Hierarchical Control of Multiple Resources in Distributed Real-time and Embedded Systems[*]

Nishanth Shankaran, Xenofon D. Koutsoukos,
Douglas C. Schmidt, and Yuan Xue
Department of EECS
Vanderbilt University

Chenyang Lu
Department of Computer Science and Engineering
Washington University, St. Louis

Abstract

There is an increasing demand to introduce adaptive capabilities in distributed real-time and embedded (DRE) systems that execute in open environments where system operational conditions, input workload, and resource availability cannot be characterized accurately a priori. To meet these needs, this paper presents the Hierarchical Distributed Resource-management Architecture (HiDRA), which provides adaptive resource management using control-theoretic techniques that adapt to workload fluctuations and resource availability. In contrast to adaptive control techniques that manage only one type of system resource, HiDRA features a hierarchical control scheme that manages both bandwidth and processor utilization simultaneously. This paper presents three contributions to research in adaptive resource management for DRE systems. First, we describe the structure and functionality of HiDRA. Second, we present an analytical model of HiDRA that formalizes its control theoretic behavior and present analytical performance guarantees. Third, we evaluate the performance of HiDRA via experiments on a representative DRE system that performs distributed target tracking in real-time. Our analytical and empirical results indicate that HiDRA yields predictable, stable, and high system performance, even in the face of changing workload.

1 Introduction

Distributed real-time and embedded (DRE) systems form the core of many mission-critical domains, including autonomous air surveillance, total ship computing environments, and supervisory control and data acquisition. Such DRE systems execute in *open* environments where system operational conditions, input workload, and resource availability cannot be characterized accurately *a priori*. Achieving end-to-end quality of service (QoS) is an important and challenging issue for these types of systems due to their unique characteristics, including (1) constraints in multiple resources (*e.g.*, limited computing power and network bandwidth) and (2) highly fluctuating resource availability and input workload.

Conventional resource management approaches, such as rate monotonic scheduling [9], are designed to manage system resources and providing QoS in *closed* environments where operating conditions, input workloads, and resource availability are known in advance. Since these approaches are insufficient for open DRE systems, there is an increasing need to introduce resource management mechanisms that can *adapt* to dynamic changes in resource availability and requirements. A promising solution is *feedback control scheduling* (FCS) [5, 1], which employs software feedback loops that dynamically control resource allocation in response to changes in input workload and resource availability. These techniques enable adaptive resource management capabilities in DRE systems that can compensate for fluctuations in resource availability and changes in application resource requirements at run-time. When FCS techniques are designed and modeled using rigorous control-theoretic techniques and implemented using QoS-enabled software platforms, they can provide robust and analytically sound QoS assurance.

Although existing FCS algorithms have been shown to be effective in managing a single type of resource, they have not managed multiple types of resources. It is still an open issue, therefore, to extend individual FCS algorithms to work together to manage multiple types of resources in a *coordinated* way, such as managing computational power and network bandwidth simultaneously. To address this issue, we have developed a control-based multi-resource management framework called *Hierarchical Distributed Resource management Architecture* (HiDRA). HiDRA employs a control-theoretic approach featuring two types of feedback controllers that coordinate the utilization of computational power and network bandwidth to prevent over-utilization of system resources. This capability is important because processor overload can cause system failure and network saturation can cause congestion and severe packet loss. Subject to the constraints of the desired utilization, HiDRA improves system QoS by modifying appropriate application parameters.

[*]This work is supported in part by DARPA, NSF CAREER award (CNS-0448554), Lockheed Martin ATL, BBN Technologies, and Raytheon.

This paper provides contributions to theoretical and experimental research. Its theoretical contribution is its use of control theory to formally prove the stability of HiDRA and derive its equilibrium resource utilization. Its experimental contribution is to evaluate how HiDRA works for a distributed target tracking application built atop *The ACE ORB* (TAO) [16], which is an implementation of Real-time CORBA [15]. Our experimental results validate our theoretical claims and show that HiDRA yields predictable and high-performance resource management and coordination for multiple types of resources.

The remainder of the paper is organized as follows: Section 2 describes the architecture and QoS requirements of our DRE system case study; Section 3 explains the structure and functionality of HiDRA; Section 4 formulates the problem described in Section 2 and presents an analysis of HiDRA; Section 5 empirically evaluates the adaptive behavior of HiDRA for our DRE system case study; Section 6 compares our research on HiDRA with related work; and Section 7 presents concluding remarks.

2 Case Study: Target Tracking DRE System

This section describes a distributed target tracking system that we use as a representative case study to investigate adaptive management of multiple system resources in open DRE systems. The tracking system provides emergency response and surveillance capabilities to help communities and relief agencies recover from major disasters, such as floods, hurricanes, or earthquakes. Figure 1 shows how multiple unmanned air vehicles (UAVs) fly over a predesignated area (known as an "area of interest") in this distributed target tracking system.

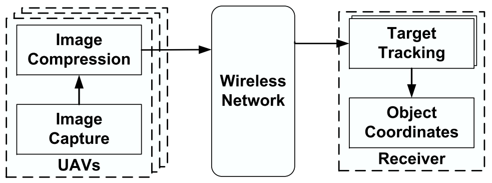


Figure 1: Target Tracking DRE System Architecture

Each UAV serves as a data source, captures live images, compresses them, and transmits them to a receiver over a wireless network. The receiver serves as a data sink, receives the images sent from the UAVs, and performs object detection. If the presence of an object of interest is detected in the received images, the tracking system determines the coordinates of the objects automatically and keeps tracking it. The coordinates of the object is reported to responders who use this information to determine the appropriate course of action, *e.g.* initiate a rescue, airlift supplies, etc. Humans, animals, cars, boats, and aircraft are typical objects of interest in our system.

