Towards Generic and Middleware-independent Support for Replicated, Distributed Objects

Jörg Domaschka, Hans P. Reiser, Franz J. Hauck

Distributed Systems Lab,
Faculty of Computer Science

Ulm University
Germany

Aspectix Research Group
http://www.aspectix.org/

20 March 2007

Motivation

Why replication?

- Object-based distributed systems have single point of failure
- Replication provides fault-tolerance

Motivation

Why replication?

- Object-based distributed systems have single point of failure
- Replication provides fault-tolerance

What are the drawbacks?

- Implementing replication support is non-trivial
- High development costs
- Bigger middleware core

Motivation

Why replication?

- Object-based distributed systems have single point of failure
- Replication provides fault-tolerance

What are the drawbacks?

- Implementing replication support is non-trivial
- High development costs
- Bigger middleware core

State of the art

- Many middleware systems provide replication support
- All of them support group communication, replica management
- Barely re-use of existing solutions

Goal

Goal

Provide a single replication framwork for multiple middleware systems!

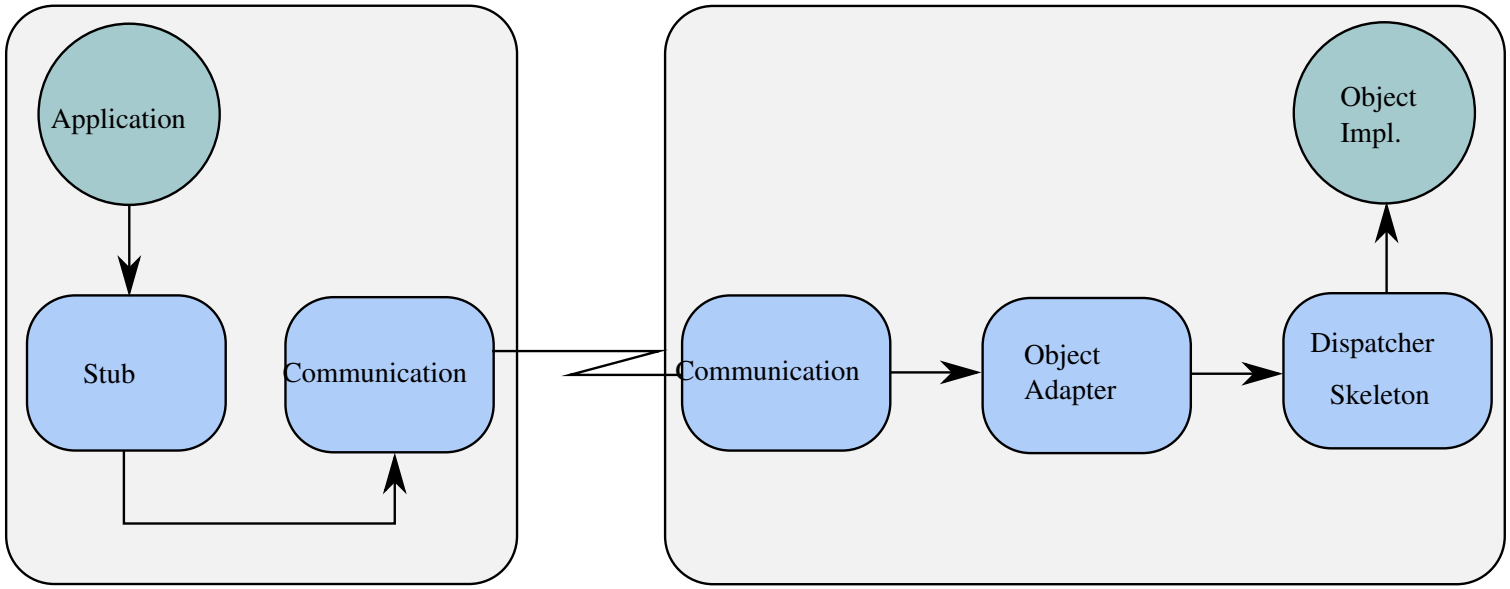
Problems to solve

- What interface does the framework have to provide?
- How to conceptually integrate the framework into the middleware systems?

Outline

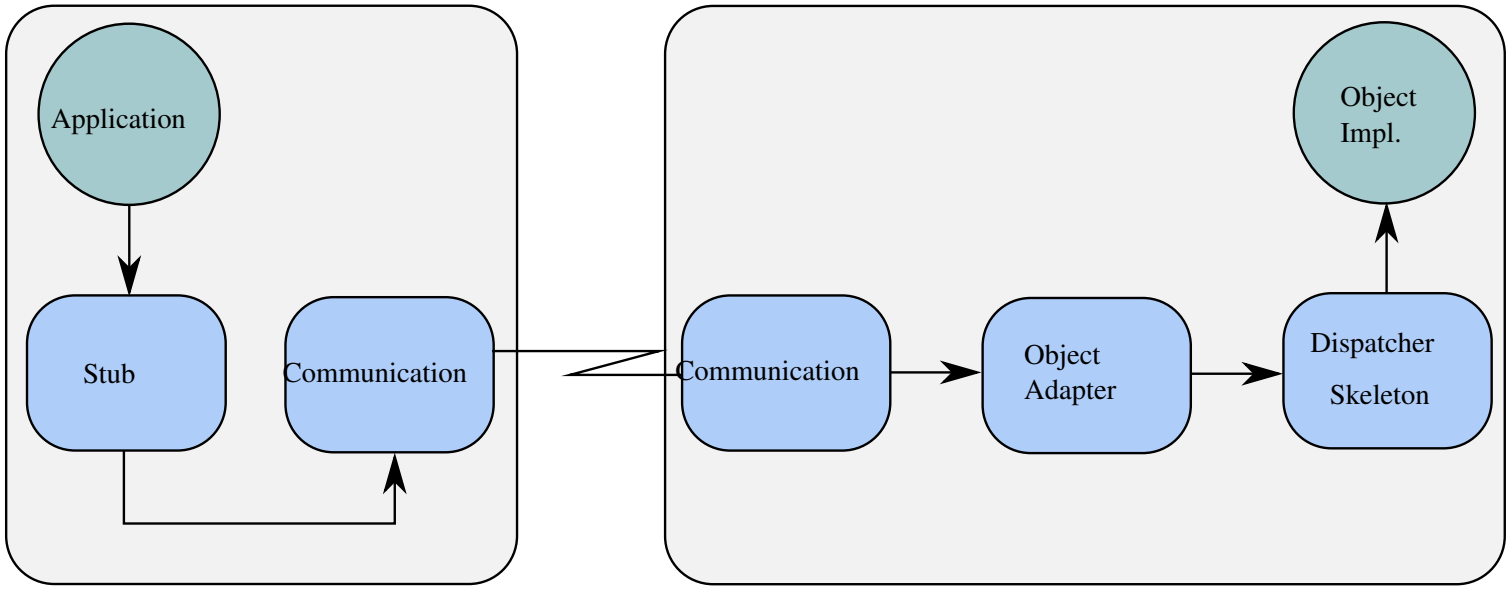
Background

Middleware for Distributed Objects



Background

Middleware for Distributed Objects

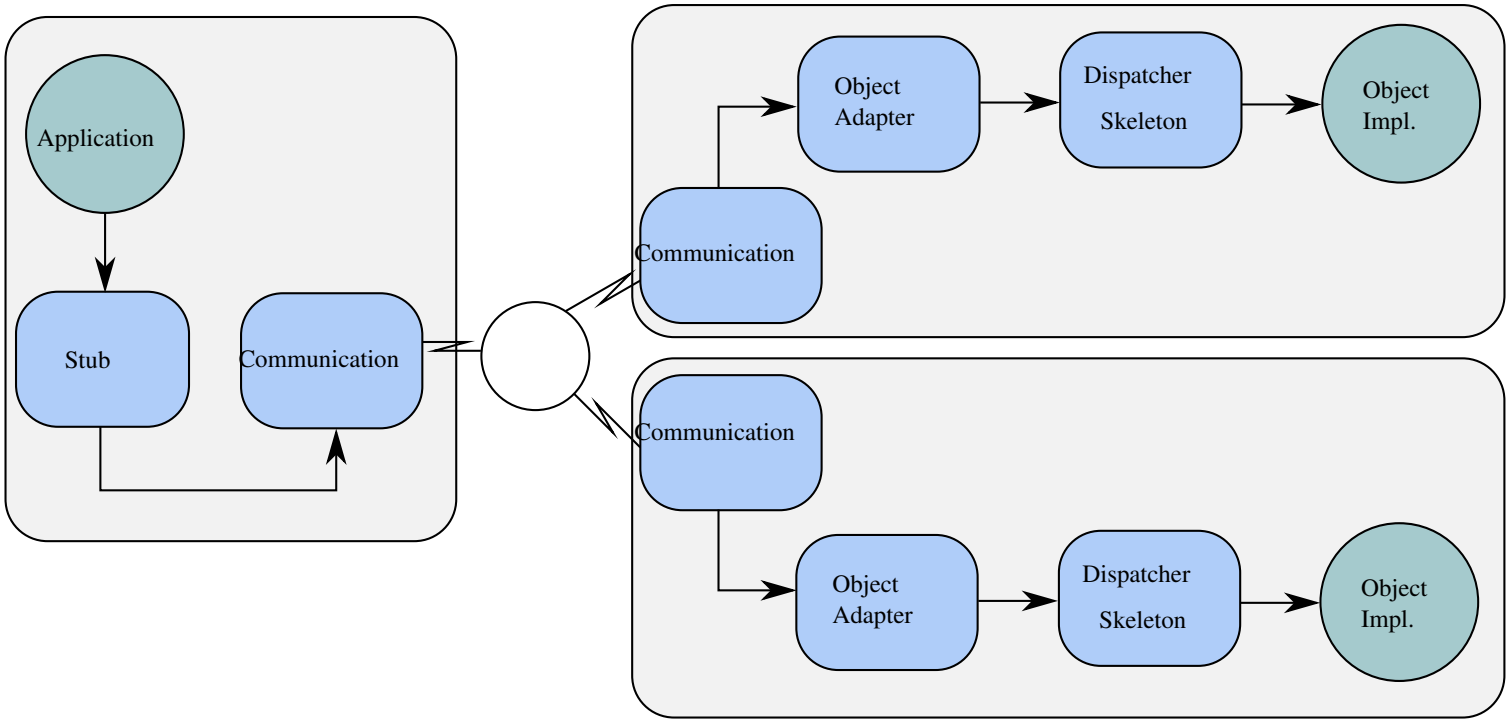


- Client accesses a single remote object
- Client-server interaction scheme
- Request followed by response

