A MONOTONICITY THEOREM FOR TANDEM QUEUES

Y.C. Tay

CS-TR-362-92

February 1992

A Monotonicity Theorem for Tandem Queues

Y.C. Tay *
Department of Computer Science
Princeton University
yct@cs.princeton.edu

Abstract Tandem queues are often used for performance modeling in production lines, computer systems and communication networks. Consider a series of tandem first-come-first-served queues with single servers. The buffers may be finite or infinite. When a finite buffer is full, the upstream server may suffer from transfer blocking or service blocking. A job's service requirements at various servers may be correlated. A transient analysis shows that, if the network is open, then response time decreases if a server is speeded up, or a finite buffer increases in size, or service blocking is changed to transfer blocking at some server; if the network is closed, then such changes increase the throughput. The analytic technique is elementary, and the equations can be used to simulate a network.

1 Introduction

Queueing networks with finite buffers have been used to model manufacturing systems, computer systems and communication systems [A, AS, BI, BS, DY, KR, P, R, SD]. These networks are difficult to analyze, and even simple, intuitively obvious properties are hard to prove. For example, one expects that the throughput of a closed network will be higher if its finite buffers are replaced by infinite buffers, but this has not been proved [O]. We address this and related problems here in the case of tandem queues.

Consider a network of first-come-first-served queues in tandem. The queues may have finite or infinite buffers. If a finite buffer is full, the blocking scheme may be transfer blocking or service blocking. (A transaction processing system, for example, may have such a mix of blocking schemes [O].) For a given job, its service times at different servers may be correlated (as is the case for the transmission times of a message as it hops from node to node in a communication network [CP]). We give a transient analysis that shows, for an open system, the response time of a job decreases if (1) the service times at some server decrease, or (2) service blocking at some finite buffer is replaced by transfer blocking, or (3) the finite buffer at some queue increases in size. The analysis also shows, for a closed system, that throughput increases under similar circumstances. Such monotonicity results can be used to bound network performance [O, SY] (which is one reason for allowing a mix of finite and infinite buffers).

We formalize the system in Section 2, derive equations that describe the path of a job through the system, and provide estimators for various performance measures. We then give the monotonicity proofs in Section 3, and contrast the results with related work in Section 4.

* On sabbatical from Dept. of Mathematics, National Univ. of Singapore (mattyc@nuscc.nus.sg).

2 Equations

Consider M first-come-first-served queues with single servers and in tandem. For each m, $1 \leq m \leq M$, queue m has a buffer consisting of B_m buffer *slots*, for some positive integer B_m. The server is at the first slot, and is referred to as a *real* server; each of the other $B_m - 1$ slots has a *virtual* server. A virtual server has zero service time.

The first $B_m - 1$ slots in a queue are *finite* — they contain at most one job each; the last (i.e B_m-th) slot may be *infinite*, in which case it can contain any number of jobs. Thus a finite buffer consists of some B_m finite slots, and an infinite buffer consists of one infinite slot and $B_m - 1$ finite slots. Figure 1 illustrates our model.

