**H.1 FORMI: An RMI Extension for Adaptive Applications**

1 Motivation

- Distributed object-oriented applications supported by middleware
  - CORBA, .NET-Remoting
  - Java Remote Method Invocation (RMI)
- Rising number of applications require non-functional requirements
  - Soft real-time conditions
  - Fault-tolerance
  - High availability

![Diagram of Streaming Service](attachment:1.png)
1 Motivation

- Java RMI framework provides extension points
  - New call semantics, transport protocols
  - Used to introduce the object group paradigm

- No general adaption of the interaction patterns and the internal structure
  - Solution: fragmented-object approach [Shapiro94]

2 Fragmented-Object Approach

- Extension of the traditional concept of stub-based distributed objects
  - Object with unique identity distributed among different nodes (fragments)
  - Distribution of state and functionality and arbitrary internal communication
  - Fragments offer the whole FO’s interface (distribution transparency)
  - Support for adaptive applications (e.g. state migration, replicas, etc.)
  - Implicit binding (passing reference creates object-specific fragment)
2 Fragmented-Object Approach

- Extension of the traditional concept of stub-based distributed objects
  - Object with unique identity distributed among different nodes (fragments)
  - Distribution of state and functionality and arbitrary internal communication
  - Fragments offer the whole FO’s interface (distribution transparency)
  - Support for adaptive applications (e.g. state migration, replicas, etc.)
  - Implicit binding (passing reference creates object-specific fragment)
3 Java RMI

- Remote Method Invocation in Java
  - Maintaining semantics of the Java object model in a distributed environment

- Semantics for object-passing
  - Primitive values, Java objects, not exported RMI-objects: call-by-value
  - Exported RMI-objects: call-by-reference

- Architecture

```
Client   Server
Stub     Skeleton
Remote Reference Layer
Remote Reference Layer
Transport Layer
```

3 Java RMI

- Remote invocation in Java RMI

```
Node A  Node B
method_xy
```
3 Java RMI

- Remote invocation in Java RMI

![Diagram of Remote Invocation in Java RMI](image-url)
4 Fragmented Objects in Java RMI

- Goal: transparent integration of fragmented objects into Java RMI

- Naive approach: Extending the Remote Reference Layer
  - Well-known approach to integrate own call semantics (e.g., JGroup system)
  - RemoteRef forwards invocations to the local fragment
  - Problem: Local calls treated like remote ones (marshalling uses Java serialization and reflection)

![Diagram of client, stub, RemoteRef, and fragment showing method calls and communication]

- FORMI approach
  - FORMI stub: merge of reference layer and stub layer
  - Marshalling of stubs possible
  - Calls processed faster

![Diagram of client, FORMI-Stub, and fragment showing method calls and communication]
4 Fragmented Objects in Java RMI

- Architecture of local fragments

Fragment Factory

View Manager

Fragment

View

Fragment Interface

Fragment Implementation

Fragment Implementation Factory

- Used as reference to local fragment, inherits RemoteStub
- Forwards calls to implementation
- Dynamic, transparent exchange of implementation at runtime
4 Fragmented Objects in Java RMI

- Architecture of local fragments

## Fragment Factory
- create

## View Manager
- create
- manage

## Fragment
- View
- Fragment Implementation
- Fragment Interface

### Fragment Factory
- Used for creation of new fragments
- Responsible to select the class of the local fragment implementation
- Serialized during marshalling

### Exchange of Implementation Interface for QoS requirements

### Fragment Implementation Factory
4 Fragmented Objects in Java RMI

- Architecture of local fragments

Fragment Factory

Create new FORMI objects

Returns reference

View Manager

create

manage

create

View

create

Fragment Implementation

create

Fragment Implementation Factory

Fragment Interface

create

Detects locally existent fragments in conjunction with the view

Creates the view

5 Conclusion

- Novel approach of integrating the concept of fragmented objects into Java RMI
  - Support for adaptive applications
  - Dynamic distribution of state and functionality
  - Arbitrary internal communication
  - Compatibility with standard RMI clients

H.2 Aufgabe #5

- Fragmented IM

```
java.rmi.Remote extends Fragment
org.aspectix.formi.FormiFragment extends FragmentImpl
```

```
Class
Interface
```
1 Aufgabe #5

- **Fragment Implementierung**
  - public interface Fragment extends Remote {}  
  - public class FragImpl extends FormiFragment implements Fragment {}

- **Fragment Initialisierung und Erzeugung remote Referenz**
  - Erzeuge einen eindeutigen Objekt ID
    
    GUID guid = new GUID();
  
  - Erzeuge eine remote Referenz
    
    Fragment frag = (Fragment) FragmentedObjectFactory.createObject
      (FragImpl.class, DefaultFragImplFactory.class, guid, null, null);
  
  - Veroeffentlichen der Referenz an einem Namensdienst (z.B. RMIRegistry)
    
    Naming.rebind("rmi://localhost/frag", frag);

- **Referenz auf fragmentiertes Objekt**
  - Fragment frag = (Fragment) Naming.lookup("rmi://localhost/frag");

2 Aufgabe #5

**Questions?**