Adaptive resource management in asynchronous real-time distributed systems using feedback control functions

B. Ravindran
Virginia Polytechnic Institute and State University

T. Hegazy

Pushkin Kachroo
University of Nevada, Las Vegas, Department of Electrical & Computer Engineering, pushkin@unlv.edu
Adaptive Resource Management in Asynchronous Real-Time Distributed Systems Using Feedback Control Functions

Binoy Ravindran, Pushkin Kachroo, and Tamir Hegazy
The Bradley Department of Electrical and Computer Engineering
Virginia Polytechnic Institute and State University
Blacksburg, VA 24061, Phone: 540-231-3777, 540-231-2976, Fax: 540-231-3362
E-mail: binoy@vt.edu, pushkin@vt.edu, thegazy@vt.edu

Abstract

In this paper, we present feedback control techniques for performing adaptive resource management in asynchronous real-time distributed systems. Asynchronous real-time distributed systems are characterized by significant execution-time uncertainties in the application environment and system resource state. Thus, such systems require adaptive resource management that dynamically monitor the system for adherence to the desired real-time requirements and perform run-time adaptation of the application to changing workloads when unacceptable timeliness behavior is observed. We propose adaptive resource management techniques that are based on feedback control theory. The controllers solve resource allocation problems that arise during run-time adaptation using the classical proportional-integral-derivative control functions. We study the performance of the controllers through simulation. The simulation results indicate that the controllers produce low missed deadline ratios and resource utilizations during situations of high workloads.

1. Introduction

Real-time computer systems that are emerging for the purpose of strategic mission management such as coordination of multiple entities that are manufacturing a vehicle, repairing a damaged reactor, or conducting combat are subject to great uncertainties at the mission and system levels. The computations in the system are "asynchronous" in the sense that processing and communication latencies do not have known upper bounds and event arrivals have non-deterministic distributions. Such real-time mission management applications require decentralization because of the physical distribution of application resources and for achieving survivability in the sense of continuous availability of application functionality that is situation-specific. Because of their physical dispersal, most real-time distributed computing systems are "loosely" coupled using communication paradigms that employ links, buses, rings, etc., resulting in additional uncertainties e.g., variable communication latencies, regardless of the bandwidth.

Most of the past efforts on real-time resource management focus on synchronous (in both the above senses), device-level, sampled data monitoring and regulatory control that is usually centralized, but occasionally distributed [1, 9, 15, 17, 18, 20]. The fundamental premise of these works is that the behavior of the application and the system can be made to be deterministic through extensive a-priori knowledge about load parameters, communications, exceptions, dependencies, and conflicts. The standard real-time theory exploits such a-priori information with static techniques and provides guarantees about application and system behavior under a set of tightly constrained mission and resource conditions that are anticipated in advance. Therefore, it is very difficult to practically employ, adapt, or scale such techniques for real-time systems that are

* Supported by the U.S. Office of Naval Research under Grant Numbers N00014-99-1-0158 and N00014-00-1-0549.
distributed and asynchronous [6, 8, 16, 19]. Asynchronous real-time computer systems and their applications are inherently posteriori in terms of their workload characteristics and thus require adaptive real-time resource management.

Recent advances in real-time distributed systems research [3] have produced technologies that allow mission management applications to specify their requirements such as timeliness and survivability as desired quality-of-service (QoS). Further, the techniques allow the applications to negotiate their service demands along the multiple dimensions of requirements as the availability of resources changes at run-time. QoS is managed through dynamic monitoring of application performance, feedback, and adaptation. Adaptation typically, involves strategies such as application scaling where processes are replicated for exploiting concurrency and load sharing [11, 12] and using the imprecise computational model where the accuracy of computations is traded-off against resource utilization [5]. Such adaptation strategies are employed when applications exhibit unacceptable QoS during situations of high workloads. Most of these works use heuristic strategies for solving dynamic resource allocation problems that occur during adaptation such as determining the (optimal) number of process replicas for load sharing using the classical proportional integral derivative (PID) control function for performing adaptive resource management. The experimental evaluation of the feedback control techniques is presented in Section 6.

