21

UNIT- II

COMMUNICATION IN DISTRIBUTED SYSTEMS

LECTURE NOTES

1) System Model

* A s/m model is the conceptual model that describes and represents a s/m. s/m modelling helps the analyst to understand the functionality of the s/m. A models are used to communicate with customers

* System models are of three types.
  1. Physical Model
  2. Architectural model
  3. Fundamental model.

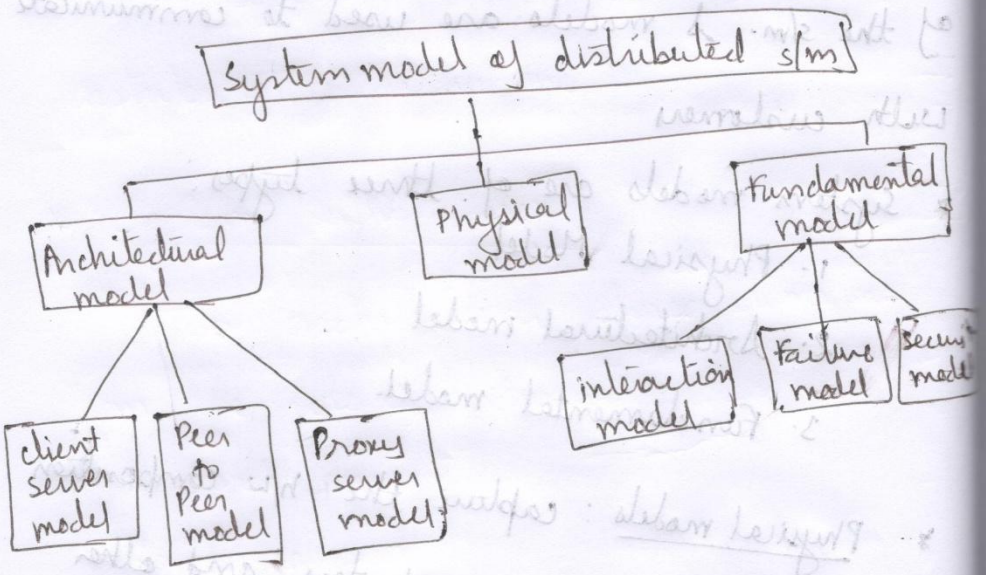
* Physical models: capture the h/w composition of a s/m in terms of computers and other devices and their interconnecting n/w.

* Fundamental models: Formal descriptions of the properties those are common to architecture models

22

Three fundamental models are interaction model, failure model & security model.

* Architecture model: defines the main components of the s/m, what their roles are and how they interact and how they are deployed in a underlying n/w of computers



2. ARCHITECTURAL MODEL

Architectural model is an abstract view of a distributed s/m. Models are constructed

To simplify reasoning about the s/ms. A model of a distributed s/m is expressed in terms of components, placements of components and interactions among components

* Process is component of a DS. A process is running pgm. Examples of processes are server process, client process and peer process.

1. Server process: A pgm executing server code
2. Client process: A pgm executing client code

* Processes interact by sending each other msgs. Architectural models can be used to determine placement of processes. More dynamic s/ms can be built as variations on the client-server model

1. Moving code
2. Adding/removing nodes or components
3. Discovery and advertisement of services

24

* Architectural Elements

To study the fundamental building blocks of a distributed s/m, following points are considered.

1. Communication entities used
2. Communication b/w entities
3. Function and responsibilities
4. Mapping of entity with physical distributed infrastructure

* System Architectures

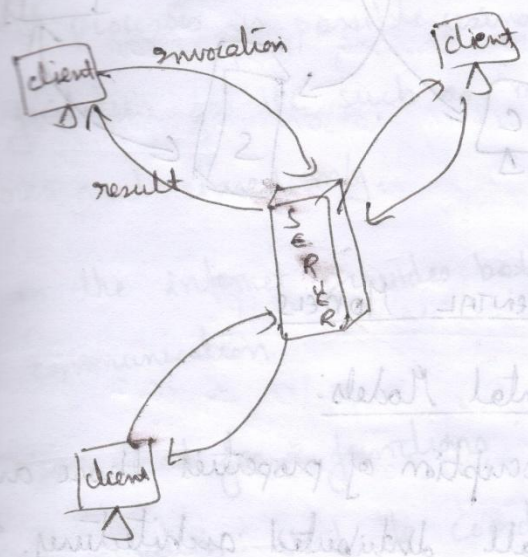
two architectural styles stemming from the role of individual process.