The QoS of our resource-constrained DRE system is measured as follows:

- *Target-tracking precision*, which is the distance between the computed center of mass of an object and the actual center of mass of the object and
- *End-to-end execution*, which is the time between image capture by the UAV and computation of the coordinates of an object of interest. End-to-end execution includes image transmission latency and execution time of the object detection and tracking system.

There are two primary types of resources that may constrain the QoS of our DRE system: (1) *processors* that provide computational power available at the UAVs and the receiver and (2) the *wireless network bandwidth* that provides communication bandwidth between UAVs and the receiver. To determine the coordinates accurately, the images captured by the UAVs must be transmitted at a higher quality and a faster rate when an object is present, which in turn requires higher network bandwidth consumption and higher computing power. The utilization of the system resources (*i.e.*, wireless network bandwidth and computing power at the receiver) are therefore subject to sudden changes caused by the presence of varying numbers of objects of interest. Moreover, the wireless network bandwidth available to transmit images from the UAVs to the receiver depends on the wireless connectivity between the UAVs and the receiver, which in-turn depends on dynamic factors, such as the speed of the UAVs and the relative distance between UAVs and the receiver.

The observations above motivate the need for adaptive management of multiple resources. To meet this need, the captured images in our system are compressed using JPEG, which supports flexible image quality. Likewise, we choose to use image streams rather than video because video compression algorithms are computationally expensive, and the computation power of the on-board processor on the UAVs is limited. Moreover, emergency response and surveillance applications and operators do not necessarily need video at 30 frames per sec.

In JPEG compression, a parameter called the *quality factor* is provided as a user-specified integer in the range 1 to 100. A lower quality factor results in smaller data size of the compressed image. The quality factor of the image compression algorithm can therefore be used as a *control knob* to manage the bandwidth utilization of an UAV. To manage the computational power of the receiver, end-to-end execution rate of applications are used as the control knob.

3 The Hierarchical Distributed Resource-management Architecture – HiDRA

This section analyzes the adaptive management of multiple resources in open DRE systems using a control-based approach and presents the *Hierarchical Distributed Resource-management Architecture* (HiDRA), which employs a control-theoretic approach to manage processors and network bandwidth simultaneously. Our control framework is shown in Figure 2 and consists of three entities: *monitors*, *controller*, and *effectors*. A monitor is associated

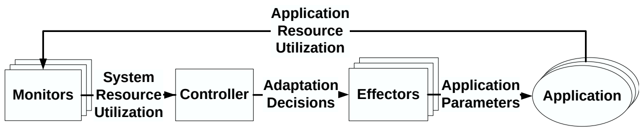


Figure 2: Control Framework

with a specific system resource and periodically updates the controller with the current resource utilization. The controller implements a particular control algorithm and computes the adaptations decisions for each (or a set of) application(s) to achieve the desired system resource utilization. Each effector is associated with an application and modifies the application parameters to achieve the controller recommended application adaptation.

We proceed to instantiate the HiDRA control framework to the domain of target tracking described in Section 2. Each application in our DRE system is composed of two subtasks: *image compression* and *target tracking*. To ensure end-to-end QoS, therefore, resource utilization of both subtasks must be controlled.

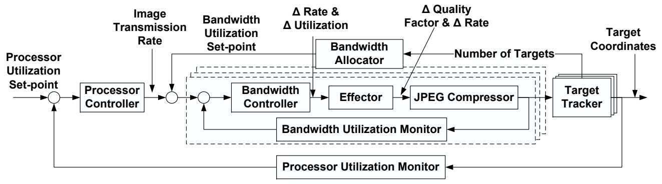


Figure 3: HiDRA Illustration.

As shown in Figure 3, HiDRA consists of two types of feedback control loops: (1) a processor control loop located at the receiver that manages the processor utilization and (2) a bandwidth control loop located at each UAV that manages the bandwidth utilization. These loops effectively control the utilization of two critical system resources and coordinate the execution of two subtasks. We structure these control loops in an *hierarchical* fashion so that the processor control loop at the receiver is viewed as the *outer* control loop and the bandwidth control loop at each UAV is viewed as the *inner* control loop.

The processor utilization monitor and processor controller serve as the resource monitor and controller of the processor control loop. The utilization set-point of the receiver processor is the input to the processor controller, and is specified during system initialization. The controlled variable for this loop is the processor utilization of the receiver, and the control input from the processor controller to the system is the image transmission rate, which is fed to the bandwidth controller of each UAV. For the processor control loop, therefore, the bandwidth controller serves as the effector.

The bandwidth utilization monitor and the bandwidth controller serve as the monitor and controller of the bandwidth control loop. The image transmission rate and bandwidth utilization set-point are the inputs to the bandwidth controller. Based on these inputs, the bandwidth controller computes an appropriate value of the JPEG quality factor to transmit the image of the highest quality, subjected to the specified bandwidth limitation. The controlled variable is the network bandwidth utilization of each UAV and the control input from the bandwidth controller to the system is the quality factor of the JPEG compression algorithm. This input is fed to the implementation of the JPEG compression algorithm, which serves as the effector for this control loop.

The bandwidth allocator is responsible for dynamically computing the bandwidth allocation to each UAV based on the presence/absence of objects of interest in the images received from the corresponding UAV. The bandwidth controller of each UAV views this allocation as bandwidth utilization set-point. The bandwidth allocator ensures that the bandwidth requirement of UAVs capturing images of one or more objects of interest is met.

4 Control Design and Analysis

This section first formalizes the resource management problem of our target tracking DRE system. We then map HiDRA to our DRE system to show how it addresses key resource management challenges. Finally, we present the stability analysis and show that HiDRA ensures the stability of our DRE system. The formalism presented below forms the foundations for the design and implementation of HiDRA and also provides analytical guarantees about system performance even under fluctuating workload.