In this paper, we propose a radically different approach for performing adaptive resource management: We propose adaptive resource management that is based on feedback control theory. In the discipline of feedback control theory, control theorists design control functions that optimize variables of physical systems or give guarantees of stable performance based on continuous or discrete time feedback. We propose feedback control functions for performing adaptive resource management to achieve real-time requirements. The controllers perform adaptive resource management through run-time monitoring of application timeliness, feedback, and adaptation by application scaling. The controllers solve resource allocation problems such as determining the number of replicas for load sharing using the classical proportional integral derivative (PID) control function and its variant. The performance of the controllers is evaluated through simulations and studied using metrics such as missed deadline ratios and resource utilizations. The simulation results indicate that the controllers are very effective (in terms of the metrics) during situations of high workloads.

2. Scope of the Work

The work presented in this paper is part of a prototyping effort in producing solutions for engineering the future surface combatants of the U.S. Navy [4]. Therefore, we are strongly motivated by the characteristics of Navy combatant systems in our effort. We summarize the characteristics of the application and the assumptions that we have made in our work as follows:

Replication of application program components is employed to achieve application-level scalability. Programs are made scalable by sharing load among replicas. The states of the replicas and their consistency is not addressed in this work, as we assume that the programs process data objects that are "continuous" in the sense that their values are obtained directly from a sensor in the application environment, or computed from values of other such objects. The replicas are thus assumed to be temporarily consistent (e.g., sufficiently up-to-date) without applying every change in value, due to the continuity of physical phenomena.

Thus, the application is constructed with features that will enable it to adapt to workload changes to achieve its timeliness requirements. In constructing an "adaptive resource manager," the question that we are trying to answer is therefore the following: What is the optimal number of replicas of application programs that are needed to achieve acceptable application timeliness during situations of high workloads?

The rest of the paper is organized as follows: We discuss a generic real-time system in Section 3. The generic system is used to reason about the asynchronous behavior of real-time systems that is due to external loads. We describe the adaptive resource management problem that we are studying in this paper in Section 4. Section 5 discusses feedback control techniques for performing adaptive resource management. The experimental evaluation of the feedback control techniques is presented in Section 6. Finally, the paper concludes with a summary of the work and ongoing efforts in Section 7.

3. A General Real-Time System

Figure 1 shows a generic real-time system. The real-time system consists of tasks that perform assessment of the environment, initiation of actions, and monitoring and guidance of the actions to their successful completion. The inter-relationship of the tasks with the environment and the intra-relationship of the tasks among themselves are illustrated in Figure 1.

---

1 It is interesting to observe that adaptive resource management is remarkably similar to feedback control.
The assessment task periodically collects data from the environment using hardware sensors. The data is filtered, correlated, classified, and then used to determine the necessity of an action by the system. When an action is necessary, the task generates an event that activates the initiation task. The initiation task determines the action that needs to be taken and causes actuators to perform the action. Since the task executes in response to an event that can occur at any time, the initiation task has an aperiodic behavior. Upon initiation of the action by the actuators, the guidance task is notified. The guidance task repeatedly uses sensors to collect data, to monitor the actions that were initiated, and to guide the actuators to successful completion of the actions. Note that the activation of the guidance task begins and terminates aperiodically, and once active, it executes periodically. Thus, the guidance task has a transient-periodic behavior.

The real-time requirements of the tasks include deadlines for the completion of each instance of task execution. Observe that during each execution period of the assessment and guidance tasks, the sensor may generate any number of data items, which must be processed by the tasks within the deadline. Furthermore, the sensor data (per period) may result in any number of aperiodic events that trigger the execution of the initiation and guidance tasks, which the tasks must respond and complete within the deadline.

After a careful study of the Anti-Air Warfare (AAW) real-time command and control (C2) system of the U.S. Navy [21], we have observed that the resource needs of the tasks are significantly influenced by the size of the data and the event streams. Size of the data stream refers to the number of data items (sensor reports) that the assessment and guidance tasks have to process during a single execution cycle, and size of the event stream refers to the arrival rate of events that trigger the execution of the initiation and guidance tasks. For systems such as the AAW, data stream sizes (radar tracks) and event (threat) arrivals have neither known upper bounds, nor deterministic distributions. Thus, we attribute the asynchronous behavior of real-time C2 mission management applications that is due to external load to two fundamental factors: (1) unknown upper bounds for the size of data streams processed by periodic and transient-periodic tasks during a single execution cycle and (2) non-deterministic distributions for the arrival rates of events that trigger the execution of transient and transient-periodic tasks.

