revision2: 9/4/'93

# 'First Come, First Served' can be unstable!

**Thomas I. Seidman**
Department of Mathematics and Statistics
University of Maryland Baltimore County
Baltimore, MD 21228, USA
e-mail: ⟨seidman@math.umbc.edu⟩

ABSTRACT: We consider flexible manufacturing systems using the 'First Come, First Served' (FCFS or FIFO) scheduling policy at each machine. We describe and discuss in some detail simple deterministic examples which have adequate capacity but which, under FCFS, can exhibit instability: unboundedly growing WIP taking the form of a repeated pattern of behavior with the repetitions on an increasing scale.

KEY WORDS: *scheduling policy, flexible manufacturing system, queueing network, stability, FIFO, 'First Come First Served'.*

## 1. Introduction

We consider network models involving multiple flows with buffering/queuing at each node (processor). Specifying a *queue discipline* (i.e., a scheduling policy for the processing at nodes) then defines the dynamics for the system. A queue discipline is called *stable* if the queue lengths (WIP) remain uniformly bounded in time for any realization — configuration, initial state — with input rates subject to the obvious capacity limitations. We quote from [3] the observation that, "We have been unable to resolve whether FCFS is stable — a significant open question." It is the point of this note to resolve that question[1] — to show by examples that the popular 'First Come, First Served' policy (FCFS — also known as FIFO = 'First In, First Out') is not a stable queue discipline.

We will use the terminology of *manufacturing systems*: we refer to the nodes as *machines* and to the flows as *product streams*, although it is clear that models of this sort arise also in other contexts. Thus, for each stream (product type) $\mathcal{P}_j$ one has a sequence of *tasks* $\{i\}$ to be done at machines $\mathcal{M}_{k(i)}$. Associated with each task is a *processing time* $\tau_i$, time units taken to process a unit of product; we do not impose any time penalty for switching between tasks at a machine. The 'obvious capacity limitations' mentioned above now take the form:

$$\sum_{i\in\mathcal{M}_k} \tau_i C_{j(i)} < 1 \qquad \text{for each machine } \mathcal{M}_k \tag{1.1}$$

where $C_j$ is the (constant) input rate for each $\mathcal{P}_j$, since this sum just gives the *utilization factor* for $\mathcal{M}_k$, i.e., the proportional time required for the processing to handle its share of the load.

This is a deterministic continuum model, but we observe that for an analysis of (potential) instability one is necessarily interested in long time scales and expects to treat amounts of product large compared to a discrete unit. Thus, even if the underlying model were discrete and stochastic, any fluctuations can be expected to be negligible in comparison with the quantities involved so a deterministic treatment, using the mean values, gives a reliable description of the behavior. We do note that this negligibility of 'small' fluctuations only applies away from the 'trivial' *ground state* — all buffers remaining empty with processing exactly matching input — so our deterministic analysis is necessarily inadequate to consider any transitions from that state to the larger scale scenarios we describe which might be induced precisely by those fluctuations.

It is easily verified — much as for the earlier analysis of clearing policies [5], [4] that — FCFS is stable within the restricted class of *acyclic* configurations, in which one can argue inductively, so our examples are necessarily nonacyclic. There is also good reason to feel that instability cannot occur without substantial discrepancies in the processing

[1] Subsequent to the original submission of this paper, we learned of related work by Bramson [1], considering a rather different configuration and demonstrating there almost sure instability in a stochastic context, i.e., that with probability 1 the total WIP has infinite liminf. A subsequent paper [2] further shows that, even subject to a stronger capacity condition (replacing (1.1 by: $\sum_{i\in\mathcal{M}_k} \tau_i C_{j(i)} < \alpha < 1$), there are such stochastically unstable configurations for arbitrarily small $\alpha$.

times. We are able to obtain a considerable simplification in our descriptions of the scenarios by considering examples in which this is carried to an extreme, assuming some of the processing is fast enough for those processing times to be neglected. We invite the reader to consider the legitimacy of this simplification while tracking the specific behavior patterns presented.

## 2. The configurations

Much as in [4], we provide two somewhat similar examples of instability mechanisms systems operating under FCFS. Each involves 12 tasks to be done at 4 machines. The first example, with 4 product streams, will be somewhat easier to analyze in detail. The second shows that instability can occur also in the case of a system with a single product.

Figures 1,2, below, show the two configurations; the numbering of tasks is less natural for Figure 2 but has been retained for comparison. The heavier dots in each Figure indicate the comparatively slow tasks; throughout, we will denote their processing times $\tau_6$, $\tau_3$, $\tau_9$, $\tau_{12}$ by $\sigma$, $\tilde{\sigma}$, $\tau$, $\tilde{\tau}$, respectively, with the requirements that

$$0 < \sigma < \tilde{\sigma} < 1, \qquad 0 < \tau < \tilde{\tau} < 1. \tag{2.1}$$

Taking the other processing times fast enough then ensures that (1.1) will be satisfied under the normalization (choice of units for product) that input rates are $\equiv 1$.

