[13] B. D. O. Anderson and J. B. Moore, "Algebraic structure of generalized positive real matrices," *SIAM J. Contr. Optim.*, vol. 6, no. 4, pp. 615–624, 1968.
[14] P. A. Ioannou and G. Tao, "Frequency domain conditions for strictly positive real functions," *IEEE Trans. Automat. Contr.*, vol. 32, pp. 53–54, Jan. 1987.
[15] T. Iwasaki and R. E. Skelton, "A complete solution to the general H^∞ problem: LMI existence conditions and state-space formulas," in *Proc. American Control Conf.*, San Francisco, CA, 1993. New York: IEEE, pp. 605–609.
[16] P. Gahinet and P. Apkarian, "A linear matrix inequality approach to H^∞ control," *Int. J. Robust Nonlinear Contr.*, vol. 4, no. 4, pp. 421–448, 1994.
[17] J. H. Ly, M. G. Safonov, and F. Ahmad, "Positive real Parrott theorem with applications to LMI controller synthesis," in *Proc. American Control Conf.*, Baltimore, MD, 1994. New York: IEEE, pp. 50–52.
[18] F. J. H. Don, "On the symmetric solutions of a linear matrix equation," *Linear Algebra and its Appl.*, vol. 93, pp. 1–7, 1987.
[19] P. Gahinet, A. Nemirovskii, and A. Laub, *LMIlab: A Package for Manipulating and Solving LMI's.* South Natick, MA: The Mathworks, 1994.
[20] I. Barkana, "Adaptive control: A simplified approach," in *Advances in Control and Dynamics*, vol. 25, C. T. Leondes, Ed. New York: Academic, 1987.
[21] K. Sobel, "Model reference adaptive control for multi-input, multi-output systems," Ph.D. dissertation, Rensselaer Polytechnic Inst., Troy, NY, June 1980.
[22] K. Sobel, H. Kaufman, and L. Mabius, "Implicit adaptive control systems for a class of multi-input, multi-output systems," *IEEE Trans. Aerosp. Electronic Syst.*, vol. 18, no. 5, pp. 576–590, 1982.
[23] P. A. Ioannou and J. Sun, *Robust Adaptive Control.* Englewood Cliffs, NJ: Prentice-Hall, 1996.

Path-Clearing Policies for Flexible Manufacturing Systems

Kevin Burgess and Kevin M. Passino

***Abstract*— In practical manufacturing settings it is often possible to obtain, in real-time, information about the operation of several machines in a flexible manufacturing system (FMS) that can be quite useful in scheduling part flows. In this brief paper the authors introduce some scheduling policies that can effectively utilize such information (something the policies in [1] do not do) and they provide sufficient conditions for the stability of two such policies.**

***Index Terms*— Boundedness, manufacturing systems, scheduling, stability, traffic control.**

I. Introduction

In this paper, we consider the use of global information for scheduling flexible manufacturing systems (FMS) of the type considered in [1] and later in [2]–[4]. This paper is unique with respect to this body of work in that it provides an analysis of a particular class of "global," rather than "local," scheduling policies. The primary advantage of an FMS composed of individual machines, each with its own scheduling policy that utilizes only local information, is that the individual machines need not communicate with one another so that real-time implementation is simplified. However, for many modern FMS's, it is quite realistic to allow intermachine communications. Here, we seek to exploit this fact by developing scheduling policies that incorporate information from other parts of the network that can be useful in making efficient scheduling decisions. In using more "global" information, we are careful to minimize the level of necessary communications so that our global policies are implementable in real-time, just as the local policies mentioned above.

In this work, which we view as only a first step toward solving the problem of how to use global FMS information to achieve high-performance scheduling, we define and analyze a class of global scheduling policies that we call path-clearing (PC) policies. Similar to the way in which local policies select a buffer to service from among the buffers of a single machine, PC policies select from among a set of *paths* to service. A path is a set of topologically consecutive buffers which can be serviced simultaneously. In general, a PC policy will choose from among *sets* of paths to process. When a PC policy chooses a set of paths to process, all buffers in each path in the set are processed simultaneously (hence, all paths in a set must be able to be processed at the same time). Once a PC policy begins servicing a set of paths, servicing continues until all paths in the set are clear of parts.

Manuscript received June 17, 1996. Recommended by Associate Editor, E. K. P. Chong.

The authors are with the Department of Electrical Engineering, The Ohio State University, Columbus, OH 43210 USA.

Publisher Item Identifier S 0018-9286(99)00562-0.

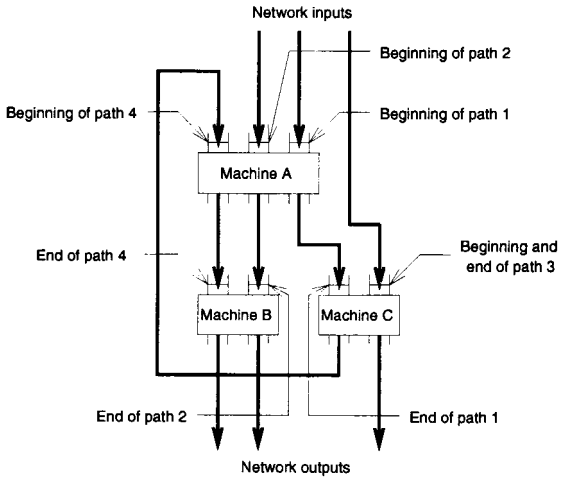

Fig. 1. Example FMS with paths labeled.

