Systems Engineering Group
Department of Mechanical Engineering
Eindhoven University of Technology
PO Box 513
5600 MB Eindhoven
The Netherlands
http://se.wtb.tue.nl/

SE Report: Nr. 2006-03

State feedback control of switching servers with setups

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

ISSN: 1872-1567

SE Report: Nr. 2006-03
Eindhoven, October 2006
SE Reports are available via http://se.wtb.tue.nl/sereports

Abstract

In this paper we study the control of switching servers, which can for example be found in manufacturing industry. In general, these systems are discrete event systems. A server processes multiple job types. Switching between the job types takes time and during that time, no jobs can be processed, so capacity is lost. How should a server switch between the job types in an efficient way? In this paper we derive the optimal process cycle with respect to work in process levels for a server with two job types and finite buffer capacities. The analysis is performed using a hybrid fluid model approximation. After the optimal process cycle has been defined, a state feedback controller is proposed that steers the trajectory of the system to this optimal cycle.
Workstations are often placed in series to form a flowline of servers. Our goal is to control flowlines of switching servers in a way that the work in process level is minimized. In a flowline, only the most downstream workstation influences the work in process level of the system, since upstream workstations simply move jobs from one server to the other. If it is possible to have the most downstream workstation process in its optimal cycle and the other workstations can make this happen, then optimal work in process levels are achieved. This paper investigates under which conditions the upstream workstations can make the most downstream workstation work optimally. Conditions on the upstream workstations are derived and the class of flowlines is characterized for which the optimal process cycle of an isolated downstream workstation can become the optimal process cycle for the flowline. For a flowline consisting of two workstations, a state feedback controller is proposed and convergence to the optimal process cycle is proved mathematically. An extensive case study demonstrates how the controller performs, for both the hybrid fluid model and in a discrete event implementation with stochastic inter-arrival and process times.

1 Introduction

In a lot of applications, servers have to share capacity over competing resources. Such servers can be found in manufacturing industry, food processing facilities, traffic flow networks and data communication networks. In general, these systems are discrete event systems. Switching between the competing resources might take time, the so-called setup time. In this paper we regard the scheduling problem of switching servers with non-zero setup times. We first derive the optimal process cycle with respect to work in process levels for a switching server, where we use hybrid fluid model dynamics to describe the discrete event behavior. Next, we propose a state feedback controller that brings the trajectory of a system to the optimal cycle. After the analysis for a single workstation has been completed, we expand the analysis to switching server flowlines.

With respect to scheduling of a single switching server, in [14, 15, 16] a method is proposed in which n queues are served. A stable limit cycle solution is found with minimal cycle period. The control policy then consists of fixed process periods for each job type, actually a feed forward controller. A disadvantage of this approach is that it does not deal with disturbances and moreover, it does not reduce the number of jobs in the system, in case this number of jobs is larger than necessary. In this study we use a feedback controller, to deal with disturbances and with techniques as described in [13], robustness can be shown. It also reduces the number of jobs in the system if possible.

Hybrid fluid model dynamics are also used in [2, 8]. In these studies, scheduling problems for two competing queues with both infinite and finite buffer capacities are considered, as we do in this paper. However, those studies are based on completely symmetric systems, i.e. equal setup times, equal arrival rates and equal process rates for both job types, whereas we drop these assumptions. Another assumption in [2, 8] is that the optimal process cycle is never influenced by the maximum buffer capacities, whereas we study this influence. On the other hand, [2, 8] also study optimal transient behavior, which we do not consider explicitly in this paper.

For queuing systems, policies to obtain stable process cycles are proposed in [3, 4]. Some papers, e.g. [12], use a heavy traffic assumption on top of the proposed policy. In most work, first a control policy is proposed and then analysis and (sometimes) optimization is done within the given policy. Clearing policies (serve a queue until it is empty then switch to another queue) of threshold services (serve a queue until a value has been reached) are mostly considered in this area for both stochastic and deterministic environments. In this paper, we first derive the desired process cycle and then look for a control policy. The control policies that are proposed in this study follow from methods described in [11]. The derivations are not given here, the interested reader is referred to [11]. Related work for stochastic systems has been done in [9], but only equal maximum process rates and infinite buffer capacities are considered therein.

In [10] a fluid model is presented for a multi-product server which has to choose between the competing resources. A convex optimization problem is defined which results in a lower bound on the work in process levels in a deterministic environment. It is stated that the polling table resulting from the optimization is rare to find and in most cases unachievable. In this paper we show that for a server with two job types, this lower bound is actually achievable.

A well known scheduling heuristic is based on the $c\mu$-rule (see e.g. [5]) where switching (without setup times) takes place according to a $c\mu$ index where c is a cost rate and μ the process rate. The job type with highest index has priority. In our research, an optimal process cycle is derived, in which a *slow-mode* may occur (also referred to in literature as 'idling' [6] or 'cruising' [10]). In this mode, lots are processed at a lower rate than the maximum rate. If the slow-mode occurs, it takes place at the queue with the highest $c\lambda$ index, even if the $c\mu$ index of the other job type is higher, as is shown in the case study in Section 6.

For flowlines of switching servers, we also want to find optimal process cycles. Since in a flowline, the work in process level is completely determined by the most downstream workstation (other workstations can not change the amount of jobs in the system), the optimal work in process level of the flowline can never be less than that of the most downstream workstation in isolation. Therefore, if the most downstream workstation can process at its optimal cycle and the other workstations can make this happen, minimal work in process levels are achieved for the

whole flowline. This paper investigates and characterizes the class of flowlines for which this is possible.
The remainder of this paper can roughly be divided into two parts and is organized as follows. In sections 2 and 3, the hybrid fluid model (ODE) dynamics of a single switching server are described and the optimal process cycle of such a server is derived. In Section 4 a state feedback controller for a single switching server is proposed and convergence is proved analytically. The influence of finite buffer capacities on the optimal process cycle and the feedback controller is then studied in Section 5. The part about the single switching server is concluded with an illustrative example. The second part of this paper deals with flowlines of switching servers. Sections 7 and 8 give a general characterization of flowlines of switching servers. The dynamics are defined and the conditions are investigated under which upstream workstations can make the most downstream workstation perform at its optimal process cycle. For a flowline consisting of two workstations, a state feedback controller is proposed in Section 9. A case study is presented in which the controller is implemented in both a hybrid fluid model simulation and a stochastic discrete event simulation. Section 11 states the conclusions and some recommendations for further research.

2 Switching server: characteristics and dynamics

Before desired trajectories for switching servers and controllers can be defined, first the characteristics and dynamics of a switching server are presented. In Figure 1 a schematic representation of a *workstation* is given. A workstation consists of a number of parallel first-in-first-out (FIFO) job-type specific buffers and a server. A *job* is the to be processed entity. The number of jobs in a buffer is denoted by x_i. The server can process only one type of jobs at a time. Switching from job type i to job type j takes $\sigma_{ij} \geq 0$ time units. Without loss of generality, we use 'hours' as time unit in this study. Furthermore, unless indicated otherwise, subscript i refers to the job type: $i \in \{1,2,\ldots,n\}$, with n the number of different job types in the system (in this study we consider $n = 2$). Jobs of type i arrive at the workstation with a constant inter-arrival time of $1/\lambda_i$ hours. The server processes this type of jobs with a constant process time of $1/\mu_i$ hours. In Figure 1, a workstation is shown that processes two job types. We use this workstation in the analysis in this paper.
Since it is hard to describe the discrete event dynamics of the workstation, we use a hybrid fluid approximation model to describe the dynamics and perform all analysis with. Instead of using the interarrival *times* and process *times* of jobs, we use the arrival *rate* $\lambda_i > 0$ and process *rate* $\mu_i > 0$. A consequence is that the buffer levels $x_i \in \mathbb{R}_+$, with $\mathbb{R}_+ = [0,\infty)$.
The partial utilization of a job type on a server is denoted by ρ_i: $\rho_i = \lambda_i/\mu_i$. For reasons of stability, the total utilization of a server must be strictly smaller than 1: $\sum_i \rho_1 < 1$. The utilization must not equal 1, since then there is no time left to process the jobs that arrive during a setup and buffer levels eventually explode.

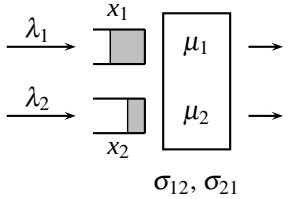

Figure 1: Switching server with two job types.

The state of the system consists not only of the buffer levels x_1 and x_2, but also of the *mode* m of

the server and the remaining setup time x_0. The mode is the job type that the server is processing or setting up for. The remaining setup time is positive while a setup is performed and zero otherwise. The state of the system at time t is given by:

$$\mathbf{x}(t) = \begin{bmatrix} x_1(t) & x_2(t) & x_0(t) & m(t) \end{bmatrix}^{\mathrm{T}} \in \mathbb{R}^3_+ \times \{1,2\}. \tag{1}$$

We assume that the process rate of the machine can be manipulated by a controller, as long as it does not exceed the maximum rate μ_i. Let manipulative input $u_i \leq \mu_i$ be the rate at which jobs are processed. Another input of the system is the required activity u_0 of the server. Possible activities are:

$u_0 = ❶$: setup for type 1 jobs
$u_0 = ①$: process type 1 jobs
$u_0 = ❷$: setup for type 2 jobs
$u_0 = ②$: process type 2 jobs

The complete input vector $\mathbf{u}$ is given by:

$$\mathbf{u}(t) = \begin{bmatrix} u_0(t) & u_1(t) & u_2(t) \end{bmatrix}^{\mathrm{T}} \in \{①,❶,②,❷\} \times \mathbb{R}^2_+. \tag{2}$$

The inputs are bound by constraints: It is not possible to process jobs while busy with a setup; when the server is processing jobs of a specific type, other job types can not be processed; if a buffer contains jobs, the process rate for that type can not exceed the maximum process rate and if a buffer is empty, the server can process jobs with at most the arrival rate. Formally, the constraints are:

$$\begin{array}{llll}
u_0 \in \{❶,❷\}, & u_1 = 0, & u_2 = 0 & \text{for } x_0 > 0 \\
u_0 \in \{①,❷\}, & 0 \leq u_1 \leq \mu_1, & u_2 = 0 & \text{for } x_0 = 0, x_1 > 0, m = 1 \\
u_0 \in \{①,❷\}, & 0 \leq u_1 \leq \lambda_1, & u_2 = 0 & \text{for } x_0 = 0, x_1 = 0, m = 1 \\
u_0 \in \{❶,②\}, & u_1 = 0, & 0 \leq u_2 \leq \mu_2 & \text{for } x_0 = 0, x_2 > 0, m = 2 \\
u_0 \in \{❶,②\}, & u_1 = 0, & 0 \leq u_2 \leq \lambda_2 & \text{for } x_0 = 0, x_2 = 0, m = 2
\end{array}$$

Input u_0 causes state variables to jump between values. Not only mode m jumps between 1 and 2, but also the remaining setup time jumps from 0 to σ_{ij} when $u_0 \in \{❶,❷\}$:

$$x_0 := \sigma_{21},\ m := 1 \text{ for } u_0 = ❶ \text{ and } m = 2 \tag{3}$$
$$x_0 := \sigma_{12},\ m := 2 \text{ for } u_0 = ❷ \text{ and } m = 1. \tag{4}$$

Note that during a setup, an intervention may cause a switch to an other job type. In that case the ongoing setup is interrupted and the new setup starts.
In addition to these discrete event dynamics, the following continuous dynamics apply:

$$\dot{x}_0(t) = \begin{cases} -1 & \text{for } u_0(t) \in \{❶,❷\} \\ 0 & \text{for } u_0(t) \in \{①,②\} \end{cases} \tag{5a}$$
$$\dot{x}_1(t) = \lambda_1 - u_1(t) \tag{5b}$$
$$\dot{x}_2(t) = \lambda_2 - u_2(t). \tag{5c}$$

The workstation with two product types has a fixed process cycle:
process type 1 → setup to type 2 → process type 2 → setup to type 1 → ...
This study first focuses on what the 'best' way is to perform this cycle. This 'best' adjective can be translated into an optimization problem. In the next section, we study the optimal process cycle with respect to averaged weighted work in process (wip) levels.

3 Optimal process cycle of single switching server

In this section, we consider a switching server with two incoming job types and constant arrival rates, as presented in Section 2. A general optimization problem is formulated for averaged weighted work in process levels. For strictly increasing cost functions, the general form of the solution is presented and for linear costs, an explicit solution is derived.

Starting from an arbitrary initial state, we want to achieve a switching policy that in the end minimizes the costs related to wip levels. The cost function J has the following form:

$$J = \lim_{t\to\infty} \frac{1}{t} \int_0^t g_1\big(x_1(s)\big) + g_2\big(x_2(s)\big)\mathrm{d}s \tag{6}$$

with $g_i : \mathbb{R}_+ \to \mathbb{R}_+$ functions that relate costs to wip levels (buffer levels). If we assume that higher buffer levels result in higher costs, i.e. if we restrict ourselves to strictly increasing functions g_i, i.e. $g_i(x_i) > g_i(y_i)$ if $x_i > y_i$, then we can make the following statements.

Lemma 3.1. *When serving type i, optimal policies first serve at the highest possible rate, after which they might idle.*

Proof. Suppose that a policy is given for which after having completed the setup to type i, buffer i contains x_i^0 jobs and at the end of mode i, buffer i contains x_i^f jobs. Then one can consider the alternative policy which serves type i equally long in mode i and first serves at the highest possible rate, i.e. at the maximal processing rate as long as the buffer contains jobs. This alternative policy processes at the arrival rate in case the buffer is empty, and in the end idles to make sure that at the end of mode i the buffer contains x_i^f jobs. Clearly, during mode i the number of jobs in the buffer can not decrease faster and in the end can not increase faster than in this alternative strategy. Therefore, for the alternative policy at each time instant the wip level of type i is minimal. In particular, if the given policy is different, there are time instants at which it is less. Since the time evolution of the other job type(s) is the same for both policies and g_i is strictly increasing, the costs are strictly less using the alternative strategy. □

Lemma 3.2. *Optimal policies do not idle.*

Proof. Suppose that an optimal policy would idle in mode i. Given the result in Lemma 3.1 this is at the end of mode i. Furthermore, suppose that for this policy service in the next mode stops at time t_f. Consider an alternative policy which does not idle in mode i, but switches immediately to the next mode and stays in this mode until t_f. For this alternative strategy the evolution of the buffer contents of type i does not change. However, (some of) the jobs that are served in the next mode are served sooner. Therefore, for some period of time the number of jobs in the buffer of the next mode is strictly less in the alternative strategy. Since g is strictly increasing, the costs are strictly less for the alternative strategy. Therefore, an optimal policy does not idle. □

Corollary 3.3. *The optimal switch policy for a switching server with respect to time averaged weighted wip levels, with $n = 2$ job types and strictly increasing costs on wip levels, always processes at maximum rate when jobs are available in the buffer. If no jobs are available, jobs are served at arrival rate. The optimal switch policy never makes the server idle.*

Remark 3.4. Notice that in the proofs of lemmas 3.1 and 3.2 we did not explicitly use the fact that $n = 2$. Therefore, Corollary 3.3 actually holds for a server serving n product types, where (6) is replaced with

$$J = \lim_{t\to\infty} \frac{1}{t} \int_0^t \sum_{i=1}^{n} g_i\big(x_i(s)\big)\mathrm{d}s, \tag{7}$$

and all g_i are strictly increasing.

