Proceedings of the 2007 American Control Conference
Marriott Marquis Hotel at Times Square
New York City, USA, July 11-13, 2007

State feedback control of switching server flowline with setups

J.A.W.M. van Eekelen, E. Lefeber and J.E. Rooda

Abstract— **We consider the control of flowlines consisting of switching servers, through which different types of jobs flow. Switching from one job type to an other takes time. Examples of such flowlines can be found in manufacturing industry, food processing industry, communication networks, and traffic flow.**

The optimal process cycle with respect to work in process levels for a single switching server with two job types is known from previous work. This optimal work in process value is an absolute lower bound on average work in process levels for larger flowlines. In this study we derive conditions for workstations in a flowline that have to be met in order to achieve this minimal work in process level for the whole flowline. Based on these conditions, the class of flowlines is characterized that can behave as if it were a single switching server.

A state feedback controller is proposed that steers a trajectory to the desired trajectories for all servers in the flowline, from any arbitrary start point. Convergence to the desired process cycles is proven mathematically. Although the analysis is performed with a hybrid fluid model, the controller has successfully been implemented in a discrete event case study.

I. INTRODUCTION

Consider a flowline of servers that processes different types of jobs. Only one job type can be processed at a time, and switching from one type to a different type takes time. One can find this kind of systems for example in manufacturing industry, food processing industry, traffic flow or (data) communication systems. In this paper we use a fluid model (ODE) approach, where switching between the job types causes jumps in the state variables. The dynamics are therefore hybrid: both discrete event and continuous dynamics. This hybrid model is used to develop a feedback controller. This controller is required in order to meet customer's demand and constraints from processing capacities.

In a lot of literature, first a control policy is determined and then the behavior of the controlled system (either open loop or closed loop) is studied and sometimes optimized, e.g. see [2], [7]. Clearing policies or threshold services are mostly considered in this area. In [8] a new approach was used. First, the minimal period during which the network is able to serve all jobs during that period is determined. This period corresponds with a specific process cycle. The durations of the different phases were implemented in a controller, actually a feed-forward controller. A disadvantage of this approach is that it can not compensate for disturbances and it does not reduce the number of jobs in the system, if it were initially larger than necessary. In this study however, we first optimize the desired behavior and then propose a controller which makes a trajectory converge to this behavior.

All authors are with the Department of Mechanical Engineering, Systems Engineering Group, Technische Universiteit Eindhoven, P.O.Box 513, 5600 MB Eindhoven, The Netherlands.
[j.a.w.m.v.eekelen,a.a.j.lefeber,j.e.rooda]@tue.nl

In general, the work in process (wip) levels are to be kept low, since both semi-finished jobs and storage space are expensive. But how to keep the wip levels low? We try to find optimal process cycles for flowlines with switching servers. In [3] the optimal process cycle (with respect to wip levels) for a single switching server processing two job types with setup times has been derived. This optimal wip level is an absolute lower bound on the average wip level for flowlines, since any other process cycle for a single machine results in higher wip levels, not to mention the work in process levels of the other workstations of a flowline. But suppose that the servers of a flowline have been chosen in a way that it is actually possible to make the whole flowline behave as if it were a single switching server. Then the minimal wip level for a single switching server becomes the minimal wip level for the whole flowline. In this study we derive the optimal process cycle for a class of flowlines consisting of switching servers that can globally behave as a single switching server. The class of flowlines for which this is possible is characterized. Loosely speaking, it is the class of flowlines for which the most downstream workstation is the 'bottleneck'. Once we know that we have a feasible set of workstations, a state feedback controller can be developed that actually makes the flowline behave as if it were a single server. This controller has to make the trajectories of the system converge to the desired (optimal) behavior for the whole flowline from any arbitrary start point.

The remainder of this paper is organized as follows. First, an example of a flowline consisting of two switching servers is presented. This system is used throughout the whole paper to derive and to explain all notions and insights. In Section II, hybrid fluid model dynamics of the system are presented and we define the desired (optimal) process cycle of the system by looking at the optimal cycle of the most downstream workstation. Next, in Section III, we derive conditions for the upstream workstations to achieve the desired process cycle for the whole flowline. In addition, we characterize the class of flowlines that fulfills these conditions. In Section IV a state feedback controller is proposed which steers a trajectory to the desired trajectory from any arbitrary start point. Convergence to the desired trajectory by means of the feedback controller is proven mathematically. In a discrete event simulation, the feedback controller is implemented to show its proper working. The discrete event character can be regarded as a disturbance to the fluid model and the controller seems to handle this well. Section VI concludes the paper with some remarks and suggestions for further research. We stress that the theory and methods presented in this paper are generally applicable for larger flowlines, with more than two servers. Some remarks regarding this are made throughout the paper.

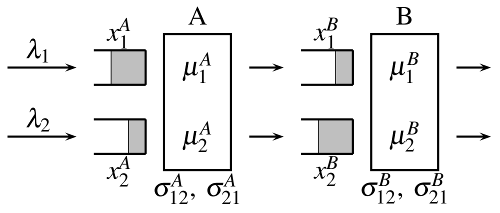

Fig. 1. Switching server flowline overview.

