DIS Important Concepts. (1)

Scalability of a System Can be measured alog with atleast 3 dimensions;

1. Size: We can add more users and resources to the System.

2. Geographically Scalable syestem: is one in which the users & resources may lie far apart.

3. Administratively scalable: easy to manage even if it spans many independent administrative organizations.

Scanned with CamScanner

Scaling Techniques :-

① Hiding Communication Latencies

achieves geographical scalability.

Basic idea is "to avoid waiting for responses to remote service requests as much as possible.

ex: Server may check for syntatic error before accepting an entry. Client side Validations are checked before sending data to server side.

② Distribution :- Involves taking a component, splitting it into smaller parts and subsequently spreading those parts across the system.

ex:- DNS (Domain Name system) name space is hierarchially organized into a tree of domains and divided into nonoverlapping zones.

③ Replication :- It increases availability and also helps to balance the load between components leading to better Performance.

Scanned with CamScanner

Distributed Objects Based Systems

Key Points & Concepts

1. Distributed Objects Plays a key role in establishing Distribution Transparency.

Architechure

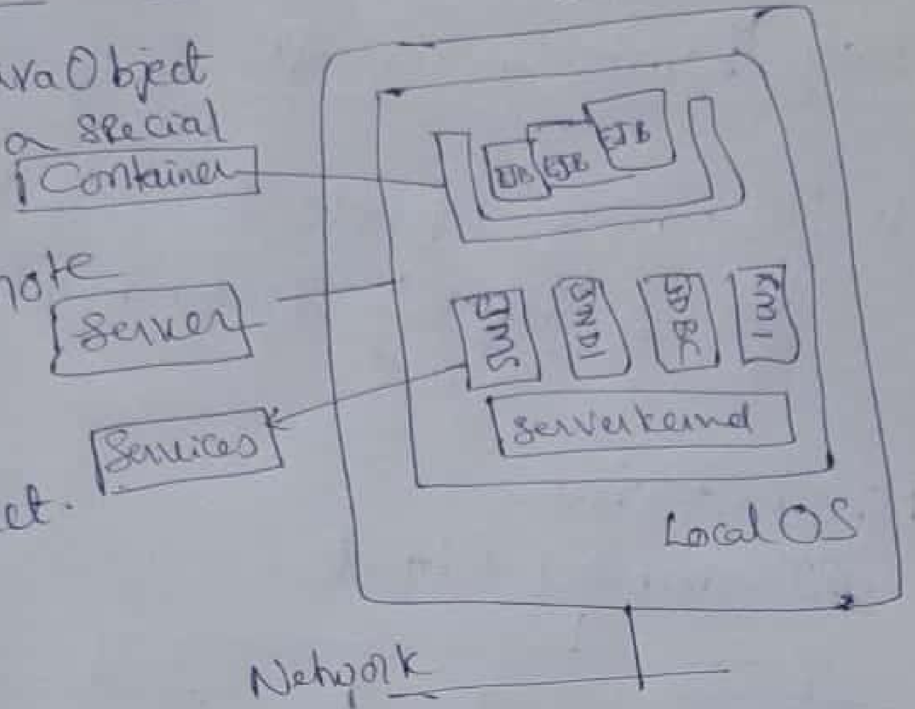
Distributed Objects

Compile Time	Runtime	Persistent	Transient
• easier to Build Distributed apps. Drawback: • Depends on Particular Programming Language •	Independent of Programming language.	• object exists even if it currently not Contained in the address space of any server process.	Object that exist exists only as long as the server that is hosting the Object.

Example:

Enterprise Java Beans [EJB]

- EJB is a Java Object
- Hosted by a special Container server
- Offering remote Clients to invoke that Object.



General Architechure of an EJB Server.

EJB Types

Stateless Session Bean

- is a Trasient Object invoked once, does its work, & then discarded.

ex: Service that lists Top Ranked books (SQL query)

Stateful Session Bean

- Maintains client related state

ex: Implementing an electronic Shopping Cart.

Entity Beans

- long lived Persistent Object
- will be stored in a database

ex: Record Customer info in e-Commerce apps.

Message Driven Beans

- used to program objects that should react to incoming messages

ex: Publish Subscribe Systems

Unit III Part II
Distribute Object Based Systems

Concepts to learn

Architechure

Processes

Communication

Naming

Synchronization

Consisteucy & Replication.