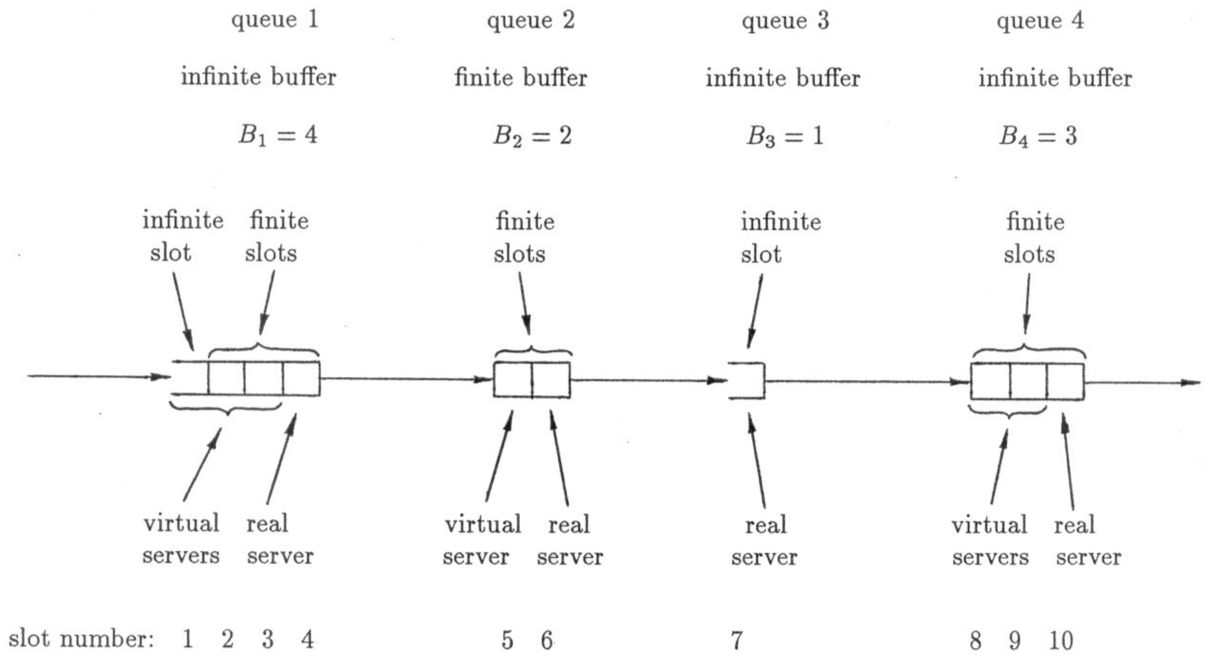

Figure 1 Terminology

We number each slot in increasing order, following the direction traveled by the jobs. Thus, a job enters the system when it enters slot 1, proceeds from slot b to slot $b+1$ for each b, and leaves the system when it finishes service at slot $B = B_1 + \cdots + B_M$. We assume slot 1 is infinite. These are illustrated in Figure 1.

Consider a real server who is at slot b, where slot $b+1$ is finite ($b < B_M$). In *transfer blocking*, service begins once a job is before the server; when service ends, the job proceeds to slot $b+1$ if the latter is empty, or else continues to occupy slot b for as long as slot $b+1$ is occupied. In *service blocking*, a job that arrives before the server does not receive service for as long as slot $b+1$ is occupied.

Let $T_0^{(j)}$ be the arrival time (at slot 1) of the j-th job, where $j = 0, 1, 2, \ldots$. For positive integer b, let $T_b^{(j)}$ be the time when job j leaves slot b. Let $S_b^{(j)}$ be the service time of job j at slot b (so $S_b^{(j)} = 0$ if slot b has a virtual server).

Open and Closed Systems

The system is *closed* if every job, upon departure from queue M (equivalently, slot B), triggers the arrival of another job. The total number of jobs in the system is therefore a constant, say N, so $T_0^{(j)} = T_B^{(j-N)}$ for $j = N, N+1, \ldots$. A system is *open* if the values of $T_0^{(j)}$ does not depend on the form of blocking, nor on B_m and $S_b^{(k)}$, for all m, b and k.

Thus, an open system is completely specified by M, B_m, the finiteness of each slot, the form of blocking, $T_0^{(j)}$, $S_b^{(j)}$, and the initialization of job 0; a closed system is completely specified by N, M, B_m, the finiteness of each slot, the form of blocking, $S_b^{(j)}$, and the initialization of the first N jobs.

Initialization

There are various ways to initialize the system. We assume that job 0 is initialized thus:

$$T_b^{(0)} = S_1^{(0)} + S_2^{(0)} + \cdots + S_b^{(0)} \qquad \text{for} \quad 1 \leq b \leq B. \tag{1}$$

For a meaningful initialization of a closed system, we assume — in addition to (1) — that $B_1 \geq N$ (equivalently, slot 1 is infinite), and the first N jobs start at queue 1; i.e.

$$T_0^{(0)} = T_0^{(1)} = \cdots = T_0^{(N-1)} = 0. \tag{2}$$

Virtual Server

Consider the j-th job entering slot b, at time $T_{b-1}^{(j)}$, where $b \geq 1$. If slot b has a virtual server, then job j passes through immediately if slot $b+1$ is empty, so $T_b^{(j)} = T_{b-1}^{(j)}$. Otherwise, it stays in slot j for as long as slot $j+1$ is occupied, so $T_b^{(j)} = T_{b+1}^{(j-1)}$. Therefore

$$T_b^{(j)} = \max(T_{b-1}^{(j)} + S_b^{(j)},\ T_{b+1}^{(j-1)} + S_b^{(j)}). \tag{3}$$

Real Server: No Blocking

Suppose slot b has a real server who is never blocked (i.e. $b = B$, or slot $b+1$ is infinite). If $T_{b-1}^{(j)} < T_b^{(j-1)}$, then job j enters slot b before the previous job leaves (this can happen if slot b is infinite), so job j must wait until $T_b^{(j-1)}$ to get service. On the other hand, if $T_{b-1}^{(j)} > T_b^{(j-1)}$, then job j gets service immediately. Thus

$$T_b^{(j)} = \max(T_{b-1}^{(j)},\ T_b^{(j-1)}) + S_b^{(j)},$$

$$\text{or} \qquad T_b^{(j)} = \max(T_{b-1}^{(j)} + S_b^{(j)},\ T_b^{(j-1)} + S_b^{(j)}). \tag{4}$$

Real Server: Transfer Blocking

Consider now a slot b with a real server who suffers from transfer blocking by (finite) slot $b+1$. As in (4), service for job j ends at $t = \max(T_{b-1}^{(j)},\ T_b^{(j-1)}) + S_b^{(j)}$. If $t > T_{b+1}^{(j-1)}$, then job j finishes service at slot b after the previous job vacates the next slot, so job j can proceed immediately to slot $b+1$. However, if $t < T_{b+1}^{(j-1)}$, then job j must wait till $T_{b+1}^{(j-1)}$ before it can leave slot b. Thus

$$T_b^{(j)} = \max(\max(T_{b-1}^{(j)},\ T_b^{(j-1)}) + S_b^{(j)},\ T_{b+1}^{(j-1)})$$

$$\text{or} \qquad T_b^{(j)} = \max(T_{b-1}^{(j)} + S_b^{(j)},\ T_b^{(j-1)} + S_b^{(j)},\ T_{b+1}^{(j-1)}). \tag{5}$$

