Online Control Design for QoS Management*

Sherif Abdelwahed† Sandeep Neema† Joseph Loyall‡ Richard Shapiro‡

Abstract – *In this paper we present an approach for QoS management that can be applied for a general class of real-time distributed computation systems. In this paper, the QoS adaptation problem is formulated based on a utility function that measures the relative performance of the system. A limited-horizon online supervisory controller is used for this purpose. The online controller explores a limited region of the state-space of the system at each time step and decides the best action accordingly. The feasibility and accuracy of the online algorithm can be assessed at design time.*

Keywords: Quality of service management, online hybrid control,switching systems

1 Introduction

In multi-process environment with performance variations, multiple applications share and compete for a limited amount of resources. Given the variations of resource availability, an adaptation mechanism needs to be implemented to ensure certain level of fairness and minimum level of services as well as satisfying user-defined priorities among the competing applications. This requires applications, in distributed computing environment, to be able to adjust their demands based on the availability of resources while preserving the requirements of their critical components. In addition, the overall system must posses a capability to rearrange the resources between various applications depending on their demands and the availability of resources.

There is a growing interest to apply automatic control techniques for QoS management in distributed computation systems. In [8] a two level control structure is used to manage QoS at the middleware level. In this framework, a PID controller is used to adjust the resource distribution at the system level. The adaptation provided from this controller is mapped to specific application adaptation using fuzzy controllers. A flexible structure for control-based adaptation is presented in [12]. The underlying project aims to provide a library to create feedback control loops for QoS management and adaptation. Another toolkit to construct feedback control loops is described in [7]. Combined estimation and compensation control has been applied in [10] to enhance the performance of a Lotus Notes server. Control theoretic techniques have also been used for congestion control in networks [2].

In the devolvement of feedback control for QoS management, a (linear) discrete-time model is usually assumed. However, the dynamics of practical distributed computation systems are typically complex involving both discrete-event and continuous-time dynamics. Systems with such mixed discrete-event and continuous-time dynamics are usually referred to as Hybrid systems. Considerable research work has been dedicated recently to the study of hybrid systems. See for example [4, 5] and the references therein.

In this paper, we present an online approach to the QoS adaptation of a general class of real-time computing systems modeled as switching hybrid system. The QoS control problem requires the system to optimize a given utility function during its operation. The proposed approach does not require the existence of a finite-state quotient equivalent for the system. Moreover, the approach can be adapted to accommodate possible changes in the system parameters that may occur as a result of a fault or parameter changes in time-varying systems.

The proposed procedure is conceptually similar to the model predictive control approach [9, 11] in which a limited time forecast of the process behavior at each state is optimized according to given criteria. Also related to our work is the limited lookahead supervision of discrete event systems (DES) [6]. In this approach a tree of all possible states is generated up to a given depth, then a control action is chosen to satisfy the specification.

This work is sponsored by the DARPA/IXO Model-Based Integration of Embedded Software program, under contract F33615-02-C-4037 with the Air Force Research Laboratory Information Directorate, Wright Patterson Air Force Base.

†emails: {sherif.abdelwahed,sandeep.k.neema}@vanderbilt.edu, Institute for Software Integrated Systems, Vanderbilt University, Nashville, TN

‡emails: {jloyall,rshapiro}@bbn.com, BBN Technologies, Cambridge, MA

2 Modeling for QoS management

In order to utilize control theory for QoS adaptation a suitable model for the underlying computational system needs to be established. System model captures the relationship between the observed system parameters - particularly, those relevant to the requirement specifications - and the control inputs used to adjust these parameters. In general, the system model can be given at the outset or identified through parameter estimation techniques. Typically, an initial model is build for those system components with known dynamics while parameter estimation and learning techniques are used to identify other unknown parts of the system.

