Stability of Pull Production Control Methods for Systems with Significant Setups

Thomas I. Seidman* **Lawrence E. Holloway†**

IEEE-Trans. Auto. Control **47**, pp. 1637–1647, (2002).

Abstract

In manufacturing, a pull production control method is a method of authorizing production based on replenishing current and past consumption, as opposed to forecasts or orders for future consumption. In this paper, we consider special classes of pull systems for operations that involve significant setup times. In particular, we present a model for variations of the *Signal Kanban* method and the *Pattern Production* method, each of which is used in industry when conventional Kanban methods are inappropriate. This paper examines such systems under demands with unpredictable load content but an upper bound on the *total* load. It is shown that, under appropriate conditions, such systems are *stable* in the sense that cumulative production at any time trails cumulative demand by no more than a constant. We determine buffer parameters under each protocol (including reorder points in the Signal Kanban case) such that the backorder queue will clear to zero and remain empty thereafter. The results are then extended to consider multiple machines fulfilling production authorizations in parallel.

KEY WORDS: *Production control, Signal Kanban, Pattern Production, lean manufacturing, round robin.*

*Department of Mathematics and Statistics, University of Maryland Baltimore County, Baltimore, MD 21250, e-mail: ⟨seidman@math.umbc.edu⟩

†Center for Robotics and Manufacturing Systems University of Kentucky, Lexington, Kentucky 40506-0108 e-mail: ⟨holloway@engr.uky.edu⟩ This work has been supported in part by NSF Grant ECS-9807106, USARO Grant DAAH04-96-1-0399, and the University of Kentucky Center for Robotics and Manufacturing Systems.

1 Introduction

Pull production control is an important tool used by companies operating under "just-in-time" manufacturing or "lean manufacturing" [9, 16, 15]. In concept, the pull production control method constrains the buildup of inventory by only authorizing production based on actual downstream consumption of product. Thus, in pull systems, authorizations for production come from downstream stations (hence, with demand arriving at the end of the line), in contrast to "push" production control systems, where orders arrive at the start of the line based on forecasts or orders for future consumption [9].

Most work in the literature on pull production control focuses on single-card Kanban[1], two-card Kanban, and pull system variants such as CONWIP. A recent survey of Kanban methods is presented by Akturk and Erhun [1], and a comparison of these and other methods is given by Bonvik, Couch, and Gershwin [2].In these classic implementations of Kanban, there are multiple production authorizations associated with a given product, and each one is associated with a standard lot of product. Whenever such a lot begins to be consumed, the authorization associated with it is released to the producing system, authorizing production to replenish it. Inventory is directly capped by only having fixed numbers of such authorizations circulating in the system.

Such a system works well when the producing station is able to change over between products relatively quickly and easily. In systems with significant costs or time associated with setups, the classic Kanban methods are inappropriate. A balance must be reached between the need to limit the number of setups to avoid sacrificing capacity and the goals of small lot sizes, minimum inventories, and production smoothing which are emphasized in a lean manufacturing operation. Instead, two different kinds of pull control are commonly used. The first of these is the "Signal Kanban". In contrast to classical Kanban methods, a single production authorization (called a signal kanban) is associated with each product, and, in particular, with a specific level of inventory. When inventory falls below that level, the authorization is released to replenish the consumption. Signal Kanban production control is discussed by Monden [9]. In traditional inventory control theory, the Signal Kanban method is often called a *reorder point policy*, or a *two-bin system* [8].

[1]As is common in the literature, we use the capitalized term *Kanban* to indicate the production control method, and the non-capitalized term *kanban* to indicate a production authorization within the method.

Such systems are commonly used in industry for operations with significant setups or minimum batch sizes. In fact, our investigation of Signal Kanban systems was motivated by the production control systems formerly in use at the metal stamping line at Toyota Motor Manufacturing Kentucky (TMMK).

A second type of production control for use in systems with significant setup costs is *Pattern Production*. In this control method, there is a fixed production sequence (*pattern*) and whenever the system receives such an authorization, it produces up to a fixed level to replenish past consumption. When the replenishment is complete, the machines are changed over to replenish the next product in the sequence. There is no specified minimum batch size and no idle time for such a system. The time between repetitions of the cycle is variable, in contrast to the cyclic productions considered in Economic Lot Scheduling Problems [4].

A key point of the systems that we consider is that the inventory behavior for each product depends on the state of all other products which consume capacity of the production system. In the case of Signal Kanban systems, the lead time of a production authorization depends on the number of other different products which are also waiting to be replenished, and in some cases on the volume of each product that must be replenished for each. Thus, the lead time depends indirectly on the past demand for all of the products that rely on a given machine or set of machines. In the case of the Pattern Production system, the period between replenishments of a product depends on the run-lengths of all other products run in the intervening period, and so again is dependent indirectly on past demand for all other products.

Our focus in this paper is on the *stability* of such pull production control methods for systems with significant setups. A production system is said to be *stable* if all its buffers are bounded so cumulative production can trail cumulative demand by no more than a constant lag [10]. Certainly, traditional capacity constraints are needed to have any possibility of stability. However, examples are known of layouts and systems with such simple control policies as clearing [7] or first-come first-served [11] which satisfy those constraints but are nevertheless unstable. Seidman and Humes [14] have observed examples of instability in the context of a production kanban system with orders based on consumption but serviced according to a clearing policy. A principal distinction between that setting and our present concerns is the treatment of multiple machines supplying each other. This is outside the scope of this paper, restricted to a single focus (whether comprised of one machine or several), but is deferred to [13].

Our main result here is that, for appropriate selections of buffer parameters, we show that Signal Kanban policies and Pattern Production policies will be stable (the queue of backorders is bounded) and even strongly clearing (the backorder queue eventually becomes empty and remains there), subject to a capacity constraint and a limit on the burstiness of total demand. One important implication of this result is that demand among different products can be reallocated, as long as the burstiness bound on the *total* demand is not exceeded. We emphasize that the policies we consider determine production solely in response to demand, and no prior schedule or demand forecasts are assumed. Thus, the policies require no prior knowledge of the composition of the load or of the nature of the burstiness except that the limit is maintained.

In the next section, we describe our basic model of a production system. Section 3 shows stability of two common variations of Signal Kanban systems, fixed-fill and fixed-batch for a single machine operation. Section 4 presents results for pattern production control for a single machine. Section 5 considers these protocols operating with a bank of machines servicing production authorizations in parallel.

2 System Overview

The structure of the systems we consider is shown in figure 1. There are N products (each associated with a buffer$_n$ of size B_n). Demand (orders) arrive to the queue $\mathcal{Q}$. If there are no backlogs, then the order immediately proceeds to the head of $\mathcal{Q}$. The order at the head of $\mathcal{Q}$ is then filled with product from the corresponding buffer$_n$ if it is nonempty. Otherwise, the order remains as a backlog and also blocks subsequent orders until it can be filled.

Let $U_n(t)$ be the cumulative number of orders for product$_n$ arriving by time t — including any 'initial condition'. Thus, $U_n(0) = 0$ means that buffer$_n$ is initially full and, in addition, that $\mathcal{Q}$ contained no orders for product$_n$.

The n-th product has a 'relative difficulty' (in terms of production effort) of ρ_n. We use the relative difficulties to scale the load, setting $u_n(t) := \rho_n U_n(t)$ and then letting $\mathbf{u}(t)$ be the N-vector with entries $u_n(t)$. Note that for non-negative N-vectors, we may write $\|\mathbf{z}\| = \mathbf{1} \cdot \mathbf{z}$ where $\mathbf{1}$ is the N-vector of '1's and we are using the ℓ_1-norm so, e.g., $\|\mathbf{u}(t)\|$ is the total cumulative load: all orders (scaled) up to time t.

Figure 1: A production system with signal kanbans (triangles) determining the production of product by the machines. The release of the signal kanbans is determined by the consumption of parts from the buffers.

We will assume that we have a bound λ on the (scaled) arrival rate of orders so, with some allowance ξ for 'burstiness', we assume that, using this ℓ_1-norm,

$$\|\mathbf{u}(t) - \mathbf{u}(s)\| \leq \lambda \cdot (t - s) + \xi \tag{2.1}$$

for arbitrary intervals $(s, t]$; when $(s, t]$ is understood, we set

$$\tilde{\xi} := \|\mathbf{u}(t) - \mathbf{u}(s)\| - \lambda \cdot (t - s), \qquad \text{requiring } \tilde{\xi} \leq \xi. \tag{2.2}$$

Figure 2(a) shows that ξ represents a bound on the burstiness of the order over any interval. Note that it is quite possible to have $\tilde{\xi} < 0$ for a particular interval, but the condition that $\tilde{\xi} \leq \xi$ must hold for every subinterval. The

Figure 2: (a.) The burstiness restriction limits demand bursts above the nominal. (b.) Extended drops in demand shifts down the demand bound for all future time.

effect of this is shown in figure 2(b). Any extended drop below the nominal rate of λ results in a downward shift in the bound, so future bursts cannot make up the loss.

