

A Hybrid Control Design for QoS Management*

Sherif Abdelwahed Sandeep Neema
Vanderbilt University
{sherif,sandeep}@isis.vanderbilt.edu

Joseph Loyall Richard Shapiro
BBN Technologies
{jloyall,rshapiro}@bbn.com

Abstract

In this paper we present an approach for QoS management that can be applied to a general class of real-time distributed computation systems. In the proposed approach a switching hybrid system model is used to represent the system. In this setting, the QoS specifications are transformed into set-point specifications and a limited-horizon online supervisory controller is used to move the system to the optimal operation point.

1. Introduction

In multi-process environments with performance variations, multiple applications compete for a limited amount of resources. Given the variations of resource availability, an adaptation mechanism needs to be implemented to ensure a certain level of fairness and minimum level of services as well as satisfying user-defined priorities among the system components.

Managing the QoS of real-time computational systems can be achieved by converting the QoS specifications into control specifications. These specifications together with the system model can then be used to design a control structure for managing the system resources efficiently in a way that satisfies the requirement specifications at both the system and the application levels.

The last decade has witnessed a growing interest to apply automatic control techniques for QoS management in distributed computation systems. A two level control structure was proposed in [5] to manage QoS of a class of real-time system 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 [10]. The underlying project aims to provide a library for designing feedback control loops for QoS management and adaptation. Another toolkit to construct feedback control loops is described in [4]. Combined estimation and compensation control has been applied in [7] to enhance the performance of a Lotus Notes server. In [6] a feedback control real-time scheduling framework for adaptive realtime systems is presented.

In most literature on the application of control theoretic techniques for QoS management, a (linear) discrete-time model for the system dynamics is usually assumed. However, the dynamics of practical distributed computation systems are typically complex involving both discrete-event and time-based dynamics. Systems with such mixed discrete-event and time-based dynamics are usually referred to as hybrid systems. In this paper, we present an online approach to the QoS adaptation of a general class of real-time distributed computing systems modeled as switching hybrid system, that is, a hybrid system with finite control set. The QoS control problem requires the system to achieve certain performance levels and in the same time optimize a given utility function during its operation.

The proposed procedure is conceptually similar to the model predictive control approach [8] 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 [3]. 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.

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. Practical distributed real-time systems usually contain both time and event-driven dynamics, namely, hybrid in nature. In the past decade several models have been proposed for the analysis and simulation of hybrid dynamic systems, see for instance [2] and the references therein. In this paper, we consider a special model for hybrid systems with finite control set referred to as *switch-*

* 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.

ing hybrid systems. A switching hybrid system is described by the discrete-time equation

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

where $k \in 0, 1, \dots$ is the time index, $\mathbf{x}(k) \in \mathbb{R}^n$ is the sampled form of the continuous state vector at time k , and $\mathbf{r}(k) \in \mathbb{R}^m$ 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. The set of inputs R is assumed finite.

The above model is general enough to describe a wide class of hybrid systems. The requirement that the input set R is finite is not uncommon in many practical computer-controlled systems, where the control inputs are usually discrete and take values from a finite 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.

QoS specifications 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 is required to be maintained at specific level or follow a certain pattern (trajectory). The other type of specification, referred to as *performance specifications*, is used to optimize the system performance by maximizing a given performance measure. The objective of QoS adaptation is to achieve and maintain the desired levels of the set-point specifications in reasonable time, and in addition optimize the given performance function. In this paper we assume that optimal points for performance functions can be computed at any given time instance and therefore the QoS specification is given as a desired operation region.

In a multiprocess computation environment, each component has its own QoS specifications. In addition a global QoS requirement for the overall system may also be specified. Due to possible interactions between the system components and the fact that they share the same resource, these specifications may conflict with each other. Such conflict can be resolved by the controller in two ways. In the first case the set of specifications are ordered based on their importance. In such situation, the controller will try to satisfy each specification in the given order. A solution exists for this scenario if the system remains controllable with respect to the low order specification after satisfying higher order ones. In the second case all QoS specification are lumped into one specification scheme with different weights reflecting the priorities among these specifications. The solution in this case is clearly similar to the single process situation.