Observe that the generic model of real-time systems that we have presented here does not capture its distributed nature. As discussed in Section I, often there exists an application "pull" for the decentralization of asynchronous real-time systems that is both involuntary and voluntary. The most common involuntary motivation for decentralization is that the assets of the application (e.g., the radars and missile launching devices of a combat system) are inherently dispersed [2]. Furthermore, real-time (response time) requirements of individual components of such systems often, cannot be met with a centralized computing facility. A primary voluntary reason for decentralization is survivability, in the sense of continued availability with a degradation of functionality or performance [2]. Often, it may be cost-effective to physically distribute a mission management system than it is to implement as a centralized system that becomes a single point of failure.


To illustrate how feedback control laws can be constructed for performing adaptive resource management in asynchronous real-time distributed systems, we consider an example resource management problem. We use the example problem as a benchmark problem throughout the paper for designing feedback control techniques.

- sub-task 1
- sub-task 2
- sub-task n

Figure 2. Sequential Sub-Tasks of an End-to-End Task

We assume a distributed system with a real-time task that is required to process data that arrives periodically. The upper bound on the size of the data that arrives during each period is assumed to be unknown a-priori. However, the task is required to complete each of its periods within a specified end-to-end deadline. The task is assumed to consist of n sub-tasks. The connectivity of the...
sub-tasks is assumed to be “sequential” i.e., sub-task $i$ needs to be completed before sub-task $i+1$ can begin its execution (see Figure 2). The sub-tasks of the task are assumed to be replicable so that the replicas can be dynamically executed on different computing nodes to exploit concurrency, achieve load sharing, and reduce end-to-end task latency when the data size increases at run-time and causes unacceptable task timeliness. The application hardware is assumed to consist of a set of homogenous processors that are distributed over a geographical region. The processors are interconnected together using a shared communication medium.

In designing a feedback controller for this benchmark problem, we define a three-fold objective: (1) to reduce the task execution time during overloaded situations that are caused due to high data stream sizes so that the task deadline can be satisfied, (2) to keep the processor and network utilization as low as possible, and (3) to use the minimum number of sub-task replicas. As the increase in the number of sub-task replicas will reduce the task execution time, but will increase the processor and network utilization, the controller has to compromise between the objectives so that the deadline can be satisfied with the minimum number of replicas and minimum resource utilizations. The controller for the problem therefore, has to make the following decision: How many replicas of each sub-task are needed for each period or what should be the change in the number of sub-task replicas for each period?

5. Feedback Control

Feedback control is usually provided in terms of a control law that gives a control input, which is defined as some function of the measurements of the system. Therefore, the system should be controllable by the control variable and the measurements should have a relationship to the states of the system that are controlled. The process of control design thus becomes determining a mapping from the system measurements to the control variable. The control law is to be designed to achieve some closed loop performance of the system [22].

\[ u(t) = k_d e(t) + k_i \int_0^t e(t) dt + k_p e(t) \]

**Figure 3. Classical Feedback Control**

In order to design a control law for a specific objective, we need to be able to assess if the objective is met by using the control law. Therefore, we need a nominal model of the system. The nominal model should be rich enough to provide the essential dynamics of the system. It may be possible to construct a highly complex model that represents the actual physical system as close as possible. However, no model can truly represent an uncertain system. Moreover, a highly complex model may represent the dynamics more accurately, but the control design can become impossible in such situations. The aim of a good (robust) controller is to provide nominal performance for the nominal model as well as robust performance against the un-modeled and other uncertainties and disturbances of the real system [10, 13].

The classical feedback control is illustrated in Figure 3. In the figure, the “Plant” block shows the nominal model and $d(t)$ is the disturbance that includes components from un-modeled dynamics, uncertainties, and other disturbances. The variable $y(t)$ represents the measurements of the system. The variable $r(t)$ represents some reference signal. The error variable $e(t)$ is the difference between the reference and the measurement variables. The controller block uses $e(t)$ as the input to calculate the control variable $u(t)$. The control variable $u(t)$ is then used as the input to the plant to produce the measured variable $y(t)$.