In order to derive the optimal steady state behavior for a server serving two types of jobs, we first look at properties of optimal behavior for one type only without considering the other type. Note that in general it might not be possible to have both types behave optimally, but at least this provides us with a lower bound for optimal system behavior. Next, we show that both types can behave optimally, i.e. that this lower bound can be achieved.

Lemma 3.5. *For optimal steady state behavior of type i, buffer i is always emptied.*

Proof. Consider a policy where at time $t = t_0$ the systems stops serving buffer i which is not yet empty, and that at $t = t_s$ type i is served again. Consider an alternative policy where the system stops serving buffer i at $t = t_0 + \varepsilon$, mimics the given policy with a time delay of ε, and then

starts serving type i at $t = t_s + \varepsilon$, where $\varepsilon > 0$ but small enough to result in a feasible alternative solution. Since some jobs of type i are served sooner in the alternative strategy, the number of jobs in buffer i is strictly less at each time instant for the alternative strategy. Since g_i is strictly increasing, the contribution to the costs by type i is strictly less for the alternative strategy. □

Remark 3.6. Note that the contribution to the costs by the other type(s) might be higher for the proposed alternative strategy, but as mentioned earlier we currently focus only at optimizing a single job type without taking into account the other job type(s).

Lemma 3.7. *For optimal steady state behavior of type i, vacations always take equally long, i.e. setups and serving the other type(s) between two successive services of type i.*

Proof. Assume that for an optimal steady state behavior, service for type i stops at time $t = 0$. From Lemma 3.5 we know that buffer i is empty then. Since jobs of type i arrive at a constant rate, the contents of buffer i grow linearly, until type i is served again. Then buffer i is emptied, the contents of buffer i grow linearly until type i is served again, and buffer i is emptied again, see also Figure 2. Now assume that both periods of not serving type i are not equally long. Without loss

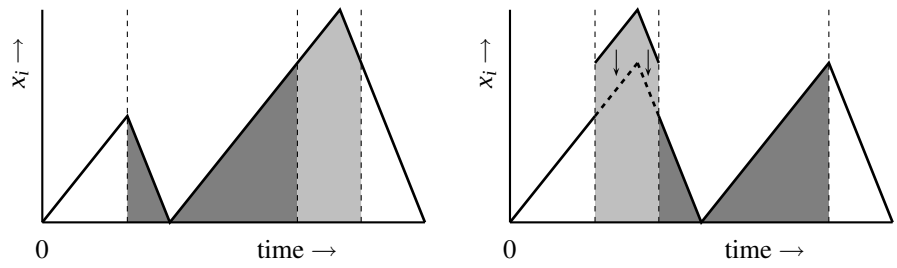

Figure 2: Times between successive services are equally long.

of generality we assume that the first period is shorter than the second. The resulting evolution of the buffer contents of buffer i is depicted in the left hand side of Figure 2. In the right hand side of Figure 2 the resulting evolution of the buffer contents of buffer i is depicted in case both periods of not serving type i are equally long. The graph of the buffer contents of type i in the left hand side figure can be divided into four parts. The first part consists of the period that type i is not served. The second part consists of the period that type i is first served and then not served, until the buffer reaches the value it would have had if both periods of not serving type i would have been equally long. The third part consists of the period that the buffer contents exceed this value and return at this value. And, finally, the fourth part consists of the remaining period. The different parts can be distinguished based on the fill color in Figure 2.
By interchanging the second (dark gray) and third (light gray) part of this graph, the right hand side of Figure 2 can be obtained. Next, it can be seen that the alternative strategy (not serving type i for equally long duration) results in strictly less jobs of type i for a certain amount of time. Since g_i is strictly increasing, the resulting costs for this alternative strategy are strictly less for type i. □

Lemmas 3.5 and 3.7 are valid if a job type is optimized in isolation. Since multiple job types occur in switching servers, optimizing each job type separately might not lead to a feasible solution. However, in case of $n = 2$ job types, this is not an issue.

Corollary 3.8. *The optimal steady state process cycle for a switching server with 2 different job types and strictly increasing costs on wip levels has in general the shape as shown in Figure 3, with the costs defined as:*

$$J = \frac{1}{T}\int_0^T g_1\big(x_1(s)\big) + g_2\big(x_2(s)\big)\mathrm{d}s. \tag{8}$$

In the left-hand side graph of Figure 3, x_1 and x_2 have been plotted against each other. The right-hand side graphs show the buffer levels over time, with the slopes of the lines annotated to them. The cycle has period 1 with period length T.

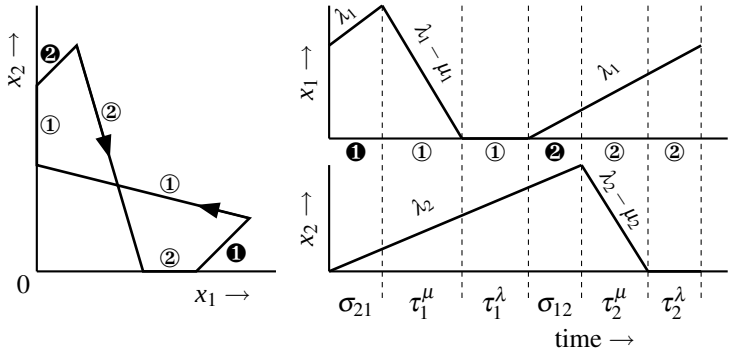

Figure 3: General form optimal process cycle for a switching server processing two job types. Left: Periodic orbit. Right: buffer levels over time, with the slopes of the lines.

In the remainder of this paper, we assume linear costs on wip levels, resulting in the following costs for the optimal process cycle:

$$J = \frac{1}{T}\int_0^T c_1 x_1(s) + c_2 x_2(s) \mathrm{d}s \tag{9}$$

where c_1 and c_2 are constant weighing factors for job type 1 and 2 respectively. In addition, without loss of generality, we assume $c_1\lambda_1 \geq c_2\lambda_2$. Moreover, we define $\sigma = \sigma_{21} + \sigma_{12}$. Now we can explicitly determine the optimal behavior as presented in Corollary 3.8.

Proposition 3.9. *The optimal process cycle with respect to time averaged weighted wip levels for a switching server with two different job types and linear costs on wip levels* (9)*, has a* slow-mode *for at most one job type (type 1). During the slow-mode, jobs are processed at their arrival rate. The slow-mode occurs if and only if* $c_1\lambda_1(\rho_1+\rho_2) + (c_2\lambda_2 - c_1\lambda_1)(1-\rho_2) < 0$.

Proof. From Corollary 3.8 we know the general shape of the optimal process cycle. Let τ_i^μ denote the duration of serving type i at maximal rate, and let τ_i^λ denote the duration of the slow-mode of type i, as indicated in Figure 3.
In steady state, the system reaches the same situation after one complete period. During processing a job type at full rate, as many lots are processed as arrive during setups and processing the other job type:

$$\lambda_1(\sigma_{12} + \tau_2^\mu + \tau_2^\lambda + \sigma_{21}) = (\mu_1 - \lambda_1)\tau_1^\mu \tag{10}$$

$$\lambda_2(\sigma_{21} + \tau_1^\mu + \tau_1^\lambda + \sigma_{12}) = (\mu_2 - \lambda_2)\tau_2^\mu. \tag{11}$$

Let $\tau_1^\lambda = \alpha_1\sigma$ and $\tau_2^\lambda = \alpha_2\sigma$ with $\alpha_1 \geq 0, \alpha_2 \geq 0$. Solving (10) and (11) for τ_i^μ we obtain:

$$\begin{bmatrix} \sigma \\ \tau_1^\mu \\ \tau_1^\lambda \\ \tau_2^\mu \\ \tau_2^\lambda \end{bmatrix} = \frac{\sigma_{12}+\sigma_{21}}{1-\rho_1-\rho_2} \begin{bmatrix} 1-\rho_1-\rho_2 \\ \alpha_1\rho_1\rho_2 + \alpha_2\rho_1(1-\rho_2) + \rho_1 \\ \alpha_1(1-\rho_1-\rho_2) \\ \alpha_1\rho_2(1-\rho_1) + \alpha_2\rho_1\rho_2 + \rho_2 \\ \alpha_2(1-\rho_1-\rho_2) \end{bmatrix}. \tag{12}$$

The weighted mean wip level is computed by determining the area underneath the right hand side graphs of Figure 3, divided by the period length T:

$$\frac{1}{T}\int_0^T c_1 x_1(s)\mathrm{d}s = \frac{c_1 \cdot \frac{1}{2} \cdot (\sigma + \tau_1^\mu + \tau_2^\mu + \tau_2^\lambda) \cdot (\mu_1 - \lambda_1)\tau_1^\mu}{\sigma + \tau_1^\mu + \tau_1^\lambda + \tau_2^\mu + \tau_2^\lambda} \tag{13}$$

$$\frac{1}{T}\int_0^T c_2 x_2(s)\mathrm{d}s = \frac{c_2 \cdot \frac{1}{2} \cdot (\sigma + \tau_1^\mu + \tau_1^\lambda + \tau_2^\mu) \cdot (\mu_2 - \lambda_2)\tau_2^\mu}{\sigma + \tau_1^\mu + \tau_1^\lambda + \tau_2^\mu + \tau_2^\lambda}. \tag{14}$$

Since $\frac{1}{2}\sigma/(1-\rho_1-\rho_2)$ is a constant, we need to solve the problem of minimizing

$$\frac{c_1\lambda_1(1-\rho_1)[1+\alpha_1\rho_2+\alpha_2(1-\rho_2)]^2 + c_2\lambda_2(1-\rho_2)[1+\alpha_1(1-\rho_1)+\alpha_2\rho_1]^2}{1+\alpha_1(1-\rho_1)+\alpha_2(1-\rho_2)} \tag{15}$$

subject to the constraints $\alpha_1 \geq 0$ and $\alpha_2 \geq 0$.
Instead of solving this optimization problem we first consider the unconstrained problem, i.e. we ignore $\alpha_1 \geq 0$ and $\alpha_2 \geq 0$. Taking the derivatives of (15) with respect to α_1 and α_2 and putting them to zero gives the solution of the unconstrained problem: $\alpha_1 = \alpha_2 = -1$ (infeasible; local maximum) or

$$\alpha_1 = 1 - \frac{2c_2\lambda_2}{c_1\lambda_1(1-\rho_2)+c_2\lambda_2(1-\rho_1)}, \alpha_2 = 1 - \frac{2c_1\lambda_1}{c_1\lambda_1(1-\rho_2)+c_2\lambda_2(1-\rho_1)}, \tag{16}$$

which is a local minimum. For this second solution, note that

$$\alpha_1 + \alpha_2 = -2\frac{c_1\lambda_1\rho_2 + c_2\lambda_2\rho_1}{c_1\lambda_1(1-\rho_2)+c_2\lambda_2(1-\rho_1)} < 0 \tag{17}$$

which means that at least one of the two constraints is active. Suppose that $\alpha_2 = 0$. Then we need to solve the following optimization problem:

$$\min_{\alpha_1 \geq 0} \frac{c_1\left[1+\alpha_1\rho_2\right]^2\lambda_1(1-\rho_1) + c_2\left[1+\alpha_1(1-\rho_1)\right]^2\lambda_2(1-\rho_2)}{1+\alpha_1(1-\rho_1)}. \tag{18}$$

From $\frac{\mathrm{d}J}{\mathrm{d}\alpha_1} = 0$ we obtain:

$$\begin{aligned}\left[c_1\lambda_1\rho_2^2(1-\rho_1) + c_2\lambda_2(1-\rho_1)^2(1-\rho_2)\right]\alpha_1^2 + \ldots \\ 2\left[c_1\lambda_1\rho_2^2 + c_2\lambda_2(1-\rho_1)(1-\rho_2)\right]\alpha_1 + \ldots \\ [c_1\lambda_1(\rho_1+\rho_2) + (c_2\lambda_2 - c_1\lambda_1)(1-\rho_2)] = 0.\end{aligned} \tag{19}$$

The coefficients in front of α_1^2 and α_1 are both strictly positive, so this parabola has a positive real root iff $c_1\lambda_1(\rho_1+\rho_2) + (c_2\lambda_2 - c_1\lambda_1)(1-\rho_2) < 0$. Note that this is only possible if $c_1\lambda_1 > c_2\lambda_2$. The value of α_1 can be obtained by solving (19) taking into account that α_1 has to be non-negative:

$$\alpha_1 = \begin{cases} 0 \text{ if } c_1\lambda_1(\rho_1+\rho_2) + (c_2\lambda_2 - c_1\lambda_1)(1-\rho_2) \geq 0 \\ \text{positive real root of (19) otherwise.} \end{cases} \tag{20}$$

For reasons of symmetry, if the other constraint would be active ($\alpha_1 = 0$), the solution would be:

$$\alpha_2 = \begin{cases} 0 \text{ if } c_2\lambda_2(\rho_1+\rho_2) + (c_1\lambda_1 - c_2\lambda_2)(1-\rho_1) \geq 0 \\ \text{positive real root of second order polynomial otherwise.} \end{cases} \tag{21}$$

Since we assumed that $c_1\lambda_1 \geq c_2\lambda_2$, we can conclude that $\alpha_2 = 0$ and α_1 is given by (20).
With this result, the optimal process cycle for a switching server with two product types, setup times and linear costs on buffer levels has completely been defined. Under certain conditions, as stated in Proposition 3.9, a slow-mode occurs in one of the job types. □

Remark 3.10. When the server processes jobs at the arrival rate, it is in slow-mode. One might argue that this slow-mode does not occur for an optimal policy since it means that capacity is lost due to processing at lower rates than the maximum rate. However, introducing a slow-mode implies that the process cycle becomes longer, which in turn implies that the system switches

less. Apparently there is a trade-off between losing capacity due to processing at lower rates and losing capacity due to switching more often. The outcome of this tradeoff depends on the specific choice for the functions g_i.

