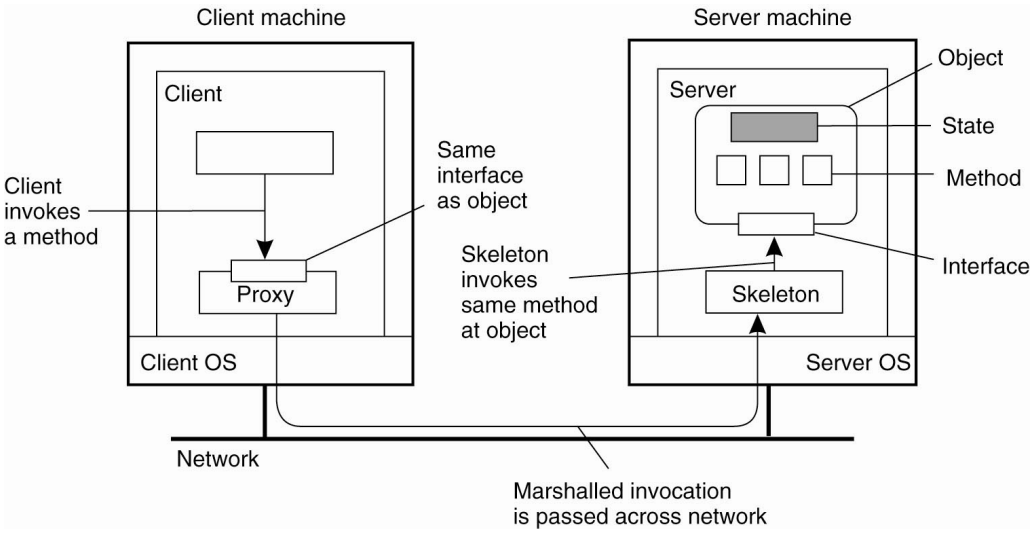
Today: Distributed Objects

- Case study: EJBs (Enterprise Java Beans)
- Case study: CORBA

Distributed Objects



- Figure 10-1. Common organization of a remote object with client-side proxy.

Distributed Objects vs. RPC

RPC : Remote Procedure Call

- Provides argument marshalling / unmarshalling
- Server handles invocation

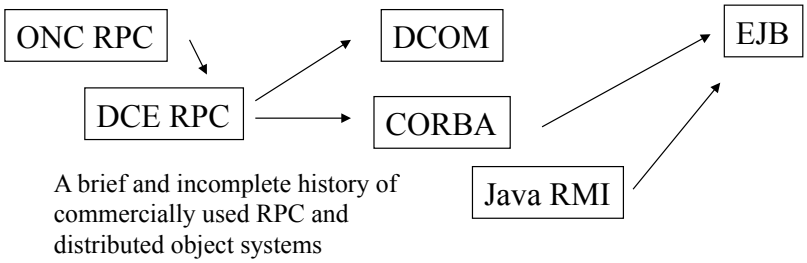
Distributed Objects

- Remote methods on remote objects
- RPC + distributed object references

Distributed object operation:

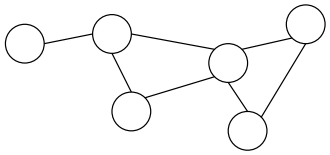
- Server side: create object, register it (register with what?) (always in this order?)
- Client side: get object reference (from where?), invoke method

Distributed Objects through History



A brief and incomplete history of commercially used RPC and distributed object systems