Fault Tolerance

Security.

Security

Security for Distributed Objects evolves arround the idea of secure method invocations.

Two Issues:-

1) Is Caller Invoking Correct Object
(secure object Binding)

2) Is caller allowed to invoke that method.
(secure method Invocation

Mechanisms deployed in Globe

1. Every object has an Public/Private key pair reffered as object key
2. Every Replica has an Replica key
3. key Pair is generated by Object server
4. each user to have a unique Public/Private key Pair known as the user key.
5. These keys are used to set various access rights in the form of Certificates.
6. 3 Types of certificates:-
   - User Certificate: is associated with a specific user, specifies exactly which methods that user is allowed to invoke.
   - Replica Certificate: specifies for a given replica server, which method is allowed to invoke or execute.
   - Administrative Certificate:- Can be used by any authorized entity to issue User and replica certificates.

Secure method Invocation in Globe.

: Draw Diagram Fig 10-22

1. Application Issues an Invocation request
2. Control Subobject check user Permission
3. Request marshalled & Passed on
4. Replication Sub Object requests middleware to set up a secure channel to a suitable replica
5. Securityobject first initiates replica lookup
6. Suitable replica When Found, establish Secure channel. Control given to Replica Subobject
7. Request now passed to Communication Subobject
8. Subobject encrypts & signs the request So that it Can pass through the Channel.
9. After its receipt the request is decrypted & Authenticated.
10. request is passed to Server side replication Subobject
11. Authorization: User Certificate Verification.
12. request unmarshalled.
13. Finally operation executed.

Scanned with CamScanner

Security Continued...

Security for Remote Objects:-

1) Basic idea is that a developer of a remote object also develops Proxy & subsequently registers Proxy with a directory service.

2) When client looking for that object it contacts directory service, retrieve the Proxy and install it.

2) Problems with this above aproach:-

a) Directory service is hijacked then attacker returns bogus proxy to the client.

b) client has no way to authenticate server.

Solutions

a) Client verifies the Origin of Proxy.

b) Proxy in turn authenticates Object with server authentication using TLS

Synchronization

(1) In Object based Distributed Systems it is important to know where and when Synchronization takes Place. Object Server location for Synchronization

Problem: or

(2) Implementation details are hidden behind interfaces may Cause Problems:

"When a Process invokes a (remote) object it has no knowledge whether that invocation will lead to invoking other object".

Consequence: "If an object is Protected against Concurrent accesses We may have Cascading set of locks that the invoking Process is unaware of" — Draw Fig 10-14 (a).

(3) No Problem: with Files/tables Protected by locks "Controlflow is visible to Process using those Resources. Draw Fig 10-14 (b).

Consequence: " Process Can give up locks when deadlock occurs".

(4) (P.T.O)

(4) Key Points of Synchronization

* Object Server is the location for Synchronization
* if Multiple invocation requests for same object arrive then server can decide to serialize those Objects and also keeps
* lock on a object when it needs to do remote invocation.
* Maintaining Locking Can be at client side, server side.
* Problems: with Locking at client side "consequence is that we need to synchronize different client at different machines".
* Alternative approach to above Problem:-
To allow blocking only at the server.
only Problem if client crashes while its invocation being handled by server.
* Solution in Java RMI
"Restrict blocking on Remote Objects only to the Proxes"
Meaning: "Threads in same Process will be Prevented from Concurrently accessing the same remote object"

Synchronization of Distributed Object based systems. (Pg 10)

Diagrams for Easy Understanding.