In addition to the optimal process cycle, another process cycle is of particular interest for the remainder of this paper: the cycle with a minimal period and with minimal maximum buffer lengths. The periodic orbit of this cycle is shown in the left hand side graph of Figure 4. As can be seen, no slow-modes occur: after having processed a specific job type until its buffer is empty, the system immediately switches to the other job type. The buffer values at specific corner points of this *(pure) bow tie* curve are indicated with an asterisk (*) and we use hats ($\hat{}$) to indicate maximum buffer levels for a given cycle. The coordinates of interest are:

$$\text{After } ①\text{: } (0,\, x_2^*) \qquad \text{with } x_2^* = \lambda_2 \left(\sigma_{21} + \frac{\sigma\rho_1}{1-\rho_1-\rho_2} \right) \tag{22a}$$

$$\text{After } ❷\text{: } (\lambda_1\sigma_{12},\, \hat{x}_2^*) \qquad \text{with } \hat{x}_2^* = \lambda_2\sigma \left(\frac{1-\rho_2}{1-\rho_1-\rho_2} \right) \tag{22b}$$

$$\text{After } ②\text{: } (x_1^*,\, 0) \qquad \text{with } x_1^* = \lambda_1 \left(\sigma_{12} + \frac{\sigma\rho_2}{1-\rho_1-\rho_2} \right) \tag{22c}$$

$$\text{After } ❶\text{: } (\hat{x}_1^*,\, \lambda_2\sigma_{21}) \qquad \text{with } \hat{x}_1^* = \lambda_1\sigma \left(\frac{1-\rho_1}{1-\rho_1-\rho_2} \right). \tag{22d}$$

These coordinates are in accordance with the values in the examples of [2, 8] for the symmetric queues.

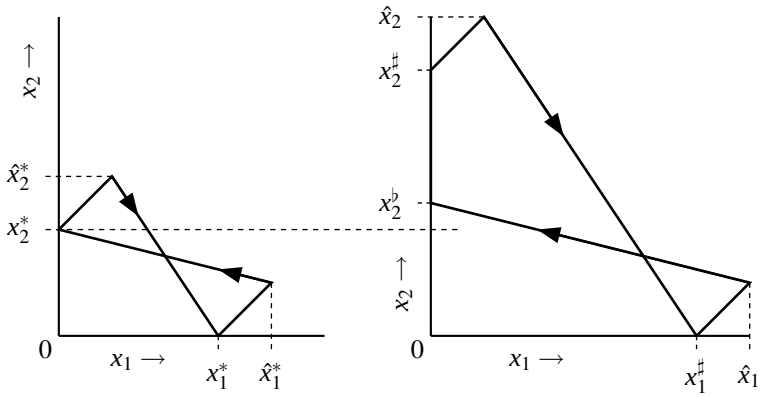

Figure 4: Pure bow tie curve and truncated bow tie curve.

In the right hand side graph of Figure 4 the periodic orbit of the optimal cycle has been shown for the case where a slow-mode occurs. This orbit is referred to as the *truncated bow tie* curve. The coordinates with flat ($^\flat$) and sharp ($^\sharp$) symbols denote the points where the slow-mode starts and ends respectively.

Remark 3.11. The lines corresponding to similar phases in the process cycle have the same slopes in the left hand side and right hand side graph. Notice that therefore $x_2^* \leq x_2^\flat$ and $x_1^* \leq x_1^\sharp$. Although it looks counterintuitive, the right hand side graph has lower mean weighted wip levels than the left hand side graph, in case a slow-mode occurs according to the condition stated in Proposition 3.9. The counterintuitiveness is due to the fact that sense of time is lost in these graphs: the left hand side graph has a shorter period than the right hand side graph. The system spends a relatively large amount of time on the vertical axis ($x_1 = 0$) of the truncated bow tie, compared to the pure bow tie.

The coordinates of interest for the truncated bow tie curve are:

$$\text{After } ① \text{ at } \mu_1: \quad (0,\, x_2^\flat) \qquad \text{with } x_2^\flat = \lambda_2 \left(\sigma_{21} + \frac{\sigma\rho_1(1+\alpha_1\rho_2)}{1-\rho_1-\rho_2} \right) \tag{23a}$$

$$\text{After } ① \text{ at } \lambda_1: \quad (0,\, x_2^\sharp) \qquad \text{with } x_2^\sharp = \lambda_2 \left(\sigma_{21} + \frac{\sigma(\alpha_1(1-\rho_1)(1-\rho_2)+\rho_1)}{1-\rho_1-\rho_2} \right) \tag{23b}$$

$$\text{After } ❷: \quad (\lambda_1\sigma_{12},\, \hat{x}_2) \qquad \text{with } \hat{x}_2 = \lambda_2\sigma \left(\frac{(1-\rho_2)(1+\alpha_1(1-\rho_1))}{1-\rho_1-\rho_2} \right) \tag{23c}$$

$$\text{After } ②: \quad (x_1^\sharp,\, 0) \qquad \text{with } x_1^\sharp = \lambda_1 \left(\sigma_{12} + \frac{\sigma\rho_2(1+\alpha_1(1-\rho_1))}{1-\rho_1-\rho_2} \right) \tag{23d}$$

$$\text{After } ❶: \quad (\hat{x}_1,\, \lambda_2\sigma_{21}) \qquad \text{with } \hat{x}_1 = \lambda_1\sigma \left(1 + \frac{\rho_2(1+\alpha_1(1-\rho_1))}{1-\rho_1-\rho_2} \right). \tag{23e}$$

Note that if $\alpha_1 = 0$, the pure bow tie trajectory actually is the optimal curve: $x_2^\sharp = x_2^\flat = x_2^*$ and consequently $x_1^\sharp = x_1^*$.
In this section, the optimal process cycle for a switching server with two job types and setup times has completely been defined. Additionally, the process cycle with minimal period and minimal extreme buffer lengths has been characterized. In the next section, a state feedback controller is proposed that steers the trajectory of the system to the desired optimal trajectory, from any arbitrary start point (initial state).

4 State feedback controller for single switching server

During the ramp-up of a factory, or due to disturbances, the trajectory of a workstation is in general not on the desired curve. Therefore, a controller is needed to steer the process trajectory to the desired (optimal) curve. A possible controller can be derived using the theory presented in [11].

Proposition 4.1. *The following state feedback control law brings the system of Figure 1 to the optimal periodic cycle with respect to minimal time averaged wip level.*

$$(u_0,u_1,u_2) = \begin{cases} (①,\mu_1,0) & \text{if } m=1,\, x_0=0,\, x_1>0 \\ (①,\lambda_1,0) & \text{if } m=1,\, x_0=0,\, x_1=0,\, x_2<x_2^\sharp \\ (❷,0,0) & \text{if } m=1,\, x_0=0,\, x_1=0,\, x_2\geq x_2^\sharp \\ (❷,0,0) & \text{if } m=2,\, x_0>0 \\ (②,0,\mu_2) & \text{if } m=2,\, x_0=0,\, x_2>0 \\ (②,0,\lambda_2) & \text{if } m=2,\, x_0=0,\, x_2=0,\, x_1<x_1^\sharp \\ (❶,0,0) & \text{if } m=2,\, x_0=0,\, x_2=0,\, x_1\geq x_1^\sharp \\ (❶,0,0) & \text{if } m=1,\, x_0>0. \end{cases} \tag{24}$$

Remark 4.2. An informal description of this controller is:

- Mode 1: ① at μ_1 as long as $x_1 > 0$, then go to Mode 2.
- Mode 2: ① at λ_1 as long as $x_2 < x_2^\sharp$, then go to Mode 3.
- Mode 3: perform ❷, after σ_{12} go to Mode 4.
- Mode 4: ② at μ_2 as long as $x_2 > 0$, then go to Mode 5.
- Mode 5: ② at λ_2 as long as $x_1 < x_1^\sharp$, then go to Mode 6.
- Mode 6: perform ❶, after σ_{21} go to Mode 1.

Dependent on the state of the system, the controller is in 1 of the 6 modes initially. This follows trivially from the controller mode descriptions. Mode 2 and Mode 5 of this controller might have a duration of zero, immediately forwarding to Mode 3 and Mode 6 respectively.

Proof. The controller loops through Modes 1–6. Given an initial state, the controller starts in one of the modes. Eventually, the system reaches the same Mode again. We are interested in how the state has changed after one complete cycle. Based on the way the state has changed, convergence is proven.

Assume the n^{th} start of ❷ (start of Mode 3) is from coordinate $(0, x_2^{(n)})$. Now we wonder from which coordinate the $(n+1)^{\text{st}}$ start of ❷ takes place. Suppose that $x_2^{(n)}$ is relatively large compared to $x_2^\sharp$. In that case, the duration of both Mode 2 and Mode 5 is zero. In one cycle, the coordinates then become:

$$\begin{aligned}
\left(0, x_2^{(n)}\right) &\xrightarrow{❷} \left(\lambda_1\sigma_{12}, x_2^{(n)} + \lambda_2\sigma_{12}\right) \\
&\xrightarrow{②} \left(\lambda_1\left(\sigma_{12} + \frac{x_2^{(n)} + \lambda_2\sigma_{12}}{\mu_2 - \lambda_2}\right), 0\right) \\
&\xrightarrow{❶} \left(\lambda_1\left(\sigma + \frac{x_2^{(n)} + \lambda_2\sigma_{12}}{\mu_2 - \lambda_2}\right), \lambda_2\sigma_{21}\right) \\
&\xrightarrow{①} \left(0, \lambda_2\left(\sigma_{21} + \frac{\lambda_1\left(\sigma + \frac{x_2^{(n)} + \lambda_2\sigma_{12}}{\mu_2 - \lambda_2}\right)}{\mu_1 - \lambda_1}\right)\right)
\end{aligned} \tag{25}$$

which results in:

$$\begin{aligned}
x_2^{(n+1)} &= \lambda_2\left(\sigma_{21} + \frac{\lambda_1\left(\sigma + \frac{x_2^{(n)} + \lambda_2\sigma_{12}}{\mu_2 - \lambda_2}\right)}{\mu_1 - \lambda_1}\right) \\
&= \frac{\rho_1\rho_2}{(1-\rho_1)(1-\rho_2)}(x_2^{(n)} - x_2^*) + x_2^*.
\end{aligned} \tag{26}$$

In case that $x_2^{(n)}$ was not large compared to $x_2^\sharp$, either Mode 2 or Mode 5 has a strictly positive duration and we end up on the optimal curve, getting:

$$x_2^{(n+1)} = x_2^\sharp. \tag{27}$$

Combining (26) and (27) results in the difference equation:

$$x_2^{(n+1)} = \max\left(\frac{\rho_1\rho_2}{(1-\rho_1)(1-\rho_2)}(x_2^{(n)} - x_2^*) + x_2^*,\ x_2^\sharp\right), \quad x_2^{(0)} = x_2^0 \tag{28}$$

which has a solution

$$x_2^{(n)} = \max\left(\left[\frac{\rho_1\rho_2}{(1-\rho_1)(1-\rho_2)}\right]^n (x_2^0 - x_2^*) + x_2^*,\ x_2^\sharp\right). \tag{29}$$

Since

$$0 < \frac{\rho_1\rho_2}{(1-\rho_1)(1-\rho_2)} = 1 - \frac{1-\rho_1-\rho_2}{(1-\rho_1)(1-\rho_2)} < 1 \tag{30}$$

we obtain

$$\lim_{n\to\infty} x_2^{(n)} = \max(x_2^*,\ x_2^\sharp) = x_2^\sharp. \tag{31}$$

Remark 4.3. For $x_2^* < x_2^\sharp$ (i.e. when a slow-mode occurs, $\alpha_1 > 0$) as well as $x_2(0) < x_2^\sharp$, we obtain convergence in finite time.

□

The controller presented in Proposition 4.1 is a double threshold policy, i.e. exhaustive processing and switch only whenever the other buffer level has reached some value. In [9] a similar double threshold policy is used, though it was done in a stochastic environment. The difference between their result and the controller presented in this study is that their analysis has only been done for workstations with equal (stochastic) process rates for both job types. Moreover, in [9], only infinite buffer capacities have been taken into account. In the next section, the optimal process cycle analysis and feedback controller are extended to situations with finite buffer capacities.

5 Finite buffer capacities: implications on optimal process cycle and feedback controller

In this section, the optimal process cycle for a switching server with two job types and setup times is determined for workstations with finite buffer capacity. These situations not only occur in physical systems, like manufacturing systems or food processing industry, but also in data flow or communication systems, where data storage is most often limited. In fact, storage capacity is always limited, but in special cases it can be regarded as virtually unlimited and then the analysis of sections 3 and 4 are applicable. The results from these sections are used to derive the optimal process cycle in the finite buffer case.
Due to buffer constraints, the optimal process cycle might change. This is the case if during the optimal cycle, a buffer level exceeds the maximum capacity. Within the buffer constraints, a new optimal cycle can be looked for. The maximum number of jobs in the buffers is denoted by $x_1^{\max}$ and $x_2^{\max}$. In Section 3, we defined the pure bow tie curve as the periodic orbit with minimal period and with minimal maximum buffer levels: $\hat{x}_1^*$ and $\hat{x}_2^*$. If the buffer capacity is less than these values, no periodic orbit can be found, so the first conditions on the new optimal process cycle are:

$$x_1^{\max} \geq \hat{x}_1^*; \quad x_2^{\max} \geq \hat{x}_2^* \tag{32}$$

with $\hat{x}_1^*$ and $\hat{x}_2^*$ as in (22).

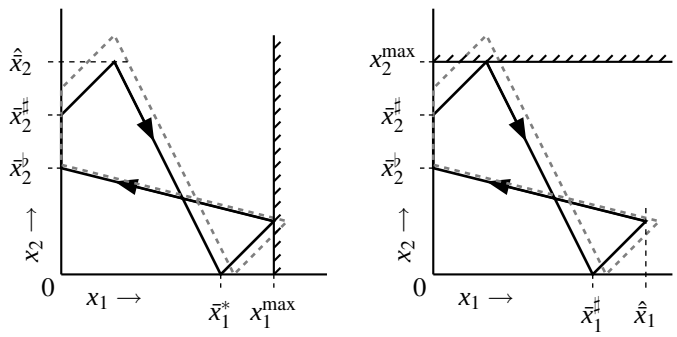

Figure 5: New optimal cycle, due to buffer capacity constraints. In dashed gray: original unconstrained optimal cycle.