3. Online control of switching systems

The set-point control of switching hybrid systems can be stated formally as follows: given a switching system H and a connected set of safe states X_s and a set of initial states $X_o \subseteq X$ where $X_s \subset X_o$, design a supervisor S that can drive the system from any state in X_o to X_s in a finite time using a finite sequence of inputs. In addition, the supervisor is required to keep the system stable within the set X_s . Conventional control approach cannot be applied directly in this case due to the fact the H is generally non-linear and the control set is finite.

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 above control problem, the selection of the next step is based on a distance map $D_s : \mathbb{R}^n \rightarrow \mathbb{R}$ that defines how close the current state is to the given safe region. The distance map can be generally defined as follows: $D_s(\mathbf{x}) = 0$ for all points $\mathbf{x} \in X_s$ and for all other points ($\mathbf{x} \notin X_s$),

$$D_s(\mathbf{x}) = \min\{a \in \mathbb{R} \mid (\exists \mathbf{x}' \in X_s) \|\mathbf{x} - \mathbf{x}'\| = a\}$$

where $\|\cdot\|$ is a proper norm for \mathbb{R}^n . In other words, $D_s(\mathbf{x})$ is the minimal distance between \mathbf{x} to the safe region X_s .

The online supervision algorithm starts by constructing the tree of all possible future states from the current state \mathbf{x}_c up to a user specified depth. The exploration procedure identifies the set of states with the minimal distance from X_s based on the distance map D_s . A state \mathbf{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 \mathbf{x}_c and the event leading to \mathbf{x}_m is used for the next step.

In certain situations it is possible to check the feasibility of the above approach, namely to determine if the online control will be able to reach the required operation region with certain accuracy. In [1], the concept of online controllability is introduced for a special class of switching systems with independent state variables. Online controllability is extended for the general class of switching hybrid systems in [9]. An algorithm that can test the feasibility of online control approach is given.

4. Case study: signal detection system

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 (PSK vs. FSK), 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 follows:

1. A random number of signals arrive at any given point of time in the field of operation of the system. The number of signals arriving at time k is denoted $B(k)$.
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 of it. In case the processed fraction was sufficient to correctly classify the signal then the signal 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.
4. A user-defined utility function continuously assesses the system performance. 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 a 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.

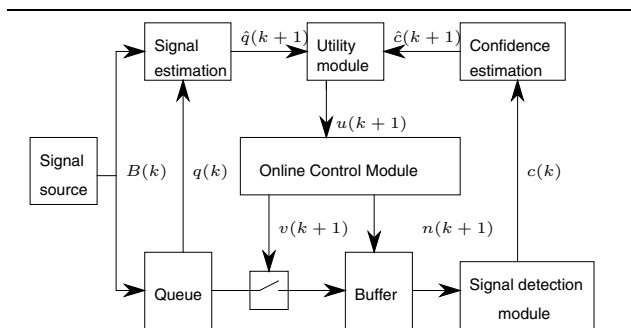


Figure 1. The signal detection system

We have realized this operational scenario in a Mat-Lab/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 At any time instant, this block adds n signals to the queue. The number of signals arriving at time k is denoted $B(k)$. Arriving signals are stored in the queue and the current level of the queue is denoted $q(k)$.

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

Signal Detection This module performs classification on the fraction of the signal delivered by the buffer module. 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 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

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

where $\hat{B}(k)$ is the estimated rate of incoming signal, $t(k)$ 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 Signal Detection module, namely, $n(k+1)$. Also, an auto-regressive model is used to estimate $B(k)$ given the previous measurement.

The quality of the processing at the feature extraction algorithm is given through a confidence measure $c(k)$. An initial model to estimate the next confidence is obtained through simulation. The estimated confidence is given by

$$\hat{c}(k+1) = \alpha n(k+1)v(k+1) + c(k) + \alpha n(k+1)c(k)v(k+1)$$

when $n(k+1) < N_s$ and is constant equal to some value β otherwise. The parameters α, β, N_s are obtained through simulation. 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 define the relative importance of the real-time versus accuracy performance of the system.

