Adaptive Fuzzy Control for Utilization Management

Mehmet H. Suzer and Kyoung-Don Kang
Department of Computer Science
State University of New York at Binghamton
{*msuzer,kang*}*@cs.binghamton.edu*

***Abstract*—An increasing number of real-time systems are embedded in mission critical systems such as target tracking systems, in which workloads may dynamically vary, for example, depending on the number of targets in the area of interest. Feedback control has been applied to support real-time performance in dynamic environments, producing promising initial results. However, mathematical system modeling necessary for feedback control is challenging. To reduce the difficulty of system modeling, we apply fuzzy control for direct nonlinear mappings between the utilization error (= target utilization − current utilization) and the workload adjustment required to achieve the target utilization via IF-THEN rules. Moreover, via online adaptation, our fuzzy controller can amplify or dampen its own fuzzy control signal, if necessary, to expedite the convergence to the desired utilization. In our simulation study, our approach quickly converges to the target utilization when the workload significantly changes. In contrast, the tested baselines oscillate between overload and underutilization.**

I. Introduction

A rapidly growing number of real-time systems, e.g., traffic control, target tracking, agile manufacturing, or SCADA (supervisory control and data acquisition) systems, run in unpredictable environments where workloads are unknown in advance and may significantly vary at run-time. For example, workloads may vary depending on the volume of the traffic or the number of targets in the area of interest [1]. During a cascading power failure, SCADA systems managing an electric grid may suffer a significant (up to an order of magnitude) load increase [2].

Feedback control has recently been applied to support real-time performance in the presence of unpredictable workloads [3], [4]. However, feedback control of real-time performance is not free of a drawback. Generally, real-time system behaviors in dynamic environments are complex and nonlinear. For example, many industrial processes are nonlinear [5]. Further, computational system dynamics are often stochastic. Thus, mathematical modeling of such systems required for feedback control is challenging. An easier-to-use yet effective approach is required for real-time performance management in dynamic environments.

In this paper, we apply *adaptive fuzzy control* technology to directly control nonlinear system behaviors to achieve the desired utilization set-point in soft real-time systems potentially having dynamic workloads. It is critical for a real-time system to avoid overloads; however, traditional real-time scheduling [6] is not directly applicable if the workload is not precisely known *a priori*. Feedback-based utilization control [3], [4] has been studied to avoid overloads by supporting the desired utilization set-point such as 90%, while avoiding severe underutilization. However, we are not aware of any prior work that applies adaptive fuzzy control to alleviate the difficulty of modeling the real-time system controlled by a feedback controller.

Fuzzy control is essentially a direct nonlinear mapping between its input, e.g., the utilization error = utilization set-point − current utilization, and output, e.g., the required workload adjustment to achieve the utilization set-point, unlike other controllers such as linear time invariant controllers, lead lag controllers, or state feedback controllers [7]. Fuzzy control provides formal techniques to represent, manipulate, and implement human experts' heuristic knowledge for controlling a plant, e.g., a real-time system, via IF-THEN rules rather than relying on mathematical modeling of the plant.

Notably, our fuzzy control system can automatically adapt its own control actions considering the current system behavior. This sharply contrasts to linear time invariant controllers such as [3], [4], which use fixed control gains regardless of the current system status. By applying adaptive fuzzy control, we aim to support the utilization set-point even under large workload variations, i.e., disturbances. In addition to the rule-base computing the fuzzy utilization control signal, we design another rule-base to either amplify or dampen the fuzzy control signal, if necessary, to expedite the convergence to the set-point, while avoiding oscillations.

We compare the performance of our adaptive fuzzy control system, via an extensive simulation study, to several baseline approaches including a linear PI controller for utilization control [3] and a non-adaptive fuzzy controller, which is similar to [8]. Overall, the adaptive fuzzy controller considerably outperforms the baselines for workloads involving large disturbances by consistently supporting the utilization set-point. In contrast, the baselines show large utilization fluctuations when the load varies.

The rest of the paper is organized as follows. Section II describes the overall structure of our adaptive fuzzy control system and gives backgrounds in fuzzy control. Section III describes the design of our adaptive fuzzy controller and rule-bases. Performance evaluation results are described in Section IV. Related work is discussed in Section V. Finally, Section VI concludes the paper and discusses future work.

II. Overall Structure and Backgrounds

In this section, the overall structure of our adaptive fuzzy control system, fuzzy control terminology, and basic fuzzy control mechanisms are discussed.

A. Overall Structure

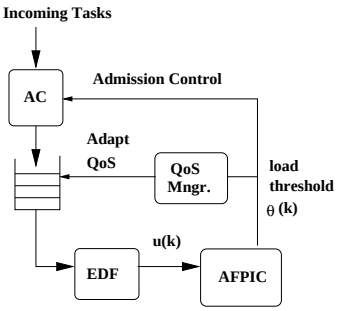

Fig. 1. Closed-Loop Real-Time System

The high level structure of our closed-loop real-time system is shown in Figure 1. Real-time tasks are scheduled in an EDF (earliest deadline first) [9] manner. To achieve the utilization set-point U_s, our AFPIC (Adaptive Fuzzy Proportional and Integral Controller) in Figure 1 computes the acceptable workload threshold $\theta(k)$ based on the error $e(k)$ and change of the error $\Delta e(k)$ for the utilization $u(k)$ measured in the k^{th} sampling period where:

$$e(k) = U_s - u(k) \tag{1}$$

and

$$\Delta e(k) = e(k) - e(k-1). \tag{2}$$

Specifically, $\theta(k)$ is computed at the k^{th} sampling instance, i.e., the end of the k^{th} sampling period, based on $e(k)$ and $\Delta e(k)$. The feedback control signal $\theta(k)$ will be used to support U_s in the $(k+1)^{th}$ sampling period. This feedback control procedure is repeated at every sampling instance to support U_s.

