Two Classes of Performance Bounds for Closed Queueing Networks *

Ching-Tarng Hsieh ** and Simon S. Lam
Department of Computer Sciences, The University of Texas at Austin, Taylor Hall 2.124, Austin, TX 78712-1188, U.S.A.

Received 3 November 1985
Revised 30 July 1986

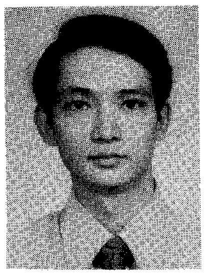
Two classes of performance bounds for separable queueing networks are presented, one for single-chain networks and one for multichain networks. Unlike most bounds for single-chain networks, our bounds are not based upon the consideration of balanced networks. Instead, they are obtained by assuming mean queue lengths to be proportional to server loads; hence, they are called *proportional bounds*. Proportional bounds are tighter than balanced bounds because individual server loads are retained as parameters in a bound's formula. For the same reason, they require more computational effort than balanced bounds. We also show how proportional bounds are related to balanced bounds. Next we present generalized bounds that are calculated iteratively over sequences of population sizes; our method extends that of Eager and Sevcik (1983). These generalized bounds are shown to have a nested property. Furthermore, we present optimal population sequences, over all sequences of the same length, for getting the tightest upper and lower bounds. The other emphasis of this paper is on performance bounds for networks with many closed chains and many service centers. Bounding techniques are especially important for multichain networks since the computation time and space requirements are often so large that an exact solution is not feasible. Models of communication networks typically have many routing chains which are characterized by a sparseness property. In the computation of our performance bounds for multichain networks, we improve their accuracy by making use of routing information and exploiting the sparseness property.

Keywords: Queueing Networks, Asymptotic Analysis, Performance Evaluation, Throughput Bounds, Product-Form Networks, Communication Networks.

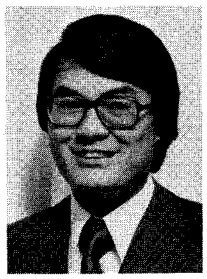
* This research was supported by the National Science Foundation under Grant No. ECS-8304734.
** AT&T Bell Laboratories, P.O. Box 3050, Naperville, IL 60566-7050, U.S.A.

1. Introduction

Separable queueing networks have been widely used as models for predicting the performance of multiprogramming systems as well as packet communication networks. The solution of separable networks requires substantially less computation time than does the solution of nonseparable networks. Yet the computation time required by the best algorithms available is nevertheless proportional to the number of customers for single-chain networks and exponential in the number of routing chains for multichain networks. Such computational requirements are very high for many mod-



Ching-Tarng Hsieh received a B.S. in Engineering Science in 1973 from National Cheng Kung University, Tainan, Taiwan, and an M.S. in Applied Mathematics in 1977 from National Tsing Hua University, Hsinchu, Taiwan. He has finished his Ph.D. degree in Computer Science at the University of Texas at Austin. He was with the Chung Shan Institute of Science and Technology from 1977 to 1980 where he participated in the research and development of Chinese character input/output and command, control, communication, and intelligence systems. At present he is with AT&T Bell Laboratories, Naperville, IL, U.S.A. His research interests include performance evaluation and modeling of computer systems, communication networks, and distributed systems. He is a member of Phi Kappa Phi, the Association for Computing Machinery, ACM Sigmetrics, and IEEE Computer Society.



Simon S. Lam received the BSEE degree (with Distinction) from Washington State University, Pullman, Washington, in 1969, and the M.S. and Ph.D. degrees in Engineering from The University of California, Los Angeles, in 1970 and 1974 respectively. From 1974 to 1977, he was a research staff member at the IBM T.J. Watson Research Center, Yorktown Heights, New York. Since September 1977, he has been with the University of Texas at Austin where he is Professor of Computer Sciences and holds the Second David Bruton Jr. Centennial Professorship. His current research interests include most aspects of communications architectures and protocols, as well as techniques for the verification and performance analysis of distributed systems.

Dr. Lam is an IEEE Fellow and was a co-recipient of the 1975 Leonard G. Abraham Prize Paper Award from the IEEE Communications Society. He serves on the editorial boards of three journals, *IEEE Transactions on Communications*, *Proceedings of IEEE*, and *Performance Evaluation*.

0166-5316/87/$3.50 © 1987, Elsevier Science Publishers B.V. (North-Holland)

els of realistic networks and systems. (This is especially true for communication networks with many routing chains.) Since the separable queueing networks are themselves approximate models of real systems and networks, an exact solution of their performance measures is not always warranted. This is often true in the early stages of system design.

1.1. Previous work on bounding techniques

Let us first consider networks with a single routing chain. Techniques for deriving upper and lower bounds of the mean delays and throughputs of separable queueing networks have been presented by several authors. The asymptotic bounds of Muntz and Wong [17] are actually applicable to a larger class of queueing networks than the class of separable networks. They also have the advantages of being simple and easy to compute. However, asymptotic bounds are in general very loose and do not provide adequate information to achieve most system design objectives. The work of Zahorjan et al. [25] was probably the first development of bounds that are restricted to the class of separable queueing networks. Their balanced job bounds (BJBs) were derived by considering related networks whose servers have identical loads and whose performance measures bound those of the original network. Separable networks with fixed-rate service centers but without delay service centers were considered. (Delay service centers are sometimes referred to in the literature as infinite-server service centers.)

Extensions of BJBs for separable networks with both fixed-rate and delay service centers were developed by Eager and Sevcik [5] and by Kriz [10]. In addition, Eager and Sevcik presented hierarchies of upper and lower bounds. Each hierarchy is a sequence of successively more accurate upper (or lower) bounds with the BJB bound as the first element in the sequence and the exact solution as its limit. Kriz also presented hierarchies of upper and lower bounds, called balanced bounds, with his extensions of BJBs at the first levels of the hierarchies. Methods for obtaining hierarchies of bounds were also developed by Suri [23] and by Stephens and Dowdy [22]. Like the method of Eager and Sevcik and the method of Kriz, Suri's method is based upon the MVA recursion equations. On the other hand, the method of Stephens and Dowdy is based upon the convolution algorithm recursion. In each method, a sequence (or hierarchy) of bounds is generated by an iterative procedure which allows one to trade computation time for accuracy. It is interesting to note that BJBs or extensions of BJBs were used as the first-level bounds in all of the methods for generating increasingly more accurate bounds. Only Kriz presented bounds in [10] that are not based upon the consideration of balanced networks.

For multichain networks, BJBs were proposed by Zahorjan et al. [25] for networks of fixed-rate service centers, which were later extended by Kriz to networks including delay service centers as well [10]. Very recently, improved throughput bounds for multichain networks were presented by Eager and Sevcik [6] and by Kerola [9].

1.2. Overview of our work

We have developed two classes of performance bounds, one for single-chain networks and one for multichain networks. Like BJBs, our bounds are derived from the MVA recursion. But unlike BJBs, our bounds are not based upon the consideration of balanced networks. Instead, they are obtained by assuming that mean queue lengths are proportional to the loads of the corresponding servers. Hence, these bounds are called *proportional bounds*. Individual server loads are included as parameters in the mathematical formulas of proportional bounds, whereas only the minimum, maximum or mean value is included in a balanced bound. For this reason, proportional bounds are proved to be more accurate than corresponding balanced bounds; for the same reason, they require more computational effort. Proportional bounds are presented in Section 2 below. We also show that the 'noniterative' bounds (proportional or balanced, upper or lower) can all be specified as special cases of a unified bounding formula.

In Section 3, we present algorithms for computing bounds iteratively over sequences of population sizes; these will be referred to as *generalized bounds*. Since we do not require population sizes in a sequence to be consecutive integers, our method extends and subsumes the Eager–Sevcik approach for trading computation time for accuracy. We prove that generalized bounds have a nested property. Furthermore, we present population sequences for getting the tightest upper and lower bounds, and prove their optimality over all allowable sequences of the same length. Our algorithms are initialized at level 0 with proportional bounds. Our results (nested property and optimal sequences) and their proofs, however, do not depend on the use of proportional bounds at level 0; they are still valid if generalized bounds are computed using balanced bounds to initialize the algorithms.

Another emphasis of this paper is a class of performance bounds for networks with many chains and many service centers. Bounding techniques are especially important for multichain networks for which the computation time and space requirements of an exact solution may be too large to be feasible.

In recent years, several authors, including us, have argued for the use of closed multichain queueing networks to predict the performance of store-and-forward communication networks and to solve network design problems such as the optimal selection of routes and channel capacities [12]. A recent experimental study of ours [14] further illustrated the inadequacy of the open queueing network model and the desirability of the closed network model. The obstacle that currently prevents the closed network model from being widely used by network designers and analysts is the large computational time and space required to calculate performance measures. Models of realistic communication networks should have tens of closed chains or more, each modeling a flow-controlled virtual channel. Such models cannot be solved by the conventional convolution and MVA algorithms [3,4,18,19]. Lam and Lien observed in [13] that models of communication networks have routes that are often characterized by sparseness and locality properties. They developed the tree convolution algorithm that exploits routing information and can solve networks with tens of closed chains. Tree MVA algorithms were subsequently developed independently by Tucci and Sauer [24] and by Hoyme et al. [7].

Tree algorithms are too expensive to be used in network design algorithms which need to evaluate very efficiently a network's performance given some design perturbations or parameter changes. Reasonably tight performance bounds are very useful for speeding up heuristic search procedures based upon the branch-and-bound technique. Another place where we have found a useful application of performance bounds of closed multichain networks is in the implementation of dynamic scaling in convolution algorithms [11] to prevent the occurrences of floating point underflows and overflows. (We employ the tree convolution algorithm and its associated tree of arrays whenever an exact solution is called for in our network design techniques [14].) In this role, the bounds can be very loose but must be efficient to compute.

We have developed two algorithms for computing performance bounds for closed multichain networks. They are presented in Section 4 below. Like the tree convolution algorithm, routing information is exploited in the computation of these performance bounds. The first algorithm is based upon the BJB idea. The second algorithm further exploits routing information to improve the bounds obtained by the first algorithm. The accuracy of these bounds is much better than BJBs for networks with many sparse routing chains. Our algorithms are similar to those of Eager and Sevcik [6] but were derived and investigated independently [8].

Throughout this paper the networks considered are BCMP networks with fixed-rate (F) servers and delay (D) service centers [2]. M denotes the total number of service centers.

2. Proportional bounds

Let us consider formulas in the MVA recursion. The mean delay $D_m(n)$ of center m in a queueing network with n customers is

$$D_m(n) = \begin{cases} \tau_m(1 + q_m(n-1)) \\ \qquad \text{if } m \text{ denotes a fixed-rate server} \\ \tau_m \quad \text{if } m \text{ denotes a delay server,} \end{cases} \tag{1}$$

where τ_m is the mean service time at center m and $q_m(n-1)$ is the mean queue length at service center m in a network with $n-1$ customers [19,21].

From Little's formula [15], the throughput $T(n)$ and mean queue length $q_m(n)$ are

$$T(n) = n \Big/ \sum_{m=1}^{M} D_m(n) \tag{2}$$

and

$$q_m(n) = T(n) D_m(n). \tag{3}$$

Equations (1), (2) and (3) form the main recursion of the MVA method [19]. Without loss of generality, assume that the fixed-rate service centers are labeled 1, 2,..., M_F, and the remaining M_D centers are delay centers, where $M_D \geqslant 0$, $M_F \geqslant 0$, and $M_D + M_F = M$. We further assume that $\tau_1 \leqslant \tau_2 \leqslant \cdots \leqslant \tau_{M_F}$. Define

$$L_F = \sum_{m=1}^{M_F} \tau_m, \quad L_D = \sum_{m=M_F+1}^{M} \tau_m, \quad L = L_F + L_D,$$

and

$$Q_F(n) = \sum_{m=1}^{M_F} q_m(n).$$

2.1. Upper bounds

We are interested in keeping the set of mean service times τ_m in our bounds instead of considering balanced networks. To do so, we shall first examine some relationships between mean queue lengths and mean service times. For proportional upper bounds, these relationships are stated in Lemmas 2.1–2.4. The bounds are then presented in two theorems and two corollaries. (Throughout this paper, proofs are postponed to Appendix A.)

