Available online at www.sciencedirect.com
SCIENCE DIRECT®
Physica A 363 (2006) 48–61
PHYSICA A
www.elsevier.com/locate/physa

Controller design for switched linear systems with setups

E. Lefeber*, J.E. Rooda

Systems Engineering Group, Department of Mechanical Engineering, Technische Universiteit Eindhoven, PO Box 513, NL-5600 MB, Eindhoven, The Netherlands

Available online 13 February 2006

Abstract

In this paper we consider the control of a complex network of servers through which many types of jobs flow, where we assume that servers require a setup time when changing between types. Such networks can be used to model complex communication, traffic or manufacturing systems. Instead of starting with a given policy for controlling the network and then study the resulting dynamics, we start from a desired steady-state behavior and *derive* a policy which achieves this behavior. By means of an example we illustrate the way to derive a feedback controller from given desired steady-state behavior. Insights from this example can be used to deal with general networks, as illustrated by a more complex network example.

Keywords: Control of networks; Hybrid dynamical systems; Queueing networks; Flexible manufacturing systems

1. Introduction

Consider a network of servers through which different types of jobs flow. One could think of a manufacturing system, i.e., a network of machines through which different types of products flow. An other example would be an urban road network of crossings with traffic lights through which cars flow. A third example would be a network of computers through which different streams of data flow.

In this paper we take a fluid model (ODE) approach, where we assume that each server can only serve one type of job at a time, i.e., no processor sharing, and furthermore that when switching from one type of job to the other, a non-zero setup time is needed. This setup time might depend on the switch, that is, switching from Type 1 to Type 2 might take a different time than switching from Type 2 to Type 1 or from Type 3 to Type 2. Furthermore, we assume that each job type arrives to the network at a constant rate and that for each job type routes are specified a priori. It is allowed that a job visits the same server more than once (a so-called re-entrant system).

The networks we consider might show some unexpected behavior. In Ref. [1] it was shown by simulation that even when each server has enough capacity to serve all jobs, these networks can be unstable in the sense that the total number of jobs in the network explodes as time evolves. Whether this happens depends on the policy used to control the flows through the network. In Ref. [2] it was shown analytically that using a clearing

*Corresponding author.
E-mail addresses: A.A.J.Lefeber@tue.nl (E. Lefeber), J.E.Rooda@tue.nl (J.E. Rooda).

0378-4371/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.physa.2006.01.049

policy (serve the queue you are currently working on until it is empty, then switch to another queue) certain networks become unstable, even for deterministic systems with no setup times.

In Ref. [3] several clearing policies have been introduced, the so-called clear a fraction (CAF) policies. It was shown that these policies are stable for a single server in isolation in a deterministic environment. Furthermore, it was shown that a CAF policy stabilizes a multiserver system, provided the network is acyclic. A network is called acyclic if the servers can be ordered in such a way that jobs can only move from one server to a server higher in the ordering. A network is called non-acyclic if such an ordering is not possible. The example in Ref. [2] shows that a CAF policy does not stabilize a non-acyclic network.

The main reason why CAF policies fail for a non-acyclic network is because they spend too long on serving one type of job. This results in starvation of other servers and therefore a waste of their capacity. Due to this waste the effective capacity of these other servers is not sufficient anymore, resulting in an unstable system. This observation has led to the development of so-called buffer regulators [4,5] or gated policies. The main idea is that each buffer contains a gate, so the buffer is split into two parts (before and after the gate). Instead of switching depending on the total buffer contents, switching is now determined based on the buffer contents after the gate. As a result, a server might now leave a buffer earlier, avoiding long periods of serving one type of job. It has been shown in Ref. [5] that under certain conditions on these regulators the (possibly non-acyclic) network is stabilized. Since non-acyclic networks are only unstable under certain conditions, applying buffer regulators is not always necessary. Needlessly applying buffer regulators results in a larger mean number of jobs in the network, which from a performance point of view is undesired. Furthermore, it is not known whether these policies result in optimal network behavior.

In Ref. [6] a new approach has been developed. First, the minimal period is determined during which the network is able to serve all jobs that arrive during that period. With this period corresponds a trajectory of the system, a periodic orbit, which serves as a basis for the control policy. The proposed policy in essence is a feedforward controller. From the periodic orbit it is known at which time instant a server should be serving a certain job type. In Ref. [6] it has been proposed to have all servers serve the job type they should be serving according to the periodic orbit at that time. In case no jobs of that type are available, the server should stay idle. In Ref. [6] it was shown that this policy guarantees that all trajectories of the closed-loop system are bounded and that for constant average arrival rates the behavior of the network eventually becomes periodic. The focus of Ref. [6] is on achieving regular behavior. If initially the number of jobs in the system is large, regular behavior is achieved rather quickly, but the number of jobs in the system remains large. Also, it is not straightforward to extend the results of Ref. [6] to a setting with stochastic serving times.

In most of the literature, first a policy (or a class of policies) has been proposed, and then the resulting behavior of the network under these policies has been considered (and sometimes optimized). As in Ref. [6], the approach in this paper is the other way around. The desired closed-loop behavior of the system is used as a starting point and then a policy is *derived*. Contrary to the results in Ref. [6] the resulting policy of this paper is a feedback policy, which can straightforwardly be applied in case serving times are stochastic. Furthermore, the feedback policy derived in this paper results in a smaller mean amount of work for the network in steady state.

The remainder of this paper is organized as follows. First, the smallest possible network of servers with setup times is considered, namely a single server which serves two different types of jobs. Starting from the periodic orbit which minimizes the mean amount of jobs in that system, it is shown how a feedback can be derived by means of a candidate Lyapunov function. The presented method can be applied to general networks. To illustrate this, a second example is presented in Section 3. This example consists of the system considered in Ref. [2] and illustrates the strengths and the weaknesses of the method presented in this paper. Finally, Section 4 concludes the paper with some remarks and suggestions for further research.

2. An illustrative example

To make clear how to design a controller for a general network, we first illustrate the basic ideas behind the controller design. Consider the smallest system possible: a single server which serves two different types of jobs, cf. Fig. 1. Assume that jobs arrive to this server at constant rates $\lambda_1 = 3$ and $\lambda_2 = 1$ and can be served at rates $\mu_1 = 8$ and $\mu_2 = 9$, respectively. All mentioned rates are measured in jobs per unit time. Additionally, the

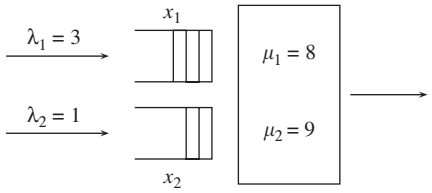

Fig. 1. Single server, two job types.

setup time for switching from Type 1 jobs to Type 2 jobs is assumed to be $\sigma_{1,2} = 3$ time units, whereas the setup time for switching from Type 2 jobs to Type 1 jobs is assumed to be $\sigma_{2,1} = 1$ time unit.