II. FLOWLINE EXAMPLE

Consider the flowline consisting of workstations A and B, each consisting of two parallel buffers and a switching server. The buffers have infinite capacity, store a specific job type and the actual contents are denoted by $x_i^j(t)$, e.g. $x_2^A(t)$ equals the number of jobs of type 2 that is stored in workstation A at time t. Jobs arrive at A with constant rates λ_1 and λ_2 for type 1 and type 2 respectively. The maximum process rate at which machine j processes type i jobs is μ_i^j. Switching from processing type 1 to type 2 jobs takes σ_{12}^j time units and σ_{21}^j vice versa. The system is shown in Fig. 1.

Partial utilizations ρ_i^j are defined as: $\rho_i^j = \frac{\lambda_i}{\mu_i^j}$. For stability reasons, total utilizations must not exceed 1 for each server: $\sum_i \rho_i^j < 1\ \forall\ j \in \{A,B\}$. Unless indicated otherwise, superscript $j \in \{A,B\}$ denotes the workstation number and subscript $i \in \{1,2\}$ represents a job type throughout the remainder of this paper. Furthermore, we refer to this flowline as $A+B$.

A. State, input and dynamics

The state of the system consists of the four buffer levels, x_1^A, x_2^A, x_1^B and x_2^B, the remaining setup times for both servers, x_0^A and x_0^B, and the job type a server is set up for or processing, mode m^A and m^B. The remaining setup times x_0^A and x_0^B equal zero if a server is busy processing jobs. The complete state is defined as: $\mathbf{x} = \begin{bmatrix} x_1^A & x_2^A & x_1^B & x_2^B & x_0^A & x_0^B & m^A & m^B \end{bmatrix}^{\mathrm{T}} \in \mathbb{R}_+^6 \times \{1,2\}^2$. Note that the buffer levels are real-valued, since we use a fluid model approximation. The symbol $\mathbb{R}_+$ denotes a non-negative real number.

Jobs can be processed at any rate smaller than the maximum rate. The rates at which jobs are processed, $u_1^A \le \mu_1^A$, $u_2^A \le \mu_2^A$, $u_1^B \le \mu_1^B$ and $u_2^B \le \mu_2^B$ are inputs of the system. A controller can use these inputs to process jobs at a desired rate. Other inputs are the required activities of the workstations, u_0^A and u_0^B. Possible activities are:

$u_0^j = \text{❶}$:	setup server j for type 1 jobs
$u_0^j = \text{①}$:	server j process type 1 jobs
$u_0^j = \text{❷}$:	setup server j for type 2 jobs
$u_0^j = \text{②}$:	server j process type 2 jobs

The complete input vector of the system is now given by: $\mathbf{u} = \begin{bmatrix} u_0^A & u_0^B & u_1^A & u_2^A & u_1^B & u_2^B \end{bmatrix}^{\mathrm{T}} \in \{\text{❶},\text{①},\text{❷},\text{②}\}^2 \times \mathbb{R}_+^4$. The inputs are constrained by the state at each time instant:

$$\begin{array}{ll}
u_0^A \in \{\text{❶},\text{❷}\},\ u_1^A = 0,\ u_2^A = 0 & \text{for } x_0^A > 0\\
u_0^A \in \{\text{①},\text{❷}\},\ 0 \le u_1^A \le \mu_1^A,\ u_2^A = 0 & \text{for } x_0^A = 0,\ x_1^A > 0,\ m^A = 1\\
u_0^A \in \{\text{①},\text{❷}\},\ 0 \le u_1^A \le \lambda_1,\ u_2^A = 0 & \text{for } x_0^A = 0,\ x_1^A = 0,\ m^A = 1\\
u_0^A \in \{\text{❶},\text{②}\},\ u_1^A = 0,\ 0 \le u_2^A \le \mu_2^A & \text{for } x_0^A = 0,\ x_2^A > 0,\ m^A = 2\\
u_0^A \in \{\text{❶},\text{②}\},\ u_1^A = 0,\ 0 \le u_2^A \le \lambda_2 & \text{for } x_0^A = 0,\ x_2^A = 0,\ m^A = 2\\
u_0^B \in \{\text{❶},\text{❷}\},\ u_1^B = 0,\ u_2^B = 0 & \text{for } x_0^B > 0\\
u_0^B \in \{\text{①},\text{❷}\},\ 0 \le u_1^B \le \mu_1^B,\ u_2^B = 0 & \text{for } x_0^B = 0,\ x_1^B > 0,\ m^B = 1\\
u_0^B \in \{\text{❶},\text{②}\},\ u_1^B = 0,\ 0 \le u_2^B \le \mu_2^B & \text{for } x_0^B = 0,\ x_2^B > 0,\ m^B = 2\\
u_0^B \in \{\text{①},\text{❷}\},\ 0 \le u_1^B \le \min(u_1^A,\mu_1^B), u_2^B = 0 & \text{for } x_0^B = x_1^B = 0,\ m^B = 1\\
u_0^B \in \{\text{❶},\text{②}\},\ u_1^B = 0, 0 \le u_2^B \le \min(u_2^A,\mu_2^B) & \text{for } x_0^B = x_2^B = 0,\ m^B = 2
\end{array}$$

In words, these constraints mean that if a server is busy with a setup, no jobs can be processed. Moreover, after a setup to a job type has been completed, only jobs of that specific type can be processed. Finally, it is always possible to stay in the current mode, or switch to the other mode.

