Some Observations on Resource Allocation in Assembly-Like Queueing Networks via Simulation Approach

Yu Song and Masayoshi Hasama

***Abstract*—This paper presents a study on assembly-like queueing models which are used for modeling assembly operations in production lines and other manufacturing systems. Based on computer simulation results, we investigate how to allocate resources such as buffers or higher speed servers to optimize throughputs of the models. We propose several heuristic policies for effective resource allocation. The policies suggest that we should give priority to allocate extra resource to the neighborhood of downstream assembly nodes. These policies are useful in designing production systems and other assembly systems.**

***Index Terms*—Queueing network, production system, allocation problem, simulation, assembly system.**

I. Introduction

Assembly operations arise in many practical situations, including assembly lines in production plants, mixing operations in chemical industries and data flow through computer systems. Assembly-like queueing systems are used for modelling such operations.

Compared to queueing systems with a tandem or merge configuration, assembly-like queueing systems are difficult to analyze. There are only several published studies on queueing network models with such assembly-like configuration, and most of them investigated systems with only a single assembly node. [1]–[4]. Harrison studied a model with infinite buffers, and found that the system cannot reach a stationary state. For models with finite buffers, Bhat studied distribution of average response time [2], Lipperand and Sengupta [3] and Hopp and Simon [4] proposed approximation methods for evaluating throughput or average inventory. For system with multiple assembly nodes, Song, Zhang and Ueda applied a decomposition method to approximate throughput of the system [5].

It is also notable that previous literature addresses evaluation of performance measures, such as stationary probabilities, throughputs, and queue length of those systems [1]–[8].

Our interest here stems from the need to solve resource allocation problems in assembly-like queueing systems. However, there is no effective technique to solve the optimal allocation problems of queueing networks. In [9], Mitra and Mitrani discussed buffer allocation in tandem queueing systems with minimal blocking. They conducted many numerical experiments using an approximation method and presented some heuristic policies for obtaining better throughput values.

The systems discussed in this paper are same as those in [5]. We consider multi-level assembly-like queueing systems (Section II). Buffers between nodes are finite-sized. Based on computer simulation results, we investigate various resource allocations and examine performance of systems in terms of throughput (Section III). Results are presented for: identical nodes with or without buffers; balanced lines, for which variability of processing times differs among nodes; and unbalanced lines. Based on the results, we propose several heuristic policies for effective resource allocation. The policies suggest that it is better to allocate extra resource to the neighborhood of downstream assembly-nodes. It is notable that the policies are quite different from policies for tandem queueing model [9]. Section IV provides concluding remarks.

Manuscript received November 5, 2013; revised February 16, 2014.

Y. Song is with Business School, Manchester Metropolitan University,UK, on leave from Department of System Management, Faculty of Information Engineering, Fukuoka Institute of Technology, Fukuoka, Japan (e-mail: song@fit.ac.jp).

M. Hasama is with the Department of Business Administration at Ube National College of Technology, Japan (e-mail: hasama@ube-k.ac.jp).

II. Model Description

Let's consider tree-structure queueing networks with K Nodes as shown in Fig. 1. There is a single server in each node and the server in Node k provides exponential service with rate μ_k .

We divide the nodes into 3 groups. Group 1 consists of K_1 nodes, Nodes 1, 2, ..., K_1. In front of each server, there is an infinite-size pool of customers waiting for service. Hence these nodes cannot be idle. Upon service completion at a node of Group 1, a customer reaches to one of the remaining $K - K_1$ nodes and receives service in the node. The inputs of nodes of Group 2 are outputs of nodes in either Group l or Group 2. There is only one node (Node K) in Group 3. A customer departs from the system after service completion of the node. Hence, node K is called the final node. A node belonging to neither Group 1 nor Group 3 belongs to Group 2.

