APPROXIMATE ANALYSIS FOR THE MERGE CONFIGURATION OF AN OPEN QUEUEING NETWORK WITH BLOCKING

Hyo-Seong Lee
Stephen M. Pollock

Technical Report #87-11

Department of Industrial and Operations Engineering
The University of Michigan
1205 Beal Avenue
Ann Arbor, Michigan 48109-2117
(313) 764-6474

ABSTRACT

A merge configuration of open queueing networks with exponential service times and finite buffers is analysed. We offer an iterative algorithm to approximate the steady-state probabilities for each queue of the system. The procedure decomposes the queueing network into individual queues and analyses each individual queue in isolation. An M/M/1/N or M/G/1/N model is used for the analysis of the merging queues; An M/M/1/N with state dependent arrival rate model is used for the receiving queue. The approximation method is easy to implement, requires little memory, is computationally fast and yields very accurate results.

1.Introduction

Queuing networks, where the output of one server is the input to others, are useful in modeling manufacturing systems, computer systems, telecommunications, etc. This is particularly true when the buffer space between facilities is finite, so that an upstream server can be blocked due to unavailability of space in the destination server. In general, queueing systems with blocking are difficult to analyse, and only under very limited conditions can the exact solutions be obtained. Therefore, most analyses available are based on approximation or simulation methods.

There are three basic structural configurations in networks of queues: tandem, split and merge, combinations of which can represent a general network. For open queueing networks (i.e., where units do not re-circulate continually), among these three the tandem configuration has been most frequently analysed. It has been studied by Hillier and Boling[11], Caseau and Pujolle[7], Altiok[2], Perros and Altiok[15], Bocharov and Rokhas[4], Brandwajn and Jow[6], Foster and Perros[8], Gershwin[9], Pollock, Birge and Alden[18]. The split and merge configurations have been reported by Boxma and Konheim[5] and Altiok and Perros[3]. Combinations of these configurations to form open queueing networks with blocking have been studied by Takahashi, Miyahara and Hasegawa[21], Labetoulle and Pujolle[12], Perros and Altiok[16] and Perros and Snyder[17]. For closed queueing networks with blocking, approximation algorithms have been reported by Suri and Diehl[19,20], Yao and Buzacott[22], Akyildiz[1] and Onvural and Perros[13]. A detailed literature survey can be found in Perros[14].

In this paper, we present an approximation method to analyse the merge configuration of an open queueing network with blocking. This algorithm is based on an earlier algorithm (SIMP) proposed by Pollock, Birge and Alden[18] to analyse tandem queues with blocking. It is also similar to the algorithm of Altiok and Perros [3], but differs in that we

describe the state of the merged queue by considering the sequence of blocked units. Our algorithm also uses a fundamentally different approximation for the effective service time distribution of the merging queues.

2.The Merge Configuration

The network we consider is identical to that in Altiok and Perros[3]. It consists of K parallel single server queues, each feeding the same single server queue (see Fig.1). We will call each of the K parallel queues (queue 1,2,···,K) a merging queue; the queue receiving the output of these merging queues is the merged queue (or queue 0). The service time at queue i follows an exponential distribution with rate μ_i; Arrivals to queue i are independent Poisson Processes with rate λ_i. There is no external arrival to the merged queue. The buffer size of the i-th merging queue is N_i (including the one in service). $N \equiv N_0$ is the buffer size of the queue 0, and $\mu \equiv \mu_0$ is service rate at queue 0.

Arrivals to any queue are served in a FIFO manner. If an arrival encounters a merging queue when it is full, the arrival is lost. When a unit completes service at the i-th queue (i=1,···,K), it proceeds to queue 0 only if space is available. However if queue 0 is full at that time, the unit waits in the i-th server until it can enter queue 0. During this time the i-th server cannot serve other units that might be waiting in its buffer: in this case, the i-th server is said to be blocked and queue 0 is blocking. Thus, the merging queues cannot be blocking, and the merged queue cannot be blocked. Since K queues feed into queue 0, there might be more than one queue blocked at any instant; in the worse case, obviously, there can be K blocked queues. In the multiple blocking case, we assume blocked units enter queue 0 on a "First-Blocked-First-Enter" basis [3].

<Insert Fig. 1>

3. Analysis of the Model

3.1. Approach and General Relationships

Our general approach is to decompose the queueing network into individual queues, and then analyse each individual queue separately. We will make important use of the fact that the time for a unit to clear service in a merging queue has two components; the actual service time plus the delay caused by being (possibly) blocked by queue 0. In particular, we make the following approximate assumptions about what is in reality an extremely complicated system:

a) There are Poisson arrivals, with rate λ_i^* (i=1, ···,K) from queue i to queue 0 as long as the i-th queue is not blocked. When the i-th queue is blocked, there is of course no arrival from queue i to queue 0.

b) The buffer size of queue 0 is augmented by K so that the total buffer of queue 0 is N+K.

Assumption b) is simply a modelling convenience made possible by noting that a blocked unit is ready to proceed to queue 0 whenever there is space in queue 0. Thus, such a unit is effectively waiting in line to be served by server 0.

The key to our approach is thus to take advantage of the fact that queue 0 is never blocked, and its service time is exponentially distributed with rate μ, so that it can be analysed by using M/M/1/N+K analysis, using approximation a). The second part of our approach is to use seperate M/G/1/N_i analyses in each merging queue i, in order to obtain

the values of λ_i^* used in approximation a).

