FARA – A Framework for Adaptive Resource Allocation in Complex Real-Time Systems*

Daniela Roşu, Karsten Schwan, Sudhakar Yalamanchili
Georgia Institute of Technology
Atlanta, Ga. 30332-0250
{ daniela,schwan}@cc.gatech.edu, sudha@ee.gatech.edu

Abstract

This paper introduces FARA, a framework that provides abstractions and mechanisms for building integrated adaptation and resource allocation services in complex real-time systems. FARA's design addresses (1) the specification of the adaptation capabilities of general-domain applications with multiple adaptive components and (2) the control of the adaptation impact on the satisfiability of an application's timing constraints during its transition to targeted steady state. We propose to use of a hierarchical adaptation model, to control the adaptation enactment overhead based on the costs of executing the application-specific adaptation procedures, and to reduce the decision overhead by use of effective solutions for mechanisms frequently invoked during the decision process, such as the decision context management and the allocation decision evaluation.

1. Introduction

Complex real-time systems often rely on dynamic allocation of available resources in order to insure efficient and correct execution despite changes in the set of active tasks, in the resource needs of the already active tasks, or in the resource availability. Adaptation is attractive because it facilitates dealing with dynamic application behavior in the context of limited resource availability. Real-time system developers have always realized the importance of adaptations, typically describing an application's functionality in terms of its execution modes [8].

*Funded by the Defense Advanced Research Projects Agency under Rome Laboratories contract F30602-96-C-0321.

Support for making adaptation decisions is provided by infrastructures embedded in the application [3, 5, 7], or in the operating system resource management component [1, 4, 6, 14]. In our research, we consider the latter approach because it eliminates the need for integrating in each application mechanisms for performance control and complex adaptation decision, and it also enables effective negotiation-based decisions[1]. However, such "system-decided" adaptations require detailed information on each application's adaptation capabilities. Previous research has only considered models for describing the adaptation capabilities of applications in specific application domains, such as multimedia [4, 9], C3I applications with components performing imprecise computation-type of algorithms [5], or applications demanding a fixed set of resources and service types [1, 6, 11].

A challenging problem associated with runtime adaptation and resource allocation (ARA) decisions is their potential negative impact on the satisfiability of an application's timing constraints. This impact is due to the delays and overheads incurred with making and enacting the adaptation that can affect the normal execution of the application[13]. Therefore, in contrast to previous research that considers the goal of adaptation to be only the attainment of acceptable steady state performance [1, 5, 6], our solution to ARA considers also the application performance during its transition to the targeted steady state, referred as *transitory state performance*.

Contributions. This paper proposes a **F**ramework for **A**daptative **R**esource **A**llocation (**FARA**) in the context of real-time systems. FARA provides abstractions and mechanisms that enable integrated, runtime adaptation and resource allocation decisions in systems with multiple applications and varied resources, while it facilitates the portability across multiple implementation platforms and applications. FARA is addressing two problems: the specification of the adaptation capabilities of a complex application, and the control of the transitory state performance through the reduction and control of the overheads incurred when making and enacting adaptation decisions. Towards addressing these issues, FARA proposes:

- a model for description of the adaptation capabilities of a complex application, that builds on top of the application-specific hierarchy of adaptable components;
- mechanisms that reduce the decision overheads by eliminating repetitive invocations of resource schedulability analysis and QoS evaluation procedures involved in the allocation decision evaluation; and
- a model for description of application-specific adaptation costs and methods for using this information in selecting adaptation solutions with low impact on transitory state performance.

Paper Outline. The remainder of this paper briefly presents the FARA architecture (see Section 2) and its solutions to several issues related to making adaptation and resource allocation decisions (see Section 3). Details on these topics and on related work are presented in [12].