Lemma 2.1. *The mean queue lengths of any two fixed-rate service centers satisfy the following inequality*:

$$\frac{q_i(n)}{q_j(n)} \leqslant \frac{\tau_i}{\tau_j} \quad \textit{for } i \leqslant j. \tag{4}$$

Lemma 2.2. *If*

$$q_j(n) \leqslant \frac{\tau_j}{L_F} Q_F(n), \quad j \leqslant M_F,$$

then

$$q_i(n) \leqslant \frac{\tau_i}{L_F} Q_F(n) \quad \textit{for all } i \textit{ such that } 1 \leqslant i \leqslant j. \tag{5}$$

Lemma 2.3. *If*

$$q_i(n) \geqslant \frac{\tau_i}{L_F} Q_F(n), \quad i \leqslant M_F,$$

then

$$q_j(n) \geqslant \frac{\tau_j}{L_F} Q_F(n) \quad \textit{for all } j \textit{ such that } i \leqslant j \leqslant M_F. \tag{6}$$

Lemma 2.4. *The mean queue lengths of the first and the last fixed-rate service centers satisfy the following inequalities*:

$$q_1(n) \leqslant \frac{\tau_1}{L_F} Q_F(n) \tag{7}$$

and

$$q_{M_F}(n) \geqslant \frac{\tau_{M_F}}{L_F} Q_F(n). \tag{8}$$

Theorem 2.5. *The network delay $D(n)$ and network throughput $T(n)$ satisfy the following inequalities*:

$$D(n) \geqslant L + \sum_{m=1}^{M_F} \frac{\tau_m^2}{L_F} [n - 1 - L_D \times T(n-1)] \tag{9}$$

and

$$T(n) \leqslant \frac{n}{L + \sum_{m=1}^{M_F} \frac{\tau_m^2}{L_F} [n - 1 - L_D \times T(n-1)]}. \tag{10}$$

Corollary 2.6. *For a network with no delay servers, the network throughput $T(n)$ is bounded above by*

$$n \bigg/ \left\{ L + \sum_{m=1}^{M_F} \frac{\tau_m^2}{L_F} [n-1] \right\},$$

which is smaller than (or equal to) the following balanced job bound in [25]:

$$\frac{n}{L + (L_F/M_F)[n-1]}.$$

For networks with one or more delay servers, the right-hand sides of equations (9) and (10) in Theorem 2.5 are functions of $T(n-1)$. Sequences of bounds for $D(n)$ and $T(n)$ can be obtained as follows [10].

Define

$$\overline{T}(n, 0) = \min\{n/L, 1/\tau_{M_F}\}$$

and

$$\underline{D}(n, 0) = \max\{L, n\tau_{M_F}\}$$

for all n. These are the asymptotic bounds. Next, define

$$\underline{D}(n, i) = \max\left\{ n\tau_{M_F},\ L + \sum_{m=1}^{M_F} \frac{\tau_m^2}{L_F} \times [n - 1 - L_D \times \overline{T}(n-1, i-1)] \right\} \tag{11}$$

and

$$\overline{T}(n, i) = n/\underline{D}(n, i) \tag{12}$$

for $1 \leqslant i \leqslant n$.

Theorem 2.7

$$D(n) \geqslant \underline{D}(n, i+1) \geqslant \underline{D}(n, i) \tag{13}$$

and

$$T(n) \leqslant \overline{T}(n, i+1) \leqslant \overline{T}(n, i) \tag{14}$$

for $0 \leqslant i \leqslant n-1$.

Corollary 2.8. *For each i, the proportional throughput upper bound in equation* (12) *is smaller than (or equal to) the corresponding balanced bound in* [10].

2.2. Lower bounds

The counterparts of Lemmas 2.1–2.4 for a proportional throughput lower bound are given below and the bound itself is presented in Theorem 2.13.

Lemma 2.9. *The mean queue lengths of any two fixed-rate service centers i and j with $i \leqslant j$ satisfy the following inequality*:

$$\frac{q_i(n)}{q_j(n)} \geqslant \left(\frac{\tau_i}{\tau_j}\right)^n \quad \text{for all } n \geqslant 1. \tag{15}$$

We next define

$$L_F^n = \sum_{m=1}^{M_F} \tau_m^n.$$

Lemma 2.10. *If*

$$q_j(n) \geqslant \frac{\tau_j^n}{L_F^n} Q_F(n), \quad 1 \leqslant j \leqslant M_F,$$

then

$$q_i(n) \geqslant \frac{\tau_i^n}{L_F^n} Q_F(n) \quad \text{for all } i \text{ such that } 1 \leqslant i \leqslant j. \tag{16}$$

Lemma 2.11. *If*

$$q_i(n) \leqslant \frac{\tau_i^n}{L_F^n} Q_F(n), \quad i \leqslant M_F,$$

then

$$q_j(n) \leqslant \frac{\tau_j^n}{L_F^n} Q_F(n) \quad \text{for all } j \text{ such that } i \leqslant j \leqslant M_F. \tag{17}$$

Lemma 2.12. *The mean queue lengths of the first and last fixed-rate service centers satisfy the following inequalities*:

$$q_1(n) \geqslant \frac{\tau_1^n}{L_F^n} Q_F(n) \tag{18}$$

and

$$q_{M_F}(n) \leqslant \frac{\tau_{M_F}^n}{L_F^n} Q_F(n). \tag{19}$$

Theorem 2.13. *The network delay $D(n)$ and network throughput $T(n)$ satisfy the following inequalities*:

$$D(n) \leqslant L + \frac{L_F^k}{L_F^{k-1}} [n - 1 - L_D \times T(n-1)], \quad \text{where } k = n, n+1, \dots, \tag{20}$$

and

$$T(n) \geqslant \frac{n}{L + \frac{L_F^k}{L_F^{k-1}} [n - 1 - L_D \times T(n-1)]} \quad \text{where } k = n, n-1, \dots. \tag{21}$$

Corollary 2.14. *For a network with no delay servers, the network throughput $T(n)$ is bounded below by*

$$\frac{n}{L + (L_F^n / L_F^{n-1})[n-1]},$$

which is larger than (or equal to) the following *balanced job bound in* [25]:

$$\frac{n}{L+\tau_{M_{\mathrm{F}}}[n-1]}.$$

For networks with dealy servers, the right-hand sides of equations (20) and (21) in Theorem 2.13 are functions of $T(n-1)$. Sequences of bounds for $D(n)$ and $T(n)$ can be obtained as follows [10].

Define $\underline{T}(n, 0)=0$ for all n. Let

$$\overline{D}(n, i)=L+\frac{L_{\mathrm{F}}^{n}}{L_{\mathrm{F}}^{n-1}} \times\left[n-1-L_{\mathrm{D}} \times \underline{T}(n-1, i-1)\right] \tag{22}$$

and

$$\underline{T}(n, i)=n / \overline{D}(n, i) \tag{23}$$

for $1 \leqslant i \leqslant n$.

Theorem 2.15

$$D(n) \leqslant \overline{D}(n, i+1) \leqslant \overline{D}(n, i), \quad 1 \leqslant i \leqslant n-1,$$

and

$$T(n) \geqslant \underline{T}(n, i+1) \geqslant \underline{T}(n, i), \quad 0 \leqslant i \leqslant n-1.$$

Corollary 2.16. *For each* i, *the proportional throughput lower bound in equation* (23) *is larger than* (*or equal to*) *the corresponding balanced bound in* [10].

Observation. When $n \rightarrow \infty$, the left-hand side of equation (21) converges to the right-hand side. The proportional bounds given in Theorem 2.15 (as well as the corresponding balanced bounds) are asymptotically exact.

2.3. Numerical examples

The following examples are taken from [10] to illustrate the accuracy of proportional bounds and of balanced bounds.

Example 2.17. The network has only fixed-rate service centers and is almost balanced, $\tau_1=0.08$, $\tau_2=0.09$ and $\tau_3=\tau_4=0.1$. The balanced job bounds and the proportional bounds are given in Table 1.

Table 1 [a]
Throughput bounds for an almost-balanced network with no delay server

Population size	Throughput bounds				
	$\underline{X}$	$\underline{T}$	Exact	$\overline{T}$	$\overline{X}$
2	4.255	4.317	4.317	4.317	4.324
5	6.494	6.660	6.715	6.729	0.757
10	7.874	8.022	8.206	8.27	8.316
20	8.811	8.867	9.168	9.338	9.401
30	9.174	9.194	9.499	9.759	9.828
40	9.368	9.376	9.654	9.984	10
60	9.569	9.570	9.792	10	10
80	9.674	9.674	9.853	10	10

[a] Where $\underline{X}$ is the balanced job lower bound, $\overline{X}$ is the balanced job upper bound, $\underline{T}$ is the proportional lower bound, and $\overline{T}$ is the proportional upper bound.

Notice that although the network is almost balanced, the proportional bounds are better than the balanced bounds. Also notice that the proportional bounds give the exact throughput when the population size is 2.

Example 2.18. The network of Example 2.17 is extended by a delay server with mean service time $\tau=1$. The first- and second-level balanced bounds of Kriz and proportional bounds are shown in Table 2.

Example 2.19. The network is unbalanced with no delay server. The mean service times at the four service centers are $\tau_1=0.04$, $\tau_2=0.05$, and $\tau_3=\tau_4=0.1$ (see Table 3).

Example 2.20. The network of Example 2.19 is extended by a delay server with mean service time $\tau=1$ (see Table 4).

2.4. Observations and discussions

Consider the following formula,

$$T_k(n)=\frac{n}{L+\dfrac{L_{\mathrm{F}}^{k}}{L_{\mathrm{F}}^{k-1}}\left[n-1-L_{\mathrm{D}} \times T(n-1)\right]},$$

$k=1,2, \ldots.$

This is a *unified formula* for the following throughput bounds:

Table 2 [a]
Throughput bonds for an almost-balanced network with one delay server

Population size	Throughput bounds								
	$\underline{Y}_1$	$\underline{Y}_2$	$\underline{T}_1$	$\underline{T}_2$	Exact	$\overline{T}_2$	$\overline{T}_1$	$\overline{Y}_2$	$\overline{Y}_1$
2	1.361	1.432	1.367	1.434	1.434	1.434	1.434	1.434	1.434
5	2.825	3.267	2.856	3.284	3.364	3.367	3.400	3.369	3.402
10	4.405	5.390	4.451	5.427	5.872	5.970	6.263	5.980	6.270
20	6.116	7.489	6.143	7.517	8.375	8.679	9.053	8.716	9.081
30	7.027	8.393	7.037	8.405	9.186	9.413	9.549	9.468	9.592
40	7.590	8.858	7.595	8.863	9.502	9.773	9.818	9.836	9.870
60	8.253	9.306	8.254	9.307	9.740	10	10	10	10
80	8.630	9.514	8.630	9.514	9.828	10	10	10	10

[a] Where T denotes proportional bounds and Y denotes Kriz's bounds.

Table 3
Throughput bounds for an unbalanced network with no delay server

Population size	Throughput bounds				
	$\underline{X}$	$\underline{T}$	Exact	$\overline{T}$	$\overline{X}$
2	5.128	5.360	5.360	5.360	5.517
5	7.246	7.341	7.803	8.033	8.621
10	8.403	8.407	8.930	9.635	10
15	8.876	8.876	9.302	10	10
20	9.132	9.132	9.483	10	10
30	9.404	9.404	9.659	10	10
40	9.547	9.547	9.746	10	10
60	9.693	9.693	9.831	10	10
80	9.768	9.768	9.874	10	10

- $T_1(n)$ is the balanced upper bound.
- $T_2(n)$ is the proportional upper bound.
- $T_n(n)$ is the proportional lower bound.
- $T_\infty(n)$ is the balanced lower bound.