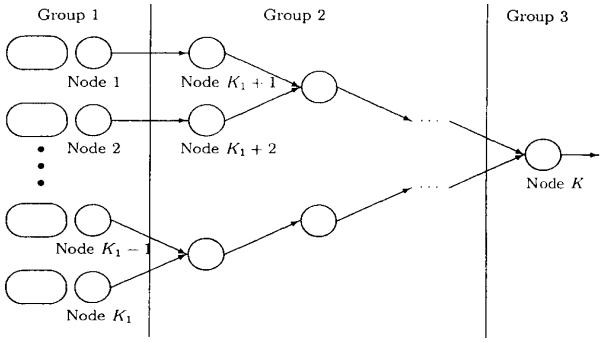

Fig. 1. Queueing network with assembly nodes.

The input of a node in Groups 2 and 3 is a customer who

DOI: 10.7763/IJMMM.2014.V2.117

finished his service at a node in either Group 1 or Group 2. Suppose that the inputs of Node k ($K_1 < k \le K$) are the outputs of i nodes, namely Nodes $u_1(k)$, $u_2(k)$, ⋯, $u_i(k)$, where $u_1(k)$, $u_2(k)$, ⋯, $u_i(k) < k$, indicating that there is no feedback flow (Fig. 2). Buffers $u_1(k)$, $u_2(k)$, ⋯, $u_i(k)$ in front of Node k are prepared to provide the waiting spaces for the outputs of Nodes $u_1(k)$, $u_2(k)$, ⋯, $u_i(k)$ (Fig. 3) . In this paper, we define Buffer k (<K) as the waiting space for customers who has completed its service of Node k (Note that this definition is different from the traditional queue buffer definition which is the waiting space for not-served customers or the input). The server of Node k can begin a bulk service (assembly operation) for the customers only when there is at least one customer in each of all input node buffers in front of Node k. A service completed customer at Node k ($k < K$) enters buffer k and waits there for a service at a downstream node which is denoted by node $d(k)$. Let the size of Buffer k be C_k –1. The server at Node k checks the state of Buffer k before beginning service for a new customer. If Buffer k is full, the customer occupies the server and blocks it. A service is begun only after a vacant space appears in Buffer k. This blocking rule is called communication blocking.

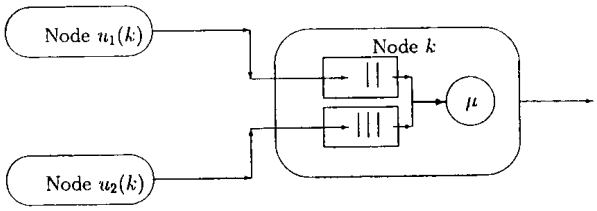

Fig. 2. Assembly-like configuration.

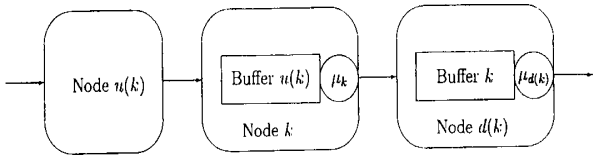

Fig. 3. Adjacent nodes.

The model with communication blocking can be considered as a kind of Kanban production system. That is, there are C_k Kanbans in Buffer k, and a customer must take a Kanban when he enters Buffer k and hold it while waiting in the buffer and receiving his service at a downstream node, Node $d(k)$. Upon completion of service, he puts back the Kanban of Buffer k and goes to Buffer $d(k)$. Therefore C_k –1 becomes the upper limit of the number of customers waiting in Buffer k. If there are no Kanbans of Buffer k available, the server in Node k is blocked and cannot serve the next customer until a Kanban is put back.

Let n_k be the number of customers holding the Kanbans of Buffer k. Since the service time distributions are exponential, we may express the state of the whole system with vector

$$(n_1, n_2, ..., n_{K-1}), n_k = 1,2,..., C_k.$$

and describe the stochastic behavior of the system with a continuous-time Markov chain with finite state space. In steady-state, we denote the throughput of the system by T. By solving a set of stationary equations for the Markov chain, we can obtain the exact value of T. However, it is hard to solve the equations practically for large scale Markov chains. Hence we conduct computer simulations to evaluate T in this research.

