A Control-Based Framework for Self-Managing Distributed Computing Systems*

Sherif Abdelwahed
Institute for Software Integrated Systems
Vanderbilt University
Nashville, TN 37203
sherif@isis.vanderbilt.edu

Nagarajan Kandasamy
Electrical and Computer Engineering Department
Drexel University
Philadelphia, PA 19104
kandasamy@ece.drexel.edu

Sandeep Neema
Institute for Software Integrated Systems
Vanderbilt University
Nashville, TN 37203
sandeep@isis.vanderbilt.edu

ABSTRACT

This paper describes an online control framework to design self-managing distributed computing systems that continually optimize their performance in response to changing computing demands and environmental conditions. An online control technique is used in conjunction with predictive filters to tune the performance of individual system components based on their forecast behavior. In a distributed setting, a global controller is used to manage the interaction between components such that overall system requirements are satisfied.

1. INTRODUCTION

Distributed computer systems host information technology (IT) applications vital to commerce, transportation, industrial process control, military command and control, among others. Such systems typically comprise numerous software and hardware components that must together satisfy stringent quality-of-service (QoS) requirements while operating in highly dynamic environments; for example, the workload to be processed may be time varying and system components may fail during operation. To operate such computer systems efficiently while achieving the desired QoS goals, multiple performance-related parameters must be carefully tuned and the current state-of-the-art requires substantial manual effort. Moreover, these parameters must adapt dynamically to operating conditions and as IT applications become more complex and the underlying computer systems more distributed in their operation, it will become difficult for human operators to effectively manage their performance.

This paper present a control-based approach to designing self-managing computing systems that aim to maintain a specified QoS over a wide range of operating conditions. A *model-predictive* (*receding horizon*) control approach [11] is used where the control actions optimizing system QoS are derived over a limited prediction horizon. The proposed approach addresses the design, online analysis and refinement, and verification of systems that continually tune their behavior in response to changes in operating environment such as resource failures and workload variations.

Recently, control-theoretic concepts, particularly feedback control, have been used to design adaptive resource management schemes for various IT applications [4, 10, 14, 12]. If the computer system of interest is correctly modeled and the effects of its operating environment estimated accurately over a finite range of operation, appropriate control laws can be derived to achieve the desired QoS objectives. Moreover, established techniques in control theory can be used to verify, *a priori*, the system design itself by analyzing properties such as stability, convergence, safety, and liveness [13].

The above methods all use classical *feedback or reactive control* to first observe the current system state and then take corrective action, if any, to achieve the desired QoS. Simple feedback control, however, has some inherent limitations. It usually assumes a linearized and discrete-time model for system dynamics with a continuous input (output) domain. Many practical systems, however, have a finite set of possible control inputs, and exhibit *hybrid* behavior comprising both discrete-event and time-based dynamics. Also, feedback control cannot be easily applied to systems with long dead times, i.e., the time delay between a control input and the corresponding system response.

We present a generic online control framework to address resource management problems for distributed computing systems whose components are modeled as *switching hybrid systems*—a special class of hybrid systems where the set of possible control inputs is finite [2]. At each time instant, the control problem of interest is to optimize a (multi-variable) objective function specifying the trade-offs between achieving the desired QoS and the corresponding cost incurred in terms of resource usage. Control actions are obtained by optimizing system behavior, as forecast by a mathematical model, for the specified QoS criteria over a limited look-ahead prediction horizon. Both the control objectives and operating constraints are represented explicitly in the optimization problem and solved at each time instant. Our method applies to various resource management problems, from those with simple dynamics to more complex ones, in-

*This work is sponsored in part 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.

WOSS'04 Oct 31-Nov 1, 2004 Newport Beach, CA, USA

cluding systems with long delay or dead times, and those with non-linear behavior. It can also accommodate changes to the behavioral model itself, caused by resource failures.

We have previously used the forementioned control scheme to address various resource management problems in computing systems with encouraging results [8, 1, 7]. This paper describes the overall control framework and introduces the hierarchical control concepts necessary to tackle complex and large-scale distributed systems. We also present case studies from [8] and [1], and discuss future work.

The rest of this paper is organized as follows. Section 2 discusses key control concepts while Section 3 presents two case studies selected from previous work. We conclude the paper with a discussion on future work in Section 4.