$\Rightarrow$ Rather simple communication

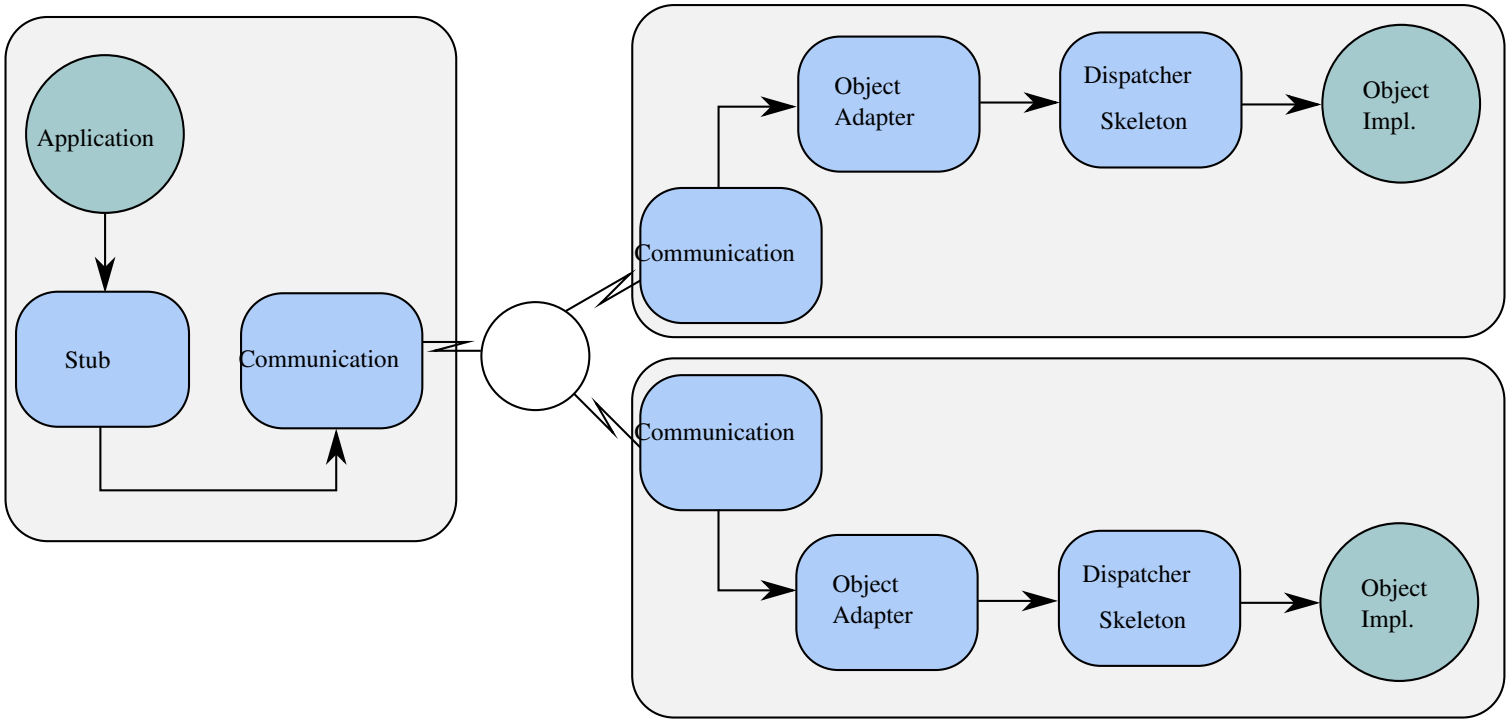
Background

Replication Support in Middleware



Background

Replication Support in Middleware

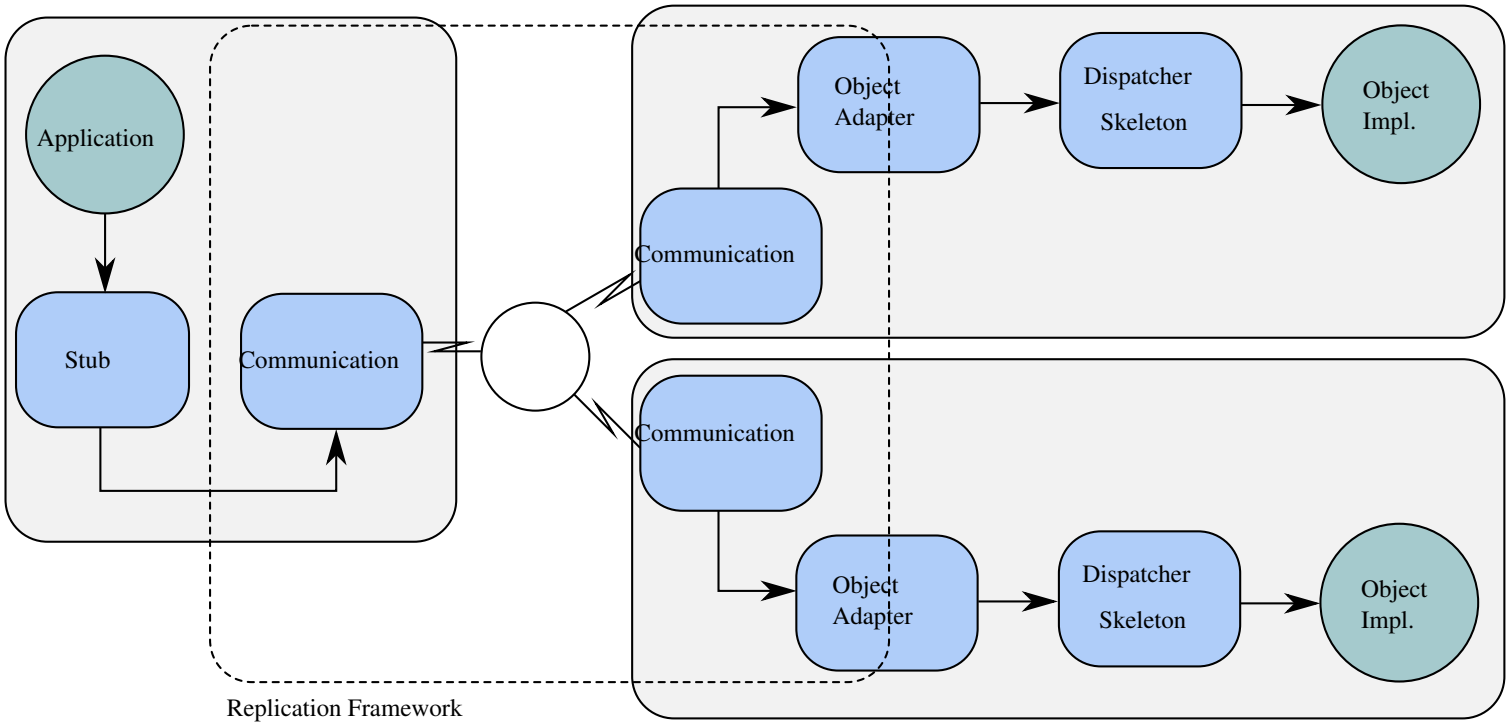


- Client accesses a group of identical remote objects
- Replica group may change
- Determinism by totally ordered multicast

