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

Controller design for switched linear systems with setups

E. Lefeber*, J.E. Rooda

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

Available online 13 February 2006

Abstract

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

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

1. Introduction

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

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

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

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

0378-4371/$ - see front matter
doi:10.1016/j.physa.2006.01.049

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

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

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

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

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

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

2. An illustrative example

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