We emphasize that our treatment assumes a bound only on the *total* demand input, without any consideration for its composition. For this treatment, then, it is not required to have any information as to how the total demand is partitioned into demand for individual products — nor need that composition remain even roughly stationary over time. A separate concern as to how one might take advantage of such compositional information if it were available is adduced in our discussion, but will not be treated in any relevant detail.

Orders are filled with product_n from buffer_n, and those buffers are replenished by a set of machines according to a production policy to be discussed later. Production authorizations (kanbans) are collected first to a queue $\mathcal{K}$, and then released to any of M machines. Each machine has a speed factor σ^m (simply σ when $M = 1$). There is also an allowance δ_n for a (relative) setup delay (in terms of production units) needed in preparation for processing product_n. This is here taken to be independent[2] of the machine used and of

[2]Since δ_n is only a *bound* on the setup delay, this represents no loss of generality — although sharper estimates than we obtain might become available at the price of further

the prior product processed. Thus, the actual processing time for product$_n$ at machine^m is $\tau_n^m \leq \rho_n/\sigma^m$ units of time per item of product, excluding setup and when considering a batch of size k processed at machine^m, the total run time, including the setup, will be bounded by $(k\rho_n + \delta_n)/\sigma^m$.

Let $V_n(t)$ be the cumulative amount of product$_n$ released from the system by time t, so $v_n(t) := \rho_n V_n(t)$ is the scaled released production for product$_n$; let $\mathbf{v}(t)$ be the N-vector with entries $v_n(t)$. If the level within buffer$_n$ is L at time t, set $x_n(t) := \rho_n[B_n - L]$; let $\mathbf{x} = \mathbf{x}(t)$ be the N-vector with entries $x_n(t)$. [Note that $\mathbf{x}$ is necessarily bounded, as we always have $x_n(t) \leq \rho_n B_n$.] Similarly, let $Y_n(t)$ be the number of orders in $\mathcal{Q}$ at time t for product$_n$ and let $\mathbf{y} = \mathbf{y}(t)$ be the N-vector with entries $y_n(t) := \rho_n Y_n(t)$.

Example: To illustrate the meaning of the scaling factors, consider a set of metal stamping machines. In such a stamping operation, a "production unit" could be a stroke of the stamping press. We assume each machine can accept each part die. If machine^4 is capable of 10 strokes/minute and machine^7 runs at 15 strokes/minute, then $\sigma^4 = 10$ and $\sigma^7 = 15$. If part 1 is produced two at a time for its die, then $\rho_1 = 1/2$, and $\tau_1^4 = 1/20$, $\tau_1^7 = 1/30$.

All orders and buffers are then scaled according to these production units (strokes). Thus:

- $x_n(t)$ is the number production units at time t that are required to refill buffer$_n$ to level B_n;
- $y_n(t)$ is the number of production units at time t required to empty the present order backlog queue, $\mathcal{Q}$, of all orders it contains for product$_n$;
- $u_n(t)$ is the cumulative number of production units for product$_n$ demanded up to time t;
- $v_n(t)$ is the cumulative number of production units corresponding to the filled demand for product$_n$ up to time t.

As an example, suppose that up to time t, only product$_1$ is being demanded, so $\|\mathbf{u}(t)\| = u_1(t)$. If demand for product$_1$ up to time t is 400 units, then $\|\mathbf{u}(t)\| = u_1(t) = 200$ press strokes.

Suppose that $\lambda = 100$. Then the nominal total demand rate for all products is 100 production units (strokes) per minute. If ξ is 20, then over a

complication of the analysis.

period of 1 minute, demand could be as high as $\lambda+\xi = 120$ production units. Over 10 minutes, however, its demand could only be as high as $10\lambda+\xi = 1020$ units.

Similar examples can be constructed for other manufacturing processes such as injection molding, extrusions, etc., where a unit of production may correspond to a fixed number of items. If we considered production of kits of like parts, then instead we may have a single "product" (kit) require multiple units of production ($\rho > 1$), instead of fractional units.

Note that $x_n(t)$ is the number of production units required to fill buffer_n to B_n, and $y_n(t)$ is the number of production units required to fill the orders for product_n in the backlog $\mathcal{Q}$. Thus, $y_n(t) > 0$ implies $x_n(t) > 0$, and the sum of them can be considered a deficit of those products (in production units) required to have no backorders and no empty portion of the buffer for that product.

We can thus look at the change in this deficit over time as the arriving orders less the filled orders over the period. The basic 'bookkeeping' identity is then:

$$[\mathbf{x}+\mathbf{y}](t) - [\mathbf{x}+\mathbf{y}](s) = [\mathbf{u}(t) - \mathbf{u}(s)] - [\mathbf{v}(t) - \mathbf{v}(s)] \tag{2.3}$$

for any (arbitrary) time interval $(s,t]$ $(0 \le s < t)$ and vectors indexed by product $(n = 1,\ldots,N)$.

Since these vectors $\mathbf{x},\mathbf{y},\mathbf{u},\mathbf{v}$ are inherently non-negative, taking the dot product with $\mathbf{1}$ in (2.3) gives

$$\|[\mathbf{x}+\mathbf{y}](t)\| = \|[\mathbf{x}+\mathbf{y}](s)\| + \left\|\mathbf{u}\Big|_s^t\right\| - \left\|\mathbf{v}\Big|_s^t\right\|. \tag{2.4}$$

where $\mathbf{u}\Big|_s^t := \mathbf{u}(t) - \mathbf{u}(s)$ is the total (scaled) units demanded over the interval $(s,t]$, and $\mathbf{v}\Big|_s^t := \mathbf{v}(t) - \mathbf{v}(s)$ is the total (scaled) units of production released from the system over the interval. Let the total deficits at times s and t be the positive scalars ζ_s and ζ_t (respectively):

$$\zeta_s := \|[\mathbf{x}+\mathbf{y}](s)\| = \|\mathbf{x}(s)\| + \|\mathbf{y}(s)\|, \quad \zeta_t := \|[\mathbf{x}+\mathbf{y}](t)\|,$$

Then equation (2.4) becomes simply:

$$\begin{aligned} \zeta_t &= \zeta_s + \left\|\mathbf{u}\Big|_s^t\right\| - \left\|\mathbf{v}\Big|_s^t\right\| \\ &= \zeta_s + \lambda\cdot(t-s) + \tilde{\xi} - \left\|\mathbf{v}\Big|_s^t\right\| \quad \text{for arbitrary intervals } (s,t]. \end{aligned} \tag{2.5}$$

Note that for each machine^m our scaling of production just gives the total demand filled over the period as

$$\left\| \mathbf{v}^m \Big|_s^t \right\| \geq \sigma^m \cdot (\,(t-s) - [\text{setup time}]^m - [\text{idle time}]^m)$$

and summing over m we have

$$\begin{array}{rl} \left\| \mathbf{v} \Big|_s^t \right\| & \geq \sigma^* \cdot (\,(t-s) - [\text{setup time}] - [\text{idle time}]) \\ t-s & \leq \left\| \mathbf{v} \Big|_s^t \right\| / \sigma^* + [\text{setup time}] + [\text{idle time}]). \end{array} \tag{2.6}$$

where $\sigma^* := \sum_m \sigma^m$ is the collective capacity of the set of M machines. In order that the backorder queue $\mathcal{Q}$ stay bounded, the cumulative orders demanded must not substantially exceed the orders fulfilled: over long time intervals we must have $\left\| \mathbf{u} \Big|_s^t \right\| - \left\| \mathbf{v} \Big|_s^t \right\| <$ bound. Comparing equation (2.6) with (2.1), we get the necessary constraint that this collective capacity σ^* must exceed the total demand rate λ:

$$\sigma^* > \lambda \qquad \text{where } \sigma^* := \sum_m \sigma^m. \tag{2.7}$$

In this paper, we consider several protocols which dictate when machines will produce a product and how many will be produced in sequence. A protocol thus specifies how the queue $\mathcal{K}$ of production authorizations is maintained and how it determines the activity of the machines (processors). A key concept of interest is whether our protocols will guarantee a bounded order backlog $\mathcal{Q}$, and further, whether that order backlog will eventually empty and remain empty.