III. Investigation on Resource Allocation

This section investigates how to allocate buffers and other resources efficiently in assembly-like queueing systems.

There are various criteria to evaluate system efficiency. The one we adopt here is throughput of a system. It is defined as the average number of customers departing from a system; hence it can be regarded as the productive capacity of the system.

We calculate throughputs of two models by computer simulation. Model 1 is a 10-node example as shown in Fig. 4. In this model, Nodes 7, 8 and 10 are assembly nodes.

Adding another node as the downstream of Node 10, we obtain Model 2. Therefore, the last node, Node 11, is not an assembly node.

We examine the effect of three types of resource allocation: identical nodes with or without buffers; unbalanced buffers with identical service rates, and balanced buffers with different service rates among nodes.

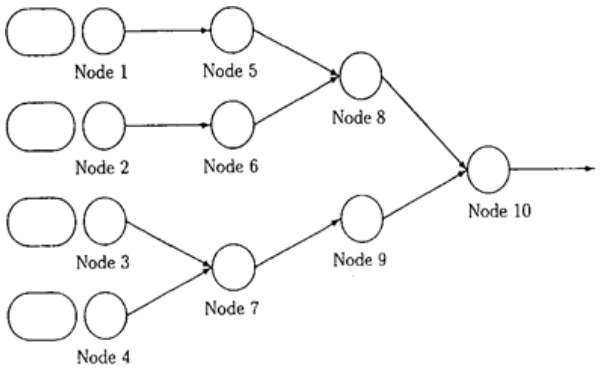

Fig. 4. Model 1.

A. Identical Nodes

First, we investigate identical nodes. In particular, for each node, the service rate μ_k = 4 and the buffer size $C_k = C$. We increase C from 1 through 20. Fig. 5 displays the throughputs of Model 1. This figure shows that the throughput increases constantly with C. As C increases from 1 to 2, the increment of throughput is 0.8. However, the increment becomes smaller and smaller as C increases. When $C > 10$, the increments are less than 0.05. A similar trend can be observed from numerical results for Model 2. This leads to the following observation.

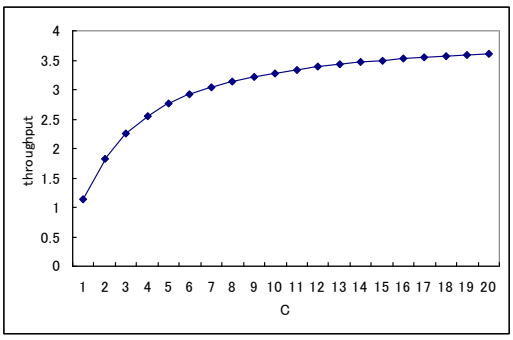

Fig. 5. Modle 1: Identical nodes ($C_1 = C_2 = \bullet\bullet\bullet = C_9 = C$, $\mu_1=\mu_2= \bullet\bullet\bullet =\mu_{10} = 4$).

Observation 1: For systems with identical nodes, throughput increases with buffer size. However, the marginal effect of increased buffer size diminishes gradually.

B. Identical Service Rates

Next, we study unbalanced buffers with identical service rate $\mu_k = 4$.

Table I shows some results of Model 1 where all buffer sizes are 0 ($C_k = 1$) except one buffer size, which has a size 1 buffer ($C_k = 2$). As shown in the table, throughputs are small for cases where C_1, C_2, C_3 or C_4 equals 2. We obtain the largest throughput $T = 1.209$ as $C_8 = 2$ and the second largest throughput $T = 1.195$ as $C_9 = 2$. Considering that C_8 and C_9 are buffer sizes in Node 10, this seems to suggest that if there is an additional buffer space, it is better to allocate it in the last downstream node.

For Model 2, however, we can see from Table II that the largest throughput is not obtained as the last downstream buffer $C_{10} = 2$. The largest throughputs are obtained as $C_8 = 2$ and $C_9 = 2$ again. The difference between Nodes 10 and 11 is that Node 10 is an assembly node while Node 11 is not. The additional buffer space in an assembly node may reduce variability of inventory more effectively.

