K^2M)$, where M is the number of queues (servers) and K is the number of chains. The second is the Linearizer algorithm, an improved version of the Bard-Schweitzer algorithm developed by Chandy and Neuse [5], which gives more accurate results at slightly higher computational cost of $O(K^3M)$. Assuming that $W_k = W$, $\forall\ k \in \mathbf{K}$, per-iteration cost is then $O(K^3MW)$ and $O(K^4MW)$ when using the Bard-Schweitzer and Linearizer algorithms, respectively.

The accuracy of the B-LZ solution technique was verified in [3, 16] using simulation. It tends to be less accurate than the first technique and its accuracy depends on the underlying algorithm used, i.e., greater accuracy is obtained using the Linearizer versus the Bard-Schweitzer approximation.

While the above two methods provide substantial savings over "exact" decomposition, it is clear that for MCPCQN's consisting of a large number of chains computational cost can still be quite significant. These costs become even more significant when the maximum population constraint (e.g., MPL for high capacity transaction processing systems) for each chain is quite

large. We illustrate this point in Table 1-1. We first give the computational complexity of the algorithms discussed above, followed by the computational complexity of the three procedures we present in Sections 4 and 5. For those algorithms which are iterative, the cost given is *per iteration*. Computational costs are expressed in terms of: $K = |\mathbf{K}|$ = number of chains, $M = |\mathbf{M}|$ = number of stations, $|\mathbf{H}_k| = H_k = H$ = number of stations visited by chain k (assumed to be equal), and $W_k = W$ = population constraint of each chain (assumed to be equal), $\forall\, k \in \mathbf{K}$.

Consider a QNM consisting of 32 closed chains (i.e., virtual routes) and 64 stations (i.e., channels), similar to the network studied in [13]. However, we now assume that there are external arrivals to each chain and there is blocking due to flow control window size constraints. We first consider a QNM in which arrival processes to all chains are fixed rate (for which we propose Procedure "1"), and then consider a QNM in which arrivals to all chains are quasi-random (for which we propose Procedures "2" and "3"). Letting $H = W = 3$, gives the computational costs shown in the last column of Table 1-1. As can be seen in Table 1-1, Procedures 1, 2, and 3 achieve considerable reductions in computational cost. Note that for this particular example (with sparsity and locality properties) the application of TCA would result in substantial savings in computational cost especially when using the B-MA solution method (refer to [13] for details). However, the solution cost would still be significantly higher than the proposed procedures.

The need for such efficient algorithms to solve very large QNM's with population constraints (particularly large QNM's of CCN's) provides the primary motivation for presenting the computational procedures in this paper. We present a solution procedure for solving such QNM's at a very low computational cost compared to the aforementioned techniques and with acceptable accuracy. As with the above approaches, our technique reduces computational costs by decomposing a multi-chain problem into a set of single chain problems. Interaction at stations between interfering user chains is taken into account by using *load concealment* [17], which has been previously used in solving various QNM's arising in computer system modeling [12, 17, 26, 27].

It is interesting to note an "inverse" problem. Due to the high cost of solving closed QNM's with large numbers of chains, there have been several attempts to replace closed chains with open chains with equal throughputs [19, 21]. In [11] the accuracy of such methods is estimated by varying the delay in a "source" node and it is shown that this approach becomes relatively inaccurate when this delay tends to zero.

Algorithm	Computational Cost	Numerical Example
B-LZ with Linearizer	$O(K^4MW)$ *per iteration*	$O(2.0 \times 10^8)$ *per iteration*
B-LZ with Bard-Schw	$O(K^3MW)$ *per iteration*	$O(6.3 \times 10^6)$ *per iteration*
B-MA	$O(KM\prod_{k=1}^{K}[W_k+1])$	$O(3.8 \times 10^{22})$
Algorithm 1	$O(KWH)$ *excluding iteration or Proc 1a, when required*	$O(2.9 \times 10^2)$
Algorithm 2	$O(KWH)$ *per iteration*	$O(2.9 \times 10^2)$ *per iteration*
Algorithm 3	$O(K^2W^2H)$ *per iteration*	$O(2.8 \times 10^4)$ *per iteration*

Table 1-1: Comparison of Computational Costs for a Large QN model

Our paper is organized as follows. In Section 2 we present a generalized MCPCQN model and define the necessary parameters. In Sections 3 and 4 we present procedures for solving several variations of the basic MCPCQN. In Section 5 results obtained using our procedures are compared against results obtained using the approximate solution techniques mentioned earlier and against simulation results when available. We present our conclusions in Section 6.

2. Multiple-Chain Population Constrained Model

The general MCPCQN model used in this paper consists of a set of **M** service stations and a set of **K** job-types or chains [15]. The number of stations and chains is denoted by M and K, respectively. Each chain is specified by it's arrival process to the network and it's service demands at each service station in the network. Service times at all stations are assumed to be exponentially distributed with parameter $\mu_{kh} = \mu_h$, $h \in \mathbf{H}_k$, where $\mathbf{H}_k$ is the set of stations visited by chain k. These assumptions are necessary when the queueing discipline at all stations is FCFS and can be relieved for a PS discipline. Service demands are given as $D_{kh} = v_{kh}\ \mu_{kh}$, where v_{kh} is the mean number of visits of a chain k customer to station h. In the case of quasi-random arrivals, this is relative to one visit to the source. In the context of computer communication networks $v_{kh} = 1$, assuming loop-free routing algorithms. Arrival processes are either Poisson with arrival rate λ_k for chain k or quasi-random. A quasi-random process might be due to a set of sources (interactive terminals), which is typically specified as follows, $\lambda_k(n_k) = (L_k - n_k) / Z_k$ where L_k is the number of sources (terminals) and Z_k is the mean source delay (user think time), and n_k is the number of outstanding requests in chain k. Lastly, W_k, is the population constraint parameter for chain k.