2. ONLINE CONTROL CONCEPTS

Fig. 1 shows the key components of a self-managing computing system: (1) the behavioral model, (2) the online controller, and (3) the QoS specification. Signal and parameter estimators may also be added to extract information about, and build an accurate model of the operating environment.

Forecasting Techniques. To estimate system behavior over the prediction horizon, changes in operating conditions such as workload variations and potential resource failures must be predicted and supplied to the corresponding model. We have previously used Box-Jenkins ARIMA models [3] and Kalman filters [6] to predict variations in web server workloads [8, 1].

System Model. The dynamics of a distributed system can be very complex depending on the number and interaction between components and the operating environment. However, the dynamics of individual components is usually much simpler. Note that the control approach assumes a switching hybrid system; only a finite number of options are available to change its behavior. The following discrete-time state-space equation describes the system dynamics:

$$x(t+1) = \Phi(x(t), u(t))$$

where $x(t)$ and $u(t) \in \{u_1, u_2, \ldots, u_r\}$ denote the sampled form of the continuous state vector and the discrete valued input vector at time t, respectively. This general model also describes both nonlinear and piecewise linear hybrid systems. Behavioral models of individual components can be composed using well-defined interaction schemes including shared variables, input/output connections, and synchronization events, to obtain the overall system model.

Complex distributed systems can be modeled using a hierarchical approach where higher-level models capture abstract and composite behavior of immediate low-level components. The high-level models only contain information relevant to the shared objectives of lower-level components. On the other hand, the lower-level models contain specific information about the corresponding system component and, therefore, may optimize performance with respect to local specifications while maintaining those restrictions imposed by the higher-level abstraction (global specification).

Performance Specifications. Computing systems must achieve QoS objectives while satisfying certain constraints imposed by the operating environment. Performance objectives can assume two main forms: set-point regulation and utility optimization. *Set-point* regulation requires that the underlying parameters or variables be maintained at a specific level (region) or follow a certain pattern (trajectory); for example, utilization level of a server, minimum acceptable data transmission rate, and maximum response time guaranteed to the end user. *Utility optimization* is used to maximize (minimize) a given performance measure represented as a function of state and input variables. A weighted norm is typically used as a performance function in which the corresponding variables are lumped together with different weights reflecting their contribution to the overall system utility and operation cost. It is easy to see that set-point regulation is a special case of utility optimization, namely, when the optimal value of the utility function is known.

Operating requirements for computing systems may also include strict and soft constraints on both system variables and control inputs. Generally, strict constraints are expressed as a feasible domain (region) for the composite space of a set of system variables (possible including control inputs). A soft constraint is also associated with a region in the composite space of a set of system variables (typically in the neighborhood of a hard constraint region)and is represented by a cost function mapping each point in the underlying domain to a value denoting the corresponding penalty. A soft constraint, therefore, is another form of an optimization specification, in which the system is required to minimize the associated cost function.

Online Control Algorithm. Given a hybrid system model, the online controller aims to satisfy the desired QoS requirements by continuously monitoring the current system state and selecting the inputs that best satisfy them. The controller must also keep the system stable within the domain satisfying the specification. In this setting, the controller is simply considered an agent that applies a given sequence of events to achieve a certain objective.

The controller explores only a limited look-ahead horizon within the system state space and selects the next event based on the available information. In the case of a set-point specification, the next step is selected using a distance map defining how close the current state is to the desired set point x_s. The distance map can be defined for each system state x as $D(x) = \|x - x_s\|$, where $\|.\|$ is a proper norm. For performance specifications, the control input minimizing (maximizing) the given utility function is selected. This function assigns to each state, a cost associated with reaching and maintaining it. Starting from the current state, the control algorithm constructs a tree of all possible future states for the specified look-ahead horizon. The exploration procedure identifies the set of states that best satisfy the given specification as discussed above. A state x_m is then chosen from this set based on certain optimality criterion, or simply picked at random. The chosen state is then traced back to the current state and the input leading to x_m is used for the next step. Online control, however, poses two main technical challenges.

- *Controller feasibility.* To guarantee a working control strategy, two questions must be addressed: (1) Does there exist a set of initial states from which a sequence of control inputs can move the system into the desired operating region in a finite number of steps?; (2) Once in the operating region, can such a trajectory be maintained inside that small neighborhood?