Inputs u_0^A and u_0^B generate events in the system, causing jumps in the state variables. The dynamics of the system are therefore hybrid. The jumps in state variables that can take place are:

$$x_0^j := \sigma_{21}^j,\ m^j := 1 \text{ for } u_0^j = \text{❶} \text{ and } m^j = 2 \tag{1}$$

$$x_0^j := \sigma_{12}^j,\ m^j := 2 \text{ for } u_0^j = \text{❷} \text{ and } m^j = 1 \tag{2}$$

The following dynamics also apply for the system:

$$\dot{x}_0^j(t) = \begin{cases} -1 & \text{for } u_0^j(t) \in \{\text{❶},\text{❷}\} \\ 0 & \text{for } u_0^j(t) \in \{\text{①},\text{②}\} \end{cases} \tag{3}$$

$$\dot{x}_i^A(t) = \lambda_i - u_i^A(t) \tag{4}$$

$$\dot{x}_i^B(t) = u_i^A(t) - u_i^B(t) \tag{5}$$

B. Desired periodic orbit for most downstream workstation

Our goal is to minimize the time averaged weighted wip level of the flowline. The cost function J is defined as:

$$J = \lim_{t \to \infty} \frac{1}{t} \int_0^t c_1(x_1^A(s) + x_1^B(s)) + c_2(x_2^A(s) + x_2^B(s))\mathrm{d}s \tag{6}$$

with c_1 and c_2 weighing factors for type 1 and type 2 jobs respectively.

The process cycle for a single switching server that minimizes the weighted wip level is known from [3]. This minimum wip level for a single server is the absolute lower bound for the wip level of a flowline of switching servers, since any other process cycle than the optimal cycle yields more wip, not to mention the additional wip in the other workstations. However, if this absolute lower bound on the wip for one single server can be achieved for the whole flowline, the lower bound of the isolated workstation becomes the actual wip level for the whole flowline. Therefore in this study, we try to characterize the (sub)class of flowlines that can behave as if they were single switching servers, with respect to wip levels. Then the most downstream workstation is the bottleneck that determines both the throughput, flow time and wip-level of jobs in the system. A schematic impression of this idea is shown in Fig. 2. The buffer lengths of a specific job type are virtually summed up and for these lumped buffer levels, we try to make them behave as the optimal process cycle for a single switching server.

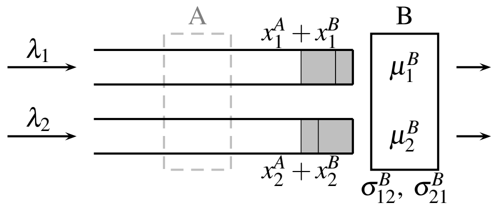

Fig. 2. General idea of flowline behaving as single switching server.

In [3] it is shown that for a single server, optimal work-in-process levels are reached if the server works with a fixed process cycle. If the most downstream workstation works at this cycle (as shown in Fig. 2) and the other workstations can make this final workstation do so, a minimal work in process level for the total flowline has been achieved.

We want to minimize the time averaged costs with respect to buffer levels. In [5] a lower bound on the costs for switching servers with both setup times and setup costs has been defined. In [3] it has been shown that this lower bound can actually be achieved in case of two job types.

In general the optimal process cycle for a single switching server with two job types and both setup times and setup costs looks as shown in Fig. 3 (see also [3]). In the left-hand side graph, x_1 and x_2 have been plotted against each other (the *periodic orbit*). The right-hand side graphs show the buffer levels over time, with the slopes of the lines annotated to them. During τ_i^μ, job type i is processed at maximum rate μ_i, while during τ_i^λ job type i is processed at its incoming rate. The buffer stays empty during τ_i^λ. This so-called *slow-mode* may seem counterintuitive because it means that capacity is lost due to processing at lower rates than the maximum rate. An explanation for this is that in fact there is a trade-off between losing capacity due to processing at lower rates and losing capacity due to relatively often switching in time. Setup costs also contribute to this trade-off. Conditions for the occurrence of slow-modes are given in [3] for situations with linear costs on buffer levels and without setup costs. In that case, one of the slow-modes does not exist ($\tau^\lambda = 0$). Two slow-modes may occur in cases where setup costs or non-linear costs on buffer levels are involved.

Now we know the shape of the optimal process cycle of the most downstream workstation. Given such optimal cycle, we can investigate when an upstream workstation can make the whole line behave as if it were only this final workstation. In the next section, this desired behavior of upstream workstations is analyzed.