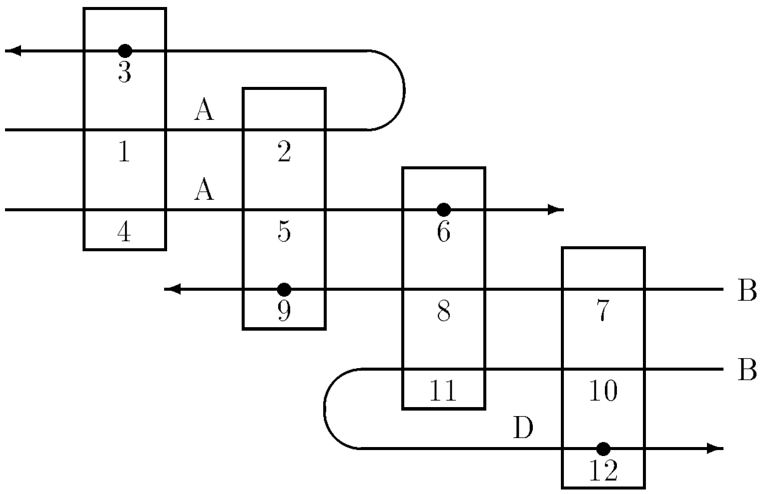


**Figure 1**

For the moment the labels $A, B, D$, above, are irrelevant. For this example we require, in addition to (2.1), that

$$(\tilde{\sigma} - \sigma),\ (\tilde{\tau} - \tau) < \sigma\tau; \qquad \tau \leq \sigma. \tag{2.2}$$

The final condition involves no loss of generality; it will become clear later how the other parameter restrictions will lead to the behavior pattern we will describe.

It may be noted already at this point that the situation would be exactly symmetric if we were to have $\sigma = \tau$ and $\tilde{\sigma} = \tilde{\tau}$ but that in any case there is an essential symmetry between the product streams $\mathcal{P}_1$, $\mathcal{P}_2$ and machines $\mathcal{M}_1$, $\mathcal{M}_2$ as compared, respectively, with the product streams $\mathcal{P}_4$, $\mathcal{P}_3$ and machines $\mathcal{M}_4$, $\mathcal{M}_3$ — i.e., pairing the tasks

$$1 \leftrightarrow 10 \qquad 2 \leftrightarrow 11 \qquad 3 \leftrightarrow 12 \qquad 4 \leftrightarrow 7 \qquad 8 \leftrightarrow 5 \qquad 6 \leftrightarrow 9. \tag{2.3}$$

The other example may be viewed as linking the product streams by feeding the output of $\mathcal{P}_2$ as input to $\mathcal{P}_3$, feeding the output of that into $\mathcal{P}_1$, and finally the output of that into $\mathcal{P}_4$.

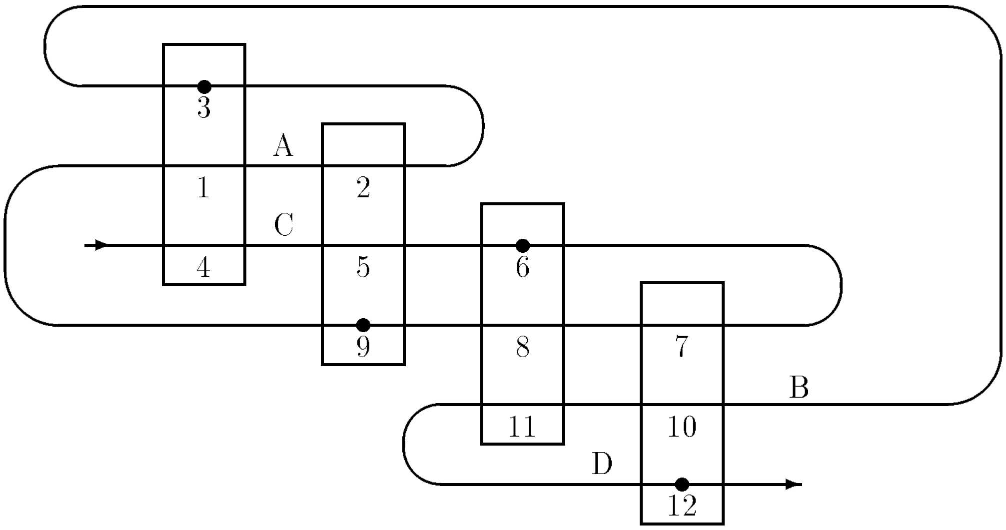


**Figure 2**

Again, for the moment the labels $A, B, C, D$, above, are irrelevant. For simplicity, we only consider here the case $\sigma = \tau$, $\tilde{\sigma} = \tilde{\tau}$; we require, subsuming (2.1), that

