

# DISTRIBUTED SCHEDULING BASED ON DUE DATES AND BUFFER PRIORITIES\*

Steve H. Lu<sup>†</sup>      P. R. Kumar<sup>†</sup>

## Abstract

We are motivated by the problem of scheduling a large semiconductor manufacturing facility, where jobs of wafers, each with a desired due date, follow essentially the same route through the manufacturing system, returning several times to many of the service centers for the processing of successive layers. Neglecting the randomness introduced by yield, such a system can be modeled as a non-acyclic flow line. In such systems, it is important to reduce the mean delay, or equivalently the mean work-in-process, as well as the variance of the delay.

We analyze several distributed scheduling policies. We show that for a single non-acyclic flow line the First Buffer First Serve Policy (FBFS), which assigns priorities to buffers in the order that they are visited, is stable, whenever the arrival rate, allowing for some burstiness, is less than the system capacity. Similarly, the Last Buffer First Serve Policy (LBFS), where the priority ordering is reversed, is also stable. However not all buffer priority policies are stable, as witnessed by a counter-example. The well known Earliest Due Date Policy (EDD), where priority is based on the due date of a part, as well as another due date based policy of interest called the Least Slack Policy (LS), where priority is based on the “slack” of a part, defined as the due date minus an estimate of the remaining delay, are also proved to be stable. For systems with many part-types following different routes, we exhibit stable extensions of these policies.

We also present simulations which confirm our intuition that LBFS may well be the best policy for minimizing the mean delay at high load factors, while LS may well be the best policy for minimizing the variance of the delay. Finally some open problems are posed.

---

\*The research reported here has been supported in part by the National Science Foundation under Grant No. ECS-88-02576, the U.S.A.R.O. under contract No. DAAL-03-88-K-0046, and the Joint Services Electronics Program under Grant No. N00014-90-J-1270.

<sup>†</sup>Department of Electrical and Computer Engineering and the Coordinated Science Laboratory, 1101 West Springfield Ave., Urbana, IL 61801, USA. Please address all correspondence to the second author.

# 1 Introduction

A particular motivation for the investigations reported in this paper has been the issue of scheduling large semiconductor manufacturing facilities. In such systems, jobs of wafers, each with a prescribed due date, follow essentially the same route through the manufacturing system, and require virtually the same processing at identical stages of their route. A characteristic feature is that each wafer indeed visits many of the service centers several times, once for the treatment of each of the several “layers” of the resistive interconnect in the personalization process of the masterslice technology; see Dillinger [1]. Figure 1, which is a much simplified version of the actual system, illustrates this basic highly “re-entrant” structure. Each service center can consist of many identical “machines” in parallel. Neglecting the random effects due to yield, such a system can be modeled as a non-acyclic flow line, to a first approximation.

More generally, if there are many types of parts which follow different routes through the manufacturing system, then one obtains a manufacturing system comprised of several such non-acyclic flow lines.

Two important goals in scheduling such systems are to reduce the mean delay as well as the variance of the delay. A small value of the mean delay, or equivalently manufacturing cycle time, guarantees small work-in-process. On the other hand, a small value of the variance of the delay allows the system to reliably meet due dates. Thus we believe that an appropriate choice of a performance measure is (mean + 3 standard deviations), since it provides a reliable estimate of the delay.

Several scheduling policies are of interest. For a system consisting of a single non-acyclic flow line, such as in Figures 1 or 2, one such policy is the First Buffer First Serve Policy (FBFS) where each service center ranks its buffers according to the order in which they are visited, and the machines in the service center select “parts” from the heads of the buffers, giving priority to buffers visited *earlier* in the route. Another is the Last Buffer First Serve Policy (LBFS) where the priority ordering of the buffers is exactly the reverse. These are two examples of scheduling policies which are based on a *priority ordering* of the buffers.

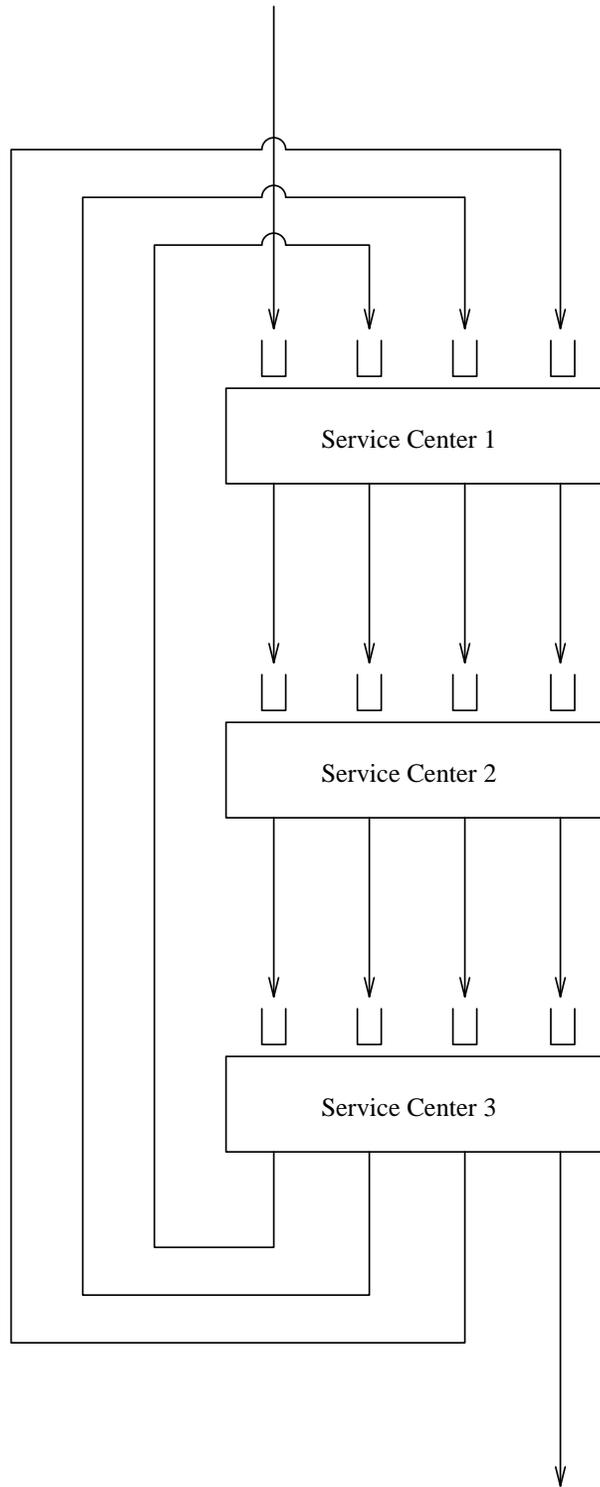


Figure 1: A much simplified schematic to illustrate the highly “re-entrant” structure of a semiconductor manufacturing system

An example of a *due date based* policy is the Earliest Due Date Policy (EDD) where at each service center the part with the earliest due date is processed next. Alternatively, in an attempt to meet due dates, one could somehow estimate the remaining delay yet to be experienced by a part, and then try to compensate for it. This gives rise to a Least Slack Policy (LS) where each service center chooses that part to work on next which has the minimum “slack,” where the slack of a part is defined as its due date minus an estimate of its remaining delay, which in turn depends on the buffer in which it is situated. Lastly, there is the well-known First Come First Serve Policy (FCFS) where the part which arrived first at the service center is chosen next, regardless of which buffer it is in. For systems with multiple non-acyclic flow lines, there are of course natural extensions of all these policies. For example, one could order all the buffers in the entire system in some way, which would yield a buffer priority policy, or one could use an LS or FCFS policy.

We are primarily interested in the LBFS and LS policies. Intuition suggests that since LBFS myopically attempts to clear parts from the system, it may well be the best candidate for minimizing the mean delay, especially at high load factors. On the other hand, since LS attempts to make all parts equally late or equally early with respect to their due dates, it should be an excellent choice for minimizing the variance of the delay. Thus, a “convex” combination of the two, which is itself an LS policy employing low values of the slacks, e.g. small proportions of the remaining delays, suggests itself as an appropriate choice for minimizing (mean + 3 standard deviations).

The first fundamental question that arises when considering such natural scheduling policies is whether they are “stable.” By stability we mean that the “delay” incurred by the parts, also known as the “cycle-time,” is bounded, whenever the arrival rate is within the capacity of the system. Equivalently, when combined with a release policy where parts are introduced into the system a certain time interval ahead of their due dates, we mean that the deviation of the actual production completion dates from the due dates is bounded. Finally, stability is also equivalent to requiring that “work in process” remains bounded.

While natural candidates for scheduling policies, as above, are mentioned frequently in the

literature, see Panwalker and Iskander [2] and Graves [3], they fall strictly outside the gamut of the available stability results for queueing networks, to the best of our knowledge. For example, in the excellent treatment of “Markovian” networks in Kelly [4], priority disciplines based on the “type” of a part are specifically excluded, see page 59 of [4]. In fact buffer priority policies, the EDD policy, and even the FCFS policy when the mean service times are different for different visits, are all excluded.

Our main results in this paper are the following. We prove that for the case of a single non-acyclic flow line, FBFS (Theorem 1) and LBFS (Theorem 3) are stable whenever the arrival rate, allowing for some burstiness, is less than the capacity of the bottleneck service center, i.e., the system capacity. The proof of stability for LBFS is based on a contractive estimate for the delay experienced by a part as a function of the number of parts in the system when it arrives (Theorem 2). This estimate is of independent interest and exhibits a “pipeline” like behavior of non-acyclic flow lines operating under LBFS. However, not all buffer priority policies are stable, as witnessed by a counter-example (Example 1). Regarding due date based scheduling policies, we show that LS (Theorem 4) and EDD (Corollary 1) are also stable. We have been unable to resolve whether FCFS is stable – a significant open question.