3. Solution Procedure for a Multiple Chain QNM with Population Constraints and Fixed Rate Arrivals

In this section we present a procedure for solving a MCPCQN in which all chains are characterized by fixed rate arrivals according to a Poisson process.

Our method is based on using decomposition such that the queueing network is represented by a FESC for each chain. The key point is to represent the effect of all other chains in a suitable manner. It is a well-known result in mixed networks, that in solving closed chains the effect of all open chains can be substituted by their contribution to station utilizations [15]. This result was proven in [19] in solving a mixed QNM for studying congestion control in a tandem channel network with interfering traffic and has been derived by others under more general contexts [14, 20, 24]. The derivation based on MVA is particularly simple and is given in the Appendix for the case of open chains sharing a single server.

We apply this result by "inflating" the service demands of the chain under consideration by the utilizations of all other open chains at all stations visited by the chain, i.e., the load concealment method [17]. The modified service demands of this chain are then used to compute the throughput characteristics of its corresponding FESC. This is an approximation since the other chains are not truly open (i.e., the distribution of the number of customers is affected by population size constraints). The utilizations due to other chains will remain unaffected, however.

It should be noted that if none of the chains are detected to be saturated then Procedure 1 is non-iterative. However, if a chain k is classified to be saturated then the contribution of chain k to station h utilization is $U_{kh} = D_{kh}\ T_k(W_k)$, $h \in \mathbf{H}_k$. We are assuming that a closed chain can be treated as an open chain [11]), which is just a reapplication of the load-concealment method we used for open chains with population-size constraints. The detection of chain k saturation is tantamount to invalidating results obtained for all preceding chains. It should be noted that false initial classifications are possible, e.g., a chain is classified to be saturated due to the fact that station utilizations by other chains, which are later determined to be saturated, were overestimated. Also $T_k(W_k)$ is an under-estimate for all chains classified tentatively to be saturated, except for the last chain to be detected saturated (in this case it is an overestimate). An iterative scheme can be adopted such that the iteration stops when consecutive iterations result in no chain re-classifications from the previous iteration and the effective chain throughputs converge.

When none of the chains is saturated, $W_k = W$ and $\mathbf{H}_k = H$, $\forall\ k \in \mathbf{K}$, and excluding the computation of state probabilities (p_n's) and performance measures, the computational cost is O(KHW). The previous procedure becomes inefficient when solving MCPCQN's consisting of many saturated chains since several iterations will be required. The following procedure obviates the need for iteration through the use of a one-pass algorithm for classifying chains. This procedure may be invoked only when Procedure 1 detects saturation. In turn, Procedure 1a invokes Procedure 1 after resetting the throughputs of closed chains to their maximum value.

Procedure 1. Analysis of a Multiple Chain Population Constrained Queueing Network with Fixed-Rate Arrivals.

- Input the arrival rate to chain k, λ_k, $\forall\, k \in \mathbf{K}$, service demands for all chains at all stations D_{kh}, $h \in \mathbf{H}_k$, $\forall\, k \in \mathbf{K}$ ($\mathbf{H}_k$ is the set of stations visited by chain k), and the population constraints for all chains W_k, $\forall\, k \in \mathbf{K}$.

- Compute U_{kh}, the chain k utilization at station h, given by $U_{kh} = \lambda_k * D_{kh}$, $h \in \mathbf{H}_k$, $\forall\, k \in \mathbf{K}$ (* Note: All chains have been implicitly classified to be unsaturated. *)

* **repeat**
  for $\forall\, k \in \mathbf{K}$ **do**
    begin
    - Account for effect of other open chains by "inflating" service demands at the stations: $D'_{kh} = D_{kh} / (1 - U'_{kh})$, where $U'_{kh} = \sum_{j \neq k} U_{jh}$, $h \in \mathbf{H}_k$.
    - Obtain $T_k(n)$, $n = 1,..,W_k$, by using an exact method for solving chain k in isolation using D'_{kh}, $h \in \mathbf{H}_k$.
    - If $\lambda_k > T_k(W_k)$, then classify chain as saturated (this classification is not final), set $\lambda_k = T_k(W_k)$, and update $U_{kh} = T_k(W_k) * D_{kh}$, $h \in \mathbf{H}_k$, $\forall\, k \in \mathbf{K}$. Else classify chain k as non-saturated. (* Alternately call Procedure 1a as a co-routine the first time a chain saturation is detected *)
    - Use decomposition by substituting the set of stations visited by chain k ($h \in \mathbf{H}_k$) with a FESC with throughputs $T_k(n)$, $n = 1,...,W_k$. Solve the corresponding M/M/1 model (with load-dependent server) for the queue-length distribution, p_n. Use following equations to obtain chain k performance measures.
      - Mean number of customers in chain k: $\bar{N}_k = \sum_{n>0} n\, p_n$ (1)
      - Mean number of customers blocked: $\bar{m}_k = \sum_{n>W_k} (n - W_k)\, p_n$ (2)
      - Mean number of customers active: $\bar{n}_k = \bar{N}_k - \bar{m}_k$ (3)
      - Mean throughput for chain k (this value is equal to λ_k unless the chain is saturated): $T_k = \sum_{n=1}^{W_k} T_k(n) p_n$, where $p_{W_k} = \sum_{n \geq W_k} p_n$ (4)
      - Mean waiting time for chain k using Little's law (should be ignored for saturated chains): $D_k = \bar{m}_k / T_k$ (5)
      - Mean active time for chain k: $A_k = \bar{n}_k / T_k$ (6)
      - Mean response time for chain k (should be ignored for saturated chains): $R_k = \bar{N}_k / T_k$ (7)
    end
* **until (No chains reclassified *and* convergence obtained on T_k's)**

