Module 17

Client-Server Software Development

Lesson 42

CORBA and COM/DCOM

Specific Instructional Objectives

At the end of this lesson the student would be able to:

- Explain what Common Object Request Broker Architecture (CORBA) is.
- Explain CORBA reference model.
- Explain CORBA architecture.
- Identify the functions of Object Request Broker (ORB).
- Identify the commercial ORBs.
- Explain what is stub.
- Explain Dynamic Invocation Interface (DII) in CORBA.
- Explain what is Component Object Model (COM).
- Explain what is Distributed Component Object Model (DCOM).
- Explain Inter-ORB communication.
- Identify the features of General Inter-ORB Protocol (GIOP).
- Differentiate between CORBA based development and COM/DCOM development.

Common Object Request Broker Architecture (CORBA)

The Common Object Request Broker Architecture (CORBA) is a specification of a standard architecture for middleware.

Using a CORBA implementation, a client can transparently invoke a method on a server object, which can be on the same machine or across a network. The middleware takes the call, and is responsible for finding an object that can implement the request, passing it the parameters, invoking its method, and returning the results of the invocation. The client does not have to be aware of where the object is located, its programming language, its operating system or any other aspects that are not part of an object's interface.

CORBA reference model

The CORBA reference model called Object Management Architecture (OMA) is shown in fig. 17.5. The OMA is itself a specification (actually, a collection of related specifications) that defines a broad range of services for building distributed client-server applications. Many services one might expect to find in a middleware product such as CORBA (e.g., naming, transaction, and asynchronous event management services) are actually specified as services in the OMA.

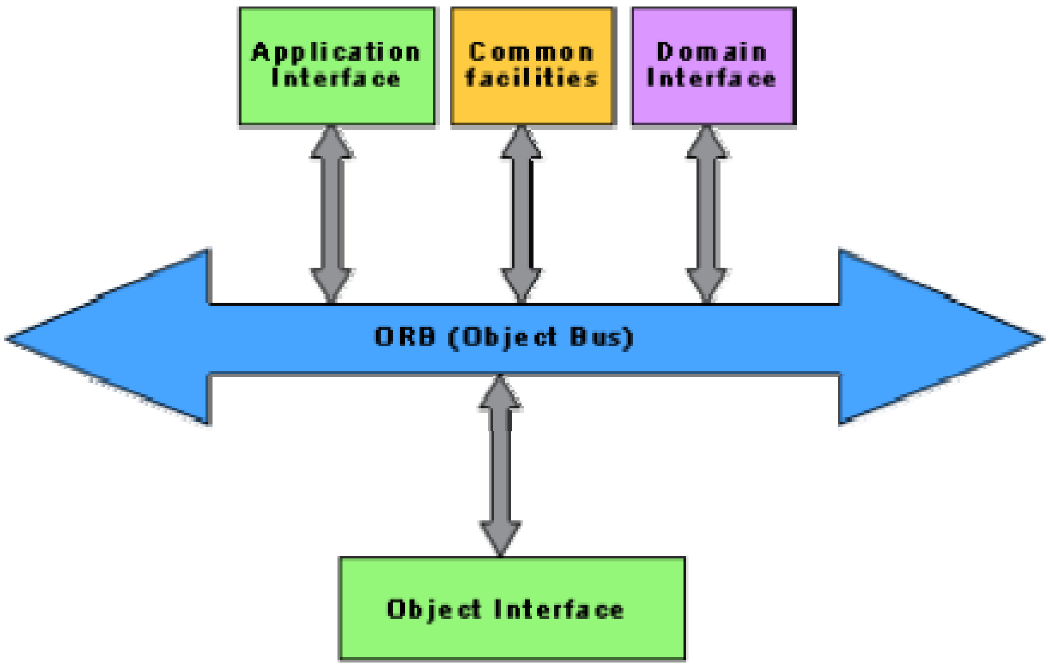

Fig. 17.5: Object Management Architecture (OMA)

Different components communicate using ORB. ORB is also known as the object bus. An example of application interface is distributed document facility. In a domain interface, it can have domain dependent services, for example, manufacturing domain. Object interface has some domain independent services:

- **Naming Service:** Naming service is also called white page service. Using naming service server-name can be searched and it's location or address found out.

- **Trading Service:** Trading service is also called yellow page service. Using trading service a specific service can be searched. This is akin to searching a service such as automobile repair shop in a yellow page directory.

There can be other services which can be provided by object interfaces such as security services, life-cycle services and so on.

Explain CORBA architecture.

Fig. 17.6 depicts the basic components and interfaces defined by CORBA.