For systems consisting of multiple non-acyclic flow lines, we show that several buffer priority policies are stable. For example, any buffer priority ordering which is an interleaving of the individual FBFS orderings for each of the various non-acyclic flow lines is stable (Theorem 5). Also, any “uniform” concatenation of the separate LBFS orderings for each of the non-acyclic flow lines is stable (Theorem 5). Finally some combinations of the above two orderings are also stable (Theorem 5).

We also provide the results of some simulations for a highly re-entrant manufacturing system of the type shown in Figure 1. These simulations confirm our intuition that LBFS may well be the best overall choice for reliably minimizing the mean delay, especially at high load factors, while LS may be the best policy for minimizing the variance of the delay.

In this paper we do not address the important problem of dealing with random

“yield,” which leads naturally to a stochastic version of the problem discussed here. Our results, such as the contractive delay estimate of Theorem 2, could provide the deterministic backbone for a sample-path based stability proof in such a situation. To set this paper in context, we note that in contrast to Perkins and Kumar [5] and Kumar and Seidman [6], the systems addressed here do not have set-up times, thus allowing the possibility of better meeting due dates part by part.

The rest of this paper is organized as follows. In Section 2 we describe the system and the policies of interest. In Section 3 we provide a motivational counter-example to show that not all buffer priority policies are stable. In Sections 4 and 5 we prove the stability of FBFS and LBFS. In Section 6 we prove the stability of EDD and LS. In Section 7 we address manufacturing systems consisting of several non-acyclic flow lines, and prove the stability of several buffer priority scheduling policies. Section 8 indicates some generalizations of the results. In Section 9 we provide an account of our simulations. Finally, Section 10 provides some concluding remarks.

## 2 Description of System and Policies

For ease of presentation, we first describe a manufacturing system consisting of a single non-acyclic flow line. Later, in Section 7, we consider the more general system consisting of multiple non-acyclic flow lines.

The manufacturing system consists of  $S$  service centers labeled  $1, 2, \dots, S$ ; see Figure 2 for an example. Service center  $\sigma \in \{1, 2, \dots, S\}$  consists of a set  $M_\sigma$  containing  $m_\sigma$  identical machines in parallel, each of which can work on only one part at a time. Parts enter the manufacturing system at the service center  $\sigma_1 \in \{1, 2, \dots, S\}$  and proceed in sequence to service centers  $\sigma_2, \sigma_3, \dots, \sigma_\ell$  where  $\sigma_\ell$  is the last service center visited. At the  $i$ -th service center  $\sigma_i$  visited by a part, it is stored in a buffer labeled  $b_i$ , and requires  $\tau_i$  time units of continuous processing which can be provided by one of the  $m_{\sigma_i}$  machines located at the service center.

Let  $u(t)$  be the total number of parts released into the system in the time interval  $[0, t]$ .

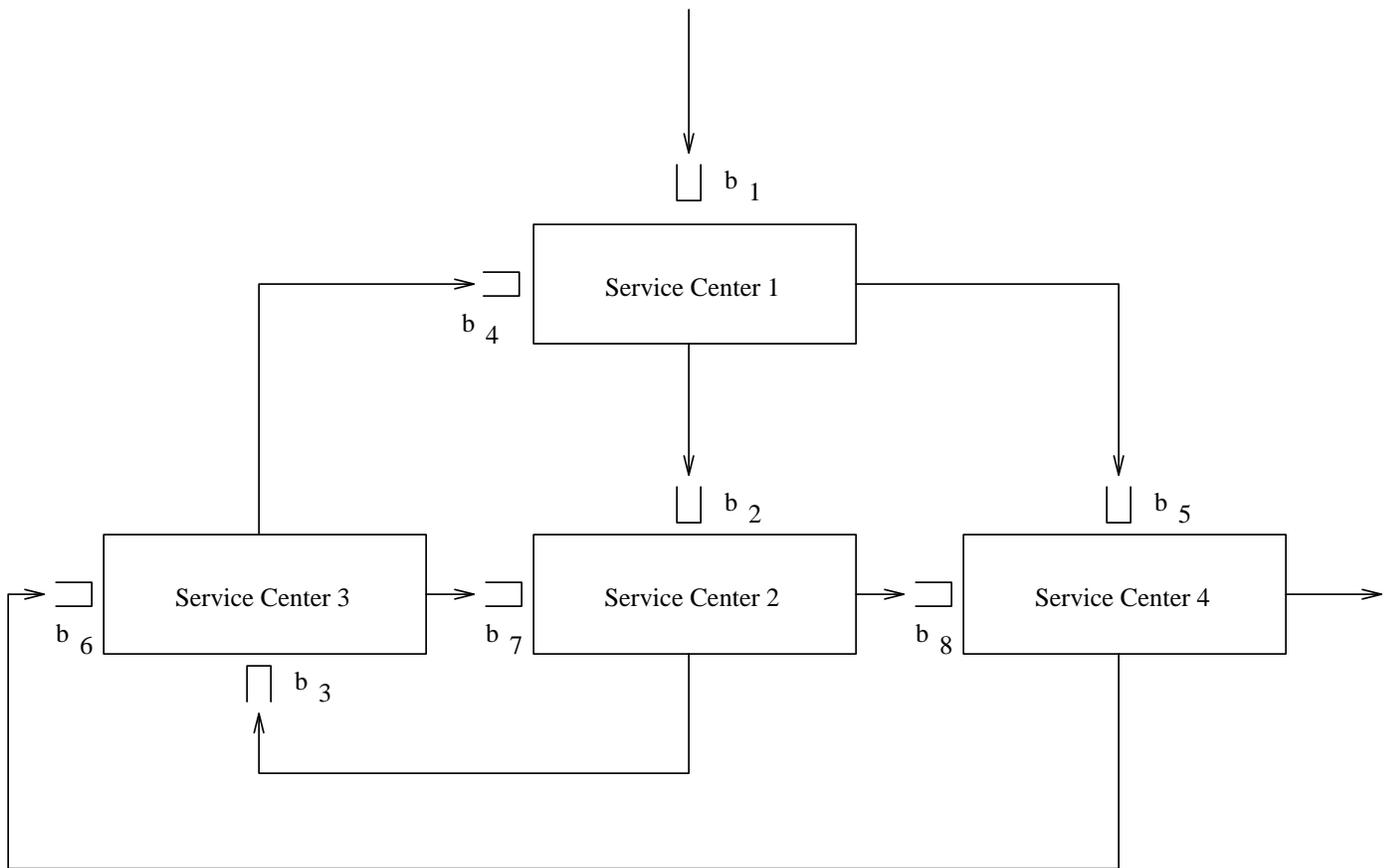


Figure 2: A non-acyclic flow line

We allow the arrivals to be bursty, but require that they satisfy the constraint:

$$u(t) - u(s) \leq \lambda(t - s) + \gamma \text{ for all } 0 \leq s \leq t, \quad (1)$$

for some constants  $\lambda \geq 0$  and  $\gamma \geq 0$ . We shall refer to  $\lambda$  as the *arrival rate*. The parameter  $\gamma$  allows for some burstiness in the arrivals. Such a non-probabilistic model of burstiness has been used earlier in Cruz [7].

Note that

$$w_\sigma := \sum_{\{i|\sigma_i=\sigma\}} \frac{\tau_i}{m_\sigma} \quad (2)$$

time units of work is brought in per machine at service center  $\sigma$  by each incoming part to the system. Accordingly, in order that the system be able to meet demand, we assume that the arrival rate  $\lambda$  satisfies

$$\rho := \max_\sigma \lambda w_\sigma < 1. \quad (3)$$

We shall refer to  $\rho$  as the *load* on the system.

Let  $B_\sigma$  denote the set of buffers served by service center  $\sigma$ . It is given by  $B_\sigma := \{b_i | \sigma_i = \sigma\}$ . All the scheduling policies we consider below are *non-idling*. By this we mean that a machine in  $M_\sigma$  is allowed to remain idle only if all the buffers in  $B_\sigma$  are empty. Otherwise, it is required to take up a part for processing, with the choice of the particular part being dictated by the scheduling rule. All our scheduling policies are *non-preemptive*, though all the results continue to hold for the preemptive case.

Let us now turn to the scheduling policies of interest.

### **First Buffer First Serve Policy (FBFS)**

An idle machine in  $M_\sigma$  takes up a part at the head of buffer  $b_i \in B_\sigma$  for processing only if all the other buffers  $b_j \in B_\sigma$  with  $j < i$ , if any, are empty. As an example, for the system of Figure 2, buffer 2 has higher priority over buffer 7 at service center 2.

### **Last Buffer First Serve Policy (LBFS)**

An idle machine in  $M_\sigma$  takes up a part at the head of buffer  $b_i \in B_\sigma$  for processing only if all  $b_j \in B_\sigma$  with  $j > i$ , if any, are empty. Thus, for the system of Figure 2, buffer 7 has higher priority over buffer 2 at service center 2.

### First Come First Serve Policy (FCFS)

An idle machine in  $M_\sigma$  takes up that part from the buffers in  $B_\sigma$  which arrived first at  $\sigma$ , on its current visit to the service center  $\sigma$ .

