Evaluating Adaptive Resource Management for Distributed Real-Time Embedded Systems

Nishanth Shankaran*, Xenofon Koutsoukos, Douglas C. Schmidt, and Aniruddha Gokhale
{nshankar,koutsoukos,schmidt,gokhale}@dre.vanderbilt.edu

Dept. of EECS, Vanderbilt University, Nashville

Abstract. *A challenging problem faced by researchers and developers of distributed real-time and embedded (DRE) systems is devising and implementing effective adaptive resource management strategies that can meet end-to-end quality of service (QoS) requirements in varying operational contexts. This paper presents three contributions to research on adaptive resource management for DRE systems. First, we describe the structure and functionality of the Hybrid Adaptive Resource-management Middleware (HyARM), which provides advanced distributed resource management based on hybrid control theoretic techniques to monitor system utilization and adapt to fluctuations in workload. Second, we present an analytical model of HyARM that formalizes the control theoretic behavior of HyARM and conveys the relationship between the system resource utilization and application QoS. Third, we highlight the adaptive behavior of HyARM via experiments on a DRE multimedia system that distributes video in real-time. Our results indicate that the HyARM yields predictable, stable, and high system performance, even in the face of fluctuating workload and resource availability.*

1 Introduction

Real-time and embedded systems have traditionally been designed for contexts where operational conditions, input workloads, and resource availability are known *a priori*, and are subject to little or no change at run-time. In such *closed* environments, conventional scheduling techniques, such as rate monotonic scheduling [1], are often effective in allocating and managing system resources. There is increasing demand, however, to introduce more *adaptive* capabilities in *distributed* real-time and embedded (DRE) systems, such as autonomous air surveillance, total ship computing environments and hot rolling mills, that execute in *open* environments where system operational conditions, input workload, and resource availability cannot be characterized accurately *a priori*. In such *open* environments, conventional scheduling techniques are often ineffective at managing system resources, due to their inability to modify allocations at run-time based on resource availability and requirements. Resource management for open DRE systems therefore need mechanisms that *adapt* to dynamic changes in resource availability and requirements.

Achieving end-to-end real-time quality of service (QoS) is particularly important for open DRE systems that face resource constraints, such as limited computing power

* Contact author: nshankar@dre.vanderbilt.edu

and network bandwidth. Over-utilization of these system resources can yield unpredictable and unstable behavior, whereas under-utilization can yield excessive system cost. A promising approach to meeting these end-to-end QoS requirements effectively, therefore, is to develop and apply *adaptive middleware* [2, 3], which is software whose functional and QoS-related properties can be modified either

- **Statically**, *e.g.*, to reduce footprint, leverage capabilities that exist in specific platforms, enable functional subsetting, and minimize hardware/software infrastructure dependencies, or
- **Dynamically**, *e.g.*, to optimize system responses to changing environments or requirements, such as changing component interconnections, power-levels, CPU and network bandwidth availability, latency/jitter; and workload.

In open DRE systems, adaptive middleware must make such modifications dependably, *i.e.*, while meeting stringent end-to-end QoS requirements, which requires the specification and enforcement of upper and lower bounds on system resource utilization to ensure effective use of system resources. To meet these requirements, we have developed the *Hybrid Adaptive Resource-management Middleware* (HyARM), which is an open-source[1] distributed resource management middleware that helps ensure efficient, flexible, and predictable adaptive resource management, which is key to supporting the requirements of open DRE systems.

HyARM is based on hybrid control theoretic techniques [4]. Hybrid control techniques provide a theoretical framework for designing control of complex system with both continuous and discrete dynamics. In our case study, the task of adaptive resource management is to control the utilization of the different resources, which are described by continuous variables. We achieve that by adapting the resolution, which is modeled as a continuous variable, and by changing the frame-rate and the compression, which are modeled by discrete actions. We have implemented HyARM atop *The ACE ORB* (TAO) [5], which is an implementation of the Real-time CORBA specification [6].

This paper shows how we applied HyARM to an emergency response and surveillance multimedia system to support its resource management requirements by modifying video qualities, such as compression scheme, picture resolution, and frame rate. Ensuring end-to-end performance for this DRE system is hard in the face of limited computing and network resources. Our results show that (1) HyARM ensures effective system resource utilization and (2) end-to-end QoS requirements of higher priority applications are met, even in the face of fluctuations in workload.

The remainder of the paper is organized as follows: Section 2 describes the architecture, functionality, and resource utilization model of our DRE multimedia system case study; Section 3 explains the structure and functionality of HyARM; Section 4 presents a formal model of the resource management challenges in our case study; Section 5 evaluates the adaptive behavior of HyARM via experiments on our multimedia system case study; Section 6 compares our research on HyARM with related work; and Section 7 presents concluding remarks.

[1] The code and examples for HyARM are available at `www.dre.vanderbilt.edu/~nshankar/HyARM/`.

2 Case Study: DRE Multimedia System

This section describes the architecture and QoS requirements of our DRE multimedia system that contains two classes of applications (1) *QoS-enabled* and *best-effort*. In these applications, remote receivers are updated via base stations with real-time video of subjects of interest captured by unmanned air vehicles (UAVs). We use this system as a case study to motivate the design of HyARM described in Section 3 and provide the context for our empirical results described in Section 5.

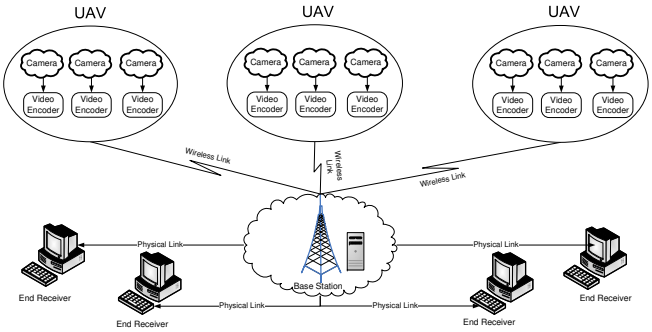

Fig. 1. DRE Multimedia System Architecture

DRE multimedia system architecture. The architecture for our DRE multimedia system is shown in Figure 1 and consists of the following entities:

- **Data source (video capture by UAV)**, where video is captured (related to subject of interest) by camera(s) on each UAV, followed by encoding of raw video using a specific encoding scheme and transmitting video to the next stage in the pipeline.
- **Data distributor (base station)**, where the video is processed to remove noise, followed by retransmission of the processed video to the next stage in the pipeline.
- **Sinks (command and control center)**, where the received video is again processed to remove noise, then decoded and finally rendered to end user displays.