II. System Description and Notation

Let there be N "paths" within the FMS. The three important attributes of any path are: 1) all of its buffers may be serviced at the same time; 2) its buffers are directly connected in the network; and 3) if a buffer is on one path it cannot be on another path, and all buffers must be on a path. For example, in the network of Fig. 1 we can define four paths which begin and end as indicated. Notice that the beginning and end of a path are defined as the first and last buffer in the path, respectively. Let each buffer be referred to by a coordinate (i, j), where i is the path number and j is the buffer

0018–9286/99$10.00 © 1999 IEEE

number along the path. For example, in the FMS of Fig. 1 the buffer that ends path 4 is referred to by coordinate (4,2). Let Q be the set of paths which originates at network inputs and let R be the set of paths which originates at network re-entry points. For the FMS of Fig. 1, $Q = \{1, 2, 3\}$ and $R = \{4\}$. Note that for a fixed FMS topology there can be many different choices for the set of N paths (indeed it is an important research direction to determine the best set of paths).

Let the level of buffer (i, j) at time t be $x_{i,j}(t)$, and let $X(t) = \{x_{i,j}(t)\}$ be the set of all of the buffer levels at time t. Let a_i be the part arrival rate (in parts per unit time) at network input buffer $(i, 1)$, $i \in Q$. For convenience, if $i \in R$, we let $a_i = 0$. Let s_i be the maximum setup time of all the buffers on path i. For example, if for path 1 it takes two time units for machine A to configure itself to process parts from buffer (1,1) and it takes one time unit for machine C to configure itself to process parts from buffer (1,2), then $s_1 = 2$. Let τ_i be the minimum processing rate for buffers on path i. All buffers in path i are serviced at rate τ_i and the sum of the transport delays between buffers on path i (including the transport delay between the last buffer in path i and the head buffer of a downstream path, if applicable) is D_i. Let P_i be the number of paths, including i, that a part in buffer $(i, 1)$ will traverse before exiting the FMS. For example, in the FMS of Fig. 1, $P_1 = 2$ and $P_i = 1$ for $i = 2, 3, 4$. If $P_i > 1$, define H_i to be the path that is fed by path i. For example, in the FMS of Fig. 1, $H_1 = 4$.

Suppose that we let S_j denote a set of paths such that all paths in the set S_j can be serviced simultaneously. Let $S = \{S_j : 1 \leq j \leq M\}$ be the set of such sets of paths for an FMS. We require that the set S satisfies the following condition: for all paths $i \in \{1, 2, \cdots, N\}$, there is some $j \in \{1, 2, \cdots, M\}$ such that $i \in S_j$ (each path is in at least one S_j). For example, in the FMS of Fig. 1, we can define S as $S = \{\{1\}, \{2, 3\}, \{3, 4\}\}$, where $S_1 = \{1\}$, $S_2 = \{2, 3, \}$, and $S_3 = \{3, 4\}$. We must emphasize that for a given FMS there may be many ways to define the S_j and S. The results of this paper are for any one that satisfies the above constraints (it is an important direction to determine the best choice of the S_j that make up S).

The production scheduling policies for the FMS that we now consider are what we will call PC policies. A PC policy is one in which a production run (i.e., a time period during which parts from a particular set of buffers are processed) is begun by choosing a set of paths $S_i \in S$. All buffers on paths in S_i are serviced until the head buffer, $(j, 1)$, $j \in S_i$, of every path in S_i is clear. At that point, the head buffers are no longer serviced and the production run continues until all buffers, except possibly the head buffers, in the paths of S_i are empty. In the following analysis, we will assume that all buffers except the head buffers are initially empty (if this is not true initially, it will be after each path has been serviced once). For convenience, we will refer to the level of the head buffer of path i, $(i, 1)$ at time t as $x_i(t)$. The goal of the following analysis is to identify PC policies which are stable (i.e., buffer levels remain bounded for all time) and to obtain bounds on the buffers of FMS controlled by such policies.

III. Path Clear-A-Fraction Policy

The first PC policy that we consider is the PCAF (path clear-a-fraction) policy. This policy is a generalization of the local CAF policy for single machines, introduced in [1]. We define the PCAF policy by specifying how it chooses the next set of paths to service, upon the ending of a production run. Define a function $V(X(t))$ as $V(X(t)) \triangleq \sum_{i=1}^{N} P_i x_i(t)$. Notice that $V(X(t))$ is the sum of all parts in the head buffers of all paths in the network (at the end of a production run, there can be no parts in buffers that are not head buffers) weighted by the number of paths that the parts must traverse, after and including the path that they are currently in, before exiting the FMS. If a scheduling decision is to be made at time t, the PCAF policy chooses to service a set of paths S_i such that $\sum_{j \in S_i} x_j(t) \geq \epsilon V(X(t))$ for some sufficiently small, fixed $\epsilon > 0$. We must choose ϵ small enough so that regardless of the distribution of parts at the end of any production run, there is some $i \in \{1, 2, \cdots, M\}$ that satisfies the above selection criterion. It is easy to identify $\epsilon^* > 0$ such that for any $\epsilon \leq \epsilon^*$ there is some $i \in \{1, 2, \cdots, M\}$ such that the selection criterion is satisfied. For instance, $\epsilon^* = \frac{1}{NP}$, where $\bar{P} = \max_i P_i$ will work.

In summary, during a production run, the PCAF policy completely clears all parts from each path processed (except for parts which may accumulate at the head buffers after processing has stopped at the head buffers). Notice that because all buffers in a path are serviced at the minimum processing rate for the path, parts never accumulate at any buffers in the path other than the head buffer. In other words, along each individual path, parts arrive at downstream buffers at a rate that is no greater than the rate at which they can be processed. This is sometimes referred to as "pipelining" parts. Next, we shall develop conditions under which the PCAF policy is stable.

