IIE Transactions (2000) **32**, 369–386

Extended kanban control system: combining kanban and base stock

YVES DALLERY[1] and GEORGE LIBEROPOULOS[2]

[1]*Laboratoire d'Informatique de Paris 6 (LIP6), Université Pierre et Marie Curie, 8, rue du Capitaine Scott, FR-75015 Paris, France*
E-mail: Yves.Dallery@Lip6.fr
[2]*Department of Mechanical and Industrial Engineering, University of Thessaly, Pedion Areos, GR-38334 Volos, Greece*
E-mail: glib@mie.uth.gr

Received March 1998 and accepted May 1999

This paper introduces a new mechanism for the coordination of multi-stage manufacturing systems. This mechanism is called the *Extended Kanban Control System* (EKCS) and depends on two parameters per stage, the number of kanbans and the base stock of finished parts. The EKCS is a combination of the classical Kanban and Base Stock control systems and includes each system as a special case. The dynamics of the EKCS are described, in particular, in relation to the dynamics of the Generalized Kanban Control System (GKCS), a known control mechanism that also includes the Kanban and Base Stock control systems as special cases. Advantages of the EKCS over the GKCS are discussed. Finally, properties of the dynamics of the EKCS are presented. One important property is that the capacity of the EKCS depends only on the number of kanbans but not on the base stock of finished parts.

1. Introduction

In many manufacturing systems the processing of parts proceeds in stages. Each stage may be seen as a production/inventory system consisting of a manufacturing process and an output buffer. The manufacturing process may consist of a single machine or a subnetwork of several machines (e.g., a production line, a job shop, a manufacturing cell, etc.) that performs a distinct processing operation (e.g., machining, forming, assembly, inspection, testing, etc.) on parts that it receives from the output buffer of the preceeding or upstream stage (or stages, in case of an assembly operation) and produces parts that are stored in the output buffer of the stage. The allocation of functions, resources, and products to stages is a major issue in the design of multi-stage manufacturing systems and addresses the questions of *what* to make and *how* to make it?

Once the manufacturing process and methods have been established and the stages of production have been defined, another very important design decision is the determination of the mechanism to control the flow of material through the manufacturing system. Material flow control is an optimization problem that addresses the question of *when* and *how much* to make in order to achieve a satisfactory customer service level, while keeping low in-process inventories. Difficulties in control arise because of queueing delays due to variability in production capacity and in demand.

One approach to tackling the material flow control problem is to formulate it as a stochastic optimal control problem and then try to determine an optimal control policy for this problem [1]. Thus far, this approach has been successful only for very simple systems. Moreover, an optimal policy, assuming one can be found even for realistic systems, will very likely be too complicated to be of any practical value. Knowledge of an optimal policy or its properties, however, may point to the design and help to assess the performance of simpler sub-optimal policies.

A more practical approach is to restrict the search for a material flow control policy to a class of simple sub-optimal policies that are easy to implement, and try to determine the optimal policy within the class. Much of the research effort in this area has focused on simple control systems that depend on a small number of parameters per stage and have often emerged from actual industrial practice [2–5].

In this paper we concentrate on material flow control systems in which production is triggered by actual demands. Such control systems are often referred to as *pull* control systems. A simple pull control system used in inventory control is the Base Stock Control System (BSCS) [2,3]. In the BSCS, every stage has a target inventory of finished parts, called *base stock*. When a demand for an end item arrives to the system, it is immediately transmitted to every stage where it authorizes the release of a new part. The advantage of this mechanism is that the system responds instantly to

demand. Its disadvantage is that is provides no limit on the number of parts in the system.

The most well-known pull control system is the Kanban Control System (KCS) [2,5]. The KCS was originally used in the Toyota production line in the mid-seventies and is often cited as the single technique most closely associated with the just-in-time philosophy [6,7]. In the KCS, a stage is authorized to start working on a new part only when a production authorization card, called *kanban* in Japanese, is available. A kanban becomes available only when a finished part of the stage is transferred to the next or downstream stage. When a kanban becomes available, it both transmits a demand for and authorizes the production of a new part. The advantage of this mechanism is that the number of parts in every stage is limited by the number of kanbans of that stage. Its disadvantage is that the system may not respond instantly to demand.

In summary, both the BSCS and the KCS are decentralized control mechanisms in that the decision to release a part into each stage depends on the state of that stage only. The advantage of the BSCS is that it uses global demand information, whereas the KCS uses local demand information. The advantage of the KCS is that it takes into consideration the limited production capacity of each stage whereas the BSCS does not.

A pull control mechanism that is more versatile but also more complex than the BSCS and the KCS is the Generalized Kanban Control System (GKCS) [8–10]. The GKCS depends on two parameters per stage, the base stock of finished parts and the number of kanbans, and includes the KCS and the BSCS as special cases. It should be noted that the Production Authorization Control (PAC) system presented by Buzacott and Shanthikumar [2,11] is equivalent to the GKCS presented in Buzocott [8], except that the PAC system also includes the options of: (i) having a delay between the arrival of a demand for a finished part and the demand for the production of a new part in any given stage; and (ii) waiting for the accumulation of a certain number of demands for the production of new parts by any given stage before transmitting these demands in a packet to that stage.

In this paper we develop a new pull control system for multi-stage manufacturing systems called the *Extended Kanban Control System* (EKCS). The EKCS, like the GKCS, depends on two parameters per stage, the base stock of finished parts and the number of kanbans, but its operation is simpler than that of the GKCS. In fact, the EKCS is a combination of the KCS and the BSCS and it includes each system as a special case.

The characteristic of the KCS is that in every stage a single parameter, the number of kanbans, plays two roles: (i) it is the number of cards used to authorize the production of new parts at the stage; and (ii) it is the base stock of finished parts of the stage. This "two-roles-in-one-parameter" characteristic may lead to bad system performance especially when demand or effective processing times are highly variable. For instance, in a situation of high demand variability, one would like to have a "large" number of kanbans at times of high demand, to quickly respond to demand. At the same time, one would like to have a "small" number of kanbans at times of low demand, to reduce inventory costs, since the number of kanbans is equal to the target inventory of finished parts. To compromize these two tendencies, one would end up setting the number of kanbans somewhere in between "large" and "small", thus meeting neither objective (quick customer response and low in-process inventories) too well. Indeed, it has often been reported that kanban control does not work well unless demand and the flow of parts are fairly constant.

This problem is remedied in the EKCS where the roles of the two parameters in every stage, the number of kanbans and the base stock of finished parts, are clearly separated. For instance, in the EKCS one could have a "low" base stock of finished parts to keep low inventories in a nominal demand context and a fairly "large" number of kanbans to quickly respond to surges in demand. Moreover, in the EKCS, demands are transmitted to all stages immediately upon their arrival to the system as is the case in the BSCS (this is not the case in the KCS), so that all stages can have this information as soon as it becomes available.

An important consequence of the separability of the role of the two parameters in the EKCS is that the production capacity of the EKCS only depends on the number of kanbans per stage and not on the base stock of finished parts per stage (see Section 6). This is not the case with GKCS, where the roles of the two parameters are not clearly separable, and where the production capacity of the system depends on both the number of kanbans and the base stock of finished parts per stage. This property becomes very useful when designing system parameters. Indeed, determining "good" values for the design parameters (the number of kanbans and the base stock levels) is very difficult for the GKCS. For the EKCS, this can be significantly simplified. We suggest a two-step approach. In the first step, the number of kanbans per stage is determined so as to achieve a given production capacity. In the second step the base stock levels are calculated so as to achieve a given service level (e.g., at least 99% of the demands must be satisfied without delay).

While installing the KCS can be done easily, implementing it requires fundamental changes in production to bring it in line with the just-in-time philosophy [3,6], and this is far from being easy. We strongly believe that the EKCS offers the opportunity to significantly improve the performance of pull production control systems. Some preliminary investigations about the improvement in performance (i.e., the reduction in operation costs) going from the KCS to the EKCS have indeed confirmed this belief [12]. Robustness is another key issue that also brings out the advantage of the EKCS. It is a known fact

that things are constantly changing in a real production environment. It is therefore very important for any production management policy to be robust to these changes, that is, to continue to perform well under conditions different from the nominal condition it was designed for. Again, because of its flexibility, the EKCS can adapt better to changes.

It should be noted that in addition to the control mechanisms mentioned above, other mechanisms have also been considered in the literature (see Buzacott and Shanthikumar [2] and Liberopoulos and Dallery [5] and references therein). In particular, a simple pull control mechanism that also depends on only one parameter per stage is the Finite Buffer Control System (FBCS) (see Buzacott and Shanthikumar [2] and references therein). A more general control mechanism than the FBCS that depends on three parameters per stage is the General Blocking Control System (GBCS) [13]. The GBCS is a combination of the KCS and the FBCS. An important limitation of the FBCS and the GBCS is that they are restricted to systems in which every stage consists of a single machine. This is a major restriction for practical applications. For this reason these two control systems are not discussed in this paper. Note, however, that even in the special case where each stage consists of a single machine, the EKCS is different from the GBCS (see Liberopoulos and Dallery [5] for more details).

The rest of this paper is organized as follows. In Section 2 we present general features of pull control systems. In Section 3 we introduce and describe the EKCS. In Section 4 we present several basic properties of the EKCS in the form of constraints. In Section 5 we derive evolution equations describing how the timings of different events in the EKCS are related to each other, and we discuss the effect of the parameters of the EKCS on these timings. In Section 6 we present an important property of the EKCS which states that the production capacity of the EKCS depends only on the number of kanbans in each stage. In Section 7 we discuss how the definition of the EKCS could be broadened by adding more parameters. Finally, conclusions are drawn in Section 8. A review of the BSCS, KCS and GKCS is given in Appendix A. Appendix B contains the proof of Property 10. Finally, in Appendix C we compare the EKCS and the GKCS using the evolution equations derived in Section 5.

2. Characteristic features of pull control systems

We consider a manufacturing system in which the production of parts proceeds in stages. Each stage is a production/inventory system made of a manufacturing process and an output buffer. The manufacturing process consists of a single machine or a subnetwork of several machines (e.g., a production line, a job shop, a manufacturing cell, etc.). It contains parts that are currently being processed in the stage (either waiting for or receiving processing on one of the workstations of the subnetwork) and are referred to as the *Work-In-Process* (WIP) of the stage. The output buffer is a storage area that contains parts that have completed processing in the stage and are referred to a the *finished parts* of the stage.

The manufacturing system is fed by raw parts that are drawn from a raw parts buffer, and releases finished parts to customers. It is assumed that either there is an infinite supply of raw parts in the raw parts buffer, or that raw parts arrive to the raw parts buffer according to a given arrival process.

For simplicity, we restrict our attention to manufacturing systems having N stages in series. In a serial system, the manufacturing process in each stage processes parts that it receives from the output buffer of the preceding or upstream stage (or the raw parts buffer, if it is the first stage), and produces parts that are stored in the output buffer of the stage. These parts are then transferred to the manufacturing process of the following or downstream stage (or to the customer, if it is the last stage). Note that the finished parts of a given stage are the raw parts of the next stage. Figure 1 shows a manufacturing system having N stages in series. We assume that there is a single type of parts and no machine setup times. Extentions that incorporate more features lead to more complex control problems and need to be explored. Some references on assembly systems are Di Mascolo and Dallery [14], Chaouiya *et al.* [15] and Sbiti *et al.* [16]. Work on multi-product systems is reported in Baynat *et al.* [17].

We are interested in mechanisms to control material flow through the manufacturing system. We restrict our attention to simple pull control systems that depend on a small number of parameters per stage and where the decision of when to release a part into the manufacturing process of a stage is triggered by external customer demands. More precisely, the decision process of when to release a part into the manufacturing process of a stage has the following characteristics.

To release a part into the manufacturing process of a stage, one requires the following: (i) a demand for the production of a new finished part in the stage; (ii) an authorization for the production of a new finished part in the stage; and (iii) a finished part in the output buffer of the preceding or upstream stage (or a raw part in the raw parts buffer, if it is the first stage). To deliver a stage-N finished part to the customer one does not require an

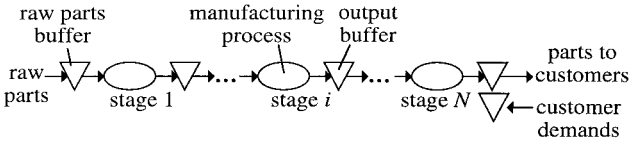

Fig. 1. Representation of a manufacturing system having N stages in series.

authorization but only; (i) a demand for a stage-N finished part; and (ii) an available stage-N finished part in the output buffer of stage N.