$$2\tau^2 < \tilde{\tau} < 1 \qquad \text{with } 1/2 < \tau \tag{2.4}$$

and so, necessarily, with $\tau < 1/\sqrt{2}$. Note that FCFS corresponds to the use of a single queue $\mathcal{Q}_k$ for each machine $\mathcal{M}_k$ — although it will be convenient for us to speak of 'buffer $i$' in referring to the inventory of product in the queue $\mathcal{Q}_{k(i)}$ awaiting processing for the task $i$. Thus,the *state* at any time (for any machine) really requires not only knowledge of the amount of product in each of the buffers $i \in \mathcal{M}_k$, but also the appropriate sequencing information. The benefit we obtain from our consideration of 'extremely fast' processing for tasks $1, 2, 4, 5, 7, 8, 10, 11$ will be precisely the simplification, *for the particular scenarios we consider*, that there will be negligible intermixing at any machine of the sequencing for slow and for fast tasks and that any intermixing with each other of sequencing for the two fast tasks at a machine will not affect the relevant features of our descriptions. Again, we invite the reader to attend to this consideration while tracking the specific behavior patterns presented.

## 3. The first example

Since the capacity condition (1.1) is satisfied, the 'ground state', in which all queues are empty, will persist if the situation is not perturbed. Let us consider, as one possible scenario, what would happen if, e.g., one would have a brief 'breakdown' at $\mathcal{M}_4$: downtime$= T_*$. When $\mathcal{M}_4$ is again working, $\mathcal{Q}_1$, $\mathcal{Q}_2$, $\mathcal{Q}_3$ are (still) empty but amounts $B_* = T_*$ of each of $\mathcal{P}_3$ and $\mathcal{P}_4$ have accumulated in $\mathcal{Q}_4$, waiting for the tasks $7, 10$, respectively; it will turn out that we have no need to know the ordering of these products in the queue, although they are presumably evenly intermixed.

*What happens then?* At $\mathcal{M}_4$, all that is now in $\mathcal{Q}_4$ will be processed during a very short time interval since we have assumed $\tau_7$, $\tau_{10}$ very small. The output from these tasks goes onto $\mathcal{Q}_3$; by assumption this was initially empty and, in such a short time, it is impossible for more than a negligible amount of $\mathcal{P}_2$ to arrive. Since 8, 11 are also 'extremely fast', this product is also processed there during what is, altogether, still only a very short time interval. Note that the output from 11 goes to the end of $\mathcal{Q}_4$ to wait for 12; even if that processing at 7, 10 will have 'finished' before all this has arrived from 11 at $\mathcal{M}_3$, we note that, since the time is very short, (a) only a negligible amount of new product ($\mathcal{P}_3$, $\mathcal{P}_4$) can have arrived meanwhile and (b) only a negligible amount of product can have gotten processed at 12, since that task is comparatively slow. Thus, at the end of this short period, $\mathcal{Q}_4$ consists (approximately) of the amount $C_* = B_*$ of $\mathcal{P}_4$ waiting for 12. Similarly, the output of 8 at $\mathcal{M}_3$ goes to $\mathcal{Q}_2$ where, since 9 is also comparatively slow, only a negligible amount could have been processed there and, at the end of our short period when $\mathcal{Q}_3$ has (approximately) emptied, $\mathcal{Q}_2$ consists (approximately) of the amount $C_* = B_*$ of $\mathcal{P}_3$ waiting for 9. For convenience of description we use these approximations as if they were exact.

The next 'significant event' is completion of the processing at $\mathcal{M}_2$ of the amount just noted as in that queue, waiting for 9. (Note that this occurs earlier than completion at $\mathcal{M}_4$ of the corresponding block of product waiting there for 12, since we have assumed that 12 is slower than 9, i.e., that $\tau < \tilde{\tau}$.) This takes place during an interval of length $T'_* = C_*/\tau$. We next ask: *What happens during this period? What is the situation at its end?*