The vision



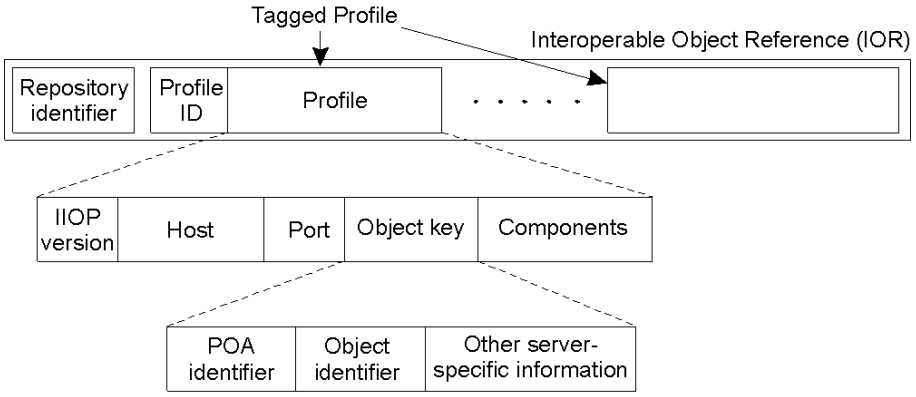
a Grand Distributed System

The reality

Client/Server

Naming: Object References

CORBA object reference



- Interoperable object reference: language-independent techniques for referring to objects

Object references and Naming

- First versions of CORBA used **opaque** object references
  - How do you locate the object? Via a location service.
  - What is the interface to the location service?
  - How do you invoke the location service?

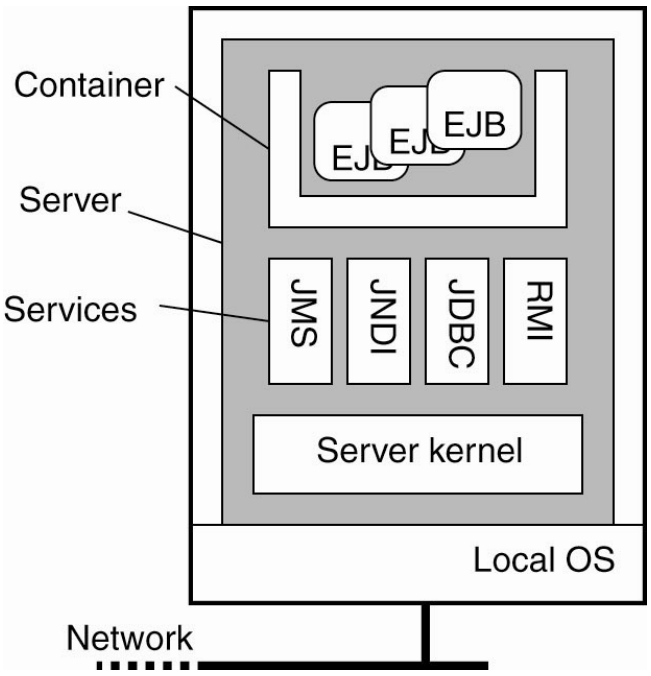
- Java (and CORBA 3.0) use **transparent** object references
  - Can be decoded at the client
  - Java reference can encode all information (e.g. code) needed to invoke an object.

Binding

- Static vs. Dynamic binding
  - What is the difference?
  - Advantages of static binding?
  - Of dynamic binding?
- What state is involved in client binding?
  - What happens if the client crashes?
  - The server?

Example: Enterprise Java Beans

- Figure 10-2. General architecture of an EJB server.



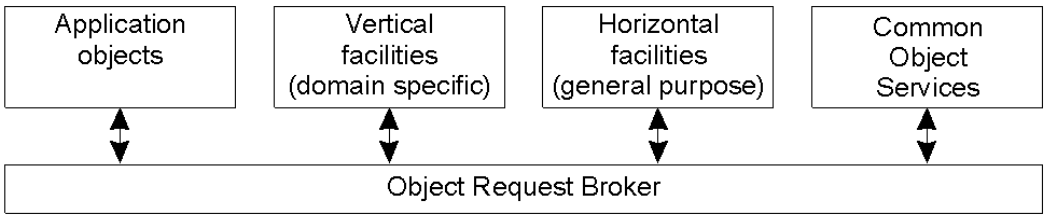
Parts of an EJB

- Home interface:
  - Object creation, deletion
  - Location of persistent objects (entity beans)
  - Object identifier is class-managed
- Remote interface
  - "business logic"
  - i.e. the object itself
- Terminology differences
  - Client/server -> web applications

