Monotonicity of the throughput in single server production and assembly networks with respect to the buffer sizes

Citation for published version (APA):

Document status and date:
Published: 01/01/1987

Document Version:
Publisher’s PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:
• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the “Taverne” license above, please follow below link for the End User Agreement:
www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at:
openaccess@tue.nl
providing details and we will investigate your claim.

Download date: 15. Nov. 2023
Memorandum COSOR 87-35

Monotonicity of the throughput in single server production and assembly networks with respect to the buffer sizes
by
Ivo Adan and Jan van der Wal

Eindhoven, the Netherlands
December 1987
Monotonicity of the throughput in single server production and assembly networks with respect to the buffer sizes

by

Ivo Adan and Jan van der Wal
University of Technology, Eindhoven

ABSTRACT

The throughput in single server production and assembly networks is shown to be monotone in the size of the local buffers. The proof is based on sample path comparison.

1. Introduction

Recently the qualitative evaluation of system performance received a lot of attention. In particular the behavior of the throughput in closed queueing networks is investigated, see e.g. Robertazzi and Lazar [6], Shanthikumar and Yao [7], Suri [8], Van der Wal [9] and Yao [10]. These papers all deal with monotonicity properties of the throughput of the so-called product form networks (see e.g. Baskett, Chandy, Muntz and Palacios [5].) Most of the proofs explicitly use the product form property.

In [1, 2, 3] we studied monotonicity properties of the throughput of more general classes of queueing networks, such as the closed queueing network with general service times, the open queueing network with general interarrival and service times and the open queueing network with finite capacity. The throughput was proved to be monotone in system parameters, as the number of circulating jobs, the speed of the servers, the arrival rate, etc. The proofs were based on coupling and sample path comparison and use no special properties of queueing networks
such as the product form property.

This paper deals with two types of networks having finite local buffers. First the tree-shaped production network is treated and next we consider the assembly network. In each of these two networks cycling and overtaking is impossible, see figure 1.

In the production network jobs arrive at machine 1. After being processed some of the jobs go to machine 2, some to 3. Of the jobs passing through 2 some also need machine 4.

In the assembly network there are five arrival streams of parts in 2, 4, 5, 6 and 7. In 3 an assembly takes place in which one part coming from 5, one from 6 and one from 7 are put together. This assembly requires all three parts at the same time, i.e. it is not possible to first assemble two of the three parts. Etc.

In each node there is a finite buffer for each of the incoming streams. So in the production network there is one buffer in each node but in the assembly network for instance, node 3 has 3 buffers.

In both systems the blocking mechanism is the so-called production protocol. After the production or assembly the machine delivers the product to the next machine unless the buffer in which it has to be placed is full. In that case the
machine stops and waits until a place becomes available. As soon as that happens the product is put in the buffer and the machine can start with a new job or assembly.

We will prove that in these two types of networks the total production increases if the size of one or more buffers is increased. The proof is based on sample path comparison.

The organization of this paper is as follows. The following two sections deal with the production network. In section 2 the model, notations and theorem are given. In section 3 the theorem is proved. In section 4 the assembly network is treated. Section 5 contains conclusions and remarks.

2. Production network; model, notations and theorem

We consider a tree-shaped production network with single server nodes (machines) 1, ..., N and independent generally distributed service times. In each node the service discipline is FCFS. Node 1 is the root of the production network. Each node i has a unique preceding node. The set of successing nodes is denoted by S(i). The network has no cycles. Every node has a finite local buffer. The total number of places in the buffer is denoted by Bi and consists of the waiting places and the service place. The routing of jobs through the network follows a Markov chain (a condition which can be relaxed.) Jobs arrive according to an arbitrary arrival process in the root. A job is rejected if upon arrival the buffer of the root is full. The arrival times, service times and transitions are independent of each other.

We will show that the throughput increases if the buffers are enlarged. For this we compare the throughput of two networks, one with small buffers, the other with large buffers. The smaller system is indicated by S and the larger system with L. Denoting the buffer sizes by B_{i}^{S} and B_{i}^{L} respectively, the assumption is
(1) \[ B_i^L \geq B_i^S, \text{ for all } i = 1, 2, ..., N. \]

For the rest the S and L system are identical. Therefore it suffices to show that for each sample path of arrival times, service times and transitions the throughput up to time \( t \) in system \( L \) is at least equal to the one of in \( S \), for all \( t \).

Let the sequences \( E_j, X_{ij}, \) and \( S_{ij} \) be any given realization of external arrival times, service times and transitions. That is, \( E_j \) is the \( j \)-th arrival time from the outside in the root, \( X_{ij} \) is the service requirement by the \( j \)-th arrived job in node \( i \) and \( S_{ij} \) is the node where the \( j \)-th departing job from node \( i \) will go to. \( S_{ij} = 0 \) if the \( j \)-th departing job leaves the network. Arrival times may coincide (equal \( E_j \)'s) to allow the jobs to arrive in batches. For convenience we assume that \( E_1 > 0 \) and that the network is empty at \( t = 0 \). In the sequel we only consider this realization.