In order to proceed, we define (where, unless otherwise stated, the index i always runs from 1 to K)

$T_i \equiv$ clearance time for server i, i.e., the time between when a unit enters service in queue i, and when it arrives at queue 0,

S_i ,(S) $\equiv$ service time at server i (server 0), (excluding any delay due to blocking)

$\lambda_i^* \equiv$ arrival rate to queue 0 from queue i as long as queue i is not blocked,

$$\lambda^* \equiv \sum_{i=1}^{K} \lambda_i^*$$

$P_i(k)$, $[P(k)] \equiv$ steady state probability that there are k units at queue i [queue 0],

$b_i(n) \equiv$ probability that queue i is blocked and the number of units at queue 0 is N+n,

$\alpha_i(k) \equiv$ conditional probability that, upon service completion at server i, there are k units at queue 0,

$f_i \equiv$ Probability {i-th queue is full} = $P_i(N_i)$

These definitions, and the structure of the system, produce the following relationships.

a) The contribution of queue i to the total system throughput, $\overline{\lambda_i}$, is given by

$$\overline{\lambda_i} = \lambda_i(1-f_i) \tag{3.1}$$

b) The total system throughput, $\overline{\lambda}$, is

$$\overline{\lambda} = \sum_{i=1}^{K} \overline{\lambda_i}$$

c) The clearance time for queue i is a random variable represented by

$$T_i = \begin{cases} S_i & \text{with probability } 1 - \sum_{j=0}^{K-1} \alpha_i(N+j) \\ S_i + S^{(j+1)} & \text{with probability } \alpha_i(N+j) \quad j=0,\cdots,K-1 \end{cases} \tag{3.2}$$

where $S^{(j+1)}$ is a gamma-distributed random variable that is the sum of $j+1$ service times at queue 0. In particular, as will be used later on, $E[S^{(j+1)}] = \frac{j+1}{\mu}$

Note that the bottom line of (3.2) follows because 1) the residual service time of a unit in queue 0 is exponential by the memoryless property of exponential distribution, and 2) given that a unit sees j other units blocked at the instant of service completion at queue i, it must wait j independent service times and one residual service time at queue 0 before it feeds into queue 0. By 1) the distribution of this blocking time is the convolution of $j+1$ independent exponentially distributed random variables with parameter μ, which is gamma distributed random variable with parameters $(j+1,\mu)$.

3.2. Analysis of Queue 0

Before we analyse queue 0 in general, we first consider the special case of $K=2$. Suppose we know the values of λ_1^*, λ_2^*. Define the state of queue 0 to be i if there are i units in queue 0 and there is no blocking. If there is blocking, the state of queue 0 is $(N+n,\mathbf{v})$ where $N+n$ is the number of units in queue 0 (including n blocked units), and $\mathbf{v}$ is an n-component vector representing, in order, the units which are being blocked.

For example, the state

$\{N+1, 1\} \equiv$ {queue 1 is blocked by queue 0};

$\{N+1, 2\} \equiv$ {queue 2 is blocked by queue 0};

$\{N+2, 12\} \equiv$ {queue 1 and 2 are blocked by queue 0 in that order};

$\{N+2, 21\} \equiv$ {queue 2 and 1 are blocked by queue 0 in that order}

The resulting state transition diagram of queue 0 is shown in Fig. 2. We are interested in computing the (steady state) occupancy probabilities of these states. We will find these, however, by looking at a smaller set of aggregated states. In particular, define

$$\{N+1\} \equiv \{N+1, 1\} \cup \{N+1, 2\}$$

$$\{N+2\} \equiv \{N+2, 12\} \cup \{N+2, 21\}$$

These new aggregated states $\{N+i\}$, $i=1,2$, represent, simply, $N+i$ units in queue 0 — including those blocked and thus waiting in their respective merging queue. We also define $\lambda_a(i)$ to be the arrival rate from state i to state $i+1$ in the aggregated system. Figure 3 shows the transition diagram for this (smaller) state space. It is straightforward to show that the following equations of balance hold for the system in figure 2:

$$P(N+1, i) = \frac{\lambda_i^*}{\lambda^*} P(N+1) \qquad i=1,2 \tag{3.3}$$

$$P(N+2, 12) = P(N+2, 21) = \frac{1}{2} P(N+2) \tag{3.4}$$

where $\lambda^* \equiv \lambda_1^* + \lambda_2^*$.

Since this new process is a simple birth and death process, it is almost trivial to find P(i) for $i=0, \cdots, N+2$. From (3.3), (3,4) we can obtain the occupancy probabilities for the original (unaggregated) queue 0.

<Insert Fig. 2, Fig. 3>

This idea can be readily extended to the general system with K merging queues. As in the above, the state of queue 0 is i, the number of units in the system. If there is blocking, the state of queue 0 is (N+n,**v**), n=1,⋯,K, where N+n represents the number of units in queue 0, including blocked ones, and **v** is the n-component vector representing the units which are being blocked, in order. Unfortunately since the order of blocked units is part of the state description, the total number of states increases geometrically in K. However, if we are able to disregard the order of units being blocked, states which have the same number of units can be aggregated into one state, and the total number of states will be only N+K+1. We now define the states of such an aggregation indexed by i, the number of units in queue 0 (i=0,⋯,N+K). Recalling that we have defined $\lambda_a(i)$ to be the arrival rate to state i+1 from state i in the aggregated system, the following theorems hold.