In this paper, we consider a periodic task model. A task T_i is defined by $< C_i, P_i, D_i, U_i, Q_i >$ where C_i is the *estimated execution time*, P_i is the task period, D_i is the deadline, and Q_i is the task QoS (quality of service) level. The actual execution time of T_i is *not* precisely known in advance. Thus, only the *estimated utilization* $U_i = C_i/P_i$ is known. Further, $D_i = P_i$. In our task model, a task T_i consists of a mandatory part and an optional part, following the imprecise computation model [10]. Under overload, only the mandatory part of T_i can be executed to reduce the workload, for example, by producing a low quality image for traffic control or target tracking. As a result, C_i, U_i, and Q_i are reduced. Specifically, the real-time system computes the total estimated utilization $U_t = \sum_{i=1}^{N} U_i$ for all the tasks currently in the system at the k^{th} sampling instance. If $U_t > \theta(k)$, the QoS manager in Figure 1 degrades the QoS of T_i in the system and decreases U_t by $C_{i,o}/P_i$ where $C_{i,o}(< C_i)$ is the estimated execution time of T_i's optional part. It repeats this step for the other tasks in the system as long as $U_t > \theta(k)$. Under severe overload, $U_t > \theta(k)$ even after degrading every task in the system. In this case, incoming tasks are rejected. By adapting the QoS before applying admission control, we can accept more tasks.

The admission controller (AC) in Figure 1 admits tasks based on the estimated utilization. To support U_s, it accepts a task T_j arriving in the $(k+1)^{th}$ sampling period only if $U_j + U_t \leq \theta(k)$. In an open-loop approach without feedback control, $\theta(k) = U_s$ for arbitrary k. As a result, the system could be overloaded (or underutilized), if the actual execution time is longer (or shorter) than the estimated execution time. In contrast, our approach adapts the acceptable load threshold to support U_s for dynamic workloads.

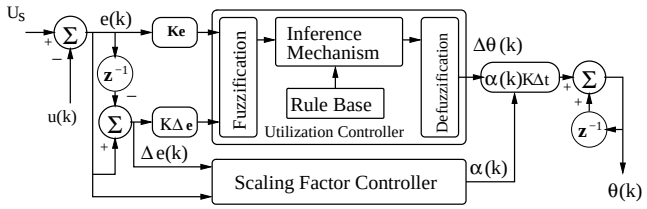

Fig. 2. Adaptive Fuzzy PI Controller

Figure 2 shows the structure of the AFPIC considered in this paper. Our AFPIC consists of a fuzzy utilization controller and a scaling factor controller for utilization control and adaptation, respectively. Although a fuzzy PI controller works well for linear and first order systems, it shows poor performance for nonlinear systems [5]. For this reason, we have a scaling factor controller that continuously adapts the scaling factor $\alpha(k)$, if necessary, to meet the set-point in the presence of dynamic workloads. Based on $e(k)$ and $\Delta e(k)$ (Eq 1 and Eq 2), the fuzzy utilization controller in Figure 2 computes the required threshold adjustment $\Delta\theta(k)$ for the next sampling period. $\Delta\theta(k)$ is multiplied by $\alpha(k)K_{\Delta t}$ where the scaling factor controller computes $\alpha(k)$ at the k^{th} sampling instance for adaptation. In contrast, $K_e, K_{\Delta e}$, and $K_{\Delta t}$ are fixed.

Fuzzy logic controllers (FLCs) are used very widely in industry. Especially, PID (proportional, integral, and derivative), PI, and PD type FLCs are most common. In this paper, we consider a fuzzy PI controller, as PD FLCs are only suitable for a limited class of systems [11]. Moreover, rule-base design and tuning for PID FLCs are complicated due to many parameters [5]. Our controller shown in Figure 2 has two inputs, i.e., the utilization error and derivative of the error. It also has an integrator at its output. Without an integrator, the controller would be a PD type FLC. One may think that, since our controller contains $e(k)$ and $\Delta e(k)$ as input and an integrator at the output, it is a PID controller. In fact, however, it is a PI controller. A standard PID controller computes an exemplar control signal $c(k)$ at the k^{th} sampling instance as follows:

$$c(k) = K_p e(k) + K_i \sum_{j=0}^{k} e(j) + K_d \Delta e(k)$$

and its iterative form, which is more efficient, is obtained by taking the derivative of the both sides of the equation:

$$\Delta c(k) = K_p \Delta e(k) + K_i e(k) + K_d \Delta^2 e(k).$$

Since $\Delta c(k) = c(k) - c(k-1)$, the above equation can be rewritten as:

$$c(k) = c(k-1) + K_p \Delta e(k) + K_i e(k) + K_d \Delta^2 e(k).$$

Compared to this equation, one can observe that our controller in Figure 2 has $e(k)$ and $\Delta e(k)$ terms, but it has no $\Delta^2 e(k)$ term. Thus, it is a PI controller. A description of fuzzy control terminology and basic fuzzy control mechanisms shown in Figure 2 follows.

