UNIVERSIDAD DE CANTABRIA

Managing Transactions in Flexible Distributed Real-Time Systems

Daniel Sangorrín (dsl@ertl.jp)
Michael González Harbour (mgh@unican.es)
Héctor Pérez Tijero (perezh@unican.es)
J. Javier Gutiérrez (gutierjj@unican.es)

Computers and Real-Time Group, University of Cantabria

Granted by the European Commission (**IFP6/2005/IST/5-034026, FRESCOR**) and by the Spanish Ministry of Science and Technology (**TIN2008-06766-C03-03, RT-MODEL**)

15th International Conference on Reliable Software Technologies
Ada-Europe 2010
14-18 June 2010, Valencia, Spain

Background: FRESCOR

UNIVERSIDAD DE CANTABRIA

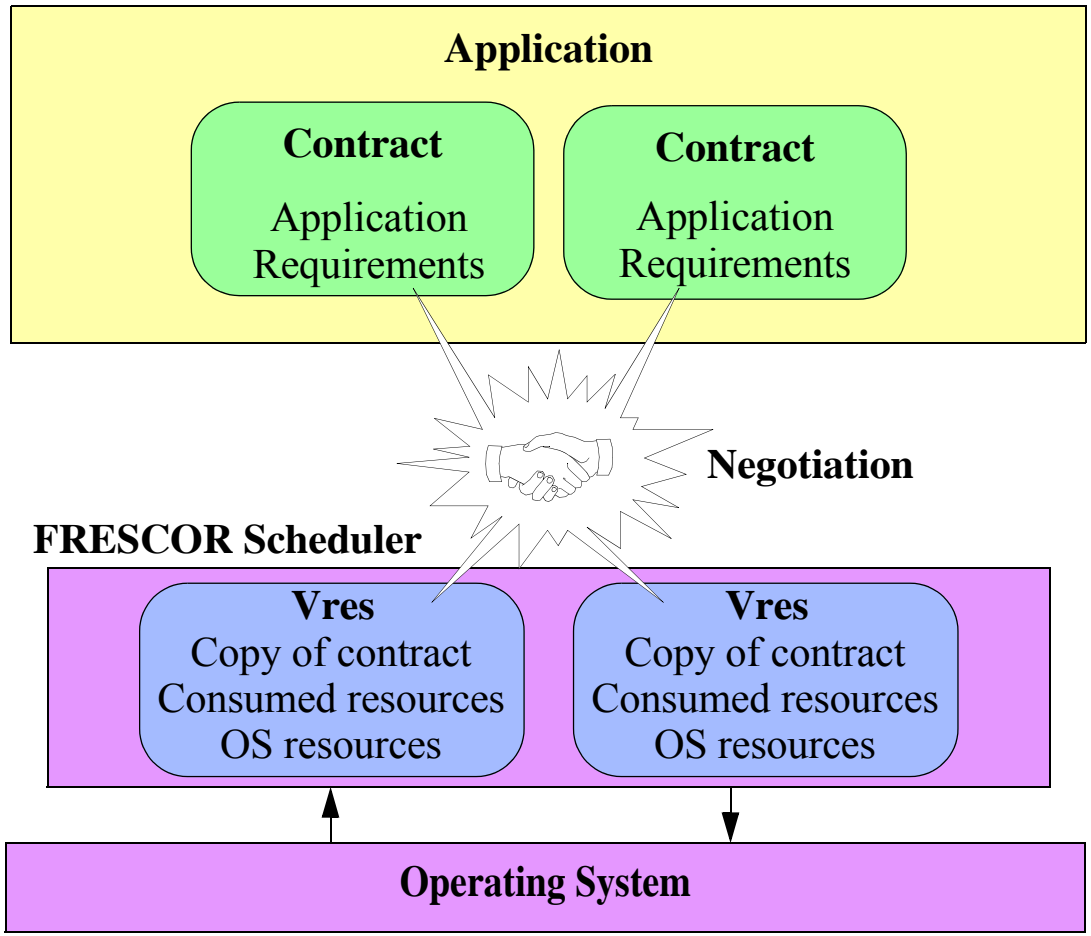
Framework for Real-time Embedded Systems based on COntRacts

- Application developer
  - *Which are my requirements?*
- System
  - *Can those requirements be guaranteed?*
  - Integrates Real-Time theory in the system
    - Built-in scheduling techniques
    - Handle overload in a safe way
  - Manages a variety of schedulable resources
    - Focus on CPUs and networks
  - Distributes spare capacity to maximize quality

Contract-based scheduling

- Contracts represent a certain resource reservation

Background: FRESCOR (cont'd)