And we have proved the following relations:

$$T_1(n) \geqslant T_2(n) \geqslant T(n) \geqslant T_n(n) \geqslant T_\infty(n).$$

The computational costs of proportional bounds are as follows. The proportional upper bound $T_2(n)$ requires $M_F + 4$ multiplications. The proportional lower bound $T_n(n)$ requires M_F exponentiations and $2M_F + 4$ multiplications (L_F^n and L_F^{n-1} can be calculated at the same time), or $(\log_2 n + 2)M_F + 4$ multiplications if exponentiation is not used. For comparison, each corresponding balanced bound requires only 4 multiplications. What do we buy with the additional computational cost? If we consider the throughput bounds as functions of network population size n, the additional parameters in proportional bounds give rise to much greater accuracy around the 'knee' of the throughput curves, as illustrated in the previous examples (Tables 1–4). On the other hand, the examples also show that for a large n, the upper bound is actually given by the asymptotic bound in each case. And since the lower bounds, both proportional and balanced, are asymptotically exact, they become close to each

Table 4
Throughput bounds for an unbalanced network with one delay server

Population size	Throughput bounds								
	$\underline{Y}_1$	$\underline{Y}_2$	$\underline{T}_1$	$\underline{T}_2$	Exact	$\overline{T}_2$	$\overline{T}_1$	$\overline{Y}_2$	$\overline{Y}_1$
2	1.439	1.524	1.457	1.528	1.528	1.528	1.528	1.531	1.531
5	2.959	3.476	2.974	3.484	3.628	3.635	3.664	3.667	3.690
10	4.556	5.684	4.567	5.685	6.422	6.586	6.858	6.743	6.960
15	5.576	6.978	5.576	6.979	8.133	8.836	9.245	9.139	9.494
20	6.270	7.676	6.270	7.767	8.945	9.703	9.814	10	10
30	7.160	8.618	7.160	8.618	9.483	10	10	10	10
40	7.707	9.042	7.707	9.042	9.659	10	10	10	10
60	8.345	9.437	8.345	9.437	9.797	10	10	10	10
80	8.705	9.614	8.705	9.614	9.856	10	10	10	10

other as n increases. Despite the above observation, proportional bounds are useful for some other reasons. First, for network design and optimization, it is very helpful to have closed-form mathematical expressions for network performance measures that include all the mean service times as parameters. (This is a significant advantage of the open queueing network model, as presented by Kleinrock, over the closed queueing network model for the design of communication networks [14].) Even though the MVA algorithm provides exact numerical solutions, it is not a closed-form mathematical formula. Second, if bounds are to be computed for several population sizes, the cost of computing L_F^n can easily be shared over all the bounds.

3. Generalized bounds

We next present algorithms which permit us to trade computation time for improved accuracy. Consider a single-chain network with M fixed-rate service centers, population size N, and a population sequence of S integers, $n_1, n_2, \ldots, n_S$, where $n_S = N$. Each algorithm begins by computing a bound for the network population size n_1. The algorithm then iterates over the population sequence using the bound computed for n_i to compute a bound for n_{i+1}, until a bound is computed for $n_S = N$. By not requiring the population sequence to consist of consecutive integers, this approach extends and subsumes the pioneering work of Eager and Sevcik [5]. We shall use proportional bounds for the initial population size n_1 in each of the algorithms presented below. However, balanced bounds can be used in place of the proportional bounds. We shall refer to bounds computed by these algorithms as generalized bounds. We show that generalized bounds have a nested property (to be defined below). We shall also present population sequences for computing the tightest upper and lower bounds that are optimal over all sequences of the same length. These properties of generalized bounds remain valid if balanced bounds replace proportional bounds in the algorithms.

3.1. Upper bounds

The following algorithm computes generalized throughput upper bounds. The population sequence is assumed to satisfy the following:

$1 < n_1 < n_2 < \cdots < n_S$ where $n_S = N$.

Algorithm 3.1—generalized_upper_bound;

```
begin
  max_throughput := 1/load[M];
  total_load := 0;
  for m := 1 to M do total_load := total_load + load[m];
  for m := 1 to M do ratio[m] := load[m]/total_load;
  for i := 1 to S do
    begin
      total_delay := 0;
      for m := 1 to M do
        begin
          delay[m] := load[m] * (1 + ratio[m] * (n[i] - 1));
          total_delay := total_delay + delay[m];
        end;
      throughput_upper := n[i]/total_delay;
      if throughput_upper > max_throughput
        then throughput_upper := max_throughput;
      for m := 1 to M do ratio[m] := delay[m]/total_delay;
    end;
end;
```

Before presenting several theorems stating some properties of the algorithm, we give three lemmas (Lemmas A.1–A.3) that form the basis of our algorithm. The lemmas, their proofs as well as proofs of the theorems can be found in the appendix. We shall use $r_j(n_i)$ to denote the value of ratio[j] when the population size is n_i during the execution of Algorithm 3.1.

For notational simplicity, these lemmas and theorems are stated and proved for networks with fixed-rate service centers only. Extension of our results to networks including delay service centers is straightforward. As in Section 2, the mean service times for the service centers have the following relation: $\tau_1 \leqslant \tau_2 \leqslant \cdots \leqslant \tau_M$.

Theorem 3.2. *For all i,* $1 \leqslant i \leqslant S$,

$$\sum_{m=1}^{M} \tau_m[1 + q_m(n_i - 1)] \geqslant \sum_{m=1}^{M} \tau_m[1 + r_m(n_{i-1}) \times (n_i - 1)], \qquad (24)$$

where $n_0 = 1$ *and* $r_m(1) = \tau_m/L$ *for* $m = 1, 2, \ldots, M$.

Equation (24) in Theorem 3.2 assures that Algorithm 3.1 computes lower bounds on delay and upper bounds on throughput.

Corollary 3.3. *If the population sequence is* $2, \ldots, N$ *then the algorithm computes the exact network throughput.*

In the next two theorems we present properties of the generalized throughput upper bounds. Given a population sequence $n_1, n_2, \ldots, n_S$ of S elements, a subsequence is said to be valid if it includes the population size n_S $(= N)$.

Applying Algorithm 3.1 to different population sequences yields different throughput upper bounds. These throughput upper bounds are said to be *nested* if the throughput upper bound computed from a population sequence is smaller than or equal to the throughput upper bound computed from any of the valid subsequences.

Theorem 3.4. *The generalized throughput upper bounds are nested.*

Theorem 3.5 (Optimal population sequence). *Given an integer* $S \leqslant N - 1$, *the population sequence of length S that yields the smallest throughput upper bound is the sequence* $N - S + 1$, $N - S + 2, \ldots, N$.

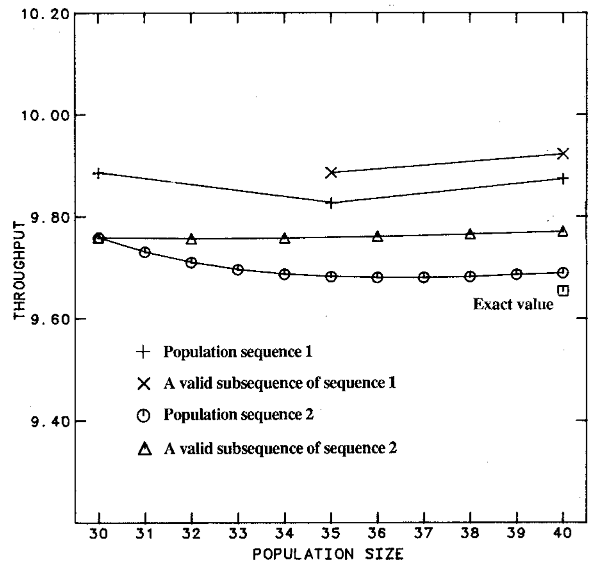


Fig. 1. Nested property of generalized throughput upper bounds.

We calculated generalized throughput upper bounds for the network considered earlier in Ex-

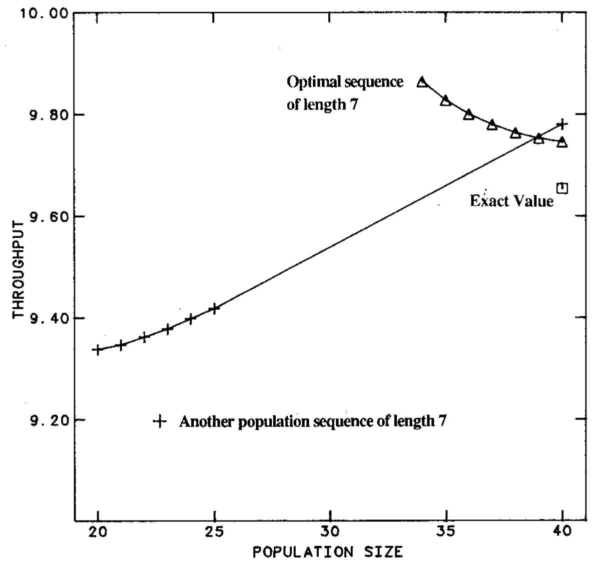


Fig. 2. Optimal population sequence of length 7 for generalized throughput upper bounds.

ample 2.17. The results are plotted in Figs. 1 and 2. Fig. 1 illustrates the nested property. The population sequence 35, 40 is a valid subsequence of 30, 35, 40. The population sequence 30, 32,...,40 is a valid subsequence of 30, 31,...,39, 40. Fig. 2 illustrates throughput upper bounds of two sequences of length 7. The optimal sequence of length 7 is 34, 35,...,40.

A slightly modified algorithm which can handle delay servers is presented in Algorithm 3.6 below. Service centers 1 to MF are fixed-rate servers and service centers $MF+1$ to M are delay servers. For convenience, $n[0]$ is set to 1.

Algorithm 3.6—Generalized_upper_bound_delay_server;

```
begin
  max_throughput := 1/load[$M$];
  throughput_upper := max_throughput;
  load_fixed := 0;
  for $m$ := 1 to $MF$ do load_fixed := load_fixed + load[$m$];
  for $m$ := 1 to $MF$ do ratio[$m$] := load[$m$]/load_fixed;
  for $i$ := 1 to $S$ do
    begin
      total_delay := 0;
      queue_$F$ := $n[i] - 1$;
      for $m$ := $MF + 1$ to $M$ do
        begin
          total_delay := total_delay + load[$m$];
          queue_$F$ := queue_$F$ − load[$m$] * throughput_upper * $(n[i]-1)/n[i-1]$;
        end;
      for $m$ =: 1 to $M$ do
        begin
          delay[$m$] := load[$m$] * (1 + ratio[$m$] * queue_$F$);
          total_delay := total_delay + delay[$m$];
        end;
      throughput_upper := $n[i]$/total_delay;
      if throughput_upper > max_throughput
        then throughput_upper := max_throughput;
      for $m$ := 1 to $M$ do ratio[$m$] := delay[$m$]/total_delay;
    end;
end;
```

3.2. Lower bounds

The following algorithm computes generalized throughput lower bounds. The population sequence is assumed to satisfy the following:

$n_1 \geqslant n_2 \geqslant \cdots \geqslant n_S = N$ where $N \geqslant 3$.

In what follows, $r_m(n_i)$ denotes the value of ratio[m] when the population size is n_i during the execution of Algorithm 3.7 (*see on top of page* 13).

Theorem 3.8. *For all i*, $1 \leqslant i \leqslant S$,

$$\sum_{m=1}^{M} \tau_m\left[1 + q_m(n_i - 1)\right] \leqslant \sum_{m=1}^{M} \tau_m\left[1 + r_m(n_{i-1}) \times (n_i - 1)\right], \qquad (25)$$

where $n_0 = n_1 - 1$ *and* $r_m(n_0) = \tau_m^{n_0}/L_F^{n_0}$.

Theorem 3.8 assures that Algorithm 3.7 com-

Algorithm 3.7—Generalized_lower_bound;

```
begin
  sum := 0;
  for $m$ := 1 to $M$ do
    begin
      ratio[$m$] := load[$m$] ** ($n$[1] − 1);
      sum := sum + ratio[$m$];
    end;
  for $m$ := 1 to $M$ do ratio[$m$] := ratio[$m$]/sum;
  for $i$ := 1 to $S$ do
    begin
      total_delay := 0;
      for $m$ := 1 to $M$ do
        begin
          delay[$m$] := load[$m$] * (1 + ratio[$m$] * ($n$[$i$] − 1));
          total_delay := total_delay + delay[$m$];
        end;
        throughput_lower := $n$[$i$]/total_delay;
      for $m$ := 1 to $M$ do ratio[$m$] := delay[$m$]/total_delay;
    end;
end;
```

putes upper bounds on delay and lower bounds on throughput. These bounds also have properties similar to those shown earlier for generalized throughput upper bounds.

Consider a population sequence $n_1, n_2, \ldots, n_S = N$. For the purpose of computing generalized throughput lower bounds, a subsequence is said to be valid if it contains both n_1 and n_S. Generalized throughput lower bounds are said to be *nested* if the throughput lower bound computed from a population sequence is larger than or equal to the throughput lower bound computed from any of the valid population subsequences.