These entities interact via the following pipeline of tasks: (1) video capture by cameras on the UAV, (2) encoding of the captured video, (3) transmission of video to a base station, (4) filtering of received video to remove noise, (5) transmission of processed video to receivers, and (6) presenting the received video to end-users via a graphical display.

Advances in video encoding and compression techniques [7–10] have made significant improvement the the mechanism used to (de)compress video. Common video compression schemes are MPEG-1 [11], MPEG-2 [12], Real Video, and MPEG-4 [13] yield high quality compressed video . Each compression scheme is characterized by its resource requirement, *e.g.*, the computational power to (de)compress the video signal, amount of network bandwidth required to transmit the compressed video signal. Properties of the compressed video, such as resolution and frame-rate determine both the quality and the resource requirement of the video.

QoS requirements in DRE multimedia system. Our multimedia system case study has the following end-to-end real-time QoS requirements:

- **Latency**, which should be minimized for end-users to view good quality full-motion video. Increased latency leads to delivery of stale video, which is undesirable.
- **Inter-frame delay**, also known as jitter, which impacts the smoothness and the clarity of the video. Due to the timeliness constraints on live video, however, our case study does not apply delayed buffering techniques [14, 15] to reduce jitter. Earlier studies [16, 17] have shown that human eyes can perceive delays more than 200ms, which we use as the upper bound on jitter for our case study.
- **Frame rate**, which determines the quality of the video [16]. Full motion picture is typically rendered at 30Hz, but smooth video with frame rate above 15Hz is acceptable for the applications in our multimedia system. Video of lower frame rates are not as smooth, but can be used as long as other qualities (such as latency and picture resolution) are acceptable.
- **Picture resolution (in pixels)**, which determines the quality of the video image. Video of lower resolution results in smaller picture and of lower clarity. Resolution should serve the purpose for which it is used, *e.g.*, video for emergency response and surveillance should be clear and large enough to distinguish various subjects of interest.

The QoS requirements described above can be classified as being either *hard* or *soft*. Hard QoS requirements should be met by the underlying system at all times, whereas soft QoS requirements can be missed occasionally.[2] For our case study, we treat QoS requirements such as latency and jitter as harder QoS requirements and strive to meet these requirements at all times. In contrast, we treat QoS requirements such as video frame rate, picture resolution, and depth of image as softer QoS requirements and modify these video properties adaptively to handle dynamic changes in resource availability effectively.

DRE multimedia system resources. There are two primary types of resources in our DRE multimedia system: (1) *processors* that provide computational power available at the UAVs, base stations, and end receivers and (2) *network links* that provide communication bandwidth between UAVs, base stations, and end receivers. Resource requirements and availability of these types of resources are subjected to the following types of dynamic changes:

- The computing power required by the video capture and encoding task depends on dynamic factors, such as speed of the UAV, speed of the subject (if the subject is mobile), and distance between UAV and the subject.
- The wireless network bandwidth available to transmit video captured by UAVs to base stations also depends on dynamic factors, such as the speed of the UAVs and the relative distance between UAVs and base stations. The bandwidth of the link

[2] Although *hard* and *soft* are often portrayed as two discrete requirement sets, in practice they are usually two ends of a continuum ranging from "softer" to "harder" rather than disjoint points.

between the base station and the end receiver is limited, but more stable than the bandwidth of the wireless network.

Two classes of applications – *QoS-enabled* and *best-effort* – use the multimedia system infrastructure described above to transmit video to their respective receivers. QoS-enabled applications has higher priority over best-effort class. In our case study, emergency response applications belong to QoS-enabled class and surveillance applications belong to best-effort class. Computing and bandwidth resources are allocated during system initialization for each class of application. For example, since a video stream from an emergency response application is of higher importance than a video stream from a surveillance application, it should receive more resources end-to-end.

In our multimedia system, resource utilization by applications are interdependent, *i.e.*, increase (or decrease) in CPU resource utilization by the application results in corresponding increase (or decrease) in network resource utilization. Since resource availability significantly affects application QoS, we use *current resource utilization* as the primary indicator of system performance. We refer to the current level of system resource availability as the *system condition*. Based on this definition, we can classify system conditions as being either *under*, *over*, or *effectively* utilized.

Under-utilization of system resources occurs when the current resource utilization is lower than the desired lower bound on resource utilization. In this system condition, large amounts of residual system resources (*i.e.*, network bandwidth and computational power) are available after meeting end-to-end QoS requirements of applications. These residual resources can be used to increase the QoS of the applications. For example, residual CPU and network bandwidth can be used to deliver better quality video (*e.g.*, with greater resolution, higher frame rate, and higher depth of image) to end receivers.

Over-utilization of system resources occurs when the current resource utilization is higher than the desired upper bound on resource utilization. This condition can arise from loss of network bandwidth, loss of computing power at base station, end receiver or at UAV, or may be due to an increase in resource demanded by the application. Over-utilization is generally undesirable since the quality of the received video (such as resolution and frame rate) and timeliness properties (such as latency and jitter) are degraded and may result in an unstable (and thus ineffective) system.

Effective resource utilization is the desired system condition since it ensures that end-to-end QoS requirements of the UAV-based multimedia system are met and utilization of both system resources, *i.e.*, network bandwidth and computational power, are within their desired utilization bounds. Section 3 describes techniques we applied to achieve effective utilization, even in the face of fluctuating resource availability.

3 Overview of HyARM

This section describes the architecture of the *Hybrid Adaptive Resource-management Middleware* (HyARM), which is an adaptive middleware for distributed resource management. HyARM employs advanced hybrid control theoretic techniques [4] to ensure efficient and predictable system performance by providing adaptive resource management, including monitoring of system resources and enforcing bounds on application resource utilization.