Contract

Mapping of application requirements for a given resource

Local admission test

Checks for available resources

Keeps guarantees on previous contracts negotiated by other apps

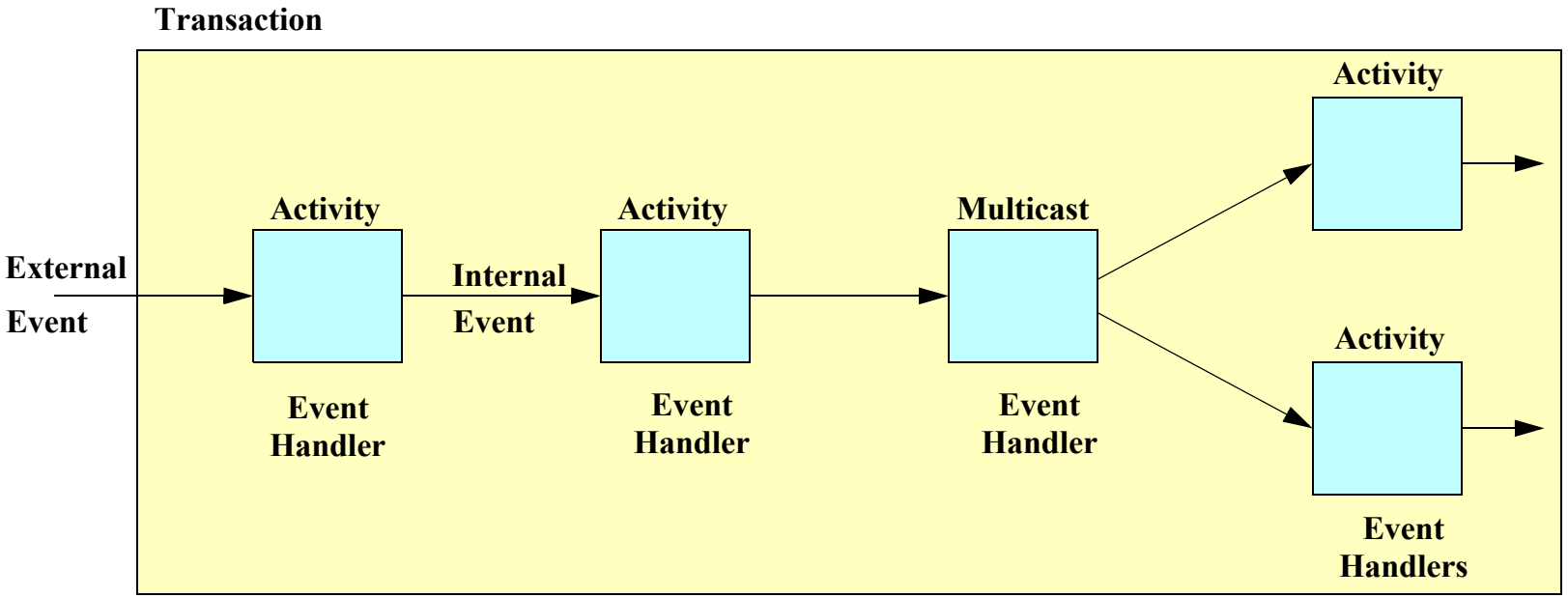
Virtual resource

Run-time representation of a contract

Background: Real-Time Situation

Event-driven transactional model:

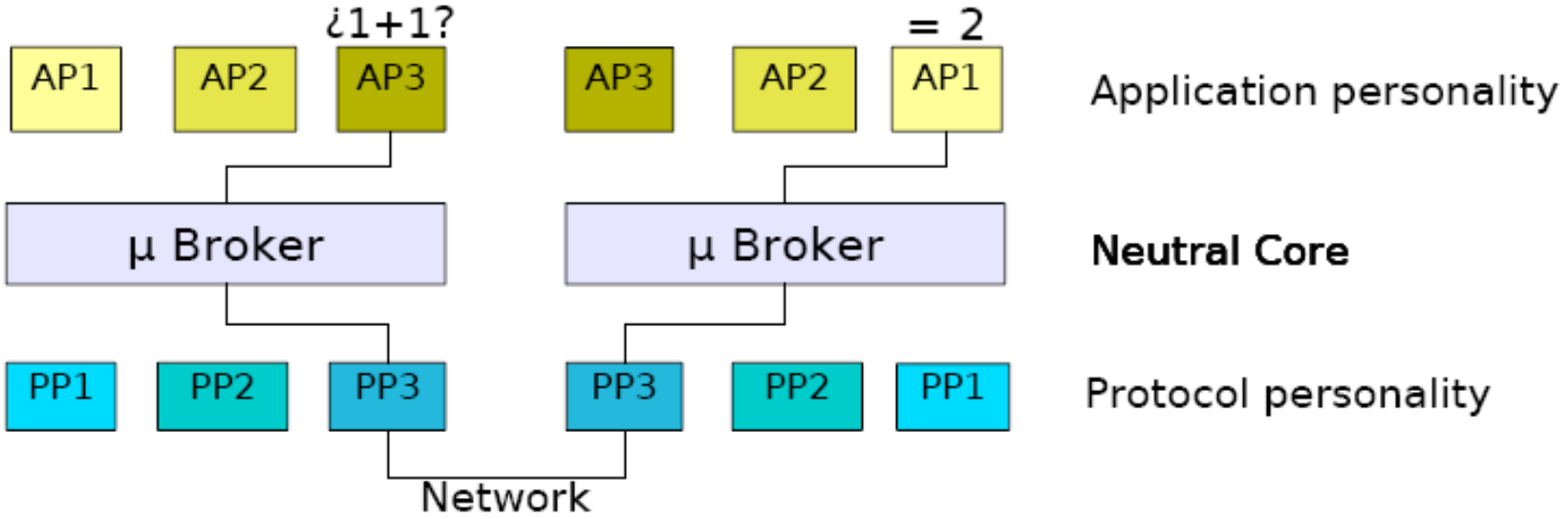
- Transaction = *end-to-end flow* in the OMG MARTE specification



Activities: code to execute by tasks in the processors or messages sent through the networks

Background: Distribution Middleware

PolyORB: The interoperable middleware by AdaCore that enables the communication between different distribution models.



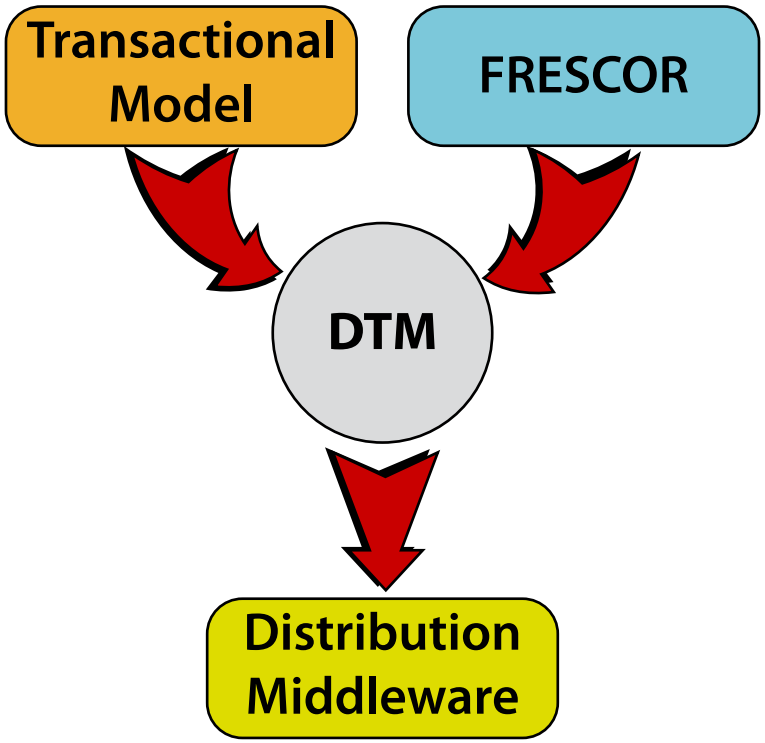
Previous work adapted this middleware to support:

- **Real-Time OS: *MaRTE OS***
- **Real-Time network protocols: *RTEP* and *FRSH-FNA***
- **Different scheduling policies: *FP*, *EDF* and *Flexible scheduling***