Real Server: Service Blocking

Finally consider a slot b with a real server who suffers from service blocking. If $T_{b-1}^{(j)} > T_{b+1}^{(j-1)}$, then slot $b+1$ is vacant when job j enters slot b, so job j is not blocked and can be served immediately. If $T_{b-1}^{(j)} < T_{b+1}^{(j-1)}$, then job j is either waiting (if slot b is infinite) or blocked at slot b, and must wait till $T_{b+1}^{(j-1)}$ to get service. Therefore,

$$T_b^{(j)} = \max(T_{b-1}^{(j)},\ T_{b+1}^{(j-1)}) + S_b^{(j)},$$

$$\text{or} \qquad T_b^{(j)} = \max(T_{b-1}^{(j)} + S_b^{(j)},\ T_{b+1}^{(j-1)} + S_b^{(j)}). \tag{6}$$

Performance Measures

We now list our estimators for the usual performance measures:

throughput $$\hat{\lambda} = \frac{n}{T_B^{(n)}}$$

response time $$\hat{R} = \frac{1}{n}\sum_{j=1}^{n}(T_B^{(j)} - T_0^{(j)})$$

Let b_m denote the slot number for the (real) server of queue m, $1 \le m \le M$.

(real) server utilization $$\hat{\rho}_m = \frac{\hat{\lambda}}{n}\sum_{j=1}^{n} S_{b_m}^{(j)}$$

slot utilization $$\hat{u}_b = \frac{\hat{\lambda}}{n}\sum_{j=1}^{n}(T_b^{(j)} - T_{b-1}^{(j)})$$

queue length $$\hat{Q}_m = \frac{\hat{\lambda}}{n}\sum_{j=1}^{n}(T_{b_m}^{(j)} - T_{b_m-1}^{(j)})$$

Let $p_{m,k}$ be the probability that the queue length (including the job in service) at queue m exceeds k. Then

queue length distribution $$\hat{p}_{m,k} = \hat{u}_{b_m-k} \qquad \text{if slot } b_m - k \text{ is finite and } k < B_m$$

In transfer blocking, we say (real) server m is *blocked* if and only if b_m is occupied by a job whose service has ended; in service blocking, server m is *blocked* if and only if slot $b_m + 1$ is full. Let $p_{m,blocked}$ be the probability that server m is blocked. Then

transfer blocking $$\hat{p}_{m,blocked} = \hat{u}_{b_m} - \hat{\rho}_{b_m}$$

service blocking $$\hat{p}_{m,blocked} = \hat{u}_{b_m+1}$$

3 Analysis

The usual starting point for analyzing tandem queues is to assume that the arrival times $T_0^{(j)}$ are independent and identically distributed random variables, and similarly for the service times $S_b^{(j)}$, for $j = 0, 1, \ldots$. Moreover, $S_b^{(j)}$ and $S_{b'}^{(j)}$ would be independent for $b \neq b'$. This latter assumption is unsatisfactory, since it is reasonable to expect that a job's service times at different servers would be correlated.

The model we have in mind consists of a network description Q, and a sample space $\mathcal{S}(Q)$ for each Q. Q is specified by the number of queues (M), the buffer sizes (B_m), the finiteness of each buffer slot, the form of blocking and, for each m, a service time s_m in time units per work unit. The sample space $\mathcal{S}(Q)$ consists of job-lists; a job-list is a sequence of jobs (from a single job class) specified by the service requirement $W_m^{(j)}$, $1 \leq m \leq M$, in work units for each job j in the sequence, and the jobs' arrival times $T_0^{(j)}$ if Q is open, or the number of jobs N if Q is closed. Thus, $S_{b_m}^{(j)} = s_m W_m^{(j)}$, where b_m is the slot number for server m.

In the case of an open system, for example, one could construct $\mathcal{S}(Q)$ as the product of two sample spaces, one of which specifies $T_0^{(j)}, j = 0, 1, \ldots$ as, say, independent, identically and exponentially distributed random variables, while the other specifies $W_m^{(j)}$ (the latter can be similarly decomposed).

In our monotonicity result, we will compare two networks Q and $\tilde{Q}$ on the same sample space ($\mathcal{S}(Q) = \mathcal{S}(\tilde{Q})$) by comparing their performance on the *same* sample point (i.e. job-list).

