10th IFAC Workshop on
Intelligent Manufacturing Systems
July 1-2, 2010, Lisbon, Portugal

Designs of optimal switching feedback decentralized control policies for re-entrant queueing networks: A case study

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: We study the problem of optimal feedback switching control of two-server re-entrant manufacturing network with nozero setup times introduced by Kumar and Seidman. The optimal steady-state (periodic) behavior is determined for the system in the analytical form. A simple feedback switching control law is proposed under which the process in the system converges to the optimal steady state behavior irrespective of the initial state. This law is 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 framework of Kumar-Seidman model is used for presentation of a methodology of both design of a switching policy and justification of its convergence.

Keywords: Control of networks, Queuing networks with setups, Flexible manufacturing systems.

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 network exterior. Every server can serve no more than one buffer at any time instant; whenever the buffer is changed, setup times are incurred. Such models are used to describe certain aspects of flexible manufacturing systems, computer, communication and transport networks, chemical kinetics, etc. Perkins and Kumar (1989).

Recently a great deal of research was concerned with these models, see e.g., Chase et al. (1993); Horn and Ramadge (1997); Matveev and Savkin (2000); Bramson (2008) and the literature therein. It was shown that they may exhibit unexpectedly complicated and counter-intuitive behavior, especially whenever decentralized control policies and nonzero setup times are involved. For instance, it was established via computer simulation in Banks and Dai (1997) that standard policies may cause instability: the total amount of work increases without limits even if each server has enough capacity to cope with the incoming flows. In Kumar and Seidman (1990), it was rigorously proved that the clearing policy (serve the buffer until emptying) is unstable for very simple networks even if the setup times are zero. 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., reentrant networks), CAF policies may fail to stabilize the system Kumar and Seidman (1990). In some cases, the so-called gated policies Humes (1994); J. R. Perkins and Kumar (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 proceeding from not the entire backlog in the buffer but its excess over the gate. This shortens the time of buffer service, thus reducing the likelihood of the detrimental situation underlying instability: a server wastes its capacity due to deficiency in work supply from another server since the latter is occupied by another activity in a side buffer for a too long time. However, gated policies carry potential for increase of the mean number of work in the system, which is undesirable from the 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. The strategy arranges the system operation in repeated cycles of the fixed duration T; within any cycle, every server visits each of the associated buffers only once in a pre-specified cycle-invariant order. From any buffer, the server removes the amount of work identical to the cumulative network income brought for time T by all inflows that affect, either directly or indirectly, this buffer. If the buffer contains not enough work to do so, it is drained out. The next setup may be prolonged to fully consume the time reserved for the step. This strategy not only keeps wip ($=$'work in progress', i.e., the total amount of work in the system) bounded but also makes all trajectories eventually periodic in the case of constant arrival rates. At the same time, it does not provide a machinery of wip reduction. For example, the more work in the system initially, the comparably more work remains afterwards. This is undesirable from the performance point of view as well.

The above references display characteristic features of other related works in the area. They start from more or less heuristically designed policies and proceed with study of the resultant system behavior. With few exceptions (see, e.g., Gaudio et al. (2001) and the literature therein) focused on two buffer systems, the issue of performance optimization, when treated, is typically limited to choice of the parameters for a pre-specified policy. However, this issue becomes one of the major concerns in recent years as a result of complication of manufacturing processes and cost increase. Though optimal scheduling of systems with setups has an extensive literature, it is primarily focused on open-loop schedules and basically assumes a static and certain environment. This approach, being a backbone of production systems planning, is not suited well to deal with real-life dynamic and uncertain environments, which causes a reported gap between the theory and practice Ouelhadj and Petrovic (2008). The basic tool to cope with these uncertainties is feedback under which decisions are made on an ongoing basis from the current events in the system.

A systematic approach to bridging this gap between optimal open-loop scheduling and feedback control of networks of switching servers with setup times was proposed in Lefeber and Rooda (2006, 2008b). 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 whenever the given process is optimal or nearly optimal. Though most schedule optimization problems are NP-hard, relatively effective optimization techniques have been developed to treat them Ouelhadj and Petrovic (2008). The main concern of this paper is not optimization but is stable feedback generation of desired periodic processes.