2. FARA Architecture

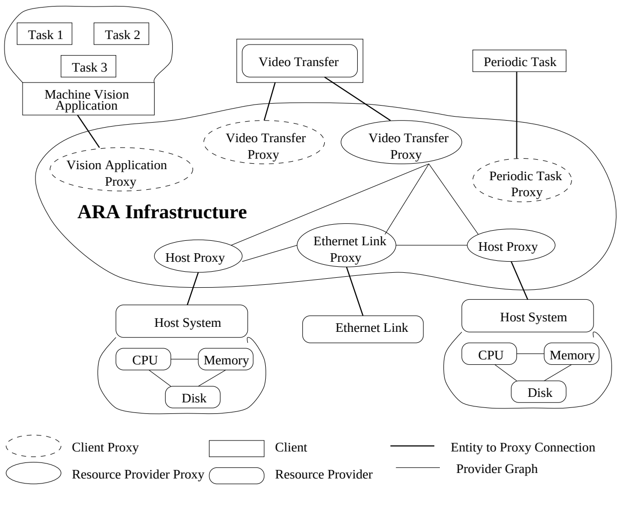

Figure 1. System Model: applications, resources, and FARA-based infrastructure

Target System Model. A FARA-based infrastructure is a middleware system component that mediates between resource providers and clients in order to guarantee acceptable reservation-based services (see Figure 1). A resource provider represents a physical or logical system resource, and a client represents an application or a logical system resource. The system includes multiple resource providers and it supports multiple concurrent, independently authored applications.

A *resource provider* may represent a set of resources with integrated allocation procedures. For instance, a "host system" provider integrates CPU, memory, communication interfaces, and disk resources. A provider offers one or more types of reservation-based services, each characterized by specific request parameters and quality metrics (e.g., a host system might offer both predictive and time-shared computation [11]). The physical connections between providers are described by a graph. This graph is used to decide how services of various providers can be used together to satisfy high-level client requests (e.g., distributed processing across several hosts and network links).

A *client* is aware of its resource and performance requirements, and it can explicitly describe its adaptation capabilities. A client is adaptable concerning (1) the **level** of requirements (e.g., frame rate for a video transfer service), and (2) the **set** of requirements (e.g., each parallel configuration in which a parallel application can correctly execute is described by a specific set of computation, memory, and communication requirements).

Clients can issue allocation requests at start-up time, and can adjust these requests at run-time during mode changes or when unexpected changes in the application behavior result in resource needs larger than those specified in the most recent allocation request. When an allocation request is received, the FARA-based infrastructure chooses an acceptable set of requirements for the requesting client, and then, it makes resource reservations such to satisfy each its performance requirements. However resources are scarce, besides the requesting client, a decision might change the adaptation and resource allocation of several other clients in the system. The infrastructure informs all the affected clients about the selected adaptation modes and provides reservation handles used in acquiring the reserved resources.

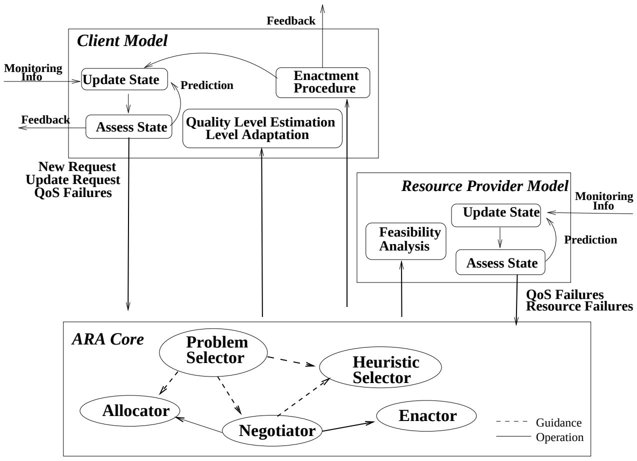

Figure 2. Components and interactions in a FARA-based infrastructure.

Architecture Principles. A FARA-based infrastructure is composed of a decision core, called "ARA core", and proxies for each client and resource provider in the system (see Figure 2). The proxies provide client and resource provider mechanisms (1) for decision evaluation and decision enactment; (2) for detection and prediction of runtime changes in a client's needs and in resource availability; and (3) for interface between applications and ARA infrastructure. Proxies interact with the system under control through monitoring streams and feedback channels. FARA makes no assumption on where these proxies are physically located, as for instance, in the client's address space or in the infrastructure's address space.

The ARA core provides the mechanisms for making ARA decisions. The core integrates a set of decision heuristics and mechanisms for guiding the decision process. The fundamental principles of FARA decisions are the following: (1) Each request has a deadline, determined by the criticality of quality requirements or by client-specific timing constraints (e.g., execution start time). This information is used to guide the decision process such to ensure a minimal impact on the satisfiability of the real-time constraints of an application. (2) Each request is solved by using the most appropriate heuristic in the available set. The goal of the heuristic selection is to reduce the decision time while providing a good quality decision. Finally, (3) allocation and negotiation decisions are separate activities. Allocation decisions concern the assignment of resource requirements to providers such to satisfy the associated performance requirements. Negotiation decisions concern the selection of the set/level of requirements that should be satisfied by the allocation decision. The separation is motivated by their different goals, input and output parameters and it enables the use of specialized heuristics.

