RELEASE POLICIES FOR ASSEMBLY SYSTEMS

Izak Duenyas and Matthew F. Keblis
Department of Industrial and Operations Engineering
The University of Michigan
Ann Arbor, MI 48109-2117

Technical Report 93-15

August 1993

RELEASE POLICIES FOR ASSEMBLY SYSTEMS

IZAK DUENYAS and MATTHEW F. KEBLIS

Department of Industrial and Operations Engineering

University of Michigan

Ann Arbor, Michigan 48109

Abstract

We consider a production system consisting of several fabrication lines feeding an assembly station. The machines in the fabrication lines and at assembly are assumed to have general processing time distributions. Releases to the system are governed by the *kanban* release mechanism. We first derive an approximation for the throughput of this system by replacing the assembly system under kanban release by an equivalent system under CONWIP release and by making use of an approximation for the throughput of a CONWIP assembly system (Duenyas, 1992). Comparisons with simulation show that this approximation is robust over a wide range of conditions.

We also address the issue of which release mechanism is more effective in obtaining a desired throughput level with the minimum possible Work-In-Process Inventory level in the system. We present both analytical and simulation results to demonstrate that the CONWIP release policy seems to be a more effective release policy for assembly systems.

1 Introduction

Assembly-like queues are prevalent in many manufacturing systems. Production systems where several sub-assemblies are produced in separate fabrication lines and then assembled together give rise to assembly-like queues. A typical example is in electronics manufacturing where multi-plane circuit boards (PCB's) are manufactured by fabricating the layers separately and then laminating them together.

Despite their prevalance in many manufacturing environments, surprisingly little work has been done on these queues. The majority of commercial queueing network packages do not handle assemblies (see Snowdon and Ammons (1988) for a survey). In fact, most of the queueing models in the literature that do handle assemblies assume that the fabrication lines feeding assembly consist of a single machine, an assumption which limits their applicability (e.g., Ammar(1980), Bhat(1986), Bonomi(1987), Hopp and Simon(1989), and Lipper and Sengupta(1986)). Gershwin (1991) and Mascolo et al. (1991) have focused on assembly systems with acyclic or tree structured networks and their models allow for multiple machines including many assembly/disassembly machines. However, both these papers focus only on systems where all machines have equal processing times. The analysis of more complicated assembly systems has been carried out through the use of simulation (e.g., Baker, Powell and Pyke, 1990, 1993). Although simulation is a powerful tool, it is very time consuming for optimization purposes.

The scarcity of models for assembly systems is even more surprising given the magnitude of research activity that has focused on release control in manufacturing systems in the last decade. Whereas release control in tandem systems is important as a mechanism to control WIP costs and cycle times in the facility, in assembly systems release control also acts as a feedback mechanism to ensure that the system is stable. Assembly systems are unstable unless some feedback mechanism is used to link releases to outputs (Harrison, 1973). Therefore, release control in assembly systems is significant both to control inventory carrying costs, and also to ensure stability of the system. However, the recent literature on release control has almost entirely focused on tandem systems. In many manufacturing systems, the method for controlling releases is MRP, in which releases are scheduled by subtracting fixed lead times from due dates. Pull systems, such as kanban (Monden (1983), Ohno (1988)), have recently become very popular due to the success of Japanese just-in-time methods. A variety of researchers have developed models of tandem systems under pull release mechanisms. Examples include Deleersnyder et al. (1989), Mitra and Mitrani (1990), Duenyas et al. (1991), Spearman (1992), Buzacott et al. (1992), and Tayur (1992a, 1992b). Uzsoy and Martin-Vega (1991) provide a recent survey of kanban-based pull systems, and Uzsoy et al. (1992) provide a comprehensive survey of models of shop-floor control mechanisms in the semiconductor industry. However, despite the great interest in kanban, most approximations for even the simplest tandem kanban lines have rather restrictive assumptions, such as exponential processing times. This is in contrast to tandem production lines under the CONWIP release mechanism, which can be modelled as closed queueing networks. Robust approximations have been developed for closed queueing networks with general processing times (e.g., Shanthikumar and Gocmen (1983), Whitt (1984)).

Several recent papers have focused on assembly systems under "pull" release mechanisms. Duenyas and Hopp (1992a) derive approximations for these systems under the assumption that all processing times are exponential. Duenyas and Hopp (1992b) treat the case where processing times are deterministic, but machines are subject to random failures. Finally, Duenyas (1992) derives an approximation for machines with general processing times. In all of these papers, the CONWIP release mechanism is used to release work into the system.

Under the CONWIP release mechanism, the release of work into the system is controlled by cards as in kanban. The total number of cards in each line is held constant. Each time a job is completed, its card is removed and sent to the front of the line to authorize the start of a new job. Hence, if the assembly operation requires several subassemblies, upon completion of the assembly operation a card would be sent to the first machine in each line, authorizing the start of work on a new unit. Once jobs are released to the first machine in each line, they are then "pushed" through the system.

When a card arrives to the first machine, a new job is not necessarily released to the system immediately. The card only authorizes the operator at the first machine to release a new job when she is done processing the job she is working on. This is due to the fact that releasing a new job before the operator at the first machine is ready to work on it would only lead to the job waiting at a queue in front of machine 1. Therefore, in a CONWIP assembly system, the first machine in each fabrication line would have at most 1 unit of WIP being worked on, although at any time, there may be several cards unattached to jobs at those machines. Figure 1 is a schematic representation of an assembly system with two lines under the CONWIP release mechanism.

An alternative to the CONWIP release mechanism described above is to use the *kanban* release mechanism for assembly systems. In a kanban assembly system, each machine is allocated one or more cards. If there is an unattached card on the bulletin board of the first machine in the fabrication line,