Assume that $x_1^{\max} < \hat{x}_1$ or $x_2^{\max} < \hat{x}_2$. The optimal process cycle now has to be adjusted, to prevent violation of the buffer constraint. In Figure 5, buffer constraints have been drawn for type 1 jobs (left hand side graph) and type 2 jobs (right hand side). The coordinates of the optimal process cycle with buffer level constraints are denoted with bars ($\bar{}$). The original (unconstrained) process cycle is shown in dashed gray. Examining the geometrical implications of buffer level constraints on the optimal periodic orbit, the following expressions can easily be derived for the coordinates of the new optimal process cycle:

$$\bar{x}_1^\sharp = \min\left(x_1^{\max} - \lambda_1\sigma_{21},\ \lambda_1\left(\sigma_{12} + \frac{x_2^{\max}}{\mu_2 - \lambda_2}\right),\ x_1^\sharp\right) \tag{33a}$$

$$\hat{\bar{x}}_1 = \min\left(x_1^{\max},\ \lambda_1\left(\sigma + \frac{x_2^{\max}}{\mu_2 - \lambda_2}\right),\ \hat{x}_1\right) \tag{33b}$$

$$\bar{x}_2^\flat = \min\left(\lambda_2\left(\sigma_{21} + \frac{x_1^{\max}}{\mu_1 - \lambda_1}\right), \lambda_2\left(\sigma_{21} + \frac{\lambda_1}{\mu_1 - \lambda_1}\left(\sigma + \frac{x_2^{\max}}{\mu_2 - \lambda_2}\right)\right),\ x_2^\flat\right) \tag{33c}$$

$$\bar{x}_2^\sharp = \min\left(\frac{\mu_2 - \lambda_2}{\lambda_1}(x_1^{\max} - \lambda_1\sigma) - \lambda_2\sigma_{12},\ x_2^{\max} - \lambda_2\sigma_{12},\ x_2^\sharp\right) \tag{33d}$$

$$\hat{\bar{x}}_2 = \min\left(\frac{\mu_2 - \lambda_2}{\lambda_1}(x_1^{\max} - \lambda_1\sigma),\ x_2^{\max},\ \hat{x}_2\right). \tag{33e}$$

Note that if no slow-mode occurs in the unconstrained optimal process cycle, $\bar{x}_2^\flat = \bar{x}_2^\sharp = x_2^\flat = x_2^\sharp = x_2^*$, $\hat{\bar{x}}_2 = \hat{x}_2 = \hat{x}_2^*$, $\bar{x}_1^\sharp = x_1^\sharp = x_1^*$ and $\hat{\bar{x}}_1 = \hat{x}_1 = \hat{x}_1^*$. Another observation is that if the optimal process cycle with slow-mode changes due to the buffer capacity constraints, the duration of the slow-mode is shortened. In the minimum expressions in (33), the first term is 'active' (the lowest) if the buffer level constraint on type 1 jobs is most restrictive. The second term is 'active' if the constraint on type 2 jobs is most restrictive and the third term is 'active' if neither buffer constraint is restrictive on the optimal process cycle.
Before the state feedback controller is adjusted to accommodate the buffer level constraints, we first take a closer look at the (x_1, x_2)-plane. This plane can be divided into regions from which it is either possible or impossible to reach the desired steady state process cycle.

Lemma 5.1. *Regardless of the state feedback policy, the (x_1, x_2)-plane can be divided into regions from which it is impossible to reach the steady state process cycle when in a certain mode, see Figure 6. The regions marked with ①† and ②† indicate that if the trajectory enters that region processing type 1 or type 2 jobs respectively, the trajectory eventually becomes infeasible (i.e. a buffer constraint is violated). If the trajectory is on the bow tie curve shifted into the upper right corner of the (x_1, x_2)-plane, the trajectory stays there.*

Proof. It is easy to see that once the trajectory has crossed the dashed lines in Figure 6), a setup to the other job type causes the buffer constraint to be violated. For the regions in or above the upper-right bow tie, the proof is included in the feedback control law proof. □

The optimal process cycle for a switching server with two product types and finite buffer capacity has completely been defined. In addition, insight has been obtained about feasible and infeasible areas in the (x_1, x_2)-plane. By using the theory presented in [11] the feedback controller as presented in Proposition 4.1 can be modified by taking into account the buffer constraints.

Proposition 5.2. *A feedback which stabilizes a trajectory to the optimal process cycle if started from a feasible start point (see Figure 6) is*

$$(u_0, u_1, u_2) = \begin{cases} (①, \mu_1, 0) & \text{if } m = 1,\ x_0 = 0,\ x_1 > 0,\ x_2 < x_2^{\max} - \lambda_2\sigma_{12} \\ (❷, 0, 0) & \text{if } m = 1,\ x_0 = 0,\ x_1 > 0,\ x_2 \geq x_2^{\max} - \lambda_2\sigma_{12} \\ (①, \lambda_1, 0) & \text{if } m = 1,\ x_0 = 0,\ x_1 = 0,\ x_2 < \bar{x}_2^\sharp \\ (❷, 0, 0) & \text{if } m = 1,\ x_0 = 0,\ x_1 = 0,\ x_2 \geq \bar{x}_2^\sharp \\ (❷, 0, 0) & \text{if } m = 2,\ x_0 > 0 \\ (②, 0, \mu_2) & \text{if } m = 2,\ x_0 = 0,\ x_2 > 0,\ x_1 < x_1^{\max} - \lambda_1\sigma_{21} \\ (❶, 0, 0) & \text{if } m = 2,\ x_0 = 0,\ x_2 > 0,\ x_1 \geq x_1^{\max} - \lambda_1\sigma_{21} \\ (②, 0, \lambda_2) & \text{if } m = 2,\ x_0 = 0,\ x_2 = 0,\ x_1 < \bar{x}_1^\sharp \\ (❶, 0, 0) & \text{if } m = 2,\ x_0 = 0,\ x_2 = 0,\ x_1 \geq \bar{x}_1^\sharp \\ (❶, 0, 0) & \text{if } m = 1,\ x_0 > 0. \end{cases} \tag{34}$$

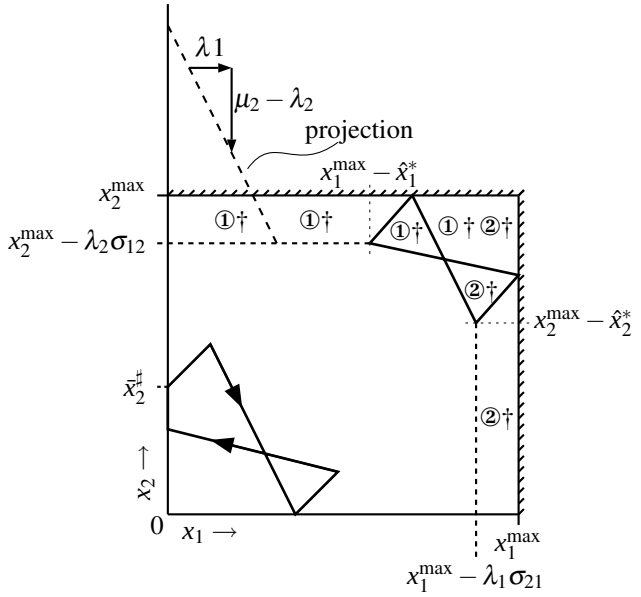

Figure 6: Feasible and infeasible regions for trajectories, subject to buffer level constraints.

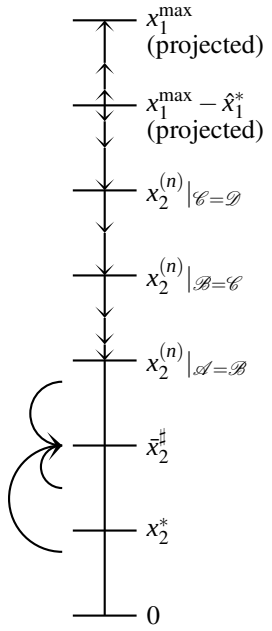

Figure 7: Evolution of $x_2^{(n)}$ at start of ❷.

Remark 5.3. An informal description of this feedback is given by:

- Mode 1: ① at μ_1 as long as $x_1 > 0$ and $x_2 < x_2^{\max} - \lambda_2\sigma_{12}$, then go to Mode 2.
- Mode 2: ① at λ_1 as long as $x_2 < \bar{x}_2^{\sharp}$, then go to Mode 3.
- Mode 3: perform ❷, after σ_{12} go to Mode 4.
- Mode 4: ② at μ_2 as long as $x_2 > 0$ and $x_1 < x_1^{\max} - \lambda_1\sigma_{21}$, then go to Mode 5.
- Mode 5: ② at λ_2 as long as $x_1 < \bar{x}_1^{\sharp}$, then go to Mode 6.
- Mode 6: perform ❶, after σ_{21} go to Mode 1.

Dependent on the state of the system, the controller is in one of these modes, which follows trivially from the mode description.

Proof. Similar to the proof of the case with infinite buffer capacities, we are interested where the $(n+1)^{\text{st}}$ start of ❷ takes place given from which coordinate the n^{th} start of ❷ took place. Setup for type 2 jobs can start at different places: on the axis between $(0,0)$ and $(0, x_2^{\max} - \lambda_2\sigma_{12})$ or on the line between $(0, x_2^{\max} - \lambda_2\sigma_{12})$ and $(x_1^{\max}, x_2^{\max} - \lambda_2\sigma_{12})$. The latter set of start points of ❷ is projected onto the vertical axis, in a way that the trajectory follows the same path in the feasible (x_1,x_2)-plane for both the unconstrained and constrained situation. The projection is shown in Figure 6. With a point $(x_1,\, x_2^{\max} - \lambda_2\sigma_{12})$ we associate the point $(0,\, x_2)$ where x_2 is given by:

$$x_2 = \frac{\mu_2 - \lambda_2}{\lambda_1} x_1 + (x_2^{\max} - \lambda_2\sigma_{12}) \tag{35}$$

and vice versa. Given $x_2^{(n)}$ we are interested in $x_2^{(n+1)}$. In case $x_2^{(n)}$ is chosen in such a way that we do not suffer from the buffer constraints, we obtain (28) again. However, in case one or two buffer constraints become active, the resulting $x_2^{(n+1)}$ is larger, since switching earlier makes the

system move along a line which is located higher in the (x_1, x_2)-plane, cf. Figure 6. For the case one constraint becomes active, we introduce auxiliary variable Z whose value depends on which constraint is active:

$$Z = \min \left(\underbrace{\frac{\mu_2 - \lambda_2}{\lambda_1}(x_1^{\max} - \lambda_1\sigma) - \hat{x}_2^*}_{\text{is the smallest if } x_1^{\max} \text{ active}}, \underbrace{\frac{\mu_1 - \lambda_1}{\lambda_2}(x_2^{\max} - \lambda_2\sigma) - \hat{x}_1^*}_{\text{is the smallest if } x_2^{\max} \text{ active}} \right). \tag{36}$$

Four situations can be distinguished:

$\mathscr{A}$**:** no active constraints, iteration as in (27);

$\mathscr{B}$**:** no active constraints, iteration as in (25);

$\mathscr{C}$**:** one active buffer constraint during iteration;

$\mathscr{D}$**:** two active buffer constraints during iteration.

The endpoint of one iteration using the feedback law of Proposition 5.2 now becomes

$$x_2^{(n+1)} = \max \begin{pmatrix} \bar{x}_2^\sharp & (\mathscr{A}) \\ \frac{\rho_1\rho_2}{(1-\rho_1)(1-\rho_2)}(x_2^{(n)} - x_2^*) + x_2^* & (\mathscr{B}) \\ x_2^{(n)} - \frac{1-\rho_1-\rho_2}{(1-\rho_1)(1-\rho_2)} \cdot Z & (\mathscr{C}) \\ \frac{(1-\rho_1)(1-\rho_2)}{\rho_1\rho_2}\left(x_2^{(n)} - [x_1^{\max} - \hat{x}_1^*]\right) + [x_1^{\max} - \hat{x}_1^*] & (\mathscr{D}) \end{pmatrix} \tag{37}$$

where the calligraphic capital refers to one of the four situations. The evolution of an arbitrary point $(0,\, x_2^{(n)})$ along the $x_1 = 0$ axis where ❷ starts can now be visualized, see Figure 7. The arrows indicate the direction in which $x_2^{(n)}$ evolves. The distance between the arrows is a measure for the rate of the evolution. First consider residing in region $\mathscr{D}$. Note that for $x_2^{(n)} > x_1^{\max} - \hat{x}_1^*$ (projected), i.e. we start right from the upper right bow tie (Figure 6), we have divergence, since $\frac{(1-\rho_1)(1-\rho_2)}{\rho_1\rho_2} > 1$, cf. (30). When starting on the upper right bow tie, we can stay on it (if we switch earlier than planned by this curve, the trajectory becomes infeasible, which can be seen on geometrical grounds in Figure 6). This completes the proof of Lemma 5.1. For $x_2^{(n)} < x_1^{\max} - \hat{x}_1^*$ we move away from the upper right bow tie in the correct direction, i.e. towards the bottom left in Figure 6, and we leave region $\mathscr{D}$ after a finite number of steps. If we are in $\mathscr{C}$ we also move in the correct direction with equidistant jumps and therefore after a finite number of steps leave region $\mathscr{C}$. So after a finite number of steps we are in either region $\mathscr{A}$ or region $\mathscr{B}$. From the proof of Proposition 4.1, convergence to $\bar{x}_2^\sharp$ follows. □

The switching server with two job types and setup times now has a suitable controller, for which convergence has been proven. In the next section, the controller is implemented in a case study, where the hybrid fluid model has been replaced with a discrete event simulation and with stochastic process times.