End of Procedure 1.

Procedure 1a. Algorithm to Determine Saturated Chains in a MCPCQN with Fixed Rate Arrivals.

- Classify all chains to be saturated: $\mathbf{S} = \mathbf{K}$, $\mathbf{N} = \{\ \}$, where **S** and **N** denote the sets of Saturated/Non-saturated chains.

repeat

- Solve closed QNM consisting of all **S** chains with the population size of each chain set to the window-size and their service-demands adjusted by the complement of the sum of utilizations by all non-saturated chains (none in the first iteration). Compute $T_s(\bullet)$, the maximum throughput for chain s, assuming all other chains are saturated. Use Linearizer or the TCA [13] to reduce computational cost.

for s ϵ S do

- if $\lambda_s < T_s(\bullet)$ then classify chain s as non-saturated: $\mathbf{N} = \mathbf{N} \bigcup \{s\}$, $\mathbf{S} = \mathbf{S} - \{s\}$

until (All remaining chains in the last pass were all classified to be in the same category)

- if $\mathbf{S} \neq \{\ \}$ then $\lambda_s = T_s(\bullet)$, $\forall\ s \epsilon \mathbf{S}$ (* Adjust chain throughputs for saturated chains *)

- Call Procedure 1 with adjusted throughputs for saturated chains. Only active time is computed for such chains. Ignore statements with (*) when Procedure 1a is executed in conjunction with Procedure 1.

End of Procedure 1a.

In case we have saturation we are in effect substituting a closed (saturated chain) in carrying out the computations in Procedure 1. This is in fact the "inverse" problem noted in Section 1, which has been shown to have relatively low accuracy when the delay at the source station (equivalent to user think times) is negligible [13]. Validation results for Procedure 1 and 1a are given in Section 5.

In Table 3-1 we illustrate the manner in which Procedures 1 and 1a determine whether a chain is saturated or non-saturated. The given classification sequences were performed during the solution of a QNM consisting of 4 chains and 5 stations, where all chains visit all stations, and $D_{kh} = 0.5$ time units, $\forall\ k \epsilon \mathbf{K}$. Arrival rates are given by $\lambda_1 = 0.40$, $\lambda_2 = 0.55$, $\lambda_3 = 0.30$, and $\lambda_4 = 0.45$ arrivals/time unit. In Table 3-1 the letters "S" and "N" are used to form a classification vector (for chain 1 through chain 4) to specify whether the chain has been classified as Saturated or Non-Saturated, respectively. As can be seen, each procedure begins with different initial classifications of the four chains, however, both eventually converge to identical classifications and produce the same performance results (not shown). Estimates of maximum throughput obtained for saturated chains compare very favorably against those obtained using the B-MA algorithm (e.g., typically errors of less than three percent).

4. Solution Procedures for a Multiple Chain QNM with Population Constraints and Quasi-Random Arrivals

In this section we present two solution procedures (Procedures 2 and 3) for solving a MCPCQN in which all chains have quasi-random arrivals (i.e., there are a finite number of sources with given source delays generating requests for each chain). Both procedures rely on the decomposition and load concealment approach used in Procedure 1, however, since the arrival rate is variable it is not possible to compute *a priori* the utilizations due to chains at the stations, i.e., U_{kh}. Instead, a solution can be obtained by starting with an initial estimate of U_{kh} and iterating until convergence is obtained.

	Algorithm 1		Algorithm 1a	
Step	Classi-fication	Comment	Classi-fication	Comment
1	S S N N	Chains 1 & 2 classified as saturated => iterate.	S S N S	Initial saturation check: chains 1,2,4 saturated. Verify in next step.
2	N S N S	Chains 1 & 4 re-classified => iterate.	N S N S	Chain 1 re-classified => verify classi-fication of saturated chains 2 & 4.
3	N S N S	No change in classi-fication. No T_k convergence => iterate.	N S N S	No change in classi-fication => STOP.
4	N S N S	No change in classi-fication *and* T_k convergence => STOP.		

