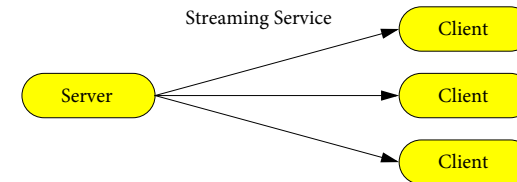


H.1 FORMI: An RMI Extension for Adaptive Applications

- Motivation
- The Fragmented-Object Approach
- Java RMI
- Fragmented Objects in Java RMI
- Conclusions

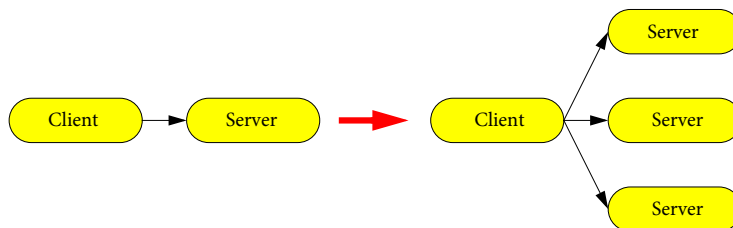
1 Motivation

- 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



1 Motivation

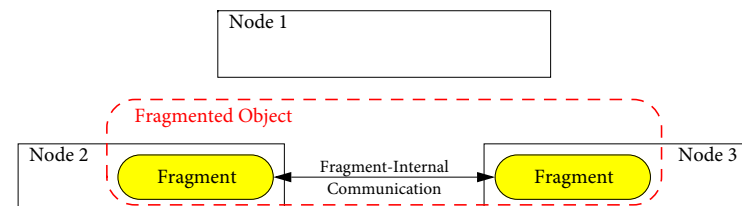
- Java RMI framework provides extension points
 - ◆ New call semantics, transport protocols
 - ◆ Used to introduce the object group paradigm



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

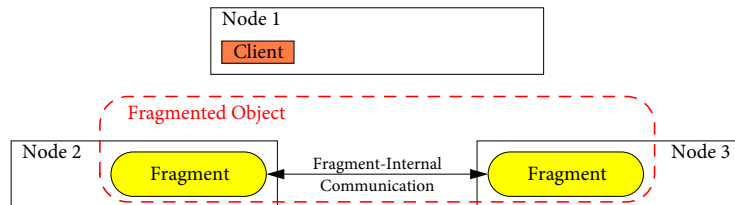
2 Fragmented-Object Approach

- 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)



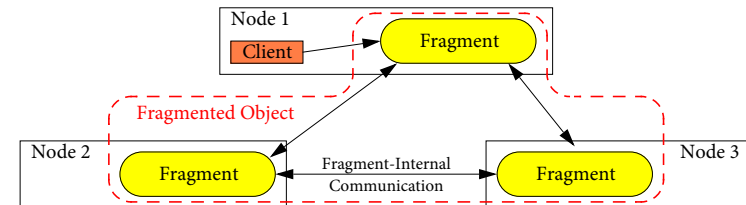
2 Fragmented-Object Approach

- 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)



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)