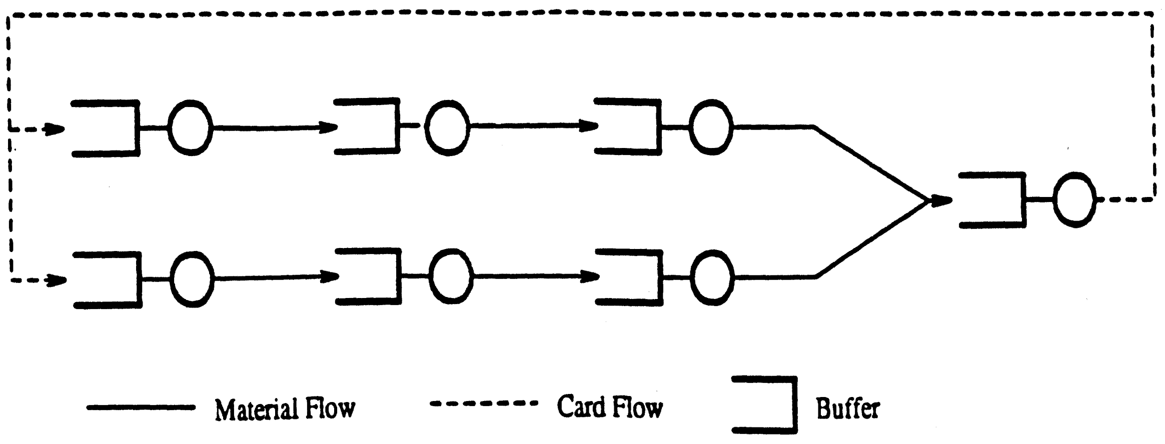

Figure 1: CONWIP Assembly System

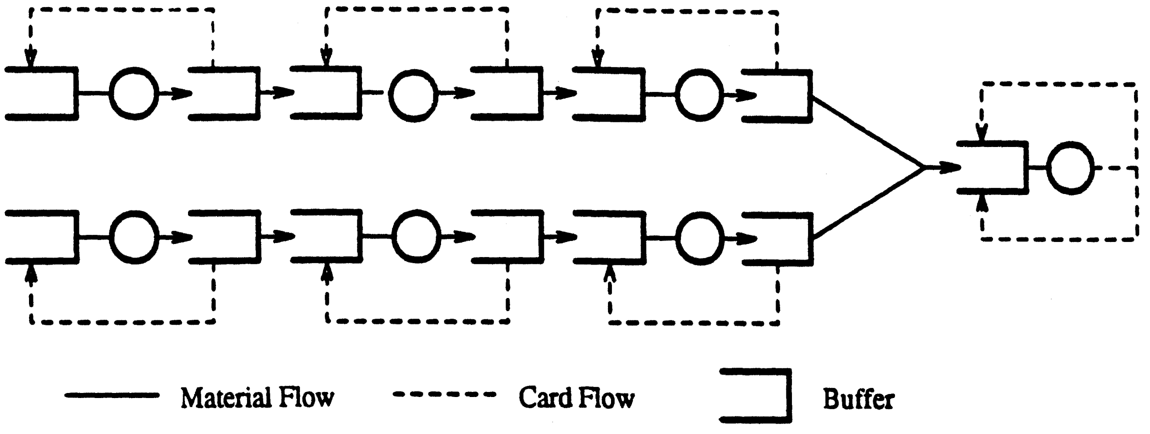

Figure 2: Kanban Assembly System

a new job is released and the card is attached to that job. Once this job's processing is finished on the first machine, the job moves to the input buffer of the next machine if there is a card (unattached to any job) on the bulletin board of the second machine. In this case, the card from machine 1 that was attached to the job is detached and sent back to the bulletin board of the first machine and the card on the bulletin board of the second machine is attached to the job. Otherwise, the job waits in the output hopper of the first machine until a card becomes available at the second machine. The movement of the jobs through the other machines in each fabrication line and to the assembly machine follows the same procedure. When a job from each line is available at the assembly machine, the assembly machine carries out the assembly operation. We note that the assembly machine has separate cards for subassemblies from different fabrication lines. Figure 2 is a schematic representation of the kanban release mechanism for assembly systems.

In this paper, we focus on assembly systems under the kanban release mechanism. Our focus on these systems is motivated by their prevalance in industry and the fact that practitioners can currently only use simulation to predict the performance of these systems due to the lack of any analytical models. Hence, the first contribution of this paper is to provide a simple, robust approximation for the throughput of these systems. To derive an approximation for the throughput of assembly systems under the kanban release mechanism, we make use of a state space approximation first introduced by Akyildiz (1988) to approximate the throughput of closed queueing networks with blocking. The state space approximation replaces the kanban assembly system by an approximately equivalent CONWIP assembly system. We can then use the results in Duenyas (1992) for the throughput of CONWIP assembly systems to obtain an approximation for the throughput of assembly systems under the kanban release mechanism.

The second contribution of this paper is to address the question of which of the two release mechanisms desribed above is more effective for assembly systems. We show that if the objective is to achieve a given throughput level with the minimum possible number of cards (which corresponds to minimizing the maximum possible WIP on the shop floor), then recent results on the superiority of the CONWIP release mechanism for a tandem system (e.g., Spearman, 1992) easily carry over to the assembly case. We also undertake a detailed simulation study to address the question of which release mechanism is more effective in achieving a given throughput level with the minimum possible average WIP in the system.

The remainder of this paper is organized as follows. In the next section, we introduce the notation and problem formulation. Section 3 describes our approximation. In Section 4, we test the performance of our approximation against simulation results. We compare the effectiveness of kanban and CONWIP release mechanisms for assembly systems in Section 5. Section 6 concludes the paper.

2 Problem Formulation

We consider k production lines with m_j machines in line j as shown in Figure 2. In line j, the i^{th} machine, machine (j, i) has $n_{j,i}$ kanban cards allocated to it. The assembly machine has seperate cards for jobs from each fabrication line. Thus, there are n_{j,m_j+1} cards at assembly for jobs from line j. Hence, when a job is completed at the last station in line j, if a card is available at assembly for that type of job, the job is moved to the assembly station. When one job from each fabrication line is available, the assembly operation begins. The movement of the jobs in the fabrication lines is according to the *kanban* discipline as described above. Raw material is always assumed to be available at the first station. We assume that successive processing times at machine (j, i) (at the assembly machine) are independent with finite mean, $x_{j,i}$ (x_A) and variance $\sigma_{j,i}^2$ (σ_A^2).

When the assembly of a job is finished, an output occurs, and the cards attached to the job (note that there are k cards attached to the job at assembly, one for each sub-assembly) are sent back to the bulletin board at assembly. We define N_t as the number of outputs until time t, starting from time 0. We are interested in finding the throughput, $\theta = \lim_{t\to\infty} N_t/t$.

3 An Approximation for Throughput

Our approximation for the throughput of the kanban assembly system is based on a state-space approximation developed by Akyildiz (1988). In Akyildiz (1988), the state space approximation was used to approximate the throughput of a closed queueing network with blocking with the throughput of an approximately equivalent closed queueing network without blocking. In this paper, we use a similar idea to approximate the throughput of tandem and assembly systems under the kanban release mechanism by tandem and assembly systems under CONWIP release. We first describe how our approximation works for a tandem kanban system, then we show how to extend it to an assembly system.

3.1 Tandem Kanban System

In this section, we derive an approximation for a tandem kanban system with M stations. Machine i is assumed to have n_i cards allocated to it. In order to compute an approximation, we need to compute the size of the state space of a tandem kanban system, where the state space consists of all the possible ways in which jobs can be distributed in the system, given the card counts for each station. As previously noted in Tayur (1992a), the distribution of jobs in a tandem kanban system with M stations can be completely characterized by an $M-1$-vector where the j^{th} element of the vector represents the difference between the number of jobs in the output hopper of cell j, and the number of unattached cards in the bulletin board of cell $j+1$. For example, in a 2-station tandem kanban line with n_1 cards in station

1 and n_2 cards in station 2, the state of the system can be described by a single number from the set $\{n_1, n_1 - 1, \ldots, -n_2\}$. Therefore, the state space is of size $n_2 + n_1 + 1$. Note that when the state of the system is n_1, n_1 jobs are in the output hopper of station 1, and there are no unattached cards in station 2 (therefore n_2 jobs are waiting to be processed in station 2), and station 1 is idle. Similarly, the state $-n_2$ corresponds to the case where there are no jobs in station 1's output hopper and n_2 unattached cards on the bulletin board of station 2. The size of the state space of a tandem kanban system can be calculated by the following recursion where Y_1 is the number of states (Tayur, 1992a).

$$\begin{aligned}
X_M &= n_M + 1 \\
Y_M &= 1 \\
X_{m-1} &= (n_{m-1} + 1)X_m + (n_{m-1}(n_{m-1} + 1)/2)Y_m \qquad \text{m=M,...,2} \\
Y_{m-1} &= X_m + n_{m-1}Y_m
\end{aligned}$$

We also note the well known fact that the number of ways N jobs can be distributed in a closed queueing network with M machines is given by $\begin{pmatrix} M + N - 1 \\ M - 1 \end{pmatrix}$. Therefore, since a tandem CONWIP system can be represented by a closed queueing network, this expression also gives the size of the state space for a tandem CONWIP system.

Given the above expressions for the size of the state space in tandem CONWIP and kanban systems, we first note that the number of states in a tandem CONWIP system with 2 machines and N cards is given by $N + 1$. As we previously stated, the size of the state space in a 2-machine kanban system is $n_1 + n_2 + 1$. Therefore, if $n_1 + n_2 = N$, then the size of the state space for kanban and CONWIP tandem systems is the same. It is easy to show that a 2-machine kanban system with $n_1 + n_2 = N$ total cards has the same throughput as a 2-machine CONWIP system with N cards. Therefore, for a 2-machine system, the kanban and CONWIP release mechanisms result in the same size state space, and the same throughput. It is rather interesting to note that Akyildiz(1988) similarly points out that for each 2-station closed queueing network with blocking, there exists a 2-station closed queueing network without blocking and fewer jobs such that the two networks have the same throughput and the same number of states.