III. PERIODIC ORBIT OF UPSTREAM WORKSTATIONS

Given the periodic orbit of the most downstream workstation, as presented in Section II-B, a feasible periodic orbit for the upstream workstation has to be found, in order to make the whole flowline behave like the final server stand-alone, with respect to wip levels. In this section, the properties and conditions for a feasible periodic orbit for

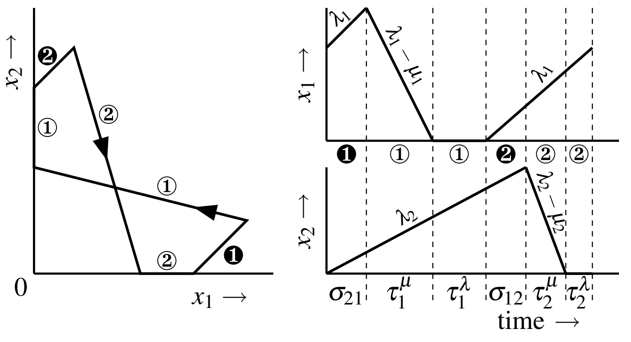

Fig. 3. General form optimal process cycle for a switching server processing two job types, with setup times and setup costs. Left: periodic orbit. Right: buffer levels over time, with slopes of the lines.

upstream workstations are explained for the flowline example of Section II. Extensions to larger flowlines (with more than two workstations) are discussed where applicable.

The optimal process cycle of workstation B must become the optimal cycle for $A+B$. We assume that the period length T of the periodic orbit of workstation B equals the length of one process cycle in A. In Fig. 4 time lines for the process cycles of A and B are shown. If $A+B$ has to behave like B stand-alone, some observations can be made:

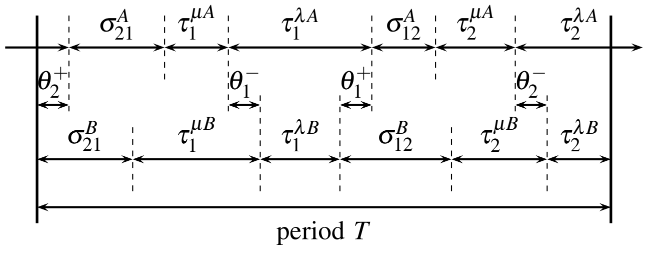

Fig. 4. The period of one cycle, T, divided into subsequent phases.

1) If the buffer level x_i^B of job type i is 0, then x_i^A must be 0 as well. Consequently, slow-modes in A should completely overlap slow-modes in B, if occurring. Define θ_i^- and θ_i^+ as the amount of time a slow-mode of job type i in A starts earlier and ends later (respectively) than the corresponding slow-mode in B, see Fig. 4. The overlap requirement yields:

$$\theta_1^- \geq 0;\ \ \theta_1^+ \geq 0;\ \ \theta_2^- \geq 0;\ \ \theta_2^+ \geq 0. \tag{7}$$

2) From observation 1 follows that at $t=0$, A starts with a setup (if $\theta_2^+ = 0$) or is still in slow-mode of type 2 jobs (if $\theta_2^+ > 0$). Therefore, σ_{21}^A starts at $t \geq 0$. Similarly, σ_{12}^A can not start earlier than the start of σ_{12}^B
3) From observations 1–2 follows:

$$\theta_2^+ + \sigma_{21}^A + \tau_1^{\mu A} + \theta_1^- = \sigma_{21}^B + \tau_1^{\mu B} \tag{8}$$

$$\theta_1^+ + \sigma_{12}^A + \tau_2^{\mu A} + \theta_2^- = \sigma_{12}^B + \tau_2^{\mu B}. \tag{9}$$

4) Buffer levels are not allowed to become negative. Therefore, if $\tau_i^{\mu B}$ starts earlier than $\tau_i^{\mu A}$ (cf. $\tau_1^{\mu B}$ and $\tau_1^{\mu A}$

in Fig. 4), the number of jobs B processes before $\tau_i^{\mu A}$ starts may not exceed the number of jobs A processes (in slow-mode) after B switched to the other mode:

$$\mu_i^B(\tau_i^{\mu B} - \theta_i^- - \tau_i^{\mu A}) \leq \lambda_i \theta_i^+ \tag{10}$$

or written differently:

$$\sigma_{21}^A + \theta_2^+ - \sigma_{21}^B \leq \rho_1^B \theta_1^+ \tag{11}$$
$$\sigma_{12}^A + \theta_1^+ - \sigma_{12}^B \leq \rho_2^B \theta_2^+ \tag{12}$$

This restriction is also valid if $\tau_i^{\mu B}$ starts after $\tau_i^{\mu A}$ started, since then the left-hand sides of (10)–(12) become negative, while the right-hand sides are positive.

5) The amount of jobs A processes of each type during one cycle must be equal to the number of jobs that is processed by B in one cycle. These mass conservation equations follow ($i \in \{1,2\}$):

$$\mu_i^A \tau_i^{\mu A} + \lambda_i \tau_i^{\lambda A} = \mu_i^B \tau_i^{\mu B} + \lambda_i \tau_i^{\lambda B} = \lambda_i T. \tag{13}$$

With these observations it is possible to derive conditions for server A which must be obeyed to make $A+B$ behave like B stand-alone. Note that observations 1–5 are also applicable for flowlines with more than two servers, e.g. flowline $A+B+C$, where firstly, B has to make $B+C$ behave like C stand-alone and secondly, find a feasible trajectory for A to make $A+B$ behave like B stand-alone.