Definition 1 *A protocol is* **stable** *— for given load scalings ρ_n , speeds σ^m , and configuration (buffer sizes) — if $\mathcal{Q}$ remains bounded (i.e., $\|\mathbf{y}(t)\|$ remains bounded for all t) for arbitrary initial state, subject only to the burstiness arrival constraint equation (2.1), and the capacity constraint (2.7). A protocol is* **uniformly stable** *if the bound on the queue size $\|\mathbf{y}\|$ depends only on the initial queue size, uniformly for all admissible input.*

Definition 2 *A protocol is* **clearing** *if, for arbitrary initial state and admissible input as above, $\mathcal{Q}$ necessarily empties; it is* **strongly clearing** *if, in*

addition, $\mathcal{Q}$ stays empty from some time on. A protocol is **uniformly clearing** *if there is some bound on the time to clearing, depending on the initial queue size, but uniform for all admissible inputs; it is* **uniformly strongly clearing** *if this also holds for the time until the queue remains clear.*

Theorem 1 *Every stable protocol is uniformly stable. Similarly, every clearing protocol is uniformly clearing and every strongly clearing protocol is uniformly strongly clearing. Finally, clearing protocols are necessarily stable and so are uniformly stable.*

PROOF: As this theorem is somewhat tangential to our principal interests we only sketch the arguments.
(stable ⇒ uniformly stable) Suppose not. Then each trajectory (i.e., sequence of states[3]) is bounded, but there is some sequence $\{\mathbf{y}^\nu(\cdot)\}$ of such trajectories for which $\|\mathbf{y}^\nu(t_\nu)\| \geq \nu$ for suitable times t_ν. Without loss of generality we can assume there are no state repetitions (at least, earlier than t_ν) in any of the trajectories under consideration here: else one can skip the intervening segment of the trajectory (noting that our definition of 'protocol' requires that the 'reduced' trajectory must again be admissible). By a diagonal argument one can find a subsequence of trajectories with common initial segments of increasing length: all trajectories agree up to the N-th event from some point $\nu(N)$ on. This determines a common 'limit trajectory', which must also be admissible and so should be bounded by the assumed stability. On the other hand, it cannot be bounded since the absence of repetitions ensures a bound on the number of steps until $\|\mathbf{y}\| \geq c$ for any c (as there are only finitely many distinct states with $\|\mathbf{y}\| \leq c$). This contradiction shows the existence of a uniform bound for fixed initial conditions whence, since there are only finitely many initial conditions satisfying any bound, we have uniform stability.

(clearing ⇒ uniform clearing) Again, suppose not. Then there would be a sequence of trajectories with increasing times to clearing and increasing number of arrivals before clearing[4] Again by a diagonal argument, we

[3]Without loss of generality we need consider only those evolutionary histories for which the input is close enough to the rate λ of (2.1) that (uniformly) one can work with sequences of orders and states (buffer and queue contents) without explicit regard to times.

[4]If there would be a long enough interval without input, then (as our definition of 'protocol' ensures the system will not idle for long when the queue $\mathcal{Q}$ is not empty) the system will clear. This clearing can only be delayed if input continues.

can extract a subsequence with increasingly long common initial segments (again determining a 'limit trajectory') and with increasingly long periods to clearing. The limit trajectory is necessarily admissible but never clears — contradicting the assumed clearing property.

(uniform clearing ⇒ uniform stability) Over a uniformly bounded time before clearing, the condition (2.1) bounds arrivals so the possible queue length is bounded.

(strong clearing ⇒ uniform strong clearing) The proof is by the same kind of diagonal argument as above. ■

In this paper we consider two protocols: the Signal Kanban protocol (in two variations) and the Pattern Production protocol.

For the **Signal Kanban** protocols we let R_n (with $0 \leq R_n \leq B_n$) be a 'reorder level' such that kanban_n is to be transmitted to $\mathcal{K}$ to 'order' production by the machine(s) to replace the 'deficit' when the buffer level gets down to R_n; i.e., the kanban is transmitted when x_n reaches $b_n := \rho_n[B_n - R_n] \geq 0$. Each kanban_n is then transferred back to buffer_n when the corresponding production run (consisting of 'setup' plus actual processing time) is completed.

There are two common variants of the Signal Kanban protocol:

- **fixed-batch:** In the fixed-batch version of a Signal Kanban, the production authorization is for a fixed lot size (scaled) of exactly b_n. This is the version of Signal Kanban discussed by Monden [9], and seems to be most common in industry. This is the same as the reorder-point ("two-bin") methods in inventory theory. We normally assume product is transferred to the appropriate buffer immediately (continuously during processing), but we note an alternative subvariant with product delivery of the entire batch made at completion of the production run.

- **fixed-fill:** In the fixed-fill protocol, the signal kanban authorizes production to fill the buffer, i.e., up to B_n. The production runs may be of different lengths depending on the demand for this product from the time of issuance of the kanban authorization until completion of the run with the buffer filled. Queuing delays in $\mathcal{K}$ due to other pending authorizations make the initial buffer level at the start of the run variable, but it is no higher than the reorder level, so b_n represents a *minimum* batch size. The disadvantage of this system in practice is that the variable run lengths require monitoring the buffer levels while

running, as opposed to the simpler fixed-batch scheme. However, from initial simulation studies, the fixed-fill appears to have a better behavior [5]. It is interesting to note that in the limit where the buffer levels become continuous variables instead of discrete, the "fixed-fill" variant is equivalent to the "switched arrival" system shown to be chaotic (the behavior is highly sensitive to initial conditions, and it is highly unlikely that the system will settle into a periodic pattern) by Chase, Serrano, and Ramadge [3].

We emphasize that for either variant the authorized production run will have a (scaled) 'batch size' at least b_n, i.e., b_n is the *minimum* run length. Signal kanbans are discussed in more detail in the next section.

For the **Pattern Production** protocols we have a specified 'cycle' of kanbans in fixed sequence $(n(1), \ldots, n(J))$ with every index n included at least once in the cycle. Each of these kanbans is transferred to the rear end of the queue $\mathcal{K}$ as soon as the corresponding production run is initiated. For the special case where each product appears exactly once in the pattern and there is only one machine, this is equivalent to a Signal Kanban system with $R_n = B_n$. Thus, an authorization is released immediately upon the filling of the buffer. Pattern Production protocols are examined in Section 4.

3 Signal kanbans: amortizing batch sizes

We wish to consider a set of M machines, working collaterally (in parallel) to serve a common set of buffers with a common input stream to the queue $\mathcal{Q}$. In a Signal Kanban system, we associate a reorder point R_n with each buffer_n. Whenever the buffer level falls to this level (corresponding to x_n reaching $b_n := \rho_n[B_n - R_n] > 0$), then a production authorization (kanban) is released to the kanban queue $\mathcal{K}$ to replenish product_n.

The specification of a minimum run length is the substantial point of the Signal Kanban protocol in the presence of setup times: we will choose each R_n (and so b_n) so as to amortize the setup times. Choosing $\alpha > 1$, set $\sigma' := \sigma^*/\alpha$. For any production run of length b (in terms of scaled product), the time taken by the run, including setup time, will then be bounded by $(b + \delta_n)/\sigma$. If we would know, by the specification of the minimum, that $b \geq \delta_n/(\alpha - 1)$, then this total run time would be bounded by b/σ' and σ' becomes a reduced 'effective capacity' for the set of machines, subsuming

consideration[5] of setup times.

We use the obvious specification of queue dynamics: If a production run is in progress, it continues according to the variant selected; in each case the minimum run length is b_n. When the machine finishes such a run, it returns that kanban to its buffer and then immediately begins the new run associated with the kanban now at the head of $\mathcal{K}$, if any. If $\mathcal{K}$ is empty, then the machine idles until a kanban arrives to $\mathcal{K}$ and that run is initiated. A production run, once initiated, consists of the setup, followed by the actual processing; it is possible that two (or more) consecutive production runs may involve the same product[6] and in that case we assume (or, as an alternative variant, may not!) that the setup time is omitted or reduced for all but the first of these runs. We allow the variants of the Signal Kanban protocol regarding 'fixed batch size' or 'fill the buffer' as above. Our analysis will accommodate any of these variants without distinction, assuming the capacity condition (2.7).