To begin with, we define several parameters that will be used in the stability analysis. First, define a set of times $T = \{t_0, t_1, t_2, \cdots\}$ where t_0 is the FMS start-up time and beginning of the first production run, t_1 is the end of the first production run and beginning of the second production run, and, in general, t_i is the end of production run i and the beginning of production run $i + 1$. Let $\max\{s_j : j \in S_i\}$ be denoted by $\bar{s}^i$, $\max\{D_j : j \in S_i\}$ by $\bar{D}^i$, $\max\{\tau_j : j \in S_i\}$ by $\bar{\tau}^i$, $\min\{\tau_j : j \in S_i\}$ by $\underline{\tau}^i$, $\max\{a_j : j \in S_i\}$ by $\bar{a}^i$, $\max\{\bar{s}^i : 1 \leq i \leq M\}$ by $\bar{s}$, and $\max\{\bar{a}^i : 1 \leq i \leq M\}$ by $\bar{a}$. Let us choose an arbitrary time $t_p \in T$. Let us assume that the PCAF policy has chosen to service buffers in the paths of S_i during the production run beginning at time t_p. For every $i \in \{1, 2, \cdots, M\}$, define a constant η_i, $0 < \eta_i \leq 1$, such that at any time t_p, $\max\{x_j(t_p) : j \in S_i\} \leq \eta_i \sum_{j \in S_i} x_j(t_p)$. Notice that η_i follows from the topology of the FMS. In many cases, we may have to take $\eta_i = 1$ because $|S_i| = 1$ or because the topology of the FMS does not allow for $\eta_i < 1$ in general (note that if an $\eta_i < 1$ cannot be found then $\eta_i = 1$ will suffice for the stability analysis below, no matter what FMS topology is being considered). As an example of a situation in which we can take $\eta_i < 1$, consider paths $S_i = \{m, n\} \subset Q$, where $m \notin S_j$ and $n \notin S_j$ for all $j \in \{1, 2, \cdots, M\}$, $j \neq i$. Because paths m and n were both last cleared at the same time, $x_m(t_p) = a_m t'$ and $x_n(t_p) = a_n t'$, for some $t' > 0$. From the above definition of η_i, we see that $\eta_i = \max\{\frac{x_m(t_p)}{x_m(t_p)+x_n(t_p)}, \frac{x_n(t_p)}{x_m(t_p)+x_n(t_p)}\}$. Hence, $\eta_i = \max\{\frac{a_m}{a_m+a_n}, \frac{a_n}{a_m+a_n}\}$. Generally, it is desirable to find an η_i that is as small as possible as the conditions for stability then become less restrictive and the bounds obtained on buffer levels become tighter, as can be seen by the following stability result.

Theorem 1: Let $F(t)$ be a function that returns the integer number of production runs that have been completed during the time interval $[t_0, t]$. If, for all $i \in \{1, 2, \cdots, M\}$, $\bar{a}^i < \underline{\tau}^i$, and

$$\eta_i \frac{\sum_{j \notin S_i} P_j a_j + \sum_{j \in S_i} (P_j - 1) a_j}{\underline{\tau}^i - \bar{a}^i} < 1$$

then the PCAF policy is stable. Furthermore, if $\gamma = \max\{\gamma_i\}$ and $\zeta = \max\{\zeta_i\}$

$$\sum_{i=1}^{N} P_i x_i(t) \leq \gamma^{F(t)} \left(V(X(t_0)) - \frac{\zeta}{1 - \gamma} \right) + \frac{\zeta}{1 - \gamma} + \sum_{i=1}^{N} P_i \bar{s} \bar{a}$$

where

$$\zeta_i = \frac{\bar{s}^i + \bar{D}^i}{1 - \frac{\bar{a}^i}{\underline{\tau}^i}} \left[\sum_{j \notin S_i, j \in Q} P_j a_j + \sum_{j \in S_i} (P_j - 1) a_j \right] + \sum_{j \in S_i} P_j \bar{D}^i a_j$$

$$\gamma_i = 1 - \epsilon \left[1 - \eta_i \frac{\sum_{j \notin S_i} P_j a_j + \sum_{j \in S_i} (P_j - 1) a_j}{\underline{\tau}^i - \bar{a}^i} \right].$$

Proof: We first bound the length of the production run beginning at t_p in terms of $\sum_{j \in S_i} x_j(t_p)$ by a value which we shall call Δ_p. There are four factors which are summed to form Δ_p: 1) a bound on the maximum amount of time to clear parts in any head buffer of a path in S_i, $\frac{\eta_i \sum_{j \in S_i} x_j(t_p)}{\underline{\tau}^i}$; 2) a bound on the maximum amount of time to clear parts which arrive at any head buffer of a path in S_i during the production run, $\frac{\bar{a}^i \Delta_p}{\underline{\tau}^i}$; 3) the maximum setup time for any buffer in the paths in S_i, $\bar{s}^i$; and 4) the maximum total transport time of any of the paths in S_i, $\bar{D}^i$. Hence

$$\begin{aligned} \Delta_p &\le \frac{\bar{a}^i \Delta_p + \eta_i \sum_{j \in S_i} x_j(t_p)}{\underline{\tau}^i} + \bar{s}^i + \bar{D}^i \\ &= \frac{\eta_i \frac{\sum_{j \in S_i} x_j(t_p)}{\underline{\tau}^i} + \bar{s}^i + \bar{D}^i}{1 - \frac{\bar{a}^i}{\underline{\tau}^i}}. \end{aligned}$$

Next we bound $V(X(t_{p+1}))$ in terms of $V(X(t_p))$. In order to accomplish this, we consider the contribution to V by several subsets of S. First, notice that

$$\sum_{j \in S_i} P_j x_j(t_{p+1}) \le \sum_{j \in S_i} P_j x_j(t_p) - \sum_{j \in S_i} P_j x_j(t_p) + \sum_{j \in S_i} P_j \bar{D}^i a_j.$$

This is because all paths in S_i are cleared during the production run and because some parts may accumulate at the head buffers during any part transport time before the end of the production run. Next, notice that

$$\sum_{j \notin S_i, j \in Q} P_j x_j(t_{p+1}) \le \sum_{j \notin S_i, j \in Q} P_j x_j(t_p) + \sum_{j \notin S_i, j \in Q} P_j \Delta_p a_j.$$