6 Case study: controller implementation in discrete event workstation

The state feedback controller as presented in Proposition 5.2 has been implemented in a case study. The derivation of the optimal process cycle had been based upon the hybrid fluid model, as presented in (1)–(5). The controller uses measurements of the state to determine its control actions. Therefore, in some sense, the controller should be robust for disturbances in the model. In this case study we implement the controller in a discrete event simulation of a switching server,

where the hybrid fluid model has been replaced with a discrete event model, specified in language χ, see [1]. The parameter settings and buffer level constraints used for the simulation are presented in Table 1. The cost weighing factors c_1 and c_2 are equal to 1.
The condition for a slow-mode is evaluated:

$$c_1\lambda_1(\rho_1+\rho_2)+(c_2\lambda_2-c_1\lambda_1)(1-\rho_2)=-\frac{23}{24}<0 \tag{38}$$

so a slow-mode occurs for type 1 jobs. Note that the slow-mode occurs for the job type with highest $c\lambda$ index and lowest $c\mu$ index, (cf. [5]). The duration of the slow-mode is computed with $\tau_1^{\lambda}=\alpha_1\sigma$ with α_1 the positive real root from (19): $\alpha_1=1/4$. This completely defines the optimal process cycle: $\vec{x}_2^{\flat}=15$, $\vec{x}_2^{\sharp}=18$, $\hat{\vec{x}}_2=24$, $\vec{x}_1^{\sharp}=27$ and $\hat{\vec{x}}_1=45$. The buffer level capacity constraints did not influence the optimal periodic orbit.
The initial state of the system is also given in Table 1. At the start of the simulation, the server processes type 2 jobs. Results of the simulation are shown in Figure 8. The trajectory goes from light-gray to black, for better visual understanding. The first and second setup that take place, are invoked by the buffer level constraint. That is why the buffer is not emptied before a setup to the other job type takes place. After three setups, the trajectory touches the $x_1=0$ axis below $\vec{x}_2^{\sharp}$. Then a slow-mode is performed until the buffer level of type 2 jobs reaches $\vec{x}_2^{\sharp}$. From that point, the trajectory stays on the optimal periodic orbit.

Table 1: Parameter settings for discrete event case study with controller implementation.

λ_1:	9 jobs/hr.	$x_1^{\max}$:	70 jobs
λ_2:	3 jobs/hr.	$x_2^{\max}$:	40 jobs
μ_1:	24 jobs/hr.	$x_0(0)$:	0 hrs.
μ_2:	27 jobs/hr.	$x_1(0)$:	50 jobs
σ_{12}:	2 hrs.	$x_2(0)$:	20 jobs
σ_{21}:	2 hrs.	$m(0)$:	2

The optimal process cycle has been defined for a single switching server with two job types. However, in many industrial environments and communication networks, workstations are coupled and form a flowline. The flow of jobs through a series of workstations is similar to the flow of jobs through one workstation, but what control actions are needed to make a flowline behave as desired? The remainder of this paper focuses on control of a flowline of switching servers.

7 Flowline of switching servers: characteristics and dynamics

We want to control a flowline of switching servers in such a way that the number of jobs in the flowline is minimal. In general, an optimal process cycle for such a flowline is hard to determine, but based on the results of Section 3, we know a lower bound on the work in process level for a flowline. The remainder of this paper focuses on achieving this lower bound on the work in process level for the entire flowline. What are the conditions / requirements for the workstations to achieve this lower bound? Can we characterize the optimal process cycle? Is there a unique solution for this optimization problem or are there more solutions? In this section, these questions are addressed.
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 contents are denoted by $x_i^j(t)$, e.g. $x_1^B(t)$ equals the number of jobs of type 1 that is stored in workstation B at time t. Jobs arrive at workstation A with constant rates λ_1 and λ_2 for type 1 and type 2 respectively. The maximum process rates are μ_i^j, with $i\in\{1,2\}$ and $j\in\{A,B\}$. Switching from

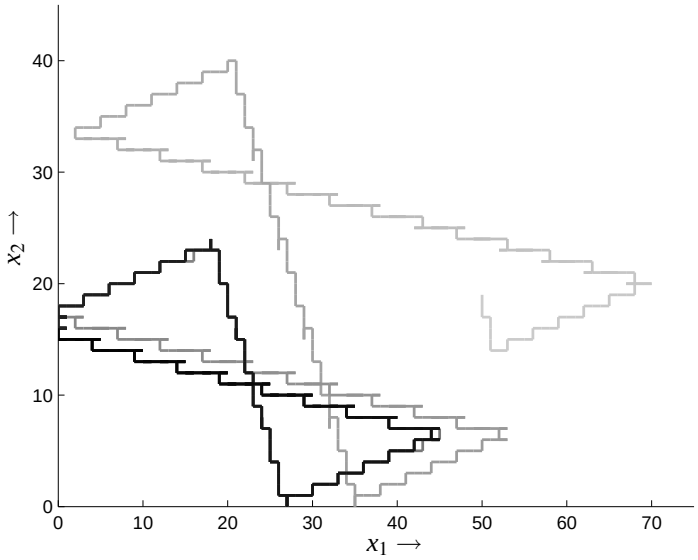

Figure 8: Simulation results of controller implementation.

processing type 1 to type 2 jobs takes σ_{12}^j time units and σ_{21}^j vice versa. The system is schematically shown in Figure 9.

For stability reasons, the total utilization must not exceed 1 for each server: $\sum_i \rho_i^j < 1 \; \forall \; j \in \{A,B\}$ with $\rho_i^j = \lambda_i/\mu_i^j$. 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 $A+B$.

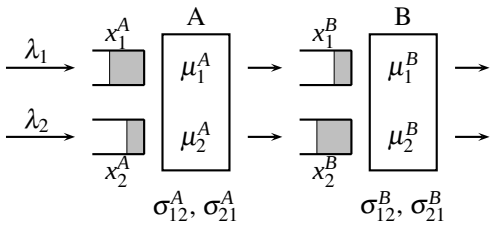

Figure 9: Switching server flowline overview.

The state and input vector for this flowline are similar to those of the single switching server example. The state consists of the buffer levels, the remaining process times and the modes of the servers:

$$\mathbf{x} = \begin{bmatrix} x_1^A & x_2^A & x_1^B & x_2^B & x_0^A & x_0^B & m^A & m^B \end{bmatrix}^{\mathrm{T}} \in \mathbb{R}_+^6 \times \{1,2\}^2 . \tag{39}$$

The input vector consists of the rates at which the servers are processing the job types and the action that has to be performed by the servers:

$$\mathbf{u} = \begin{bmatrix} u_0^A & u_0^B & u_1^A & u_2^A & u_1^B & u_2^B \end{bmatrix}^{\mathrm{T}} \in \{❶,①,❷,②\}^2 \times \mathbb{R}_+^4 . \tag{40}$$

Possible actions of the servers are:

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

and similar to the single switching server, the inputs are constrained by the state at each time instant:

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

These constraints mean that if a server is busy with a setup, no jobs can be processed and 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.

The discrete and continuous dynamics of the flowline look similar to the dynamics of the single switching server and need no further explanation:

$$x_0^j(t) := \sigma_{21}^j,\ m^j(t) := 1 \text{ for } u_0^j(t) = ❶ \text{ and } m^j(t) = 2 \tag{41a}$$

$$x_0^j(t) := \sigma_{12}^j,\ m^j(t) := 2 \text{ for } u_0^j(t) = ❷ \text{ and } m^j(t) = 1 \tag{41b}$$

$$\dot{x}_0^j(t) = \begin{cases} -1 & \text{for } u_0^j(t) \in \{❶,❷\} \\ 0 & \text{for } u_0^j(t) \in \{①,②\} \end{cases} \tag{41c}$$

$$\dot{x}_1^A(t) = \lambda_1 - u_1^A(t) \tag{41d}$$

$$\dot{x}_2^A(t) = \lambda_2 - u_2^A(t) \tag{41e}$$

$$\dot{x}_1^B(t) = u_1^A(t) - u_1^B(t) \tag{41f}$$

$$\dot{x}_2^B(t) = u_2^A(t) - u_2^B(t). \tag{41g}$$

The goal is to minimize the time averaged weighted work in process level of the flowline. The cost function J is defined as:

$$J = \lim_{t\to\infty} \frac{1}{t} \int_0^t g_1\big(x_1^A(s) + x_1^B(s)\big) + g_2\big(x_2^A(s) + x_2^B(s)\big) \mathrm{d}s \tag{42}$$

with $g_i : \mathbb{R}_+ \to \mathbb{R}_+$ strictly increasing functions. An important observation is that the switch policy of workstation A does not affect the work in process level. Workstation A just moves work from A to B, but the work remains in the flowline. Workstation B actually removes work from the flowline. The switching policy of B therefore determines the wip level of the whole system. In general, the most downstream workstation of a flowline determines the wip in the system. From Section 3 the optimal process cycle of a single switching server is known. If it is possible to have the most downstream workstation process at its optimal cycle and have the other workstations make this possible, then optimal behavior for the complete flowline has been achieved. The buffer levels of a job type can then virtually be added. For these lumped buffer levels, the optimal cycle must be performed. The switching policy of the upstream workstations must then accommodate these virtual lumped buffer levels. This idea is schematically shown in Figure 10 (cf. Figure 1).

In this paper we consider the class of flowlines for which it is possible to achieve the wip level of a single switching server. We investigate the conditions on the upstream workstations to make this possible, characterize the class of workstations explicitly and define the optimal process cycle. In Section 9 a state feedback controller is proposed for the flowline with two workstations.

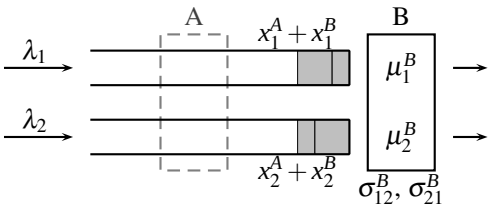

Figure 10: General idea of flowline behaving as single switching server.

Section 10 is an implementation of this controller in a discrete event simulation with disturbances on the arrival times and process times.

8 Optimal process cycle of flowline: conditions and derivation

The general form of an optimal process cycle for a single switching server is shown in Figure 3 and made explicit in Proposition 3.9. This optimal cycle must be performed by workstation B and must also become the optimal cycle with respect to wip levels for $A+B$. If workstation A has to switch between job types in such a way that it makes $A+B$ behave like B stand alone, then some observations can be made, which are given below. (Recall the notational aspects: τ is a period in which jobs can be processed, the subscript indicates the job type. Superscript μ means processing at maximum rate, whereas superscript λ means processing at incoming rate. Capital A or B is the workstation identification.)

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 (see the conditions in Section 3). 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:
$$\theta_i^- + \tau_i^{\lambda B} + \theta_i^+ = \tau_i^{\lambda A}, \quad i \in \{1,2\}. \tag{43}$$
The overlap requirement yields:
$$\theta_1^- \geq 0;\ \ \theta_1^+ \geq 0;\ \ \theta_2^- \geq 0;\ \ \theta_2^+ \geq 0. \tag{44}$$

2. Without loss of generality, let time $t = 0$ be the start of σ_{21}^B. 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. The period of the process cycle of A must be equal to the period of B. This period is denoted by T.

4. From observations 1–3 follows:
$$\theta_2^+ + \sigma_{21}^A + \tau_1^{\mu A} + \theta_1^- = \sigma_{21}^B + \tau_1^{\mu B} \tag{45}$$
$$\theta_1^+ + \sigma_{12}^A + \tau_2^{\mu A} + \theta_2^- = \sigma_{12}^B + \tau_2^{\mu B}. \tag{46}$$

5. Buffer levels are not allowed to become negative. Therefore, if $\tau_i^{\mu B}$ starts earlier than $\tau_i^{\mu A}$, 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:
$$\mu_i^B(\tau_i^{\mu B} - \theta_i^- - \tau_i^{\mu A}) \leq \lambda_i \theta_i^+ \tag{47}$$

or written differently:

$$\sigma_{21}^A + \theta_2^+ - \sigma_{21}^B \leq \rho_1^B \theta_1^+ \tag{48}$$

$$\sigma_{12}^A + \theta_1^+ - \sigma_{12}^B \leq \rho_2^B \theta_2^+. \tag{49}$$

This restriction is also valid if $\tau_i^{\mu B}$ starts after $\tau_i^{\mu A}$ started, since then the left-hand sides of (47)–(49) become negative, while the right-hand sides are always positive. Therefore, these constraints may always be required.

6. 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 (with $i \in \{1,2\}$):

$$\mu_i^A \tau_i^{\mu A} + \lambda_i \tau_i^{\lambda A} = \mu_i^B \tau_i^{\mu B} + \lambda_i \tau_i^{\lambda B} = \lambda_i T. \tag{50}$$

These observations have been summarized in Figure 11. The process cycles for workstations A and B are presented. Note that the timeline for B is the same as in Figure 3. The overlapping slow-modes (observation 1) are clearly visible. Note that the situation which observation 5 refers to is visible for type 1 jobs in the figure: the amount of jobs workstation A processes during θ_1^+ must at least equal the number of jobs that are processed by B before $\tau_1^{\mu A}$ starts again.

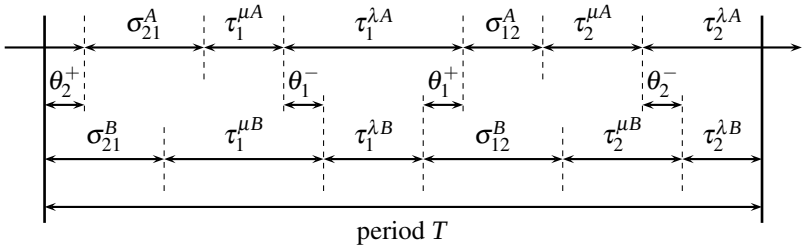

Figure 11: The period of one cycle, T, divided into subsequent phases.

With the given observations and Figure 11 it is possible to derive conditions for server A which must be obeyed to make $A+B$ behave like B stand-alone with respect to work in process levels.

Remark 8.1. Observations 1–6 are also applicable for flowlines with more than two servers, e.g. flowline $A+B+C$. In that case, first 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. Similar reasoning goes for larger flowlines.