Every customer demand arriving to the system may be thought of as a $(N+1)$-element vector whose components are a demand for a stage-N finished part and a demand for the production of a new finished part in stage $i, i = 1, \ldots, N$; however, the instant at which each component of the demand vector is transferred to its respective stage depends on the particular control system in place. The control system also determines the instant at which an authorization for the production of a new part is made available at each stage. Once a part is released into a stage, it receives processing in the manufacturing facility of the stage, and, upon completion of its processing, it is immediately stored in the output store of the stage, irrespectively of the control system in place.

To be able to compare different pull control systems having the above characteristics, we use a common notation and a common modeling approach. The common notation we use is the following:

p_i,	$i = 1, \ldots, N$:	a stage-i finished part;
q_i,	$i = 1, \ldots, N$:	a part currently being processed in stage i;
d_i,	$i = 1, \ldots, N$:	a demand for the production of a new p_i;
d_{N+1} :		a demand for a p_N;
$\mathbf{d}_i$,	$i = 1, \ldots, N+1$:	an i-element vector $(d_1, d_2, \ldots, d_i)$; note that $\mathbf{d}_1 = (d_1)$;
a_i,	$i = 1, \ldots, N$:	an authorization for the production of a new p_i.

The common modeling approach we use is that we model any pull control system as a queueing network with synchronization stations. A synchronization station consists of a server with instant service time, fed by two or more queues. A synchronization station operates as follows. As soon as there is at least one customer in each of the queues that feed the server, these customers instantaneously move through the station. As a result, at least one of the queues that feed the server is always empty. Customers that enter the server may be separated into more, or joined into fewer, customers upon exiting the server. The operation of synchronization stations will become more obvious in the following sections. Synchronization stations in queueing systems are related to *transitions* in Petri nets.

The queueing network models of all pull control systems have the following common elements. All the queues that are between stages i and $i+1$ are contained in a *linkage station* denoted $J_{i,i+1}$. $J_{0,1}$ contains all queues that are between the supply of raw parts and stage 1. $J_{N,N+1}$ contains all queues that are between stage N and the customer. MP_i is a queueing subnetwork representing the manufacturing facility in stage-i.

When studying a pull control system, our primary concern is to understand the dynamics and evaluate the performance of the system as it is driven by customer demands. It is also very important, however, to determine what is the maximum demand rate that the system can meet, that is, the *production capacity* of the system. To determine the production capacity of a pull control system we study the so-called *saturated* version of the system, that is the original system under the assumption that there is an infinite number of raw parts and customer demands. The throughput of the saturated system is the production capacity of the original system.

In the following sections we present a new pull control system called the Extended Kanban Control System (EKCS). The EKCS is a combination of the BSCS and the KCS and includes each of these systems as a special case. It is worth noting that the EKCS does not belong to the class of Production Authorization Card (PAC) systems [2,9], in which the BSCS, KCS, and GKCS belong. A review of the BSCS, KCS and GKCS is given in Appendix A. Also, it is worth repeating that even in the special case of a single machine per stage, the EKCS (as well as the GKCS) is different from the GBCS [13]. More information on the exact operation of these and other pull control mechanisms is provided in Liberopoulos and Dallery [5].

3. Extended kanban control system

This section describes the Extended Kanban Control System (EKCS). Figure 2 shows the queueing network model of an EKCS having N stages in series. The contents and initial values of each queue (or network of queues, in the case of MP_i) are shown in Table 1.

In the EKCS each stage i has K_i kanbans a_i that are used to authorize the production of new stage-i finished parts. Initially, S_i of these kanbans $(0 \leq S_i \leq K_i)$ are attached onto an equal number of finished parts p_i and are stored in queue PA_i as pairs (p_i, a_i). The remaining $K_i - S_i$ kanbans a_i are stored in queue A_i. Queue P_0 represents the raw parts buffer. The initial number of raw parts in P_0 and the arrival of new raw parts into P_0 fall outside the scope of the control mechanism and are considered as

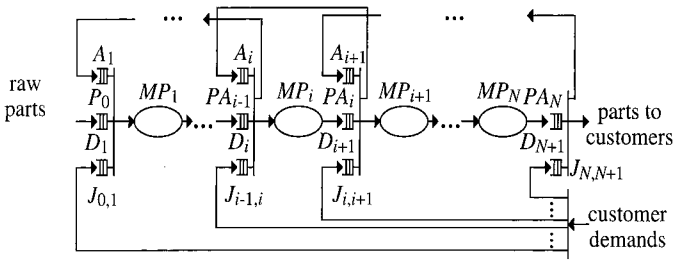

Fig. 2. Queueing network model of an EKCS having N stages in series.

Table 1. Contents and initial values of queues in the queueing network model of an EKCS having N stages in series.

	Queue	*Contents*	*Initial value*
MP_i	$i = 1,\ldots,N$	(q_i, a_i)	0
PA_i	$i = 1,\ldots,N$	(p_i, a_i)	S_i
A_i	$i = 1,\ldots,N$	a_i	$K_i - S_i$
D_i	$i = 1,\ldots,N+1$	d_i	0

given. The EKCS, therefore, depends on two parameters per stage, namely S_i and $K_i, i = 1, \ldots, N$.

Queues PA_i, A_{i+1}, and D_{i+1} in linkage station $J_{i,i+1}, i = 1, \ldots, N-1$, are joined in a synchronization station. If there is a pair (p_i, a_i) in PA_i, a kanban a_{i+1} in A_{i+1}, and a demand d_{i+1} in D_{i+1}, then: (i) the kanban a_i is detached from p_i; (ii) the kanban a_{i+1} is attached onto p_i, which is relabeled q_{i+1}, and together they are transferred downstream into MP_{i+1} as a pair (q_{i+1}, a_{i+1}); (iii) the demand d_{i+1} is satisfied and is therefore dropped; and (iv) the detached kanban a_i is transferred upstream into A_i. When the part q_{i+1} finishes its processing in MP_{i+1} it is relabeled p_{i+1} and together with the kanban a_{i+1} that was attached onto it, they join queue PA_{i+1} as a pair (p_{i+1}, a_{i+1}).

In the last stage there is no need for an authorization a_{N+1} to release a finished part p_N to the customer; therefore, as soon as there is a pair (p_N, a_N) in PA_N and a demand d_{N+1} in D_{N+1}, then: (i) the kanban a_N is detached from p_N; (ii) p_N is released to the customer; (iii) the demand d_{N+1} is satisfied and is therefore dropped; and (iv) the kanban a_N is transferred upstream into A_N.

Also, in the first stage raw parts do not have any kanbans attached to them; therefore, as soon as there is a raw part in P_0, a kanban a_1 in A_1, and a demand d_1 in D_1, then: (i) the kanban a_1 is attached onto the raw part, which is labeled q_1, and together they are transferred downstream into MP_1 as a pair (q_1, a_1); and (ii) the demand d_1 is satisfied and is therefore dropped.

Production in the system is driven by customer demands. When a customer demand vector $\mathbf{d}_{N+1}$ arrives to the system, then the following happens: (i) the demand vector $\mathbf{d}_{N+1}$ is separated into its components, $d_i, i = 1, \ldots, N+1$, and each component d_i is immediately transferred upstream to its respective queue, D_i; (ii) if there is a pair (p_{i-1}, a_{i-1}) in $PA_{i-1}, i = 2, \ldots, N+1$, (or a raw part in P_0) and a kanban a_i in A_i, then the kanban a_{i-1} is detached from p_{i-1} and is transferred upstream to A_{i-1}, the kanban a_i is attached onto the part p_{i-1}, which is relabeled q_i, and together they are transferred downstream to MP_i (or to the customer, if $i = N+1$), and the demand d_i is satisfied. If there are no pairs (p_{i-1}, a_{i-1}) in PA_{i-1}, or no kanbans a_i in A_i, then the demand d_i stays in D_i as a backordered demand until a pair (p_{i-1}, a_{i-1}) and a kanban a_i become available in PA_{i-1} and A_i, respectively.

The dynamics of the EKCS are a combination of the dynamics of the BSCS and the KCS. In the EKCS each demand $d_i, i = 1, \ldots, N+1$, is transferred upstream to D_i immediately upon its arrival to the system, as is the case in the BSCS. A finished part $p_i, i = 1, \ldots, N$ (or raw part, if $i = 0$), however, is transferred downstream into MP_{i+1} (or to the customer, if $i = N$) only if one of a finite number of kanbans a_{i+1} is available in A_{i+1}, as is the case in the KCS (in stage N no kanbans are needed). When the latter happens, the part p_i releases its stage-i kanban a_i and engages the stage-$(i+1)$ kanban a_{i+1}. The released kanban a_i is transferred upstream into A_i. Thus, in the EKCS the role of the kanbans is only to authorize the transfer of finished parts p_i downstream and not to also authorize the transfer of demands upstream, as is the case in the KCS. To clarify matters, Fig. 3 shows the entire queueing network model of an EKCS having two stages in series.

The EKCS, like the GKCS, has two parameters per stage. One advantage of the EKCS over the GKCS is that the EKCS is simpler than the GKCS. This is because the EKCS has only one synchronization station between two consecutive stages, whereas the GKCS has two. Another advantage of the EKCS over the GKCS, from a practical point of view, is that in the EKCS all parts always have kanbans attached to them so they are easy to identify, whereas in the GKCS finished parts do not have kanbans attached to them. Yet another advantage of the EKCS over the GKCS is that in the EKCS the roles of the two parameters in each stage are clearly distinguishable. One important consequence of this is that the production capacity of the EKCS only depends on the parameters K_i and not on S_i (see Section 6 that follows). As we shall see in the following section, the EKCS also includes the BSCS and KCS as special cases.

In the sections that follow we give several properties of the EKCS.

4. Basic properties of the EKCS

In this section we present several properties of the EKCS. These include invariants and bounds on the contents of

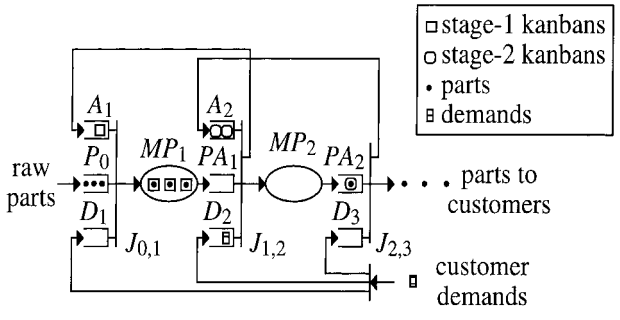

Fig. 3. Queueing network model of an EKCS having two stages in series.

various queues in the queueing network model of the EKCS and two special cases of the EKCS. For any queue or network of queues, Q, in the system, we use the notation $M(Q)$ to denote the number of customers present in Q. For example, $M(MP_i)$ denotes the WIP in the manufacturing process of stage i. Actually, $M(Q)$ is also a function of time, but for notational simplicity we omit this dependence on time, because all the relations involving $M(Q)$ that we develop hold at all times.

By definition of the initial state of the EKCS, queue $A_i, i = 1, \ldots, N$, has $K_i - S_i$ free kanbans, where clearly $K_i - S_i$ must be greater than or equal to zero. The two parameters of stage i are therefore constrained by:

$$K_i \geq S_i, \quad i = 1, \ldots, N. \tag{1}$$

Also, the following relationships hold.

$$M(A_i)M(PA_{i-1})M(D_i) = 0, \quad i = 2, \ldots, N, \tag{2}$$

$$M(A_1)M(P_0)M(D_1) = 0, \tag{3}$$

$$M(PA_N)M(D_{N+1}) = 0. \tag{4}$$

Relationships (2)–(4) hold by definition of a synchronization station. They state that at least one of the queues that feed any synchronization station must be empty.