- *Computational efficiency.* Since the control algorithm constructs a search tree up to the specified look-ahead horizon, its worst-case complexity is exponential with

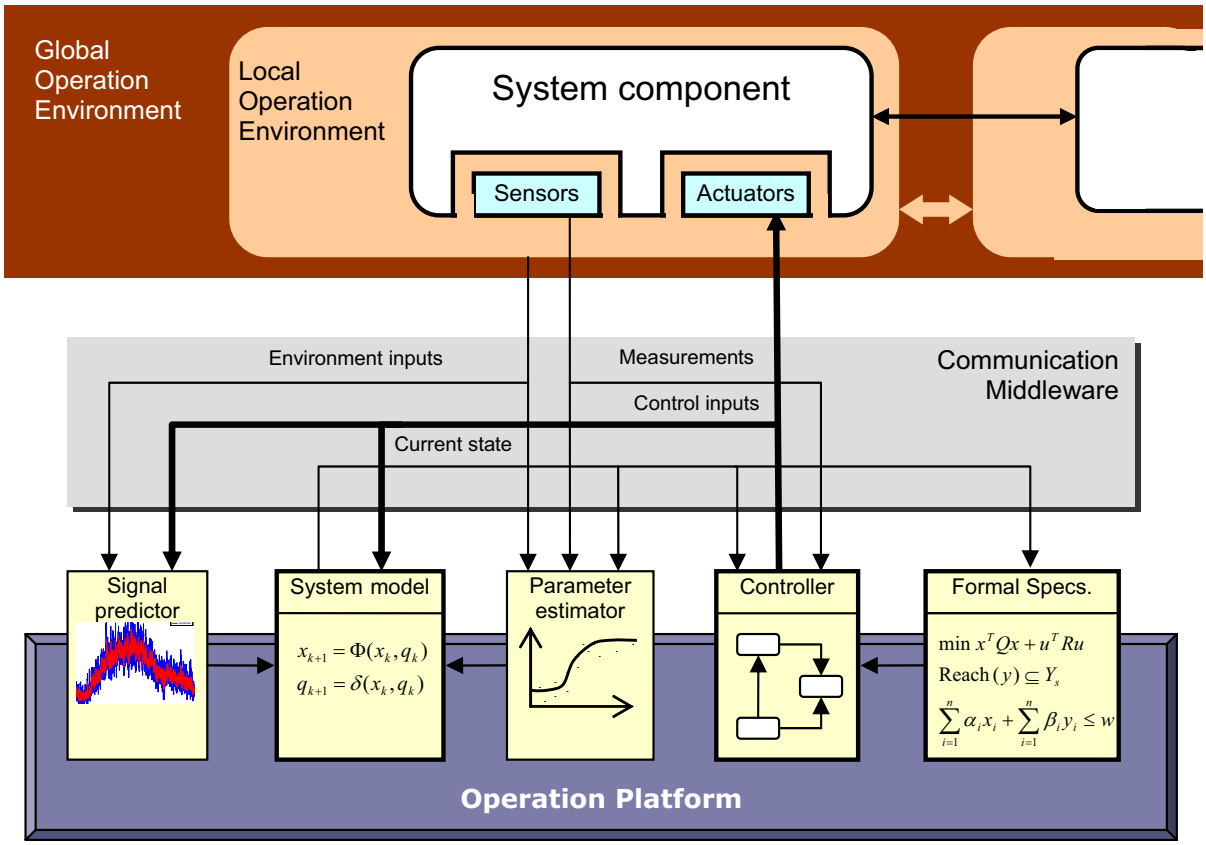

Figure 1: Key components of a self-managing computing system

the depth of the exploration tree and the number of control inputs. Therefore, information about the system dynamics must be utilized to restrict the search to a limited subset of possible future states.

Model Learning. Component behavior can be initially captured using standard system identification techniques [9]. The parametric form of the component model is assumed known, and the identification task is to estimate its parameters for various known operating modes. In most cases, components are dynamic processes whose behavior may be abstracted as simplified ARMA/ARIMA input-output models. Least-square estimation techniques may therefore be applied to determine model coefficients using real or simulated data available from the individual components. This task may be computationally intensive, but is performed at design time in off-line fashion.