Simulation result Figures 2 below shows the results from a simulation run. As shown in the utility plot below, the controller maintains a higher utility compared with the no control situation ($v(k+1) = 1$ and $n(k+1) = 0$). The average utility with control is 223% higher than the no control case.

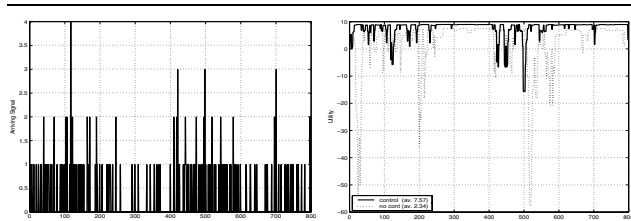


Figure 2. Simulation results

5. System implementation

Our basic approach is in developing a semantically rich, domain-specific modeling language supporting the high-level design of QoS adaptation strategies and concerns, from which we can derive runtime interfaces and low-level programming artifacts using code-generation techniques. This domain-specific modeling language, that we call Dynamic QoS Adaptation Modeling Environment (DQME), allows the system designer to represent the specifics of his system relevant to the design and synthesis of a controller for the system. The aspects modeled include:

- The application's functional structure defining the data flow within the application components/tasks.
- Controllable and observable parameters defining the variables that can be measured/manipulated at runtime.
- Bounds defining acceptable and preferred region of operation for different observable variables.
- Controller model defining the intended action and its association with other variables in the system.
- Mission requirements, namely the overall control objectives in terms of decision variables.

In this approach, we rely on code-generators to translate the specifications captured as models into code that can be instantiated at runtime. Our runtime environment is built around QuO [11], a middleware framework that allows instantiating adaptive behaviors in distributed real-time systems. The controller that is being modeled is instantiated as a QuO Contract, and is connected to the observable and controller parameters, instantiated as QuO System Condition, at runtime. The controller then performs its actions on a deployed system, maintaining the desired QoS.

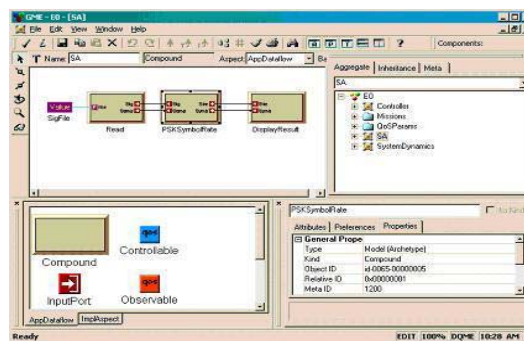


Figure 3. GME model of the signal detector

Figure 3 illustrates a system that we have modeled and simulated in our prototype development of these concepts. It implements the adaptive control strategy for the signal detection application described in the previous section.

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, 2002.
- [2] P. Antsaklis, editor. *Special Issue on Hybrid Systems*. Proceedings of the IEEE, July 2000.
- [3] 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, Dec. 1992.
- [4] A. Goel, D. Steere, C. Pu, and J. Walpole. SWIFT: A feedback control and dynamic reconfiguration toolkit. Technical Report CSE-98-009, Oregon Graduate Institute, 1998.
- [5] 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.
- [6] C. Lu, J. Stankovic, G. Tao, and S. Son. Feedback control real-time scheduling: Framework, modeling and algorithms. *Journal of Real-Time Systems*, 23(1/2):85–126, 2002.
- [7] 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 Int. Symp. on Integrated Network Management*, 2001.
- [8] S. Qin and T. Badgewell. An overview of industrial model predictive control technology. *Chemical Process Control*, 93(316):232–256, 1997.
- [9] R. Su, S. Abdelwahed, and S. Neema. On the online controllability of switching systems. Submitted to The American Control Conference, Boston, MA, 2004.
- [10] 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.
- [11] J. Zinky, D. Bakken, and R. Schantz. Architectural support for quality of service for corba objects. *Theory and Practice of Object Systems*, 3(1):1–20, 1997.