At $\mathcal{M}_1$, there was adequate capacity to process all the arriving product so $\mathcal{Q}_1$ remains empty; the amounts $A_0 = T'_*$ of each of $\mathcal{P}_1$, $\mathcal{P}_2$ arrived, were processed at 1, 2, respectively, (i.e., tasks 1, 2 at $\mathcal{M}_1$) without delay, and so went as output to the end of the queue $\mathcal{Q}_2$ (in a mixed order, which we have no need to know) after the $\mathcal{P}_3$ we already saw queued there for 9. At $\mathcal{M}_4$, effort was devoted to task 12, since the head of $\mathcal{Q}_4$ was just this. During this time, the amount $B_0 = T'_*$ of each of $\mathcal{P}_3$, $\mathcal{P}_4$ arrived to wait (in some mixed order) at the end of $\mathcal{Q}_4$ for processing at 7, 10, respectively. From our descriptions so far of the activity at $\mathcal{M}_2$, $\mathcal{M}_4$, we see that $\mathcal{M}_3$ must idle during this period — since $\mathcal{Q}_3$ was empty at its beginning and there are no outputs from 5, 7, 10 to provide input into $\mathcal{Q}_3$. Thus, $\mathcal{Q}_3$ (like $\mathcal{Q}_1$) is still empty at the end of the period and our descriptions of the activity at $\mathcal{M}_2$, $\mathcal{M}_4$ were complete: there could be negligible further input for 9, 12 during this time. Hence, at the end of the period, $\mathcal{Q}_2 \approx [A_0$ *each of* $\mathcal{P}_1$, $\mathcal{P}_2$ *waiting for* 2, 5] and $\mathcal{Q}_4 \approx [D_0$ *of* $\mathcal{P}_4$ *waiting for* 12; *followed by* $B_0$ *each of* $\mathcal{P}_3$, $\mathcal{P}_4$ *waiting for* 7, 10]. Here $D_0$ is the portion of the earlier amount $C_*$ at the head of this queue which has not yet been processed: $D_0 = C_* - T'_*/\tilde{\tau} = C_*[1 - \tau/\tilde{\tau}] > 0$.

This state has the form of the state indicated in Figure 1 and we remark that various other scenarios also lead to states of this same form, although we do not take the time to describe any of the others. Note that, for a state of this format, we may take $A$ as determining an 'amplitude' and then define the state by specifying $\gamma = B/A$ (here, $\gamma = 1$) and $D/A$.

We now describe the behavior pattern occurring when one starts with the initial state indicated in Figure 1:

$$
\begin{array}{lrl}
 & \boxed{\text{at } t = t_0} & \\
 & A & \text{at each of } 2,5; \\
(3.1) & B = \gamma A & \text{at each of } 7,10; \\
 & D & \text{at } 12; \\
 & 0 & \text{at all others: } 1,3,4,6,8,9,11 \\
[*] & & \text{all at 12 arrived before any at } 7,10
\end{array}
$$

with $A, B, D > 0$. What this means, in terms of 'machine queue' state specifications, is that $\mathcal{Q}_1$ and $\mathcal{Q}_3$ are empty, $\mathcal{Q}_2$ consists of the amounts $A$ corresponding to each of the buffers (tasks) $2,5$ in some irrelevant sequence within the queue, and in $\mathcal{Q}_4$ one has the amount $D$ of $\mathcal{P}_4$ at the 'front' of the queue, noting (3.1[$*$]), waiting for 12 followed by an intermixture (in some sequence) of amounts $B$ for each of $7,10$. We do not yet specify $\gamma := B/A > 0$, but will leave this as a free parameter for the moment. We also do not specify $D$ (e.g., in relation to $A$) except to require that

$$0 < \tilde{\tau} D < \sigma A \tag{3.2}$$

in this initial state; later, this will ensure that we do, indeed, have $t_2$ before $t_4$.

We now describe the behavior pattern which we expect — and which will indeed occur, subject to the restrictions we will find necessary for that. We will only describe the first four steps out of eight in detail, relying on symmetry for the remainder. For our present description we formally take $\tau_1 = \tau_2 = \tau_4 = \tau_5 = \tau_7 = \tau_8 = \tau_{10} = \tau_{11} = 0$ so what was just described as a 'very short interval' will now be described as 'instantaneous', etc.

**Step 1:** $\boxed{t_0 < t < t_1 = t_0+}$

This step is defined by the 'instantaneous' processing of the initial inventories $A$ at $2,5$ (i.e., queued in $\mathcal{Q}_2$ to await processing for the tasks $2,5$), then arriving at buffers $3,6$. By (3.1[$*$]), under FCFS we have $\mathcal{M}_4$ processing the inventory $D$ at 12, which is at the front of the queue, so buffer 8 (hence, also, buffer 9) and 11 remain empty; we may view this, alternatively, as meaning that tasks 7 and 10 are 'blocked' by the priority of 12. Since the step is 'instantaneous', nothing happens elsewhere.

