Evaluating Adaptive Resource Management for Distributed Real-Time Embedded Systems

Nishanth Shankaran, * Xenofon Koutsoukos, Douglas C. Schmidt, and Aniruddha Gokhale
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 conditions. This paper presents two contributions to research in adaptive resource management for DRE systems. First, we describe the structure and functionality of the Hybrid Adaptive Resource-management Middleware (HyARM), which provides adaptive resource management using hybrid control techniques for adapting to workload fluctuations and resource availability. Second, we evaluate the adaptive behavior of HyARM via experiments on a DRE multimedia system that distributes video in real-time. Our results indicate that HyARM yields predictable, stable, and high system performance, even in the face of fluctuating workload and resource availability.

Keywords

Distributed Systems, Real-time and Embedded Systems, Quality of Service, Hybrid Systems

1. INTRODUCTION

Achieving end-to-end real-time quality of service (QoS) is particularly important for open *distributed real-time and embedded* (DRE) systems that face resource constraints, such as limited computing power 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* [10, 15], which is software whose functional and QoS-related properties can be modified either **statically** or **dynamically**. Static modifications are carried out to reduce footprint, leverage capabilities that exist in specific platforms, enable functional subsetting, and/or minimize hardware/software infrastructure dependencies. Objectives of dynamic modifications include optimizing 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.

HyARM is based on hybrid control theoretic techniques [8], which provide a theoretical framework for designing control of complex system with both continuous and discrete dynamics. In our case study, which involves a distributed real-time video distribution system, the task of adaptive resource management is to control the utilization of the different resources, whose utilizations are described by continuous variables. We achieve this by adapting the resolution of the transmitted video, 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) [13], which is an implementation of the Real-time CORBA specification [12]. 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 evaluates the adaptive behavior of HyARM via experiments on our multimedia system case study; Section 5 compares our research on HyARM with related work; and Section 6 presents concluding remarks.

2. CASE STUDY: DRE MULTIMEDIA SYSTEM

This section describes the architecture and QoS requirements of our DRE multimedia system.

*Contact author:nshankar@dre.vanderbilt.edu

RM '05, November 28- December 2, 2005 Grenoble, France

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

2.1 Multimedia System Architecture

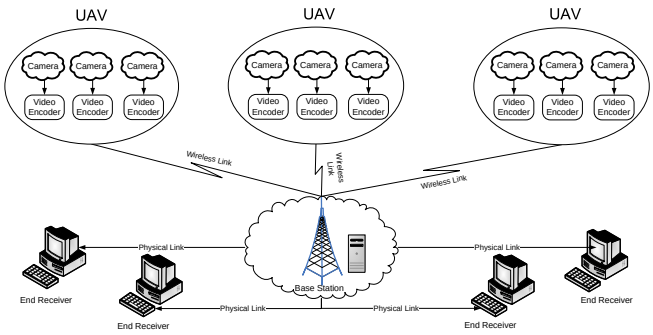

Figure 1: DRE Multimedia System Architecture

The architecture for our DRE multimedia system is shown in Figure 1 and consists of the following entities: (1)**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 the video to the next stage in the pipeline. (2)**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. (3) **Sinks (command and control center)**, where the received video is again processed to remove noise, then decoded and finally rendered to end user via graphical displays.

Significant improvements in video encoding/decoding and (de)compression techniques have been made as a result of recent advances in video encoding and compression techniques [14]. Common video compression schemes are MPEG-1, MPEG-2, Real Video, and MPEG-4. Each compression scheme is characterized by its resource requirement, *e.g.*, the computational power to (de)compress the video signal and the 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 requirements of the video.

Our multimedia system case study has the following end-to-end real-time QoS requirements: (1) latency, (2) inter-frame delay (also know as jitter), (3) frame rate, and (4) picture resolution. These QoS requirements 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 and picture resolution as softer QoS requirements and modify these video properties adaptively to handle dynamic changes in resource availability effectively.

2.2 DRE Multimedia System Rresources

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. The computing power required by the video capture and encoding tasks 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 the wireless connectivity between the UAVs and the base station, which in-turn depend on dynamic factors such as the speed of the UAVs and the relative distance between UAVs and base stations. The bandwidth of the link between the base station and the end receiver is limited, but more stable than the bandwidth of the wireless network. Resource requirements and availability of resources are subjected to dynamic changes.

