
Stability of Real-Time Scheduling Policies In Flexible Manufacturing Systems

Abhay K. Parekh

Abstract

In this report we review recent work on the stability of scheduling policies in manufacturing systems, and present some extensions of this work. Flexible manufacturing systems are modeled as queueing networks, and issues of stability considered for a wide range of scheduling policies. These scheduling policies are distributed and may be implemented in real-time.

The tradeoff between machine set-up time and stability is examined in detail for *single machine* systems, and it is shown that a large class of policies, called Clear-a-Generalized Fraction, is stable. Next, stable idling policies are found to minimize average buffer levels in certain *lightly loaded* systems. The decision to idle is found to be independent of the set-up times (when all of the set-up times are identical). A lower bound on the average buffer level of any stable policy is presented and discussed. Candidates for good scheduling policies on single machine systems are examined.

Next, networks of machines are considered, and the starvation or under-utilization of machines is identified as an important cause of instability. This is illustrated by examples of *simple* systems. It is found that a way to cope with this phenomenon is to follow a buffer priority scheduling scheme, with the buffers ordered in relation to the flow line. Under reasonable assumptions, due date based policies are found to behave much like the buffer priority scheme, Last Buffer First Served. This assures their stability as well. We find that a number of stable policies are *limited overtaking policies* for which a contractive delay estimate property holds.

This work was supported by a Vinton Hayes Fellowship.

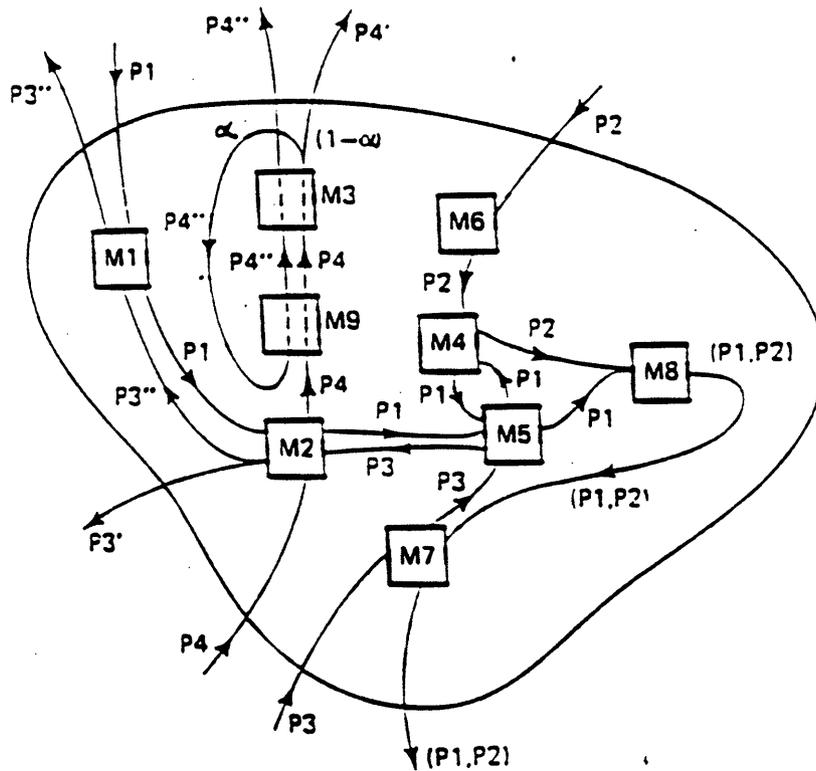


Fig. 1. An example of a general system.

1 Introduction

Consider a factory (manufacturing system) that produces a variety of products from a number of different raw materials. A given product type is manufactured out of parts, by a series of machines that are accessible to each other through a transportation network (conveyer belts, for example). Certain machines may combine part types (i.e. assemble them) and others may take them apart (disassemble them). The machines are sophisticated enough to be able to work on more than one part type, and so a number of part types may compete to be scheduled for production on a particular machine. Machines may only work on *one* part type at a time (with the exception of assembly), and require time to set up before switching part types. Arriving parts wait at buffers, usually separated by part type, and are selected for production according to a *scheduling* policy. The goal of this scheduling policy is to ensure that the production rates of the final products track their demand rates closely, and at low cost.

An example of the type of system we have been describing is represented in Figure 1 of [1]. Even a cursory inspection of it reveals that the competition of part-types for machines could result in fairly complicated dynamics within the manufacturing system. It is the object of this report to understand these dynamics in some detail, through a review of [1], [2], and [3].

We make the following assumptions:

- (1) The number of machines, and the underlying transportation network are fixed.
- (2) There are no machine failures or maintenance periods. This may be reasonable for systems in which failures and maintenance periods occur very infrequently compared to machine operations (see Section 2.1). Thus one could argue that the model applies to the (long) periods of time when there are no failures or maintenance periods. However, when a machine failure, *does* occur, it may last for a long time, and have deleterious effects on other machines (for example, they may become seriously underutilized), and lead to instability. A scheduling policy should ideally be designed to adapt to such failures.
- (3) Processing and set-up times are fixed. If the machines are well maintained, and all the part types of a given part type are (almost) identical, this seems like a good assumption.
- (4) The transportation delays along the links of the transportation network are bounded.
- (5) The maximum inflow of part i in any finite interval of size Δ is bounded by a known function of Δ . Thus the inflow is "deterministically well behaved." This

assumption is reasonable since we are concerned with issues of *stability in the worst-case*, as opposed to the average case.

- (6) It is desirable for the scheduling policies to be *distributed*. By this we mean that machines implement such policies autonomously, and do not communicate with each other. This is a good assumption, if in fact, communication among machines would take a long time, and increase the set-up times significantly.
- (7) It is desirable for a machine to make its next scheduling decision based solely on *current* buffer levels (at that machine), rather than on prior buffer levels. Again, this makes sense only if such computation would slow down the machines considerably.

Suppose the system is empty at time 0, and let $u_p(t)$ be the amount of part type p inflow into the system in the interval $[0, t]$. Also, let $y_p(t)$ be the cumulative outflow of part type p in the interval $[0, t]$. Then we define the manufacturing system to be *stable* if there exists a constant, M_p , for every incoming part type such that

$$u_p(t) \leq y_p(t) + M_p, \quad \forall t \geq 0. \quad (1.1)$$

This condition implies that in a stable system all the buffer sizes will be bounded. Also, since we have assumed bounded delays on the transportation links, only a finite amount of material can be in transit between any two connected machines at any given time. Thus bounded buffer sizes implies (1.1) as well.

In this report we will examine a number of policies from the standpoint of stability. It is clear that stability is a very basic property of a good scheduling policy, since an unstable policy can result in unbounded buffer sizes, and therefore unbounded delay. (Moreover, it is not feasible to build systems with unbounded buffer sizes!) Understanding the dynamics that lead to instability will be seen to be quite difficult; yet by examining the work of [1], [2] and [3], we will get considerable insight into the nature of the problem. The rest of the report is organized as follows:

In the next section we put the objectives mentioned above into context from three different perspectives: Gershwin's hierarchical flow control framework for scheduling in manufacturing systems; traditional scheduling in queuing networks; and flow control in high speed data networks. Our treatment of these perspectives will be brief and thus quite incomplete.

In Section 3 we examine the work of Perkins and Kumar [1], and Chase and Ramadge [2] for a single machine system. In such systems, the set-up times are the only interesting parameters in determining stability. We begin by proving that a fairly large class of policies, known as Clear-a-Generalized Fraction policies is stable. We establish

upper bounds on the total buffer size that match those of Theorem 1 in [1], when specialized to the class of Clear-a-Fraction policies considered in that theorem. Next, we attempt to understand the phenomenon of "idling", (the importance of which appears to have been overlooked by Perkins and Kumar in [1]), by analyzing a simple two part type example. The lower bound result of Chase and Ramadge for the maximum total buffer size of any stable policy is then explained and interpreted. Finally we present some observations on good scheduling policies for single machine systems in line with those of Section IV of [1] and Section 2.2 of [2].

In Section 4 we extend our manufacturing system to many machines. After dispensing with the easy case of acyclic networks, we try to understand the phenomenon of starvation, or underutilization in non-acyclic networks, through a few simple examples. After establishing that set-up times are not crucial to this phenomenon, we eliminate them completely, and follow the very interesting work of Lu and Kumar in [3]. Our model in this section is somewhat limited since there is only one part type, and no rework, assembly or disassembly is allowed. Buffer Priority and Due Date based policies are examined and their stability established. Our treatment of this material is somewhat different than in [3], although we provide no original results. Our approach is to emphasize the similarities among the various policies considered by specifying properties that they share. In particular, we show that they are all *limited overtaking policies*, and that any such policy for which the contractive delay estimate property of [3] holds, must be stable.

In Section 5 we close with a few remarks on the limitations of the work reviewed, and with some suggestions for further work.

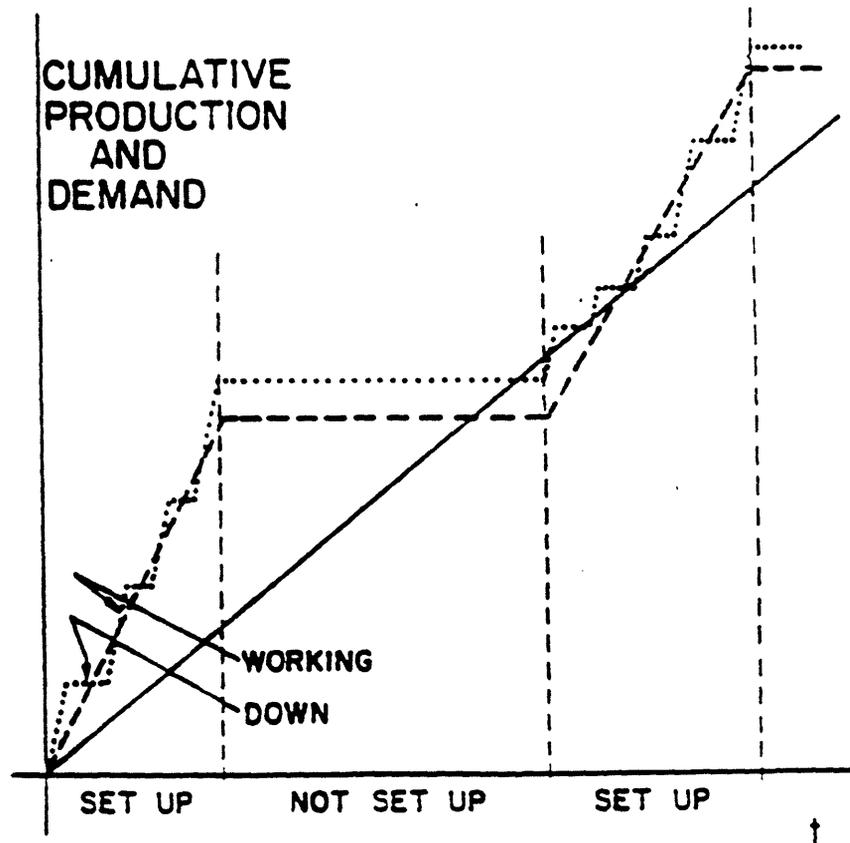


Fig. 1. Production and other events.