Four Types of EJBs

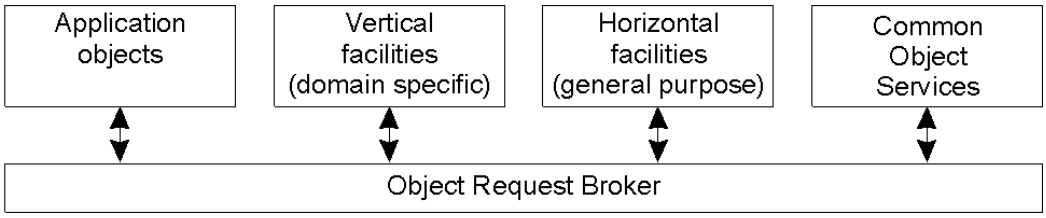
- Stateless session beans
- Stateful session beans
- Entity beans
- Message-driven beans

Overview of CORBA



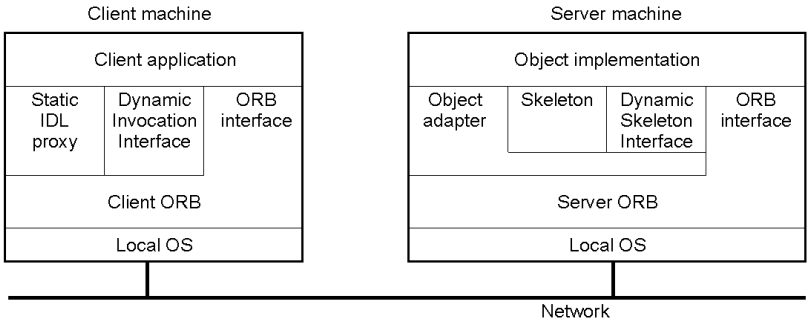
- Common Object Request Broker Architecture
  - Specification of a distributed middleware
  - Specs drawn up by Object Management Group (OMG)
  - http://www.omg.org
- Goal: Interoperability with distributed applications on various platforms

CORBA Overview



- Object request broker (ORB)
  - Core of the middleware platform
  - Handles communication between objects and clients
  - Handles distribution and heterogeneity issues
  - May be implemented as libraries
- Facilities: composition of CORBA services

Object Model



- Objects & services specified using an Interface Definition language (IDL)
  - Used to specify interface of objects and/or services
- ORB: run-time system that handles object-client communication
- Dynamic invocation interface: allows object invocation at run-time
  - Generic *invoke* operation: takes object reference as input
  - Interface repository stores all interface definitions

CORBA Services

- Collection service: group objects into lists, queues,..
- Query service: use query language to query for service(s)
- Concurrency control service: locking services
- Event service: interrupt upon a specific event
- Many more…

- Q: Do CORBA objects have a corresponding class?

Corba Services

Service	Description
Collection	Facilities for grouping objects into lists, queue, sets, etc.
Query	Facilities for querying collections of objects in a declarative manner
Concurrency	Facilities to allow concurrent access to shared objects
Transaction	Flat and nested transactions on method calls over multiple objects
Event	Facilities for asynchronous communication through events
Notification	Advanced facilities for event-based asynchronous communication
Externalization	Facilities for marshaling and unmarshaling of objects
Life cycle	Facilities for creation, deletion, copying, and moving of objects
Licensing	Facilities for attaching a license to an object
Naming	Facilities for systemwide name of objects
Property	Facilities for associating (attribute, value) pairs with objects
Trading	Facilities to publish and find the services on object has to offer
Persistence	Facilities for persistently storing objects
Relationship	Facilities for expressing relationships between objects
Security	Mechanisms for secure channels, authorization, and auditing
Time	Provides the current time within specified error margins