4.1. *Invariants*

Properties 1–3 that follow are invariants relating the contents of various queues in the queueing network model of the EKCS to a certain constant. Property 1, in particular, expresses the fact that within the queueing network model of the EKCS shown in Fig. 2, there are N closed subnetworks, one in each stage $i = 1, \ldots, N$. The closed subnetwork of stage i has K_i customers and corresponds to the circulation loop of stage-i kanbans a_i.

Property 1. *In the EKCS the following holds*:

$$M(A_i) + M(MP_i) + M(PA_i) = K_i, \quad i = 1, \ldots, N. \tag{5}$$

Proof. When the EKCS is in its initial state, A_i has $K_i - S_i$ kanbans a_i, MP_i is empty, and PA_i has S_i pairs (p_i, a_i); therefore, Equation (5) is true initially. By observing the events that can modify the state of the system, it is clear that as the EKCS evolves, starting from its initial state, (5) remains true, since: (i) when a pair (q_i, a_i) leaves MP_i, it joins PA_i as a pair (p_i, a_i); (ii) when a pair (p_i, a_i) leaves PA_i, the kanban a_i is transferred upstream to A_i; and (iii) when a kanban a_i leaves A_i, that kanban is attached to a part p_{i-1} and together they enter MP_i as a pair (q_i, a_i). ■

Property 2. *In the EKCS the following holds*:

$$M(A_i) + M(D_{i+1}) - M(D_i) = K_i - S_i, \quad i = 1, \ldots, N. \tag{6}$$

Proof. When the EKCS is in its initial state, A_i has $K_i - S_i$ kanbans a_i and D_i and D_{i+1} are empty; therefore, Equation (6) is true initially. By observing the events that can modify the state of the system, it is clear that as the EKCS evolves starting from its initial state, (6) remains true, since: (i) when a kanban a_i leaves A_i, a demand d_i also leaves D_i; (ii) when a demand d_{i+1} leaves D_{i+1}, a kanban a_i is transferred upstream to A_i; and (iii) when a demand d_i joins D_i, a demand d_{i+1} also joins D_{i+1}. ■

Property 3. *In the EKCS the following holds*:

$$M(PA_i) - M(D_{i+1}) = S_i - M(MP_i) - M(D_i), \quad i = 1, \ldots, N. \tag{7}$$

Proof. Equation (7) follows after substituting $M(A_i)$ from (5) into (6) and rearranging terms. ■

4.2. *Bounds*

Properties 4–7 that follow are inequalities expressing bounds on the contents of various queues. These bounds follow easily from the invariants (5)–(7).

Property 4. *In the EKCS the following hold*:

$$0 \leq M(A_i) \leq K_i, \quad i = 1, \ldots, N, \tag{8}$$

$$0 \leq M(MP_i) \leq K_i, \quad i = 1, \ldots, N, \tag{9}$$

$$0 \leq M(PA_i) \leq K_i, \quad i = 1, \ldots, N, \tag{10}$$

$$0 \leq M(MP_i) + M(PA_i) \leq K_i, \quad i = 1, \ldots, N. \tag{11}$$

Proof. Inequalities (8)–(11) follow directly from Equation (5) since all variables in (5) are non-negative. ■

The lower bound (zero) and upper bound (K_i) are attainable in all of the inequalities (8)–(11), except for inequality (10) when $i = N$. When $i = N$, the variable in the middle of inequality (10), i.e., $M(PA_N)$, has a stricter attainable upper bound. This is expressed by Property 7 that follows. To see why the bounds in inequalities (8)–(11) are attainable (except for inequality (10) when $i = N$), consider the following three plausible sequences of events.

(i) Suppose that processing in MP_{i-1}, for some i, is very slow. In this case PA_{i-1} eventually becomes empty and causes MP_i to be starved. Eventually then, all parts are flushed out of MP_i and PA_i, and all stage-i kanbans accumulate in A_i where they are blocked, since there are no parts in PA_{i-1}. Thus, in the end, A_i has K_i kanbans a_i and PA_i and MP_i are empty.

(ii) Suppose that processing in MP_i, for some i, is very slow. In this case, PA_i eventually becomes empty, since MP_i sends it no parts, so that all stage-i kanbans are in A_i and MP_i. Eventually then, all free kanbans in A_i are attached to parts in PA_{i-1} and are

transferred to MP_i. Thus, in the end, MP_i has K_i pairs (q_i, a_i) and PA_i and A_i are empty.

(iii) Suppose that MP_i has K_i pairs (q_i, a_i), while A_{i+1} is empty and will not be receiving any kanbans for a long time to come. Eventually then, all K_i pairs (q_i, a_i) in MP_i are transferred into PA_i as pairs (p_i, a_i). There, they are blocked, because there are no kanbans a_{i+1} in A_{i+1}. Thus, in the end, PA_i has K_i pairs (p_i, a_i), whereas A_i and MP_i are empty.

Property 5. *In the EKCS the following holds*:

$$M(PA_i) - M(D_{i+1}) \leq S_i, \quad i = 1, \ldots, N. \tag{12}$$

Proof. Inequality (12) follows directly from Equation (7) since all variables in (7) are non negative. ■

Property 6. *In the EKCS the following holds*:

$$-S_i \leq M(D_{i+1}) - M(D_i) \leq K_i - S_i, \quad i = 1, \ldots, N. \tag{13}$$

Proof. Inequality (13) follows after substituting $M(A_i)$ from (6) into (8) and rearranging terms. ■

When $i = N$, the variable in the middle of inequality (10), i.e., $M(PA_N)$ has a stricter attainable bound. In particular, we have the following.

Property 7. *In the EKCS the following holds*:

$$0 \leq M(PA_N) \leq S_N. \tag{14}$$

Proof. Equation (4) states that when $M(PA_N) > 0$, then $M(D_{N+1}) = 0$. With this in mind, inequality (14) follows directly from inequality (12) evaluated at $i = N$. ■

4.3. *Special cases*

Properties 8 and 9 that follow express two special cases where the EKCS is equivalent to the KCS and the BSCS, respectively.

Property 8. *The EKCS with $K_i = \infty, S_i \geq 0, i = 1, \ldots, N$, is equivalent to the BSCS having a base stock of S_i finished parts in stage i.*

Proof. Consider the EKCS shown in Fig. 2, with $K_i = \infty, S_i \geq 0, i = 1, \ldots, N$. Queues A_i have an infinite number of kanbans a_i and therefore play no role in the synchronization station they belong to, since they never block the passage of parts through that synchronization station; hence they can be eliminated. Once queues A_i are eliminated from the queueing network in Fig. 2, the remaining network is the same as the queueing network model of the BSCS in Fig. A1 and has the same initial conditions. ■

Property 9. *The EKCS with $K_i = S_i, i = 1, \ldots, N$, is equivalent to the KCS having K_i kanbans in stage i.*

Proof. Consider the EKCS shown in Fig. 2, with $K_i = S_i, i = 1, \ldots, N$. In the initial state of the system there are no available kanbans a_i in A_i since $K_i - S_i = 0$. Instead, all kanbans a_i are attached to parts in PA_i. A kanban a_i becomes available only when a finished part p_i is transferred downstream to MP_{i+1}, as is the case in the KCS. Also, since $K_i - S_i = 0$, (6) implies that $M(D_i) \geq M(A_i)$, $i = 1, \ldots, N$. Queues D_i, $i = 1, \ldots, N$, therefore, play no role in the synchronization station they belong to, since they never block the passage of parts through that synchronization station; hence they can be eliminated. Once queues $D_i, i = 1, \ldots, N$, are eliminated from the queueing network in Fig. 2, the remaining network is the same as the queueing network model of the KCS in Fig. A2, where queues $A_i, i = 1, \ldots, N$, in Fig. 2 play the same role as queues $\mathbf{D}A_i$ in Fig. A2. ■

5. Evolution equations of the EKCS

The purpose of this section is to describe in detail the dynamics of kanbans and material flow in the EKCS in order to determine the impact of the choice of system parameters on the departure times of parts from various points in the system. To do this we use the fact that the EKCS can be classified as a Fork/Join Queueing Network with Blocking (FJQN/B). FJQN/B's have been studied by Dallery *et al.* [18,19]. The dynamics of the EKCS, and indeed of any FJQN/B, can be described by recursive evolution equations that utilize the operators "+" and "max" only [2,18–20]. These equations relate the timing of a particular event in the EKCS to the timings of events that must precede it. To elaborate, let $D_{(i-1,i),n}$, $i = 2, \ldots, N+1$, be the departure time of the nth pair (p_{i-1}, a_{i-1}) from the synchronization station in $J_{i-1,i}$, i.e., the time of the nth release of a pair (q_i, a_i) into MP_i (or of a finished part p_N to the customer, if $i = N+1$), and the transfer of the nth kanban a_{i-1} into A_{i-1}. Let $D_{i,n}, i = 1, \ldots, N$, be the departure time of the nth pair (q_i, a_i) from MP_i, i.e., the time of the nth arrival of a pair (p_i, a_i) into PA_i. Let $D_{0,n}$ be the release time of the nth raw part into P_0 and S_0 be the initial number of raw parts in P_0. Let $D_{d,n}$ be the arrival time of the nth customer demand vector $\mathbf{d}_{N+1}$ to the system. These times are shown in Fig. 4. For the sake of simplicity assume that MP_i consists of a single machine, and let $\sigma_{i,n}, i = 1, \ldots, N$, be the processing time of the nth part at the machine in MP_i. Clearly, for any of the above times, $D_{\cdot,n}$, the following holds.

$$D_{\cdot,n-m} \leq D_{\cdot,n}, \quad n = 1, 2, \ldots, \ m = 0, 1, \ldots. \tag{15}$$

Also, by convention, we define

$$D_{\cdot,n} = \infty, \quad n \leq 0. \tag{16}$$

We now have the following result.

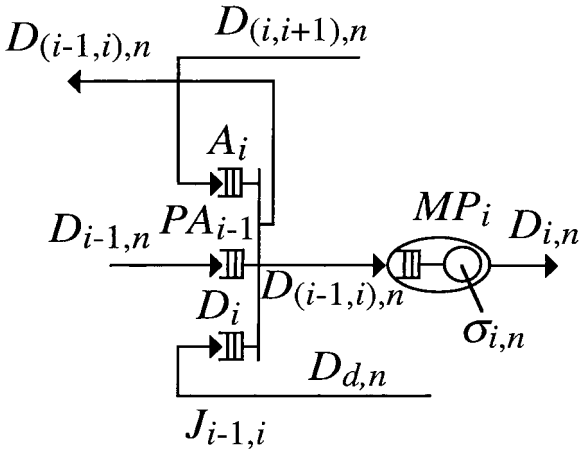

Fig. 4. Times of events at the ith stage of an EKCS having N stages in series.

Proposition 1. *In the EKCS the timings of events are related by the following evolution equations*:

$$D_{i,n} = \sigma_{i,n} + \max(D_{i,n-1}, D_{(i-1,i),n}), \quad i = 1,\ldots,N,\ n = 1,2,\ldots, \tag{17}$$

$$D_{(i-1,i),n} = \max\left(D_{d,n}, D_{i-1,n-S_{i-1}}, D_{(i,i+1),n-(K_i-S_i)}\right), \quad i = 1,\ldots,N+1,\ n = 1,2,\ldots, \tag{18}$$

where, by convention, the maximum over an empty set is $-\infty$ *and* $D_{(N+1,N+2),\cdot} = -\infty$.

Proof. Equation (17) represents the time at which the nth pair (q_i, a_i) has completed processing in MP_i and is transferred to PA_i as a pair (p_i, a_i). This time is equal to the time at which the nth part q_i begins its processing at the machine in MP_i plus its processing time. The first term in the right-hand side of (17), $\sigma_{i,n}$, is the processing time of the nth part at the machine in MP_i. The second term in the right-hand side of (17) represents the time at which the nth part begins its processing at the machine in MP_i. Indeed, to begin processing the nth part at the machine in MP_i, two conditions must be met: (i) the $(n-1)$th part must have completed its processing at the machine in MP_i so that the machine is free; and (ii) the nth part must have been released into MP_i.