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 class of applications have higher priority over best-effort class of application. In our study, emergency response applications belong to QoS-enabled and surveillance applications belong to best-effort class. For example, since a stream from an emergency response application is of higher importance than a video stream from a surveillance application, it receives more resources end-to-end.

Since resource availability significantly affects QoS, we use *current resource utilization* as the primary indicator of system performance. We refer to the current level of system resource utilization 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, residual system resources (*i.e.*, network bandwidth and computational power) are available in large amounts 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 and higher frame rate) 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 resources - network bandwidth and/or computing power at base station, end receiver or at UAV - or may be due to an increase in resource demands by applications. 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 and/or demand.

3. OVERVIEW OF HYARM

This section describes the architecture of the *Hybrid Adap-*

[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 two disjoint points.

tive Resource-management Middleware (HyARM). HyARM ensures efficient and predictable system performance by providing adaptive resource management, including monitoring of system resources and enforcing bounds on application resource utilization.

3.1 HyARM Structure and Functionality

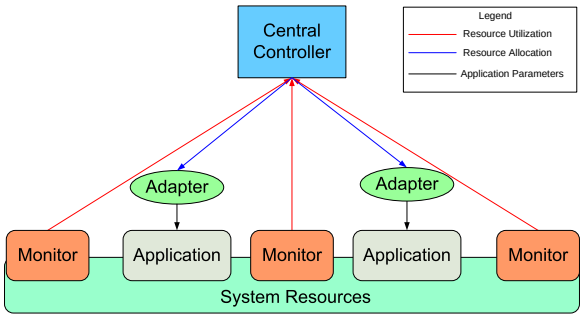

Figure 2: HyARM Architecture

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) wireless network link between UAVs and the base station and (2) wired network 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 [8], 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 ensures that the operation of the application is in accordance with the recommended parameters. In the current mplementation of HyARM, the application adapter modifies the input parameters to the application that affect application QoS and resource utilization - compression scheme, frame rate, and picture resolution. In our future implementations, we plan to use resource reservation mechanisms such as Differentiated Service [7, 3] and Class-based Kernel Resource Management [4] to provision/reserve network and CPU resources. 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.

3.2 Applying HyARM to the Multimedia System Case Study

HyARM is built atop TAO [13], which is a widely used open-source implementation of Real-time CORBA [12]. 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 the CORBA A/V Streaming Service specification. 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 entities of

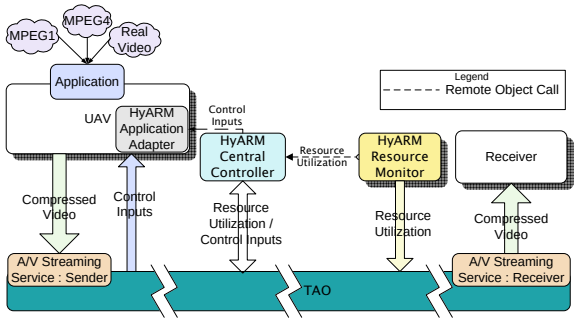

Figure 3: Developing the DRE Multimedia System with HyARM

HyARM, namely the 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 periodically with the current resource utilization. Application adapters are collocated with applications since the two interact closely.

As shown in Figure 3, UAVs compress the data using various compression schemes, such as MPEG1, MPEG4, and Real Video, and uses 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 the central controller with the current utilization. [3]

The interaction between the controller and the resource monitors uses the Observer pattern [5]. When the controller receives resource utilization updates from monitors, it computes the necessary modifications to application(s) parameters and notifies application adapter(s) via a remote operation call. 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.

4. PERFORMANCE RESULTS AND ANALYSIS

This section first describes the testbed that provides the infrastructure for our DRE multimedia system, which was

[3]The base station is not included in the figure since it only retransmits the video received from UAVs to end receivers.

used to evaluate the performance of HyARM. We then describe our experiments and analyze the results obtained to empirically evaluate how HyARM behaves during under- and over-utilization of system resources.

4.1 Overview of the Hardware and Software Testbed

Our experiments were performed on the Emulab testbed at University of Utah. The hardware configuration 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 configured with the following properties: average packet loss ratio of 0.3 and bandwidth 1 Mbps. The network 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. From the base station, the video was retransmitted to the end receiver via a reliable wireline link of 10 Mbps bandwidth with no packet loss.