For a kanban system with more than 2 machines, a tandem CONWIP system that has the same number of states as a tandem kanban system does not necessarily exist. In fact, as Tayur(1992a) has shown, for the same number of total cards in the system, CONWIP always has a larger state space than kanban. Let the size of the state space of the kanban system, as computed by the recursion above be given by S_{kan}. For the same system with CONWIP release, let N_1 be the largest card count resulting in a tandem CONWIP system with fewer states than S_{kan}. We let $S_C(N_1)$ be the number of states for this CONWIP system (hence, $S_C(N_1) < S_{kan} \leq S_C(N_1 + 1)$) and $\theta_C(N_1)$ represent the throughput of this CONWIP system. Our throughput approximation for the tandem kanban system is then given by a linear interpolation of the throughput of the CONWIP systems with N_1 and $N_1 + 1$ cards. Letting θ_{ap} represent the approximation for the throughput of the kanban system, we get

$$\theta_{ap} = \theta_C(N_1) + \frac{(S_{kan} - S_C(N_1))(\theta_C(N_1 + 1) - \theta_C(N_1))}{S_C(N_1 + 1) - S_C(N_1)}. \tag{1}$$

The remaining issue in our approximation is the throughput of the CONWIP system, $\theta_C(N_1)$. Clearly, if all the processing times are exponential, then the throughput of the tandem CONWIP system can be computed exactly, by using Mean Value Analysis (MVA) (Reiser and Lavenberg, 1980). However, exact results for the throughput of closed queueing networks with general processing times do not exist. Therefore, when processing times are general, we use an approximation due to Shanthikumar and Gocmen (1983) for the throughput of a closed queueing network. This approximation is very simple to code and has been shown to be robust for a wide variety of cases when processing times at each station have coefficients of variation less than 1. Since, in almost all practical situations, processing time distributions encountered in manufacturing satisfy this condition, we make use of this approximation by Shanthikumar and Gocmen. Despite the fact that we are using approximations at two levels, the results in Section 4 show that our approximation is robust. We next describe our approximation for the throughput of an assembly system under the kanban release mechanism.

3.2 Kanban Assembly Systems

Our procedure for estimating the throughput of a kanban assembly system is based on approximating the assembly system under the kanban release mechanism by one under CONWIP release. We can then use the approximation developed in Duenyas (1992) for the throughput of an assembly system under

CONWIP release to derive an estimate of the throughput of the original system. In an assembly system under CONWIP release, as described in Section 2, card counts have to be specified for each fabrication line.

Given an assembly system with k fabrication lines, and m_j stations in line $j, j = 1, \ldots, k$, the first step of our approximation finds the equivalent card counts in each line of a CONWIP assembly system. To do that, we treat each fabrication line along with the assembly machine as a single tandem line. We can then use the procedure described above for a tandem kanban line, and find the equivalent card counts for a CONWIP line. For example, for the tandem line consisting of fabrication line i, along with the assembly machine, we first compute the number of states under kanban release. We let this number be S^i_{kan}. As described above, we let N_i be largest card count resulting in a smaller size state space than S^i_{kan} under CONWIP release. We let $S_C(N_i)$ be the number of states for this tandem line under CONWIP release and N_i cards. We denote the throughput of the CONWIP assembly system with N_i cards in line $i, i = 1, \ldots, k$ as $\theta_C(N_1, \ldots, N_k)$. In this case, since there are two possibilities for card count for each line (e.g., N_1 or $N_1 + 1$ for line 1, N_2 or $N_2 + 1$ for line 2 etc.), we use a very simple interpolation heuristic to obtain our approximation. The first step is to compute the marginal decrease in throughput resulting from decreasing the number of cards in line i by one, which we denote by d_i.

$$d_i = \theta_C(N_1 + 1, N_2 + 1, \ldots, N_i + 1, \ldots, N_k + 1) - \theta_C(N_1 + 1, N_2 + 1, \ldots, N_i, \ldots, N_k + 1) \tag{2}$$

Having computed d_i for each i, we use a simple linear interpolation to compute our approximation for the throughput of the kanban assembly system. The approximation, θ_{ap} is given by

$$\theta_{ap} = \max\{\theta_C(N_1, N_2, \ldots, N_k); \theta_C(N_1 + 1, N_2 + 1, \ldots, N_k + 1) - \sum_{i=1}^{k} d_i \frac{(S_C(N_i + 1) - S^i_{kan})}{S_C(N_i + 1) - S_C(N_i)}\} \tag{3}$$

Notice that (3) involves the maximum of two terms. The maximum in (3) guarantees that the simple linear interpolation we use does not result in any value smaller than $\theta_C(N_1, N_2, \ldots, N_k)$. The remaining issue is that of computing the terms $\theta_C(.)$ in (3). As we have previously noted, we can make use of an approximation recently developed by Duenyas(1992) for the throughput of CONWIP assembly systems for this purpose. We describe how this approximation works for assembly systems with 2 fabrication lines in the Appendix. Readers are referred to Duenyas (1992) for description of the algorithm for assembly systems with more than 2 lines and for further details on this approximation.

4 Computational Results

In this section, we report the results of our simulation study in which we tested the performance of our approximations. We tested our approximation both on tandem and assembly systems. We used a GPSS/H program to simulate the tandem and assembly kanban systems. Each simulation run lasted 21000 time units, and we initialized the system after the first 1000 time units to account for initial bias. The average of 10 runs gave us our simulation estimate for the throughput. Whereas each simulation run took several minutes on a 486 machine, our approximation always gave results in less than a second. Below, we report some representative examples to demonstrate the performance of our approximation.

As an example of the performance of our approximation on tandem lines, we used an example from Mitra and Mitrani (1990). This is a tandem kanban line with 6 machines and each machine has an exponential processing time distribution with rate 4. Figure 3 displays the throughput of the system when the number of cards that each machine is allocated is varied from 1 to 6. Figure 3 displays the simulation value for the throughput, as well as the value obtained by the Mitra and Mitrani approximation and our approximation. In this case, both approximations behaved very well, although the approximation by Mitra and Mitrani (1990) was slightly better. The maximum error in the Mitra and Mitrani approximation was 2.7 %, while the maximum error in our approximation was 3.1 %. However, we must note that our approximation produced results much more rapidly than the Mitra and Mitrani approximation which became considerably slower as the number of cards in the system was increased. Furthermore, as we previously noted, a significant advantage of our approximation is that for tandem systems, it can easily be used along with the Shanthikumar and Gocmen approximation for closed queueing networks to obtain approximations when processing times are not exponential. In Figure 4, we display the results for the same network considered above, but with all the machines having Erlang-2 processing times with rate 4. Again, our approximation behaved very well. For each case it took a 486 computer less than a second to compute the approximation. These examples are representative of our experience with tandem systems.