5.1 PID-Controller for Adaptive Resource Management

We present the design of a proportional integral derivative (PID) control function for the benchmark adaptive resource management problem. The function uses the sum of weighted error, integral of error, and derivative of error terms as the control variable. To design a PID controller, we first define the sampling time. We define the sampling time to be the end of each period of data arrival. A decision is taken at the beginning of each period based on the behavior of the system in the past period(s). The controller input (i.e., the error) $e(k)$ can be defined as:

\[ e(k) = w_1(x(k) - X(k)) + w_2(u(k) - U(k)) + w_3\tau(k) + w_4m(k) \]

(1)

where $k$ is the sampling instant, $x(k)$ is the actual execution time of the task for processing the data that arrived in the previous period, $X(k)$ is the desired execution time for the task, $u(k)$ is the average actual utilization of the processors during the previous period, $U(k)$ is the desired utilization, $\tau(k)$ is the actual average actual network utilization throughout the previous period, and $m(k)$ is the missed deadline ratio. Based on the error term, a PID control function that computes the change in the number of replicas for each sub-task of the task is given by:

\[ \Delta s_i(k) = k_0e(k) + k_1\sum_{j=0}^i e(j) + k_2e(k-1); \quad i \in [0, 1, n] \]

(2)
\[
\Delta s_t(k) = -A_2 \\
\Delta s_t(k) = -A_1 \\
\Delta s_t(k) = 0 \\
\Delta s_t(k) = \infty
\]

\[ T_3 \quad T_2 \quad 0 \quad T_1 \]

\[ A_1 > 0, \quad A_2 > A_1, \quad T_1 < 0, \quad T_3 > 0 \]

**Figure 4. A P-Controller for Adaptive Resource Management**

5.2 A P-Controller for Adaptive Resource Management

We can also design a P-controller that determines the change in the number of sub-task replicas using the error described in Equation (1). The P-control function characterizes the error values into four levels. For each level, the controller will perform a different decision as shown in Figure 4. The idea behind the function is summarized as follows:

1. If the error value exceeds a positive specified threshold \( T_1 \), then the performance of the task in the last period is assumed to be worse than the desired performance. This is because a large data stream was received in the last period and could not be satisfied by the current number of sub-task replicas. Furthermore, most likely, the system is going to encounter a larger data stream in the current period. So the controller will make the maximum possible positive change in the number of sub-task replicas to enable the system to handle the large data size.

2. If the error value falls between \( T_1 \) (which is positive) and \( T_2 \) (which is negative), then the task performance is assumed to be satisfactory and no change in the number of sub-task replicas is required.

3. If the error value falls between \( T_2 \) and \( T_3 \) (which are both negative), the task performance is assumed to be much better than the desired performance. This is because the system is using a large number of sub-task replicas and therefore, the task deadline could be possibly satisfied with less number of sub-task replicas. So the controller will reduce the number of sub-task replicas by a constant factor \( A_1 \).

4. If the error value is less than \( T_3 \), then the assumption made in step 3 holds. However, we will assume that the number of sub-task replicas is much more than what is needed. So, the number of sub-task replicas will be reduced by another constant factor \( A_2 \) that is larger than \( A_1 \).

6. Performance Evaluation

We evaluate the performance of the feedback control techniques through simulation. We present the resource management architecture in Section 6.1. The baseline parameters of the simulation study are discussed in Section 6.2. We discuss the simulation results for the feedback controllers in Section 6.3.
on the respective processors using the EDF scheduling scheme.

Data

<table>
<thead>
<tr>
<th>Maximum</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Minimum</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 6. Input Data Arrival Pattern**

At the beginning of each period, data is generated according to the step-wise pattern shown in Figure 6. The generated input data is fed to the resource allocator. Information regarding resource utilization such as CPU utilization and task performance such as sub-task execution times and missed deadlines are also fed to the controller at the beginning of each period. The controller computes the change in the number of sub-task replicas at the beginning of the period and conveys this information to the resource allocator. The allocator determines a processor allocation for the sub-tasks and their replicas and conveys the allocation decision to the local schedulers. Further, the allocator splits the input data and distributes the data to the sub-task replicas. We use a load-balancing algorithm for processor allocation that determines the replica-to-processor assignment by trying to maintain the same utilization for all the processors.

6.2 Baseline Parameters of Simulation Study

We evaluate the performance of the feedback control algorithms by comparing their performance with two “fixed” controllers — control functions that use a constant number of sub-task replicas. The two fixed controllers that are used in the study include: (1) a controller that uses the maximum number of sub-task replicas possible to exploit maximum concurrency and (2) a controller that uses half the maximum number of sub-task replicas that are possible. Thus, the resource management algorithms that are the focus of the study include: (1) PID-controller called PID, (2) P-controller called P, (3) number of sub-task replicas fixed at the maximum called FIXED6, and (4) number of sub-task replicas fixed at half the maximum called FIXED3.