4.1 Problem Formulation

The following formal notations are used throughout the remaining of the paper. The target tracking system consists of n UAVs, and therefore, n end-to-end tasks $\{T_i|1 \leq i \leq n\}$, each with two subtasks, *i.e.*, image compression subtask executed at the UAV$_i$ and target-tracking subtask executed at the receiver. The sampling period of the processor controller (outer feedback loop) and the bandwidth controller (inner feedback loop) are represented by T_s^{out} and T_s^{in}, respectively. Each end-to-end task T_i is invoked periodically at a rate $r_i(k)$ at the k^{th} sampling instant of the processor controller. The rate $r_i(k)$ is assumed to take values within the range $[r_i^{min}, r_i^{max}]$.

During the k^{th} sampling instant of the processor controller, images are compressed and transmitted by T_i's data source, UAV$_i$, to the receiver at the rate of $r_i(k)$ images/second. The sampling periods T_s^{out} and T_s^{in} are selected to be larger than the maximum task period. T_s^{out} is selected to be greater than or equal to T_s^{in}. In our model, k^{th} and κ^{th} sampling period represent the k^{th} sampling period of the processor controller and the κ^{th} sampling period of the bandwidth controller respectively. The processor utilization set-point of the receiver and the wireless bandwidth utilization set-point are represented as U^s and B^s, respectively. Although the net available bandwidth B^s is assumed to be constant, the capacity of the wireless network may change dynamically at runtime. However, the available wireless bandwidth can be measured [17] and modeled as a time varying reference signal. The stability of the system can be proved even for a time varying reference signal.

4.1.1 Bandwidth Allocator

During each sampling period of the processor controller, the task of the bandwidth allocator is to compute a desirable bandwidth allocation for each task T_i. The wireless network bandwidth allocation to each task T_i is recomputed by the bandwidth allocator if the presence of an object of interest was detected by any of the target-tracking subtasks during the previous sampling period. For each task, bandwidth is allocated such that the net bandwidth utilization is below the set-point of B^s, *i.e.*:

$$\sum_{i=1}^{n} b_i^s(k) \leq B^s \tag{1}$$

where $b_i^s(k)$ is the bandwidth allocation (utilization set-point) for task T_i during the k^{th} sampling period of the processor controller.

Let $p(k)$ and $p_i(k)$ represent the total number of objects of interest tracked by the system and the number of objects being tracked by T_i during the k^{th} sampling period, respectively. Let b_{min} represent the minimum bandwidth allocation to each task so that images of the lowest quality can be transmitted to the receiver. Bandwidth is allocated to each end-to-end task as a function of $p(k)$ and $p_i(k)$ as follows:

$$b_i^s(k) = \begin{cases} B^s/n & \text{if } p(k) = 0 \\ b_{min} + \frac{(B^s - nb_{min})p_i(k)}{p(k)} & \text{if } p(k) > 0 \end{cases}$$
$$\forall \quad T_i \quad | \quad 1 \leq i \leq n. \tag{2}$$

If the total number of objects of interest tracked by the system is 0, bandwidth is equally allocated to each task. If the total number of objects of interest tracked by the system is greater than 0, bandwidth allocation to tasks is based on the number of objects currently being tracked by that task. This design ensures that a greater amount of bandwidth is allocated to tasks that are currently tracking objects of interest as compared to the ones that are not.

4.1.2 Processor Utilization Controller

We use the approach in [12] to model processor utilization. Section 4.2 uses the following model in the stability analysis of HiDRA. The target-tracking subtask of each end-to-end task T_i has an *estimated* execution time of c_i known at design time. The estimated processor utilization by the target-tracking subtask of task T_i during the k^{th} sampling period is denoted as $E_i(k)$ and is computed as $E_i(k) = c_i r_i(k)$, where $r_i(k)$ is the invocation rate of end-to-end task T_i during the k^{th} sampling period. The net estimated processor utilization during the k^{th} sampling period is therefore

$$E(k) = \sum_{i=1}^{n} c_i r_i(k). \tag{3}$$

At runtime, however, the *actual* execution times may be different since they depend on the presence (and number) of objects in the images. At runtime, therefore, the actual processor utilization $U(k)$ can be written as

$$U(k) = G_p(k)E(k) \tag{4}$$

where $G_p(k)$ is the processor utilization ratio. Although, $G_p(k)$ is unknown, it is reasonable to assume that the worst case utilization ratio $G_p = \max_k\{G_p(k)\}$ is known. From (4), the process utilization model can be written as

$$\Delta U(k+1) = \Delta U(k) + G_p v_p(k) \tag{5}$$

where $\Delta U(k) = U(k) - U^s$ and $v_p(k) = E(k+1) - E(k)$. The task of the feedback controller is to compute $v_p(k)$ so that $U(k)$ converges to U^s (or $\Delta U(k) \to 0$).

We consider a linear proportional controller

$$v_p(k) = K_p \Delta U(k) \tag{6}$$

where K_p is a control gain which will be selected so that the system is stable. The control signal $v_p(k)$ is implemented by the actuators by changing the invocation rate of end-to-end tasks. The closed-loop system is described by

$$\Delta U(k+1) = [1 + K_p G_p]\Delta U(k) \tag{7}$$

The control algorithm is implemented as follows. During each sampling period, the controller compares the current processor utilization $U(k)$ with the utilization set-point U^s, and computes the net estimated utilization $E(k+1)$ for the next sampling period based on the equation $E(k+1) = E(k) + K_p\Delta U(k)$. Since the presence of one or more objects of interest in the received images increases the execution time the target-tracking subtask, computational power is allocated to target tracking subtasks based on the number of objects of interest that are present in the received images. We therefore have

$$E_i(k+1) = \begin{cases} \frac{E(k+1)}{n} & \text{if } p(k) = 0 \\ \frac{E(k+1)p_i(k)}{p(k)} & \text{if } p(k) > 0 \end{cases}$$
$$\forall \quad T_i \quad | \quad 1 \leq i \leq n \tag{8}$$

