Preprints of the 18th IFAC World Congress
Milano (Italy) August 28 - September 2, 2011

# On Optimal Switching Interactive Decentralized Control of Networked Manufacturing Systems ⋆

**V. Feoktistova** * **A. Matveev** * **E. Lefeber** ** **J.E. Rooda** **

* *Department of Mathematics and Mechanics, Saint Petersburg University, Universitetskii 28, Petrodvoretz, St.Petersburg, 198504, Russia, (e-mail: horsa@yandex.ru, alexeymatveev@hotmail.com)*
** *Department of Mechanical Engineering, Eindhoven University of Technology, 5600 MB, Eindhoven, The Netherlands, (e-mail: A.A.J.Lefeber@tue.nl, j.e.rooda@tue.nl)*

**Abstract:** The paper considers standard fluid models of multi-product multiple-server production systems where setup times are incurred whenever a server changes product. Concerned is the general approach to the problem of optimizing the long-run average cost per unit time that offers to first determine an optimal steady state (periodic) behavior and then to design a feedback scheduling protocol ensuring convergence to this behavior as time progresses. The second part of this program is treated and a systematic presentation of a novel approach to it is offered. This approach gives rise to protocols that are cyclic and distributed: the servers do not need information about the entire system state. Each of them proceeds basically from the local data concerning only the currently served queue, although a fixed finite number of one-bit notification signals should be exchanged between the servers during every cycle. The approach is illustrated by simple albeit instructive examples concerning polling systems, single server systems with processor sharing scheme, and the re-entrant two-server manufacturing network with non-negligible setup times introduced by Kumar and Seidman. For the last network considered in the analytical form, the optimal steady-state (periodic) behavior is first determined. Based on the desired steady state behavior and the presented theory, we designed simple distributed feedback switching control laws for all examples. These laws not only obtain the required behaviors but also make them globally attractive, irrespective of the system parameters and initial state.

Keywords: Hybrid dynamical systems, Optimal switched control, Control of networks, Fluid models, Queueing.

## 1. INTRODUCTION

The paper deals with fluid models of production systems. They represent the system as a network that receives incoming product flows, interpreted as deterministic fluid streams, and processes them by means of servers. The servers move products (also called work) among internal buffers and ultimately dispatch work into the exterior of the network. The servers can alter their locations, which requires nonzero setup times. Such models are used to describe certain aspects of flexible manufacturing systems, computer, communication and transport networks, chemical kinetics, etc. Baker [1974], Perkins and Kumar [1989], Kelly and Williams [2004].

Recently a great deal of research was concerned with these models, see e.g., Savkin [2001], Dai and Vate [2000], Bramson [2008], Gamarnik and Hasenbein [2005] and the literature therein. It was shown that they may exhibit unexpectedly complicated and counter-intuitive behavior, especially if decentralized control policies and non-zero setup times are involved, see also Banks and Dai [1997], Kumar and Seidman [1990]. In Perkins and Kumar [1989], clear a fraction (CAF) policies were introduced and shown to achieve stability for single server systems, as well as for multi-server networks such that under some enumeration of the servers, work visits them in the ascending order. If such enumeration is impossible (which holds for e.g., re-entrant networks), CAF policies may fail to stabilize the system. In some cases, the so-called gated policies proposed in Humes [1994], Perkins et al. [1994] are able to overcome this drawback. The main idea behind them is to assign a certain level (gate) to every buffer and switch the servers based on the buffer contents after the gate. However, gated policies carry potential for increase of the mean number of jobs in the system, which is undesirable from a performance point of view.

In Savkin [1998], a universal decentralized switching strategy was proposed and shown to stabilize very general multiple server networks with time-varying rates of the outer inflows. However, it does not provide a machinery of wip reduction. For example, the more work in the system

⋆ This work was supported by the Netherlands Organization for Scientific Research (NWO-VIDI grant 639.072.072; NWO visitors grant 040.11.131), the Russian Foundation for Basic Research (grant 09-08-00803), and Russian Federal Program (grant N 2010-1.1-111-128-033).

initially, the comparably more work remains afterwards. This is also undesirable from a performance point of view.

The above references display characteristic features of other works on feedback control of fluid networks. They start from more or less heuristically designed policies and proceed with study of the resultant system behavior. A different approach was proposed in Lefeber and Rooda [2006, 2008]. Assuming a pre-determined periodic process that represents the desired open-loop schedule, the approach sets as the objective development of a general technique to design a feedback switching policy that gives rise to this process and makes it globally attractive, thus equalizing the asymptotic qualities of service along any and the given processes, respectively. The last property is of especial interest if the given process is optimal or nearly optimal. The main concern of this paper is not optimization but is stable feedback generation of a desired periodic process.

In Feoktistova and Matveev [2009] a Poincaré-type technique was reported preliminary form. It is aimed at designs of decentralized controllers and offers to partition the required process into relatively simple phases, each associated with a specific combination of activities of different servers. The policy is to periodically repeat the resultant cycle of the phases, each governed by an individual control rule on the basis of local information. When the server completes the task for the phase, it broadcasts one-bit notification to the others and proceeds to the next phase as soon as it collects all notifications. Design of the phase control rules (PCR) is the core occupation. According to Feoktistova and Matveev [2009], the design objective is confined into the phase itself: it should be ensured that a certain set of properties hold for the phase dynamical operator, which maps the system state at the phase beginning into that at its end.

This paper offers an extended and systematic presentation of this technique and demonstrates, by means of examples, that it fits to handle the entire range of cases encountered in the optimization problem.