The specification of system dynamics requires some further consideration when $M > 1$: How should one handle the possibility of *partial idling* (some, but not all, of the machines being active at t)? If we accept this as a possibility while imposing the capacity condition in the form (2.7), then partial idling means that the battery of machines may be operating below its collective capacity σ^* and it should not then be surprising that stability might fail.[7] To

[5]It will be important to take advantage of the strict inequality in the capacity condition (2.7), to choose $\alpha < \sigma^*/\lambda$ so $\sigma' > \lambda$. Otherwise, one easily sees that with run lengths short enough that the capacity condition now fails — i.e., if $\sigma' < \lambda$ in (2.7) — then the system cannot be stable: too high a proportion of the time would be taken up by the setups and the effectively reduced capacity becomes inadequate to keep up with demand input.

The significance of our choice of $\alpha > 1$ is that a larger α (farther from 1) means smaller batch sizes and requires smaller buffers – but somewhat limits the load which could be handled, corresponding to a possible increase in λ.

[6]This can happen if there is an intervening idle period or if we are working with a fixed batch size and, when the kanban is returned to the buffer, the buffer level is already at or below the reorder level (due to further demand since the kanban was previously transmitted to $\mathcal{K}$) and the kanban is then immediately retransmitted to $\mathcal{K}$. Of course, this also assumes that no other kanban is transmitted between these occurrences.

[7]Consider the scenario (say, in the 'fill the buffer' variant) in which, from some moment on, the external demand stream becomes entirely composed of orders for some one product$_{\bar{n}}$. The kanban$_{\bar{n}}$ has initiated a production run for product$_{\bar{n}}$ (e.g., at machine$^{\bar{m}}$) and, after a while, all the other machines would be idle while machine$^{\bar{m}}$ works at speed $\sigma^{\bar{m}}$. The capacity condition (2.7) permits arrival of orders at a (scaled) rate λ. Although $\lambda < \sigma^*$ by assumption, it remains possible that $\lambda > \sigma^{\bar{m}}$ — the input rate dominating the pro-

obtain a positive result we must include some 'assistance rule' in extending the protocol to the collateral case, rather than having a kanban *exclusively* assign a production run to a single one of the machines. We will think of $\mathcal{K}$ as consisting of two disjoint parts: $\mathcal{K}_a := \{$those kanbans in $\mathcal{K}$ already assigned to some machine (so a production run is already in progress)$\}$ and $\mathcal{K}_0 := \{$those kanbans in $\mathcal{K}$ awaiting assignment$\}$. A machine, on completing a production run, is assigned a new run from $\mathcal{K}_0$ if that is nonempty and otherwise initiates a run 'in assistance' from $\mathcal{K}_a$ (e.g., the first, although this choice does not affect our argument); if $\mathcal{K}_0$ and $\mathcal{K}_a$ are both empty, then the system is (entirely) idle with $\mathcal{K}$ empty. While we have described a particular version of the assistance rule for definiteness, we will really need only two properties of this for our proof:

- When the system is idle each buffer must be above its reorder level.
- When several machines participate in a compound production run, we will have a setup time for each (with the possibility of omitting this applicable on a machine-by-machine basis), but cannot have more than M such altogether for any single such run.

It would be convenient to have also the property that all machines involved in a compound production run finish simultaneously (with a 'priority rule' to determine order of subsequent assignment). In this situation, we would be concerned with the possibility of a machine assigned to to the production of a part being within a setup process when the production is completed by the others machines. In the case of 'fixed batch size' this would necessarily be predictable and we modify the assistance rule to assign this machine to that run, but not have setup actually performed — although the time until the run is completed might be counted as setup time, rather than as idling. In the case of 'fill the buffer' this predictability might fail and we will allow for this possibility of incomplete setup in our analysis.

Theorem 2 *Consider a Signal Kanban protocol as described above for collateral operation of a set of M machines satisfying* (2.7)*. We choose α so $1 < \alpha < \sigma^*/\lambda$ and suppose the reorder levels $R_n \geq 0$ are set so that*

$$b_n := \rho_n[B_n - R_n] \geq M\delta_n/(\alpha - 1) \tag{3.1}$$

duction rate so the system cannot keep up, i.e., instability. The problem obviously is that σ^* truly represents system capacity only in that multiple machines can combine to handle the load and, for our setting, this must be allowed to hold even when the composition of the load might be concentrated on a set of products smaller than the number of machines.

for each n. The following statements can then be made for either the fixed batch or fixed fill variants of the protocol:

1. *The protocol is uniformly stable.*

2. *The protocol is uniformly clearing: the maximum length of any active interval $(s,t]$ is at most*

$$T_{max} := \frac{\zeta_s + \xi}{\sigma^*/\alpha - \lambda} \tag{3.2}$$

   where we note that, except initially, $\zeta_s < b_{tot} := \sum_n b_n$.

3. *If the buffer sizes are large enough [see (3.5) and, e.g., (3.7), below], then the protocol is (uniformly) strongly clearing.*

PROOF: Fix a time s at which the system becomes active (i.e., either $s = 0$ or s ends an idle period. Our protocol description ensures that if any machine is 'idle' then all machines are idle and $\mathcal{Q}$ is empty so $\mathbf{y} = 0$ and $\|\mathbf{x}\| < b_{tot} := \sum_n b_n$, whence $\zeta_s < b_{tot}$ when $s \neq 0$ here. We then fix an interval $(s,t]$ during which the system remains active without any idling. Indexing production runs (whether simple or compound) during this interval by ν, we have

$$\sigma^m(t-s) \leq \sum_\nu [b+\delta]_\nu^m$$

where $[b,\delta]_\nu^m$ here represent the shares of (scaled) production and setup, respectively, for run$_\nu$ within $(s,t]$ by machine^m and we have no idle time to consider. Summing over m we get

$$\sigma^*(t-s) \leq \left\{ \sum\nolimits_\nu{}' + \sum\nolimits_\nu{}'' \right\} [b+\delta]_\nu$$

where we have separated the sum into $\sum'$, consisting of those (complete) runs occurring entirely within $(s,t]$, and $\sum''$, the remaining (partial) runs: note that this latter sum consists of those runs not yet finished by t since our choice of s ensures that none of the runs involved here is incomplete because it was initiated before s. We then have $\left\{ \sum' + \sum'' \right\} [b]_\nu = \left\| \mathbf{v} \Big|_s^t \right\|$.

For each of the complete runs in $\sum'$ we know that $b \geq b_n$ with $n = n(\nu)$ and that we have at most one setup time for each of the machines,

whence $[\delta]_\nu \le M\delta_n$. Thus, using (3.1), we have $[b+\delta]_\nu \le \alpha[b]_\nu$ for each of these runs. To estimate $\sum''$ we note that there can be at most M machines involved in those runs in process at time t so, altogether, their setup 'costs' can be at most $M\delta_{max}$ with $\delta_{max} := \max_n\{\delta_n\}$: i.e., $\sum''[\delta]_\nu \le M\delta_{max}$. Combining these gives, since $\alpha > 1$,

$$\begin{aligned} \sigma^*(t-s) &= \left\{ \sum_\nu{}' + \sum_\nu{}'' \right\} [b+\delta]_\nu \\ &\le \alpha \sum_\nu{}' [b]_\nu + \left\{ M\delta_{max} + \sum_\nu{}'' [b]_\nu \right\} \\ &\le M\delta_{max} + \alpha \left\| \mathbf{v}\Big|_s^t \right\|. \end{aligned} \tag{3.3}$$

Using (2.5) in (3.3) gives

$$\zeta_t \le \zeta_s + \xi + (\lambda/\sigma^*) M\delta_{max} - \left(1 - \frac{\lambda\alpha}{\sigma^*}\right) \left\| \mathbf{v}\Big|_s^t \right\|$$

and, as $\lambda\alpha/\sigma^* < 1$ by assumption, the last term above can be omitted and we have

$$\zeta_t \le \zeta_s + \xi + (\lambda/\sigma^*) M\delta_{max}. \tag{3.4}$$

Since $\zeta_t := \|[\mathbf{x}+\mathbf{y}](t)\|$ and ζ_s is uniformly bounded (either ζ_0 or $< b_{tot}$), this shows that the protocol is uniformly stable, so statement *1* of this theorem is proved.

Next we determine the maximum time the system can be busy without returning to idle. Taking t above to be the end of the active interval, we have $\sum''$ empty so we can omit the $M\delta_{max}$ term of (3.3). From (2.5) we then have

$$\zeta_t \le \zeta_s + \lambda \cdot (t-s) + \tilde{\xi} - (\sigma^*/\alpha)\,(t-s).$$

Solving for $(t-s)$ and recognizing that $\zeta_t \ge 0$, we arrive at

$$(t-s) \le \frac{\zeta_s + \xi}{\sigma^*/\alpha - \lambda},$$

which gives us the desired bound (3.2).