Theorem 3.9. *The generalized throughput lower bounds are nested.*

Theorem 3.10 (Optimal population sequence). *Given an integer S, the population sequence that yields the largest throughout lower bound among all population sequences of length S is $N, N, \ldots, N$.*

We calculated generalized throughput lower bounds for the network considered earlier in Example 2.17. Fig. 3 illustrates the nested property. The population sequence 50, 40 is a valid subsequence of 50, 45, 40. The sequence 50, 48, ..., 42, 40 is a valid subsequence of 50, 49, ..., 41, 40. Fig. 4 illustrates throughput lower bounds computed

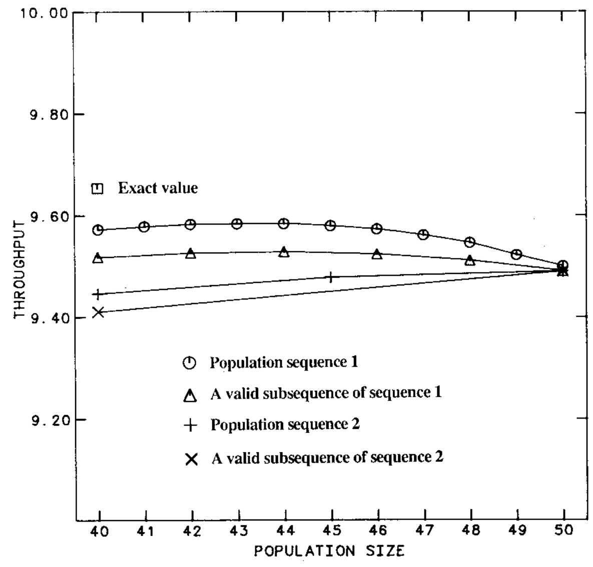


Fig. 3. Nested property of generalized throughput lower bounds.

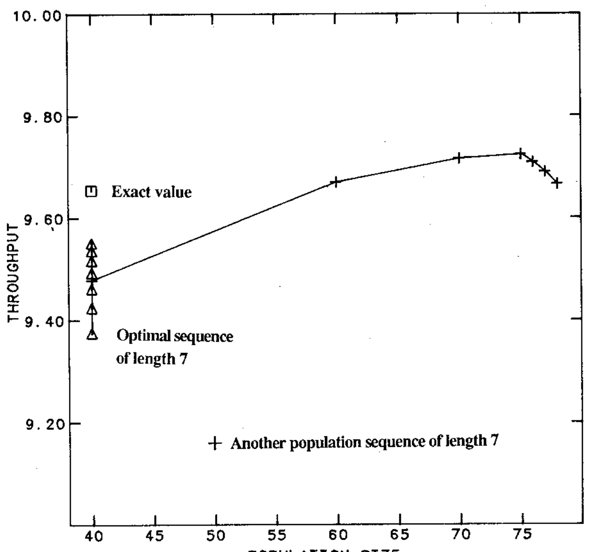


Fig. 4. Optimal population sequence of length 7 for generalized throughput lower bounds.

for two sequences of length 7, including the optimal one.

A slightly modified algorithm which can handle delay servers is presented in Algorithm 3.11 below. For convenience, $n[0]$ is set to 1.

3.3. Observations and discussions

We emphasize that balanced bounds can be used instead of proportional bounds in the initial steps of Algorithms 3.1, 3.6, 3.7 and 3.11. The use of proportional bounds gives rise to more accurate generalized bounds but incurs more computational cost for the initial step of each algorithm than using balanced bounds. The computational costs for the initial proportional bounds are M multiplications and M exponentiations for Algorithms 3.1 and 3.7 respectively. The computational cost of each iteration is $3M+1$ multiplications for each algorithm.

Algorithm 3.11—Generalized_lower_bound_delay_server;

```
begin
  sum := 0;
  throughput_lower := 0;
  for m := 1 to MF do
    begin
      ratio[m] := load[m] * * (n[1] - 1);
      sum := sum + ratio[m];
    end;
  for m := 1 to MF do ratio[m] := ratio[m]/sum;
  for i := 1 to S do
    begin
      total_delay := 0;
      for m := MF + 1 to M do
        begin
          total_delay := total_delay + load[m];
          queue_F := queue_F - load[m] * throughput_lower * (n[i] - 1)/n[i - 1];
        end;
      for m := 1 to MF do
        begin
          delay[m] := load[m] * (1 + ratio[m] * queue_F);
          total_delay := total_delay + delay[m];
        end;
        throughput_lower := n[i]/total_delay;
      for m := 1 to M do ratio[m] := delay[m]/total_delay;
    end;
end;
```

Generalized bound algorithms are especially useful if bounds for many different population sizes are needed. In particular, the additional computational cost of using proportional bounds in the initial step is a fixed cost.

The generalized lower bound algorithm for an optimal population sequence is similar to the Schweitzer approximate solution technique [20]. Our results show that for single-chain networks, the Schweitzer heuristic provides a lower bound (if initialized with a lower bound) rather than merely an approximate solution.

4. Bounds for closed multichain networks

We next present performance bounds for closed multichain networks. Algorithms for computing such bounds are given in Section 4.1. In the development of these algorithms, we were motivated by models of communication networks that typically have numerous routing chains. In the computation of performance bounds for multichain networks, we improve their accuracy by making use of routing information and exploiting the sparseness of routes.

For multichain networks, we shall use the same notation as for single-chain networks except that an additional subscript, h or k, is used to denote a specific chain.

Theorem 4.1. *The mean delay $D_k(\boldsymbol{n})$ of chain k customers satisfies the following inequalities*

$$\underline{D}_k(\boldsymbol{n}) \leqslant D_k(\boldsymbol{n}) \leqslant \overline{D}_k(\boldsymbol{n}),$$

where

$$\begin{aligned}\underline{D}_k(\boldsymbol{n}) = L_k + \sum_{\substack{m=1\\ m\text{ in chain }k}}^{M_F} \tau_{mk} &\times \sum_{\substack{h=1\\ \text{chain }h\text{ visits }m}}^{K} \tau_{mh}\underline{T}_h(\boldsymbol{n}-\mathbf{1}_k)\\ &+\tau_{\min,k}\big[n_k - 1 - L_{\mathrm{D},k}\overline{T}_k(\boldsymbol{n}-\mathbf{1}_k)\\ &\quad - L_{\mathrm{F},k}\underline{T}_k(\boldsymbol{n}-\mathbf{1}_k)\big] \qquad (26)\end{aligned}$$

and

$$\begin{aligned}\overline{D}_k(\boldsymbol{n}) = L_k + \sum_{\substack{m=1\\ m\text{ in chain }k}}^{M_F} \tau_{mk} &\times \sum_{\substack{h=1\\ \text{chain }h\text{ visits }m}}^{K} \tau_{mh}\underline{T}_h(\boldsymbol{n}-\mathbf{1}_k)\\ &+ \sum_{\substack{h=1\\ \text{chain }h\\ \text{intersects chain }k}}^{K} \tau_{\max,h,k}\\ &\times\big[n_h - L_h\underline{T}_h(\boldsymbol{n}-\mathbf{1}_k)\big] - \tau_{\max,k,k}, \qquad (27)\end{aligned}$$

where

- $\tau_{\max,h,k}$ *is the maximum mean service time among the fixed-rate service centers traversed by both chain h and chain k customers,*
- $\tau_{\min,k}$ *is the minimum mean service time among the fixed-rate service centers traversed by chain k customers,*
- $\underline{T}_h(\boldsymbol{n}-\mathbf{1}_k)$ *is a lower bound of* $T_h(\boldsymbol{n}-\mathbf{1}_k)$, *and*
- $\overline{T}_k(\boldsymbol{n}-\mathbf{1}_k)$ *is an upper bound of* $T_k(\boldsymbol{n}-\mathbf{1}_k)$.

Chain h is said to ***intersect*** *chain k if it visits a fixed-rate service center that is also visited by chain k.*

Corollary 4.2. *The throughput $T_k(\boldsymbol{n})$ of chain k satisfies the following inequalities*:

$$\underline{T}_k(\boldsymbol{n}) \leqslant T_k(\boldsymbol{n}) \leqslant \overline{T}_k(\boldsymbol{n}), \qquad (28)$$

where

$$\underline{T}_k(\boldsymbol{n}) = n_k/\overline{D}_k(\boldsymbol{n}) \qquad (29)$$

and

$$\overline{T}_k(\boldsymbol{n}) = n_k/\underline{D}_k(\boldsymbol{n}). \qquad (30)$$

4.1. Algorithms

The procedure to compute throughput bounds involves the following steps:

(i) Find fast lower bounds of $T_h(\boldsymbol{n}-\mathbf{1}_k)$ for all h, $h = 1, 2, \ldots, K$ and fast upper bounds of $T_k(\boldsymbol{n}-\mathbf{1}_k)$ for all $k = 1, 2, \ldots, K$.

(ii) Plug the fast upper bounds of $T_k(\boldsymbol{n}-\mathbf{1}_k)$ into equation (26) and the fast lower bounds of $T_k(\boldsymbol{n}-\mathbf{1}_k)$ and $T_h(\boldsymbol{n}-\mathbf{1}_k)$ into equations (26) and (27) to calculate bounds of mean delay. Apply equations (29) and (30) to obtain throughput bounds.

Some of the variables used in the algorithms are defined in the following. The meaning of other variables is self-explanatory.

- load_total[k] is the sum of mean service times of chain k at fixed-rate and delay service centers,
- load_max[h, k] is the largest mean service time for chain k among all fixed-rate service centers visited by both chain h and chain k,
- visit_common_queue[h, i] is true if chain h and chain i visit at least one common fixed-rate service center,
- load[k, m] is the mean service time of chain k at service center m,
- load_D[k] is the sum of mean service times at delay centers for chain k,
- load_F[k] is the sum of mean service times at fixed-rate centers for chain k.
- load_min[k] is the smallest mean service time for chain k among fixed-rate service centers visited by chain k, and
- visit[h, m] is true if service center m is visited by chain h.

Algorithm 4.3, given below, finds a fast lower bound of $T_h(\boldsymbol{n} - \mathbf{1}_k)$. It utilizes some routing information.

Algorithm 4.3—Fast_throughput_lower_bounds;

```
begin
  for k := 1 to num_chains do
  begin
    population[k] := population[k] - 1;
    for h := 1 to num_chains do
      if population[h] > 0
      then
      begin
        delay := (population[h] - 1) * load_max[h, h];
        for i := 1 to num_chains do
        if visit_common_queue[h, i] and (h ≠ i)
        then delay := delay + population[i] * load_max[h, i];
        delay := delay + load_total[h];
        throughput_lower[h, k] := population[h]/delay;
      end
      else throughput_lower[h, k] := 0;
    population[k] := population[k] + 1;
  end;
end;
```

The above procedure is actually a special case of Algorithm 4.4 below. Its throughput bound of $T_h(\boldsymbol{n} - \mathbf{1}_k)$ is obtained by replacing all throughput lower bounds in equation (27) with zero.

Algorithm 4.3 calculates throughput lower bounds only. There are two methods to obtain fast throughput upper bounds of $T_k(\boldsymbol{n} - \mathbf{1}_k)$. First, we can use BJB upper bounds for a multichain network [25]. Second, we can consider a network in which all chains, except chain k, are removed and use the proportional upper bound for such a single-chain network. Note that fast throughput upper bounds are needed only if the network has delay service centers.

The second procedure, to be given next, uses the fast bounds described above to calculate improved bounds for mean delays. It then applies Little's formula to obtain bounds for $T_k(\boldsymbol{n})$, for all $k = 1, 2, \ldots, K$. The computation sequence follows equations (26) and (27) exactly.