**Step 2:** $\boxed{t_1 < t < t_2 \text{ with } t_2 = t_0 + \tilde{\tau} D}$

This step is defined by completion at $\mathcal{M}_4$ (after time $\tilde{\tau} D$) of the processing of the (residual) inventory $D$ waiting for 12 in $\mathcal{Q}_4$. As noted above, there is not yet any processing of inventory at 7 and 10 so the buffers for 8, 9, and 11 still remain empty. Similarly, at $\mathcal{M}_1$ any new inputs to buffers 1,4 arriving after $t_1$ will necessarily follow in $\mathcal{Q}_1$ behind the inventory which arrived for buffer 3 during the first step just as the 'contents of buffer 6' are already at the front of $\mathcal{Q}_3$ with established priority. Thus, inputs accumulate at the ends of $\mathcal{Q}_1$ and $\mathcal{Q}_4$ for tasks 1,4,7,10.

**Step 3:** $\boxed{t_2 < t < t_3 = t_2+}$

This is another 'instantaneous' step, now defined by the processing of the accumulated inventory at buffers 7,10, once $\mathcal{Q}_4$ is emptied of the 'older' product initially awaiting 12, ending Step 2. Note that when the inventories from 7,10 now arrive for 8,11, they necessarily follow in $\mathcal{Q}_3$ *after* the inventory which arrived for buffer 3 during Step 2; this prioritization is the essential significance of Step 2 for our behavior pattern. Again, since this step is 'instantaneous', nothing happens elsewhere.

**Step 4:** $\boxed{t_3 < t < t_4 \text{ with } t_4 = t_0 + \sigma A}$

This step is defined by completion of the processing of the (residual) inventory at buffer 6; this takes time $\sigma A$ altogether from when that processing started at $t_0+$. All the original buffer contents at 7,10 together with the inputs which had accumulated during Step 2 went to 8,11 during Step 3 and this, together with what enters 7,10 during the current step and is 'passed through', will now still be in $\mathcal{Q}_3$ (corresponding to buffers 8,11) since, as noted, the WIP corresponding to buffer 6 was at the front of the queue there. Since the total time of accumulation of inputs was $\sigma A$, the amount which will accumulate in $\mathcal{Q}_3$ for 8 and 11 will be $A' := \sigma A + B$; buffers 9,12 remain empty. Of the amount $A$ which went to buffer 3 in Step 1, the amount $(\sigma A)/\tilde{\sigma}$ has been processed (with processing time $\tilde{\tau}$) in the available time $\sigma A$, so the amount remaining there by the end of this step will be $D' := (1 - \sigma/\tilde{\sigma})A$. During the entire time $\sigma A$, the machine $\mathcal{M}_1$ has been processing work (at the front of the queue) corresponding to buffer 3, so buffers 1,4 accumulate their inputs at the end of $\mathcal{Q}_1$, thus ending the step with the same amount $B' := \sigma A$ for each. The ensuing state is:

$$\boxed{\text{at } t = t_4 := t_0 + \sigma A}$$

$$(3.3)\qquad \begin{array}{rl} A' & \text{at each of } 8, 11; \\ B' & \text{at each of } 1, 4; \\ D' & \text{at } 3; \\ 0 & \text{at all others: } 2, 5, 6, 7, 9, 10, 12 \end{array}$$

with

$$(3.4)\qquad A' = B + \sigma A \qquad B' = \sigma A \qquad D' = (1 - \sigma/\tilde{\sigma})A.$$

Note that at $t_4$ the front of $\mathcal{Q}_1$ is all product awaiting task 3, since that arrived during Step 3 before any of the input for tasks 1,4 which all arrived during Step 4. Corresponding to (3.2), we verify that

$$(3.5)\qquad 0 < \tilde{\sigma} D' < \tau A' \qquad \text{i.e., } \tilde{\sigma} - \sigma < \tau(\gamma + \sigma)$$

for any $\gamma \geq 0$ by (2.1), (2.2). Thus, the state in (3.3) is the symmetric image of that given in (3.1) in terms of (2.3).

One will then have a symmetrically corresponding description of Steps 5,6,7,8, noting that it is (3.5) which will ensure $t_6 < t_8$. This leads to the 'final' state:

$$
(3.6)\qquad
\begin{array}{rrl}
 & \boxed{\text{at } t = t_8 := t_4 + \tau A'} & \\
 & A'' & \text{at each of } 8, 11; \\
 & B'' & \text{at each of } 1, 4; \\
 & D'' & \text{at } 3; \\
 & 0 & \text{at all others: } 1, 3, 4, 6, 8, 9, 11 \\
[*] & & \text{all at 12 arrived before any at } 7, 10
\end{array}
$$

with

$$
(3.7)\qquad A'' = B' + \tau A' \qquad B'' = \tau A' \qquad D'' = (1 - \tau/\tilde{\tau})A'.
$$

Now replacing $A, D$ by $A'', D''$ in (3.2), we see that this is equivalent to

$$
(3.8)\qquad \tilde{\tau} - \tau < \sigma(\gamma' + \tau) = \sigma\tau + \frac{\sigma^2}{\gamma + \sigma}
$$

which holds for any $\gamma \geq 0$ by (2.2).