Note that this system has sufficient capacity for serving the arriving jobs, since $\lambda_1/\mu_1 + \lambda_2/\mu_2 = \frac{3}{8} + \frac{1}{9} = \frac{35}{72} < 1$. Also, since doing a setup takes time, it is impossible to stay in a fixed point. We consider the problem of controlling the system towards an arbitrary given periodic orbit. Before we can address the problem of making the system converge to a desired periodic orbit, we need to be precise when addressing the state, the input and the dynamics of this system. Next, we can specify the desired periodic orbit.

2.1. State, input and dynamics

The state consists of the buffer contents of Type 1 jobs, x_1, the buffer contents of Type 2 jobs, x_2, and of the remaining setup time, x_0, which is zero in case the setup has finished and the server is serving. The state is completed by the type of job the server is currently either serving or being setup for, the mode of the server, which is denoted by $m \in \{1, 2\}$.

The server might decide to serve job types at any rate, so the rates $u_1 \leqslant \mu_1$ and $u_2 \leqslant \mu_2$ at which, respectively, Type 1 and Type 2 jobs are being served are inputs. Another input is the required activity of the server, u_0, i.e., what the server should be doing at the moment. The following activities can be distinguished:

$u_0 = ❶$: setup for Type 1 jobs;
$u_0 = ①$: serve Type 1 jobs;
$u_0 = ❷$: setup for Type 2 jobs;
$u_0 = ②$: serve Type 2 jobs.

The input u_0 is a special kind of input, since it can only take a finite number of values. A change in u_0 generates an event. This event might cause jumps in the state variables. First of all, if it has been decided that the server should setup for Type 1 jobs ($u_0 = ❶$) but the server is currently serving of being setup for Type 2 jobs ($m = 2$), then x_0 becomes $\sigma_{2,1}$ (the setup time for switching from Type 2 to Type 1). Similarly, for $m = 1$ and $u_0 = ❷$, x_0 becomes $\sigma_{1,2}$, i.e.,

$$x_0 := \sigma_{2,1} \quad \text{if } u_0 = ❶ \quad \text{and} \quad m = 2,$$

$$x_0 := \sigma_{1,2} \quad \text{if } u_0 = ❷ \quad \text{and} \quad m = 1.$$

Second, we have

$$m := \begin{cases} 1 & \text{if } u_0 \in \{❶, ①\} \quad \text{and} \quad m = 2, \\ 2 & \text{if } u_0 \in \{❷, ②\} \quad \text{and} \quad m = 1, \end{cases} \tag{1a}$$

that is, if the server should be serving or setting up for Type i jobs, the current mode becomes Mode i ($i = 1, 2$).

In addition to this discrete event dynamics, we also have the following dynamics:

$$\dot{x}_0(t) = \begin{cases} -1 & \text{if } u_0 \in \{❶, ❷\}, \\ 0 & \text{if } u_0 \in \{①, ②\}, \end{cases} \tag{1b}$$

$$\dot{x}_1(t) = \lambda_1 - u_1(t), \tag{1c}$$

$$\dot{x}_2(t) = \lambda_2 - u_2(t). \tag{1d}$$

Furthermore, at each time instant the input (u_0, u_1, u_2) is subject to the following constraints:

$$u_0 \in \{❶, ❷\}, \quad u_1 = 0, \ u_2 = 0 \ \text{for} \ x_0 > 0, \tag{2a}$$

$$u_0 \in \{①, ❷\}, \quad 0 \leqslant u_1 \leqslant \mu_1, \ u_2 = 0 \ \text{for} \ x_0 = 0, \ x_1 > 0, \ m = 1, \tag{2b}$$

$$u_0 \in \{①, ❷\}, \quad 0 \leqslant u_1 \leqslant \lambda_1, \ u_2 = 0 \ \text{for} \ x_0 = 0, \ x_1 = 0, \ m = 1, \tag{2c}$$

$$u_0 \in \{❶, ②\}, \quad u_1 = 0, \ 0 \leqslant u_2 \leqslant \mu_2 \ \text{for} \ x_0 = 0, \ x_2 > 0, \ m = 2, \tag{2d}$$

$$u_0 \in \{❶, ②\}, \quad u_1 = 0, \ 0 \leqslant u_2 \leqslant \lambda_2 \ \text{for} \ x_0 = 0, \ x_2 = 0, \ m = 2. \tag{2e}$$

In words, these constraints say that in case the server is setting up, no jobs can be served (2a). In case a setup has been completed, only the job type can be processed for which the server has been setup. This processing takes place at a rate which is at most μ_i if jobs of Type i are available in the buffer, (2b) and (2d), and at a rate of at most λ_i if no jobs of Type i are available in the buffer, (2c) and (2e), ($i \in \{1, 2\}$). Also, it is possible to either stay in the current mode, or to switch to the other mode. In particular, it is possible during setup to leave that setup and start a setup to the other type again. The latter setup is assumed to take the entire setup time.

To summarize, the state of this single server system which serves two different types of jobs is given by $(m, x_0, x_1, x_2) = (m, x_0, x) \in \{1, 2\} \times \mathbb{R}_+ \times \mathbb{R}_+^2$, the input is $(u_0, u_1, u_2) = (u_0, u) \in \{❶, ①, ❷, ②\} \times \mathbb{R}_+^2$ and the dynamics are given by (1a). Furthermore, the input needs to satisfy the constraints (2).

2.2. Desired periodic orbit

Having defined the state, input, and dynamics of the system under consideration, we can now consider the problem of controlling this system. Setups take time, as mentioned before. Therefore, the system cannot be controlled towards a fixed point. In this paper we consider the problem of controlling the system towards an arbitrary given periodic orbit.

For the system depicted in Fig. 1, we are interested in controlling our system towards the periodic orbit which minimizes the mean number of jobs in the system. That is, the orbit which minimizes

$$\frac{1}{T} \int_0^T [x_1(t) + x_2(t)] \, \mathrm{d}t,$$

where T denotes the period of the orbit.