Typical distributed computation systems contain both time and event-driven dynamics. The timed-driven dynamics are usually sampled in practical systems so without loss of generalization we will assume that such dynamics is described by discrete-time difference equations. In real-time computation systems queues, filters, and sampling units can be modeled as (linear) discrete-time systems. On the other hand communication protocols and certain software components are event-driven structures which can be described by finite state machines possibly with timing information. Such components of different dynamics can interact through an interface composed of event generators that maps discrete time signals into events (for example a threshold detection event), and actuators which maps events to discrete-time signals (for example a desired level of the queue).

There are several available models to describe hybrid dynamic systems. A general class of hybrid systems can be represented by hybrid automata [3]. In this paper, however, we consider a special class of hybrid systems, referred to as *switching systems*, that can describe switching dynamics associated with most real-time computation systems. Switching hybrid system can be described by the discrete-time equation

$$\boldsymbol{x}(k+1) = \boldsymbol{\phi}(\boldsymbol{x}(k), \boldsymbol{r}(k))$$

where $k \in 0, 1, \ldots$ is the time index, $\boldsymbol{x}(k) \subset \mathbb{R}^n$ is the sampled form of the continuous state vector at time k, and $\boldsymbol{r}(k) \subset \mathbb{R}^m$ is the discrete valued input vector at time k. We will use X and R to denote the state space and the input set for the system, respectively. For each input $\boldsymbol{r} \in R$, the function $\phi(., \boldsymbol{r})$ is continuous in X and meets the conditions for existence and uniqueness of solutions for a set of initial states $X_o \subseteq X$. We assume that the set of inputs R is finite. Boldface letters are used here to denote vectors and vector-valued signals. We will use the subscript i to distinguish the ith component for a vector, for example, x_i denotes the ith component of the state vector $\boldsymbol{x}$, and $\phi_i(\boldsymbol{x}, \boldsymbol{r})$ denotes the ith component of the map $\phi(\boldsymbol{x}, \boldsymbol{r})$.

The above model is general enough to describe a wide class of hybrid systems, including nonlinear systems and piece-wise linear systems. The requirement that the input set R is finite is typical in many practical computer-controlled systems, where the input is usually discrete and restricted to a finite set. It is important to note, however, that the proposed online control approach is more suitable for systems with small number of control inputs as, in general, the size of the search tree grows exponentially with the number of input switching signals which is proportional to the size of the input set. Many real-time computation systems have a limited finite (quantized) set of control inputs and therefore can be adequately captured using the above model.

In real-time computation systems, performance specifications can be classified into two categories. The first type is set-point specifications in which the underlying parameter (variable) is required to be maintained at specific level or follow a certain pattern. Examples of this type include utilization level of the server, minimal acceptable frame rate, and queue size. The other type of specification is used for performance enhancement where it is required to maximize or minimize the underlying variable. In this case several variables can be lumped together with different weights in one function, typically referred to as utility or cost function. In general, it is assumed that the two sets of variables involved in these two types of specifications are disjoint.

The objective of adaptation is to achieve the desired levels of the set-point specifications in reasonable time while maintaining the system stable at the desired value. In addition, it is required to optimize the given utility functions. In most situations, set-point specifications take precedence over utility optimization. Due to the nature of the computation environment, it is common that variables that contribute to the utility function are evaluated over a quantized finite domain. For example, the quality of the result of a given task varies with respect to the size of the input which can only take a finite set of values.

In this paper we are primarily dealing with the second type of specifications. We assume that the optimal performance of system is expressed in a utility function defined over a finite domain of systems variables and parameters. However, it is easy to see that a utility function can also be used for set point control by considering the difference between the current value and the desired level. The precedence factor can be approximated by considering high weights for the set point variables relative to the utility variables.

3 Control of switching systems

The problem of optimal performance is stated as follows. Given a switching hybrid system H and a utility function U and a range of operation $X_r \subseteq X$, design a supervisor S that can maximize the given utility function for the given region of operation. In this paper we assume that the set X_r is convex. The key requirement

to the success of this control action is the ability of the controller to keep minimizing the given utility function at any point in the operation region of the system. It is also required that the system is maintained within a predefined operation region which is typically defined as a limit on the values of some or all of the system variables.