When component behavior changes at run time due to varying environmental and operating conditions, model learning must be applied online. To feasibly achieve this, we assume that the new parameters affecting component behavior are known, and their values are measurable. The task, therefore, is to estimate the incremental changes to the model due to the new environmental parameters. Again, assuming that the model form is known, and the effects of the environmental variables are decoupled, the learning problem can be posed as one of parameter estimation. Computationally efficient algorithms must be developed to derive these estimates. Model learning also affects system performance; due to the delay in estimating new model parameters, controllers may operate in the interim with incomplete (and incorrect) models leading to short-term performance degradation.

Multilevel Distributed Control. Typically, in a distributed system with several components, each has its own requirement specification defining a desired operating region. In addition, a global performance requirement for the overall system may also be specified. Therefore, the controller must effectively coordinate (complex) interactions between the various components to ensure overall system performance. The nature of such systems suggests a decentralized and hierarchical control structure where each component has a local controller. Interaction between these controllers is managed via a global controller that aims to satisfy the global specifications of the overall system. Fig. 2 shows the structure of a two-level control scheme.

In Fig. 2, local-controller interactions are managed by a higher-level controller using an abstract system model containing information relevant to its objectives. The model includes, for instance, details of interactions between system components in terms of specific local variables contributing to a global objective. The abstract dynamics then represents how these variables would change in response to certain settings that the global controller can enforce via commands to the local controllers.

We envision a hierarchical structure where the high-level controller takes a long-term perspective of system dynamics, while the local ones act to optimize their components on a shorter-term basis. High-level commands are directed towards satisfying global QoS objectives, and act as a set

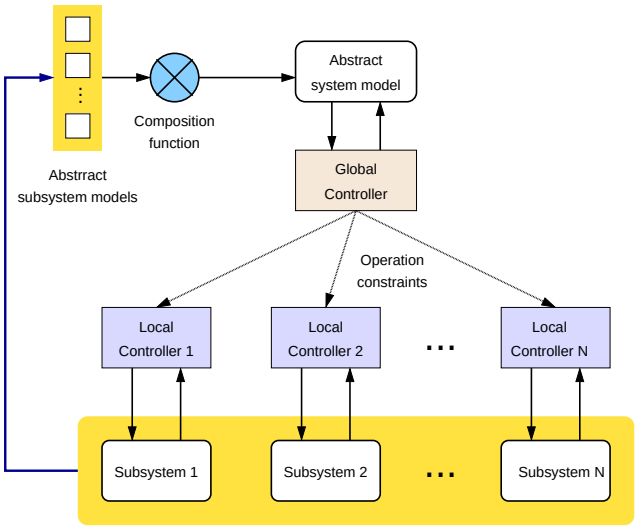

Figure 2: The Multilevel control structure

of operating constraints on each local controller. Each local controller then tries to optimize the performance of its underlying component using specific utility functions while satisfying any constraints imposed on it. Interaction between these controllers takes place as follows. Each local controller has a finite set of operating modes corresponding to specific parameter settings within its controlled component, e.g., a different operational requirement or input domain. The global controller then places (or restricts) each one in a mode aimed at satisfying the QoS objective. Local controllers optimize relevant parameters within that mode.

3. CASE STUDIES

The control approach proposed in Section 2 has been applied to some real-world applications. We briefly describe two systems currently under development using elements of our framework.

Low-Power Computing. We have applied the concepts presented in Section 2 to manage the power consumed by a computer processing a time-varying workload comprising HTTP and e-commerce related requests [7, 8]. Assuming a processor with multiple operating frequencies, an online controller is developed to achieve a specified response time $\omega_r ef$ for these requests while minimizing the operating frequency, and therefore, energy consumption (relates quadratically to the supply voltage which can be reduced at lower frequencies). Unlike classical feedback control where a continuous input (output) domain is assumed, the controller optimizes processor operation over a discrete state-space comprising a small number of control inputs.

Fig. 3 shows the key components of the control framework. Fig. 3(a) shows the processor model. Fig. 3(b) shows the overall structure of the controller where a queuing model, detailed in Fig. 3(c), describes processor behavior in terms of its queue size $q(t)$, average response time $\omega(t)$ and energy consumption $E(t)$, and an optimizer minimizes the utility function shown in Fig. 3(d). Future processor outputs, for a pre-determined prediction horizon are estimated during each sampling instant t using this model. These predictions depend on known values (past inputs and outputs) up to the sampling instant t, and on the future control signals which are inputs to the processor that must be calculated. A sequence of control signals (frequency inputs) resulting in the desired processor behavior is obtained for each step of the prediction horizon by minimizing the utility function in Fig. 3(d). The control signal corresponding to the first frequency in this sequence is applied as input to the processor during sampling instant t; the other inputs are rejected. The above steps are repeated again during the next sampling instant.

