Math. Control Signals Syst. (2012) 24:477–503
DOI 10.1007/s00498-012-0086-y

ORIGINAL ARTICLE

Designs of optimal switching feedback decentralized control policies for fluid queueing networks

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

Received: 12 September 2010 / Accepted: 26 March 2012 / Published online: 11 April 2012

Abstract The paper considers standard fluid models of multi-product multiple-server production systems where setup times are incurred whenever a server changes product. We consider a general approach to the problem of optimizing the long-run average cost per unit time that consists of first determining an optimal steady state (periodic) behavior and then to design a feedback scheduling protocol ensuring convergence to this behavior as time progresses. In this paper, we focus on the latter part and introduce a systematic approach. 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 instructive examples concerning polling systems, single server systems with

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), the Russian Federal Program (grant N 1.1-111-128-033) and the Russian Federal Program "Research and Teaching Cadres" (contract N 16.740.11.0042).

V. Feoktistova (✉) · A. Matveev
Department of Mathematics and Mechanics, Saint Petersburg University,
Universitetskii 28, Petrodvoretz, St. Petersburg 198504, Russia
e-mail: horsa@yandex.ru

A. Matveev
e-mail: alexeymatveev@hotmail.com

E. Lefeber · J. E. Rooda
Department of Mechanical Engineering, Eindhoven University of Technology,
5600 MB, Eindhoven, The Netherlands
e-mail: A.A.J.Lefeber@tue.nl

J. E. Rooda
e-mail: j.e.rooda@tue.nl

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, some cases of optimal steady-state (periodic) behavior are first recalled. For all examples, based on the desired steady state behavior and using the presented theory, we designed simple distributed feedback switching control laws. These laws not only give rise to 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 standard fluid models of production systems. We 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. [2,22,32].

Recently, a great deal of research was concerned with these models, see e.g., [5,6,8,12,18,35,36,38] 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. For instance, it was shown via computer simulation in [3] that some 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 [23], 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 [32], 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 [23]. In some cases, the so-called gated policies proposed in [20,21] 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 jobs in the system, which is undesirable from a performance point of view.

In [34], 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 a 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' = the total amount of work in the system) bounded but also makes all trajectories eventually periodic in the case of constant arrival rates. However, 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 a performance point of view as well.

The above references display characteristic features of other works on feedback control of fluid networks (see e.g., [11] for a recent reference). They start from more or less heuristically designed policies and proceed with study of the resultant system behavior. Performance optimization, when treated, is typically limited to the choice of the parameters for a pre-specified policy, with a few exceptions [1,7,9,13,19] which focused on two buffer systems. However, the issue of performance becomes one of the major concerns 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 (see, e.g., [33] and the literature therein). 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 [31]. 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 bridge this gap between optimal open-loop scheduling and feedback switching control protocols was proposed in [26,27]. Assuming a pre-determined periodic process that represents the desired open-loop schedule, the objective of the approach is to develop a general technique to design a feedback switching policy that not only gives rise to this periodic process but also makes it globally attractive. The last property is of especial interest if 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 [31,33]. The main concern of this paper is not optimization, but is a stable feedback generation of a desired periodic process. This is similar to the standard hard problem in control engineering, i.e., excitation of the oscillations in a technical device: a feedback controller should make the desired cyclic trajectory globally attracting.

A Lyapunov-type technique of the above kind was proposed in [27]. The resultant controllers are central: every server has access to the entire system state. In certain cases, decentralized controllers driven by only local data are needed. In [26], it was shown, by means of an example, that within the new view of the problem introduced in [27], decentralized controllers can be derived as well. The discussed technique relies on computation of a Lyapunov function and suffers much from the 'curse of dimensionality'.

To break the curse, a computationally non-demanding Poincaré-type technique was reported in [10] in a 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 the 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 notifications to the other servers 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 [10], 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 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 the MO. The above properties are such that being established for the MO, they guarantee the global stability of the equilibrium point of the iteration process and as a result 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 [10], this technique was probed by application to an example, also treated in [26]. It concerns the Kumar–Seidman system [23] 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 systematic presentation of this technique and demonstrates, by means of examples[1], that it fits to handle the entire range of cases encountered in the optimization problem. To the latter end, we start with polling systems, i.e., ones with a single server attending several buffers typically, in a cyclic order. Such systems have a wide range of applications and have been the subject of extensive research; see e.g., [24,37,39] for surveys of this area. Mostly concerned were the performance of specific policies and performance estimates. Only a relatively small body of research dealt with optimal designs under non-zero setup times, with the focus on open-loop schedules and efficient visit orders, where only the latter topic was partly handled with the aid of feedback protocols; see e.g., [4,14–16,24] and the literature therein.

When dealing with polling systems, we mean to highlight the ease of application of the proposed theory, which promises well regarding more complicated networks. Our choice of the desired periodic behavior is based on many studies showing that for a whole variety of performance criteria (e.g., time-averaged wip, maximum wip, throughput, etc.), service at the rates maximal under the current circumstances[2] is required for optimal system performance; see e.g., [9,24]. So we assume that the pre-specified process has this property, with no other assumptions being imposed. A simple distributed feedback switching control law is designed that not only gives

[1] Application of this technique to the general multiple-server fluid networks is the topic of ongoing research. As compared with the examples, this requires essentially more technical developments, which makes the general case not the best arena for the first presentation of the approach.

[2] i.e., at the maximal rate if the served buffer is not empty, and at the input rate otherwise.

rise to the required steady state behavior but also makes it globally attractive, irrespective of the system parameters.

Being of self-importance, polling systems are not interesting enough for illustration of the general theory: for these systems the phase dynamical operators are affine, whereas the general theory deals with non-affine ones. As the simplest example involving non-affine operators, we extend the above results on a generalization of the polling system where the server can simultaneously serve several buffers. This model is of interest for e.g., high speed data and computer networks [40], transport and manufacturing networks [28], etc., with the simplest example being an intersection of two-way roads.

Our third illustration concerns the more complicated two-server re-entrant network introduced by Kumar and Seidman [23] and traditionally employed as a testbed in the area. We first recall the analytical solution to the problem of optimizing the system behavior, as presented in [25]. Next, we design a distributed feedback switching control law that gives rise to the optimal steady state behavior and makes it globally attractive. Compared with [10], this requires new phase control rules, with an emphasis on the concept of the flexible phase. This is a 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 and based on only the local data, thus achieving complete decentralization of control.

The body of the paper is organized as follows. Section 3 presents the proposed general guidelines for switching policies design and the related mathematical background.[3] To make presentation of these specific guidelines, Sect. 2 introduces a rather general model of multi-product multiple server system with setups.[4] Sections 4, 5, and 6 deal with polling systems, single server networks with processor sharing scheme, and the Kumar–Seidman system, respectively. There are three appendices providing the technical facts and their proofs.

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 priory. A setup activity is required to switch a product type at any server.

This system is represented by a directed graph with the set of nodes $\mathfrak{N} := \{1, \dots, N, \circledast_1, \dots, \circledast_F, \circledast_{\text{out}}\}$. Here $\circledast_i$ is the *source* of the ith flow and $\circledast_{\text{out}}$ 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

[3] This material was partly reported in [10].

[4] These guidelines can be extended and the proposed mathematical background can be directly applied to even more general systems.

of buffers. The sources $\circledast_j$ have no incoming arcs, the exterior $\circledast_{\text{out}}$ has no outgoing arcs. There is only one arc starting at $n \neq \circledast_{\text{out}}$; 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 piece-wise 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[5];
2. the function $X(\cdot)$ is absolutely continuous and for any buffer $n \in [1 : N]$,

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

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 buffer should be served next. The problem to be treated in this paper is as follows:

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

For the definition of process convergence, we refer the reader to [35,36]. Briefly, this means asymptotic convergence. Notice that a necessary and sufficient condition for existence of periodic processes is provided in [17,34]: all servers should have enough capacity to process the job inflow.

Given a switching policy, the process is determined by the initial state. So the first requirement means that there exists an initial state that gives rise to π^0. By the second requirement, sooner or later, the system evolution closely follows π^0 irrespective of the initial state. This is of especial interest if π^0 is suboptimal. Then the policy ensures automatic transition to the suboptimal system behavior. The main concern of this paper is not determining optimal system behavior, but is generating stable feedbacks that make all processes converge to a given desired periodic processes. So in what follows, the process π^0 is treated as pre-specified.