Observation 2: If there is any additional buffer space, it is more effective to allocate it to a downstream assembly node.

It is noticeable to compare this with the case of tandem queueing systems. In [9], Mitra and Mitrani indicate that for tandem queueing systems, it is better to give additional buffer spaces to middle nodes than to extreme ones. This is of an entirely different feature from the observation for assembly-like queueing systems.

Table III shows results of two additional buffer spaces in Model 1. We can see the following.

- If we allocate two spaces in the same buffer, as stated in Observation 2, it is better to allocate them in the downstream node. However, the largest throughput $T = 1.240$ (in the case of $C_8 = 3$) is just an intermediate one compared to other results in the table.
- The largest throughputs are obtained in cases of $C_7 = C_8 = 2$ and $C_8 = C_9 = 2$.

As shown in Fig. 4, in Node 10, which is an assembly node, Buffers 8 and 9 are on different sides. Also although Buffer 7, which is in Node 9, is not a buffer of Node 10, it strongly affect outputs to Buffer 9. Hence, Buffers 7 and 8 are also on the different buffer sides of Node 10.

TABLE I: *MODEL 1*: ONE ADDITIONAL BUFFER SPACE ($\mu_1=\mu_2=\cdots=\mu_{10}=4$)

Other C_k=1	T	Other C_k=1	T
C_1=*2*	1.148	C_6=2	1.179
C_2=*2*	1.148	C_7=2	1.185
C_3=*2*	1.152	C_8=2	1.209
C_4=*2*	1.151	C_9=2	1.195
C_5=*2*	1.180		

TABLE II: MODEL 2: ONE ADDITIONAL BUFFER SPACE ($\mu_1=\mu_2=\cdots=\mu_{11}=4$)

Other C_k=1	T	Other C_k=1	T
C_1=*2*	1.116	C_6=2	1.147
C_2=*2*	1.119	C_7=2	1.134
C_3=*2*	1.118	C_8=2	1.173
C_4=*2*	1.110	C_9=2	1.154
C_5=*2*	1.149	C_{10}=2	1.125

Observation 3: If there is more than one additional buffer space, it is more effective to allocate them on different buffer sides of a downstream assembly node rather than in one buffer.

TABLE III: MODEL 1: TWO ADDITIONAL BUFFERS ($\mu_1=\mu_2=\cdots=\mu_{10}=4$)

Other C_k=1	T	Other C_k=1	T
C_1=3	1.148	C_6=*3*	1.148
C_2=3	1.151	C_7=*3*	1.155
C_3=3	1.189	C_8=*3*	1.190
C_4=3	1.206	C_9=*3*	1.240
C_5=3	1.223		
C_1=C_2=2	1.170	C_1=C_3=*2*	1.173
C_1=C_4=2	1.171	C_1=C_5=*2*	1.186
C_1=C_6=2	1.198	C_1=C_7=*2*	1.212
C_1=C_8=2	1.228	C_1=C_9=*2*	1.220
C_2=C_3=2	1.176	C_2=C_4=*2*	1.178
C_2=C_5=2	1.207	C_2=C_6=*2*	1.186
C_2=C_7=2	1.218	C_2=C_8=*2*	1.223
C_2=C_9=2	1.223		
C_3=C_4=2	1.172	C_3=C_5=*2*	1.211
C_3=C_6=2	1.211	C_3=C_7=*2*	1.196
C_3=C_8=2	1.236	C_3=C_9=*2*	1.216
C_4=C_5=2	1.213	C_4=C_6=*2*	1.212
C_4=C_7=2	1.197	C_4=C_8=*2*	1.238
C_4=C_9=2	1.217		
C_5=C_6=2	1.231	C_5=C_7=*2*	1.255
C_5=C_8=2	1.252	C_5=C_9=*2*	1.257
C_6=C_7=2	1.257	C_6=C_8=*2*	1.252
C_6=C_9=2	1.257		
C_7=C_8=2	1.268	C_7=C_9=*2*	1.234
C_8=C_9=2	1.261		

C. Identical Buffer Size

Finally, we examine balanced buffers with different service rates among nodes. Suppose that one gets some provisional funding to replace only one machine with a higher speed one, where should it be allocated?