To achieve the above objective, we propose an online supervision algorithm that explores only a limited part of the system state space and selects the next input based on the available information about the current state. For the safety control problem, the selection of the next step is based on a utility map $U : \mathbb{R}^n \to \mathbb{R}$ that defines the current level of performance of the system. We assume that the argument of the map U is the current state vector $\boldsymbol{x}$. However, this can be easily extended to include input variables as well. The decision of the controller is also influenced by the boundary of the given operation region where it is required to maintain the system variables within.

The online supervision algorithm starts by constructing the tree of all possible future states from the current state $\boldsymbol{x}_c$ up to a specified depth. To avoid the Zeno effect, in which the controller may try to preempt time indefinitely through continuous switching, we require that any input switching event is followed by at least one sampling period. The exploration procedure identifies the set of states with the utility based on the map U. A state $\boldsymbol{x}_m$ is then chosen from this set based on certain optimality criterion (for example, minimal input switching), or simply picked at random. The chosen state is then traced back to the current state $\boldsymbol{x}_c$ and the event leading to $\boldsymbol{x}_m$ is used for the next step.

Giving the limited exploration nature of the online algorithm, it is important to obtain a measure of feasibility to determine if the online control will be able to maximize (minimize) the given performance measure in a finite time. Such measure is particularly important in uncertain environments where the system variables may be influenced with some stochastic factors. Such situation is not uncommon in real-time distributed computation system where, for instance, system performance may be affected by noise, communication delay, and measurement inaccuracy. The feasibility measure can provide an assurance that the system will continue to improve its performance even in the presence of such unpredictable factors.

The utility optimization problem can be formulated as a safety control problem as presented in [1]. This is the case when the optimal value of the utility within the given region can be calculated online. In this case, let $\boldsymbol{x}_o$ be the value of the system variables corresponding to the optimal utility. In the online control, the selection of the next step would then be based on a distance map $D_s : \mathbb{R}^n \to \mathbb{R}$ that defines how close the current state is to the optimal state. The distance map can be generally defined as follows

$$(\forall \boldsymbol{x} \in X_r) \quad D_s(\boldsymbol{x}) = \|\boldsymbol{x} - \boldsymbol{x}_o\|$$

where $\|.\|$ is a proper norm for $\mathbb{R}^n$. In other words, $D_s(\boldsymbol{x})$ defines the minimal distance between $\boldsymbol{x}$ to the optimal state. The above measure can be extended easily to the case when optimal utility is associated with a region X_o rather than a point. In this case, $D_s(\boldsymbol{x})$ will be defined as the minimal radius of all spheres centered at $\boldsymbol{x}$ that contain at least one point from X_s.

Given the above settings, a hybrid system H is said to be *online controllable* in the region $X_r \subseteq X$ if there exists $\check{\delta}_{X_r} > 0$ such that for any partition $\{n^+, n^-\}$ of the set $[1 \dots n]$ there exists an input $\boldsymbol{r} \in R$ such that $(\forall \boldsymbol{x} \in X_o)(\forall i \in [1 \dots n])$,

$$i \in n^+ \Rightarrow \phi_i(\boldsymbol{x}, \boldsymbol{r}) - x_i > \check{\delta}_{X_o},$$
$$i \in n^- \Rightarrow x_i - \phi_i(\boldsymbol{x}, \boldsymbol{r}) > \check{\delta}_{X_o}$$

That is, H is online controllable in the region X_r if at any state $\boldsymbol{x} \in X_r$ it is always possible to find an input that can control the next step direction by incrementing some components of $\boldsymbol{x}$ and decrementing the other components. The requirement that the increments (decrements) be greater than a given positive number is needed to ensure that the system does not converge to a fixed point within the region X_r.