The hardware configuration of all the nodes was chosen as follows: 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: (1) **Ffmpeg 0.4.9-pre1**, which is an open-source library (`http://www.ffmpeg.sourceforge.net/download.php`) that compresses video into MPEG-2, MPEG-4, Real Video, and many other video formats. (2) **Iftop 0.16**, which is an open-source library (`http://www.ex-parrot.com/~pdw/iftop/`) we used for monitoring network activity and bandwidth utilization. (3) **ACE 5.4.3 + TAO 1.4.3**, which is an open-source (`http://www.dre.vanderbilt.edu/TAO`) implementation of the Real-time CORBA [12] specification upon which HyARM is built. TAO provides the CORBA Audio/Video (A/V) Streaming Service that we use to transmit the video from the UAVs to end receivers via the base station.

4.2 Experiment Configuration

Our experiment consisted of two (emulated) UAVs that simultaneously send video to the base station using the experimentation setup described in Section 4.1. 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 (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 it 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 the QoS of the applications, such as latency, jitter, frame rate, and 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 [9]. Since studies [6] 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 1.

4.3 Empirical Results and Analysis

This section presents the results obtained from running the experiment described in Section 4.2 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 4 and Figure 5 compare the system resource utilization with and without HyARM. Figure 4 shows that HyARM maintains system 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.

Figure 5 shows that without HyARM, network utilization was as high as 0.9 during increase in workload conditions, which is greater than the utilization set point of 0.7 by 0.2. As a result of over-utilization of resources, QoS of the received video, such as average latency and jitter, was affected significantly. Without HyARM, 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 workload. 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.

Comparison of application QoS. Figures 6, Figure 7, and Table 2 compare latency, jitter, resolution, and frame-rate of the received video, respectively. Table 2 shows that HyARM increases the resolution and frame video of QoS-enabled applications, but decreases the resolution and frame rate of best effort applications. During over utilization of

Class	Resolution	Frame Rate	Latency (msec)	Jitter (msec)
QoS Enabled	1024 x 768	25	200	200
Best-effort	320 x 240	15	300	250

Table 1: Application QoS Requirements

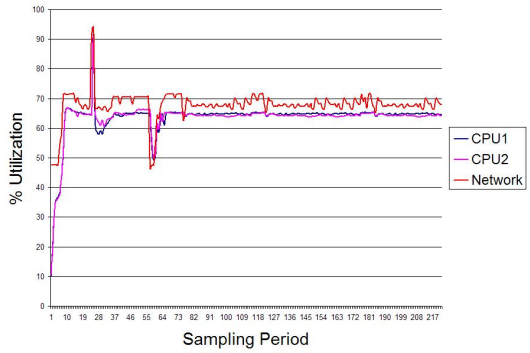

Figure 4: Resource utilization with HyARM

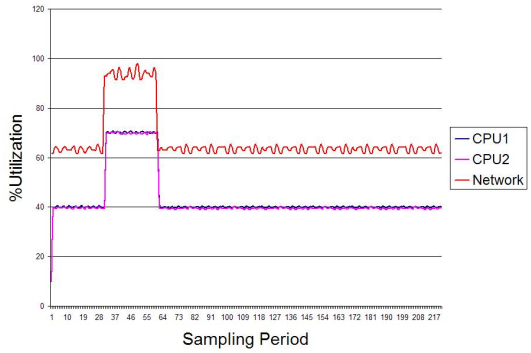

Figure 5: Resource utilization without HyARM

system resources, resolution and frame rate of lower priority applications are reduced to *adapt* to fluctuations in application workload and to maintain the utilization of resources at the specified set point.

It can be seen from Figure 6 and Figure 7 that 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 4, 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. Figures 6 and 7 show 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 in packet loss. HyARM avoids this by reducing video parameters such as resolution, frame-rate, and/or modifying the compression scheme used to compress the video.

Our conclusions from analyzing the results described above are that applying adaptive middleware via hybrid control 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 resource utilization, efficient system workload management, and adaptive resource provisioning by means of HyARM's network/CPU resource monitors, application adapter, and central controller, respectively.

5. 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 poorly and result in a rigidly scheduled system that adapts poorly to change. A survey of these techniques is presented in [1].