where $p(k)$ represents the total number of objects of interest captured by all the tasks in the system and $p_i(k)$ represents the number of objects of interest being captured by T_i during the k^{th} sampling period. If the total number of objects of interest tracked by the system is 0, computational power is equally allocated to each task. If the total number of objects of interest tracked by the system is greater than 0, however, allocation of computational resource to tasks is weighted based on the number of objects currently being tracked by that task. This design ensures that a greater amount of computational power is allocated to tasks that are currently tracking objects of interest as compared to the ones that are not. From equations (3) and (8) we derive the task execution rate, as follows:

$$r_i(k+1) = \begin{cases} \frac{E(k)+(U^s-U(k))/G_p}{nc_i} & \text{if } p(k)=0 \\ \frac{p_i(k)(E(k)+(U^s-U(k))/G_p)}{p(k)c_i} & \text{if } p(k)>0 \end{cases} \quad \forall \quad T_i \quad | \quad 1 \le i \le n \qquad (9)$$

4.1.3 Bandwidth Utilization Controller

We present the analytical model of the bandwidth controller for each UAV. The following notations are used in this model:

- $b(\kappa)$: Bandwidth utilization in the κ^{th} sampling period.
- $b^s(k)$: Desired bandwidth utilization (set-point) computed by the bandwidth allocator in the k^{th} sampling period as shown in equation (2).
- $r(k)$: Task rate computed by the processor controller in the k^{th} sampling period, as shown in equation (9).
- s: Size of an uncompressed image, which is a constant and known at design time.
- $q(\kappa)$: Quality factor of image compression algorithm (JPEG) computed by the bandwidth controller in the κ^{th} sampling period.
- $\phi(q)$: Estimated size of the compressed image compressed with quality factor q.

The controlled variable of this feedback control loop is the bandwidth utilization, $b(\kappa)$, and the control input from the controller to the UAV is the quality factor of the image compression algorithm, $q(\kappa)$. The controller computes an appropriate value of $q(\kappa)$ to ensure $b(\kappa)$ converges to $b^s(k)$.

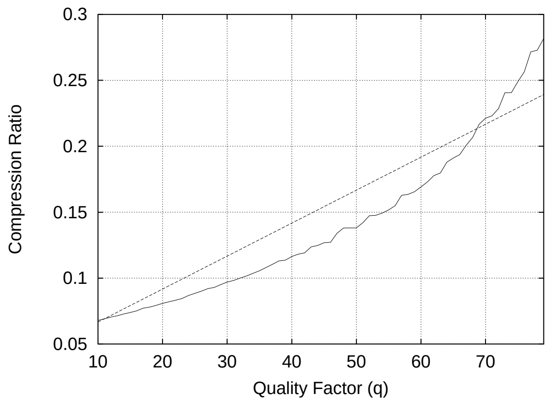


Figure 4: Linearization of $\phi(q)$

The average size of the compressed image, $\phi(q)$, is related to the quality factor of the image compression algorithm, q, by a non-linear function as shown in Figure 4. For the purpose of our control design, however, we choose q within the range $[10, 70]$ where this function can be approximated by a linear one. A piecewise linear function can also be used. For $10 \le q \le 80$, we have

$$\phi(q) = sgq + \omega \qquad (10)$$

where g is the slope and ω is the y-intersect of the linear approximation of the function in Figure 4.

Images are compressed with a quality factor q and transmitted at the rate of $r(\kappa)$ images per second from the UAV to the receiver. The following expression gives the estimated bandwidth utilization by the UAV:

$$\begin{aligned} b(\kappa) &= r(\kappa)\phi(q) \\ &= r(\kappa)sgq(\kappa) + r(\kappa)\omega \end{aligned} \qquad (11)$$

from which we get the network utilization model as

$$\Delta b(\kappa+1) = \Delta b(\kappa) + r(\kappa)sgv_b(\kappa) \qquad (12)$$

where $\Delta b(\kappa) = b(\kappa) - b^s(\kappa)$ and $v_b(\kappa) = q(\kappa+1) - q(\kappa)$.

We consider a linear controller

$$v_b(\kappa) = K_b \Delta b(\kappa) \qquad (13)$$

where K_b is the control gain that will be selected so that the system is stable. The closed-loop system is

$$\Delta b(\kappa+1) = [1 + r(\kappa)sgK_b]\Delta b(\kappa). \qquad (14)$$

During each sampling period, the controller compares the current bandwidth utilization $b(\kappa)$ with the utilization set-point $b^s(\kappa)$, and computes the quality factor $q(\kappa+1)$ by

$$q(\kappa+1) = q(\kappa) + K_b \Delta b(\kappa). \qquad (15)$$

4.2 Stability Analysis

A control system is said to be stable if and only if the system converges to an equilibrium for any set of initial conditions. In our case, the initial conditions are used to represent the abrupt changes in the workload due to the change of the images' content. Our target tracking system is therefore stable if resource utilization of both the system resources (*i.e.*, processor utilization at the receiver and the network bandwidth utilization), converges to their respective utilization set-points in the presence of workload changes. Although the controller is designed based on a time-invariant model (constant upper bounds on resource utilization), we show that the system is stable even when resource utilization changes at run time, *i.e.*, the system is time varying.

We can stabilize each of the two types of feedback control loops by selecting the gains K_p and K_b so that the corresponding poles are in the unit circle. Such a design, however, does not necessarily guarantee the stability of the hierarchical control architecture since it does not take into consideration the interaction between the feedback loops (due to the presence of $r(\kappa)$ in equation (14)). Next, we present an analysis result that allows us to select the control gains so that the overall stability is guaranteed.

Assuming that the input buffer of the receiver is never empty, it is clear that the processor utilization is independent of the bandwidth utilization. If we select K_p so that $-2/G_p < K_p < 0$ then

$$\Delta U(k) = [1 + K_p G_p(k)]^k \Delta U(k_0), k \geq k_0$$

and $\Delta U(k) \to 0$ since $|1 + K_p G_p(k)| < 1$.

