Assignment 3

Problem Statement

To develop any distributed application with CORBA program using JAVA IDL.

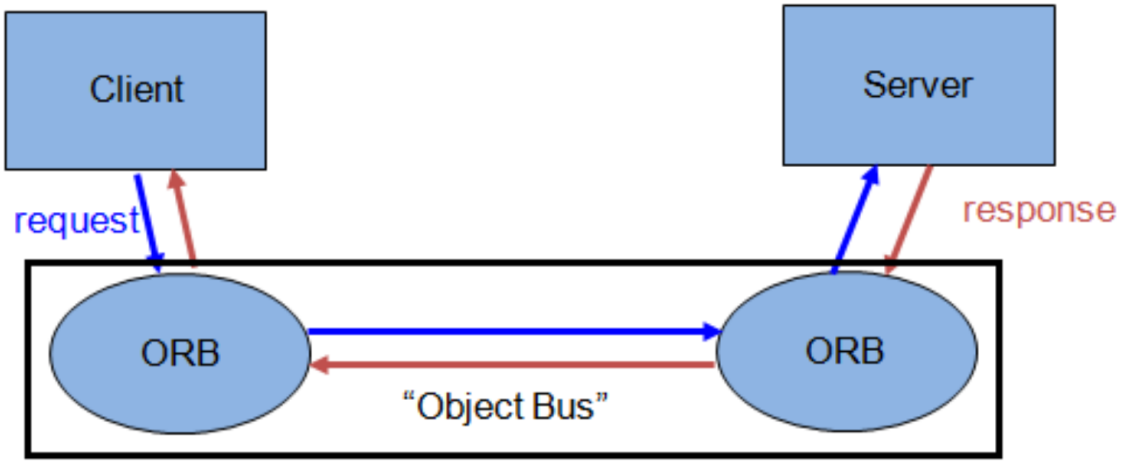
Objective

By the end of this assignment, we should be able to implement any distributed applications based on CORBA.

Relevant Theory

CORBA

- Stands for Common Object Request Broker Architecture.
- A CORBA application is developed using IDL (Interface Definition Language).
- IDL is used to define interfaces and the Java IDL compiler generates skeleton code.
- CORBA technology is an integral part of the Java platform. It consists of an Object Request Broker (ORB), APIs for the RMI programming model, and APIs for the IDL programming model.
- It is a specification for creating distributed objects and NOT a
- programming language.
- It promotes design of applications as a set of cooperating objects.
- Clients are isolated from servers by interface.
- CORBA objects run on any platform, can be located anywhere on the network and can be written in any language that has IDL mapping.

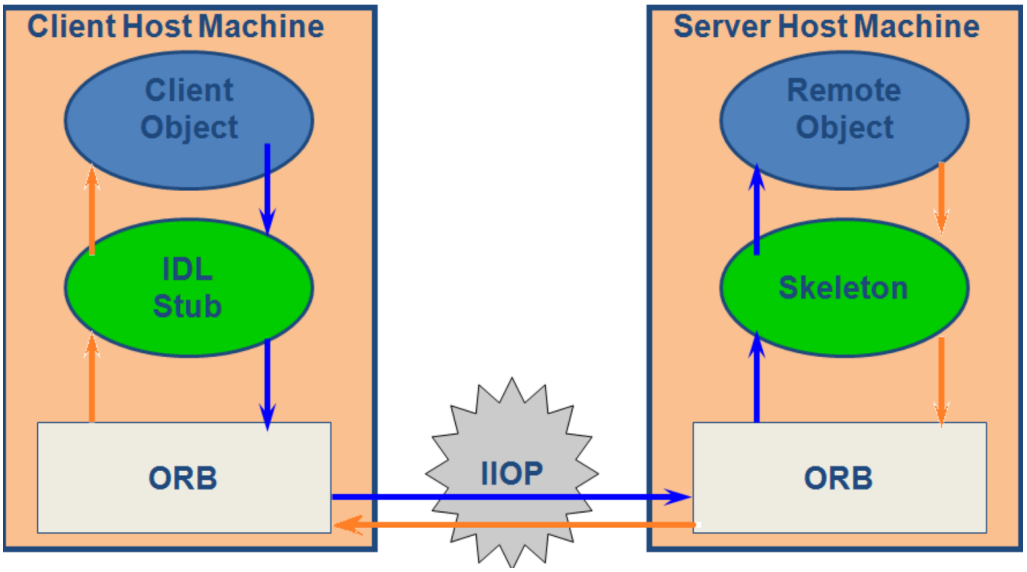


COBRA Architecture

- Object Request Broker is an Object Manager in CORBA.
- It is present on the client side as well as server side (allows agents to act as both clients and servers of remote objects).
- On client side the ORB is responsible for
    - accepting requests for a remote object
    - finding implementation of the object
    - accepting client-side reference to the remote object(converted to a language specific form, e.g., a Java stub object)
    - routing client method calls through the object reference to the object implementation.
- On server side the ORB
    - lets object servers register new objects
    - receives requests from the client ORB
    - uses object's skeleton interface to invoke object's activation method
    - creates references for new objects and sends it back to the client.
- Between the ORBs, Internet Inter-ORB Protocol is used for 67 communications.