A Lyapunov-type approach to this problem was proposed in Lefeber and Rooda (2006). The resultant controllers are central: access to the information about the entire system state is assumed for every server. In certain cases, decentralized controllers driven by only local data are needed. In Lefeber and Rooda (2008b), it was shown, by means of an example, that within the new view of the problem introduced in Lefeber and Rooda (2006), decentralized controllers can be derived as well. The proposed technique relies on computation of a Lyapunov function and so suffers much from the standard in the area 'curse of dimensionality'.

To break the curse, a computationally non-demanding Poincaré-type approach aimed at designs of decentralized controllers was reported in Feoktistova and Matveev (2009). It 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 all notifications are collected. Design of the phase control rules is the core occupation. According to Feoktistova and Matveev (2009), the design objective is confined into the phase itself. Specifically, 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 set is elaborated so that these properties are inherited by compositions and so inevitably hold for the monodromy operator (MO). This is the composition of the dynamical operators over the entire cycle of the phases; the system in fact evolves via iterations of MO. The above properties are such that being established for MO, they guarantee the global stability of the equilibrium point of the iteration process and whereby that of the required periodic behavior. The curse becomes broken by avoiding computation and analysis of the entire MO, which is typically cumbersome up to intractable. In Feoktistova and Matveev (2009), this technique was probed by application to an example, also treated in Lefeber and Rooda (2008b). It concerns the Kumar-Seidman system Kumar and Seidman (1990) and a periodic process optimal for only particular numerical values of the system parameters. Under them, this process features special properties, unnecessary in general, which were essentially utilized in the design and proofs.

This paper offers an extended and detailed presentation of the proposed approach. The objective is to demonstrate, by means of an example, that the approach fits to handle the entire range of cases encountered in the optimization problem. To this end, we first provide an exhaustive analysis of the optimal periodic behaviors of the Kumar-Seidman system treated in the general (analytical) form. This research is based on Lefeber and Rooda (2008a), where the optimal behavior was characterized in terms of the separate activities of the servers. In this paper, we disclose both the combination of these activities and its evolution over time. A simple distributed feedback switching control law is proposed that not only gives rise to the optimal steady state behavior but also makes it globally attractive, irrespective of the system parameters. Compared with Feoktistova and Matveev (2009), this requires elaboration of new phase control rules. In particular, a special emphasis is given to the concept of the flexible phase that encompasses several activities of every server and within which the servers are given the freedom to proceed to the next activity independently of each other.

The body of the paper is organized as follows. Section 2 introduces the system to be studied; its optimal periodic behavior is established in Section 3. Section 4 presents the proposed general guidelines of switching policies design and the related mathematical background. The main result is stated in Section 5, where the optimal switching policy is introduced. There is an appendix providing the necessary technical facts.

2. THE KUMAR-SEIDMAN SYSTEM

We consider the manufacturing system that is assembled of four buffers and two servers (machines), see Fig. 1 The content

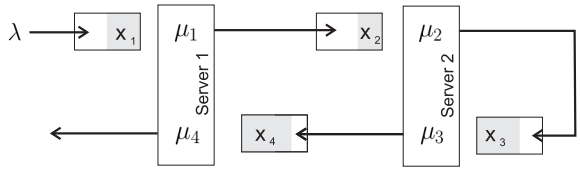

Fig. 1. The Kumar-Seidman model

of the buffers is called *work* and interpreted as fluid. Work arrives at the first buffer at the 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. Service of buffer n consists in withdrawal of its content $x_n(t)$ at the rate $u_n(t) \in [0, \mu_n]$, where $\mu_n > 0$ is given. 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:

$$1-\rho_1-\rho_4 > 0, \quad 1-\rho_2-\rho_3 > 0, \quad \text{where} \quad \rho_i := \mu_i^{-1}\lambda. \quad (1)$$

The model at hand was introduced in Kumar and Seidman (1990) to demonstrate that the clearing policy is unappropriate since it may cause instability: even if (1) holds, the total amount of work may explode. Moreover, this inevitably holds whenever

$$\rho_2 + \rho_4 > 1. \quad (2)$$

In this paper, we assume that (1) and (2) are true. Then

$$\mu_1 > \mu_2, \quad \mu_3 > \mu_4, \quad (3)$$

i.e., $x_2 \uparrow$ (or $x_4 \uparrow$) if buffers 1 and 2 (or 3 and 4) are simultaneously served at the maximal rates.