$\Rightarrow$ Complex communication logic

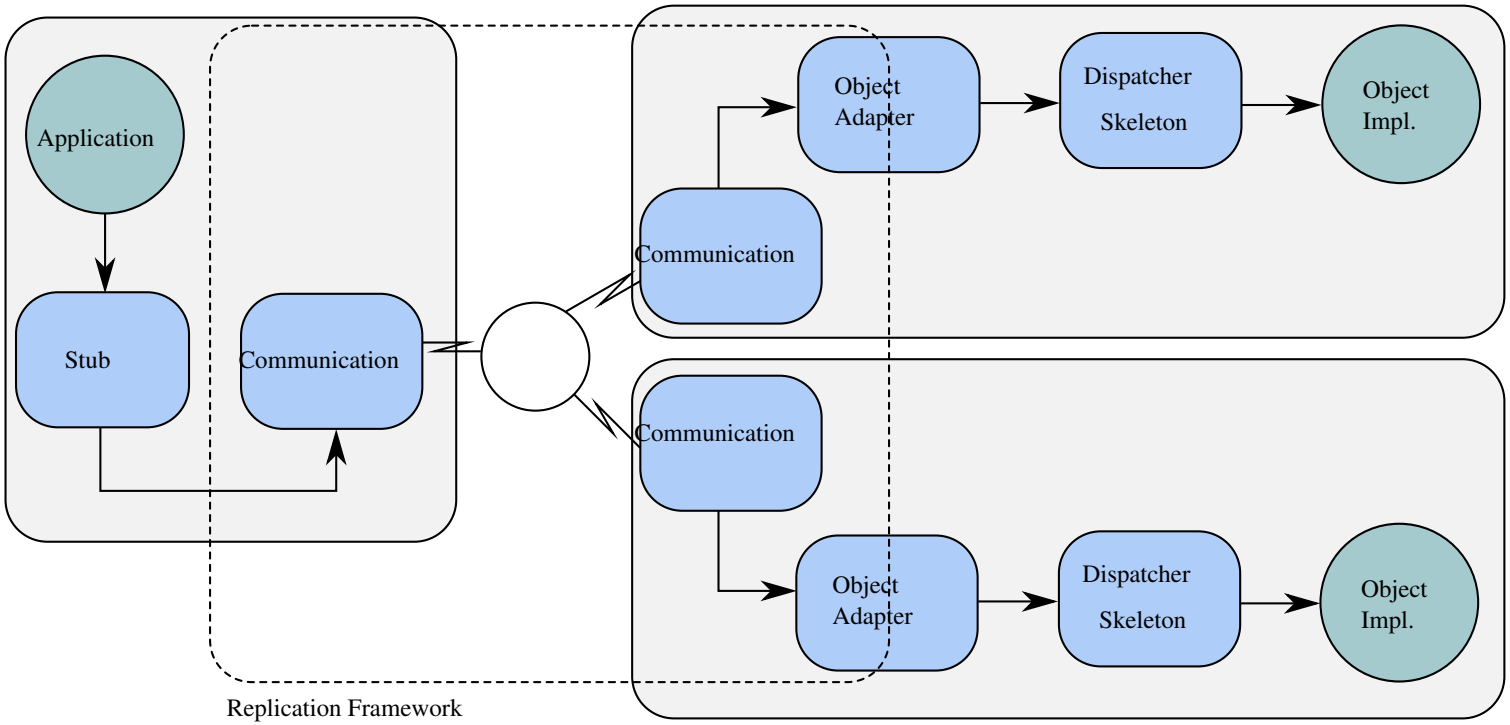
Background

Replication support: Separation of concerns



Background

Replication support: Separation of concerns

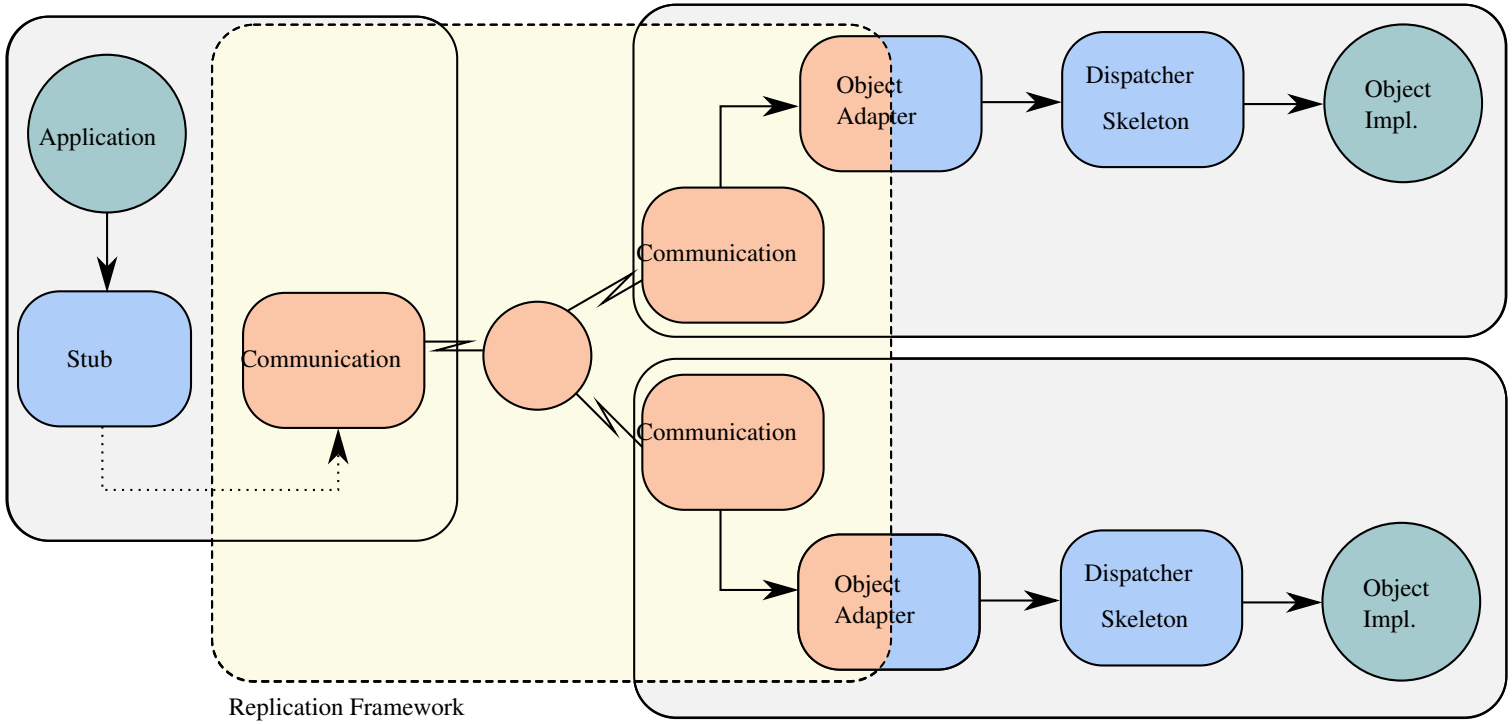


Responsibility of a replication framework

- Ensure determinism and consistency
- Follow changes of the replica group
- Ensure replication transparency

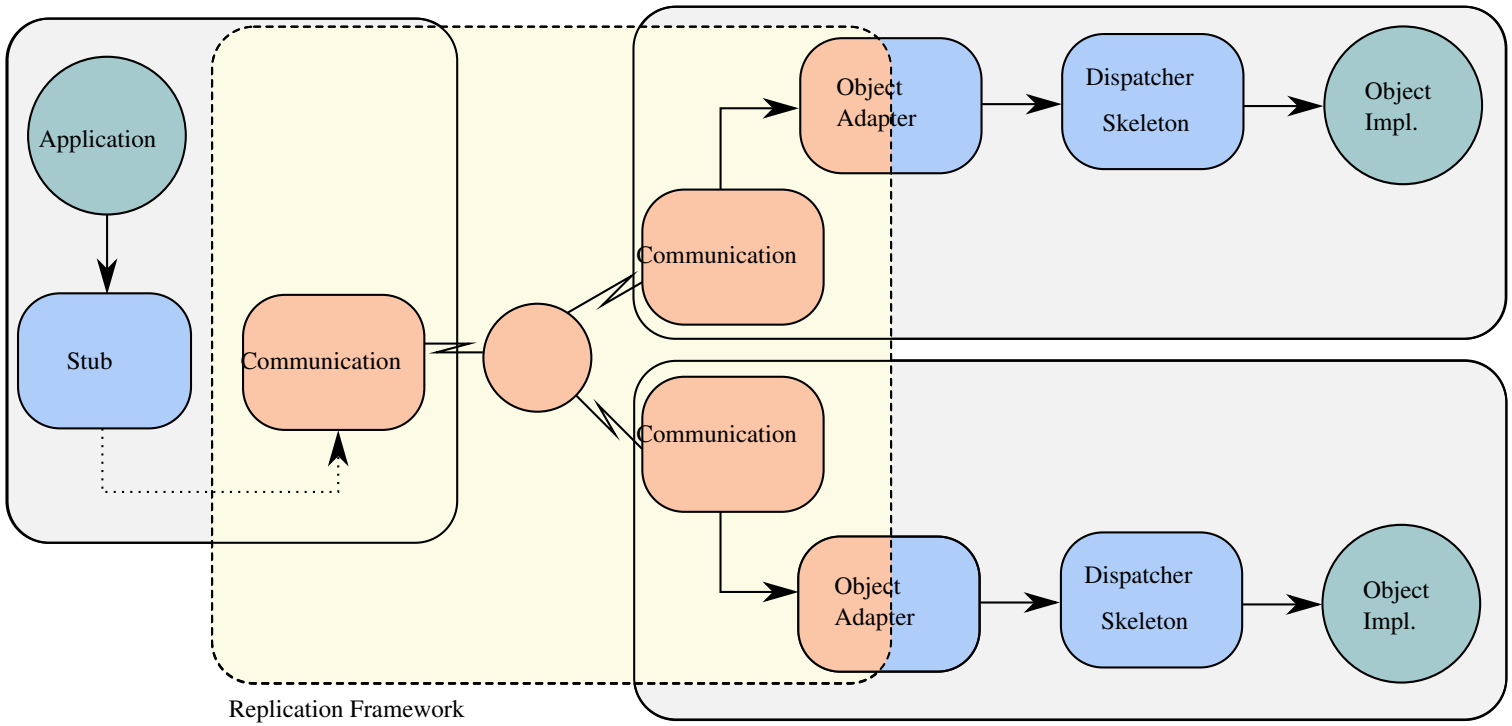
An Architecture for Replication Support

Initial step



An Architecture for Replication Support

Initial step

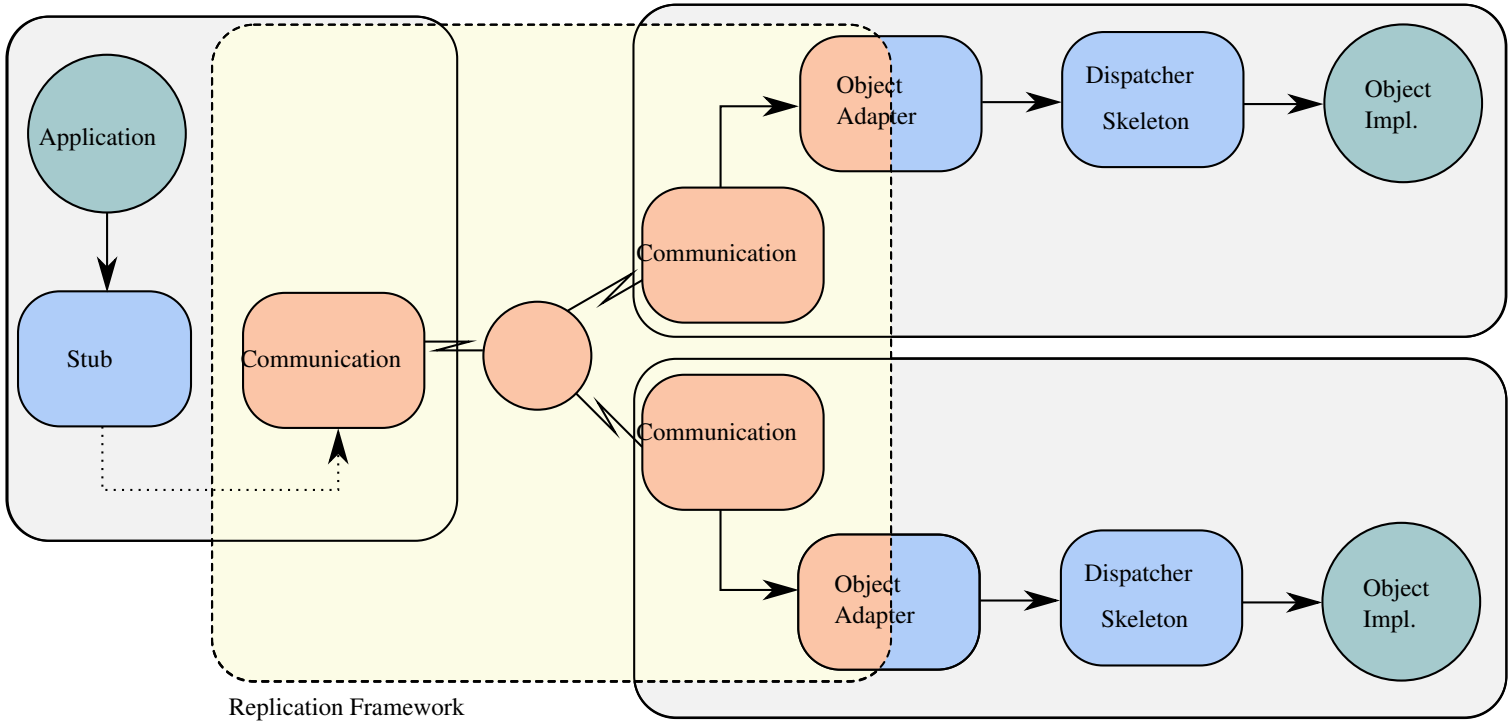


Generic replication framework:

- Transports any data
- Handles data in an opaque way

An Architecture for Replication Support

Initial step



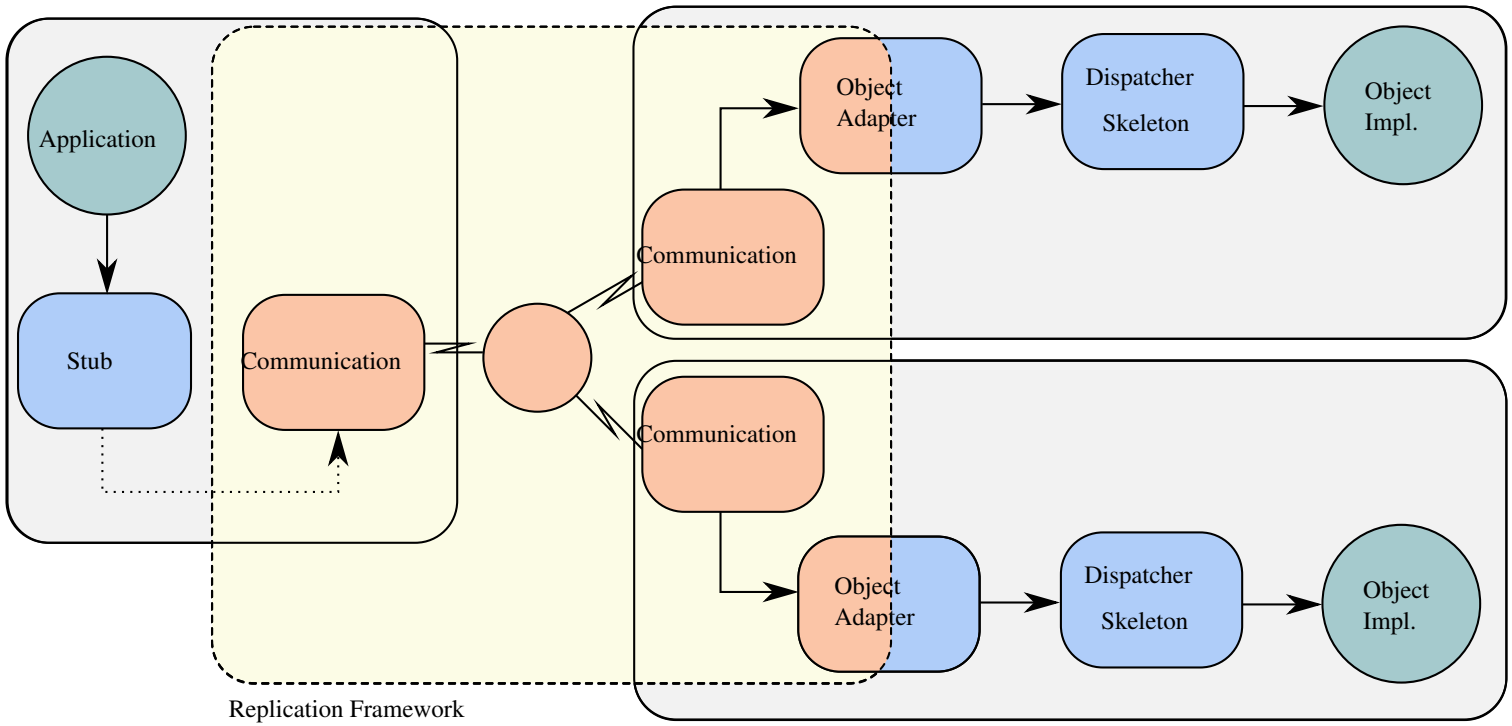
Generic replication framework:

- Transports any data
- Handles data in an opaque way

$\Rightarrow$ Interface supports only `byte[]`

An Architecture for Replication Support

Initial step



Generic replication framework:

- Transports any data
- Handles data in an opaque way

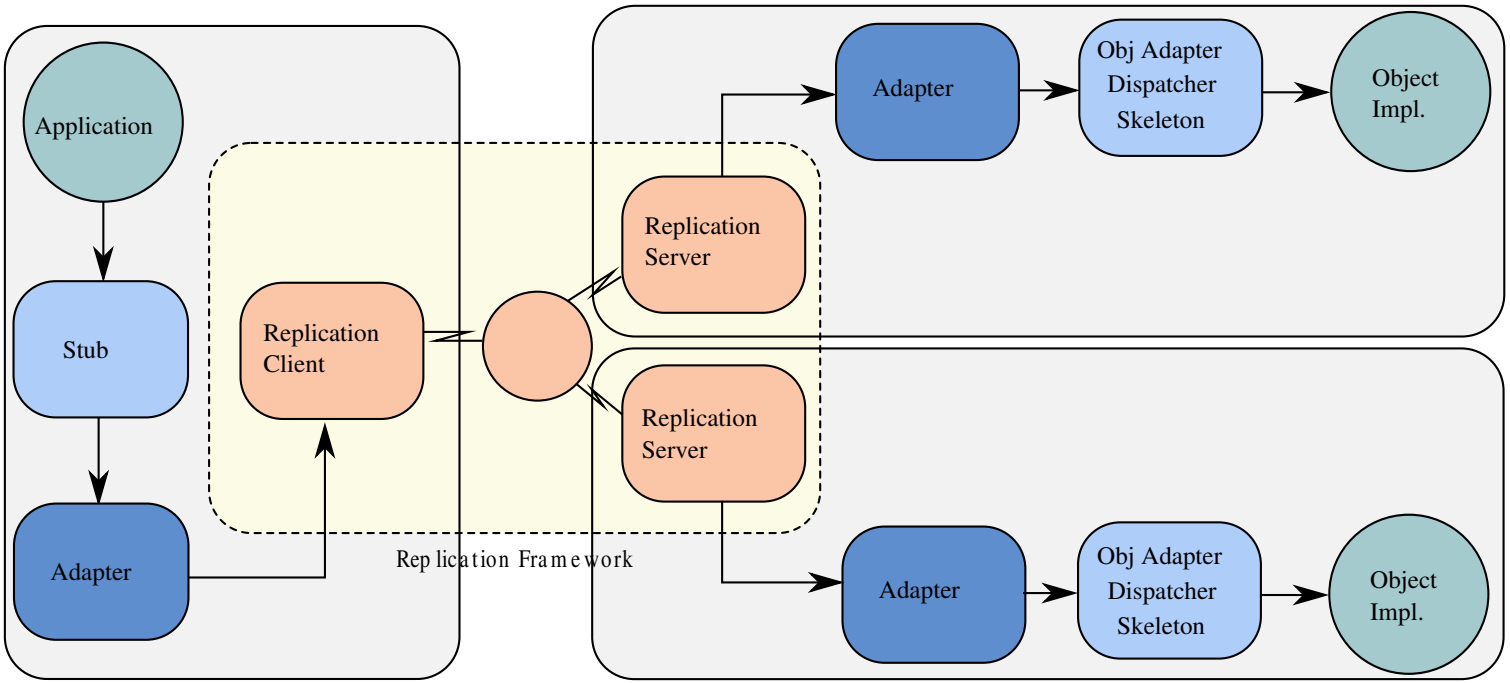
⇒ Interface supports only `byte[]`

Integration into middleware

- Not all stubs support `byte[]`
- Redirecting calls to replication framework necessary