### Earliest Due Date Policy (EDD)

Each part  $\pi$  entering the system has a (desired) due date  $\delta(\pi)$ . An idle machine in  $M_\sigma$  takes up that part  $\pi$  from the buffers in  $B_\sigma$  which has the earliest due date  $\delta(\pi)$ .

### Least Slack Policy (LS)

For each buffer  $b_i$ , let there be assigned a number  $\zeta_i \geq 0$ . This number is meant to be an estimate of the remaining delay in the system for a part commencing service from  $b_i$ . (However,  $\zeta_i$  need not be such an estimate; it could also be based on some other consideration and the results would still apply). For a part  $\pi$  in  $b_i$ , we define its *slack* as  $\delta(\pi) - \zeta_i$ , i.e., as the due date minus the estimated future delay. An idle machine in  $M_\sigma$  takes up that part from the buffers in  $B_\sigma$  which has the smallest value for its slack.

It should be noted that FBFS and LBFS are examples of *buffer priority* policies, where an ordering of the buffers at each service center dictates the priority discipline. On the other hand EDD and LS are *due date based* policies.

Let us denote by  $\alpha(\pi)$  the time that a part is released into the system, i.e., the time that it *arrives* at the system. If parts arrive into the system in the order of their due dates, i.e.,  $\delta(\pi) \leq \delta(\pi') \Rightarrow \alpha(\pi) \leq \alpha(\pi')$ , then it is easy to check that LBFS coincides with EDD. The policy FCFS is of course well known. Finally LS is also a natural candidate for a scheduling policy. It is based on making a reasonable effort to meet due dates. In that sense one would expect that it attains a low “variance” of the difference between the actual finish time of parts and their desired due dates. We shall comment more about the performances of all the policies in Section 9. Note also that EDD is a special case of LS, since we can simply choose  $\zeta_i \equiv 0$ .

All the scheduling policies above are “distributed,” in that they can be implemented separately at each service center, without requiring any coordination of action or sharing

of information between service centers. Panwalker and Iskander [2] refer to such policies as “local dispatch rules.”

Let  $e(\pi)$  denote the time that a part  $\pi$  exits the system. We shall say that a scheduling policy is *stable* if for any arrival rate  $\lambda$  satisfying (3),

$$e(\pi) - \alpha(\pi) \leq \Gamma \text{ for all } \pi \tag{4}$$

for some  $\Gamma \geq 0$  (which could depend on the initial condition as well as  $\lambda$ ). We emphasize that our definition above requires “stability” for *all* arrival rates  $\lambda$  within the capacity of the system as given by (3).

Since  $e(\pi) - \alpha(\pi)$  is the *delay* or *cycle time* of a part, the stability of a policy simply guarantees a bounded delay for all parts. By Little’s Theorem, it also guarantees that the maximum “work in process” in the manufacturing system is bounded.

We note that LBFS myopically attempts to empty the system of parts. Thus one would expect that it tends to minimize the number of parts in the system, or equivalently the *mean* delay, especially at high load factors. On the other hand, LS attempts to make all parts equally late or early, and thus tends to minimize the *variance* of the delay. Both these intuitions are confirmed by the simulations presented in Section 9.

### 3 An Unstable Buffer Priority Policy

Both the FBFS and LBFS policies are particular examples of scheduling policies based on a buffer priority ordering. Are *all* buffer priority policies stable for all arrival rates satisfying (3)? As the following counter-example shows, the answer is in the negative. It clearly exhibits the need to address the issue of stability of scheduling policies.

**Example 1.** Consider a system with two service centers, each consisting of just one machine, as shown in Figure 3. A single part-type visits  $b_1$  located at service center 1, then  $b_2$  located at service center 2, then  $b_3$  located at service center 2, and finally  $b_4$  located at service center 1. The arrival rate is  $\lambda = 1$ , and we simply assume that parts arrive periodically. The

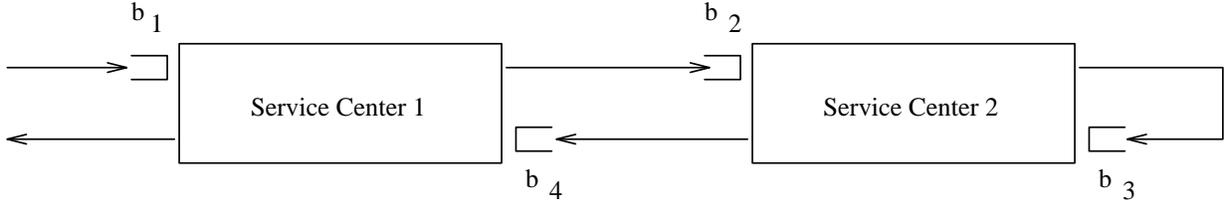


Figure 3: System of Example 1.

processing times are  $\tau_1 = \tau_3 = 0$  and  $\tau_2 = \tau_4 = \frac{2}{3}$ . At service center 1, priority is given to buffer  $b_4$ , while at service center 2 priority is given to buffer  $b_2$ .

Suppose that at time  $0^-$ ,  $b_1$  has  $x_1$  parts, while  $b_2$ ,  $b_3$  and  $b_4$  are empty. At time  $0^-$ , these  $x_1$  parts are immediately finished with their servicing (since  $\tau_1 = 0$ ) and passed on to buffer  $b_2$ , which then begins to work on them. At time  $2x_1^-$ ,  $b_2$  is empty, since these  $x_1$  parts as well as those arriving at the times  $0, 1, \dots, 2x_1 - 1$  have been processed. At time  $2x_1^-$  service center 2 then works on the  $3x_1$  parts in  $b_3$  and instantly transmits them to  $b_4$  which then starts work on them, since it has higher priority over  $b_1$ . These  $3x_1$  parts are completed at time  $4x_1^-$ . At time  $4x_1^-$ , buffer  $b_1$  has  $2x_1$  parts which have arrived at the times  $2x_1, 2x_1 + 1, \dots, 4x_1 - 1$ . This is the same initial condition as at  $0^-$ , except that the contents of  $b_1$  are doubled. The cycle thus repeats itself indefinitely, yielding instability.  $\square$

It would be useful to obtain another counterexample which does not involve zero processing times at any service centers.

## 4 The FBFS Policy

The main result of this section is that the FBFS policy is stable.

**Theorem 1: Stability of FBFS.** *The First Buffer First Serve Policy (FBFS) is stable whenever the arrivals satisfy (1) and the arrival rate  $\lambda$  is within the capacity of the system, i.e., (3) is satisfied.*

**Proof.** Let  $x_i(t)$  denote the number of parts in buffer  $b_i$  at time  $t$ , not counting any in service. At time 0, we shall allow the system to have some nonempty initial state  $x_i(0)$ , plus

some parts in service.

Let  $\alpha_i(\pi)$  denote the time instant that a part  $\pi$  arrives at buffer  $b_i$ . (Thus  $\alpha_1(\pi) = \alpha(\pi)$ , the release time of part  $\pi$  into the system).

We shall say that a time interval  $[T_1, T_2]$  is an *i-busy period* if at every time instant in this interval there is some part in some buffer  $b_j$  at the service center  $\sigma_i$  with  $j \leq i$ , which is *waiting* for service, or more formally,

$$\sum_{\{j|\sigma_j=\sigma_i \text{ and } j \leq i\}} x_j(t) > 0 \text{ for all } T_1 \leq t \leq T_2.$$

Our proof is by induction on the following hypothesis: “There exist  $T^{(i)}, \Gamma^{(i)}$  for  $i = 1, 2, \dots, \ell$  such that if  $[T_1, T_2]$  is an *i-busy period* with  $T_1 > T^{(i)}$ , then

$$T_2 - T_1 \leq \Gamma^{(i)},$$

i.e., after a certain finite time  $T^{(i)}$  all *i-busy periods* have duration less than  $\Gamma^{(i)}$ .”

Consider  $i = 1$ . Let  $T^{(1)}$  be the first time that buffer  $b_1$  empties. (Clearly  $T^{(1)}$  is finite, or else the service center is forever working on parts in  $b_1$ , completing service at a rate faster than the arrivals by (3), which is impossible). Consider any time  $T_1 > T^{(1)}$  at which a 1-busy period  $[T_1, T_2]$  commences. Thus at time  $T_1^-$  no part in  $b_1$  is waiting for service, and a part arrives into buffer  $b_1$  at time  $T_1$ . By the FBFS priority, and allowing for the non-preemptive service discipline, this newly arriving part has to wait no more than  $\bar{\tau} := \max_j \tau_j$  time units to commence service. If  $T_2 \geq T_1 + \bar{\tau}$ , then throughout the time interval  $[T_1 + \bar{\tau}, T_2]$  all the  $m_\sigma$  machines at the service center  $\sigma_1$  are busy working on parts from buffer  $b_1$ . Hence the number of service *completions* in  $[T_1 + \bar{\tau}, T_2]$  is  $\geq \left(\frac{T_2 - T_1 - 2\bar{\tau}}{\tau_1}\right) m_{\sigma_1}$ , while the number of arrivals in  $[T_1, T_2]$  is  $\leq \lambda(T_2 - T_1) + \gamma$ . Hence  $\lambda(T_2 - T_1) + \gamma \geq \frac{(T_2 - T_1 - 2\bar{\tau})}{\tau_1} m_{\sigma_1}$ , which yields  $T_2 - T_1 \leq \Gamma^{(1)}$  where  $\Gamma^{(1)} := \left(2\bar{\tau} + \frac{\gamma\tau_1}{m_{\sigma_1}}\right) \left(1 - \frac{\lambda\tau_1}{m_{\sigma_1}}\right)^{-1}$ . This proves the induction hypothesis for  $i = 1$ .