1. client-server
2. Peer to peer

* client-server Model

Distributed services which are called on by clients. servers that provide services are treated differently from clients

that use services. Processes divided into two groups: server and client process

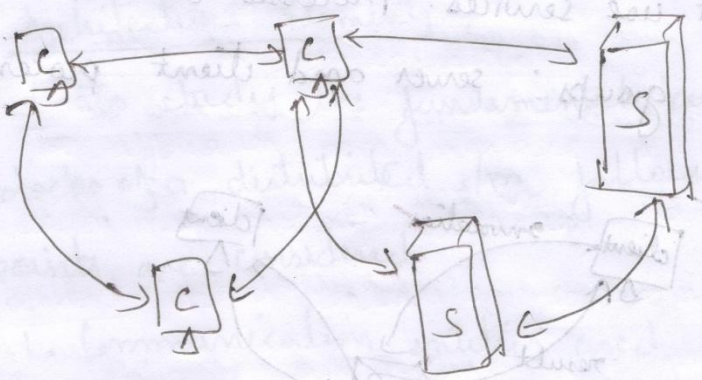


Peer to Peer

→ P2P s/m offer an alternative to traditional client-server s/m's for some appln domains.

→ P2P n/w is a distributed n/w composed of a large no. of distributed heterogeneous, autonomous and highly dynamic peers in which participants share a part of their own resources such as processing power, storage capacity, s/w's and files content

26



3. FUNDAMENTAL MODELS

* Fundamental Models:

Description of properties those are present in all distributed architectures. It is also an abstraction of essential properties of a natural phenomenon, for the purpose of understanding and analysis.

* Purpose of a model is

1. Make explicit all relevant assumptions
2. Make generalization concerning what is possible or impossible.

4. Inter Process Communication

* Exchange of data b/w 2 or more seperate independent processes is possible using IPC. OS provide facilities for IPC such as msg queues, semaphores and shared m/y.

* IPC in the internet provides both datagram and stream communication.

* IPC is used for 2 functions.

1) Synchronization: Used to coordinate access to resources among processes and also to coordinate the execution of these processes. They are record locking, semaphores, mutex and condition variables.

2) Message passing: Used when processes wish to exchange information. Message passing takes several forms such as: Pipes, FIFOs, Message queues and shared m/y.

* Java API provides both datagram and stream communication.

28

5. The API for Internet Protocols

* The application pgms interface to UDP provides a msg passing abstraction. The independent packets containing messages are called datagrams.

* General Characteristics of IPC

1) IPC is the transfer of data among processes. eg. a web browser may request a web page from a web server, which then sends HTML data. This transfer of data usually uses sockets in a telephone-like connection.

2) Exchange of data b/w two or more seperate, independent processes/ threads

3) Msg passing b/w a pair of processes supported by 2 commn operations: Send and Receive.

*

* Message destinations:

1) A local port is a msg destination within a computer, specified as an integer. A port has exactly one receiver but can have many senders.

2) Processes may use multiple ports from which to receive messages. Any process that knows the number of a port can send a message to it.

* Reliability:

1) Reliability is defined in terms of validity and integrity.

2) Integrity: Messages must arrive uncorrupted and without duplication.

3) Validity: Point to point msg service can be described as reliable if msgs are guaranteed to be delivered despite a reasonable number of packets being dropped or lost.

* Ordering:

1) Some applications require that msgs be delivered in sender order. i.e. the order in

which they were transmitted by the sender.

2) The delivery of msgs out of sender order is regarded as a failure by such applns.