One such approach is *feedback control scheduling* (FCS) [2, 11]. FCS algorithms dynamically adjust resource allocation by means of software feedback control loops. FCS algorithms are modeled and designed using rigorous control-theoretic methodologies. These algorithms provide robust and analytical performance assurances despite uncertainties in resource availability and/or demand. Although existing FCS algorithms have shown promise, these 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.

6. 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

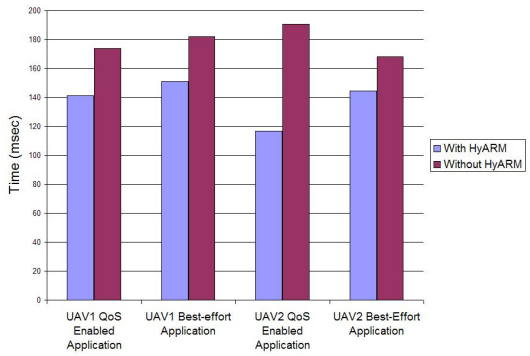

Figure 6: Comparison of Video Latency

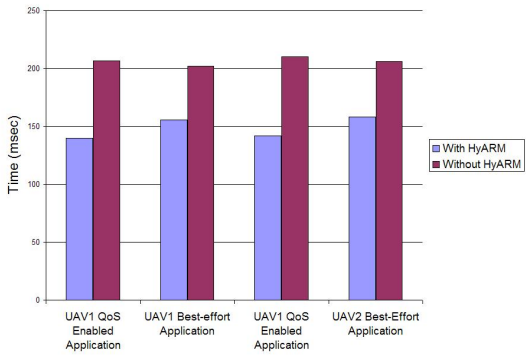

Figure 7: Comparison of Video Jitter

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 2: Comparison of Video Quality

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 enables efficient resource provisioning for executing applications.

This paper described HyARM, an adaptive middleware, 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 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 (1) efficient resource utilization by maintaining the resource utilization of system resources within the specified utilization bounds, (2) QoS requirements of QoS-enabled applications are met at all times. Overall, HyARM ensures efficient, predictable, and adaptive resource management for DRE systems.

7. REFERENCES

[1] T. F. Abdelzaher, J. Stankovic, C. Lu, R. Zhang, and Y. Lu. Feddback Performance Control in Software Services. *IEEE: Control Systems*, 23(3), June 2003.

[2] L. Abeni, L. Palopoli, G. Lipari, and J. Walpole. Analysis of a reservation-based feedback scheduler. In *IEEE Real-Time Systems Symposium*, Dec. 2002.

[3] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An architecture for differentiated services. *Network Information Center RFC 2475*, Dec. 1998.

[4] H. Franke, S. Nagar, C. Seetharaman, and V. Kashyap. Enabling Autonomic Workload Management in Linux. In *Proceedings of the International Conference on Autonomic Computing (ICAC)*, New York, New York, May 2004. IEEE.

[5] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, Reading, MA, 1995.

[6] 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*, pages 49–54, Bristol, United Kingdom, 1998. ACM Press.

[7] Internet Engineering Task Force. Differentiated Services Working Group (diffserv) Charter. www.ietf.org/html.charters/diffserv-charter.html, 2000.

[8] 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, Mar. 2005.

[9] 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 (RTSS 1989)*, pages 166–171. IEEE Computer Society Press, 1989.

[10] 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)*, pages 625–634. IEEE, Apr. 2001.

[11] C. Lu, J. A. Stankovic, G. Tao, and S. H. Son. Feedback Control Real-Time Scheduling: Framework, Modeling, and Algorithms. *Real-Time Systems Journal*, 23(1/2):85–126, July 2002.

[12] Object Management Group. *Real-time CORBA Specification*, OMG Document formal/02-08-02 edition, Aug. 2002.

[13] D. C. Schmidt, D. L. Levine, and S. Mungee. The Design and Performance of Real-Time Object Request Brokers. *Computer Communications*, 21(4):294–324, Apr. 1998.

[14] Thomas Sikora. Trends and Perspectives in Image and Video Coding. In *Proceedings of the IEEE*, Jan. 2005.

[15] X. Wang, H.-M. Huang, V. Subramonian, C. Lu, and C. Gill. CAMRIT: Control-based Adaptive Middleware for Real-time Image Transmission. In *Proc. of the 10th IEEE Real-Time and Embedded Tech. and Applications Symp. (RTAS)*, Toronto, Canada, May 2004.