2 Background and Perspective

2.1 Hierarchical Framework for Scheduling

In [4] Gershwin proposes a hierarchical framework for scheduling in manufacturing systems, in which the levels of the hierarchy correspond to “classes of events that occur with distinct frequencies.” Events at higher levels occur less frequently than those at lower levels. For example, the event of a machine failure would be at a lower level than is the event of a machine operation. An important assumption of this model is that the frequencies characterizing the events on two different levels in the hierarchy are *well separated*.

Suppose we wish to schedule events on level l of the hierarchy. Events that occur much less frequently than those on l are treated as static, or slow varying. On the other hand, events that occur much more frequently than those on l may be treated as continuous variables, or in any other manner that ignores their (fast) variation.

For example, when fixing long term production rates, failures may be approximated by fixed (time) average rates. Similarly, when scheduling machine operations on parts, the desired production rate may be viewed as constant.

The hierarchy may be better understood through the figure on the left (Figure 1 of [4]), which illustrates the demand and production curves for a part type p that is one of many part types sharing a machine m . The solid line represents (the integral of) the desired long term production rate, r^1 of parts of type p on m . This production rate cannot be maintained at all times, since m is not set up for p during certain time intervals, and so when m is working on parts of type p , it must do so at a higher rate than r^1 so as to approximate the long-term schedule. But the dashed line does not take into account the possibility of failures, during which no production of p can take place. This results in the dotted line that approximates the dashed line. As explained in [4], the actual cumulative production graph represents times at which parts are loaded on the machine, and has resolution too fine to be visible on the graph.

The model described in Section 1 deals with the lower levels of the hierarchy since it is concerned with the events of loading and transporting material through the manufacturing system.

2.2 Some General Approaches to Scheduling

Much research has been done on the scheduling of jobs to machines, and we will not even attempt to survey it comprehensively. In static models of scheduling all the jobs to be processed arrive simultaneously and the schedule must assign jobs to machines (subject to precedence constraints) so as to minimize quantities such as lateness, makespan etc. The optimization of these quantities often leads to computationally intractable combinatorial formulations (i.e. NP-complete problems).

In dynamic models, such as the one considered here, the parts arrive according to some random process. In most approaches the arrivals are modeled as stochastic processes and the objective is to minimize the *expected* lateness, makespan etc. In this report, we assume that the inflows follow assumption 5 of Section 1, and thus we eliminate a number of sample paths that, while extremely unlikely, could still lead to instability. This allows us to examine traditional criteria from a worst-case point of view.

In [3] Lu and Kumar refer to a stochastic queueing network model proposed by Kelly in chapter 3 of his book [5]. In this model, the customers are differentiated by type according to the route they follow through the network. Priority schedules are permissible only *within* a particular type, and must be based on the position of the customer in the queue. Further, while customers may revisit servers, their mean service times upon each visit will be the same. In Section 4 we will use a fixed route model that allows different service times for each revisit, and we will examine a number of priority rules that allow priority *by* type of customer (i.e. part).

2.3 Relevance to Flow Control in High Speed Data Networks

In a data network, the objective of flow control is to reduce congestion in the network by regulating fairly, the input traffic of a number of unpredictable, bursty users. With the advent of high speed networks, in which propagation delay has become an important factor, it has become necessary to use flow control schemes that do not entail much end to end communication. This has led to a burgeoning of interest in directly regulating the input of traffic, as opposed to using window-based schemes. In implementing any such scheme it is important to limit the burstiness, or departure from expected rate of a source. A model of traffic that accomplishes this is the one we described briefly in assumption 5 of the introduction, and which we will elaborate on in Section 4.4. There, we will see that the model used by Lu and Kumar in [3] has also been recently used by Cruz [6][7] to study worst case delay in fixed route data networks.

A significant point of departure between data networks and manufacturing networks is that routes never revisit a node of a data network, while certain semiconductor manufacturing systems have highly re-entrant routes [3]. Also, set up times are not relevant in current models of packet networks.

3 The Single Machine System

Consider a single machine system consisting of P incoming part types. There is no assembly or disassembly, so every incoming part corresponds to an outgoing part type. Also, no parts revisit the machine, i.e. there are no loops in the transportation network. Let τ_p be the time taken for the machine to process one unit of part type p flow. Parts of type p are stored in buffer b_p and the size of b_p at any time t is $x_p(t)$. For most of our results we assume that the parts arrive at a constant rate, d_p for part type p (although the results continue to hold for more general arrival streams as well). Let $\rho_p = d_p \tau_p$. Then a necessary condition on stability is

$$\rho = \sum_p \rho_p < 1. \quad (3.1)$$

In the following we assume that (3.1) always holds. If all set-up times are zero, stability is assured for any policy. Thus the interesting parameters in this section are the set-up times. Let δ be the time taken to switch from one part type to another, for all part types. This assumption simplifies the analysis, however, the results of this section will apply to systems with different set-up time values as well.

A *clearing scheduling policy* is one in which the machine continues to serve a buffer until it has completely cleared that buffer of work. The amount of work in a buffer, b_p , at time t is

$$w_p(t) = x_p(t) \tau_p.$$

Also, the total amount of work in the system at time t is

$$w(t) = \sum_p w_p(t).$$

For any policy, let $T(n)$ be the time that the machine begins setting up to serve a new part type for the n^{th} time, and let this part type be $P(n)$. Note that $P(n) \neq P(n+1)$.

The scheduling policy is *nonidling* if the machine switches to another part type as soon as a buffer has been cleared. Thus in a non-idling policy, buffer $b_{P(n-1)}$ is cleared at time $T(n)$.

In an *idling* policy the machine may continue to serve a part type at its incoming rate, thus operating at less than maximum rate.

A *feasible* scheduling policy is one that never deadlocks. A trivial example of a scheduling policy that is not feasible is a policy that requires: $x_{P(n)}(T_n) > \frac{2}{P} \sum_p x_p(T_n)$, for all n .

On the other hand the clearing policy *Least WorkFirst*, which always picks the buffer with the least work in it is a feasible clearing policy.

A *Clear-a-Generalized Fraction Policy (CGF)* is any feasible clearing policy, which at time T_n , chooses to produce any part-type $P(n)$, that satisfies:

$$x_{P(n)}(T_n) \geq f\left(\sum_p x_p(T_n)\right)$$

for some fixed, unbounded, monotonic increasing $f : R^+ \rightarrow R^+$. (R^+ is the set of non-negative reals.)

Note that a necessary condition on f (for feasibility) is that $f(x) \leq x$, for all $x \in R^+$.

A special case of CGF policies are the *Clear-a-Fraction Policies (CAF)* considered extensively in [1]. CAF policies are CGF policies with $f(\sum_p x_p(T_n)) = \epsilon \sum_p x_p(T_n)$, for some fixed $\epsilon > 0$.

Another interesting special case of CGF polices is the class of policies characterized by $m > 1$, and

$$\begin{aligned} f(y) &= \frac{1}{P} y^m, \quad 0 \leq y \leq 1 \\ &= \frac{1}{P} y^{\frac{1}{m}}, \quad y \geq 1. \end{aligned}$$

3.1 Stability Results

In this section we analyze the stability of the scheduling policy classes outlined earlier. We would like to understand “how bad” a policy has to be in order to be unstable. For example the policy *Least Work First* appears suicidal, and in fact, can easily be seen to be unstable. Perkins and Kumar show in Theorem 1 of [1] that all CAF policies are stable, and also provide an upper bound on the maximum buffer size (in the limit). However, they point out that this bound is probably “somewhat gross.” This statement, and a desire to understand why a better bound would be hard to get, prompted me to look *closely* at the proof of Theorem 1 of [1]. At a very minimum I hoped to simplify the proof, which appeared to me, to hide exactly what information about the system was being used to get the bound.

It turned out that by simplifying the proof, I was able to prove that the more general class of CGF policies is stable. When specialized to the CAF case, the result yields the same bounds as those of Theorem 1 in [1].

Theorem 1. *Performance of CGF policies:*

- (i) *All CGF policies are stable.*
- (ii) *In particular,*

$$\sum_p w_p(t) \leq \delta\rho + \max\{\tau_{max} f^{-1}\left(\frac{\delta}{1-\rho} \max_p \left(\frac{\rho - \rho_p}{\tau_p}\right)\right), \sum_p w_p(0)\},$$

for all $t \geq 0$, where τ_{min} and τ_{max} are the smallest and largest values of τ respectively.

- (iii)

$$\sum_p x_p(t) \leq \frac{\delta\rho}{\tau_{min}} + \frac{\tau_{max}}{\tau_{min}} \max\{f^{-1}\left(\frac{\delta}{1-\rho} \max_p \left(\frac{\rho - \rho_p}{\tau_p}\right)\right), \sum_p x_p(0)\},$$

for all $t \geq 0$, where τ_{min} and τ_{max} are the smallest and largest values of τ respectively.

Proof: The key idea of the proof is the following: Suppose we start observing the system at time T_n . This system is indistinguishable from a system that starts out with $x'_p(T_0) = x_p(T_n)$, for all p .