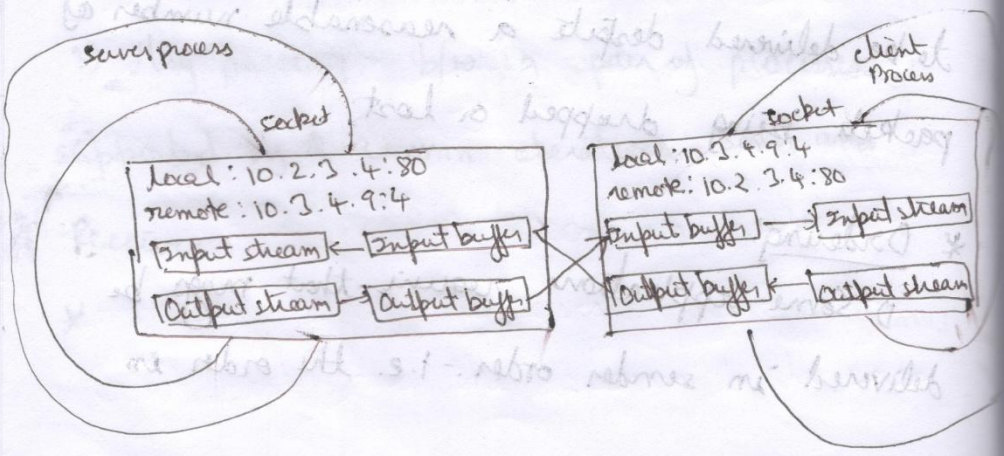
* Socket

1) A socket is a bidirectional comm. device that can be used to communicate with another process on the same m/c or with a process running on other m/c

2) Socket interface is a protocol independent interface to multiple transport layer primitives.

In order to write applications which need to communicate with other applications

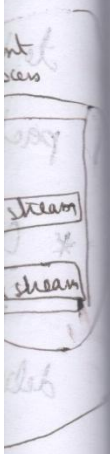
3)



31

socket functional calls

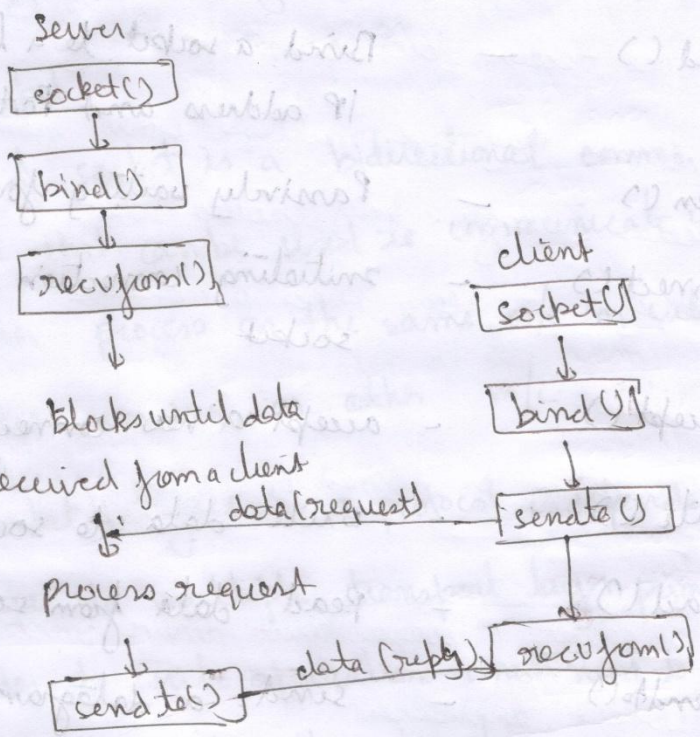
Call	Remarks
1) socket()	— Create a socket
2) bind()	— Bind a socket to a local IP address and Port no.
3) listen()	— Passively waiting for connections
4) connect()	— initiating connection to another socket
5) accept()	— accept a new connection
6) write()	— write data to socket
7) read()	— Read data from socket
8) sendto()	— send a datagram to another UDP socket
9) recefrom()	— read a datagram from UDP socket
10) close()	— close a socket.



32

* UDP Datagram Communication

The interaction b/w a UDP client and server



1) There are some instances when it makes to use UDP instead of TCP. Some popular applns built around UDP are DNS, NFS and SNMP.