In Ref. [7] this periodic orbit has been determined (cf. Ref. [8]), and the resulting orbit is depicted in Fig. 2. This desired periodic orbit switches between the two modes of the system. The first mode starts in $(m, x_0, x_1, x_2) = (1, 1, 12, 0)$, i.e., the server is setup for serving Type 1 jobs, this setup will (still) take one time unit and 12 jobs of Type 1 are in the buffer and no jobs of Type 2. Using the input $(u_0, u_1, u_2) = (❶, 0, 0)$ for 1 time unit makes that the setup to Type 1 jobs is completed and brings the system in

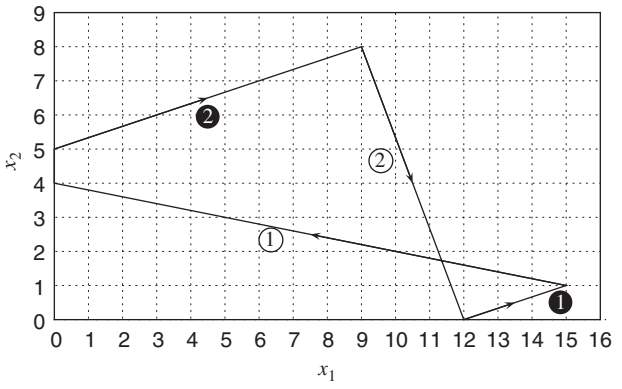

Fig. 2. Desired periodic orbit.

$(m, x_0, x_1, x_2) = (1, 0, 15, 1)$. Next, the system serves Type 1 jobs at full rate for a period of three time units. That is, using the input $(u_0, u_1, u_2) = (①, \mu_1, 0)$ the system moves to $(m, x_0, x_1, x_2) = (1, 0, 0, 4)$. Then, the system remains serving Type 1 jobs for one time unit, but now at the arrival rate (see also Remark 1). So the input $(①, \lambda_1, 0)$ brings the system to $(1, 0, 0, 5)$, which ends Mode 1. Next, for a period of three time units the input $(❷, 0, 0)$ is used. As a result the system first jumps from $(1, 0, 0, 5)$ to $(2, 3, 0, 5)$. After three time units the system is in $(2, 0, 9, 8)$. Then the system starts serving Type 2 jobs at full rate. That is, the input $(②, 0, \mu_2)$ brings the system to $(2, 0, 12, 0)$. This also ends Mode 2. Applying the input $(❶, 0, 0)$ makes the system jump to $(1, 1, 12, 0)$ and the cycle is completed.

During a cycle the amount of jobs of Type 1 increases from 0 to 15 and decreases to 0 again for a duration of 8 time units. Furthermore, it stays zero for a duration of 1 time unit. Therefore, the mean amount of jobs of Type 1 equals $(\frac{1}{2} \cdot 8 \cdot 15 + 1 \cdot 0)/9 = 6\frac{2}{3}$. As the mean amount of jobs of Type 2 equals 4, the mean amount of jobs in the system for the optimal periodic orbit equals $10\frac{2}{3}$.

Remark 1. As shown in Ref. [7] the described period orbit minimizes the mean amount of jobs in the system. At first sight it might seem strange that serving for a certain period at a lower rate can be optimal. As shown above, using the optimal periodic orbit, the mean number of jobs in the systems equals $10\frac{2}{3}$. However, a clearing policy which always serves at full rate yields a mean number of jobs in the system of $10\frac{28}{37}$. This counterintuitive result can be understood from the fact that jobs of Type 2 arrive at a low rate and can be cleared relatively quickly. Since jobs of Type 1 arrive at a rather high rate, it is better to keep on serving Type 1 for a while and make sure no jobs of Type 1 are in the buffer. The fact that the number of jobs of Type 2 increases a little bit more is compensated by enlarging the period of a cycle. As a result on the average the system switches less and the mean number of jobs of Type 1 decreases more than the mean number of jobs of Type 2 increases. For more details on determining the optimal periodic orbit for arbitrary input and serving rates, as well as for using different weights for each job type, the interested reader is referred to Ref. [7].

2.3. Controller design: approach

Given both a correct system description and a desired periodic orbit, the next step is to design a controller which brings our system to this desired periodic orbit. This controller design is based on Lyapunov's direct method, cf. Refs. [9–11]. The basic idea behind Lyapunov's direct method is that if the total energy of a mechanical or electrical system is continuously dissipated, then the system must eventually settle down to an equilibrium point. So if we are able to come up with a kind of "energy-function" for the system under consideration, and if we are able to design our controller such that this energy is decreasing all the time, the system should settle down to a constant energy level and we may conclude stability of our system. This we can do by examining the variation of a single scalar function. The value of this energy function should reflect the magnitude of the state vector, so it can also be interpreted as a kind of measure of the distance between the current state and desired states. Some required properties of this energy function are the following:

- zero energy corresponds to the desired steady-state;
- convergence of energy to zero implies asymptotic stability;
- instability implies growth of energy.

In the remainder of this section we first propose a candidate for this energy function, a so-called Lyapunov function candidate. This candidate is defined for a large subset of the state space. Next, we consider the time derivative of the Lyapunov function candidate along solutions of the system. Clearly, this time derivative depends on the input we use for our system. The input is chosen such that in each point the time derivative of the Lyapunov function candidate is minimized over the set of all allowed inputs, resulting in a non-negative time derivative. Since the minimizing input depends on the current state, we obtain a state feedback. As a next step, since the controller which has been derived by means of the Lyapunov function candidate is valid for a subset of the state space, it needs to be extended to one which is defined for the entire state space. Finally, since a non-negative time derivative of the Lyapunov function candidate in general only guarantees convergence of the Lyapunov function candidate, but not necessarily to zero, it is shown that for the example of this section the derived controller assures convergence to the desired periodic orbit.

2.4. Lyapunov function candidate

Before we construct a Lyapunov function candidate we first associate a number with a periodic orbit, namely the mean amount of work for that periodic orbit. The amount of work associated with a job that is waiting to be served is given by its serving time. So work is measured in time units. The reason for considering the amount of work, is because this always decreases at a constant rate if the server is serving at full rate. This rate is independent of the type of job the server is currently working on. For the example introduced in this section, the server reduces work at a rate of one hour per hour, whereas work arrives at a rate of $\frac{3}{8}+\frac{1}{9}=\frac{35}{72}$.

For the desired periodic orbit, the curve in Fig. 2, we can determine the mean amount of work. The mean amount of jobs of Type 1 is 4, whereas the mean amount of jobs of Type 2 is $6\frac{2}{3}$. A job of Type 1 needs $\frac{1}{8}$ time units of serving, whereas a job of Type 2 needs $\frac{1}{9}$ time units of serving. Therefore, the mean amount of work associated with the desired period orbit depicted in Fig. 2 equals $4\cdot\frac{1}{8}+6\frac{2}{3}\cdot\frac{1}{9}=1\frac{13}{54}$ time units. Similarly, the mean amount of work associated with the solid curve in Fig. 3, which is a translation by $(0,3)$ of the desired periodic orbit, equals $1\frac{13}{54}+0\cdot\frac{1}{8}+3\cdot\frac{1}{9}=1\frac{31}{54}$, and the mean amount of work associated with the dotted curve in Fig. 3 equals $1\frac{13}{54}+15\cdot\frac{1}{8}+0\cdot\frac{1}{9}=3\frac{25}{216}$.