We will first show that the total work to the machine is always decreased at T_1 whenever $w(T_0)$ is above a certain threshold, K , that depends on the input traffic and the CGF policy function, f .

$$w(T_1) - w(T_0) \geq 0 \Rightarrow w(T_0) \leq K. \tag{C-1}$$

A. K. PAREKH

$$K = \tau_{max} f^{-1}\left(\frac{\delta}{1-\rho} \max_p \left(\frac{\rho - \rho_p}{\tau_p}\right)\right).$$

Next, we will show that if the total work is ever below this threshold, K , then it will continue to remain below K for subsequent set-up times, T_n . i.e.

$$w(T_0) \leq K \Rightarrow w(T_1) \leq K. \quad (C-2)$$

These two steps will give us an upper bound on $w(t)$, the total work to the machine at time t . We will use the fact that

$$w(t) \leq w(T_n + \delta) = w(T_n) + \rho\delta, \quad \text{for } T_n \leq t \leq T_{n+1}.$$

Finally, we will relate $w(t)$ to the total buffer level at time t to get the result.

(C-1): As pointed out earlier, the only time that the system is not serving parts is when it is setting up. Thus,

$$w(T_n) = w(T_0) + n\delta - (1-\rho)(T_n - T_0).$$

If $n = 1$ and $T_0 = 0$,

$$w(T_1) = w(0) + \delta - (1-\rho)T_1. \quad (A)$$

Thus

$$w(T_1) - w(0) \geq 0 \iff \delta \geq (1-\rho)T_1.$$

Also, since the policy is clearing,

$$T_n = T_{n-1} + \frac{\delta + \tau_{P(n-1)}x_{P(n-1)}}{1 - \rho_{P(n-1)}},$$

which is just (5) of [1]. Again, if $n = 1$ and $T_0 = 0$,

$$T_1 = \frac{\delta + \tau_{P(0)}x_{P(0)}}{1 - \rho_{P(0)}}. \quad (B)$$

From (A) and (B):

$$w(T_1) - w(T_0) \geq 0 \iff \frac{\delta}{1-\rho} \geq \frac{\delta + \tau_{P(0)}x_{P(0)}}{1 - \rho_{P(0)}}$$

Thus

$$x_{P(0)} \leq \frac{\rho - \rho_{P(0)}}{1-\rho} \frac{\delta}{\tau_{P(0)}} \quad \text{and} \quad f^{-1}(x_{P(0)}) \leq f^{-1}\left(\frac{\rho - \rho_{P(0)}}{1-\rho} \frac{\delta}{\tau_{P(0)}}\right)$$

Stability in Manufacturing Systems

(since f is unbounded and f^{-1} is monotonically increasing). Now by assumption,

$$\sum_p x_p(T_0) \leq f^{-1}(x_{P(0)}).$$

This yields:

$$\sum_p x_p(T_0) \leq f^{-1}\left(\frac{\rho - \rho_{P(0)}}{1 - \rho} \frac{\delta}{\tau_{P(0)}}\right).$$

But

$$f^{-1}\left(\frac{\rho - \rho_{P(0)}}{1 - \rho} \frac{\delta}{\tau_{P(0)}}\right) \leq \max_p f^{-1}\left(\frac{\rho - \rho_p}{1 - \rho} \frac{\delta}{\tau_p}\right) \leq f^{-1}\left(\frac{\delta}{1 - \rho} \max_p\left(\frac{\rho - \rho_p}{\tau_p}\right)\right). \quad (*)$$

Notice that $w(T_0) \leq \sum_p \tau_{max} x_p = \tau_{max} \sum_p x_p$, to get the result

$$w(T_0) \leq \tau_{max} f^{-1}\left(\frac{\delta}{1 - \rho} \max_p\left(\frac{\rho - \rho_p}{\tau_p}\right)\right) = K.$$

This shows (C-1).

(C-2): From (A): (setting $T_0 = 0$):

$$T_1 = \frac{\delta - w(T_1) + w(T_0)}{1 - \rho}.$$

Comparing with (B), and rearranging terms:

$$w(T_1) - w(T_0) = \delta \left(\frac{\rho - \rho_{P(0)}}{1 - \rho_{P(0)}}\right) - \tau_{P(0)} x_{P(0)}(T_0) \left(\frac{1 - \rho}{1 - \rho_{P(0)}}\right)$$

Thus for CGF policies:

$$w(T_1) \leq \delta \left(\frac{\rho - \rho_{P(0)}}{1 - \rho_{P(0)}}\right) + w(T_0) - \tau_{P(0)} f\left(\frac{w(T_0)}{\tau_{max}}\right) \left(\frac{1 - \rho}{1 - \rho_{P(0)}}\right).$$

Now note that since the CGF policies characterized by f must be feasible, it must be that $f(y) \leq y$ for all $y \geq 0$. Thus

$$w(T_0) - \tau_{P(0)} f\left(\frac{w(T_0)}{\tau_{max}}\right) \left(\frac{1 - \rho}{1 - \rho_{P(0)}}\right) > 0.$$

But now we are done, since if $w(T_0) \leq K$ we have:

$$w(T_1) \leq \delta \left(\frac{\rho - \rho_{P(0)}}{1 - \rho_{P(0)}}\right) + K - \tau_{P(0)} f\left(\frac{K}{\tau_{max}}\right) \left(\frac{1 - \rho}{1 - \rho_{P(0)}}\right)$$

A. K. PAREKH

$$\begin{aligned} &\leq K + \delta \left(\frac{\rho - \rho_{P(0)}}{1 - \rho_{P(0)}} \right) - \frac{1 - \rho}{1 - \rho_{P(0)}} \frac{\delta \tau_{P(0)}}{1 - \rho} \max_p \left(\frac{\rho - \rho_p}{\tau_p} \right) \\ &\leq K \end{aligned}$$

From (C-1) and (C-2) we have:

$$w(T_n) \leq \max\{K, w(0)\}, \quad n = 0, 1, 2, \dots$$

But

$$w(t) \leq w(T_n + \delta) = w(T_n) + \rho\delta, \quad \text{for } T_n \leq t \leq T_{n+1}$$

i.e.,

$$w(t) \leq \rho\delta + \max\{K, w(0)\}, \quad t \geq 0. \quad (C)$$

This yields (ii) and therefore (i).

Now note that

$$\sum_p x_p(t) = \frac{1}{\tau_{min}} \sum_p \tau_{min} x_p(t) \leq \frac{1}{\tau_{min}} w(t).$$

Also,

$$\frac{w(0)}{\tau_{min}} = \frac{1}{\tau_{min}} \sum_p \tau_p x_p(0) \leq \frac{\tau_{max}}{\tau_{min}} x_p(0).$$

Substituting in (C) we have (iii).

This completes the proof of theorem 1.

Notice that our bound K is independent of the value of $P(0)$. This is what makes it rather loose. Thus the weakening step is (*) in which take the maximum over all the part types, including the type $P(0)$. From the above reasoning, it should follow that the bound is quite good when all the part types are identical in terms of d_p and τ_p . We will discuss this point more fully in Section 3.4.

By a simple modification of the proof of Theorem 1 (similar to those made in Lemma 6 of [1]), we may show:

Corollary 1. *The results of Theorem 1 hold when the input of every part type p , $u_p(t)$ satisfies, for some $\gamma \geq 0$, and all t :*

$$td_p - \gamma \leq u_p(t) \leq td_p + \gamma.$$

3.2 A Two-Part Example—Example 2.1 of [2]

An effective way to gain some initial insight into manufacturing networks is to examine a simple example in some detail. In particular, we want to understand the tradeoff between set-up time and buffer levels, and the role of idling in minimizing these buffer levels.

The notion that using a machine at less than its maximum allowable rate can reduce buffer levels seems counter-intuitive, especially since the machine is being “wasted” during set-up times anyway. This “intuition” perhaps comes from the fact that in many situations, idling is indeed a terrible thing to do. For example if all the sources are identical, and $\rho \sim 1$, then idling is likely to drive the system into instability. Suppose, however, that there are only two part types, and $\rho_2 = 0$. At time 0 the machine is set up to serve buffer 2, and after clearing the initial buffer level, i.e. $x_2(0)$, it sets up to serve buffer 1. Clearly, it doesn’t make any sense to ever switch back to serve part type 2. Thus, in this case, idling is the *obvious* thing to do. Now consider the same situation, but with ρ_2 slightly larger than 0. As buffer 1 is being served, parts of type 2 arrive very slowly, and so when buffer 1 is cleared, hardly any work has accumulated in buffer 2. Switching to buffer 2, clearing it in almost 0, time and then switching back to buffer 1 entails a time of at least 2δ in which negligible work has been done. On the other hand, idling at buffer 1 is better, since the machine can work at a rate of about ρ if it does so. After a while, buffer 2 will grow significantly, and it is *then* that it becomes advantageous for the machine to switch.

To be more concrete, let’s look at an example. This example is example 2.1 in [2]. The treatment of this example is rather cryptic in [2] and so we’ll fill in some of the gaps.

Suppose there are two parts and we allow idling on the first part-type. Also, suppose that the policy is clearing. We can choose initial buffer levels so that the buffer level trajectories are periodic for both part-types. (This should be clear from the figure.)

Thus, there is a period T such that $x_i(a) = x_i(a + T)$, for $a \geq 0$, $i = 1, 2$.

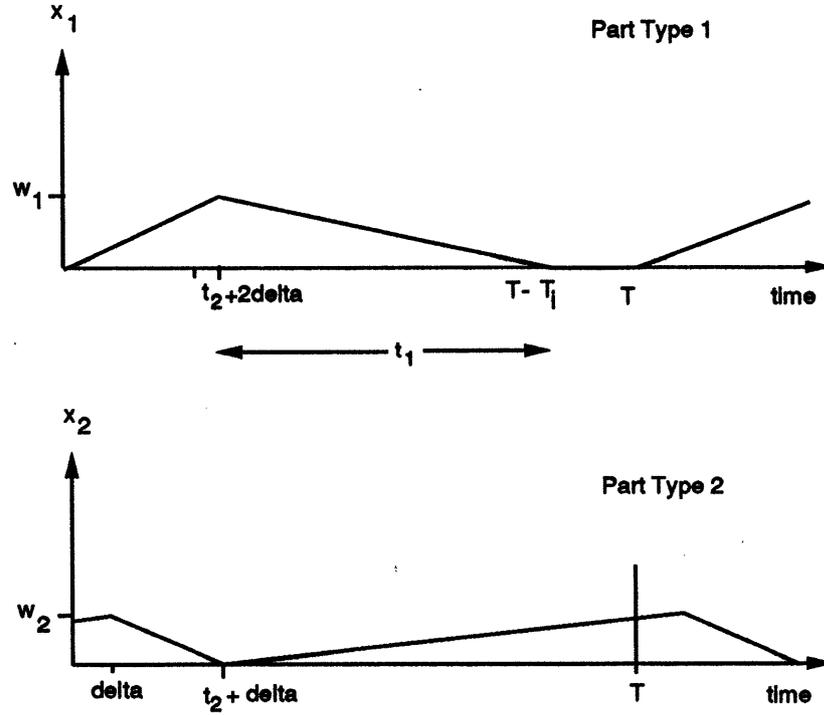


Figure 1. Idling in a 2-Part System.