2) Initially client does not establish connection with the server instead the client just send the datagram to the server

33

using the sendto() function which requires address of the destination as a parameter.

- Similarly, the server does not accept a connection from a client instead the server just calls the recvfrom() function which waits until data arrives from some client.
- recvfrom returns the protocal address of the client along with the datagram so that server can send a response to the client.

* UDP failure model:

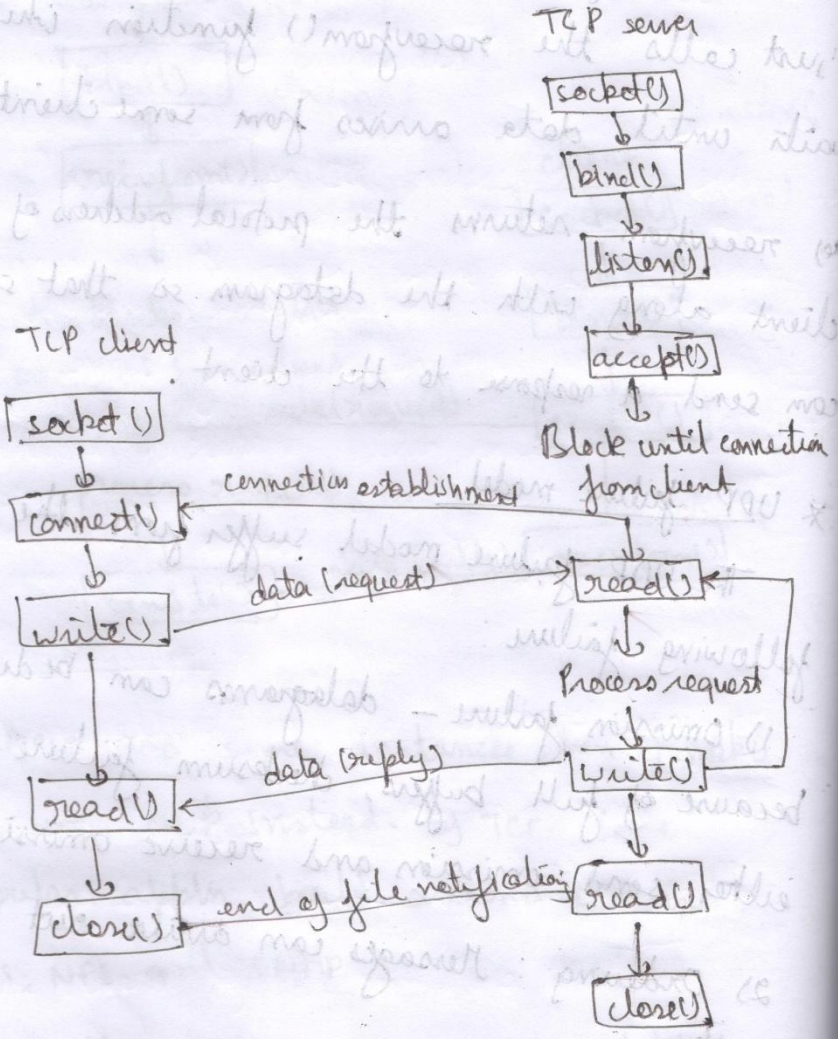
The UDP failure model suffer from the following failure:

1) Omission failure – datagrams can be dropped because of full buffers, checksum failure, either send-omission and receive omission.

2) Ordering: Messages can arrive out of order.

* TCP stream communication

The following dig shows a timeline of the typical scenario that takes place b/w a TCP client and server.



* An abstraction provided by TCP API as a stream of bytes to which data may be read.

35

* Stream abstraction offers the following characteristics

⇒ Message size: No restriction about msg size. application can choose how much data written to a read from a stream.

⇒ Lost message: guarantee msg delivery using acknowledgement scheme. The sender retransmit the msg if it doesnot receive an acknowledgement within a time out

⇒ Flow ctrl: attempt to match speed of reader and writer process. The writer may be blocked until the reader has consumed sufficient data

⇒ msg duplication & ordering: guarantee and duplication and ordering of msgs. msgs identifier enable the receiver to detect and reject duplicates are to re-order msgs not arrived in sender order.