## 4. The stability analysis

Clearly, (3.6) has the same form as (3.1). Thus, the behavior pattern just described will repeat ad infinitem. To understand the asymptotics of this repetition, we first note that, using (3.4), (3.7), one has

$$
(4.1)\qquad \gamma'' := \frac{B''}{A''} = \frac{\tau A'}{B' + \tau A'} = \frac{\tau[B + \sigma A]}{\sigma A + \tau[B + \sigma A]} = \frac{\gamma + \sigma}{(\sigma/\tau) + \gamma + \sigma}
$$

so, recursively,

$$
(4.2)\qquad \gamma_n = \Gamma(\gamma_{n-1}) \quad \text{with} \quad \Gamma(\gamma) := \frac{\gamma + \sigma}{(\sigma/\tau) + \gamma + \sigma} = 1 - \frac{\sigma/\tau}{(\sigma/\tau) + \gamma + \sigma}.
$$

We compute

$$
(4.3)\qquad \frac{d\Gamma}{d\gamma} = \frac{\sigma/\tau}{[(\sigma/\tau) + \gamma + \sigma]^2}
$$

and, using the assumption that $\sigma \geq \tau$, we see that $0 < \Gamma' < 1/(\sigma + \sigma/\tau) < 1$ so the function $\Gamma(\cdot)$ is uniformly contractive[2] on $\mathbb{R}_+$. It follows that the iteration sequence $\{\gamma_n\}$ always converges to a unique fixpoint $\bar{\gamma} = \Gamma(\bar{\gamma})$ which is the positive root of the quadratic equation

$$
(4.4)\qquad \bar{\gamma}^2 + [\sigma + (\sigma/\tau) - 1]\bar{\gamma} - \sigma = 0.
$$

Again from (3.4), (3.7), we obtain

$$
(4.5)\qquad \lambda := A''/A = \sigma + \tau[\gamma + \sigma]
$$

[2] REMARK: It is interesting to note that we may define $\tilde{\Gamma}(\gamma) := 1/(1 + \gamma/\sigma) = \sigma/(\gamma + \sigma)$ and will get $\gamma' := B'/A' = \tilde{\Gamma}(\gamma)$ and then $\gamma'' := B''/A'' = \tilde{\Gamma}([\sigma/\tau]\gamma')$ so $\Gamma(\gamma) = \tilde{\Gamma}([\sigma/\tau]\tilde{\Gamma}(\gamma))$. In the symmetric case, when $\sigma = \tau$, one has $\Gamma = \tilde{\Gamma} \circ \tilde{\Gamma}$ and it would seem simpler to work with $\tilde{\Gamma}$ — but even in this case the contractivity is available only when one combines both 'halves' of the full pattern.

for arbitrary $\gamma$. Recursively, of course, since $\gamma_n \to \bar\gamma$ we have

$$\lambda_n = \sigma + \tau[\gamma_n + \sigma] \to \bar\lambda := \sigma + \tau[\bar\gamma + \sigma]. \tag{4.6}$$

Taking $\gamma := B/A = \bar\gamma$, we also have $B''/A'' = \bar\gamma$ giving $\bar\lambda = A''/A = B''/B$. Then (3.4), (3.7) give $(\bar\lambda - \tau)B = \sigma\tau A$ and $(\bar\lambda - \sigma - \sigma\tau)A = \tau B$ whence

$$\bar\lambda^2 - [\sigma + \tau + \sigma\tau]\bar\lambda + \sigma\tau = 0. \tag{4.7}$$

It is not difficult to verify that with $\sigma, \tau > 0$ one will always have two distinct positive real roots for (4.7) and we note from (4.5) that it is the larger of these which corresponds to the larger root of (4.4) — which is the meaningful positive one. Clearly, $\bar\lambda = 1$ satisfies (4.7) if and only if $\sigma + \tau = 1$ and it easily follows that

$$\text{(4.7) has a root } \bar\lambda > 1 \qquad \Longleftrightarrow \qquad \sigma + \tau > 1. \tag{4.8}$$