The following notations will be used:

- \( A_{1j} \) The time of the \( j \)-th accepted arrival at the root.
- \( A_{ij} \) The time of the \( j \)-th arrival at node \( i \).
- \( D_{ij} \) The time of the \( j \)-th departure from node \( i \).
- \( A_1(t) \) The total number of accepted arrivals at the root up to and including time \( t \).
- \( A_i(t) \) The total number of arrivals at node \( i \) up to and including time \( t \).
- \( D_i(t) \) The total number of departures from node \( i \) up to and including time \( t \).
\[ D_{ik}(t) \]  The total number of departures from node \( i \) to node \( k \) up to and including time \( t \).

\[ D_{i0}(t) \]  The total number of departures from node \( i \) to the outside up to and including time \( t \).

These variables will have a superscript \( L \) or \( S \) to indicate whether they correspond to the \( L \) or \( S \) system. Note that arrivals in a node are departures from the preceding one. So for all \( i \) and \( k \in S(i) \)

\[(2) \quad D_{ik}(t) = A_k(t).\]

In order to have compare the two systems only at discrete points in time we define the sequence \( t_0, t_1, \ldots \) of time instants upon which an event occurs in at least one of the two systems. Let \( e_1^C < e_2^C < \ldots \) be the time instants in system \( C \) upon which one or more services are completed or jobs arrive from the outside. Then

\[ t_0 := 0 \]
\[ t_n := \min \{ \min \{ e_i^S | e_i^S > t_{n-1} \}, \min \{ e_i^L | e_i^L > t_{n-1} \} \}, \quad n \geq 1. \]

Let us make the following assumption about the arrival and service times.

Assumption

(i) \( E_j \to \infty \quad (j \to \infty). \)

(ii) \( X_{ij} > 0 \) for all \( i, j. \)

The first condition guarantees that in every finite interval \([0, t]\) only finitely many jobs arrive in the root, so only a finite number will be admitted. Each job gives rise to at most \( N+1 \) events (one arrival in the root plus a departure from each node), thus the total number of events up to and including time \( t \) is finite. Hence \( t_n \to \infty \quad (n \to \infty). \)

The second condition guarantees that a job can complete only one service at a
time and hence make at most one transition at a time.

Now we can state our main theorem.

**Theorem (production network)**

For all \( t \geq 0 \), all \( i = 1, \ldots, N \) and all \( k = 0, 1, \ldots, N \)

\[
(3) \quad D_{ik}^L(t) \geq D_{ik}^S(t)
\]

\[
(4) \quad A_t^L(t) \geq A_t^S(t).
\]

Since the functions \( D_{ik}^C \) and \( A_t^C \) are constant on the intervals \([t_m, t_{m+1})\) and \( t_n \to \infty \) \((n \to \infty)\) it suffices to prove the theorem for the time instants \( t_0, t_1, \ldots\)

3. **Proof of the theorem (production network)**

To study the departure times we need more notation. Let \( k \) be a succeeding node of node \( i \), then

\( Y_{ikj} \) The total number of the first \( j \) departing jobs from node \( i \), which go to node \( k \).

\( T_{ikj} \) The time upon which a place becomes available in node \( k \) for the \( j \)-th departing job from node \( i \). This may be earlier than the departure time of the \( j \)-th departing job from node \( i \).

Define \( D_{ij} := 0 \) if \( j \leq 0 \) and \( \delta(i, j) = 1 \) if \( i = j \) and \( 0 \) otherwise, then

\[
Y_{ikj} = \sum_{l=1}^{j} \delta(S_{il}, k)
\]

\[
T_{ikj} = D_{k, Y_{ikj} - 1 - B_{k+1}}, \quad k \neq 0, \quad T_{i0j} = 0
\]

For the time of the \( j \)-th departure from node \( i \) we have, if \( S_{ij} = k \),
(6) \[ D_{ij} = \max( \max( A_{ij} , D_{ij-1} ) + X_{ij} , T_{ik} ) \]

Now we can prove the theorem.
Assume we have shown for all \( t \leq t_m \)

(3) \[ D_{ik}^L(t) \geq D_{ik}^S(t) \] for all \( i, k \)

(4) \[ A_{ij}^L(t) \geq A_{ij}^S(t) \]

Since arrivals are departures from the preceding node this implies

(7) \[ A_{ij}^L \leq A_{ij}^S \] for all \( i, j \) and all \( i \leq A_{ij}^S(t) \)

and clearly (3) gives \( D_{ij}^L(t) \geq D_{ij}^S(t) \) for all \( t \leq t_m \), so

(8) \[ D_{ij}^L \leq D_{ij}^S \] for all \( i, j \) and all \( j \leq D_{ij}^S(t) \)

Note that (3) and (4) trivially hold for \( t = 0 \) as \( D_{ik}(0) = A_{ij}(0) = 0. \)

In order to prove (3) and (4) for all \( t \leq t_{m+1} \) we start to consider the leaves (nodes without successors) and then descend the tree ending in the root.