[5] 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.

3 Transformation of a periodic process into a switching policy: general guidelines and mathematical background

According to [10], transformation of the periodic process $\pi^0 = [X^0(\cdot), Q^0(\cdot)]$ into a switching policy is arranged along the following lines:

- The periodic process π^0 is partitioned into finitely many *phases*

$$\mathfrak{C} = \mathfrak{P}_0, \mathfrak{P}_1, \mathfrak{P}_2, \ldots, \mathfrak{P}_{c-1}, \tag{1}$$

  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):

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

The problem **(P)** from page 482 is solved whenever PCR's ensure the following:

(i) Any PCR generates the related piece of π^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 π^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 π^0 by the standard argument presented in e.g., [34–36].

To ensure (i), the idea is to design PCR's so that they enforce the system to copycat the desired process π^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.

Definition 1 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) *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 $A_j \in \mathbb{R}^{p\times p}$, $b_j \in \mathbb{R}^p$;
(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 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 Theorem 1 is given in Appendix A.

With respect to the monodromy operator $\mathcal{T} := M$, the assumptions of continuity, monotonicity, and piece-wise affinity can be checked 'phase-wise' since they are evidently inherited by compositions of the maps. As for as 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.

Piece-wise affinity is usually absolutely clear from the formula of the operator. To check the continuity of piece-wise affine operators it is required to establish that the formulas that are active at the different sides of the boundary of any cell produce a common result at any boundary point. Another useful fact is that the composition $f = g \circ h$ of a continuous and piece-wise affine operator g with an affine operator h is continuous and piece-wise affine as well. For example, by taking here $g(y_1, \ldots, y_m) = \max\{y_1, \ldots, y_m\}$, $h(x) = \{\sum_{j=1}^n a_{ij}x_j + b_i\}_{i=1}^m$, $x \in \mathbb{R}^n$, we see that the function $f(x) = \max_{i=1,\ldots,m}\left(\sum_{j=1}^n a_{ij}x_j + b_i\right)$ is continuous and piece-wise affine. Linear combinations of continuous and piece-wise affine functions clearly inherit these properties. Finally, it can be shown that a piece-wise affine continuous operator is monotone if and only if all matrices A_j from (ii) Definition 1 have non-negative entries.

4 Polling systems

We consider a particular case of the system from Sect. 2: only one server, $N \geq 2$ buffers, and N outer flows (see Fig. 1a, where $N = 6$). The nth flow arrives at buffer n at a constant rate $\lambda_n > 0$ and after service at a rate $0 \leq u_n(t) \leq \mu_n$, leaves the system. Switching between buffers requires a given and nonzero setup time.

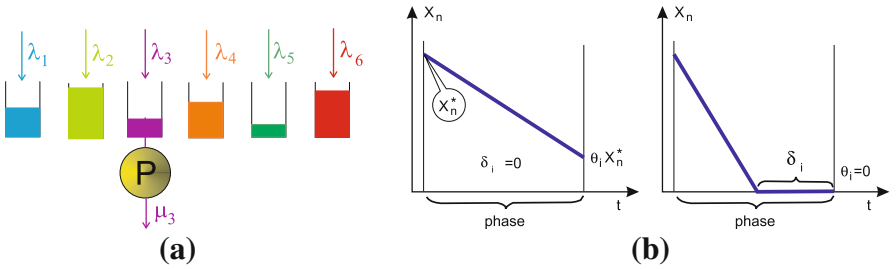

Fig. 1 **a** Polling system, **b** basic quantities underlying the phase control rule

Consider a T-periodic process[6] $\pi^0 = \left[\{x_n^0(t)\}_{n=1}^N, q_1^0(t) \in [1:N] \cup \{\ominus\}\right]$ 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 Sect. 3, we decompose π^0 into the simplest phases (1) by partitioning the period $0 = t_0^0 < t_1^0 < t_2^0 < \cdots < t_{c-1}^0 < t_c^0 = T$ 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

$$q^0 \mapsto q^1 = \ominus \mapsto q^2 \mapsto q^3 = \ominus \mapsto \cdots \mapsto q^{c-2} \mapsto q^{c-1} = \ominus. \tag{3}$$

Now we introduce the phase control rules (PCR).

PCR for the switching phase $q^i = \ominus$. Switching is implemented for a duration of σ_i^0 time units, where $\sigma_i^0 > 0$ is its duration along the process π^0.

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

- θ_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 π^0, i.e.,

$$\theta_i^n := \frac{x_n^0(t_{i+1}^0)}{x_n^0(t_i^0)} \in [0, 1]; \tag{4}$$

- δ_i—the duration of the service at the rate λ_n at this phase for π^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 μ_n until its content reduces to the level $\theta_i^n x_n(t_i)$ and then at the input rate δ_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.

[6] Existence of a periodic process is equivalent to $\sum_{n=1}^N \lambda_n/\mu_n < 1$ [17,34]. In fact, this inequality is the minimal requirement: whenever it is violated, no control policy keeps the total amount of work in the system bounded [17,34].

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

Proof The proposed PCR's trivially meet (i) from Sect. 3. So the entire policy does generate π^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., [34–36], it suffices to show that the assumptions of Theorem 1 are true for M.

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

$$T^{\mathfrak{P}_i} X = \begin{cases} AX + b & \text{if } q^i = n \neq \ominus \\ \sigma_i^0 \cdot \begin{bmatrix} \lambda_1 & \dots & \lambda_N \end{bmatrix}^\top & \text{if } q^i = \ominus, \end{cases}$$

where

$$AX = \begin{bmatrix} x_1 + \lambda_1 \frac{1-\theta_i^n}{\mu_n - \lambda_n} x_n \\ \vdots \\ x_{n-1} + \lambda_{n-1} \frac{1-\theta_i^n}{\mu_n - \lambda_n} x_n \\ \theta_i^n x_n \\ x_{n+1} + \lambda_{n+1} \frac{1-\theta_i^n}{\mu_n - \lambda_n} x_n \\ \vdots \\ x_N + \lambda_N \frac{1-\theta_i^n}{\mu_n - \lambda_n} x_n \end{bmatrix} \quad b = \begin{bmatrix} \lambda_1 \delta_i \\ \vdots \\ \lambda_{n-1} \delta_i \\ 0 \\ \lambda_{n+1} \delta_i \\ \vdots \\ \lambda_N \delta_i \end{bmatrix}.$$

They are clearly affine, continuous, monotone, and dominated. So evidently their composition is 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 piece-wise affine.

5 Single server networks with processor sharing scheme

Now we consider a 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.[7]

[7] Though the system at hand is not a particular case of the network from Sect. 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 rth of them is formed by the rth elements of J_m's, the switch between two elements consumes the time of the switch between the related

Let π^0 be a T-periodic process[8] for which any service of any buffer n starts at the maximal rate μ_n and proceeds at the input rate λ_n, where any of these periods may be of zero duration, i.e., does not occur in effect. Like in Sect. 4, the interest to such processes is inspired by optimization issues.

Like in Sect. 4, it can be assumed that T is the end of a switching period. To transform π^0 into a switching policy, we still decompose π^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

$$m_0 \mapsto \ominus \mapsto m_2 \mapsto \ominus \mapsto \cdots \mapsto m_{c-2} \mapsto \ominus. \tag{5}$$

PCR for the switching phase $\mathfrak{P}_i \sim \ominus$. Switching is implemented for the duration of σ_i^0 time units, where $\sigma_i^0 > 0$ is its duration along π^0.

PCR for the service phase $\mathfrak{P}_i \sim m_i$. To state this rule, we employ the quantity θ_i^n from (4) in Sect. 4. Let δ_i^n denote the duration of the service of buffer n at the input rate at phase $\mathfrak{P}_i$ for process π^0. The PCR is as follows:

(1) Every buffer $n \in J_{m_i}$ is served at the maximal rate μ_n until its content reduces to the level $\theta_i^n x_n(t_i)$, where t_i is the time when $\mathfrak{P}_i$ has 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 λ_n and maintains it no less than δ_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 δ_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 (5) 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 π^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:

$$T^{\mathfrak{P}_i} X = \begin{cases} \{y_n\}_{n=1}^N + \{a_n\}_{n=1}^N \cdot \tau & \text{for } \mathfrak{P}_i \sim m_i \neq \ominus \\ \sigma_i^0 \cdot \left[\lambda_1 \ \ldots \ \lambda_N\right]^\top & \text{for } \mathfrak{P}_i \sim \ominus \end{cases}, \tag{6}$$

Footnote 7 continued
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.