Algorithm 4.4—Multichain_throughput_bounds;

```
begin
  for k := 1 to num_chains do
  begin
    population[k] := population[k] − 1;
      /* remove one chain k customer */
    min_delay := load_total[k];
    delay_others := 0;
    for m := 1 to num_queues do
    if visit[k, m]
    then
      begin
        queue := 0;
        for h := 1 to num_chains do if visit[h, m]
        then queue := queue + load[h, m] * throughput_lower[h, k];
        min_delay := min_delay + load[k, m] * queue;
      end;
    for h := 1 to num_chains do
    if visit_common_queue[h, k]
    then
      begin
        queue_others := population[h] − load_total[h] * throughput_lower[h, k];
        delay_others := delay_others + load_max[h, k] * queue_others;
      end;
    population[k] := population[k] + 1;
    queue_others := population[k] − load_D[k] * throughput_upper[k]
                    − load_F[k] * throughput_lower[k, k];
    delay_lower := min_delay + load_min[k] * queue_others;
    delay_upper := min_delay + delay_others;
    final_throughput_upper[k] := population[k]/delay_lower;
    bottleneck := min(1/load_max[k, k], population[k]/load_total[k]);
    if final_throughput_upper[k] > bottleneck
    then final_throughput_upper[k] := bottleneck;
    final_throughput_lower[k] := population[k]/delay_upper;
  end;
end;
```

4.2. Numerical examples

The first network used is a 26-node network with 32 full-duplex communication links and 32 virtual channels. The window size (chain population size) is 2 for each virtual channel. Because we assume full-duplex virtual channels with symmetric traffic, the network reduces to a queueing network model with 32 fixed-rate service centers and 16 closed chains. Additionally, we employ 16 fixed-rate servers, one for each closed chain, to model the external sources of virtual channels. The mean service time for each service center is 0.1 s and is the same for all virtual channels. The mean service time at the source servers is 1.0 s. Table 5 below shows the routes for the 16 virtual channels. The calculated throughput bounds and actual throughput of each of the 16 chains are shown in Fig. 5. The maximum, minimum, and average utilizations of the 31 fixed-rate service

Table 5
Routes of virtual channels

VC	Route (in node sequence)
1	16 17 18 19 4 5
2	1 2 3 17 18 19
3	6 25 22 23 24 26
4	24 10 11 12 13
5	13 1 2 3 4
6	17 3 4 5 6
7	1 2 3 4 5 6 7 8 9
8	1 13 12
9	9 10 24
10	21 20 25
11	15 16 17
12	23 24
13	21 22 25 6 7 8
14	23 22 21 20
15	21 15 14 13 12 11 10 9
16	1 13 14 15 21 22 25

centers actually used in the network are 0.272, 0.0895, and 0.193 respectively. (One service center has zero utilization and was excluded.) In this case, the network is lightly loaded.

In Fig. 6, for the same network, the mean service time at each source server is set to 0.1 s. The maximum, minimum, and average utilizations of the 31 communication channels are 0.753, 0.212, and 0.519, respectively. This represents a fairly heavily loaded network.

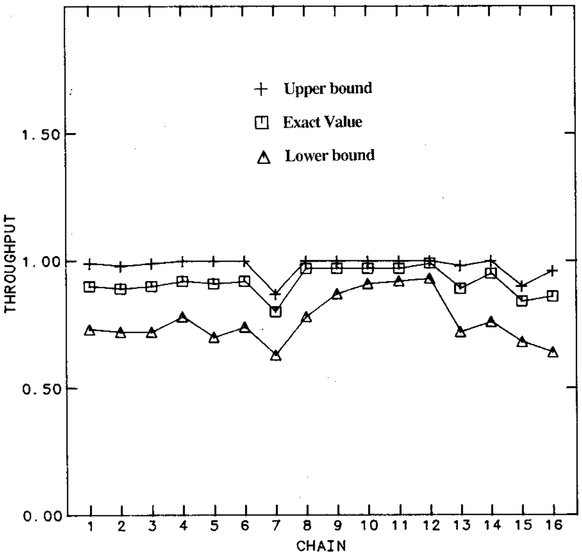


Fig. 5. Throughputs and throughput bounds of individual chains in the first network example (low utilization).

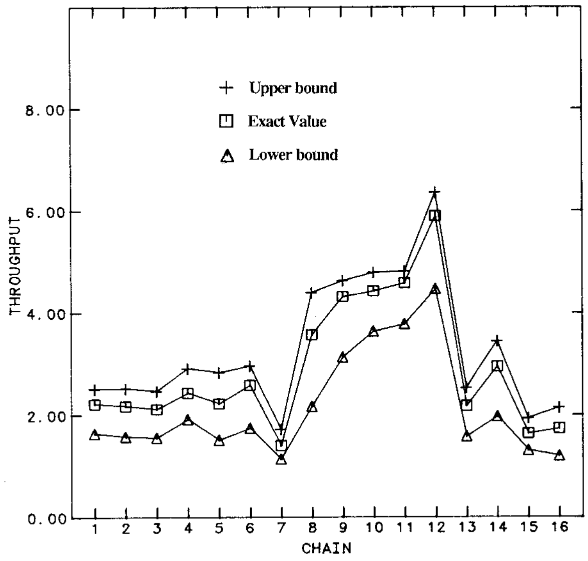


Fig. 6. Throughputs and throughput bounds of individual chains in the first network example (high utilization).

The second network used in our numerical study is a randomly generated network with 12 nodes, 30 virtual channels and 34 communication channels. The communication channels and their mean service times are shown in Table 6. The notation (i, j) in Table 6 denotes a communication channel from node i to node j. The route, window size, and mean service time of the source server for each virtual channel are given in Table 7. The maximum, minimum, and average utilizations of the 32 communication channels with nonzero utilizations are 0.998, 0.101, and 0.517 respectively. Because of symmetric traffic, only the results of 15 virtual channels are shown in Fig. 7.

From Fig. 7 and the table on routes, we observe that if a virtual channel does not interact much with other virtual channels, then its throughput bounds are tight and the exact value is close to the upper bound. On the other hand, if a virtual channel interacts significantly with many other virtual channels then its throughput bounds are not so tight and the exact throughput is closer to the lower bound than the upper bound. Notice that virtual channel 3 does not intersect any other virtual channel; both its upper bound and its lower bound obtained are equal to the exact throughput. Thus the tightness of the throughput bounds presented in this section is affected by the degree of sparseness of routes in a network.

Table 6
Mean service times of communication channels in the second network example

Communication channel	Mean service time (sec)
(9, 4)	0.200
(2, 7)	0.200
(3, 11)	0.200
(5, 12)	0.050
(6, 10)	0.025
(9, 7)	0.200
(1, 2)	0.050
(3, 1)	0.050
(5, 11)	0.200
(6, 8)	0.200
(9, 2)	0.100
(10, 8)	0.100
(12, 3)	0.050
(4, 7)	0.025
(5, 9)	0.200
(6, 12)	0.200
(10, 7)	0.100
(4, 9)	0.200
(7, 2)	0.200
(11, 3)	0.200
(12, 5)	0.050
(10, 6)	0.025
(7, 9)	0.200
(2, 1)	0.050
(1, 3)	0.050
(11, 5)	0.200
(8, 6)	0.200
(2, 9)	0.100
(8, 10)	0.100
(3, 12)	0.050
(7, 4)	0.025
(9, 5)	0.200
(12, 6)	0.200
(7, 10)	0.100

Table 7
Routes, window sizes and mean service time of source servers for virtual channels in the second network example

Virtual channel	Route (in node sequence)	Window size	Mean service time of source server (sec)
1	11 5	2	0.10
2	2 7 4	2	0.20
3	8 6	2	0.10
4	11 3 1 2 9 4	3	0.30
5	10 6 12	2	0.30
6	6 10 8	3	0.20
7	9 2 7 4	2	0.30
8	8 10 7 9 4	2	0.10
9	4 7 9 2 1 3	2	0.10
10	5 12 3 1 2 7 9 4	2	0.20
11	2 9	2	0.30
12	3 11 5	2	0.30
13	3 12 5	2	0.30
14	7 2	2	0.10
15	10 7 9	2	0.10
16	5 11	2	0.10
17	4 7 2	2	0.20
18	6 8	2	0.10
19	4 9 2 1 3 11	3	0.30
20	12 6 10	2	0.30
21	8 10 6	3	0.20
22	4 7 2 9	2	0.30
23	4 9 7 10 8	2	0.10
24	3 1 2 9 7 4	2	0.10
25	4 9 7 2 1 3 12 5	2	0.20
26	9 2	2	0.30
27	5 11 3	2	0.30
28	5 12 3	2	0.30
29	2 7	2	0.10
30	9 7 10	2	0.10

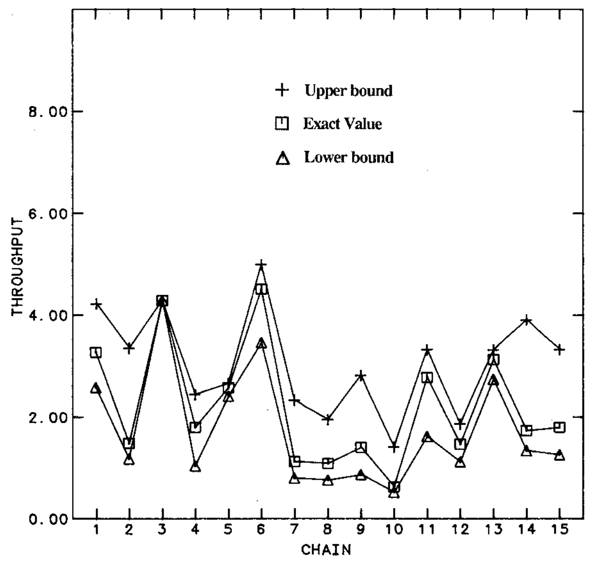


Fig. 7. Throughputs and throughput bounds of individual chains in the second network example.

4.3. Discussions

The maximum number of operations required by Algorithm 4.3 is $K^2(K+1)$ multiplications. Algorithm 4.4 requires a maximum of $K(M+2K+6)$ multiplications to calculate the upper and lower bounds for each chain. The actual computational cost of Algorithm 4.3 or Algorithm 4.4 depends upon specific routes, and is smaller than the maximum number for a sparse network. Our algorithms are similar to the level 2 bounds of Eager and Sevcik published recently [6]. Our algorithms, however, were developed and investigated independently [8]. Also, our study of performance bounds for networks with large numbers of sparse routine chains appears to be unique.

The composite bound method of Kerola [9] calculates upper bounds only assuming that the lower bounds are known. His bounds are derived for saturated networks and can be viewed as extensions of asymptotic bounds [17]. An asymptotic expansion algorithm for computing performance bounds was presented by McKenna and Mitra [16]. This algorithm gives bounds that are better than other bounds for networks where the loads at delay service centers are much higher than loads at fixed-rate service centers. But it is required that each chain visits a delay service center and there are also restrictions on population sizes and loads.

Acknowledgment

This paper has greatly benefitted from the constructive criticisms of the anonymous reviewers.

Appendix A. Proofs of lemmas, theorems, and corollaries

Proof of Lemma 2.1. By induction.

(1) *Base case*: $q_i(1)/q_j(1) = \tau_i/\tau_j$. Therefore it is true for $n = 1$.

(2) *Induction step*: Assume it is true for $n = k$, i.e.,

$$q_i(k)/q_j(k) \leqslant \tau_i/\tau_j.$$

From the MVA formula,

$$\frac{q_i(k+1)}{q_j(k+1)} = \frac{D_i(k+1)T(k+1)}{D_j(k+1)T(k+1)} = \frac{D_i(k+1)}{D_j(k+1)} = \frac{\tau_i(1+q_i(k))}{\tau_j(1+q_j(k))} \leqslant \frac{\tau_i}{\tau_j}$$

$$\big(\text{because } q_i(k) \leqslant q_j(k)\big).$$

Therefore, it is also true for $n = k + 1$. □

Proof of Lemma 2.2. If $q_j(n) \leqslant (\tau_j/L_{\mathrm{F}})Q_{\mathrm{F}}(n)$, then from Lemma 2.1, for any $i \leqslant j$

$$q_i(n) \leqslant \frac{\tau_i}{\tau_j} q_j(n) \leqslant \frac{\tau_i}{\tau_j}\frac{\tau_j}{L_{\mathrm{F}}} Q_{\mathrm{F}}(n) = \frac{\tau_i}{L_{\mathrm{F}}} Q_{\mathrm{F}}(n). \qquad \square$$

Proof of Lemma 2.3. Similar to the proof of Lemma 2.2. □

Proof of Lemma 2.4. From Lemma 2.3,

$$q_m(n) \geqslant \frac{\tau_m}{\tau_1} q_1(n) \quad \text{for } m = 1, 2, \ldots, M_{\mathrm{F}}.$$

We then have

$$Q_{\mathrm{F}}(n) \geqslant \sum_{m=1}^{M_{\mathrm{F}}} \frac{\tau_m}{\tau_1} q_1(n) = \frac{L_{\mathrm{F}}}{\tau_1} q_1(n).$$

Hence, we have

$$q_1(n) \leqslant \frac{\tau_1}{L_{\mathrm{F}}} Q_{\mathrm{F}}(n).$$

Similarly,

$$q_{M_{\mathrm{F}}}(n) \geqslant \frac{\tau_{M_F}}{L_F} Q_{\mathrm{F}}(n). \qquad \square$$