Consider a hybrid system H which is online controllable within a region X_r. For a state $\boldsymbol{x}_r \in X_r$ write $\boldsymbol{\phi}_c(\boldsymbol{x}_r, k)$ for the state $\boldsymbol{x}(k)$ obtained from the online control algorithm after k time steps starting from $\boldsymbol{x}_r$. The *accuracy error* of the online controller within the region X_r and for a distance map D_s after k time steps is defined as follows,

$$E_k(X_r, D_s) = \min\{D_s(\boldsymbol{\phi}_c(\boldsymbol{x}_r, k)) \mid \boldsymbol{x}_r \in X_r\}$$

That is, $E_k(X_r, D_s)$ is the minimal distance to the safe region that can be obtained starting at any state in X_o after k time steps. The accuracy error of the online controller can be estimated based on the upper limit of the distance covered by the system in a single step. Write $\hat{\delta}_{X_r}$ for the maximal single step absolute change to any component in a state $\boldsymbol{x} \in X_r$ under any input from R, namely $\hat{\delta}_{X_r}$ is equal to

$$\max\{|\phi_i(\boldsymbol{x}, \boldsymbol{r}) - x_i| \mid \boldsymbol{x} \in X_r, \boldsymbol{r} \in R,\ i \in [1 \dots n]\}$$

Write $\hat{\boldsymbol{\delta}}_{X_r}$ for the vector $(\hat{\delta}_{X_r}, \dots, \hat{\delta}_{X_r})$. Then,

Proposition 1 There exists $N > 0$ such that

$$(\forall k > N) \quad E_k(X_r, D_s) \leq 0.5\|\hat{\boldsymbol{\delta}}_{X_r}\|$$

□

Therefore, an online controllable system H can be driven by the online controller in finite time to the optimal state $\boldsymbol{x}_o \in X_r$ with a maximum accuracy error of

$0.5\|\hat{\boldsymbol{\delta}}_{X_r}\|$. This means it is always possible to reach a near optimal performance (depending on the accuracy) in a finite time interval. In the implementation of the online control, the set X_r may need to be adjusted to take into account this accuracy error as well as the existence of measurement noise.

The parameters $\check{\delta}_{X_r}$, $\hat{\check{\delta}}_{X_r}$ can be used to reduce the search tree in the online control algorithm. The algorithm can safely stop exploring if there is no prospect of further reduction in the current minimal distance along any path starting from the current node up to the limit of the search tree. This can easily be determined using the values $\check{\delta}_{X_r}$, $\hat{\check{\delta}}_{X_r}$, and the predefined depth of the search tree.

4 Case study

This section describes a prototype signal detection system, developed by Southwest Research Institute, San Antonio, Texas, that we are using to evaluate the QoS adaptation approach presented above. A typical signal detection system accepts as input a finite-duration, time-domain signal and classifies it according to the properties of interest, such as modulation (Phase Shift Keying vs. Frequency Shift Keying), carrier frequency, symbol rate, etc. The signal classification takes a finite amount of time, and provides a confidence measure in the quality of the results. The classification is done using highly involved signal processing algorithms. It is often the case that several of these detection algorithms are parameterized and may be "tuned" to trade-off the quality (accuracy) of the result with the computation time in order to achieve the desired real-time performance. The system operates as follow:

1. A random number of new signals arrive at any given point of time in the field of operation of the system. The number of signal arriving at time k is denoted $B(k)$. In certain situations, the number of arriving signals can be approximated as an auto-regressive stochastic process.

2. Typically there are more than one signal arriving at each time unit. These signals are added to a FIFO queue in order to be processed by the system.

3. The signal detection module removes one signal from the queue, buffers it temporarily and processes a fraction (we refer to this fraction as feature level in the rest of this section) of it. In case the processed fraction was sufficient to correctly classify the signal, as determined by the lower threshold on the confidence measure, then the signals is discarded from the buffer and a new signal is fetched from the queue. Otherwise, a larger fraction of the signal is taken from the buffer and classification is reattempted on this larger fraction. The end result of the signal detection is the estimated symbol rate, the computation time required, and a confidence measure. Note that the confidence as well as the computation time required depends upon the fraction of the signal that is used for classification.