The body of the paper is organized as follows. Section 2 introduces a rather general model of multi-product multiple server system with setups. Section 3 presents general guidelines of switching policies design and the related mathematical background. [1] Sections 4, 5, and 6 respectively deal with polling systems, single server networks with processor sharing scheme, and the Kumar-Seidman system. Section 7 concludes the paper with some simulation results.

## 2. GENERAL MULTI-PRODUCT MULTIPLE SERVER SYSTEM WITH SETUPS

We consider a system that receives $F$ product flows, interpreted as fluid streams, and processes them by means of $S$ *servers*. The servers move products among $N$ internal *buffers* and ultimately dispatch them into the exterior of the system. We do not consider the case where a job may dynamically choose the server to be processed at, and assume that the production routes are specified a priori. A setup activity is required to switch a product type at any server.

[1] This material is partly from Feoktistova and Matveev [2009].

This system is represented by a directed graph with the set of nodes $\mathfrak{N} := \{1, \dots, N, \circledast_1, \dots, \circledast_F, \circledast_{\rm out}\}$. Here $\circledast_i$ is the *source* of the $i$th flow and $\circledast_{\rm out}$ is the *exterior* where work is ultimately delivered. Other nodes represent buffers enumerated by $n \in [1:N]$. The graph arcs display the paths along which work is moved. Any server $s$ has its own *service area* $I_s \subset [1:N]$, which form a partition of the set of buffers. The sources $\circledast_j$ have no incoming arcs, the exterior $\circledast_{\rm out}$ has no outgoing arcs. There is only one arc starting at $n \neq \circledast_{\rm out}$; its end is denoted by $\boldsymbol{next}(n)$. The graph contains no cycles and every buffer $n$ has incoming arcs. The rate $\lambda_i \geq 0$ of the flow from the source $\circledast_i$ is constant. Any server can serve only one buffer at a given time. Service of buffer $n$ consists in withdrawal of its content to $\boldsymbol{next}(n)$ at a rate $0 \leq u_n(t) \leq \mu_n$, where $\mu_n > 0$ is given. Switching the server from $n'$ to $n''$ consumes $\sigma_{n' \to n''} > 0$ time units.

The (*feasible*) state $(X, Q)$ consists of the *continuous state* $X = \{x_n \geq 0\}_{n=1}^N$ and the *discrete state* $Q = \left\{q_s \in I_s \cup \{\ominus\}\right\}_{s=1}^S$. Here $x_n$ is the content of buffer $n$ and $q_s$ is the state of server $s$, i.e., $q_s$ either indicates the buffer served or is the 'switching in progress' symbol $\ominus$. A *process* refers to a feasible evolution of the feasible state $[X(t), Q(t)]$ over time, i.e., evolution such that

(1) any function $q_s[\cdot]$ is piecewise constant and in the chronological list of its values, any two successive 'buffer' entries $n', n''$ are different $n' \neq n''$ and separated by the 'switching' one $\ominus$, which is maintained no less than $\sigma_{n' \to n''}$ time units [2];
(2) the function $X(\cdot)$ is absolutely continuous and for any buffer $n \in [1:N]$,

$$x_n(t) \geq 0, \quad \dot{x}_n(t) = \sum_{j \in \mathfrak{N}: n = \boldsymbol{next}(j)} u_j(t) - u_n(t),$$

where $u_{\circledast_i}(t) \equiv \lambda_i$ and for $j \neq \circledast_i \ \forall i$, $0 \leq u_j(t) \leq \mu_j \ \forall t$ and $u_j(t) = 0$ whenever $q_s \neq j \ \forall s$.

In practice, the system is usually governed by a *switching policy*. It endows each server with a rule to determine the current service rate $u_n(t)$ and to decide when this service should be terminated and which server should be served next. The problem to be treated in this paper is as follows:

**P)** *Given a periodic process $\pi^0$, a switching policy should be designed such that*
**P1)** *The process $\pi^0$ is generated by this policy;*
**P2)** *All processes converge to $\pi^0$ as $t \to \infty$.*

For the definition of process convergence, we refer the reader to Savkin [2001].

Given a switching policy, the process is determined by the initial state. So P1) means that there exists an initial state that gives rise to $\pi^0$. By P2), sooner or later, the system evolution closely follows $\pi^0$ irrespective of the initial state. This is of especial interest if $\pi^0$ is optimal or suboptimal. Then the policy ensures automatic transition to the optimal or suboptimal system behavior and its subsequent globally stable maintenance. The main

[2] Dropout of 'no less than' might seem more natural. 'No less than' is taken for the technical convenience. This is possible since prolonging the switching period is equivalent to continuing the previous service at the zero rate.

concern of this paper is not optimization but is stable feedback generation of desired periodic processes. So in what follows, the process $\pi^0$ is treated as pre-specified.

## 3. TRANSFORMATION OF A PERIODIC PROCESS INTO A SWITCHING POLICY: GENERAL GUIDELINES AND MATHEMATICAL BACKGROUND

According to Feoktistova and Matveev [2009], transformation of the periodic process $\pi^0 = [X^0(\cdot), Q^0(\cdot)]$ into a switching policy is arranged along the following lines:

- The optimal periodic process is partitioned into finitely many *phases*
$$\mathfrak{C} = \mathfrak{P}_0, \mathfrak{P}_1, \mathfrak{P}_2, \dots, \mathfrak{P}_{c-1}, \tag{1}$$
each associated with the discrete state transitions involved;
- Every phase is equipped with a *phase control rule* (PCR) to govern the system within the phase;
- The entire policy is to progress through the periodically repeated sequence of phases (1) while applying the relevant PCR within every phase;
- When a server completes the task for the phase, it broadcasts one-bit notification to the others and proceeds to the next phase as soon as it collects all notifications.

