Assignment - 2

Core Distributed computing Technologies-

1) client/server.
II) CORBA
III) JAVA RMI
IV) Microsoft DCOM

1) client / server-
this is the early age technology which separates the roles of computers as client and server. this technology its still powerful and popular amongst the network technologies to establish communication between two or more machines.

The early stage of this technology used two-tier business Applications.
In this model, the first (upper) tier handles the presentation and bussiness logic of the user application (client), and the second / lower tier handles the application organization and its data storage (server).

- In general, the server is a database server that is mainly responsible for the organization and retrival of data. The application client handles the user interaction through variety of graphical user interface of the application.

for example- the client - server model has been widely used in Enterprise

Resource Planning (ERP), billing, and Inventory application systems, banking etc.

2) CORBA (common object Request Broker Architecture)

→ It is an industry wide, open standard initiative.

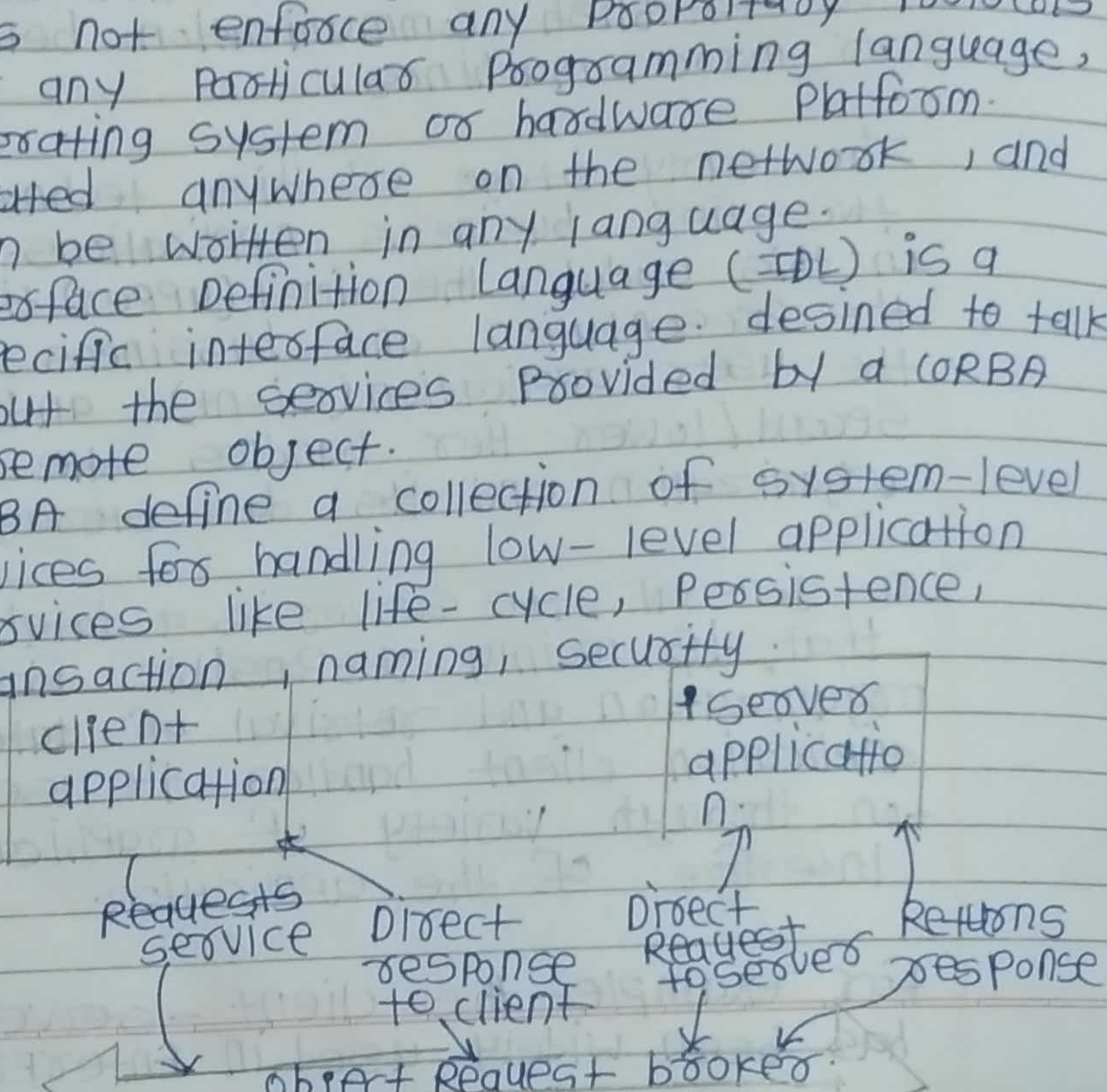
It is developed by the object management Group (OMG)

It is developed to enable distributed computing that supports a wide range of applications environments.