Proposition 8.2. *Workstation A can make flowline $A+B$ perform like B stand-alone with respect to work in process levels if and only if:*

$$R_2 \left[\tau_1^{\mu B} + \tau_2^{\lambda B} + \sigma_{21}^B - \sigma_{21}^A - T + R_1(\tau_1^{\lambda B} - T) \right] + \tau_2^{\mu B} + \sigma_{12}^B - \sigma_{12}^A \geq 0 \tag{51}$$

and

$$R_1 \left[\tau_2^{\mu B} + \tau_1^{\lambda B} + \sigma_{12}^B - \sigma_{12}^A - T + R_2(\tau_2^{\lambda B} - T) \right] + \tau_1^{\mu B} + \sigma_{21}^B - \sigma_{21}^A \geq 0 \tag{52}$$

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

Proof. The proof consists of two parts. First, if a periodic orbit of A makes $A+B$ behave like B stand-alone, it fulfills (43)–(50). 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:

$$\tau_i^{\mu A} + \theta_i^- \geq \rho_i^A (T - \theta_i^+ - \tau_i^{\lambda B}). \tag{53}$$

Note that the mass conservation requirement has been replaced with inequality constraints. Combining this result with (45) and (46) results in:

$$\sigma_{21}^{B} - \sigma_{21}^{A} + \tau_1^{\mu B} - \theta_2^{+} \geq \rho_1^{A}(T - \theta_1^{+} - \tau_1^{\lambda B}) \tag{54}$$

which can be rewritten as:

$$\rho_1^{A}(\theta_1^{+} + \tau_1^{\lambda B} - T) + \tau_1^{\mu B} - \theta_2^{+} \geq \sigma_{21}^{A} - \sigma_{21}^{B} \tag{55}$$

and similar for the other job type:

$$\rho_2^{A}(\theta_2^{+} + \tau_2^{\lambda B} - T) + \tau_2^{\mu B} - \theta_1^{+} \geq \sigma_{12}^{A} - \sigma_{12}^{B}. \tag{56}$$

From (50) we know that:

$$\tau_i^{\mu B} - \rho_i^{B}(T - \tau_i^{\lambda B}) = 0. \tag{57}$$

Adding this to (48) and (49) respectively results in:

$$\rho_1^{B}(\theta_1^{+} + \tau_1^{\lambda B} - T) + \tau_1^{\mu B} - \theta_2^{+} \geq \sigma_{21}^{A} - \sigma_{21}^{B} \tag{58}$$

$$\rho_2^{B}(\theta_2^{+} + \tau_2^{\lambda B} - T) + \tau_2^{\mu B} - \theta_1^{+} \geq \sigma_{12}^{A} - \sigma_{12}^{B} \tag{59}$$

which looks rather similar to (55) and (56). Combining the results (55), (58) and (56), (59) results in:

$$\max(\rho_1^{A}, \rho_1^{B})(\theta_1^{+} + \tau_1^{\lambda B} - T) + \tau_1^{\mu B} - \theta_2^{+} \geq \sigma_{21}^{A} - \sigma_{21}^{B} \tag{60a}$$

$$\max(\rho_2^{A}, \rho_2^{B})(\theta_2^{+} + \tau_2^{\lambda B} - T) + \tau_2^{\mu B} - \theta_1^{+} \geq \sigma_{12}^{A} - \sigma_{12}^{B}. \tag{60b}$$

These inequalities, together with $\theta_1^{+} \geq 0$ and $\theta_2^{+} \geq 0$, enclose a feasible area in the $(\theta_2^{+},\ \theta_1^{+})$-plane (Figure 12).

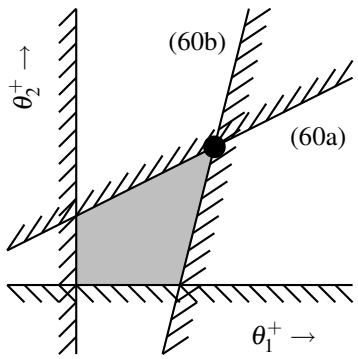

Figure 12: $(\theta_2^{+},\ \theta_1^{+})$-plane with feasible area (gray).

The intersection point of the two linear borders defined by (60) lies in the pos-pos quarter of this plane. The intersection point is given by:

$$\theta_1^{+} = \frac{R_2\left[\tau_1^{\mu B} + \tau_2^{\lambda B} + \sigma_{21}^{B} - \sigma_{21}^{A} - T + R_1(\tau_1^{\lambda B} - T)\right] + \tau_2^{\mu B} + \sigma_{12}^{B} - \sigma_{12}^{A}}{1 - R_1 \cdot R_2} \tag{61a}$$

$$\theta_2^{+} = \frac{R_1\left[\tau_2^{\mu B} + \tau_1^{\lambda B} + \sigma_{12}^{B} - \sigma_{12}^{A} - T + R_2(\tau_2^{\lambda B} - T)\right] + \tau_1^{\mu B} + \sigma_{21}^{B} - \sigma_{21}^{A}}{1 - R_1 \cdot R_2} \tag{61b}$$

which are only both non-negative if (51) and (52) are fulfilled.
The second part of the proof is to show that conditions (51) and (52) are sufficient to guarantee that A can make $A+B$ behave like B stand-alone. For arbitrary θ_i^{+}, the values for the processing

intervals $\tau_i^{\mu A}$ and $\tau_i^{\lambda A}$ can be computed. These can be solved from (43), (45), (46) and (50), guaranteeing mass conservation and the requirement that the process cycle of A fits in the process cycle of B for both job types:

$$\tau_1^{\mu A} = \frac{\rho_1^A(\sigma_{12}^B + \tau_2^{\mu B} + \tau_2^{\lambda B} - \theta_1^+ + \theta_2^+ + \sigma_{21}^A)}{1-\rho_1^A} \tag{62}$$

$$\theta_1^- = \frac{\sigma_{21}^B - \theta_2^+ - \sigma_{21}^A + \tau_1^{\mu B} + \rho_1^A(\theta_1^+ - \tau_1^{\mu B} - \tau_2^{\mu B} - \tau_2^{\lambda B} - \sigma_{21}^B - \sigma_{12}^B)}{1-\rho_1^A} \tag{63}$$

$$\tau_2^{\mu A} = \frac{\rho_2^A(\tau_1^{\mu B} + \tau_1^{\lambda B} + \sigma_{21}^B - \theta_2^+ + \theta_1^+ + \sigma_{12}^A)}{1-\rho_2^A} \tag{64}$$

$$\theta_2^- = \frac{\sigma_{12}^B - \theta_1^+ - \sigma_{12}^A + \tau_2^{\mu B} + \rho_2^A(\theta_2^+ - \tau_1^{\mu B} - \tau_2^{\mu B} - \tau_1^{\lambda B} - \sigma_{21}^B - \sigma_{12}^B)}{1-\rho_2^A} \tag{65}$$

$$\tau_1^{\lambda A} = \theta_1^- + \tau_1^{\lambda B} + \theta_1^+ \tag{66}$$

$$\tau_2^{\lambda A} = \theta_2^- + \tau_2^{\lambda B} + \theta_2^+. \tag{67}$$

If θ_1^+ and θ_2^+ are chosen in a way that $\tau_1^{\mu A}$, θ_1^-, $\tau_2^{\mu A}$ and θ_2^- are non-negative and moreover, the buffer levels do not become negative ((48) and (49)), then a feasible solution has been found. From (66) and (67), it can easily be seen that $\tau_1^{\lambda A}$ and $\tau_2^{\lambda A}$ are also non-negative then. If (51) and (52) hold, then (61) gives feasible values for θ_1^+ and θ_2^+, since they are non-negative. In (61) the period length T and the process intervals $\tau_i^{\mu B}$ are substitued with:

$$\tau_i^{\mu B} = \rho_i^B(T - \tau_i^{\lambda B}) \quad \text{(follows from (50))} \tag{68}$$

$$T = \tau_1^{\mu B} + \tau_2^{\mu B} + \tau_1^{\lambda B} + \tau_2^{\lambda B} + \sigma_{12}^B + \sigma_{21}^B. \tag{69}$$

Substituting (61) in (62)–(65) gives expressions for $\tau_1^{\mu A}$, $\tau_2^{\mu A}$, θ_1^- and θ_2^-:

$$\begin{aligned}\tau_1^{\mu A} = \frac{\rho_1^A(1-R_1)}{(1-\rho_1^A)(1-R_1R_2)(1-\rho_1^B-\rho_2^B)}\Big[&(1-\rho_1^B-\rho_2^B)\sigma_{12}^A \\ &+ R_2(1-\rho_1^B-\rho_2^B)\sigma_{21}^A + (R_2(1-\rho_1^B)+\rho_1^B)\sigma_{12}^B + (1-\rho_2^B+R_2\rho_2^B)\sigma_{21}^B \\ &+ (R_2(1-\rho_1^B)+\rho_1^B\rho_2^B(1-R_2))\tau_1^{\lambda B} + (\rho_1^B\rho_2^B(R_2-1)+1-\rho_2^B)\tau_2^{\lambda B}\Big]\end{aligned} \tag{70}$$

$$\begin{aligned}\tau_2^{\mu A} = \frac{\rho_2^A(1-R_2)}{(1-\rho_2^A)(1-R_1R_2)(1-\rho_1^B-\rho_2^B)}\Big[&R_1(1-\rho_1^B-\rho_2^B)\sigma_{12}^A \\ &+ (1-\rho_1^B-\rho_2^B)\sigma_{21}^A + (1-\rho_1^B+R_1\rho_1^B))\sigma_{12}^B + (R_1(1-\rho_2^B)+\rho_2^B)\sigma_{21}^B \\ &+ (\rho_1^B\rho_2^B(R_1-1)+1-\rho_1^B)\tau_1^{\lambda B} + (R_1(1-\rho_2^B)+\rho_1^B\rho_2^B(1-R_1))\tau_2^{\lambda B}\Big]\end{aligned} \tag{71}$$

$$\begin{aligned}\theta_1^- = \frac{R_1-\rho_1^A}{(1-\rho_1^A)(1-R_1R_2)(1-\rho_1^B-\rho_2^B)}\Big[&(1-\rho_1^B-\rho_2^B)\sigma_{12}^A \\ &+ R_2(1-\rho_1^B-\rho_2^B)\sigma_{21}^A + (\rho_1^B+R_2(1-\rho_1^B))\sigma_{12}^B + (1-\rho_2^B+R_2\rho_2^B)\sigma_{21}^B \\ &+ (R_2(1-\rho_1^B)+\rho_1^B\rho_2^B(1-R_2))\tau_1^{\lambda B} + (\rho_1^B\rho_2^B(R_2-1)+1-\rho_2^B)\tau_2^{\lambda B}\Big]\end{aligned} \tag{72}$$

$$\begin{aligned}\theta_2^- = \frac{R_2-\rho_2^A}{(1-\rho_2^A)(1-R_1R_2)(1-\rho_1^B-\rho_2^B)}\Big[&R_1(1-\rho_1^B-\rho_2^B)\sigma_{12}^A \\ &+ (1-\rho_1^B-\rho_2^B)\sigma_{21}^A + (1-\rho_1^B+R_1\rho_1^B)\sigma_{12}^B + (\rho_2^B+R_1(1-\rho_2^B))\sigma_{21}^B \\ &+ (\rho_1^B\rho_2^B(R_1-1)+1-\rho_1^B)\tau_1^{\lambda B} + (R_1(1-\rho_2^B)+\rho_1^B\rho_2^B(1-R_1))\tau_2^{\lambda B}\Big]\end{aligned} \tag{73}$$

Each expression in (70)–(73) consists of six terms. Recalling that $0 \leq \rho_i^j < 1$, $\sum_i \rho_i^j < 1$ and $R_i = \max(\rho_i^A,\ \rho_i^B)$, one can easily verify that all six terms in each expression are non-negative. Finally, (48) and (49) are checked, resulting in:

$$\frac{R_1-\rho_1^B}{(1-R_1R_2)(1-\rho_1^B-\rho_2^B)}\Big[-(1-\rho_1^B-\rho_2^B)\sigma_{12}^A - R_2(1-\rho_1^B-\rho_2^B)\sigma_{21}^A \\ -(\rho_1^B+R_2(1-\rho_1^B))\sigma_{12}^B-(1-\rho_2^B(1-R_2))\sigma_{21}^B \\ -(\rho_1^B\rho_2^B(1-R_2)+R_2(1-\rho_1^B))\tau_1^{\lambda B}-(\rho_1^B\rho_2^B(R_2-1)+1-\rho_2^B)\tau_2^{\lambda B}\Big]\leq 0 \quad (74)$$

and

$$\frac{R_2-\rho_2^B}{(1-R_1R_2)(1-\rho_1^B-\rho_2^B)}\Big[-R_1(1-\rho_1^B-\rho_2^B)\sigma_{12}^A-(1-\rho_1^B-\rho_2^B)\sigma_{21}^A \\ -(1-\rho_1^B(1-R_1))\sigma_{12}^B-(\rho_2^B+R_1(1-\rho_2^B))\sigma_{21}^B \\ -(\rho_1^B\rho_2^B(R_1-1)+1-\rho_1^B)\tau_1^{\lambda B}-(\rho_1^B\rho_2^B(1-R_1)+R_1(1-\rho_2^B))\tau_2^{\lambda B}\Big]\leq 0. \quad (75)$$

For these two inequalities, it can easily be verified that all terms at the left hand side are non-positive, so the inequalities hold. With these results, it has been proven that (51) and (52) are sufficient conditions to guarantee that workstation A can make $A+B$ perform like B stand-alone with respect to work in process levels. □

When more than one upstream workstation is present, similar conditions can be derived for these workstations. Each additional workstation adds two constraints similar to (51) and (52) to Proposition 8.2. Notice, however, that for larger flowlines, checking the conditions for all workstations in upstream direction might involve some iterations. In general, there is some freedom in the choice of process interval lengths within all conditions as described in the observations earlier in this section (the gray feasible area in Figure 12). The one choice may give feasible results for other upstream servers, whereas the other choice might give infeasible results. Eliminating this freedom and developing explicit relations has not been investigated in this study. To avoid this issue, a way to find feasible trajectories for upstream servers all at once given the parameters (μ and σ) is to cast the problem into a linear program with design variables τ_1^μ, τ_1^λ, τ_2^μ, τ_2^λ, θ_1^+ and θ_2^+ for all servers. All constraints (43)–(50) are linear in the design variables. Any arbitrary objective function results in a feasible solution, (unless the problem is infeasible according to (51) and (52) and corresponding conditions for other workstations). In this way, for larger flowlines, feasible trajectories can be found relatively easy. The linear program solver can also be used to check if a feasible solution exists at all.
The desired (optimal) process cycle has been defined for the entire flowline now and conditions for the upstream workstations have been derived. In the next section, we look for a controller that steers the state $\mathbf{x}(t)$ to the desired periodic orbits from any arbitrary initial state $\mathbf{x}(0)$.

9 State feedback controller for switching server flowline

A state feedback controller that brings any arbitrary trajectory to the desired periodic orbits as defined in sections 3 and 8 is presented in this section. The controller can be obtained using the ideas presented in [11].