To prove result *3* of the theorem, our simplest argument is to note that $x_{\bar{n}}(\tau) < b_{\bar{n}}$ for τ within any idling period while for τ in a non-initial active period $(s,t]$ we have, as $s \le \tau \le t \le s + T_{max}$,

$$x_{\bar{n}}(\tau) \le x_{\bar{n}}(s) + [\lambda(\tau - s) + \xi] < b_{\bar{n}} + \lambda T_{max} + \xi$$

with T_{max} computed as in (3.2) using $\zeta_s < b_{tot}$. Clearly, if the buffer size $B_{\bar{n}}$ is set large enough that

$$\rho_{\bar{n}} B_{\bar{n}} \geq b_{\bar{n}} + \xi + \frac{b_{tot} + \xi}{\sigma^*/\alpha - \lambda}, \tag{3.5}$$

then this buffer could not empty as would be necessary to have an order for product$_{\bar{n}}$ at the head of the backlog queue $\mathcal{Q}$. If this holds for *every* $\bar{n}$, then $\mathcal{Q}$ must remain empty since no order could be at its head.

While proving statement *3*, this universal and explicit estimate (3.5) is clearly unreasonably pessimistic, since an input stream maximizing the length of the active interval as in (3.2) would not be expected to maximize $x_{\bar{n}}(\tau)$ at $\tau = t$.

We would prefer a sharper estimate of the lower bounds for the buffer sizes which will ensure strong clearing. For the general case with $M > 1$ the situation is rather complicated, as one must consider combinatorial aspects of the machine assignments and assistance to obtain a 'good' bound for $x_{\bar{n}}(\tau)$ in an active period. We will seek to obtain such an estimate only under the additional assumption[8] that it requires *all* the machines to keep up with demand arriving at the maximal rate of (2.1) — i.e., with (2.7) we would have

$$\sigma^* - \sigma^m < \lambda < \sigma^* \qquad \forall m. \tag{3.6}$$

We also continue to assume that (3.1) holds for each n. A simplifying consequence of (3.6) is that a time τ at which $x_{\bar{n}}(\tau)$ attains its maximum must immediately precede a period of production of product$_{\bar{n}}$ with assistance by all machines so all production runs during $(s, \tau]$ for products $n \neq \bar{n}$ are completed and for such n we have[9]

$$[\text{production}]_n = K_n b_n \text{ with } K_n := \left\lfloor \frac{x_n(s) + [\text{input}]_n}{b_n} \right\rfloor.$$

We note here that the expression $(\alpha\lambda/\sigma^*)K_n b_n - [\text{input}]_n$ is then maximized (least negative) when $x_n(s)$ is as large as possible (for $s-$ ending an

[8]This holds automatically for the single machine case $M = 1$ and we note that the requirement we obtain will be sharp in that case. When $M > 1$ it seems possible that some slight further improvement might still be achieved by a more detailed analysis (e.g., to show that one typically has one, rather than M setups for each production run), but we do not pursue this and consider (3.7) sufficiently sharp.

[9]We also write $[\text{production}]_{\bar{n}} = K_{\bar{n}} b_{\bar{n}}$ with $K_{\bar{n}} \geq 1$, although $K_{\bar{n}}$ will not, in general, be an integer.

idle period: $x_n(s) = b_n - \rho_n$) and the input is just what is needed to trigger a run (i.e., $[\text{input}]_n = \rho_n$ so $K_n = 1$) for $n \neq \bar{n}$. Further, each of these complete runs is amortized so

$$[\text{production} + \text{setup time}]_{n\neq\bar{n}} \leq \alpha[\text{production}]_{n\neq\bar{n}} = \alpha \sum_{n\neq\bar{n}} K_n b_n$$

while, as $K_{\bar{n}}$ need not be an integer, we similarly have

$$[\text{production} + \text{setup time}]_{\bar{n}} \leq \alpha K_{\bar{n}} b_{\bar{n}} + \tilde{\delta} \qquad \left(\tilde{\delta} \leq \delta_{\bar{n}}\right).$$

Note that (2.6) then gives

$$\tau - s \leq [\text{production} + \text{setup time}]/\sigma^* \leq \left(\alpha \sum_n K_n b_n + \tilde{\delta}\right)/\sigma^*.$$

At this point we are ready to bound $x_{\bar{n}}$: using the observations above and (2.1), we have

$$\begin{aligned}
x_{\bar{n}}\Big|_s^\tau &= [\text{input}]_{\bar{n}} - [\text{production}]_{\bar{n}} \\
&= [\text{input}] - [\text{input}]_{n\neq\bar{n}} - K_{\bar{n}} b_{\bar{n}} \\
&\leq \lambda(\tau - s) + \xi - [\text{input}]_{n\neq\bar{n}} - K_{\bar{n}} b_{\bar{n}} \\
&\leq (\lambda/\sigma^*)\left(\alpha \sum_n K_n b_n + \tilde{\delta}\right) + \xi - [\text{input}]_{n\neq\bar{n}} - K_{\bar{n}} b_{\bar{n}} \\
&= \xi + (\lambda/\sigma^*)\delta_{\bar{n}} - (1 - (\alpha\lambda/\sigma^*)K_{\bar{n}} b_{\bar{n}} + \sum_{n\neq\bar{n}} ((\alpha\lambda/\sigma^*)K_n b_n - [\text{input}]_n) \\
&\leq \xi + (\lambda/\sigma^*)\delta_{\bar{n}} - (1 - (\alpha\lambda/\sigma^*)b_{\bar{n}} + \sum_{n\neq\bar{n}} ((\alpha\lambda/\sigma^*)b_n - \rho_n)
\end{aligned}$$

$$\begin{aligned}
x_{\bar{n}}(\tau) &= x_{\bar{n}}(s) + x_{\bar{n}}\Big|_s^\tau \leq b_{\bar{n}} - \rho_{\bar{n}} + x_{\bar{n}}\Big|_s^\tau \\
&\leq \xi + (\lambda/\sigma^*)\delta_{\bar{n}} + \sum_n ((\alpha\lambda/\sigma^*)b_n - \rho_n).
\end{aligned}$$

Thus, as earlier for (3.5), we see that requiring

$$\rho_{\bar{n}} B_{\bar{n}} > \xi + \frac{\lambda}{\sigma^*}\delta_{\bar{n}} + \sum_n \left(\frac{\alpha\lambda}{\sigma^*} b_n - \rho_n\right) \tag{3.7}$$

for each $\bar{n}$ is sufficient, in the context of (3.6) and subject to (3.1), to ensure strong clearing. ■

Example: To illustrate the above results, consider a manufacturing facility with two "machines", with scaled production rates of $\sigma^1 = \sigma^2 = 9$ units/min. Our machines produce six products with scaled setup for each product of $\delta_1 = \delta_2 = \delta_3 = \delta_4 = 80$ (production units) and $\delta_5 = \delta_6 = 40$. These represent worst-case setups – when switching between certain pairs of products, common settings between particular products may reduce the setup effort. In our example, a production unit equals an item of product, so $\rho_i = 1$ for $1 \leq i \leq 6$. For the production rates given, the average setup time on machine^1 thus corresponds to 8.9 minutes.

Suppose that we have a demand of $\lambda = 15$ units/minute. The collective capacity is then $\sigma^* = \sigma^1 + \sigma^2 = 18$ units/minute, and the utilization is $\lambda/\sigma^* = 83.3\%$. The parameter α must be chosen between $1 < \alpha < \sigma^*/\lambda = 1.2$. First let us consider the case where $\alpha := 1.15$. In this case, by equation 3.1, $b_1|_{\alpha=1.15} \geq 1067$ units, which would correspond to at least 71 minutes of supply if demand swings entirely to product 1. If demand did swing entirely to one product, then the two machines must both eventually help, since the rate of demand would exceed the capability of either machine by itself. Note that regardless of the demand, each production run will take $(b_n + \delta_n)/\sigma^1 = 127$ minutes if run on just one machine.

Note that product$_6$ setup is only $\delta_6 = 40$. For this, $b_6|_{\alpha=1.15} \geq 533.3$ units. This is significantly lower than for product$_1$, illustrating (as we would expect) that the minimum scaled buffer size (and thus the investment in inventory) is reduced proportional to scaled setup time δ_i.