FARA Abstractions. A FARA-based infrastructure is built from four categories of abstractions: representation, operation, guidance, and infrastructure.

Representation abstractions describe clients and resource providers. The main representation abstractions are the following: resource requirement, quality requirement, dependency group, contract request, adaptable entity, and resource provider.

Resource and *quality requirements* are described by a type, a set of acceptable levels, and a target level. A quality requirement provides methods for evaluation of the expected quality level based on the levels guaranteed for the dependent resource requirements. A *dependency group* represents a set of resource or quality requirements whose acceptable levels are interdependent. The group provides methods for correlated adjustment of the target levels of these requirements, which are usually invoked upon unsuccessful allocation decisions. A *contract request* represents a set of resource requirements, quality requirements, and dependency groups. An *adaptable entity* represents a client, and is described by one or more contract requests, called "adaptation modes". A *resource provider* performs operations related to the estimation of resource load and the feasibility analysis of allocation requests.

Operation abstractions provide methods for making and enacting ARA decisions. The main abstractions in this group are the following: detector, negotiator, allocator, and enactor. A *detector* provides a method for identifying runtime changes in a client's or provider's behavior. For example, a detector may check whether a given percentage of the raw or smoothed observed values are below some threshold, or it may apply a Bayesian model over the result of another detector [10].

A *negotiator* represents a negotiation heuristic. In FARA, the primary goal of negotiation is to select a set of adaptation modes with a high likelihood of being satisfied. Criteria considered by a negotiation heuristic are:

- steady state performance, reflected in the

values associated to client's adaptation modes, and in the guaranteed quality levels;
- transitory state performance, reflected in the expected enactment delay (e.g., adaptation procedure overhead, number of adaptable entities affected by the decision), and in the allocation decision overhead (e.g., number of new or increased resource requirements).

An *allocator* represents an allocation heuristic, and an *enactor* represents a protocol for ARA decision enactment. An enactor's goal is to order the execution of the adaptation procedures for each of the affected adaptable entities such to avoid long blocking delays upon new resource acquisition.

Guidance abstractions provide methods for selection of the operation objects to use during each ARA decision process. These abstractions are the problem selector and the heuristic selector. A *problem selector* decides which of the pending requests should be addressed first. For instance, one might give priority to requests triggered by runtime changes in the needs of an already active client over requests of newly arrived clients. Further, among requests in the former category, one might service the requests in the non-decreasing order their deadlines.

A *heuristic selector* identifies the allocator, negotiator, and enactor objects that are expected to solve a given request within the corresponding deadline. Possible criteria are: the type of request (e.g., new versus adjustment, resource failure versus detection of service needs larger than specification), and type of quality requirements to satisfy (e.g., rate versus latency).

Infrastructure abstractions provide the functionality needed by FARA objects in order to interact with each other and with the system under control, including abstractions for evaluation of an allocation decision, for manipulation of decision context, and for monitoring and feedback.

The FARA abstractions are used to build the components of a FARA architecture as follows: (1) client and provider proxies include representation, operations, and infrastructure abstractions, (2) the ARA core includes operation, guidance, and infrastructure abstractions.

3. Solutions to ARA Challenges

In this section we present the solutions offered by FARA to the following problems:

- description of the adaptation model for complex applications;
- control of the adaptation enactment overhead; and
- reduction of the adaptation decision overhead.

Hierarchical Adaptation Model. A complex client will have several interacting components, each with specific requirements and adaptation capabilities, and with constraints on how per-component adaptations may be combined into acceptable client-level adaptations. For instance, in the vision application in Figure 1, Task1 executes optic flow estimation, Task2 executes model estimation, and Task3 executes region classification. These tasks may each execute with different levels of parallelism (i.e., adaptation modes), but Task2 and Task3 levels always have to operate with the same level in order to minimize the loss in output accuracy.