The system state (X, Q) consists of the *continuous state* $X(t) = \{x_i(t)\}_{i=1}^4$ and the *discrete state* $Q(t) = [q_1(t), q_2(t)]$, $q_1(t) \in \{1, 4, \ominus\}, q_2(t) \in \{2, 3, \ominus\}$. Here q_i is the state of machine i, i.e., q_i is either the serial number of the buffer served or the 'switching in progress' symbol $\ominus$. In practice, the system is usually governed by a switching policy. It endows each server with a rule to determine the service rate $u_n(t)$ and to decide when the service should be terminated and switch to the other buffer should be commenced. A *process* refers to feasible evolution of the system. The formal definition is as follows.

Definition 2.1. A *process* π is a functional dependency $[X(t), Q(t)]$ of the system state on time t such that the function $Q[\cdot]$ is piece-wise constant; time $\sigma_{n\to m}$ elapses between any two consecutive discontinuities of the forms $q_s(t'-0) = n \mapsto q_s(t'+0) = \ominus$ and $q_s(t''-0) = \ominus \mapsto q_s(t''+0) = m$ for $s = 1, 2$; the functions x_i are absolutely continuous, and

$$x_i(t) \geq 0, \dot{x}_i(t) = u_{i-1}(t) - u_i(t) \quad i = 1, \dots, 4, \quad \text{where}$$

$$u_0(t) := \lambda, u_i(t) \left\{ \begin{array}{ll} = 0, & \text{if } i \neq q_1(t), q_2(t) \\ \in [0, \mu_i], & \text{if } i \in \{q_1(t), q_2(t)\} \end{array} \right\} \text{for } i \geq 1.$$

This paper is concerned with design of the optimal switching policy that minimizes the long-run averaged weighted wip:

$$W(\pi) := \varlimsup_{T\to\infty} \frac{1}{T} \int_0^T w(t)\, dt \to \min,$$

$$\text{where} \quad w(t) := \sum_{n=1}^4 c_n x_n(t), \quad c_n > 0. \quad (4)$$

The design is accomplished via two steps. First, optimization is performed over periodic processes. Second, we propose a feedback interactive switching policy that makes all processes converging to the optimal periodic one, thus equalizing their asymptotic qualities of service (4).

3. OPTIMAL PERIODIC BEHAVIOR

A periodic processes is said to be *simple* if every machine processes each of the associated buffers only ones during the period. The recent publication Lefeber and Rooda (2008a) characterizes the optimal simple periodic process in terms of the separate activities of the first and second servers. For switching policy design, we need to know more: how these activities are combined and how this combination evolves over time. Now we provide the required complement, thus giving the exhaustive description of the optimum, under the following technical assumption borrowed from Lefeber and Rooda (2008a).

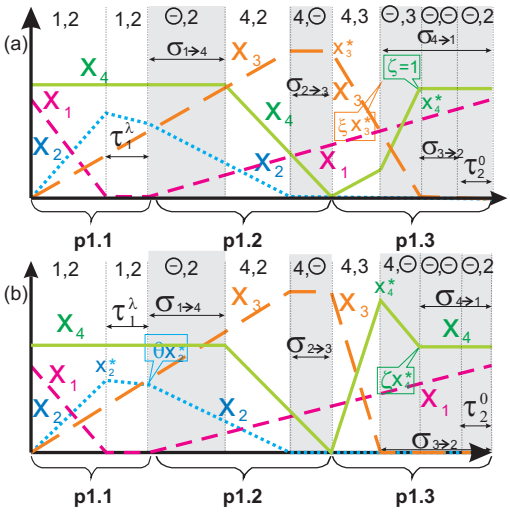

Fig. 2. Case 1 of the optimal periodic behavior.

Assumption 3.1. No downstream buffer values more than an upstream one, i.e., in (4), $c_1 \geq c_2 \geq c_3 \geq c_4 > 0$.

Now we are in a position to state the main result of the section.