Equation (18) represents the time at which the nth pair $(q_i, a_i), i = 1,\ldots,N$, is released into $MP_i, i = 1,\ldots,N$. To see why (18) holds, note that in order to release a pair (q_i, a_i) into MP_i one needs: (i) a demand d_i in D_i; (ii) a pair (p_{i-1}, a_{i-1}) in PA_{i-1} (or a raw part in P_0, if $i = 1$); and (iii) a kanban a_i in A_i.

Indeed, to release the nth pair (q_i, a_i) into MP_i, three conditions must be met: (i) the nth demand d_i must have arrived at D_i; (ii) the $(n - S_{i-1})$th pair (p_{i-1}, a_{i-1}) must have been transferred into PA_{i-1}, since there are initially S_{i-1} pairs (p_{i-1}, a_{i-1}) in PA_{i-1}; and (iii) the $(n-(K_i - S_i))$th kanban a_i must have been released into A_i, since there are initially $K_i - S_i$ kanbans a_i in A_i. Equation (18), for $i = N+1$, represents the time at which the nth part q_{N+1} is released to the customer. To see why (18) holds for $i = N+1$, note that in order to release a finished part p_N to the customer, one needs only: (i) a demand d_{N+1} in D_{N+1}; and (ii) a pair (p_N, a_N) in PA_N, i.e., there is no need for a kanban a_{N+1}. Indeed, to release the nth finished part p_N to the customer, two conditions must be met: (i) the nth demand d_{N+1} must have arrived at D_{N+1}; and (ii) the $(n - S_N)$th pair (p_N, a_N) must have been transferred into PA_N, since there are initially S_N pairs (p_N, a_N) in PA_N. ■

Equation (18) is recursive in that it expresses $D_{(i-1,i),\cdot}$ in terms of $D_{(i,i+1),\cdot}, i = 1,\ldots,N+1$. Expanding this recursion backwards, starting from $i = N+1$, and noting that, by definition, $D_{N+1,N+2,\cdot} = 0$, yields

$$\begin{aligned}
D_{(N,N+1),n} &= \max\left(D_{d,n}, D_{N,n-S_N}\right),\\
D_{(N-1,N),n} &= \max\left(D_{d,n}, D_{N-1,n-S_{N-1}}, D_{N,n-K_N}\right),\\
D_{(N-2,N-1),n} &= \max\big(D_{d,n}, D_{N-2,n-S_{N-2}}, D_{N-1,\,n-K_{N-1}},\\
&\qquad D_{N,n-K_N-(S_{N-1}-K_{N-1})}\big),\\
&\ \ \vdots\\
D_{(i-1,i),n} &= \max\big(D_{d,n}, D_{i-1,n-S_{i-1}}, D_{i,n-K_i},\\
&\qquad D_{i+1,n-K_{i+1}-(K_i-S_i)} \cdots\\
&\qquad D_{N,n-K_N-(K_{N-1}-S_{N-1})-\cdots-(K_i-S_i)}\big).
\end{aligned}$$

This can be written in a more compact form as

$$D_{(i-1,i),n} = \max\left[D_{d,n}, D_{i-1,n-S_{i-1}}, \max_{j=1}^{N}\left(D_{j,n-S_j-\sum_{m=1}^{j}(K_m-S_m)}\right)\right],\ i = 1,\ldots,N+1. \tag{19}$$

The evolution equations in Proposition 1 imply the following property.

Property 10. *Consider the EKCS, with parameters* S_i', K_i' *in place of* S_i, K_i, $i = 1,\ldots,N$, *and let* $D_{i,n}', i = 1,\ldots,N$, *and* $D_{(i-1,i),n}'$, $i = 1,\ldots,N+1$, $n = 1,2\ldots$, *denote the corresponding event times. Then, the following hold.*

(i) *If* $K_l' > K_l$, *for some* $l \in \{1,\ldots,N\}$, $K_i' = K_i$, *for all* $i \in \{1,\ldots,N\} - \{l\}$, *and* $S_i' = S_i$, *for all* $i \in \{1,\ldots,N\}$, *then*

$$D_{(i-1,i),n}' \le D_{(i-1,i),n}, \quad i = 1,\ldots,N+1,\ n = 1,2,\ldots, \tag{20}$$

$$D_{i,n}' \le D_{i,n}, \quad i = 1,\ldots,N,\ n = 1,2,\ldots\ . \tag{21}$$

(ii) *If* $S_l' > S_l$, *for some* $l \in \{1,\ldots,N\}$, $S_i' = S_i$, *for all* $i \in \{1,\ldots,N\} - \{l\}$, *and* $K_i' = K_i$, *for all* $i \in \{1,\ldots,N\}$, *then*

$$D_{(i-1,i),n}' \le D_{(i-1,i),n}, \quad i = l+1,\ldots,N+1,\ n = 1,2,\ldots, \tag{22}$$

$$D'_{i,n} \le D_{i,n}, \quad i = l+1, \ldots, N, \ n = 1, 2, \ldots, \tag{23}$$

$$D'_{(i-1,i),n} \le D_{(i-1,i),n+(S'_l - S_l)}, \quad i = 1, \ldots, l, \ n = 1, 2, \ldots, \tag{24}$$

$$D'_{i,n} \le D_{i,n+(S'_l - S_l)}, \quad i = 1, \ldots, l, \ n = 1, 2, \ldots. \tag{25}$$

The proof of Property 10 is presented in Appendix B. Part (i) of property 10 states that if the number of kanbans in stage l is increased from K_l to K'_l, the departure time of the nth part from *all* the synchronization stations and manufacturing processes in the EKCS will not increase but may decrease. Part (ii) of property 10 states that if the base stock in stage l is increased from S_l to S'_l, the departure time of the nth part from those synchronization stations and manufacturing processes that are *downstream* of stage l will not increase but may decrease. Moreover, the departure time of the nth part from those synchronization stations and manufacturing processes that are *upstream* of stage l (including stage l) will not increase but may decrease with respect to the departure time of the $n + (S'_l - S_l)$th part from the same synchronization stations and manufacturing processes in the original system. To see why the latter is true, note that an increase in the base stock of stage l from S_l to S'_l has the same effect as that of having $(S'_l - S_l)$ extra parts enter the system and receive processing all the way up to and including stage l, before the first demand arrives to the system. Therefore, the departure of the nth part in the EKCS with S'_l from any point upstream of $J_{l-1,l}$ corresponds to the departure of the $n + (S'_l - S_l)$th part in the EKCS with S_l from the same point.

The evolution equations in Proposition 1 also clearly imply the following property that we state without proof (see Baccelli and Liu [20] for similar arguments).

Property 11. *The departure times* $D_{(i-1,i),n}, i = 1, \ldots, N+1, n = 1, 2, \ldots,$ *and* $D_{i,n}, i = 1, \ldots, N, \ n = 1, 2, \ldots,$ *are non-decreasing in* $\sigma_{l,m}, D_{d,m}$ *and* $D_{0,m}$, *for any* $m \ge 1$ *and* $l \in \{1, \ldots, N\}$.

Property 11 states that if a processing times $\sigma_{l,m,}$ or a demand vector arrival time $D_{d,m}$, or a raw part arrival time $D_{0,m}$ is increased, the departure time of the nth part from all the synchronization stations and manufacturing processes in the EKCS will not decrease but may increase.

6. Production capacity of the EKCS

As was mentioned in Section 2, the production capacity of the EKCS is the throughput of the saturated EKCS. Figure 5 shows the queueing network model of the saturated version of the EKCS having N stages in series that

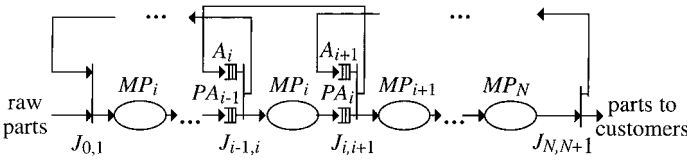

Fig. 5. Queueing network model of a saturated EKCS having N stages in series.

was shown in Fig. 2. Figure 5 is obtained from Fig. 2 as follows. In the saturated EKCS, queue P_0 in Fig. 2 has by definition an infinite number of raw parts. Also, queues $D_i, i = 1, \ldots, N+1$, have by definition an infinite number of demands d_i, since in the saturated EKCS there is an infinite number of customer demand vectors $\mathbf{d}_{N+1}$, all of which are immediately separated into their components $d_i, i = 1, \ldots, N+1$, and each component d_i is transferred upstream into its respective queue D_i. In the saturated EKCS, therefore, queues P_0 and $D_i, i = 1, \ldots, N+1$, play no role in the synchronization station they belong to, since they never block the transfer of parts through that synchronization station. Hence, they can be eliminated. Once P_0 and D_1 are eliminated from Fig. 2, A_1 remains the only queue in the synchronization station in $J_{0,1}$. Similarly, once D_{N+1} is eliminated, PA_N remains the only queue in the synchronization station in $J_{N,N+1}$. Clearly, if there is only one queue feeding a synchronization station, this queue can be eliminated, since any customer entering this queue immediately goes through the synchronization station. In the saturated EKCS, therefore, queues A_1 and PA_N can be eliminated. The resulting network after the above queues have been eliminated is shown in Fig. 5.

It can be shown in a similar way that in the queueing network model of the saturated version of the KCS having N stages in series, shown in Fig. A2, queues P_0, DA_1, PA_N, and $\mathbf{D}_{N+1}$ are eliminated. Thus, the queueing network model of the saturated version of the KCS having N stages in series is shown in Fig. 6.

We now have the following important properties.

Property 12. *The production capacity of the EKCS depends only on* $K_i, i = 1, \ldots, N$, *and is independent of* $S_i, i = 1, \ldots, N$.

Proof. The saturated EKCS shown in Fig. 5 is a basic FJQN/B containing N distinct elementary closed subnetworks, one in each stage, corresponding to the

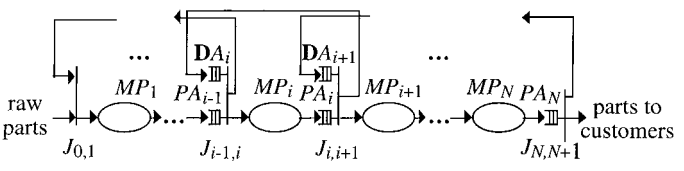

Fig. 6. Queueing network model of a saturated KCS having N stages in series.

circulation loop of the kanbans in that stage. Thus, the stage-i closed subnetwork has K_i customers. According to Theorem 2.3 of Dallery *et al.* [19], the throughput of a basic FJQN/B containing N distinct elementary closed subnetworks depends only on the fixed number of customers in each closed subnetwork – in this case $K_i, i = 1, \ldots, N$ – and not on the initial allocation of these customers among the different queues of this closed subnetwork. Parameter $S_i, i = 1, \ldots, N$, only affects this initial allocation, since initially in the EKCS, queue PA_i has S_i pairs (p_i, a_i) and queue A_i has $K_i - S_i$ kanbans a_i. The production capacity of the EKCS, therefore, depends only on $K_i, i = 1, \ldots, N$, and is independent of $S_i, i = 1, \ldots, N$. ■

Property 13. *The production capacity of the EKCS, with parameters K_i and $S_i, i = 1, \ldots, N$, is equal to the production capacity of the KCS with the same parameters $K_i, i = 1, \ldots, N$, as those in the EKCS.*

Proof. By comparing Figs. 5 and 6 it is clear that the saturated EKCS is equivalent to the saturated KCS, where queues $A_i, i = 2, \ldots, N$, in the saturated EKCS correspond to queues $\mathbf{D}A_i, i = 2, \ldots, N$, in the saturated KCS. The only difference between the two networks is their initial state. Both the saturated EKCS and the saturated KCS are basic FJQN/B's containing N distinct elementary closed subnetworks, one in each stage. The stage-i closed subnetwork in both saturated system has K_i customers. According to Theorem 2.3 of Dallery *et al.* [19], the throughput of a basic FJQN/B containing N distinct elementary closed subnetworks depends only on the fixed number of customers in each closed subnetwork and not on the initial state (marking) of the network. Therefore, by Theorem 2.3 of Dallery *et al.* [19], the throughput of both saturated systems, i.e., the production capacity of the original systems is the same. ■

An important concept that led to the EKCS is that the role of parameters S_i and $K_i, i = 1, \ldots, N$, should be clearly separated. The main reason for keeping a base stock of S_i stage-i finished parts p_i is to be able to quickly satisfy the demands d_i brought in by customers. The main role of stage-i kanbans a_i, on the other hand, is to authorize the production of new finished parts p_i to replenish the base stock of finished parts, when these parts are consumed by demands. In other words, S_i is related to the consumption of existing finished parts, which itself is related to the satisfaction of demands, whereas K_i is related to the production of new parts, which itself is related to the production capacity of the system. Property 12 precisely expresses the fact that the production capacity of the EKCS only depends on K_i and not on $S_i, i = 1, \ldots, N$. With this in mind, a reasonable design procedure for the EKCS could be to first design parameters K_i to obtain a desirable production capacity level, and subsequently design parameters S_i to obtain a desirable customer satisfaction level. Property 12 is not true for the GKCS, i.e., the production capacity of the GKCS depends on both K_i and $S_i, i = 1, \ldots, N$. This constitutes an important disadvantage of the GKCS relative to the EKCS.