Objectives

Create a mechanism to extend the negotiation capabilities of FRESCOR

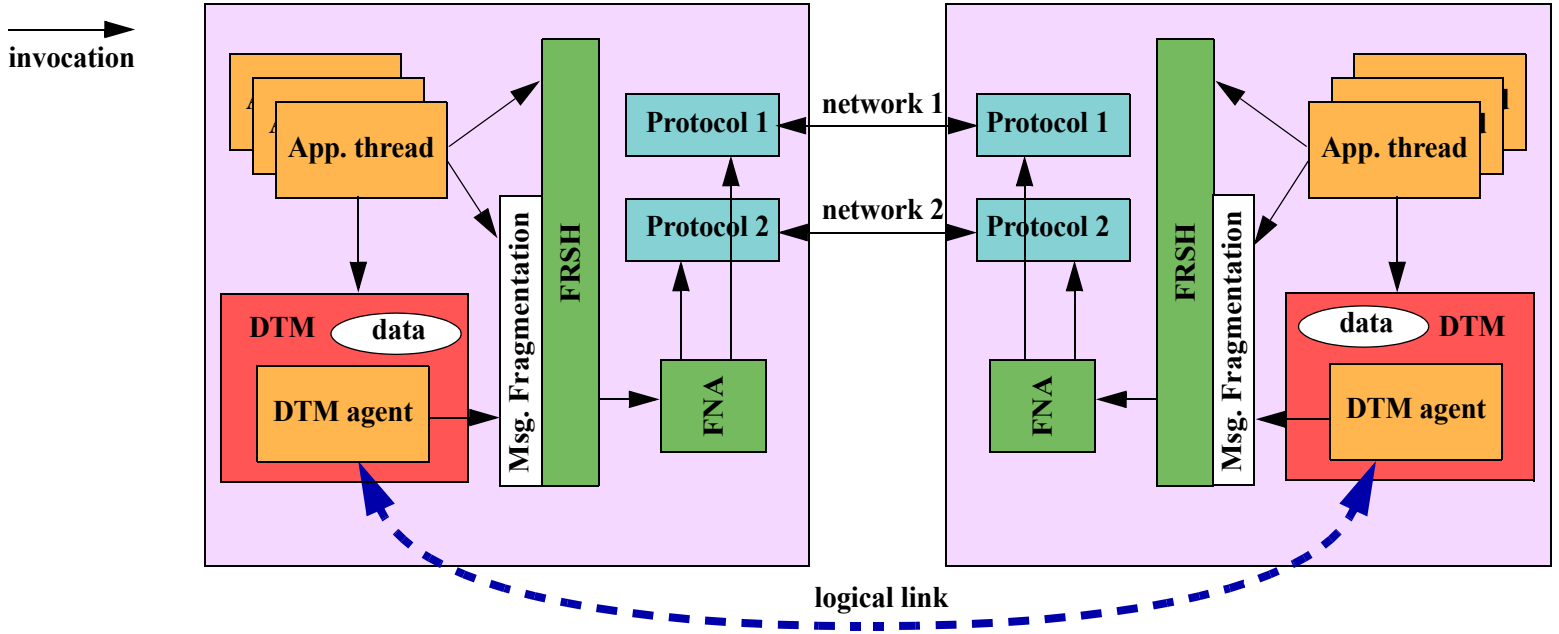
- *Distributed Transaction Manager* (DTM)
  - Negotiation of distributed transactions
    - Remote negotiation of contracts
    - Management of the results



Integration of this mechanism within a distribution middleware

- Increasing portability
- Hidding distribution details
  - Through the use of distribution standards

Distributed Transaction Manager



Distributed application requiring:

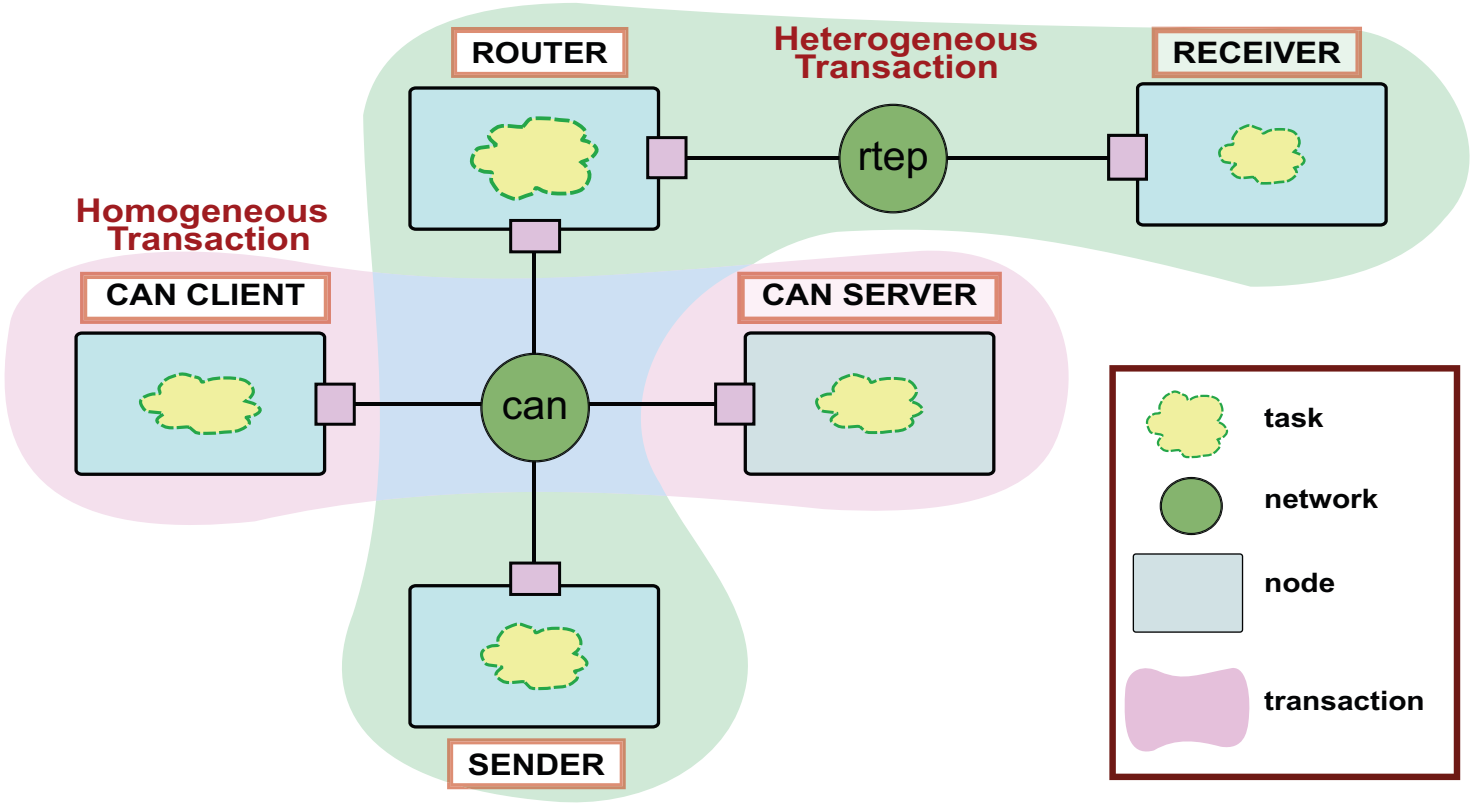
- One DTM agent per node
  - Management of the negotiation messages
- Data structures for the negotiation process
- Fragmentation layer

Distributed Transaction Manager (cont'd)

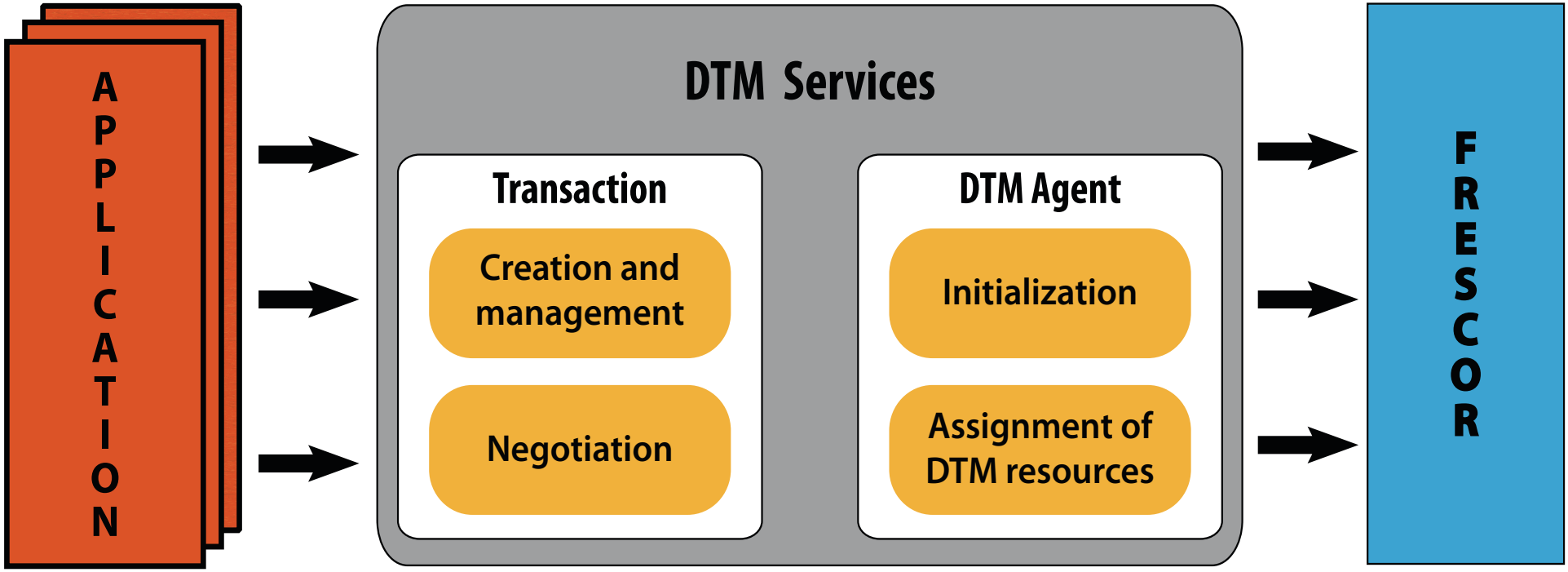
UNIVERSIDAD DE CANTABRIA

DTM also requires:

- Bidirectional communication with a *Routing Service*



DTM Services



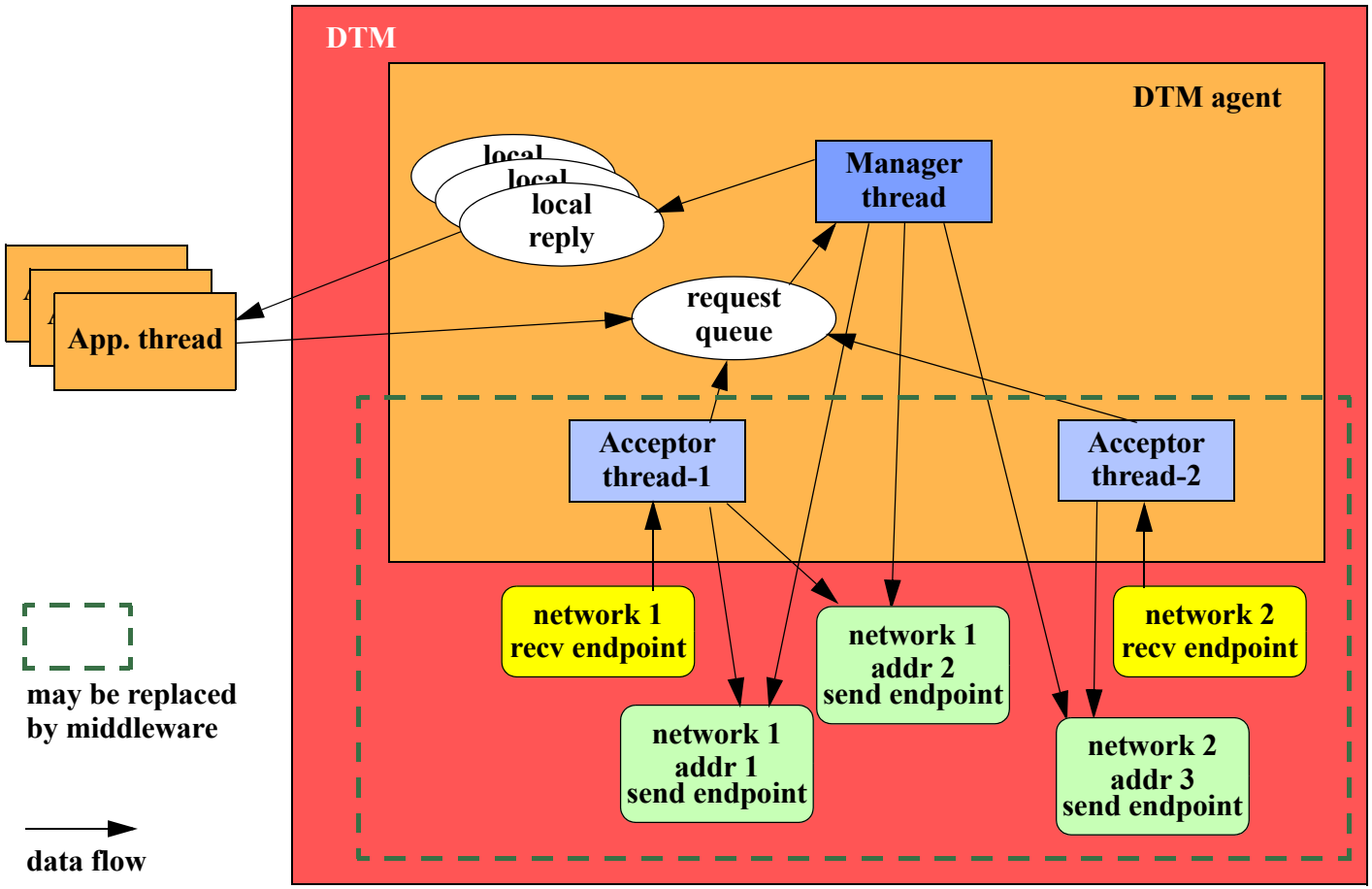
DTM Implementation

UNIVERSIDAD DE CANTABRIA

Development of two different implementations:

1. Stand-alone
   - Direct use of FRESCOR services
   - Communications through message passing mechanism
2. Integrated within middleware
   - Full support for the transactional model

Stand-alone architecture



Multithreaded

- **DTM Manager**
  - -Management and negotiation of transactions
- **Acceptor threads**
  - -Checking & Handling
  - -Routing messages

Endpoints

- **Send & Receive messages**

DTM Implementation

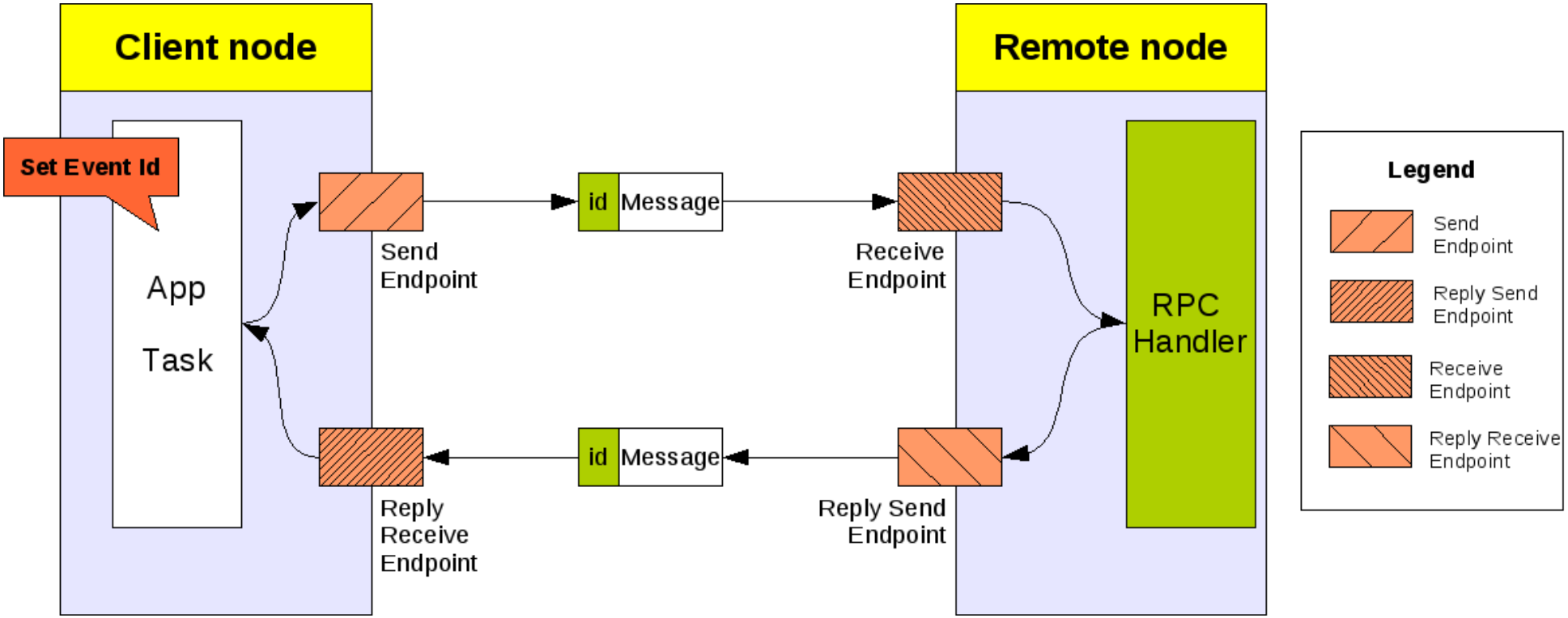
UNIVERSIDAD DE CANTABRIA

Development of two different implementations:

1. Stand-alone
   - Direct use of FRESCOR services
   - Communications through message passing mechanism
2. Integrated within middleware
   - Full support for the transactional model

Transactional model support in MW

- Two schedulable entities
  - *RPC Handlers* for the processing nodes
  - *Endpoints* for the networks



DTM: Middleware Integration

UNIVERSIDAD DE CANTABRIA

Benefits of integrating the DTM with distribution middleware:

- **Simplicity of the DTM application itself**
  - Avoid unnecessary complexity in the management of communication-related data
- **Easing the initialization process**
  - Discovery of DTM Agents performed internally
- **Replacement of the Acceptor threads**
  - Same services provided by RPC Handler tasks
- **Fragmentation service**
  - Most distribution standards implicitly integrate it
- **Full support for the transactional model**
  - Internal management of all transaction-related details