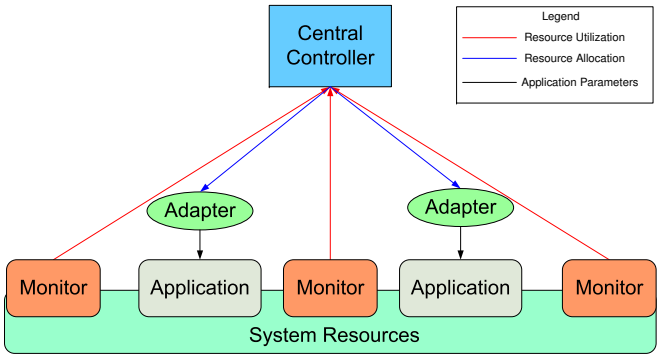

Fig. 2. HyARM Architecture

HyARM structure and functionality. HyARM is composed of three types of entities shown in Figure 2 and described below:

- **Resource monitors** observe the overall resource utilization for each type of resource and resource utilization per application. In our multimedia system, there are resource monitors for CPU utilization and network bandwidth. CPU monitors observe the CPU resource utilization of UAVs, base station, and end receivers. Network bandwidth monitors observe the network resource utilization of (1) network between UAVs and the base station and (2) physical link between the base station and end receivers.
- The **central controller** maintains the system resource utilization below a desired bound by (1) processing periodic updates it receives from resource monitors and (2) modifying the execution of applications accordingly, *e.g.*, by using different execution algorithms or operating the application with increased/decreased QoS. This adaptation process ensures that system resources are utilized efficiently and end-to-end application QoS requirements are met. In our multimedia system, the HyARM controller determines the value of application parameters, such as (1) video compression schemes, such as Real Video and MPEG-4, and/or (2) frame rate, and (3) picture resolution. From the perspective of hybrid control theoretic techniques [4], the different video compression schemes and frame rate form the *discrete variables* of application execution and picture resolution forms the *continuous variables*.
- **Application adapters** modify application execution according to parameters recommended by the controller and ensure that the operation of the application is in accordance the recommended parameters. In our multimedia system, the application adapter ensures that the video is encoded at the recommended frame rate and resolution using the specified compression scheme.

HyARM adapts to changes in resource requirements and availability by (1) *online* monitoring of resource utilization (via resource monitors), (2) dynamically modifying resource utilization by modifying application parameters (via the central controller), and (3) enforcing bounds on application resource utilization by operating the applications at above computed parameters (via application adapters). For example, network

monitors observe the utilization of network bandwidth and notify the central controller about any increases/decreases in bandwidth utilization. Depending on changes in resource utilization, the central controller can decrease or increase the QoS of lower and higher priority applications. The central controller modifies the video parameters and notifies the corresponding application adapter with the revised application parameters so it can modify the application operation accordingly.

Applying HyARM to the Multimedia System Case Study HyARM is built atop of TAO [5], which is a widely used open-source implementation of Real-time CORBA [6]. HyARM is a middleware that enables adaptive resource management for DRE systems. HyARM can be applied to ensure predictable, efficient, and adaptive resource management of any DRE system where resource availability and/ requirements are subject to dynamic change.

Figure 3 shows the interaction of various parts of the DRE multimedia system developed with HyARM, TAO, and TAO's A/V Streaming Service. TAO's A/V Streaming service is an implementation of of the CORBA A/V Streaming Service specification [18]. TAO's A/V Streaming Service is a QoS-enabled video distribution service that can transfer video in real-time to one or more receivers. We use the A/V Streaming Service to transmit the video from the UAVs to the end receivers via the base station. Three

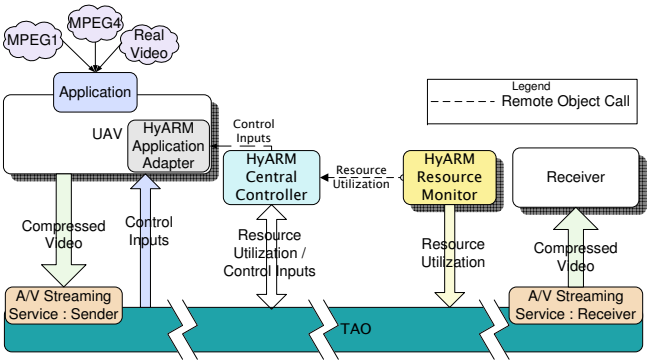

Fig. 3. Developing the DRE Multimedia System with HyARM

entities of HyARM, resource monitors, central controller, and application adapters are built as CORBA servants, so they can be distributed throughout a DRE system. Resource monitors are remote CORBA objects that update the central controller with the current resource utilization periodically via TAO. Application adapters are collocated with applications since they interact closely.

As shown in Figure 3, UAVs compress the data using various compression schemes, such as MPEG1, MPEG4, and Real Video and use TAO's A/V streaming service to transmit the video to end receivers. HyARM's resource monitors continuously observe the system resource utilization and notify central controller with the current resource utilization. The base station is not included in the figure since the it only retransmits the video received from UAVs to end receivers.

The interaction between the controller and the resource monitors uses the Observer pattern [19]. When the controller receives resource utilization updates from monitors it performs necessary modifications to application parameters and notifies application adapter(s) via a remote operation call using TAO. Application adapter(s), that are collocated with the application, modify the input parameters to the application, in our case video encoder, to modify the application resource utilization and QoS. Table 1 summarizes the number of lines of code of various entities in our middleware and DRE multimedia system case study.[3]

Entity	Total Source Lines of Code
Multimedia System	157,191
HyARM	2,872
CORBA A/V Streaming Service	19,504
The ACE ORB (TAO)	258,902

Table 1. Source Lines of Code for Middleware and DRE Multimedia System

4 Formal Model of the DRE Multimedia System

This section presents a formal model of our distributed multimedia system case study discussed in Section 2. We also present a formal model of the resource management challenges of our case study and describe how these challenges are resolved using HyARM by presenting a detailed analytical description of HyARM's adaptive behavior. This analytical model ensures that HyARM will restore the system utilization to the desired state even during fluctuation in resource demand.

End-to-end system model. The multimedia system comprises *n* applications $\{T_i \mid 1 \leq i \leq n\}$, executing on *m* resources $\{R_j \mid 1 \leq j \leq m\}$. The utilization of each resource is monitored periodically and the sampling period is denoted by T_s. The utilization at sampling period k is specified as $U_j(k)$.[4] We assume that a desired utilization set point $U_j^{(s)}$ of each resource R_j is specified. Let U^s represent the desired utilization set point of all system resources and $U(k)$ represent the utilization of all resources at sampling period k. The objective of HyARM is to increase the utilization $U(k)$ while satisfying the utilization bound described by the set point U^s even in the presence of dynamic workload changes, *(i.e)*

$$max \sum_{j=1}^{m} U_j(k) \mid subject\ to\ U_j(k) \leq U_j^s\ \{1 \leq j \leq m\} \tag{1}$$

During system startup, resources are allocated to *QoS-enabled* and *best-effort* application classes by selecting desired utilization set points. For the QoS-enabled applications, we select the desired utilization set point for all resources U_g^s and we have

[3] Lines of source code was measured using SLOCCount.

[4] We represent the utilization of all system resources by U that is a vector of size m. U takes values within $[0, 1]^m$.