To promote decentralization, PCR are welcome to drive every server on the basis of only its own local data (i.e., that about the currently served buffer). Then within any phase, the control is completely decentralized and cooperation of servers comes to exchange of finitely many bits at the end of every phase.

The *dynamical operator* $T^{\mathfrak{P}_i}$ of phase $\mathfrak{P}_i$ maps the continuous state $X$ at the beginning of $\mathfrak{P}_i$ into that at the end (for a given PCR). The *monodromy operator* is the similar map for the entire cycle (1):
$$M = T^{\mathfrak{P}_{c-1}} \circ T^{\mathfrak{P}_{c-2}} \circ \dots \circ T^{\mathfrak{P}_1} \circ T^{\mathfrak{P}_0}. \tag{2}$$
The problem **P)** is solved whenever PCR's ensure:

- **i)** Any PCR generates the related piece of $\pi^0$;
- **ii)** Any trajectory of the iterated system $X(k+1) = M[X(k)], X(0) \geq 0$ converges to $X_0^0 := X^0(0)$ as $k \to \infty$.

Here i) guarantees that the entire switching policy does generate the required periodic process $\pi^0$ and also that $X_0^0$ is the equilibrium point of the iterated system. If the phase dynamical operators $T^{\mathfrak{P}_i}$ are continuous, ii) ensures convergence of all processes in the original fluid network to the desired periodic behavior $\pi^0$ by the standard argument presented in e.g., Savkin [1998, 2001].

To ensure i), the idea is to design PCR's so that they enforce the system to copycat the desired process $\pi^0$. Property ii) brings more trouble partly due to the curse of dimensionality: computation of the monodromy operator becomes cumbersome up to intractable as the numbers of servers or buffers increase. This burden is especially hard at the stage of design, where there is no specific monodromy operator to compute, and the actual task is to display and employ the relationships between this operator and particular designs of PCR's in order to choose the proper ones. The following new criterion for stability of equilibria of iterative dynamic systems aids to remove this blockage since this criterion can be verified and ensured 'phase-wise', thus annihilating the need to deal with the entire monodromy operator.

### *3.1 Mathematical Background*

The inequalities $x \leq y$ and $x < y$ for $x, y \in \mathbb{R}^p$ are meant component-wise. The operator $\mathcal{T} : K_+^p \to K_+^p := \{x \in \mathbb{R}^p : x \geq 0\}$ is said to be:

i) *monotone*, if $x \leq y \;\Rightarrow\; \mathcal{T}(x) \leq \mathcal{T}(y)$;
ii) *piecewise affine*, if a partition $K_+ = \bigcup_{j=1}^m S_j$ exists such that each set $S_j$ (called *cell*) has an interior point and is described by finitely many linear inequalities (both strict and non-strict), and all restrictions $T|_{S_j}$ are affine, i.e., $T(x) = A_j x + b_j \;\; \forall x \in S_j$, where $A_j \in \mathbb{R}^{p \times p}, b_j \in \mathbb{R}^p$;
iii) *dominated* if $b_j \geq 0 \, \forall j$; and *strictly dominated* if $b_j > 0 \, \forall j$.

The following theorem is the main result of this section.

*Theorem 1.* Suppose that an iteration $\mathcal{T}^m$ of a piecewise affine continuous monotone map $\mathcal{T}$ has a fixed point $\mathcal{T}[x_*] = x_* \in K_+^p$ and is strictly dominated in the domain $x : \{x \geq x_*\}$ and in the domain $x : \{x \leq x_*\}$. Then this fixed point is unique and attracts $x_k \xrightarrow{k \to \infty} x_*$ all trajectories of the iterated system $x_{k+1} = \mathcal{T}[x_k], x_0 \in K_+^p$.

With respect to the monodromy operator $\mathcal{T} := M$, the assumptions of continuity, monotonicity, and piecewise affinity can be checked 'phase-wise' since they are evidently inherited by compositions of the maps. As for the strict dominance, it can be shown that the composition not only inherits this property but also acquires it even if the composed maps are not strictly dominated.

## 4. POLLING SYSTEMS

We consider the particular case of the system from Section 2: there is only one server, $N \geq 2$ buffers, and $N$ outer flows (see Fig. 1(a)). The $n$th flow arrives at buffer $n$ at the constant rate $\lambda_n > 0$ and after service at a rate $0 \leq u_n(t) \leq \mu_n$, leaves the system. Switching between buffer requires a nonzero setup time.

Let we be given a $T$-periodic process[3] $\pi^0 = \Big[\{x_n^0(t)\}_{n=1}^N, q_1^0(t) \in [1:N] \cup \{\ominus\}\Big]$ for which the service rate of buffer $n = q_1(t)$ is maximal $u_n(t) = \mu_n$ if $x_n(t) > 0$, and equals the input rate $u_n(t) = \lambda_n$ if $x_n(t) = 0$. Without any loss of generality, we assume that $T$ is the end of a switching period. Following the lines of Section 3, we decompose $\pi^0$ into the simplest phases (1) by partitioning the period $0 = t_0^0 < t_1^0 < t_2^0 < \dots < t_{c-1}^0 < t_c^0 = T$ into the intervals where the discrete state is constant $q_1^0(t) \equiv q^i, t \in [t_i^0, t_{i+1}^0)$, $q_1^0(t_i - 0) \neq q_1^0(t_i + 0)$. Then any phase $\mathfrak{P}_i$ is associated with a discrete state $q^i$, which form the sequence
$$q^0 \mapsto q^1 = \ominus \mapsto q^2 \mapsto q^3 = \ominus \mapsto \dots \mapsto q^{c-2} \mapsto q^{c-1} = \ominus. \tag{3}$$

[3] The existence of such process implies that $\sum_{n=1}^N \lambda_n/\mu_n < 1$ Gross and Harris [1985].