36

6) External Data Representation And Marshalling

* Different h/w & OS platform use different representation for elementary data type such as integers and characters.

* Most modern OS represent 16 bit integers as 2 bytes where most significant byte comes first. older m/c such as IBM mainframe represent this integers exactly other way around.

* Marshalling

1) marshalling is the process of assembling the collection of data items in a form suitable for transmission & unmarshalling is the reverse process of disassembling them on arrival.

2) Marshalling pattern complex data structures

translatable representation usually a stream of bytes which may be split into sequence of msgs if necessary.

CORBA CDR

CORBA stands for common object request broker architecture. It is an industry std. developed by OMG to Aid in Distributed object programming.

CORBA is not a programming language. CORBA architecture is based object model.

A CORBA based s/m is a collection of objects that isolates requesters of services from the providers of services (servers). By a well defined encapsulating interface.

Java Object Serialization

In Java RMI, ~~objects~~ ~~obj~~ both object & primitives data values may be passed as arguments & results of mtd invocations. In

object is a instance (java class)

* Serialization is the process of converting a set of object instances that contain references to each other into a linear stream of bytes, which can then be sent through a socket, stored to a file or simply manipulated as a stream of data.

* Serialization is usually used when there is need to send data over n/w or to store in files.

* XML

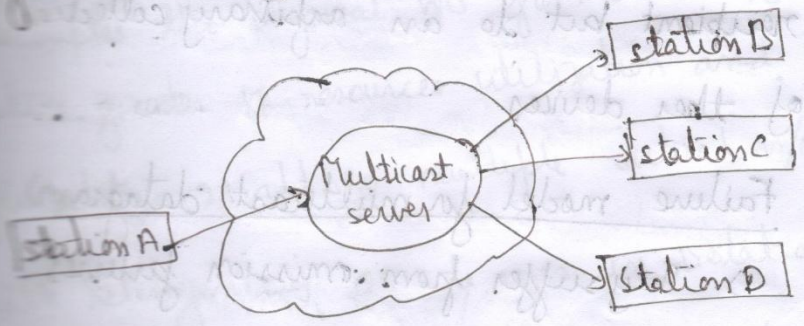
1) eXtensible: By applying identifiers for elements of information in a neutral way, stored in a neutral form, independent of s/m's, devices and applications.

2) Markup: For adding information to a document relating to its structure and/or content.

3) Language: A std methodology with formal syntax.

2) MULTICAST COMMUNICATION

* Multicast communication allows a process to send the same msg to a group of processes.
* multicast operations can provide the programmer with delivery guarantees that are difficult to realize for the application programmer using ordinary unicast operations.



40

* Multicast msg provides a useful infrastructure for constructing distributed s/m's with the following characteristics:

1) Replicated services

2) Better performance

3) Propagation of event notification

* IP Multicast

1) IP Multicast is built on top of the Internet protocol. IP multicasting is to allow a device on an IP internetworking to send datagrams not to just one recipient but to an arbitrary collection of ther devices

* Failure model for multicast datagrams

1) They suffer from omission failures.

2) Omission Failure:
   - msgs dropped
   - checksum errors

41

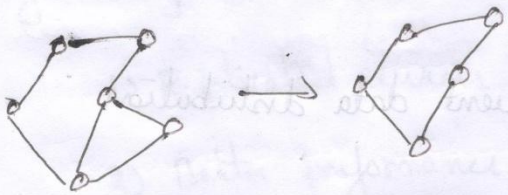
- Both send-omissions and receive-omissions can occur in multicast datagram.
- Applications that use UDP need to provide their own checks
- Advantages: efficient data distribution
- Disadvantages:
  - Limited router support
  - Per-group/source state in routers
  - Reliability difficulties
  - Congestion-ctrl difficulties

NETWORK VIRTUALIZATION

- Virtualization is an abstraction layer that decouples the physical h/w from the OS & gives greater IT resources utilization and flexibility. It allows multiple virtual m/c, with heterogeneous OS to run in isolation, side-by-side on the same physical m/c.