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

where

$$y_n = \begin{cases} \theta_i^n x_n & \text{if } n \in J_{m_i} \\ x_n & \text{otherwise} \end{cases}, \quad a_n = \begin{cases} 0 & \text{if } n \in J_{m_i} \\ \lambda_n & \text{otherwise} \end{cases},$$
$$\tau = \max_{n \in J_{m_i}} \left[\delta_i^n + \frac{(1-\theta_i^n)x_n}{\mu_n - \lambda_n} \right].$$

Note that $T^{\mathfrak{P}_i}$ is a piece-wise affine monotone continuous function of X, whose cells (see Definition 1) are enumerated by $n \in J_{m_i}$ and look as follows: $\delta_i^n + \frac{(1-\theta_i^n)x_n}{\mu_n - \lambda_n} \geq \delta_i^{n'} + \frac{(1-\theta_i^{n'})x_{n'}}{\mu_{n'} - \lambda_{n'}}$ for all $n' \in J_{m_i} \setminus \{n\}$. On this cell, τ is equal to the expression on the left. It follows that the entire dynamical operator $T^{\mathfrak{P}_i}$ is not only piece-wise affine, continuous, and monotone but also dominated. So evidently is the composition M of these operators. It is also strictly dominated since so is the operator $T^{\mathfrak{P}_c - 1}$ (by (5)) that is the last to act in the composition (2). Thus the assumptions of Theorem 1 are satisfied. □

The following lemma addresses existence of a periodic process and shows that such process exists if and only if the system is stabilizable (there is a way to control the system so that the total queue is kept bounded).

Lemma 1 *The following statements are equivalent for the system at hand:*

(i) *There exists a periodic process of the kind that we have considered (i.e., such that any service of any buffer n starts at the maximal rate μ_n and proceeds at the input rate λ_n);*
(ii) *There exists a (not necessarily periodic) process along which the total amount of work in the system remains bounded as time progresses;*
(iii) *The following inequality holds*

$$\sum_{m=1}^{M} \max_{n \in J_m} \frac{\lambda_n}{\mu_n} < 1; \tag{7}$$

(iv) *There exists a cyclic switching policy that generates a unique periodic process, which attracts all other processes.*

An example of such policy is that given by any periodically repeated production cycle (5) equipped with the above PCR's, where the parameters $\theta_i^n \in [0; 1)$, $\delta_i > 0$, $\sigma_{2i+1}^0 \geq \sigma_{2i \to 2i+2}$ are arbitrary chosen, provided that any mode m is encountered in the chain (5) and $\theta_i^n = \theta_i^m \ \forall n \in J_m$.

The proof of this lemma is given in Appendix B. The last requirement from the lemma in fact is not necessary and is imposed to simplify the proof in the face of the paper length limitations.

Since the system at hand generalizes that from Sect. 4, Lemma 1 extends on polling system. For them, (7) shapes into $\sum_{n=1}^{N} \frac{\lambda_n}{\mu_n} < 1$ since modes are associated with buffers $m \sim n$ and every J_m contains only one buffer n.

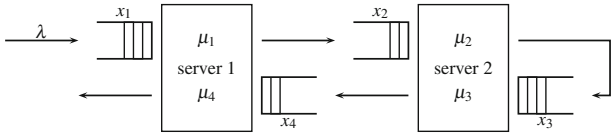

Fig. 2 The Kumar–Seidman model

6 The Kumar–Seidman manufacturing network

This network consists of four buffers and two servers and processes a single job flow, see Fig. 2. Work arrives 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 continuous state $X = \{x_n\}_{n=1}^4$ and the service areas are as follows $I_1 = \{1, 4\}$, $I_2 = \{2, 3\}$.

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 [17]:

$$1 - \rho_1 - \rho_4 > 0, \quad 1 - \rho_2 - \rho_3 > 0, \quad \text{where} \quad \rho_i := \lambda/\mu_i. \tag{8}$$

The model at hand was introduced in [23], also analyzed in [29], to demonstrate that the clearing policy (serve any buffer until emptying) is inappropriate since it may cause instability: even if (8) holds, the total amount of work may explode. Moreover, this inevitably holds whenever

$$\rho_2 + \rho_4 > 1. \tag{9}$$

It is this case that is examined: (8) and (9) are assumed to be true. Then

$$\mu_1 > \mu_2, \quad \mu_3 > \mu_4, \tag{10}$$

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 [25] optimal periodic behavior for this network has been determined with respect to the long-run time-averaged weighted wip (work in progress):

$$W(\pi) := \varlimsup_{T\to\infty} \frac{1}{T} \int_0^T w(t)\, \mathrm{d}t, \quad \text{where} \quad w(t) := \sum_{n=1}^4 c_n x_n(t) \tag{11}$$

under the following technical assumption.

Assumption 1 No downstream buffer values more than an upstream one:

$$c_1 \geq c_2 \geq c_3 \geq c_4 > 0. \tag{12}$$

More precisely, a periodic processes is said to be *simple* if every server processes each of the associated buffers only once during the period. The paper [25] 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. The following theorem summarizes the results of [25].

Theorem 4 *An optimal simple periodic process exists. For this optimal periodic behavior, server 1 repeatedly goes through the following successive phases:*

- *Setup from 4 to 1 for a duration of* $\sigma_{4\to 1}$ ($x_2 = 0$ *upon completion*),
- *Serve 1 at rate* μ_1 *for a duration of* $\frac{\rho_1}{1-\rho_1}(\rho_4 T + \sigma_{1\to 4} + \sigma_{4\to 1})$ ($x_1 = 0$ *upon completion*),
- *Serve 1 at rate* λ_1 *for a duration of* $\frac{1}{1-\rho_1}[(1-\rho_1-\rho_4)T - (\sigma_{1\to 4} + \sigma_{4\to 1})]$,
- *Setup from 1 to 4 for a duration of* $\sigma_{1\to 4}$,
- *Serve 4 at rate* μ_4 *for a duration of* $\rho_4 T$.

and server 2 repeatedly goes through the following successive phases:

- *Setup from 3 to 2 for a duration of* $\sigma_{3\to 2}$,
- *Serve 2 at maximal rate for a duration of* $(1-\rho_3)T - (\sigma_{2\to 3}) + \sigma_{3\to 2})$, *which is either at rate* μ_2 *as long as* $x_2 > 0$ *or at rate* 0 *when* $x_2 = 0$,
- *Setup from 2 to 3 for a duration of* $\sigma_{2\to 3}$ ($x_4 = 0$ *upon completion*),
- *Serve 3 at rate* μ_3 *for a duration of* $\rho_3 T$ ($x_3 = 0$ *upon completion*).

where T *denotes the optimal duration of the period* (*see* [25] *for the explicit expression*). *Furthermore, depending on the parameters* c_i, λ, μ_i, $\sigma_{i\to j}$, *either the setups of duration* $\sigma_{4\to 1}$ *and* $\sigma_{3\to 2}$ *are finished simultaneously, or the setups of duration* $\sigma_{1\to 4}$ *and* $\sigma_{2\to 3}$ *are finished simultaneously.*

As explained in [25], eight different time evolutions of buffer contents result, depending on the above parameters.[9] In this paper, we discuss, for three of these cases, how to arrive at a distributed feedback switching control law that gives rise to the optimal steady state behavior and makes it globally attractive. Since two of the cases can be merged, we call the cases Case 1(a), Case 1(b), and Case 2, respectively.

All other cases can be easily treated along the same lines; their discussion is omitted only to meet the paper length limit.

For the considered cases, the optimal periodic behavior is illustrated by Figs. 3 and 4, respectively. For each of them, the buffers are served at the maximal feasible rates.

In Case 1 from Fig. 3, the optimal behavior consists of periodic repetition of the following successive phases:

P_0 The servers simultaneously start services of buffers 1 and 2; during the phase, buffer 1 is drained out and then served at the input rate;

[9] An explicit description of this dependence is also provided.