Fig1

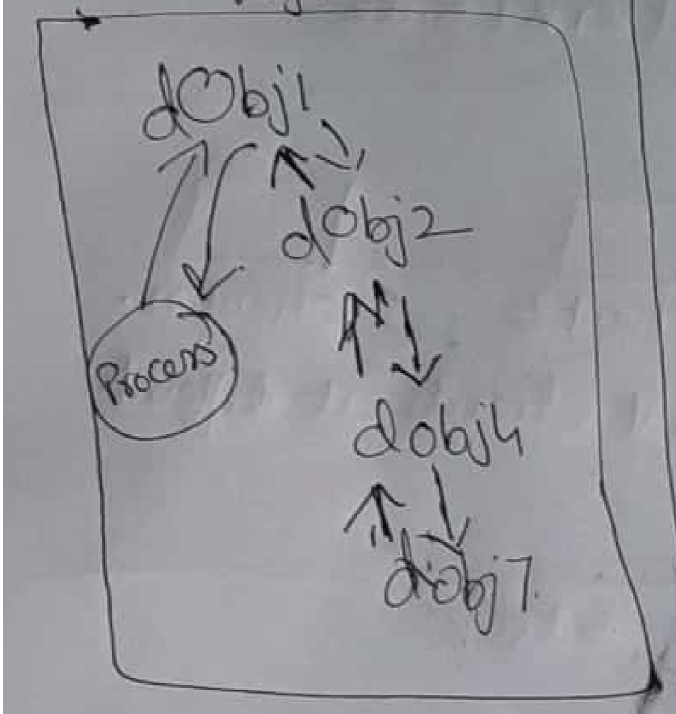


Fig 1 A Process invoking Remote Object Unaware about invoking other Objects

Fig2

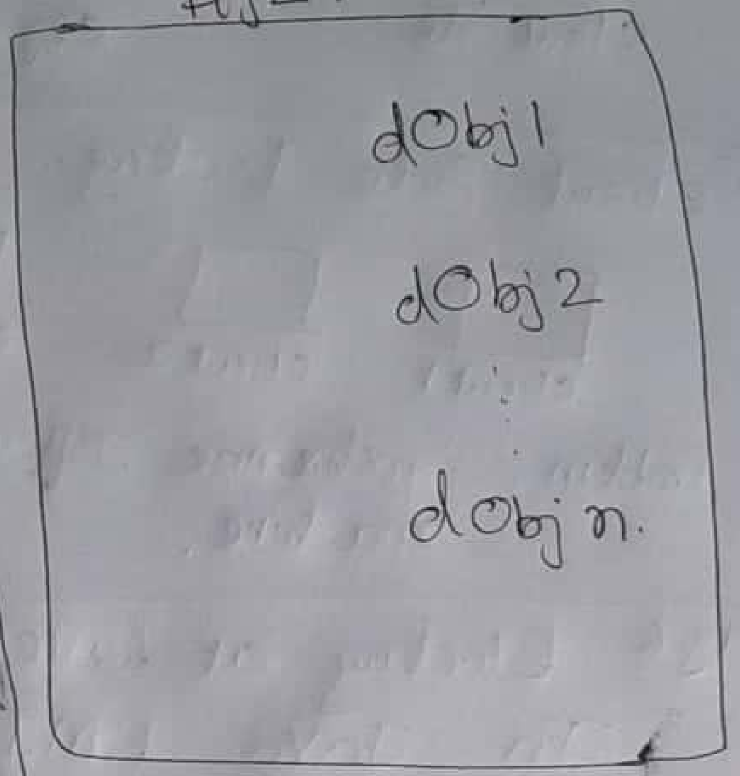
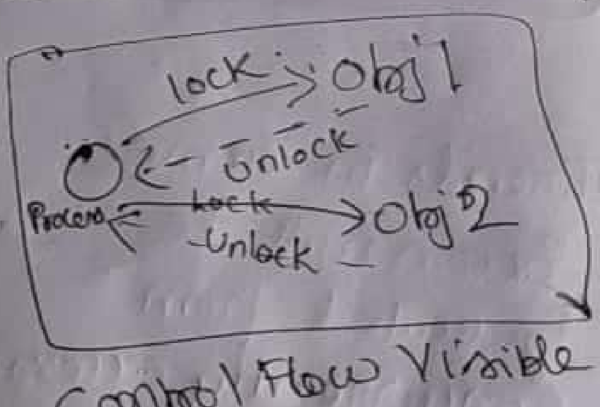


Fig 2 A Object Server Holds distributed Objects.