Notice that t_p is the time devoted to clearing the part p buffer (at rate $d_p - \tau_p^{-1}$), in one time period. Also, T_I is the time the machine spends idling on part type 1 in one time period. Also:

$$T = 2\delta + t_1 + t_2 + T_I$$

$$x_1(T) = x_1(0) + d_1T - d_1T_I - t_1\tau_1^{-1} \quad (3.2.1)$$

$$x_2(T) = x_2(0) + d_2T - t_2\tau_2^{-1}. \quad (3.2.2)$$

From (3.2.1) and (3.2.2) and the definition of T :

$$t_1 = \rho_1(T - T_I), \quad t_2 = \rho_2T. \quad (3.2.3)$$

Thus:

$$T = \frac{2\delta + (1 - \rho_1)T_I}{1 - \rho}$$

Now notice that

$$w_1 = (2\delta + t_2)d_1$$

$$w_2 = (2\delta + t_1 + T_I)d_2.$$

Suppose we want to minimize some weighted time average of the buffer levels. i.e. we want to determine

$$B = \liminf_{t \rightarrow \infty} \frac{1}{t} B(t) = \liminf_{t \rightarrow \infty} \frac{1}{t} \int_0^t \left[\sum_p \gamma_p \tau_p x_p(s) \right] ds$$

From the figure it is easy to see that the average weighted buffer level B is just

$$B = \frac{1}{T} \left(\frac{1}{2} \gamma_1 \tau_1 w_1 (T - T_I) + \frac{1}{2} \gamma_2 \tau_2 w_2 T \right).$$

As explained on page 4 of [2] B is convex in T_I can be minimized over T_I by just taking derivatives wrt T_I . However, the authors of [2] do not reveal the outcome of this minimization. After some crunching I got:

$$T_I = \frac{2\delta(L - M)}{\rho_1(1 - \rho_1)(\rho_2^2 + (1 - \rho_1)(1 - \rho_2)K)}, \quad (3.2.4)$$

where

$$L = (1 - \rho) \sqrt{K(1 - \rho_1)(1 - \rho_2) + 1}, \quad M = \rho_2 + K(1 - \rho_1)(1 - \rho_2).$$

and

$$K = \frac{\rho_2 \gamma_2}{\rho_1 \gamma_1}.$$

Notice that the amount of idling depends linearly on the set up time, δ . From (3.2.4) we see that idling reduces $B \iff L > M$ i.e.

$$(1 - \rho) \sqrt{K(1 - \rho_1)(1 - \rho_2) + 1} > \rho_2 + K(1 - \rho_1)(1 - \rho_2)$$

or

$$\frac{\rho_2 + K(1 - \rho_1)(1 - \rho_2)}{\sqrt{1 + K(1 - \rho_1)(1 - \rho_2)}} < (1 - \rho) \quad (3.2.5).$$

Figure 2 shows the combinations of ρ_1 and ρ_2 for which idling is beneficial. $g = \frac{\gamma_2}{\gamma_1}$.

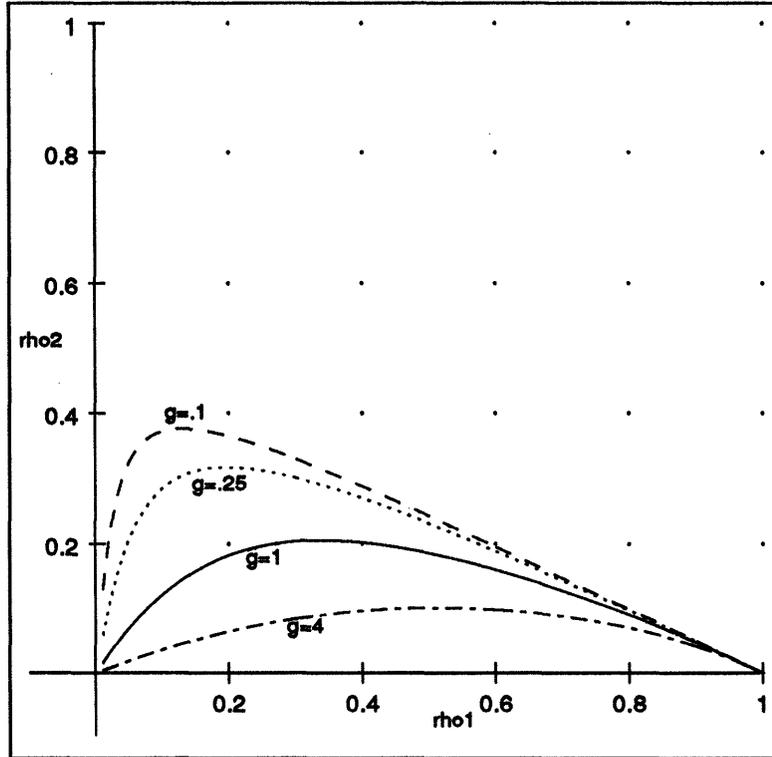


Figure 2. Idling Regions for various buffer weights.

The curves in this figure reveal the tradeoffs between the ρ_i 's and T_I . Consider the curve for $g = 1$, in which both buffers are given equal priority by the machine. Idling occurs only for small values of ρ_2 , which is not surprising since time spent idling on part type 1, is time that could have been spent serving part type 2 (minus set up times).

Also, the curve indicates that when ρ_1 is either very small or very large, ρ_2 must be particularly small for idling to be beneficial. It is easy to see that idling on part type 1 is not attractive when ρ_2 is much larger than ρ_1 . On the other hand, if ρ_1 is large, the time spent clearing buffer 1 will be correspondingly large. (This is because of (3.2.3)) So unless ρ_2 is very small, a significant amount of type 2 parts will have arrived in the interim, and idling will not be an attractive option.

Comparing the different curves, we see that as g increases, the Idling region expands, but the increase is the most for small values of ρ_1 . This is because idling in a system with ρ close to 1 will be driven to instability.

3.3 Lower Bounds on the Average Buffer Levels

While Theorem 1 shows that a number of policies are stable, our upper bounds are rather weak for cases in which the sources are not identical, and start off with initial buffer levels much smaller than K . This means that we still do not have much insight into the trade-off between set-up times and buffer levels. However, if we focus on average weighted buffer levels, i.e. the quantity

$$B = \liminf_{t \rightarrow \infty} \frac{1}{t} B(t) = \liminf_{t \rightarrow \infty} \frac{1}{t} \int_0^t [\sum_p \gamma_p \tau_p x_p(s)] ds = \liminf_{t \rightarrow \infty} \frac{1}{t} \int_0^t [\sum_p \gamma_p w_p(s)] ds$$

then we can find very good lower bounds. In addition, the proofs of these bounds in [1] and [2] are almost constructive in nature, and provide considerable information on how to construct good scheduling policies.

The most general result of this form is by Ramadge and Chase [2] for stable policies with idling. As mentioned earlier, the machine is said to be idling if it does not switch part types after clearing a buffer, but continues to serve the current part type, p , at rate d_p . Perkins and Kumar give lower bounds for non-idling policies, and it appears that they overlooked the possibility of idling altogether. Figure 1 of [2] shows that idling can reduce B to the extent that it violates the lower bound given in [1], although this is not surprising given our study of the example in Section 3.2.

Ramadge and Chase go on to present a lower bound on B that *does* take idling into account. It is useful to examine the ideas used in deriving this bound, since the bound appears to be quite tight.

Consider a stable policy \mathcal{S} , and pick an interval $[0, T]$. Let n_p be the number of part p production runs in the interval, and let T_p be the time taken to do those runs. The trajectory $x_p(t)$ is piece-wise linear, and consists of a sequence of *buffer build-up*, *buffer depleting* and *idling* phases. A buffer build-up consists of a linear segment of slope d_p , and a depleting phase consists of a segment of slope $d_p - \tau_p^{-1}$. An idling phase consists of a segment of slope 0. There are clearly n_p depleting phases. Finally, since the policy is stable, there exists a c such that $x_p(t) \leq c$ for all $t \geq 0$.

Now treating n_p and T_p as given, and ignoring inter-part dependencies, Ramadge and Chase find a trajectory $\bar{x}_p(t)$ that has the above properties, and is such that the

area under the trajectory is minimized. By series of appealing geometric arguments they show that such a \bar{x}_p has the form of Figure 8 of [2]:

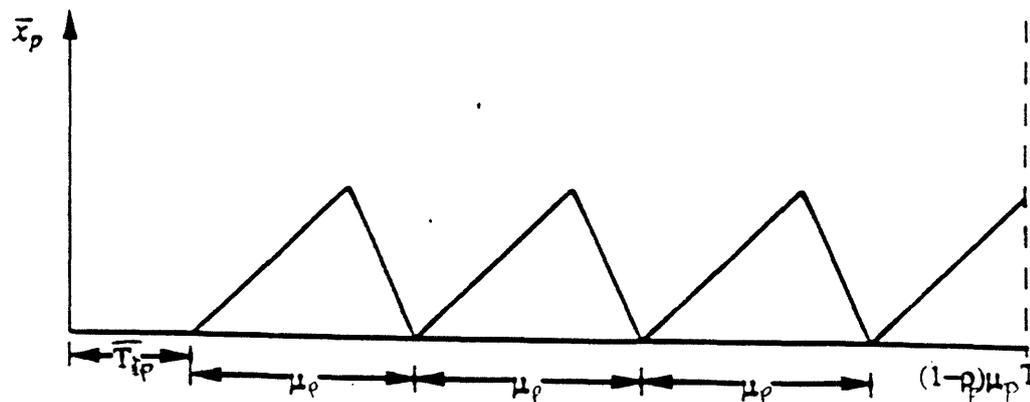


Fig. 8. Qualitative form of $\bar{x}_p(\cdot)$.