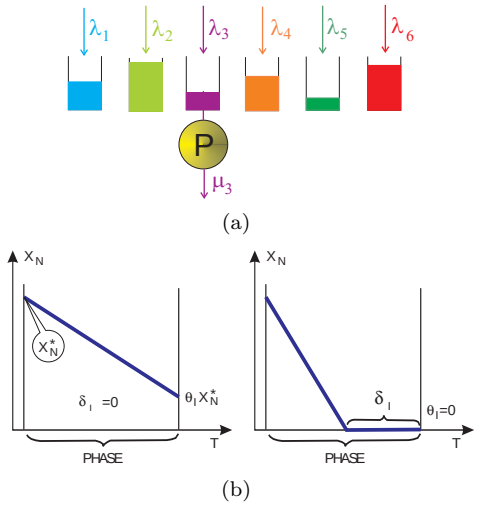



Fig. 1. (a) Polling system; (b) Basic quantities underlying the phase control rule

Now we introduce the phase control rules (PCR).

**Switching phase $q^i = \ominus$.** Switching is implemented during $\sigma_i^0$ time units, where $\sigma_i^0 > 0$ is its duration along the required periodic process $\pi^0$.

**Service phase $q^i = n \neq \ominus$.** We first introduce the following (see Fig. 1(b)):

**q.1)** $\theta_i^n$ — the fraction of the initial content of buffer $n$ at this phase that is retained in the buffer at the phase end for $\pi^0$, i.e., $\theta_i^n := \frac{x_n^0(t_{i+1}^0)}{x_n^0(t_i^0)} \in [0,1]$;

**q.2)** $\delta_i$ — the duration of the service at the rate $\lambda_n$ at this phase for $\pi^0$.

Note that $\theta_i^n \cdot \delta_i = 0$. Let $t_i$ stand for the time when the phase commences.

*Phase control rule:*

> Buffer $n = q^i$ is served at the maximal rate $\mu_n$ until its content reduces to the level $\theta_i^n x_n(t_i)$ and then at the input rate $\delta_i$ time units more.

The entire policy is to progress through the periodically repeated sequence of phases (3) while applying the relevant phase control rule within every phase.

Thus the server is driven by local data about the currently served buffer.

*Theorem 2.* The proposed policy gives rise to a unique periodic process, which is equal to $\pi^0$ and attracts all other processes.

**Proof.** The proposed PCR's trivially meet i) from Section 3. So the entire policy does generate $\pi^0$ and $X_0^0 := X^0(0)$ is the equilibrium of the monodromy operator $M$. By Theorem 1 and the standard argument presented in e.g., Savkin [1998, 2001], it suffices to show that the assumptions of Theorem 1 are true for $M$.

By elementary computation, the phase dynamical operators are as follows:

$$T^{\mathfrak{P}_i}X = \begin{pmatrix} x_1 + \lambda_1\tau \\ \vdots \\ x_{n-1} + \lambda_{n-1}\tau \\ \theta_i^n x_n \\ x_{n+1} + \lambda_{n+1}\tau \\ \vdots \\ x_N + \lambda_N\tau \end{pmatrix}, \quad \text{where } \tau := \delta_i + \frac{(1-\theta_i^n)x_n}{\mu_n - \lambda_n}, \quad \text{if } q^i = n \neq \ominus;$$

$$T^{\mathfrak{P}_i}X = \sigma_i^0 \cdot (\lambda_1, \ldots, \lambda_N)^\top \quad \text{if } q^i = \ominus.$$

They are clearly affine, continuous, monotone, and dominated. So evidently is their composition $M$. It is also strictly dominated since so is the operator $T^{\mathfrak{P}_{c-1}}$ (where $q^{c-1} = \ominus$ by (3)) that is the last to act in the composition (2). Thus the assumptions of Theorem 1 are satisfied. □

For this example, the phase dynamical operators are affine, which is not the case for the next example, where they are only piecewise affine.

## 5. SINGLE SERVER NETWORKS WITH PROCESSOR SHARING SCHEME

Now we consider modification of the previous example where the server can operate in several *modes*, enumerated by $m \in [1:M], M \geq 2$. In mode $m$, it simultaneously serves the buffers from a set $J_m \neq \emptyset$; these sets form a partition of $[1:N]$. Switching from mode $m_1$ to $m_2$ requires $\geq \sigma_{m_1 \to m_2} > 0$ time units. [4]

Let $\pi^0$ be a $T$-periodic process [5] for which any service of any buffer $n$ starts at the maximal rate $\mu_n$ and proceeds at the input rate $\lambda_n$, where any of these periods may be of zero duration, i.e., does not occur in effect. Like in Section 4, the interest to such processes is inspired by optimization issues.