Suppose now that the induction hypothesis is true for  $1, 2, \dots, i - 1$ . Consider  $i$ . Since for  $j = 1, 2, \dots, i - 1$ , the length of no *j-busy period* after  $T^{(j)}$  exceeds  $\Gamma^{(j)}$ , it follows that the delay of a part at  $b_j$  is  $\leq \Gamma^{(j)} + \tau_j$ , if it arrived at buffer  $b_j$  after  $T^{(j)}$ . Hence the delay from

the entry of a part into the system till it exits from  $b_j$  is  $\leq \bar{\Gamma}$  where  $\bar{\Gamma} := \sum_{j=1}^{i-1} (\Gamma^{(j)} + \tau_j)$ , if it arrived into the system after  $\bar{T} := \max_{1 \leq j \leq i-1} T^{(j)}$ . Let  $T' := \max(\bar{\Gamma} + \bar{T}$ , time that all of the original parts in the buffers  $b_1, b_2, \dots, b_i$  at time 0 have exited from  $b_i$ ). Define  $T^{(i)} :=$  first time instant after  $T'$  that all the buffers  $b_j$  with  $j \leq i$  located at service center  $\sigma_i$  are simultaneously empty. It is easy to see that  $T^{(i)} < +\infty$ .

Thus parts arriving at a buffer  $b_j$  with  $j \leq i$  in any time interval  $[s, t]$  with  $s \geq T^{(i)}$  must have arrived into the system at some time in the time interval  $[s - \bar{\Gamma}, t]$ . By the arrival assumption (1), the total number of parts that have entered  $b_j$  in  $[s, t]$  is  $\leq \lambda(t - s) + (\lambda\bar{\Gamma} + \gamma)$  for  $j = 1, 2, \dots, i$ , and  $T^{(i)} \leq s \leq t$ . Let  $[T_1, T_2]$  be an  $i$ -busy period commencing at  $T_1 > T^{(i)}$ . By an argument similar to that made earlier, if  $T_2 \geq T_1 + \bar{\tau}$ , then throughout  $[T_1 + \bar{\tau}, T_2]$ , the  $m_{\sigma_i}$  machines at service center  $\sigma_i$  are busy working on the parts that arrived to the buffers  $b_j \in B_{\sigma_i}$  with  $j \leq i$  during the time interval  $[T_1, T_2]$ . These arrivals bring in

$$\leq \sum_{\{j|\sigma_j=\sigma_i \text{ and } j \leq i\}} [\lambda\tau_j(T_2 - T_1) + \lambda\tau_j\bar{\Gamma} + \gamma\tau_j]$$

time units of work for the  $m_{\sigma_i}$  machines. Hence

$$(T_2 - T_1 - 2\bar{\tau})m_{\sigma_i} \leq \sum_{\{j|\sigma_j=\sigma_i \text{ and } j \leq i\}} [\lambda\tau_j(T_2 - T_1) + \lambda\tau_j\bar{\Gamma} + \gamma\tau_j].$$

This yields  $(T_2 - T_1) \leq \Gamma^{(i)}$  where

$$\Gamma^{(i)} := \left[ 2\bar{\tau} + \sum_{\{j|\sigma_j=\sigma_i \text{ and } j \leq i\}} \frac{(\lambda\tau_j\bar{\Gamma} + \gamma\tau_j)}{m_{\sigma_i}} \right] \left[ 1 - \sum_{\{j|\sigma_j=\sigma_i \text{ and } j \leq i\}} \frac{\lambda\tau_j}{m_{\sigma_i}} \right].$$

This completes the induction proof.

Note that we have thereby also shown that the delay of a part in the system is  $\leq \sum_{j=1}^{\ell} (\Gamma^{(j)} + \tau_j)$  for parts arriving after  $T^{(\ell)}$ , thus proving the Theorem.  $\square$

A careful reading of the above proof shows that the bound on the delays experienced by parts entering after the transient period  $[0, T^{(\ell)}]$ , is independent of the initial state  $\{x_i(0)\}$ . On the other hand, the duration of the transient period  $[0, T^{(\ell)}]$  does depend on the initial state  $\{x_i(0)\}$ , as it should.

## 5 The LBFS Policy

We now turn to the LBFS policy, which is more in the nature of a “pull” policy. Its proof of stability is much more subtle than that for the FBFS policy. It relies on the following key “contractive” estimate of the delay experienced by a part as a function of the number in the system at its arrival time.

**Theorem 2: Contractive Estimate for the Delay under LBFS.** *With  $w_\sigma$  defined as in (2), let  $\bar{w} := \max_\sigma w_\sigma$  be the maximum work brought per machine at a service center by an incoming part. There exists a constant  $c(\epsilon)$  such that if there are  $x$  parts in the system when a part  $\pi$  arrives, then the delay experienced by  $\pi$  satisfies*

$$e(\pi) - \alpha(\pi) \leq c(\epsilon) + (\bar{w} + \epsilon)x \text{ for every } \epsilon > 0. \quad (5)$$

**Proof.** The key idea on which the proof hinges is that under the LBFS policy, parts entering the system *after* a part  $\pi$  cannot interfere with part  $\pi$  except for the minor delay caused by the non-preemptive discipline. Thus the delay experienced by a part is almost completely due to its being blocked by parts *ahead* of it in the system.

Consider the system consisting of only the buffers  $B^{(k)} := \{b_k, b_{k+1}, \dots, b_\ell\}$ . Let

$$w^{(k)} := \max_{k \leq j \leq \ell} \left[ \sum_{\{i | \sigma_i = \sigma_j \text{ and } i \geq k\}} \frac{\tau_i}{m_{\sigma_j}} \right] \quad (6)$$

be the maximum work brought per machine at a service center due to a part in buffer  $b_k$  (i.e., compared to  $\bar{w}$ , we simply ignore the loads on the service centers due to the service needs of  $\{b_1, \dots, b_{k-1}\}$ ).

Our proof is by induction starting from the end of the system (consistent with the “pull” nature of the policy), and is based on the following induction hypothesis:

“For every  $\epsilon > 0$ , there exists a constant  $c^{(k)}(\epsilon)$  such that if at a certain time instant a part  $\pi$  is in  $B^{(k)}$  and there are  $x$  parts ahead of it, then the *remaining* delay of part  $\pi$  in the system is  $\leq c^{(k)}(\epsilon) + (w^{(k)} + \epsilon)x$  for every  $\epsilon > 0$ .”

Consider  $k = \ell$ , where the truncated system  $B^{(\ell)}$  consists only of the last buffer  $b_\ell$ . Thus the part  $\pi$  is the  $(x + 1)$ -th part in  $b_\ell$ . The remaining delay of  $\pi$  is  $\leq (\bar{\tau} + \tau_\ell) + \left(\frac{\tau_\ell}{m_{\sigma_\ell}}\right)x$ , with

$$\bar{\tau} := \max_j \tau_j. \quad (7)$$

(The  $\bar{\tau}$  term allows for the delay caused by the non-preemptive discipline, the  $\left(\frac{x}{m_{\sigma_\ell}}\right)\tau_\ell$  term accounts for the time taken to process the  $x$  parts ahead of  $\pi$  in  $b_\ell$ , and  $\tau_\ell$  represents the processing time of  $\pi$  itself). Thus the induction hypothesis is true for  $k = \ell$ , with

$$c^{(\ell)}(\epsilon) := (\bar{\tau} + \tau_\ell) \text{ for all } \epsilon > 0, \quad (8)$$

since  $w^{(\ell)} = \frac{\tau_\ell}{m_{\sigma_\ell}}$  by (6).

Suppose now that the induction hypothesis is true for  $k + 1, k + 2, \dots, \ell$ . We will now show that it is valid for  $k$  with  $c^{(k)}(\epsilon)$  defined as,

$$c^{(k)}(\epsilon) := c^{(k+1)}(\epsilon) + \max\{2\ell\bar{\tau}, (w^{(k+1)} + \epsilon) \left\lceil \frac{2c^{(k+1)}(\frac{\epsilon}{2})}{\epsilon} \right\rceil + 2\bar{\tau}, c^{(k+1)}(\epsilon)\}. \quad (9)$$

Thus our task is to show that if at a certain time instant, say  $t = 0$  for convenience, a part  $\pi$  is in  $b_k$  and there are  $x$  parts ahead of it in  $B^{(k)}$ , then the exit time of  $\pi$  satisfies,

$$e(\pi) \leq c^{(k)}(\epsilon) + (w^{(k)} + \epsilon)x \text{ for all } \epsilon > 0. \quad (10)$$

Let  $\pi'$  be the part just ahead of  $\pi$  in  $B^{(k)}$ . (Thus, while  $\pi$  is the  $(x + 1)$ -th part in  $B^{(k)}$ ,  $\pi'$  is the  $x$ -th part in  $B^{(k)}$ ). We claim that the exit times  $e(\pi)$  and  $e(\pi')$  satisfy

$$e(\pi) \leq e(\pi') + \left( \sum_{j=k}^{\ell} \tau_j \right) + (\ell - k + 1)\bar{\tau}. \quad (11)$$

To see why this is true, consider the system at the time instant  $e(\pi')$ . At that time,  $\pi$  has the highest priority at all remaining buffers in the system. Hence it can be delayed at each buffer by at most  $\bar{\tau}$  (due to the non-pre-emptive discipline) plus a service time. Thus it exits within  $\sum_{j=k}^{\ell} (\tau_j + \bar{\tau})$  time units after  $e(\pi')$ , justifying (11).