We assume, henceforth,[3] that $\sigma + \tau > 1$ so $\bar\lambda > 1$ — indeed, one sees that then $\bar\lambda > \sigma + \tau$. Consider the special 'limit pattern' in which one has

$$B := \frac{\sigma\tau}{\bar\lambda - \tau} A \qquad D := \frac{\tilde\tau - \tau\,\sigma}{\bar\lambda - \tau\,\tau} A \tag{4.9}$$

with $\bar\lambda > 1$ given by (4.7); here $\gamma = \bar\gamma$ so (4.9) is invariant under the repetitions. Thus, we will have the amount $A_n = \bar\lambda^n A$ in buffers 2,5 after the *n-th* repetition of the behavior pattern, i.e., at time $t_{8n} =: T_n$. From the description in the previous section one easily sees that $T_1 - T_0 = t_8 - t_0 = A'' = A_1$ and that in general one has $T_n - T_{n-1} = A_n$. An easy calculation then gives

$$A_n = \frac{\bar\lambda^n}{\bar\lambda^n - 1} \cdot \frac{\bar\lambda - 1}{\bar\lambda} [T_n - T_0] \tag{4.10}$$

so, asymptotically, one has linear growth with time of the envelope of the queue lengths:

$$A(T) \sim (1 - 1/\bar\lambda)T \tag{4.11}$$

which, of course, estimates the WIP retained in the system. Since the accumulated input is just $T$, this means that the total throughput must be $T/\bar\lambda$, i.e., the effective throughput *rate* is $1/\bar\lambda$. Since the throughput lag is just the time needed to process the WIP, this will be $A(T) \cdot 1/\bar\lambda \sim (\bar\lambda - 1)T$, also linear but growing unboundedly for $\bar\lambda > 1$.

For the general situation of (3.1) one would certainly, from (4.6), have $\lambda_n > 1$ for large $n$ so the behavior pattern then 'repeats with amplification' and one has instability. More precisely, in view of (4.6) the general asymptotic behavior will be the same as that of the limit pattern, satisfying (4.11), etc.

We conclude by noting that we have shown that the behavior pattern described is robust and the limit pattern an attractor, reachable from a variety of initial states;

[3] At this point we note that if one would only consider initial data with $\gamma$ not too far from $\bar\gamma$ — sufficient for the establishment of 'instability' — then the parameter restriction $\tilde\sigma - \sigma < \sigma\tau$ in (2.2) can be weakened to (3.8) with $\bar\gamma$ replacing $\gamma$ and the restriction $\tilde\sigma - \sigma < \sigma\tau$ can be omitted entirely since (3.5) is equivalent to $\tilde\sigma < \bar\lambda$ at $\gamma = \bar\gamma$ — which is, of course, always true for $\bar\lambda > 1$.

this does not mean that it is a *global* attractor: the ground state persists. We have constructed several scenarios leading into this behavior pattern — and presented one such above — but have not exhaustively determined the dynamics arising from all possible initial states.

## 5. A single-product example

For comparison with [3], we next provide an example involving only a single product. The configuration is, as noted earlier, a 'simple' adaptation of that of the previous example, but the dynamics are rather different. We acknowledge with gratitude the assistance of A. Yershov, who tracked the behavior pattern occurring here and determined the appropriate condition (2.4).

As indicated in Figure 2, we begin tracking (e.g., at a moment marked by the completion of processing at 9, emptying $\mathcal{Q}_2$) with the following state:

$\mathcal{Q}_1 \approx$ [ *amounts* $A$, $C$ *waiting for* $2, 5$, *respectively* ]

$\mathcal{Q}_4 \approx$ [$D$ *waiting for* 12*; followed by* $B$ *for* 10]

$\mathcal{Q}_2$, $\mathcal{Q}_3 \approx$ [ *empty* ]; the sequencing in $\mathcal{Q}_1$ is irrelevant for our purposes.

We will set $t_1 = t_0+$, $t_3 = t_2+$, $t_5 = t_4+$, $t_7 = t_6+$ with

$$(5.1) \qquad t_0 := 0, \qquad t_2 := \tilde{\tau} D, \qquad t_4 := \tau C, \qquad t_6 := \tilde{\tau} A, \qquad t_8 := 2\tau C$$

and will later verify that

$$(5.2) \qquad 0 < \tilde{\tau} D < \tau C < \tilde{\tau} A < 2\tau C$$

so these do have the indicated order. Note also the later imposition of (5.4).

The nature of the 'behavior tracking' will be quite similar to that of Section 3, so we present it somewhat cursorily here.

**Step 1:** Amount $A$ processed at 2 goes for 3 to head of $\mathcal{Q}_1$; similarly, $C$ for 6 at head of $\mathcal{Q}_3$.

**Step 2:** $\mathcal{Q}_2$ remains empty. Outputs from 6 (at $\mathcal{M}_3$) and 3 (at $\mathcal{M}_1$) go to end of $\mathcal{Q}_4$ to wait for 7, 10, respectively. This step ends at $t_2$ with the completion of processing of 12 at head of $\mathcal{Q}_4$, leaving $\mathcal{Q}_4 \approx$ [ *?* *for* 7; ( *?*+$B$) *for* 10] with irrelevant sequencing.