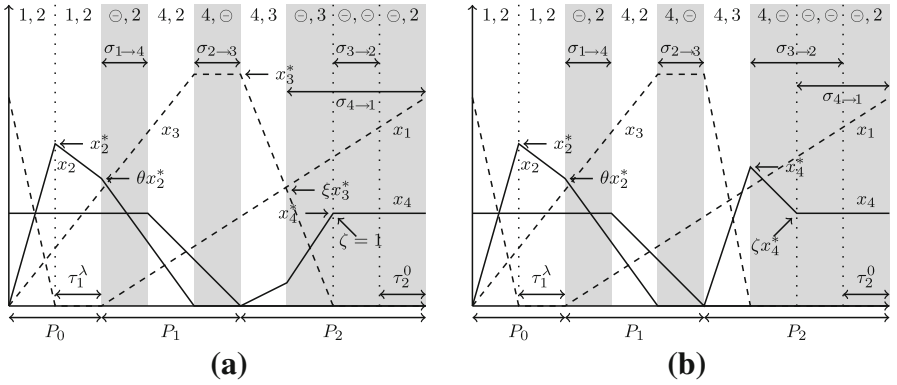

Fig. 3 Case 1 of the optimal periodic behavior

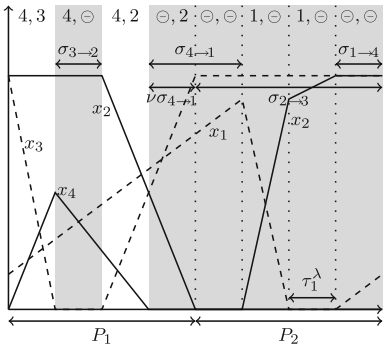

Fig. 4 Case 2 of the optimal periodic behavior

P_1 Server 1 goes to buffer 4 and serves it until emptying; server 2 empties buffer 2 and switches to buffer 3, the switch is completed when buffer 4 is drained out;

P_2 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.

In Case 2 from Fig. 4, the optimal behavior consists in periodic repetition of the following successive phases:

P_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. Server 1 serves buffer 4 until emptying and then begins switching to buffer 1. When this switch is in progress, buffer 2 is emptied, which is the end of the phase;

P_2 After emptying buffer 2, server 2 goes to buffer 3. Server 1 completes the switch to buffer 1, serves it until emptying and then even longer, and finally goes to buffer 4. Switches to buffers 3 and 4 are completed synchronously.

The difference between Fig. 3a and b concerns only the evolution of buffer 4 at phase P_2. 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 the two optimal behaviors can be related to the fact that the problem is reducible to that of optimization of a linear function over a polytope. The solution to this optimization problem may abruptly jump from one vertex to another under the continuous change of the parameters.

To design the switching policy, we also need the following parameters of the optimal process:

Notation 1 τ_2^0: the idle time of server 2 at phase P_2, see Fig. 3 and (13a);
τ_1^λ: the duration of the period when server 1 serves the emptied buffer 1 at the input rate λ at phases P_0 and P_2, see Figs. 3, 4 and (13b);
θ: the fraction of the maximal content of buffer 2 at phase P_0 that remains in this buffer at the phase end, see Fig. 3b and (13c);
ξ: 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 and (13d);
ζ: 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 and (13d);
ν: the percentage of the switching period $\sigma_{4\to1}$ that elapses until buffer 2 is emptied at phase P_2, see Fig. 4 and (13e);
For a given value of T,[10] these parameters are given by:

$$\tau_2^0 = (1-\rho_2-\rho_3)T - (\sigma_{2\to3}+\sigma_{3\to2}), \tag{13a}$$

$$\tau_1^\lambda = (1-\rho_1)^{-1}\left[(1-\rho_1-\rho_4)T - (\sigma_{1\to4}+\sigma_{4\to1})\right], \tag{13b}$$

$$\theta = 1 - \frac{\mu_2-\lambda}{\mu_1-\mu_2}\,\frac{(1-\rho_1-\rho_4)T-(\sigma_{1\to4}+\sigma_{4\to1})}{\rho_1\left[\rho_4 T+(\sigma_{1\to4}+\sigma_{4\to1})\right]}, \tag{13c}$$

$$[\xi;\zeta] = \begin{cases} \left[\mu_3\frac{\sigma_{4\to1}-\sigma_{3\to2}-\tau_2^0}{\lambda T};\,1\right] & \text{in case 1(a)} \\ \left[0;\,1-\mu_4\frac{\tau_2^0+\sigma_{3\to2}-\sigma_{4\to1}}{(\mu_3-\mu_4)\rho_3 T}\right] & \text{in case 1(b)} \end{cases}, \tag{13d}$$

$$\nu = \frac{\sigma_{32}}{\sigma_{4\to1}} + [\rho_2+\rho_3-\rho_4]\frac{T}{\sigma_{4\to1}}. \tag{13e}$$

6.2 Optimal switching policy

Now by following the guidelines from Sect. 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 described in Theorem 4.

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.

[10] An explicit expression of the period T of the optimal process is given in [25].

The partition (1) of the process into phases is merely borrowed from Theorem 4. The phase control rules are designed so that the system behavior copycats that from Fig. 3 or 4.

Switching policy 1 (to be applied in Case 1 illustrated by Fig. 3)

1. Whenever any buffer n is served, the service is at the maximal feasible rate:

$$u_n = \mu_n^{max} := \begin{cases} \mu_n & \text{if } x_n > 0 \\ u_{n-1} & \text{if } x_n = 0 \end{cases}, \quad \text{where } u_0 := \lambda; \tag{14}$$

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

$$\begin{aligned} \to \underbrace{(1,2)}_{P_0} \xrightarrow{(a)} \underbrace{(\ominus, 2) \longrightarrow \left| \begin{array}{c} (4,2) \\ \text{or} \\ (\ominus, \ominus) \end{array} \right| \longrightarrow (4, \ominus)}_{P_1} \to \\ \xrightarrow{(b)} \underbrace{(4,3) \longrightarrow \left| \begin{array}{c} (\ominus, 3) \\ \text{or} \\ (4, \ominus) \end{array} \right| \longrightarrow (\ominus, \ominus) \longrightarrow (\ominus, 2)}_{P_2} \xrightarrow{(c)}; \end{aligned} \tag{15}$$

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 τ is the time when event 3(a) occurs, and θ 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 τ_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 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 for a duration of $\sigma_{4\to 1}$ time units and then possibly 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 Notation 1;

7. Transition (c) is implemented as soon as switching of server 1 is completed and the compulsory idling time τ_2^0 of server 2 is expired.

Remark 1 (i) Formula (15) 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 (15) 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.

Switching policy 2 (to be applied in Case 2 illustrated by Fig. 4)

1. Whenever a buffer is served, the service is at the maximal feasible rate (14);
2. The servers are switched so that the discrete state $Q(t) = [q_1(t), q_2(t)]$ periodically repeats the following cycle:

$$\rightarrow \underbrace{(4,3) \rightarrow (4,\ominus) \rightarrow (4,2) \rightarrow (\ominus,2)}_{P_1} \overset{(a)}{\rightarrow} \underbrace{(\ominus,\ominus) \rightarrow (1,\ominus) \rightarrow (\ominus,\ominus)}_{P_2} \overset{(b)}{\rightarrow}; \tag{16}$$

3. During phase P_1,
   - Server 2 serves buffer 3 until emptying, then switches to buffer 2 for $\sigma_{3\rightarrow 2}$ time units, then serves buffer 2 until emptying, and finally possibly idles, waiting for server 1 to complete its mission for this phase;
   - Server 1 empties buffer 4, then undergoes the first part of switching to buffer 1 for $\nu\sigma_{4\rightarrow 1}$ time units, and finally possibly idles, waiting for emptying buffer 2 by server 2. (Here ν is introduced in Notation 1.)
4. Transition (a) is implemented as soon as buffer 2 is emptied and server 1 completes the required percentage of switching $4 \rightarrow 1$.
5. During phase P_2,
   - Server 1 completes switching $4 \rightarrow 1$ for $(1-\nu)\sigma_{4\rightarrow 1}$ time units, then empties buffer 1, continues to serve it at the input rate τ_1^{λ} time units and possibly even longer so that it leaves buffer 1 no sooner than $\sigma_{2\rightarrow 3} - \sigma_{1\rightarrow 4}$ time units elapses since the phase beginning, and finally switches to buffer 4 for a duration of $\sigma_{1\rightarrow 4}$ time units;

- Server 2 switches from buffer 2 to buffer 3 for $\sigma_{2\to 3}$ time units and then possibly idles, waiting for server 1 to complete its mission.