7. Broader definition of the EKCS

The main goal of this paper was to introduce and analyze the EKCS in its basic form; therefore, this paper only focused on the essential elements of the EKCS. These elements are: (i) the control of the transfer of a finished part p_i downstream into MP_{i+1} (or to the customer, if $i = N$) parameterized by K_i and S_i, for each stage i; and (ii) the immediate transfer of a demand d_i to $D_i, i = 1, \ldots, N + 1$, upon its arrival to the system. We note, however, that one could generalize the way finished parts are transferred downstream of the system and the way demands are transferred upstream of the system by adding more parameters to the system. For instance, a policy for the transfer of finished parts p_i from PA_i into MP_{i+1} may be to transfer them in lots of size r_i, that is, transfer r_i finished parts p_i as soon as there are r_i pairs (p_i, a_i) in PA_i, r_i kanbans a_{i+1} in A_{i+1}, and r_i demands d_{i+1} in D_{i+1}. In addition, for each stage there may be a delay τ_i between the arrival of a customer demand to the system and its transfer to queue D_i. Adding these two sets of parameters to the EKCS permits a broader definition of the EKCS to a similar extent to which the PAC system [2,11] is a broader definition of the GKCS [8,10]. Let us informally call this broader definition of the EKCS, "BEKCS".

With this in mind, the BEKCS, like the PAC system, can be specialized into a variety of classical coordination approaches through the appropriate choice of parameters (see, Buzacott and Shanthikumar [2] for special cases of the PAC system). Notably, a BEKCS in which $S_i \geq 0, K_i = \infty, r_i \geq 1, \tau_i \geq 0, i = 1, \ldots, N$, is equivalent to an MRP-controlled system, where τ_i is a delay in the transfer of a demand for the production of a new stage-i finished part determined by the lead times used in the MRP calculations. Similarly, a BEKCS in which $S_i \geq 0, K_i = \infty, r_i \geq 1, \tau_i = 0, i = 1, \ldots, N$, may be used to implement "s,S" type policies in a BSCS setting, in which production of new finished parts is ordered in lots of one or more parts.

8. Conclusions

The popularity and success of the KCS for coordinating multi-stage manufacturing systems has inspired the search for improved kanban-type policies. The GKCS (or PAC) is a significant improvement over the KCS as it loosens the coupling between the transfer of demands to produce new finished parts upstream of the line and the

transfer of finished parts downstream of the line, present in the KCS. The GKCS depends on two parameters for each stage i, the number of kanbans, K_i, and the base stock of finished parts, S_i. The production capacity of the GKCS depends on both these parameters. The EKCS, presented in this paper, like the GKCS, depends on the same two parameters, K_i and S_i, for each stage i, and includes the KCS and BSCS as special cases. In the EKCS customer demands are transferred to all stages immediately upon their arrival, as is the case in the BSCS. Unlike in the BSCS, however, in the EKCS parts are transferred downstream only if one of a finite number of kanbans is available, as is the case in the KCS. A consequence of this mechanism is that production capacity in the EKCS only depends on parameters K_i and does not depend on parameters S_i. The response time of the EKCS to a customer demand is shorter than that of the GKCS with the same system parameters. This is, however, at the expense of higher bounds on the number of finished parts in the system. The operation of the EKCS is simpler than that of the GKCS since the EKCS has only one synchronization station per stage, whereas the GKCS has two. This is very important from a practical point of view. Another advantage of the EKCS over the GKCS, from a practical point of view, is that in the EKCS all parts always have kanbans attached to them, so they are easy to identify, whereas in the GKCS in-process parts have kanbans attached onto them, but finished parts do not.

This paper introduced the EKCS and some of its properties. An important task that remains to be done is to develop a method to optimize the parameters K_i and S_i and eventually compare the performance of the optimized EKCS with the performance of the optimized GKCS. The simplest way to do the optimization of the EKCS is via a gradient based optimization algorithm. For this one needs to develop a simulation algorithm or an analytical, probably approximate technique (for instance, along the lines of the technique presented in Di Mascolo *et al.* [21, 22]) to evaluate the performance of the EKCS given parameters K_i and S_i.

Acknowledgements

The authors are grateful to the anonymous referee for his/her suggestions, which improved the presentation and clarity of the paper. This work was partly supported by the Human Capital and Mobility Program of the European Union under contract No. ERBCHBICT940938.

References

[1] Gershwin, S.B. (1994) *Manufacturing Systems Engineering*, Prentice-Hall, Englewood Cliffs, NJ.

[2] Buzacott, J.A. and Shanthikumar, J.G. (1993) *Stochastic Models of Manufacturing Systems*, Prentice-Hall, Englewood Cliffs, NJ.

[3] Hopp, W.J. and Spearman, M.L. (1996) *Factory Physics*, McGraw-Hill, New York, NY.

[4] Veatch, M.H. and Wein, L.M. (1994) Optimal control of a two-station tandem production/inventory system. *Operations Research*, **42**(2), 337–350.

[5] Liberopoulos, G. and Dallery, Y. (1998) A unified framework for pull control mechanisms in multi-stage manufacturing systems. *Annals of Operations Research*, *Volume on Performance Evaluation of Production Lines*.

[6] Zipkin, P. (1991) Does manufacturing need a JIT revolution? *Harvard Business Review*, January–February, 40–50.

[7] Groenvelt, H. (1993) The just-in-time system, in *Handbooks in Operations Research and Management Science*, Vol. 4: *Logistics of Production and Inventory*, Graves, S.C., Rinnooy Kan, A.H.G. and Zipkin, P.H. (eds), Elsevier Science, (North Holland), Amsterdam, pp. 629–670.

[8] Buzacott, J.A. (1989) Queueing models of kanban and MRP controlled production systems. *Engineering Cost and Production Economics*, **17**, 3–20.

[9] Zipkin, P. (1989) A kanban-like production control system: analysis of simple models. Research Working Paper No. 89–1, Graduate School of Business, Columbia University, New York, NY 10027.

[10] Frein, Y., Di Mascolo, M. and Dallery, Y. (1995) On the design of generalized kanban control systems. *International Journal of Operations and Production Management*, **15**(9), 158–184.

[11] Buzacott, J.A. and Shanthikumar, J.G. (1992) A general approach for coordinating production in multiple-cell manufacturing systems. *Production and Operations Management*, **1**(1), 34–52.

[12] Karaesmen, F. and Dallery, Y. (1999) A performance comparison of pull control mechanisms for multi-stage manufacturing systems. *International Journal of Production Economics*, **63**(1) (in press).

[13] Cheng, D.Y. and Yao, D.D. (1993) Tandem queues with general blocking: a unified model and comparisons results. *Journal of Discrete Event Dynamic Systems: Theory and Applications*, **2**, 207–234.

[14] Di Mascolo, M. and Dallery, Y. (1996) Performance evaluation of kanban-controlled assembly systems, in *Proceedings of the Symposium of Discrete Events and Manufacturing Systems of the Multiconference on Computational Engineering in Systems Applications (CESA96 IMACS)*, Lille, France.

[15] Chaouiya, C., Liberopoulos, G. and Dallery, Y. (1998) The extended kanban system for production control of assembly systems. Technical Report LIP6 1998/024, Université Pierre et Marie Curie, Paris.

[16] Sbiti, N., Di Mascolo, M. and Amghar, M. (1999) Analysis of Base Stock Assembly Systems, in *Proceedings of the 2nd Aegean Conference on Analysis and Modeling of Manufacturing Systems*, Tinos Island, Greece, Editions Ziti, Thesaloniki, Greece. pp. 141–151.

[17] Baynat, B., Buzacott, J.A. and Dallery, Y. (1999) Multi-product kanban control systems. Manuscript in preparation, LIP6, Université Pierre et Marie Curie, Paris.

[18] Dallery, Y., Liu, Z. and Towsley, D. (1994) Equivalence, reversibility, symmetry and concavity properties in fork-join queueing networks with blocking. *Journal of the Association for Computing Machinery*, **41**(5), 903–942.

[19] Dallery, Y., Liu, Z. and Towsley, D. (1997) Properties of fork/join queueing networks with blocking under various operating mechanisms. *IEEE Transactions on Robotics and Automation*, **13**(4), 503–518.

[20] Baccelli, F. and Liu, Z. (1992) Comparison properties of stochastic decision free nets. *IEEE Transactions on Automatic Control*, **37**(12), 1905–1920.

[21] Di Mascolo, M., Frein, Y., Baynat, B. and Dallery, Y. (1993) Queueing network modeling and analysis of generalized kanban

systems, in *Proceedings of the Second European Control Conference (ECC93)*, Groningen, The Netherlands. pp. 170–175.

[22] Di Mascolo, M., Frein, Y. and Dallery, Y. (1996) An analytical method for performance evaluation of kanban controlled production systems. *Operations Research*, **44**(1), 50–64.

Appendices

Appendix A. Review of the BSCS, KCS, and GKCS Below, we review the BSCS, KCS, and GKCS.

Appendix A1. Base stock control system

A simple pull control mechanism for coordinating multi-stage manufacturing systems is the Base Stock Control System (BSCS) [2] where the term "base stock" is borrowed from Inventory Control Theory. Figure A1 shows the queueing network model of a BSCS having N stages in series. The contents and initial value of each queue, or, in the case of an MP_i, network of queues, are listed in Table A1.

Queues P_i and D_{i+1} in linkage station $J_{i,i+1}$, $i = 0, \ldots, N$, are joined in a synchronization station. If there is a finished part (or raw part, if $i = 0$) p_i in P_i and a demand d_{i+1} in D_{i+1}, then: (i) the finished part p_i is transferred downstream into MP_{i+1} (or to the customer, if $i = N$), where it is relabeled q_{i+1}; and (ii) the demand d_{i+1} is satisfied and is therefore dropped. When the part q_{i+1} finishes its processing in MP_{i+1}, it is relabeled p_{i+1} and joins queue P_{i+1}. In the BSCS no authorization is needed in order to transfer a part p_i downstream of the line. Another way of viewing this is that in the BSCS there is an infinite number of authorizations $a_i, i = 1, \ldots, N$, and therefore there is no need to represent them in the synchronization stations.

Production in the system is driven by customer demands. As was mentioned in Section 2, a customer demand is a vector $(d_1, \ldots, d_{N+1})$, denoted $\mathbf{d}_{N+1}$. When a customer demand vector $\mathbf{d}_{N+1}$ arrives at the system, then the following happens:

(i) The demand vector $\mathbf{d}_{N+1}$ is separated into its components, d_i, $i = 1, \ldots, N+1$, and each component, d_i is immediately transferred upstream to its respective queue D_i.

(ii) If there is a finished part (or a raw part, if $i = 1$) p_{i-1} in P_{i-1}, $i = 1, \ldots, N+1$, this part is transferred downstream to MP_i (or to the customer, if $i = N+1$), after being relabeled q_i, and the demand d_i is satisfied. If there is no finished part p_{i-1} in P_{i-1}, the demand d_i stays in D_i as a backordered demand until a finished part p_{i-1} becomes available in P_{i-1}.