$U_g^s < U^s$ (component wise). Due to changes in workload at runtime, HyARM controls the resource utilization of the *best-effort* class of applications to achieve the system objective outlined in equation 1. This objective is achieved by setting a time varying utilization set point for the *best-effort* class defined by

$$U_{be}^s(k) = max\{(U^s - U_g(k)), 0\} \tag{2}$$

The task of HyARM's controller is to select application parameters (such as frame rate, resolution, and compression scheme) dynamically to ensure that resource utilization of each application class is close to the utilization set point of that class. The utilization set point of the *QoS-enabled* class is a constant, while utilization set point of *best-effort* class is time varying.

Resolution is a continuous parameter and can take values between $[S^{min}, S^{max}]$, *(i.e)* $S^{min} \leq S(k) \leq S^{max}$. HyARM achieves modification to resolution by modifying the width of the picture, and the length of the picture is computed to maintain an aspect ration of 4:3. Frame-rate and compression scheme are discrete parameters and can take values from a fixed set, *(i.e)* $F(k) \in \{F^1, F^2, F^3, \ldots, F^p\}$, and $C(k) \in \{C^1, C^2, C^3, \ldots, C^q\}$. Specifically, the controller employs change in resolution $\Delta S(k)$ as a *continuous control variable*, and transitions between different frame rates and compression schemes as *discrete control variables*.

4.1 System Dynamics

At sampling period k, the contribution of the application T_i to the utilization of resource R_j by application T_i is denoted as $U_{j,i}(k)$. The resource monitors observe these values and update the central controller with the current values resource utilization, $U_{j,i}(k)$. The total resource utilization of resource R_j can therefore be written as

$$U_j(k) = \sum_{i=1}^{n} U_{j,i}(k) \tag{3}$$

We now establish a dynamic model that characterizes the relationship between the *control inputs* from the central controller to the application adapter, $\Delta S(k)$, $F(k)$, and $C(k)$ and the utilization $U_{j,i}(k)$. The dynamics of the system, $U_j(k)$, can be obtained from $U_{j,i}$ as shown by equation 3. The model is characterized by *hybrid* (continuous and discrete) dynamics. As discussed below, the continuous dynamics represent the effect of changes in resolution to the utilization for fixed frame rate and compression scheme, whereas the discrete dynamics describe the utilization effect for changing either the frame rate or the compression scheme.

Continuous dynamics. Let $\Delta S_i(k) = S_i(k) - S_i(k-1)$ denote the change in resolution of application T_i. The relationship between $\Delta S_i(k)$ and utilization of resource R_j by application T_i, $U_{j,i}(k)$, for a fixed compression scheme and frame-rate, is as follows:

$$U_{j,i}(k+1) = U_{j,i}(k) + g_{j,i,F_i(k),C_i(k)} * \Delta S_i(k) \tag{4}$$

where the *utilization gain* $g_{j,i,F_i(k),C_i(k)}$ represents the estimated change in resource utilization of application T_i running on resource R_j for a unit change in resolution. In

general, the utilization gain is not constant and not known *a priori*. We use an estimated value that is obtained by profiling the system and monitoring the change in resource utilization of various resources for unit change in resolution. The controller is based on this estimated value of the gain. Given that we employ feedback control loop, however, we expect to obtain good performance for a reasonable varying gain.

There is a limit on the change in resource utilization that can be obtained by changing the resolution of the picture, and is limited by the minimum and maximum resolution, S_i^{min}, S_i^{max}. In contrast, changes in frame rate and compression scheme alter the resource utilization and application QoS significantly. If desired changes in resource utilization cannot be achieved by modifying resolution, HyARM modifies frame rate and/or the compression scheme as described next in the discrete dynamics discussion.

Discrete dynamics. Discrete dynamics describe the change in utilization by an application by switching frame rates and/or the compression scheme. A combination of frame rate and compression scheme makes up a discrete operational state of an application. The *hybrid automata* of an application is shown in Figure 4.

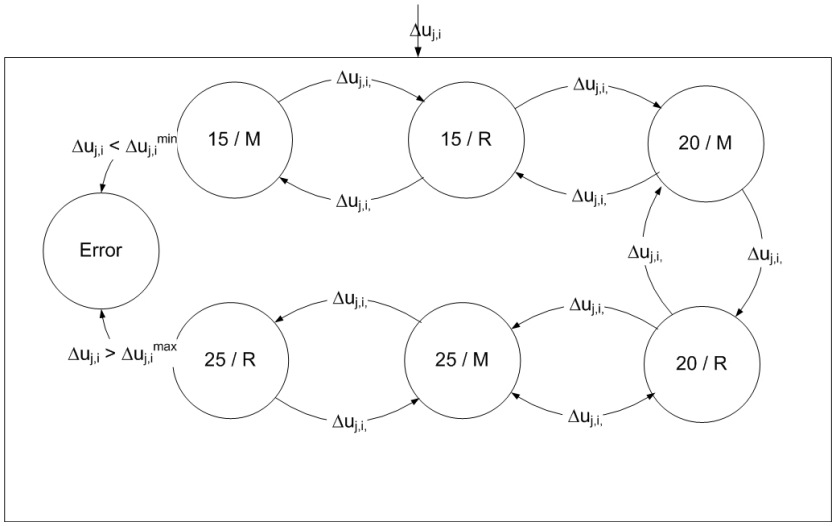

Fig. 4. Hybrid Automata Model of the Application

Each state in Figure 4 represents a compression scheme and a constant frame rate. The error state indicates that the current resource utilization by the application is at a maximum (or maximum), and therefore, cannot be further increased (or decreased). Input to the *hybrid automata* is $\Delta U_{j,i}^{'}$, the desired change in utilization of resource R_j by application T_i. A transition from one state to other indicates in change in frame-rate and/or compression scheme, and therefore, results in a change in resource utilization. Estimated change in resource utilization by the application T_i as a result of the transition is represented by $\Delta U_{j,i}$ on each transition.

For example, the state $\{20$ / M$\}$ indicates that the video is transmitted at 20 frames per second and compressed using MPEG-4 compression scheme. A transition from $\{20$ / M$\}$ to $\{15$ / R$\}$ indicates a modification in the video properties and reduces the frame-rate from 20 to 15 and modifies the compression scheme from MPEG-4 to Real Video. The estimated change in resource utilization is given by the parameter $\Delta U_{j,i}$ on the transition from $\{20$ /M$\}$ to $\{15$ / R$\}$. The relationship between the discrete transition and application resource utilization is modeled as follows:

$$U_{j,i}(k+1) = U_{j,i}(k) + \Delta U_{j,i,l,m} \tag{5}$$

where $\Delta U_{j,i,l,m}$ is the estimated change in resource utilization of resource R_j by application T_i as a result of the transition from state l to m. Value of $\Delta U_{j,i,l,m}$ are not constant and are obtained by profiling, similar to the utilization gain $g_{j,i,F_i(k),C_i(k)}$.