Here τ_1^{λ} is introduced in Notation 1;

6. Transition (b) is implemented as soon as the switch $1 \to 4$ is completed.

Remark 2 • 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 flexible phases P_1 and P_2.
- For the definiteness, we assume that $Q(0) = (4, 3)$. Then given the initial state $X(0)$, policy 2 uniquely determines a process in the system.

The following theorem shows that the proposed policies ensure asymptotically optimal performance of the closed-loop system.

Theorem 5 *Let the conditions* (8) *and* (10) *hold. Suppose that policy 1 is applied in Case 1 and policy 2 is put in use in Case 2. Then any of these policies 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 described in Theorem* 4.

6.3 Proof of Theorem 5

The proposed phase control rules trivially meet the requirement (i) from Sect. 3. So the entire switching policy generates the periodic process described in Theorem 4 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., [34–36], it suffices to show that the assumptions of this theorem are true for this operator.

Policy 1. The phase dynamical operators are easily computed:

$$\mathcal{T}_{P_0} X = \begin{pmatrix} 0 \\ \theta \frac{\mu_1-\mu_2}{\mu_1-\lambda} x_1 \\ x_3 + \mu_2 \left[\frac{x_1}{\mu_1-\lambda} + \frac{1-\theta}{\mu_2-\lambda} \left(x_2 + \frac{\mu_1-\mu_2}{\mu_1-\lambda} x_1 \right) \right] \\ x_4 \end{pmatrix}$$

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

$$\mathcal{T}_{P_2} X = \begin{pmatrix} x_1 + \lambda \max \left\{ \frac{x_3}{\mu_3} + \sigma_{3\to 2} + \tau_2^0 \, ; \, c + \frac{1-\zeta}{\mu_4} b + \sigma_{4\to 1} \right\} \\ x_2 \\ 0 \\ \zeta b \end{pmatrix},$$

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 piece-wise 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 $\mathcal{T}_{P_0} \circ MX$ contains $+\text{const}(>0)$ at the second and third positions; $\mathcal{T}_{P_1} \circ \mathcal{T}_{P_0} \circ MX$ contains $+\text{const}(>0)$ at the first and third positions; $\mathcal{T}_{P_2} \circ \mathcal{T}_{P_1} \circ \mathcal{T}_{P_0} \circ MX = M^2X$ 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. □

Policy 2. In this case, the phase dynamical operators are as follows:

$$\mathcal{T}_{P1}X = \begin{pmatrix} x_1 + \lambda \max\left\{\frac{x_3}{\mu_3} + \frac{x_2}{\mu_2} + \sigma_{3\to 2}\,;\, \frac{x_3+x_4}{\mu_4} + \nu\sigma_{4\to 1}\right\} \\ 0 \\ x_2 \\ 0 \end{pmatrix},$$

$$\mathcal{T}_{P2} \overset{\text{(b)}}{=} \begin{pmatrix} \lambda\sigma_{1\to 4} \\ x_1 + x_2 + \lambda \max\left\{\sigma_{2\to 3} - \sigma_{1\to 4}\,;\, \tau_1^\lambda + \frac{x_1+\mu_1(1-\nu)\sigma_{4\to 1}}{\mu_1-\lambda}\right\} \\ x_3 \\ x_4 \end{pmatrix},$$

where (b) follows from the computation of the phase duration

$$\tau = \max\left\{\sigma_{2\to 3}\,;\, \tau_1^\lambda + \frac{x_1 + \mu_1(1-\nu)\sigma_{4\to 1}}{\mu_1 - \lambda} + \sigma_{1\to 4}\right\}$$

and noting that since $\lambda\tau$ units of work arrives at buffer 1 during the phase, $x_1 + \lambda\tau - \lambda\sigma_{1\to 4}$ units should be removed to buffer 2 to make the final level of buffer 1 equal to $\lambda\sigma_{1\to 4}$. We see that these operators are piece-wise affine, continuous, monotone, and dominated. So evidently is the monodromy operator $M = \mathcal{T}_{P2} \circ \mathcal{T}_{P1}$. As for the strict dominance, we still examine M^2. Irrespective of the cell, the expression for $\mathcal{T}_{P2}$ contains $+\text{const}(>0)$ at the first and second positions. So $\mathcal{T}_{P1} \circ M$ contains $+\text{const}(>0)$ at the first and third positions; $\mathcal{T}_{P2} \circ \mathcal{T}_{P1} \circ M = M^2$ contains $+\text{const}(>0)$ at the first, second and third positions. The forth position of MX is always zero. So by truncating the state space to $\mathbb{R}^3 = \{\mathbf{col}\,(x_1, x_2, x_3)\}$, we make M^2 strictly dominated. So the assumptions of Theorem 1 are satisfied, which completes the proof. □

Appendiz A: Proof of Theorem 1

Lemma A.2 *Suppose that $\mathcal{T}$ is a continuous piece-wise affine operator. It is dominated if and only if $\mathcal{T}(\theta x) \le \theta\mathcal{T}(x)\ \ \forall\theta \ge 1,\ \ x \ge 0$, and it is strictly dominated, if and only if $\mathcal{T}(\theta x) < \theta T(x)\ \ \forall\theta > 1, x \ge 0$. In the second case, there exists $b > 0$, such that*

$$\mathcal{T}(\theta x) \le \theta \mathcal{T}(x) - (\theta - 1)b \qquad \forall \theta \ge 1,\ x \ge 0. \tag{17}$$

Proof is focused on the case of strictly dominated operator, the case of dominated one is considered likewise. *Necessity.* Let $\mathcal{T}$ be strictly dominated, $x \in K_+^p$, $\theta > 1$. Consider the partition from (ii) of Definition 1. There exists a matching partition $1 = \theta_0 < \theta_1 < \theta_2 < \cdots < \theta_k = \theta$ of $[1, \theta]$ such that $\{z : z = \tau x, \tau \in [\theta_i, \theta_{i+1}]\} \subset S_{j(i)}$ for $i = 0, \ldots, k-1$. Then

$$\begin{aligned}
\mathcal{T}(\theta x) &= \mathcal{T}(\theta_k x) = \mathcal{T}\left(\frac{\theta_k}{\theta_{k-1}}\theta_{k-1}x\right) = A_{j(k-1)}\frac{\theta_k}{\theta_{k-1}}\theta_{k-1}x + b_{j(k-1)} \\
&= \frac{\theta_k}{\theta_{k-1}}\left[A_{j(k-1)}\theta_{k-1}x + b_{j(k-1)}\right] + \left(1 - \frac{\theta_k}{\theta_{k-1}}\right)b_{j(k-1)} = \frac{\theta_k}{\theta_{k-1}}\mathcal{T}(\theta_{k-1}x) \\
&\quad + \left(1 - \frac{\theta_k}{\theta_{k-1}}\right)b_{j(k-1)} \\
&= \frac{\theta_k}{\theta_{k-1}}\mathcal{T}\left(\frac{\theta_{k-1}}{\theta_{k-2}}\theta_{k-2}x\right) + \left(1 - \frac{\theta_k}{\theta_{k-1}}\right)b_{j(k-1)} \\
&= \frac{\theta_k}{\theta_{k-1}}\left\{A_{j(k-2)}\frac{\theta_{k-1}}{\theta_{k-2}}\theta_{k-2}x + b_{j(k-2)}\right\} + \left(1 - \frac{\theta_k}{\theta_{k-1}}\right)b_{j(k-1)} \\
&= \frac{\theta_k}{\theta_{k-1}}\left\{\frac{\theta_{k-1}}{\theta_{k-2}}\left[A_{j(k-2)}\theta_{k-2}x + b_{j(k-2)}\right] + \left(1 - \frac{\theta_{k-1}}{\theta_{k-2}}\right)b_{j(k-2)}\right\} \\
&\quad + \left(1 - \frac{\theta_k}{\theta_{k-1}}\right)b_{j(k-1)} \\
&= \frac{\theta_k}{\theta_{k-2}}\mathcal{T}(\theta_{k-2}x) + \left(\frac{\theta_k}{\theta_{k-1}} - \frac{\theta_k}{\theta_{k-2}}\right)b_{j(k-2)} + \left(1 - \frac{\theta_k}{\theta_{k-1}}\right)b_{j(k-1)}.
\end{aligned}$$