Proposition 3.1: Workstation A can make flowline $A+B$ perform like B stand-alone with respect to work-in-process levels if and only if:

$$R_2\left[\tau_1^{\mu B} + \tau_2^{\lambda B} + \sigma_{21}^B - \sigma_{21}^A - T + R_1(\tau_1^{\lambda B} - T)\right] + \tau_2^{\mu B} + \sigma_{12}^B - \sigma_{12}^A \geq 0 \tag{14}$$

and

$$R_1\left[\tau_2^{\mu B} + \tau_1^{\lambda B} + \sigma_{12}^B - \sigma_{12}^A - T + R_2(\tau_2^{\lambda B} - T)\right] + \tau_1^{\mu B} + \sigma_{21}^B - \sigma_{21}^A \geq 0 \tag{15}$$

with $R_1 = \max(\rho_1^A, \rho_1^B)$ and $R_2 = \max(\rho_2^A, \rho_2^B)$.

Proof: The proof goes in two parts. First, if a periodic orbit of A makes $A+B$ behave like B stand-alone, it fulfills (7)–(13). During $\tau_i^{\mu A} + \theta_i^-$, A has to process $\mu_i^B \tau_i^{\mu B} - \lambda_i \theta_i^+ = \lambda_i(T - \theta_i^+ - \tau_i^{\lambda B})$ jobs. This takes at least $\lambda_i(T - \theta_i^+ - \tau_i^{\lambda B})/\mu_i^A$ time units (if $\theta_i^- = 0$). This gives:

$$\tau_i^{\mu A} + \theta_i^- \geq \rho_i^A(T - \theta_i^+ - \tau_i^{\lambda B}). \tag{16}$$

Note that the mass conservation requirement has been translated into inequality constraints now. Combining this result with (8) and (9) results in:

$$\sigma_{21}^B - \sigma_{21}^A + \tau_1^{\mu B} - \theta_2^+ \geq \rho_1^A(T - \theta_1^+ - \tau_1^{\lambda B}) \tag{17}$$
$$\rho_1^A(\theta_1^+ + \tau_1^{\lambda B} - T) + \tau_1^{\mu B} - \theta_2^+ \geq \sigma_{21}^A - \sigma_{21}^B \tag{18}$$

and similar for the other job type:

$$\rho_2^A(\theta_2^+ + \tau_2^{\lambda B} - T) + \tau_2^{\mu B} - \theta_1^+ \geq \sigma_{12}^A - \sigma_{12}^B. \tag{19}$$

From (13) we know that:

$$\tau_i^{\mu B} - \rho_i^B(T - \tau_i^{\lambda B}) = 0. \tag{20}$$

Adding this up with (11) and (12) results in:

$$\rho_1^B(\theta_1^+ + \tau_1^{\lambda B} - T) + \tau_1^{\mu B} - \theta_2^+ \geq \sigma_{21}^A - \sigma_{21}^B \tag{21}$$
$$\rho_2^B(\theta_2^+ + \tau_2^{\lambda B} - T) + \tau_2^{\mu B} - \theta_1^+ \geq \sigma_{12}^A - \sigma_{12}^B \tag{22}$$

which looks very similar to (18) and (19). The results can be combined:

$$\max(\rho_1^A, \rho_1^B)(\theta_1^+ + \tau_1^{\lambda B} - T) + \tau_1^{\mu B} - \theta_2^+ \geq \sigma_{21}^A - \sigma_{21}^B \tag{23}$$
$$\max(\rho_2^A, \rho_2^B)(\theta_2^+ + \tau_2^{\lambda B} - T) + \tau_2^{\mu B} - \theta_1^+ \geq \sigma_{12}^A - \sigma_{12}^B. \tag{24}$$

These inequalities, together with $\theta_1^+ \geq 0$ and $\theta_2^+ \geq 0$, enclose a feasible area in the (θ_2^+, θ_1^+)-plane. The intersection point of the two linear borders defined by (23) and (24) lies in the pos-pos quarter of this plane. The intersection point is given by:

$$\theta_1^+ = \frac{R_2\left[\tau_1^{\mu B} + \tau_2^{\lambda B} + \sigma_{21}^B - \sigma_{21}^A - T + R_1(\tau_1^{\lambda B} - T)\right] + \tau_2^{\mu B} + \sigma_{12}^B - \sigma_{12}^A}{1 - R_1 \cdot R_2} \tag{25}$$

$$\theta_2^+ = \frac{R_1\left[\tau_2^{\mu B} + \tau_1^{\lambda B} + \sigma_{12}^B - \sigma_{12}^A - T + R_2(\tau_2^{\lambda B} - T)\right] + \tau_1^{\mu B} + \sigma_{21}^B - \sigma_{21}^A}{1 - R_1 \cdot R_2} \tag{26}$$

which are only positive if (14) and (15) are fulfilled.

For the second part of the proof, we know that (14) and (15) are fulfilled and we want to find a feasible periodic orbit for A. Because of the lengthy (but straightforward) proof, the reader is referred to [4, Proposition 8.2]. ■

In case of more than one upstream workstation, similar conditions can be derived for these workstations. Each additional workstation adds two constraints similar to (14) and (15) to Proposition 3.1.

In order to find a feasible trajectory for upstream servers given the parameters (μ and σ) the problem can be casted into a linear program (LP) with design variables $\tau_1^{\mu A}$, $\tau_1^{\lambda A}$, $\tau_2^{\mu A}$, $\tau_2^{\lambda A}$, θ_1^+ and θ_2^+. All constraints (7)–(13) are linear in the design variables. Additional workstations can be put in the same LP. Any arbitrary objective function will result in a feasible solution, (unless infeasible according to (14) and (15)). In this way, for larger flowlines, feasible trajectories can be found fairly easy. Also, the LP solver can be used to check if a feasible solution exists at all.