Equation 4, equation 5 and the *hybrid automata* shown in Figure 4 model the *hybrid* dynamics of an application T_i. At every time step k HyARM modifies the continuous parameter, $\Delta S(k)$, or discrete parameters, $F(k), C(k)$, of the application(s), if needed.

4.2 Modeling Adaptive Behavior in HyARM

The inputs to HyARM include (1) resource utilization set point for all m resources, U^s, (2) utilization set point for QoS-enabled class of applications, U^s_g, (3) applications with their minimum and maximum resolution, discrete set of frame-rate and compression scheme. Every sampling period, the controller of HyARM receives the net utilization $U(k)$ and utilization of each class of application $U_g(k),\ U_{be}(k)$ for all system resources. Upon receiving the resource utilization, the controller, if needed, computes the new set of application parameters (frame-rate, compression scheme, and resolution) and notifies the corresponding application adapter(s). In response to this, the application adapter(s) modifies input to the video encoder accordingly. The detailed layout of the interaction of HyARM's various entities and the DRE multimedia system is shown in Figure 3.

In a DRE system containing many resources, such as our DRE multimedia system, resource utilization of various resources may differ significantly. As a result, utilization of certain resource may be within desired bounds whereas utilization of certain resources may be below (or above) the desired bound. Every sampling period k, HyARM uses the following algorithm to determine whether the system is over-utilized or under-utilized:

if $\exists R_j \mid U_j(k) > U^s_j\{j = 1, 2, \ldots, m\}$ **then**
  return over-utilized
else
  if $\exists R_j | U_j(k) < U^s_j\{j = 1, 2, \ldots, m\}$ **then**
    return under-utilized
  else
    return efficient-utilized
  end if
end if

HyARM responds to changes in resource availability and/or demand by reallocating system resources to QoS-enabled and best-effort classes of applications as follows:

Over-utilization of system resources. To reduce the system resource utilization, HyARM first identifies the resource, R_j, that is over-utilized to the maximum, *(i.e)*, $j = arg(max\{U_j(k) - U_j^s\})$. To achieve a reduction in the utilization of resource R_j, HyARM then uses the following algorithm to compute the change in resource allocation to QoS-enabled and/or best-effort class of applications.

if $U_{j,g}(k) > U_{j,g}^s$ **then**
  $\Delta U_{j,g}(k) = U_{j,g}^s - U_{j,g}(k)$
else
  $U_{j,be}^s(k) = max\{(U_j^s - U_j(k)), 0\}$
  $\Delta U_{j,be}(k) = U_{j,be}^s(k) - U_{j,be}(k)$
end if

Under-utilization of system resources. To increase the system resource utilization, HyARM first identifies R_j, the resource that is least under-utilized, *(i.e)*, $j = arg(min\{U_j^s - U_j(k)\})$. To achieve an increase in the utilization of resource R_j, HyARM then uses the following algorithm to compute the change in resource allocation to QoS-enabled and/or best-effort class of applications.

if $U_{j,g}(k) < U_{j,g}^s$ **then**
  $\Delta U_{j,g}(k) = U_{j,g}^s - U_{j,g}(k)$
else
  $U_{j,be}^s(k) = max\{(U_j^s - U_j(k)), 0\}$
  $\Delta U_{j,be}(k) = U_{j,be}^s(k) - U_{j,be}(k)$
end if

HyARM's central controller employs the *hybrid dynamics* described above to achieve the above computed change in utilization ($\Delta U_{j,g}$ and/or $\Delta U_{j,be}$) to compute the new set of application parameters. Following this, the controller notifies the corresponding application adapters(s) with new set of application parameters. With the help of resource monitors, central controller, and the application adapter, HyARM thus maintains system resource utilization within the specified bounds.

5 Performance Results and Analysis

This section describes the testbed that provides the infrastructure for our DRE multimedia system, which was used to evaluate the performance of HyARM. We then describe the experiments and analyze the results we obtained to empirically evaluate how HyARM's adaptive resource model from Section 4 behaves during under- and over-utilization of system resources.

5.1 Overview of the Hardware and Software Testbed

Our experiments were performed on the University of Utah's Emulab network testbed [20]. The hardware configuration is shown in Figure 5 and consists of two nodes acting as UAVs, one acting as base station, and one as end receiver. Video from the two UAVs were transmitted to a base station via a LAN. Network properties of the LAN were chosen as follows: average packet loss ratio of 0.3 and bandwidth 1 Mbps (the network

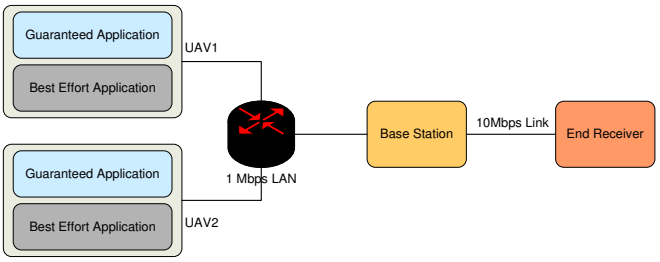

Fig. 5. Experimentation Testbed Configuration

bandwidth was chosen to be 1 Mbps since each UAV in the DRE multimedia system is allocated 250 Kbps). These parameters were chosen to emulate an unreliable wireless network with limited bandwidth between the UAVs and the base station.[5] From the base station, the video was retransmitted to the end receiver via a wireline link of 10 Mbps bandwidth.

The hardware configuration of all the nodes is 600 MHz Intel Pentium III processor, 256 MB physical memory, 4 Intel EtherExpress Pro 10/100 Mbps Ethernet ports, and 13 GB hard drive. A real-time version of Linux – TimeSys Linux/NET 3.1.214 based on RedHat Linux 9 – was used as the operating system for all nodes. The following software packages were also used for our experiments:

- **Ffmpeg 0.4.9-pre1**, which is an open-source library (`ffmpeg.sourceforge.net/download.php`) that compresses video into MPEG-2, MPEG-4, Real Video, and many more video formats. This package encodes the video at the UAV(s), before transmitting to the base station. We used this library to compress video in MPEG-4 and Real Video format for our experiments.
- **Iftop 0.16**, which is an open-source library (`www.ex-parrot.com/~pdw/iftop/`) we used to monitor network activity and bandwidth utilization. Iftop is built atop of pcap and provides current network activity in a user interface similar to the `top` CPU monitor.
- **ACE 5.4.3 + TAO 1.4.3**, which is an open-source (`www.dre.vanderbilt.edu/TAO`) implementation of the Real-time CORBA [6] specification upon which HyARM is built. TAO provides the CORBA Audio/Video (A/V) Streaming Service [18] that we use to transmit the video from the UAVs to end receivers via the base station.