B. Fuzzy Control Terminology and Mechanisms

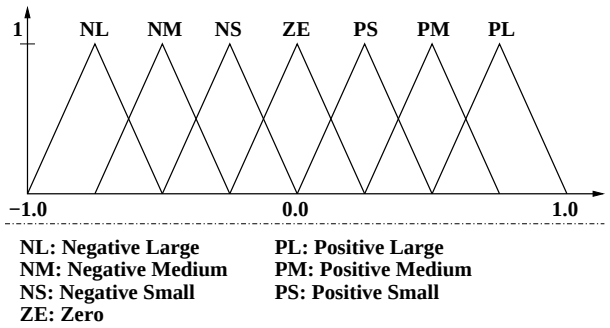

Fig. 3. Input and Output Membership Functions for the Utilization Controller

- **Universe of Discourse:** The universe of discourse is the domain of the inputs to a fuzzy controller. As shown in Figure 3, the universe of discourse for utilization errors is [-1, 1], since $e(k) = -1$ if the set-point $U_s = 0$ and current utilization $u(k) = 1$ in Eq. 1.[1] On the other hand, $e(k) = 1$ if $U_s = 1$ and $u(k) = 0$. In this paper, the universe of discourse for $\Delta e(k)$ is [-1, 1] too.
- **Linguistic Variables:** Linguistic variables describe the inputs and output(s) of a fuzzy controller. For instance, two inputs to the fuzzy controller at the k^{th} sampling instance are described as *error* (fuzzified $e(k)$) and *change in error* (fuzzified $\Delta e(k)$), while the output from the fuzzy system is *control signal*, i.e., the load threshold adjustment required to achieve the set-point U_s expressed linguistically. The **fuzzification interface** in Figure 2 converts $e(k)$ and $\Delta e(k)$ to the corresponding linguistic values defined next. (A description of the fuzzification process is discussed in Section III.)
- **Linguistic Values:** Linguistic variables are associated with linguistic values to describe characteristics of the variables. A linguistic variable *error*, for example, could be associated with linguistic values Large, Small, or Zero at a sampling instance. Figure 3 shows linguistic values for the linguistic variables *error*, *change in error*, and threshold *control signal* used in this paper.
- **Linguistic Rules:** A set of IF *premise* THEN *consequent* rules are used to map the inputs to output(s) of a fuzzy controller. For example, if $error = NL$ (negative large) and *change in error* $= NL$ at the k^{th} sampling instance, then the system is overloaded, i.e., $U_s < u(k)$, and the degree of overload is increasing significantly. Thus, the corresponding rule generates a NL signal to significantly reduce the workload by largely decreasing the acceptable load threshold $\theta(k)$. In fact, *error* and *change in error* can only be NL for a low U_s such as 0.3. For a high U_s such as 0.9, NS (negative small) already indicates overload. To avoid potential performance oscillations under overload, for a high U_s, our controller generates a relatively small control signal rather than generating aggressive control signals. Thus, our controller can be used for a broad range of set-points. This flexibility is another virtue of fuzzy control. Although we have verified that AFPIC can support various set-points, in Section IV, we only present the performance evaluation results for a relatively high $U_s = 0.9$, which is hard to support without saturating the CPU.
- **Rule-Base:** The *rule-base* in Figure 2 has a set of IF-THEN rules dictating how to achieve U_s according to the fuzzified linguistic values of $e(k)$ and $\Delta e(k)$, i.e., *error* and *change in error*. The *inference mechanism* in Figure 2 evaluates which control rules are relevant at the current time. It also decides what the fuzzy control signal should be by looking up the rule-base table based on the fuzzified $e(k)$ and $\Delta e(k)$ values. The *defuzzification interface* in Figure 2 converts the fuzzy *control signal* reached by the inference mechanism into the control signal $\Delta\theta(k)$ expressed as a real number. $\theta(k)$ in Figure 2 is the control input to the real-time system that adjusts the workload according to $\theta(k)$ by QoS adaptation and admission control. Our linguistic rules, inference, and defuzzification are described in Section III.
- **Membership Functions:** The horizontal axis of Figure 3 represents $e(k)$, $\Delta e(k)$, or $\Delta\theta(k)$ and the vertical axis indicates the membership value. A membership function (MF) quantifies the *certainty* an $e(k)$, $\Delta e(k)$, or $\Delta\theta(k)$ value to be associated with a certain linguistic value. For MFs, we use symmetric triangles of an equal base and 50% overlap with adjacent MFs, similar to [5], [7]. In Figure 3, for example, if $e(k) = 0.25$, the membership function for PS (Positive Small) $\mu_{PS}(0.25) = 1$ and it is 0 for the other linguistic values. If $\Delta e(k) = 0.0625$, $\mu_{ZE}(0.0625) = 0.75$, $\mu_{PS}(0.0625) = 0.25$, and it is 0 for the other linguistic values.

[1] Although setting U_s to 0 is impractical in reality, we intend to consider the full range of possible error values for fuzzy control as recommended by [5].

III. Adaptive Fuzzy Utilization Control

In this section, the design of our adaptive fuzzy control system is discussed.