This is because all paths not in S_i that are fed by network inputs have their buffer levels grow at the rate of part arrival during the production run. Let $H^*_{S_i}$ be the set of all paths fed by paths in S_i so that $H^*_{S_i} = \{j = H_k : k \in S_i, P_k > 1\}$. Then we see that

$$\sum_{j \in H^*_{S_i}} P_j x_j(t_{p+1}) \le \sum_{j \in H^*_{S_i}} P_j x_j(t_p) + \sum_{j \in S_i} (P_j - 1)[x_j(t_p) + \Delta_p a_j].$$

This is because a path which is fed by some path $j \in S_i$ has its head buffer increase during the production run by no more than $x_j(t_p) + \Delta_p a_j$, a bound on the total number of parts processed through path j during the production run. Notice that the P_{j-1} term in the final sum is zero if path j feeds a network output. Finally, notice that

$$\sum_{\substack{j \notin S_i, j \in R \\ j \notin H^*_{S_i}}} P_j x_j(t_{p+1}) = \sum_{\substack{j \notin S_i, j \in R \\ j \notin H^*_{S_i}}} P_j x_j(t_p).$$

This is because paths not in S_i that are fed neither by network inputs nor paths in S_i have their head buffers remain unchanged during the production run. If we sum the four preceding (in) equalities, we see that

$$\begin{aligned} V(X(t_{p+1})) \le V(X(t_p)) &- \sum_{j \in S_i} P_j x_j(t_p) \\ &+ \sum_{j \in S_i} P_j \bar{D}^i a_j + \sum_{j \notin S_i} P_j \Delta_p a_j \\ &+ \sum_{j \in S_i} (P_j - 1)[x_j(t_p) + \Delta_p a_j]. \end{aligned}$$

Manipulating the above expression and using the bound on Δ_p, we see that an upper bound on $V(X(t_{p+1}))$ is

$$\begin{aligned} &V(X(t_p)) - \sum_{j \in S_i} x_j(t_p) + \left(\frac{\eta_i \sum_{j \in S_i} x_j(t_p)}{\underline{\tau}^i - \bar{a}^i} + \frac{\bar{s}^i + \bar{D}^i}{1 - \frac{\bar{a}^i}{\underline{\tau}^i}} \right) \\ &\quad \times \left[\sum_{j \notin S_i} P_j a_j + \sum_{j \in S_i} (P_j - 1) a_j \right] + \sum_{j \in S_i} P_j \bar{D}^i a_j \\ &\quad = V(X(t_p)) + \zeta_i - \left(\sum_{j \in S_i} x_j(t_p) \right) \\ &\qquad \times \left[1 - \eta_i \frac{\sum_{j \notin S_i} P_j a_j + \sum_{j \in S_i} (P_j - 1) a_j}{\underline{\tau}^i - \bar{a}^i} \right] \\ &\quad \le V(X(t_p)) + \zeta_i - \epsilon V(X(t_p)) \\ &\qquad \times \left[1 - \eta_i \frac{\sum_{j \notin S_i} P_j a_j + \sum_{j \in S_i} (P_j - 1) a_j}{\underline{\tau}^i - \bar{a}^i} \right] \\ &\quad = \gamma_i V(X(t_p)) + \zeta_i \end{aligned}$$

where

$$\zeta_i = \frac{\bar{s}^i + \bar{D}^i}{1 - \frac{\bar{a}^i}{\underline{\tau}^i}} \left[\sum_{j \notin S_i} P_j a_j + \sum_{j \in S_i} (P_j - 1) a_j \right] + \sum_{j \in S_i} P_j \bar{D}^i a_j$$

and

$$\gamma_i = 1 - \epsilon \left[1 - \eta_i \frac{\sum_{j \notin S_i} P_j a_j + \sum_{j \in S_i} (P_j - 1) a_j}{\underline{\tau}^i - \bar{a}^i} \right].$$

Due to the sufficient condition of Theorem 1, $0 < \gamma_i < 1$. Notice that the sufficient condition of Theorem 1 is satisfied if $\underline{\tau}^i$ is sufficiently large with respect to $\bar{a}^i$ for all $i \in \{1, 2, \cdots, M\}$. If we let $\gamma = \max\{\gamma_i : i = 1, 2, \cdots, M\}$ and $\zeta = \max\{\zeta_i : i = 1, 2, \cdots, M\}$, we see that for any time $t_p \in T$, $V(X(t_{p+1})) \le \gamma V(X(t_p)) + \zeta$. We can solve this first-order difference inequality to yield $V(X(t_k)) \le \gamma^k (V(X(t_0)) - \frac{\zeta}{1-\gamma}) + \frac{\zeta}{1-\gamma}$. Choose any $t_p \in T$ such that the buffers of path S_i are serviced during the production run beginning at time t_p. It is clear that in the closed time interval $[t_p, t_{p+1}]$, the maximum of $V(X(t))$ occurs either at t_p, t_{p+1}, or at a point t' which occurs sometime during the span of time during which servicing begins on the head buffers of paths in S_i. Specifically, $t_p + \underline{s}^i \le t' \le t_p + \bar{s}^i$. Consequently, if we let $\bar{s} = \max\{\bar{s}^i : i = 1, 2, \cdots M\}$, we see that for all $t \ge t_0$

$$V(X(t)) \le \gamma^{F(t)} \left(V(X(t_0)) - \frac{\zeta}{1 - \gamma} \right) + \frac{\zeta}{1 - \gamma} + \sum_{i \in Q} P_i \bar{s} a_i.$$