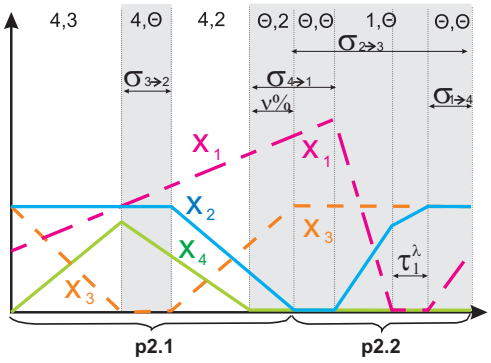

Fig. 3. Case 2 of the optimal periodic behavior

Proposition 1. The optimal simple periodic process does exist. There are two different types of the optimal periodic behavior illustrated by Figs. 2 and 3, respectively. [1] For each of them, the buffers are served at the maximal feasible rates.

In Case 1 from Fig. 2, the optimal behavior consists in periodic repetition of the following successive phases:

1.1) The servers simultaneously start services of buffers 1 and 2, serve them until buffer 1 is emptied and then any longer;

1.2) Server 1 switches to buffer 4 and serves it until emptying; server 2 empties buffer 2 and then switches to buffer 3. This switch is completed when buffer 4 is drained out;

1.3) Server 2 empties buffer 3 and then switches to buffer 2, where it idles for some time τ_2^0. Server 1 serves buffer 4 and then switches to buffer 1. This switch is completed synchronously with the end of the idling period of server 2, which is the end of the phase.

[1] Appendix A offers a rule to determine which of these cases holds. The grey vertical stripes in the figures highlight switching activities.

In Case 2 from Fig. 3, the optimal behavior consists in periodic repetition of the following successive phases:

2.1) The servers simultaneously start services of buffers 3 and 4. Server 2 empties buffer 3, then switches to buffer 2 and serves it until emptying, whereas server 1 serves buffer 4 until emptying and then begins switching to buffer 1. When this switching is in progress, buffer 2 is emptied, which is the end of the phase;

2.2) After emptying buffer 2, server 2 switches to buffer 3. Server 1 completes the switch to buffer 1, serves it until emptying and then any longer, and switches to buffer 4. Switches to buffers 3 and 4 are completed synchronously.

The proof of this proposition will be given in the full version of the paper.

The difference between Figs. 1(a) and (b) concerns only the evolution of buffer 4 at phase **1.3)**. Case 1(b) occurs if and only if $\sigma_{4\to1} < \widehat{\sigma}_{3\to2} := \sigma_{3\to2} + \tau_2^0$, where τ_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), $\sigma_{4\to1} \geq \widehat{\sigma}_{3\to2}$, and the content of buffer 4 never decreases at this phase.

Dissimilarities in Figs. 2 and 3 are underlaid by the fact that the problem is reducible to optimization of a linear function over a polytop, which solution may abruptly jump from one vertex to another under the continuous change of the parameters.

To design the switching policy, we need the following parameters, which explicit expressions are given in Appendix A.

p1) τ_2^0 — the time of server 2 idling at phase 1.3), see Fig. 2;

p2) τ_1^λ — the time during which the emptied buffer 1 is served at the input rate λ at phases 1.1) and 2.2), see Figs. 2, 3;

p3) θ — the fraction of buffer 2 maximal content at phase 1.1) that remains in this buffer at the phase end, see Fig. 2b ;

p4) ξ — the fraction of buffer 3 content x_3^* at the start of phase 1.3) that remains in this buffer at the start of server 1 switching $4 \to 1$, see Fig. 2;

p5) ζ — the fraction of buffer 4 content x_4^* at the start of server 2 switching $3 \to 2$ at phase 1.3) that is in this buffer at the first time instant when both servers are involved in switching within the phase, see Fig. 2;

p6) ν — the percentage of the switching period $\sigma_{4\to1}$ that elapses until buffer 2 is emptied at phase 2.2), see Fig. 3.

By combining Proposition 1 with the results of Lefeber and Rooda (2008a), exhaustive analytical description of the entire optimal periodic process can be obtained. However, this is not needed to design the optimal switching policy.