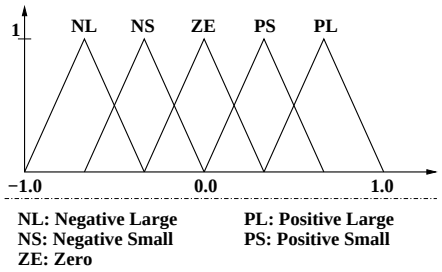

Fig. 4. Input MFs for the Scaling Factor Controller

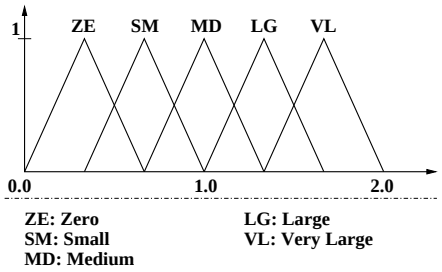

Fig. 5. Output MFs for the Scaling Factor Controller

A. Fuzzification, Inference, and Defuzzification

All MFs for the utilization controller's inputs and output as well as the scaling factor controller's inputs, i.e., $e(k)$, $\Delta e(k)$, and $\Delta\theta(k)$ at the k^{th} sampling instance, are defined in the common normalized domain [-1, 1] as shown in Figures 3 and 4. In contrast, as shown in Figure 5, the MFs for the scaling factor controller's output are defined in the normalized domain [0, 2], because the scaling factor either amplifies or shrinks the fuzzy control signal $\Delta\theta(k)$ in Figure 2 without changing the sign.

Based on the fuzzified $e(k)$ and $\Delta e(k)$, the inference mechanism in Figure 2 determines which rules to apply at the k^{th} sampling instance. For example, suppose $e(k)$ = 0.25 and $\Delta e(k)$ = 0.0625. According to Figure 3, the certainty $\mu_{PS}(0.25)$ = 1 for $e(k)$ and $\mu_{ZE}(0.0625)$ = 0.75 and $\mu_{PS}(0.0625)$ = 0.25 for $\Delta e(k)$ as discussed in Section II-B. To compute the certainty value of the premise in the corresponding IF *premise* THEN *consequent* rule(s), we take the minimum between the certainty values of $e(k)$ and $\Delta e(k)$, following one of the most common approaches [7], [5], [12]. Thus, $\mu(PS, ZE) = min\{1, 0.75\} = 0.75$ and $\mu(PS, PS) = min\{1, 0.25\} = 0.25$.

The inference engine looks Table I up to find that $rule(PS, ZE)$ = PS and $rule(PS, PS)$ = PM. Let $\mu(i, j)$ denote the membership function and $c(i, j)$ denote the center of the MF of the consequent of the $rule(i, j)$. For triangle MFs, the center is the value on the x axis at the middle of the triangle and the fuzzy utilization control output is [7]:

$$\Delta\theta(k) = \frac{\sum_{i,j} c(i,j) \cdot \mu(i,j)}{\sum_{i,j} \mu(i,j)} \quad (3)$$

For example, in Figure 3, the center of PS and PM is 0.25 and 0.5, respectively. Thus, $\Delta\theta(k) = (0.25 \cdot 0.75 + 0.5 \cdot 0.25)/(0.75 + 0.25) = 0.3125$. The scaling factor controller computes the scaling factor in a similar way using $e(k)$, $\Delta e(k)$, and the rule-base for scaling factor control shown in Table II. A detailed discussion of the rule-bases for utilization and scaling factor control follows.

B. Fuzzy Rules

For utilization control, we define fuzzy control rules using linguistic variables associated with linguistic values NL, NM, NS, ZE, PS, PM, and PL as shown in Figure 3 where each value has a different sign and/or size from the others. As shown in Figure 6, there are five zones that characterize the utilization controller's action, from which we derive the **rule-base for fuzzy utilization control** described in Table I as follows.

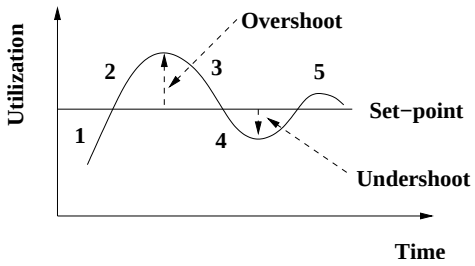

Fig. 6. Fuzzy Utilization Control Characteristics

1) $e(k) > 0$ and $\Delta e(k) < 0$ at the k^{th} sampling instance: In this zone, the actual utilization is smaller than the target utilization, but it comes closer to the set-point according to Eq. 1 and Eq. 2. The utilization controller should apply a small positive $\Delta\theta(k)$ to avoid a potential overshoot, i.e., overload, while increasing the utilization.
2) $e(k) < 0$ and $\Delta e(k) < 0$: In this zone, the actual utilization is larger than the set-point and it is further increasing. Hence, the controller should apply $\Delta\theta(k) < 0$ to reverse the current trend.
3) $e(k) < 0$ and $\Delta e(k) > 0$: In this zone, the actual utilization is higher than the target utilization, but it comes closer to the set-point. Since the actual utilization is converging to the target, the controller should apply a small negative $\Delta\theta(k)$ to avoid a potential undershoot, i.e., underutilization, while reducing the utilization.
4) $e(k) > 0$ and $\Delta e(k) > 0$: In this zone, the actual utilization is lower than the set-point and it is further decreasing. Thus, $\Delta\theta(k) > 0$ to reverse the current trend.
5) $|e(k)| < \epsilon$ and $|\Delta e(k)| < \epsilon$ where ϵ is a predefined small positive real number: The utilization converges to the set-point and $|e|$ and $|\Delta e|$ are small. This means that the real-time system is in the steady state. Therefore, the controller applies a zero control signal.