Now that the desired (optimal) process cycle has been defined for the whole flowline and conditions for the upstream workstations have been derived, we look for a controller that steers the state $\mathbf{x}(t)$ to the desired periodic orbits from any arbitrary initial state $\mathbf{x}(0)$.

IV. CONTROLLER

A state feedback controller that brings any arbitrary trajectory to the desired periodic orbits as defined in sections II-B and III of this paper is presented in this section. The controller can be obtained using the theory and method presented in [6], based on Lyapunov's direct method.

Lemma 4.1: Under conditions (14) and (15), the following inequality holds: $R_1 + R_2 < 1$.

Proof: See [4, Lemma 9.1]. ■

Proposition 4.2: The following state feedback controller steers the system to the desired (optimal) periodic orbits, from any arbitrary initial state $\mathbf{x}(0)$: (due to limited space, an 'informal' state feedback law has been presented, instead of a formal specification)

- If at $t = 0$ the modes of the machines are unequal in the initial state, then make A switch to the same mode as B, according to (1) and (2):

$$m^A = 1 \wedge m^B = 2 \rightarrow u_0^A := ❷ \rightarrow x_0^A := \sigma_{12}^A;\ m^A := 2 \quad (27)$$
$$m^A = 2 \wedge m^B = 1 \rightarrow u_0^A := ❶ \rightarrow x_0^A := \sigma_{21}^A;\ m^A := 1 \quad (28)$$

- After initial switching (if necessary), the controller loops the following lines from top to bottom. Based on the state of the system, the controller (trivially) starts in one of the lines for each server.

Workstation A	Workstation B
① until $x_1^A = x_1^B = 0$	① until $x_1^B = x_1^A = 0$
① until $x_1^B \geq x_1^{B\natural}$ and $m^B = 2$	① until $x_2^A \geq x_2^{A\natural}$
perform ❷	perform ❷
② until $x_2^A = x_2^B = 0$	② until $x_2^B = x_2^A = 0$
② until $x_2^B \geq x_2^{B\natural}$ and $m^B = 1$	② until $x_1^A \geq x_1^{A\natural}$
perform ❶	perform ❶

in which $x_1^{A\natural} = \lambda_1(\sigma_{12}^A + \tau_2^{\mu A} + \tau_2^{\lambda A} - \theta_2^+)$, $x_1^{B\natural} = \lambda_1 \theta_1^+$, $x_2^{A\natural} = \lambda_2(\sigma_{21}^A + \tau_1^{\mu A} + \tau_1^{\lambda A} - \theta_1^+)$ and $x_2^{B\natural} = \lambda_2 \theta_2^+$.

Proof: In both the desired (optimal) trajectory and the transient, the system loops these modes (m^A, m^B): $(1,1) \rightarrow (1,2) \rightarrow (2,2) \rightarrow (2,1) \rightarrow (1,1) \rightarrow \ldots$ For the desired trajectory, the buffer levels after leaving modes (m^A, m^B) are:

$$\text{After } (1,1): \quad \begin{bmatrix} x_1^A & x_1^B & x_2^A & x_2^B \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} 0 & 0 & x_2^{A\natural} & x_2^{B\natural} \end{bmatrix}^{\mathrm{T}}$$

$$\text{After } (1,2): \quad \begin{bmatrix} x_1^A \\ x_1^B \\ x_2^A \\ x_2^B \end{bmatrix} = \begin{bmatrix} 0 \\ x_1^{B\natural} \\ x_2^{A\natural} + \lambda_2 \theta_1^+ \\ x_2^{B\natural} - \mu_2^B \max(\theta_1^+ - \sigma_{12}^B, 0) \end{bmatrix}$$

$$\text{After } (2,2): \quad \begin{bmatrix} x_1^A & x_1^B & x_2^A & x_2^B \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} x_1^{A\natural} & x_1^{B\natural} & 0 & 0 \end{bmatrix}^{\mathrm{T}}$$

$$\text{After } (2,1): \quad \begin{bmatrix} x_1^A \\ x_1^B \\ x_2^A \\ x_2^B \end{bmatrix} = \begin{bmatrix} x_1^{A\natural} + \lambda_1 \theta_2^+ \\ x_1^{B\natural} - \mu_1^B \max(\theta_2^+ - \sigma_{21}^B, 0) \\ 0 \\ x_2^{B\natural} \end{bmatrix}$$

The duration of mode (1,1) equals $x_2^{A\natural}/\lambda_2$, whereas the duration of mode (2,2) equals $x_1^{A\natural}/\lambda_1$. Furthermore, mode (1,2) always takes θ_1^+ and mode (2,1) always takes θ_2^+.

Suppose that we enter mode (1,2) in the transient for the nth time ($n > 1$). The buffer levels are at this point:

$$\begin{bmatrix} x_1^A & x_1^B & x_2^A & x_2^B \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} 0 & 0 & x_2^{A\natural} + X^{(n)} & x_2^{B\natural} \end{bmatrix}^{\mathrm{T}} \quad (29)$$

where $X^{(n)} \geq 0$ represents the e*X*tra buffer content with respect to the steady state value, when starting mode (1,2) for the nth time. Now we can wonder what the buffer levels are after mode (2,2), and consequently after mode (1,1), i.e. we would like to express $X^{(n+1)} = f(X^{(n)})$. Instead of deriving the map f explicitly, we compute an easy to find upper bound for $X^{(n+1)}$ by means of an alternative control strategy.