Note that \bar{T}_{I_p} is the total length for which the slope of $\bar{x}_p(t)$ is zero. The area under this curve is easily lower bounded, enabling us to find a lower bound for $w_p(t)$, the part type p work at time t . By weighting w_p by γ_p and summing over all p we get a lower bound, $\bar{B}(t)$ on $B(t)$,

$$\bar{B}(t) = \sum_p \frac{\gamma_p \rho_p (1 - \rho_p) (T - \bar{T}_{I_p})^2}{2T(n_p + 1 - \rho_p)}.$$

We now need to account for the fact that the machine can work on only one part-type at a time, and must incur a set-up time of δ when switching part types. This is obtained in (5) of [2]:

$$\delta n_p + \sum_p \bar{T}_{I_p} (1 - \rho_p) \leq (1 - \rho)T + \delta + \hat{c},$$

where $\hat{c} = \sum_p c\tau_p$.

(This is a generalized form of the corresponding constraint obtained by Perkins and Kumar, (13) of [1].)

The next step is to minimize $\bar{B}(t)$ subject to the above constraint. The minimization is set up on page 12 of [2]. There are now two sets of non-negative variables to minimize, i.e. the n_p 's and the \bar{T}_{I_p} 's. The Kuhn Tucker conditions, which any optimum solution must satisfy are in (6)-(7) of [2]. In the rest of the proof Chase and Ramadge

find all solutions to (6) and (7) that are feasible in the constraint region (extremal points), and then determine which solutions minimize $\bar{B}(t)$.

They find that if all the idle time variables, \bar{T}_{I_p} , are set to zero, there is only one extremal point, \mathcal{P}_1 , and therefore only one solution to the optimization problem exists. This solution matches exactly (as $T \rightarrow \infty$), the solution found in theorem 2 of [1] in which idling was ignored. The optimizing set of n_p 's is of the form:

$$n_p = k\alpha_p - (1 - \rho_p), \quad (3.3.1)$$

where

$$\alpha_p = \frac{\sqrt{\gamma_p \rho_p (1 - \rho_p)}}{\sum_j \sqrt{\gamma_j \rho_j (1 - \rho_j)}},$$

and $k > 0$ is some constant that depends on T , δ , and c .

When the \bar{T}_{I_p} 's are allowed to be greater than 0, only one more extremal point, \mathcal{P}_2 emerges. The question to be resolved then, is when this point yields a smaller value of $\bar{B}(t)$ than does the point \mathcal{P}_1 . Chase and Ramadge find that this occurs only when there exists a part type j such that

$$\gamma_j \rho_j \geq \gamma_p \rho_p \quad (3.3.2)$$

for all p , and

$$\alpha_j > \frac{1}{2} + \frac{1}{2} \left(\frac{\rho - \rho_j}{1 - \rho_j} \right), \quad (3.3.3)$$

The inequalities (3.3.2) and (3.3.3) form an *Idling Condition*, (IC). The condition (3.3.3) is a more clear form of the condition (ii) on page 4 of [2].

Now it is easy to see from (3.3.3) that (IC) can only hold for one part type j . (Notice that $\alpha_j > \frac{1}{2}$, $\sum_j \alpha_j = 1$ and $\alpha_j \geq 0$ for all j .) Also, (3.3.3) can only hold if

$$\rho < \frac{1 + \rho_j}{2}.$$

When (IC) holds for a part type j the optimizing n_p 's have the form:

$$n_p = \bar{K} \frac{\alpha_p}{\alpha_j} - (1 - \rho_p), \quad p \neq j, \quad (3.3.4)$$

and

$$n_j = \bar{K}(1 - \alpha_j) + \frac{T}{\delta}(\rho - \rho_j) + \theta_1, \quad (3.3.5)$$

for some constants θ_1 , and $\bar{K} > 0$.

The optimizing \bar{T}_{I_j} is found to be (see (10) of [2]):

$$\bar{T}_{I_j} = T - \frac{2\delta}{1 - \rho_j}(n_j + 1 - \rho_j).$$

Finally, by substituting these solutions into the expression for $\bar{B}(t)$ and determining its average value as $T \rightarrow \infty$, we have Theorem 1 in [2]:

Theorem 2. Let $\{\gamma_p\}$ be a set of positive weighting factors. For any stable policy:

$$B = \liminf_{t \rightarrow \infty} \frac{1}{t} B(t) \geq B^*$$

where

$$B^* = \frac{2\delta}{1 - \rho_j} \gamma_j \rho_j (1 - \rho_j) \left(\alpha_j^{-1} - \frac{1 - \rho}{1 - \rho_j} \right)$$

if the idling condition (IC) (3.3.2)–(3.3.3) holds for j . And

$$B^* = \frac{\delta}{2(1 - \rho)} \left(\sum_p \sqrt{\gamma_p \rho_p (1 - \rho_p)} \right)^2$$

otherwise.

Assuming that this bound is tight, we can estimate the benefits of idling when (IC) holds for part type j . In this case we find that the difference between optimal idling and non-idling policy to be of the form:

$$\Delta B = \frac{\delta}{2(1 - \rho)} \left(\sum_p \sqrt{\gamma_p \rho_p (1 - \rho_p)} - 2 \left(\frac{1 - \rho}{1 - \rho_j} \right) \sqrt{\gamma_j \rho_j (1 - \rho_j)} \right)^2,$$

which is seen to be proportional to δ and

$$\alpha_j - \frac{1}{2} - \frac{1}{2} \left(\frac{\rho - \rho_j}{1 - \rho_j} \right).$$

•

3.4 Devising Good Policies

In this section we attempt to use the insight gained in the earlier sections to devise good policies, i.e. those that have small values of B for single machine systems.

There are three reasons for assuming that the bound in Theorem 2 is tight:

- (1) Figure 4 of [2] shows the performance of the periodic idling schedule we studied in Section 3.2 versus the lower bound of theorem 2. The bound does remarkably well for the chosen values of γ_i and ρ_i , $i = 1, 2$.
- (2) We derived our own Idling Condition (3.2.5) which is compared to (IC) in the figure below:

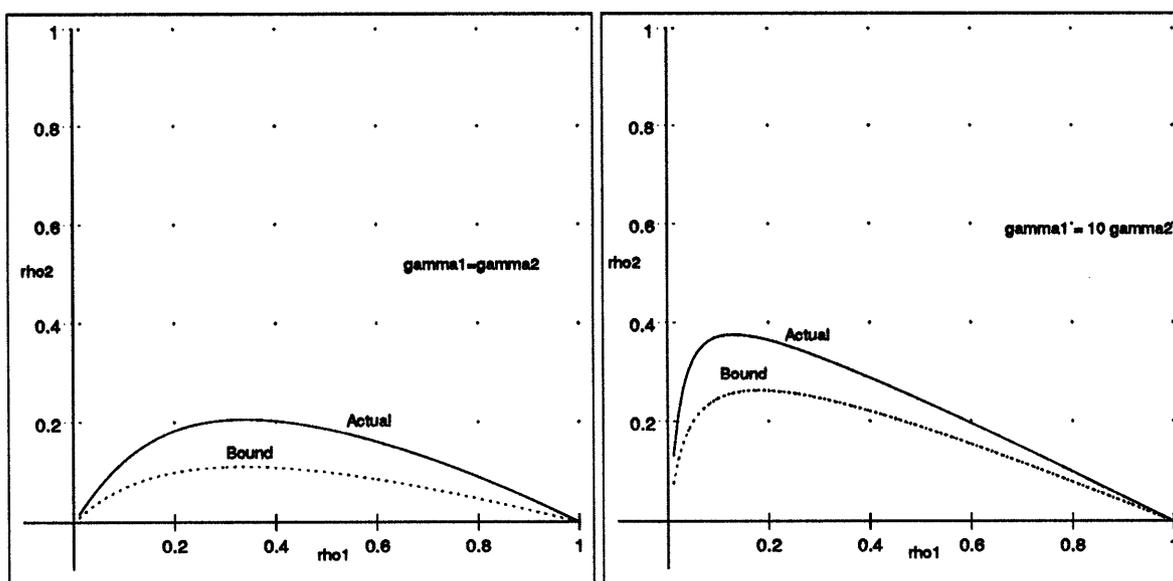


Figure 3. Comparison of Actual to Predicted Idle Regions.

The Figure above suggests that for decreasing values of g , the area in the difference of the two curves goes to zero. Even for $g = 1$ we see that both curves have essentially the same form.

- (3) Recall the observation in Section 3.1, that the bounds of Theorem 1 are tight when $\rho^* = \rho_p$ for all p . We can check that the lower bound is tight in this case as well. For example, pick $\tau_p = 1$ for all p , and assume that the initial buffer levels are small. Our scheduling policy is a CAF policy with $\epsilon = 1$. From Theorems 1 and 2 we find the difference in the two bounds is just $\delta\rho^*$ which is especially small, since the bound in Theorem 2 is on the *average* as opposed to the maximum buffer level.

While the above arguments do not prove anything, they strongly suggest (to me, anyway), that the bound of Theorem 2 is very tight. It should then be our objective to devise policies that come close to the lower bound:

The average area under the trajectory $\bar{x}_p(t)$ can be calculated from equations in the proof of Theorem 1 in [2], for all p . If the average buffer level for part type p under a scheduling policy is close to this area, the policy will be very good. This is the idea suggested by Ramadge and Chase in Section 2.2 of [2] (and by Kumar and Perkins for the non-idling case, in [IV,1]):

Let T_n be the time of the n^{th} set up, and let the part type selected at T_n be $P(n)$. Also, let the part type for which the idling condition holds, be part type 1. At time $T(n)$ the machine decides to serve the part type which at time $T(n) + \delta$ would have the largest ratio of actual to desired buffer size. The exception to this is when part type 1 has just been cleared. In this case, the machine idles on part type 1 until the actual buffer levels exceed the desired buffer level for at least one part type. Simulation results by Perkins and Kumar for the non-idling case strongly suggest that this policy works very well.