Object Invocation Models

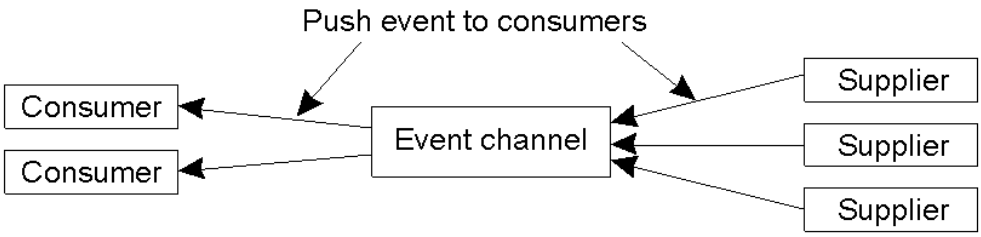
Request type	Failure semantics	Description
Synchronous	At-most-once	Caller blocks until a response is returned or an exception is raised
One-way	Best effort delivery	Caller continues immediately without waiting for any response from the server
Deferred synchronous	At-most-once	Caller continues immediately and can later block until response is delivered

- Invocation models supported in CORBA.
  - Original model was RMI/RPC-like
  - Current CORBA versions support additional semantics

What went wrong with CORBA?

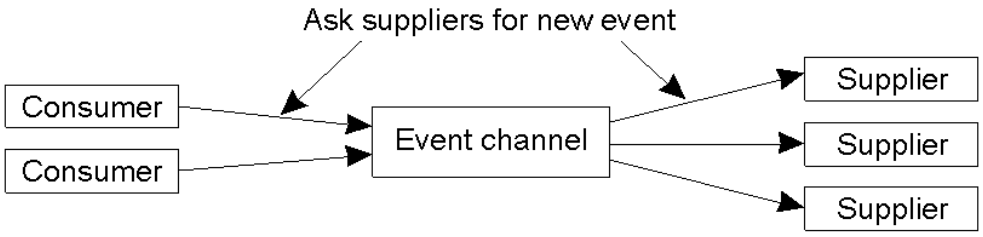
- Where is it now?
  - Inside EJB, I think
  - Gnome desktop
  - Embedded CORBA?
- Design by committee
  - Competing commercial interests
  - … time to go teach….

Event and Notification Services (1)



- The logical organization of suppliers and consumers of events, following the push-style model.

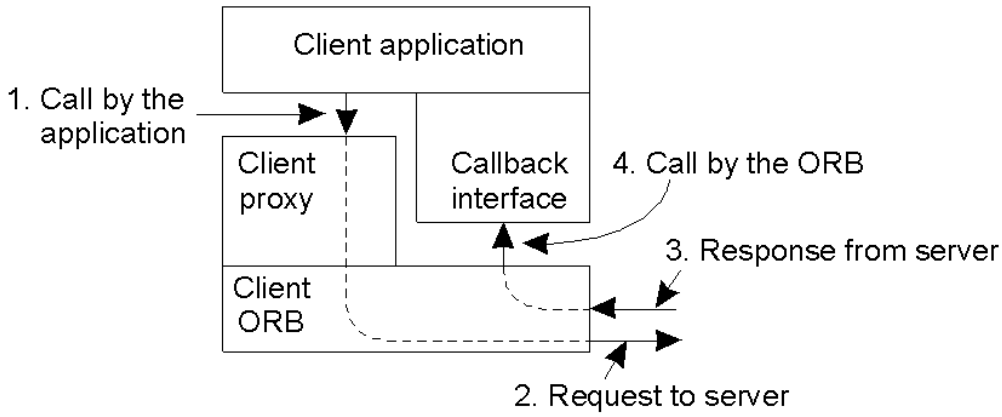
Event and Notification Services (2)