Theorem 3.1. The aggregated queue 0 is equivalent (in terms of producing identical P(i)) to the original queue 0 if the arrival rate to each aggregated states is:

$$\lambda_a(i) = \lambda^* \qquad \text{for } i=0,\cdots,N \tag{3.5}$$

$$\lambda_a(N+n) = \frac{(n+1)\Omega_{n+1}}{\Omega_n} \qquad \text{for } n=1,\cdots,K\text{-}1 \tag{3.6}$$

where $\Omega_k = \sum_{i_1<\cdots<i_k} \prod_{j=1}^{k} \lambda_{i_j}^*, \quad i_j \in \{1,\cdots,K\}$

Note that if the merging queues are symmetric, then $\lambda_a(N+n)$ satisfies the simple expression

$$\lambda_a(N+n) = \frac{K-n}{K}\lambda^* \qquad \text{for } n=1,\cdots,K\text{-}1 \tag{3.7}$$

Theorem 3.2. The relationship between the steady-state probabilities of the original queue 0 and those of aggregated queue 0 is:

$$P(N+n,i_1 \cdots i_n) = \frac{\prod_{j=1}^{n} \lambda_{i_j}^{*}}{n!\ \Omega_n} P(N+n) \qquad \text{for } n=1,\cdots,K \tag{3.8}$$

(The proof is in the appendix)

To use (3.8) we note that the aggregated occupancy probability P(i) can be found from

$$P(i)=P(0) \prod_{j=1}^{i} \frac{\lambda_a(j)}{\mu} \qquad i=1,2,\cdots,N+K$$

$$\sum_{i=0}^{N+K} P(i) = 1 \tag{3.8.a}$$

In order to use these results, however, we need to know the values of λ_i^{*}. Suppose we have available the values of $\overline{\lambda_i}$ and $b_i(n)$. To find the λ_i^{*}, we can use the following equation.

$$\lambda_i^{*}\left(1 - \sum_{n=1}^{K} b_i(n)\right) = \overline{\lambda_i} \tag{3.9}$$

Equation (3.9) is a conservation equation which yields the value of the arrival rate λ_i^{*} needed in order to produce the given value of throughput $\overline{\lambda_i}$.

3.3 Analysis of Merging Queues

The above analysis depended upon known values of $b_i(n)$ and $\overline{\lambda_i}$. But $b_i(n)$ can be

obtained directly from P(i), as will be explained later. Also, from $b_i(n)$, $\alpha_i(k)$ can be straightforwardly found. Once values of $\alpha_i(k)$ are obtained, we can find the clearance time distribution at merging queues using equation (3.2), which makes it possible to analyse each merging queue by separate $M/M/1/N_i$ or $M/G/1/N_i$ analyses. Then we obtain the full probability f_i at each merging queue, which produces $\overline{\lambda}_i$. Given $\overline{\lambda}_i$ and $b_i(n)$, we once more find an updated λ_i^* using equation (3.9). This procedure is used iteratively until the expected clearance times at merging queues (or other appropriate variable values) converge.

We now examine each merging queue to find $b_i(n)$, the probability {queue i is blocked and the number of units at queue 0 is N+n},

$$b_i(n) = \sum_{i\in\{i_1,\cdots,i_n\}} P(N+n,i_1\cdots i_n)$$

By simply inserting equation (3.8) into the above, one can show that:

$$b_i(n) = \frac{\lambda_i^* \Omega_{n-1\backslash i}}{\Omega_n} P(N+n) \qquad i=1,\cdots,K \quad j=0,\cdots,K \tag{3.10}$$

where $\Omega_{n-1\backslash i} = \sum_{i_1<\cdots<i_{n-1}} \prod_{j=1}^{n-1} \lambda_{i_j}^*, \quad i_j\in\{1,\cdots,i-1,i+1,\cdots,K\}$

Note that if we sum (3.9) over the merging queues i using equation (3.10), we obtain the following conservation flow equation for the aggregated system.

$$\sum_{i=0}^{N+K-1} \lambda_a(i)P(i) = \overline{\lambda} \tag{3.11}$$

We now find the conditional probability $\alpha_i(k)$ that, upon service completion at server i, a unit is blocked and there are k units at queue 0. To do this, we will assume that a unit

at server i, at the instant service is completed, sees queue 0 in steady state. From this assumption and the fact that there can be no service completion at queue i if it is blocked, we see that $\alpha_i(N+j)$ is the conditional probability that there are $N+j$ units at queue 0 given that queue i is not blocked. Here, the steady state probability that queue i is blocked is $\sum_{n=1}^{K} b_i(n)$ since $b_i(n)$ is the probability that there are n blocked units at queue 0 including one from queue i. From these, $\alpha_i(N+j)$ can be obtained:

$$\alpha_i(N+j) = \frac{P(N+j) - b_i(j)}{1 - \sum_{n=1}^{K} b_i(n)} \qquad i=1,\cdots,K \quad j=0,\cdots,K \tag{3.12}$$

where $b_i(0)$ is defined to be 0.

We now can use this value of $\alpha_i(N+j)$ in equation (3.2) to obtain the distribution of T_i, the clearance time at queue i. This, in turn, allows us to analyse queue i by using M/M/1/N or M/G/1/N methods, since the arrivals to each are Poissons. Which of these we use depends upon the approximation assumption we are willing to make about the distribution of T_i:

Approximation 1 : Model each merging queue i ($i=1,\cdots,K$) as an M/M/1/N system. To do this, we assume the clearance time T_i is exponentially distributed, with an equivalent service rate equal to the reciprocal of the expected clearance time.

Approximation 2 : Model each merging queue i ($i=1,\cdots,K$) as an M/G/1/N system, using (3.2) to describe the random variable T_i.

Both of these lead to iterative algorithms that use queue 0 analysis to produce values of $\alpha_i(\cdot)$ and queue i analysis to produce λ_i^*.

3.3.1. M/M/1/N Analysis

If we assume that the clearance time at server i has an exponential distribution, the information needed to analyse the queue is only the arrival rate and the expected clearance time. But the arrival rate is given as λ_i, and the expected clearance time can be obtained from (3.2),

$$E(T_i) = \frac{1}{\mu_i} + \sum_{j=0}^{K-1} \alpha_i(N+j) \cdot \frac{j+1}{\mu} \qquad i=1,\cdots,K \tag{3.13}$$

An approximate solution to the system's steady state probabilities is then gotten from the following iterative algorithm.