We display eleven cases as examples of our experience with the approximation in the case of assembly

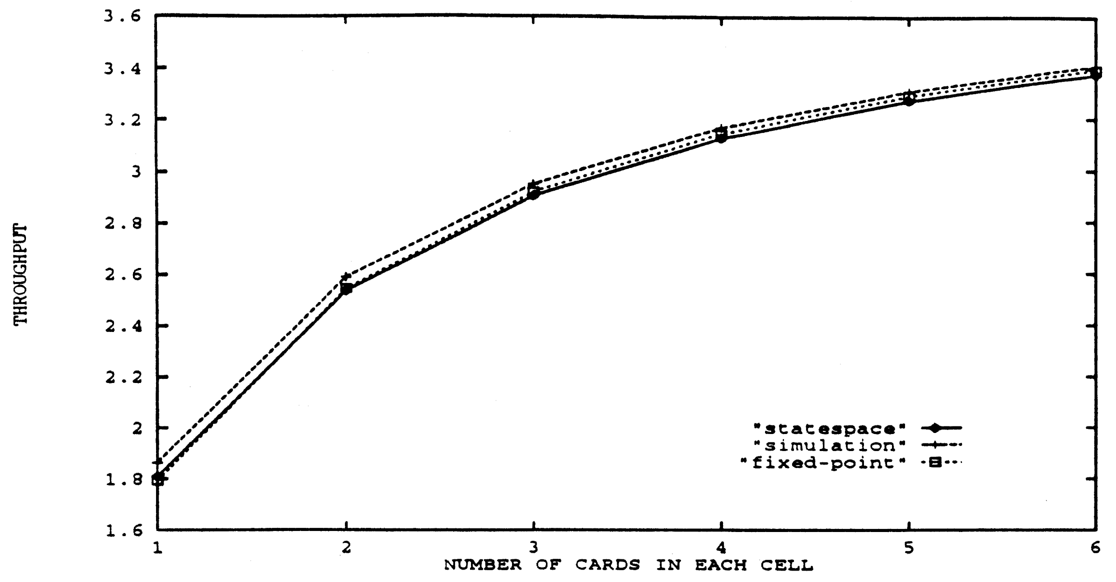

Figure 3: Example from Mitra and Mitrani (1990)

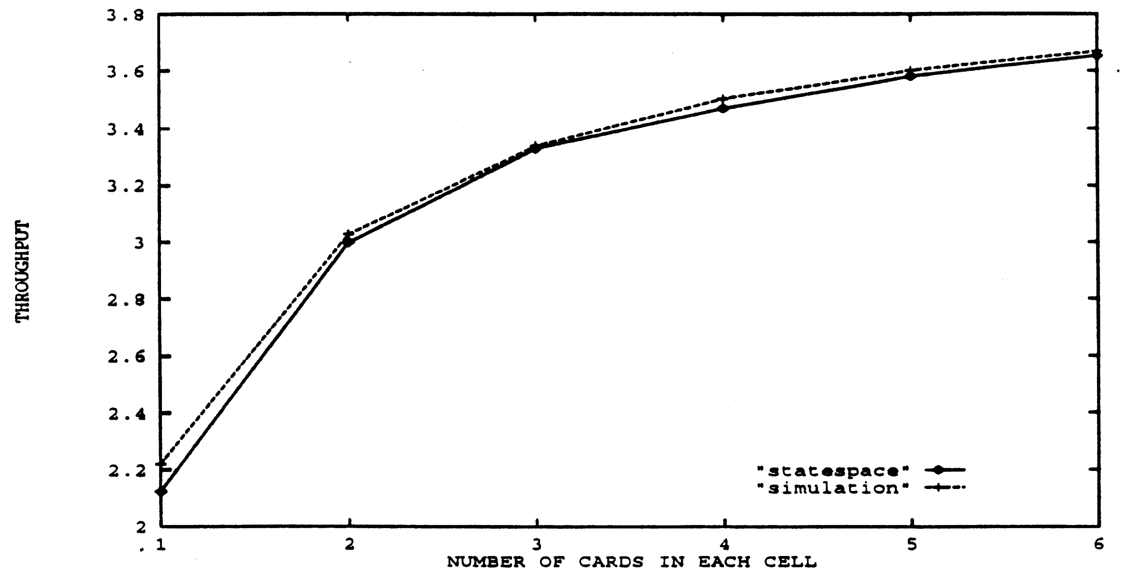

Figure 4: Example from Mitra and Mitrani (1990)

systems. In eight of these cases, there are 2 fabrication lines, and each fabrication line has 3 machines. These examples include cases where the assembly machine is the bottleneck, the bottleneck is in fabrication, as well as balanced cases. In Examples 1-4, all machines have exponential processing times. In Example 1, all the machines have mean processing times equal to 1. In Example 2, all the machines in the two fabrication lines have mean processing times equal to 1, however the assembly machine is the bottleneck with mean processing time equal to 2. Example 3 is the same as Example 2, but in this case the assembly machine's mean processing time is 0.4, and therefore the assembly machine is faster than all other machines. Whereas Examples 1-3 have balanced fabrication lines, in Example 4, the two lines are not balanced. In this example, all machines in fabrication line 2 as well as the assembly machine have mean processing times equal to 1. However, all machines in fabrication line 1 have mean processing times equal to 3. In Examples 5-8, all processing times have Erlang-2 processing times. Apart from this difference in processing time distribution, the networks in Examples 5,6,7 and 8 are exactly the same as in, respectively, Examples 1,2,3, and 4. Therefore, these examples test the effect of a different coefficient of variation on the approximation.

In Examples 9 and 10, there are 2 fabrication lines, but each line has 4 machines. Furthermore, whereas Examples 1-8 had lines where all the machines in each line had the same mean processing time, in Examples 9 and 10 the machines in each line have mean processing times that are not all identical. In Example 9 the machines in line 1 have Erlang-2 processing times while in line 2 the machines have Erlang-4 processing times. In line 1, the first two machines have mean processing times of 1.5, the third machine has a mean of 2.0 and the fourth machine has a mean of 2.5. The first three machines in the second line have mean processing times of 1.0 and the fourth machine has a mean of 1.5. The processing time of the assembly machine has an exponential distribution with mean 3.0. In Example 10 the 4 machines in line 1 have Erlang-3 processing times, where the first machine has mean 1.5, the second machine has mean 2.5, the third machine has mean 1.0 and the fourth machine has mean 2.0. The machines in the other line and assembly have Erlang-4 processing times with means 1.2, 2.2, 2.0, 2.0 and 2.0 respectively. An important difference between Examples 9 and 10 is that the bottleneck machine has been moved from assembly to fabrication. Example 11 has three fabrication lines and four machines

Kanban Allocation	θ_s	θ_{ap}	$\%err$
1,2,1,1;1,2,1,1	0.517	0.513	-0.8
1,2,2,1;1,2,2,1	0.584	0.570	-2.4
1,3,2,1;1,3,2,1	0.604	0.610	1.0
2,2,2,2;2,2,2,2	0.641	0.620	-3.3
1,3,3,1;1,3,3,1	0.659	0.646	-2.0
1,8,7,1;1,7,7,1	0.803	0.799	-0.5
1,17,18,1;1,17,18,1	0.902	0.902	0.0

Table 1: Results for Example 1

Kanban Allocation	θ_s	θ_{ap}	$\%err$
1,1,1,1;1,1,1,1	0.363	0.357	-1.7
1,1,2,1;1,1,2,1	0.413	0.405	-1.9
1,3,2,1;1,3,2,1	0.442	0.456	3.0
2,2,2,2;2,2,2,2	0.464	0.460	-0.9
2,3,3,2;2,3,3,2	0.483	0.484	0.2
1,4,4,1;1,4,4,1	0.484	0.487	0.6

Table 2: Results for Example 2

in each line. All machines except the assembly machine have Erlang-2 distributions with mean 1.0, while the assembly machine has an Erlang-2 distribution with mean 2.0.

The results for Examples 1-11 are in Tables 1-11. In each table, the first column gives the allocation of cards to machines. For example, the first allocation in Table 1 (1,2,1,1;1,2,1,1) means that the three machines in each fabrication line were allocated, respectively 1,2 and 1 cards, while the assembly machine was allocated 1 card each for jobs from each line. In these tables, θ_s denotes the throughput obtained by simulation, while θ_{ap} denotes the throughput obtained by our approximation. Finally, we also give the percentage difference between the simulation results and the approximation results.

As it can be seen from Tables 1-11, our approximation behaved very well for all of the test problems considered. The maximum error, in all cases considered, was less than 4 %, and the average error was much lower than that. The approximation behaved very well for different values of coefficient of variation of processing time and also for different network configurations. These examples are representative of our general experience with our approximation.