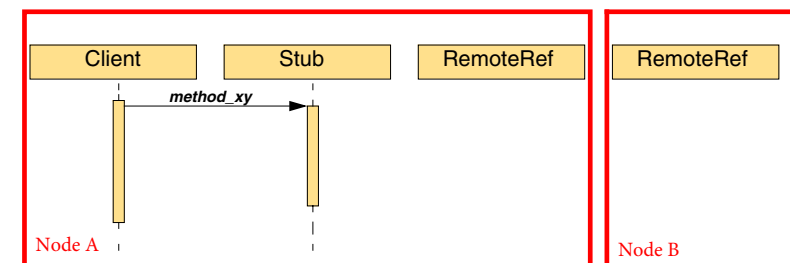
3 Java RMI

- 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



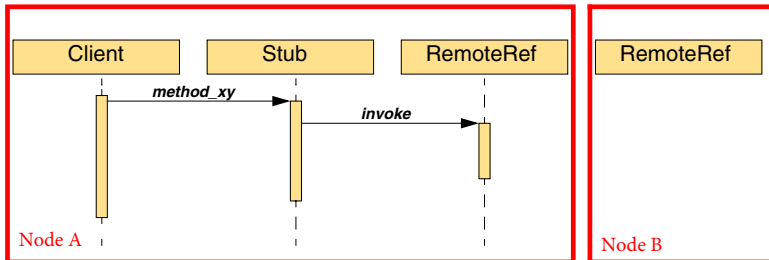
3 Java RMI

- Remote invocation in Java RMI



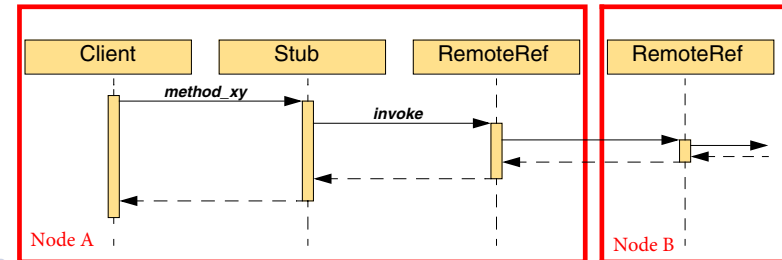
3 Java RMI

Remote invocation in Java RMI



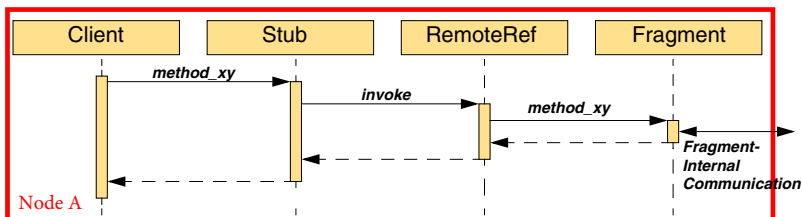
3 Java RMI

Remote invocation in Java RMI



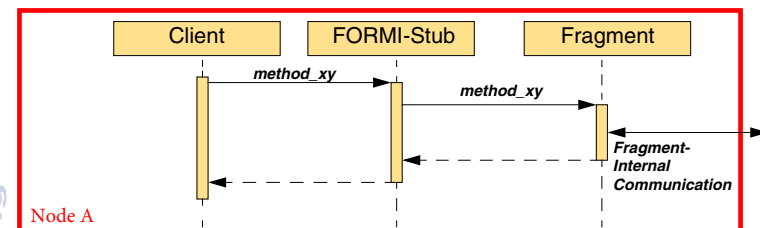
4 Fragmented Objects in Java RMI

- 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)