We have evaluated the performance of a prototype using e-commerce workloads with encouraging results. We are currently using concepts discussed in Section 2 to develop, and apply a hierarchical control scheme to distributed systems comprising multiple computers. During their operation, these system can be optimized for power consumption at multiple levels. For example, computers may be switched on (off) as needed in anticipation of future workload variations. Also, those operational computers can process their workload at reduced operating frequencies. This implies the need for a two-level control scheme where a global controller uses the forecast workload to determine how many processors to operate and the corresponding operating modes. Local controllers then aim to operate each processor in energy-efficient fashion within the specified mode.

Signal Detection System. In [1], we discuss how the control framework can be applied to a real-time signal detection application comprising multiple processors operating on digital signals to extract features such as human voice and speech from them. Signal processing is performed in distributed fashion. Incoming signals are stored in a global buffer and distributed to individual processors where they are locally queued. Each processor then examines a chunk of signal to identify designer-specified features. Clearly, detection accuracy improves with chunk size at the cost of increased computational complexity.

The control approach addresses the trade-off between accuracy and responsiveness to optimize overall system performance. A global controller estimates the signal arrival rate and receives average queue size and detection accuracy information from each processor over the past sampling period. This information is used to distribute a fraction of the new arrivals to individual processors and set its operating mode. We assume that the local controllers on processors operate in the following qualitative modes: (1) low mode, used when the signal arrival rate is low and large signal chunks may, therefore, be processed; (2) medium mode which applies to medium arrival rates; and (3) high mode used when signals arrive at a high rate and signal processing must be done in small chunks. Based on the signal arrival rate estimate, the global controller selects the appropriate modes of operation for local controllers as well as their share of the incoming signals to satisfy the system-level QoS goals. The controller optimizes system utility, a weighted norm of the quality of the signal detection and detection latency in terms of response time or queue size. A prototype system was developed for the Southwest Research Institute.

4. CONCLUSIONS

We have presented a generic online control framework to design self-optimizing computer systems where actions governing system operation are obtained by optimizing its behavior, as forecast by a mathematical model, over a limited time horizon. As specific applications of our approach,