Suppose that at time 0,  $\pi$  is the  $n$ -th part in  $b_k$ . We will show (10) by induction on  $n$ .

Consider  $n = 1$ . Then clearly  $\pi'$ , the part just ahead of  $\pi$ , is in  $B^{(k+1)}$ , since  $\pi$  is the only part in  $b_k$ . Since the induction hypothesis (10) is true for  $k + 1$ ,

$$e(\pi') \leq c^{(k+1)}(\epsilon) + (w^{(k+1)} + \epsilon)(x - 1).$$

From (11) we obtain

$$\begin{aligned} e(\pi) &\leq \left[ c^{(k+1)}(\epsilon) + \sum_{j=k}^{\ell} \tau_j + (\ell - k + 1)\bar{\tau} \right] + (w^{(k+1)} + \epsilon)(x - 1) \\ &\leq c^{(k+1)}(\epsilon) + 2\ell\bar{\tau} + (w^{(k)} + \epsilon)x \\ &\leq c^{(k)}(\epsilon) + (w^{(k)} + \epsilon)x \end{aligned}$$

where the second inequality follows from (6) by noting that  $w^{(k)} \geq w^{(k+1)}$ , and the third inequality follows from (9).

Suppose now that the result (10) has been proved if  $\pi$  is the 1st, 2nd,  $\dots$ , or  $(n - 1)$ th part in  $b_k$ . Suppose  $\pi$  is the  $n$ -th part in  $b_k$ .

Let  $T$  be the time instant at which  $\pi$  enters  $B^{(k+1)}$ , and let  $y$  be the number of parts ahead of  $\pi$  at the time  $T$  when it enters  $B^{(k+1)}$ . There are three cases to consider.

**Case 1.** Suppose  $y \leq \Gamma$ , where

$$\Gamma := \left\lceil \frac{2c^{(k+1)}\left(\frac{\epsilon}{2}\right)}{\epsilon} \right\rceil. \quad (12)$$

Then note that,

$$e(\pi) \leq T + c^{(k+1)}(\epsilon) + (w^{(k+1)} + \epsilon)\Gamma \text{ for all } \epsilon > 0, \quad (13)$$

since after entering  $B^{(k+1)}$  at time  $T$  part  $\pi$  spends only a further time  $\leq c^{(k+1)}(\epsilon) + (w^{(k+1)} + \epsilon)\Gamma$  before exiting the system. Let us now estimate  $T$ . If  $T \geq 2\bar{\tau}$ , then allowing for the non-preemption discipline, during the time interval  $[\bar{\tau}, T - \bar{\tau}]$  all the  $m_{\sigma_k}$  machines at the service center  $\sigma_k$  are continuously working on parts ahead of  $\pi$ . The total work for the service center  $B_{\sigma_k}$  contained in these  $x$  parts is  $\leq \sum_{\{i|\sigma_i=\sigma_k \text{ and } i \geq k\}} \tau_i x$ . Hence

$$m_{\sigma_k}(T - 2\bar{\tau}) \leq x \sum_{\{i|\sigma_i=\sigma_k \text{ and } i \geq k\}} \tau_i.$$

Thus

$$T \leq 2\bar{\tau} + \left[ \sum_{\{i|\sigma_i=\sigma_k \text{ and } i \geq k\}} \frac{\tau_i}{m_{\sigma_k}} \right] x.$$

Hence combining with (13), we get

$$e(\pi) \leq [c^{(k+1)}(\epsilon) + (w^{(k+1)} + \epsilon)\Gamma + 2\bar{\tau}] + \left[ \sum_{\{i|\sigma_i=\sigma_k \text{ and } i \geq k\}} \frac{\tau_i}{m_{\sigma_k}} \right] x \text{ for all } \epsilon > 0,$$

which satisfies (10) by virtue of (12), (9) and (6).

**Case 2.** Suppose  $\Gamma < y \leq n - 1$ . This corresponds to the case where the part  $\pi''$  with the highest priority in the system at the time instant  $T$  (i.e., the part which is closest to the exit), was originally in position  $(n - y)$  in the buffer  $b_k$  at time 0, with  $y > \Gamma$ . Since the induction hypothesis is true for position  $n - y < n$ , it applies to part  $\pi''$  which has  $x - y$  parts ahead of it in  $B^{(k)}$  at time 0, and so,

$$T \leq e(\pi'') \leq c^{(k)}(\epsilon) + (w^{(k)} + \epsilon)(x - y).$$

Also, at time  $T$ , part  $\pi$  is in  $B^{(k+1)}$  and there are  $y$  parts ahead of it in  $B^{(k+1)}$ . Hence after entering  $B^{(k+1)}$  at time  $T$  it spends  $\leq c^{(k+1)}\left(\frac{\epsilon}{2}\right) + (w^{(k+1)} + \frac{\epsilon}{2})y$  time units before exiting.

Thus,

$$\begin{aligned} e(\pi) &\leq c^{(k)}(\epsilon) + c^{(k+1)}\left(\frac{\epsilon}{2}\right) + \left(w^{(k+1)} + \frac{\epsilon}{2}\right)y + (w^{(k)} + \epsilon)(x - y) \\ &\leq c^{(k)}(\epsilon) + c^{(k+1)}\left(\frac{\epsilon}{2}\right) - \frac{\epsilon}{2}y + (w^{(k+1)} - w^{(k)})y + (w^{(k)} + \epsilon)x \\ &\leq c^{(k)}(\epsilon) + c^{(k+1)}\left(\frac{\epsilon}{2}\right) - \frac{\epsilon}{2}\Gamma + (w^{(k)} + \epsilon)x, \quad \text{since } y > \Gamma \text{ and } w^{(k+1)} \leq w^{(k)}, \\ &\leq c^{(k)}(\epsilon) + (w^{(k)} + \epsilon)x, \text{ from (12),} \end{aligned}$$

which again satisfies (10).

**Case 3.** Suppose  $y > \Gamma$  and  $y \geq n$ . Then the part  $\pi''$  with the highest priority in  $B^{(k+1)}$  at time  $T$  was originally in  $B^{(k+1)}$  at time 0 and had  $x - y$  parts ahead of it. Since the induction hypothesis is true for  $k + 1$ ,

$$T \leq e(\pi'') \leq c^{(k+1)}(\epsilon) + (w^{(k+1)} + \epsilon)(x - y).$$

Moreover, after entering  $B^{(k+1)}$  at time  $T$ , part  $\pi$  thereafter spends  $\leq c^{(k+1)}(\epsilon) + (w^{(k+1)} + \epsilon)y$  time units before leaving the system, and so,

$$e(\pi) \leq 2c^{(k+1)}(\epsilon) + (w^{(k+1)} + \epsilon)x$$

which again satisfies (10) by virtue of (9).

This completes the induction and the proof.  $\square$

The above contractive estimate for the delay shows that non-acyclic systems operating under LBFS exhibit a “pipeline-like” behavior, similar to acyclic systems.

**The Pipeline Property of LBFS.** *Under the same conditions as in Theorem 2,*

$$e(\pi) - \alpha(\pi) \leq \bar{w}x + o(x).$$

**Proof.** From (8) and (9) we can compute  $c(\epsilon) := c^{(1)}(\epsilon)$ . This gives the bound (5) on the delay as,

$$e(\pi) - \alpha(\pi) \leq \sum_{i=0}^{\ell-1} \frac{a_i}{\epsilon^i} + (\bar{w} + \epsilon)x \text{ for all } \epsilon > 0,$$

for some constants  $\{a_i\}$ . The result is then obtained by choosing  $\epsilon$  as a function of  $x$  to minimize the right hand side above.  $\square$

Whether the term  $o(x)$  above can be replaced by  $O(1)$  is an open question.

With the “contractivity” property for the delay given by Theorem 2 in hand, we can establish the stability of LBFS.

**Theorem 3: Stability of LBFS.** *Suppose that the arrivals satisfy (1) with the arrival rate satisfying the capacity condition (3). Let  $x(0)$  denote the number of parts that are initially in the system at time 0. Then the number  $x(t)$  of parts in the system at time  $t$  satisfies:*

$$i) \ x(t) \leq \max \left\{ (1 + \rho + \lambda\epsilon)x(0) + \lambda c(\epsilon) + \gamma, \frac{2(\lambda c(\epsilon) + \gamma)}{1 - \rho - \lambda\epsilon} \right\} \text{ for all } t \geq 0, \text{ which bounds the transient behavior.}$$

$$ii) \ \limsup_{t \rightarrow \infty} x(t) \leq \frac{\lambda c(\epsilon) + \gamma}{1 - \rho - \lambda\epsilon}, \text{ which bounds the asymptotic behavior.}$$

*The above statements are valid for all  $\epsilon > 0$  small enough that  $(1 - \rho - \lambda\epsilon) > 0$ .*

**Proof.** 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  $t_{k-1}$ .

(Above, if buffers  $b_i$  for  $i < j$  are empty while buffer  $b_j$  is nonempty, then the part at the tail end of buffer  $b_j$  is what we mean by the part at the “beginning” of the system. There is a slight modification of the following proof if the system is empty of parts at time  $t_{k-1}$ ; the minor details are left to the reader). Since there are  $x(t_{k-1}) - 1$  parts ahead of such a part at time  $t_{k-1}$ , we have

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

by Theorem 2. Moreover, since parts exit the system in the order that they enter it,  $x(t_k)$  is equal to the number of parts that arrived into the system in the time interval  $[t_{k-1}, t_k]$ .