0. (Set-up) Set the values of λ_i, μ_i, N_i for $i=0,\cdots,K$

1. (Initialization – The conditions here are as if all queues are unblocked.)

   Set $E(T_i) = \frac{1}{\mu_i}$, $\rho_i = \lambda_i \cdot E(T_i)$, for $i=1,\cdots,K$

   Find $\overline{\lambda_i} = \lambda_i \cdot (1-f_i)$ for $i=1,\cdots,K$ where

$$f_i = \frac{(1-\rho_i)\rho_i^{N_i}}{1-\rho_i^{N_i+1}} \tag{3.14}$$

   Set $\lambda_i = \overline{\lambda_i}$ for $i=1,\cdots,K$

$$\overline{\lambda} = \sum_{i=1}^{K} \overline{\lambda_i}, \quad \lambda^* = \overline{\lambda}$$

2. (Find full and blocking probabilities of queue 0)

   Find $\lambda_a(i)$ using equations (3.5), (3.6) for $i=0, \cdots,N+K-1$

   P(i) using birth and death equations (3.8.a) for $i=0, \cdots,N+K$

   $b_i(n)$ using equation (3.10) for $i=1, \cdots,K$, $n=0,\cdots,K$

$\alpha_i(N+j)$ using equation (3.12) for $i=1, \cdots, K$, $j=0, \cdots, K$

3. (Find expected clearance times at each merging queues.)

Solve for $E(T_i)$ using equation (3.13) and set $\rho_i = \lambda_i E(T_i)$ for $i=1, \cdots, K$.

4. (Convergence check)

If updated values of $E(T_i)$ show little change from the previous ones for all i (i.e., convergence) go to step 6. Else, go to step 5.

5. (Find full probabilities of merging queues and λ_i^*)

Obtain full probabilities for each of the merging queues using equation (3.14)

Find $\overline{\lambda_i}$ using equation (3.1) for $i=1, \cdots, K$

Set $\overline{\lambda} = \sum_{i=1}^{K} \overline{\lambda_i}$

Find λ_i^* and λ^* using equation (3.9).

Go to step 2.

6. (Calculate occupancy probabilities)

For queue 0, these have already been obtained in step 2. For queue 1 through K,

$$P_i(n) = \frac{(1-\rho_i)\rho_i^n}{1-\rho_i^{N_i+1}} \quad \text{for } i=1, \cdots, K, \ n=0, \cdots, N_i \tag{3.15}$$

3.3.2 M/G/1/N Analysis

The clearance time of each merging queue is in fact not exponentially distributed (as assumed in the section above) since there might be delay due to blocking. Therefore, in order to get a more accurate analysis, we can treat the clearance time of a merging queue as having a general distribution, which requires replacing the M/M/1/N analysis of step 3

of algorithm 1 with an M/G/1/N analysis.

The standard approach for M/G/1/N analysis is to consider the state occupancies only at departure epochs. Following this approach, define for each queue i the following;

$a_i^n \equiv$ Probability of n arrivals during a clearance time,

$\pi_i^n \equiv$ steady state probability of n units in queue i,

$\phi_i(t) \equiv$ p.d.f. of T_i,

$h_i(t)$, $[h(t)] \equiv$ p.d.f. of S_i $[S]$,

$\phi_i^{*(n)}(s)$, $[h_i^{*(n)}(s)] \equiv$ n-th derivatives of Laplace transforms of $\phi_i(t)$,$[h_i(t)]$,

Then, (see Pollock et.al [18] for details).

$$a_i^n = \frac{(-\lambda_i)^n}{n!}\phi_i^{*(n)}(\lambda_i) \tag{3.16}$$

In order to evaluate the right hand side of (3.16) we note that, from (3.2),

$$\phi_i^*(s) = \{1 - \sum_{j=0}^{K-1} \alpha_i(N+j)\}h_i^*(s) + \sum_{j=0}^{K-1} \alpha_i(N+j)\{h_i^*(s)\{h^*(s)\}^{j+1}\} \tag{3.17}$$

Taking the n-th derivative of both sides of (3.17)

$$\phi_i^{*(n)}(s) = \{1 - \sum_{j=0}^{K-1} \alpha_i(N+j)\}h_i^{*(n)}(s)$$

$$+ \sum_{j=0}^{K-1} \alpha_i(N+j) \sum_{k=0}^{n} \binom{n}{k} h_i^{*(n-k)}(s)\{(h^*(s))^{j+1}\}^k, \tag{3.18}$$

which allows (3.16) to be evaluated by setting $s=\lambda_i$.

Once the values of a_i^n are obtained, the steady state probabilities π_i^n follow from the usual M/G/1/N analysis. (see, e.g., Gross and Harris[10])

The steps in the resulting algorithm are as follows:

0,1,2. (See algorithm 1)

3. (Find expected clearance times at each merging queues.)

Calculate $\phi_i^{*(n)}(\lambda_i)$ using equation (3.18) and calculate a_i^n using equation (3.16) for $i=1,\cdots,K$ and $n=0,\cdots,N_i$

Set $E(T_i) = -\phi_i^{*(1)}(0)$ for $i=1,\cdots,K$

4. (Convergence check)

If updated values of $E(T_i)$ show little change from the previous ones for all i (i.e., convergence) go to step 6. Else, go to step 5.

5. (Find full probabilities of merging queues and λ_i^*)

Use M/G/1/N analysis to solve for $\pi_i^{N_i}$.

Set $f_i = \pi_i^{N_i}$ for $i=1,\cdots,N$.