To begin, we recall equations (3)–(6):

$$T_b^{(j)} = \begin{cases} \max(T_{b-1}^{(j)} + S_b^{(j)},\ T_b^{(j-1)} + S_b^{(j)}) & \text{real server, no blocking} \\ \max(T_{b-1}^{(j)} + S_b^{(j)},\ T_b^{(j-1)} + S_b^{(j)},\ T_{b+1}^{(j-1)}) & \text{real server, transfer blocking} \\ \max(T_{b-1}^{(j)} + S_b^{(j)},\ T_{b+1}^{(j-1)} + S_b^{(j)}) & \text{real server, service blocking} \\ \max(T_{b-1}^{(j)} + S_b^{(j)},\ T_{b+1}^{(j-1)} + S_b^{(j)}) & \text{virtual server} \end{cases} \tag{7}$$

For a given network Q, let $\alpha(Q)$ denote the variable α for that network, e.g. $T_b^{(j)}(Q)$ and $\hat{\lambda}(Q)$.

Lemma 1
Suppose Q and $\tilde{Q}$ are identical networks except, for some server m, $s_m(Q) \leq s_m(\tilde{Q})$. Then $T_b^{(j)}(Q) \leq T_b^{(j)}(\tilde{Q})$ for all b and j.

Proof
Consider induction on b and j. Clearly, from the way Q and $\tilde{Q}$ are initialized, $T_b^{(0)}(Q) \leq T_b^{(0)}(\tilde{Q})$ for all $1 \leq b \leq B$. Now suppose there is $k \geq 1$ such that $T_b^{(j)}(Q) \leq T_b^{(j)}(\tilde{Q})$ for all $1 \leq b \leq B$ and $j < k$.

If Q and $\tilde{Q}$ are open, then $T_0^{(k)}(Q) = T_0^{(k)}(\tilde{Q})$; if they are closed, then $T_0^{(k)}(Q) = T_0^{(k)}(\tilde{Q})$ if $k < N$ and $T_0^{(k)}(Q) = T_B^{(k-N)}(Q) \leq T_B^{(k-N)}(\tilde{Q}) = T_0^{(k)}(\tilde{Q})$ if $k \geq N$, by the induction hypothesis.

In any case, we have $T_0^{(k)}(Q) \leq T_0^{(k)}(\tilde{Q})$. Starting with this fact, it now follows from the induction hypothesis and equation (7) that $T_b^{(k)}(Q) \leq T_b^{(k)}(\tilde{Q})$ for $1 \leq b \leq B$. □

Lemma 2
Consider (real) server m in a network Q, $m < M$.

(i) Suppose server m suffers from transfer blocking. If this is changed to service blocking, and the resulting network is $\tilde{Q}$, then $T_b^{(j)}(Q) \leq T_b^{(j)}(\tilde{Q})$ for all b and j.

(ii) Suppose slot $b_m + 1$ is infinite. If this is changed into a finite slot, and the resulting network is $\tilde{Q}$, then $T_b^{(j)}(Q) \leq T_b^{(j)}(\tilde{Q})$ for all b and j.

Proof

(i) All equations in Q and $\tilde{Q}$ are the same, except

$$\begin{aligned} & T_{b_m}^{(j)}(Q) = \max(T_{b_m-1}^{(j)}(Q) + S_{b_m}^{(j)},\ T_{b_m}^{(j-1)}(Q) + S_{b_m}^{(j)},\ T_{b_m+1}^{(j-1)}(Q)) \\ \text{and} \quad & T_{b_m}^{(j)}(\tilde{Q}) = \max(T_{b_m-1}^{(j)}(\tilde{Q}) + S_{b_m}^{(j)},\ T_{b_m+1}^{(j-1)}(\tilde{Q}) + S_{b_m}^{(j)})\ . \end{aligned} \tag{8}$$

The proof is by induction on b and j, as in Lemma 1, and the induction step follows from (8) (and $T_b^{(j)} \leq T_{b+1}^{(j)}$).

(ii) Consider first the case where server m in $\tilde{Q}$ suffers from transfer blocking. All equations in Q and $\tilde{Q}$ remain the same, except

$$T_{b_m}^{(j)}(Q) = \max(T_{b_m-1}^{(j)}(Q) + S_{b_m}^{(j)},\ T_{b_m}^{(j-1)}(Q) + S_{b_m}^{(j)})$$

and $$T_{b_m}^{(j)}(\tilde{Q}) = \max(T_{b_m-1}^{(j)}(\tilde{Q}) + S_{b_m}^{(j)},\ T_{b_m}^{(j-1)}(\tilde{Q}) + S_{b_m}^{(j)},\ T_{b_m+1}^{(j-1)}(\tilde{Q}))\ .$$

Induction proves the claim.

The case where server m in $\tilde{Q}$ suffers from service blocking now follows from (i). □

Lemma 3
Suppose slot k is finite, and is the last slot for queue m ($m > 1$) in network Q (i.e. $k = b_{m-1} + 1$). Assume queue m has more than one slot in its (finite) buffer, so $B_m > 1$. Delete slot k to obtain a network $\tilde{Q}$ (so queue m in $\tilde{Q}$ has $B_m - 1$ slots), but retain the slot numbering (so $\tilde{Q}$ has slot numbers $1, 2, \ldots, k-1 = b_{m-1}, k+1 = b_{m-1}+2, \ldots, b_M$). If server $m-1$ suffers from transfer blocking, then $T_b^{(j)}(Q) \le T_b^{(j)}(\tilde{Q})$ for all j and $b \neq k$.