Fig3



Control Flow Visible to Process using those resources Can lock + unlock

Scanned with CamScanner

Fig 4 Blocking / locking to achieve Synchronization

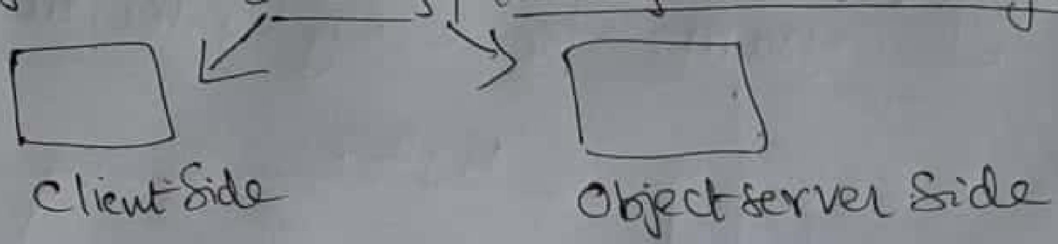
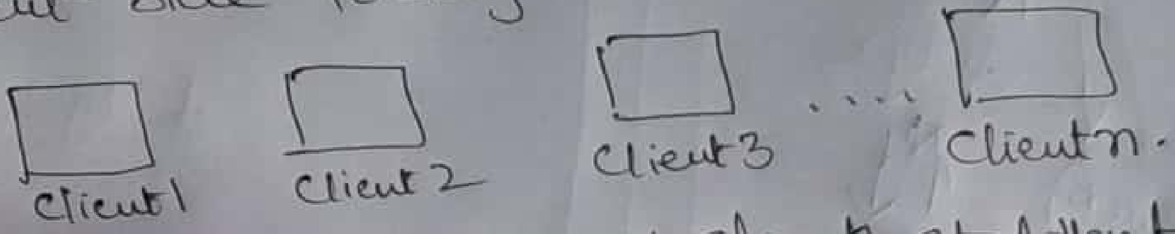
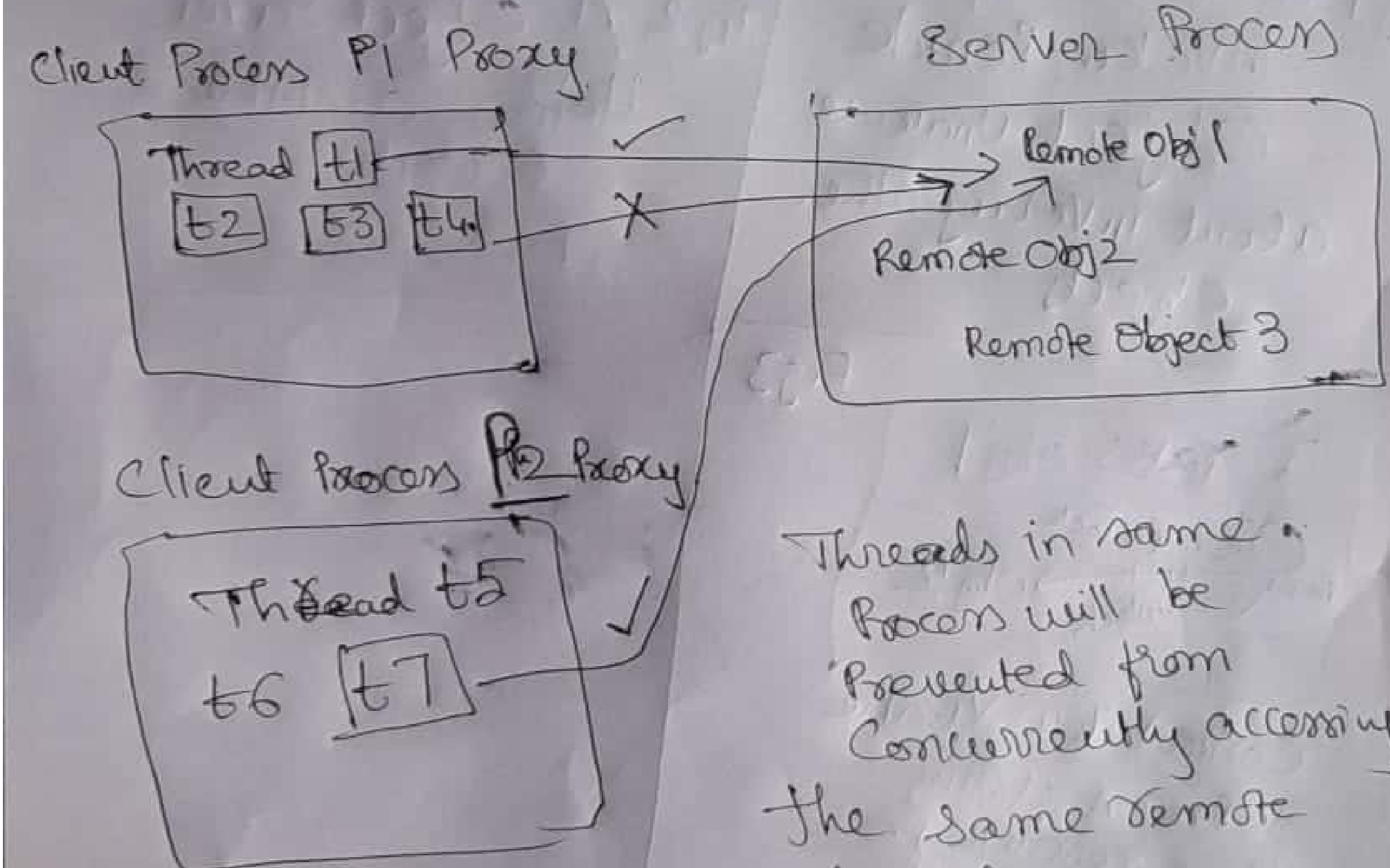


