**Distributed Applications:**

Distributed object system is a technology that combines networking and object oriented programming.

A distributed system contains a number of nodes that perform computations. A node may be a PC, a mainframe computer or any other device. Nodes of distributed system are scattered. The node in use is the local node and any other node is the remote node. Network makes distributed computing possible.

**Distributed Computing Technologies:**

**RMI vs. CORBA**

• RMI

– Java-only distributed object model

– relatively simple to use

• especially if already familiar with Java

– software needed is free

– new so lacks maturity

• many RMI designers participated in the design of CORBA

• CORBA

– language independent distributed object model

• but nearly all CORBA development is done in C++

– relatively complex to use

– software needed to use is expensive

**RMI Architecture:**

RMI architecture consists of four layers:

1. Application Layer ii. Proxy layer iii. Remote Reference Layer iv. Transport Layer
2. **Application Layer:** This layer contains the actual implementation of the client and server applications. It is in this layer that high level calls are made to access and export remote objects.
3. **Proxy Layer:** This layer contains the client stub objects. The application layer communicates with this proxy layer directly. All calls to remote objects and marshalling(converting remote method arguments to a stream of bytes in a stub )of parameters and return objects are done through these proxies.
4. **Remote Reference Layer:** It handles packaging of method call and its parameters and its return values for transport over the network. The remote reference layer uses a server-side and client-side component to communicate via the network layer.
5. **Transport Layer:** The transport layer sets up connections, manages existing connections and handles remote objects residing in its address space.

**Remote Method Invocation**

Java’s *Remote Method Invocation (RMI)* facility lets you make method calls on objects that

exist outside your JVM. RMI hides many of the details that you have to take care of when you

use object streams, including knowledge of the server port number, socket and stream maintenance, and the need for a command class.

**CORBA(Common Object Request Broker Architecture)**

CORBA, which stands for Common Object Request Broker Architecture, is an industry standard developed by the OMG (Object Management Group) a consortium of more than 700 companies; to aid in distributed

objects programming. CORBA is just a *specification* for creating and using distributed objects. CORBA is *not* a programming language.

**CORBA Architecture**

CORBA is composed of five major components: ORB(Object Request Broker), IDL(Interface Definition Language), Dynamic invocation interface (DII), interface repositories (IR), and object adapters (OA).

**Object Request Broker:** The CORBA specification must have software to implement it. The software that implements the CORBA specification is called the ORB. The ORB, which is the heart of CORBA, is responsible

for all the mechanisms required to perform these tasks:

* Find the object implementation for the request.
* Prepare the object implementation to receive the request.
* Communicate the data making up the request.

**Interface definition language**

As with RMI, CORBA objects are to be specified with interfaces, which are the contract between the client and server. In CORBA’s case, however, interfaces are specified in the special definition language IDL.

The IDL defines the types of objects by defining their interfaces. An interface consists of a set of named operations and the parameters to those operations. Note that IDL is used to describe interfaces only, not implementations. Despite the fact that IDL syntax is similar to C++ and Java, IDL is not a programming language.

**Dynamic invocation interface**

Invoking operations can be done through either static or dynamic interfaces. Static invocation interfaces are determined at compile time, and they are presented to the client using stubs. The DII, on the other hand, allows client applications to use server objects without knowing the type of those objects at compile time. It allows a client to obtain an instance of a CORBA object and make invocations on that object by dynamically constructing requests. DII uses the interface repository to validate and retrieve the signature of the operation on which a request is made. CORBA supports both the dynamic and the static invocation interfaces.

**Interface Repository**

The IR provides another way to specify the interfaces to objects. Interfaces can be added to the interface repository service. Using the IR, a client should be able to locate an object that is unknown at compile time, find information about its interface, then build a request to be forwarded through the ORB.

**Object adapters**

An object adapter is the primary way that an object implementation accesses services provided by the ORB. Such services include object reference generation and interpretation, method invocation, security of interactions, and object and implementation activation and deactivation.

**CORBA Services**

***Object naming service***This service supports a name-to-object association called a name binding, which is always defined relative to a naming context. Different names can be bound to an object in the same or different context at the same time. This service supports a number of operations, including bind, unbind, and lookup.

• ***Event service*** This service supports notification of events to interested objects. It provides asynchronous communications between cooperating, remote objects.

• ***Persistent object service***This service provides common interfaces for the mechanisms used for retaining and managing the persistent state of objects in a data-store independent manner. Of course, the object has the responsibility of managing its state, but it can use or delegate to this service for the actual work.

• ***Concurrency control service***This service defines how an object mediates simultaneous access by one or more clients, so that the objects it accesses remain consistent and coherent.