The FARA *hierarchical adaptation model* describes an application as a hierarchy of adaptable entities, built on top of the application's hierarchy of functional components. Each application component is described by an adaptable entity object with one or more adaptation modes. Besides the resource and quality requirements, each adaptation mode includes:

- the list of the acceptable adaptation modes for the components at the next lower level in the hierarchy; and
- hierarchical dependency groups, that describe relationships between the levels of quality requirements in the particular contract request and the level of resource and quality requirements of components on the next lower level.

Potential advantages of FARA's hierarchical adaptation model are: (1) the ability to capture the adaptation capabilities of complex adaptive applications with constraints imposed on combining components' adaptation modes; (2) the compact representation, with no replication of an adaptation mode's specifications when it is part of multiple higher-level adaptation modes; and (3) the ability to minimize the number of application components affected by the adaptation by understanding what component is affected by a failed resource and quality requirement and how components' adaptations can be combined into acceptable higher level adaptations.

Overhead of Adaptation Enactment. In order to reduce the impact of adaptation on transitory application performance, it is necessary to evaluate and control the enactment overhead. This overhead has (1) a system component, resulting from operations like thread/process migration, activation, and shutdown, and (2) a client component, resulting from the execution of application-specific procedures, such as state redistribution, and component initialization and shutdown.

While the system-related overheads could be a-priori known to the ARA infrastructure, the client-related overheads have to be explicitly specified for each adaptive application. We submit that this information, called *adaptation cost*, has three dimensions: *quantitative*, describing the amount of resources needed to change between adaptation modes, *qualitative*, describing the drop in quality level that might occur due to the execution of the adaptation procedure, and *temporal*, describing the constraints that can delay the execution of the adaptation procedure.

FARA enables to describe this information for each transition between adaptation modes. The information is used in the ARA decision process to quantify the adaptation overhead and the delay in reaching the targeted steady state, called enactment delay. More specifically, for each adaptable entity, the *adaptation overhead* can be estimated by the time needed to satisfy these requirements with the steady state resource reservations, or simpler, by the sum of all of the overheads associated with each "active" resource – e.g., computation, communication, and disk. The enactment delay depends on the adaptation overhead and the enactment delays of the adaptable entities that have to release the resources to be acquired during the adaptation enactment.

Note that this adaptation cost information also enables making resource reservations for the execution of the adaptation procedures. This is necessary in a reservation-based system, and especially when the needs in the steady state and the needs of the adaptation procedure are different.

ARA Decision Overhead. The decision overhead is a significant component of the delay to reaching the steady state after an ARA decision is triggered by runtime changes in the resource needs of an application. Deriving from our interest in reducing the impact of adaptation on the transitory state performance, the FARA design is also focused on reducing the decision overhead. We have identified two mechanisms that are frequently invoked during the decision process and propose solutions that eliminate part of the associated overhead. More specifically, we propose new solutions for the management of decision contexts and the evaluation of allocation decisions.

Decision Context Management. It is typical for resource allocation algorithms to maintain and manipulate multiple contexts describing partial solutions (e.g., in a branch-and-bound algorithm) or complete solutions (e.g., in a best-fit algorithm). Each such context includes parameters describing the current allocation and adaptation state and, also, entity-specific parameters which are derived from this state. For instance, for a CPU provider such derived parameters are the schedule description and the utilization level. Decision overhead can be reduced if the derived parameters can be saved and restored together with the (partial) solution description, instead of reevaluating them each time the solution is reconsidered.

In order to address this need, FARA implements a *collaborative approach* to decision context management. More specifically, a decision context is a container for pointers to object-state descriptors. Any object involved in the decision process (e.g., resource provider, quality requirement, dependency group) can register its state with a context thereby providing whatever detail is deemed necessary for future decision steps. As the context description includes pointers and not data structures, no copy overhead is incurred when the object's state is updated. In addition, the management mechanism keeps track of context precedence (i.e., context A precedes context B, if B is obtained by applying some allocation or adaptation decision to A), thereby eliminating the need for replication when an object's state does not change from one decision context to another.

Allocation Decision Evaluation. The typical solution to analyzing the feasibility of an assignment request is to invoke the appropriate service provider to analyze the request and then, to invoke all the quality requirements that might be affected by the request in order to reevaluate their predicted values. This per-request invocation technique might result in unnecessarily large overheads when the allocator generates the assignment requests in batch (e.g., such as for a resource reallocation) and several requests are addressed to the same service provider or affect the same quality requirements. For instance, the time taken to analyze a set of CPU allocation requests for a deadline-based scheduling method[2] depends on the order in which the requests are considered.