Table II shows the **rule-base for scaling factor control**. The linguistic variables of scaling factor control assume linguistic values NL, NS, ZE, PS, PL for input and ZE, SM, MD, LG, VL for output as shown in Figures 4 and 5. Note that the output of the scaling factor controller does not have any negative

e/Δe	NL	NM	NS	ZE	PS	PM	PL
NL	NL	NL	NL	NL	NM	NS	ZE
NM	NL	NL	NL	NM	NS	ZE	PS
NS	NL	NL	NM	NS	ZE	PS	PM
ZE	NL	NM	NS	ZE	PS	PM	PL
PS	NM	NS	ZE	PS	PM	PL	PL
PM	NS	ZE	PS	PM	PL	PL	PL
PL	ZE	PS	PM	PL	PL	PL	PL

TABLE I
UTILIZATION CONTROL RULES

e/Δe	NL	NS	ZE	PS	PL
NL	MD	LG	VL	LG	MD
NS	SM	MD	LG	MD	SM
ZE	SM	SM	ZE	SM	SM
PS	SM	MD	LG	MD	SM
PL	MD	LG	VL	LG	MD

TABLE II
SCALING FACTOR CONTROL RULES

linguistic value, because it either amplifies or shrinks the control signal $\Delta\theta(k)$ without changing the sign to compensate the limitations of the fuzzy utilization controller as follows.

1) $|e(k)|$ is large and $|\Delta e(k)|$ is large or $|e(k)|$ is small and $|\Delta e(k)|$ is small: In these zones, the utilization controller's action is sufficient to correct the error. Thus, it must be left the same. We set $\alpha(k) = 1$ in this case.
2) $|e(k)|$ is large and $|\Delta e(k)|$ is small or $|e(k)|$ is small and $\Delta e(k) = 0$: In these zones, the utilization controller is correcting the error, but with an insufficient control signal. Thus, the control signal $\Delta\theta(k)$ must be amplified to make the correction faster. Specifically, we choose $\alpha(k) = 1.5$.
3) $|e(k)|$ is large and $\Delta e(k) = 0$: In this zone, $\Delta\theta(k)$ is so small that the controlled system does not respond to that signal. Therefore, it must be significantly amplified to get a response from the plant. Specifically, we set $\alpha(k) = 2$.
4) $e(k) = 0$ and $|\Delta e(k)|$ is small or $e(k)$ is small and $|\Delta e(k)|$ is large: In these zones, the utilization is fluctuating around the desired level, but cannot converge to the set-point. Thus, $\Delta\theta(k)$ must be weakened to let the plant reach the desired set-point. To this end, we select $\alpha(k) = 0.5$.
5) $e(k) = 0$ and $\Delta e(k) = 0$: In this zone, the utilization is equal to the set-point. Hence, $\alpha(k) = 0$.

Overall, our adaptive fuzzy control only requires small rule-bases, efficient table look-ups, and control signal computation, which all finish in constant time. Thus, it is computationally lightweight. Specifically, we choose K_e = 1.0, $K_{\Delta e}$ = 0.5, and $K_{\Delta t}$ = 0.1, since $e(k)$ needs to be fully considered, while $\Delta e(k)$ and $\Delta\theta(k)$ need to be damped to avoid oscillations. The stability of our fuzzy control system can be proved by the Lyapunov method [7]. This is reserved for future work.

Notation	Value
EET_i	uniform [5ms, 20ms]
$slack$	uniform [10, 20]
$D_i(= P_i)$	$slack \cdot EET_i$
QoS levels	low, high

TABLE III
WORKLOAD SPECIFICATION

IV. PERFORMANCE EVALUATION

For performance evaluation, we have developed a simulator to model the soft real-time system architecture depicted in Figure 1. For performance evaluation, the admission control and QoS adaptation components can be turned on or off. Also, the adaptive fuzzy controller can be replaced with different controllers. We measure the utilization, success ratio (i.e., the fraction of the submitted tasks finishing within their deadlines), and QoS for our AFPIC and the tested baseline approaches. For performance analysis, one simulation is run for 20 minutes. Each performance data is the average of 10 simulation runs with different seed numbers. We have derived 90% confidence intervals; however, we omit the confidence intervals as most of them are less than 3%. A discussion of the workloads, baselines, and performance results follows.

A. Baselines

For performance comparisons, we consider several baselines: `Open-Loop` simply admits all incoming tasks and provides the full QoS regardless of the current system status. `AC` applies admission control to incoming tasks based on their estimated utilization values and fixed load threshold $\theta = U_s$. `PI` employs the linear PI controller [3] to achieve the target utilization in real-time systems. We strictly follow the design procedure described in [3]. To support the stability, we have tuned the controller via the Root Locus method [13] as done in [3]. Due to space limitations, we do not repeat their control model. Interested readers are referred to [3]. `FPIC` employs the non-adaptive fuzzy PI utilization controller using the rule-base shown in Table I. FPIC is analogous to [8], which applies non-adaptive fuzzy control to QoS management in visual tracking. For AFPIC, PI, and FPIC, we use the 5s sampling period for fair comparisons. Further, the same QoS adaptation and admission control methods are used under overload. Hence, the only difference among AFPIC, PI, and FPIC is the way to compute the control signal.