4. TRANSFORMATION OF A PERIODIC PROCESS INTO A SWITCHING POLICY: GENERAL GUIDELINES

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 *phases*
$$\mathfrak{C} = \mathfrak{P}_0, \mathfrak{P}_1, \mathfrak{P}_2, \ldots, \mathfrak{P}_{c-1}, \tag{5}$$
each identified by the discrete state transitions involved;
- Every phase is equipped with a *phase control rule* (**PhCR**) to govern the system within the phase;
- The entire policy is to progress through the periodically repeated sequence of phases (5) while applying the relevant phase control rule within every phase;
- On completion the task for the phase, each server broadcasts one-bit notification to the others and proceeds to the next phase as soon as all notifications are collected.

Let $X^0[b|\mathfrak{P}_i]$ and $X^0[e|\mathfrak{P}_i]$ be the values of $X^0(\cdot)$ at the start and end of phase $\mathfrak{P}_i$, respectively; and let $T^{\mathfrak{P}_i}$ map X at the beginning of $\mathfrak{P}_i$ into that at the end (for a given PhCR). The *monodromy operator* is the similar map for the entire cycle (5):

$$M = T^{\mathfrak{P}_{c-1}} \circ T^{\mathfrak{P}_{c-2}} \circ \cdots \circ T^{\mathfrak{P}_1} \circ T^{\mathfrak{P}_0}. \tag{6}$$

PhCR should be designed so that the following properties hold:

i) Every PhCR generates the related piece of π^0, i.e., $T^{\mathfrak{P}_i} X^0[b|\mathfrak{P}_i] = X^0[e|\mathfrak{P}_i]\ \forall i$;

ii) Any trajectory of the iterated system $X(k+1) = M[X(k)]$ converges to $X_0^0 := X^0[b|\mathfrak{P}_0]$ as $k \to \infty$.

Thanks to i), the entire switching policy does generate the required periodic process π^0 and X_0^0 is the equilibrium point of the iterated system. If the operators $T^{\mathfrak{P}_i}$ are continuous, ii) ensures convergence of all processes in the original fluid network to the desired behavior π^0 by the standard argument presented in e.g., Matveev and Savkin (2000); Savkin (1998).

To ensure i), PhCR are designed so that the system copycats the desired process. Property ii) brings more trouble since computation of the monodromy operator M 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 M and particular designs of PhCR 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.

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:

monotone, if $x \leq y \Rightarrow \mathcal{T}(x) \leq \mathcal{T}(y)$;

piece-wise 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 $A_j \in \mathbb{R}^{p\times p}, b_j \in \mathbb{R}^p$;

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

The proof of this theorem will be given in the full version of the paper.

With respect to the monodromy operator $\mathcal{T} := M$, continuity, monotonicity, and piece-wise 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.

5. OPTIMAL SWITCHING POLICY

Now by following the above lines, we propose a simple interactive switching policy that ensures that after a transient and irrespective of the initial state, the system exhibits the optimal periodic behavior described in Proposition 1, thus making the overall asymptotic quality of service (4) optimal. Since there are two qualitatively different optimal behaviors, two switching policies are offered to handle the cases where the first or second behavior occurs, respectively. The partition (5) of the process into phases is borrowed from Proposition 1. PhCR are designed so that the system behavior copycats that from Fig. 2 or 3.

Switching policy 1 (to be applied in Case 1 from Fig. 2)

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

$$u_i = \mu_i^{max} := \begin{cases} \mu_i & \text{if } x_i > 0 \\ u_{i-1} & \text{if } x_i = 0 \end{cases}; \qquad (7)$$

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

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

$$\underbrace{(4,3) \longrightarrow \left| \begin{array}{c} (\ominus,3) \\ \text{or} \\ (4,\ominus) \end{array} \right| \longrightarrow (\ominus,\ominus) \longrightarrow (\ominus,2)}_{P_2} \xrightarrow{(c)}; \qquad (8)$$

3) Transition (a) is implemented as soon as buffer 1 is emptied (at some time τ) and after this the level of buffer 2 is reduced to $\theta x_2(\tau)$, where θ is introduced in p3);

4) Within phase P_1,
- server 1 switches from buffer 1 to 4 during $\sigma_{1\to4}$ time units and serves buffer 4 until emptying and maybe, longer, waiting for switch of server 2 to be completed;
- Server 2 first serves buffer 2 until emptying, then switches to buffer 3 during $\sigma_{2\to3}$ time units and then maybe, idles waiting for emptying buffer 4.