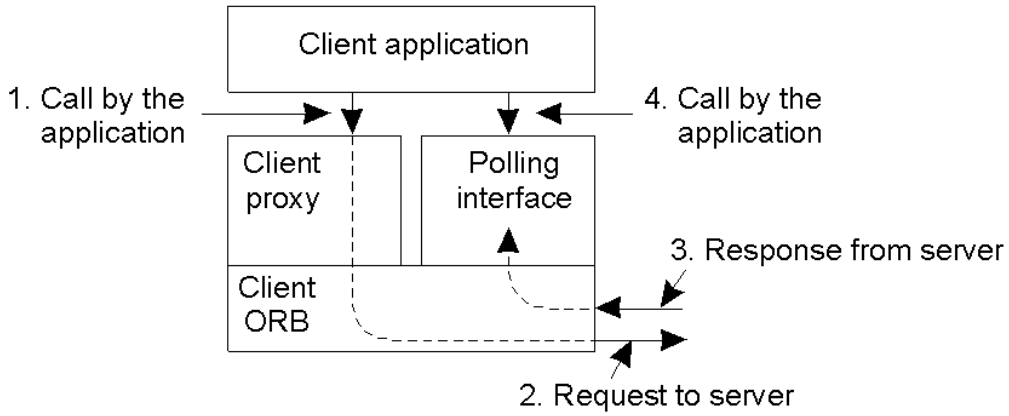
- The pull-style model for event delivery in CORBA.

Messaging: Async. Method Invocation

- CORBA's callback model for asynchronous method invocation.

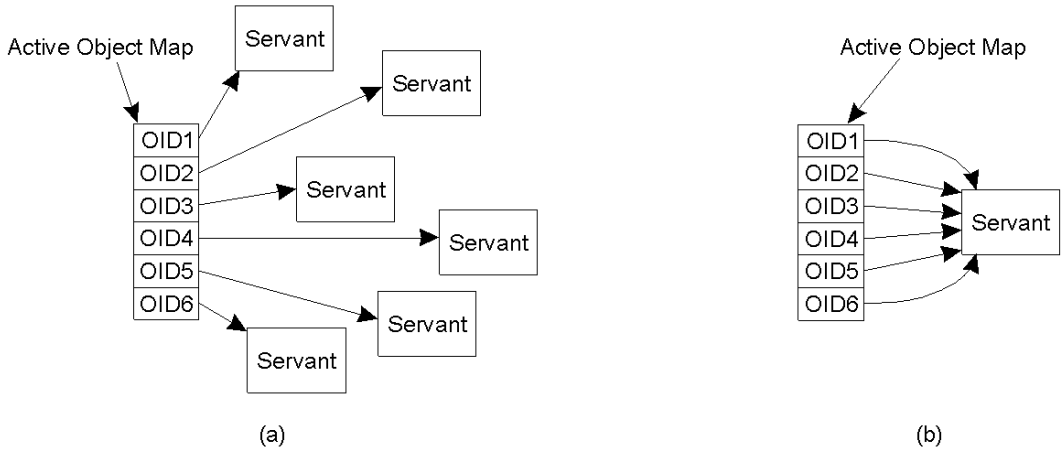


Messaging (2)



- CORBA'S polling model for asynchronous method invocation.

Portable Object Adaptor (1)



- POA: Wrappers for server-side code (makes code look like objects)

a) The POA supports multiple servants.

b) The POA supports a single servant.