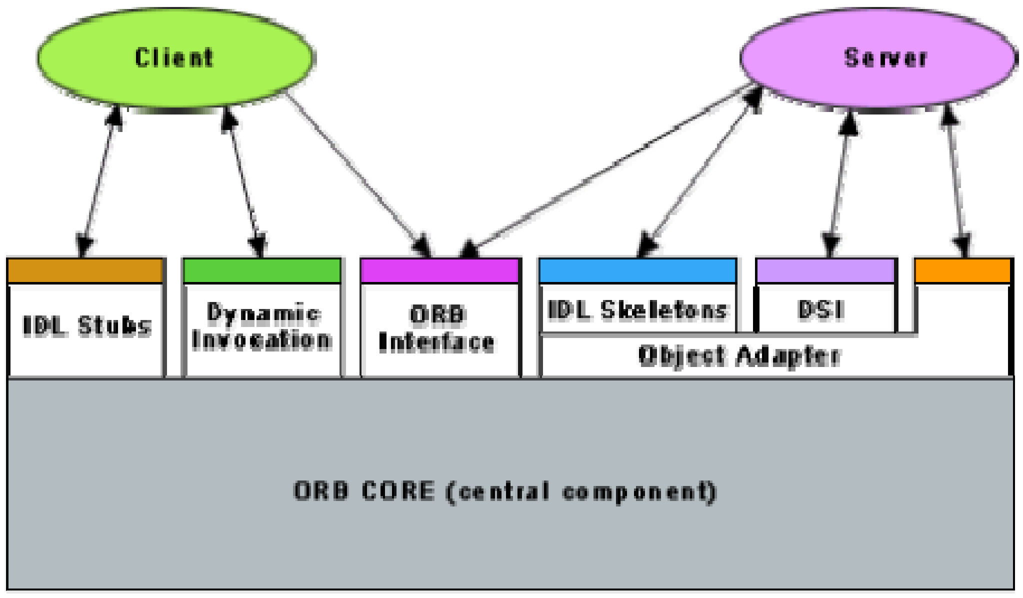

Fig. 17.6: CORBA Architecture

Using a CORBA implementation clients can communicate to the server in two ways:

- Stub

- Dynamic Invocation Interface (DII)

A client can invoke the server services through a Stub and the server gets the requests through the Skeleton. Alternatively, a client can avail services from server through Dynamic Invocation Interface (DII).

Functions of Object Request Broker (ORB)

ORB is the central component of the CORBA architecture. The main responsibility of ORB is to transmit the client request to the server and get the response back to the client. ORB abstracts out many procedures involved in service invocation and makes service invocation by client seamless and easy. The main responsibilities of ORB are the following:

- Server location
- Server state management
- Communication between clients and servers

An object request broker provides directory services and helps establish connections between clients and these services [CORBA 96, Steinke 95]. Fig. 17.7 illustrates some of the key ideas.

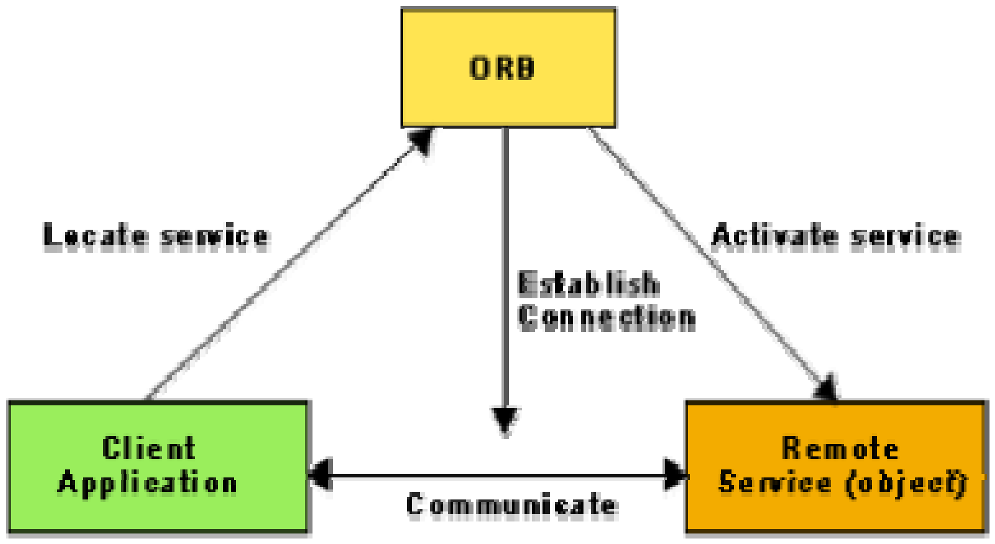

Fig. 17.7: Object Request Broker

The ORB must support a large number of functions in order to operate consistently and effectively. The ORB implements much of these functionality as pluggable modules to simplify the design and implementation of ORB and to make it efficient.

ORB allows objects to hide their implementation details from clients. This can include programming language, operating system, host hardware, and object location.