5) Transition (b) is implemented as soon as buffer 4 is empty and switching of server 2 from buffer 2 to 3 is completed;

6) Within phase P_2,
- Server 2 empties buffer 3, switches to buffer 2 for $\sigma_{3\to2}$ time units, and finally idles for τ_2^0 time units and, maybe, longer, waiting for the switch of server 1 to be completed.
- Server 1 serves buffer 4 until the content of buffer 3 decays to ξx_3^0 and after this the level of buffer 4 is reduced to $\zeta x_4(\tau_*)$, where x_3^0 is the buffer level at the start of the phase and τ_* is the time instant when the first of these reductions is completed. After this, server 1 switches to buffer 1 during $\sigma_{4\to1}$ time units and then maybe, idles, waiting for the compulsory idling time τ_2^0 of server 2 to be expired.

Here τ_2^0, ξ, and ζ are introduced in p1), p4), and p5);

7) Transition (c) holds as soon as server 1 completes the switch and the compulsory idling of server 2 is ceased.

Formula (8) displays the longest chains of discrete state Q 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 (8) at hand. Such phases are said to be *flexible*.

To recognize the phase end, any server needs one-bit 'end of mission' notification from the companion server. Within phase P_2 in case 1(b) and phase P_1, every server acts proceeding from the current level of the served buffer with no regard to the other server. As for P_2 in case 1(a), server 1 needs a one-bit notification that the required decrease of x_3 is achieved.

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 since the start of the unload and until the phase end.

We assume that $Q(0) = (1,2)$. Then given the initial state $X(0)$, Policy 1 uniquely determines a process in the system.

Switching policy 2 (to be applied in Case 2 from Fig. 3)

1) Rule **1)** of Policy 1 is employed;

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

$$\longrightarrow \underbrace{(4,3) \longrightarrow (4,\ominus) \longrightarrow (4,2) \longrightarrow (\ominus,2)}_{P_1} \xrightarrow{\text{(a)}}$$

$$\underbrace{(\ominus,\ominus) \longrightarrow (1,\ominus) \longrightarrow (\ominus,\ominus)}_{P_2} \xrightarrow{\text{(b)}}; \qquad (9)$$

3) During phase P_1,
- Server 2 serves buffer 3 until it is emptied, then switches to buffer 2 during $\sigma_{3\to2}$ time units, then serves buffer 2 until emptying, and finally maybe idles waiting for server 1 to complete its mission for this phase;
- Server 1 empties buffer 4, then performs the first part of switch to buffer 1 during $\nu\sigma_{4\to1}$ time units, and finally maybe idles waiting for emptying buffer 2 by server 2.

Here ν is introduced in p6);

4) Transition (a) holds as soon as buffer 2 is emptied and the required percentage of switching $4 \to 1$ is completed;

5) During phase P_2 ,
- Server 1 completes switching $4 \to 1$ during $(1-\nu)\sigma_{4\to1}$ time units, empties buffer 1, continues to serve it at the input rate τ_1^λ (see p2)) time units and maybe, longer so that it leaves buffer 1 no sooner than $\sigma_{3\to2} - \sigma_{1\to4}$ time units elapses since the phase beginning, and finally switches to buffer 4 during $\sigma_{1\to4}$ time units;
- Server 2 switches from buffer 2 to 3 during $\sigma_{2\to3}$ time units and then maybe idles, waiting for server 1 to complete its mission.

6) Transition (b) holds when the switch $1 \to 4$ is completed.

The duration of service of the emptied buffer 1 at phase P_2 is adjusted so that switching $1 \to 4$ is completed at the earliest occasion after the end of the switch $2 \to 3$.

The servers operate independently and on the basis of data from only the currently served buffer within both phases P_1 and P_2.

For the definiteness, we assume that $Q(0) = (4,3)$. Then given the initial state $X(0)$, the switching policy 2 uniquely determines a process in the system.

By the following main result of the paper, the proposed policies ensure asymptotically optimal performance of the system.

Theorem 3. Let the conditions (1) and (3) hold. Suppose that policy 1 is applied in case 1 and policy 2 is put in use otherwise. Then any of these policies gives rise to a unique periodic pro-

cess, which attracts all other processes in the Kumar-Seidman system. Moreover, this periodic process represents the optimal behavior described in Proposition 1.