Table 3-1: Comparison of Algorithm 1 and 1a Classification Algorithms

	Case 1: Moderate		Case 2: Heavy		
	B-LZ with Linearizer (1)	Alg 1 (2)	B-LZ with Linearizer (3)	Algorithm 1 All (4)	Algorithm 1 LF<0.8 (5)
Dk % Mean Err	25.90	1.84	---	---	5.21
% Max Err	72.10	22.14	---	---	20.70
Ak % Mean Err	0.94	0.32	1.95	7.71	0.93
% Max Err	8.45	1.76	8.50	81.17	3.10
Rk % Mean Err	1.58	0.38	---	---	1.20
% Max Err	9.62	1.94	---	---	3.60

Table 5-1: Algorithm 1 Validation Results

Except for the required iteration, Procedure 2 is a direct extension of Procedure 1 for the solution of MCPCQN's with quasi-random arrivals. Procedure 3, on the other hand, is developed within the framework of Reiser's approximate-MVA algorithm for the solution of product-form QNM's with a very large number of chains [21]. However, since Reiser's algorithm also solves each chain in isolation, using load concealment to account for the effect of interfering chains at a given queue, the overall approach is very similar in spirit to that of Procedure 2. The major difference is the manner in which the FESC load-dependent throughputs, $T_k(n)$, are computed. Instead of directly using the inflated service demands to obtain $T_k(n)$, as in Procedures 1 and 2, the inflated service demands are used to compute estimates of chain k queue-length statistics which are incorporated into Reiser's MVA-based heuristic for computing $T_k(n)$. The reader is referred to [21] for a detailed description of Reiser's algorithm.

We now present formal descriptions of both procedures.

Procedure 2: Analysis of a Multiple Chain Population Constrained Queueing Network with Quasi-Random Arrivals.

- Input service demands for all chains at all stations D_{kh}, $h \in \mathbf{H}_k$, $\forall k \in \mathbf{K}$, the population constraints for all chains W_k, $\forall k \in \mathbf{K}$, the number of sources for chain k, L_k, $\forall k \in \mathbf{K}$, and the source delays Z_k, $\forall k \in \mathbf{K}$.
- Initialize utilizations of all stations due to all chains: $U_{kh} = 0$, $h \in \mathbf{H}_k$, $\forall k \in \mathbf{K}$ and all mean response times: $R_k = 0$, $\forall k \in \mathbf{K}$.

repeat

- $R'_k = R_k$, $\forall k \in \mathbf{K}$ (* Save previous values for checking convergence *)

for $\forall k \in \mathbf{K}$ **do**
begin

- Compute utilization at station $h \in \mathbf{H}_k$ by all other chains besides k:

$$U'_{kh} = \sum_{j \neq k} U_{jh}, \quad h \in \mathbf{H}_k.$$

- Account for effect of other open chains by "inflating" service demands for chain k at the stations:

$$D'_{kh} = D_{kh} / (1 - U'_{kh}), \quad h \in \mathbf{H}_k.$$

- Obtain $T_k(n)$ by solving chain k in isolation using an exact method for $n = 1,\ldots,W_k$, and using D'_{kh}, $h \in \mathbf{H}_k$.
- Compute chain k queue-length distribution, p_n, by solving M/M/1//M queueing system, using specified variable-rate arrival process and $T_k(n)$, $n = 1,\ldots,W_k$.
- Obtain T_k, mean throughput for chain k, using equation 4.
- Obtain station utilizations due to chain k, $U_{kh} = T_k D_{kh}$, $h \in \mathbf{H}_k$.

end

until $(|R'_k - R_k| / R_k < \epsilon, \forall k \in \mathbf{K})$

- Compute performance measures using equations 1 to 7 in Procedure 1.

End of Procedure 2.

Procedure 3: Analysis of a Multiple Closed Class Population Constrained Queueing Network using Reiser's Algorithm.

- Input service demands for all chains at all stations D_{kh}, $h \in \mathbf{H}_k$, $\forall k \in \mathbf{K}$, the population constraints for all chains W_k, $\forall k \in \mathbf{K}$, the number of sources for chain k, L_k, $\forall k \in \mathbf{K}$, and the source delays Z_k, $\forall k \in \mathbf{K}$.
- Initialize: Utilizations of all stations due to all chains: $U_{kh} = 0$, $h \in \mathbf{H}_k$, $\forall k \in \mathbf{K}$, all mean response times: $R_k = 0$, $\forall k \in \mathbf{K}$, and mean queue-lengths of all chains at all stations: $L_{kh}[N] = 0$, $h \in \mathbf{H}_k$, $\forall k \in \mathbf{K}$, where $[N] = [\bar{n}_1,...,\bar{n}_k,...,\bar{n}_K]$

repeat

- $R'_k = R_k$, $\forall k \in \mathbf{K}$ (* Save previous values for checking convergence *)

for $\forall k \in \mathbf{K}$ **do**
begin

- Compute utilization at station $h \in \mathbf{H}_k$ by all other chains besides k:

  $U'_{kh} = \sum_{j \neq k} U_{jh}$, $h \in \mathbf{H}_k$.

- Account for the effect of other open chains by "inflating" service demands for chain k at the stations:

  $D'_{kh} = D_{kh} / (1 - U'_{kh})$, $h \in \mathbf{H}_k$.