Find $\overline{\lambda}_i$ using equation (3.1) for $i=1,\cdots,K$

Set $\overline{\lambda} = \sum_{i=1}^{K} \overline{\lambda}_i$

Find λ_i^* and λ^* using equation (3.9).

Go to step 2.

6. (Calculate occupancy probabilities)

For queue 0, these have already been obtained in step 2. For queue 1 through K, use M/G/1/N analysis to obtain the occupancy probabilities.

If the merging queues have infinite buffer spaces, since $\overline{\lambda}_i$ is simply equal to λ_i at

every iteration, the iterative procedure becomes simpler. For stability conditions when merging queues have infinite buffers, see Altiok and Perros[3].

5. Computational Results

The two approximation methods were tested for their accuracy in estimating steady state occupancy probabilities. In those cases where analytic solutions are not available the results are compared with simulations. Tables 1-6 shows a comparison with the method of Altiok and Perros [3], for systems consisting of from 2 to 4 merging queues. Tables 1 through 4 treat problems with finite buffers for the merging queues, and table 5 and 6 treat the problems with infinite buffers for the merging queues. Each table gives average and maximum absolute deviations of the approximate values from the exact or simulation ones.

In all experiments we have conducted, the number of iterations needed for convergence was under 7 and CPU time required was under 0.05 seconds on an IBM 3090-400. We should note that the Altiok and Perros method always overestimates the probability of queue 0 being full. Ours does not, and usually gives more accurate values. This is due to the fact that, in analysing queue 0, Altiok and Perros do not consider a reduction of arrival rates for the states where blocking exists, which fact our method takes into account. As can be seen in the tables, both algorithm 1 and algorithm 2 give better results than those of Altiok and Perros, in both average and maximum absolute deviations.

We can also see that algorithm 2 yields only slightly improved results over algorithm 1. This suggests that the exponential approximation for the clearance time is not unacceptable, at least for the cases we examined. This also suggests that a great part of the error may come from the (fundamentally faulty) Poisson assumption about the input

process to queue 0, not from the exponential assumption for the clearance time.

Algorithm 2 does, however, have the distinct advantage of allowing the service time distribution at the merging queues to be non-exponential, although this paper does not present numerical results for such cases.

6. Conclusions

We presented two approximation algorithms for analysing a merge configuration of queueing networks with blocking. These algorithms converge rapidly acceptably. Considering the simplicity of algorithm 1, we are optimistic about using it as a "building block" in the analysis of more general configurations of open queueing networks with blocking. It remains to be seen whether we can provide any error bounds, or provide explicit advice on parameter values for which the approximation is unequivocally recommended.

Acknowledgement

This study was supported in part by a contract from the General Motors Corporation to the University of Michigan, for the development of analytical tools for production systems. We wish, in particular, to thank Jeff Alden and Larry Burns, of G.M., for their continuing interest and support.

<Appendix>

(Proof of Theorem 3.1, 3.2)

Suppose we set $\lambda_a(i)$ as in (3.5),(3.6) and solved for P(i), i=0,⋯,N+K. Also suppose we obtained $P(N+n,i_1\cdots i_n)$ from (3.8). If the $P(N+n,i_1\cdots i_n)$ obtained thus satisfy the balance equation of the original system, then Theorem 3.1 and 3.2 must be true. The balance equations of the original system are

$$\overset{*}{\lambda} P(0) = \mu P(1) \qquad i=0 \tag{A.1}$$

$$(\overset{*}{\lambda}+\mu)P(i) = \overset{*}{\lambda} P(i-1) + \mu P(i+1) \qquad i=1,\cdots,N-1 \tag{A.2}$$

$$(\overset{*}{\lambda}+\mu)P(i) = \overset{*}{\lambda} P(i-1) + \mu \sum_{j=1}^{K} P(N+1, j) \qquad i=N \tag{A.3}$$

$$P(N+n, i_1\cdots i_n)\{\sum_{j\in\{i_1,\cdots,i_n\}} \overset{*}{\lambda}_j + \mu\} = \overset{*}{\lambda}_{i_n} P(N+n-1, i_1\cdots i_{n-1})$$

$$+ \sum_{j\in\{i_1,\cdots,i_n\}} \mu P(N+n+1, j i_1\cdots i_n) \qquad 1\le n\le K-1 \tag{A.4}$$

$$P(N+K,i_1\cdots i_K)\mu = \overset{*}{\lambda}_{i_k} P(N+K-1,i_1\cdots i_{K-1}) \qquad n=K \tag{A.5}$$

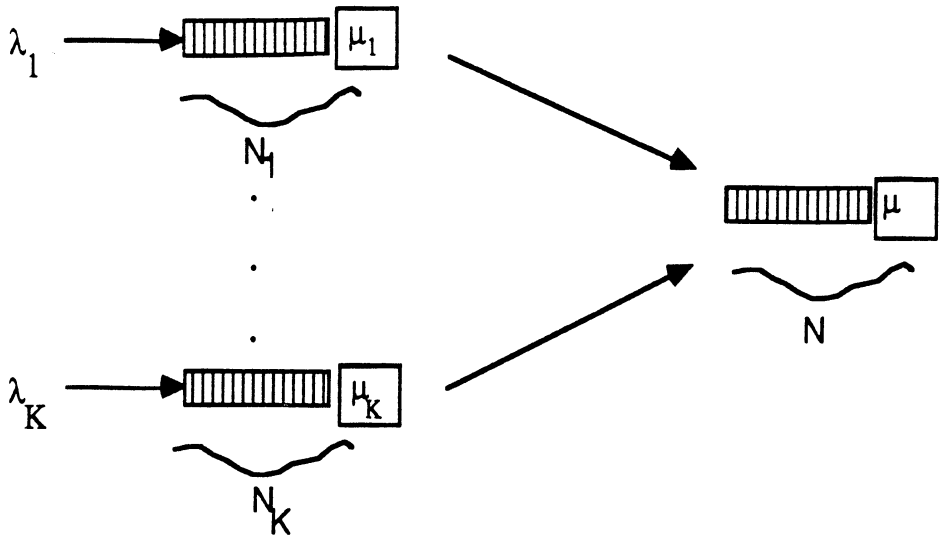
We now show equation (A.4) is satisfied by the solution of the aggregated system together with relationship (3.8). It is trivial to show that the balance equations of the aggregated system lead to the solutions