B. Workloads

Our workload summarized in Table III is similar to [3], [14]. The workload used in [3] is developed to evaluate the performance of their linear PI controllers for utilization (and deadline miss ratio) management. Further, the workload used in [14] is originally derived from air traffic control. For task T_i, its estimated execution time EET_i is uniformly selected in a range [5ms, 20ms]. Its actual execution time $AET_i = etf \cdot EET_i$ where etf is the execution time factor [3]. If $etf > 1$, the system could admit too many

tasks based on the estimated execution times shorter than the actual execution times. Every task is periodic and task T_i's deadline and period are: $D_i = P_i = slack \cdot EET_i$ where $slack = uniform[10, 20]$. Also, we consider two QoS levels: low and high for executing only the mandatory part of a task and both the mandatory and optional parts, respectively. The QoS of a task is 1 if the both mandatory and optional parts are executed. It is considered 0.5 if only the madatory part is executed. Note that our approach is not limited to a specific number of QoS levels. We will consider more QoS levels in the future.

For performance evaluation, we model abrupt workload changes to stress the tested approaches. Initially, the load is 100% and $etf = 1$ but etf is suddenly increased to 2, 6, or 10 at 600s and maintained at the level until 1200s. As a result, the load becomes twice, six times, or ten times the system capacity at 600s. These situations may happen, for example, when real-time tasks are required to process too many traffic or surveillance images within the deadlines due to traffic accidents or appearances of multiple targets in the area of interest. We have also considered other etf values, the performance results are consistent with the ones reported here. Note that the tested approaches including AFPIC are unaware of etf changes. As a result, they can be overloaded, if they are not reactive enough. In the rest of the paper, we only show the performance of PI, FPIC, and AFPIC. OPEN-LOOP shows poor performance, because it simply admits all incoming tasks regardless of the current system behavior. Neither does it degrade QoS. AC admits too many tasks as $etf > 1$ under disturbances. Thus, it becomes overloaded, showing poor performance.

C. Performance Results

We have observed that PI, FPIC, and AFPIC closely support U_s in terms of the average. Due to space limitations, we do not plot the average utilization, which has little meaning under abrupt workload changes. Figures 7, 8, and 9 show the transient utilization for the 200%, 600%, and 1000% disturbances. In these figures, at the beginning of the simulation, PI shows an overshoot and its settling time is about 100s. Both FPIC and AFPIC cause no overshoot at start-up. Also, AFPIC's rise time−the time taken to reach the set-point from the beginning−is slightly shorter than FPIC's. Generally, a shorter rise time is better, if it does not cause an overshoot [13]. Hence, AFPIC shows the best performance up to approximately 100s, because it has no overshoot and the shortest rise time.

After 100s, the three controllers, i.e., PI, FPIC, and AFPIC, maintain the utilization near the set-point (90%) until the disturbance point at 600s as shown in Figures 7, 8, and 9. When the load suddenly increases at 600s, the utilization saturates at 100%. The controllers are required to recover from the saturation and re-converge to the set-point. As shown in the figures, PI suffers the largest utilization fluctuations, involving overshoots and undershoots, among the tested approaches. The magnitude of its utilization fluctuations is bigger than 30%

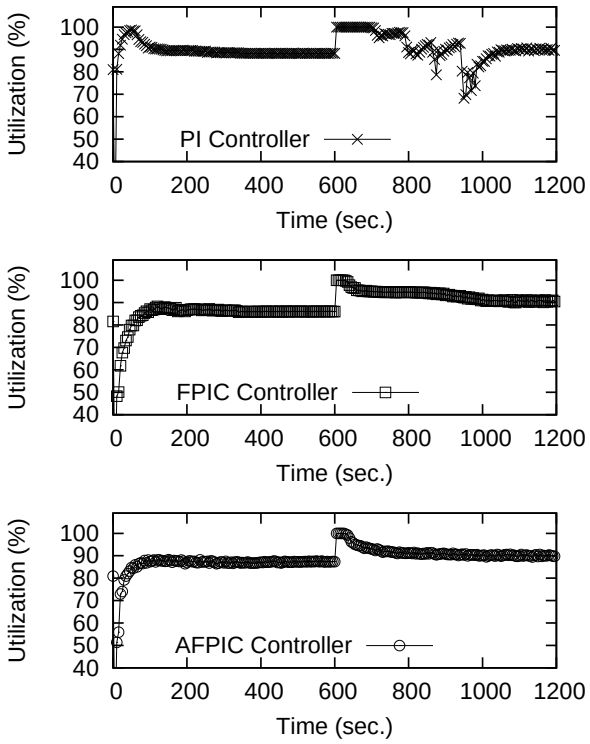

Fig. 7. Transient Utilization for 200% Disturbance

for several times. Further, it cannot converge to the target utilization for more than 400s after the disturbance at 600s. In Figure 9, PI cannot re-converge to the 90% set-point by the end of the simulation. Such large fluctuations and instability are unacceptable in real-time systems.