By continuing likewise, we see that:

$$\mathcal{T}(\theta x) = \frac{\theta_k}{\theta_0}\mathcal{T}(\theta_0 x) + \sum_{l=0}^{k-1}\left(\frac{\theta_k}{\theta_{l+1}} - \frac{\theta_k}{\theta_l}\right)b_{j(l)} = \theta\mathcal{T}(x) + \sum_{l=0}^{k-1}\left(\frac{\theta_k}{\theta_{l+1}} - \frac{\theta_k}{\theta_l}\right)b_{j(l)}.$$

Let $b := \min_{1 \le i \le m} b_i > 0$,where min is meant component wise, m is the number from (ii) of Definition 1, and the inequality holds by (iii) of Definition 1. Then,

$$\mathcal{T}(\theta x) \le \theta\mathcal{T}(x) + \sum_{l=0}^{k-1}\left(\frac{\theta_k}{\theta_{l+1}} - \frac{\theta_k}{\theta_l}\right)b = \theta\mathcal{T}(x) + (1-\theta)b \quad \Rightarrow (17).$$

Sufficiency. Suppose that $\mathcal{T}(\theta x) < \theta\mathcal{T}(x)$ for all $\theta > 1$ and $x \ge 0$. Pick a set S_i from (ii) of Definition 1 and its interior point x. Then $\theta x \in S_i$ for $\theta > 1$, $\theta \approx 1$ and so:

$$\theta A_i x + b_i = \mathcal{T}(\theta x) < \theta\mathcal{T}(x) = \theta(A_i x + b_i) \Rightarrow (\theta - 1)b_i > 0 \Rightarrow b_i > 0.$$

□

Lemma A.3 *The continuous piece-wise affine operator $\mathcal{T}$ is monotonous if and only if the entries of the matrix A_j from* (ii) *of Definition 1 are non-negative for $j = 1, \ldots, m$.*

Proof Since necessity is obvious, we focus on sufficiency. Let $0 \le x \le y$. Owing to (ii) of Definition 1, a partition $0 = \theta_0 < \theta_1 < \cdots < \theta_k = 1$ exists such that

$$\{z : z = z(\theta) := (1-\theta)x + \theta y, \theta \in [\theta_i, \theta_{i+1}]\} \subset S_{j(i)} \quad i = 0, \ldots, k-1.$$

Within any set S_j, the operator $\mathcal{T}(x) = A_j x + b_j$ is evidently monotonous. So

$$z(\theta_i) \le z(\theta_{i+1}); z(\theta_i), z(\theta_{i+1}) \in S_{j(i)} \Rightarrow \mathcal{T}[z(\theta_i)] \le \mathcal{T}[z(\theta_{i+1})] \quad \forall i.$$

Hence, $\mathcal{T}(x) = \mathcal{T}[z(0)] = \mathcal{T}[z(\theta_0)] \le \mathcal{T}[z(\theta_k)] = \mathcal{T}[z(1)] = \mathcal{T}(y)$. □

Corollary A.1 *Fixed points $x_* \ge 0$ of continuous piece-wise affine monotonous strictly dominated operators are strictly positive $x_* > 0$.*

Lemma A.4 *Suppose that $\mathcal{T}$ is a continuous monotone operator. Let $\{x_t(a)\}_{t=0}^{\infty}$ denote the trajectory of the iterated system $x_{t+1} = \mathcal{T}(x_t)$ starting at $x_0 = a$. Then*

(i) $a_1 \le a_2 \Rightarrow x_t(a_1) \le x_t(a_2) \ \forall t = 0, 1, 2, \ldots;$
(ii) $\pm[x_1(a) - x_0(a)] \le 0 \Rightarrow \pm[x_{t+1}(a) - x_t(a)] \le 0, t \ge 0;$
(iii) *If $x_t(a) \longrightarrow \tilde{x}$ as $t \to \infty$, then $\mathcal{T}(\tilde{x}) = \tilde{x}$*

Proof (i) and (iii) are obvious; (ii) $\Leftarrow$ (i). □

Lemma A.5 *Let the continuous operator $\mathcal{T}$ be piece-wise affine, monotonous, and strictly dominated, x_* be its fixed point $\mathcal{T}[x_*] = x_*$, and $\theta \ge 1$. Then $x_t(\theta x_*) \to x_*$ as $t \to \infty$.*

Proof By Corollary A.1, $x_* > 0$. In (17), $b \ge \rho x_*$ with some $\rho > 0$, and hence

$$0 \le \mathcal{T}(\tau x_*) \le \tau \mathcal{T}(x_*) - \rho(\tau - 1)x_* = \underbrace{[(1-\rho)\tau + \rho]}_{r(\tau)} x_* \quad \forall \tau \ge 1. \tag{18}$$

Letting $\tau \to \infty$ shows that $\rho \le 1$. So the map $r(\cdot) : \mathbb{R} \to \mathbb{R}$ has the unique fixed point $\tau = 1$,

$$\tau_t \ge 1 \ \forall t, \quad \tau_t \to 1 \quad \text{as } t \to \infty, \quad \text{where } \tau_0 := \theta \text{ and } \tau_{t+1} = r[\tau_t], t = 0, 1, \ldots \tag{19}$$

Now we are going to show that

$$x_* \le x_t[\theta x_*] \le \tau_t x_* \qquad \forall t = 0, 1, \ldots, \tag{20}$$

arguing via induction on t. For $t = 0$, this is obvious. Let (20) be true for some $t \ge 0$. Then $x_{t+1}[\theta x_*] \ge x_*$ by i) of Lemma A.4. Furthermore,

$$x_{t+1}[\theta x_*] = \mathcal{T}\{x_t[\theta x_*]\} \overset{(20)}{\le} \mathcal{T}[\tau_t x_*] \overset{(18)}{\le} r[\tau_t] x_* = \tau_{t+1} x_* \Rightarrow (20) \text{ with } t := t+1 .$$

The proof is completed by (20) and the second relation in (19). □

Corollary A.2 *Under the assumptions of Lemma A.5, $x_t[a] \to x_*$ as $t \to \infty$ if $a \geq x_*$.*

Proof Since $x_* > 0$ by Corollary A.1, $a \leq \theta x_*$ for some $\theta \geq 1$. Then $x_* \leq x_t[a] \leq x_t[\theta x_*]$ $\forall t \geq 0$ thanks to i) of Lemma A.4. The proof is completed by Lemma A.5. □

Corollary A.3 *There is only one fixed point under the assumptions of Lemma A.5.*

Proof Let x'_* and x''_* be fixed points of $\mathcal{T}[\cdot]$. We pick a such that $a \geq x'_*, a \geq x''_*$. By Corollary A.2, x'_* and x''_* are the limits of the common sequence $\{x_t[a]\}$ and so $x'_* = x''_*$. □

Lemma A.6 *Theorem 1 is true if $m = 1$ in this theorem.*

Proof The fixed point is unique by Corollary A.3. Owing to i) and ii) of Lemma A.4,

$$0 \leq x_*, \;\; 0 = x_0(0) \leq x_1(0) \Rightarrow x_t(0) \leq x_{t+1}(0) \leq x_{t+1}(x_*) = x_*,$$

i.e., $\lim_{t\to\infty} x_t(0)$ exists. By iii) of Lemma A.4 and Corollary A.3, this limit is equal to x_*.