Further, we see from the definition of $V(X(t))$ that

$$\sum_{i=1}^{N} P_i x_i(t) \le \gamma^{F(t)} \left(V(X(t_0)) - \frac{\zeta}{1 - \gamma} \right) + \frac{\zeta}{1 - \gamma} + \sum_{i=1}^{N} P_i \bar{s} \bar{a}$$

for all $t \ge t_0$ so that clearly we have boundedness. □

Notice that while the condition for the stability of the PCAF policy is not as intuitive as the capacity condition of the CAF policy, it is,

in some sense, a generalization of the capacity condition because, for each $i \in \{1, 2, \cdots, M\}$, it requires that the rate of growth of some fraction (η_i) of the growth of V during a production run be less than the rate at which V is decreased during the production run. Notice that the PC policies do not truly require global information (i.e., all buffer levels and the status of processing at each machine). Because of the manner in which a PC policy mandates part pipelining by synchronizing production in consecutive machines, the policy only needs to know: 1) when each head buffer has been cleared during a production run (so that the policy can stop production at the head buffers when they all have been cleared); 2) when all paths currently being serviced are empty downstream from the head buffers (so that the policy can begin a new production run); and 3) the buffer levels of all head buffers at the end of each production run (so that the policy can select a new set of paths to service). Similar to the way in which the clear-largest-buffer (CLB) policy is a special case of policy CAF [1], we can define the path clear-largest-buffer (PCLB) policy as a special case of PCAF. The PCLB policy chooses to process the paths in the set S_i, $i \in \{1, 2, \cdots, M\}$, with the largest sum of parts. More precisely, PCLB chooses at time t_p to process paths of some set S_i, $i \in \{1, 2, \cdots, M\}$, such that $\sum_{j \in S_i} x_j(t_p) \geq \sum_{j \in S_q} x_j(t_p)$, for all $q \in \{1, 2, \cdots, M\}$, $q \neq i$. PCLB is considered to be a special case of PCAF because at each time PCLB picks an S_i to process, this S_i could have also been chosen by PCAF. Due to this fact, Theorem 1 also applies to the PCLB policy. The primary drawback of the above analysis is the conservative nature of the sufficient condition for stability. The fact is that many FMS which do not satisfy the condition are indeed stable. In the next section, we consider a priority PC policy that for a large number of FMS topologies will alleviate the conservative nature of the stability condition and produce much sharper buffer bounds than the PCAF analysis.

IV. Path Periodic Clearing Policy

The scheduler that we now describe is inspired by the observation that when a PC policy is allowed to control an FMS, it often falls into a periodic pattern of servicing choices. This periodicity often occurs whether or not the resulting FMS is stable. What we call the Path Periodic Clearing (PPC) Policy will enforce a given periodic service schedule. There are several ways to view such a policy. In some cases, intuition may be used to specify a periodic policy which will perform well. In other cases, service periods which perform well may be identified by simulation or actual implementation. Finally, we may identify a stable service period and use it as a "stabilizing safety net" for some other PC policy that we cannot rigorously guarantee is stable.

For our stability analysis, we begin by specifying a special model of the FMS being considered. Let $y(p)$ be the vector of the head buffer levels at time t_p so that

$$\begin{aligned} y(p) &= [y_1(p), y_2(p), \cdots, y_N(p)]^T \\ &= [x_1(t_p), x_2(t_p), \cdots, x_N(t_p)]^T. \end{aligned}$$

The key to our analysis in this section is the fact that we can write $y(p+1) = A_{i,j}y(p) + b_{i,j}$ for some $A_{i,j} \in \mathbb{R}^{N \times N}_{[0,\infty)}$ and $b_{i,j} \in \mathbb{R}^N$ ($\mathbb{R}^{N \times N}_{[0,\infty)}$ is the set of all real $N \times N$ matrices with nonnegative entries). The subscript "i,j" in the above expression denotes that paths in the set S_i are serviced during the production run beginning at time t_p and that path $j \in S_i$ is a *critical path*. In calling path j a "critical path," we mean that it is the last of the paths in S_i to have its head buffer cleared during the production run (i.e., when the head buffer of path j is cleared, processing ceases at all of the head buffers of paths in S_i). Let n_i be the number of possible critical paths in S_i. If j is the critical path (ties are resolved arbitrarily) and paths in S_i are serviced during the production run beginning at time t_p, we can write Δ_p, the time it takes to complete the production run begun at time t_p, as

$$\begin{aligned} \Delta_p &= \frac{y_j(p)}{\tau_j} + \frac{a_j(\Delta_p - D_j)}{\tau_j} + s_j + D_j \\ &= \frac{\frac{y_j(p) - a_j D_j}{\tau_j} + s_j + D_j}{1 - \frac{a_j}{\tau_j}} \\ &= \frac{y_j(p)}{\tau_j - a_j} + \frac{s_j + D_j - \frac{a_j D_j}{\tau_j}}{1 - \frac{a_j}{\tau_j}}. \end{aligned} \tag{1}$$

Notice that this expression for Δ_p is similar, but not exactly the same as, the expression for Δ_p in the PCAF analysis. The difference arises because this is an exact expression for Δ_p, where the expression in the PCAF analysis is an upper bound.