From equation (9), it follows that in the steady state the utilization for each task $U_i(k)$ will be stable (it will converge to a set-point U_i^s that depends on the presence of objects in the image data) and we can write

$$\Delta U_i(k+1) = \alpha_i(k) \Delta U_i(k) \tag{16}$$

where the function $\alpha_i(k)$ satisfies $|\alpha_i(k)| < 1$.

Denote r_i^s the rate of the i^{th} task at the steady state, then $r_i(k) = r_i^s + \Delta r_i(k)$ where $\Delta r_i(k) \to 0$. From equation (14), the bandwidth utilization model for the i^{th} UAV is

$$\Delta b_i(\kappa+1) = [1 + (r_i^s + \Delta r_i(\kappa)) sg K_b^i] \Delta b_i(\kappa) \tag{17}$$

Our objective is to deduce the stability properties of the system (16-17) by studying the *isolated system*

$$\begin{aligned} \Delta U_i(k+1) &= \alpha_i(k) \Delta U_i(k) && (18) \\ \Delta b_i(\kappa+1) &= [1 + r_i^s sg K_b^i] \Delta b_i(\kappa) && (19) \end{aligned}$$

where the equations have been decoupled by setting $\Delta r_i(\kappa) = 0$.

Theorem 1. *The system (16-17) is stable if and only if the isolated system (18-19) is stable.*

Proof. Define the norm $||[x_1, x_2]|| = ||[x_1, x_2]||_\infty = \max\{|x_1|, |x_2|\}$ and denote $\Delta U_i(k), \Delta b_i(\kappa)$ and $\Delta U_i^I(k), \Delta b_i^I(\kappa)$ the solutions of (16-17) and (18-19) respectively.

"Only-if": If the system (16-17) is stable, then there exists function $\alpha(\kappa)$ with $\alpha(\kappa) \to 0$ such that

$$||[\Delta U_i(\kappa), \Delta b_i(\kappa)]^T|| \leq \alpha(\kappa) ||[\Delta U_i(\kappa_0), \Delta b_i(\kappa_0)]^T|| \tag{20}$$

$\forall \kappa \geq \kappa_0$ and for every initial condition $[\Delta U_i(\kappa_0), \Delta b_i(\kappa_0)]^T$ where $\Delta U_i(\kappa) = \Delta U_i(k), k \leq \kappa < k+1$.

In particular, suppose that the initial condition is $[0, \Delta b_i(\kappa_0)]^T$, then by equation (20) $\forall \kappa \geq \kappa_0$, $|\Delta b_i^I(\kappa)| \leq \alpha(\kappa) |\Delta b_i^I(\kappa_0)|$, which shows that the system (18-19) is stable.

"If": It is easy to see that $\Delta U_i(k) = \Delta U_i^I(k)$ so we have to analyze only $\Delta b_i(\kappa)$. Define $\eta_I(\kappa) = 1 + r_i^s g K_b^i$ and $\eta(\kappa, \Delta r_i(\kappa)) = 1 + (r_i^s + \Delta r_i(\kappa)) g K_b^i$. From the stability of (18-19), we have that $|\eta_I(\kappa)| < 1$ and there exists a function $\alpha_2(\kappa)$ with $0 \leq \alpha_2(\kappa) \to 0$ such that

$$\Delta b_i^2(\kappa)(\eta_I^2(\kappa) - 1) \leq -\alpha_2(\kappa) \Delta b_i^2(\kappa_0)$$

for every $\Delta b_i(\kappa_0)$ and $\kappa \geq \kappa_0$. But we can write

$$\begin{aligned} \Delta b_i^2(\kappa+1) - \Delta b_i^2(\kappa) &= \Delta b_i^2(\kappa)(\eta_I^2(\kappa) - 1) + \\ &\quad \Delta b_i^2(\kappa)(\eta^2(\kappa, \Delta r_i(\kappa)) - \eta_I^2(\kappa)) \\ &\leq -\alpha_2(\kappa) \Delta b_i^2(\kappa_0) + \gamma(\kappa) \end{aligned}$$

where $\gamma(\kappa) \to$ since $\Delta r_i(\kappa) \to 0$. $\Delta b_i(\kappa) \to 0$ and the system (16-17) is therefore stable. $\square$

Using the above theorem, we can select the control gains so that our hierarchical control architecture is stable. For the processor utilization feedback loop, the gain could be selected to satisfy $-2/G_p < K_p < 0$ that guarantees stability [12, 14]. Similarly, for the bandwidth utilization control loop, the gain should be selected so that (19) is stable. Since r_i^s is not known at design time, we can select the gain to satisfy $-2/(r_i^{\max}) < K_b^i < 0$. A reasonable choice for selecting the control gains is to use deadbeat control [7] based on the worst case utilization ratio and maximum task rate respectively, i.e. $K_p = -1/G$ and $K_b^i = -1/r_i^{\max}$. This selection tries to minimize the settling time keeping the overshoot equal to zero. Other criteria for selection of the gain can be found in [12].

5 Performance Results and Analysis

This section presents the testbed for our target tracking DRE system, which was used to evaluate the performance of HiDRA. We then describe our experiments and analyze the results obtained to empirically evaluate the performance of our DRE system with and without HiDRA under varying input workload. The goal is to validate our theoretical claims and show that HiDRA yields predictable and high-performance resource management and coordination for multiple types of resources.

5.1 Hardware and Software Testbed

Our experiments were performed on the Emulab testbed at University of Utah (`www.emulab.net`). The hardware configuration consists of three nodes acting as UAVs and one receiver node. Images from the UAVs were transmitted to a receiver via a wireless LAN configured with a channel capacity of 1 Mbps. The network bandwidth was chosen to be 1 Mbps since each UAV in the DRE system required a minimum of 350Kbps to transmit 5 images per second, each image of size 320x240 compressed with a quality factor of 30. The hardware configuration of all the nodes was a 3 GHz Intel Pentium IV processor, 1 GB physical memory, 802.11 a/b/g wifi interface (Atheros 5212 chipset), and 120 GB hard drive. The Redhat 9.0 operating system with wireless support was used for all the nodes.