Like in Section 4, it can be assumed that $T$ is the end of a switching period. To transform $\pi^0$ into a switching policy, we still decompose $\pi^0$ into the simplest phases (1) by partitioning $[0,T]$ into the intervals where the discrete state is constant. Then any phase $\mathfrak{P}_i$ from (1) is associated with either an active mode $\mathfrak{P}_i \sim m_i$ or switching $\mathfrak{P}_i \sim \ominus$, which are arranged in the sequence

$$m_0 \mapsto \ominus \mapsto m_2 \mapsto \ominus \mapsto \cdots \mapsto m_{c-2} \mapsto \ominus. \qquad (4)$$

**PCR for the switching phase $\mathfrak{P}_i \sim \ominus$** is to implement switching during $\sigma_i^0$ time units, where $\sigma_i^0 > 0$ is its duration along the process $\pi^0$.

[4] Though the system at hand is not a particular case of the network from Section 2, it can be emulated on such a network. To this end, we first equalize the sizes $|J_m|$ of all sets $J_m$ by inserting 'void' buffers $n$ with $\lambda_n := \mu_n := 0$ if necessary. Then we enumerate the buffers in every set $J_m$ by $r \in [1:k]$, where $k := |J_m|$, and replace the 'real' server by $k$ 'fictitious' ones. The service area of the $r$th of them is formed by the $r$th elements of $J_m$'s, the switch between two elements consumes the time of the switch between the related modes. The processes in the original system can be identified with those in the auxiliary $k$-server system for which all servers first, are switched synchronously and second, always serve buffers from a common set $J_m$. So if a switching policy designed for the auxiliary system gives rise only to processes with these properties, it can be interpreted as a policy for the original system.

[5] Existence of such process clearly implies that $\mu_n > \lambda_n \ \forall n$.

**PCR for the service phase $\mathfrak{P}_i \sim m_i$.** To state this rule, we employ the quantity $\theta_i^n$ from **q.1** in Section 4. Let $\delta_i^n$ denote the duration of the service of buffer $n$ at the input rate at phase $\mathfrak{P}_i$ for process $\pi^0$. PCR is as follows:

1) Every buffer $n \in J_{m_i}$ is served at the maximal rate $\mu_n$ until its content reduces to the level $\theta_i^n x_n(t_i)$, where $t_i$ is the time when $\mathfrak{P}_i$ is commenced;
2) When task 1) is completed for a buffer $n \in J_{m_i}$, the server reduces the service rate for this buffer to the input rate $\lambda_n$ and maintains it no less than $\delta_i^n$ time units and until the phase end;
3) The phase is terminated as soon as task 1) is accomplished and the compulsory time $\delta_i^n$ of service at the input rate is expired for all buffers $n \in J_{m_i}$.

The entire policy is to progress through the periodically repeated sequence of phases (4) while applying the relevant phase control rule within every phase.

Thus the server is driven only by data about the currently served buffers.

*Theorem 3.* The proposed policy gives rise to a unique periodic process, which is equal to $\pi^0$ and attracts all other processes.

**Proof.** Similarly to the proof of Theorem 2, it suffices to show that the assumptions of Theorem 1 are true for the monodromy operator $M$. By elementary computation, the phase dynamical operators are as follows:

$$T^{\mathfrak{P}_i} X = \{y_n\}_{n=1}^N,$$

$$\text{where} \quad y_n = \begin{cases} \theta_i^n x_n & \text{if } n \in J_{m_i} \\ x_n + \lambda_n \tau & \text{otherwise} \end{cases}$$

$$\tau := \max_{n \in J_{m_i}} \left[ \delta_i^n + \frac{(1-\theta_i^n) x_n}{\mu_n - \lambda_n} \right] \qquad \text{for} \quad \mathfrak{P}_i \sim m_i \neq \ominus;$$

$$T^{\mathfrak{P}_i} X = \delta_i^0 \cdot (\lambda_1, \ldots, \lambda_N)^\top \qquad \text{for} \quad \mathfrak{P}_i \sim \ominus.$$

They are clearly piecewise affine, continuous, monotone, and dominated. So evidently is their composition $M$. It is also strictly dominated since so is the operator $T^{\mathfrak{P}_c - 1}$ (by (4)) that is the last to act in the composition (2). Thus the assumptions of Theorem 1 are satisfied. □

## 6. THE KUMAR-SEIDMAN MANUFACTURING NETWORK

This network is assembled of four buffers and two servers and processes a single job flow, see Fig. 2 Work arrives

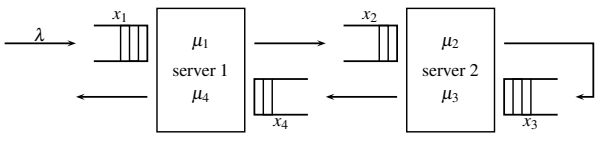



Fig. 2. The Kumar-Seidman model

at the first buffer at a constant rate $\lambda > 0$, then is consecutively processed by server 1, then twice by server 2, and finally by server 1 once more, and then leaves the system. Any server is capable to serve only one buffer at a given time. Switching between buffers consumes setup times $\sigma_{1\to4},\ \sigma_{4\to1},\ \sigma_{2\to3},\ \sigma_{3\to2} > 0$, respectively. The maximal service rate is $\mu_n > 0$ for buffer $n$. Thus the

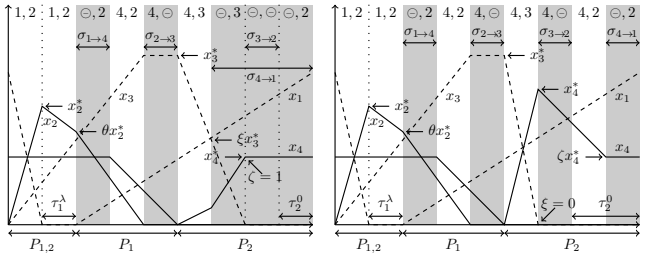



Fig. 3. The optimal periodic behavior.

continuous state $X = \{x_n\}_{n=1}^4$ and the service areas are as follows $I_1 = \{1, 4, \ominus\}, I_2 = \{2, 3, \ominus\}$.