Hence

$$\begin{aligned} x(t_k) &\leq \lambda(t_k - t_{k-1}) + \gamma \\ &\leq (\lambda c(\epsilon) + \gamma) + \lambda(\bar{w} + \epsilon)x(t_{k-1}). \end{aligned} \tag{14}$$

This gives,

$$\limsup_k x(t_k) \leq \frac{\lambda c(\epsilon) + \gamma}{1 - \rho - \lambda \epsilon}, \text{ since } \rho = \lambda \bar{w}.$$

Moreover note from (14) that if  $x(0) \leq \frac{\lambda c(\epsilon) + \gamma}{1 - \rho - \lambda \epsilon}$ , then  $x(t_k) \leq \frac{\lambda c(\epsilon) + \gamma}{1 - \rho - \lambda \epsilon}$  for all  $k \geq 0$ . On the other hand from (14) we also see that if  $x(0) > \frac{\lambda c(\epsilon) + \gamma}{1 - \rho - \lambda \epsilon}$ , then the sequence  $\{x(t_k)\}$  is monotone decreasing until it dips below  $\frac{\lambda c(\epsilon) + \gamma}{1 - \rho - \lambda \epsilon}$ , after which it stays below  $\frac{\lambda c(\epsilon) + \gamma}{1 - \rho - \lambda \epsilon}$  for all further  $k$ . Hence, allowing for both these possibilities,

$$x(t_k) \leq \max \left\{ x(0); \frac{\lambda c(\epsilon) + \gamma}{1 - \rho - \lambda \epsilon} \right\} \text{ for all } k \geq 0.$$

Now, for any  $t \in [t_{k-1}, t_k]$ ,

$$\begin{aligned} x(t) &\leq x(t_{k-1}) + \text{number of parts that arrived in } [t_{k-1}, t_k] \\ &\leq x(t_{k-1}) + \lambda(t_k - t_{k-1}) + \gamma \\ &\leq (1 + \rho + \lambda \epsilon)x(t_{k-1}) + \lambda c(\epsilon) + \gamma \end{aligned}$$

which yields,

$$\limsup_{t \rightarrow \infty} x(t) \leq \frac{2(\lambda c(\epsilon) + \gamma)}{1 - \rho - \lambda \epsilon} \quad (15)$$

as well as,

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

The factor 2 in (15) is unnecessary and can be removed by a “backward” recursion. Consider a time  $t \geq t_{k+1}$ . Let  $t^{(k)} := t$  and recursively define  $t^{(j)}$  as the arrival time of the part which is at the *finish* of the system at  $t^{(j+1)}$ . (If buffers  $b_i$  for  $i > j$  are empty while buffer  $b_j$  is nonempty, then the part at the head of buffer  $b_j$  is what we mean by the part at the “finish” of the system). Note that  $t^{(0)} \geq 0$ , and as earlier,

$$t^{(j)} - t^{(j-1)} \leq c(\epsilon) + (\bar{w} + \epsilon)x(t^{(j-1)}),$$

and

$$x(t^{(j)}) \leq \lambda(t^{(j)} - t^{(j-1)}) + \gamma.$$

This yields,

$$x(t^{(j)}) \leq (\rho + \lambda \epsilon)x(t^{(j-1)}) + (\lambda c(\epsilon) + \gamma)$$

and therefore

$$x(t) = x(t^{(k)}) \leq (\rho + \lambda \epsilon)^k x(t^{(0)}) + \frac{\lambda c(\epsilon) + \gamma}{1 - \rho - \lambda \epsilon}.$$

The above is true for all  $t \geq t^{(k+1)}$ . Hence,

$$\limsup_{t \rightarrow \infty} x(t) \leq \frac{\lambda c(\epsilon) + \gamma}{1 - \rho - \lambda \epsilon},$$

proving the Theorem. □

It should be noted that the bound (i) is valid for all time, and it covers the transient period as well. Therefore it is necessarily  $x(0)$  dependent. On the other hand, (ii) bounds only the asymptotic behavior and it is independent of  $x(0)$ .

## 6 The EDD and LS Policies

We now take up the due-date based policies EDD and LS for consideration.

In order to obtain any reasonable results regarding the ability of any due date based policies to actually meet due dates, it is clearly necessary to assume that

$$-\gamma_1 \leq \delta(\pi) - \alpha(\pi) \leq \gamma_2 \text{ for all } \pi \quad (16)$$

for some  $\gamma_1, \gamma_2 \geq 0$ , i.e., that parts are released into the system neither too far in advance of the due date, nor too late after it. In fact, under this assumption, the stability of a policy is equivalent to guaranteeing that for any  $\lambda$  satisfying (3),

$$|e(\pi) - \delta(\pi)| \leq \Gamma' \text{ for all } \pi, \quad (17)$$

for some  $\Gamma' \geq 0$ , i.e., that the policy is able to meet the due dates for all parts to within a bounded time  $\Gamma'$ .

Note that the assumption (16) allows the flexibility of specifying the due date of a part as occurring even before the release date, which is a useful artifice to expedite the processing of some parts on occasion.

**Theorem 4: Stability of LS.** *Consider a system where the arrivals satisfy (1,3) while the due dates satisfy (16). Consider the LS Policy with arbitrary estimates  $\{\zeta_i\}$  where  $\zeta_i \geq 0$ . If  $x(t)$  denotes the number of parts in the system at time  $t$ , with  $x(0)$  denoting the initial number, then*

$$\sup_t x(t) < f(x(0)),$$

and

$$\limsup_{t \rightarrow \infty} x(t) \leq \Gamma$$

where  $\Gamma$  does not depend on  $x(0)$ .

**Proof:** There are two differences between LS and LBFS. First, under LS a part  $\bar{\pi}$  arriving after  $\pi$  may overtake  $\pi$ . This happens when both  $\pi$  and  $\bar{\pi}$  are in the same buffer  $b_i$ , and

$\delta(\bar{\pi}) - \zeta_i \leq \delta(\pi) - \zeta_i$ , i.e.,  $\delta(\bar{\pi}) \leq \delta(\pi)$ . However, by (16), it must then be the case that

$$\alpha(\pi) \leq \alpha(\bar{\pi}) \leq \delta(\bar{\pi}) + \gamma_1 \leq \delta(\pi) + \gamma_1 \leq \alpha(\pi) + \gamma_2 + \gamma_1.$$

Hence only a part  $\bar{\pi}$  arriving in  $[\alpha(\pi), \alpha(\pi) + \gamma_2 + \gamma_1]$  can possibly overtake  $\pi$ , and by (1), there can be no more than  $\lambda(\gamma_2 + \gamma_1) + \gamma$  such parts. Thus at any buffer  $b_i$ , the maximum delay  $\xi_1$  that a part can experience due to being overtaken is bounded by  $\xi_1 := [\lambda(\gamma_2 + \gamma_1) + \gamma]\bar{\tau}$ , where  $\bar{\tau} := \max_i \tau_i$ . Also note that  $\bar{\pi}$  can only overtake  $\pi$  once.

Second, under LS a part  $\pi$  at a buffer  $b_i$  may, in contrast to LBFS, have lower priority than a part  $\bar{\pi}$  in a buffer  $b_j$  with  $j < i$ , located at the same service center. This happens if  $\delta(\bar{\pi}) - \zeta_j \leq \delta(\pi) - \zeta_i$ . There are two possibilities. If  $\alpha(\bar{\pi}) \geq \alpha(\pi)$ , then

$$\alpha(\pi) \leq \alpha(\bar{\pi}) \leq \delta(\bar{\pi}) + \gamma_1 \leq \delta(\pi) - \zeta_i + \zeta_j + \gamma_1 \leq \alpha(\pi) + \gamma_2 - \zeta_i + \zeta_j + \gamma_1.$$

On the other hand, if  $\alpha(\bar{\pi}) < \alpha(\pi)$ , then  $\pi$  must have already overtaken  $\bar{\pi}$  at some prior buffer  $b_k$ ,  $k < j$ , and so,

$$\alpha(\pi) > \alpha(\bar{\pi}) \geq \delta(\bar{\pi}) - \gamma_2 \geq \delta(\pi) - \gamma_2 \geq \alpha(\pi) - \gamma_1 - \gamma_2.$$

Hence, either  $\alpha(\bar{\pi}) \in [\alpha(\pi), \alpha(\pi) + \gamma_2 - \zeta_i + \zeta_j + \gamma_1]$ , or  $\alpha(\bar{\pi}) \in [\alpha(\pi) - \gamma_1 - \gamma_2, \alpha(\pi)]$ . Thus the number of such parts  $\bar{\pi}$  is limited to  $\lambda \max\{\gamma_1 + \gamma_2, \gamma_1 + \gamma_2 - \zeta_i + \zeta_j\} + \gamma$  by (1). So the delay thus incurred by a part  $\pi$  due to such parts  $\bar{\pi}$  is limited to

$$\xi_2 := \bar{\tau} \lambda \max\{\gamma_1 + \gamma_2, \max_{i,j}(\gamma_1 + \gamma_2 - \zeta_i + \zeta_j)\} + \bar{\tau} \gamma.$$

Let us define  $\xi := \xi_1 + \xi_2$ , and the delay experienced by a part  $b_i$  at a service center due to having to wait for a part *behind* it at the service center is limited to  $\xi$ .