DTM support for the transactional model

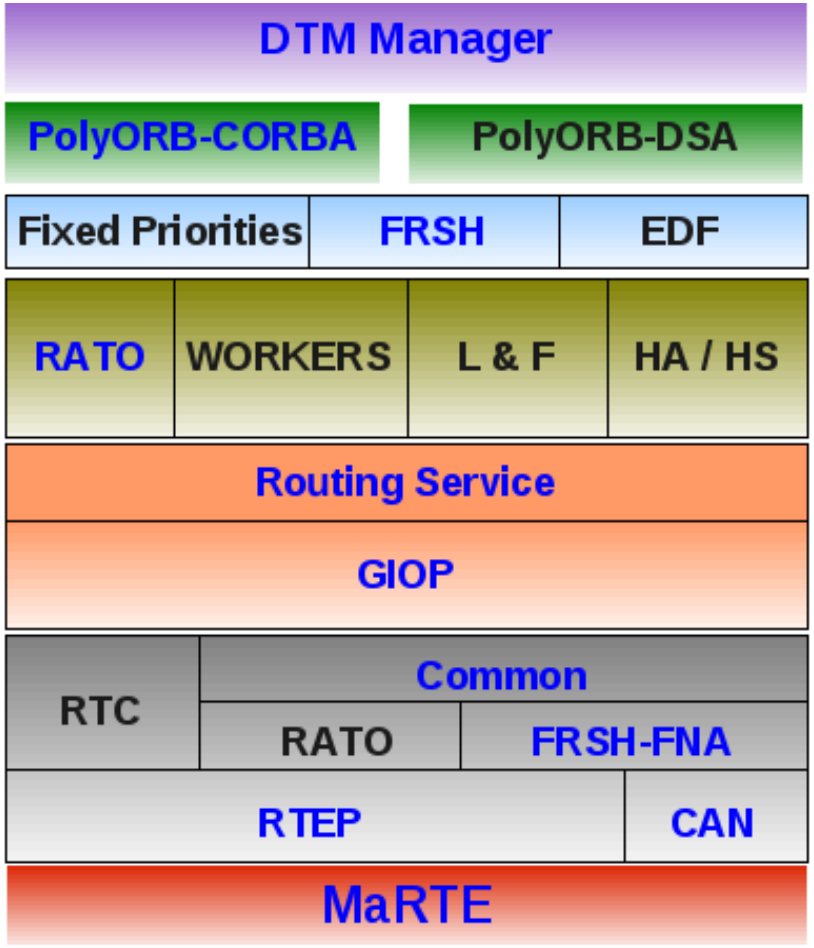
UNIVERSIDAD DE CANTABRIA

Automatic deployment of the new accepted transaction

- Creation of handlers and endpoints to enable the communication
- Extension of the DTM application with:
  - Management of the all transaction details
    - Contract-related parameters
    - Flow associated to the new transaction
  - Choice of unused Event_Ids
    - Decentralized architecture increases the complexity of this operation
  - Choice of unused ports
    - To receive messages from the new transaction
    - Information to share with the nodes involved in the transaction

Implementing the DTM over PolyORB

- The DTM application is distributed using CORBA standard
- DTM services are described through IDL
  - Definition of asynchronous remote operations and DTM messages
- Routing Service
  - Performed internally within GIOP protocol
- Framentation layer
- Initialization Service
  - Through CORBA Naming Service



Evaluation of DTM performance

UNIVERSIDAD DE CANTABRIA

Evaluation of the time required to negotiate a distributed transaction using middleware:

- Hardware platform
  - 2 to 4 embedded nodes
  - 800 Mhz
  - 100 Mbps Ethernet
- Software platform
  - GNAT GPL 2008
  - MaRTE OS v 1.9
  - PolyORB v 2.4 extended
- Negotiation of 10 independent asynchronous transactions
  - one contract per CPU
  - one contract per network link

Num. of Nodes	Max (μs)	Avg (μs)	Min (μs)	Std. deviation
Two nodes	19508	19383	19133	111
Three nodes	25546	25219	24470	330
Four nodes	33255	32713	32565	199

Conclusions

UNIVERSIDAD DE CANTABRIA

The Distributed Transaction Manager (DTM) extends FRESCOR capabilities allowing:

- **Dynamic admission of new application components through:**
  - **Remote negotiation and renegotiation of contracts**
  - **Consistent management of the results**

Two different implementations have been partially developed as a proof of concepts:

- **Stand-alone DTM using FRESCOR services**
  - **Communications through message passing mechanism**
- **Integrated within middleware**
  - **Full support for the transactional or *end-to-end flows* model**

UNIVERSIDAD DE CANTABRIA

Questions?