4 Stability of Manufacturing Networks

We will now look at systems of more than one machine. It is here that the problems of assembly, disassembly, rework and transportation come into play. Since the transportation delays are bounded, stability results are essentially untouched by the existence of such delays. However, there is currently a lack of understanding of how to deal with assembly, disassembly, and rework (or proportional routing) in non-acyclic networks, unless Clearing policies with backoff (Section 4.3) are used. We comment more on this limitation in Section 5.1.

4.1 Acyclic Networks

Acyclic networks are those that do not have any directed cycles, i.e. there is no feedback.

Fact 1. *For any acyclic network there exists a labeling of the machines so that parts never move from a machine of higher to lower label.*

Proof: Let the directed graph associated with the acyclic network be denoted G , and define a node that has no incoming arcs to be a start node. There must be at least one start node in G . (Otherwise G will contain a cycle.) Pick any start node, and label it 1. Now the graph $G - \{1\}$ is also acyclic, and must contain a start node as well. Pick such a start node, and label it 2. Continue this process until all nodes are labeled. Now it is easy to see that the Fact holds for this labeling.

In Theorem 5 of [1] this fact is used to show that all feasible CAF policies are stable in acyclic networks. The proof proceeds by induction on the labels of the machines. We know from Theorem 1 that machine 1 must be stable since all the inflows are inputs into the system. Now suppose that machines 1 through $m - 1$ are stable i.e. they have bounded buffer sizes. The part inflows to machine m can only come from these machines. Let $u_{pm}(t)$ be the total of part p into m in the interval $[0, t]$. We have

$$d_p t \geq u_{pm}(t) \geq d_p t - \left(\sum_p x_{pm}(t) + \text{amount of part } p \text{ flow in transit at } t \right).$$

But since delays are bounded by σ , there can be at most $\sum_{n=1}^{m-1} \tau_{pn}^{-1} \sigma$ part type p parts in transit from machine 1 to m . Thus we have

$$d_p t \geq u_{pm}(t) \geq d_p t - \gamma_{pm}, \quad \forall p$$

for some $\gamma_{pm} > 0$. So the inflows to machine m are almost linear i.e. Corollary 1 of Section 3.1 applies. Thus machine m is stable, and the proof is done.

In fact we have shown:

Theorem 4. *All feasible CGF policies are stable in acyclic networks.*

4.2 Non-Acyclic Networks-Examples

The problem of determining stability becomes much more interesting in networks with directed cycles. This permits parts to revisit machines, although there are many interesting non-acyclic networks in which no part revisits a machine.

In order to understand the dynamics involved, consider the following two machine network.

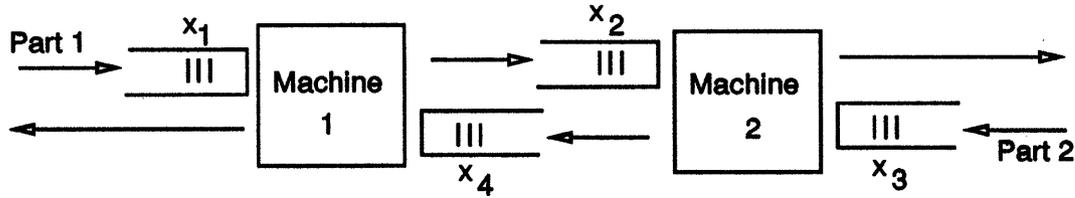


Figure 4. A Potentially Unstable Two Machine Network With No Revisits.

This is the example of Section 3 in [2], and is also considered in [8]. The system starts at time $t = 0$ and the size of buffer b_i , at time t , is $x_i(t)$. We will assume that $\tau_1 = \tau_3 = 0$, and $d_1 = d_2 = 1$. Thus a necessary condition for stability is $\tau_2 < 1$ and $\tau_4 < 1$.

Case 1: Set up times are all zero, $x_2(0) = K$, all other buffers are empty at $t = 0$: Machine 1 begins to serve b_2 at $t = 0$ and clears it at time

$$t_1 = \frac{K}{\tau_2^{-1} - 1}.$$

Now $x_3(t_1) = t_1$, since no type 2 parts can be served while machine 2 is clearing b_2 . At t_1 , machine 2 clears b_3 instantaneously, and so we have $x_4(t_1) = t_1$. Buffer b_4 is cleared at time

$$t_1 + t_2 = t_1 + \frac{t_1}{\tau_4^{-1} - 1} = t_1 + \frac{K\tau_2\tau_4}{(1 - \tau_2)(1 - \tau_4)}.$$

Again, no type 1 parts can be served in the interval $[t_1, t_1 + t_2]$, and so $x_1(t_1 + t_2) = t_2$. Since $\tau_1 = 0$, machine 1 will clear b_1 instantaneously, and we have

$$x_2(t_1 + t_2) = \frac{K\tau_2\tau_4}{(1 - \tau_2)(1 - \tau_4)}.$$

Now observe that all the other buffers are empty at time $t_1 + t_2$, and so we are back to the situation at time 0. The system will be unstable if

$$\frac{\tau_2\tau_4}{(1 - \tau_2)(1 - \tau_4)} > 1 \Rightarrow \tau_2 + \tau_4 > 1.$$

Let us understand why $\tau_2 + \tau_4 > 1$ leads to instability. The problem is that if $x_i(0) > 0$ for any i , then the system is forever constrained to work only on *one part type at a time*. Thus one of the machines is always underutilized, or as Kumar puts in [3], *starved* for parts. One (non-robust) way to combat this starvation is to introduce appropriate set-up times as explained below.

Case 2: Let $x_2(0) = x_4(0) = K$, and we introduce set up times at buffers 1 and 3 (but not at buffers 2 and 4). We set δ_1 so that when the initial buffer level, $x_2(0) = K$ is cleared by machine 2, (at time $K\tau_2$), machine 1 is still setting up to serve b_1 . Similarly, we set δ_3 so that when the initial buffer level, $x_4(0) = K$ is cleared by machine 1 (at time $K\tau_4$), machine 2 is still setting up to serve b_3 .

Thus $\delta_1 + K\tau_4 > K\tau_2$, and $\delta_3 + K\tau_2 > K\tau_4$.

Machine 2 can begin setting up for b_3 at time $K\tau_2$, and can begin serving b_3 at time $\delta_3 + K\tau_2$. Since $\tau_3 = 0$,

$$x_4(\delta_3 + K\tau_2) = x_3(\delta_3 + K\tau_2) = \delta_3 + K\tau_2.$$

Similarly, Machine 1 can begin setting up for b_1 at time $K\tau_4$, and can begin serving b_3 at time $\delta_1 + K\tau_4$. Since $\tau_1 = 0$,

$$x_2(\delta_1 + K\tau_4) = x_3(\delta_1 + K\tau_4) = \delta_1 + K\tau_4.$$

Now if $\delta_3 + K\tau_2 = \delta_1 + K\tau_4 = K$, then the system is periodic—i.e. if

$$\delta_3 = K(1 - \tau_2), \quad \delta_1 = K(1 - \tau_4),$$

then the system is periodic.

As we have seen from Case 2, the stability of this system is quite fragile. The problem is that large set-up times will always drive the system into instability because machines are non-productive during set-ups, but very small set up times may lead to unduly large production cycles, thus exacerbating starvation (as in Case 1).

Thus there may be a very small range of set-up times that ensures stability. Chase and Ramadge [2], view the problem slightly differently, in that they view the initial conditions (i.e. initial buffer levels) as the important parameter. Both views are complementary.

In the remaining sections of the paper, we will try to understand the problem of instability due to starvation (versus instability due to large set up times). Thus Case 1 is of more interest to us. The example below is a “cute” variation on Case 1 since it

consists of a *single* part revisiting machine 1:

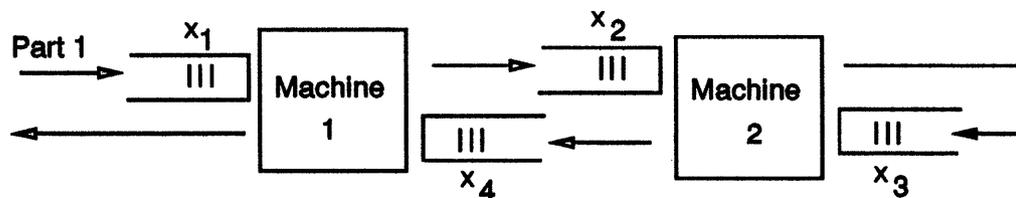


Figure 5. An Unstable Two Machine Network With Revisits.

Here $\tau_1 = \tau_3 = 0$, $d_1 = 1$ and $\tau_2(1 + \tau_4) > 1$. As in case 1, of the earlier example, $x_2(0) = K$ and all the other buffers are empty at time $t = 0$. The analysis is virtually identical to that of Case 1 and is contained in Section 5 of [3].

We have seen that the feedback associated with directed cycles introduces the possibility of instability. Thus any stable policy must be able to circumvent these effects of feedback. Since a clearing policy must continue to serve a buffer as long as it takes it to empty it, it can become stuck in a long production run during which machines are being starved of parts. A scheduling policy that can protect against this phenomenon has more of a chance of being stable.

4.3 Clearing Policies with Backoff

One strategy to deal with feedback information is to ignore it entirely. Clearing policies with backoff [1] adopt this strategy. They work as follows:

Fix a machine type m and a clearing policy, \mathcal{C} that is stable in the single machine case. Suppose that there are b buffers $B_{1,m}, \dots, B_{b,m}$ at m and buffer $B_{i,m}$ stores parts of type $p(i)$. We first suppose this machine is in isolation, i.e. it forms a single machine system of its own, and parts arrive at $B_{i,m}$ according to a constant rate of $d_{p(i)}$. If we apply policy \mathcal{C} to this (imagined) inflow then we know the single machine system is stable. Now let $\{T_n^{p(i)}\}$ be the sequence of times when the machine begins to set up to clear buffer B^i , and $\{\bar{T}_n^{p(i)}\}$ be the sequence of times when m has just cleared $B_{i,m}$. We now define the clearing policy with backoff, given these two sets of sequences of time:

At machine m , for all time t , produce parts of type $p(i)$ if and only if t is in the interval $[T_n^{p(i)} + \delta, \bar{T}_n^{p(i)}]$.