Now we are able to associate a number with a periodic orbit, we can use this as a starting point for proposing a Lyapunov function candidate which can be used in the final controller design. A first observation is that, as illustrated in Fig. 3, by translating the desired periodic orbit into the positive x_1 and/or x_2 direction, we obtain other feasible periodic orbits for the system. As a result, if the system is in a state which can be on a translated periodic orbit, it is possible for the system to stay on that translated periodic orbit. In particular, this means that if the state of the system is in the set of points through which at least one translated periodic orbit goes, it is possible to stay in that set. Since the desired periodic orbit can be characterized as

$$\text{for } m=1:\begin{cases} x_1=15-3x_0 \quad\text{and}\quad x_2=1-x_0 & \text{if } x_0>0,\\ \frac{1}{5}x_1+x_2=4 & \text{if } x_0=0,\end{cases}$$

$$\text{for } m=2:\begin{cases} x_1=9-3x_0 \quad\text{and}\quad x_2=8-x_0 & \text{if } x_0>0,\\ x_1+\frac{3}{8}x_2=12 & \text{if } x_0=0,\end{cases}$$

the set of states through which at least one translated desired periodic orbit is possible can be characterized as

$$\mathscr{D}=\left\{(m,x_0,x_1,x_2)\;\middle|\;\begin{array}{lll} x_1\geqslant 15-3x_0 \quad\text{and}\quad x_2\geqslant 1-x_0 & \text{if } m=1, & x_0>0\\ \frac{1}{5}x_1+x_2\geqslant 4 & \text{if } m=1, & x_0=0\\ x_1\geqslant 9-3x_0 \quad\text{and}\quad x_2\geqslant 8-x_0 & \text{if } m=2, & x_0>0\\ x_1+\frac{3}{8}x_2\geqslant 12 & \text{if } m=2, & x_0=0\end{array}\right\}.$$

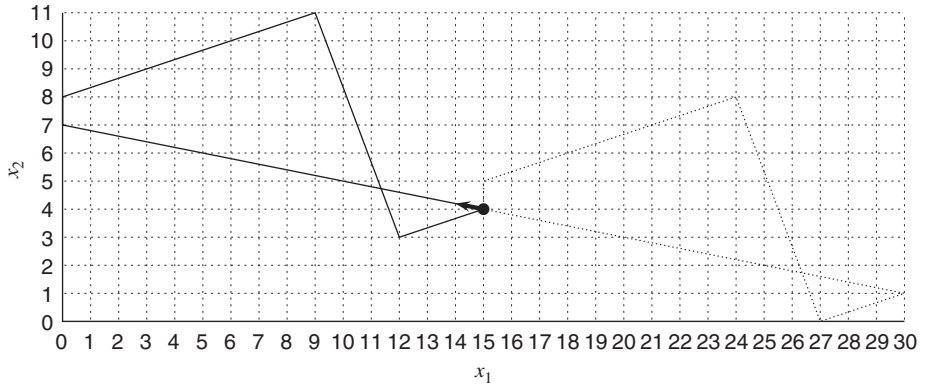

Fig. 3. Translated desired periodic orbits going through $X=(1,0,15,4)$.

For defining the Lyapunov function candidate V we use $\mathscr{D}$ as the domain, i.e., we restrict ourselves to the set $\mathscr{D}$. Note that the set $\mathscr{D}$ covers almost the entire state space. In words we define the Lyapunov function candidate as follows:

Definition 2. For an arbitrary point $X = (m, x_0, x_1, x_2) \in \mathscr{D}$, consider the set of all translated desired periodic orbits going through this point. By definition of $\mathscr{D}$ this set is non-empty. With each of these translated desired periodic orbits we can associate the mean amount of work. Next, we take the smallest of these numbers and subtract $1\frac{13}{54}$ from it (the mean amount of work associated with the desired periodic orbit). This is the number we associate with an arbitrary point $X \in \mathscr{D}$, i.e., this defines $V(X)$.

Note that this definition makes sure that for all $x \in \mathscr{D}$ we have $V(X) \geqslant 0$ and $V(X) = 0$ if and only if X is at the desired periodic orbit. Furthermore, $V(X)$ increases for increasing x_1 or x_2. This relates to the positive definiteness and the radially unboundedness that a Lyapunov function candidate should satisfy, cf. Refs. [9–11].

Given the above-mentioned definition in words of a Lyapunov function candidate V for points $X \in \mathscr{D}$, how does this $V(X)$ look like as an equation? Which of all possible translated desired periodic orbits going through that point has the smallest mean amount of work?

In case the server is performing a setup, i.e., if $x_0 > 0$, only one translated desired periodic orbit is going through this point, so then $V(X)$ is given by the extra amount of work in the system compared to the desired periodic orbit:

$$V = \begin{cases} \frac{1}{8}(x_1 - 15 + 3x_0) + \frac{1}{9}(x_2 - 1 + x_0) & \text{if } m = 1, \ x_0 > 0, \\ \frac{1}{8}(x_1 - 9 + 3x_0) + \frac{1}{9}(x_2 - 8 + x_0) & \text{if } m = 2, \ x_0 > 0. \end{cases} \tag{3}$$

In case the server is not performing a setup, in general more than one translated desired periodic orbit is going through a point. Assume that the system is in the point $X = (1, 0, 15, 4)$, that is: the system is currently in Mode 1, no setup time is remaining (so Type 1 can be served), and the buffer contents for Type 1 and Type 2 are, respectively, 15 and 4. In Fig. 3 this point is marked and two of the infinitely many translated desired periodic orbits going through this point are shown. These curves are two extremes. For the solid curve, the system just started serving Type 1 at full rate, while for the dashed curve the system just finished serving Type 1 at full rate and is about to start serving Type 1 at the arrival rate. Not that any curve where Type 1 has been served at full rate for some time is also a curve that goes through the given point. For the point $X = (1, 0, 15, 4)$ these are the only curves going through it. In case x_2 would have been larger ($\geqslant 5$) all curves where Type 1 has been served at the arrival rate for a while, with as an extreme the case where the system is about to switch to Type 2, would have been allowed too.

So which of all these curves is the one for which the associated mean amount of work is minimal? In order to answer this question it provides insight to consider for each curve the amount of work as a function of time. For the two extreme curves of Fig. 3 the amount of work as a function of time has been shown in Fig. 4, where it is assumed that current time equals 0. From this graph it can be seen that, as mentioned before, the amount of work in the system is decreasing at a constant rate in case the system serves buffers at full rate. The solid curve corresponds to the case where this processing at full rate has just started, whereas the dotted curve corresponds to the case where this processing at full rate is about to end, cf. Fig. 3.