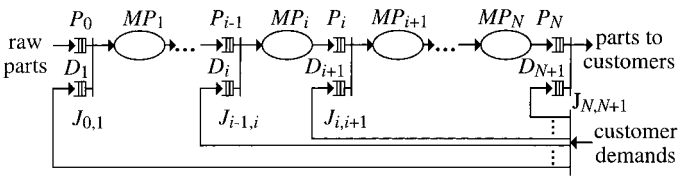

Fig. A1. Queueing network model of a BSCS having N stages in series.

Table A1. Contents and initial values in the queueing network model of a BSCS having N stages in series

	Queue	*Contents*	*Initial value*
MP_i	$i = 1,\ldots,N$	q_i	0
P_i	$i = 1,\ldots,N$	p_i	S_i
D_i	$i = 1,\ldots,N+1$	d_i	0

Initially, queue P_i, $i = 1, \ldots, N$, has S_i finished parts p_i. Queue P_0 represents the raw parts buffer. The initial number of raw parts in P_0 and the arrival of new raw parts into P_0 fall outside the scope of the control mechanism and are considered as given. In Inventory Control Theory S_i, $i = 1, \ldots, N$, is referred to as the *base stock* of stage i, hence the name BSCS. A BSCS with $S_i = 0$, $i = 1, \ldots, N$, is equivalent to a make to order rather than a make to stock system [2]. A make to order system is a system in which production of a new part is initiated to meet a given customer demand, rather than to replenish an inventory of parts depleted by customer demands, as is the case in a make to stock system.

The BSCS is a simple control mechanism that depends only on one parameter per stage, namely S_i, $i = 1, \ldots, N$. S_i influences the transfer of parts downstream of the system. A demand d_i, on the other hand, is transferred upstream to stage i immediately upon its arrival to the system and, therefore, independently of the transfer of parts downstream the system and of S_i. It can be shown that, since the transfer of demands upstream does not depend on S_i, the production capacity of the system also does not depend on S_i. Actually, the production capacity of the BSCS is given by the production capacity of the stage that has the lowest production capacity in isolation among all stages $i = 1, \ldots, N$. In Frein *et al.* [10] it is shown that $M(P_i) \leq S_i$, $i = 1, \ldots, N$, i.e., that the number of finished parts in P_i is bounded by S_i. The WIP in MP_i, however, may grow with no bound, since each demand d_i that arrives to the system triggers the release into MP_i of a finished part p_{i-1} in P_{i-1} (or a raw part, if $i = 1$), provided that such a part is available.

Appendix A2. The kanban control system

The most well-known pull control mechanism for coordinating multi-stage manufacturing systems is the Kanban Control System (KCS). Figure A2 shows the queueing network model of a KCS having N stages in

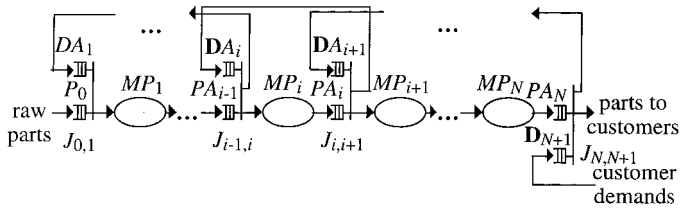

Fig. A2. Queueing network model of a KCS having N stages in series.

series. The contents and initial value of each queue, or, in the case of MP_i, network of queues, are listed in Table A2.

In the KCS each stage i has K_i authorization cards a_i that authorize the production of new stage-i finished parts. These authorizations are called kanbans. Initially, all the authorizations a_i in stage i are attached onto an equal number of finished parts p_i and are stored as pairs (p_i, a_i) in PA_i, $i = 1, \ldots, N$; therefore, K_i is also the base stock of finished parts p_i in PA_i. As is the case in the BSCS, queue P_0 represents the raw parts buffer. The initial number of raw parts in P_0 and the arrival of new raw parts into P_0 fall outside the scope of the control mechanism and are considered as given.

Queues PA_i and $\mathbf{D}A_{i+1}$ in linkage station $J_{i,i+1}$, $i = 1, \ldots, N-1$, are linked in a synchronization station. If there is a pair (p_i, a_i) in PA_i and a pair $(\mathbf{d}_{i+1}, a_{i+1})$ in $\mathbf{D}A_{i+1}$, then: (i) the kanban a_i is detached from p_i; (ii) the kanban a_{i+1} is attached onto p_i, which is relabeled q_{i+1}, and together they are transferred downstream into MP_{i+1}, as a pair (q_{i+1}, a_{i+1}); (iii) the component d_{i+1} of the demand vector $\mathbf{d}_{i+1}$ is satisfied and is therefore dropped; and (iv) the kanban a_i is attached onto the remaining demand vector $\mathbf{d}_i$, and together they are transferred upstream into $\mathbf{D}A_i$ as a pair $(\mathbf{d}_i, a_i)$. When the part q_{i+1} finishes its processing in MP_{i+1}, it is relabeled p_{i+1} and together with the kanban a_{i+1} that is attached onto it, they join queue PA_{i+1} as a pair (p_{i+1}, a_{i+1}).

In the last stage there is no need for an authorization a_{N+1} to release a finished part p_N to the customer; therefore, as soon as there is a pair (p_N, a_N) in PA_N and a demand vector $\mathbf{d}_{N+1}$ in $\mathbf{D}_{N+1}$, then: (i) the kanban a_N is detached from p_N; (ii) p_N is released to the customer; (iii) the component d_{N+1} of the demand vector $\mathbf{d}_{N+1}$ is satisfied and is therefore dropped; and (iv) the kanban a_N is attached onto the remaining demand vector $\mathbf{d}_N$, and together they are transferred upstream into $\mathbf{D}A_N$.

Also, in the first stage, raw parts do not have any kanbans attached to them; therefore, as soon as there is a raw part in P_0 and a pair (d_1, a_1) in DA_1, then: (i) the kanban a_1 is attached onto the raw part, which is labeled q_1, and together they are transferred downstream into MP_1, as a pair (q_1, a_1); and (ii) the demand d_1 is satisfied and is therefore dropped.

Production in the system is driven by customer demands. When a customer demand vector $\mathbf{d}_{N+1}$ arrives to the system, then the following happens:

(i) The demand vector $\mathbf{d}_{N+1}$ is immediately transferred upstream to queue $\mathbf{D}_{N+1}$.

(ii) If there is a pair (p_N, a_N) in PA_N, the kanban a_N is detached from p_N, the finished part p_N is transferred downstream to the customer, and the component d_{N+1} of vector $\mathbf{d}_{N+1}$ is satisfied. The kanban a_N is attached to the remaining demand vector $\mathbf{d}_N$, and together they are transferred upstream into $\mathbf{D}A_N$.

(iii) If there are no pairs (p_N, a_N) in PA_N, the customer demand vector $\mathbf{d}_{N+1}$ stays in $\mathbf{D}_{N+1}$ as a backordered demand until a pair (p_N, a_N) becomes available in PA_N. Therefore, a demand $\mathbf{d}_N$ is not transferred upstream to $\mathbf{D}A_N$ until a finished part p_N in PA_N is transferred downstream to the customer. Similarly, a demand $\mathbf{d}_{N-1}$ is not transferred upstream to $\mathbf{D}A_{N-1}$ until a finished part p_N in PA_N is transferred downstream to the customer and a part p_{N-1} in PA_{N-1} is transferred downstream into MP_N. More generally, a demand $\mathbf{d}_i$ is not transferred upstream to $\mathbf{D}A_i$ until a finished part p_N in PA_N is transferred downstream to a customer, a finished part p_{N-1} in PA_{N-1} is transferred into $MP_N, \ldots$, and a finished part p_i in PA_i is transferred downstream into MP_{i+1}. In Frein *et al.* [10] it is demonstrated that a KCS with a single stage is equivalent to a CONWIP system [3], in which an authorization to release a new raw part into the system is given as soon as a finished product is transferred to a customer.

The KCS is a simple control mechanism that depends only on one parameter per stage, namely K_i, $i = 1, \ldots, N$. K_i influences the transfer of finished parts p_i downstream of the system and the transfer of demands d_i upstream of the system. In fact, the transfer of a finished part p_i from PA_i into MP_{i+1} (or to the customer, if $i = N$) is completely synchronized with the transfer of a demand vector $\mathbf{d}_i$ from $\mathbf{D}A_{i+1}$ (or $\mathbf{D}_{N+1}$, if $i = N$) into $\mathbf{D}A_i$. This synchronization takes place when the kanban a_i is detached from p_i, at the moment p_i is transferred downstream, and is attached onto demand vector $\mathbf{d}_i$, authorizing the transfer of $\mathbf{d}_i$ upstream. Since the transfer of demands upstream depends on K_i, it can be shown

Table A2. Contents and initial values of queues in the queueing network model of a KCS having N stages in series

	Queue	*Contents*	*Initial value*
MP_i	$i = 1,\ldots,N$	(q_i, a_i)	0
PA_i	$i = 1,\ldots,N$	(p_i, a_i)	K_i
$\mathbf{D}A_i$	$i = 1,\ldots,N$	$(\mathbf{d}_i, a_i)$	0
$\mathbf{D}_{N+1}$		$\mathbf{d}_{N+1}$	0

that the production capacity of the system depends on K_i (see Property 13 in Section 6). In Frein *et al.* [10] it is shown that $M(MP_i) + M(PA_i) + M(\mathbf{D}A_i) = K_i, i = 1, \ldots, N$. This implies that the WIP and the number of finished parts in stage i are bounded by K_i.

The view that queue $\mathbf{D}A_{i+1}$ contains pairs of kanbans a_i and demand vectors $\mathbf{d}_{i+1}$ was employed in the interest of homogenizing the presentation of all the control systems discussed in this paper. According to this view, when a finished part p_i is being transferred downstream of stage i, the component d_{i+1} of a vector $\mathbf{d}_{i+1}$ in $\mathbf{D}A_{i+1}$ is satisfied and therefore dropped, and the remaining demand vector $\mathbf{d}_i$ is being transferred upstream of stage i. It should be noted, however, that a more widely encountered view of a KCS is one where, queue $\mathbf{D}A_{i+1}$ contains pairs of kanbans a_i and demands d_{i+1}. According to this view, when a finished part p_i is transferred downstream of stage i, a demand d_{i+1} in $\mathbf{D}A_{i+1}$ is satisfied and therefore dropped, and a *new* demand d_i is *created* and transferred upstream of stage i.

A3. The generalized kanban control system

The KCS has been generalized into a pull control mechanism called the Generalized Kanban Control System (GKCS) [8,9]. Figure A3 shows the queueing network model of a GKCS having N stages in series. The contents and initial value of each queue, or, in the case of MP_i, network of queues, are listed in Table A3.

In the GKCS each stage i has K_i kanbans a_i that authorize the production of new stage-i finished parts. Initially, all the kanbans a_i in stage i are stored in queue A_i. Initially, also, queue P_i, $i = 1, \ldots, N$, has S_i finished parts p_i. As in the previous two systems, queue P_0 represents the raw parts buffer. The initial number of raw parts in P_0 and the arrival of new raw parts into P_0 fall outside the scope of the control mechanism and are considered as given.

In the GKCS linkage station $J_{i,i+1}$, $i = 1, \ldots, N$, has two synchronization stations; $J_{0,1}$ has only one synchronization station. Queues P_i and DA_{i+1} in linkage station $J_{i,i+1}$, $i = 0, \ldots, N-1$, are linked in a synchronization station. If there is a finished part p_i in P_i and a pair (d_{i+1}, a_{i+1}) in DA_{i+1}, then: (i) the kanban a_{i+1} is attached

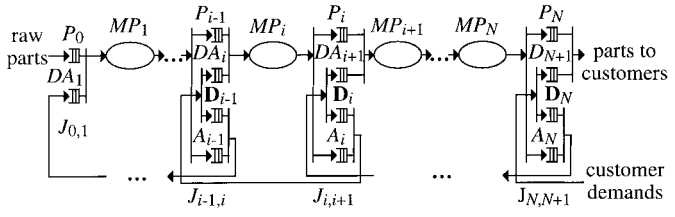

Fig. A3. Queueing network model of a GKCS having N stages in series.