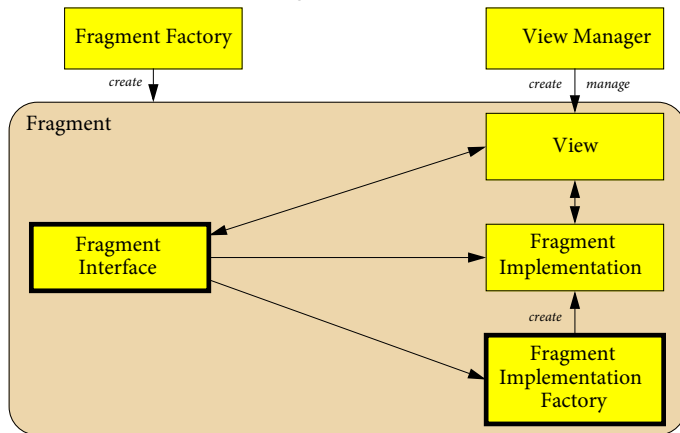
4 Fragmented Objects in Java RMI

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



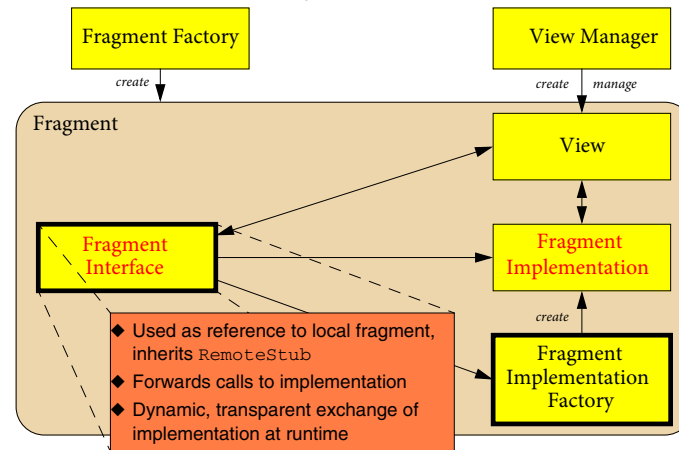
4 Fragmented Objects in Java RMI

Architecture of local fragments



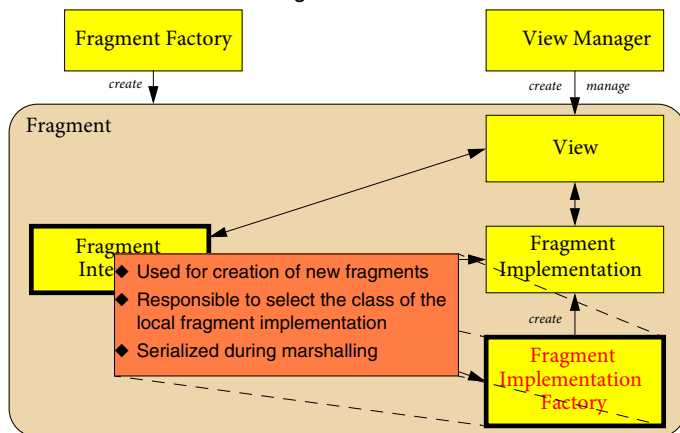
4 Fragmented Objects in Java RMI

Architecture of local fragments



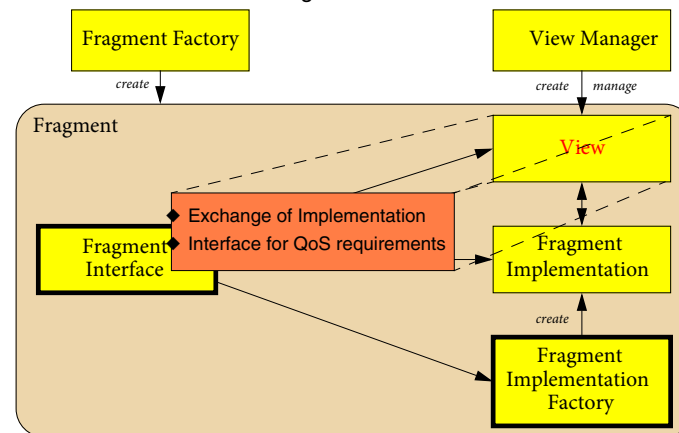
4 Fragmented Objects in Java RMI

Architecture of local fragments



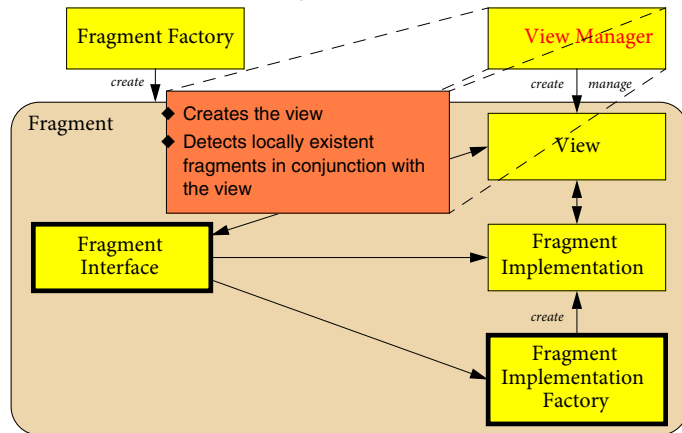
4 Fragmented Objects in Java RMI

Architecture of local fragments



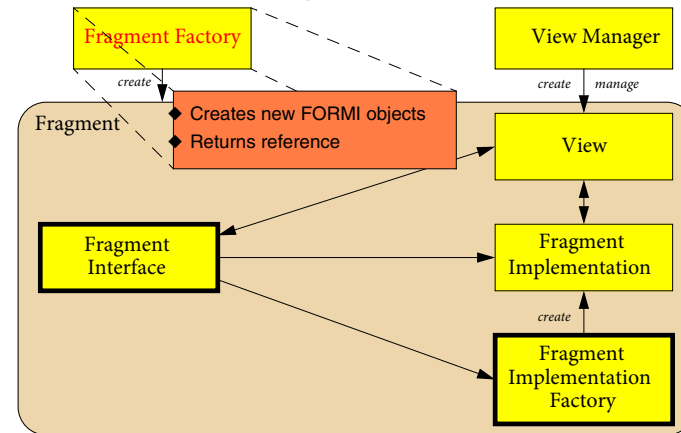
4 Fragmented Objects in Java RMI

Architecture of local fragments



4 Fragmented Objects in Java RMI

Architecture of local fragments

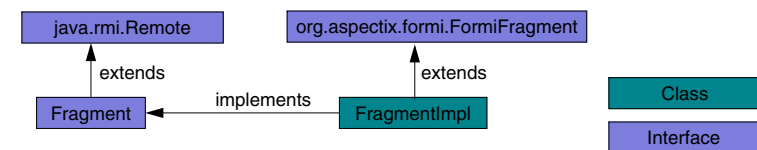


5 Conclusion

- 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



1 Aufgabe #5

H.2 Aufgabe #5

■ Fragment Implementierung

- ◆ `public interface Fragment extends Remote {}`
- ◆ `public class FragImpl extends FormiFragment implements Fragment {}`

■ Fragment Initialisierung und Erzeugung remote Referenz

- ◆ Erzeugen einer eindeutigen Objekt ID
`GUID guid = new GUID();`
- ◆ Erzeugen einer remote Referenz
`Fragment frag = (Fragment) FragmentedObjectFactory.createObject
(FragImpl.class, DefaultFragImplFactory.class, guid, null, null);`
- ◆ Veröffentlichen 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

H.2 Aufgabe #5

Questions?