Commercial ORBs

There are several ORBs that are commercially available.

- **Visigenic:** This is probably most popular one. Netscape browser supports Visigenic. CORBA applications can be run using Netscape web browser. In other words, Netscape browser can act as client for CORBA applications. Netscape is extremely popular and there are several millions of copies installed on desktops across the world.
- **IONa**
- **Orbix**
- **Java IDL**

Steps to develop application in CORBA

Service can be invoked by a client through either stub or Dynamic Invocation Interface (DII). Before developing a client-server application, the problem is split into two parts: client part and the server part. Next the exact client and server interfaces are determined.

- To specify an interface, IDL (Interface Definition Language) is used.

IDL is very similar to C++ and Java except that it has no executable statements. Using IDL only data interface between clients and servers can be defined. It supports inheritance so that interfaces can be reused. It also supports exception.

After the client-server interface is specified in IDL an IDL compiler is used to compile the IDL specification. Depending on whether the target language in which the application is to be developed is Java, C++, C, etc. IDL2Java, IDL2C++, IDL2C etc. can be used appropriately. When the IDL specification is compiled, it generates the skeletal code for stub and skeleton as shown in fig. 17.8. The stub and skeleton contain interface definition, but the methods (services) are to be filled in by the programmers.

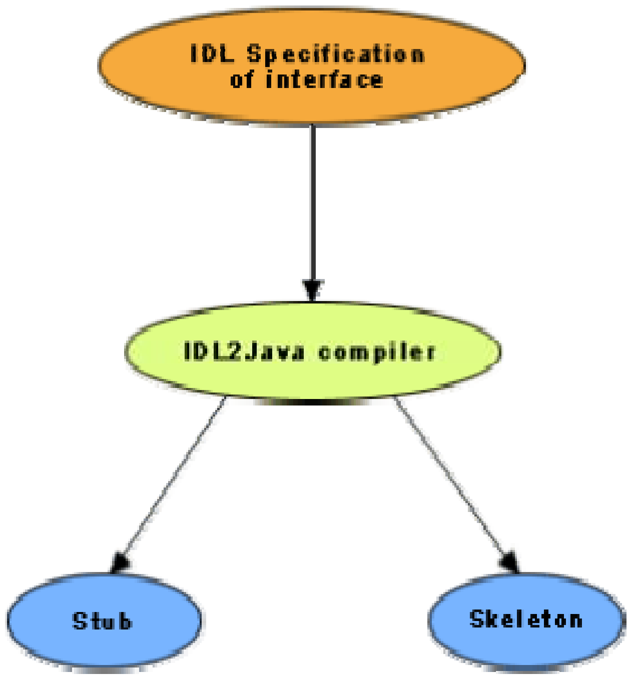

Fig. 17.8: Creation of stub and skeleton using IDL

Service invocation by client through stub is suitable when the interface between the client and server is fixed and it does not change with time. If Interface is known before starting to develop client and the server parts then stubs can effectively be used for service invocation.

The stub part will reside in the client computer and that would basically act as a proxy for the server which may reside in the remote computer. That is the reason why stub is also known as a proxy.

Dynamic Invocation Interface (DII) in CORBA

Service invocation through Dynamic Invocation Interface (DII) transparently accesses the interface repository (IR). When an object gets created, it registers information about itself with IR. DII gets the relevant information from the IR and lets the client know about the interface being used. DII is inefficient as compared to stubs.

Component Object Model (COM)

The main idea in the Component Object Model (COM) is that:

- Different vendors can sell binary components.
- Application can be developed by integrating off-the-shelf and proprietary components.

COM runs on a single computer. The concepts used are very similar to CORBA. The components are known as binary objects. These can be generated using languages such as Visual Basic, Delphi, Visual C++ etc. These languages have the necessary features to create COM components. COM components are binary objects and they exist in the form of .exe or .dll (dynamic link library). .exe COM components have separate existence. But .dll COM components are in-process servers. So they get linked to a process. For example, ActiveX is a dll type server. ActiveX can get loaded on the client-side using the dll.

Distributed Component Object Model (DCOM)

Distributed Component Object Model (DCOM) is the extension of the Component Object Model (COM). The restriction that clients and servers reside in the same computer is released here. So, DCOM and CORBA both operate on networked computers.

Here development is much easier as compared to CORBA development. Many of the things are transparent to the programmer such as proxy generation, service invocation etc.

Inter-ORB communication

How does ORB do in service invocation when different components exist in different LANS? The answer is Inter-ORB Communication. CORBA 1.0 did not permit Inter-ORB Communication. CORBA 2.0 removes the shortcoming. CORBA 2.0 defines general interoperability standard.

In bridge-based interoperability, the bridge is basically responsible for translating ORB specification information from one ORB to other ORB. For example, one service is known as one reference number in one ORB but that service is known as different reference number in the other ORB.