The system is *stabilizable*, i.e., the total amount of work can be kept bounded provided that the system is properly controlled. This holds if and only if every server has enough capacity to process the job inflow Gross and Harris [1985]:

$$1-\rho_1-\rho_4 > 0, \quad 1-\rho_2-\rho_3 > 0, \quad \text{where} \quad \rho_i := \lambda/\mu_i. \tag{5}$$

The model at hand was introduced in Kumar and Seidman [1990] to demonstrate that the clearing policy is inappropriate since it may cause instability: even if (5) holds, the total amount of work may explode whenever

$$\rho_2 + \rho_4 > 1. \tag{6}$$

It is this case that is examined: (5) and (6) are assumed to be true. Then $\mu_1 > \mu_2$, $\mu_3 > \mu_4$, i.e., $\dot{x}_2 > 0$ (or $\dot{x}_4 > 0$) if buffers 1 and 2 (or 3 and 4) are simultaneously served at the maximal rates.

### *6.1 Optimal periodic behavior of the Kumar-Seidman system*

In Lefeber [2011] it has been determined which periodic behavior minimizes the weighted wip (work in progress):

$$W(\pi) := \overline{\lim_{T\to\infty}} \frac{1}{T} \int_0^T \sum_{n=1}^4 c_n x_n(t)\, dt \tag{7}$$

We restrict ourselves to the case $\rho_2 \leq \rho_4$ and $\sigma_{4\to1} + \sigma_{2\to3} \leq (1-\rho_2)T$. Then the optimal trajectory is as shown in Fig. 3. The gray vertical stripes in the figures highlight switching activities. The buffers are served at the maximal feasible rates. The difference between Figs. 3(a) and (b) concerns only the evolution of buffer 4 at phase $P_2$. Case 1(b) occurs if an only if $\sigma_{4\to1} < \sigma_{3\to2} + \tau_2^0$, where $\tau_2^0$ is the idling time of server 2. Then the content of buffer 4 decreases during some sub-phase of this phase. In case 1(a), $\sigma_{4\to1} \geq \sigma_{3\to2} + \tau_2^0$, and the content of buffer 4 never decreases at this phase.

To design the switching policy, we also need the following parameters of the optimal process, for which explicit expressions can be easily found based on the formulas from Lefeber [2011].

*Notation 1.* $\tau_2^0$ — the idle time of server 2 within phase $P_2$, see Fig. 3;
$\tau_1^\lambda$ — the duration of the period when server 1 serves the emptied buffer 1 at the input rate $\lambda$ at phases $P_{1,2}$, see Fig. 3;
$\theta$ — the fraction of the maximal content of buffer 2 at phase $P_{1,2}$ that remains in this buffer at the phase end, see Fig. 3(b);

$\xi$ — the fraction of the buffer 3 initial content $x_3^*$ at phase $P_2$ that remains there at the start of server 1 switching $4 \to 1$, see Fig. 3;
$\zeta$ — the fraction of the buffer 4 content $x_4^*$ at the start of server 2 switching $3 \to 2$ at phase $P_2$ that is in this buffer at the first time instant when both servers are involved in switching within the phase, see Fig. 3;

### *6.2 Optimal switching policy*

Now by following the guidelines from Section 3, we propose a simple interactive switching policy that ensures that after a transient and irrespective of the initial state, the system inevitably exhibits the optimal periodic behavior illustrated in Fig. 3. The phase control rules are designed so that the system behavior copycats that from Fig. 3.

**Switching policy**

(1) Whenever any buffer $i$ is served, the service is at the maximal feasible rate:

$$u_n = \begin{cases} \mu_n & \text{if } x_n > 0 \\ u_{n-1} & \text{if } x_n = 0 \end{cases}, \text{where } u_0 := \lambda; \qquad (8)$$

(2) The servers are switched so that the discrete state $Q(t) = [q_1(t), q_2(t)]$ periodically repeats the cycle:

$$\underbrace{\to (1,2) \xrightarrow{(a)} (\ominus, 2) \to \left| \begin{matrix} (4,2) \\ \text{or} \\ (\ominus, \ominus) \end{matrix} \right| \to (4, \ominus)}_{P_1} \xrightarrow{(b)}$$

$$\underbrace{\to (4,3) \to \left| \begin{matrix} (\ominus, 3) \\ \text{or} \\ (4, \ominus) \end{matrix} \right| \to \left| \begin{matrix} (4,2) \\ \text{or} \\ (\ominus, \ominus) \end{matrix} \right| \to (\ominus, 2)}_{P_2} \xrightarrow{(c)}; \qquad (9)$$

(3) Transition $(a)$ is implemented as soon as
3.a) buffer 1 is emptied
3.b) and after this the level of buffer 2 is reduced to the value $\theta x_2(\tau)$.
Here $\tau$ is the time when event 3.a) occurs, and $\theta$ is introduced in Notation 1;
(4) Within phase $P_1$,
- server 1 switches from buffer 1 to 4 for $\sigma_{1\to 4}$ time units and then serves buffer 4 until emptying and possibly longer, waiting for the switch of server 2 to be completed;
- Server 2 serves buffer 2 until emptying, then switches to buffer 3 for $\sigma_{2\to 3}$ time units and then possibly idles, waiting for emptying buffer 4.