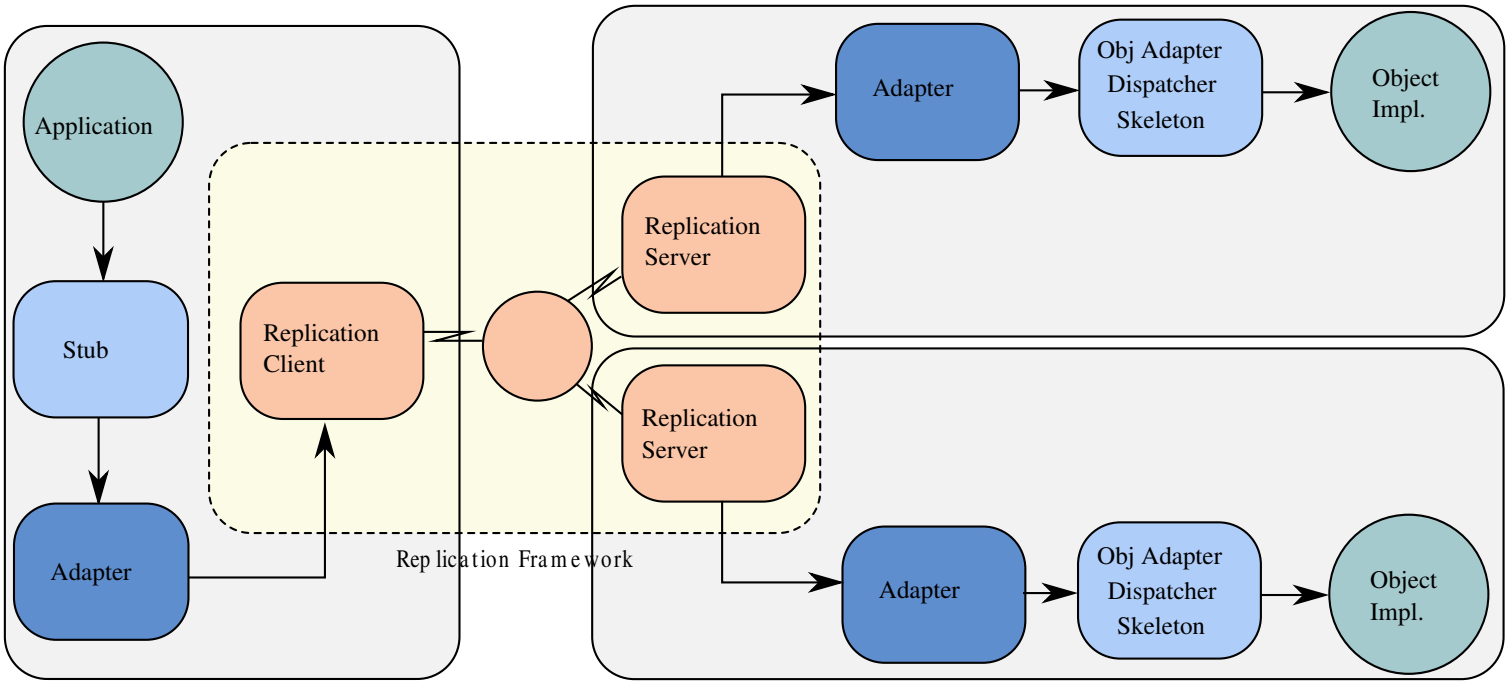
An Architecture for Replication Support

Step II: The Adapter



An Architecture for Replication Support

Step II: The Adapter

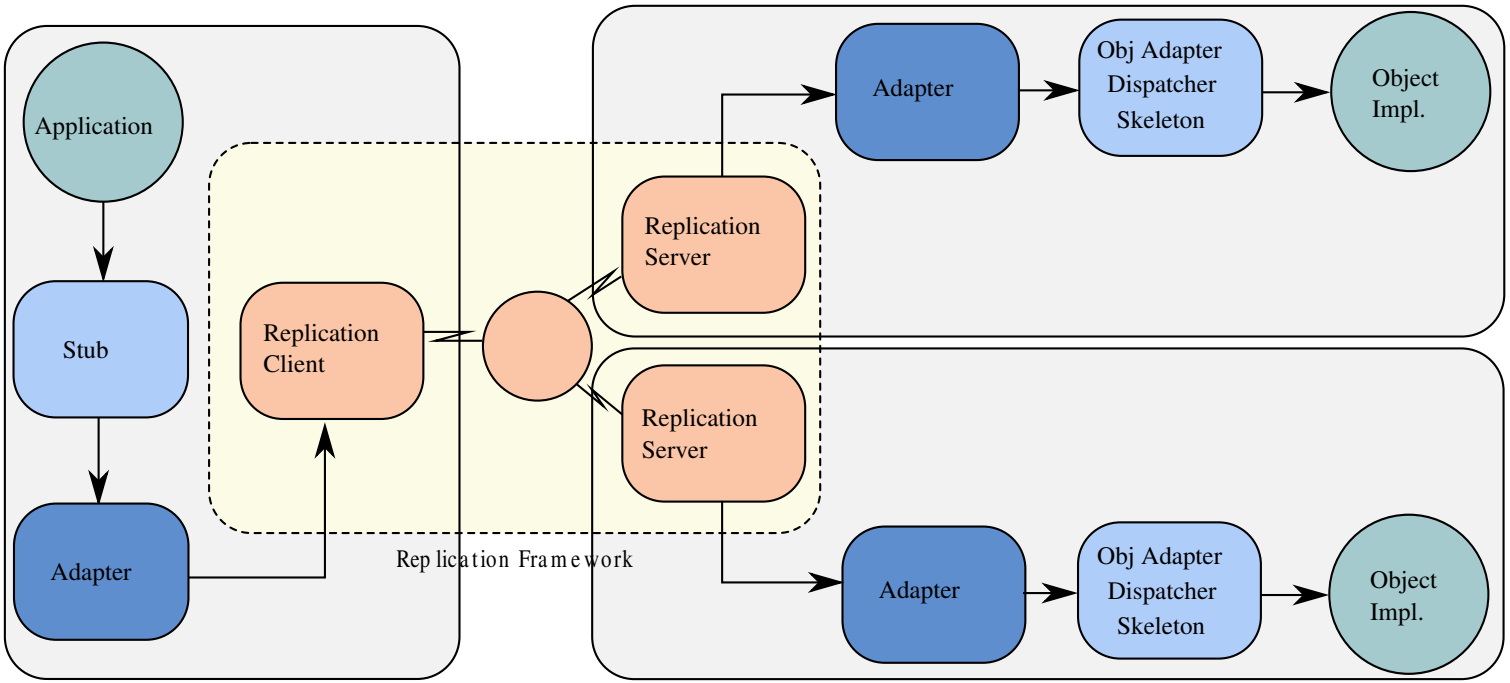


Adapter characteristics

- Conceptually part of the middleware
- Initialises replication framework
- Redirects calls to replication framework

An Architecture for Replication Support

Step II: The Adapter

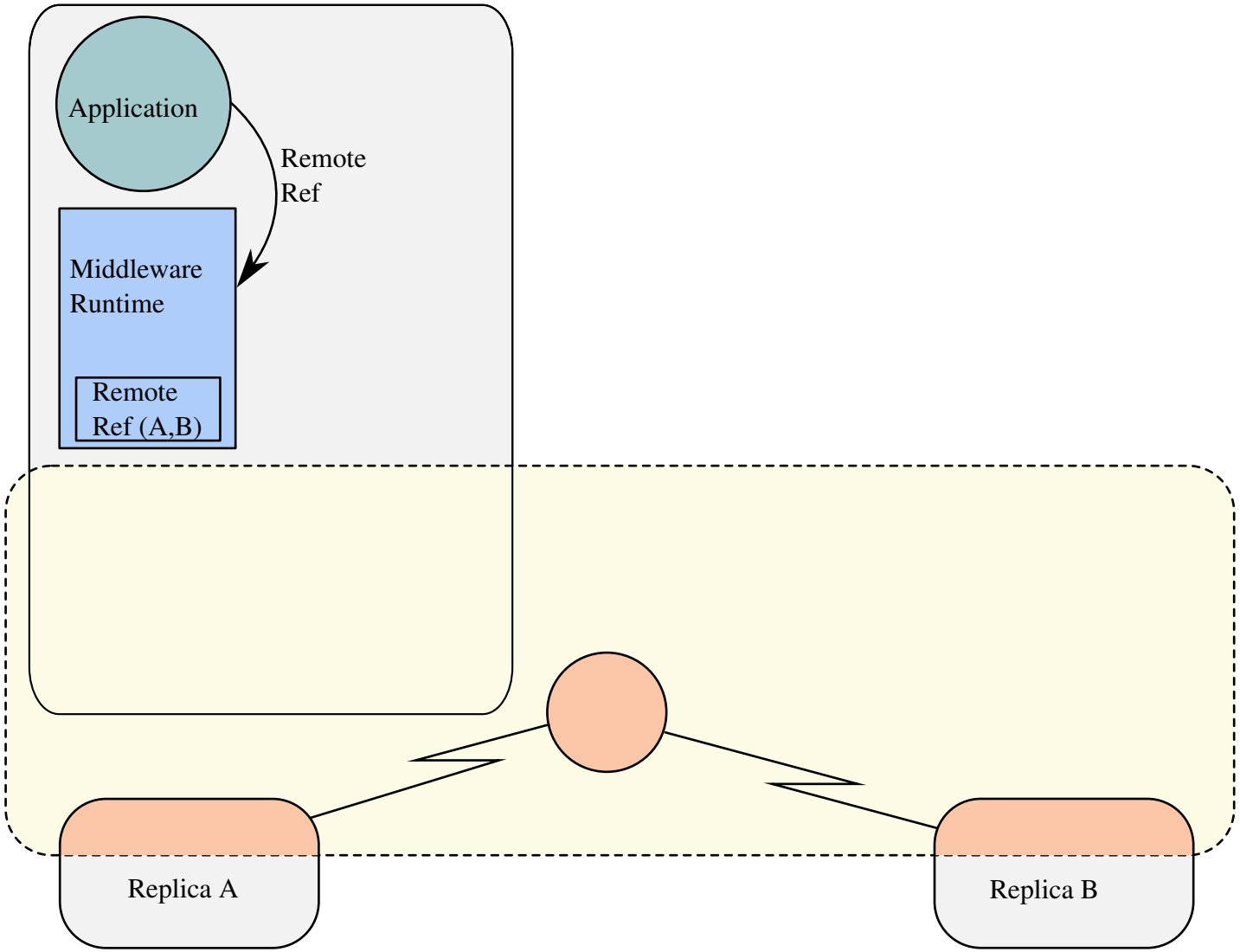


Adapter characteristics

- Conceptually part of the middleware
- Initialises replication framework
- Redirects calls to replication framework
- Does not solve all problems