Consider a curve with a certain mean amount of work. For this curve, the amount of work when processing at full rate starts is larger than the amount of work when processing at full rate finishes. Therefore, if a certain amount of work is given and a curve going through this point needs to be found, a curve where processing at full rate starts has a smaller mean amount of work than a curve where processing at full rate is about to finish, as shown in Fig. 4. In this figure it can be seen that a smaller amount of work corresponds to the solid curve than to the dotted curve. This is because the given amount of work corresponding to the point $X = (1, 0, 15, 4)$, which is $\frac{1}{8} \cdot 15 + \frac{1}{9} \cdot 4 = 2\frac{23}{72}$, is a local maximum for the solid curve, whereas it is a local minimum for the dashed curve. Therefore, the solid curve has a lower mean amount of work.

Using this insight the translated desired periodic orbit with the smallest mean amount of work can be associated with a point $X \in \mathscr{D}$. Of all translated desired periodic orbits going through a point, the orbit for

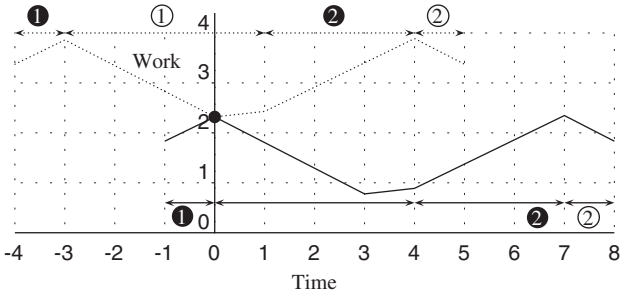

Fig. 4. Amount of work corresponding to translated desired periodic orbits.

which in that point the amount of work is maximal is the orbit with the smallest mean amount of work. To understand this somewhat strange statement consider again Fig. 3. For the solid curve the point $0, 2\frac{23}{72}$ is a local maximum, whereas it is a local minimum for the dashed curve. As a result, the solid curve is the one with the smallest mean amount of work. Since the graph in Fig. 4 describes the evolution of the amount of work at the curves in Fig. 3, the orbit with the smallest mean amount of work in Fig. 3 is also the solid curve.

In most of the states ($m=1$, $x_0=0$, $x_1\geqslant 15$, or $m=2$, $x_0=0$, $x_2\geqslant 8$) the orbit with the smallest mean amount of work would be the one where serving the current type at full rate is about to start. In case $m=1$, $x_0=0$, $x_1\leqslant 15$ it is either the orbit which has started serving Type 1 latest, or the orbit that is about to switch to serving Type 2. For $x_2\leqslant 5$ the latter needs to be after some period of serving Type 1 at the arrival rate. In case $m=2$, $x_0=0$, $x_2\leqslant 8$ this is the orbit which has started serving Type 2 the latest.

As a result, we obtain for all $X\in\mathscr{D}$:

$$V=\begin{cases}\frac{1}{8}(x_1-15+3x_0)+\frac{1}{9}(x_2-1+x_0), & m=1,\ x_0>0,\\ \frac{1}{8}(x_1-15)+\frac{1}{9}(x_2-1), & m=1,\ x_0=0,\ x_1\geqslant 15,\\ \min[\frac{1}{9}(\frac{1}{5}x_1+x_2-4),\frac{1}{8}x_1+\frac{1}{9}(x_2-5)], & m=1,\ x_0=0,\ x_1\leqslant 15,\ x_2\geqslant 5,\\ \min[\frac{1}{9}(\frac{1}{5}x_1+x_2-4),\frac{1}{8}x_1], & m=1,\ x_0=0,\ x_1\leqslant 15,\ x_2\leqslant 5,\\ \frac{1}{8}(x_1-9+3x_0)+\frac{1}{9}(x_2-8+x_0), & m=2,\ x_0>0,\\ \frac{1}{8}(x_1-9)+\frac{1}{9}(x_2-8), & m=2,\ x_0=0,\ x_2\geqslant 8,\\ \frac{1}{8}(x_1+\frac{3}{8}x_2-12), & m=2,\ x_0=0,\ x_2\leqslant 8.\end{cases} \tag{4}$$

Given this Lyapunov function candidate, we can use it for deriving a feedback controller. This is the next step.

2.5. Derivation of controller

Once we have the Lyapunov function candidate as defined in Definition 2 and explicitly written down in (4), we can use it for our controller design. For each point $X\in\mathscr{D}$ we first consider the set of allowed inputs, cf. (2), which assure that the system remains in $\mathscr{D}$. By definition of $\mathscr{D}$ this set is non-empty. Next, we minimize $\dot{V}(X)$ over this set of allowed inputs that assure that the systems remains in $\mathscr{D}$. Note that by definition of $\mathscr{D}$ it is always possible to stay on the translated desired periodic orbit that corresponds to $V(X)$, i.e., the translated desired periodic orbit going through X which minimizes the mean amount of work. The proposed method of constructing our feedback guarantees that $\dot{V}\leqslant 0$. This also implies that buffer contents remain bounded, i.e., stability of the system.

Before deriving the feedback controller we first make two important observations. The first observation is that the periodic orbit going through a point X with $x_0=0$ (i.e., setups have been completed) and minimizing the mean amount of work is never switching in X when $x_m>0$ (i.e., when the buffer that is being served is non-empty). In case $x_m=0$ the periodic orbit which minimizes the mean amount of work is the one that is

switching in X, except for $m=1$, $x_1=0$, $4\leqslant x_2\leqslant 5$, since then the desired periodic orbit determines $V(X)$. A second observation is that serving jobs at a lower than maximal rate never minimizes $\dot{V}(X)$, since making the amount of work decrease at a lower than maximal rate makes $V(X)$ decrease at a lower than maximal rate.

These two observations imply that in case the buffer that is being served is non-empty, the system should continue serving that type at maximal rate. In case the buffer that is being served is empty, we need to switch to the other type, unless $m=1$, $x_1=0$, $4\leqslant x_2\leqslant 5$. In that case we need to serve Type 1 at the arrival rate.

The only thing that remains to be determined is the control action in case the system is doing a setup. Note that from (2) we have two options: either to continue the current setup, or to switch to the other type. This means we have to compare $(1,x_0,x_1,x_2)$ to $(2,3,x_1,x_2)$ and $(2,x_0,x_1,x_2)$ to $(1,1,x_1,x_2)$. From (3) it can be seen that if we are setting up for Type 1, we should continue the setup. If we are setting up for Type 2 we should continue the setup only when $x_0\leqslant\frac{37}{35}$, i.e., the remaining setup time is not more than $\frac{37}{35}$. However, it seems better to switch to Type 1 in case $x_0>\frac{37}{35}$. This is not precisely true, since switching to Type 1 is only allowed if the systems stays in $\mathscr{D}$. This imposes the additional constraint that $x_1>12$ before switching to Type 1 is allowed.

