RMI-HRT:
Remote Method Invocation for Hard Real Time Systems

D. Tejera, A. Alonso, M.A de Miguel
Universidad Politécnica de Madrid

JTRES 2007, Vienna
Introduction

- Provide support for the development of Distributed Hard Real-Time Systems with Java
  - There is an industrial need for such support

RTSJ (Real-Time Java Specification):

- Defines extensions for RTS
- No support for distribution (JSR-50)
- No support for hard real-time systems (JSR-302)

RMI-HRT: Oriented to Hard Real-Time Systems

- Available RMI implementations are not suitable
HRTJ: Hard Real-Time Profile

- Reliable and predictable Java concurrent model

**Main features:**

- Schedulable objects: periodic or sporadic with MIT
- Only no-heap RTSJ objects
- Application executes in two phases:
  - Initialization: load and creation of required elements
  - Mission: execution of business code
- Limited memory model to achieve predictability:
  - No object creation on immortal during mission phase
  - Each SO has its own scoped memory
- Defined in HIJA project and one basis of JSR-302
**RMI-HRT**

- Compliant with HRT profile
- Time and memory predictability
  - Off-line schedulability and memory analysis
- RMI model has been simplified
  - System configuration is known in advance
  - Resource needs, release parameters, activation patterns, invocation patterns, etc.
Two execution phases: Initialization and Mission
Configuration parameters are modeled as classes,
Static creation of threads, connections, etc.
References:
  - Each one is associated with one connection
  - Characterized by a set of real-time parameters
New serialization approach
Memory usage:
  - Scope memory for temporal objects
  - Immortal memory for initialization and keeping state
1. Initializer: Creation of basic objects
2. Connection creation and parameter passing
3. Handler creation and associated to a connection and remote object
Predictable Serialization

- Relies on the static nature of HRT applications: a-priori knowledge of communicating objects
- RMI compiler performs preliminary activities to simplify run-time serialization
  - Calculates the worst-case objects size
  - Generates classes for serialization operations
- At run-time:
  - Internal buffers are created at initialization phase
  - No need for creating objects during mission phase
  - The knowledge of worst-case streams size is used for a precise end-to-end response time calculation
Response Time Analysis

- System composed by a set of transactions
- Transactions composed by a set of actions
- An action is a portion of code in a thread or a message
- Actions can only be activated by an event which can only activate an action

Linear action

\[ T_{i,j-i} = T_i = T_{i,j+1} \]

Diagram:

1. CPU-1
   - Action: \( a_1 \)
   - Event: \( e_1 \)

2. Net
   - Action: \( a_2 \)
   - Events: \( e_{1,2}, e_{2,3} \)

3. CPU-2
   - Action: \( a_3 \)
Memory Usage

Client

RMI-HRT Core

Serialization Classes

Net Module

IM 1

Hrt-Stub

IM 2

Scoped Memory

ReturnValue

Server

RMI-HRT Core

Net Module

Handler

IM 3

Hrt-Skel

IM 4

Scoped Memory

Parameters

Temporary Objects

Remote Object

IM 5

IM = Immortal memory  , Bx = Buffer x
Industrial Assessment

Integrated Modular Avionics (IMA) module

- Guidance manager
- Autopilot manager
- A/C simulator
- Reprediction manager
- Flight plan manager
- Database manager

PC

HRT RMI
UDP RMI

© DIT/UPM
Conclusions

- RMI-HRT allows development of DHRTS
- Compliant with HRTJ profile
- Its design and implementation allows for memory and time response predictability

Future work:
- Further experimentation
- Precise characterization of memory usage and overhead in an industrial platform
- Other improvements: serialization, error handling, etc
- Alignment with future standards