Lemma 9.1. *Under conditions* (51) *and* (52)*, i.e. workstation A can make $A+B$ behave like B stand-alone, the following inequality holds: $R_1+R_2<1$.*

Proof. Choose θ_i^+ in a way that they are as small as possible, but within the feasible area of

Figure 12. The values for θ_i^+ are then:

$$\theta_1^+ = \max(0,\, R_2(\tau_2^{\lambda B} - T) + \tau_2^{\mu B} - \sigma_{12}^A + \sigma_{12}^B) \tag{76a}$$

$$\theta_2^+ = \max(0,\, R_1(\tau_1^{\lambda B} - T) + \tau_1^{\mu B} - \sigma_{21}^A + \sigma_{21}^B). \tag{76b}$$

Rewriting (51) and (52) gives:

$$(1-R_1-R_2)(\tau_1^{\mu B} + \tau_2^{\mu B} + \sigma_{12}^B) - (\sigma_{21}^A + \sigma_{12}^A + R_1(\tau_2^{\lambda B} + \sigma_{21}^B) + R_2\tau_2^{\lambda B}) \geq (1-R_2)(R_1(\tau_1^{\lambda B} - T) + \tau_1^{\mu B} - \sigma_{21}^A) \tag{77a}$$

$$(1-R_1-R_2)(\tau_1^{\mu B} + \tau_2^{\mu B} + \sigma_{21}^B) - (\sigma_{21}^A + \sigma_{12}^A + R_2(\tau_1^{\lambda B} + \sigma_{12}^B) + R_1\tau_1^{\lambda B}) \geq (1-R_1)(R_2(\tau_2^{\lambda B} - T) + \tau_2^{\mu B} - \sigma_{12}^A). \tag{77b}$$

Suppose that $R_1 + R_2 \geq 1$, implying that the left hand sides of (77) are negative. Combining this with (76), this results in $\theta_1^+ < \sigma_{12}^B$ and $\theta_2^+ < \sigma_{21}^B$, meaning that A and B are never working at different job types simultaneously (cf. Figure 11) if working at the desired process cycles. A consequence is that a *virtual station* can be put between workstations A and B, with maximum process rates $\min(\mu_1^A, \mu_i^B)$ and $\min(\mu_2^A, \mu_2^B)$, which does not influence the process cycles of A and B (the virtual station paradigm has been elaborated in [7, Ch.8]). In Figure 13, the virtual station (VS) has been put between the timelines of A and B. During τ_i^{VS}, the virtual station processes type i jobs (with maximum process rate $\min(\mu_i^A, \mu_i^B)$). The process intervals τ_i^{VS} start with the first start of $\tau_i^{\mu j}$ and end at the same moment as θ_i^+. For reasons of stability, the utilization of the virtual station must not exceed 1: $R_1 + R_2 < 1$. This is in contradiction with our assumption, so under conditions (51) and (52), $R_1 + R_2 < 1$. □

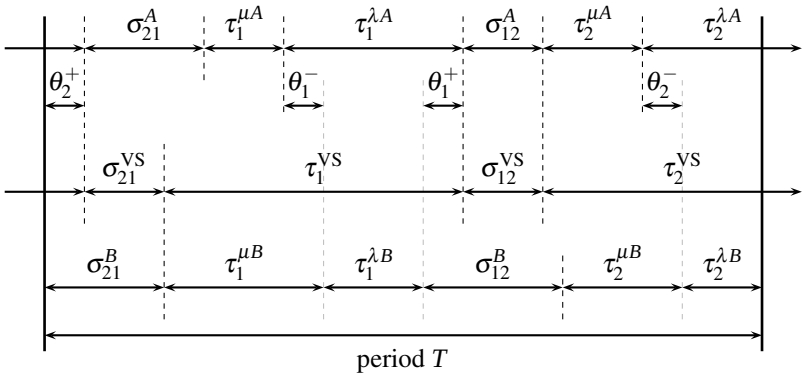

Figure 13: Virtual station between workstations A and B.

Proposition 9.2. *The following state feedback controller steers the system to the desired (optimal) periodic orbits, from any arbitrary initial state* $\mathbf{x}(0)$*:*
Initially, the workstations have to get into the same mode $\mathbf{m} = (m^A,\, m^B)$*. Without loss of generality, we adjust the mode of A to equal the mode of B (if necessary). This is done by means of the following feedback:*

$$\begin{pmatrix} u_0^A & u_0^B & u_1^A & u_2^A & u_1^B & u_2^B \end{pmatrix} = \begin{cases} (❶,*,0,0,*,*) & \text{if } \mathbf{m} = (2,1) \\ (❷,*,0,0,*,*) & \text{if } \mathbf{m} = (1,2) \end{cases} \tag{78}$$

where $*$ means 'unchanged value'. Immediately after this action (if necessary), the following feedback is used:

$$\begin{pmatrix} u_0^A \\ u_0^B \\ u_1^A \\ u_2^A \\ u_1^B \\ u_2^B \end{pmatrix}^{\mathrm{T}} = \begin{cases} (①,①,\mu_1^A,0,\mu_1^B,0) & \text{if } \mathbf{m}=(1,1),\ x_0^A=0,\ x_0^B=0,\ x_1^A>0,\ x_1^B>0 \\ (①,①,\mu_1^A,0,\min(\mu_1^A,\mu_1^B),0) & \text{if } \mathbf{m}=(1,1),\ x_0^A=0,\ x_0^B=0,\ x_1^A>0,\ x_1^B=0 \\ (①,①,\lambda_1,0,\mu_1^B,0) & \text{if } \mathbf{m}=(1,1),\ x_0^A=0,\ x_0^B=0,\ x_1^A=0,\ x_1^B>0 \\ (①,①,\lambda_1,0,\lambda_1,0) & \text{if } \mathbf{m}=(1,1),\ x_0^A=x_0^B=x_1^A=x_1^B=0,\ x_2^A<x_2^{A\sharp} \\ (①,②,\lambda_1,0,0,\mu_2^B) & \text{if } \mathbf{m}=(1,2),\ x_0^A=0,\ x_0^B=0,\ x_1^A=0,\ x_2^B>0 \\ (①,②,\lambda_1,0,0,0) & \text{if } \mathbf{m}=(1,2),\ x_0^A=0,\ x_0^B=0,\ x_1^A=0,\ x_2^B=0 \\ (①,❶,\mu_1^A,0,0,0) & \text{if } \mathbf{m}=(1,1),\ x_0^A=0,\ x_0^B>0,\ x_1^A>0 \\ (①,❶,\lambda_1,0,0,0) & \text{if } \mathbf{m}=(1,1),\ x_0^A=0,\ x_0^B>0,\ x_1^A=0 \\ (①,❷,\lambda_1,0,0,0) & \text{if } \mathbf{m}=(1,2),\ x_0^A=0,\ x_0^B>0,\ x_1^A=0,\ x_1^B<x_1^{B\sharp} \\ (①,❷,\lambda_1,0,0,0) & \text{if } \mathbf{m}=(1,1),\ x_0^A=x_0^B=x_1^A=x_1^B=0,\ x_2^A\geq x_2^{A\sharp} \\ (②,①,0,\lambda_2,\mu_1^B,0) & \text{if } \mathbf{m}=(2,1),\ x_0^A=x_0^B=x_2^A=0, x_1^B>0,\ x_2^B<x_2^{B\sharp} \\ (②,①,0,\lambda_2,0,0) & \text{if } \mathbf{m}=(2,1),\ x_0^A=x_0^B=x_2^A=x_1^B=0,\ x_2^B<x_2^{B\sharp} \\ (②,②,0,\mu_2^A,0,\mu_2^B) & \text{if } \mathbf{m}=(2,2),\ x_0^A=0,\ x_0^B=0,\ x_2^A>0,\ x_2^B>0 \\ (②,②,0,\mu_2^A,0,\min(\mu_2^A,\mu_2^B)) & \text{if } \mathbf{m}=(2,2),\ x_0^A=0,\ x_0^B=0,\ x_2^A>0,\ x_2^B=0 \\ (②,②,0,\lambda_2,0,\mu_2^B) & \text{if } \mathbf{m}=(2,2),\ x_0^A=0,\ x_0^B=0,\ x_2^A=0,\ x_2^B>0 \\ (②,②,0,\lambda_2,0,\lambda_2) & \text{if } \mathbf{m}=(2,2),\ x_0^A=x_0^B=0, x_2^A=x_2^B=0,\ x_1^A<x_1^{A\sharp} \\ (②,❶,0,\lambda_2,0,0) & \text{if } \mathbf{m}=(2,2),\ x_0^A=x_0^B=0, x_2^A=x_2^B=0,\ x_1^A\geq x_1^{A\sharp} \\ (②,❶,0,\lambda_2,0,0) & \text{if } \mathbf{m}=(2,1),\ x_0^A=0,\ x_0^B>0,\ x_2^A=0,\ x_2^B<x_2^{B\sharp} \\ (②,❷,0,\mu_2^A,0,0) & \text{if } \mathbf{m}=(2,2),\ x_0^B>0,\ x_2^A>0 \\ (②,❷,0,\lambda_2,0,0) & \text{if } \mathbf{m}=(2,2),\ x_0^B>0,\ x_2^A=0 \\ (❶,①,0,0,\mu_1^B,0) & \text{if } \mathbf{m}=(1,1),\ x_0^A>0,\ x_0^B=0,\ x_1^B>0 \\ (❶,①,0,0,\mu_1^B,0) & \text{if } \mathbf{m}=(2,1),\ x_0^A=x_0^B=x_2^A=0, x_1^B>0,\ x_2^B\geq x_2^{B\sharp} \\ (❶,①,0,0,0,0) & \text{if } \mathbf{m}=(1,1),\ x_0^A>0,\ x_0^B=0,\ x_1^B=0 \\ (❶,①,0,0,0,0) & \text{if } \mathbf{m}=(2,1),\ x_0^A=0,\ x_0^B=0,\ x_1^B=0,\ x_2^B\geq x_2^{B\sharp} \\ (❶,❶,0,0,0,0) & \text{if } \mathbf{m}=(1,1),\ x_0^A>0,\ x_0^B>0 \\ (❶,❶,0,0,0,0) & \text{if } \mathbf{m}=(2,1),\ x_0^A=0,\ x_0^B>0,\ x_2^B\geq x_2^{B\sharp} \\ (❷,②,0,0,0,\mu_2^B) & \text{if } \mathbf{m}=(2,2),\ x_0^A>0,\ x_0^B=0,\ x_2^B>0 \\ (❷,②,0,0,0,\mu_2^B) & \text{if } \mathbf{m}=(1,2),\ x_0^A=0,\ x_0^B=0,\ x_2^B>0,\ x_1^B\geq x_1^{B\sharp} \\ (❷,②,0,0,0,0) & \text{if } \mathbf{m}=(2,2),\ x_0^A>0,\ x_0^B=0,\ x_2^B=0 \\ (❷,②,0,0,0,0) & \text{if } \mathbf{m}=(1,2),\ x_0^A=0,\ x_0^B=0,\ x_2^B=0,\ x_1^B\geq x_1^{B\sharp} \\ (❷,❷,0,0,0,0) & \text{if } \mathbf{m}=(2,2),\ x_0^A>0,\ x_0^B>0 \\ (❷,❷,0,0,0,0) & \text{if } \mathbf{m}=(1,2),\ x_0^A=0,\ x_0^B>0,\ x_1^A=0,\ x_1^B\geq x_1^{B\sharp} \end{cases} \tag{79}$$

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

Remark 9.3. An informal description of this feedback is given by:

- If at $t=0$ the modes $\mathbf{m}=(m^A,\ m^B)$ of the machines are unequal in the initial state, then make A switch to the same mode as B:

$$\mathbf{m}=(1,2)\rightarrow u_0^A:=❷\rightarrow x_0^A:=\sigma_{12}^A;\ m^A:=2 \tag{80}$$

$$\mathbf{m}=(2,1)\rightarrow u_0^A:=❶\rightarrow x_0^A:=\sigma_{21}^A;\ m^A:=1. \tag{81}$$

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

Note that the feedback always makes the servers process at the highest possible rate, obeying Lemma 3.1.

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

$$\begin{aligned}
\text{After mode } (1,1): &\quad \begin{bmatrix} x_1^A \\ x_1^B \\ x_2^A \\ x_2^B \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ x_2^{A\sharp} \\ x_2^{B\sharp} \end{bmatrix} \\
\text{After mode } (1,2): &\quad \begin{bmatrix} x_1^A \\ x_1^B \\ x_2^A \\ x_2^B \end{bmatrix} = \begin{bmatrix} 0 \\ x_1^{B\sharp} \\ x_2^{A\sharp} + \lambda_2 \theta_1^+ \\ x_2^{B\sharp} - \mu_2^B \max(\theta_1^+ - \sigma_{12}^B, 0) \end{bmatrix} \\
\text{After mode } (2,2): &\quad \begin{bmatrix} x_1^A \\ x_1^B \\ x_2^A \\ x_2^B \end{bmatrix} = \begin{bmatrix} x_1^{A\sharp} \\ x_1^{B\sharp} \\ 0 \\ 0 \end{bmatrix} \\
\text{After mode } (2,1): &\quad \begin{bmatrix} x_1^A \\ x_1^B \\ x_2^A \\ x_2^B \end{bmatrix} = \begin{bmatrix} x_1^{A\sharp} + \lambda_1 \theta_2^+ \\ x_1^{B\sharp} - \mu_1^B \max(\theta_2^+ - \sigma_{21}^B, 0) \\ 0 \\ x_2^{B\sharp} \end{bmatrix}.
\end{aligned} \tag{82}$$

The duration of mode $\mathbf{m} = (1,1)$ equals $x_2^{A\sharp}/\lambda_2$, whereas the duration of mode $(2,2)$ equals $x_1^{A\sharp}/\lambda_1$. Furthermore, mode $(1,2)$ always takes θ_1^+ and mode $(2,1)$ always takes θ_2^+ (this follows directly from the controller description).
Suppose that we enter mode $(1,2)$ in the transient for the n^{th} time $(n > 1)$. The buffer levels are at this point:

$$\begin{bmatrix} x_1^A & x_1^B & x_2^A & x_2^B \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} 0 & 0 & x_2^{A\sharp} + X^{(n)} & x_2^{B\sharp} \end{bmatrix}^{\mathrm{T}} \tag{83}$$

where $X^{(n)} \geq 0$ represents the additional buffer content with respect to the steady state value, when starting mode $(1,2)$ for the n^{th} time. Now we can wonder what the buffer levels are after mode $(2,2)$, and consequently after mode $(1,1)$. So we would like to express $X^{(n+1)} = f(X^{(n)})$. Instead of deriving the map f explicitly, we determine 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 $x_1^{A\sharp}/\lambda_1$, as if it were on the desired orbit. The resulting buffer levels

are then:

$$\begin{bmatrix} x_1^A & x_1^B & x_2^A & x_2^B \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} 0 & 0 & X^{(n)} & x_2^{B\sharp} \end{bmatrix}^{\mathrm{T}}. \tag{84}$$

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 after this step are then:

$$\begin{bmatrix} x_1^A & x_1^B & x_2^A & x_2^B \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} x_1^{A\sharp} + \frac{\lambda_1}{\min(\mu_2^A, \mu_2^B) - \lambda_2} X^{(n)} & x_1^{B\sharp} & 0 & 0 \end{bmatrix}^{\mathrm{T}}. \tag{85}$$

With the original controller of Proposition 9.2, 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 (cf. Lemma 3.1). For this reason, we know that for the real controller at the end of mode $(2,2)$, $\begin{bmatrix} x_1^B & x_2^A & x_2^B \end{bmatrix} = \begin{bmatrix} x_1^{B\sharp} & 0 & 0 \end{bmatrix}$ and for x_1^A:

$$x_1^{A\sharp} \leq x_1^A \leq x_1^{A\sharp} + \frac{\lambda_1}{\min(\mu_2^A, \mu_2^B) - \lambda_2} X^{(n)}. \tag{86}$$

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

$$0 \leq X^{(n+1)} \leq \frac{\lambda_1}{\min(\mu_2^A, \mu_2^B) - \lambda_2} \cdot \frac{\lambda_2}{\min(\mu_1^A, \mu_1^B) - \lambda_1} \cdot X^{(n)} \tag{87}$$

or rewritten:

$$0 \leq X^{(n+1)} \leq \frac{R_1}{1 - R_1} \cdot \frac{R_2}{1 - R_2} \cdot X^{(n)}. \tag{88}$$

Since $R_1 + R_2 < 1$ (result of Lemma 9.1), we can conclude:

$$\lim_{n \to \infty} X^{(n)} = 0 \tag{89}$$

which means that the system converges to the desired (optimal) periodic orbit (cf. (83) with the result of (89)). □

Convergence to the desired steady state process cycle has been proven. In the next section, we show the robustness of this controller by means of an example where disturbances with respect to the original hybrid fluid model occur.