(5) Transition $(b)$ from phase $P_1$ to $P_2$ is implemented as soon as first, buffer 4 is empty and second, switching of server 2 from buffer 2 to 3 is completed;
(6) Within phase $P_2$,
- Server 2 empties buffer 3, then switches to buffer 2 for $\sigma_{3\to 2}$ time units, and finally idles for $\tau_2^0$ time units and possibly longer, waiting for the switch of server 1 to be completed.
- Server 1 serves buffer 4 until the content of buffer 3 decays to $\xi x_3^*$ and after this the level of buffer 4 is reduced to $\zeta x_4(\tau_*)$, where $x_3^*$ is the buffer level at the start of the phase and $\tau_*$ is the time instant when the first of these reductions is completed. After this, server 1 switches to buffer 1 for a duration of $\sigma_{4\to 1}$ time units and then possibly idles, waiting for the compulsory idling time $\tau_2^0$ of server 2 to be expired.

Here $\tau_2^0, \xi$, and $\zeta$ are introduced in Notation 1;
(7) Transition (c) is implemented as soon as switching of server 1 is completed and the compulsory idling time $\tau_2^0$ of server 2 is expired.

*Remark 6.1.* **i)** Formula (9) displays the longest chains of discrete state transitions that may be observed during phases $P_1$ and $P_2$; the rigorous definitions of these phases are given in 4 and 6, respectively. Some sub-phases, like $(4, 2)$ in $P_1$, may be missed depending on the initial state and the serial number of the cycle (9) at hand. Such phases are said to be *flexible*.
**ii)** To determine the end of the current phase, any server needs the one-bit 'end of mission' notification from the companion server.
**iii)** Within the flexible phase $P_2$ in case 1(b) and phase $P_1$, operation of every server is based on data about the current level of the buffer served. In particular, each server operates with no regard to what is going on with the other server. As for $P_2$ in case 1(a), server 1 needs a one-bit notification that the required decrease in the level of buffer 3 is achieved.
**iv)** Rule 6 is well-defined since buffer 4 should be unloaded only in case 1(b), where server 2 does not supply work to it from the start of the unload and until the phase ends.
**v)** For the definiteness, we assume that $Q(0) = (1, 2)$. Then given the initial state $X(0)$, policy 1 uniquely determines a process in the system.

The the proposed policy ensures asymptotically optimal performance of the closed-loop system.

*Theorem 4.* Let the conditions (5) and (6) hold. The policy 1 gives rise to a unique periodic process, which attracts all other processes in the Kumar-Seidman system. Moreover, this periodic process represents the optimal behavior illustrated in Fig. 3.

### *6.3 Proof of Theorem 4*

The proposed phase control rules trivially meet the requirement i) from Section 3. So the entire switching policy generates the periodic process illustrated in Fig. 3 and $X_0^0 := X^0(0)$ is the equilibrium of the monodromy operator $M$. By Theorem 1 and the standard argument presented in e.g., Savkin [1998, 2001], it suffices to show that the assumptions of this theorem are true for this operator. The phase dynamical operators are easily computed:

$$\mathcal{T}_{1,2}X = \begin{pmatrix} 0 \\ \theta\dfrac{\mu_1 - \mu_2}{\mu_1 - \lambda} x_1 \\ x_3 + \mu_2 \left[ \dfrac{x_1}{\mu_1 - \lambda} + \dfrac{1-\theta}{\mu_2 - \lambda} \left( x_2 + \dfrac{\mu_1 - \mu_2}{\mu_1 - \lambda} x_1 \right) \right] \\ x_4 \end{pmatrix}$$

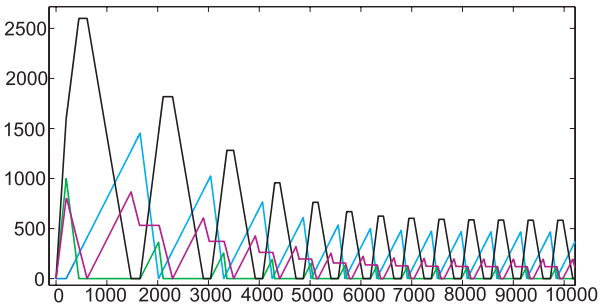
$$\mathcal{T}_{P_1}X = \begin{pmatrix} x_1 + \lambda \max\left\{ \dfrac{x_2}{\mu_2} + \sigma_{2\to 3}; \dfrac{x_4}{\mu_4} + \sigma_{1\to 4} \right\} \\ 0 \\ x_2 + x_3 \\ 0 \end{pmatrix}$$

$$\mathcal{T}_{P\,2}X = \begin{pmatrix} x_1 + \lambda \max\left\{ \dfrac{x_3}{\mu_3} + \sigma_{3\to 2} + \tau_2^0 \;;\; c + \dfrac{1-\zeta}{\mu_4} b + \sigma_{4\to 1} \right\} \\ x_2 \\ 0 \\ \zeta b \end{pmatrix},$$