Portable Object Adaptor (2)

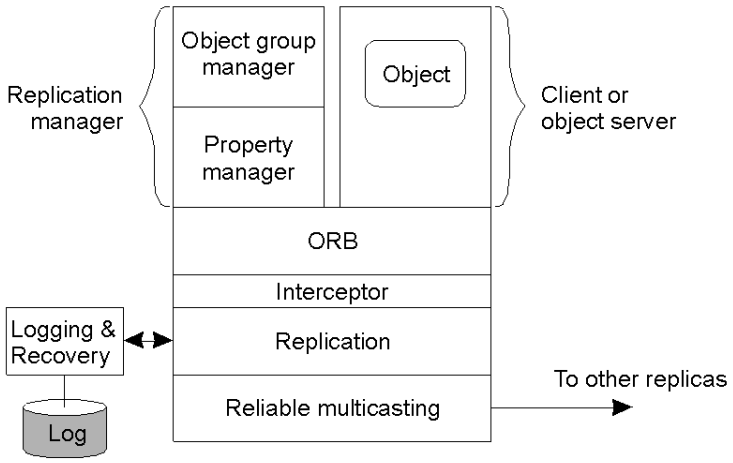
```
My_servant *my_object;              // Declare a reference to a C++ object
CORBA::Objectid_var oid;            // Declare a CORBA identifier

my_object = new MyServant;          // Create a new C++ object
oid = poa ->activate_object (my_object);
                                    // Register C++ object as CORBA OBJECT
```

- Changing a C++ object into a CORBA object.

An Example Architecture

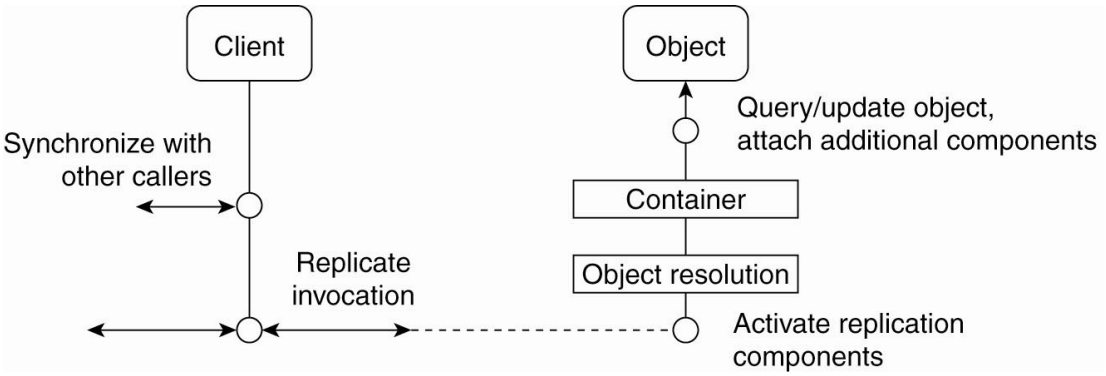
- An example architecture of a fault-tolerant CORBA system.



Replication Frameworks (1)

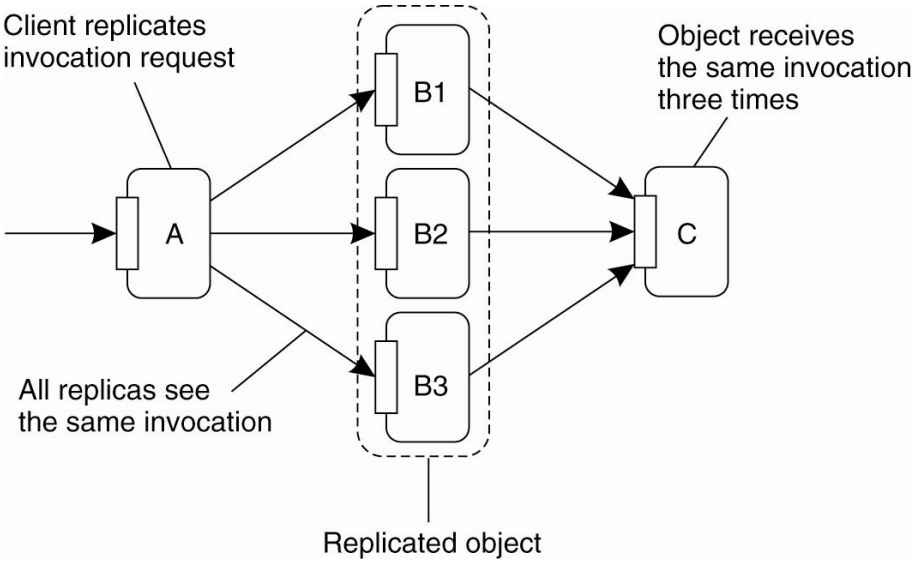
- Invocations to objects are intercepted at three different points:
- At the client side just before the invocation is passed to the stub.
- Inside the client's stub, where the interception forms part of the replication algorithm.
- At the server side, just before the object is about to be invoked.

