Applying Meta-Programming Techniques to Reconcile Heterogeneous Scheduling Policies in Open Distributed Real-Time Systems

Angelo Corsaro, Chris Gill, Ron Cytron

Dept. Of Computer Science
Washington University
St. Louis, MO
Challenges

- Emergent QoS Enabled Middleware like RT-CORBA, RT-Java and Distributed RT-Java is Leading to the simplification of distributed system with complex QoS requirements

- Challenges arising in this types of DRE systems involve communicating and enforcing the relative importance of entities that compete for system resources

- Resolving this challenges is essential to build DRE systems that are:
  - Open
  - Dependable
Limitation of Current Middleware

- Most of the QoS Enabled Middleware present today tie together QoS requirements of *competitors* with the *Scheduling Policy* used to enforce/guarantee the requested QoS.

- As a result many QoS Enabled Middleware cannot cope with heterogeneous DRE systems in which
  - Different end-system have different scheduling policies and,
  - *Competitors* originated at one end-system moves to foreign end-systems
An Brief Example

End System (A) Running an EDF Scheduler

End System (B) Running an RM Scheduler

End System (C) Running a Generic Value Based Scheduler

Competitor Created at End-System (A)
Competitor Created at End-System (B)
Competitor Created at End-System (C)
Proposed Solution 1/2

- **QoS requirements should be expressed in terms of Properties**
  - A property describes a feature such as a criticality level, a deadline or a constraint on the jitter.

- **Competitors (which denotes entities that contend for common system resources) expose properties that describe their features or QoS requirements**

- **Schedulers (which grant competitor access to shared resources) are associated with a characteristic function that describes the properties that the scheduler “looks at” when deciding how to order the competitors.**
Schedulers are seen as an “Ordering of Equivalence Classes” over the set of properties in the characteristic set of the scheduler.

For each competitors there is a function “\(\rho\)” that allows access to the properties it exposes.

Adapters, which are mappings from “Compound Property Domain” are used to reconcile the properties exposed by a competitor with the property in the characteristic set of a scheduler.

Special case of Adapter are those that perform “restrictions” or “extension” on the set of properties to be adapted.
The formal solution that we have come up with is better implemented through Meta-Programming technique.

Juno, represent the generic structure of the solution that we propose, which can be cast to fit to different instance of the interoperability problem outlined before.
The DSRT-RTCORBA (see Section 5.4 of the joint-submission) does not address interoperability between heterogeneous dynamic schedulers.

It does not address how to manage the interoperability between Dynamic, and non Dynamic scheduled ORBs.

Juno Meta-Programming Architecture can be applied to a DSRT-RTCORBA ORB to solve the interoperability problems outlined above.
Applying Juno to DSRT-RTCORBA 2/2

- In this mapping, Distributable Thread represent Juno’s competitors
- Juno’s Meta-Object are in part placed in the ORB-Core, and in part associated with the Stub

- Each time a distributable thread transition from one orb to another, reconciliation might need to occur
- Properties Reconciliation can take place either in the client ORB or in the Server ORB
- Adapters take care of performing the needed conversion.
The client ORB obtain the Meta-Properties that describe the server ORB scheduler into Tagged Components embedded in and IOR.

The adapter used to perform the conversion is maintained into the Stub.

Whenever a call is performed the client ORB adapts the properties of the distributable thread to the target ORB scheduler characteristic set.
Server Side Properties Reconciliation

- The Server ORB obtains the Meta-Properties that describe the server ORB scheduler inside the GIOP request, or at connection time.

- The Meta-Scheduler uses a MOP to perform the right property adaptation.
The two ORB exchanging a *Distributable Thread* use respectively EDF and LLF scheduling policy.

In the case EDF -> LFF an extension adapter can be used to reconcile properties.

In the case LLF -> EDF a restriction adapter can be used to reconcile properties.
The two ORB exchanging a *Distributable Thread* use respectively EDF and MAU scheduling policy.

- In the case EDF -> MAU an extension adapter can be used to reconcile properties.
- In the case MAU -> EDF a restriction adapter can be used to reconcile properties.
The two ORB exchanging a *Distributable Thread* use respectively LLF and MAU scheduling policy.

In the case LLF -> MAU an extension/restriction adapter can be used to reconcile properties.

In the case MAU -> LLF a extension/restriction adapter can be used to reconcile properties.
The key concept emphasized by our solution is that Properties Belong to Competitors.

Properties should clearly separated by the scheduling policy if interoperability wants to be achieved.

Meta-Programming technique provides the perfect match for implementing our solution.

Juno’s Meta-Programming architectures provide a general solution to the problem of interoperability which can be applied to many different systems.