Fig 5 client side locking.



Problem: Synchronize Different clients at different machine.

Fig 6 Blocking at server side.
In Java RMI.



Threads in same Process will be Prevented from Concurrently accessing the same remote object.

Distributed Object Based Systems:- (Pg 12)

Naming in CORBA : key Points.

① Current CORBA Systems support language independent representation of an Object refference which is called "Interoperable Object Refference" (IOR).

② IOR contains all info to identify an Object

③ Draw + Refer Fig 10-11. IOR Contains

a) Repository Identifier → Used To Identify Interfaces.

b) Profile Id & Tagged Profile → Has (i) to (v) fields.

(i) IIOP Version → version of IIOP

(ii) Host → String identifying on which host object is located

(iii) Port → Port No to which object server listening for incoming requests.

(iv) Object key → Contains server specific info for demultiplexing incoming requests to the appropriate object.

(v) Component → Contains more info needed for Properly invoking referenced object ex: Security Info.

Distributed Object Based Systems :- (Pg13)

Naming in Globe :- Key Points

(1) Globally unique identifier (OID) is a 256 Bit String'

(2) Globe OIDs can be used only for comparing object references.

(3). ex:- If Process A+B are bound to a shared distributed object. If OIDs are same the A+B are considered to be bounded to same object.

(4) Globe OIDs cannot be used to directly contact an object.

(5) To locate an Object in Globe it is necessary to look up ~~object~~ a contact address for that Object in a location Servic

(6) 2 kinds of address supported currently:-

(a) Stacked address with 3 fields:-

(i) Protocol Identifier: A constant representing a known Protocol.

(ii) Protocol Address: A Protocol specific address

(iii) Implementation Handle: Refference to a file in a class repository.

Scanned with CamScanner

Stacked Address

Protocol Identifier ; ex ; TCP, UDP, IP

Protocol address ; ex ; TCP Port no
or IPV4 network
address.

Implementation handle : implementation
of Protocol
referenced as URL

b) Instance Address :- contains 2 fields.

(i) Implementation handle : Reference to a file
in a class repository

(ii) Initialization String : string that is used
to initialize an implementation

Note :

CORBA references Contain exact info where
to Contact an object.

Globe refferences require an addiitonal
lookup step to retrieve that information.

Scanned with CamScanner

Distributed Object Based Systems

Communication :- Key Points

① Mostly based on RPC.

② When a Process holds an object reference it must first bind to the refferenced Object. before invoking any of its methods.

③ Binding 2 types :-

(i) Implicit Binding : Client is directly allowed to invoke methods using only a reference to an Object.

```
Distributed object * object_ref.
ex: obj_ref -> method();
```

(ii) Explicit Binding :- Client should first Call a special function to bind to that Object then invoke its method.

```
ex:  Distributed_obj *obj-ref; //step1
     Local_Object   *obj-ptr; //step2
     obj-ref = ...; //step3
     obj-ptr = bind (obj-ref); //step4
     obj-ptr -> method(); //step5
```

step1 -> Declare a System wide Object Reference

step2 ⇒ Declare a Pointer to local objects

step3 -> Initialize the Refrence to a distributed object

step4 -> Explicitly bind & get Ptr to local Proxy

step5 -> Invoke a method on the local Proxy.

communication Continued....