In Figures 7, 8, and 9, FPIC shows the more stable utilization than PI does. In Figure 7, its utilization is saturated only briefly when the load is increased to 200% at 600s, even though its settling time−the time taken to converge to the set-point−is approximately 400s. This is because FPIC is designed by directly considering nonlinear system behaviors represented by e and Δe. In contrast, PI has no mechanism to directly handle such conditions. As a result, PI has the longest settling time after the disturbance at 600s, if it ever re-converges to the set-point, as shown in Figures 7, 8, and 9. Although FPIC shows the better performance than PI in the figures, its utilization control performance is unsatisfactory under 600% and 1000% disturbances.

Overall, AFPIC shows the best transient utilization in Figures 7, 8, and 9. It quickly re-converges to the target utilization after a disturbance as shown in the figures. It re-converges in only about 50 seconds, i.e., 10 sampling periods, and the utilization fluctuates by less than 1% for the 200% disturbance and 3% for the larger disturbances after it becomes stable, considerably outperforming FPIC and PI.

We have evaluated the success ratio and QoS for all the tested approaches too. AFPIC achieved the highest success ratio followed by FPIC and PI in the sequence. When the workload changes abruptly at 600s, via online adaptation,

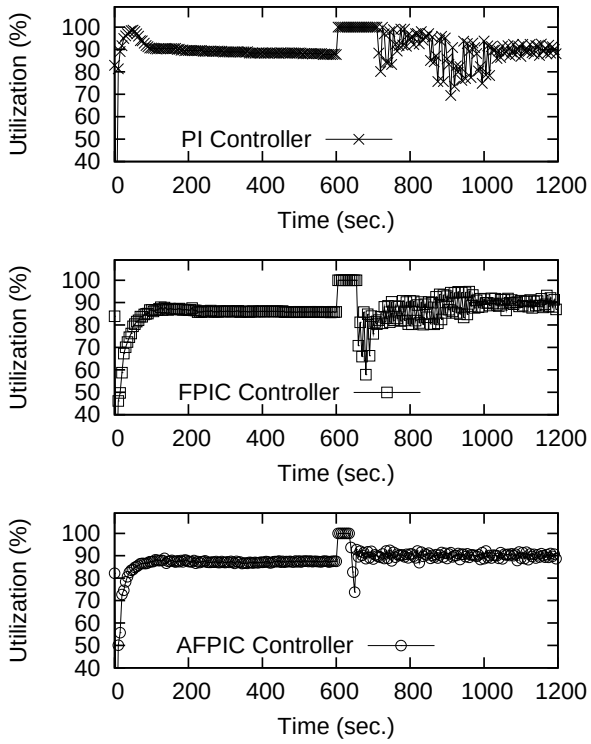

Fig. 8. Transient Utilization for 600% Disturbance

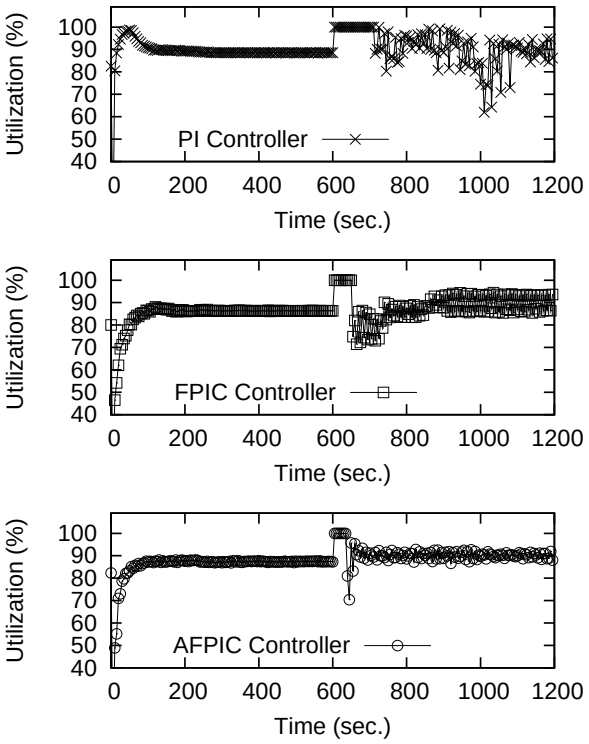

Fig. 9. Transient Utilization for 1000% Disturbance

AFPIC adapts the QoS faster than FPIC and PI do.

V. Related Work

Feedback control has recently been applied to real-time performance management in dynamic environments. A number of existing approaches for feedback control of real-time performance such as [3], [4] mathematically model real-time system behaviors via difference equations. To apply classical linear control theory, real-time system behaviors are approximated in a piecewise linear manner. However, linear approximation is not always applicable due to dynamic, potentially nonlinear behaviors of real-time systems interacting with physical environments. As control gains are determined offline, the closed-loop systems of these approaches cannot adjust its own control actions at run time considering the current status.

Model predictive control techniques are applied to dynamically identify the relation between the real-time task execution rate and utilization over a prediction horizon, which consists of a specified number of sampling periods, for utilization control in a multiprocessor environment [15]. Specifically, least square equations are solved for online system identification (SYSID) [13], [16] aiming to approximate the real-time system behavior. Unfortunately, this procedure further complicates the system modeling and controller tuning. It is not trivial to determine the appropriate prediction window size and system order [16], which can considerably affect control performance. Fuzzy control theory [7] is originally developed to address these modeling and design difficulties by allowing direct nonlinear mappings between system inputs and outputs. In this paper, we develop an adaptive fuzzy controller and show its effectiveness for utilization control.