$$P(k) = P(k-1)\frac{\lambda_a(n)}{\mu} \qquad k = 1,\cdots,N+K \tag{A.6}$$

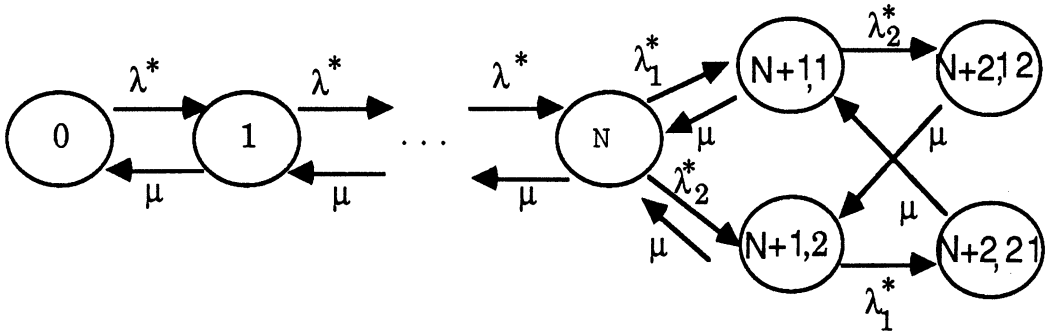
since the aggregated system is a simple birth-and-death process.

If we insert (A.6) together with the relationship (3.8) into (A.4), it is easily proved that (A.4) is satisfied. For the other balance equations, ,i.e., (A.1), (A.2), (A.3), (A.5), the same

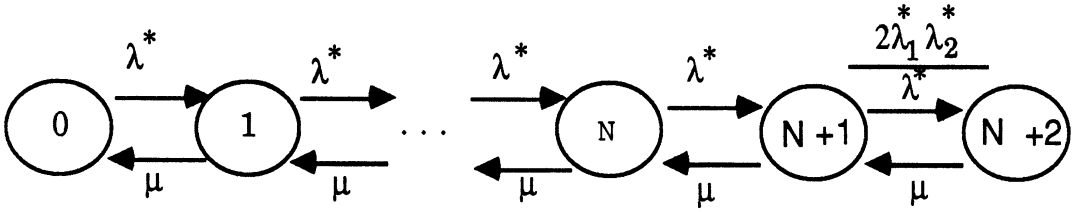
procedure is used. •



< Fig . 1 > The Merge Configuration



< Fig. 2> State Transition Diagram for Queue 0



< Fig. 3 > Aggregated State Transition Diagram

Table 1.

Problem Description		N_1=4 N_2=2 N =4; λ_1=4 λ_2=2; μ_1=5 μ_2=3 μ =7			
Measure		Exact	Altiok	Algo.1	Algo.2
Queue 1	$P_1(0)$	0.2408	0.2470	0.2519	0.2494
	$P_1(1)$	0.2196	0.2219	0.2228	0.2237
	$P_1(2)$	0.2004	0.1986	0.1970	0.1989
	$P_1(3)$	0.1835	0.1772	0.1742	0.1761
	$P_1(4)$	0.1556	0.1553	0.1541	0.1521
Queue 2	$P_2(0)$	0.4392	0.4621	0.4449	0.4422
	$P_2(1)$	0.3312	0.3200	0.3220	0.3275
	$P_2(2)$	0.2296	0.2179	0.2332	0.2302
Queue 0	P (0)	0.2820	0.2939	0.2975	0.2955
	P (1)	0.2214	0.2183	0.2231	0.2224
	P (2)	0.1708	0.1621	0.1672	0.1673
	P (3)	0.1294	0.1204	0.1254	0.1259
	P (4)	0.1963	0.2053	0.1869	0.1888
max. abs. deviation		0.0000	0.0229	0.0155	0.0135
avg. abs. deviation		0.0000	0.0080	0.0062	0.0047
throughput		4.9184	4.9430	4.9174	4.9314

Table 2

Problem Description		$N_1=N_2=2, N=3\ \lambda_1=\lambda_2=2,\ \mu_1=\mu_2=3,\ \mu=4$			
Measure		Exact	Altiok	Algo. 1	Algo. 2
Queue i=1,2	$P_i(0)$	0.3921	0.4205	0.4008	0.3972
	$P_i(1)$	0.3425	0.3283	0.3291	0.3353
	$P_i(2)$	0.2653	0.2512	0.2702	0.2674
Queue 0	P (0)	0.2551	0.2512	0.2702	0.2674
	P (1)	0.2248	0.2091	0.2222	0.2212
	P (2)	0.1891	0.1740	0.1828	0.1830
	p (3)	0.3310	0.3657	0.3248	0.3284
max. abs. deviation		0.0000	0.0347	0.0151	0.0123
avg. abs. deviation		0.0000	0.0180	0.0082	0.0056
throughput		2.9388	2.9952	2.9193	2.9303