Kanban Allocation	θ_s	θ_{ap}	$\%err$
1,3,2,1;1,3,2,1	0.675	0.660	-2.2
1,3,3,1;1,3,3,1	0.706	0.699	-1.0
3,3,4,3;1,6,5,1	0.791	0.804	1.6
1,6,5,1;1,6,5,1	0.805	0.810	0.6
1,13,14,1;1,13,13,1	0.902	0.917	1.7

Table 3: Results for Example 3

Kanban Allocation	θ_s	θ_{ap}	$\%err$
1,1,1,1;1,1,1,1	0.187	0.192	2.7
1,2,1,1;1,1,1,1	0.215	0.215	0.0
1,2,2,1;1,1,1,1	0.229	0.232	1.3
1,3,3,1;1,1,1,1	0.251	0.254	1.2
1,4,3,1;1,1,1,1	0.261	0.261	0.0
1,11,11,1;1,1,1,1	0.302	0.304	0.7

Table 4: Results for Example 4

Kanban Allocation	θ_s	θ_{ap}	$\%err$
1,2,1,1;1,2,1,1	0.614	0.611	-0.5
1,2,2,1;1,2,2,1	0.693	0.702	1.3
2,3,3,2;2,3,3,2;	0.806	0.792	-1.7
1,4,4,1;1,4,4,1	0.812	0.803	-1.1
1,9,9,1;1,8,9,1	0.901	0.899	-0.2

Table 5: Results for Example 5

Kanban Allocation	θ_s	θ_{ap}	$\%err$
1,1,1,1;1,1,1,1	0.417	0.410	-1.7
2,1,1,1;2,1,1,1	0.422	0.430	1.9
1,2,2,1;1,2,1,1	0.451	0.466	3.3
1,3,2,1;1,2,2,1	0.480	0.485	1.0
2,2,2,2;2,2,2,2	0.494	0.493	-0.2
2,3,3,2;2,3,3,2	0.499	0.499	0.0

Table 6: Results for Example 6

Kanban Allocation	θ_s	θ_{ap}	$\%err$
1,1,2,1;1,1,2,1	0.656	0.656	0.0
1,2,2,1;1,2,1,1	0.708	0.684	-3.3
1,2,2,1;2,2,1,1	0.727	0.706	-2.9
1,3,3,1;1,3,3,1	0.806	0.804	-0.2
1,7,7,1;1,6,7,1	0.900	0.911	1.2

Table 7: Results for Example 7

Kanban Allocation	θ_s	θ_{ap}	$\%err$
1,2,1,1;1,1,1,1	0.244	0.249	2.0
2,2,2,2;2,2,2,2	0.270	0.280	3.7
1,3,2,1;1,1,1,1	0.272	0.276	1.5
3,3,3,3;1,1,1,1	0.291	0.298	2.4
1,6,6,1;1,1,1,1	0.302	0.306	1.3

Table 8: Results for Example 8

Kanban Allocation	θ_s	θ_{ap}	$\%err$
1,1,1,2,1;1,1,1,1,1	0.262	0.254	-3.1
1,2,1,2,1;1,1,1,1,1	0.265	0.266	0.4
1,2,1,2,1;1,1,1,2,1	0.271	0.275	1.5
1,2,1,3,1;1,1,1,2,1	0.289	0.287	-0.7
1,2,2,3,1;1,1,1,2,1	0.295	0.295	0.0
1,2,2,3,1;1,1,2,2,1	0.295	0.300	1.7

Table 9: Results for Example 9

Kanban Allocation	θ_s	θ_{ap}	$\%err$
1,2,1,1,1;1,2,1,1,1	0.331	0.320	-3.3
1,2,1,2,1;1,2,1,1,1	0.344	0.335	-2.6
1,2,1,2,1;1,2,2,1,1	0.359	0.354	-1.4
1,2,1,2,1;1,2,2,2,1	0.373	0.360	-3.5
1,3,1,2,1;1,2,2,2,1	0.380	0.371	-2.4
1,3,2,2,1;1,2,2,2,1	0.384	0.380	-1.0

Table 10: Results for Example 10

Kanban Allocation	θ_s	θ_{ap}	$\%err$
1,1,2,1,1;1,1,2,1,1;1,1,2,1,1	0.421	0.431	2.4
1,2,2,2,1;1,2,2,2,1;1,2,2,2,1	0.475	0.480	1.1
2,1,1,1,1;2,1,1,1,1;2,1,1,1,1	0.404	0.418	3.5
1,1,1,1,1;1,1,1,1,1;1,1,1,1,1	0.401	0.386	-3.9
2,2,2,2,2;2,2,2,2,2;2,2,2,2,2	0.493	0.493	0.0

Table 11: Results for Example 11

The computational results displayed above indicate that our approximation is very useful for initial analysis. This is due to the fact that our approximation is very fast (in all of the examples considered above, we obtained results in less than a second on a 486 computer) and pretty robust. Hence, it can be used in the process of design to rule out many cases. For example, an important consideration in the design of many kanban systems is to ensure that a given production level will be met with the minimum possible inventory. This problem has two components. The first is to ensure that there are "enough" kanban cards in the system. The second is to ensure that these cards are allocated optimally. Our approximation can be used for these problems in order to rule out many possibilities in a very rapid manner. Simulation can then be used on a limited number of cases to choose the best possible design. For example, if a facility would like to achieve a throughput level θ, then our approximation can be used initially to eliminate many different possible designs. Then, a conservative approach would be to use simulation on all the possible designs for which our approximation estimated a throughput level of between 0.97θ and 1.03θ, given that our approximation's worst case errors seem to be between 3 and 4 percent.

An interesting problem that has been addressed before for tandem kanban lines (e.g., Mitra and Mitrani(1990), and Tayur (1992b)) is the question of how to allocate a fixed number of cards to maximize throughput. Tayur(1992b) discovered that for balanced or nearly balanced systems, the heuristic that allocates cards in order to achieve the largest state space works very well. We note that our approximation also predicts that throughput is highest when cards are allocated in this manner *within* each line in an assembly system. However, this allocation is not necessarily optimal when the system is unbalanced. Yet, our approximation still behaved very well because often the difference in throughput obtained by the *best* allocation and the state space maximizing allocation of cards *within* each line is very small (e.g., less than 2-3 %). We note, however, that the best allocation of cards to each line and to each machine is an open and difficult problem for kanban assembly systems. Given this much greater difficulty of allocation of cards in a kanban system as compared to a CONWIP system, a question of practical importance is whether all the extra work involved in designing a kanban assembly system is worth it. This question is especially significant given kanban's widespread use. We address this question in the next section.

5 Comparison of Release Mechanisms

In Section 3, we developed an approximation to estimate the throughput of an assembly system under kanban release. In previous work (e.g., Duenyas and Hopp, 1992a and Duenyas 1992), we had also derived an approximation for assembly systems under CONWIP release. In this section, we compare the performance of these two systems.

This question has been previously addressed in several papers (e.g., Spearman, 1992, Spearman and Zazanis (1992), Tayur (1992a)). One result that has previously been shown in these papers is that given a fixed number of cards, CONWIP achives a higher throughput than kanban. This result immediately generalizes to assembly systems in the following

Proposition 1 *Assume that we are given an assembly system with k lines and m_j machines in each fabrication line, and a total of N cards to control releases. Let θ_C be the throughput of the system when CONWIP is used to release jobs, and the N cards are allocated to the fabrication lines optimally. Let θ_K be the throughput of the system when Kanban is used to control releases, and the N cards are allocated to each line and each machine optimally. Then*

$$\theta_C \geq \theta_K.$$

Proof: The proof is similiar to the proof of this result for the tandem case in Spearman (1992) and Spearman and Zazanis (1992) and is omitted.