The parameter α must lie between 1 and σ^*/λ. Moving α closer to one effectively limits the load by forcing larger batches and less frequent setups. Thus, it in effect reserves capacity that could cover a potential long-term increase in λ. To illustrate the effect of decreasing α, suppose now that $\alpha = 1.05$. Then $b_1|_{\alpha=1.05} \geq 3200$ units.

The buffer bounds calculated above ensure that the system will be stable and the backlog will be bounded. However, typically we are more concerned with preventing a backlog from occuring. Assume that our burstiness bound ξ is 5 units. From above, we have $\sum_{i=1}^{6} b_i|_{\alpha=1.15} \geq 5336$. Then from equation 3.7, we have the buffer size for product$_1$ must be

$$\rho_1 B_1|_{\alpha=1.15} > 5179.3$$

Thus, if the buffer for product$_1$ is at least 5180 parts deep (approximately 5.75 hours of maximum demand), then it is sufficiently deep such that no

part shortage will occur, even under the "worst case" situation of all kanbans being released simultaneously. We should emphasize that this this value is primarily due to the term $\sum_{i=1}^{6} \alpha\lambda b_i/\sigma^*$, which corresponds to replenishing when all products are at their reorder points simultaneously — possible, although unlikely.

4 Pattern production

We continue to consider a set of M machines working collaterally (in parallel) to serve a common set of buffers with a common input stream to the queue $\mathcal{Q}$. In this section we wish to show stability and strong clearing for Pattern Production protocols, such as simple 'Round Robin', in which there is a predetermined cyclical specification of the sequencing of production runs. In general the system will be continuously active, since the queue $\mathcal{K}$ will always contain the full (cyclical) set of kanbans. However, there will be no positive minimum for the batch sizes and the problem is to verify that the setup times are amortized by an *automatic* adjustment of the batch sizes so the setup times just fill the capacity gap $(\sigma^* - \lambda)$. While there are no 'reserve levels' to specify for these protocols, it will be necessary to require lower bounds on the buffer sizes.

Two comments are in order here as to the precise specification of the protocol. First, much as in our description of the Signal Kanban protocols, we wish to allow (as one alternative possibility) that we omit the setup when it is clearly unnecessary: here, if the $\text{buffer}_{n(j)}$ is already completely full when one comes to the j-th kanban of the cycle. Conceivably there might be some interval, necessarily with no external demand input, during which *all* buffers remain full and, in the no-setups variant, we do not take this as meaning there are infinitely many (totally degenerate) 'cycles' in no time, but simply move the 'head' of the kanban queue to the kanban for the next arriving order, with some selection rule or arbitrary choice if that might be ambiguous due to multiple occurrence of this n in the cyclical pattern. Second, we note — by the same logic as preceding Theorem 2 — that 'assistance' may be necessary. However, this is now implicit in what we have described of the protocol without introducing a special 'assistance rule': if machine^m becomes available and the next kanban is one for which a production run is already under way with[10] that buffer not yet full, then machine^m initiates an 'assisting

[10]It is *almost* necessary that the buffer be not yet full if a production run is under way,

run' for the same product.

Theorem 3 *Consider a 'Pattern Production' (cyclical) protocol for collateral operation of a set of M machines satisfying* (2.7)*. We assume that all the buffer sizes have been set so*

$$\begin{aligned} b_n := \rho_n B_n > \zeta_* := \frac{(\lambda/\sigma^*)\Delta^*}{1-\lambda/\sigma^*} \\ \textit{with} \quad \Delta^* := M\Delta + (M-1)\delta_{max} \qquad \Delta := \textstyle\sum_j \delta_{n(j)}. \end{aligned} \tag{4.1}$$

Then

1. *The protocol is uniform clearing and stable. Moreover, for $M = 1$ the same conclusion holds even if we admit one exception to (4.1).*

2. *If the buffer sizes satisfy the stronger condition*

$$b_n > \zeta^* := \frac{2-\lambda/\sigma^*}{1-\lambda/\sigma^*}(\lambda/\sigma^*)\Delta^* + \xi, \tag{4.2}$$

   then the protocol is uniform strongly clearing.

PROOF: The key to the proof will be the property:

> *If j is the first time $n = n(j)$ occurs, then the production of product$_n$ in the j-th run of the cycle must be at least $x_n(s)$.* (4.3)

This property is clear when $M = 1$, but our first concern here is to determine a suitable replacement for it, since (4.3) is no longer valid, as stated, in the more general context. The difficulty is that, unless $x_n(s) = 0$, a production run for product$_{n(j)}$ will be initiated, but need not now terminate within the cycle: this is, after all, the reason we have found it necessary to introduce the notion of 'assistance'. Hence the desired lower bound on production in (4.3) may fail in considering a single cycle.

---

but not quite: a first machine might initiate a run of product$_n$ and then a second machine initiate assistance, after which it is possible that a third machine will become available with this again the 'next kanban' after the first machine has already filled the buffer, but while the second machine is still in its setup time, so the run remains 'under way'.

Instead, we will analyze production over an interval $(s,t]$ consisting of several consecutive cycles, defining such intervals by requiring that each production run associated with a 'first occurrence' j of product$_{n(j)}$ in the first cycle of the interval must have terminated by the end of the interval with buffer$_{n(j)}$ filled. Then (4.3) holds in the present setting with 'cycle' replaced by 'interval'. Summing this over j — i.e., over n, since each n occurs for its first time exactly once in $(s,t]$ — we see that we always have

$$\left\| \mathbf{v}\Big|_s^t \right\| \geq \sum_n x_n(s) = \|\mathbf{x}(s)\|. \tag{4.4}$$

Excluding the possibility that the input stream of orders is finite, we have a well-defined sequence of intervals$_\nu$ for $\nu = 1, 2, \ldots$ with $\nu \to \infty$ as $t \to \infty$. We let t_ν denote the completion time of the ν-th interval (with $t_0 = 0$) and consider $(s,t] = (t_{\nu-1}, t_\nu]$. Here (2.6) becomes

$$\sigma^* \cdot (t-s) = \left\| \mathbf{v}\Big|_s^t \right\| + \tilde{\Delta} \qquad (\tilde{\Delta} = \tilde{\Delta}_\nu := [\text{total of setup times}])$$

since no time is occupied by idling.

To obtain an estimate for the [total setup time]$_\nu =: \tilde{\Delta}_\nu$ associated with such an interval, we first observe that one of our 'intervals' can consist of at most[11] M cycles. It is then clear that the setup times associated with all runs (including 'assisting runs') initiated within the cycles comprising the interval$_\nu$ cannot exceed $M\Delta$. We do note, however, the possibility that at the time $t_{\nu-1}$ at which the interval$_\nu$ began, perhaps as many as $(M-1)$ of the machines were involved in doing setups and to estimate $\tilde{\Delta}_\nu$ we must also allow for these. Altogether, $\tilde{\Delta} = \tilde{\Delta}_\nu \leq \Delta^*$ with Δ^* as in (4.1) and using this in (2.5) gives the recursive identity

$$\zeta_t = \zeta_s - (1 - \lambda/\sigma) \left\| \mathbf{v}\Big|_s^t \right\| + \left[(\lambda/\sigma)\tilde{\Delta}_\nu + \tilde{\xi}_\nu \right]. \tag{4.5}$$

[11] To see this, consider any j which is a first occurrence of $n = n(j)$ in the cycle pattern and the production run associated with this, starting in the first cycle of the interval. If this run has not terminated (with buffer$_n$ filled) by the occurrence of j in the second cycle of the interval, then a second machine will initiate an assisting run of the same product$_n$. By the occurrence of j in the M-th cycle of the interval — if the interval has extended this long because this run remains incomplete — all M machines will be working on processing this same product$_n$ so the M-th cycle could only end after the completion of this compound run. (As before, we note that the capacity condition (2.7) ensures that this must eventually happen.)

We now assume that (4.1) holds for all n and distinguish two cases for the system state at the time $s = t_{\nu-1}$: either there is no backlog at that time (i.e., $\mathcal{Q}$ is empty and $\mathbf{y}(s) = 0$) or there is a backlog (i.e., $\mathcal{Q}$ is nonempty).

Case 1: If the queue $\mathcal{Q}(s)$ is empty so $\mathbf{y}(s) = 0$, then $\zeta_s = \|\mathbf{x}(s)\|$ and we may use (4.4) in (4.5) to get

$$\zeta_t \leq (\lambda/\sigma)\zeta_s + \left[(\lambda/\sigma)\Delta + \tilde{\xi}_\nu\right]. \tag{4.6}$$

[So far, this has used no assumption about the buffer sizes.]