Table 3

Problem Description		$N_1 = N_2 = N_3 = N_4 = 3$ $N = 5$ $\mu = 20$ $\lambda_1 = 2$ $\lambda_2 = 3$ $\lambda_3 = 4$ $\lambda_4 = 5$ $\mu_1 = 4$ $\mu_2 = 5$ $\mu_3 = 6$ $\mu_4 = 7$			
Measure		Simulation	Altiok	Algo. 1	Algo. 2
Queue 1	$P_1(0)$	0.5200	0.5314	0.5255	0.5250
	$P_1(1)$	0.2720	0.2673	0.2680	0.2695
	$P_1(2)$	0.1399	0.1341	0.1367	0.1368
	$P_1(3)$	0.0680	0.0672	0.0697	0.0687
Queue 2	$P_2(0)$	0.4466	0.4558	0.4501	0.4492
	$P_2(1)$	0.2706	0.2764	0.2763	0.2779
	$P_2(2)$	0.1732	0.1671	0.1696	0.1702
	$P_2(3)$	0.1094	0.1006	0.1041	0.1028
Queue 3	$P_3(0)$	0.4116	0.4095	0.4046	0.4036
	$P_3(1)$	0.2741	0.2774	0.2767	0.2783
	$P_3(2)$	0.1937	0.1873	0.1892	0.1901
	$P_3(3)$	0.1204	0.1257	0.1294	0.1280
Queue 4	$P_4(0)$	0.3765	0.3780	0.3745	0.3734
	$P_4(1)$	0.2780	0.2760	0.2751	0.2765
	$P_4(2)$	0.1990	0.2009	0.2020	0.2031
	$P_4(3)$	0.1462	0.1451	0.1484	0.1470
Queue 0	$P(0)$	0.3715	0.3833	0.3856	0.3847
	$P(1)$	0.2468	0.2376	0.2404	0.2402
	$P(2)$	0.1547	0.1473	0.1498	0.1500
	$P(3)$	0.1094	0.0913	0.0934	0.0936
	$P(4)$	0.0563	0.0566	0.0582	0.0585
	$P(5)$	0.0619	0.0839	0.0726	0.0731
	$P(6)$				
max. abs. deviation		0.0000	0.0220	0.0160	0.0158
avg. abs. deviation		0.0000	0.0066	0.0054	0.0053
throughput		12.3232	12.3355	12.2886	12.3070

Table 4

Problem Description		N_i =5 (i=0,1,2,3,4) $\lambda_1=\lambda_3=2$ $\lambda_2=\lambda_4=1$ $\mu_1=\mu_3=3$ $\mu_2=\mu_4=2$ $\mu=8$			
Measure		Simulation	Altiok	Algo. 1	Algo. 2
Queue i=1,3	$P_i(0)$	0.3239	0.3436	0.3334	0.3314
	$P_i(1)$	0.2531	0.2380	0.2364	0.2388
	$P_i(2)$	0.1649	0.1667	0.1676	0.1693
	$P_i(3)$	0.1157	0.1155	0.1188	0.1193
	$P_i(4)$	0.0857	0.0798	0.0842	0.0838
	$P_i(5)$	0.0563	0.0545	0.0597	0.0574
Queue i=2,4	$P_i(0)$	0.5002	0.5012	0.4846	0.4837
	$P_i(1)$	0.2627	0.2550	0.2547	0.2580
	$P_i(2)$	0.1355	0.1291	0.1339	0.1345
	$P_i(3)$	0.0599	0.0652	0.0704	0.0696
	$P_i(4)$	0.0299	0.0329	0.0370	0.0360
	$P_i(5)$	0.0113	0.0165	0.0194	0.0182
Queue 0	P (0)	0.2675	0.2814	0.2847	0.2833
	P (1)	0.2183	0.2057	0.2110	0.2104
	P (2)	0.1721	0.1503	0.1564	0.1563
	P (3)	0.1049	0.1099	0.1159	0.1161
	P (4)	0.0911	0.0803	0.0859	0.0863
	P (5)	0.1457	0.1725	0.1462	0.1476
max. abs. deviation		0.0000	0.0268	0.0172	0.0165
avg. abs. deviation		0.0000	0.0091	0.0080	0.0075
throughput		5.7522	5.7490	5.7224	5.7340

Table 5

Problem Description		$N_1 = N_2 = \infty$ $N = 3$ $\lambda_1 = \lambda_2 = \frac{1}{3}$ $\mu_1 = \mu_2 = \frac{1}{2}$ $\mu = 1$			
Measure		Simulation	Altiok	Algo. 1	Algo. 2
Queue i=1,2	$P_i(0)$	0.2563	0.2824	0.2545	0.2545
	$P_i(1)$	0.1999	0.2055	0.1898	0.1949
	$P_i(2)$	0.1412	0.1476	0.1415	0.1454
	$P_i(3)$	0.1077	0.1053	0.1055	0.1074
	$P_i(4)$	0.0676	0.0750	0.0786	0.0791
	$P_i(5)$	0.0600	0.0533	0.0586	0.0581
Queue 0	P (0)	0.3419	0.3333	0.3333	0.3333
	P (1)	0.2374	0.2363	0.2412	0.2412
	P (2)	0.1704	0.1676	0.1746	0.1746
	P (3)	0.2500	0.2628	0.2509	0.2509
max. abs. deviation		0.0000	0.0261	0.0110	0.0115
avg. abs. deviation		0.0000	0.0080	0.0044	0.0042
throughput		0.6667	0.6667	0.6667	0.6667