It Provide an object-oriented solution that does not enforce any proprietary Protocols or any Particular Programming language, operating system or hardware Platform. located anywhere on the network, and can be written in any language.

- Interface Definition Language (IDL) is a specific interface language. desined to talk about the services Provided by a CORBA remote object.

- CORBA define a collection of system-level services for handling low-level application services like life-cycle, Persistence, transaction, naming, security.



The application written in C, C++ and Java can be integrated through IDL bindings.

The CORBA architecture is composed of the following components:

1) IDL : CORBA use IDL contracts to specify the application boundaries and to establish interfaces with its clients. The IDL Provides a mechanism by which the distributed application component's interfaces inherited classes, events, attributes and exceptions can be specified.

ii) ORB - ~~I~~ acts as the object bus or the bridge, Providing the communication infrastruc ture to send and receive request/ responses from the client and server. It establishes the foundation for the distribut -ed application objects, achieving interoperability in a heterogeneous environment.

3) Java RMI ~~Method~~ (Remote method invocation)
At the name specifies Java invented RMI APIS for communicating methods on any machine remotely.
- This is pure - Java solution for handling distributed communication.
- Through RMI object Running on a client computer can invoke methods on an object present at server.

Working of RMI -
There are two special objects oriented to establish communication between client and server.
i) Stub object (client side)
ii) Skeleton object (server side)

i) Stub object - The stub object on the client machine builds an information block and sends this information to the server.
The block consists of -
a. An identifier of the remote object to be used.
b. Method name which is to be invoked.
c. Parameters to the remote JVM.

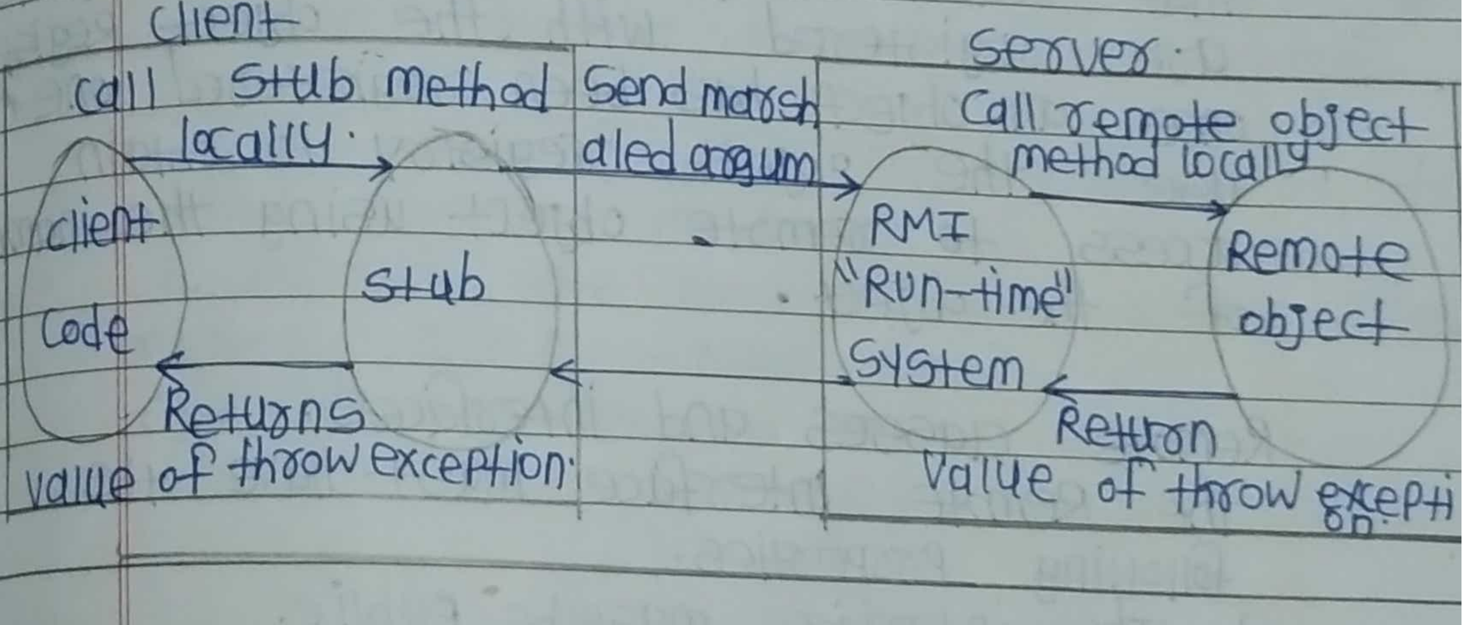
ii) Skeleton object - The skeleton object passe the request from the stub object to the remote object it works as:

a. It calls the desired method on the real object present on the server.

b. It forwards the Parameters received from the stub object to the method.

Steps to implement interface.
I. Defining a remote interface.
II implementing the remote interface.
III) creating Stub and Skeleton objects from the implementation class using (rmi compiler)
IV) ~~Start~~ the rmiregistry.
V) create and execute the server application Program.
VI) create and execute the client application Program.

· ~~Java~~ Internet



client
call Stub method locally.
Send marshaled argum
client code
Stub
Returns value of throw exception.
Server.
Call remote object method locally
RMI "Run-time" System
Remote object
Return Value of throw exception

Java RMI is mechanism that allows one ~~to~~ invoke a method on an object that exists in another ~~class~~ address space.
- The other address space should be on the same machine or a different one.
- The RMI mechanism is basically an object oriented RPC mechanism.

There are three process that participate in supporting remote method invocation.
a. The client is the process that is invoking a method on a remote project.
b The server is the process that owns the remote object. the remote object is an ordinary object in the address space of the server process.
c. The object Registry is a name server that releates objects with names. objects are registered with the object Registry. once an object has been registered one can use the object registry to obtain access to remote object using the name of the object.

Remote classes and interfaces.
The Remote Interface must have the following properties.
1) The interface must be public.
2) The interface must extend the interface java.rmi.Remote.
3) Every method in the interfaces must declare that it throws java.rmi.RemoteException other exceptions may also be thrown.

The Remote class itself has the following properties.

i) It must implement a Remote interface.
ii) It can have methods that are not in its remote interface. These can only be invoked locally.

* The skeleton interface -
• The interface skeletonis used solely by the implementation of skeletons generated by the rmic computer. A skeleton for a remote object is a server side entity that dispatches calls to the actual remote object implementation.
- The skeleton is responsible for dispatching the call to the actual remote object implementation. When a skeleton receives an incoming method invocation it does the following.
a. unmarshals (reads) the parameters for the remote method.
b. invokes the method on the actual remote object implementation, and.
c. marshals (writes and transmits) the result (return value or exception) to the caller.

Stub ~~object~~ -
are surrogates that support exactly the
same set of remote ~~object~~ interfaces
defined by the actual implementation
of a remote object.

- A stub for a remote object acts as
a client's local representative or
proxy for the ~~$~~ remote object
- The caller invokes a method on the local stub
which is responsible for carrying out the
method call on the remote object.
- In RMI a stub for a remote object
implementation~~s~~ the same set of remote
interfaces that a remote object implem

* The Registry interface -

• The REGISTRY _ PORT is the default port
of the registry

• the lookup method returns the ~~create~~
remote object bound to the specified
name.

• The bind method associates the name
with the remote object, obj

• The unbind method removes the
binding between the name & the
remote object, obj.

• the list method returns an array of string
containing a snapshot of the names bound
in the registry

4) Microsoft DCOM (Distributed component object model) -
It is a remote Protocol designed by Microsoft to invoke RPCS. It consists of a set of extensions layered on the Microsoft Remote Procedure call Extensions.

High level Application/ Protocol	
DCOM	
RPCS.	

(DCOM Protocol Stack): Higher level applications use the DCOM client to obtain object references or make ORPC Calls on the object. The DCOM client uses the Remote Procedure call Protocol Extensions to communicate with the object server.
The object server constitutes an object resolver service and one or more object exporters. objects are contained in object exporters.
- DCOM is language. is language. & Platform independent.
DCOM is a binary standard.
DCOM Provides the ability to use and reuse component dynamically, without recompiling or Platform & language neutral Principal.