- Compute chain k load-dependent throughputs $T_k[n]$, $n = 1,...,W_k$, using **Procedure 3a** (Reiser's heuristic [21]).
- Compute chain k queue-length distribution, p_n, by solving M/M/1//M queueing system, using specified variable-rate arrival process and $T_k(n)$, $n = 1,...,W_k$.
- Obtain T_k, mean throughput for chain k, using equation 4.
- Obtain chain k station utilizations, $U_{kh} = T_k D_{kh}$, $h \in \mathbf{H}_k$.
- Obtain mean queue-lengths $L_{kh}[N]$ for chain k:

  $L_{kh}[N] = \sum_{n=1}^{W_k} \tilde{L}_{kh}(n)\, p_n$, where $p_{W_k} = \sum_{n=W_k}^{L_k} p_n$

- Compute mean chain k response time, R_k, using equation 7.

end

until $(|R'_k - R_k| / R_k < \epsilon, \forall k)$

- Compute final performance measures using equations 1 to 7 in Procedure 1.

End of Procedure 3.

Procedure 3a: Computation of chain k throughputs $T_k[n]$, $n = 1,...,W_k$.

- Input Parameters: $L_{jh}[N]$, $h \in H_j$, $\forall j \in \mathbf{K} - \{k\}$.
- Solve chain k in isolation with single chain MVA using $D'_{k\,h}$, $h \in \mathbf{H}_k$, and obtain queue-lengths: $\widetilde{L}_{kh}[n]$, $h \in \mathbf{H}_k$, $n = 1,...,W_k$.
- Compute $\epsilon_{kh}[n] = \widetilde{L}_{kh}[n] - \widetilde{L}_{kh}[n-1]$, $h \in \mathbf{H}_k$, $n = 1,...,W_k$

for $n = 1,\ldots,W_k$ **do**

begin

repeat

- Compute chain k residence time at station h using Arrival Theorem [15,17]:

$$R_{kh}[n] = D_{kh}(1 + \sum_{j \neq k} L_{jh}[N] + \widetilde{L}_{kh}[n] - \epsilon_{kh}[n]),\ h \in \mathbf{H}_k.$$

- Compute total chain k residence time (used for testing convergence):

$$R_k[n] = \sum_{h \in \mathbf{H}_k} R_{kh}[n].$$

- Compute chain k throughput: $T_k[n] = n / R_k[n]$.
- Compute chain k mean queue-length at station h:

$\widetilde{L}_{kh}[n] = T_k[n] * R_{kh}[n]$, $h \in \mathbf{H}_k$.

until (Convergence on $R_k[n]$**)**

end

End of Procedure 3a.

Assuming $W_k = W$, $H = \mathbf{H}_k$, $\forall k \in \mathbf{K}$, and excluding the computation of state probabilities (p_n's) and performance measures, the computational cost for Procedure 2 *per iteration* is $O(KHW)$. Under the same assumptions the computational cost for Procedure 3 is approximately the equivalent of solving Reiser's algorithm W times. The computational cost of solving Reiser's algorithm is given by [21]: $O[KH\,(2 + \sum_{k=1}^{K} W_k)]$ $\approx O[K^2HW]$. Thus the *per iteration* computational cost of Procedure 3 is $O[K^2HW^2]$. It should be noted that Procedure 3, unlike the iteration in the B-LZ algorithm, does not require any complete solutions of a K-chain QNM for obtaining the load-dependent throughputs $T_k[n]$. Rather, Procedure 3 only requires the "partial" single chain computation required in Procedure 3a. This partial solution approach alone reduces the per-iteration computational cost of Procedure 3 by a factor of K (approximately) with respect to a comparable B-LZ method using Reiser approximate-MVA rather than the Bard-Schweitzer or Linearizer approximate solution methods at the lower level. We assess the accuracy of this procedure in Section 5.

5. Procedure Validation

5.1. Procedure 1 Validation

In this section we validate Procedure 1 against the B-MA approximate solution technique based on manifold equivalence and decomposition [3, 18]. Although the B-MA approach is also approximate, its high degree of accuracy should provide a reliable "benchmark" against which our procedure can be tested and verified. Furthermore, based on experience using the B-MA approach, we observed that the most accurate results are obtained by permuting the order in which the chains are solved. Specifically, for a QNM consisting of the chains $\mathbf{K} = \{1,...,K\}$, the QNM is solved K times where the order in which the chains are solved are specified by the following sequence: $\mathbf{K}^{(1)} = \{1,...,K\text{-}1,K\}$, $\mathbf{K}^{(2)} = \{2,...,K,1\},...,\mathbf{K}^{(K)} = \{K,1,..,K\text{-}1\}$, i.e., the order of a set is obtained by cyclically shifting the members of the "predecessor" sequence. The most accurate results of each sequence are obtained for the first chain (which is the last chained solved in the B-MA algorithm), and are the only results saved for that solution. B-MA results for validation are obtained in this manner.

Our validation consists of solving a QNM consisting of $K = 4$ chains and $M = H = 5$ stations (all chains visit all stations). We consider two sets of fixed arrival rates (Case 1 and Case 2): (1) Moderate network load with $\lambda_1 = 0.3$, $\lambda_2 = 0.25$, $\lambda_3 = 0.2$, $\lambda_4 = 0.15$ arrivals per time unit, and (2) Heavy network load with $\lambda_1 = 0.4$, $\lambda_2 = 0.4$, $\lambda_3 = 0.2$, $\lambda_4 = 0.15$ arrivals per time unit. All population size constraints are equal, $W_k = 4$, $\forall\, k \in \mathbf{K}$ and service demands, D_{kh}, at each device are randomly generated from a uniform distribution over [0,1]. For each QNM generated, we obtain the mean response time, R_k, mean active time, A_k, and mean waiting time, $D_k = R_k - A_k$, for all 4 chains. For D_k, A_k, and R_k we compute the Mean Error (relative error with respect to results obtained using the B-MA algorithm) and the Maximum Error over all chains using Procedure 1 and the B-LZ algorithm (for fixed rate arrivals).