Consider the alternative control strategy that first goes through mode (1,2) during θ_1^+ and then stays in mode (2,2) during $x_1^{A\natural}/\lambda_1$, as if it were on the desired orbit. The resulting buffer levels are then:

$$\begin{bmatrix} x_1^A & x_1^B & x_2^A & x_2^B \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} 0 & 0 & X^{(n)} & x_2^{B\natural} \end{bmatrix}^{\mathrm{T}} \quad (30)$$

Assume that A and B both process type 2 jobs at rate $\min(\mu_2^A, \mu_2^B)$ to empty buffer x_2^A. This takes another $X^{(n)}/(\min(\mu_2^A, \mu_2^B) - \lambda_2)$ time units. The resulting buffer levels are then:

$$\begin{bmatrix} x_1^A & x_1^B & x_2^A & x_2^B \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} x_1^{A\natural} + \frac{\lambda_1}{\min(\mu_2^A, \mu_2^B) - \lambda_2} X^{(n)} & x_1^{B\natural} & 0 & 0 \end{bmatrix}^{\mathrm{T}} \quad (31)$$

With the original controller (Proposition 4.2), different values for the buffer levels are obtained. However, the original controller processes at least as much jobs as the alternative controller, at each time instant, because the original controller always processes jobs at the highest possible rate. For this reason, we know that for the real controller at the end of mode (2,2), $\begin{bmatrix} x_1^B & x_2^A & x_2^B \end{bmatrix} = \begin{bmatrix} x_1^{B\natural} & 0 & 0 \end{bmatrix}$ and x_1^A:

$$x_1^{A\natural} \leq x_1^A \leq x_1^{A\natural} + \frac{\lambda_1}{\min(\mu_2^A, \mu_2^B) - \lambda_2} X^{(n)} \quad (32)$$

Completing the controller cycle for modes (2,1) and (1,1), similar reasoning leads to the following result:

$$0 \leq X^{(n+1)} \leq \frac{\lambda_1}{\min(\mu_2^A, \mu_2^B) - \lambda_2} \cdot \frac{\lambda_2}{\min(\mu_1^A, \mu_1^B) - \lambda_1} \cdot X^{(n)} \quad (33)$$

or rewritten:

$$0 \leq X^{(n+1)} \leq \frac{R_1}{1 - R_1} \cdot \frac{R_2}{1 - R_2} \cdot X^{(n)}. \quad (34)$$

Since $R_1 + R_2 < 1$ (result of Lemma 4.1), we can conclude:

$$\lim_{n \to \infty} X^{(n)} = 0 \quad (35)$$

which means that the system converges to the desired (optimal) periodic orbit (cf. (29) with (35)). ∎

V. DISCRETE EVENT IMPLEMENTATION

Both the analysis of the servers has been performed and the feedback controller has been proposed based on a hybrid fluid model. In this section, we implement the controller in a discrete event simulation. Loosely speaking, the integer-valued buffer lengths of this simulation can be regarded as a disturbance with respect to the fluid model and thereby, we test the controller for robustness, in some sense. The discrete event system and controller have been modeled using specification language χ, see [1]. The parameter settings used for the implementation are presented in Table I. Setup costs are zero. From [3] we know the optimal process cycle of B: $\tau_1^{\mu B} = 3$, $\tau_1^{\lambda B} = 1$, $\tau_2^{\mu B} = 1$, $\tau_2^{\lambda B} = 0$ and $T = 9$. The periodic orbit of A has been computed as proposed in the second part of the proof of Proposition 3.1: $\tau_1^{\mu A} = 3.47$, $\tau_1^{\lambda A} = 2.06$, $\tau_2^{\mu A} = 0.40$, $\tau_2^{\lambda A} = 1.07$, $\theta_1^+ = 1.06$ and $\theta_2^+ = 0.53$.

The results of the simulation have been shown in Fig. 5 (buffer levels over time) and Fig. 6 (periodic orbit $A+B$, from light-gray to black for better visual understanding). As can be seen, $A+B$ converge to the optimal periodic orbit of B (cf. Fig. 8 in [3]), resulting in minimal work-in-process levels. Despite the disturbances due to the integer-valued buffer, convergence is reached, as a result of the state feedback control, in which measurements of the current situation are used for to compute the control action.

TABLE I

PARAMETER SETTINGS FOR IMPLEMENTATION OF CONTROLLER.