In order to eliminate such unnecessary overheads, FARA provides an asynchronous, batch, and prioritized invocations of the entities affected by such decisions that permits to invoke a resource provider or quality requirement object only once for an entire batch of requests. The implementation of this invocation mechanism is based on a signaling infrastructure. The signal receivers are the entities whose reevaluation depends on other entities, and the signal transmitters are the entities that reevaluate their state. These signals, called "update signals", have types and priorities. An *"update carrier"* abstraction intermediates between transmitters and receivers. The carrier has two execution modes: collection mode, when it gathers the transmitted signals, and delivery mode, when it invokes all of the receivers interested in one or more of the collected signals. Invocation order is determined by signal priority and triggering order, and all signals of interest to a receiver are delivered at once.

4. Conclusions

This paper proposes a framework, called FARA, for building ARA infrastructures as middleware components in modern real-time systems. FARA offers: (1) a hierarchical model for describing the adaptation capabilities of multi-component applications; (2) a description of adaptation costs based on which the enactment delay may be evaluated; and (3) mechanisms that reduce the decision overhead through collaborative manipulation of decision context and batch evaluation of allocation decisions.

A FARA-based ARA infrastructure is under development for a cluster of UltraSPARC workstations connected by an ATM network. It will be evaluated with multiple real-time applications including a video transmission, computer vision, and synthetic real-time benchmarks.

References

[1] T. F. Abdelzaher, E. M. Atkins, and K. G. Shin. QoS Negotiation in Real-Time Systems and Its Application to Automated Flight Control. *Real-Time Technology and Applications Symposium*, 1997.

[2] N. C. Audsley, A. Burns, M. F. Richardson, and A. J. Wellings. Hard Real-Time Scheduling: The Deadline-Monotonic Approach. *8th RTOSS*, May, 1991.

[3] T. Bihari and K. Schwan. Dynamic Adaptation of Real-Time Software. *ACM Transactions on Computer Systems*, May, 1991.

[4] J. Huang and P.-J. Wan. On Supporting Mission-Critical Multimedia Applications. *3rd IEEE International Conference on Multimedia Computing and Systems*, June 1996.

[5] D. Hull, A. Shankar, K. Nahrstedt, and J. W. S. Liu. An end-to-end QoS model and management architecture. *IEEE Workshop on Middleware for Distributed Real-time Systems and Services*, pages 82–89, Dec.,1997.

[6] J. Jehuda. Automated Meta-Control for Adaptable Real-Time Software. *Real-Time Systems Journal*, (to appear).

[7] M. B. Jones, P. J. Leach, R. Draves, and J. Barrera III. Modular Real-Time Resource Management in the Rialto Operating System. *5th Workshop on Hot Topics in Operating Systems, Orcas Island, WA*, pages 12–17, May, 1995.

[8] D. J. Musliner, E. H. Durfee, and K. G. Shin. CIRCA: A Cooperative Intelligent Real-Time Control Architecture. *IEEE Transactions on Systems, Man, and Cybernetics*, 23(6):1561–1574, Nov./Dec. 1993.

[9] K. Nahrstedt and J. M. Smith. The QOS Broker. *IEEE MultiMedia*, Spring, 1995.

[10] D. Paul. Decisions Models for On-Line Adaptive Resource Managament'. *Masters Thesis, School of Electrical and Computer Engineering, Georgia Institute of Technology*, 1997.

[11] R. Rajkumar, K. Juvva, A. Molano, and S. Oikawa. Resource Kernels: A Resource-Centric Approach to Real-Time Systems. *SPIE/ACM Conference on Multimedia Computing and Networking*, Jan.,1998.

[12] D. Rosu, K. Schwan, and S. Yalamanchili. FARA – A Framework for Adaptive Resource Allocation in Complex Real-Time Systems. *Georgia Inst. of Technology, GIT-CC-98-10*, Mar.,1998.

[13] D. Rosu, K. Schwan, S. Yalamanchili, and R. Jha. On Adaptive Resource Allocation for Complex Real-Time Applications. *18th IEEE Real-Time Systems Symposium*, Dec., 1997.

[14] D. I. Rosu and K. Schwan. Improving Protocol Performance by Dynamic Control of Communication Resources. *2nd IEEE International Conference on Engineering Complex Computer Systems*, 1996.