For Case 1, error statistics were computed from 50 randomly generated sets of service demand data. As can be seen in Table 5-1, results obtained using Procedure 1 (column 2) were very accurate when compared against B-MA results. Mean errors for Procedure 1 averaged less than 2 percent for all performance measures, outperforming the B-LZ algorithm (column 1).

For Case 2, error statistics were computed from 25 randomly generated sets of service demand data. Most of the 25 cases contained at least one chain operating near saturation, while in 6 cases at least one chain was actually saturated. In Table 5-1 we show results only for A_k (since statistics for D_k and R_k are not particularly meaningful when saturated chains exist). From Table 5-1 (column 4) it can be seen that the mean error for Procedure 1 is still quite acceptable, however, the procedure is susceptible to large errors when there are chains at or near saturation, as indicated by the large maximum error obtained. Results (column 3) using the B-LZ algorithm are far more robust.

The last result shown in Table 5-1 (column 5), represents statistics for a special group of Case 2 results. The subset of Case 2 results used in computing these error statistics are selected as follows. First we define the chain k Load Factor as $LF_k = \lambda_k$ / (Maximum Chain k Throughput). We then only use results obtained from those sets of data in which $LF_k < 0.8$, $\forall\, k \in \mathbf{K}$. Using this screening procedure we filter out results which produce abnormally high errors due to saturation/near-saturation conditions, while retaining those results which still stress Procedure 1 under more realistic heavy load conditions (from a CCN modeling point of view). Under such conditions, results obtained using Procedure 1 are still quite acceptable.

5.2. Procedure 2 and 3 Validation

In this section we validate Procedures 2 and 3 by comparing them against simulations performed by Sauer [23] for a timesharing QNM consisting of a CPU (queue 1), four equally loaded disks (queues 2 through 5), and two interactive customer classes with independent memory constraints. The service demands for customer classes 1 and 2 are given by D_{1h} = (0.1, 0.0875, 0.0875, 0.0875, 0.0875) and D_{2h} = (2.0, 0.175, 0.175, 0.175, 0.175), respectively. Mean user think times are $Z_1 = 5$ and $Z_2 = 10$ time units. Simulations were conducted for 9 models, varying in the number of terminal users, L_k, and memory size constraints, W_k, for each class, as shown in Table 5-2. Models were chosen such that examples of low, moderate, and high memory contention were represented. Each simulation was terminated when the relative width of the 90% confidence interval for class-independent mean response time was 5%.

Tables 5-3, 5-4, and 5-5 summarize results for class 1 and 2 response times, and CPU utilization, respectively, as obtained using decomposition, simulation, Procedure 2, and Procedure 3. Decomposition results were obtained by Sauer [23] by solving the underlying Markov process of a flow-equivalent model using a recursive technique due to Herzog, Woo, and Chandy [9]. In addition, for comparison purposes, we indicate results obtained using the B-LZ algorithm (with Linearizer), the B-LZ algorithm (with Bard-Schweitzer) and the B-MA algorithm. Results pertaining to this

No.	Sauer's Decomp	Simulation C.I.	B-LZ w/ Linear	B-LZ w/ Bd-Sch	Alg 2	Alg 3	B-MA
1	0.80	(0.77, 0.80)	0.798	0.808	0.820	0.803	0.795
2	0.93	(0.90, 0.95)	0.926	0.935	0.964	0.931	0.922
3	4.83	(4.68, 4.93)	4.922	4.948	4.981	4.950	4.830
4	1.06	(1.03, 1.08)	1.066	1.091	1.165	1.079	1.061
5	1.17	(1.13, 1.19)	1.173	1.190	1.314	1.184	1.171
6	4.10	(3.97, 4.17)	4.155	4.202	4.445	4.189	4.102
7	1.50	(1.47, 1.54)	1.526	1.559	1.741	1.541	1.504
8	1.69	(1.67, 1.74)	1.704	1.750	1.919	1.727	1.683
9	2.33	(2.30, 2.42)	2.338	2.366	2.735	2.354	2.333

Table 5-2: Comparison of Class 1 Mean Response Time

No.	Sauer's Decomp	Simulation C.I.	B-LZ w/ Linear	B-LZ w/ Bd-Sch	Alg 2	Alg 3	B-MA
1	4.66	(4.49, 5.09)	4.701	5.267	4.727	4.590	4.675
2	5.08	(4.71, 5.31)	5.135	5.756	5.142	5.534	5.096
3	4.09	(3.86, 4.31)	4.099	4.352	4.191	4.252	4.090
4	7.59	(6.70, 7.64)	7.793	9.269	7.606	8.625	7.599
5	9.13	(8.42, 9.69)	9.396	11.413	9.020	11.114	9.139
6	6.44	(6.08, 6.95)	6.464	7.280	6.499	7.127	6.443
7	13.30	(12.15, 14.10)	14.087	16.279	12.698	15.520	13.303
8	12.46	(11.98, 13.26)	12.803	14.909	12.115	14.117	12.484
9	14.47	(13.48, 15.15)	14.551	18.161	13.552	17.890	14.474