5.2 DRE Multimedia System Experiment Configuration

Our experiment consisted of two (emulated) UAVs that simultaneously send video to the base station using the experimentation setup described in Section 5.1. At the base station, video was retransmitted to the end receivers (without any modifications), where it was stored to a file. Each UAV hosted two applications, one QoS-enabled application

[5] Since the wireless testbed provided by Emulab is still work-in-progress and unstable, we simulated a wireless LAN using a physical LAN with the above specified network properties.

(emergency response), and one best-effort application (surveillance). Within each UAV, *computational power* is shared between the applications, while the *network bandwidth* is shared among all applications.

To evaluate the QoS provided by HyARM, we monitored CPU utilization at the two UAVs, and network bandwidth utilization between the UAV and the base station. CPU resource utilization was not monitored at the base station and the end receiver since they performed no computationally-intensive operations. The resource utilization of the 10 Mpbs physical link between the base station and the end receiver does not affect QoS of applications and is not monitored by HyARM since the is nearly 10 times the 1 MB bandwidth of the LAN between the UAVs and the base station. The experiment also monitors properties of the video that affect determine the QoS of the applications, such as (1) latency, (2) jitter, (2) frame rate, (4) resolution.

The set point on resource utilization for each resource was specified at 0.69, which is the upper bound typically recommended by scheduling techniques, such as rate monotonic algorithm [1]. Since studies [16, 17] have shown that human eyes can perceive delays more than 200ms, we use this as the upper bound on jitter of the received video. QoS requirements for each class of application is specified during system initialization and is shown in Table 2.

Class	Resolution	Frame Rate	Latency (msec)	Jitter (msec)
QoS Enabled	1024 x 768	25	30	200
Best-effort	320 x 240	15	50	250

Table 2. Application QoS Requirements

5.3 Empirical Results and Analysis

This section presents the results obtained from running the experiment described in Section 5.2 on our DRE multimedia system testbed. We used system resource utilization as a metric to evaluate the adaptive resource management capabilities of HyARM under varying input work loads. We also used application QoS as a metric to evaluate HyARM's capabilities to support end-to-end QoS requirements of the various classes of applications in the DRE multimedia system. We analyze these results to explain the significant differences in system performance and application QoS.

Comparison of system performance is decomposed into comparison of resource utilization and application QoS. For system resource utilization, we compare (1) network bandwidth utilization of the local area network and (2) CPU utilization at the two UAV nodes. For application QoS, we compare mean values of video parameters, including (1) picture resolution, (2) frame rate, (3) latency, and (4) jitter.

Comparison of resource utilization. Over-utilization of system resources in DRE systems can yield an unstable system. In contrast, under-utilization of system resources increases system cost. Figure 6 presents the comparison of system resource utilization with and without HyARM. Figure 6 shows that HyARM maintains system utilization

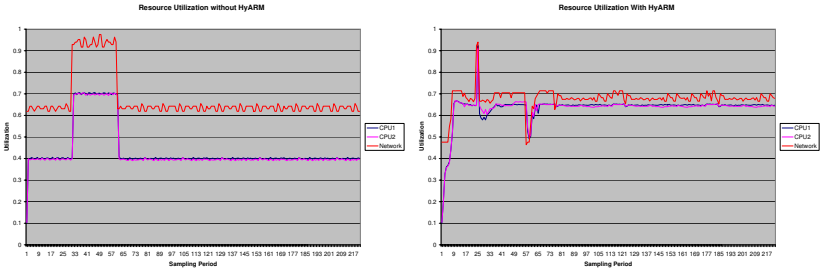

Fig. 6. Comparison of Resource Utilization

close to the desired utilization set point during fluctuation in input work load by transmitting video of higher (or lower) QoS for QoS-enabled (or best-effort) class of applications during over (or under) utilization of system resources. HyARM tries to achieve the objective function outlined by equation 1 throughout the lifetime of the system by lowering the utilization set point of best-effort class of applications, as per equation 2, and ensuring resource requirements of QoS enabled applications are met.

Figure 6 shows that without HyARM, network utilization was as high as 0.9 during increase in work load conditions, which is greater than the utilization set point of 0.7 by 0.2. As a result of over-utilization of resources, the QoS of the received video, such as average latency and jitter, was affected significantly and system resources were either under-utilized or over-utilized, both of which are undesirable. In contrast, with HyARM, system resource utilization is always close to the desired set point, even during fluctuations in application work load. During sudden fluctuation in application workload, system conditions may be temporarily undesirable, but are restored to the desired condition within several sampling periods. Temporary over-utilization of resources is permissible in our multimedia system since the quality of the video may be degraded for a short period of time, though application QoS will be degraded significantly if poor quality video is transmitted for a longer period of time.

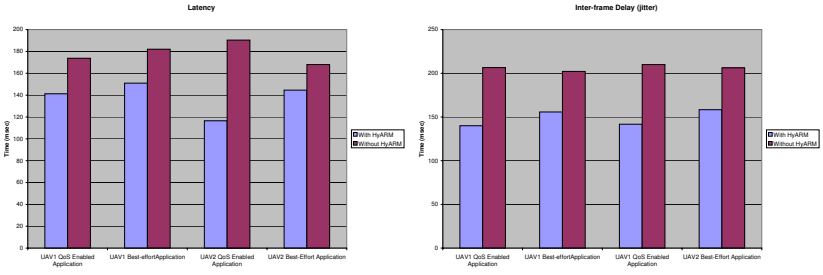

Fig. 7. Comparison of Image Latency and Jitter

Comparison of application QoS. Figure 7 and Table 3 compare latency, jitter, resolution, and frame-rate of the received video, respectively. Table 3 shows that HyARM increases the resolution and frame video of QoS-enabled applications, but decreases the resolution and frame rate of best effort applications. Resolution and frame rate of lower priority applications are reduced to *adapt* to fluctuations in application work load and to maintain the utilization of resources at the specified set point.