As we previously noted, the number of cards in a kanban or CONWIP system corresponds to the maximum possible inventory on the shop floor since at any point in time, there may be cards that do not have any inventory attached to them in the system. As we noted in Section 1, in a CONWIP system, there may be cards unattached to any jobs in front of machine 1 in each line. Hence, the above proposition only compares the performance of the two release mechanisms with respect to the objective of maximizing throughput subject to a given level of maximum WIP on the shop floor. An alternative objective is to meet a given throughput level with the minimum *average* WIP on the shop floor. Unfortunately, analytical results on the superiority of one release mechanism over another under this objective do not exist even for tandem systems. Spearman (1992), and Muckstadt and Tayur (1991) have compared the performance of CONWIP and kanban for serial systems using simulation, however they have reached conflicting conclusions. Spearman (1992) considered a serial make-to-order system in which customer demand occurs at random time intervals. He focused on the mean and variance of the time until the customer's order is filled. In the limited number of examples in his paper, for the same average WIP in the system, CONWIP seems to outperform kanban in customer service where customer service is measured by the mean wait time a customer experiences until her order is filled. However, Spearman (1992) notes that in his study, card counts for kanban were set according to a procedure advocated in several practitoner texts on kanban (e.g., Hall (1983)). Since these procedures do not necessarily allocate cards to machines *optimally*, the results of this study do not necessarily show that CONWIP would outperform kanban when cards are allocated to machines *optimally* in kanban.

The other study where such a comparison is made is Muckstadt and Tayur (1991). In this case, the authors address the question of which release mechanism achieves a target throughput level in a serial system with less average WIP. Muckstadt and Tayur compute the best allocation of kanbans by simulation. In their experiments, CONWIP is always outperformed by kanban by significant margins. However, their count of average WIP in CONWIP includes the inventory in front of machine 1. As we have noted in Section 1, there is really no point in releasing a new unit to the first machine of a line in a CONWIP system until the machine is ready to process it. Furthermore given the assumption that raw material is always available, *even if* raw material is released to the first machines in each line as soon as a card arrives to the first machine, this would only mean that raw material that was already bought is moved from one location in the plant (e.g., raw materials storage) to another (to the first machine).

No value is added to the material until the first machine begins processing it. Hence, when accounting for WIP in a CONWIP or kanban system, we should only take into account those jobs to which value has already been added. (Muckstadt and Tayur claim that even if WIP is accounted for in this manner, kanban still outperforms CONWIP in their examples, although by a smaller margin.) Clearly, by the same reasoning, in a kanban system there should be only one unit of WIP on the first machine as well. In fact, Tayur (1992a) has shown that the first and last machine in a serial kanban line is always allocated only 1 card. This result also carries over to assembly systems.

Proposition 2 *The optimal solution to the problem of allocating a fixed number of kanbans to machines in a kanban assembly system will always have only 1 card allocated to the first machine on each fabrication line. Furthermore, the assembly machine will also have only 1 card for sub-assemblies from each line.*

Proof: The proof is similiar to that in Tayur (1992a) for serial systems and is omitted.

In comparing the performance of kanban and CONWIP release mechanisms in the context of assembly systems, we need to take the difficulty of finding the optimal allocations of cards into account. Whereas in CONWIP, the only problem is the allocation of cards to lines, in kanban there is also the problem of allocating cards in each line to the particular machines. Clearly, the allocation problem is a much more difficult combinatorial problem in kanban. However, this additional difficulty might be worth it if kanban significantly decreased average WIP required to achieve a throughput level compared to CONWIP. To discover whether this is true, we undertook a simulation study. For example networks 1,2,4,5,6 and 8 in the previous section, we determined the kanban and CONWIP allocations that meet a given throughput level with the minimum possible *average* WIP in the system. We also did the same for a new example (Example 12) with two fabrication lines where the three machines in line 1 have mean processing times of 1, 3 and 2, each of the machines in line 2 has a mean processing time of 1, and the assembly has mean 0.4. Each of the machines in Example 12 has an exponential processing time. We note once again that these examples include cases with balanced and unbalanced lines, as well as the bottleneck in different locations. Hence, they cover a wide range of situations.

We present our results in Tables 12-18 in which information is presented in the following manner: θ_{tar} = the throughput value targeted, Kanban Allocation is the best allocation of cards that achieves

θ_{tar}	Kanban Allocation	θ_{kan}	WIP_{kan}	CONWIP	θ_{con}	WIP_{con}
0.44	1,1,1,1;1,1,1,1	0.462	6.695	3,3	0.442	5.638
0.50	1,1,2,1;1,1,2,1	0.528	7.937	4,4	0.516	7.359
0.55	1,2,2,1;1,1,2,1	0.550	9.048	5,5	0.571	9.041
0.60	1,2,3,1;1,2,2,1	0.603	10.758	6,6	0.616	10.701
0.65	1,3,3,1;1,3,3,1	0.659	13.364	7,7	0.652	12.350
0.80	1,7,8,1;1,7,7,1	0.805	26.456	15,15	0.804	25.128
0.90	1,17,18,1;1,17,18,1	0.902	57.714	34,34	0.902	55.101

Table 12: WIP Comparison for Example 1

θ_{tar}	Kanban Allocation	θ_{kan}	WIP_{kan}	CONWIP	θ_{con}	WIP_{con}
0.40	1,1,2,1;1,1,2,1	0.413	8.983	4,4	0.400	7.623
0.42	1,1,2,1;1,1,3,1	0.425	9.851	5,5	0.433	9.488
0.44	1,1,3,1;1,1,3,1	0.442	10.608	6,5	0.444	10.436
0.46	1,1,4,1,;1,1,4,1	0.461	12.249	6,7	0.462	12.319
0.48	1,2,5,1;1,2,5,1	0.485	16.610	8,8	0.482	15.178

Table 13: WIP Comparison for Example 2

the target throughput, θ_{kan} = the throughput of the kanban allocation, WIP_{kan} = the average work in process in the kanban system, CONWIP is the allocation of cards that achieves the throughput target using CONWIP, θ_{con} = the throughput of the CONWIP allocation, WIP_{con} = the average work in process in the CONWIP system.

The results in Tables 12-18 clearly show that the kanban release mechanism does not seem to perform better with respect to average WIP in the system. Although there were a couple of cases where kanban outperformed CONWIP by a very small margin, in many other cases the CONWIP release mechanism

θ_{tar}	Kanban Allocation	θ_{kan}	WIP_{kan}	CONWIP	θ_{con}	WIP_{con}
0.17	1,1,1,1;1,1,1,1	0.187	6.503	3,1	0.177	3.786
0.20	1,1,2,1;1,1,1,1	0.204	6.819	5,1	0.210	5.525
0.22	1,2,2,1;1,1,1,1	0.229	7.717	6,1	0.221	6.409
0.24	1,3,3,1;1,1,1,1	0.251	8.892	6,2	0.243	6.923
0.26	1,4,3,1;1,1,1,1	0.261	9.682	8,2	0.261	8.413
0.30	1,11,11,1;1,1,1,1	0.302	18.049	20,2	0.301	17.017

Table 14: WIP Comparison for Example 4

θ_{tar}	Kanban Allocation	θ_{kan}	WIP_{kan}	CONWIP	θ_{con}	WIP_{con}
0.60	1,1,2,1;1,1,2,1	0.627	8.131	4,4	0.604	7.467
0.65	1,1,2,1;1,2,2,1	0.650	9.284	5,5	0.669	9.157
0.70	1,2,3,1;1,2,2,1	0.711	11.028	6,6	0.717	10.804
0.75	1,3,3,1;1,3,3,1	0.766	13.585	7,7	0.755	12.431
0.80	1,4,4,1;1,3,4,1	0.801	15.762	9,9	0.807	15.635
0.90	1,9,9,1;1,8,9,1	0.901	31.295	18,18	0.900	29.979

Table 15: WIP Comparison for Example 5

θ_{tar}	Kanban Allocation	θ_{kan}	WIP_{kan}	CONWIP	θ_{con}	WIP_{con}
0.40	1,1,1,1;1,1,1,1	0.417	7.574	3,4	0.415	6.831
0.42	1,1,2,1;1,1,1,1	0.436	8.531	4,4	0.450	7.770
0.44	1,1,2,1;1,1,2,1	0.466	9.395	4,4	0.450	7.770
0.46	1,1,2,1;1,1,2,1	0.466	9.395	4,5	0.461	8.742
0.48	1,1,3,1;1,1,3,1	0.486	11.219	6,5	0.483	10.676