Proof of Theorem 2.5. From equation (1), we have

$$D(n) = \sum_{m=1}^{M} D_m(n) = \sum_{m=M_F+1}^{M} \tau_m + \sum_{m=1}^{M_F} \tau_m(1 + q_m(n-1))$$

$$= \sum_{m=M_F+1}^{M} \tau_m + \sum_{m=1}^{M_F} \tau_m + \sum_{M=1}^{M_F} \tau_m q_m(n-1)$$

$$= \sum_{m=1}^{M} \tau_m + \left(\tau_1 q_1(n-1) + \tau_2 q_2(n-1) + \cdots + \tau_{M_F} q_{M_F}(n-1)\right)$$

$$= L + \left[\left(\frac{\tau_1^2}{L_F} + \frac{\tau_2^2}{L_F} + \cdots + \frac{\tau_{M_F}}{L_F}\right) \times (n - 1 - L_D \times T(n-1))\right] + \Delta,$$

where

$$\Delta = \tau_1\left[q_1(n-1) - \frac{\tau_1}{L_F}(n - 1 - L_D \times T(n-1))\right]$$

$$+ \tau_2\left[q_2(n-1) - \frac{\tau_2}{L_F}(n - 1 - L_D \times T(n-1))\right]$$

$$+ \cdots + \tau_{M_F}\left[q_{M_F}(n-1) - \frac{\tau_{M_F}}{L_F}(n - 1 - L_D \times T(n-1))\right].$$

From Lemmas 2.2, 2.3, and 2.4, there exists an m, $1 \leqslant m \leqslant M_F$ such that

$$q_i(n-1) \leqslant \frac{\tau_i}{L_F}(n - 1 - L_D \times T(n-1)) \quad \text{for all } 1 \leqslant i \leqslant m$$

and

$$q_i(n-1) \geqslant \frac{\tau_i}{L_F}(n - 1 - L_D \times T(n-1)) \quad \text{for all } m < i \leqslant M_F.$$

Therefore, we have $\Delta = \Delta_1 - \Delta_2$ where

$$\Delta_1 = \sum_{i=m+1}^{M_F} \tau_i\left[q_i(n-1) - \frac{\tau_i}{L_F}(n - 1 - L_D \times T(n-1))\right]$$

and

$$\Delta_2 = \sum_{i=1}^{m} \tau_i\left[\frac{\tau_i}{L_F}(n - 1 - L_D \times T(n-1)) - q_i(n-1)\right].$$

Replacing $(n - 1 - L_D \times T(n-1))$ with $\sum_{j=1}^{M_F} q_j(n-1)$, we have

$$\Delta_1 = \sum_{i=m+1}^{M_F} \tau_i\left[q_i(n-1) - \frac{\tau_i}{L_F}\sum_{j=1}^{M_F} q_j(n-1)\right]$$

$$\geqslant \tau_{m+1}\left[\sum_{i=m+1}^{M_F} q_i(n-1) - \sum_{i=m+1}^{M_F} \frac{\tau_i}{L_F}\sum_{j=1}^{M_F} q_j(n-1)\right] \tag{A.1}$$

and

$$\Delta_2 = \sum_{i=1}^{m} \tau_i\left[\frac{\tau_i}{L_F}\sum_{j=1}^{M_F} q_j(n-1) - q_i(n-1)\right] \leqslant \tau_m\left[\sum_{i=1}^{m} \frac{\tau_i}{L_F}\sum_{j=1}^{M_F} q_j(n-1) - \sum_{i=1}^{m} q_i(n-1)\right]. \tag{A.2}$$

The expressions inside the brackets of the right-hand sides of equations (A.1) and (A.2) are equal because

$$\sum_{i=1}^{m} q_i(n-1) + \sum_{i=m+1}^{M_F} q_i(n-1)$$
$$= \sum_{i=1}^{m} \frac{\tau_i}{L_F} \sum_{j=1}^{M_F} q_j(n-1) + \sum_{i=m+1}^{M_F} \frac{\tau_i}{L_F} \sum_{j=1}^{M_F} q_j(n-1) = \sum_{j=1}^{M_F} q_j(n-1).$$

Hence, $\Delta_1 \geqslant \Delta_2$. Therefore,

$$D(n) \geqslant L + \sum_{m=1}^{M_F} \tau_m^2 [n-1-L_D \times T(n-1)]/L_F. \qquad \square$$

Proof of Theorem 2.7. When $n=1$ it is clearly true. Therefore we only have to prove the theorem for $n>1$. The proof can be divided into two parts.

(1) $D(n) \geqslant \underline{D}(n, i)$ and $T(n) \leqslant \overline{T}(n, i)$ for $0 \leqslant i \leqslant n-1$. This can be proved by induction on i and by using Theorem 2.5.

(2) $\underline{D}(n, i+1)$ and $\overline{T}(n, i+1) \leqslant \overline{T}(n, i)$ for $0 \leqslant i \leqslant n-1$. This is proved by induction as follows:
  (i) When $i=1$, $\underline{D}(n, 1) \geqslant \underline{D}(n, 0)$ and $\overline{T}(n, 1) \leqslant \overline{T}(n, 0)$ from equations (11) and (12).
  (ii) Assume that it is true for $i=k$; then from Theorem 2.5 it is also true for $i=k+1$. $\square$

Proof of Corollaries 2.6 and 2.8. Compare the balanced delay bound of Kriz (which generalizes the balanced job bound)

$$L + \frac{L_F}{M_F}[n-1-L_D \times \overline{T}(n-1, i-1)]$$

with the corresponding proportional bound

$$L + \sum_{m=1}^{M_F} \frac{\tau_m^2}{L_F}[n-1-L_D \times \overline{T}(n-1, i-1)].$$

For the proportional bound to the tighter, it is sufficient to show

$$\sum_{m=1}^{M_F} \tau_m^2/L_F \leqslant L_F/M_F,$$

which is true by virtue of the Chebyshev inequality [1], and then apply induction on i. $\square$

Proof of Lemma 2.9. By induction.

(1) *Base case*: $q_i(1)/q_j(1) = \tau_i/\tau_j$. Therefore it is true for $n=1$.

(2) *Induction step*: Assume it is true for $n=k$, i.e.,

$$q_i(k)/q_j(k) \geqslant (\tau_i/\tau_j)^k.$$

From the MVA formula,

$$\frac{q_i(k+1)}{q_j(k+1)} = \frac{D_i(k+1)T(k+1)}{D_j(k+1)(k+1)} = \frac{D_i(k+1)}{D_j(k+1)} = \frac{\tau_i(1+q_i(k))}{\tau_j(1+q_j(k))}$$
$$\geqslant \frac{\tau_i}{\tau_j}\left(\frac{q_i(k)}{q_j(k)}\right) \quad (\text{because } q_i(k) \leqslant q_j(k))$$
$$\geqslant (\tau_i/\tau_j)^{k+1}.$$

Therefore it is also true for $n=k+1$. $\square$

Proofs of Lemmas 2.10–2.12 and Theorems 2.13 and 2.15 are similar to those of Lemmas 2.2–2.4 and Theorems 2.5 and 2.7 and are therefore omitted.

Proof of Corollaries 2.14 and 2.16. Compare the balanced delay upper bound of Kriz (which generalizes the balanced job bound)

$$L = \tau_{M_F}\left[n-1-L_D \times \underline{T}(n-1,\, i-1)\right]$$

with the corresponding proportional bound

$$L + \frac{L_F^n}{L_F^{n-1}}\left[n-1-L_D \times \underline{T}(n-1,\, i-1)\right].$$

For the proportional bound to be tighter, it is sufficient to show that

$$L_F^n / L_F^{n-1} \leqslant \tau_{M_F},$$

which is true by the fact that $\tau_j \leqslant \tau_{M_F}$ for all $j \leqslant M_F$. □

The following lemma is from [5].

Lemma A.1. *For any i, $1 \leqslant i \leqslant S$, and all j, $1 \leqslant j \leqslant M$, if*

$$r_j(n_{i-1}) \Big/ \sum_{m=j}^{M} r_m(n_{i-1}) \geqslant q_j(n_i - 1) \Big/ \sum_{m=j}^{M} q_m(n_i - 1),$$

then

$$\sum_{m=j}^{M} q_m(n_i - 1) \geqslant (n_i - 1) \sum_{m=j}^{M} r_m(n_{i-1}),$$

$$\left[\sum_{m=j}^{M} \tau_m q_m(n_i - 1)\right] \Big/ \left[\sum_{m=j}^{M} q_m(n_i - 1)\right] \geqslant \left[\sum_{m=j}^{M} \tau_m r_m(n_{i-1})\right] \Big/ \left[\sum_{m=j}^{M} r_m(n_{i-1})\right],$$

and

$$r_j(n_i) \Big/ \sum_{m=j}^{M} r_m(n_i) \geqslant q_j(n_i) \Big/ \sum_{m=j}^{M} q_m(n_i).$$

The proof of this lemma is similar to the proof of the corresponding lemmas in [5] and is omitted.

Lemma A.2. *For two arrays whose elements are ratios of mean queue lengths $r_m(i_1)$, $r'_m(i_2)$, $m = 1, 2, \ldots, M$ and i_1, $i_2 \geqslant 0$, if*

$$r_j(i_1) \Big/ \sum_{m=j}^{M} r_m(i_1) \leqslant r'_j(i_2) \Big/ \sum_{m=j}^{M} r'_m(i_2) \quad \text{for } 1 \leqslant j \leqslant M,$$

then

$$\left\{\tau_j\left[1 + r_j(i_1) \times l_1\right]\right\} \Big/ \left\{\sum_{m=j}^{M} \tau_m\left[1 + r_m(i_1) \times l_1\right]\right\}$$

$$\leqslant \left\{\tau_j\left[1 + r'_j(i_2) \times l_2\right]\right\} \Big/ \left\{\sum_{m=j}^{M} \tau_m\left[1 + r'_m(i_2) \times l_2\right]\right\}$$

for all nonnegative integers l_1 and l_2 such that $l_1 \geqslant l_2$ and $1 \leqslant j \leqslant M$.

The proof is similar to part of the proof of Lemma A.1.

Lemma A.3

$$r_j(n_{i-1}) \Big/ \sum_{m=j}^{M} r_m(n_{i-1}) \geqslant r_j(n_i) \Big/ \sum_{m=j}^{M} r_m(n_i)$$

for all i and j such that $1 \leqslant j \leqslant M$ and $1 \leqslant i \leqslant S-1$.

Proof. By induction on i.

(i) When $i=1$,

$$r_j(1) \Big/ \sum_{m=j}^{M} r_m(1) - r_j(n_1) \Big/ \sum_{m=j}^{M} r_m(n_1)$$

$$= \tau_j \Big/ \sum_{m=j}^{M} \tau_m - \tau_j(L + \tau_j \times n_0) \Big/ \sum_{m=j}^{M} \tau_m(L + \tau_m \times n_0)$$

$$= \frac{\tau_j \sum_{m=j}^{M} \tau_m(L + \tau_m \times n_0) - \tau_j(L + \tau_j \times n_0) \sum_{m=j}^{M} \tau_m}{\sum_{m=j}^{M} \tau_m \sum_{m=j}^{M} \tau_m(L + \tau_m \times n_0)}$$

$$= n_0 \times \left[\tau_j \sum_{m=j}^{M} \tau_m^2 - \tau_j \sum_{m=j}^{M} \tau_m \tau_j \right] \Big/ \left[\sum_{m=j}^{M} \tau_m \sum_{m=j}^{M} \tau_m(L + \tau_m \times n_0) \right] \geqslant 0.$$

This establishes the induction base.

(ii) Assume that it is true for $i=k$. From Lemma A.2, it is also true for $i=k+1$. □

Notice that Lemmas A.1, A.2, and A.3 are also true for exact mean queue lengths $q_m(\cdot)$, $m = 1, 2, \ldots, M$, since they correspond to the special case in which the population sequence is $2, 3, \ldots, N$.

Proof of Theorem 3.2. We only have to prove that

$$r_j(n_{i-1}) \Big/ \sum_{m=j}^{M} r_m(n_{i-1}) \geqslant q_j(n_i - 1) \Big/ \sum_{m=j}^{M} q_m(n_i - 1) \tag{A.3}$$

for all i, $1 \leqslant i \leqslant S$ and all j, $1 \leqslant j \leqslant M$.

The proof is by induction on i.

(i) When $i=1$, $r_j(1) = q_j(1)$ for $1 \leqslant j \leqslant M$. Equation (A.3) is clearly true.