Proof
All equations in $\tilde{Q}$ are the same as for Q, except for $b = k-1$ and $b = k+1$. Accordingly, we rewrite the equations for $k-1$ and $k+1$.

$$\begin{aligned} T_{k-1}^{(j)}(Q) &= \max(T_{k-2}^{(j)}(Q) + S_{k-1}^{(j)},\ T_{k-1}^{(j-1)}(Q) + S_{k-1}^{(j)},\ T_{k}^{(j-1)}(Q)) \\ &= \max(T_{k-2}^{(j)}(Q) + S_{k-1}^{(j)},\ T_{k-1}^{(j-1)}(Q) + S_{k-1}^{(j)},\ \max(T_{k-1}^{(j-1)}(Q),\ T_{k+1}^{(j-2)}(Q))) \\ &= \max(T_{k-2}^{(j)}(Q) + S_{k-1}^{(j)},\ T_{k-1}^{(j-1)}(Q) + S_{k-1}^{(j)},\ T_{k+1}^{(j-2)}(Q)) \end{aligned}$$

$$\begin{aligned} T_{k-1}^{(j)}(\tilde{Q}) &= \max(T_{k-2}^{(j)}(\tilde{Q}) + S_{k-1}^{(j)},\ T_{k-1}^{(j-1)}(\tilde{Q}) + S_{k-1}^{(j)},\ T_{k+1}^{(j-1)}(\tilde{Q})) \\ &= \max(T_{k-2}^{(j)}(\tilde{Q}) + S_{k-1}^{(j)},\ T_{k-1}^{(j-1)}(\tilde{Q}) + S_{k-1}^{(j)},\ T_{k+1}^{(j-2)}(\tilde{Q}),\ T_{k+1}^{(j-1)}(\tilde{Q})) \\ &\qquad \text{since} \quad T_{k+1}^{(j-1)}(\tilde{Q}) \ge T_{k+1}^{(j-2)}(\tilde{Q}) \end{aligned}$$

For slot $k+1$, consider the following three cases:

(1) If slot $k+1$ has a real server with no blocking, then

$$\begin{aligned} T_{k+1}^{(j)}(Q) &= \max(T_{k}^{(j)}(Q) + S_{k+1}^{(j)},\ T_{k+1}^{(j-1)}(Q) + S_{k+1}^{(j)}) \\ &= \max(\max(T_{k-1}^{(j)}(Q),\ T_{k+1}^{(j-1)}(Q)) + S_{k+1}^{(j)},\ T_{k+1}^{(j-1)}(Q) + S_{k+1}^{(j)}) \\ &= \max(T_{k-1}^{(j)}(Q) + S_{k+1}^{(j)},\ T_{k+1}^{(j-1)}(Q) + S_{k+1}^{(j)}) \end{aligned}$$

$$T_{k+1}^{(j)}(\tilde{Q}) = \max(T_{k-1}^{(j)}(\tilde{Q}) + S_{k+1}^{(j)},\ T_{k+1}^{(j-1)}(\tilde{Q}) + S_{k+1}^{(j)}).$$

(2) If slot $k+1$ has a real server with transfer blocking, then

$$\begin{aligned}T_{k+1}^{(j)}(Q) &= \max(T_k^{(j)}(Q) + S_{k+1}^{(j)},\ T_{k+1}^{(j-1)}(Q) + S_{k+1}^{(j)},\ T_{k+2}^{(j-1)}(Q)) \\ &= \max(\max(T_{k-1}^{(j)}(Q),\ T_{k+1}^{(j-1)}(Q)) + S_{k+1}^{(j)},\ T_{k+1}^{(j-1)}(Q) + S_{k+1}^{(j)},\ T_{k+2}^{(j-1)}(Q)) \\ &= \max(T_{k-1}^{(j)}(Q) + S_{k+1}^{(j)},\ T_{k+1}^{(j-1)}(Q) + S_{k+1}^{(j)},\ T_{k+2}^{(j-1)}(Q))\end{aligned}$$

$$T_{k+1}^{(j)}(\tilde{Q}) = \max(T_{k-1}^{(j)}(\tilde{Q}) + S_{k+1}^{(j)},\ T_{k+1}^{(j-1)}(\tilde{Q}) + S_{k+1}^{(j)},\ T_{k+2}^{(j-1)}(\tilde{Q})).$$

(3) If slot $k+1$ has a real server with service blocking, or a virtual server, then

$$\begin{aligned}T_{k+1}^{(j)}(Q) &= \max(T_k^{(j)}(Q) + S_{k+1}^{(j)},\ T_{k+2}^{(j-1)}(Q) + S_{k+1}^{(j)}) \\ &= \max(\max(T_{k-1}^{(j)}(Q),\ T_{k+1}^{(j-1)}(Q)) + S_{k+1}^{(j)},\ T_{k+2}^{(j-1)}(Q) + S_{k+1}^{(j)}) \\ &= \max(T_{k-1}^{(j)}(Q) + S_{k+1}^{(j)},\ T_{k+1}^{(j-1)}(Q) + S_{k+1}^{(j)},\ T_{k+2}^{(j-1)}(Q) + S_{k+1}^{(j)}) \\ &= \max(T_{k-1}^{(j)}(Q) + S_{k+1}^{(j)},\ T_{k+2}^{(j-1)}(Q) + S_{k+1}^{(j)})\end{aligned}$$

$$T_{k+1}^{(j)}(\tilde{Q}) = \max(T_{k-1}^{(j)}(\tilde{Q}) + S_{k+1}^{(j)},\ T_{k+2}^{(j-1)}(\tilde{Q}) + S_{k+1}^{(j)}).$$