4. A user-defined utility function continuously assesses the utility of the system which. The utility is defined with respect to the quality of the results i.e. confidence measure, the throughput i.e. the number of signals classified per unit of time which is a function of the average compute time, and the latency i.e. time between when signal enters the queue, and when it is classified, which is a function of the queue-size.

The above operational scenario is augmented with an online controller module that monitors various system variables and estimates the utility of the system, and adjusts the feature level in a closed loop such that the overall utility of the system is maximized.

We have realized this operational scenario in a MatLab/Simulink model. Figure 1 depicts the Simulink model of the signal detection system integrated with an online controller. The individual modules in the Simulink model are described below.

Signal Source This module simulates the arrival of new signals in the field of operation of the system. At any time instant, this block adds n signals to the queue. The signals are real data signals, read from pre-recorded wave files. The number of signals arriving at time k is denoted $B(k)$. In many situations, it is reasonable to approximate the incoming signals as an auto-regressive stochastic process.

Queue This module simulates a queueing function that updates the number of unprocessed signals in the queue based on the number of new signals received, and the signals being processed by the signal detection module. The current level of the queue is denoted $q(k)$.

Buffer This module simulates a signal buffer. Depending on the state of the switch $v(k+1)$, as set by the controller module, it either retrieves a new signal from the queue, and dispatches a fraction $n(k+1)$ of it to the signal detection module, or delivers another fraction of the same (previously retrieved) signal to the signal detection module.

Signal Detection This module performs classification on the fraction of the signal delivered by the buffer module. It uses a PSK feature extraction algorithm provided by Southwest Research Institute. The module outputs the PSK symbol rate, as well as a confidence measure $c(k)$ which estimates the quality of the computations, and computation time.

The online control module uses the Utility module to obtain an estimation of the next step utility given

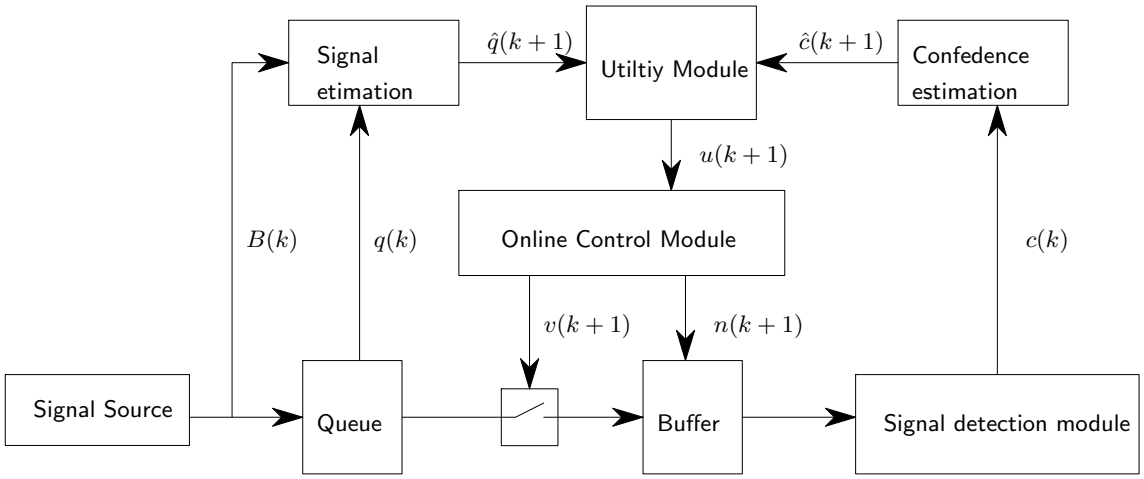

Figure 1: The Signal Detection System with Online Control

the current state and inputs. The Utility module relies on two estimation modules. The first is the signal estimation module which estimates the next level of the queue based on the current level and the estimated signal model. The estimated next queue level is given by

