Introduction

The Real-time (RT) CORBA 1.0 specification addresses the problem of allocating resources to competing real-time schedulable entities using fixed priorities. However, many real-time systems have resource requirements that vary significantly at run-time, so that it is difficult to create a successful schedule using fixed priorities that also achieves reasonable resource utilization. Dynamic scheduling defers at least some resource scheduling decisions until run-time, which allows the system to create successful schedules with higher resource utilization.

Key Design Forces

The following key design forces in the real-time systems domain must be addressed by the standard for Real-time CORBA dynamic scheduling:

1. Real-time systems with hard timing constraints for critical operations must insulate critical operations from the resource demands of non-critical operations.
2. Real-time systems must balance heterogeneous and sometimes competing resource management goals, e.g., maximizing utilization of the CPU or sharing link bandwidth fairly between threads at the same priority.
3. Developers of real-time distributed systems must be able to trade off the relative complexity of different programming models, which can increase or decrease system development time and effort, with the real-time system performance benefits each programming model provides.
4. Developers of real-time distributed systems must be able to control the internal concurrency, priority management, and resource utilization configurations in the ORB itself, in order to provide deterministic end-to-end quality of service to applications.

Relationship to RT CORBA 1.0

We see Dynamic Scheduling as a necessary and appropriate extension to the static priority model in the RT CORBA 1.0 standard. Sensor-driven systems with hard real-time processing requirements can benefit greatly from dynamic scheduling capabilities, particularly to make effective use of over-provisioned resources during non-peak loads. Statically allocated priority banding, enforced by preemptive thread priorities, is a valuable feature of these systems, because higher priority activities can be shielded from the resource demands of lower priority operations. Hybrid static-dynamic scheduling techniques\(^1\) offer a way to preserve the off-line scheduling guarantees for critical operations, while increasing overall system utilization. As real-time systems are increasingly interconnected both with each other and with

---

non-real-time systems, the need to support flexible and pluggable scheduling capabilities\(^2\) becomes increasingly important. We believe the Dynamic Scheduling RFP response should offer an extension of the static scheduling capabilities in the RT CORBA 1.0 specification, in which static priorities and mechanisms for their enforcement coexist with additional capabilities for dynamic resource scheduling.

**Necessary Requirements**

We believe the following requirements must be met in the Dynamic Scheduling RFP response to support developers of dynamically scheduled real-time distributed OO systems in meeting the key design forces described above:

- All components of the RT CORBA 1.0 standard must be supported, including the Real-time CORBA manager, the Real-time POA, global and native thread priorities, pluggable priority mappings, transforms, and policies, and mutexes, lanes and thread pools.
- Pluggable dynamic scheduling policies must be supported at different points along the request-response (or in the case of AMI, request-execution) path.
- Flexible interfaces for specifying which attributes are considered by a scheduling policy, propagated with a request, or passed by the application.
- Capabilities in the real-time CORBA dynamic scheduling spec should be based on operational systems or prototypes that are used in practice, so that the standard will reflect rather than attempt to extend the state of the art in dynamic scheduling.

**Desired Capabilities**

We believe the following features would be a beneficial part of the Real-time Dynamic Scheduling standard, but are perhaps not strategic to cast as requirements. These are perhaps best cast as optional portions of the standard, but should not interfere with by other constraints imposed by the standard.

- Reference implementations of well-known scheduling policies, such as RMS, EDF, MLF, and MUF\(^2\).
- Identifiers for scheduling decision points along a request-and-response path, so that real-time aspects such as deadlines can be specified with respect to these individual scheduling decision points.
- Local request handling policies, e.g., the ability to specify whether the passing of a deadline before a request-response path is completed causes the ORB to drop the request or continue processing.
- Exception propagation back along the request-response path when real-time semantics fail, e.g., a deadline is missed.
- A generalization of the optional Scheduling Service from the RT CORBA 1.0 specification, which will provide appropriate policy, configuration and static priority information to each of the (identified) scheduling decision points. The intention of this service is the same as for the RT CORBA 1.0 specification: to provide valid sets of policies, priorities, and configuration information so that correct systems can be built without an undue burden of complexity on the system developers.

---

Necessary Exclusions

We believe it is necessary to make the following exclusions in order to preserve the flexibility of developers to build correct implementations of diverse real-time features within the standard:

- No restrictions on which attributes, e.g., execution time, criticality, etc., are considered by a scheduling policy, forwarded along a request-response path, or supplied by the application.
- No removal of features from the RT CORBA specification or from the general CORBA specification unless a fundamental contradiction with mandatory dynamic scheduling features is discovered.
- No constraints on the semantics of scheduling policies to address such “non-real-time” issues as throughput and fairness, as well as such “real-time” issues as priority preservation and deterministic timing bounds.

Concluding Remarks

Dynamic Scheduling is a necessary and appropriate extension to the static priority-based RT CORBA 1.0 standard. By preserving the static capabilities specified in the existing RT CORBA 1.0 standard, and generalizing those capabilities to include dynamic and hybrid static/dynamic capabilities, Dynamic Scheduling can address the needs of broad application categories.