The proof now proceeds with induction on b and j. From the initialization step, $T_b^{(0)}(Q) = T_b^{(0)}(\tilde{Q})$ for all b, since $S_k^{(0)} = 0$ (the deleted slot has a virtual server). Now suppose for some $h \geq 1$, $T_b^{(j)}(Q) \leq T_b^{(j)}(\tilde{Q})$ for all $j < h$ and all b.

If Q and $\tilde{Q}$ are open, then $T_0^{(h)}(Q) = T_0^{(h)}(\tilde{Q})$; if they are closed, then $T_0^{(h)}(Q) = T_0^{(h)}(\tilde{Q})$ if $h < N$ and $T_0^{(h)}(Q) = T_B^{(h-N)}(Q) \leq T_B^{(h-N)}(\tilde{Q}) = T_0^{(h)}(\tilde{Q})$ if $h \geq N$, by the induction hypothesis. In any case, we have $T_0^{(h)}(Q) \leq T_0^{(h)}(\tilde{Q})$.

Now consider the equations for $T_b^{(h)}(Q)$ and $T_b^{(h)}(\tilde{Q})$ for $b \geq 1$. By induction on b (using the equations above for $b = k-1$ and $b = k+1$) and applying the induction hypothesis, we get $T_b^{(h)}(Q) \leq T_b^{(h)}(\tilde{Q})$ for all b. □

Lemma 4

Suppose slot k is finite, and is the last slot for queue m ($m > 1$) in network Q. Assume queue m has more than one slot in its buffer, so $B_m > 1$. Delete slot k to obtain a network $\tilde{Q}$, but retain the slot numbering. If server $m-1$ suffers from service blocking, then $T_b^{(j)}(Q) \leq T_b^{(j)}(\tilde{Q})$ for all j and $b \neq k$.

Proof

As in Lemma 4, we first derive the equations for $b = k-1$ in Q and $\tilde{Q}$.

$$\begin{aligned}T_{k-1}^{(j)}(Q) &= \max(T_{k-2}^{(j)}(Q) + S_{k-1}^{(j)},\ T_k^{(j-1)}(Q) + S_{k-1}^{(j)}) \\ &= \max(T_{k-2}^{(j)}(Q) + S_{k-1}^{(j)},\ \max(T_{k-1}^{(j-1)}(Q),\ T_{k+1}^{(j-2)}(Q)) + S_{k-1}^{(j)}) \\ &= \max(T_{k-2}^{(j)}(Q) + S_{k-1}^{(j)},\ T_{k-1}^{(j-1)}(Q) + S_{k-1}^{(j)},\ T_{k+1}^{(j-2)}(Q) + S_{k-1}^{(j)})\end{aligned}$$

$$\begin{aligned}T_{k-1}^{(j)}(\tilde{Q}) &= \max(T_{k-2}^{(j)}(\tilde{Q}) + S_{k-1}^{(j)},\ T_{k+1}^{(j-1)}(\tilde{Q}) + S_{k-1}^{(j)}) \\ &= \max(T_{k-2}^{(j)}(\tilde{Q}) + S_{k-1}^{(j)},\ T_{k-1}^{(j-1)}(\tilde{Q}) + S_{k-1}^{(j)},\ T_{k+1}^{(j-2)}(\tilde{Q}) + S_{k-1}^{(j)},\ T_{k+1}^{(j-1)}(\tilde{Q}) + S_{k-1}^{(j)}) \\ &\qquad \text{since}\quad T_{k+1}^{(j-1)}(\tilde{Q}) \geq T_{k-1}^{(j-1)}(\tilde{Q}) \quad\text{and}\quad T_{k+1}^{(j-1)}(\tilde{Q}) \geq T_{k+1}^{(j-2)}(\tilde{Q})\end{aligned}$$

For slot $k+1$, the equations are as in the proof for Lemma 4. An induction on b and j, similar to the one for Lemma 4, completes the proof. □

We reiterate that in the comparison below, the networks are evaluated on the same job-list.

Theorem
Let Q be a network.

(I) Let $\tilde{Q}_1$ be identical to Q except, for some server m, $s_m(Q) \leq s_m(\tilde{Q}_1)$.

(II) Let $\tilde{Q}_2$ be identical to Q except, for some server m in Q that suffers from transfer blocking, server m in $\tilde{Q}_2$ suffers from service blocking.

(III) Let $\tilde{Q}_3$ be identical to Q except some infinite buffer in Q becomes a finite buffer in $\tilde{Q}_3$.

(IV) Let $\tilde{Q}_4$ be identical to Q except some finite buffer in Q has one slot less in $\tilde{Q}_4$.