This completes the derivation of the feedback for $X\in\mathscr{D}$. The feedback can be expressed mathematically as follows:

$$(u_0,u_1,u_2)=\begin{cases}(❶,0,0) & \text{if } m=1,\ x_0>0,\\ (①,\mu_1,0) & \text{if } m=1,\ x_0=0,\ x_1>0,\\ (①,\lambda_1,0) & \text{if } m=1,\ x_0=0,\ x_1=0,\ x_2<5,\\ (❷,0,0) & \text{if } m=1,\ x_0=0,\ x_1=0,\ x_2\geqslant 5,\\ (❶,0,0) & \text{if } m=2,\ x_0>\frac{37}{35},\ x_1>12,\\ (❷,0,0) & \text{if } m=2,\ x_0>\frac{37}{35},\ x_1\leqslant 12,\\ (❷,0,0) & \text{if } m=2,\ x_0\leqslant\frac{37}{35},\\ (②,0,\mu_2) & \text{if } m=2,\ x_0=0,\ x_2>0,\\ (❶,0,0) & \text{if } m=2,\ x_0=0,\ x_2=0.\end{cases}$$

In words the feedback can be expressed by means of the following rules:

- If initially the system is setting up for Type 2, but the remaining setup time is more than $\frac{37}{35}$ and the buffer contents of Type 1 is more than 12, switch to Type 1 instead.
- If in Mode 1, serve Type 1 at maximal rate until both $x_1=0$ and $x_2\geqslant 5$. Then switch to Type 2.
- If in Mode 2, serve Type 2 at maximal rate until $x_2=0$. Then switch to Type 1.

The above-mentioned controller has been derived for $X\in\mathscr{D}$. So the final step in our controller design is to extend this controller to a controller which is defined for the entire state space. This can be done rather straightforwardly. In fact, the feedback controller as mentioned in words does not contain any text that cannot be used for $X\notin\mathscr{D}$, so the above-mentioned controller in words is also defined for the entire state space. However, for symmetry reasons (and to match the general case) a small modification is needed. In words, the feedback which is defined for the entire state space becomes:

- If initially the system is setting up for Type 2, but the remaining setup time is more than $\frac{37}{35}$ and the buffer contents of Type 1 is more than 12, switch to Type 1 instead.
- If in Mode 1, serve Type 1 at maximal rate until both $x_1=0$ and $x_2\geqslant 5$. Then switch to Type 2.
- If in Mode 2, serve Type 2 at maximal rate until both $x_2=0$ and $x_1\geqslant 12$. Then switch to Type 1.

In equations this looks like

$$(u_0,u_1,u_2)=\begin{cases}(\textcircled{1},0,0) & \text{if } m=1,\ x_0>0,\\ (①,\mu_1,0) & \text{if } m=1,\ x_0=0,\ x_1>0,\\ (①,\lambda_1,0) & \text{if } m=1,\ x_0=0,\ x_1=0,\ x_2<5,\\ (\textcircled{2},0,0) & \text{if } m=1,\ x_0=0,\ x_1=0,\ x_2\geqslant 5,\\ (\textcircled{1},0,0) & \text{if } m=2,\ x_0>\frac{37}{35},\ x_1>12,\\ (\textcircled{2},0,0) & \text{if } m=2,\ x_0>\frac{37}{35},\ x_1\leqslant 12,\\ (\textcircled{2},0,0) & \text{if } m=2,\ x_0\leqslant\frac{37}{35},\\ (②,0,\mu_2) & \text{if } m=2,\ x_0=0,\ x_2>0,\\ (②,0,\lambda_2) & \text{if } m=2,\ x_0=0,\ x_2=0,\ x_1<12,\\ (\textcircled{1},0,0) & \text{if } m=2,\ x_0=0,\ x_2=0,\ x_1\geqslant 12.\end{cases}\tag{5}$$

In Ref. [7] it has been shown that the controller (5) guarantees that the resulting closed-loop system is stable and that solutions converge to the desired periodic orbit. For sake of completeness that proof is repeated here, for details see Ref. [7].

Let x_2^k denote the value of x_2 when Mode 1 is left for the kth time. Then we have

$$x_2^{k+1}=\max(5,\tfrac{3}{40}(x_2^k-3\tfrac{32}{37})+3\tfrac{32}{37}),$$

which yields

$$\lim_{k\to\infty}x_2^k=\lim_{k\to\infty}\max(5,(\tfrac{3}{40})^k\cdot(x_2^0-3\tfrac{32}{37})+3\tfrac{32}{37})=5,$$

from which it can be concluded that the controller (5) results in a stable closed-loop system and that solutions converge to the desired orbit. For details of the general case (arbitrary constant λ_i and μ_i) the reader is referred to Ref. [7].

2.6. Summary

With this example of the smallest network possible, one server and two job types, we illustrated the basic ideas behind the controller design. The initial feedback controller design is restricted to the set of points through which at least one translated desired period orbit goes, the set $\mathscr{D}$. On this set a Lyapunov function candidate is proposed being the smallest possible mean amount of work that can be associated with a point (relative to that of the desired periodic orbit). Next it turns out that this Lyapunov function candidate is a non-increasing function of time if one tries to keep on serving at the highest possible rate until the decreasing buffer contents hit their level corresponding to that in the desired periodic orbit (namely zero). Finally, the feedback is extended to the entire state space.

3. A second example: the Kumar–Seidman case

The insights from the controller design in the previous section can also be used for the feedback controller design for a general network. Even though the explicit derivation of a Lyapunov function candidate, i.e., deriving (4) might at first sight seem involving, deriving a feedback controller for a general network, given a desired periodic orbit, is relatively easy. To illustrate both the simplicity of the feedback controller design for a general network and the limitations of the derived feedback controller we consider the case as presented by Kumar and Seidman in Ref. [2]

The network consists of 2 servers serving 1 type of job. This type of job arrives at a rate $\lambda=1$ and needs service at consecutively servers 1, 2, 2, and 1. Server 1 serves steps 1 and 4 at rates $\mu_1=\frac{1}{0.3}$ and $\mu_4=\frac{1}{0.6}$, respectively, whereas Server 2 serves steps 2 and 3 at rates $\mu_2=\frac{1}{0.6}$ and $\mu_3=\frac{1}{0.3}$, respectively. All setups take 50 time units (Fig. 5).

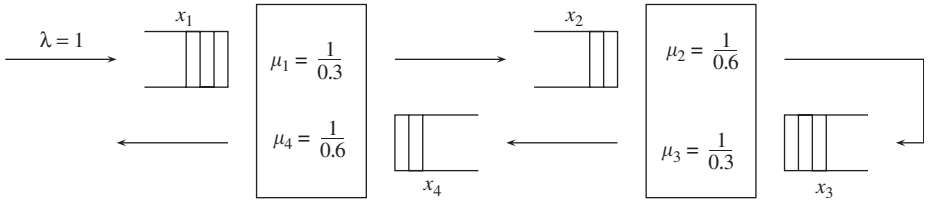

Fig. 5. Kumar–Seidman case.