10 Case study: controller implementation in switching server flowline

Consider the switching server flowline consisting of two workstations, processing two job types. A schematic overview of this flowline is given in Figure 9. The buffers have infinite capacity and the system parameters are as given in Table 2. The inter-arrival times and process times are exponentially distributed. This is a disturbance with respect to the original hybrid fluid model, but a more realistic instance of reality than constant process times and arrival times. The state feedback controller of Proposition 9.2 is implemented in a discrete event simulation. In this discrete event simulation, the fluid model is abandoned, i.e. buffer levels only take on integer (natural) values and processing jobs takes a real amount of time, contrary to the fluid model approximation.

The optimal process cycle of workstation B does not contain a slow-mode, so the periodic orbit has the pure bow tie shape (see Figure 4). The optimal process cycle can be characterized as follows: $\tau_1^{\mu B} = 1\frac{5}{6}$, $\tau_2^{\mu B} = 1\frac{5}{6}$, which results in $x_2^\sharp = 9\frac{1}{3}$ and $x_1^\sharp = 27\frac{1}{3}$. The mean number of jobs in the system is $29\frac{1}{3}$, which is $14\frac{2}{3}$ per job type (symmetric workstation with respect to process rates and arrival rates). Using Little's law, the mean flow time of both job types is $3\frac{2}{3}$ hours.

For workstation A, first conditions (51) and (52) are checked and they are fulfilled. The process cycle of A is computed in the way as described in the second part of the proof of Proposition 8.2.

Table 2: System parameters of flowline case study.

λ_1:	4 jobs/hr. (exp.)	μ_1^A:	20 jobs/hr. (exp.)	μ_1^B:	20 jobs/hr. (exp.)
λ_2:	4 jobs/hr. (exp.)	μ_2^A:	40 jobs/hr. (exp.)	μ_2^B:	20 jobs/hr. (exp.)
c_1:	1	σ_{12}^A:	0.5 hrs.	σ_{12}^B:	5.0 hrs.
c_2:	1	σ_{21}^A:	1.0 hrs.	σ_{21}^B:	0.5 hrs.

Table 3: Initial conditions for hybrid fluid model and stochastic discrete event simulations with state feedback controller.

$m^A(0)$:	1	$x_1^A(0)$:	25 jobs
$m^B(0)$:	1	$x_2^A(0)$:	25 jobs
$x_0^A(0)$:	0 hrs.	$x_1^B(0)$:	25 jobs
$x_0^B(0)$:	0 hrs.	$x_2^B(0)$:	25 jobs

The process cycle of workstation A has the following characteristics: $\tau_1^{\mu A} = \frac{11}{12}$, $\tau_1^{\lambda A} = 4\frac{7}{12}$, $\tau_2^{\mu A} = \frac{7}{9}$, $\tau_2^{\lambda A} = 1\frac{7}{18}$, $\theta_1^+ = 4\frac{7}{12}$ and $\theta_2^+ = \frac{5}{12}$.
For the controller implementation, the values of the buffers at which setup to the other job type takes place have to be computed: $x_1^{A\sharp} = 9$, $x_2^{A\sharp} = 7\frac{2}{3}$, $x_1^{B\sharp} = 18\frac{1}{3}$ and $x_2^{B\sharp} = 1\frac{2}{3}$.
First, a simulation with the original hybrid fluid model (with constant arrival and process rates) and the controller has been carried out (with Matlab). Initial conditions for this simulation are presented in Table 3. Simulation results are shown in Figure 14. These results are compared to a discrete event simulation, which has been carried out in χ [1]. The same initial conditions have been used (Table 3). Simulation results are shown in Figure 15 (trajectory of the system) and Figure 16 (individual buffer levels). Notice that the discrete event simulation results look very much like the hybrid fluid model results.

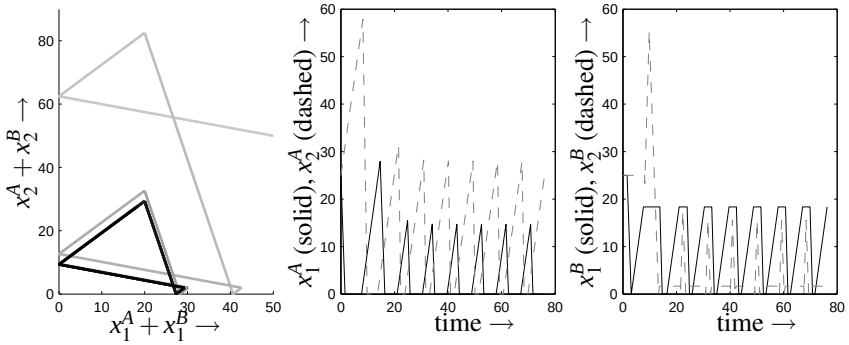

Figure 14: Results of hybrid fluid model simulation with controller implementation.

A number of simulations have been carried out with the stochastic discrete event model. In Table 4 the mean flow times of jobs and mean number of jobs in the system are listed, and the standard deviation. Results have been obtained for 20 different simulations. To exclude the transient effects, these simulations have been carried out with an initial point (almost) on the desired optimal orbit. As can be seen in the table, the results differ from the theoretic values of the hybrid fluid model. The differences are due to the variability on the inter arrival times and exponential process times and the discrete event characteristics of the simulation.

Remark 10.1. A different choice for the process cycle of workstation A may result in different simulation output and system performances. Since there is some freedom to choose the process cycle of A, the robustness for disturbances might be influenced. This is not investigated further here.

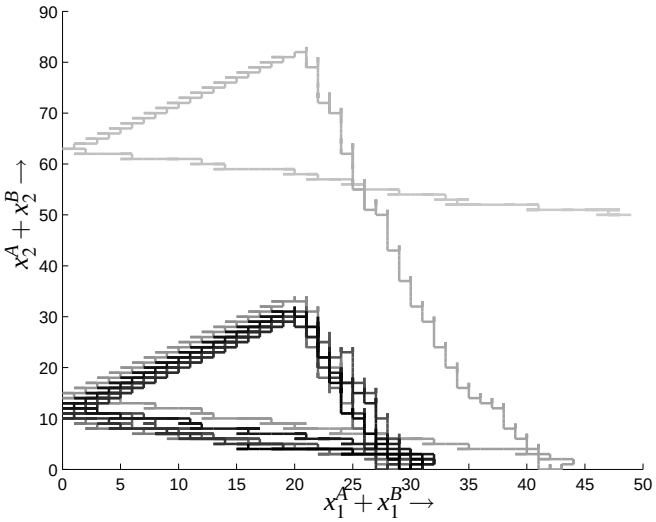

Figure 15: Trajectory of flowline with feedback controller implementation.

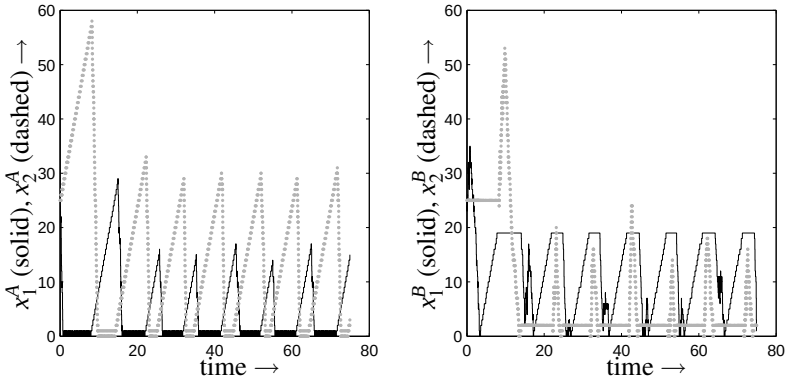

Figure 16: Buffer levels for workstation A (left) and B (right).

Table 4: Mean flow times ($\pm$ standard deviation) and number of jobs for the two job types.

	type 1 jobs	type 2 jobs
flow time:	3.89 ± 0.020	3.94 ± 0.016
number of jobs:	15.53 ± 0.082	15.75 ± 0.063

11 Conclusions and recommendations for further research

In this paper we studied switching servers with setup times. The optimal process cycle for a single server with two job types has been derived, with respect to time averaged weighted work in process levels. The analysis was based on a hybrid fluid model approximation of the workstation. Under certain circumstances, which have been fully characterized, a slow-mode occurs: jobs are processed at arrival rate after a buffer has been emptied, instead of immediate switching to the other job type. This slow-mode represents a trade-off between losing capacity due to processing at lower rates than the maximum rate and losing capacity due to relatively often switching between job types.

For a single workstation with bounded buffer capacities, a controller has been proposed that steers a trajectory to the optimal process cycle from any arbitrary start point. Convergence of this controller to the desired periodic orbit has been proven mathematically. The controller has been implemented successfully in a discrete event simulation study.

Switching servers are often placed in series. For such a flowline of switching servers, we want to find an optimal process cycle and develop feedback controllers. If a flowline consists of more than one workstation, only the most downstream server influences the work in process level, since all upstream workstations simply move work within the system. The minimal work in process level for a single switching server therefore is a lower bound on the work in process level for a flowline of switching servers. In this paper, we investigated under which conditions upstream workstations can make the most downstream workstation perform its optimal cycle. In that way, the lower bound on the work in process level can be achieved for the entire flowline. The class of flowlines that fulfill the conditions has been characterized.

For a flowline consisting of two workstations, a state feedback controller has been proposed. For this controller, convergence to the desired process cycles has been proven mathematically. A simulation study has been carried out for the flowline with two workstations. First, the controller has been implemented in the hybrid fluid model. Next, the controller was used in a discrete event simulation, with exponentially distributed inter arrival and process times. The controller performs very well in this stochastic and discrete event environment.

Although the analysis in this paper has been performed for a single switching server and for a flowline consisting of two workstations, the results are generally applicable for larger flowlines. Some remarks about this issue have been made throughout this paper.

Within the current line of research, a lot of challenges are still to be treated. For situations where we want to achieve the lower bound on the wip level of the most downstream workstation for the entire flowline, it can be useful to derive explicit feasibility conditions for all servers at once, instead of residing to a linear program. In the search for optimal process cycles for flowlines of switching servers, a general optimal cycle for all flowlines is yet to be determined. The results in this paper then are a special case of flowlines for which the optimal cycle for a single server can be achieved for the entire flowline. In addition, the influence of finite buffer capacities on flowlines of switching servers can be investigated. Another interesting topic is expanding the servers to more than two job types. Challenge is then to find an optimal process cycle. Difficulty is the fact that the sequence of process steps is not known beforehand, as it was in the case of two job types. Finally, a completely different challenge is to define and determine optimal transient behavior for controlled systems of switching servers and to develop controllers that yield optimal transients.

Acknowledgment

The authors gratefully thank Willem de Koning, M.Sc. student at the Department of Mechanical Engineering, Systems Engineering Group, TU Eindhoven, for his work on the flowline case study.

Bibliography

[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] M. Boccadoro and P. Valigi. A modelling approach for the dynamic scheduling problem of manufacturing systems with non negligible setup times and finite buffers. In *Proceedings of the 42nd IEEE Conference on Decision and Control*, pages 5472–5477, 2003.

[3] 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.

[4] O.J. Boxma, H. Levy, and J.A. Westrate. Efficient visit frequencies for polling tables: minimization of waiting cost. *Queueing systems theory and applications*, 9(1–2):133–162, 1991.

[5] C. Buyukkoc, P. Varaiya, and J. Walrand. The $c\mu$-rule revisited. *Advances in applied probability*, 17:237–238, 1985.

[6] C. Chase and P.J. Ramadge. On real-time scheduling policies for flexible manufacturing systems. *IEEE transactions on automatic control*, 37(4):491–496, 1992.

[7] H. Chen and D.D. Yao. *Fundamentals of queueing networks.* Springer, 2001.

[8] M. Del Gaudio, F. Martinelli, and P. Valigi. 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*, pages 2355–2360, 2001.

[9] M. Hofri and K.W. Ross. On the optimal control of two queues with server setup times and its analysis. *SIAM Journal on computing*, 16(2):399–420, 1987.

[10] 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.

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

[12] 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.

[13] A.S. Matveev and A.V. Savkin. *Qualitative theory of hybrid dynamical systems.* Birkhäuser, Boston, 2000.

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

[15] A.V. Savkin. Optimal distributed real-time scheduling of flexible manufacturing networks modeled as hybrid dynamical systems. In *Proceedings of the 42th IEEE Conference on Decision and Control*, pages 5468–5471, 2003.

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