We see that the “goal” of this policy is to try to keep the behavior at each machine as close to its behavior in the single machine case, in the hope of achieving stability. It

is interesting that this goal is, in fact, achieved:

Theorem 5. (*Theorem 8 of [1]*) *All CAF policies with backoff are stable.*

Proof: The idea of the proof is quite clever and is worth examining. It is clever because it can be represented conveniently as a graphical argument (figure 7 of [1]) that is easily grasped. We won't reproduce the argument here, but its essence is to show that if a buffer b accepts parts of type p , then the inflow of parts to that buffer can be bounded below by a line of slope d_p . This inflow to b is the outflow from some other machine, m' , and this means that the outflow of part p from m' is bounded below by a line of slope d_p as well. By showing this argument to hold for all the buffers in the system they demonstrate that the size of buffer b is bounded by the distance between two parallel lines of slope d_p , thus completing the proof.

It is perhaps worth noting that theorem 5 can be generalized to include any clearing policy with backoff, and that the machines do not have to all use the same clearing policy.

The "isolationist" nature of this class of policies enables it to deal with rework, assembly and disassembly, as explained in [VIII,1]. Every machine incorporates these operations by modifying its model of the imagined inflows. Rerouting a proportion α , of part type p at machine, m , is accomplished by considering two part types $p1$ and $p2$ such that $d_{p1} = \alpha d_p$, $d_{p2} = (1 - \alpha)d_p$. Disassembling a part type p into parts $p1$ and $p2$ is handled by replacing part type p with incoming part types $p1$ and $p2$, such that $d_{p1} = d_{p2} = d_p$.

Finally, assembly of parts $p1$ and $p2$ into a part type p at machine m is accomplished as follows: An coming part type p is defined at machine m such that the incoming rate rate of p is exactly the demand for the combined part, d_p and $x_{pm}(t) = \min\{x_{p1,m}, x_{p2,m}\}$.

A simple modification of the proof in Theorem 8 of [1] yields the result that the that the above modifications do not affect the stability of CAF (CGF) policies with backoff.

4.4 Non-Clearing Policies in Zero Set-Up Time Networks

In this section we will look at a number of policies on non-acyclic networks. Since we would like to learn more about the effects of starvation on stability, all the set-up times are assumed to be zero. Also, there will be only one part type, so that all the parts follow the same route through the network. This route is called the *flow line*.

As Lu and Kumar point out in [3], semiconductor manufacturing lines may be modeled by such systems.

Figure 1 of [3] is an example of the type of system we are considering. We use it to go over the notation to be used in the following sections. Note that this notation is in Section 2 of [3] and we are repeating it here for the reader's convenience.

There are 4 service centers, labeled 1-4, and 8 buffers, b_i $i = 1, 2, \dots, 8$. In general we will let there be S service centers and l buffers. The set of buffers at machine m is B_m . Service center $\sigma \in \{1, \dots, S\}$ contains a set, M_σ of m_σ machines that work on the buffers in parallel. Parts enter at σ_1 and the flow is defined by $\sigma_1, \sigma_2, \dots, \sigma_l$. Parts at b_i take τ_i units of time to serve.

In the previous sections we have treated the incoming parts as a continuous flow of material; here we break this flow up into parts. Let τ_i be the time taken to process a part in buffer i . The number of arrivals into the system, $u(t)$ is constrained by two parameters λ and γ so that for every interval $[s, t]$, $0 \leq s \leq t$:

$$u(t) - u(s) \leq \lambda(t - s) + \gamma, \quad \gamma, \lambda \geq 0. \quad (4.4.1)$$

This model of bursty input traffic has been recently studied by Cruz [6],[7] in the context of data communications networks, and it has been used by others such as Hakimi in the past [6]. Following Cruz, we say that the input flow, $u(t) \sim (\gamma, \lambda)$ if it obeys the arrival condition (4.4.1).

Every part in b_i , in service center σ requires $w_{\sigma_i} = \frac{\tau_i}{m_\sigma}$ units of work. Thus a necessary condition on stability is that

$$\rho = \max_{\sigma} \lambda \sum_{i:\sigma_i=\sigma} w_{\sigma_i} < 1. \quad (4.4.2)$$

In the remainder of Section 4, we assume that (4.4.2) always holds. Now the following "Fact", which we state without proof, will be useful to us in the following:

Fact 2. Consider a single service center, σ for which (4.4.2) holds. There are p arrival streams and the i^{th} arrival stream is $\sim (\Gamma_i, \lambda)$. At time 0 all the buffers at σ are empty. Then for every time $t \geq 0$, there exists a $t' \geq t$ such that all p buffers will be simultaneously empty at least once in the interval $[t, t + t']$.

Another simple, but useful result is due to Cruz:

Fact 3. Suppose an input stream $u(t) \sim (\gamma, \lambda)$ is delayed by a network N by a maximum of D time units. Then the output stream $v(t) \sim (\gamma + \lambda D, \lambda)$.

Proof: Note that any flow exiting the system in the interval $[t_1, t_2]$ had to have arrived in the interval $[t_1 - D, t_2]$. But only $\gamma + \lambda D + \lambda(t_2 - t_1)$ units could have arrived in that interval. So the result follows.

Following the notation of [3], we denote the arrival time of a part π to be $\alpha(\pi)$ and its time of exit to be $e(\pi)$. Thus the system is stable iff

$$e(\pi) - \alpha(\pi) \leq \Gamma, \text{ for all } \pi,$$

for some $\Gamma \geq 0$. Alternatively, if $x(t)$ is the total number of parts in the system at time t , the system is stable iff

$$x(t) \leq M, \text{ for all } t \geq 0, \text{ for some } M \geq 0.$$

While the policies considered in [3] are not clearing, they are not pre-emptive either. So once a machine accepts a part, it must complete production on that part. The non-preemptive aspect of these schedules introduces small delays in the processing of parts that the schedule gives high priority.

Lu and Kumar consider two kinds of schedules: Buffer Priority schedules and Due Date based schedules. In Buffer Priority schemes every buffer has a priority associated with it, and a machine always picks the next part to be processed from the head of the non-empty buffer of highest priority. In due date policies, every part, π , in buffer b_i has associated with it a number, $\mathcal{L}(\pi, i)$. A machine always picks the part with the smallest value of $\mathcal{L}(\pi, i)$, over all buffers at that machine, as the next part to be processed.

Let us define a *limited overtaking scheduling policy* to be one in which an incoming part may overtake (i.e. exit before) at most $\omega \geq 0$ parts already in the system.

All buffer priority schemes are *non-overtaking*, i.e. $\omega = 0$. To see this, observe that a part π may only overtake a part π' when they are both in the same buffer, but buffer priority schemes are first come first serve within any particular buffer.

In Section 4.4.4 we will show that under very reasonable assumptions, the Due Date based schemes considered in [3], are also limited overtaking schemes.

4.4.1 First Buffer First Served FBFS

In FBFS every machine in M_σ follows the following rule: serve the buffer $b_j \in B_\sigma$ iff every buffer $b_i \in B_\sigma$ such that $i < j$, is empty. Since the flow of parts from b_j can only be impeded by parts in buffers of smaller indices, we need only consider *these* buffers in determining whether the size of b_j is bounded over all time. Define $A^j = \{b_j : b_j \in B_{\sigma_j}, j \leq i\}$. Thus a part at the head of buffer b_j will be accepted by a machine in σ_j at time t only if all the buffers in $A^j - \{b_j\}$ are empty at t .

Notice that in FBFS the machines of the system “push” parts through the flow line. Incoming parts have the highest priority, and this priority decreases with each visit to a service center. In what follows we will provide a simple proof for the stability of FBFS when the system starts with all its buffers empty. We also assume that every service center has just one machine. Our proof captures the basic ideas of Theorem 1 in [3].

Define an i -busy period to be a period in which at least one of the buffers in A^i is not empty. We will show by induction on i that all i -busy periods are bounded.

$i = 1$: Suppose a 1-busy period begins at time T_1 . The machine in σ_1 must begin to serve b_1 by $T_1 + \bar{\tau}$, where $\bar{\tau} = \max_p \tau_p$. Now we show that the 1-busy period terminates at some finite time T_2 : We know that the arrival stream of parts is (γ, λ) and that the machine of σ_1 exclusively serves these arrivals at a rate $\frac{1}{\tau_1} > \lambda$. From fact 2, b_1 must be cleared in finite time, T_2 , and the 1-busy period is bounded.

Now assume that all i -busy periods, $i = 1, 2, \dots, i-1$ are bounded. By the induction hypothesis every j -busy period, $j < i$ is bounded in duration by some Z^j . Thus the time taken for a part to traverse b_1 through b_{j-1} , $j < i$ is bounded by some constant $\Gamma^j = \sum_{k=1}^j Z^k$. By fact 3 we see that inflow into buffer $b_i \sim (\lambda\Gamma^{i-1} + \gamma, \lambda)$. Similarly, the inflow of parts into buffer $b_j \in A^i - \{b_i\} \sim (\lambda\Gamma^{j-1} + \gamma, \lambda)$. Thus from fact 2 we are done.

4.4.2 Contractive Delay Estimates

The stability of FBFS was easy to determine since we could start our induction from the first buffer in the flow line, and this allowed us to exploit the well behavedness of the arrivals into the system. We could then argue that the priority system was set up to *maintain* this well behavedness at the other buffers as well. For policies other than FBFS this approach may not work, and consequently the issue of stability is much more difficult to resolve. In their proof for the stability of LBFS Lu and Kumar discovered a very useful property that may apply to a variety of different schedules.

Contractive Delay Estimate Property: *Let \bar{w} be the maximum amount of work brought per machine at a service center by an incoming part. The schedule S has the contractive delay estimate property if for every $\epsilon > 0$ there exists a constant $c(\epsilon)$ such that, if there are x parts in the system when a part π arrives then the delay it experiences satisfies:*

$$e(\pi) - \alpha(\pi) \leq c(\epsilon) + (\bar{w} + \epsilon)x, \quad \text{for every } \epsilon > 0.$$

Now we show the following:

Theorem 6. *Any limited overtaking scheduling policy that has the contractive delay estimate property is stable.*

Proof: This argument generalizes, but is essentially the same as, the one in the proof of Theorem 3 of [3]. That theorem deals with the stability of LBFS, but we want to generalize to any limited overtaking policy.