If Q is open, then $\hat{R}(Q) \leq \hat{R}(\tilde{Q}_i)$ for $i = 1,2,3,4$. If Q is closed, then $\hat{\lambda}(Q) \geq \hat{\lambda}(\tilde{Q}_i)$ for $i = 1,2,3,4$.

Proof
If Q is open, we have

$$\hat{R}(\tilde{Q}_i) - \hat{R}(Q) = \frac{1}{n}\sum_{j=1}^{n}\left(T_B^{(j)}(\tilde{Q}_i) - T_B^{(j)}(Q)\right).$$

If Q is closed, then

$$\hat{\lambda}(Q) - \hat{\lambda}(\tilde{Q}_i) = \frac{n}{T_B^{(n)}(Q)} - \frac{n}{T_B^{(n)}(\tilde{Q}_i)}.$$

The lemmas now prove the claim. □

4 Related Work

In the discussion below, we gloss over minor differences in the models; e.g. for open networks, Tsoucas and Walrand have a finite buffer for the first queue, and discard arrivals who find the buffer full [TW]; for closed networks, Shanthikumar and Yao allow the jobs to be arbitrarily distributed when the network is initialized [SY].

For an open network, monotonicity in (II) and (IV) of our theorem was proved by Tsoucas and Walrand in the case where all buffers are finite and the servers (except the last) either all suffer from transfer blocking or all suffer from service blocking. Their results extend to multiserver queues.

For a closed network, monotonicity in (I) and (IV) was proved by Shanthikumar and Yao. Again, all servers have finite buffers and the same blocking scheme, but may have queuedependent service times. They also proved monotonicity with respect to population size (for $N < \max(B_1, \ldots, B_M)$).

Thus, there is currently no result in the literature that allows the arbitrary mix of finite and infinite buffers, as well as blocking schemes, that we consider in our theorem. However, there is considerable overlap between this work and recent work by Cheng, Glasserman and Yao. *

Cheng and Yao introduced the $(\mathbf{a}, \mathbf{b}, \mathbf{k})$ model, where each queue m is described by a triple (a_m, b_m, k_m). For example, $(B_m, 0, B_m)$ specifies service blocking for server m, while $(B_m, 1, B_m)$ specifies transfer blocking, so the blocking scheme is more general than what we consider here (it also includes *kanban* blocking [SKCU]). The authors proved monotonicity in (I), (II) and (IV) for an open network of $(\mathbf{a}, \mathbf{b}, \mathbf{k})$ queues. They also presented results on throughput variability and convexity, and two modes for initiating arrivals at the first queue.

Finally, Glasserman and Yao used a powerful technique (that views the network as a *generalized semi-Markov process*) to prove several results on the allocation of buffers, including monotonicity in (II) and (IV) for both open and closed networks of $(\mathbf{a}, \mathbf{b}, \mathbf{k})$ queues [GY].

We now return to the other three papers, and consider their technique. Our idea of deriving recursive equations for job departure times is the same as theirs, and independently explored by Adan and Van der Wal in unpublished work [AV]. For example, Cheng and Yao used departure times D_j^m, where $D_j^m = T_{b_m}^{(j)}$ in our notation. Thus, they have

$$D_j^m = \max(D_j^{m-1}, D_{j-1}^m, D_{j-k_{m+1}}^{m+1}) + S_{b_m}^{(j)} \qquad \text{for service blocking}$$

$$\text{and} \qquad D_j^m = \max(\max(D_j^{m-1}, D_{j-1}^m) + S_{b_m}^{(j)}, D_{j-k_{m+1}}^{m-1}) \qquad \text{for transfer blocking,}$$

where k_m is the buffer size. We can derive these from the equations for $T_b^{(j)}$, by iterating as in the proofs of Lemmas 3 and 4 (for the case $b = k - 1$).

Although these papers did not consider infinite buffers for open systems, their results can be extended to permit an arbitrary mix of finite and infinite buffers (as we have done) by setting $D_j^m = 0$ for $j < 0$, so their equations for D_j^m is the same as ours for $T_{b_m}^{(j)}$ where there is no blocking (i.e. $k_m = \infty$). In this way, their equations can be used to prove the monotonicity in (III).

The $T_b^{(j)}$ equations here have an advantage over the D_j^m equations, in that estimators for the queue length distribution and probability of blocking can be directly expressed in terms of $T_b^{(j)}$. This is why we consider an infinite buffer as consisting of $B_m - 1$ finite slots, followed by one infinite slot. One application of the $T_b^{(j)}$ equations is as a simulator for tandem queues — instead of setting up a discrete event simulator with data structures for events, jobs, queues and monitors, we can simply generate one job at a time and evaluate the $T_b^{(j)}$ equations iteratively. In such a simulation, we can estimate a queue length distribution for an infinite buffer up to an arbitrarily

* The results were obtained independently. I was aware of Shanthikumar and Yao's paper, but did not bring a copy of it with me when I arrived at Princeton in May 1991. After proving the theorem in January 1992, I requested copies of related work from David Yao, and received the three papers mentioned here [CY, GY, SY].

small tail by picking an arbitrarily large B_m. (Another possible use of the equations is as the basis for approximations.)