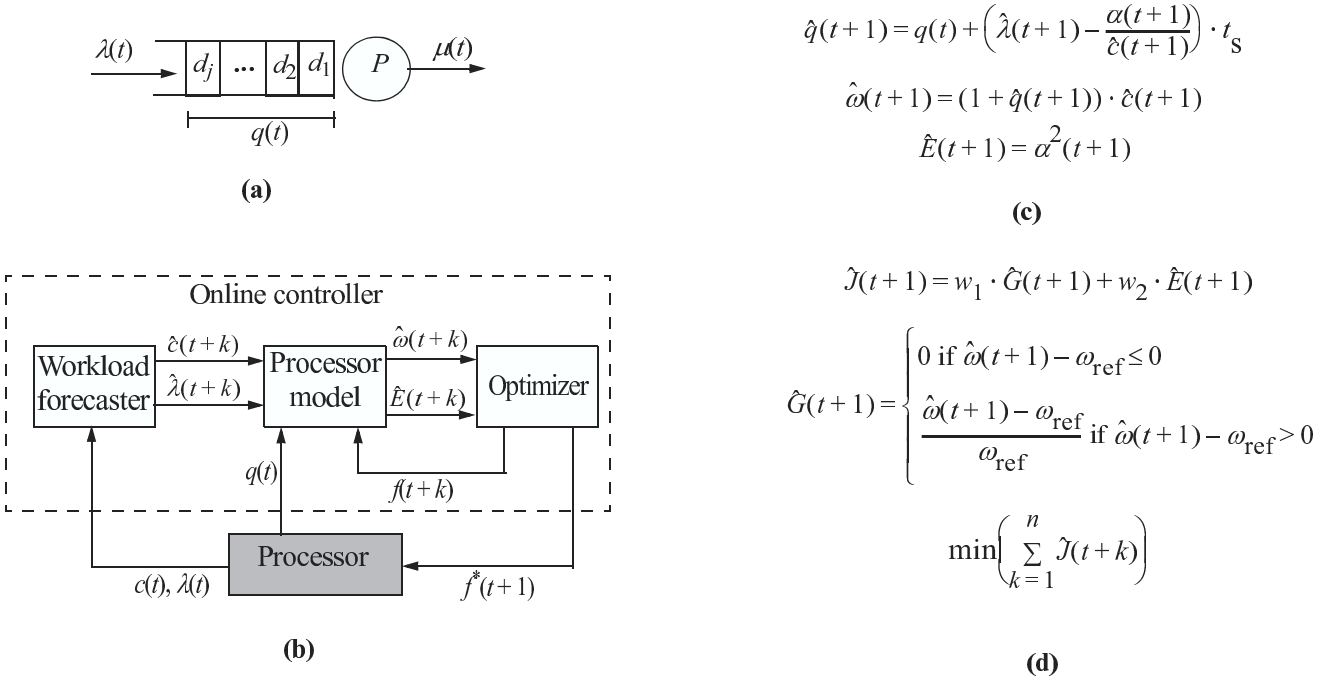

Figure 3: Application of online control to processor power management; (a) processor model, (b) overall structure of the online controller, (c) behavioral model of the processor, and (d) the cost function as a set-point specification

we presented two case studies. First, we developed an online controller to efficiently manage power consumption in processors under a time-varying workload. We then extended the online control method to distributed systems and applied it to a signal detection application.

We believe that the proposed control approach is applicable to other resource management problems in computer systems; for example, energy-efficient load balancing in clusters. A similar approach may also help design self-healing distributed systems. Certain failures due to design mistakes (configuration errors by system administrators) and hardware faults may be predicted shortly before their occurrence by analyzing corresponding performance variables [5]. The controller can then initiate the appropriate reconfiguration action such as switching on a backup computer in anticipation of such failures to prevent service disruptions.

5. REFERENCES

[1] S. Abdelwahed, N. Kandasamy, and S. Neema. Online control for self-management in computing systems. In *IEEE Real-Time & Embedded Tech. & Applications Symp.*, pages 368–375, 2004.

[2] S. Abdelwahed, G. Karsai, and G. Biswas. Online safety control of a class of hybrid systems. In *IEEE Conf. Decision and Control*, pages 1988–1990, 2002.

[3] G. P. Box, G. M. Jenkins, and G. C. Reinsel. *Time Series Analysis: Forecasting and Control.* Prentice-Hall, Upper Saddle River, New Jersey, 3 edition, 1994.

[4] A. Cervin, J. Eker, B. Bernhardsson, and K. Arzen. Feedback-feedforward scheduling of control tasks. *J. Real-Time Syst.*, 23(1–2), 2002.

[5] R. V. et al. Predictive algorithms in the management of computer systems. *IBM Systems Journal*, 41(3):461–474, 2002.

[6] A. C. Harvey. *Forecasting Structural Time Series Models and the Kalman Filter.* Cambridge University Press, Cambridge, 1989.

[7] N. Kandasamy and S. Abdelwahed. Designing self-managing distributed systems via online predictive control. Tech. Report ISIS-03-404, Vanderbilt University, 2003.

[8] N. Kandasamy, S. Abdelwahed, and J. P. Hayes. Self-optimization in computer systems via online control: Application to power management. In *IEEE Int'l Conf. Autonomic Computing*, pages 54–62, 2004.

[9] L. Ljung. *System Identification: Theory for the User.* Prentice Hall, Englewood Cliffs, NJ, 2 edition, 1998.

[10] C. Lu, J. Stankovic, G. Tao, and S. Son. Feedback control real-time scheduling: Framework, modeling and algorithms. *J. Real-Time Syst.*, 23(1/2):85–126, 2002.

[11] J. M. Maciejowski. *Predictive Control with Constraints.* Prentice Hall, Englewood Cliffs, NJ, 2002.

[12] S. Mascolo. Classical control theory for congestion avoidance in high-speed internet. In *Conf. Decision & Control*, pages 2709–2714, 1999.

[13] K. Ogata. *Modern Control Engineering.* Prentice Hall, Englewood Cliffs, NJ, 1997.

[14] S. Parekh, N. Gandhi, J. Hellerstein, D. Tilbury, T. Jayram, and J. Bigus. Using control theory to achieve service level objectives in performance management. 23(1/2):127–141, 2002.