Table IV shows some numerical results for Model 1 where $C_1 = \cdots = C_{10} = 2$ and all μ_k's are set equal to 4 except one server with $\mu_k = 6$. We can see from the table that the largest throughput ($T = 1.914$) is obtained for the case $\mu_8 = 6$; next is the case $\mu_{10} = 6$. For Model 2 (Table V), the largest throughputs are T=1.885 for $\mu_{10} = 6$ and T=1.872 for $\mu_8 = 6$. Since these nodes are downstream assembly nodes, we may infer that, similar to Observation 2, it is effective to allocate higher speed servers to downstream assembly nodes.

Observation 4: If there is a higher speed server, it is more effective to allocate it in a downstream assembly node.

Further numerical experiments are for cases with more than one higher speed servers. Table VI shows result for Model 1 with 2 higher speed servers, the largest throughput is 1.998 for $\mu_7=\mu_8=6$. And large throughputs can also be obtained as $\mu_8=\mu_{10}=6$ and $\mu_7=\mu_{10}=6$. Combining with numerical results for other parameters, we obtain the following observation similar to Observation 3.

Observation 5: If there are more than one higher speed servers, it is more effective to allocate them on the different end sides of a downstream assembly node rather than in one buffer.

Moreover, from Observations 2 – 5, we can conclude that more generally as follows:

Observation 6: The assembly nodes are likely to become the bottleneck of a system. Therefore, to maximize the throughput of a system, we should give priority to the

assembly nodes when allocate resources.

IV. Conclusion

In this paper, we modelled assembly production systems by assembly-like queueing systems and investigate effective resource allocation based on simulation results. We determined the following heuristic policies for resource allocation.

TABLE IV: Model 1: Upgrade 1 Server (C_k=2)

Other μ_k=4	T	Other μ_k=4	T
μ_1=6	1.859	μ_6=6	1.876
μ_2=6	1.849	μ_7=6	1.888
μ_3=6	1.856	μ_8=6	1.914
μ_4=6	1.855	μ_9=6	1.892
μ_5=6	1.874	μ_{10}=6	1.900

TABLE V: Model 2: Upgrade 1 Server (C_K=2)

Other μ_k=4	T	Other μ_k=4	T
μ_1=6	1.812	μ_7=6	1.848
μ_2=6	1.814	μ_8=6	1.872
μ_3=6	1.819	μ_9=6	1.855
μ_4=6	1.817	μ_{10}=6	1.885
μ_5=6	1.842	μ_{11}=6	1.827
μ_6=6	1.840		

TABLE VI: Model 1: Upgrade 2 Servers (C_K=2)