Table 16: WIP Comparison for Example 6

θ_{tar}	Kanban Allocation	θ_{kan}	WIP_{kan}	CONWIP	θ_{con}	WIP_{con}
0.20	1,1,1,1;1,1,1,1	0.212	6.663	4,1	0.220	4.747
0.22	1,1,2,1;1,1,1,1	0.230	6.972	4,1	0.220	4.747
0.24	1,2,1,1;1,1,1,1	0.244	7.520	4,2	0.241	5.493
0.26	1,2,3,1;1,1,1,1	0.264	8.188	6,2	0.271	6.966
0.30	1,6,6,1;1,1,1,1	0.301	12.402	12,2	0.302	11.171

Table 17: WIP Comparison for Example 8

θ_{tar}	Kanban Allocation	θ_{kan}	WIP_{kan}	CONWIP	θ_{con}	WIP_{con}
0.255	1,1,1,1;1,1,1,1	0.256	6.444	3,2	0.266	4.899
0.280	1,1,2,1;1,1,1,1	0.281	6.730	4,2	0.288	5.876
0.300	1,2,2,1;1,1,1,1	0.302	7.797	5,2	0.303	6.858
0.310	1,2,3,1;1,1,1,1	0.311	8.057	6,2	0.312	7.849
0.320	1,3,3,1;1,1,1,1	0.321	9.126	7,2	0.320	8.840

Table 18: WIP Comparison for Example 12

outperformed kanban by a large margin. In fact, the average WIP in one example was as much as 30 % higher when kanban was used than when CONWIP was used. These results seem to indicate that for assembly systems, the CONWIP release mechanism may be a more appropriate release mechanism than the kanban release mechanism since it is simpler to implement and at least in our experiments seems to perform at least as well as kanban. However, further research needs to be carried out to find out if there are cases where kanban might be more appropriate.

6 Conclusions and Further Research

In this paper, we derived an approximation for the throughput of kanban assembly lines. The approximation is based on a state-space approximation originally developed by Akyildiz. We tested our approximation on a variety of examples and found that the approximation works very well. Some advantages of this approximation is that unlike most other work on kanban systems, it does not require exponential processing times and can handle general processing times, the robustness of the approximation, and the fact that it can easily be used for design of kanban assembly lines since the approximation is not computationally expensive. Since the kanban release mechanism is used widely in industry, our approximation should aid manufacturers in analyzing and designing their production lines.

We also addressed the question of whether the kanban or CONWIP release mechanism is more appropriate for assembly systems. We showed that the well known result for tandem systems that CONWIP reaches a target throughput level with less *maximum* WIP in the system easily carries over to the assembly case. We also conducted a simulation experiment to find out which release mechanism is more successful in obtaining a throughput level with less *average* WIP in the system. For the experiments that we conducted, CONWIP performed better than kanban with respect to this objective as well. However, further research is necessary to find out if there are certain conditions that make kanban a more appropriate release mechanism.

Further research should also characterize the variance of the output process from kanban and CONWIP assembly systems. In a recent and very comprehensive survey of the research on manufacturing flow line systems, Dallery and Gershwin (1992) write that this issue has been entirely neglected by researchers even though the standard deviation of weekly production in many factories can be over 10 % of the mean. Recently, some approximations for the variance of the number produced in a given period in a tandem CONWIP line were derived in Duenyas and Hopp (1990) and Duenyas et al. (1991). However, further research is necessary to develop approximations and structural results for more general systems.

Appendix

In this appendix, we describe the approximation for throughput developed in Duenyas (1992) for assembly systems under CONWIP release. For brevity, we describe the approximation for the 2-line case, readers are referred to Duenyas (1992) for more details.

Consider a CONWIP assembly system with two lines (with n_1 cards in line 1, and n_2 cards in line 2) as in Figure 2. We let W_1 be the amount of time that a job from line 1 has to wait at assembly for a job from line 2. To calculate EW_1, we condition on the position of the jobs in line 2 at the time that the job from line 1 arrives at assembly. Let N_i be the number of jobs in line 2, machine i. Also, denote the number of jobs from line 2 waiting in front of assembly as N_{m_2+1}. Obviously, if $N_{m_2+1} > 0$, then there is at least one job from line 2 already waiting at assembly, and hence the expected delay that a job from line 1 experiences waiting for a job from line 2 at assembly is 0 in this case. In general, we have

$$EW_1 = \sum_{i=1}^{m_2+1} E[W_1 | N_i > 0, \sum_{p=i+1}^{m_2+1} N_p = 0] P(N_i > 0, \sum_{p=i+1}^{m_2+1} N_p = 0) \tag{4}$$

$$EW_1^2 = \sum_{i=1}^{m_2+1} E[W_1^2 | N_i > 0, \sum_{p=i+1}^{m_2+1} N_p = 0] P(N_i > 0, \sum_{p=i+1}^{m_2+1} N_p = 0) \tag{5}$$

and

$$Var[W_1] = EW_1^2 - (EW_1)^2 \tag{6}$$

Calculating the conditional expectations in (4) and (5) requires that we also condition on the amount of processing that the job in line 2 "closest to" assembly has had at the machine that it is being processed at the time the job from line 1 arrives at assembly. However, since our aim is to get a rough estimate of the expected waiting time, we ignore the amount of processing that the job may already have had and approximate the conditional expectation by

$$E[W_1 | N_i > 0, \sum_{p=i+1}^{m_2+1} N_p = 0] \simeq \sum_{p=i}^{m_2} x_{2,p} \tag{7}$$

Similarly, we let

$$E[W_1^2 | N_i > 0, \sum_{p=i+1}^{m_2+1} N_p = 0] \simeq \sum_{p=i}^{m_2} \sigma_{2,p}^2 + (\sum_{p=i}^{m_2} x_{2,p})^2 \tag{8}$$

Approximating the probabilities in (4) is more difficult however, since we do not know the distribution of the jobs in the network. Hence, we approximate these by supposing that while jobs in line 1 have to wait for jobs in line 2 for their assembly operation, jobs in line 2 are independent of jobs in line 1 and start their assembly operation regardless of whether or not there are jobs from line 1 in assembly. This makes line 2 a regular closed queueing network, and we can use the approximation by Shanthikumar and Gocmen to calculate utilizations for line 2. Furthermore, the approximation procedure by Shanthikumar and Gocmen approximates the closed queueing network by a load-dependent exponential queueing network, and calculates Buzen's coefficients (Buzen (1983)) for this network at each iteration. Let $G(i,j), i = 0, \ldots, n_2, j = 1, \ldots, m_2 + 1$ denote the Buzen's coefficients calculated by the Shanthikumar and Gocmen approximation procedure on its last iteration for the closed queueing network formed by line 2 and the assembly machine. Then, we have

$$P[N_i > 0, \sum_{p=i+1}^{m_2+1} N_p = 0] \simeq \frac{G(n_2, i) - G(n_2, i-1)}{G(n_2, m_2+1)} \tag{9}$$

Hence, we can estimate the first and second moments of the expected delay that jobs from line 1 experience waiting for jobs from line 2 by using (4), (5), (6) along with (7), (8), and (9).