The following software packages were also used for our experiments:

- **Ffmpeg 0.4.9-pre1** with **Fobs-0.4.0** front-end, which is an open-source library that decodes video encoded in MPEG-2, MPEG-4, Real Video, and many other video formats to yield raw images.

- **ImageMagick 6.2.5**, which is an open-source software suite that we used to compress the raw images to JPEG image format.
- **TAO 1.4.7**, which is an open-source implementation of Real-time CORBA [15] that HiDRA and our DRE system case study are built upon.

5.2 Target Tracking DRE System Implementation

The entities in our target tracking DRE system are implemented as CORBA objects and communicate over the *TAO* [16] Real-time CORBA Object Request Broker to achieve desired real-time performance. The end-to-end application consists of pairs of CORBA objects: the UAV data source and the receiver data sink. The UAV data source object that executes on each UAV's on-board processor "pushes" the compressed images to the data sink object via a CORBA oneway method invocation. A data sink object at the receiver processes the images received from the corresponding UAV. Each data sink object contains two functional modules: one that determines the presence of one or more objects of interest in the received images, and the other tracks the coordinates of objects of interest in the received image, if present. The second functional module is executed only if the presence of one of more objects of interest is detected by the first module.

To perform target tracking, received images are compared with a reference image, that is given during system initialization. The received images are converted from color to gray-scale, and the processed image is "subtracted" from the reference image to obtain the difference image. If the average pixel value of the difference image is greater than a threshold (which indicates the presence of one of more objects of interest), the center of mass of the objected is computed. This approach is common, *e.g.*, [6] track the coordinated of moving objects using a Kalman filter.

5.3 Experiment Configuration

Our experiments consisted of three (emulated) UAVs containing the data source object that (1) decoded the video from a file, (2) extracted the raw images, (3) compressed them using JPEG compression, and (4) transmitted the compressed images to the corresponding data sink object at the receiver node. Wireless network bandwidth was shared between the three data source/data sink object pairs, and the computational power at the receiver node was shared between the three data sink CORBA objects.

To evaluate the performance of HiDRA, we monitored the processor utilization at the receiver, and wireless network bandwidth utilization between the UAVs and the receiver. We assume that the channel capacity of the wireless network is constant (1 Mbps). Bandwidth consumption by each UAV is measured by the rate a which data is written to the underlying network stack by the UAV data source CORBA object. Processor utilization at the receiver was measured using the data from the `/proc/stat` file. A smoothing filter such as [2] can be used to suppress the disturbances in the measured resource utilization.

Processor utilization at each UAV node was not monitored since the computational power the UAV on-board processor was sufficiently large to compress images of the highest quality and resolution and transmit them to the receiver without overloading the processor. In our experiments, we also measured application QoS properties such as target-tracking precision and end-to-end execution time. For our experiment, we chose the sampling period of the processor controller and the bandwidth controller as 10 seconds and 1 second respectively. The minimum and maximum image transmission rate $[r_{min}, r_{max}]$ was 5 and 15 images/second. The control gains for the processor controller (K_p) and the bandwidth controller (K_b) were -0.1 and -0.15, respectively. The processor utilization set-point was selected to be 0.7, which is slightly lower than RMS [9] utilization bound of 0.77. Since an IEEE 802.11 DCF-based network has a utilization of approximately 0.7 with 20 active nodes [3], the wireless bandwidth utilization set-point was also selected to be 0.7.

5.4 Analysis of Empirical Results

This section presents the results obtained from running the experiment described in Section 5.3 on our DRE system testbed. We used *system resource* as a metric to evaluate the adaptive resource management capabilities of HiDRA under varying input workloads. Comparison of system performance is decomposed into comparison of resource utilization and application QoS. For system resource utilization, we compare (1) wireless network bandwidth utilization and (2) processor utilization of the receiver node. For application QoS, we compare (1) tracking-precision and (2) mean value of end-to-end execution time.

5.4.1 Comparison of Resource Utilization

Figures 5 and 6 compare the processor utilization at the receiver node and wireless network bandwidth utilization with and without HiDRA. Table 1 summarizes the number of objects of interests that were tracked as a function of time.

Time (sec)	Number of Objects
0 - 200	0
200 - 600	1
600 - 900	2
900 - 1,100	1
1,100 - 1,600	0
1,600 - 1,900	1
1,900 - 2,000	0

Table 1: Objects of Interest as a Function of Time

Figure 5 and Table 1 show that the increase in the processor utilization at $T = 200s$ is due to the presence of the first object of interest. Figure 5 also shows that although the processor utilization reached 0.8, within the next several sampling periods, HiDRA restored the processor utilization to the desired set-point of 0.7. This was achieved by reducing the execution rates of data-source/receiver pair(s) deemed less important, *i.e.*, ones that captured images where objects of interest were absent. Figure 5 shows that when the system was operated without HiDRA the processor utilization remained at 0.9, which is significantly higher than the utilization set-point of 0.7.

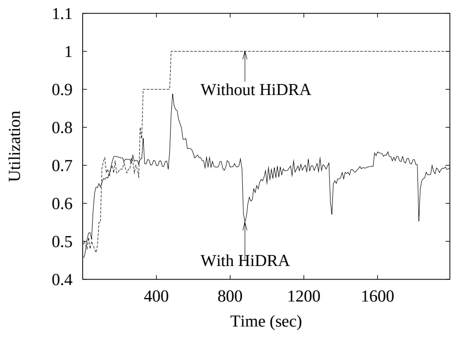


Figure 5: Comparison of Processor Utilization

At $T = 600s$, the presence of the second object of interest was detected. As a results, the processor utilization increased to 0.89 with HiDRA, and the receiver node crashed when the system was operated without HiDRA, which is represented as a utilization of 1.0 for remaining duration of the experiment. When the system was operated without HiDRA, although the receiver crashed at $T = 600s$, the sender continues to send images at a constant rate. This accounts for the network bandwidth utilization shown in Figure 6. Within several sampling periods, HiDRA once again restored the system utilization to the desired utilization set-point.