We consider the problem of designing a feedback based on the desired periodic orbit as depicted in Fig. 6. The desired periodic orbit has a period of 1000 time units. For this periodic orbit Server 1 is in the Mode $m_1 = 1$ from $t = 0$ till $t = 350$ and in the Mode $m_1 = 4$ from $t = 350$ till $t = 1000$. Server 2 is in the Mode $m_2 = 2$ from $t = 0$ till $t = 650$ and in the Mode $m_2 = 3$ from $t = 650$ till $t = 1000$. Similarly, we say that for this periodic orbit the system is in Mode $m = (1, 2)$ from $t = 0$ till $t = 350$, in Mode $m = (4, 2)$ from $t = 350$ till $t = 650$, and in Mode $m = (4, 3)$ from $t = 650$ till $t = 1000$.

From looking at the desired periodic orbit as depicted in Fig. 6 we obtain the following mode descriptions:

Mode $(1, 2)$: Only x_1 and x_2 can decrease, and their values at the end of the mode, are, respectively, 0 and 500. Furthermore, x_2 and x_3 can increase and their values at the end of the mode are both 500.

Mode $(4, 2)$: Only x_2 and x_4 can decrease, and their values at the end of the mode are, respectively, 0 and $83\frac{1}{3}$. Furthermore, x_1 and x_3 can increase and their values at the end of the mode are, respectively, 300 and 1000.

Mode $(4, 3)$: Only x_3 and x_4 can decrease, and their values at the end of the mode are, respectively, 0 and 500. Furthermore, x_1 and x_4 can increase and their values at the end of the mode are, respectively, 650 and 500.

Using the same approach as explained in detail in the previous section, the following feedback results, which closely relates to the above-mentioned descriptions:

- If initially in Mode $(1, 3)$, switch to Mode $(1, 2)$.
- If in Mode $(1, 2)$, serve Type 1 and Type 2 at maximal rate until $x_1 = 0$. Then switch to Mode $(4, 2)$, provided that both $x_2 \geqslant 500$ and $x_3 \geqslant 500$. In case $x_2 < 500$ and $x_3 \geqslant 500$, set Server 2 idle.
- If in Mode $(4, 2)$, serve Type 2 and Type 4 at maximal rate until either $x_2 = 0$ or $x_4 \leqslant 83\frac{1}{3}$. Then switch to Mode $(4, 3)$ provided that both $x_1 \geqslant 300$ and $x_3 \geqslant 1000$ (if possible).
- If in Mode $(4, 3)$, serve Type 3 and Type 4 at maximal rate until $x_3 = 0$. Then switch to Mode $(1, 2)$, provided that both $x_1 \geqslant 650$ and $x_4 \geqslant 500$ (if possible).

Note the "if possible". For $X \notin \mathscr{D}$ we can have in Mode $(4, 2)$ that $x_3 < 1000$ and $x_2 = 0$. In that case we cannot wait for x_3 to reach the value of 1000, so then a switch to Mode $(4, 3)$ is allowed. Similarly for $X \notin \mathscr{D}$ we can have in Mode $(4, 3)$ that $x_1 \geqslant 650$, $x_4 < 500$ and $x_3 = 0$. In that case, a switch to Mode $(1, 2)$ is allowed.

By design we have for this feedback that $\dot{V} \leqslant 0$. Unfortunately, this does not guarantee that V tends to zero (as was the case in the example of Section 2). To make this clear, consider the desired periodic orbit of Fig. 6, where some positive amount, say 1000, is added to the x_4-signal. If we happen to be on that curve, using the above-mentioned feedback makes that we stay on that curve. Assume we start in Mode $(4, 3)$. Since the initial value of x_3 is the same as for the desired periodic orbit, the duration of Mode $(4, 3)$ is the same. Next, since the initial value of x_1 is the same as for the desired periodic orbit, the duration of Mode $(1, 2)$ is the same. Finally, since the initial value of x_2 is the same but the initial value of x_4 is higher, the departure from Mode $(4, 2)$ is determined by the event that x_2 hits zero. Therefore, the duration of Mode $(4, 2)$ is also the same. This shows that using the above-mentioned feedback we might in general converge to a translated desired periodic orbit. So the feedback does result in a stable network and in regular behavior according to the given periodic orbit, but we might converge to a translated periodic orbit. Still, this result is an improvement of the results in Ref. [6], since first of all the derived controller is a feedback, and second it results in a smaller mean amount of work, since for the policy derived in Ref. [6] we actually have $\dot{V} \geqslant 0$.

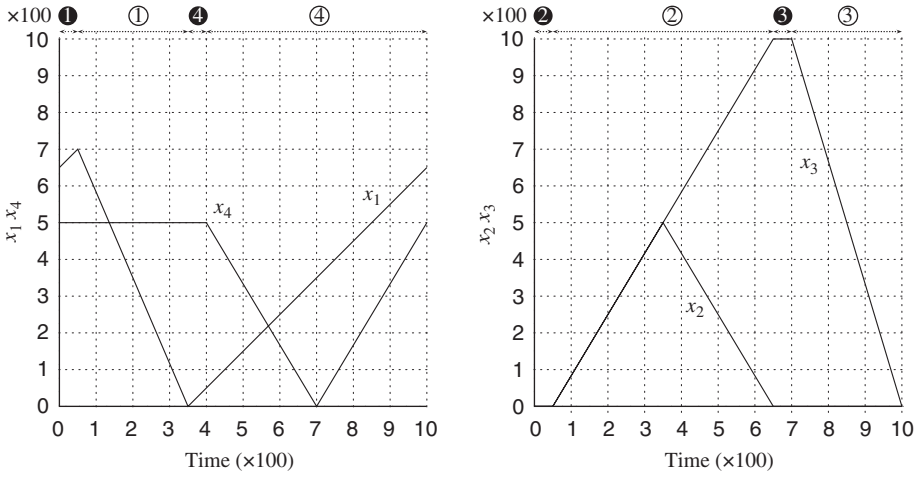

Fig. 6. Desired periodic orbit Kumar–Seidman case.

The derived controller can be improved by using for Mode (4, 2):

- If in Mode (4, 2), serve Type 2 and Type 4 at maximal rate until *both* $x_2 = 0$ *and* $x_4 \leqslant 83\frac{1}{3}$. Then switch to Mode (4, 3) provided that both $x_1 \geqslant 300$ and $x_3 \geqslant 1000$ (if possible). The maximal serving rate for serving Type 2 equals μ_2 as long as $x_2 > 0$ and 0 in case $x_2 = 0$. The maximal serving rate for serving Type 4 equals μ_4 as long as $x_4 > 83\frac{1}{3}$ and 0 in case $x_4 \leqslant 83\frac{1}{3}$.