Notice that just as jobs from line 1 wait for jobs from line 2 at assembly, the reverse is true as well. Hence, to capture the effect that both lines have on each other, we also need to calculate EW_2. In fact, the approximation does this recursively. The first step in the approximation is to start with the network formed by fabrication line 1 and the assembly machine (we denote this network by $\{F_1/F_A/n_1\}$) and to calculate EW_2, and $Var[W_2]$ by assuming jobs in line 1 do not have to wait for jobs from line 2. Since we have approximated the time that jobs in line 2 wait for jobs from line 1 by a random variable with mean EW_2 and variance $Var[W_2]$, we next compute EW_1 and $Var[W_1]$ by assuming that the network formed by line 2 and the assembly machine where jobs experience an additional wait W_2 is a regular closed network, and we use $\{F_2/F_A + W_2/n_2\}$ (where $F_A + W_2$ denotes a processing time with mean $x_A + EW_2$ and variance $\sigma_A^2 + Var[W_2]$) and the above equations to compute EW_1 and $Var[W_1]$, and continue in this manner until throughput converges. Below is a summary for the procedure to obtain an approximate value (θ_{ap}) for the throughput of a CONWIP assembly system:

Procedure for Computing θ_{ap} (2 lines)

1. For $i = 1, 2$ compute $\theta\{F_i/F_A/n_i\}$, using the throughput approximation for closed queueing networks in Shanthikumar and Gocmen . Let $h = argmin_i \theta\{F_i/F_A/n_i\}$. Renumber line h as line 1. Let $\theta_1 = \min_i \theta_i\{F_i/F_A/n_i\}$. Let $EW_1 = 0$, and $Var[W_1 = 0]$.

2. Compute EW_2, and $Var[W_2]$ using (4) and (6), and $\{F_1/F_A + W_1/n_1\}$.

3. Compute EW_1, and $Var[W_1]$ using (4) and (6), and $\{F_2/F_A + W_2/n_2\}$.

4. $\theta_{ap} = \theta\{F_1/F_A + W_1/n_1\}$. If $|\theta_{ap} - \theta_1| < \epsilon$ for a prespecified ϵ, then stop. Else, let $\theta_1 = \theta_{ap}$, go to 2.

Bibliography

[1] Akyildiz, I.F., 1988. "On the Exact and Approximate Throughput Analysis of Closed Queueing Networks with Blocking," *IEEE Transactions on Software Engineering* **14,** 62.

[2] Ammar, M.H., 1980. "Modelling and analysis of unreliable manufacturing assembly networks with finite storage," MIT Laboratory for Information and Decision Sciences, Report LIDS-TH-1004.

[3] Baker, K.R., S.G. Powell, and D.F. Pyke, 1990. "Buffered and Unbuffered Assembly Systems with Variable Processing Times," *Journal of Manufacturing and Operations Management* **3,** 200.

[4] Baker, K.R., S.G. Powell, and D.F. Pyke, 1993. "Optimal Allocation of Work in Assembly Systems," *Management Science* **39,** 101.

[5] Bhat, U.N., 1986. "Finite capacity assembly-like queues," *Queueing Systems: Theory and Applications* **1,** 85.

[6] Bitran, G.R., and L. Chang, 1987. "A mathematical programming approach to a deterministic kanban system," *Management Science* **33,** 427.

[7] Bonomi, F., 1987. "An approximate analysis for a class of assembly-like queues," *Queueing Systems: Theory and Applications* **1,** 289.

[8] Buzacott, J.A., S.M. Price, and J.G. Shanthikumar, 1992. "The Performance of Kanban Controlled Serial Production Systems," to appear in Proceedings of US/German Conference on New Devlopments in OR in Production Planning and Control, June 1992, Hagen Germany.

[9] Buzen, J.P., 1973. "Computational Algorithms for Closed Queueing Networks with Exponential Servers," *Communications of the ACM* **16,** 527.

[10] Deleersnyder, J., T.J. Hodgson, H. Muller, and P. O'Grady, 1989. "Pull Systems: An analytic approach," *Management Science* **35,** 1079.

[11] Dallery, Y., and S.B. Gershwin, 1992. "Manufacturing flow line systems: a review of models and analytical results," *Queueing Systems, Theory and Applications* **12,** 3.

[12] Duenyas, I., 1992. "Estimating the Throughput of a Cyclic Assembly System," to appear in *International Journal of Production Research.*

[13] Duenyas, I., and W.J. Hopp, 1990. "Estimating Variance of Output from Cyclic Exponential Queueing Systems," *Queueing Systems: Theory and Applications* **7,** 337.

[14] Duenyas, I., and W.J. Hopp, 1992a. "Estimating the Throughput of an Exponential CONWIP Assembly System," to appear in *Queueing Systems: Theory and Applications.*

[15] Duenyas, I., and W.J. Hopp, 1992b. "CONWIP Assembly with Deterministic Processing and Random Outages," *IIE Transactions* **24,** 97.

[16] Duenyas, I., W.J. Hopp, and M.L. Spearman, 1991. "Characterizing the Output Process of a CONWIP Line with Deterministic Processing and Random Outages," to appear in *Management Science.*

[17] Gershwin, S.B., 1991. "Assembly/Disassembly Systems: An Efficient Decomposition Algorithm for Tree-Structured Networks," *IIE Transactions* **23,** 302.

[18] Hall, R.W., 1983. *Zero Inventories,* Dow Jones-Irwin, Homewood, IL.

[19] Harrison, J.M., 1973. "Assembly-like queues," *Journal of Applied Probability* **10,** 354.

[20] Hopp, W.J., and J.T. Simon, 1989. "Bounds and Heuristics for Assembly-Like Queues," *Queueing Systems: Theory and Applications* **4,** 137.

[21] Lipper, E.H., and B. Sengupta, 1986. "Assembly-like queues with finite capacity: bounds, asymptotics and approximations," *Queueing Systems: Theory and Applications* **1,** 67.

[22] Mascolo, M.D., R. David, and Y. Dallery, 1991. "Modeling and Analysis of Assembly Systems with Unreliable Machines and Finite Buffers," *IIE Transactions* **23,** 315.

[23] Mitra, D., and I. Mitrani, 1990. "Analysis of a Kanban Discipline for Cell Coordination in Production Lines," *Management Science* **36,** 1548.

[24] J.A. Muckstadt, and S. Tayur, 1991. "A comparison of alternative control mechanisms," Technical Report No: 962, Cornell University.

[25] Monden, Y., 1983. *Toyota Production System,* Industrial Engineering and Management Press.

[26] Ohno, T., 1988. *Toyota Production System: Beyond Large Scale Production,* Productivity Press, Cambridge, MA (original Japanese 1978).

[27] Reiser, M., and S. Lavenberg, 1980. "Mean Value Analysis of Closed MultiChain Queueing Networks," *J.ACM* **27,** 313.

[28] Shanthikumar, J., M. Gocmen, 1983. "Heuristic Analysis of Closed Queueing Networks," *International Journal of Production Research* **21,** 675.

[29] Spearman, M.L., 1992. "Customer Service in Pull Systems," *Operations Research* **40,** 948.

[30] Spearman, M.L., D.L. Woodruff, and W.J. Hopp, 1990. "CONWIP: A Pull Alternative to Kanban," *International Journal of Production Research* **28,** 879.

[31] Spearman, M.L., and M.A. Zazanis, 1992. "Push and Pull Production systems: Issues and Comparisons," *Operations Research* **40,** 521.

[32] Snowden, J.L., and J.C. Ammons, 1988. "A Survey of Queueing Network Packages for the Analysis of Manufacturing Systems," *Manufacturing Review* **1,** 14.

[33] Tayur, S.R., 1992a. "Structural Properties and a Heuristic for Kanban Controlled Serial Lines," Technical Report, Graduate School of Industrial Administration, Carnegie-Mellon University, to appear in *Management Science.*

[34] Tayur, S.R., 1992b. "Properties of serial kanban systems," *Queueing Systems* **12,** 297-318.

[35] Uzsoy, R., C.Y. Lee, and L.A. Martin-Vega, 1992. "A Survey of Production Planning and Scheduling Models in the Semiconductor Industry Part II: Shop Floor Control," Research Report, School of Industrial Engineering, Purdue University.

[36] Uzsoy, R., and L.A. Martin-Vega., 1991."Modelling Kanban-based Demand-Pull Systems: A Survey and Critique," *Manufacturing Review,* **3,** 155.

[37] Whitt, W. 1984. "Open and closed models for networks of queues," *AT&T Technical Journal* **63,** 1911-1979.