At $T = 900s$ when the total number of objects currently being tracked reduced from 2 to 1, processor utilization reduced to 0.55. HiDRA restored the processor utilization of 0.7 by increasing the execution rate of important data-source/data sink pair(s). Similarly, HiDRA ensured that the processor utilization converges to the desired set-point for the reaming duration of the experiment.

The results of these experiments show how HiDRA ensures that the processor utilization of the receiver node converges to the desired set-point within bounded time, even under fluctuating workloads. Similarly, from Figure 6 it can be seen that HiDRA ensures that the wireless bandwidth utilization converges to the desired set-point of 0.7 within bounded time, even under fluctuating workloads. We therefore conclude that HiDRA ensures utilization of system resources is maintained within the specified bounds, thereby ensuring system stability.

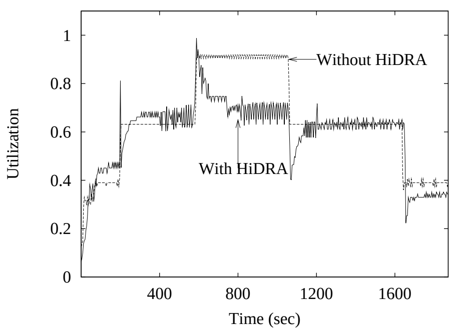


Figure 6: Comparison of Bandwidth Utilization

5.4.2 Comparison of QoS

We now compare the application QoS – (1) target-tracking precision, and (2) average end-to-end execution time.

Target-tracking precision is defined as the inverse of *target-tracking error*, which is the distance between the computed center of mass of an object and the actual center of mass of the object. To compute the actual center of mass of the object, we identified an object present in the video as the object of interest, performed target-tracking on the raw images extracted from the video, and used this value as a reference. At the data sink object, the target-tracking results were then compared with this reference value.

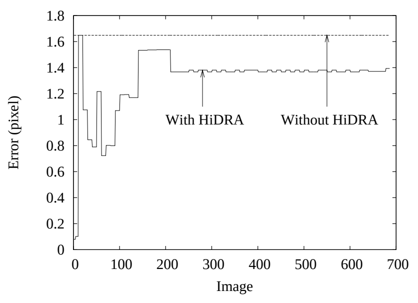
Figure 7 compares the target-tracking error that were obtained when the system was operated with and without HiDRA. As described in Section 5.4.1, the system crashed when operated without HiDRA when the presence of the second object of interest was detected. We therefore use the target-tracking error that was obtained in tracking the first object of interest as the baseline for our comparison. Figures 7a, 7b, and 7c show that average target-tracking error is lower when the system was operated with HiDRA compared to when operated without HiDRA. HiDRA therefore improves the target-tracking precision of our DRE system.

Average end-to-end execution time consists of (1) network transmission latency and (2) processing time at the receiver node. Table 2 compares the end-to-end execution time when the system was operated with and without HiDRA. Since the system crashed when the number of objects being tracked increased to 2, we represent the end-to-end execution time as ∞. Table 2 shows that end-to-end execution time is much lower when the system is operated with HiDRA than when it operates without HiDRA.

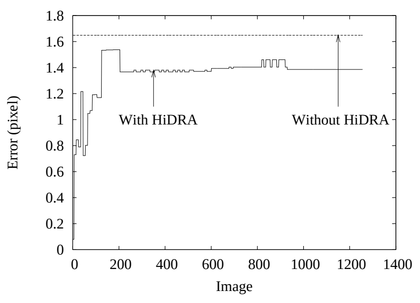
Number of Objects	End-to-End Execution Time (msec)	
	With HiDRA	Without HiDRA
0	10	10
1	40	50
2	80	∞

Table 2: Comparison of End-to-End Execution Time

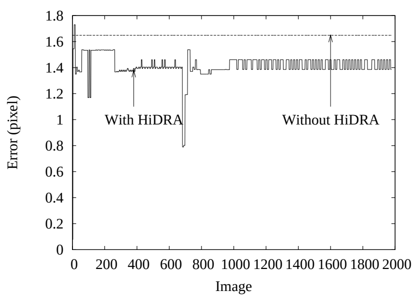
HiDRA responds to fluctuation in resource requirements by constant monitoring of resource utilization. Figures 5



(a) UAV-1



(b) UAV-2



(c) UAV-3

Figure 7: Target-tracking Precision

and 6 show that when resources utilization increases above the desired set-point, HiDRA lowers the utilization by modifying application parameters such as execution rates and JPEG quality factor. These adaptations ensures that (1) system resources are not over-utilized and (2) enough resources are available for important applications. Figure 7 and Table 2 show that the system QoS is significantly higher when the system is operated with HiDRA compared to when it operates without HiDRA.

Our conclusions from analyzing the results described above are that applying hierarchical adaptive resource management to our target tracking system helps to (1) maintain system resource utilization within specified bounds and (2) improve system QoS. These improvements are achieved largely due to monitoring of system resource utilization, adaptive resource provisioning, and efficient system workload management by means of HiDRA's resource monitors, hierarchical controllers, and effectors respectively.

6 Related Work

A number of control-theoretic approaches have been applied to DRE systems to overcome limitations with traditional scheduling approaches that are not suited to handle dynamic changes in resource availability and result in a rigidly scheduled system that adapts poorly to change. A survey of these techniques is presented in [1].