So let us prove (3) and (4) for \( t = t_{m+1} \).
Let \( i \) be a leaf and \( D_{ij}^S = t_{m+1} \). From (6), and \( X_{ij} > 0 \) follows \( A_{ij}^S \leq t_m \) and \( D_{ij-1}^S \leq t_m \). Hence, with (7) and (8)

\[ A_{ij}^L \leq A_{ij}^S \text{ and } D_{ij-1}^L \leq D_{ij-1}^S. \]

Since \( T_{i0j} = 0 \) this leads to

\[ D_{ij}^L = \max( \max( A_{ij}^L , D_{ij}^L ) + X_{ij} , 0 ) \]
\[ \leq \max( \max( A_{ij}^S , D_{ij}^S ) + X_{ij} , 0 ) = D_{ij}^S. \]

So
Next assume we already established

\[ D^L_{ij}(t_{m+1}) \geq D^S_{ij}(t_{m+1}) \]

Let \( D^S_{ij} = t_{m+1} \) with \( S_{ij} = k, k \neq 0 \) (the case \( k = 0 \) is simpler and can be treated as a leaf.) Since \( X_{ij} > 0 \), a node can complete at most one job at any time. Thus for \( l \neq k \)

\[ D^L_{kl}(t_{m+1}) \geq D^L_{il}(t_m) \geq D^S_{il}(t_m) = D^S_{il}(t_{m+1}) \]

and it remains only to verify the case \( l = k \).

As before (6) and \( X_{ij} > 0 \) imply \( A^S_{ij} \leq t_m, D^S_{ij-1} \leq t_m \) and also

\[ T^S_{ikj} = D^S_{k,Y_{ikj-1}} - B^S_k + 1 \leq t_{m+1}. \]

But then also

\[ D^L_{k,Y_{ikj-1}} - B^S_k + 1 \leq D^S_{k,Y_{ikj-1}} - B^S_k + 1. \]

So with \( B^L_k \geq B^S_k \) we get

\[ T^L_{ikj} = D^L_{k,Y_{ikj-1}} - B^L_k + 1 \leq D^L_{k,Y_{ikj-1}} - B^S_k + 1 \leq D^S_{k,Y_{ikj-1}} - B^S_k + 1 = T^S_{ikj}. \]

Using (6) again we obtain \( D^L_{ij} \leq D^S_{ij} \) whence

\[ D^L_{ik}(t_{m+1}) \geq D^S_{ik}(t_{m+1}). \]

Finally let us prove

\[ A^L_1(t_{m+1}) \geq A^S_1(t_{m+1}). \]

Assume that if a job departs from node 1 just at the moment a job arrives at node
1, it departs just before the arrival and thus it is not counted as being in node 1 upon the arrival instant. So $A_1(t_m) - D_1(t_{m+1})$ is the number of jobs in the root just before $t_{m+1}$. Let $N(t_{m+1})$ be the number of jobs arriving at $t_{m+1}$ in the root (recall the $E_j$ may be equal.)

Then the number of jobs admitted in system $L$ at time $t_{m+1}$ is

$$\min(B_1^L - A_1^L (t_m) + D_1^L (t_{m+1}), N(t_{m+1}))$$

Further

$$\min(B_1^L - A_1^L (t_m) + D_1^L (t_{m+1}), N(t_{m+1}))$$

$$\geq \min(B_1^S - A_1^S (t_m) + D_1^S (t_{m+1}), N(t_{m+1}))$$

$$= \min(B_1^S - A_1^S (t_m) + D_1^S (t_{m+1}) - [A_1^L (t_m) - A_1^S (t_m)], N(t_{m+1}))$$

$$\geq \min(B_1^S - A_1^S (t_m) + D_1^S (t_{m+1}), N(t_{m+1}) - [A_1^L (t_m) - A_1^S (t_m)]$$

Hence

$$A_1^L (t_{m+1}) = A_1^L (t_m) + \min(B_1^L - A_1^L (t_m) + D_1^L (t_{m+1}), N(t_{m+1}))$$

$$\geq A_1^S (t_m) + \min(B_1^S - A_1^S (t_m) + D_1^S (t_{m+1}), N(t_{m+1})) = A_1^S (t_{m+1})$$

so (3) and (4) also hold for $t_{m+1}$. \(\square\)