Table A3. Contents and initial values of queues in the queueing network model of a GKCS having N stages in series

Queue		*Contents*	*Initial value*
MP_i	$i = 1,\ldots,N$	(q_i, a_i)	0
P_i	$i = 1,\ldots,N$	p_i	S_i
A_i	$i = 1,\ldots,N$	a_i	K_i
DA_i	$i = 1,\ldots,N$	(a_i, d_i)	0
D_{N+1}		d_{N+1}	0
$\mathbf{D}_i$	$i = 1,\ldots,N$	$\mathbf{d}_i$	0

onto p_i, which is relabeled q_{i+1}, and together they are transferred downstream into MP_{i+1}, as a pair (q_{i+1}, a_{i+1}); and (ii) the demand d_{i+1} is satisfied and is therefore dropped. When a part q_{i+1}, $i = 0, \ldots, N-1$, finishes its processing in MP_{i+1}, it releases the kanban a_{i+1} that was attached onto it. The part q_{i+1} is relabeled p_{i+1} and joins queue P_{i+1}, and the kanban a_{i+1} joins queue A_{i+1}.

Queues $\mathbf{D}_i$ and A_i in linkage station $J_{i,i+1}, i = 1, \ldots, N$, are also linked into a synchronization station. If there is a demand vector $\mathbf{d}_i$ in $\mathbf{D}_i$ and a kanban a_i in A_i, they are attached to each other and together they are transferred upstream. There, they are separated into a pair (d_i, a_i), which joins DA_i, and a vector $\mathbf{d}_{i-1}$, which joins $\mathbf{D}_{i-1}$.

As is the case in the KCS, in the last stage there is no need for an authorization a_{N+1} to release a finished part p_N to the customer. Also, in the first stage, if there is a demand vector $\mathbf{d}_1$ in $\mathbf{D}_1$ and a kanban a_1 in A_1, they are attached to each other and together they are transferred upstream to DA_1.

The GKCS is a generalization of the KCS in which: (i) queue PA_i, $i = 1, \ldots, N$, in the KCS, is separated into queues P_i and A_i, in the GKCS; (ii) queue $\mathbf{D}A_i$, $i = 1, \ldots, N$, in the KCS, is separated into queues DA_i and $\mathbf{D}_{i-1}$, $i = 2, \ldots, N$, in the GKCS; and (iii) queue $\mathbf{D}_{N+1}$ in the KCS is separated into queues D_{N+1} and $\mathbf{D}_N$, in the GKCS. This separation breaks the synchronization of the transfer of a finished part p_i downstream and the transfer of a demand d_i upstream of the system, which in the KCS are completely synchronized.

In Buzacott and Shanthikumar [2] it is shown that the GKCS with $K_i = S_i$, $i = 1, \ldots, N$, is equivalent to the KCS having K_i kanbans. This is because when $K_i = S_i$, $i = 1, \ldots, N$, in the GKCS, the two synchronization stations in $J_{i,i+1}$, $i = 1, \ldots, N$, "fire" at the same time. It is also shown that the GKCS with $K_i = \infty$, $S_i \geq 0$, $i = 1, \ldots, N$, is equivalent to the BSCS having the same S_i values as those of the GKCS. This is because when $K_i = \infty$, $i = 1, \ldots, N$, in the GKCS, the demands d_i are immediately transferred to stage-i upon their arrival to the system.

The GKCS is a more complex control mechanism than either the KCS or the BSCS, which, as was already mentioned, are special cases of the GKCS. First, because the GKCS depends on two parameters per stage, namely

S_i and K_i, $i = 1, \ldots, N$, and second because the GKCS has two synchronization stations in every stage. It is shown in Frein *et al.* [10] that in the GKCS, $M(DA_i) + M(MP_i) + M(A_i) = K_i$ and $M(P_i) - M(DA_{i+1}) + M(\mathbf{D}_i) - M(A_i) = S_i - K_i$. These two invariants imply that the WIP and the number of finished parts in stage i are bounded by K_i and S_i, respectively. In Frein *et al.* [10] it is also shown that a GKCS in which $K_i > K_{i+1} + S_i$, for some stage i, is equivalent to a GKCS with $K_i = K_{i+1} + S_i$ for that same stage i. Therefore, one need only consider GKCSs for which $K_i \leq K_{i+1} + S_i$, for all stages i.

The advantage of the GKCS over the KCS is that in the GKCS the transfer of a demand d_i from $\mathbf{D}_i$ upstream to DA_i is not completely synchronized with the transfer of a finished part p_i from P_i downstream to MP_{i+1} (or to the customer, if $i = N$), as is the case in the KCS. In both the KCS and the GKCS these transfers are accomplished via the use of a kanban a_i. The advantage of the GKCS over the BSCS is that in the GKCS the WIP in each stage i is bounded by K_i, whereas it is unbounded in the BSCS. The production capacity of the GKCS depends on both K_i and S_i, since the transfer of demands upstream depends on both these parameters, whereas the production capacity of the BSCS is independent of S_i [10].

Appendix B. Proof of Property 10

The proof of Property 10 is similar to the proof of "Stochastic Monotonicity with Respect to the Initial Marking" in Baccelli and Liu [20]. It is by induction on (i, n).

(i) To prove inequalities (20) and (21), assume that these inequalities hold up to $n - 1$ (they hold for $n \leq 0$), i.e., assume that

$$D'_{(i-1,i),n-m} \leq D_{(i-1,i),n-m}, \quad i = 1, \ldots, N+1, \ m = 1, 2, \ldots, \tag{A1}$$

$$D'_{i,n-m} \leq D_{i,n-m}, \quad i = 1, \ldots, N, \ m = 1, 2, \ldots \ . \tag{A2}$$

Then, to prove inequality (20) it suffices to show that the terms on the right-hand side of Equation (18) in the EKCS with parameter K'_l are smaller than or equal to the equivalent terms in the EKCS with parameter K_l, where $K'_l > K_l$. Indeed: (i) $D_{d,n}$ is the same in both systems; (ii) $D'_{i-1,n-S'_{i-1}} \leq D_{i-1,n-S_{i-1}}$, $i = 1, \ldots, N$, by (A2), since $S'_{i-1} = S_{i-1}$, $i = 1, \ldots, N$; (iiia) $D'_{(i,i+1),n-(K'_i-S'_i)} \leq D_{(i,i+1),n-(K_i-S_i)}$, $i \neq l$, by (A1), since $K'_i - S'_i = K_i - S_i$, $i \neq l$; and (iiib) for $i = l$, $D'_{(l,l+1),n-(K'_l-S'_l)} \leq D'_{(l,l+1),n-(K_l-S_l)} \leq D_{(l,l+1),n-(K_l-S_l)}$, where the first inequality holds by (15), since $n - (K'_l - S'_l) < n - (K_l - S_l)$, and the second inequality holds by (A1).

Similarly, to prove inequality (21), it suffices to show that the terms inside the parentheses on the right hand side of Equation (17) for the EKCS with parameter K'_l are smaller than or equal to the equivalent terms for the EKCS with parameter K_l, where $K'_l > K_l$, since the terms $\sigma_{i,n}$ are the same in both systems. Indeed; (i) $D'_{i,n-1} \leq D_{i,n-1}$, $i = 1, \ldots, N$, by (A2); and (ii) $D'_{(i-1,i),n} \leq D_{(i-1,i),n}$, $i = 1, \ldots, N+1$, by (20).

(ii) To prove inequalities (22)–(25) assume that these inequalities hold up to $n - 1$ (they hold for $n \leq 0$), i.e., assume that

$$D'_{(i-1,i),n-m} \leq D_{(i-1,i),n-m}, \quad i = l+1, \ldots, N+1, \quad m = 1, 2, \ldots, \tag{A3}$$

$$D'_{i,n-m} \leq D_{i,n-m}, \quad i = l+1, \ldots, N+1, \ m = 1, 2, \ldots, \tag{A4}$$

$$D'_{(i-1,i),n-m} \leq D_{(i-1,i),n+(S'_l-S_l)-m}, \ i = 1, \ldots, l, \ m = 1, 2, \ldots, \tag{A5}$$

$$D'_{i,n-m} \leq D_{i,n+(S'_l-S_l)-m}, \quad i = 1, \ldots, l, \ m = 1, 2, \ldots \tag{A6}$$

Then, to prove inequality (22) it suffices to show that the terms on the right-hand side of Equation (18) in the EKCS with parameter S'_l are smaller than or equal to the equivalent terms in the EKCS with parameter S_l, for $i \geq l + 1$, where $S'_l > S_l$. Indeed: (i) $D_{d,n}$ is the same in both systems; (iia) $D'_{i-1,n-S'_{i-1}} \leq D_{i-1,n-S_{i-1}}$, for $i > l + 1$, by (A4), since $S'_{i-1} = S_{i-1}, i > l + 1$; (iib) for $i = l + 1$, $D'_{l,n-S'_l} \leq D'_{l,n-S_l} \leq D_{l,n-S_l}$, where the first inequality holds by (15), since $n - S'_l < n - S_l$, and the second inequality holds by (A4); and (iii) $D'_{(i,i+1),n-(K'_i-S'_i)} \leq D_{(i,i+1),n-(K_i-S_i)}$, $i \geq l + 1$, by (A3), since $K'_i - S'_i = K_i - S_i$, $i \geq l + 1$.

Similarly, to prove inequality (23), it suffices to show that the terms inside the parentheses on the right hand side of Equation (17) in the EKCS with parameter S'_l are smaller than or equal to the equivalent terms in the EKCS with parameter S_l, for $i \geq l + 1$, where $S'_l > S_l$, since the terms $\sigma_{i,n}$ are the same in both systems. Indeed: (i) $D'_{i,n-1} \leq D_{i,n-1}$, $i \geq l + 1$, by (A4); and (ii) $D'_{(i-1,i),n} \leq D_{(i-1,i),n}$, $i \geq l + 1$, by (22).

Moreover, to prove inequality (24) it suffices to show that the terms on the right hand side of Equation (18) in the EKCS with parameter S'_l are smaller than or equal to the equivalent terms – shifted by $S'_l - S_l$ – in the EKCS with parameter S_l, for $i \leq l$, where $S'_l > S_l$. Indeed: (i) $D_{d,n}$ in the EKCS with parameter S'_l is smaller than or equal to $D_{d,n+(S'_l-S_l)}$, by (15); (ii) $D'_{i-1,n-S'_{i-1}} \leq D_{i-1,n+(S'_l-S_l)-S_{i-1}}$, $i \leq l$, by (A5), since $S'_{i-1} = S_{i-1}$, $i \leq l$; (iiia) $D'_{(i,i+1),n-(K'_i-S'_i)} \leq D_{(i,i+1),n+(S'_l-S_l)-(K_i-S_i)}$, $i < l$, by (A5), since $K'_i - S'_i = K_i - S_i$, $i < l$; and (iiib) for $i = l$, $D'_{(l,l+1),n-(K'_l-S'_l)} \leq D_{(l,l+1),n-(K_l-S'_l)} = D_{(l,l+1),n+(S'_l-S_l)-(K_l-S_l)}$, where the inequality holds by (A3), since $K'_l = K_l$.

Finally, to prove inequality (25), it suffices to show that the terms inside the parentheses on the right hand side of Equation (17) in the EKCS with parameter S'_l are smaller

than or equal to the equivalent terms – shifted by $S'_l - S_l$ – in the EKCS with parameter S_l, for $i \leq l$, where $S'_l > S_l$. Indeed: (i) $D'_{i,n-1} \leq D_{i,n+(S'_l-S_l)-1},\ i \leq l$, by (A6); and (ii) $D'_{(i-1,i),n} \leq D_{(i-1,i)n+(S'_l-S_l)}$, for $i \leq l$, by (24).

Appendix C. Comparison between the EKCS and GKCS

A hasty, visual comparison between the queueing network model of the EKCS shown in Fig. 2 and the queueing network model of the GKCS shown in Fig. A3 may lead to the conjecture that the EKCS responds faster to customer demands than does the GKCS, given that the two systems have the same parameters. This is because in the EKCS a demand d_i is transferred upstream to D_i immediately upon its arrival to the system, whereas in the GKCS a demand d_i is transferred from $\mathbf{D}_N$ to DA_i in $N - i + 1$ steps after the transfer of a kanban a_N from A_N into DA_N, a kanban a_{N-1} from A_{N-1} into $DA_{N-1}, \ldots$, a kanban a_i from A_i into DA_i.