Feedback control scheduling (FCS) [12] is designed to address the challenges of applications with stringent end-to-end QoS executing in open DRE systems. These algorithms provide robust and analytical performance assurances despite uncertainties in resource availability and/-or demand. FC-U and FC-M [13] and HySUCON [8] to manage the processor utilization. CAMRIT [18] 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.

A hierarchical control scheme that integrates resource reservation mechanisms [5, 11] with application specific QoS adaptation [4] is proposed in [14]. This control scheme features a two-tier hierarchical structure: (1) a global QoS manager that is responsible for allocating computational resources to various applications in the system and (2) application-specific QoS managers/adapters that modify application execution to use the allocated resources efficiently and improves application QoS.

Although these approaches are similar to HiDRA, these algorithms/mechanisms perform resource management of only one type of system resource, *i.e.*, either computing power *or* network bandwidth. HiDRA performs resource management of both network and computing resources, which is crucial for real-world DRE systems.

One approach to manage both computing power and network bandwidth might use either the hierarchical control structure proposed in [14], FC-U/FC-M, or HySUCON to manage the processor utilization, and use CAMRIT to manage the network bandwidth utilization. Unfortunately, this approach does not take into consideration the coupling between the two types of system resources and does not necessarily guarantee system stability.

The work of [10] utilizes task control model and fuzzy control model to enhance the QoS adaptation decision of multimedia DRE systems. However, the control framework established in this work is still confined to single type of resource, *(i.e.)*, transmission rate in a distributed visual tracking system.

As described in Section 3, HiDRA uses hierarchical control feedback loops – the processor utilization feedback loop and bandwidth utilization loop – to manage the utilization of system resources, which can be extended to handle more types of resources and end-to-end applications without significant modifications to the existing architecture. Moreover, HiDRA's feedback loops are designed so that the adaptation decisions made by one does not conflict with the decisions made by the other. As shown in Section 4.2, this design results in a hierarchical control architecture that ensures system stability.

7 Concluding Remarks

This paper described HiDRA, which is a hierarchical distributed resource management architecture based on control-theoretic techniques that provides adaptive resource management, such as resource monitoring and application adaptation, that are key to supporting open DRE systems. We first presented the stability analysis of the system to obtain theoretical guarantees that HiDRA ensures system stability of our DRE system. We then evaluated the performance of HiDRA using a representative target tracking DRE system implemented using Real-time CORBA and composed of two types of system resources (computational power at the receiver and wireless network bandwidth) and three applications (UAV data sender/receiver pairs). Our theoretical analysis and empirical results show that HiDRA ensures efficient resource utilization by maintaining the resource utilization of system resources within the specified utilization bounds even under fluctuating work loads, thereby ensuring system stability and delivering effective QoS.

The lessons learned by applying HiDRA to our target tracking system thus far include:

- HiDRA's Control-theoretic approaches yielded in an *adaptive* resource management architecture that can gracefully handle fluctuations in resource availability and/or demand for open DRE systems.
- The formalism of a resource management framework forms the foundation for the design and implementation of a resource management framework based on control-theoretic principles. Moreover, using the formalism, stability analysis of the system can be performed to obtain theoretical guarantees about system performance.
- Developing applications that have various parameters that can be fine-tuned 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.

References

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

[2] M. Amirijoo, J. Hansson, S. Gunnarsson, and S. H. Son. Enhancing Feedback Control Scheduling Performance by Online Quantifi cation and Suppression of Measurement Disturbance. In *RTAS '05: Proceedings of the 11th IEEE Real Time on Embedded Technology and Applications Symposium*, pages 2–11, Washington, DC, USA, 2005. IEEE Computer Society.

[3] G. Bianchi. Performance Analysis of the IEEE 802.11 Distributed Coordination Function. *IEEE Journal on Selected Areas in Communications*, 18(1-2):535–547, Mar 2000.

[4] S. Brandt, G. Nutt, T. Berk, and J. Mankovich. A Dynamic Quality of Service Middleware Agent for Mediating Application Resource Usage. In *RTSS '98: Proceedings of the IEEE Real-Time Systems Symposium*, page 307, Washington, DC, USA, 1998. IEEE Computer Society.

[5] T. Cucinotta, L. Palopoli, L. Marzario, G. Lipari, and L. Abeni. Adaptive Reservations in a Linux Environment. In *IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 238–245, 2004.

[6] F. Dellaert and C. Thorpe. Robust Car Tracking Using Kalman Filtering and Bayesian Templates. In *Conference on Intelligent Transportation Systems*, 1997.

[7] G. F. Franklin, J. D. Powell, and M. Workman. *Digital Control of Dynamic Systems, 3rd edition*. Addition-Wesley, 1997.

[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] B. Li and K. Nahrstedt. A Control-based Middleware Framework for QoS Adaptations. *IEEE Journal on Selected Areas in Communications*, 17(9):1632–1650, Sept. 1999.

[11] G. Lipari, G. Lamastra, and L. Abeni. Task Synchronization in Reservation-Based Real-Time Systems. *IEEE Trans. Computers*, 53(12):1591–1601, 2004.

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

[13] C. Lu, X. Wang, and C. Gill. Feedback Control Real-time Scheduling in ORB Middleware. In *Proceedings of the 9th IEEE Real-time and Embedded Technology and Applications Symposium (RTAS)*, Washington, DC, May 2003. IEEE.

[14] Luca Abeni and Giorgio Buttazzo. Hierarchical QoS Management for Time Sensitive Applications. In *RTAS '01: Proceedings of the Seventh Real-Time Technology and Applications Symposium (RTAS '01)*, page 63, Washington, DC, USA, 2001. IEEE Computer Society.

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

[16] 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.

[17] S. H. Shah, K. Chen, and K. Nahrstedt. Dynamic Bandwidth Management for Single-hop Ad Hoc Wireless Networks. *Mob. Netw. Appl.*, 10(1-2):199–217, 2005.

[18] 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.