Features of General Inter-ORB Protocol (GIOP)

The General Inter-ORB Protocol (GIOP) is an abstract meta-protocol. It specifies a standard transfer syntax (how data is represented as bits and bytes) and a set of message formats for object requests. The GIOP is designed to work over many different transport protocols.

The features of GIOP are as follows:

- Designed to be simple, scalable and easy to implement. Every ORB must support GIOP mapped onto local transport.
- GIOP can be used almost any connection-oriented bytestream transport.
- Common Data Representation (CDR) encoding on data types.

One popular connection-oriented transport on which GIOP is implemented is TCP/IP. So the implementation of GIOP on TCP/IP is known as IIOP (Internet Inter-ORB Protocol). It is very popular and frequently used.

CORBA vs. COM/DCOM

If it is the case that all applications reside on PCs and are to run fully on Microsoft platforms then it will be better to use COM/DCOM because development would be much easier here.

- If an application is to be developed for a heterogeneous environment then it will be better to use CORBA.

- Microsoft is very strong on desktop applications i.e. GUI-based applications. Whereas, CORBA-based development is stronger on server side. Java Beans promise to overcome the shortcoming on desktop side i.e. the client part.

The following questions have been designed to test the objectives identified for this module:

1. Briefly specify the reasons for the popularity of client-server software development.

2. What are the advantages of client-server software development?

3. What are the disadvantages of client-server software development?

4. Can we say, “Two-tier architecture is the practical solution”? If so, then give the reasons. Briefly specify the limitations of two-tier architecture.

5. What are the functions of a middleware in a three-tier architecture? Mention two popular middleware standards.

6. Briefly specify the domain independent services provided by object interface in the Object Management Architecture (OMA).

7. What are the things that Object Request Broker (ORB) abstracts out? Mention the functions of ORB. Mention some available ORBs.

8. How does Dynamic Invocation Interface (DII) know what format data or what exact data required by the server for providing the service and how does the client recognize the data?

9. What is GIOP? What are the features of General Inter-ORB Protocol (GIOP)?

10. Compare between CORBA based development with COM/DCOM development.

Mark all options which are true.

1. What are the reasons for the recent popularity of the client-server style of software development?

   ☐ computers have become small, decentralized and cheap
   ☐ networking has become affordable, reliable, and efficient
   ☐ client-server systems divide up the work of computing among many separate machines
   ☐ all of the above

2. Which of the following functions are performed by middleware?

- ☐ it can identify the server from either its id or its service type
- ☐ it knows client protocols and server protocols
- ☐ it can deliver client-request to the server and server-response to the client
- ☐ all of the above

3. Which of the following domain independent services are provided by object interface in an Object Management Architecture (OMA)?

- ☐ naming service
- ☐ trading service
- ☐ security service
- ☐ all of the above

4. Which of the following functions does Object Request Broker (ORB) perform?

- ☐ to transmit the client request to the server and get the response back to the client
- ☐ location and possible activation of remote objects
- ☐ interface definition
- ☐ all of the above

5. Before developing client-server application in CORBA, interface between the client part and the server must specified using

- ☐ Interface Definition Language
- ☐ Dynamic Invocation Interface
- ☐ ORB
- ☐ none of the above

6. In CORBA if the server interface would not change with time, then it is more efficient to use

- ☐ stubs and skeletons
- ☐ DSI and DII
- ☐ none of the above

7. In CORBA Dynamic Service Invocation requires

- ☐ previous knowledge of the interface between the client and the server part

☐ we do not need to know the interface between the client and the server part
☐ none of the above

8. What are the properties General Inter-ORB Protocol (GIOP) hold?

☐ scalable
☐ easy to implement
☐ can be used any connection-oriented bytestream transport
☐ all of the above

9. If some applications run entirely on Microsoft platforms then it will be better to use

☐ CORBA
☐ COM/DCOM
☐ all of the above

10. If clients and servers run on heterogeneous platforms then it will be better to use

☐ CORBA
☐ COM/DCOM
☐ all of the above

Mark the following statements as either True or False. Justify your answer.

1. Client-server software development is synonymous with component-based development.

2. Fault-tolerance is more difficult to provide in a monolithic application compared to its client-server implementation.

3. Client-server based software development is more secure than a monolithic software.

4. Two-tier client-server architecture is a practical solution for distributed computing applications.

5. The object adapter component in CORBA is responsible for translating the client data formats into server data formats and vice versa.

6. The term marshalling in CORBA refers to encryption of client data for added security.

7. CORBA is the name of a software product that facilitates development of client-server solutions.

8. A CORBA-based client-server solution is constrained to run on a single Local Area Network (LAN).

9. Using Internet Inter-ORB Protocol (IIOP), a web browser such as Netscape can serve as a CORBA client.