Aside from the open problem concerning the replacement of finite buffers by infinite buffers, Onvural also posed two questions concerning the monotonicity in slot utilization and probability of blocking for a (real) server. None of the papers mentioned here considered these performance measures. We have shown (in Section 2) how $T_b^{(j)}$ can be used to estimate these two performance measures. We expect that, if their monotonicity can be established via the recursive relationship among departure times, then it would be achieved through the $T_b^{(j)}$ equations.

There is another significant difference in our approach. The monotonicity results in the other papers [CY, SY, TW] were stated in terms of the stochastic ordering $\leq_{st}$, where $X \leq_{st} Y$ if and only if $\text{Prob}(X \geq z) \leq \text{Prob}(Y \geq z)$ for all real z [KKO]. For example, Cheng and Yao proved monotonicity in (I) by considering random variables $S_{b_m}^{(j)}$ and $\tilde{S}_{b_m}^{(j)}$ such that $(S_{b_m}^{(j)})_{j=1}^n \leq_{st} (\tilde{S}_{b_m}^{(j)})_{j=1}^n$. In other words, they stochastically compare the performance of Q and $\tilde{Q}_1$ by picking one sample point each from $\mathcal{S}(Q)$ and $\mathcal{S}(\tilde{Q}_1)$. In our approach, however, we decoupled the network and job descriptions, so that Q and $\tilde{Q}_1$ share the same sample space, and a comparison can be made with a single sample point. This makes the proofs entirely elementary, and the technique more accessible.

Acknowledgment

Many thanks to Paul Spirakis, who — long long ago, in a galaxy far far away — encouraged me to work on finite-buffer queues.

References

[A] I.F. Akyildiz, *Mean value analysis for blocking queueing networks*, IEEE Trans. Software Engineering 14 (1988), 418–428.

[AS] T.M. Altiok and S. Stidham, *A note on transfer lines with unreliable machines, random processing ttimes, and finite buffers*, IIE Trans. 14 (1982), 125–127.

[AV] I. Adan and J. Van der Wal, *Monotonicity of the throughput of a closed queueing network in the number of jobs*, Technical Report, Eindhoven University of Technology (1987).

[BI] S. Balsamo and G. Iazeolla, *Some equivalent properties for queueing networks with and without blocking*, In *Performance '83*, Agrawala and Tripathi (eds.), North Holland, Amsterdam (1983), 351–360.

[BS] J.A. Buzacott and J.G. Shanthikumar, *Stochastic Models of Manufacturing Systems*, Prentice Hall, Englewood Cliffs, to appear.

[CP] P. Caseau and G. Pujolle, *Throughput capacity of a sequence of queues with blocking due to finite waiting room*, IEEE Trans. Software Engineering 5 (1979), 631–642.

[CY] D.W. Cheng and D.D. Yao, *Tandem queues with general blocking: a unified model and comparison results*, manuscript (Jan. 1991; revised: Aug. 1991), submitted for publication.

[DY] Y. Dallery and D.D. Yao, *Modeling a system of flexible manufacturing cells*, In *Modeling and Design of Flexible Manufacturing Systems*, Kusiak (ed.), Elsevier North Holland, Amsterdam (1986), 289–300.

[GY] P. Glasserman and D.D. Yao, *Structured buffer-allocation problems in production lines*, manuscript (June 1991), submitted for publication.

[KKO] T. Kamae, U. Krengel and G. O'Brien, *Stochastic inequalities on partially ordered spaces*, Annals of Probability 5 (1977), 899–912.

[KR] A.G. Konheim and M. Reiser, *A queueing model with finite waiting room and blocking*, J. ACM 23 (1976), 328–341.

[O] R.O. Onvural, *Survey of closed queueing networks with blocking*, ACM Computing Surveys 22 (1990), 83–121.

[P] H.G. Perros, *Open queueing networks with blocking*, In *Stochastic Analysis of Computer and Communications Systems*, Takagi (ed.), Elsevier North Holland, New York (1989).

[R] M. Reiser, *A queueing network analysis of computer communications networks with window flow control*, IEEE Transactions on Communications 27 (1979), 1199–1209.

[SD] R. Suri and G.W. Diehl, *A new building block for performance evaluation of queueing networks with finite buffers*, Proc. ACM SIGMETRICS Conference Measurement and Modeling of Computer Systems (1984), 134–142.

[SKCU] Y.K. Sugimori, F. Kusunoki, Cho and S. Uchikawa, *Toyota production system and kanban system: materialization of just-in-time and respect-for-human system*, Int. J. Prod. Res. 15 (1977), 553–564.

[SY] J.G. Shanthikumar and D.D. Yao, *Monotonicity and concavity properties in cyclic queueing networks with finite buffers*, In *Queueing Networks with Blocking*, Perros and Altiok (eds.), North Holland, Amsterdam (1989), 325–344.

[TW] P. Tsoucas and J. Walrand, *Monotonicity of throughput in non-Markovian networks*, J. Applied Probability 16 (1989), 134–141.