Let $[A_{i,j}]_{m,n}$ denote the element in row m and column n of $A_{i,j}$, and let $[b_{i,j}]_m$ denote element m of $b_{i,j}$. We now describe how to build $A_{i,j}$ and $b_{i,j}$ for an FMS of arbitrary topology. First, consider $m \in S_i$. Because all paths in S_i are cleared, except for parts which arrive in the time segment of length $\bar{D}^i$ between the end of head buffer servicing and the end of the production run, we see that $[A_{i,j}]_{m,n} = 0, n \in \{1, 2, \cdots, N\}$ and $[b_{i,j}]_m = \bar{D}^i a_m$. Secondly, consider $m \notin S_i$, $m \in Q$ (i.e., path m is not being serviced and is fed by a network input). Clearly, $y_m(p+1) = y_m(p) + a_m \Delta_p$ so that if j is the critical path and we use (1)

$$[A_{i,j}]_{m,n} = \begin{cases} 1, & n = m \\ \frac{a_m}{t_j - a_j}, & n = j \\ 0, & \text{otherwise} \end{cases}$$

and

$$[b_{i,j}]_m = a_m \frac{s_j + \bar{D}^i - \frac{a_j \bar{D}^i}{\tau_j}}{1 - \frac{a_j}{\tau_j}}.$$

Thirdly, consider $m \in H^*_{S_i}$ with $H_q = m$, $q \in S_i$ (i.e., path m is not being serviced, but is fed by path q that is being serviced). Because m is fed by $q \in S_i$, we see that $y_m(p+1) = y_m(p) + y_q(p) + a_q(\Delta_p - \bar{D}^i)$. Hence, if j is the critical path and we use one,

$$[A_{i,j}]_{m,n} = \begin{cases} 1, & n = m \\ 1, & n = q \quad \text{and} \quad q \neq j \\ \frac{a_q}{t_j - a_j}, & n = j \quad \text{and} \quad q \neq j \\ 1 + \frac{a_j}{t_j - a_j}, & n = q \quad \text{and} \quad q = j \\ 0, & \text{otherwise} \end{cases}$$

(note that the second and third lines correspond to the case in which the path which feeds path m is not the critical path and that the fourth line corresponds to the case in which the path which feeds path m is the critical path) and

$$[b_{i,j}]_m = a_q \left(\frac{s_j + \bar{D}^i - \frac{a_j \bar{D}^i}{\tau_j}}{1 - \frac{a_j}{\tau_j}} - \bar{D}^i \right).$$

Finally, consider $m \notin S_i$, $m \in R$, and $m \notin H^*_{S_i}$ (i.e., path m is not being serviced and is fed neither by a network input nor by a path which is being serviced). Because path m is fed neither by a network input nor by some path in S_i, it is clear that $y_m(p+1) = y_m(p)$ so that

$$[A_{i,j}]_{m,n} = \begin{cases} 1, & n = m \\ 0, & \text{otherwise} \end{cases}$$

and $[b_{i,j}]_m = 0$.

In the following, we shall let $\|\cdot\|$ denote the vector two-norm or the matrix norm induced by the vector two-norm: $\|x\| = (\sum_{i=1}^N x_i^2)^{\frac{1}{2}}$,

$x \in \mathbb{R}^N$, $\|A\| = \max_{\|x\|=1} \|Ax\|$, $A \in \mathbb{R}^{N \times N}$, and we shall let $\lambda(A)$ denote the set of eigenvalues of the matrix $A \in \mathbb{R}^{N \times N}$. Next, we formally specify the PPC policy. To do this, we specify an ordered list, ρ, of length N_ρ, whose members are elements of $\{1, 2, \cdots, M\}$. The elements of ρ are referenced by $\rho(1), \rho(2), \cdots, \rho(N_\rho)$. The list ρ specifies one period of a periodic servicing sequence. For example, once the PPC policy is activated, the periodic servicing sequence is $S_{\rho(1)}, S_{\rho(2)}, \cdots, S_{\rho(N_\rho)}, S_{\rho(1)}, S_{\rho(2)}, \cdots, S_{\rho(N_\rho)}, \cdots$. Of course, our intuition tells us that a necessary condition for the periodic servicing sequence to produce stable operation is that the sequence contain at least one occurrence of each S_i, $i \in \{1, 2, \cdots, M\}$.

Let the notation $\prod_{i=n,-1}^{m} A_{\phi(i)}$, $n \geq m$, where $A_{\phi(i)}$ denotes a particular square matrix for all $i \in \{m, m+1, \cdots, n\}$, denote the product $A_{\phi(n)} A_{\phi(n-1)} \cdots A_{\phi(m+1)} A_{\phi(m)}$. Let t_0 be the time at which the PPC policy is initiated. Using the model developed above, we can write

$$\begin{aligned}
y(1) &= A_{\rho(1),j_0} y(0) + b_{\rho(1),j_0} \\
y(2) &= A_{\rho(2),j_1} y(1) + b_{\rho(2),j_1} \\
&= A_{\rho(2),j_1} A_{\rho(1),j_0} y(0) + A_{\rho(2),j_1} b_{\rho(1),j_0} + b_{\rho(2),j_1} \\
&\vdots \\
y(N_\rho) &= \left(\prod_{i=N_\rho,-1}^{1} A_{\rho(i),j_{i-1}} \right) y(0) + b_{\rho(N_\rho),j_{N_\rho-1}} \\
&\quad + \sum_{i=1}^{N_\rho-1} \left(\prod_{k=N_\rho,-1}^{i+1} A_{\rho(k),j_{\rho(k)-1}} \right) b_{\rho(i),j_{\rho(i)-1}}
\end{aligned}$$

where j_i denotes the critical path for the production run beginning at time t_i.

There are $n_\rho \triangleq n_{\rho(1)} n_{\rho(2)} \cdots n_{\rho(N_\rho)}$ ways in which the functions of matrices $\prod_{i=N_\rho,-1}^{1} A_{\rho(i),j_{i-1}}$ and $b_{\rho(N_\rho),j_{N_\rho-1}} + \sum_{i=1}^{N_\rho-1} (\prod_{k=N_\rho,-1}^{i+1} A_{\rho(k),j_{\rho(k)-1}}) b_{\rho(i),j_{\rho(i)-1}}$ might be formed. Notice that n_i is the number of different critical paths that can feasibly result when paths in S_i are serviced. We form the sets $A^* \triangleq \{A_1^*, A_2^*, \cdots, A_{n_\rho}^*\}$ and $b^* \triangleq \{b_1^*, b_2^*, \cdots b_{n_\rho}^*\}$ to contain all possible formations of the matrix functions. This completes the definition of the PPC policy and the associated FMS model.

