Global Scheduling and Binding in a Real-Time Embedded Distributed System

Lisa Cingiser DiPippo, Victor Fay-Wolfe, Oleg Uvarov, Angela Uvarova University of Rhode Island
Trudy Morgan U.S. Navy SPAWAR Systems Center San Diego

In a real-time embedded application, it is crucial to ensure that timing constraints be met when processing requests between clients and servers. We are developing middleware services and techniques for global *real-time scheduling* and *real-time binding* within a dynamic real-time distributed system.

Our global Real-Time Binding Service allows a client with real-time constraints (e.g. deadline and/or period) to bind to the server that will most likely be able to service this request, and future requests, within timing constraints. Each application server registers with the Real-Time Binding Service to specify the kinds of services that the application server can provide, including their execution times. When a client makes a request to the Real-Time Binding Service for a specific application service, the Real-Time Binding Service finds all application servers that have indicated an ability to provide the requested service. The Real-Time Binding Service then uses information about the system (tasks already executing, execution times, etc) to determine to which application server the client should be bound to best enforce real-time constraints. The algorithm used by the Real-Time Binding Service uses a best-fit strategy based on the slack times of tasks already scheduled on the node. It chooses the node with the minimum slack time (of all tasks on the node) that is closest to the execution time of the requested task.

Our global Real-Time Scheduling Service dynamically assigns priorities to all tasks that will execute in the real-time distributed system. When a client requests a service from an application server, it can specify quality of service parameters such as its Deadline and Importance. The application server then sends the request to the Real-Time Scheduling Service to get a priority for the execution of the task that represents the service. The Real-Time Scheduling Service performs schedulability analysis to determine if the requested task can be scheduled. If so, the Real-Time Scheduling Service assigns a priority to the execution of the task based on the earliest-deadline-first policy. If Real-Time Scheduling Service determines that system is overloaded when the new task in included, then the Real-Time Scheduling Service performs a *load shedding algorithm* that chooses a task to shed from the system in order to maintain schedulability of the entire system. The heuristic used to choose the task(s) to shed uses a weighting of Importance and remaining execution time of the tasks as the key factors in the decision. This load shedding technique is similar to admission control which denies new tasks if they don't fit; however, our technique looks at the entire list of scheduled tasks as candidates for shedding so that new, important tasks may be scheduled. Shedding is performed by the Real-Time Scheduling Service lowering the global priority of the shed tasks to lower than all non-shed tasks and notifying these shed tasks through the middleware exception mechanism.

Both our Real-time Binding Service and our Real-Time Scheduling Service have been designed in such a way as to work within a Real-time CORBA system. The Real-Time Binding Service is currently designed as an extension of the existing CORBA Trader Service. However, the binding techniques and algorithms can likely be adopted to work within the proposed CORBA Load Balancing Service as well. Our Real-Time Scheduling Service is consistent with the requirements specified by the Dynamic Real-time CORBA specification.

The workshop presentation will describe the Real-Time Binding Service and the Real-Time Scheduling Service interfaces, algorithms, and implementations.