So the only change is that if only one of the two conditions $x_2 = 0$ and $x_4 \leqslant 83\frac{1}{3}$ is met, the other server idles until both conditions are met. As a result, during that period $V(X)$ is actually *increasing*. Nevertheless, this temporary increase of $V(X)$ makes that later on, $V(X)$ can decrease more.

The statement that this modification leads to a feedback which makes the system converge to the desired periodic orbit can actually be shown to be correct: let $(x_1^k, x_2^k, x_3^k, x_4^k)$ denote the buffer contents when Mode (4, 3) is left for the kth time, while using the modified feedback. Then we have

$$x_1^{k+1} = 100 + \tfrac{3}{7}x_1^k + \tfrac{3}{10}x_2^k + \tfrac{3}{10}x_3^k + \max(\tfrac{3}{7}x_1^k, \tfrac{3}{5}x_2^k + \tfrac{3}{5}x_4^k),$$

$$x_2^{k+1} = 0,$$

$$x_3^{k+1} = 0,$$

$$x_4^{k+1} = \tfrac{5}{7}x_1^k + \tfrac{1}{2}x_2^k + \tfrac{1}{2}x_3^k.$$

So assume that $k > 1$, then we have $x_2^k = x_3^k = 0$. Furthermore, let $x_1^k = 700 + 7y_1^k$ and $x_4^k = 500 + 5y_4^k$. This results in

$$y_1^{k+1} = \tfrac{3}{7}y_1^k + \tfrac{3}{7}\max(y_1^k, y_4^k),$$

$$y_4^{k+1} = y_1^k$$

or

$$\max(|y_1^{k+2}|, |y_4^{k+2}|) \leqslant \tfrac{6}{7}\max(|y_1^k|, |y_4^k|).$$

This shows convergence of (y_1, y_4) to zero, from which we can conclude convergence of (x_1, x_2, x_3, x_4) to $(700, 0, 0, 500)$. The latter implies convergence to the desired periodic orbit.

4. Concluding remarks

This paper deals with the problem of controlling a network consisting of a finite number of servers serving a finite number of job types where each job type requires a finite number of processing steps. We assume finite constant arrival rates and sufficient capacity at the servers. Furthermore, we assume finite strictly positive setup times when servers switch from one job type to the other.

Most literature on this control problem starts from a policy and then analyzes the resulting closed-loop system. In this paper we start from a desired closed-loop behavior and then design a policy which achieves this behavior. The resulting policy is a *feedback* policy. The benefit of a feedback policy is that the closed-loop system is also robust against disturbances. Even though the policy has been derived for a deterministic system, the resulting policy can also be applied in case processing times and setup times are stochastic.

The way to derive such a feedback policy has been illustrated extensively by means of an example. Based on the given desired periodic orbit, an "energy" of the system can be defined by considering the amount of work in the system. By controlling the network in such a way that this "energy" in the system is never increasing, the system stabilizes at a fixed energy level.

Often the resulting policy can be readily obtained from the desired periodic orbit, as illustrated in Section 3. The policy guarantees a stable closed-loop dynamics, as well as behavior according to the desired periodic orbit. However, it might be that we end up on a translated desired periodic orbit. Nevertheless, the resulting steady-state behavior has less work than when we apply the policy proposed in Ref. [6].

We conjecture that a modification of the derived policy similar to the modification as presented in Section 3 also results in convergence to the desired periodic orbit for the general case. Future research consists of proving this conjecture. Most likely a different Lyapunov function candidate is needed, since the candidate proposed in this paper might be increasing when servers serve at a lower rate. We conjecture that the mapping between successive Mode departures (from the same mode) is a contraction, as was shown for the two examples in this paper. Unfortunately, a proof for the general case still needs to be developed.

Though the way to derive a feedback policy has only been presented by means of two examples, the method works for general networks consisting of a finite number of servers and a finite number of processing steps for each job type. Furthermore, constraints on buffer capacities can be easily incorporated, cf. Ref. [7]: the set of translated desired periodic orbits going through an $X \in \mathscr{D}$ becomes smaller. Transportation delays between servers can be included similarly (though the definition of the state deserves more attention in that case).

As a final remark, we note that most literature considers exhaustive policies, gated policies or k-limited policies. Since we did not start from a policy, but *derived* a policy from a given desired periodic orbit, we obtained a different policy. The policy derived in this paper can be summarized as "serve not exhaustively, but until the buffer contents reaches a certain threshold (not necessarily zero). Then either serve at a lower rate, or switch to an other buffer. This decision depends on the contents of other buffers". For manufacturing systems this so-called "global policy" is feasible, maybe also for some urban traffic networks. However, for other networks, e.g. communication networks or computer networks, this global information might not be available. It seems that using the control concept of *observers* might be helpful to overcome some of these disadvantages, resulting in "local policies", but this is also subject of future research.

Furthermore, it remains an open issue if policies similar to the ones derived in this paper might improve on policies so far considered in literature.

Acknowledgements

The authors sincerely thank J. A. W. M. van Eekelen and W. Lefeber-Fikse for a careful reading of this manuscript, as well as the participants of the Thematic Institute on Information and Material Flows in Complex Networks for stimulating discussions regarding the topic, in particular Stephan Lämmer.

References

[1] J. Banks, J.G. Dai, IIE Trans. 29 (1997) 213–219.

[2] P.R. Kumar, T.I. Seidman, IEEE Trans. Automatic Control 35 (3) (1990) 289–298.
[3] J. Perkins, P.R. Kumar, IEEE Trans. Automatic Control 34 (2) (1989) 139–148.
[4] C. Humes Jr., IEEE Trans. Automatic Control 39 (1) (1994) 191–196.
[5] J.R. Perkins, C. Humes Jr., P.R. Kumar, IEEE Trans. Robot. Automatics 10 (2) (1994) 133–141.
[6] A.V. Savkin, Syst. Control Lett. 35 (1998) 291–299.
[7] J.A.W.M. v. Eekelen, E. Lefeber, J.E. Rooda, Feedback control of 2-product server with setups and bounded buffers, in: Proc. 2006 American Control Conference, Minneapolis, Minnesota, USA, 2006.
[8] T.L. Olsen, W.-M. Lan, Multi-product systems with both setup times and costs: fluid bounds and schedules, Oper. Res., in press, 2006, http://iol-a.informs.org/site/OperationsResearch/index.php?c=10&kat=Forthcoming+Papers.
[9] H.K. Khalil, Nonlinear Systems, Prentice-Hall, Upper Saddle River, NJ, 1996.
[10] J.-J.E. Slotine, W. Li, Applied Nonlinear Control, Prentice-Hall, Englewood Cliffs, NJ, 1991.
[11] M. Vidyasagar, Nonlinear Systems Analysis, Prentice-Hall, Englewood Cliffs, NJ, 1993.