Other μ_k =4	T	Other μ_k =4	T
μ_1=μ_2=6	1.836	μ_1=μ_3=6	1.187
μ_1=μ_4=6	1.870	μ_1=μ_5=6	1.882
μ_1=μ_6=6	1.896	μ_1=μ_7=6	1.908
μ_1=μ_8=6	1.934	μ_1=μ_9=6	1.920
μ_1=μ_{10}=6	1.926		
μ_2=μ_3=6	1.869	μ_2=μ_4=6	1.860
μ_2=μ_5=6	1.894	μ_2=μ_6=6	1.880
μ_2=μ_7=6	1.917	μ_2=μ_8=6	1.932
μ_2=μ_9=6	1.921	μ_2=μ_{10}=6	1.928
μ_3=μ_4=6	1.868	μ_3=μ_5=6	1.903
μ_3= μ_6=6	1.901	μ_3=μ_7=6	1.897
μ_3= μ_8=6	1.955	μ_3=μ_9=6	1.912
μ_3= μ_{10}=6	1.933		
μ_4=μ_5=6	1.902	μ_4=μ_6=6	1.901
μ_4=μ_7=6	1.896	μ_4=μ_8=6	1.953
μ_4=μ_9=6	1.914	μ_4=μ_{10}=6	1.935
μ_5=μ_6=6	1.913	μ_5=μ_7=6	1.945
μ_5=μ_8=6	1.944	μ_5=μ_9=6	1.949
μ_5=μ_{10}=6	1.953		
μ_6=μ_7=6	1.944	μ_6=μ_8=6	1.954
μ_6=μ_9=6	1.958	μ_6=μ_{10}=6	1.953
μ_7=μ_8=6	1.998	μ_7=μ_9=6	1.936
μ_7=μ_{10}=6	1.966		
μ_8=μ_9=6	1.944	**μ_8=μ_{10}=6**	**1.984**
μ_9=μ_{10}=6	1.961		

- For systems with identical nodes, throughput increases with buffer size. However, the marginal effect of increased buffer size diminish gradually.
- It is more effective to allocate extra resources (additional buffer spaces or higher speed servers) to downstream assembly nodes.
- If multiple extra resources are available, it is more effective to allocate them among different end sides of two-end queues.

Note that throughput is only one of criteria to evaluate queueing system performance. In practice, other criteria are used for evaluating system performance. For example, Toyota's Just-In-Time production systems emphasizes decreasing inventory level as much as possible.

Future study should address resource allocation problem with respect with such performance criteria.

References

[1] J. M. Harrison, "Assembly-like queues," *Journal of Applied Probability*, vol. 18, pp. 354, 1981.

[2] U. N. Bhat, "Finite capacity assembly-like queues," *Queueing Systems*, vol. 1, pp. 85–101, 1986.

[3] E. H. Lipperand and B. Sengupta, "Assembly-like queues with finite capacity: bounds, aysmptotics, and approximations," *Queueing Systems*, vol. 1, pp. 67 – 83, 1986.

[4] W. J. Hopp and J. T. Simon, "Bounds and heuristics for assembly-like queues," *Queueing Systems*, vol. 4, pp. 137–156, 1989.

[5] Y. Song, Z. G. Zhang, and F. Ueda, "An approximation analysis for the assembly-like queueing systems," *International Journal of Management Literature*, vol. 1, pp. 49–58, 2001.

[6] B. Prabhakar, N. Bambos, and T. S. Mountford, "The synchronization of poisson processes and queueing networks with service and synchronization nodes," *Advances in Applied Probability*, vol. 32, no. 3, pp. 824–843, 2000.

[7] M. Manitz, "Queueing-Model based analysis of assembly lines with finite buffers and general service times," *Computers & Operations Research*, vol. 35, pp. 2520 – 2536, 2008.

[8] D. Alexander, I. Premachandra, and T. Kimura, "Transient and asymptotic behavior of synchronization processes in assembly-like queues," *Annals of Operations Research*, vol. 181, no. 1, pp. 641-659, 2010.

[9] D. Mitra and I. Mitrani, "Analysis of a novel discipline for cell coordination in production lines," *Management Science*, vol. 36, pp. 1548-1566, 1990.

Yu Song was born in China. He earned his Ph.D. at Tohoku University, Japan in 1992. Currently he is a professor of Fukuoka Institute of Technology, Japan, and is on a sabbatical leave to Business school, Manchester Metropolitan University, UK. He is a member of IACSIT.

Masayoshi Hasama was born in Japan. He received his Ph.D. degree in engineering from Fukuoka Institute of Technology, Japan in 2007. Currently he is a lecturer of the Department of Business Administration at Ube National College of Technology, Japan. His research interests are in areas of supply chain management and operations research. His articles have appeared in journals such as Trans. of the Japan Society for Production Management, Journal of Japan Industrial Management Association and proceedings of major conferences such as APIEMS, ISORA and WSEAS.