Source	Picture Size / Frame Rate	
	With HyARM	Without HyARM
UAV1 QoS Enabled Application	1122 X 1496 / 25	960 X 720 / 20
UAV1 Best-effort Application	288 X 384 / 15	640 X 480 / 20
UAV2 QoS Enabled Application	1126 X 1496 / 25	960 X 720 / 20
UAV2 Best-effort Application	288 X 384 / 15	640 X 480 / 20

Table 3. Comparison of Video Quality

Figure 7 shows how HyARM reduces the latency and jitter of the received video significantly. These figures show that the QoS of QoS-enabled applications is greatly improved by HyARM. Although application parameters, such as frame rate and resolutions, which affect the *soft* QoS requirements of best-effort applications may be compromised, the *hard* QoS requirements, such as latency and jitter, of all applications are met.

HyARM responds to fluctuation in resource availability and/or demand by constant monitoring of resource utilization. As shown in Figure 6, when resources utilization increases above the desired set point, HyARM lowers the utilization by reducing the QoS of best-effort applications. This adaptation ensures that enough resources are available for QoS-enabled applications to meet their QoS needs. Figure 7 shows that the values of latency and jitter of the received video of the system with HyARM are nearly half of the corresponding value of the system without HyARM. With HyARM, values of these parameters are well below the specified bounds, whereas without HyARM, these value are significantly above the specified bounds due to over-utilization of the network bandwidth, which leads to network congestion and results packet loss. HyARM avoids this by reducing video parameters such as resolution and /or frame-rate, and /or modifying the compression scheme used to compress the video.

Our conclusions from analyzing the experiments described above are that applying adaptive middleware via hybrid control theory to DRE system helps to (1) improve application QoS, (2) increase system resource utilization, and (3) provide better predictability (lower latency and inter-frame delay) to QoS-enabled applications. These improvements are achieved largely due to monitoring of system resources, efficient system workload management, and adaptive resource provisioning by means of HyARMS's network/CPU resource monitors, application adapter, and central controller, respectively.

6 Related Work

A number of control theoretic approaches have been applied to DRE systems recently. These techniques aid in overcoming limitations with traditional scheduling approaches that handle dynamic changes in resource availability ineffectively and yield in a rigidly scheduled systems that adapt poorly to change. A survey of these techniques is presented in [21].

One such approach is *feedback control scheduling* (FCS) [22–26]. FCS algorithms dynamically adjust resource allocation by means of software feedback control loops. FCS algorithms are modeled and designed using rigorous control-theoretic methodologies, and as a result, these algorithms provide robust and analytical performance assurances despite uncertainties. Although existing FCS algorithms have shown promise, existing algorithms often assume that the system has continuous control variable(s) that can continuously be adjusted. While this assumption holds for certain classes of systems, there are many classes of DRE systems, such as avionics and total-ship computing environments that only support a finite a priori set of discrete configurations. The control variables in such systems are therefore intrinsically discrete.

HyARM handles both continuous control variables, such as picture resolution, and discrete control variable, such as discrete set of frame-rates. HyARM can therefore be applied to system that support continuous and/or discrete set of control variables. The DRE multimedia system as described in Section 2 is an example DRE system that offers both continuous (picture resolution) and discrete set (frame-rate) of control variables. These variables are modified by HyARM to achieve efficient resource utilization and improved application QoS.

CAMRIT [3] applies control theoretic approaches to ensure transmission deadlines of images over an unpredictable network link and also presents analytic performance assurance that the transmission deadlines are met. CAMRIT monitors the TCP buffer length and is used as an indicator of current network bandwidth availability. to meet the transmission deadlines, CAMRIT modifies application properties, such as *quality factor* of the JPEG image compression scheme. Although this approach is similar to the adaptation mechanisms of HyARM, CAMRIT monitors and performs resource management only for one resource, the network, and handles only one QoS requirement, transmission deadline. HyARM provides *adaptive* resource management of multiple resources simultaneously, such as CPU utilization of multiple hosts and multiple network links. In addition, HyARM also supports differentiated classes of services to applications based on their relative priority that is not supported by CAMRIT.

Quality of Service for Objects (QuO) [27] is an open-source middleware framework that provides a bridge between QoS capabilities offered by the underlying network and QoS requirements of the application. QuO translates application QoS requirements into network and endsystem QoS parameters and relies on the underlying network and middleware infrastructure to handle fluctuations in resource availability and /or demand. Although the architecture of QuO is similar to that of HyARM, adaptation decisions in HyARM, such as modification of application parameters and resource (re)allocation to applications, are based on advanced hybrid theoretic model that captures the dynamics of the system. Therefore, HyARM handles fluctuation in resource availability /or

demands in a graceful manner and ensures that utilization of the system resources are below the desired bounds within a finite period of time.

7 Concluding Remarks

Many distributed real-time and embedded (DRE) systems demand end-to-end quality of service (QoS) enforcement from their underlying platforms to operate correctly. These systems increasingly run in open environments, where resource availability is subject to dynamic change. To meet end-to-end QoS in dynamic environments, DRE systems can benefit from an adaptive middleware that monitors system resources, performs efficient application workload management, and enable efficient resource provisioning to executing applications.

Resource management mechanisms based on control theoretic techniques are emerging as a promising solution to handle the challenges of applications with stringent end-to-end QoS executing in DRE systems. These mechanisms enable *adaptive* resource management capabilities in DRE systems and *adapt* gracefully to fluctuation in resource availability and application resource requirement at run-time.

This paper described an adaptive middleware called HyARM that provides effective resource management to DRE systems. HyARM employs hybrid control techniques to provide the adaptive middleware capabilities, such as resource monitoring and application adaptation, that are key to providing the dynamic resource management capabilities for open DRE systems. We employed HyARM to a representative DRE multimedia system that is implemented using Real-time CORBA and the CORBA A/V Streaming Service.

We evaluated the performance of HyARM in a system composed of three distributed resources and two classes of applications with two applications each. Our empirical results indicate that HyARM ensures efficient resource utilization by maintaining the resource utilization of system resources within the specified utilization bounds. HyARM also ensures QoS requirements of QoS-enabled applications are met at all times ensures efficient, predictable, and adaptive resource management for DRE systems.

The lessons learned by applying hybrid control theoretic approach to resource management to DRE systems thus far include:

- Hybrid control theoretic approaches yield in an *adaptive* resource management middleware that can handle fluctuations in resource availability and/or demand in a graceful manner. Analytical design of middleware yields in an implementation that is *correct by construction.*
- In a DRE system with heterogeneous resources, achieving efficient utilization of all system resources may not be possible, as some parts of the system may be over engineered that the rest of the system. In our multimedia the physical network link between the base station and the end receiver was over-provisioned compared to the wireless link between the UAVs and the base station. This resource was therefore under-utilized for the entire system lifetime.
- Developing applications that have various parameters that can be fine tuned in order to modify the application operation and utilization of system resources aid in

achieving higher QoS of applications. This also enables in maintaining the system resource utilization within the desired bounds.
- The current design of HyARM features a centralized controller. Although the results in Section 5 show that this approach is feasible for a DRE system with small number of resources and applications, a centralized approach will not be scalable for a larger DRE systems. We therefore plan to extend the current design of HyARM to feature a distributed and/or hierarchical controller.