For any $c \geq 0$, there exists $a \geq 0$ such that $a \geq x_*, a \geq c$. By invoking i) of Lemma A.4 once more, we see that $x_t[0] \leq x_t[c] \leq x_t[a]$ $\forall t \geq 0$, where $x_t[0] \to x_*$ and $x_t[a] \to x_*$ as $t \to \infty$ by the foregoing and Corollary A.2, respectively. Hence $x_t[a] \to x_*$ as $t \to \infty$. □

Proof of Theorem 1 The iteration $\mathcal{T}^m$ is clearly a piece-wise affine continuous monotone map. Applying Lemma A.6 to this iteration shows that it has a unique fixed point attracting all trajectories. Since any fixed point of $\mathcal{T}[\cdot]$ is a fixed point of the iteration, the fixed point of $\mathcal{T}[\cdot]$ is also unique. For the trajectory $\{x_k\}$ from Theorem 1 and any $s \in [0 : m-1]$, the sequence $y_t := x_{t\cdot m+s}, t = 0, 1, \ldots$ is the trajectory of the iterated system $y_{t+1} = \mathcal{T}^m[y_t]$. Hence $y_t \to x_*$ as $t \to \infty$ by Lemma A.6. Since s is arbitrary, this completes the proof.

Appendix B: Proof of Lemma 1

(i) ⇒ (ii) is trivial.
(ii) ⇒ (iii) For any mode m, we pick a buffer $n = n(m)$ furnishing $\max_{n\in J_m} \frac{\lambda_n}{\mu_n}$ and consider the polling system that results from the system at hand by discarding all buffers in every group J_m except for $n(m)$. By neglecting everything concerned with the discarded buffers, any process in the original system is converted into a process in this polling system. The process from (ii) evidently gives rise to a process along which the total amount of work in the polling system stays bounded as time progresses. This implies that $\sum_{m=1}^{M} \frac{\lambda_{n(m)}}{\mu_{n(m)}} < 1$ [17,34], which is identical to (7).
(iii) ⇒ (iv) Consider any of the policies described in the last claim of the lemma and the related dynamical operators (6) of the phases. We are going to show that the function

$$V(X) := \sum_{m=1}^{M} \max_{n\in J_m} \frac{x_n}{\mu_n}, \qquad X = \{x_n\}_{n=1}^{N} \in \mathbb{R}^N \tag{21}$$

displays a scant Lyapunov-like property under the action of these operators. Indeed, for any active phase $\mathfrak{P}_i \sim m_i \neq \ominus$, we have by (6)

$$V[T^{\mathfrak{P}_i} X] = \sum_{m\neq m_i} \max_{n\in J_m} \frac{x_n+\lambda_n\tau}{\mu_n} + \max_{k\in J_{m_i}} \frac{\theta_i^k x_k}{\mu_k}, \quad \text{where} \quad \tau = \max_{k\in J_{m_i}} \left[\delta_i^k + \frac{(1-\theta_i^k)x_k}{\mu_k-\lambda_k}\right].$$

By invoking that $\theta_i^n = \theta_i^m$ and $\max_n[a_n + b_n] \le \max_n a_n + \max_n b_n$, we conclude that

$$\begin{aligned} V[T^{\mathfrak{P}_i} X] \le \sum_{m\neq m_i} \left[\max_{n\in J_m} \frac{x_n}{\mu_n} + \max_{n\in J_m} \frac{\lambda_n}{\mu_n} \max_{k\in J_{m_i}} \delta_i^k + (1-\theta_i^{m_i}) \max_{n\in J_m} \frac{\lambda_n}{\mu_n} \max_{k\in J_{m_i}} \frac{x_k}{\mu_k-\lambda_k}\right] \\ + \theta_i^{m_i} \max_{k\in J_{m_i}} \frac{x_k}{\mu_k} = V[X] + (1-\theta_i^{m_i}) \left[\sum_{m\neq m_i} \max_{n\in J_m} \frac{\lambda_n}{\mu_n}\right] \max_{k\in J_{m_i}} \frac{x_k}{\mu_k-\lambda_k} \\ + \left(\theta_i^{m_i} - 1\right) \max_{k\in J_{m_i}} \frac{x_k}{\mu_k} + \underbrace{\sum_{m\neq m_i} \max_{n\in J_m} \frac{\lambda_n}{\mu_n} \max_{k\in J_{m_i}} \delta_i^k}_{\varphi_i}. \end{aligned}$$

Now we observe that due to (7)

$$1 - \max_{k\in J_{m_i}} \frac{\lambda_k}{\mu_k} > \sum_{m\neq m_i} \max_{n\in J_m} \frac{\lambda_n}{\mu_n} \Rightarrow 1 - \max_{k\in J_{m_i}} \frac{\lambda_k}{\mu_k} \ge \varkappa^{-1} \sum_{m\neq m_i} \max_{n\in J_m} \frac{\lambda_n}{\mu_n},$$

where $\varkappa \in (0, 1)$ may be chosen independent of i. Hence

$$\begin{aligned} \max_{k\in J_{m_i}} \frac{x_k}{\mu_k-\lambda_k} &= \max_{k\in J_{m_i}} \frac{x_k}{\mu_k} \frac{1}{1-\frac{\lambda_k}{\mu_k}} \le \frac{1}{1-\max_{k\in J_{m_i}} \frac{\lambda_k}{\mu_k}} \max_{k\in J_{m_i}} \frac{x_k}{\mu_k} \\ &\le \varkappa \left(\sum_{m\neq m_i} \max_{n\in J_m} \frac{\lambda_n}{\mu_n}\right)^{-1} \max_{k\in J_{m_i}} \frac{x_k}{\mu_k}. \end{aligned}$$

Overall

$$V[T^{\mathfrak{P}_i} X] \le V[X] - (1-\varkappa)(1-\theta_i^{m_i}) \max_{k\in J_{m_i}} \frac{x_k}{\mu_k} + \varphi_i \le V[X] - \omega \max_{k\in J_{m_i}} \frac{x_k}{\mu_k} + \varphi_i, \tag{22}$$

where $\omega := (1-\varkappa) \min_{i:\mathfrak{P}_i\sim m_i\neq\ominus}(1-\theta_i^{m_i}) > 0$. Hence $V[T^{\mathfrak{P}_i} X]$ does not exceed $V[X]$ plus the constant φ_i that does not depend on X, with this property being evidently

true for any switching phase (with another constant φ_i) since the time of switching is given.

Based on this scant Laypunov-like property, we are going to show that the monodromy operator (2) maps the set $C_r := \{X \in \mathbb{R}^N : 0 \le X, V[X] \le r\}$ into itself provided that $r > 0$ is large enough. To this end, we start with analysis of $X \in C_r$ such that $V[X] = r$. By assumption, any mode m is encountered in the chain (5); let $i(m)$ stand for the index of the first phase in this mode $\mathfrak{P}_i \sim m$. We also enumerate the modes $m_1, \ldots, m_M$ to arrange $i(m)$ in the descending order $i(m_1) > i(m_2) > \ldots > i(m_M)$. Then

$$V[MX] \overset{(2)}{=} V\left[\underbrace{T^{\mathfrak{P}_{c-1}} \ldots T^{\mathfrak{P}_{i(m_1)+1}}}\, T^{\mathfrak{P}_{i(m_1)}} \ldots T^{\mathfrak{P}_0} X\right]$$
$$\le V\left[T^{\mathfrak{P}_{i(m_1)}} \underbrace{T^{\mathfrak{P}_{i(m_1)-1}} \ldots T^{\mathfrak{P}_0} X}_{X^1}\right] + \sum_{j=i(m_1)+1}^{c-1} \varphi_j.$$

Now we employ (22) (where $i := i(m_1)$ and $X := X^1$) and note that since $i(m_1)$ is the first phase in mode m_1, the contents of all buffers $n \in J_{m_1}$ constantly increase duting all previous phases and so $x_n^1 > x_n \ \forall n \in J_{m_1}$. Hence

$$V[MX] \le V\left[\underbrace{T^{\mathfrak{P}_{i(m_1)-1}} \ldots T^{\mathfrak{P}_{i(m_2)+1}}} \ldots T^{\mathfrak{P}_0} X\right] + \sum_{j=i(m_1)}^{c-1} \varphi_j - \omega \max_{k \in J_{m_1}} \frac{x_k}{\mu_k}.$$

By continuing likewise, we establish that

$$V[MX] \le V[X] + \sum_{j=0}^{c-1} \varphi_j - \omega \sum_{\nu=1}^{M} \max_{k \in J_{m_\nu}} \frac{x_k}{\mu_k} \overset{(a)}{=} V[X] + \sum_{j=0}^{c-1} \varphi_j - \omega \sum_{m=1}^{M} \max_{k \in J_m} \frac{x_k}{\mu_k}$$
$$\overset{(21)}{=} V[X] + \sum_{j=0}^{c-1} \varphi_j - \omega V[X] \overset{(b)}{=} (1-\omega) r + \sum_{j=0}^{c-1} \varphi_j.$$

Here (a) holds since $m_1, \ldots, m_M$ is a permutation of $[1 : M]$; (b) is true since $V[X] = r$. So far as $\omega > 0$, the last expression in the displayed formula is less than r provided that r is large enough. Thus $V[X] = r \Rightarrow V[MX] \le r \Rightarrow MX \in C_r$ for such r.