λ_1:	9	μ_1^A:	18	μ_1^B:	24	$m^A(0)$:	2	$x_1^A(0)$:	30
λ_2:	3	μ_2^A:	60	μ_2^B:	27	$m^B(0)$:	2	$x_2^A(0)$:	10
c_1:	1	σ_{12}^A:	1	σ_{12}^B:	2	$x_0^A(0)$:	0	$x_1^B(0)$:	30
c_2:	1	σ_{21}^A:	1	σ_{21}^B:	2	$x_0^B(0)$:	0	$x_2^B(0)$:	15

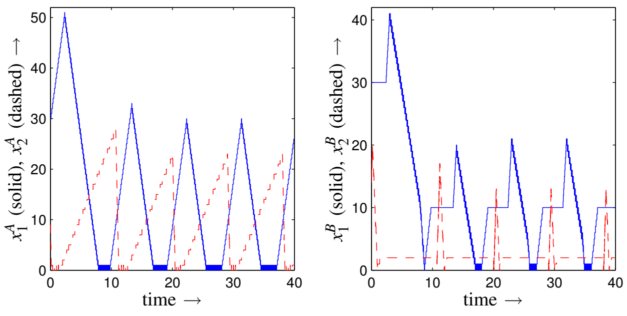

Fig. 5. Buffer levels of A (left) and B (right).

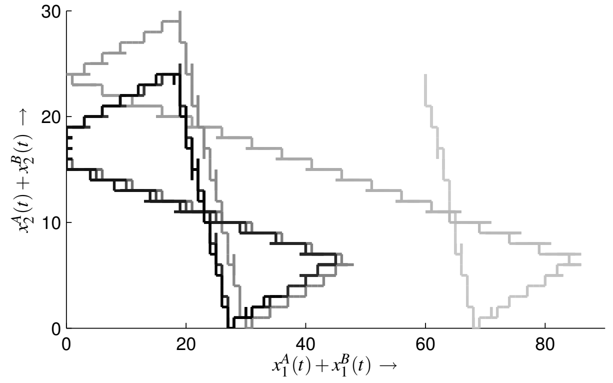

Fig. 6. Plot of total amount of type 2 jobs against amount of type 1 jobs.

VI. CONCLUSIONS AND FUTURE WORKS

In this paper we studied flowlines consisting of switching servers. The servers process more than one job type and switching between job types takes time. In previous work, the optimal process cycle for a single switching server with respect to work in process level has been derived. This minimal wip level is an absolute lower bound for average the wip level of a flowline. In this paper, we characterized the class of flowlines that actually can be controlled in such a way that the lower bound of the wip for a single switching server becomes the actual wip level for the flowline. Given an arbitrary (possibly optimal) process cycle for a single switching server, we derived conditions for the other workstations of the flowline which must be obeyed in order to make the flowline behave like the given cycle for one server. We assumed that both workstations have equal lengths of their process cycles. Once the process cycles for all servers have been determined, a controller has to make the system process in the desired way. In this paper, we proposed a state feedback controller (which can be derived using Lyapunov's direct method [6]) and we proved convergence to the desired process cycles mathematically. So instead of optimizing within a given control policy, we first determined the desired optimal behavior of the system, and then proposed a control strategy to achieve this behavior.

Although the methods and derivations have been performed for a system consisting of two servers, it can easily be extended to larger flowlines. We gave remarks on this issue throughout the paper. If the number of job types becomes more than two, the theory and methods still hold. Challenge is then to find the *optimal* process cycle for a single server. Given a feasible (not necessarily optimal) process cycle with more than two job types, all notions and methods in this paper remain valid.

Suggestions for further study on this topic are:
- finite buffer analysis for flowlines;
- optimal cycles for all flowlines, not only the flowlines that can behave as a single switching server stand-alone;
- workstation dependent weighing factors for wip, instead of type specific weighing factors;
- derivation of optimal process cycles for workstations serving more than two job types;
- defining 'optimal transient behavior' and developing controllers that guarantee optimal transient behavior.

ACKNOWLEDGMENT

The authors gratefully thank Willem de Koning, MSc. student at the Dept. of Mechanical Engineering, TU Eindhoven, for his work on the flowline example and for carefully reading of the manuscript.

REFERENCES

[1] D.A. van Beek, K.L. Man, M.A. Reniers, J.E. Rooda, and R.R.H. Schiffelers. Syntax and consistent equation semantics of hybrid Chi. *Journal of Logic and Algebraic Programming*, 68(1–2):129–210, 2006.

[2] O.J. Boxma and D.G. Down. Dynamic server assignment in a two-queue model. *European Journal of Operational Research*, 103(3):595–609, 1997.

[3] J.A.W.M. van Eekelen, E. Lefeber, and J.E. Rooda. Feedback control of 2-product server with setups and bounded buffers. In *Proceedings of the 2006 American Control Conference*, pages 544–549, 2006.

[4] J.A.W.M. van Eekelen, E. Lefeber, and J.E. Rooda. State feedback control of switching servers with setups. SE Report 2006-03, Eindhoven University of Technology, Systems Engineering Group, Department of Mechanical Engineering, Eindhoven, The Netherlands, 2006. http://se.wtb.tue.nl/sereports.

[5] W.-M. Lan and T.L. Olsen. Multiproduct systems with both setup times and costs: Fluid bounds and schedules. *Operations Research*, 54(3):505–522, 2006.

[6] E. Lefeber and J.E. Rooda. Controller design for switched linear systems with setups. *Physica A*, 363:48–61, 2006.

[7] D.M. Markowitz and L.M. Wein. Heavy traffic analysis of dynamic cyclic policies: a unified treatment of the single machine scheduling problem. *Operations Research*, 49(2):246–270, 2001.

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