Distributed Objects

Figure 10-1. Common organization of a remote object with client-side proxy.
Example: Enterprise Java Beans

Figure 10-2. General architecture of an EJB server.
Four Types of EJBs

- Stateless session beans
- Stateful session beans
- Entity beans
- Message-driven beans
Globe Distributed Shared Objects (1)

Figure 10-3. The organization of a Globe distributed shared object.
Globe Distributed Shared Objects (2)

Figure 10-4. The general organization of a local object for distributed shared objects in Globe.
Object Adapter

Figure 10-5. Organization of an object server supporting different activation policies.
Example: The Ice Runtime System

```c
main(int argc, char* argv[]) {  
    Ice::Communicator    ic;
    Ice::ObjectAdapter   adapter;
    Ice::Object          object;

    ic = Ice::initialize(argc, argv);
    adapter =
        ic->createObjectAdapterWithEndPoints( "MyAdapter","tcp -p 10000");
    object = new MyObject;
    adapter->add(object, object::ID);
    adapter->activate();
    ic->waitForShutdown();
}
```

Figure 10-6. Example of creating an object server in Ice.
Binding a Client to an Object

(a) An example with implicit binding using only global references.

Distr_object* obj_ref;
obj_ref = ...;
obj_ref->do_something();

(b) An example with explicit binding using global and local references.

Distr_object obj_ref;
Local_object* obj_ptr;
obj_ref = ...;
obj_ptr = bind(obj_ref);
obj_ptr->do_something();

Figure 10-7. (a) An example with implicit binding using only global references. (b) An example with explicit binding using global and local references.
Parameter Passing

Figure 10-8. The situation when passing an object by reference or by value.
Object-Based Messaging (1)

Figure 10-9. CORBA’s callback model for asynchronous method invocation.

Tanenbaum & Van Steen, Distributed Systems: Principles and Paradigms, 2e, (c) 2007 Prentice-Hall, Inc. All rights reserved. 0-13-239227-5
Object-Based Messaging (2)

Figure 10-10. CORBA’s polling model for asynchronous method invocation.
CORBA Object References

Figure 10-11. The organization of an IOR with specific information for IIOP.
### Globe Object References (1)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protocol identifier</td>
<td>A constant representing a (known) protocol</td>
</tr>
<tr>
<td>Protocol address</td>
<td>A protocol-specific address</td>
</tr>
<tr>
<td>Implementation handle</td>
<td>Reference to a file in a class repository</td>
</tr>
</tbody>
</table>

**Figure 10-12.** The representation of a protocol layer in a stacked contact address.
### Globe Object References (2)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Implementation handle</td>
<td>Reference to a file in a class repository</td>
</tr>
<tr>
<td>Initialization string</td>
<td>String that is used to initialize an implementation</td>
</tr>
</tbody>
</table>

*Figure 10-13. The representation of an instance contact address.*
Synchronization

Figure 10-14. Differences in control flow for locking objects
Entry Consistency

Figure 10-15. Deterministic thread scheduling for replicated object servers.
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.
Figure 10-18. (b) Returning a reply from one replicated object to another.
Example: Fault-Tolerant CORBA

Figure 10-19. A possible organization of an IOGR for an object group having a primary and backups.
An Example Architecture

Figure 10-20. An example architecture of a fault-tolerant CORBA system.
Example: Fault-Tolerant Java

Causes for nondeterministic behavior:

1. JVM can execute native code, that is, code that is external to the JVM and provided to the latter through an interface.
2. Input data may be subject to nondeterminism.
3. In the presence of failures, different JVMs will produce different output revealing that the machines have been replicated.
Overview of Globe Security

<table>
<thead>
<tr>
<th>User certificate</th>
<th>Replica certificate</th>
<th>Administrative certificate</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K_{Alice}^+$</td>
<td>$K_{Repl}^+$</td>
<td>$K_{Adm}^+$</td>
</tr>
<tr>
<td>U: 0010011100</td>
<td>R: 11000111100</td>
<td>R: 11011111100</td>
</tr>
<tr>
<td>sig(O, {U, $K_{Alice}^+$})</td>
<td>sig(O, {R, $K_{Repl}^+$})</td>
<td>U: 0110011111</td>
</tr>
<tr>
<td></td>
<td></td>
<td>D: 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sig(O, {R, U, D, $K_{Adm}^+$})</td>
</tr>
</tbody>
</table>

(a)                                                                                              (b)                                                                                              (c)

Figure 10-21. Certificates in Globe: (a) a user certificate, (b) a replica certificate, (c) an administrative certificate.
Secure Method Invocation (1)

Figure 10-22. Secure method invocation in Globe.
Secure Method Invocation (2)

Steps for securely invoking a method of a Globe object:

1. Application issues a invocation request by locally calling the associated method
2. Control subobject checks the user permissions with the information stored in the local security object.
3. Request is marshaled and passed on.
4. Replication subobject requests the middleware to set up a secure channel to a suitable replica.
Secure Method Invocation (3)

5. Security object first initiates a replica lookup.
6. Once a suitable replica has been found, security subobject can set up a secure channel with its peer, after which control is returned to the replication subobject.
7. Request is now passed on to the communication subobject.
8. Subobject encrypts and signs the request so that it can pass through the channel.
Secure Method Invocation (4)

9. After its receipt, the request is decrypted and authenticated.
10. Request then passed on to the server-side replication subobject.
11. Authorization takes place:
12. Request is then unmarshaled.
13. Finally, the operation can be executed.