**Conclusion**

For each realization of arrival times, service times and transitions the throughput up to time $t$ in node $i$ increases if the buffers are enlarged. Clearly this implies monotonicity for notions as the average throughput per unit of time.
4. Assembly network; model, notations and theorem

The tree-shaped assembly network consists of single-server nodes 1, 2, ..., N. A node has at most one successor but may have several preceding nodes, denoted by $P(i)$ for node $i$. There are no cycles. In each node several streams of parts may arrive. These streams are labelled $N+1$, ..., $M$. $E(i)$ denotes the subset of external streams arriving in $i$. The assembly times are independent and generally distributed. The arrival streams are arbitrary but independent of the assembly times and the state of the network.

Each arrival stream, internal or external, has its own buffer. The buffer sizes in node $i$ are denoted by $B_{hi}$. If $h \leq N$ the buffer is for the internal stream of parts from $h$ to $i$, if $h > N$ the buffer is used by an external stream.

Again we compare two systems, one with smaller buffers denoted by $S$, the other having the larger buffers denoted by $L$. We assume

\[
B_{hi}^L \geq B_{hi}^S \text{ for all } h, j.
\]

Let $E_{hj}$ and $X_{ij}$ be a given realization of the external arrival and assembly times, $h = N+1$, ..., $M$, $i = 1$, ..., $N$, $j = 1, 2, ...$. I.e., $E_{hj}$ is the time of the $j$-th arrival for the external stream $h$, $X_{ij}$ the time required for the $j$-th assembly in node $i$. For convenience we assume $E_{h1} > 0$ for all $h = N+1$, ..., $M$ and both $S$ and $L$ to be empty at $t = 0$.

We will use the same notations introduced in the previous sections.

Furthermore define, for $h = 1, ..., N$

\[
A_{hj} \text{ The time of the } j \text{-th arrival of a part at } i \text{ which comes from } h.
\]

\[
A_{hi}(t) \text{ The total number of arrivals at } i \text{ which come from } h \text{ up to and including time } t.
\]

and for $h = N+1$, ..., $M$
\( A_{hij} \) The time of the \( j \)-th accepted arrival at \( i \) of external stream \( h \).

\( A_{hi}(t) \) The total number of accepted arrivals at \( i \) of external stream \( h \) up to and including time \( t \).

Assumption (i) is replaced by

\[ E_{hj} \to \infty \ ( j \to \infty ) \text{ for } h = N+1, \ldots, M. \]

The main result can be stated as

**Theorem (assembly network)**

For all \( t \geq 0 \), all \( i = 1, \ldots, N \) and all \( h = N+1, \ldots, M \)

\[
(10) \quad D_i^L(t) \geq D_i^S(t)
\]

\[
(11) \quad A_{hi}^L(t) \geq A_{hi}^S(t).
\]

**Proof of the theorem (assembly network)**

Let us study the departure times from node \( i \). Let \( k \) be the successor of \( i \) then \( Y_{ikj-1} = j-1 \), so (5) reduces to

\[
(12) \quad T_{ikj} = D_{kj-B}_{ik}
\]

Recall that an assembly requires all parts at the same time, so the \( j \)-th assembly starts at time

\[
t = \max( \max_{heP(i)\cup E(i)} A_{hij}, D_{ij-1} ).
\]

So that relation (6) expands to

\[
(13) \quad D_{ij} = \max( \max_{heP(i)\cup E(i)} A_{hij}, D_{ij-1} ) + X_{ij}, T_{ikj} ).
\]
From here on the proof of the theorem is analogous to the one in the previous section, apart from some minor differences.

5. Conclusion and extensions.

In this paper we considered the single-server tree-shaped production and assembly network with arbitrary arrival streams, Markovian routing and nonzero, independent and generally distributed service times. It was shown that the throughput is monotone in the sizes of the local buffers.

Most of the assumptions can be relaxed, such as independent service times and Markovian routing.

**Independent service times**

The service times may be dependent as long as they can be taken the same in both systems L and S. A natural sort of dependency, for which the coupling still works, is that the service times of a job are nearly the same in each node.

**Markovian routing**

Any routing is allowed. We only need that the transitions in the L and S system can be taken to be the same. For instance alternating routing is possible.

With the same techniques one can show that the throughput is monotone in system parameters as the speed of the servicing and the arrival stream. Furthermore the monotonicity can be shown to hold for the *multi-server* production and assembly network as well [4].

As a final note, it is not necessary to have a tree-shaped network, but it is sufficient that there are no cycles in the network. For instance the monotonicity also holds for a network as depicted in figure 2.
Figure 2

References