Adaptive control [16] is applied to differentiated web caching services [17]. Diao et al [18] apply non-adaptive fuzzy control to maximize the profit in an email server. eQoS [12] supports, via adaptive fuzzy control, service differentiation in a web server. Different from them, our work focuses on supporting the desired utilization threshold to meet timing constraints in real-time systems operating in dynamic environments.

Very little prior work has applied fuzzy control to real-time performance management. It is applied to visual tracking [8]; however, their work only applies non-adaptive fuzzy control. Our approach is complementary to the control theoretic work discussed in this section in that we support adaptive fuzzy control of utilization by directly considering dynamic, nonlinear system behaviors, while reducing the complexity for real-time system modeling. Thus, our work can be considered a design alternative for real-time performance management based on control theory. To our knowledge, no previous work has applied adaptive fuzzy control to real-time performance management.

VI. Conclusions

Feedback control has been applied to real-time performance management in dynamic environments. In this paper, we aim to alleviate the difficulty of mathematical modeling necessary for feedback control of real-time performance. We apply adaptive fuzzy control to support, via IF-THEN rules, direct nonlinear mappings between the utilization error and workload

adjustment required to achieve the target utilization. Further, our adaptive controller can adjust its own behavior by amplifying or weakening the utilization control signal computed by the fuzzy controller, if necessary, to shorten the settling time, while avoiding potential oscillations. In the performance evaluation, our adaptive fuzzy controller considerably outperforms the tested baselines. In the future, we will investigate more effective approaches to adaptive fuzzy control of real-time performance.

ACKNOWLEDGMENT

This work was supported, in part, by a NSF grant CNS-0614771.

REFERENCES

[1] J. Loyall, R. Schantz, D. Corman, and J. P. andS. Fernandez, "A Distributed Real-Time Embedded Application for Surveillance, Detection, and Tracking of Time Critical Targets," in *The 11th IEEE Real-Time Embedded Technology and Applications Symposium*, 2005.

[2] R. Carlson, "Sandia SCADA Program High-Security SCADA LDRD Final Report," SANDIA, Tech. Rep., 2002, sANDIA Report SAND2002-0729.

[3] C. Lu, J. A. Stankovic, G. Tao, and S. H. Son, "Feedback Control Real-Time Scheduling: Framework, Modeling and Algorithms," *Real-Time Systems, Special Issue on Control-Theoretical Approaches to Real-Time Computing*, vol. 23, no. 1/2, May 2002.

[4] M. Amirijoo, J. Hansson, S. H. Son, and S. Gunnarsson, "Experimental Evaluation of Linear Time-Invariant Models for Feedback Performance Control in Real-Time System," in *Real-Time Systems*, vol. 35, no. 3, 2007.

[5] R. K. Mudi and N. R. Pal, "A Self-Tuning Fuzzy PI Controller," *Fuzzy Sets and Systems*, vol. 115, no. 2, pp. 327–338, April 2000.

[6] G. C. Buttazzo, *Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications*. Kluwer Academic Publishers, 1997.

[7] K. M. Passino and S. Yurkovich, *Fuzzy Control*. Addison-Wesley, 1998.

[8] B. Li and K. Nahrstedt, "A Control-Based Middleware Framework for Quality of Service Adaptations," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 9, 1999.

[9] C. L. Liu and J. W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment," *Journal of the ACM*, vol. 20, no. 1, pp. 46–61, 1979.

[10] K. J. Lin, S. Natarajan, and J. W. S. Liu, "Imprecise Results: Utilizing Partial Computations in Real-Time Systems," in *Real-Time System Symposium*, December 1987.

[11] H. Malki, H. Li, and G. Chen, "New Design and Stability Analysis of Fuzzy Proportional-Derivative Control Systems," *IEEE Transactions on Fuzzy Systems*, vol. 2, no. 4, pp. 245–254, 1994.

[12] J. Wei and C.-Z. Xu, "eQoS: Provisioning of Client-Perceived End-to-End QoS Guarantees in Web Servers," *IEEE Transactions on Computers*, vol. 55, no. 12, December 2006.

[13] C. L. Phillips and H. T. Nagle, *Digital Control System Analysis and Design (3rd edition)*. Prentice Hall, 1995.

[14] M. Xiong, K. Ramamritham, J. A. Stankovic, D. Towsley, and R. Sivasankaran, "Scheduling Transactions with Temporal Constraints: Exploiting Data Semantics," *IEEE Transactions on Knowledge and Data Engineering*, vol. 14, no. 5, pp. 1155–1166, September/October 2002.

[15] X. Wang, D. Jia, C. Lu, and X. Koutsoukos, "DEUCON: Decentralized End-to-End Utilization Control for Distributed Real-Time Systems," *IEEE Transactions on Parallel and Distributed Systems*, to Appear.

[16] K. J. Åström and B. Wittenmark, *Adaptive Control*, 2nd ed. Addison-Wesley, 1994.

[17] Y. Lu, A. Saxena, and T. F. Abdelzaher, "Differentiated Caching Services; A Control-Theoretical Approach," in *the 21st International Conference on Distributed Computing Systems*, 2002.

[18] Y. Diao, J. L. Hellerstein, and S. Parekh, "Using Fuzzy Control to Maximize Profits in Service Level Management," *IBM Systems Journal*, vol. 41, no. 3, 2002.