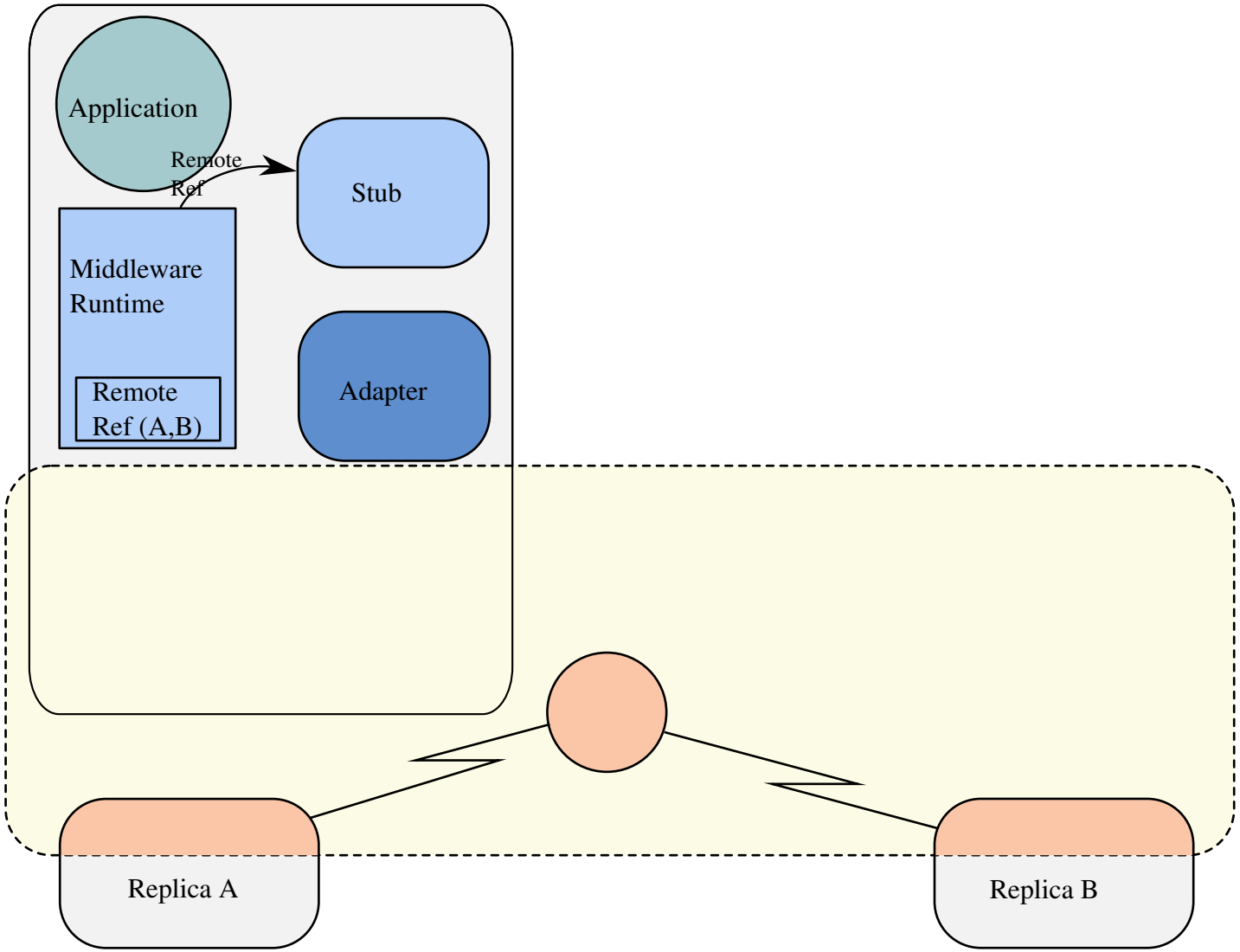
An Architecture for Replication Support

The binding problem



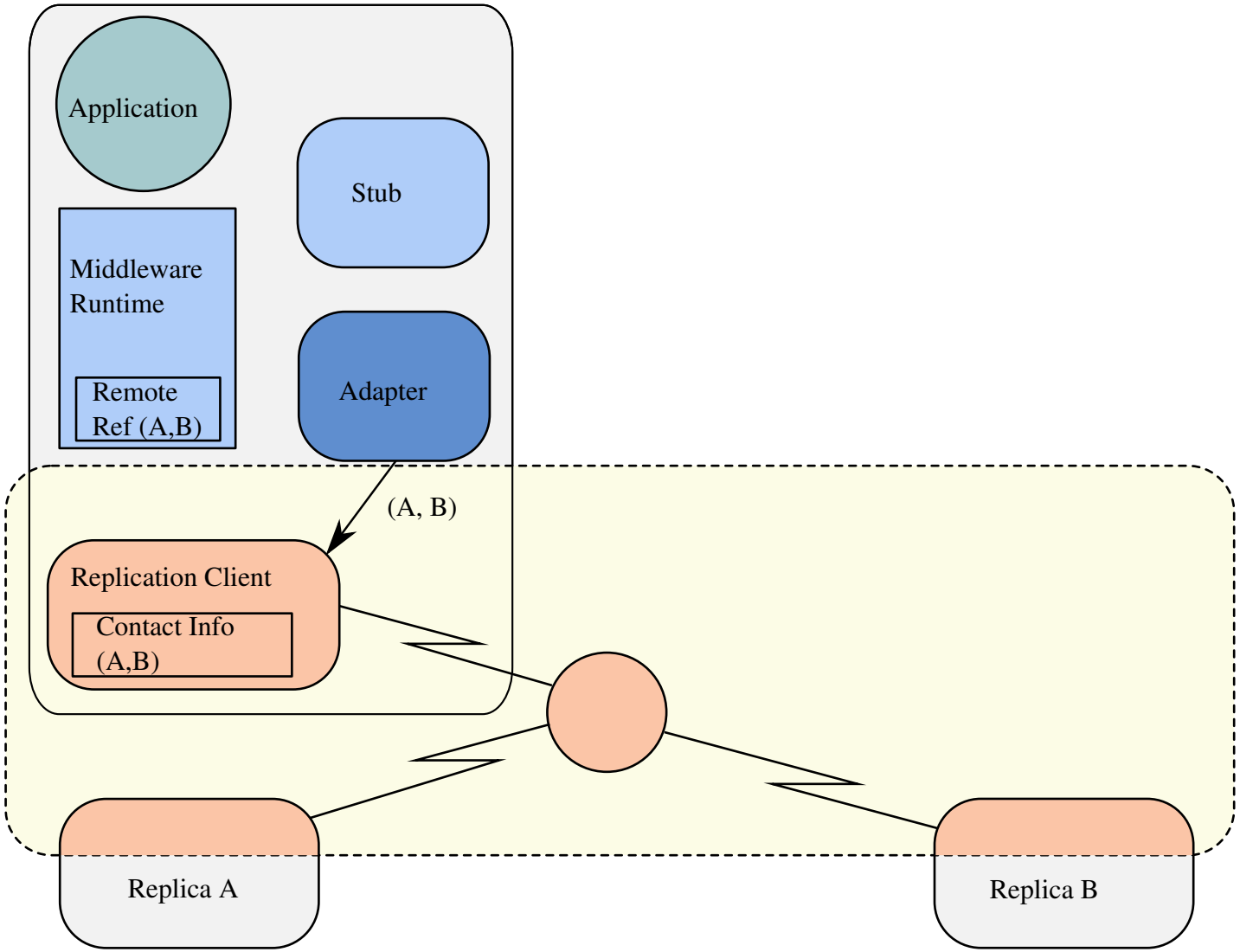
An Architecture for Replication Support

The binding problem



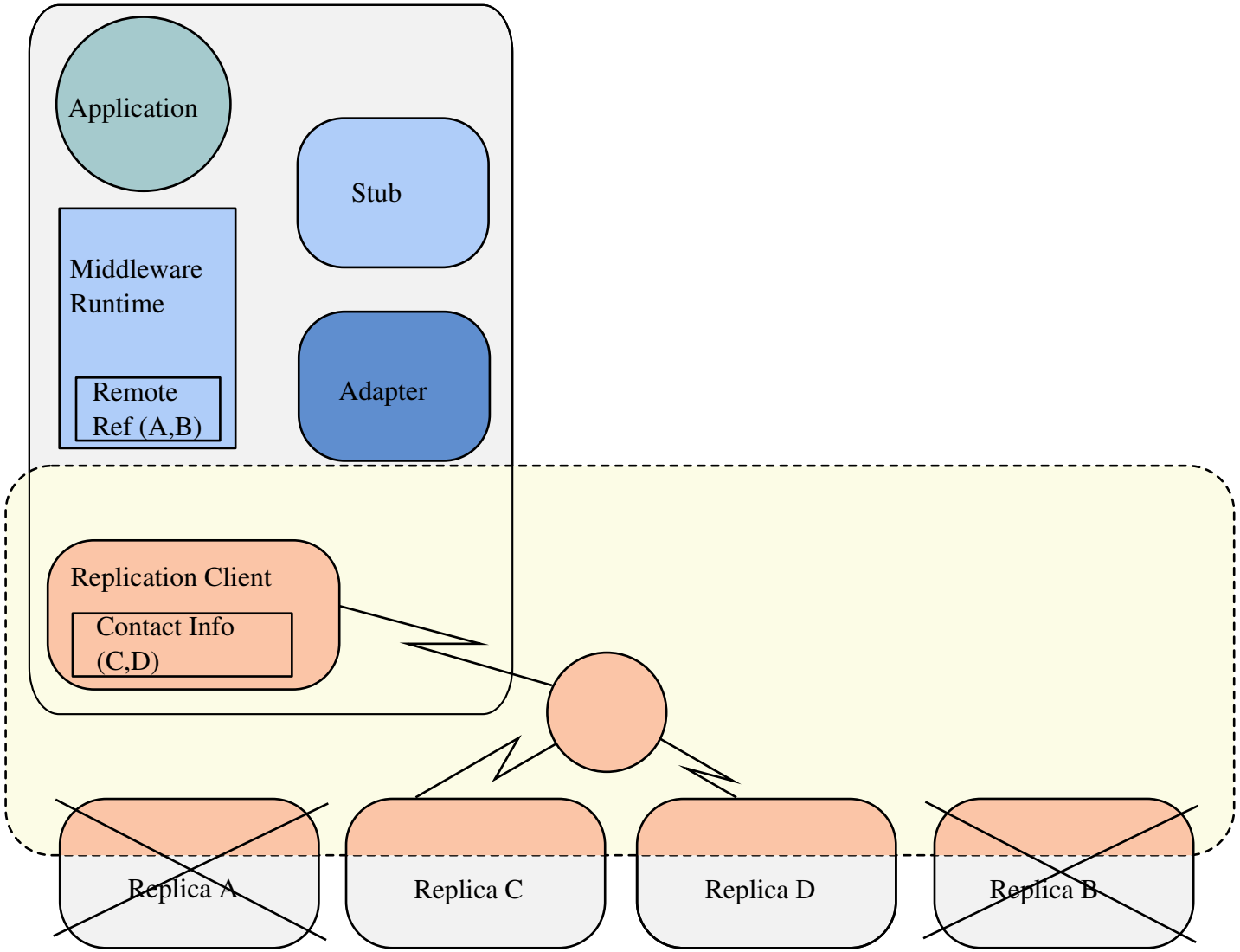
An Architecture for Replication Support

The binding problem



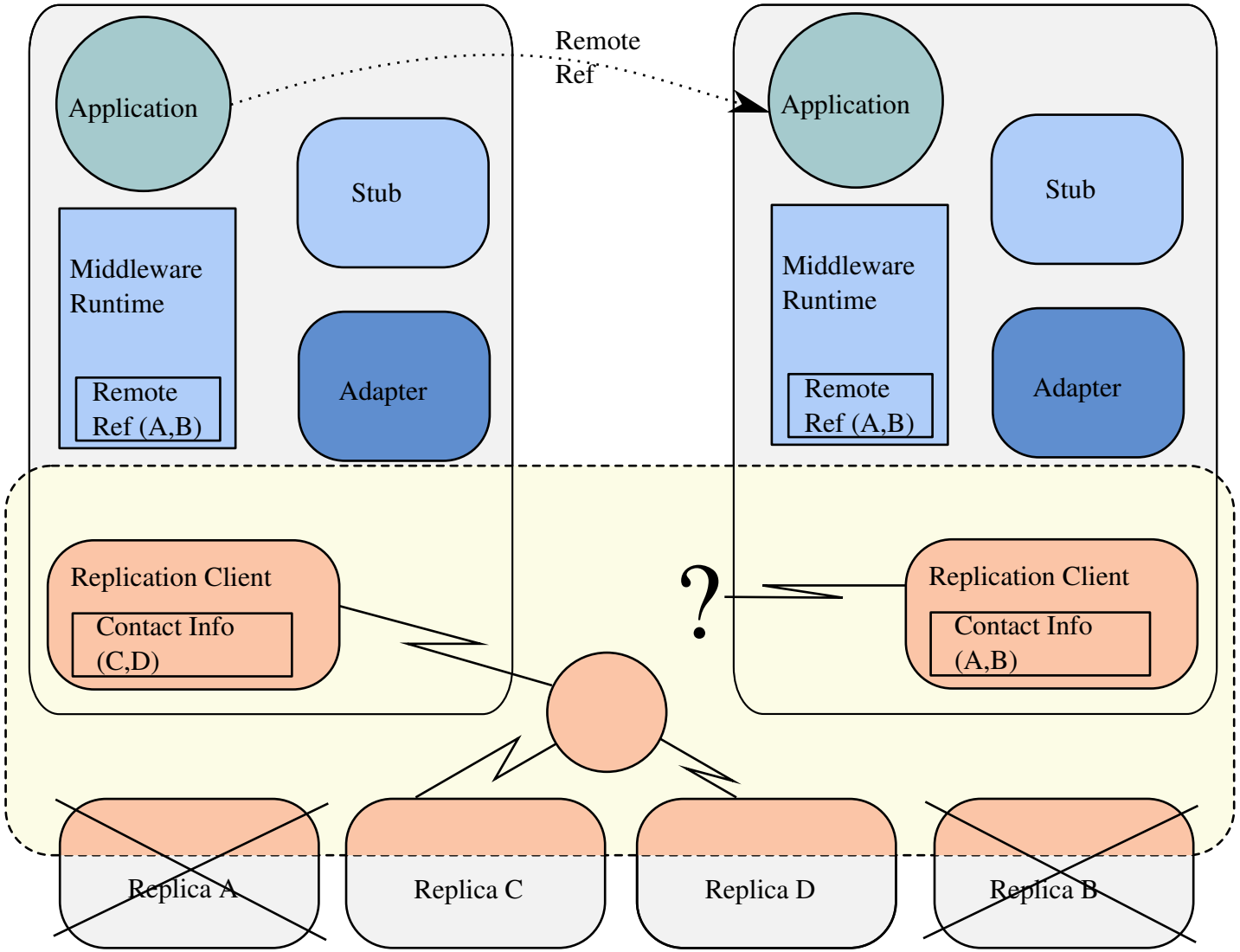
An Architecture for Replication Support

The binding problem



An Architecture for Replication Support

The binding problem



Solving the Binding Problem

Remote Reference should reflect lastest known state

Solving the Binding Problem

Remote Reference should reflect lastest known state

- Middleware system has to get the state

Solving the Binding Problem

Remote Reference should reflect lastest known state

- Middleware system has to get the state
  - Middleware pulls information from adapter/stub
    $\Rightarrow$ Not supported by all middleware systems

Solving the Binding Problem

Remote Reference should reflect lastest known state

- Middleware system has to get the state
  - Middleware pulls information from adapter/stub
    $\Rightarrow$ Not supported by all middleware systems
  - Replication framework pushes state into middleware
    $\Rightarrow$ No generic middleware interface