Except for such a time delay  $\xi$  at each service center, a part  $\pi$  is only delayed by parts *ahead* of it in the system, and thus behaves similar to an LBFS policy.

It is easy to show that the induction hypothesis of Theorem 2 still holds, and as in Theorem 3, we still obtain stability; the details of the minor modifications, which essentially replace the non-preemptive delay  $\bar{\tau}$  in (7) by  $\xi + \bar{\tau}$ , are left to the reader.  $\square$

Finally, since EDD is just a special case of LS where the “estimates” are set to  $\zeta_i \equiv 0$ , we obtain the following stability result for EDD.

**Corollary 1: Stability of EDD.** *Consider a system where the arrivals satisfy (1,3) while the due dates satisfy (16). Then, under the EDD policy, the number  $x(t)$  of the parts in the system at time  $t$  satisfies,*

$$\sup_t x(t) < f(x(0)),$$

and

$$\limsup_{t \rightarrow \infty} x(t) \leq \Gamma$$

where  $\Gamma$  does not depend on  $x(0)$ .

We have not been able to establish the stability of FCFS. It is curious that in the restricted case when all the buffers located at the same service center require the same mean processing time, Kelly [4] has established the stability of FCFS under the “Markovian” assumption of exponential service and inter-arrival times. However, we have been unable to prove stability even under the analogous deterministic restriction that all processing times at a service center are equal. The difficulty stems from the fact that a part can be substantially delayed by both parts arriving before as well as after it. Given the widespread mention of FCFS this is an interesting open question.

## 7 Systems With Multiple Non-Acyclic Flow Lines

Consider now a set  $\{1, 2, \dots, S\}$  of service centers, where service center  $\sigma$  contains  $m_\sigma$  identical machines in parallel. There are  $P$  types of parts labeled  $1, 2, \dots, P$ . Parts of type  $p$  visit the service centers  $\sigma_{p,1}, \sigma_{p,2}, \dots, \sigma_{p,\ell_p}$  in succession, where they are stored in buffers  $b_{p,1}, b_{p,2}, \dots, b_{p,\ell_p}$ . At buffer  $b_{p,i}$  they require a processing time  $\tau_{p,i}$  from one of the machines at the service center.

(We should mention that the usage of the word “part-type” is not strictly correct. In semiconductor manufacturing, even in a single non-acyclic flow line there may be many different “types” of parts which use the same route and have the same processing requirements. However, for convenience of exposition, we regard all parts having the same route and processing requirements as being of the same “type”).

Let  $u_p(t)$  denote the number of arrivals of parts of type  $p$  to the system in the time interval  $[0, t]$ . We assume that

$$u_p(t) - u_p(s) \leq \lambda_p(t - s) + \gamma_p \text{ for } 0 \leq s \leq t \text{ and all } p. \quad (18)$$

The work brought in per machine at service center  $\sigma$  by a part of type  $p$  is

$$w_{p,\sigma} := \sum_{\{i|\sigma_p,i=\sigma\}} \frac{\tau_{p,i}}{m_\sigma}.$$

Clearly in order for the system to be stabilizable, we require that,

$$\sum_p \lambda_p w_{p,\sigma} < 1 \text{ for all } \sigma = 1, \dots, S. \quad (19)$$

Consider now the following distributed scheduling policies based on a buffer priority ordering.

### **Interleaved First Buffer First Serve Policy (IFBFS)**

Consider a total ordering  $\{\beta_1, \beta_2, \dots, \beta_{\sum_p \ell_p}\}$  of all the buffers in the system, which satisfies the following condition. If  $\beta_m = b_{p,i}$  and  $\beta_n = b_{p,j}$  with  $i < j$ , then  $m < n$ . (Note that such an ordering is obtained whenever the several FBFS orderings of the various part-types are “interleaved”).

### **Uniform Last Buffer First Serve Policy (ULBFS)**

Consider a total ordering  $\{\beta_1, \beta_2, \dots, \beta_{\sum_p \ell_p}\}$  of all the buffers in the system, which satisfies the following condition. Buffer  $b_{p,i}$  precedes buffer  $b_{q,j}$  in this ordering if and only if either  $p < q$  or  $(p = q \text{ and } i > j)$ . (It is clear that in place of the lexicographic ordering  $1, 2, \dots, P$ , any other ordering of the part-types also suffices).

### **A Combination of FBFS and LBFS (CFLBFS)**

This is a generalization which includes both IFBFS as well as ULBFS. First we divide the set of all part-types into two sets  $P_F$  and  $P_L$ . Then we consider a total ordering  $\{\beta_1, \beta_2, \dots, \beta_{\sum_p \ell_p}\}$  of all the buffers in the system, which satisfies the following restrictions.

- (i) If  $\beta_n = b_{p,\ell_p}$  for some  $p \in P_L$ , then  $\beta_{n+1} = b_{p,\ell_p-1}, \beta_{n+2} = b_{p,\ell_p-2}, \dots, \beta_{n+\ell_p-1} = b_{p,1}$ .
- (ii) If  $\beta_m = b_{p,i}$  and  $\beta_n = b_{p,j}$  for some  $p \in P_F$  with  $i < j$ , then  $m < n$ .

In each of the above policies, priority is given to buffers according to the total ordering, i.e., buffer  $\beta_m$  gets higher priority over another buffer  $\beta_n$  located at the same service center if  $m < n$ .

The following theorem shows that all the above policies are stable.

**Theorem 5: Stability of IFBFS, ULBFS, and CFLBFS.** *The policies IFBFS, ULBFS and CFLBFS are stable whenever the arrivals satisfy (18, 19).*

**Proof.** Let us first consider IFBFS. By the same argument as in Theorem 1, it follows that the busy periods spent working on  $\beta_1$  are bounded. Define now an *i-busy period* as a time interval during which, at every time instant, at least one of the buffers  $\beta_j$  located at the same service center as  $\beta_i$ , with  $j \leq i$ , has some part waiting for service. By exactly the same induction argument as in Theorem 1, it follows that the *i-busy periods* are all bounded. This also proves that the delays experienced by parts are bounded, concluding the proof of stability of IFBFS.

Now we turn to ULBFS. Consider first part-type  $p = 1$ . Since it gets higher priority over every other part-type, its blockage by parts of type  $q \geq 2$  is limited to the delay caused by non-preemption. Hence the result of Theorem 2 applies to parts of type 1; and its proof by induction is the same. By an application of Theorem 3 to part-type  $p = 1$ , we conclude that the delays experienced by parts of type  $p = 1$  are bounded.

Now the proof is by induction on  $p$ . Suppose that the delays of parts of types  $1, 2, \dots, p-1$  are bounded, and thus also the buffer levels of these part-types. This implies that for each buffer  $b_i$  of a part-type  $q$  with  $q < p$ , in any given time interval  $[s, t]$ , a number of parts  $\leq \lambda_q(t - s) + \xi$  are processed, where  $\xi$  is a constant. Hence the sum, over the machines at the service center  $\sigma$ , of the lengths of the time intervals that they are available to process part-types  $p, p+1, \dots, P$  in any interval  $[s, t]$ , is  $\geq m_\sigma[1 - \sum_{q=1}^{p-1} \lambda_q w_{q,\sigma}](t - s) - \xi'$ , where  $\xi'$  is a constant. Therefore, enough time is available process parts of types  $p, p+1, \dots, P$ , on the average, i.e., the capacity condition is met for part-types  $p, p+1, \dots, P$ . By repeating the argument of Theorem 2, the contractive estimate of the delay for parts of type  $p$  is

established, i.e., if a part of type  $p$  enters the system when there are already  $x$  parts of type  $p$  in the system, then it leaves within  $c_p(\epsilon) + (\max_{\sigma} w_{p,\sigma} + \epsilon)x$  time units. By repeating the argument of Theorem 3, the boundedness of the buffers of part-type  $p$  holds. This completes the induction and the proof of the stability of ULBFS.

Finally, for the policy CFLBFS, the proof is a combination of the above two proofs. It is based on induction, according to the buffer priority ordering, where for a part-type  $p \in P_L$  the stability of all its buffers is proved simultaneously, while for a part-type  $p \in P_F$  only the stability of an individual buffer is proved at each stage of the induction argument. The details, which are a combination of the above two proofs, are left to the reader.  $\square$

We conjecture that if one only orders the buffers separately at each service station, so that, for example, the FBFS ordering is respected *locally* at each service center, but without taking into account the *global* ordering of the buffers, then that alone is not enough for stability.

We have not been able to establish the stability of an “interleaved” version of LBFS; in fact we conjecture that some such interleavings may lead to instability.

We have also been unable to establish whether EDD is stable for systems consisting of multiple non-acyclic flow lines. The difficulty lies in the fact that a part may be interfered with several times at several buffers by a part traversing a different route, which precludes a contractive estimate of its delay. Given the widespread mention of EDD, the resolution of its stability is an interesting open question.

## 8 Generalizations

All the stability results in this paper can be generalized to the following situations.

- i) Parts may differ with respect to their processing times. If  $\tau_i(\pi)$  is the processing time of a particular part  $\pi$  at the buffer  $b_i$ , then we only need assume that  $\tau_i(\pi) \leq \tau_i$  for all  $\pi$ .
- ii) The machines at a service center need not be identical; they may differ in “speed.” Let  $s_{\sigma,k}$  be the speed of the  $k$ -th machine at the service center  $\sigma$ . So a part requiring  $\tau$

time units worth of processing is completed in  $\frac{\tau}{s_{\sigma,k}}$  time units if it is processed at this machine. Then, the “work” brought per part to a “standard” machine with speed 1 is,

$$w_{\sigma} := \sum_{\{i|\sigma_i=\sigma\}} \frac{\tau_i}{\sum_{k=1}^{m_{\sigma}} s_{\sigma,k}},$$

and the required capacity condition is (3).