$$\hat{q}(k+1) = q(k) + \hat{B}(k)\hat{t}(k) - v(k+1)$$

where $\hat{B}(k)$ is the estimated rate of incoming signal, $\hat{t}(k)$ is the estimated computation time per data unit. Note that in the above setting, the time between the k and $k+1$ instances, denoted $t(k)$ is not fixed and depends on the amount of data input to the Feature extraction module, namely, $n(k+1)$. Typically $t(k)$ grows almost linearly with $n(k+1)$. Also, an auto-regressive model is used to estimate $B(k)$ given the previous measurement. The parameters of this model is update at each time instance.

The quality of the processing at the feature extraction algorithm is given through a confidence measure $c(k)$. The confidence $c(k)$ depends to a large extent on the size of the feature level $n(k+1)$. An initial model to estimate the next confidence is obtained initially through simulation. The estimated confidence is given by

$$\hat{c}(k+1) = \begin{cases} \alpha\, n(k+1) & \text{if } v(k+1) = 1 \\ c(k) + \alpha\, n(k+1) & \text{if } v(k+1) = 0 \\ & \quad \text{and } n(k+1) > N_s \\ \beta & \text{otherwise} \end{cases}$$

The parameters α, β, N_s are set initially from the simulation data and updated with the new information at each time instant. The objective of the online controller is to maximize the following utility function

$$U(k) = a_1\,[q(k)]^2 + a_2\,[c(k)]^2$$

The factors a_1 and a_2 are user specified and defines the relative importance of the real-time versus accuracy performance of the system. The online controller uses the estimated utility $\hat{U}(k+1)$ (based on $\hat{q}(k+1)$ and $\hat{c}(k+1)$) for given set of inputs up to a finite number of forward steps to decide the best input to maximize the utility.

Simulation result

Figures 2 and 3 below show some results from a simulation run. It can be observed from the results shown below that the controller tunes the feature level, to maximize the utility, which decrease because of increasing queue size over some periods. In the absence of such adaptation it can be expected that the queues will overflow, resulting in either system failure or missing of potentially interesting signals.

5 Conclusion

In this paper we presented an online hybrid control approach for the QoS management which can be applied to a wide range of real-time distributed computation systems. The proposed approach explores a limited depth tree of all possible transitions from the current states. The online controller can be tested at design time for feasibility and accuracy.

References

[1] S. Abdelwahed, G. Karsai, and G. Biswas. Online safety control of a class of hybrid systems. In *41st IEEE Conference on Decision and Control*, pages 1988–1990, Las Vegas, NV, 2002.

[2] E. Altman, T. Başar, and R. Srikant. Congestion control as a stochastic control problem with action delay. *Automatica*, 35:1937–1950, 1999.