(4) Implenetation of Object References:-

a) Simple Object reference:- Includes Network address of machine where actual object resides along with the an End Point identifying the server that manages the Object, Plus indication of which object.

Object Adapter Provides

Network address + End Point + Object name

Problem:
- Server's machine crashes
- After Recover Server assigned different endpoint. So
- All refferences become invalid.

Solution: End Point Table maintained by Local deamon Per machine.

Server needs to register its details with daemon server.

Better Solution to have a location Server that keeps track of machine where an Object Server is currently running.

Object Refference = Network Address of Location Server + Systemwide Id for the server

Contains.

Communication continued....

Implementing Object-References Continued

(b) Client and Server must use same protocol for setting upan inital connection, handle errors, flow control the same way.

(c) Including implementation handle in the object refference. Which refers to a complete implementation of a proxy that the client can dynamically load when binding to that object.

ex:- ftp://ftp.clientaware.org/proxies/java/Proxy-v1.1a.zip

↳ The binding protocol should prescribe a file that should be dynamically downloaded, unpacked, installed & subsequently instantiated.

Special security measures to ensure the client that it can trust the downloaded code.

5 Communication Continued ...

Static Vs Dynamic RMI

a) RMI :- after a client is bound to an object, it can invoke the object's methods through the Proxy.

RMI :- Supports Systemwide objects refferences.
Supports general Purpose client side & server side stubs, object specific stubs

(RMI) Remote Method Invocation 2 types

(i) Static Invocation :- require that the interfaces of an object are known when the client application is being developed. If interfaces change then client application must be recompiled. → $\boxed{\text{obj.add}(x,y)}$

(ii) Dynamic Invocation :- Composing method invocation at runtime.

↳ invoke (object, method, input_param, output_param);

Communication Continued.

E Parameter Passing. Key Points

1. Parameter Passing in RMI less restricted than RPC

2. When Invoking a method with an object refference as Parameter that refference is copied and passed :—

   (1) By Value when it is local object

   (ii) By Refference - when it is a remote object

Side effect of invoking a method with object refference as Parameter is that we maybe copying the Object.

Draw Fig 10-8. Pg 461 Textbook.

Example :- RMI Theory & Program of DIS lab.

7 Communication continued..... Pg 20

Object Based Messaging.

a) RMI is Preffered way of Communication.

b) Messaging also ~~Poss~~ important.

c) Ex: CORBA Messaging Combines method invocation and Message Oriented Communication.

d) Messaging takes Place by invoking object.

e) Assynchronous Method Invocation (ASMI):- The caller continues after initiating the invocation without waiting for a result.

ASMI Two steps:-

Step (i) Implement 2 interfaces.

Interface 1 contains specification of methods that client can call.

Interface 2 is the Callback interface.

Step (ii) Compiling the generated interfaces.

f) CORBA's Callback Model.
Client Provides an Object that implements an interface Containing Callback methods. These methods Can be called by the underlying Communication system to Pass the result of an assynchronous invocations. Draw Fig 10-9

Communication continued....

9) CORBA Polling model :-

Inthis model the client is offered a Collection of operations to poll to its local RTS (Runtime System) for incoming results.

CORBA
← Difference →

Call Back model

1. Client responsible for transforming the original synchronous method Invocations into Asynchronous ones.

2. ex:

```
Void send_add(in int i, int j);
//downcall by client
Void reply_add(in int ret_val, in int k);
// Upcall to the client
```

in — Incomming Parameter.
out → outgoing Parameter

3) Draw Fig 10.9

Polling Model

1. The response method have to be implemented by client (RTS)

2. ex:-

```
Void sendpoll_add
    (....)
// called by client
Void replypoll_add
    (out....);
// Also called by client.
```

4) Draw Fig 10.10.

Scanned with CamScanner

Commuication in Distributed Object Based Systems

Summary sheet For easy Remberance

* How To Bind object

Binding
- Implicit Binding
- Explicit Binding

* How to refer the object.

Implementation of Object References
- Network details
- Protocol
- File Info.

* How to Invoke the methods in an object

RMI
- Static
- Dynamic

* How Parameters are Passed [object References as parameters].

Parameter Passing
- By Value (local Object)
- By Refference (Remote object)

*

Object Based Messaging

Assynchronous Method Invocation
- CORBA's Callback Model
- CORBA's Polling model.