DRE systems are increasingly being used for mission-critical applications that operate in hostile environments and are subjected to constant *QoS attacks* that aim at significantly degrade the performance of the system, which results in significant degrades of QoS of these mission critical applications. These attacks often results in loss of system resources. In future work, we will extend HyARM to detect such attacks early and prevent them from reducing the QoS of mission critical applications.

References

1. J. Lehoczky, L. Sha, and Y. Ding, "The Rate Monotonic Scheduling Algorithm: Exact Characterization and Average Case Behavior," in *Proceedings of the 10th IEEE Real-Time Systems Symposium*, pp. 166–171, IEEE Computer Society Press, 1989.
2. J. Loyall, J. Gossett, C. Gill, R. Schantz, J. Zinky, P. Pal, R. Shapiro, C. Rodrigues, M. Atighetchi, and D. Karr, "Comparing and Contrasting Adaptive Middleware Support in Wide-Area and Embedded Distributed Object Applications," in *Proceedings of the 21st International Conference on Distributed Computing Systems (ICDCS-21)*, pp. 625–634, IEEE, Apr. 2001.
3. X. Wang, H.-M. Huang, V. Subramonian, C. Lu, and C. Gill, "CAMRIT: Control-based Adaptive Middleware for Real-time Image Transmission," in *Proceedings of the 10th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, (Toronto, Canada), IEEE, May 2004.
4. X. Koutsoukos, R. Tekumalla, B. Natarajan, and C. Lu, "Hybrid Supervisory Control of Real-Time Systems," in *11th IEEE Real-Time and Embedded Technology and Applications Symposium*, (San Francisco, California), IEEE, Mar. 2005.
5. D. C. Schmidt, D. L. Levine, and S. Mungee, "The Design and Performance of Real-Time Object Request Brokers," *Computer Communications*, vol. 21, pp. 294–324, Apr. 1998.
6. Object Management Group, *Real-time CORBA Specification*, OMG Document formal/02-08-02 ed., Aug. 2002.
7. Thomas Sikora, "Trends and Perspectives in Image and Video Coding," in *Proceedings of the IEEE*, Jan. 2005.
8. I. Richardson, *Video Codec Design: Developing Image and Video Compression Systems*. New York, New York: John Wiley & Sons, 2002.
9. A. N. Netravali and B. G. Haskell, *Digital Pictures: Representations, Compression, and Standards (Applications of Communications Theory)*. New York, New York: Plenum Publishing Corporation, 1995.
10. W. Effelsberg and R. Steinmetz, *Video compression techniques*. dpunkt—Verlag fur digitale Technologie GmbH, 1998.
11. D. L. Gall, "Mpeg: a video compression standard for multimedia applications," *Commun. ACM*, vol. 34, no. 4, 1991.
12. B. G. Haskell, A. Puri, and A. N. Netravali, *Digital Video : An introduction to MPEG-2 (Digital Multimedia Standards Series)*. Springer, 1995.

13. F. P. Touradj Ebrahimi, *The MPEG-4 Book*. Prentice Hall, 2002.
14. D. L. Stone and K. Jeffay, "An empirical study of delay jitter management policies," *Multimedia Syst.*, vol. 2, no. 6, pp. 267–279, 1995.
15. D. Ferrari, "Design and applications of a delay jitter control scheme for packet-switching internetworks," in *Proceedings of the Second International Workshop on Network and Operating System Support for Digital Audio and Video*, pp. 72–83, Springer-Verlag, 1992.
16. G. Ghinea and J. P. Thomas, "Qos impact on user perception and understanding of multimedia video clips," in *MULTIMEDIA '98: Proceedings of the sixth ACM international conference on Multimedia*, (Bristol, United Kingdom), pp. 49–54, ACM Press, 1998.
17. M. Claypool and J. Tanner, "The effects of jitter on the peceptual quality of video," in *MULTIMEDIA '99: Proceedings of the seventh ACM international conference on Multimedia (Part 2)*, (Orlando, Florida, United States), pp. 115–118, ACM Press, 1999.
18. S. Mungee, N. Surendran, Y. Krishnamurthy, and D. C. Schmidt, "The Design and Performance of a CORBA Audio/Video Streaming Service," in *Design and Management of Multimedia Information Systems: Opportunities and Challenges* (M. Syed, ed.), Hershey, PA: Idea Group Publishing, 2000.
19. E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. Reading, MA: Addison-Wesley, 1995.
20. Robert Ricci and Chris Alfred and Jay Lepreau, "A Solver for the Network Testbed Mapping Problem," *SIGCOMM Computer Communications Review*, vol. 33, pp. 30–44, Apr. 2003.
21. T. F. Abdelzaher, J. Stankovic, C. Lu, R. Zhang, and Y. Lu, "Feddback Performance Control in Software Services," *IEEE: Control Systems*, vol. 23, June 2003.
22. S. Abdelwahed, N. Kandasamy, and S. Neema, "Online Control for Self-Management in Computing Systems," in *Proceedings of the 10th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, (Toronto, Canada), IEEE, May 2004.
23. L. Abeni, L. Palopoli, G. Lipari, and J. Walpole, "Analysis of a reservation-based feedback scheduler," in *IEEE Real-Time Systems Symposium*, Dec. 2002.
24. A. Cervin, J. Eker, B. Bernhardsson, and K.-E. Årzn, "Feedback‧feedforward scheduling of control tasks," *Real-Time Syst.*, vol. 23, no. 1-2, pp. 25–53, 2002.
25. C. Lu, J. A. Stankovic, G. Tao, and S. H. Son, "Feedback Control Real-Time Scheduling: Framework, Modeling, and Algorithms," *Real-Time Systems Journal*, vol. 23, pp. 85–126, July 2002.
26. D. C. Steere, A. Goel, J. Gruenberg, D. McNamee, C. Pu, and J. Walpole, "A feedback-driven proportion allocator for real-rate scheduling," in *Operating Systems Design and Implementation*, pp. 145–158, 1999.
27. J. A. Zinky, D. E. Bakken, and R. Schantz, "Architectural Support for Quality of Service for CORBA Objects," *Theory and Practice of Object Systems*, vol. 3, no. 1, pp. 1–20, 1997.