Observe that the EDF scheduling algorithm requires deadlines for sub-tasks in order to schedule the sub-tasks on the processor. We derive deadlines for sub-tasks from the task deadline using the equal-flexibility (EQF) strategy that is proposed in [7]. EQF assigns deadlines to sub-tasks that are proportional to the estimated execution times of the sub-tasks.

<table>
<thead>
<tr>
<th>Table 1. The Baseline Parameters of Simulation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of sub-tasks</td>
</tr>
<tr>
<td>Minimum data size (1 unit)</td>
</tr>
<tr>
<td>Execution time of sub-task per datum</td>
</tr>
<tr>
<td>Network transmission rate</td>
</tr>
<tr>
<td>Data Period</td>
</tr>
<tr>
<td>Simulation time per experiment</td>
</tr>
<tr>
<td>Number of CPUs</td>
</tr>
<tr>
<td>Maximum number of replicas</td>
</tr>
<tr>
<td>Task deadline</td>
</tr>
</tbody>
</table>

The baseline parameters of the simulation study are shown in Table 1. We obtained parameters such as minimum data size and execution time for a unit data size from actual measurements of a real-time benchmark application [14].

6.3 Simulation Results for PID and P Controllers

Figures 7, 8, 9, and 10 show the missed deadline percentages, average number of sub-task replicas, average CPU utilization, and average network utilization, as the maximum data stream size varies (Figure 6) for the feedback controllers, respectively. Each data point in the figures is obtained from a single experiment. The data size that corresponds to the data point in the figures is the maximum data size that was used in the experiment.

<table>
<thead>
<tr>
<th>Missed Deadlines</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
</tr>
<tr>
<td>PID</td>
</tr>
<tr>
<td>FIXED3</td>
</tr>
<tr>
<td>FIXED6</td>
</tr>
</tbody>
</table>

**Figure 7. Missed Deadline Percentages**

The results shown in the figures are highly encouraging. They show superior performance of the feedback controllers over that of the fixed controllers in terms of simultaneously reducing the missed deadline ratio and using the minimum number of sub-task replicas. Note that the FIXED6 controller gives the lowest possible
missed deadline ratio among all other techniques shown above. However, the controller uses a large number of replicas (more than what is necessary) and therefore its CPU and network utilization is high (see Figures 9 and 10). On the other hand, both the feedback control functions give a non-optimal but very low missed deadline ratios (compared to the fixed controllers) with lower number of replicas. This is true for all the data values chosen for simulation. Thus, the PID and P controllers outperform the fixed controllers.

Figure 8. Average Number of Sub-task Replicas

Figure 9 also illustrates that using the maximum number of sub-task replicas will cause high CPU utilization because every available processor in the network is running a replica of each sub-task all the time. However, when half the maximum number of sub-task replicas is used (i.e. in FIXED3), we observe that the resulting CPU utilization is lower than that of PID and P. This is because whenever a sub-task misses its deadline, we abort its parent task. Hence, the CPUs will remain idle for longer periods.

Figure 9. Average CPU Utilization

Finally, Figure 10 shows that the FIXED6 controller causes higher network utilization than the other algorithms. This is because, the FIXED6 controller causes more communication activity as the data is always divided and distributed among all the processors. Also, note that compared to the FIXED6 controller, the feedback controllers causes lower network utilization.

7. Conclusions

In this paper, we present feedback control functions for performing adaptive resource management in asynchronous real-time distributed systems. We design PID and P-control functions for an example adaptive resource management problem – determining the number of replicas that are needed to adapt the application to workload changes. The performance of the control functions is studied through a set of simulation experiments. The experimental results illustrate superior performance of the control functions when compared to controllers that use no feedback control for the same problem. This indicates the promise of feedback controllers for such types of problems.

However, it is important to note that control design is performed here using a generic, controller architecture and the gains of the control laws are chosen based on heuristics that rely upon the control experience of the designer. This might work in many practical applications, but may not give a generic model and performance guarantees for most problems.

In the discipline of feedback control theory, control theorists analytically design control functions that give guarantees of stable performance based on continuous or discrete time feedback. However, control functions that are analytically designed and studied in control theory for the most part, are linear-time invariant systems in continuous and discrete times. Asynchronous real-time
distributed systems are inherently nonlinear. Thus, analytically designing feedback control functions for performing adaptive resource management so that their behavior can be theoretically studied becomes an important problem. This is the focus of ongoing efforts.

References