To extend this conclusion on all $X \in C_r$, we note that $X \in C_r \Rightarrow \widehat{X} := r/V[X]\ X \in C_r \& V[\widehat{X}] = r \& X \le \widehat{X}$. The last inequality implies that $MX \le M\widehat{X}$ since the operator M is monotone [11]. So far as the function $V[\cdot]$ is evidently monotone as well, we have $V[MX] \le V[M\widehat{X}] \le r$, where the last inequality is given be the foregoing. Thus $X \in C_r \Rightarrow MX \in C_r$.

Overall we see that the continuous (see footnote 11) operator M maps the convex compact subset $C_r \subset \mathbb{R}^N$ into itself. By Brouwers fixed point theorem [30, p. 117],

[11] Which was established in the proof of Theorem 3.

this operator has a fixed point $X_* = MX_*$. The last equation means that the process that starts at the initial state X_* returns in this state after the entire production cycle is completed; thus it is periodic. The proof is finalized by the arguments from the concluding part of the proof of Theorem 3.

Since the policy employed in this part of the proof clearly generate only processes for which any service of any buffer n starts at the maximal rate μ_n and proceeds at the input rate λ_n, we have also proved that (iii) $\Rightarrow$ (i)

To complete the proof, it is suffices to show that (iv) $\Rightarrow$ (ii). However, this is evident.

References

1. Bai S, Elhafsi M (1994) Optimal feedback control of a manufacturing system with setup changes. In: Proceedings of of the 4th international conference on computer integrated manufacturing and automation technology, pp 191–196
2. Baker KR (1974) Introduction to sequencing and scheduling
3. Banks J, Dai JG (1997) Simulation studies of multiclass queueing networks. IIE Trans 29:213–219
4. Boxma O, Levy H, Weststrate J (1991) Efficient visit frequencies for polling tables: minimization of waiting cost. Queueing Syst 9:133–162
5. Bramson M (2008) Stability of queueing networks. Springer, Berlin
6. Chase C, Serrano J, Ramadge PJ (1993) Periodicity and chaos from switched flow systems: contrasting examples of discretely controlled continuous systems. IEEE Trans Autom Control 38(1):70–83
7. Connolly S, Dallery Y, Gershwin S (1992) A real-time policy for performing setup changes in a manufacturing system. In: Proceedings of the 31st IEEE conference on decision and control, pp 764–770. Tucson, AZ
8. Dai JG, Vande Vate JH (2000) The stability of two-station multitype fluid networks. Oper Res 48:721–744
9. van Eekelen JAWM (2007) Modelling and control of discrete event manufactoring flow lines. Ph.D. thesis, Technical University of Eindhoven, Eindhoven, The Netherlands
10. Feoktistova V, Matveev A (2009) 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, pp 568–573. Moscow, Russia
11. Flieller D, Riedinger P, Louis JP (2006) Computation and stability of limit cycles in hybrid systems. Nonlinear Anal 64:352–367
12. Gamarnik D, Hasenbein JJ (2005) Instability in stochastic and fluid queueing networks. Queueing Syst 15:1652–1690
13. Gaudio MD, Martinelli E, Valigi P (2001) A scheduling problem for two competing queues with finite capacity and non negligible setup times. In: Proceedings of the 40th IEEE conference on decision and control, pp 2355–2360. Orlando, FL
14. Gaujal B, Hordijk A, van der Laan D (2007) On the optimal open-loop policy for deterministic and exponential polling systems. Probab Eng Inf Sci 27(2):157–187
15. Gaujal B, Hyon E (2000) Optimal routing policy in two deterministic queues. Calc Paralleles 13:601–634
16. Gaujal EAB, Hordijk A (2000) Optimal open-loop control of vacations, polling and service assignment. Queueing Syst 36:303–325
17. Gross D, Harris G (1985) Fundamentals of queueing theory. Wiley Series in Probability and Mathematical Statistics
18. Horn C, Ramadge PJ (1997) A topological analysis of a family of dynamical systems with nonstandard chaotic and periodic behavior. Int J Control 67(6):979–1020
19. Hu J, Caramanis M (1992) Near optimal set-up scheduling for flexible manufacturing systems. In: Proceedings of the 3rd international conference on computer integrated manufacturing and automation technology, pp 192–201

20. Humes C (1994) A regulator stabilization technique: Kumar–Seidman revisited. IEEE Trans Autom Control 39(1):191–196
21. Perkins JR, Humes C Jr, Kumar PR (1994) Distributed scheduling of flexible manufacturing systems: stability and performance. IEEE Trans Robotics Autom 10(2):133–141
22. Kelly FP, Williams R (2004) Fluid model for a network operating under a fair bandwidth sharing policy. Ann Appl Probab 14:1055–1083
23. Kumar PR, Seidman TI (1990) Dynamic instabilities and stabilization methods in distributed real-time scheduling of manufacturing systems. IEEE Trans Autom Control 35(3):289–298
24. Lan WM, Olsen T (2006) Multiproduct systems with both setup times and costs: fluid bounds and schedules. Oper Res 54(3):505–525
25. Lefeber E (2011) Optimal performance of multiclass queueing networks with setup times: the Kumar–Seidman network as an example. SE Report, Eindhoven University of Technology, Systems Engineering Group, Department of Mechanical Engineering, Eindhoven, The Netherlands. http://se.wtb.tue.nl/sereports. Mathematical Methods of Operations Research (submitted)
26. Lefeber E, Rooda J (2008) Controller design for flow networks of switched servers with setup times: the Kumar–Seidman case as an illustrative example. Asian J Control 10(1):55–66
27. Lefeber E, Rooda JE (2006) Controller design of switched linear systems with setups. Physica A 363(1)
28. Lefeber E, Lämmer S (2008) Rooda, J Tech. Rep. SE 2008-09, Department of Mechanical Engineering, Eindhoven University of Technology. http://se.wth.tue.nl/sereports
29. Lu SH, Kumar PR (1991) Distributed scheduling based on due dates and buffer priorities. IEEE Trans Autom Control 36(12):1406–1416
30. Munkres JR (1991) Analysis on manifolds. Addison-Wesley Publishing Company, Reading
31. Ouelhadj D, Petrovic S (2008) A survey of dynamic scheduling in manufacturing systems. J Sched 1099–1425
32. Perkins J, Kumar PR (1989) Stable, distributed, real-time scheduling of flexible manufactoring/assembly/disassembly systems. IEEE Trans Autom Control 34(2):139–148
33. Pinedo M (2008) Scheduling theory: algorithms and systems, 3rd edn. Prentice Hall, New York
34. Savkin A (1998) Regularizability of complex switched server queueing networks modelled as hybrid dynamical systems. Syst Control Lett 35:291–299
35. Savkin AV (2000) Cyclic linear differential automata: a simple class of hybrid dynamical systems. Automatica 36(5):727–734
36. Savkin AV (2001) A switched single server system of order n with all its trajectories converging to $(n-1)!$ limit cycles. Automatica 37(2):303–306
37. Takagi H (2000) Analysis and application of polling models. In: Haring G, Lindemann C, Reiser M (eds) Performance evaluation: origins and directions. Lecture Notes in Computer Science, vol 1769. Springer, Berlin, pp 423—442
38. Ushio T, Ueda H, Hirai K (1996) Stabilization of periodic orbits in switched arrival systems with n buffers. In: Proceedings of the 35th IEEE conference on decision and control, pp 1213–1214. Kobe, Japan
39. Wierman A, Winands E, Boxma O (2007) Scheduling in polling systems. Perform Eval 64:1009–1028
40. Zhang Z, Towsley D, Kurose J (1995) Statistical analysis of the generalized processor sharing scheduling discipline. IEEE J Selected Areas Commun 13:1071–1080