In what follows, we derive stability conditions and buffer bounds for FMS's that satisfy different topological conditions. We first consider an important class of systems with the topological property that $n_i = 1$ for all i, $1 \leq i \leq M$. Notice that this condition is satisfied by systems for which $|S_i| = 1$ for all i, $1 \leq i \leq M$, and that it is also effectively satisfied by systems in which any S_i with $|S_i| > 1$ contains only paths which are fed by network inputs. The last part of the previous statement is true because in set S_i with $|S_i| > 1$ which contain only paths which are fed by network inputs, after the initial production run, the critical path for all future production runs is determined by the fixed input and processing rates of the paths of S_i. Two examples of FMS with $n_i = 1$ are certain types of re-entrant lines and a feedforward line. In all systems with $n_i = 1$ for all i, $1 \leq i \leq M$, the sets A^* and b^* each contains just one element, A_1^* and b_1^*, respectively. Let $A \triangleq A_1^*$ and $b \triangleq b_1^*$. In terms of A and b, then, we have the iterative relation $y((k+1)N_\rho) = Ay(kN_\rho) + b$.

Theorem 2: If $n_i = 1$ for all i, $1 \leq i \leq M$, and if $\max(|\lambda(A)|) < 1$, then the PPC-controlled FMS is stable. Furthermore, $y(kN_\rho) = A^k y(0) + (\sum_{i=0}^{k-1} A^i) b$ and $\lim_{k\to\infty} y(kN_\rho) = (I - A)^{-1} b$.

Proof: That $y(kN_\rho)$ can be written as $y(kN_\rho) = A^k y(0) + (\sum_{i=0}^{k-1} A^i) b$ is easily shown via induction on the iterative relation $y((k+1)N_\rho) = Ay(kN_\rho) + b$. Given this, it is easy to see that $\lim_{k\to\infty} y(kN_\rho) = (I - A)^{-1} b$, because: 1) $\lim_{k\to\infty} A^k = 0$ and 2) $\sum_{i=0}^{\infty} A^i = (I - A)^{-1}$. Hence, the system is stable because $\|(I - A)^{-1}\| \|b\| < \infty$. □

Notice that the "bounds" provided by this result are actually exact characterizations of FMS behavior. In this case, we do not need to resort to norm-based buffer bounds. Notice also that the result is only valid once in every N_ρ times. In order to find the behavior at all times k, we must form N_ρ different A and b matrices and compute the behavior for each set. Each of the A and b matrix sets should correspond to the length-N_ρ service period starting at a different point. For example, if the original service period is S_1, S_2, S_3, we need to calculate three different sets of A and b matrices, with one corresponding to each of the following periods: S_1, S_2, S_3, S_2, S_3, S_1, and S_3, S_1, S_2. Next, notice that it is always possible to define the S_i so that $n_i = 1$ (e.g., choose the S_i so that $|S_i| = 1$ for all i). However, from a performance (and stability) perspective, this is clearly not always the best choice. For example, an FMS that, due to its topology, requires the defining of many short paths (such as the cellular structure I FMS of Section V) may perform badly or even become unstable if we take $|S_i| = 1$ for all i. These types of FMS generally require more than a single path to be processed at once.

We are now left to consider systems with $n_i > 1$ for some i, $1 \leq i \leq M$. Ideally, we would like to show that the sequence of critical paths of such systems eventually falls into periodic behavior. Such a result would eliminate the need for Theorem 3 below. While we have not been able to produce a simulation in which this eventual periodicity does not occur, we have not yet been able to prove that it must occur. In any case, we can find stability results for systems of this type; however, the resulting bounds are not nearly so sharp as the bounds in the $n_i = 1$ case.

Before presenting the stability result, we define some convenient notation. For a fixed product of matrices from the set A^*, $\prod_{i=r,-1}^{1} A_{\phi(i)}^*$, where $\phi(i)$ is an index into the set A^*, we define the *corresponding sum of products* to be the vector b that is appropriate in the iterative relation $y((k+1)rN_\rho) = (\prod_{i=r,-1}^{1} A_{\phi(i)}^*) y(krN_\rho) + b$. In other words, $b = b_{\phi(r)}^* + \sum_{i=1}^{r-1} (\prod_{j=r,-1}^{i+1} A_{\phi(j)}^*) b_{\phi(i)}^*$.

Theorem 3: Suppose that there exists integer $r > 0$ such that any product of r matrices from the set A^* has matrix two-norm less than one. Let A denote the r-length product with the largest norm, and let b denote the corresponding sum of products. The PPC-controlled FMS is stable and

$$\|y(krN_\rho)\| \leq \|A\|^k \|y(0)\| + \sum_{i=0}^{k-1} \|A\|^i \|b\|$$

so that $\lim_{k\to\infty} \|y(krN_\rho)\| \leq \frac{\|b\|}{1 - \|A\|}$.

Proof: From the definitions of A and b, it is clear that we can write $\|y((k+1)rN_\rho)\| \leq \|A\| \|y(krN_\rho)\| + \|b\|$. By applying induction to the above iterative relation, it easily follows that $\|y(krN_\rho)\| \leq \|A\|^k \|y(0)\| + \sum_{i=0}^{k-1} \|A\|^i \|b\|$. Because we have required that $\|A\| < 1$, it is also apparent that $\lim_{k\to\infty} \|y(krN_\rho)\| \leq \frac{\|b\|}{1 - \|A\|} < \infty$. □

The conditions of Theorem 3 are not completely satisfying for two reasons. First of all, we must check all length-r multiplicative combinations of the A_i^* matrices, regardless of whether all of the combinations are physically realizable. Secondly, we would like to avoid a norm-based result. A more exact characterization of FMS behavior, similar to that in Theorem 2, would require showing that after the PPC policy is enforced that eventually the resulting sequence of critical buffers becomes periodic. This appears to be a very difficult problem. Next, notice that even though the PPC policy prescribes a periodic service schedule, it is still a feedback policy in that it must know when production runs have been completed. Next, notice that for either $n_i = 1$ or $n_i > 1$, it may be the case that the production engineer may not be aware of a good choice of service period for

the PPC policy. In this case, one approach is to identify a stable service period and use it as a supervisor for another, more intuitive (e.g., PCLB) policy. As a supervisor, the purpose of the PPC policy is simply to guarantee stability of the system. In practice, the supervising PPC policy is invoked when some measure of system performance (e.g., the sum of the buffer levels) exceeds some preset threshold (i.e., once the sum of FMS buffer levels grows beyond some preset limit, a predefined, stable periodic service sequence is implemented).