Table 5-3: Comparison of Class 2 Mean Response Time

No.	Sauer's Decomp	Simulation C.I.	B-LZ w/ Linear	B-LZ w/ Bd-Sch	Alg 2	Alg 3	B-MA
1	0.62	(0.60, 0.63)	0.617	0.606	0.615	0.613	0.618
2	0.60	(0.59, 0.61)	0.602	0.581	0.600	0.595	0.603
3	0.49	(0.48, 0.50)	0.485	0.480	0.482	0.481	0.487
4	0.84	(0.83, 0.85)	0.832	0.804	0.827	0.816	0.836
5	0.80	(0.79, 0.81)	0.795	0.765	0.791	0.769	0.800
6	0.69	(0.69, 0.71)	0.692	0.673	0.681	0.677	0.695
7	0.96	(0.96, 0.97)	0.945	0.914	0.946	0.926	0.958
8	0.95	(0.95, 0.96)	0.948	0.914	0.940	0.927	0.954
9	0.87	(0.87, 0.88)	0.871	0.827	0.857	0.830	0.872

Table 5-4: Comparison of Total CPU Utilization

example using a "delay technique" are reported in [28] and shown to be more accurate than the mean population technique (the B-LZ algorithm) however are not discussed in this paper.

For Procedure 2, all of the class 2 response times obtained were within the simulation confidence intervals, while class 1 response times were all slightly outside the given confidence intervals (i.e., results for cases with low to moderate CPU utilizations were 1 to 8 percent outside their confidence intervals, while results for cases with high CPU utilizations were slightly further outside their confidence intervals). Over half of the values obtained for CPU utilization were within their confidence intervals, while the maximum distance of those outside their confidence intervals was less than 1.5%.

Results obtained using Procedure 3 show that results were comparable to those results obtained using the B-LZ algorithm with Bard-Schweitzer, with Procedure 3 giving slightly better results for class 2 performance measures. Over half of the mean response times obtained for class 1 were within their respective confidence intervals; all other class 1 mean response time results were within one percent of their confidence intervals. For class 2 mean response times, results for cases 1 and 3 were within their confidence intervals, with deviations ranging from 2 to 18 percent for the remaining cases. Using a convergence criteria of $\epsilon = 0.001$ (see Procedure 2 and 3 descriptions) for solving the nine models, both Procedures 2 and 3 required approximately 4 to 5 iterations for convergence, while the B-LZ approaches required approximately 3 to 4 iterations.

6. Conclusions

We presented three procedures for solving multiple class population constrained QNM's. Our emphasis was on developing computationally efficient procedures for solving very large QNM's associated with computer communication networks.

Procedure 1, which is based on load concealment, was shown to be a fast and accurate procedure for solving MCPCQN's in which there are fixed rate arrivals to all chains. The procedure was observed to be comparable in accuracy to computationally more expensive techniques using manifold equivalence and decomposition. Largest deviations occurred when the QNM contained chains which were at or near saturation.

Procedures 2 and 3 were proposed for solving MCPCQN's in which all chains are characterized by quasi-random arrivals. Procedure 2 was a direct extension of Procedure 1, while Procedure 3 was an adaptation of Reiser's approximate MVA algorithm for solving product-form QNM's [21]. Results obtained using both procedures compared favorably against techniques [3, 16, 18] having significantly higher costs. Procedure 2 has the lower computational cost of the two procedures, while Procedure 3 tends to be more robust.

Acknowledgements

The first author wishes to acknowledge a discussion with Dr. Alexandre Brandwajn, which motivated him to write this paper.

	Terminals		MPL Constraint	
Case	Class 1	Class 2	Class 1	Class 2
1	20	2	4	2
2			3	1
3			1	1
4	30	3	7	2
5			5	1
6			2	1
7	40	4	14	4
8			9	3
9			5	1

Table 5-1: Parameters for Sauer's Simulation Test Cases

I. Appendix

Consider a single-server station with a PS queueing discipline, which services K chains with Poisson arrivals. The arrival rates are λ_k and the mean service times are X_k, k = 1,...,K. The utilization of the server by each chain is denoted by $\rho_k = \lambda_k X_k$, k = 1,...,K. The total server utilization is $\rho = \sum_{k=1}^{K} \rho_k$ and $\rho < 1$, i.e., none of the chains is saturated. Since Poisson arrivals see time averages, the mean response time for chain k is,

$$\bar{R}_k = X_k \left(1 + \sum_{k=1}^{K} \bar{n}_k\right) \qquad (8)$$

$$= X_k \left(1 + \bar{n}\right), \quad k = 1,\ldots,K$$

where $\bar{n}_k$ is the mean population of chain k, $\bar{n}$ is the total population, and the second factor expresses the expansion of service time due to the presence of other chains. Multiplying the above equations by λ_k and summing the equations we obtain: $\bar{n} = (1 + \bar{n})\,\rho$. It follows that: $\bar{n} = \rho/(1 - \rho)$. Substituting into equation 8 we have:

$$\bar{n}_k = \rho_k/(1 - \rho) \qquad (9)$$

The same result could have been obtained by inflating the service times by dividing them by the remaining capacity of the server, i.e., using

$$X'_k = X_k/(1 - \sum_{j \neq k} \rho_j).$$

References

1. Bard, Y. Some Extensions to Multi-Class Queueing Network Analysis. In *Performance of Computer Systems*, M. Arato et al., Eds., North-Holland, Amsterdam, 1979, pp. 51-62.