For the rigorous definition of the process convergence, we refer the reader to Matveev and Savkin (2000). The proof of Theorem 3 will be given in the full version of the paper.

REFERENCES

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

Bramson, M. (2008). Stability of queueing networks. volume 1950 of *Lecture Notes in Mathematics*. Springer-Verlag.

Chase, C., Serrano, J., and Ramadge, P.J. (1993). Periodicity and chaos from switched flow systems: Contrasting examples of discretely controlled continuous systems. *IEEE Trans. on Autom. Control*, 38(1), 70–83.

Feoktistova, V. and Matveev, A. (2009). Generation of production cycles in multiple server systems with setup times: The case study. In *Proc. of the 13th IFAC Symp. on Inf. Control Problems in Manufacturing*, 568–573. Moscow, Russia.

Gaudio, M.D., Martinelli, E., and Valigi, P. (2001). A scheduling problem for two competing queues with finite capacity and non negligible setup times. In *Proc. of the 40th IEEE Conf. on Decision and Control*, 2355–2360. Orlando, FL.

Horn, C. and Ramadge, P.J. (1997). A topological analysis of a family of dynamical systems with nonstandard chaotic and periodic behavior. *Int. Journal of Control*, 67(6), 979–1020.

Humes, C. (1994). A regulator stabilization technique: Kumar-Seidman revisited. *IEEE Trans. Aut. Cont.*, 39(1), 191–196.

J. R. Perkins, C.H.J. and Kumar, P.R. (1994). *Distributed scheduling of flexible manufacturing systems: stability and performance*, 10(2):133,141. IEEE Trans.Robotics and Autom.

Kumar, P.R. and Seidman, T.I. (1990). *Dynamic instabilities and stabilization methods in distributed real-time scheduling of manufacturing systems*, 35(3):289–298. IEEE Transactions on Automatic Control.

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

Lefeber, E. and Rooda, J.E. (2008a). Optimal behavior for the Kumar-Seidman network of switching servers. In *Proceedings of the 6th EUROMECH Nonlinear Dynamics Conference (ENOC'08)*. Saint Petersburg, Russia. Available at http://seweb.se.wtb.tue.nl/∼lefeber/refereed_proceedings.php.

Lefeber, E. and Rooda, J. (2008b). 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.

Matveev, A.S. and Savkin, A.V. (2000). *Qualitative Theory of Hybrid Dynamical Systems*. M. A., Birkhauser, Boston.

Ouelhadj, D. and Petrovic, S. (2008). A survey of dynamic scheduling in manufacturing systems. *Journal of Scheduling*, 1099–1425. ,Published online.

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

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

Appendix A

The following notations employ the quantities $\lambda, \mu_i, \rho_i, c_j$ from Fig. 1, (1) and (4), respectively, as well as the setup times $\sigma_{i\to j}$:

$$\alpha_0 := (c_1-c_2)\frac{\lambda(\sigma_{1\to4}+\sigma_{4\to1})^2}{2(1-\rho_1)}, \alpha_i := \sum_{j=1}^{4} c_j\alpha_{j,i}\ i=1,2,$$

$$\text{where } \alpha_{1,1} := \frac{\lambda\rho_4(\sigma_{1\to4}+\sigma_{4\to1})}{(1-\rho_1)}, \alpha_{2,2} := \frac{1}{2}\lambda(\rho_2-\rho_1),$$

$$\alpha_{2,1} := \frac{\lambda(1-\rho_1-\rho_4)(\sigma_{4\to1}+\sigma_{1\to4})}{(1-\rho_1)}, \quad \alpha_{3,1} := \lambda\sigma_{23},$$

$$\alpha_{4,1} := (\mu_4-\lambda)(\rho_2+\rho_4-1)+\frac{1}{2}\lambda(\rho_4-\rho_3), \alpha_{1,2} := \frac{\lambda\rho_4^2}{2(1-\rho_1)},$$

$$\alpha_{3,2} := \frac{1}{2}\lambda(\rho_2+\rho_3), \quad \alpha_{4,2} := (\mu_4-\lambda)(\sigma_{4\to1}+\sigma_{2\to3}),$$

$$T_{14} := \frac{\sigma_{1\to4}+\sigma_{4\to1}}{1-\rho_1-\rho_4}, \quad T_{23} := \frac{\sigma_{2\to3}+\sigma_{3\to2}}{1-\rho_2-\rho_3}, \quad T_* := \sqrt{\frac{\alpha_0}{\alpha_2}}.$$