Let $t_0 = 0$ and recursively define:

$t_k :=$ the exit time of the part which is at the *beginning* of the system at time t_{k-1} .

The beginning of the system at time t is the first buffer in the flow line that is non-empty at time t .

Since there are $x(t_{k-1}) - 1$ parts in the system ahead of the part arriving at t_{k-1} :

$$t_k - t_{k-1} \leq c(\epsilon) + (\bar{w} + \epsilon)x(t_{k-1}).$$

By assumption a part may overtake at most ω parts before exiting the system. Thus

$$x(t_k) \leq \omega + \gamma + \lambda(t_k - t_{k-1}).$$

Substituting for $t_k - t_{k-1}$ and simplifying, we have

$$x(t_k) \leq (\lambda c(\epsilon) + \gamma + \omega) + \lambda(\bar{w} + \epsilon)x(t_{k-1}).$$

(This looks a lot like equation (A) of Section 3.2.) We see that

$$x(t_k) - x(t_{k-1}) \geq 0 \Rightarrow (1 - \lambda\bar{w} - \lambda\epsilon)x(t_{k-1}) \leq \lambda c(\epsilon) + \gamma + \omega.$$

Substituting $\rho = \lambda\bar{w}$ and applying arguments similar to those in Theorem 3 of [3], we have:

$$\limsup_k x(t_k) \leq \frac{\lambda c(\epsilon) + \gamma + \omega}{1 - \rho - \lambda\epsilon}.$$

and

$$x(t_k) \leq \max\left\{x(0), \frac{\lambda c(\epsilon) + \gamma + \omega}{1 - \rho - \lambda \epsilon}\right\}$$

for all $k \geq 0$. Now all we need do is to note that for any $t \in [t_{k-1}, t_k]$:

$$x(t) \leq x(t_{k-1}) + \gamma + \lambda(t_k - t_{k-1}).$$

Substituting for $x(t_{k-1})$ and $t_k - t_{k-1}$ we see that

$$x(t) \leq \max\left\{(1 + \rho + \lambda \epsilon)x(0) + \lambda c(\epsilon) + \gamma + \omega, \frac{2(\lambda c(\epsilon) + \gamma + \omega)}{1 - \rho - \lambda \epsilon}\right\},$$

for all $t \geq 0$. This shows the theorem.

In what follows we will consider three policies that are limited overtaking: Last Buffer First Served (LBFS), Earliest Due Date First Served (EDD) and Least Slack First Served (LS), and prove them to be stable by showing that they each have the contractive delay estimate property.

4.4.3 Last Buffer First Served

LBFS is the diametrically opposite strategy of FBFS: Every machine in M_σ follows the following rule: serve the buffer $b_j \in B_\sigma$ iff every buffer $b_i \in B_\sigma$ such that $i > j$, is empty. First notice that since LBFS is a buffer priority scheme, it is non-overtaking i.e. it is a limited overtaking policy with $\omega = 0$.

In LBFS the service centers attempt to “pull” out from the system as many parts as they can. Thus every time a part revisits a service center, it waits in a higher priority buffer.

In theorem 2 of [3], Lu and Kumar provide a proof showing that LBFS has the contractive delay estimate property. Their argument is powerful since it can be generalized to show the contractive delay estimate property of EDD and LS as well.

As explained on page 14 of [3], “the proof hinges on the fact that parts entering the system after π cannot interfere with part π except for causing a *minor* delay because of the non-preemptive discipline.”

By the word “minor” Kumar and Lu mean that no part, π , is *delayed at any service center for more than a constant \bar{d} units of time by parts arriving after π* . For LBFS we see that $\bar{d} = \bar{\tau} = \max_j \tau_j$.

Let us call all policies for which such a \bar{d} exists, *Quasi Least Buffer First Serve (QLBFS) Policies*.

Then we can restate theorem 2 of [3] as follows:

Theorem 7. *Any QLBFs policy has the contractive delay estimate property.*

From theorems 7 and 6 we obtain:

Theorem 8. *All limited overtaking QLBFs policies are stable.*

and

Corollary 2. *LBFS is stable.*

4.4.4 Due-Date Based Policies

Two Due Date policies are considered by Lu and Kumar. In Earliest Due Date First (EDD) the due dates $\mathcal{L}(\pi, i)$ are independent of the buffer number, i.e. they are of the form $\delta(i)$. In Least Slack Served First (LS) the due dates are of the form:

$$\mathcal{L}(\pi, i) = \delta(\pi) - \eta_i, \quad i = 1, 2, \dots, l, \quad \forall \pi,$$

for $\eta_i \geq 0$.

We may consider η_i to be an estimate of how long the part will take to exit the system, given that it is at buffer b_i , and $\delta(\pi)$ to be a due date on when the part desires to exit the system.

While in theory, one should be able to assign any due date to a part π it is very likely that in practice these due dates will bear some relationship to $\alpha(\pi)$, the arrival time of π . Kumar and Lu make the assumption that for any manufacturing system, there exist γ_1 and γ_2 , both ≥ 0 such that:

$$-\gamma_1 \leq \delta(\pi) - \alpha(\pi) \leq \gamma_2, \quad \forall \pi.$$

Now notice that EDD is a special case of LS when all the η_i 's are zero. Thus we need only the stability of LS to ensure the stability of both policies.

Theorem 9. *LS is a QLBFs policy with $\bar{d} = d_1 + d_2$ where*

$$d_1 = (\lambda(\gamma_2 + \gamma_1) + \gamma)\bar{\tau}$$

$$d_2 = \bar{\tau}\lambda \max_{i,j} \{\gamma_1 + \gamma_2 \max(\gamma_1 + \gamma_2 - \eta_i + \eta_j)\} + \bar{\tau}\gamma.$$

Proof: These bounds are given on page 22 of [3]. d_1 bounds the amount of time a part π would have to wait due to overtaking parts, and d_2 bounds the amount of time π would have to wait while parts of higher priority, in lower indexed buffers are cleared.

Now by arguments similar to those used in obtaining d_1 , we can show that LS is a limited overtaking policy with $\omega = \frac{\bar{d}_1}{\bar{\tau}}$. From this and from Theorem 8 have the result:

Corollary 3. *LS and EDD are stable.*

5 Conclusions

5.1 Summary of Results

The following is a summary of the major conclusions reached in this report:

- (1) **Single Machine Systems:** All CGF policies are stable under a variety of arrival assumptions. Idling occurs when the system is lightly loaded and when a particular part type, j has large values of α_j , and $\gamma_j \rho_j$. Idling reduces average buffer levels by an amount proportional to the set up times, δ , although the decision to idle is made independently of δ . Good policies may be devised by exploiting the tight lower bound derived in Section 3.3.
- (2) **Networks:** Starvation is an important cause of instability. Clearing Policies are especially susceptible to starvation since they entail long production runs. Buffer priority schemes can reduce the effects of starvation since they may not be clearing. Also, since the buffers are ordered relative to the flow line they can take advantage of this information to “push” or “pull” flow through the network. Any limited overtaking policy that has the contractive delay estimate property is stable. All limited overtaking QLBFS policies are stable.

5.2 Limitations and Suggestions for Further Work

While the work of the papers reviewed in this paper provides considerable insight into the dynamics of manufacturing networks, there remain a number of unanswered questions:

- (1) Tighter upper bounds for CGF policies are probably not very difficult to obtain, since the current method ignores the fact that two consecutive production runs must be on different part types. It would be interesting to see how close they relate to the lower bound established in Section 3.3.
- (2) The performance of clearing policies with backoff was not analyzed or commented on. This is especially important since no other class of policies has been found that can handle re-work, assembly and disassembly in non-acyclic networks.
- (3) The effect of set-up times in networks was only briefly considered (in Case 2 of Section 4.2.) The interaction between starvation and set-up induced instability would be interesting to study.

Stability in Manufacturing Systems

- (4) Regions of stability for policies on specific networks, relative to initial buffer levels, might provide insight into the nature of instability.
- (5) Lu and Kumar mention that the stability of FCFS is an important open question. A totally different approach than the contractive delay estimate one in [3], will be necessary to resolve the question.
- (6) Stochastic models of re-work, (which appears to be an important requirement in semiconductor assembly lines) would be interesting to look at. As Lu and Kumar suggest, contractive delay estimate properties might be useful in such analyses.
- (7) Simple models of failures could be introduced with fixed alternate routes. For example, we could bound the maximum number of failures present at any time, assume fixed time repair, and also require that the total number of possible failures in any interval of time be bounded. The point of looking at such failures would be to test the robustness of stable policies.

References

- [1] J. R. Perkins and P. R. Kumar, "Stable Distributed real-time scheduling of flexible manufacturing systems," *IEEE Transactions on Automatic Control*, AC-34(2):139-148, February 1989.
- [2] C. J. Chase and P. J. Ramadge, "On the real time control of flexible manufacturing systems," *Proceedings of the 28th IEEE Conference on Decision and Control*, Tampa, Florida, 1989.
- [3] C. Lu and P. R. Kumar, "Distributed scheduling based on due dates and buffer prioritization," *University of Illinois Technical Report*, 1990.
- [4] S. B. Gershwin, "Hierarchical Flow Control: A framework for scheduling and planning discrete events in manufacturing systems," *Proceedings of the IEEE*, vol 77, no. 1, January, 1989.
- [5] F. P. Kelly, *Reversibility and Stochastic Networks* John Wiley and Sons, New York, 1987.
- [6] R. L. Cruz, "A calculus for network delay, Part I: Network elements in isolation," *University of California at San Diego Technical Report*, October 1989.
- [7] R. L. Cruz, "A calculus for network delay, Part II: Network analysis," *University of California at San Diego Technical Report*, October 1989.
- [8] P. R. Kumar and T. I. Seidman, "Dynamic instabilities and stabilization methods in distributed real time scheduling of manufacturing systems," *IEEE Transactions on Automatic Control*, AC-35(3):289-298, March 1990.