(ii) Assume that it is true for $i=k$. From Lemma A.1 it follows that

$$r_j(n_k) \Big/ \sum_{m=j}^{M} r_m(n_k) \geqslant q_j(n_k) \Big/ \sum_{m=j}^{M} q_m(n_k). \tag{A.4}$$

From Lemma A.3, we have

$$q_j(n_{k+1} - 1) \Big/ \sum_{m=j}^{M} q_m(n_{k+1} - 1) \leqslant q_j(n_k) \Big/ \sum_{m=j}^{M} q_m(n_k). \tag{A.5}$$

From equations (A.4) and (A.5), it is also true for $i = k+1$. From Lemma A.1, we know that equation (24) is true. □

Proof of Corollary 3.3. We only have to prove that

$$r_j(n_{i-1}) \Big/ \sum_{m=j}^{M} r_m(n_{i-1}) = q_j(n_i - 1) \Big/ \sum_{m=j}^{M} q_m(n_i - 1)$$

for all i, $1 \leqslant i \leqslant S$ and all j, $1 \leqslant j \leqslant M$. The proof is similar to that of Theorem 3.2. □

Proof of Theorem 3.4. We only have to prove that the upper bound obtained from population sequence $1 < n_1 < n_2 < \cdots < n_S = N$ is smaller than that obtained from population sequence $m_1, m_2, \ldots, m_{S-1}$, where $m_i = n_i$ for $i = 1, 2, \ldots, l-1$, and $m_i = n_{i+1}$ for $i = l, l+1, \ldots, S-1$, where l is an integer such that $1 \leqslant l \leqslant S-1$. It is equivalent to showing that

$$r_j(n_i) \Big/ \sum_{m=j}^{M} r_m(n_i) \leqslant r'_j(m_{i-1}) \Big/ \sum_{m=j}^{M} r'_m(m_{i-1}) \quad \text{for } 1 \leqslant j \leqslant M \quad \text{and } 1 \leqslant i \leqslant S-1,$$

where $r'_j(m_i)$ denotes the value of ratio[j] in Algorithm 3.1 when the sequence $m_1, m_2, \ldots, m_{S-1}$ is used.

The proof is by induction on i.

(i) When $i = 1$, from Lemma A.3

$$r_j(n_1) \Big/ \sum_{m=j}^{M} r_m(n_1) \leqslant r_j(n_0) \Big/ \sum_{m=j}^{M} r_m(n_0) = r'_j(m_0) \Big/ \sum_{m=j}^{M} r'_m(m_0) \quad \text{for all } 1 \leqslant j \leqslant M,$$

where $r_j(n_0)$ and $r'_j(m_0)$ denote the initial values of ratio[j] in Algorithm 3.1 when the corresponding sequences $n_1, n_2, \ldots, n_S$ and $m_1, m_2, \ldots, m_{S-1}$ are used. This establishes the base of the induction.

(ii) Suppose that it is true for $i = k$, i.e.,

$$r_j(n_k) \Big/ \sum_{m=j}^{M} r_m(n_k) \leqslant r'_j(m_{k-1}) \Big/ \sum_{m=j}^{M} r'_m(m_{k-1}) \quad \text{for all } 1 \leqslant j \leqslant M.$$

From Lemma A.2 and the fact that $n_k \geqslant m_{k-1}$ it is also true for $i = k+1$. □

Proof of Theorem 3.5. With Lemmas A.1 and A.2, the proof is similar to the proof of Theorem 3.4 and is omitted.

The following lemma, which is similar to Lemma A.1, is used in the proofs of Theorems 3.8–3.10. Its proof is similar to that of Lemma A.1 and is omitted.

Lemma A.4. *For any i, $1 \leqslant i \leqslant S$, and all j, $1 \leqslant j \leqslant M$ if*

$$r_j(n_{i-1}) \Big/ \sum_{m=j}^{M} r_m(n_{i-1}) \leqslant q_j(n_i - 1) \Big/ \sum_{m=j}^{M} q_m(n_i - 1),$$

then

$$\sum_{m=j}^{M} q_m(n_i - 1) \leqslant (n_i - 1) \sum_{m=j}^{M} r_m(n_{i-1}),$$

$$\left[\sum_{m=j}^{M} \tau_m q_m(n_i - 1)\right] \Big/ \left[\sum_{m=j}^{M} q_m(n_i - 1)\right] \leqslant \left[\sum_{m=j}^{M} \tau_m r_m(n_i - 1)\right] \Big/ \left[\sum_{m=j}^{M} r_m(n_{i-1})\right],$$

and

$$r_j(n_i)\Big/\sum_{m=j}^{M} r_m(n_i) \leqslant q_j(n_i)\Big/\sum_{m=j}^{M} q_m(n_i).$$

Proof of Theorem 3.8. We only have to prove that

$$r_j(n_{i-1})\Big/\sum_{m=j}^{M} r_m(n_{i-1}) \leqslant q_j(n_i-1)\Big/\sum_{m=j}^{M} q_m(n_i-1) \tag{A.6}$$

for all i, $1 \leqslant i \leqslant S$ and all j, $1 \leqslant j \leqslant M$.

The proof is by induction on i.

(i) When $i=1$, from Lemma 2.9,

$$r_j(n_0)/r_m(n_0) \leqslant q_j(n_0)/q_m(n_0) \quad \text{for all } 1 \leqslant j \leqslant m \leqslant M.$$

Therefore equation (A.6) is true.

(ii) Assume that it is true for $i=k$. From Lemma A.4 it follows that

$$r_j(n_k)\Big/\sum_{m=j}^{M} r_m(n_k) \leqslant q_j(n_k)\Big/\sum_{m=j}^{M} q_m(n_k). \tag{A.7}$$

Because $n_{k+1} \leqslant n_k$, from Lemma A.3, we have

$$q_j(n_{k+1}-1)\Big/\sum_{m=j}^{M} q_m(n_{k+1}-1) \geqslant q_j(n_k)\Big/\sum_{m=j}^{M} q_m(n_k). \tag{A.8}$$

From equations (A.7) and (A.8), it is also true for $i=k+1$. From Lemma A.4, we know that equation (25) is true. □

In Lemma A.5, to be given in the following, $r_j(n_i)$ denotes the value of ratio[j] when the population size is n_i during the execution of Algorithm 3.7. This lemma is used in the proof of Theorem 3.9.

Lemma A.5

$$r_j(n_0)\Big/\sum_{m=j}^{M} r_m(n_0) \leqslant r_j(n_1)\Big/\sum_{m=j}^{M} r_m(n_1)$$

for all j such that $1 \leqslant j \leqslant M$.

Proof. It is sufficient to prove

$$r_j(n_0)/r_i(n_0) \leqslant r_j(n_1)/r_i(n_1) \quad \text{for } 1 \leqslant j < i \leqslant M.$$

For the sake of clarity, we shall replace n_0 with n for the balance of this proof. We then have $r_i(n_0)=\tau_i^n$ and $r_i(n_1)=\tau_i(1+n\times\tau_i^n/L^n)$ for $1 \leqslant i \leqslant M$.

Thus

$$\frac{r_j(n_1)}{r_i(n_1)} - \frac{r_j(n)}{r_i(n)} = \frac{\tau_j(1+n\times\tau_j^n/L^n)}{\tau_i(1+n\times\tau_i^n/L^n)} - \frac{\tau_j^n}{\tau_i^n}. \tag{A.9}$$

Multiply equation (A.9) by $L^n\tau_i^{n+1}(1+n\times\tau_i^n/L^n)/\tau_j$. We obtain

$$\begin{aligned} &L^n\tau_i^{n-1}(1+n\times\tau_j^n/L^n) - L^n\tau_j^{n-1}(1+n\times\tau_i^n/L^n)\\ &\quad = L^n\left(\tau_i^{n-1}-\tau_j^{n-1}\right) - n\tau_i^{n-1}\tau_j^{n-1}(\tau_i-\tau_j). \end{aligned} \tag{A.10}$$

We shall prove that equation (A.10) is greater than or equal to zero by induction on M.

(i) Case of $M = 2$ (i.e., there are only two fixed-rate service centers): we shall prove this base case by induction on n.

(a) When $n = 2$ (i.e., $n_1 = 3$), we have

$$L^2(\tau_i - \tau_j) - n\tau_i\tau_j(\tau_i - \tau_j) = (\tau_i - \tau_j)\left[\tau_i^2 + \tau_j^2 - 2\tau_i\tau_j\right] = (\tau_i - \tau_j) \times (\tau_i - \tau_j)^2 \geqslant 0,$$

where $L^2 = \tau_i^2 + \tau_j^2$. This establishes the induction base for n.

(b) Assume that equation (A.10) is true for $n = k$. After factorizing equation (A.10) and eliminating $(\tau_i - \tau_j)$, the induction hypothesis becomes

$$L^k\left(\tau_i^{k-2} + \tau_i^{k-3}\tau_j + \cdots + \tau_i\tau_j^{k-3} + \tau_j^{k-2}\right) - k\tau_i^{k-1}\tau_j^{k-1} \geqslant 0. \tag{A.11}$$

Multiply equation (A.11) by $\tau_i\tau_j$, we get

$$\begin{aligned}&\tau_j L^k\left(\tau_i^{k-1} + \tau_i^{k-2}\tau_j + \cdots + \tau_i^2\tau_j^{k-3} + \tau_i\tau_j^{k-2}\right) - k\tau_i^k\tau_j^k \\ &\quad = \tau_j L^k\left(\tau_i^{k-1} + \tau_i^{k-2}\tau_j + \cdots + \tau_i\tau_j^{k-2} + \tau_j^{k-1}\right) - (k+1)\tau_i^k\tau_j^k - \left(\tau_j^k L^k - \tau_i^k\tau_j^k\right) \geqslant 0.\end{aligned} \tag{A.12}$$

When $n = k + 1$, the left-hand side of equation (A.11) becomes

$$\begin{aligned}&L^{k+1}\left(\tau_i^{k-1} + \tau_i^{k-2}\tau_j + \cdots + \tau_i\tau_j^{k-2}\tau_j^{k-1}\right) - (k+1)\tau_i^k\tau_j^k \\ &\quad > \tau_j L^k\left(\tau_i^{k-1} + \tau_i^{k-2}\tau_j + \cdots + \tau_i\tau_j^{k-2}\tau_j^{k-1}\right) - (k+1)\tau_i^k\tau_j^k \geqslant 0.\end{aligned} \tag{A.13}$$

Equation (A.13) is true because $(\tau_j^k L^k - \tau_i^k\tau_j^k)$ in equation (A.12) is nonnegative and τ_j is smaller than τ_i. We have thus proved the base case for M.

(ii) Assume that equation (A.10) is true for $M = l$, i.e.,

$$\left[\sum_{m=1}^{l} \tau_m^n\right]\left(\tau_i^{n-1} - \tau_j^{n-1}\right) - n\tau_i^{n-1}\tau_j^{n-1}(\tau_i - \tau_j) \geqslant 0.$$

It is clear that

$$\left[\sum_{m=1}^{l+1} \tau_m^n\right]\left(\tau_i^{n-1} - \tau_j^{n-1}\right) - n\tau_i^{n-1}\tau_j^{n-1}(\tau_i - \tau_j) \geqslant 0.$$

Therefore,

$$\frac{r_j(n_0)}{r_i(n_0)} \leqslant \frac{r_j(n_1)}{r_i(n_1)} \quad \text{for } 1 \leqslant j < i \leqslant M. \quad \square$$

Proof of Theorem 3.9. We only have to prove that the lower bound obtained from population sequence $n_1, n_2, \ldots, n_S = N$ is larger than that obtained from population sequence $m_1, m_2, \ldots, m_{S-1}$, where $m_i = n_i$ for $i = 1, 2, \ldots, l-1$, and $m_i = n_{i+1}$ for $i = l, l+1, \ldots, S-1$, where l is an integer such that $1 < l \leqslant S - 1$. It is equivalent to showing that

$$r_j(n_i) \Big/ \sum_{m=j}^{M} r_m(n_i) \geqslant r_j'(m_{i-1}) \Big/ \sum_{m=j}^{M} r_m'(m_{i-1}) \quad \text{for } 1 \leqslant j \leqslant M \text{ and } 1 \leqslant i \leqslant S-1,$$

where $r_j'(m_i)$ denotes the value of ratio[j] in Algorithm 3.7 when the sequence $m_1, m_2, \ldots, m_{S-1}$ is used.

The proof is by induction on i.

(i) When $i = 1$, from Lemma A.5,

$$r_j(n_1) \Big/ \sum_{m=j}^{M} r_m(n_1) \geqslant r_j(n_0) \Big/ \sum_{m=j}^{M} r_m(n_0) = r_j'(m_0) \Big/ \sum_{m=j}^{M} r_m'(m_0) \quad \text{for all } 1 \leqslant j \leqslant M,$$

where $r_j(n_0)$ and $r_j'(m_0)$ denote the initial values of ratio[j] in Algorithm 3.7 when the corresponding sequences $n_1, n_2, \ldots, n_S$ and $m_1, m_2, \ldots, m_{S-1}$ are used. This establishes the base of the induction.