Next, note that in simulation studies for the policies introduced here we have uncovered some interesting points [5]. First, in attempting to formulate a general rule of thumb for determining the suitability of PC control for a given FMS, we make the following observation: The less variance there is among processing rates along individual paths, the better PC policies will perform (with respect to distributed policies). The reason for this is that because PC policies mandate that all buffers on a given path be processed at a single rate (the minimum processing rate of all buffers on the path), any buffers on the path that are able to be processed at a faster rate than the minimum processing rate are constrained to be processed at a lower rate than they would be processed at in a distributed control scheme. In general, then, for systems with very high processing rate "skew" along individual paths, we may be wiser to choose a distributed policy. However, it may be possible to choose paths intelligently so as to minimize the adverse affects of processing rate skew.

Finally, we would like to emphasize that PC policies will not yield stability for all FMS that would be stable under a distributed scheduling approach where the FMS satisfies a capacity constraint. For example, if there is a high amount of "processing rate skew" along the paths PC policies may not be stable and a distributed policy may be.

REFERENCES

[1] J. R. Perkins and P. Kumar, "Stable, distributed, real-time scheduling of flexible manufacturing/assembly/disassembly systems," *IEEE Trans. Automat. Contr.*, vol. 34, pp. 139–148, Feb. 1989.

[2] P. Kumar and T. J. Seidman, "Dynamic instabilities and stabilization methods in distributed real-time scheduling of manufacturing systems," *IEEE Trans. Automat. Contr.*, vol. 35, pp. 289–298, Mar. 1990.

[3] S. H. Lu and P. Kumar, "Distributed scheduling based on due dates and buffer priorities," *IEEE Trans. Automat. Contr.*, vol. 36, pp. 1406–1416, Dec. 1991.

[4] C. Humes, Jr., "A regular stabilization technique: Kumar–Seidman revisited," *IEEE Trans. Automat. Contr.*, vol. 39, pp. 191–196, Jan. 1994.

[5] K. M. Passino and K. L. Burgess, *Stability Analysis of Discrete Event Systems*. New York: Wiley, 1998.

Decentralized H_∞-Controller Design for Nonlinear Systems

Guang-Hong Yang, Jianliang Wang, C. B. Soh, and James Lam

***Abstract*—This paper considers the decentralized H_∞-controller design problem for nonlinear systems. Sufficient conditions for the solution of the problem are presented in terms of solutions of Hamilton–Jacobi inequalities. The resulting design guarantees local asymptotic stability and ensures a predetermined L_2-gain bound on the closed-loop system.**

***Index Terms*—Decentralized control, Hamilton–Jacobi inequality, nonlinear H_∞ control, nonlinear system.**

I. INTRODUCTION

In the area of the decentralized control of large scale systems, numerous important advances have been accomplished in the past two decades [9], [11]. Recently, the decentralized H_∞-control problem for linear systems has been considered in [7], [8], [10], and [13]. In particular, Veillette *et al.* [13] presented a decentralized H_∞-controller design procedure in terms of solutions of the modified algebraic Riccati equations, and the result has also been extended to discrete-time linear systems [7]. In [8], another sufficient condition for the decentralized H_∞-control problem is derived, under which the decentralized solution can be constructed from the central controller solution from the standard H_∞-control theory in [2].

In recent years, the problem of central controller design to solve the H_∞-control problem (or in short, the central H_∞-control problem) for nonlinear systems has been extensively investigated by several authors [1], [3]–[6], [12]. In particular, Van der Schaft [12] has shown that the solution of the H_∞-control problem via state feedback can be determined from the solution of a Hamilton–Jacobi equation (or inequality), which is the nonlinear version of the Riccati equation for the corresponding linear H_∞-control problem. In the case of measurement feedback, a set of sufficient conditions has also been given in [1], [4], and [6] in terms of the solutions of a pair of Hamilton-Jacobi inequalities, and the necessity of these sufficient conditions has been discussed in [1] and [5].

In this paper, we consider the decentralized H_∞-control problem for nonlinear systems by using the Hamilton–Jacobi inequality approach. The results given in this paper are extensions of existing results on the linear decentralized H_∞-control problem [13], [8] and nonlinear central H_∞-control problem [4]. The paper is organized as follows. The system description and problem statement are given in Section II. The main results are given in Section III, followed by a numerical example in Section IV to illustrate the design procedure and the effectiveness of the proposed method. Finally, some concluding remarks are given in Section V.

Manuscript received September 11, 1996. Recommended by Associate Editor, J.-B. Pomet.

G.-H. Yang and C. B. Soh are with the School of Electrical and Electronic Engineering, Nanyang Technological University, 639798 Singapore.

J. Wang is with the School of Electrical and Electronic Engineering, Nanyang Technological University, 639798 Singapore (e-mail: ejlwang@ntu.edu.sg).

J. Lam is with the Department of Mechanical Engineering, University of Hong Kong, Hong Kong.

Publisher Item Identifier S 0018-9286(99)01320-3.

0018–9286/99$10.00 © 1999 IEEE