Case 2: If, however, $\mathcal{Q}(s)$ is nonempty, we denote by $\bar{n}$ the index of the item at the head of that queue. This is possible only if the buffer$_{\bar{n}}$ is empty so $x_{\bar{n}}(s) = b_{\bar{n}}$ and we note from (4.4) that this gives: $\left\|\mathbf{v}\Big|_s^t\right\| \geq b_{\bar{n}}$. Using this in (4.5) gives

$$\begin{aligned} \zeta_t \;&\leq \zeta_s - (1-\lambda/\sigma)b_{\bar{n}} + \left[(\lambda/\sigma)\tilde{\Delta}_\nu + \tilde{\xi}_\nu\right] \\ &\leq \zeta_s + \tilde{\xi}_\nu - c \end{aligned} \tag{4.7}$$

where

$$c := \min_n\{(1-\lambda/\sigma^*)b_n - (\lambda/\sigma^*)\Delta\}. \tag{4.8}$$

The assumption (4.1) just ensures that $c > 0$.

> Now, for $M = 1$ (so our intervals are simple cycles), consider the modifications needed if we admit a single exception to (4.1). Without loss of generality we may assume the exception is for $n = 1$ and that we have have taken the 'beginning' of the cycle there, so $j(1) = 1$. We modify the case distinction to consider also the state at the time s' at which the first production run of cycle$_\nu$ (for product$_1$) terminates: Case 1 now corresponds to "$\mathcal{Q}(s')$ is empty" (rather than to "$\mathcal{Q}(s')$ is empty" as earlier) and Case 2 corresponds to "$\mathcal{Q}(s')$ nonempty." More specifically, Case 1 now includes both Case 1′ with $\mathcal{Q}$ also empty at s (as above, with the identical analysis) and also Case 1″ where $\mathcal{Q}(s)$ is nonempty and $\bar{n} = 1$ but $\mathcal{Q}(s')$ is empty (thus, $\mathcal{Q}(s)$ contained only product$_1$, and no new backorders entered $\mathcal{Q}$ while processing product$_1$ over the interval $(s, s']$). For Case 2, we now either have $\bar{n} \neq 1$ so (4.1) applies and the analysis is as before, or we have $\bar{n} = 1$ with $\mathcal{Q}(s')$ nonempty (thus, backorders entered $\mathcal{Q}$ during the production run of *product*$_1$ over the interval $(s, s']$). In the latter subcase the index of the item at the head of the queue at s' will be $\bar{n}'$, necessarily with $\bar{n}' \neq 1$ so (4.1) will apply and, since this $\bar{n}'$ must occur later in the cycle than $\bar{n} = 1$ with $j(\bar{n}) = 1$, we note that cycle$_\nu$ contains a production run for product$_{\bar{n}'}$

starting with buffer$_{\bar{n}'}$ empty so $\left\|\mathbf{v}\Big|_s^t\right\| \geq b_{\bar{n}'}$.

Either way, Case 2 gives $\zeta_t \leq \zeta_s + \tilde{\xi}_\nu - c$, i.e., (4.7), with $c > 0$ now defined by taking the minimum over $n \neq 1$ in (4.8). For Case 1$''$ all of $\mathbf{y}(s)$ will have been subtracted from the respective buffers by the time s' to give $\mathbf{y}(s') = 0$ and we note that the production of product$_1$ during the first run of the cycle must be at least $x_1(s) + y_1(s)$ and the production of each other product ($n \neq 1$) occurs subsequent to s' and so must be at least $x_n(s') \geq x_n(s) + y_n(s)$: altogether we must have $\left\|\mathbf{v}\Big|_s^t\right\| \geq \sum_n [x_n(s) + y_n(s)] = \zeta_s$ as earlier so, for either subcase, we still have (4.6) for Case 1.

A first consequence of (4.7) is then that any backlog occurring must eventually clear: one can only have finitely many[12] consecutive cycles in Case 2. Setting

$$k(\nu) := [\text{number of cycles in Case 1 through cycle}_\nu],$$

we note that (4.1) ensures that: $k(\nu) \to \infty$ as $\nu \to \infty$. We now observe that (4.6) and (4.7) give

$$\begin{aligned} \zeta_t \;&\leq (\lambda/\sigma^*)^{k(\nu)}\zeta_0 \\ &\quad + \left[1 - (\lambda/\sigma^*)^{k(\nu)}\right] \frac{(\lambda/\sigma^*)\Delta}{1 - \lambda/\sigma^*} \\ &\quad + \sum_{\mu=1}^{\nu} (\lambda/\sigma^*)^{k(\nu)-k(\mu)} \tilde{\xi}_\mu \end{aligned} \tag{4.9}$$

for $\nu = 0, 1, \ldots$. This is trivially true for $\nu = 0$ with $k(0) = 0$, and then proceeds easily by induction — using for this, as appropriate, either (4.6) with $k(\nu+1) = k(\nu) + 1$ or else (4.7) with $k(\nu+1) = k(\nu)$.

For an arbitrary time τ (falling in the $(\nu+1)$-th cycle, so $t_\nu \leq \tau \leq t_{\nu+1}$) the same analysis which gave (4.5) can also be applied to the time interval $(t_\nu, \tau]$, giving[13]

$$\zeta_\tau \leq \zeta_t + \left[(\lambda/\sigma^*)\Delta^* + \tilde{\xi}_\tau\right] \tag{4.10}$$

[12]From (4.7) we have: $\zeta_{t_{\nu+k}} \leq \zeta_{t_{\nu-1}} + \tilde{\xi} - kc$ if interval$_\nu$ through interval$_{\nu+k}$ were Case 2 intervals (with $\tilde{\xi}$ corresponding to the entire concatenated time interval so $\tilde{\xi} \leq \xi$). With $c > 0$ this would contradict the positivity of $\zeta_{t_{\nu+k}}$ for large enough k, greater than $(\zeta_s + \xi)/c$.

[13]In (4.10) we could replace the bound on setup time within $(s, \tau]$ by summing $\delta_{n(j)}$ only over those setups actually occurring by τ, but we use Δ^* as in (4.1) for simplicity.

and combining this with (4.9) gives

$$\begin{aligned} \zeta_\tau &\leq (\lambda/\sigma^*)^{k(\nu)}\zeta_0 + \frac{(2-\lambda/\sigma^*)(\lambda/\sigma^*)\Delta}{1-\lambda/\sigma^*} + \Xi_\tau \\ \text{with} \qquad \Xi_\tau &:= \tilde{\xi}_\tau + \sum_{\mu=1}^{\nu} (\lambda/\sigma^*)^{k(\nu)-k(\mu)}\tilde{\xi}_\mu. \end{aligned} \tag{4.11}$$

To estimate Ξ_τ, begin by defining

$$\tilde{\Xi}_\kappa := \tilde{\xi}_\tau + \sum_{\mu=\kappa}^{\nu} \tilde{\xi}_\mu = \|\mathbf{u}(\tau) - \mathbf{u}(t_{\kappa-1})\| - \lambda \cdot (\tau - t_{\kappa-1})$$

$$\beta_\kappa := (\lambda/\sigma^*)^{k(\nu)-k(\kappa)} \qquad (\beta_{\nu+1} = 1)$$

for $\kappa = 1, \ldots, \nu+1$. Note that each $\tilde{\Xi}_\kappa \leq \xi$ by (2.2) and that $\tilde{\xi}_\mu = \tilde{\Xi}_\mu - \tilde{\Xi}_{\mu+1}$ for $\mu = 1, \ldots, \nu$ with $\tilde{\xi}_\tau = \tilde{\Xi}_{\nu+1}$. Further, since $k(\cdot)$ is nondecreasing, we have $\beta_\kappa \geq \beta_{\kappa+1}$ for each κ. Simplifying the summations by setting $\tilde{\Xi}_{\nu+2} := 0$ and $\beta_0 = 0$, we may use Abel's formula for summation by parts to get

$$\begin{aligned} \Xi_\tau &= -\sum_{\mu=1}^{\nu+1} \beta_\mu \left[\tilde{\Xi}_{\mu+1} - \tilde{\Xi}_\mu\right] = \sum_{\mu=1}^{\nu+1} \left[\beta_\mu - \beta_{\mu-1}\right] \tilde{\Xi}_\mu \\ &\leq \sum_{\mu=1}^{\nu+1} \left[\beta_\mu - \beta_{\mu-1}\right] \xi = \left[\beta_{\nu+1} - \beta_0\right] \xi \qquad = \xi. \end{aligned}$$

This shows that $\Xi_\tau \leq \xi$ for all τ, which gives uniform stability in view of (4.11) — i.e., subject to (2.7), (4.1). Asymptotically,

$$\zeta_\tau \leq (\lambda/\sigma^*)^{k(\nu)}\zeta_0 + \zeta^* \;\to\; \zeta^* \qquad (\text{as } \tau \to \infty)$$

since then $k(\nu) \to \infty$.