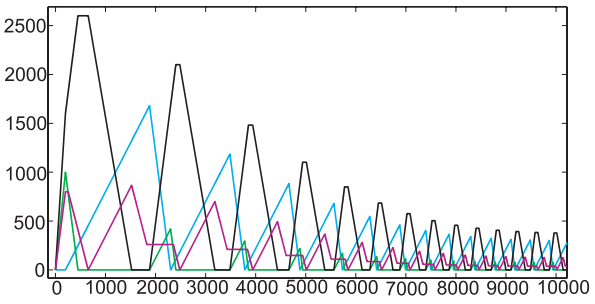
where $b = x_4 + \frac{\mu_3 - \mu_4}{\mu_3}(1-\xi)x_3$ and $c = \frac{1-\xi}{\mu_3}x_3$.
They are clearly piecewise affine, continuous, monotone, and dominated. So evidently is their composition $M = \mathcal{T}_{P\,2} \circ \mathcal{T}_{P\,1} \circ \mathcal{T}_{1,2}$. As for the strict dominance, we are going to examine $M^2$. It is easy to see that $+\lambda\sigma_{2\to 3}$ or $+\lambda\sigma_{1\to 4}$ in the first line of the formula for $\mathcal{T}_{P\,1}X$ is converted into $+\text{const}(>0)$ at the first position of $\mathcal{T}_{P\,2} \circ \mathcal{T}_{P\,1}X$, in addition to the constant addend $\lambda[\sigma_{3\to 2} + \tau_2^0]$ or $\lambda\sigma_{4\to 1}$. It follows that $\mathcal{T}_{12} \circ MX$ contains $+\text{const}(>0)$ at the second and third positions; $\mathcal{T}_{P\,1} \circ \mathcal{T}_{12} \circ MX$ contains $+\text{const}(>0)$ at the first and third positions; $\mathcal{T}_{P\,2} \circ \mathcal{T}_{P\,1} \circ \mathcal{T}_{12} \circ MX = M^2X$ contains $+\text{const}(>0)$ at the first and forth positions. The second and third positions of $MX$ are always zero. So by truncating the state space to $\mathbb{R}^2 = \{\mathbf{col}\,(x_1, x_4)\}$, we make $M^2$ strictly dominated. So the assumptions of Theorem 1 are satisfied. □

## 7. SIMULATION

Consider the Kumar-Seidman manufacturing network with following initial data: $\lambda = 1$, $\mu_1 = 5$, $\mu_2 = 4$, $\mu_3 = 3$, $\mu_4 = 2$(see Fig. 4(a) and 4(b)). The initial level of each buffer is equal to 800, T=366,6667. As was mentioned, the difference between cases 1(a) and 1(b) concerns the duration of switching times. For case 1(a), $\sigma_{1\to 4} = 10$, $\sigma_{2\to 3} = 100$, $\sigma_{3\to 2} = 10$, $\sigma_{4\to 1} = 100$, which implies $\theta = 1$, $\zeta = 1$, $\xi = 0,3864$, $\tau_2^0 = 42,7778$. For case 1(b), all switching times are equal to 55, which implies $\theta = 1$, $\zeta = 0,3$, $\xi = 0$, $\tau_2^0 = 42,7778$. The simulation results demonstrate high enough rate of convergence of the proposed policy.




(a)




(b)

Fig. 4. Case 1(a) and 1(b), correspondingly

## REFERENCES

K. R. Baker. *Introduction to Sequencing and Scheduling.* New York: Wiley & Sons, 1974.

J. Banks and J. G. Dai. Simulation studies of multiclass queueing networks. *IIE Transactions*, 29:213–219, 1997.

M. Bramson. *Stability of Queueing Networks*, volume 1950 of *Lecture Notes in Mathematics*. Springer-Verlag, 2008.

J. G. Dai and J. H. Vande Vate. The stability of two-station multitype fluid networks. *Operations Research*, 48:721–744, 2000.

V. Feoktistova and A. Matveev. Generation of production cycles in multiple server systems with setup times: The case study. In *Proceedings of the 13th IFAC Symposium on Information Control Problems in Manufacturing*, pages 568–573, Moscow, Russia, July 2009.

D. Gamarnik and J. J. Hasenbein. Instability in stochastic and fluid queueing networks. *Annals of Applied Probability*, 15:1652–1690, 2005.

D. Gross and G.M. Harris. *Fundamentals of Queueing Theory.* Wiley Series in Probability and Mathematical Statistics, 1985.

C. Humes. A regulator stabilization technique: Kumar-Seidman revisited. *IEEE Transactions on Automatic Control*, 39(1):191–196, 1994.

F. P. Kelly and R.J. Williams. Fluid model for a network operating under a fair bandwidth sharing policy. *Annals of Applied Probability*, 14:1055–1083, 2004.

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*, 35(3):289–298, March 1990.

E. Lefeber. Optimal performance of multiclass queueing networks with setup times: The Kumar-Seidman network as an example. SE Report 2011-02, Eindhoven University of Technology, Systems Engineering Group, Department of Mechanical Engineering, Eindhoven, The Netherlands, 2011. URL `http://se.wtb.tue.nl/sereports`.

E. Lefeber and J. E. Rooda. *Controller design of switched linear systems with setups*, page 363(1). Physica A, 2006.

E. Lefeber and J.E. Rooda. Controller design for flow networks of switched servers with setup times: the Kumar-Seidman case as an illustrative example. *Asian Journal of Control*, 10(1):55–66, 2008.

J. Perkins and P. R. Kumar. Stable, distributed, real-time scheduling of flexible manufacturing / assembly / disassembly systems. *IEEE Transactions on Automatic Control*, 34(2):139–148, February 1989.

J. R. Perkins, C. Humes Jr., and P. R. Kumar. *Distributed scheduling of flexible manufacturing systems: stability and performance*, pages 10(2):133,141. IEEE Transactions on Robotics and Automation, April 1994.

A. V. Savkin. A switched single server system of order $n$ with all its trajectories converging to $(n-1)!$ limit cycles. *Automatica*, 37(2):303–306, 2001.

A.V. Savkin. Regularizability of complex switched server queueing networks modelled as hybrid dynamical systems. *Systems & Control Letters*, 35:291–299, 1998.