Replication Frameworks (2)



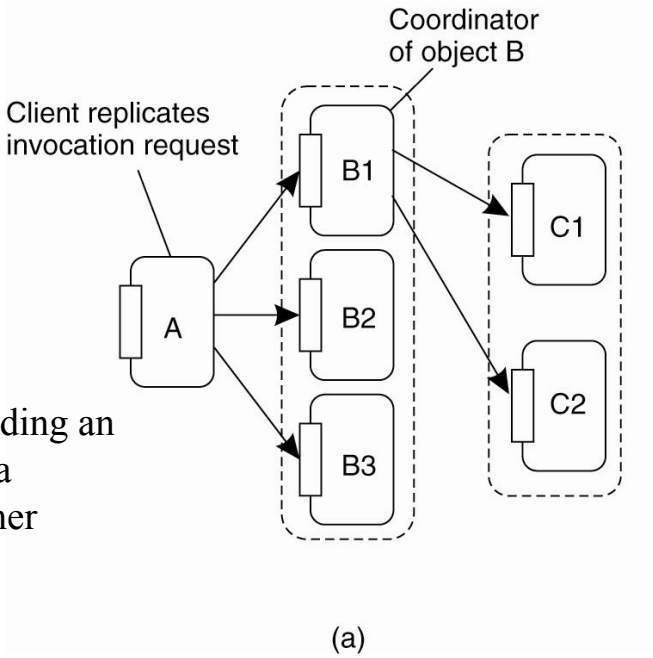
- Figure 10-16. A general framework for separating replication algorithms from objects in an EJB environment.

Replicated Invocations (1)



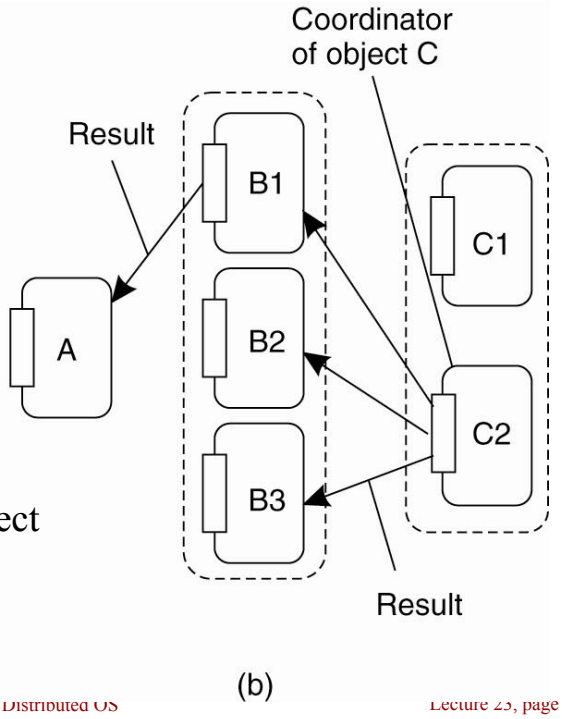
- Figure 10-17. The problem of replicated method invocations.

Replicated Invocations (2)



- Figure 10-18. (a) Forwarding an invocation request from a replicated object to another replicated object.

Replicated Invocations (3)



- Figure 10-18. (b) Returning a reply from one replicated object to another.