- iii) In the LS scheduling policy, we can allow for a more dynamic policy where the “estimates” of future delay depend on the current state of the network. All we require is that there be a uniform bound for these dynamic estimates. We note that this flexibility also allows for some heuristics which attempt to “optimize” short-range sequencing decisions.
- iv) There may be a state-dependent transportation delay as a part moves between service centers; all we require is that the transportation times be uniformly bounded.

## 9 Some Simulation Results

To obtain a better idea of the performance of the scheduling policies studied in this paper, we have conducted simulations of a highly re-entrant flow line such as in Figure 1, using the SLAM II simulation software. Specifically, we simulated such a system with 5 service centers, each consisting of one machine, with each machine being visited 5 times.

We conducted 30 simulations, at each of the values of the load factors,  $\rho = 0.8, 0.9$  and  $0.999$ . In each simulation, the arrivals were chosen as a Poisson process of rate 1.

The processing times were chosen randomly as follows. First, a set  $\{x_i \mid 1 \leq i \leq S^2\}$  of  $S^2$  pseudo-random numbers, uniformly distributed in  $[0, 1]$ , were generated. From these, we derived

$$\tau_i^{(\rho)} := \frac{\rho x_i}{\sum_{\{j|j \bmod s=i \bmod s\}} x_j} \text{ for } 1 \leq i \leq S^2, \text{ and } \rho = 0.8, 0.9, \text{ and } 0.999.$$

This procedure yields three sets of processing times  $\{\tau_i^{(\rho)} \mid 1 \leq i \leq S^2\}$  for the three values of  $\rho$ , where  $\lambda w_{\sigma} = \rho$  for all  $\sigma$  in each set, i.e., all service centers have identical loads.

These processing times were then used to conduct one simulation for each of the three load factors. By using the same  $\{x_i\}$ , we were also able to study the performance of the policies as processing times were scaled. Finally this entire procedure was repeated 30 times.

Each simulation was started with an empty system and was terminated when the 400-th part left the system. To eliminate transient effects, statistics were collected beginning with the 201-th part.

One may wonder how the “estimates”  $\{\zeta_i\}$  of the “remaining delays” from the buffers  $\{b_i\}$ , necessary for the implementation of the LS policy, were obtained, given that the delays depend on the scheduling policy, while the policy itself depends on estimates of the delays. We have found a quite effective procedure in our simulations. Essentially, at iteration 0, we set all  $\zeta_i^{(0)}$  identically to zero, and then run a simulation of the LS policy, which gives us empirical “mean remaining delays,” which we call  $\{\zeta_i^{(1)}\}$ . Then we use the LS scheduling policy defined by  $\{\zeta_i^{(1)}\}$  to run a simulation, which in turn gives estimates  $\{\zeta_i^{(2)}\}$ , and so on. We have observed that these estimates are monotone increasing with a tendency towards convergence, and we used  $\{\zeta_i^{(10)}\}$  as our estimates of slacks.

We measured the empirical mean as well as the empirical standard deviation of delay in the system for each policy, in each simulation run. We believe that the best measure of “performance” may well be captured by a combination of mean and standard deviation, such as (mean + 3 standard deviations), since we are guaranteed (by Chebyshev’s inequality) that at least 8/9 (= 88.8%) of the parts will incur lesser delay, providing some reliability in meeting due dates.

The results of the simulations are summarized in Table 1. To avoid confusion, the word “Mean” in the table refers to the average taken over the individual parts in a particular simulation run, while “Average” refers to the average taken over the 30 statistics gathered from the 30 simulation runs.

In our simulation results, LS had the least variance for the delay in 81 out of 90 runs. On the other hand, LBFS had the least mean delay in 80 runs, and the least (mean + 3 standard deviations) in 65 runs (specifically in all 30 runs at  $\rho = 0.999$ , 23 out of 30 runs

$\rho$		<i>FBFS*</i>	<i>FCFS</i>	<i>LBFS</i>	<i>LS</i>
0.8	Average of 30 Means	33.63	11.32	10.06	10.52
	Average of 30 Std. Deviations	14.16	4.19	3.44	3.01
	Average of 30 Mean+3 Std. Deviation	76.11	23.88	20.40	19.56
	Average Rank of Mean	4.00	2.80	1.40	1.80
	Average Rank of Std. Deviation	4.00	2.87	2.00	1.13
	Average Rank of Mean+3 Std. Deviation	4.00	2.87	1.70	1.43
0.9	Average of 30 Means	164.04	31.54	22.42	26.56
	Average of 30 Std. Deviations	18.29	7.64	5.52	4.76
	Average of 30 Mean+3 Std. Deviation	218.92	54.46	39.00	40.86
	Average Rank of Mean	4.00	2.90	1.20	1.90
	Average Rank of Std. Deviation	4.00	3.00	1.97	1.03
	Average Rank of Mean+3 Std. Deviation	4.00	3.00	1.23	1.77
0.999	Average of 30 Means	491.65	83.62	49.32	65.43
	Average of 30 Std. Deviations	15.22	4.04	4.41	3.35
	Average of 30 Mean+3 Std. Deviation	537.30	95.74	62.54	75.48
	Average Rank of Mean	4.00	3.00	1.00	2.00
	Average Rank of Std. Deviation	4.00	2.17	2.70	1.13
	Average Rank of Mean+3 Std. Deviation	4.00	3.00	1.00	2.00

\* Due to an extremely large build up of queue size, only 14 sets of values were recorded at the load factor 0.999.

Table 1: Summary of Simulation Results

at  $\rho = 0.9$ , and 12 out of 30 runs at  $\rho = 0.8$ ). These observations confirm our intuition (see Section 2) regarding the suitability of LBFS for minimizing mean delay, and that of LS for minimizing variance of the delay.

The ability of the LS policy to reduce mean delay is especially noticeable by a significant margin at high load factors (0.9 and 0.999). At the lighter load factor of  $\rho = 0.8$ , however, LS tended to outperform LBFS with respect to (mean + 3 standard deviations), due to its lower standard deviations.

The FCFS policy was generally worse than LBFS and LS with respect to both mean and standard deviation. Finally, the FBFS policy was the worst among all four policies in all 90 runs. Its exceedingly large delay is apparently due to its “push” nature, which is in sharp contrast with the “pull” nature of LBFS.

Of course, none of these “results” is conclusive in any sense of the word, but merely indicative of our experience for a highly specific situation.

In general, due to the high degree of cross-consistency across the simulation runs, it appears that such simulation studies can indeed help us in understanding the performance of the various policies. Hence, a much more thorough simulation study appears to be warranted, to see whether similar results also hold for larger systems, when service centers contain several machines, when random yields cause rework and scrapping, when parts arrive possibly out of order, etc.

## 10 Concluding Remarks

This paper is a start towards determining the behavior of several scheduling policies, all of which, while commonly mentioned in the literature, appear not to be well understood. The simulation results confirm our intuition that LBFS may well be the best policy for minimizing the *mean* delay, especially at high load factors, and that LS may well be the best policy for minimizing the *variance* of the delay. We therefore recommend a “convex” combination of the two, which is itself an LS policy employing small values of “slacks” (say small proportions of the remaining delays) for minimizing (mean + 3 standard deviations).

Several interesting open problems remain. For example, the stability of the First Come First Serve Policy is an open question even for systems with even just one non-acyclic flow line. For systems with multiple non-acyclic flow lines, the stability of even the Earliest Due Date Policy is unresolved.

In application areas such as semiconductor manufacturing, the problem of “yield” management gives rise to several additional issues involving rework and scrapping, for which a stochastic extension of our results would be useful. What we have provided could be the deterministic backbone of a sample-path based stability argument for such stochastic extensions, which may also incorporate other sources of randomness such as processing times, machine failures, etc.

**Acknowledgements** The authors are grateful to C. Abraham, D. Connors, S. Hood, R. Jayaraman, P. Kamesam and R. Wittrock, all of the IBM Yorktown Research Center, for informative discussions regarding the problems of semiconductor manufacturing. We are also grateful to T. I. Seidman, University of Maryland Baltimore County, for suggesting the counter-example of Section 3.

## References

- [1] T. E. Dillinger, *VLSI Engineering*. Englewood Cliffs, NJ: Prentice-Hall, 1988.
- [2] S. S. Panwalker and W. Iskander, “A survey of scheduling rules,” *Operations Research*, vol. 25, pp. 45–61, January-February 1977.
- [3] S. C. Graves, “A review of production scheduling,” *Mathematics of Operations Research*, vol. 29, pp. 646–675, July-August 1981.
- [4] F. P. Kelly, *Reversibility and Stochastic Networks*. New York, NY: John Wiley and Sons, 1987.
- [5] J. R. Perkins and P. R. Kumar, “Stable distributed real-time scheduling of flexible manufacturing/assembly/disassembly systems,” *IEEE Transactions on Automatic Control*, vol. AC-34, pp. 139–148, February 1989.
- [6] 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*, vol. AC-35, pp. 289–298, March 1990.

- [7] R. L. Cruz, “A calculus for network delay, part I: Network elements in isolation,” *IEEE Transactions on Information Theory*, vol. 37, pp. 114–131, January 1991.