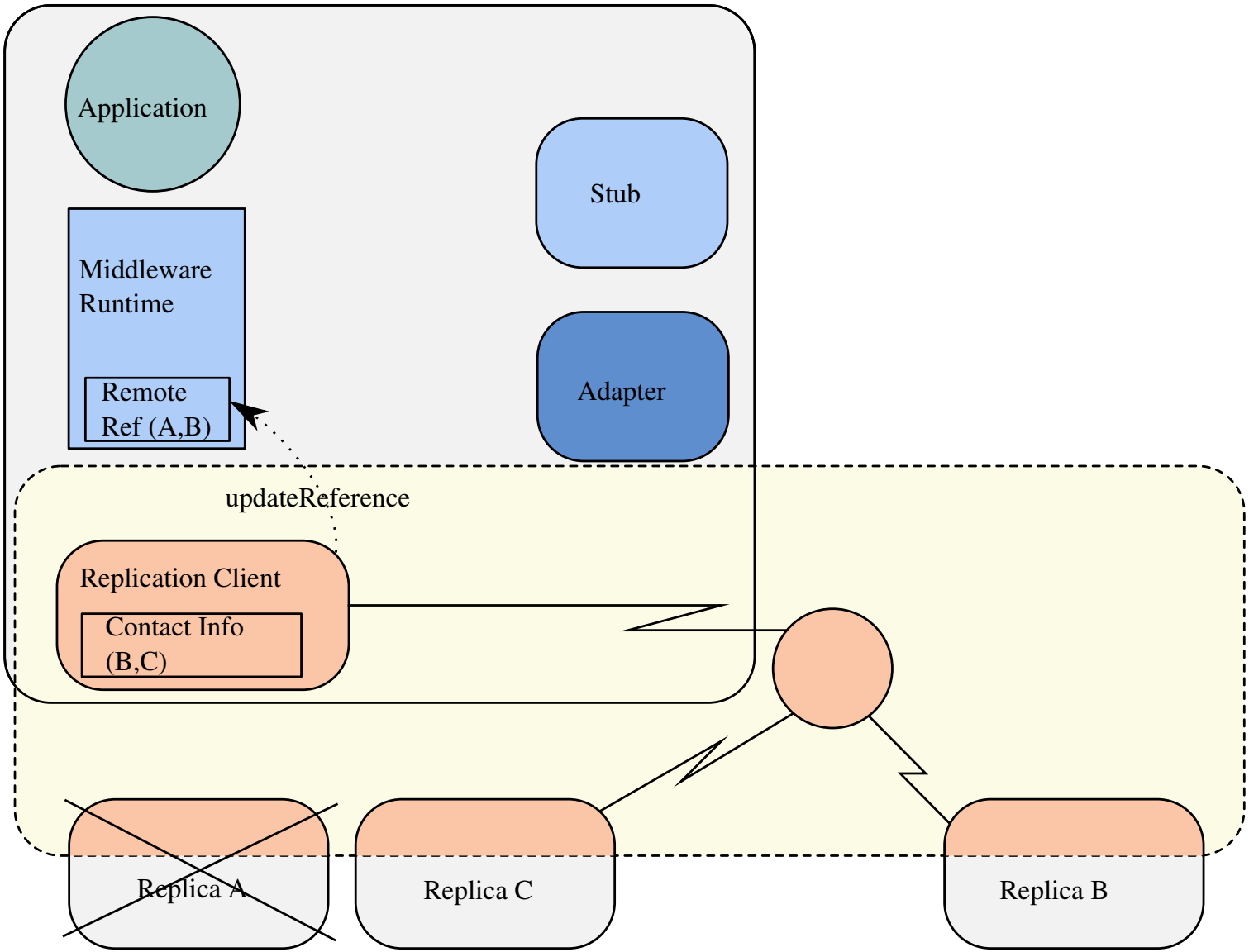
Solving the Binding Problem

Remote Reference should reflect lastest known state

- Middleware system has to get the state
  - Middleware pulls information from adapter/stub
    $\Rightarrow$ Not supported by all middleware systems
  - Replication framework pushes state into middleware
    $\Rightarrow$ No generic middleware interface
- Conversion step required
  - Converter as callback handler

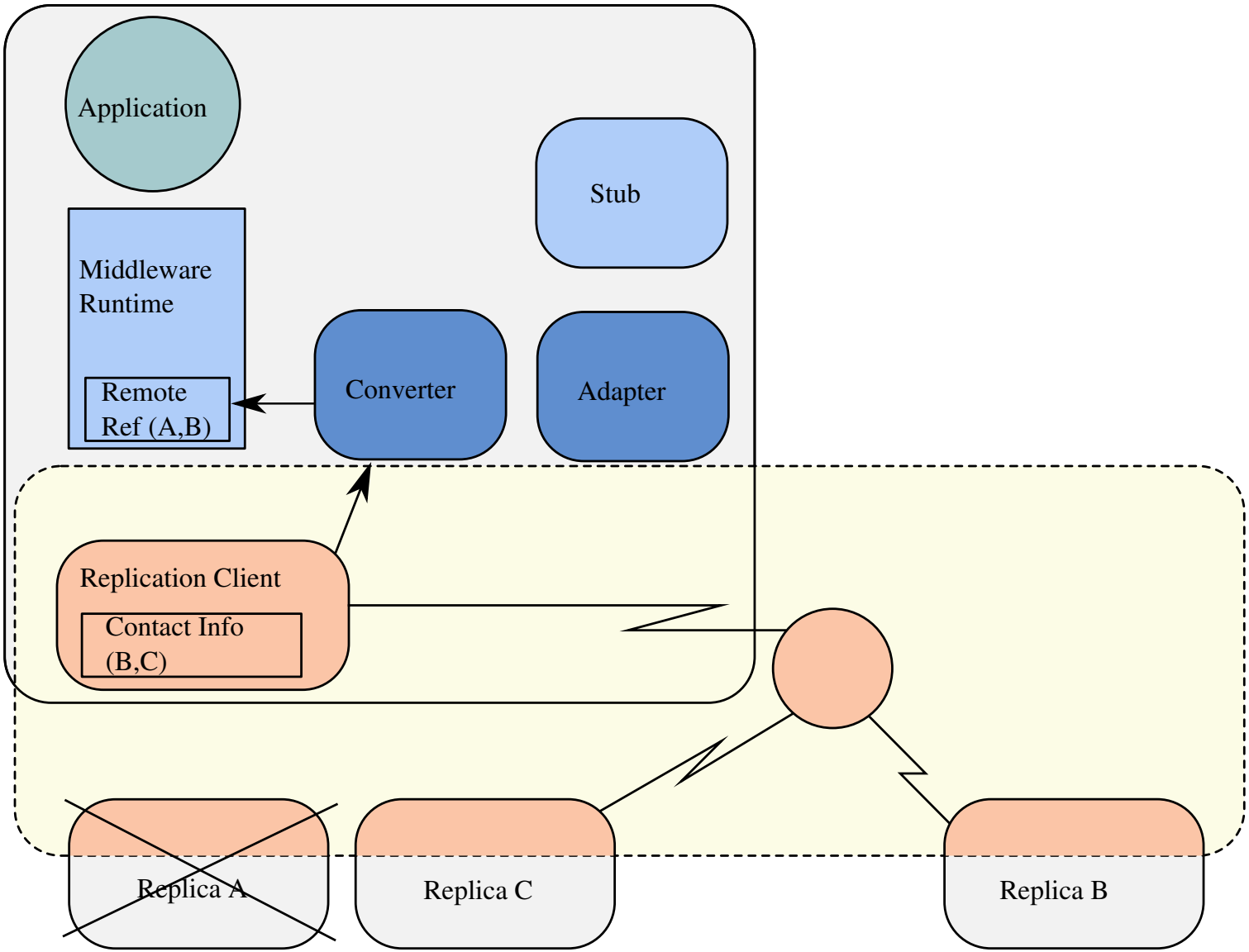
An Architecture for Replication Support

Step III: The Converter



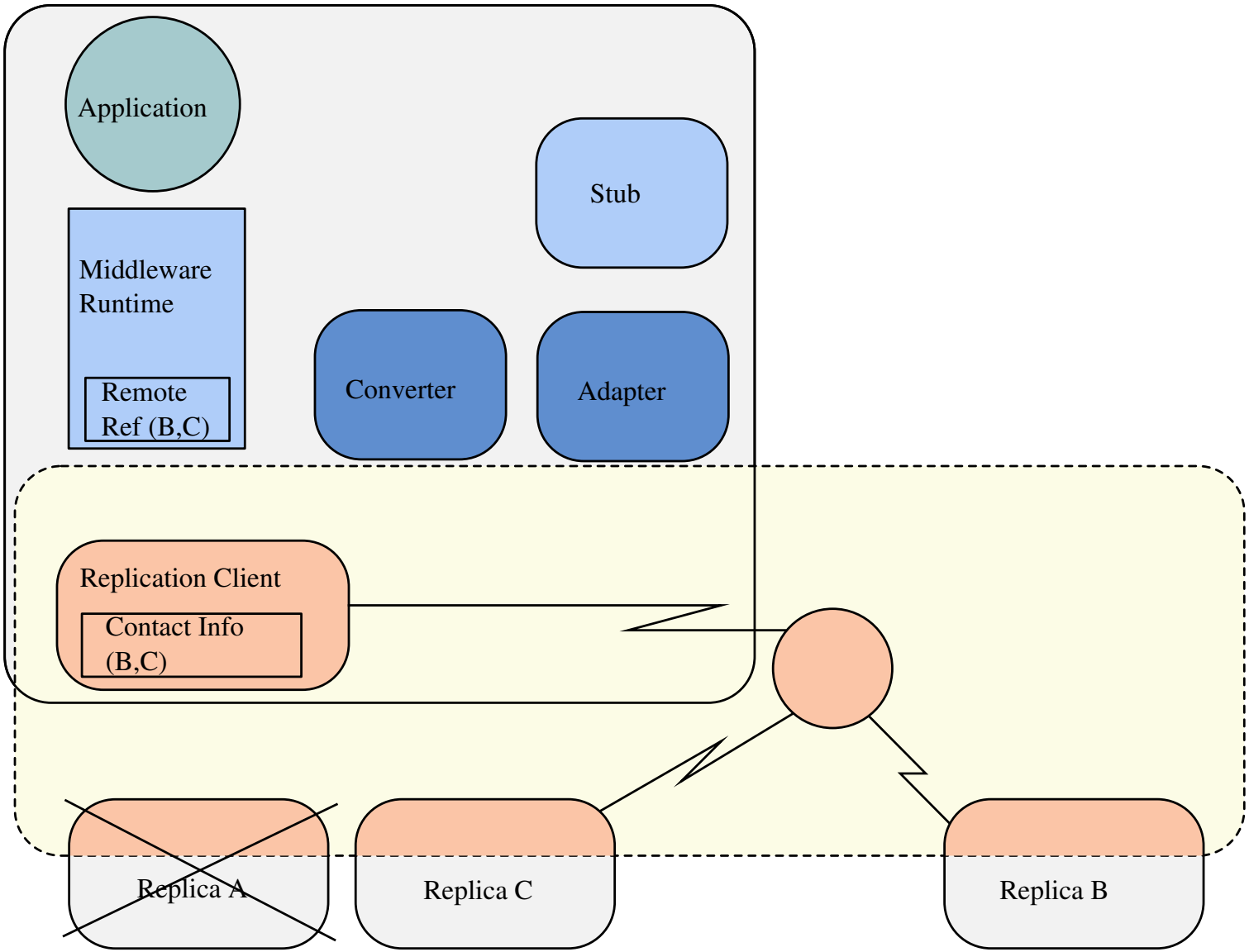
An Architecture for Replication Support

Step III: The Converter



An Architecture for Replication Support

Step III: The Converter



Separation in Practice

Starting Point:

FT*flex* replication framework integrated into Aspectix middleware

Separation in Practice

Starting Point:
FT*flex* replication framework integrated into Aspectix middleware

Coupling:

- IOR
  - Aspectix: Remote reference, binding
  - FT*flex*: Store state of replica group
- Coupling code: Code in framework unrelated to replication

Separation in Practice

Starting Point:
FT*flex* replication framework integrated into Aspectix middleware

Coupling:

- IOR
  - Aspectix: Remote reference, binding
  - FT*flex*: Store state of replica group
- Coupling code: Code in framework unrelated to replication

Separation

- IOR
  - Used for binding only
  - Extended FT*flex* with a *replication state management* facility
  - Converter translates between both systems
- Coupling Code: Moved to adapter

Support for Arbitrary Middleware Architectures

- Adapter and converter heavily depend on middleware
- In general four approaches to include the adapter
  - Integration: modify the middleware source code
  - Interception: intercept messages at OS level
  - Service approach: application calls replication services
  - Customised stub: All further stubs are copies of the first one
- Existing adapters
  - Aspectix-FT*flex* adapter
  - Java RMI-FT*flex*

Evaluation

Replication:

Java RMI	81.7 μs
RMI-FT*flex*	126.7 μs
Costs	55 %

Separation:

FT*flex* integrated	124.9 μs
FT*flex* adapter	130.4 μs
Costs	4 %

Set-up:

- One client, one replica
- On a single physical machine
- Different virtual machines

Conclusion

Conclusion:

- Middleware-independent replication support is feasible
  - Genericity by the use `byte[]` at interface
  - Handle data in an opaque way
- Light-weight Integration into middleware
  - Adapter and Converter
- Overhead introduced by replication logic

Ongoing work:

- Adapter for JacORB
- Comparison with FT-CORBA implementation

Separation of Concerns

Object reference, Interface definition, Object adapter, Binding, Code generation, Consistency management, Replication management

Separation of Concerns

Interface definition, Object adapter, Binding, Code generation, Consistency management, Replication management

Object reference

Middleware concerns

Replication concerns

Separation of Concerns

Interface definition, Object adapter, Binding, Code generation, Consistency management, Replication management

Middleware concerns

- Object reference

Replication concerns

Separation of Concerns

Object adapter, Binding, Code generation, Consistency management, Replication management

Interface definition

Middleware concerns

- Object reference

Replication concerns

Separation of Concerns

Object adapter, Binding, Code generation, Consistency management, Replication management

Middleware concerns

- Object reference
- Interface definition

Replication concerns

Separation of Concerns

Binding, Code generation, Consistency management, Replication management

Object adapter

Middleware concerns

- Object reference
- Interface definition

Replication concerns

Separation of Concerns

Binding, Code generation, Consistency management, Replication management

Middleware concerns

- Object reference
- Interface definition
- Object adapter

Replication concerns

- Object adapter

Separation of Concerns

Code generation, Consistency management, Replication management

Binding

Middleware concerns

- Object reference
- Interface definition
- Object adapter

Replication concerns

- Object adapter

Separation of Concerns

Code generation, Consistency management, Replication management

Middleware concerns

- Object reference
- Interface definition
- Object adapter
- Binding

Replication concerns

- Object adapter

Separation of Concerns

Consistency management, Replication management

Code generation

Middleware concerns

- Object reference
- Interface definition
- Object adapter
- Binding

Replication concerns

- Object adapter

Separation of Concerns

Consistency management, Replication management

Middleware concerns

- Object reference
- Interface definition
- Object adapter
- Binding
- Code generation
  - Dispatching
  - Marshalling in Stub and Skeleton

Replication concerns

- Object adapter

Separation of Concerns

Replication management

Consistency management

Middleware concerns

- Object reference
- Interface definition
- Object adapter
- Binding
- Code generation
  - Dispatching
  - Marshalling in Stub and Skeleton

Replication concerns

- Object adapter

Separation of Concerns

Replication management

Middleware concerns

- Object reference
- Interface definition
- Object adapter
- Binding
- Code generation
  - Dispatching
  - Marshalling in Stub and Skeleton
- Consistency management
  - Calling semantics

Replication concerns

- Object adapter
- Consistency management
  - Message ordering
  - Determinism

Separation of Concerns

Replication management

Middleware concerns

- Object reference
- Interface definition
- Object adapter
- Binding
- Code generation
  - Dispatching
  - Marshalling in Stub and Skeleton
- Consistency management
  - Calling semantics

Replication concerns

- Object adapter
- Consistency management
  - Message ordering
  - Determinism

Separation of Concerns

Middleware concerns

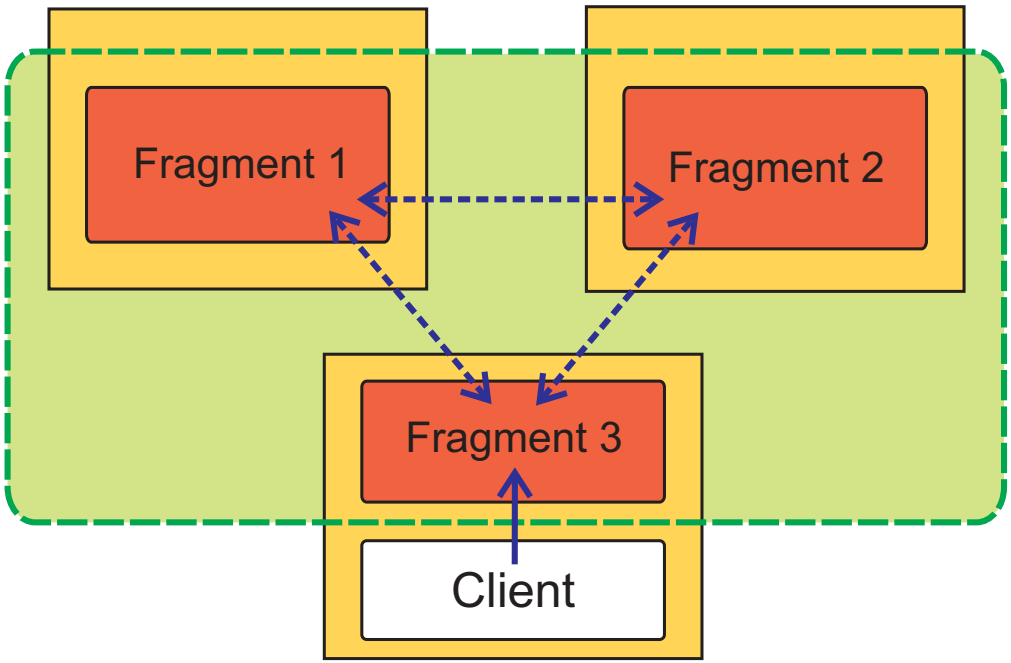
- Object reference
- Interface definition
- Object adapter
- Binding
- Code generation
  - Dispatching
  - Marshalling in Stub and Skeleton
- Consistency management
  - Calling semantics

Replication concerns

- Object adapter
- Consistency management
  - Message ordering
  - Determinism
- Replication Management
  - Monitoring of replicas
  - Start new replicas
  - migrate replicas

Aspectix

- CORBA-based
- Supports fragmented objects
  - An object consits of multiple fragments
  - Fragments are arbitrarily distributed over multiple address spaces
  - Distribution of state and functionality is arbitrary
  - Communication is an object-internal issue

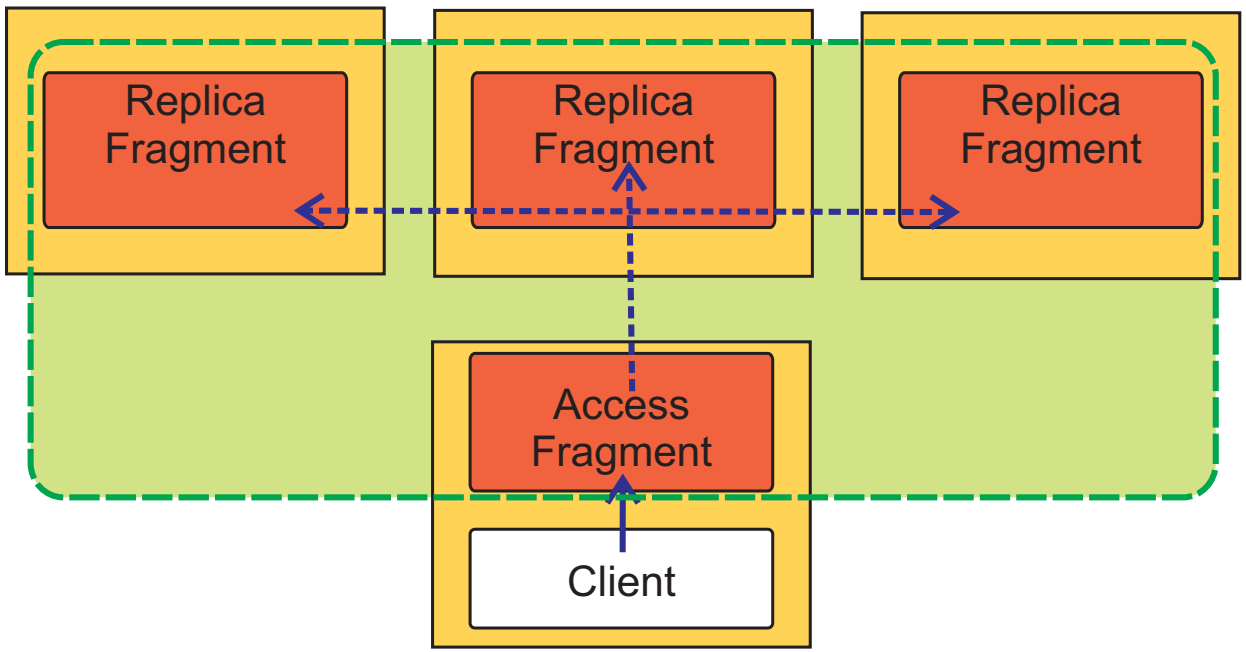


Aspectix

- CORBA-based
- Supports fragmented objects
  - An object consits of multiple fragments
  - Fragments are arbitrarily distributed over multiple address spaces
  - Distribution of state and functionality is arbitrary
  - Communication is an object-internal issue
- Features
  - Implicit binding
  - Interface Definition
  - Code generation
- but arbitrary distribution of state and functionality disables generation of marshalling and dispatching

FT*flex*

- Replication infrastructure based on Aspectix
  - Fragmented objects
- Has a more concrete object structure
  - Access fragments
  - Replica fragments
  - Code generation results in marshalling and dispatching code



FT*flex*

- Replication infrastructure based on Aspectix
  - Fragmented objects
- Has a more concrete object structure
  - Access fragments
  - Replica fragments
  - Code generation results in marshalling and dispatching code
- Object adapter part of replica fragment
  - Message Management
  - Dispatching
  - Error handling
- Own group communication system