(ii) Suppose that it is true for $i = k$, i.e.,

$$r_j(n_k) \Big/ \sum_{m=j}^{M} r_m(n_k) \geqslant r_j'(m_{k-1}) \Big/ \sum_{m=j}^{M} r_m'(m_{k-1}) \quad \text{for all } 1 \leqslant j \leqslant M.$$

From Lemma A.2 and the fact that $n_k \leqslant m_{k-1}$ it is also true for $i = k + 1$. □

Proof of Theorem 3.10. We shall prove a stronger result. I.e., the lower bound obtained from population sequence $n_1 \geqslant n_2 \geqslant \cdots \geqslant n_S = N$ is larger than or equal to that obtained from population sequence $m_1 \geqslant m_2 \geqslant \cdots \geqslant m_S = N$, where $m_i \geqslant n_i \geqslant N$ for $i = 1, 2, \ldots, S - 1$. It is equivalent to showing that

$$r_j(n_{i-1}) \Big/ \sum_{m=j}^{M} r_m(n_{i-1}) \geqslant r_j'(m_{i-1}) \Big/ \sum_{m=j}^{M} r_m'(m_{i-1}) \quad \text{for } 1 \leqslant j \leqslant M \text{ and } 1 \leqslant i \leqslant S,$$

where $r_j'(m_i)$ denotes the value of ratio[j] in Algorithm 3.7 when the sequence $m_1, m_2, \ldots, m_{S-1}$ is used. Let $n_0 = n_1 - 1$ and $m_0 = m_1 - 1$.

The proof is by induction on i.

(i) When $i = 1$, we have $n_0 \leqslant m_0$ and

$$r_j(n_0)/r_m(n_0) = \tau_j^{n_0}/\tau_m^{n_0} \geqslant \tau_j^{m_0}/\tau_m^{m_0} = r_j'(m_0)/r_m'(m_0) \quad \text{for all } 1 \leqslant j \leqslant m \leqslant M.$$

Therefore,

$$r_j(n_0) \Big/ \sum_{m=j}^{M} r_m(n_0) \geqslant r_j'(m_0) \Big/ \sum_{m=j}^{M} r_m'(m_0) \quad \text{for all } 1 \leqslant j \leqslant M.$$

This establishes the base of the induction.

(ii) Suppose that it is true for $i = k$, i.e.,

$$r_j(n_k) \Big/ \sum_{m=j}^{M} r_m(n_k) \geqslant r_j'(m_k) \Big/ \sum_{m=j}^{M} r_m'(m_k) \quad \text{for all } 1 \leqslant j \leqslant M.$$

From Lemma A.2 and the fact that $n_k \leqslant m_k$ it is also true for $i = k + 1$. □

Proof of Theorem 4.1. From MVA, we have

$$D_k(\boldsymbol{n}) = \sum_{m=1}^{M} \tau_{mk}\left[1 + \sum_{h=1}^{K} q_{mh}(\boldsymbol{n} - \boldsymbol{1}_k)\right]$$

$$\left(\text{if } m \text{ is a delay service center, then } \sum_{h=1}^{K} q_{mh}(\boldsymbol{n} - \boldsymbol{1}_k) = 0\right)$$

$$= \sum_{m=1}^{M} \tau_{mk} + \sum_{m=1}^{M_F} \tau_{mk} \sum_{h=1}^{K} q_{mh}(\boldsymbol{n} - \boldsymbol{1}_k) = L_k + \sum_{\substack{m=1 \\ m \text{ in chain } k}}^{M_F} \tau_{mk} \sum_{\substack{h=1 \\ \text{chain } h \text{ visits } m}}^{K} q_{mh}(\boldsymbol{n} - \boldsymbol{1}_k)$$

$$= L_k + \sum_{\substack{m=1 \\ m \text{ in chain } k}}^{M_F} \tau_{mk} \sum_{\substack{h=1 \\ \text{chain } h \text{ visits } m}}^{K} \tau_{mh} \underline{T}_h(\boldsymbol{n} - \boldsymbol{1}_k) + \alpha,$$

where

$$\alpha = \sum_{\substack{m=1 \\ m \text{ in chain } k}}^{M_F} \tau_{mk} \sum_{\substack{h=1 \\ \text{chain } h \text{ visits } m}}^{K} \left[q_{mh}(\boldsymbol{n} - \boldsymbol{1}_k) - \tau_{mh} \underline{T}_h(\boldsymbol{n} - \boldsymbol{1}_k)\right].$$

We then have

$$\begin{aligned}
\alpha &\leqslant \sum_{\substack{h=1\\ \text{chain } h\\ \text{intersects chain } k}}^{K} \tau_{\max,h,k} \sum_{\substack{m=1\\ m \text{ in chains } h,k}}^{M_{\mathrm{F}}} \left[q_{mh}(\boldsymbol{n}-\mathbf{1}_k) - \tau_{mh}\underline{T}_h(\boldsymbol{n}-\mathbf{1}_k)\right] \\
&\leqslant \sum_{\substack{h=1\\ \text{chain } k\\ \text{intersects chain } k}}^{K} \tau_{\max,h,k} \left[n_h - L_{D,h}\underline{T}_h(\boldsymbol{n}-\mathbf{1}_k) - \sum_{\substack{m=1\\ m \text{ in chain } h\\ m \text{ not in chain } k}}^{M_{\mathrm{F}}} \tau_{mh}\underline{T}_h(\boldsymbol{n}-\mathbf{1}_k) \right. \\
&\qquad \left. - \sum_{\substack{m=1\\ m \text{ in chains } h,k}}^{M_{\mathrm{F}}} \tau_{mh}\underline{T}_h(\boldsymbol{n}-\mathbf{1}_k) \right] - \tau_{\max,k,k} \\
&= \sum_{\substack{h=1\\ \text{chain } h\\ \text{intersects chain } k}}^{K} \tau_{\max,h,k} \left[n_h - L_{\mathrm{D},h}\underline{T}_h(\boldsymbol{n}-\mathbf{1}_k) - \sum_{\substack{m=1\\ m \text{ in chain } h}}^{M_{\mathrm{F}}} \tau_{mh}\underline{T}_h(\boldsymbol{n}-\mathbf{1}_k) \right] - \tau_{\max,k,k} \\
&= \sum_{\substack{h=1\\ \text{chain } h\\ \text{intersects chain } k}}^{K} \tau_{\max,h,k} \left[n_h - L_{\mathrm{D},h}\underline{T}_h(\boldsymbol{n}-\mathbf{1}_k) - L_{\mathrm{F},h}\underline{T}_h(\boldsymbol{n}-\mathbf{1}_k)\right] - \tau_{\max,k,k} \\
&= \sum_{\substack{h=1\\ \text{chain } h\\ \text{intersects chain } k}}^{K} \tau_{\max,h,k} \left[n_h - L_h\underline{T}_h(\boldsymbol{n}-\mathbf{1}_k)\right] - \tau_{\max,k,k}.
\end{aligned}$$

On the other hand, we have

$$\begin{aligned}
\alpha &\geqslant \sum_{\substack{m=1\\ m \text{ in chain } k}}^{M_{\mathrm{F}}} \tau_{\min,k}\left[q_{mk}(\boldsymbol{n}-\mathbf{1}_k) - \tau_{mk}\underline{T}_k(\boldsymbol{n}-\mathbf{1}_k)\right] \\
&= \tau_{\min,k}\left[\sum_{\substack{m=1\\ m \text{ in chain } k}}^{M_{\mathrm{F}}} q_{mk}(\boldsymbol{n}-\mathbf{1}_k) - \sum_{\substack{m=1\\ m \text{ in chain } k}}^{M_{\mathrm{F}}} \tau_{mk}\underline{T}_k(\boldsymbol{n}-\mathbf{1}_k) \right] \\
&\geqslant \tau_{\min,k}\left[n_k - 1 - L_{\mathrm{D},k}\overline{T}_k(\boldsymbol{n}-\mathbf{1}_k) - L_{\mathrm{F},k}\underline{T}_k(\boldsymbol{n}-\mathbf{1}_k)\right]. \qquad \square
\end{aligned}$$

Proof of Corollary 4.2. Immediate from Theorem 4.1. □

References

[1] M. Abramowitz and I.E. Stegun, eds., *Handbook of Mathematical Functions* (Dover Publications, New York, 1972).

[2] F. Baskett, K.M. Chandy, R.R. Muntz and F. Palacios, Open, closed and mixed networks of queues with different class of customers, *J. ACM* **22** (1975) 248–260.

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

[4] K.M. Chandy, U. Herzog and L. Woo, Parametric analysis of queueing networks, *IBM J. Res. Develop.* **19** (1975) 36–42.

[5] D.L. Eager and K.C. Sevcik, Performance bound hierarchies for queueing networks, *ACM Trans. Comput. Systems* **1** (2) (1983) 99–115.

[6] D.L. Eager and K.C. Sevcik, Bound hierarchies for multiple-class queueing networks, *J. ACM* **33** (1) (1986) 179–206.

[7] K.P. Hoyme, S.C. Bruell, P.V. Afshari and R.Y. Kain, A tree-structured mean value analysis algorithm, *ACM Trans. Comput. Systems* **4** (2) (1986) 178–185.

[8] C.-T. Hsieh and S.S. Lam, *Two Classes of Performance Bounds for Closed Queueing Networks*, Tech. Rept. TR-85-09, Univ. of Texas at Austin, TX, June 1985.

[9] T. Kerola, The composite bound method for computing throughput bounds in multiple class environments, *Performance Evaluation* **6** (1) (1986) 1–9.

[10] J. Kriz, Throughput bounds for closed queueing networks, *Performance Evaluation* **4** (1) (1984) 1–10.

[11] S.S. Lam, Dynamic scaling and growth behavior of queueing network normalization constants, *J. ACM* **29** (1982) 492–513.

[12] S.S. Lam and J.W. Wong, Queueing network models of packet switching networks, Part 2: Networks with population size constraints, *Performance Evaluation* **2** (1982) 161–180.

[13] S.S. Lam and Y.L. Lien, A tree convolution algorithm for the solution of queueing networks, *Comm. ACM.* **26** (3) (1983) 203–215.

[14] S.S. Lam and C.-T. Hsieh, Models and algorithms for the design of store-and-forward communication networks, *Conf. Record Internat. Conf. on Communications*, Chicago, June 1985.

[15] J.D.C. Little, A proof of the queueing formula $L = \lambda W$, *Oper. Res.* **9** (1961) 383–387.

[16] J. McKenna and D. Mitra, Asymptotic expansions and integral representations of moments of queue lengths in closed Markovian networks, *J. ACM* **31** (2) (1984) 346–360.

[17] R.R. Muntz and J. Wong, Asymptotic properties of closed queueing network models, *Proc. 8th Ann. Princeton Conf. on Information Science and Systems*, Princeton University, March 1974.

[18] M. Reiser and H. Kobayashi, Queueing networks with multiple closed chains: Theory and computational algorithms, *IBM J. Res. Develop.* **21** (1975) 283–294.

[19] M. Reiser and S. Lavenberg, Mean value analysis of closed multichain queueing networks, *J. ACM* **27** (2) (1980) 313–322.

[20] P. Schweitzer, Approximate analysis of multiclass closed networks of queues, *Proc. Internat. Conf. on Stochastic Control and Optimization*, Amsterdam, The Netherlands, 1979.

[21] K.C. Sevcik and I. Mitrani, The distribution of queueing network states at input and output instants, *Proc. 4th Internat. Symp. on Modeling and Performance Evaluation of Computer Systems* (North-Holland, Amsterdam, and The International Institute for Systems Analysis, Vienna, Austria, 1979).

[22] L.E. Stephens and L.W. Dowdy, Convolutional bound hierarchies, *Proc. 1984 ACM SIGMETRICS Conf.*, Cambridge, MA (1984) 120–133.

[23] R. Suri, Generalized quick bounds for performance of queueing networks, *Computer Performance* **5** (2) (1984) 116–120.

[24] S. Tucci and C.H. Sauer, The tree MVA algorithm, *Performance Evaluation* **5** (3) (1985) 187–197.

[25] J. Zahorjan, K.C. Sevcik, D.L. Eager and B. Galler, Balanced job bound analysis of queueing networks, *Comm. ACM* **25** (2) (1982) 132–141.