Finally, if (4.2) holds — so *a fortiori* (4.1) holds — then, from some time on (such that $k(\nu)$ is large enough to make the first term on the right of (4.11) negligible) one has $\zeta_\tau < b_n$ for every n. Thus, as each $x_{\bar{n}}(\tau) \leq \zeta_\tau$, no $\text{buffer}_{\bar{n}}$ can be empty so no $\bar{n}$ can index the item at the head of the queue. This means that $\mathcal{Q}$ must then be empty — which is just the strong clearing property. ■

Example: To illustrate our results for a pattern production system, we consider the same example as at the end of section 3. From equation 4.1,

$b_n := \rho_n B_n > \zeta_* = 4400$ production units. This equates to 4.8 hours of maximum demand for the product.

To ensure strongly clearing, from equation 4.2, $b_n > \zeta^* = 5138.3$. Thus, if the buffer is at least 5139 production units (products) deep, then we are assured that there will be sufficient inventory to prevent backlogs even under changes in demand mix and under demand burstiness.

Now, suppose that the sequence pattern of production can be chosen to take advantage of similarities of adjacent products in the sequence, thus reducing the setup times.[14] In particular, suppose the resulting savings in setups due to sequencing leads to $\Delta' = 200$ and $\delta'_{max} = 60$, which then gives $\zeta_* = 2300$ as a bound to ensure stability, and $\zeta^* = 2688.3$ as a bound to ensure strongly clearing and thus no backlogs. This illustrates that our analysis can be applied to cases where the production sequence is carefully selected to reduce setup times.

5 Discussion

In this paper, we have examined several variants of production protocols for manufacturing systems where setups can have a significant impact on capacity. Each protocol examined is a pull protocol, so production is directly in response to prior and current consumption (demand). We considered systems where the demand is unknown. A maximum average long-term demand rate is assumed known, but the demand can exceed this bound subject to a 'burstiness' constraint. There are no assumptions on the distribution of this demand or the burstiness among the different products being produced. We examined variants of the Signal Kanban systems and Pattern Production systems, for the situation of single machine and parallel machines.

Our focus in this paper was on determining whether these protocols were stable and strongly clearing. A protocol is stable if the backorder queue remains bounded, and is strongly clearing if the backorder queue eventually empties and remains empty thereafter. For each of the policies we considered (subject to a capacity constraint), conditions on the buffer sizes were found in order to ensure that the policies would be both stable and strongly clearing.

[14]This is very reasonable when $M = 1$ when the sequence of products in the pattern directly determines the sequence of products on the machine. Using the sequence to reduce setups when $M > 1$ seems less reasonable, but we nonetheless continue with our multi-machine example.

We should emphasize that we assumed no structure to the distribution of demand among products, or of the distribution of the burstiness among products. As such, the bounds that were determined for setting buffer sizes and reorder points would be conservative in practice. In lean manufacturing, where pull production methods are commonly used, there is a strong emphasis on *heijunka*, leveling demand over time and among products. This has an impact on our results in two ways. First, there would be a conscious effort to minimize the burstiness ξ. Secondly, for each product$_n$, we would have a maximum demand rate of $\lambda_n \leq \lambda$ such that $\lambda \leq \sum_{n=1}^{N} \lambda_n \leq N\lambda$. The developments in this paper have considered the worst case where $\lambda_n = \lambda$ for all n, but reducing each λ_n to more accurately reflect knowledge of demand would reduce the buffer level bounds accordingly.

There are several directions for continued work on these protocols. First, it would be useful to determine better buffer sizes and reorder points when knowledge of the distribution of demand among products is known. A second direction for investigation is the performance of the protocols. Average inventory is one performance measure of importance in many industries. Yang [17] used simulation to compare the average inventory of a fixed-fill signal Kanban system (a reorder point policy) with a model intended to approximate traditional Kanban methods, but which in fact resembles the behavior of a pattern production system (specifically, in his model, inventories are always replenished to their maximum level, and an authorization is then immediately reissued when the inventory falls below the maximum level). From the results of the simulations, it is concluded that the pattern-production control system requires less inventory than the fixed-fill Kanban system to achieve a similar level of service. It would be worthwhile for future research to examine these conclusions within the analytical framework that we have considered.

Finally, we should note that the dynamic behavior of these protocols should be examined. Under the unstructured demands that we considered, the buffer sizes and reorder points determined for the single machine Signal Kanban protocol are both given by Theorem 1. However, simulation results show that for simple two-product systems, given demand information for the different products, the fixed-fill variation of the Signal Kanban policy performs better (lower average inventory) than the fixed-batch variation of the policy [5]. This appears to be due to long-term cyclic behaviors that can occur in the fixed-batch policy. Further investigation of these behaviors and the dynamic behavior of the pattern production system should be done.

In this paper, we did not consider the behavior of signal kanbans oper-

ating over a set of machines in series. One series arrangement commonly found in industry has two machines and consequently two signal kanbans for each product, with each kanban having a separate reorder point. The signal kanban with the earliest reorder point is called a material requisition kanban [9]. This kanban requests the first machine in the series to produce material for the second machine in the series, which then uses it when the its signal kanban is released. The behavior of serial kanban systems is a subject of current research, and some initial results are presented in [13].

References

[1] M. S. Akturk and F. Erhun, *An overview of design and operational issues of kanban systems,* International Journal of Production Research, vol. 37(17), pp. 3859-3881, (1999).

[2] A.M. Bonvik, C.E. Couch, and S.B. Gershwin, *A comparison of production-line control mechanisms,* International Journal of Production Research, vol. 35(3), pp. 789-804, (1997).

[3] C. Chase, J. Serrano, and P.J. Ramadge, *Periodicity and chaos from switched flow systems: contrasting examples of discretely controlled continuous systems*, IEEE Transactions on Automatic Control, Vol 38(1) (1993).

[4] S. E. Elmaghraby, *The economic lot scheduling problem (ELSP), review and extensions*, in Management Science, Vol 24(6), (1978).

[5] L. E. Holloway, *Modeling and Simulation of Manufacturing Systems under Signal Kanban Policies*, 2nd International Symposium on Scale Modeling, Lexington, Kentucky, May 1997.

[6] A.F.P.C. Humes and C. Humes Jr., *A clearing round-robin-based stabilization mechanism*, in Proc. Allerton–94, Allerton (IL), 1994.

[7] P.R. Kumar and T.I. Seidman, *Dynamic instabilities and stabilization methods in distributed real time scheduling of manufacturing systems,* IEEE Trans. Autom. Control **AC-35**, pp. 289–298 (1990)

[8] C. D. Lewis, *'Scientific inventory control'*, American Elsevier, New York, 1970.

[9] Yasuhiro Monden, *'Toyota Production System'* (2^{nd} edition), Inst. Industrial Engineers Press, Norcross (GA), 1993.

[10] J.R. Perkins and P.R. Kumar, *Stable distributed real-time scheduling of flexible manufacturing/ assembly/ disassembly systems*, IEEE Trans Autom. Control **AC-34**, pp. 139–148 (1989).

[11] T.I. Seidman, *'First Come, First Served' can be unstable!*, IEEE Trans. Autom. Control **AC-39**, pp. 2166–2171 (1994).

[12] T.I. Seidman and L.E. Holloway, *Stability of a 'signal kanban' manufacturing system,* in *Proc. 1997 Amer. Control Conf.* (vol. 1), pp. 590–594, Amer. Automatic Control Council, Evanston (1997).

[13] T.I. Seidman and L.E. Holloway, *Stability of signal kanbans for machines in series,* paper in progress.

[14] T.I. Seidman and C. Humes, Jr., *Some kanban–controlled manufacturing systems: a first stability analysis,* IEEE Trans. Autom. Control, **AC-41**, pp. 1013–1018 (1996).

[15] Shigeo Shingo, *A Study of the Toyota Production System from an Industrial Engineering Viewpoint, Revised Edition.* Productivity Press, Portland Oregon, (1989).

[16] James P. Womack, Daniel T. Jones, and Daniel Roos, *The machine that changed the world: the story of lean production.* Harper Collins, New York, (1990).

[17] Kum Khiong Yang, *A comparison of reorder point and kanban policies for a single machine production system*, Production Planning and Control, Vol. 9(4), pp 385-390 (1998).