The comparison between the two systems is of course valid only for parameter values that satisfy inequality (1). For parameter values that do not satisfy inequality (1) the EKCS is not defined. It should be noted however that a GKCS with $S_i > K_i,\ i = 1, \ldots, N$, becomes a very restrictive system known as the "Local Control" system [2], in which a finished part p_i is released into MP_{i+1} depending only on the availability of space in MP_{i+1} and P_{i+1}.

What is a little puzzling, when comparing the EKCS and the GKCS, is that in the two special cases, when $K_i = \infty$ and when $K_i = S_i,\ i = 1, \ldots, N$, both the EKCS and the GKCS are equivalent to the BSCS and the KCS, respectively, and are therefore equivalent to each other. This raises the question: what is the difference between the EKCS and GKCS? To answer this question we compare the evolution equations relating equivalent event times in the two systems.

To distinguish equivalent event times in the two systems, we denote by $D^E_{(i-1,i),n}$ the departure time of the nth pair (p_{i-1}, a_{i-1}) from the synchronization station in $J_{i-1,i},\ i = 1, \ldots, N+1$, and by $D^E_{i,n}$ the departure time of the nth pair (q_i, a_i) from $MP_i,\ i = 1, \ldots, N$, in the EKCS. The evolution equations relating $D^E_{(i-1,i),n}$ and $D^E_{i,n}$ are therefore given by Proposition 1, where $D_{(i-1,i),n}$ and $D_{i,n}$ are replaced by $D^E_{(i-1,i),n}$ and $D^E_{i,n}$, respectively. Similarly, we denote by $D^G_{(i-1,i),n}$ and $D^G_{i,n}$ the respective event times in the GKCS, i.e., $D^G_{(i-1,i),n},\ i = 1, \ldots, N+1$, is the departure time of the nth part p_i from the top synchronization station in $J_{i-1,i}$ and $D^G_{i,n}$ is the departure time of the nth pair (q_i, a_i) from $MP_i,\ i = 1, \ldots, N$. In addition, in the GKCS we denote by $D^{G*}_{(i-1,i),n},\ i = 2, \ldots, N+1$, the departure time of the nth kanban a_{i-1} from the bottom synchronization station in $J_{i-1,i}$, i.e., the time of the nth arrival of a pair (d_{i-1}, a_{i-1}) in DA_{i-1} and a vector $\mathbf{d}_{i-2}$ in $\mathbf{D}_{i-2}$ (or no vector, if $i = 2$). These times are shown in Fig. A4.

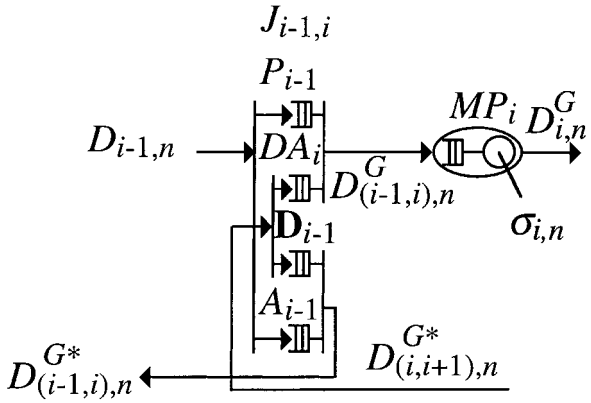

Fig. A4. Times of events at the ith stage of an GKCS having N stages in series.

The evolution equations relating $D^G_{(i-1,i),n}, D^G_{i,n}$, and $D^{G*}_{(i-1,i),n}$ are given in the following proposition (see Buzacott and Shanthikumar [2] for similar expressions).

Proposition 2. *In the GKCS in which MP_i consists of a single machine, the timings of events are related by the following evolution equations*:

$$D^G_{i,n} = \sigma_{i,n} + \max(D^G_{i,n-1},\ D^G_{(i-1,i),n}), \quad i = 1, \ldots, N, \tag{A7}$$

$$D^{G*}_{(i-1,i),n} = \max(D^G_{i-1,n-K_{i-1}},\ D^{G*}_{(i,i+1),n}), \quad i = 2, \ldots, N+1, \tag{A8}$$

$$D^G_{(i-1,i),n} = \max(D^G_{i-1,n-S_{i-1}},\ D^{G*}_{(i,i+1),n}), \quad i = 1, \ldots, N+1, \tag{A9}$$

where, *by convention, the maximum over an empty set is* $-\infty$, *and* $D^{G*}_{(N+1,N+2),n} = D_{d,n}$.

Proof. Equation (A7) can be explained in a similar way as Equation (17) in the EKCS. Equations (A8) and (A9) can be explained in a similar way as Equation (18) in the EKCS. ■

Equation (A8) is recursive in that it expresses $D^{G*}_{(i-1,i),n}$ in terms of $D^{G*}_{(i,i+1),n}$. Expanding this recursion, starting from $i = N+1$, yields

$$D^{G*}_{(i-1,i),n} = \max\left[D_{d,n}, \max_{j=i-1}^{N}(D^G_{j,n-K_j})\right], \quad i = 2, \ldots, N+2,$$

which, alternatively, can be written as

$$D^{G*}_{(i,i+1),n} = \max\left[D_{d,n}, \max_{j=i}^{N}(D^G_{j,n-K_j})\right], \quad i = 1, \ldots, N+1. \tag{A10}$$

Substituting $D^{G*}_{(i,i+1),n}$ from (A10) into (A9) yields

$$D^G_{(i-1,i),n} = \max\left[D_{d,n}, D^G_{i-1,n-S_{i-1}}, \max_{j=i}^{N}(D^G_{j,n-K_j})\right], \quad i = 1,\ldots,N+1. \tag{A11}$$

Equation (A11) in the GKCS is the equivalent to Equation (19) in the EKCS. Both equations relate the release times of parts into MP_i (or to the customer, if $i = N+1$) to the arrival times of customer demands to the system and the departure times of parts from MP_j, $j = i-1,\ldots,N$.

Comparing the evolution equations of the EKCS and the GKCS, given be Propositions 1 and 2, respectively, leads to the following property.

Property 14. *Consider two systems, the EKCS and the GKCS, having the same parameters K_i and S_i, $i = 1,\ldots,N$, the same sequence of service times, $\sigma_{i,n}$, $i = 1,\ldots,N$, $n = 1,2,\ldots$, and the same sequence of customer demand and raw part arrival times, $D_{d,n}$ and $D_{0,n}$, $n = 1,2,\ldots$, respectively. Then, the following holds*:

$$D^E_{(i-1,i),n} \le D^G_{(i-1,i),n}, \quad i = 1,\ldots,N+1,\ n = 1,2,\ldots, \tag{A12}$$

$$D^E_{i,n} \le D^G_{i,n}, \quad i = 1,\ldots,N,\ n = 1,2,\ldots\ . \tag{A13}$$

Proof. The proof is similar to the proof of Property 10 in Appendix B. To prove inequalities (A12) and (A13), assume that these inequalities hold up to $n-1$ (they hold for $n \le 0$), i.e., assume that

$$D^E_{(i-1,i),n-m} \le D^G_{(i-1,i),n-m}, \quad i = 1,\ldots,N+1,\ m = 1,2,\ldots, \tag{A14}$$

$$D^E_{i,n-m} \le D^G_{i,n-m}, \quad i = 1,\ldots,N,\ m = 1,2,\ldots\ . \tag{A15}$$

Then, to prove inequality (A12) it suffices to show that the terms on the right hand side of Equation (19) for the EKCS are smaller than or equal to the equivalent terms on the right hand side of Equation (A11) for the GKCS. Indeed: (i) $D_{d,n}$ is the same in both systems; (ii) $D^E_{i-1,n-S_{i-1}} \le D^G_{i-1,n-S_{i-1}}$, by (A15); and (iii) $D^E_{j,n-S_j-\sum_{m=i}^{j}(K_m-S_m)} \le D^E_{j,n-K_j} \le D^G_{j,n-K_j}$, where the first inequality holds by (15), since $n - S_j - \sum_{m=i}^{j}(K_m - S_m) \le n - K_j$, and the second inequality holds by (A15).

Similarly, to prove inequality (A13) it suffices to show that the terms on the right hand side of Equation (17) for the EKCS are smaller than or equal to the equivalent terms on the right hand side of Equation (A7) for the GKCS. Indeed: (i) $D^E_{i,n-1} \le D^G_{i,n-1}$, by (A15); and (ii) $D^E_{(i-1,i),n} \le D^G_{(i-1,i),n}$, by (A12). ■

Property 14 states that the departure time of the nth part p_{i-1} from the synchronization station in $J_{i-1,i}$ in the EKCS is smaller than the departure time of the nth part p_{i-1} from the top synchronization station in $J_{i-1,i}$ in the GKCS, given the same parameter values for the two systems. This means that demands are satisfied earlier in the EKCS than in the GKCS, but it does not necessarily also mean that the EKCS has an overall better performance than the GKCS, since inventory storage costs are not taken into account. In fact, the EKCS is likely to incur higher inventory storage costs than does the GKCS. This is because in the EKCS the bound on the number of pairs (p_i, a_i) in PA_i is higher than the bound on the number of finished parts p_i in P_i in the GKCS. Namely, in the EKCS the number of pairs (p_i, a_i) in PA_i is bounded by K_i (except in the last stage), whereas in the GKCS, the number of finished parts p_i in P_i is bounded by S_i, where $S_i \le K_i$, by (1). The WIP in MP_i as well as the WIP + number of finished parts in stage i, however, is bounded by K_i in both systems (see Equation (11) for the EKCS and Frein *et al.* [10] for the GKCS).

From Property 14, which states that parts move faster through the EKCS than through the GKCS when production is driven by demands, it can be shown that parts move faster through the EKCS than through the GKCS in the presence of infinite demands too. This then leads to the following property.

Property 15. *The production capacity of the EKCS with parameters K_i and S_i, $i = 1,\ldots,N$, is higher than the production capacity of the GKCS with the same parameters K_i and S_i.*

Further comparison between (19) and (A11) reveals the following property.

Property 16. *The EKCS with $K_i = S_i$ or $K_i = \infty$, $i = 1,\ldots,N-1$, is equivalent to the GKCS with the same parameters K_i and S_i.*

Proof. When $K_i = S_i$ or $K_i = \infty$, $i = 1,\ldots,N-1$, the evolution Equation (19) of the EKCS and (A11) of the GKCS are the same. The two systems are, therefore, equivalent. ■

Property 16 states that in order for the GKCS and the EKCS to be equivalent to each other it suffices that $K_i = S_i$, or $K_i = \infty$, for all but the last stage, i.e., for $i = 1,\ldots,N-1$. Of course, by Properties 8 and 9, the EKCS and the GKCS with $K_i = S_i$, and $K_i = \infty$, $i = 1,\ldots,N$, are both equivalent to the KCS and BSCS, respectively. In the case where there is only one stage, this stage *is* the last stage. In this case the two systems are equivalent no matter what parameters K_1 and S_1 are. This is stated as the following property.

Property 17. *The single-stage EKCS with parameters K_1 and S_1 is equivalent to the single-stage GKCS with the same parameters.*

Biographies

Yves Dallery received his Ph.D. and the degree of Habilitation Diriger des Recherches from the Institut National Polytechnique de Grenoble (INPG) in 1984 and 1989, respectively. He is currently Directeur de Recherche at the Centre National de la Recherche Scientifique (CNRS). In 1984–1985, he was a post-doctoral fellow at Harvard University. In 1991–1992, he was a visiting scientist at M.I.T. and in 1992–1993 he was an Associate Professor of Manufacturing Engineering at Boston University. His research interests are in stochastic models of discrete event systems, modeling, performance evaluation and control of manufacturing systems, and operations management.

George Liberopoulos received his B.S. and M.Eng. from Cornell University in 1985 and 1986, respectively, and his Ph.D. from Boston University in 1993. In 1994–96 he was a visiting scientist at Universite Paris IV. He is currently an Assistant Professor of Production Management at the University of Thessaly. His research interests are in the analysis and control of manufacturing systems. He serves in the Editorial Board of *IIE Transactions on Design and Manufacturing*.

Contributed by the Manufacturing Systems Control Department