The rule to distinguish Cases 1 and 2 illustrated in Figs. 2 and 3, respectively, and described in Proposition 1. Case 2 from Fig. 3 holds if and only if Lefeber and Rooda (2008a)

$$\begin{aligned} c_1 > c_2 \wedge T_* > T_{23} \wedge \mu_4 > \mu_2 + (c_2-c_4)(\mu_2-\lambda)c_4^{-1} \\ \wedge\ 2\sqrt{\alpha_0\alpha_2} + \alpha_1 \le \alpha_2 T_{23} + \alpha_1 + \frac{\alpha_0}{T_{23}} \\ + \left[c_2(\mu_2-\lambda) - c_2(\mu_4-\lambda)\right] T_{23} \times \\ \times \left[(\rho_2+\rho_4-1)T_{23} + \sigma_{2\to3} + \sigma_{3\to2}\right]; \end{aligned} \tag{A.1}$$

otherwise, Case 1 from Fig. 2 takes place.

Parameters p1)–p6). We need the formulas for the period T of the optimal periodic process Lefeber and Rooda (2008a):

$$T = \begin{cases} T_{14} \text{ if } T_{14} > T_{23} \text{ and either } c_1 = c_2 \text{ or } T_{14} \ge T_* \\ T_* \quad \text{if} \begin{cases} T_{14} > T_{23} \wedge c_1 > c_2 \wedge T_* > T_{14}; \quad \textbf{or} \\ T_{14} \le T_{23} < T_* \wedge c_1 > c_2 \\ \wedge \mu_4 \le \mu_2 + (c_2-c_4)(\mu_2-\lambda)c_4^{-1}; \ \textbf{or} \\ \begin{bmatrix} T_{14} \le T_{23} < T_* \wedge c_1 > c_2 \\ \wedge\ \mu_4 > \mu_2 + (c_2-c_4)(\mu_2-\lambda)c_4^{-1} \\ \text{and (A.1) does not hold} \end{bmatrix} \end{cases} \\ T_{23} \text{ if } T_{14} \le T_{23} \text{ and } \begin{cases} c_1 = c_2; \quad \textbf{or} \\ T_* \le T_{23}; \quad \textbf{or} \\ \text{(A.1) holds} \end{cases} \end{cases}$$

Straightforward computation based on Lemmas 6 and 7 Lefeber and Rooda (2008a) shows that the parameters introduced in **p1)–p6)** are given by

$$\tau_2^0 = (1-\rho_2-\rho_3)T - (\sigma_{2\to3}+\sigma_{3\to2}),$$

$$\tau_1^\lambda = \frac{1}{1-\rho_1}\left[(1-\rho_1-\rho_4)T - (\sigma_{1\to4}+\sigma_{4\to1})\right],$$

$$\theta = 1 - \frac{\mu_2-\lambda}{\mu_1-\mu_2}\frac{(1-\rho_1-\rho_4)T - (\sigma_{1\to4}+\sigma_{4\to1})}{\rho_1\left[\rho_4 T + \sigma_{1\to4} + \sigma_{4\to1}\right]},$$

$$\xi = \begin{cases} \mu_3\dfrac{\sigma_{4\to1}-\sigma_{3\to2}-\tau_2^0}{\lambda T} & \text{in case 1(a)} \\ 0 & \text{in case 1(b)} \end{cases},$$

$$\zeta = \begin{cases} 1 & \text{in case 1(a)} \\ 1-\mu_4\dfrac{\tau_2^0+\sigma_{3\to2}-\sigma_{4\to1}}{(\mu_3-\mu_4)\rho_3 T} & \text{in case 1(b)} \end{cases},$$

$$\nu = \frac{\sigma_{32}}{\sigma_{4\to1}} + [\rho_2+\rho_3-\rho_4]\frac{T}{\sigma_{4\to1}}.$$