**Step 3:** Output from $7 \mapsto 8$, from $10 \mapsto 11$, emptying $\mathcal{Q}_4$.

**Step 4:** $\mathcal{Q}_2$, $\mathcal{Q}_4$ remain empty. Output from 6 is processed at 7 without delay and goes on to 8 with total accumulation of $C$ since this step ends at $t_6$ with the clearing of 6 at the head of $\mathcal{Q}_3$. Output from 3 passes through 10 to 11 with accumulation of $(t_4 - t_1)/\tilde{\tau} = (\tau/\tilde{\tau})C$ joined to earlier $B$.

**Step 5:** Product at $8 \mapsto 9$ (amount $= C$), at $11 \mapsto 12$ (amount $= [B + (\tau/\tilde{\tau})C] =: X$), emptying $\mathcal{Q}_3$.

**Step 6:** $\mathcal{Q}_3$ remains empty. Output from 9 goes to 1; output from 3 goes to tail of $\mathcal{Q}_4$ for 7 (amount $= [A - (\tau/\tilde{\tau})C] =: B'$). Input to 4 accumulates at tail of $\mathcal{Q}_1$. Ends at clearing of 3 at head of $\mathcal{Q}_1$ at $t_2$.

**Step 7:** Product at $1 \mapsto 2$, at $4 \mapsto 5$ [mixed at tail of $\mathcal{Q}_2$], emptying $\mathcal{Q}_1$.

**Step 8:** $\mathcal{Q}_1$, $\mathcal{Q}_3$ remain empty. Output from 9 passes through 1 to 2 with total accumulation $= C =: A'$. Ends at clearing of 9 at head of $\mathcal{Q}_3$ at time $t_5 + \tau C = 2\tau C =:$

$t_8$. Input passes through 4 to 5 with total accumulation $= (t_8 - t_1) = 2\tau C =: C'$. What remains at 12 (at the head of $\mathcal{Q}_4$) is then

$$[X - (t_8 - t_5)/\tilde{\tau}] = [B + (\tau/\tilde{\tau})C] - (2\tau C - \tau C)/\tilde{\tau} = B =: D'.$$

We then have a new 'initial' state of the same form as before with

$$A' = C, \qquad B' = [A - (\tau/\tilde{\tau})C], \qquad C' = 2\tau C, \qquad D' = B. \tag{5.3}$$

We wish (5.3) to be simply a scaled version of the original state. Clearly the scale factor must be $\lambda := 2\tau$ to obtain the third equation of (5.3) and one then obtains $C = A' = 2\tau A$, $B = D' = 2\tau D$ so

$$2\tau B = B' = A - (\tau/\tilde{\tau})C = \left[1 - \tfrac{2\tau^2}{\tilde{\tau}}\right] A$$

whence, in terms of $A$, one must have

$$B = \frac{1 - 2\tau^2/\tilde{\tau}}{2\tau} A, \qquad C = 2\tau A, \qquad D = \frac{1 - 2\tau^2/\tilde{\tau}}{4\tau^2} A. \tag{5.4}$$

It is the necessity that $B, D$ be positive which imposes the first requirement of (2.4), that $2\tau^2 < \tilde{\tau}$. We note that to have the 'amplification factor' $\lambda > 1$ requires $\tau > 1/2$ and one easily sees that this, along with the requirement that $\tilde{\tau} < 1$, ensures the necessary inequalities (5.1).

Obviously, if $\tau > 1/\sqrt{2}$ one cannot have $2\tau^2 < \tilde{\tau} < 1$. In this case one can attempt an alternative behavioral tracking but a preliminary investigation suggests that the sequencing will now be chaotic and it is unclear whether or not one has instability. It would, of course, be an interesting paradox if, say with $\tilde{\tau} = .9$, slowing the processing time $\tau$ from .6 to .7 could somehow convert an unstable system to a stable one.

**Acknowledgment:** I would like to thank P.R. Kumar for the discussion in New Hampshire which originally stimulated this analysis — as well as for other stimulating discussions before and since. I would like to thank A. Yershov for his contributions to this paper. Finally, I would also like to thank the referees of the original version for comments which, I hope, have led to a clearer exposition.

# References

[1] M. Bramson, *Instability of FIFO queueing networks*, Ann. Appl. Prob., pp. 414-431, (1994).

[2] M. Bramson, *Instability of FIFO queueing networks with quick service times*, Ann. Appl. Prob., to appear.

[3] S.H. Lu and P.R. Kumar, *Distributed scheduling based on due dates and buffer priorities,*
IEEE Trans Autom. Control **AC-36**, pp. 1406–1416 (1991).

[4] P.R. Kumar and T.I. Seidman, *Dynamic instabilities and stabilization methods in distributed real time scheduling of manufacturing systems,* IEEE Trans Autom. Control **35**, pp. 289–298 (1990) [see also, pp. 2028–2031 in *Proc. 28*$^{\rm th}$ *IEEE CDC*, IEEE (1989)].

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