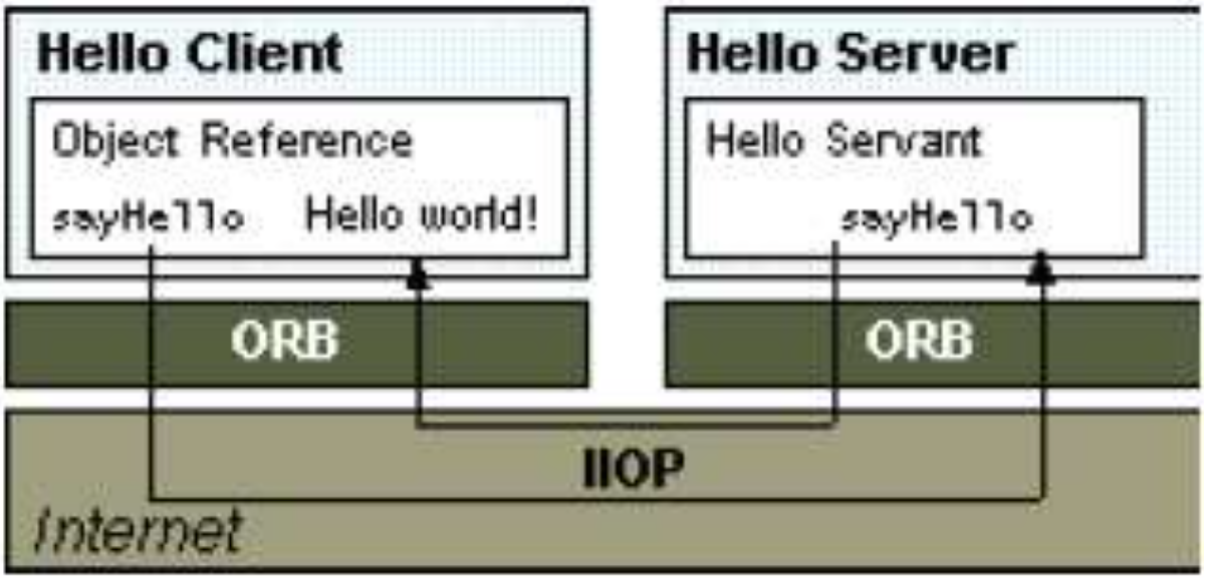
IDL

- IDL is Interface Definition Language which defines protocol to access objects.
- Stub lives on a client and pretends to be a remote object.
- Skeleton lives on the server , receives requests from stub, talks to true remote objects and delivers responses to stub.



JAVA IDL - Using COBRA from JAVA

- Java – IDL is a technology for distributed objects -- that is, objects interacting on different platforms across a network.
- Translates IDL concepts into Java Language Constructs.
- Java IDL supports distributed objects written entirely in the Java
- programming language.
- Java IDL enables objects to interact regardless of whether they're written in the Java programming language or another language such as C, C++.
- This is possible because Java IDL is based on the Common Object Request Brokerage Architecture (CORBA), an industry-standard distributed object model.
- Each language that supports CORBA has its own IDL mapping--and as its name implies, Java IDL supports the mapping for Java.
- To support interaction between objects in separate programs, Java IDL provides an Object Request Broker, or ORB.
- The ORB is a class library that enables low-level communication between Java IDL applications and other CORBA-compliant applications.
- On the client side, the application includes a reference for the remote object. The object reference has a stub method, which is a stand-in for the method being called remotely.
- The stub is actually wired into the ORB, so that calling it invokes the ORB's connection capabilities, which forwards the invocation to the server.



- On the server side, the ORB uses skeleton code to translate the remote invocation into a method call on the local object.

- The skeleton translates the call and any parameters to their implementation-specific format and calls the method being invoked. When the method returns, the skeleton code transforms results or errors, and sends them back to the client via the ORBs. Between the ORBs, communication proceeds by means of IIOP.

Building a COBRA Distributed Application using JAVA IDL

1. **Define the remote constructor -** Define the interface for the remote object using Interface Definition Language (IDL).
2. **Compile the remote interface -** The interface definition file generates the Java version of the interface, as well as the class code files for the stubs and skeletons that enable your applications to hook into the ORB.
3. **Implement the Server -** Use the skeletons it generates to put together your server application. In addition to implementing the methods of the remote interface, the server code includes a mechanism to start the ORB and wait for invocation from a remote client.

Conclusion

Thus in this assignment we've learnt about COBRA and successfully implemented a distributed application using JAVA IDL.