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

3. Brandwajn, A. Fast Approximate Solution of Multiprogramming Models. Proc. ACM/SIGMETRICS Conf. on Measurement and Modeling, September, 1982, pp. 141-149.

4. Chandy, K.M. and Sauer, C.H. "Approximate methods for analyzing queueing network models of computer systems." *Computing Surveys 10*, 3 (September 1978), 281-317.

5. Chandy, K.M., and Neuse, D. "Linearizer: A Heuristic Algorithm for Queueing Network Models of Computing Systems." *Commun. ACM 25*, 2 (February 1982), 126-134.

6. Chow, W.M. "Approximations for Large Scale Closed Queueing Networks." *Performance Evaluation 3*, 1 (February 1983), 1-12.

7. de Souza e Silva, E., Lavenberg, S.S., and Muntz, R.R. A Perspective on Iterative Methods for the Approximate Analysis of Closed Queueing Networks. Proc. International Workshop on Applied Mathematics and Performance/Reliability Models of Computer/Communication Systems, Pisa, Italy, September, 1983.

8. Dowdy, L.W. and Gordon, K.D. Algorithms for Nonintegral Degrees of Multiprogramming in Closed Queueing Networks. Tech. Rept. CS-82-04, Vanderbilt University, April, 1982.

9. Herzog, U., Woo, L., and Chandy, K.M. "Solution of Queueing Problems by a Recursive Technique." *IBM J. Res. Develop. 19* (May 1975), 295-300.

10. Lam, S.S. "Queueing Networks with Population Size Constraints." *IBM J. Res. Develop. 21*, 4 (July 1977), 370-378.

11. Lam, S.S. and Lien, Y.L. Modeling and Analysis of Flow Controlled Packet Switching Networks. Proc. 7th Data Communications Symposium, 1981, pp. 98-107.

12. Lam, S.S., and Wong, J.W. "Queueing Network Models of Packet Switching Networks, Part 2: Networks with Population Size Constraints." *Performance Evaluation 2* (1982), 161-180.

13. Lam, S.S. and Lien, Y.L. "A Tree Convolution Algorithm for the Solution of Queueing Networks." *Commun. ACM 26*, 3 (March 1983), 203-215.

14. Langer, A. and Shum, A. Product-Form Queueing Networks Containing Mixed Customer Classes. Harvard University, 1982.

15. Lavenberg, S.S. (Ed.). *Computer Performance Modeling Handbook*. Academic Press, 1983.

16. Lazowska, E.D., and Zahorjan, J. Multiple Class Memory Constrained Queueing Network. Proc. 1982 SIGMETRICS Conference on Measurement and Modeling of Computer Systems, ACM, Seattle, Washington, August, 1982, pp. 130-140.

17. Lazowska, E.D. et al. *Quantitative System Performance: Computer System Analysis Using Queueing Network Models.* Prentice-Hall, 1984.

18. Menasce, D.A. and Almeida, V.A.F. "Operational Analysis of MultiClass Systems with Variable Multiprogramming Level and Memory Queueing." *Computer Performance 3*, 3 (September 1982), 145-159.

19. Pennotti, M. and Schwartz, M. "Congestion Control in Store and Forward Tandem Links." *IEEE Trans. Comm. COM-23*, 12 (December 1975), 1434-1443.

20. Reiser, M. and Kobayashi, H. "Queueing Networks with Multiple Closed Chains: Theory and Computational Algorithms." *IBM J. Res. Develop. 19*, 23 (May 1975), 283-294.

21. Reiser, M. "A Queueing Network Analysis of Computer Communication Networks with Window Flow Control." *IEEE Trans. Comm. COM-27*, 8 (August 1979), 1199-1209.

22. Sauer, C.H. Numerical Solution of Some Multiple Chain Queueing Networks. Tech. Rept. RC 8986, IBM T.J. Watson Research Center, August, 1981.

23. Sauer, C.H. "Approximate Solution of Queueing Networks with Simultaneous Resource Possession." *IBM J. Res. Develop. 25*, 6 (November 1981), 894-903.

24. Sauer, C.H. "Computational Algorithms for State-Dependent Queueing Networks." *ACM Trans. Computer Systems 1*, 1 (February 1983), 67-92.

25. Schweitzer, P. Approximate Analysis of Multiclass Closed Networks of Queues. Proc. of International Conf. Stochastic Control and Optimization, Amsterdam, 1979.

26. Thomasian, A. and Bay, P. "Analysis Techniques for Queueing Network Models of Multicomputer Systems with Shared Resources." *Computer Performance 4*, 3 (September 1983), 151-166.

27. Thomasian, A., and Bay, P. Analytic Solution of an Integrated Performance Model of a Computer Communication Network with Window Flow Control. Proc. 1984 SIGCOMM Symp: Communication Architectures and Protocols, ACM, Montreal, Canada, June, 1984. 8 pages.

28. Zahorjan, J. Workload Representations of Queueing Models of Computer Systems. Proc. ACM/SIGMETRICS Conf. on Measurement and Modeling of Computer Systems, Minneapolis, MN., August, 1983, pp. 70-81.