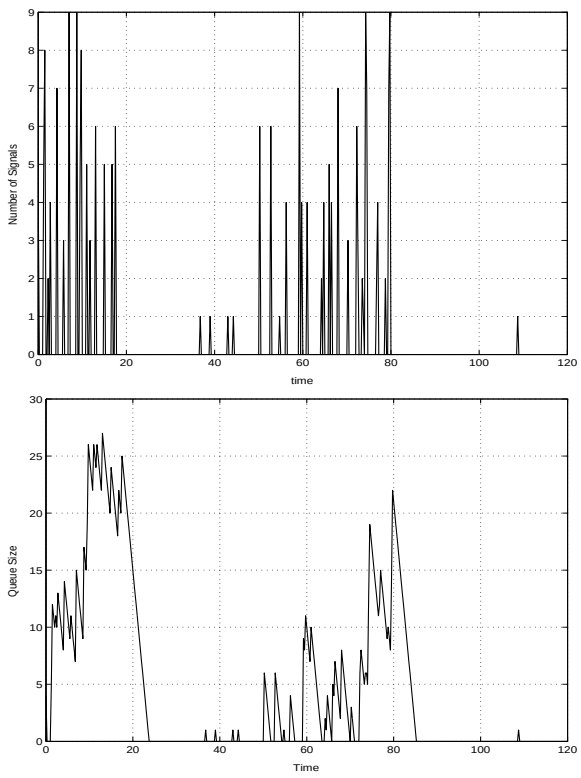

Figure 2: Number of Incoming Signals and the Queue Level

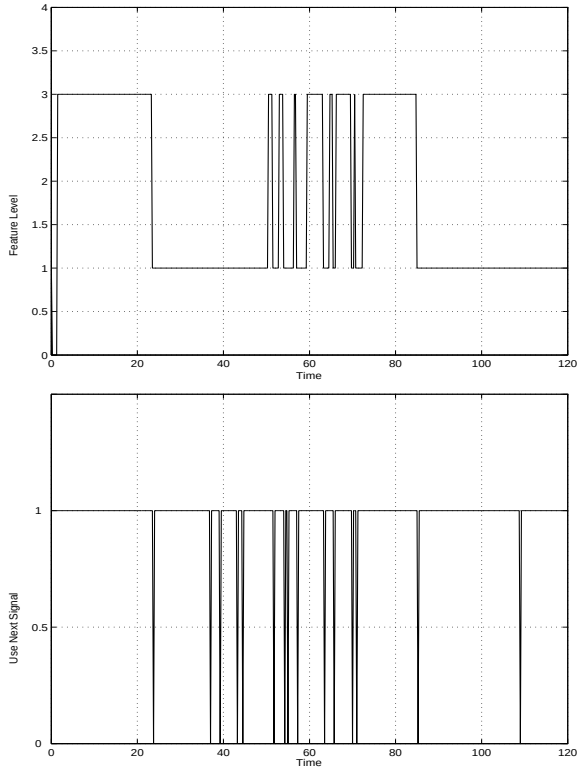

Figure 3: Controller Outputs: Feature Level $n(k)$ and Next Signal $v(k)$

[3] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138(1):3–34, 1995.

[4] P. Antsaklis, editor. *Special Issue on Hybrid Systems*. Proceedings of the IEEE. July 2000.

[5] P. Antsaklis, X. Koutsoukos, and J. Zaytoon. On hybrid control of complex systems: a survey. *European Journal of Automation*, 32:1023–1045, 1998.

[6] S. L. Chung, S. Lafortune, and F. Lin. Limited lookahead policies in supervisory control of discrete event systems. *IEEE Trans. Autom. Control*, 37(12):1921–1935, December 1992.

[7] A. Goel, D. Steere, C. Pu, and J. Walpole. SWiFT: A feedback control and dynamic reconfiguration toolkit. Technical Report CSE-98-009, Department of Computer Science and Engineering, Oregon Graduate Institute, 1998.

[8] B. Li and K. Nahrstedt. A control-based middleware framework for quality of service adaptations. *IEEE Journal on Selected Areas in Communications*, 17(9):1632–1650, 1999.

[9] M. Morari and J. Lee. Model predictive control: Past, present and future. *Computers and Chemical Engineering*, 23:667–682, 1999.

[10] S. Parekh, N. Gandhi, J. Hellerstein, D. Tilbury, T. Jayram, and J. Bigus. Using control theory to achieve service level objectives in performance management. In *Proc. IFIP/IEEE International Symposium on Integrated Network Management*, Seattle, WA, 2001.

[11] S. Qin and T. Badgewell. An overview of industrial model predictive control technology. *Chemical Process Control*, 93(316):232–256, 1997.

[12] R. Zhang, C. Lu, T. Abdelzaher, and J. Stankovic. Controlware: A middleware architecture for feedback control of software performance. In *Proceedings of the ICDCS*, 2002.