Table 6

Problem Description		$N_i = \infty$ (i=1,2,3,4) N =5 $\lambda_1 = \lambda_3 = 2$ $\lambda_2 = \lambda_4 = 1$ $\mu_1 = \mu_3 = 3$ $\mu_2 = \mu_4 = 2$ $\mu = 8$			
Measure		Simulation	Altiok	Algo. 1	Algo. 2
Queue i=1,3	$P_i(0)$	0.2899	0.2985	0.2824	0.2821
	$P_i(1)$	0.2154	0.2110	0.2031	0.2068
	$P_i(2)$	0.1355	0.1481	0.1460	0.1487
	$P_i(3)$	0.1070	0.1036	0.1050	0.1061
	$P_i(4)$	0.0747	0.0723	0.0755	0.0755
	$P_i(5)$	0.0600	0.0504	0.0543	0.0536
Queue i=2,4	$P_i(0)$	0.4712	0.4913	0.4677	0.4676
	$P_i(1)$	0.2570	0.2509	0.2490	0.2534
	$P_i(2)$	0.1269	0.1274	0.1325	0.1335
	$P_i(3)$	0.0691	0.0645	0.0706	0.0697
	$P_i(4)$	0.0379	0.0326	0.0376	0.0363
	$P_i(5)$	0.0185	0.0165	0.0200	0.0189
Queue 0	P (0)	0.2588	0.2500	0.2510	0.2509
	P (1)	0.1874	0.1919	0.1970	0.1970
	P (2)	0.1508	0.1474	0.1546	0.1546
	P (3)	0.1056	0.1132	0.1214	0.1214
	P (4)	0.0957	0.0868	0.0953	0.0953
	P (5)	0.2014	0.2107	0.1807	0.1809
max. abs. deviation		0.0000	0.0201	0.0207	0.0205
avg. abs. deviation		0.0000	0.0068	0.0065	0.0062
throughput		6.0000	6.0000	6.0000	6.0000

REFERENCES

[1] Akyildiz,I.F., "Analysis of Closed Queuing Networks with Blocking", CS Dept., 85-022(1985),Lousiana State Univ.,1985

[2] Altiok,T.,"Approximate Analysis of Exponential Tandem Queues with Blocking",Europ. J. Oper. Res. 11, 390-398,1982

[3] Altiok,T.,and H.G. Perros, "Open Networks of Queues with Blocking: Split and Merge Configurations",AIIE Trans. 18,251-261,1986

[4] Bocharov,P.P. and G.P. Rokhas, "On an Exponential Queueing System in Series with Blocking", Problems of Control and Information Theory 9,441-455,1980

[5] Boxma.O.J. and A.G.Konheim, "Approximate Analysis of Exponential Queueing Systems with Blocking,"Acta Informatica 15,19-66,1981

[6] Brandwajn,A. and Y.L.Jow, "An Approximate Method for Tandem Queues with Blocking",Manuscript,Amdahl Corp.,1985

[7] Caseau,P. and G.Pujolle, "Throughput Capacity of a Sequence of Queues with Blocking due to Finite Waiting Room,"IEEE Trans. Soft. Eng. SE-5,631-642,1979

[8] Foster,F.G. and H.G. Perros, "On the Blocking Process in Queue Networks", Europ. J. Oper. Res. 5,276-283,1980

[9] Gershwin,S.B., "An Efficient Decomposition Method for the Approximate Evaluation of Tandem Queues with Finite Storage and Blocking", Manuscript,Lab. for Information and Decision Sciences,MIT,1983

[10] Gross,D., and C.M.Harris, Fundamentals of Queueing Theory, New York, John Wiley and Sons, 1985

[11] Hillier,F.S. and R. Boling, "Finite Queues in Series with Exponential or Erlang Service Times – Numerical Approach", Oper. Res. 15,286-303,1967

[12] Labetoulle,J.and G.Pujolle, "Isolation Method in a Networks of Queues", IEEE

UNIVERSITY OF MICHIGAN
3 9015 04732 7120

Trans. Soft. Eng. SE-6,1980

[13] Onvural,R.O. and H.G. Perros, "On Equivalencies of Blocking Mechanisms in Queueing Networks with Blocking", Oper. Res. Letter 5,293-297,1986

[14] Perros,H.G., "A Servey of Queueing Networks with Blocking-Part I", Computer Science Report,N.C.State Univ.,1986

[15] Peross,H.G. and T. Altiok, "Approximate Analysis of Open Networks of Queues with Blocking: Tandem Configurations",IEEE Trans on Soft. Eng., SE-12,1986

[16] Perros,H.G. and T. Altiok, "Approximate Analysis of Arbitrary Configurations of Open Queueing Networks with Blocking",to Appear in the Annals of OR

[17] Perros,H.G. and P.M. Snyder, "A Computationally Efficient Approximation Algorithm for Analyzing Open Queueing Networks with Blocking",Manuscript, CS Dept., N.C. State Univ.,1986

[18] Pollock,S.M.,J.R.Birge and J.M.Alden, "Approximation Analysis for Open Tandem Queues with Blocking: Exponential and General Service Distributions", Technical Report,IOE Dept.,85-30, Univ. of Michigan,1985

[19] Suri,R. and G.W.Diehl, "A Variable Buffer-size Model and its use in Analytic Closed Queueing Networks with Blocking,"Proc. ACM SIGMETRICS on Measurement and Modelling of Computer Systems,134-142,1984

[20] Suri,R. and G.W.Diehl, "A Variable Buffer-size Model and its use in Analyzing Closed Queueing Networks with Blocking",Management Sci.,Vol 32, No.2,1986

[21] Takahashi,Y,H.Miyahara and T.Hasegawa, "An Approximation Method for Open Restricted Queueing Networks",J.Oper. Res. 28,594-602,1980

[22] Yao,D. and J.A.Buzacott, "Modelling a Class of Flexible Manufacturing Systems with Reversible Routing",Manuscript,I.E.Dept., Columbia Univ.,1983