42

* Making a physical n/w appear as multiple logical ones



Physical n/w | Virtualized n/w-1 | Virtualized n/w-2

* n/w virtualization is a n/wing environment that allows multiple service providers to dynamically compose multiple heterogeneous virtual n/w that co-exist together in isolation from each other and to deploy customized end-to-end services on-the-fly as well as manage them as those virtual n/w's for the end-users by effectively sharing and utilizing underlying n/w resources leased from multiple infrastructure providers.

* Virtualization is NOT a mtd to increase resiliency.

* Advantages of n/w Virtualization
  1) Lower number of physical devices
  2) Multiple devices with separate roles and simpler configurations
  3) Easier to manage.

* Overlay n/w
  1) A n/w built on top of one or more existing n/w. It adds an additional layer of virtualization. An overlay n/w is a virtual n/w of nodes and logical links that is built on top of an existing n/w with the purpose to implement a n/w service that is not available in the existing n/w.

* Overlay n/w Types:
  a) Distributed hash tables
  b) Peer to peer file sharing
  c) Content distribution n/w.

44

d) wireless adhoc n/w's

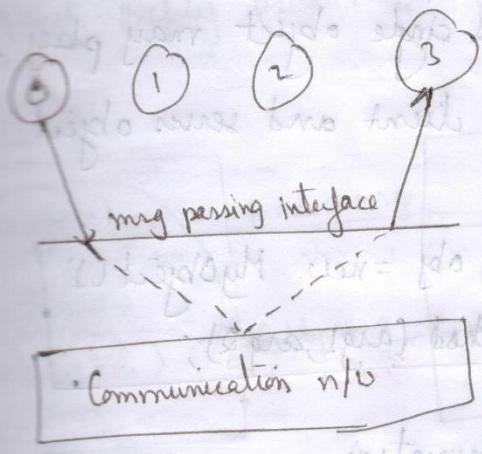
e) CASE STUDY : MPI

* MPI is an Application Programming Interface (API) for communication b/w seperate process. The most widely used approach for distributed parallel computing. MPI is portable, scalable flexible and comprehensive

* Message passing interface (MPI) is an industrial std that specifies library routines needed for writing msg passing pgms. MPI allows the development of scalable portable msg passing pgms. It is a std supported pretty much by everybody in the field.

* MPI uses a library approach to support IIel programming. It specifies the API for msg passing. MPI pgm are

compiled with a regular compiler and linked with an MPI library

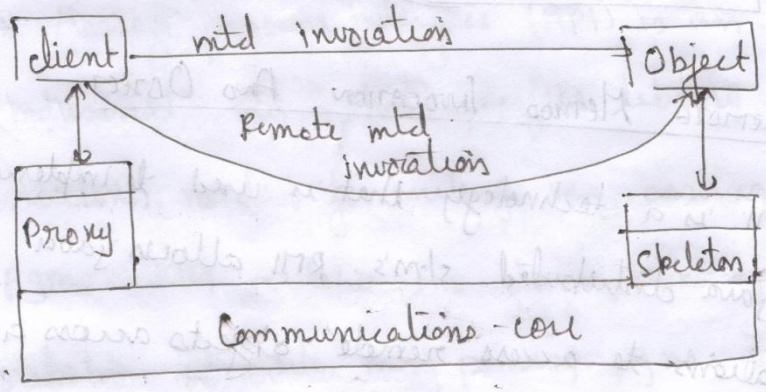


3) Remote Method Invocation And Objects.

* RMI is a technology that is used to implement pure java distributed s/m's. RMI allows 'java' applications to access "remote" objects across a distributed s/m. clients and servers in an RMI application can be distributed over a n/w or can reside in virtual m/c running on the same physical m/c

* When an object invokes an operation in another object, the former is called the client object, and the latter is called the server object. A single object may play the role of both a client and server object.

```
MyObject obj = new MyObject();
obj.method (arg1, arg2);
```



* Request - Reply Protocols

1) Basically all client - server comm n. follows the pattern of a request - reply protocol:

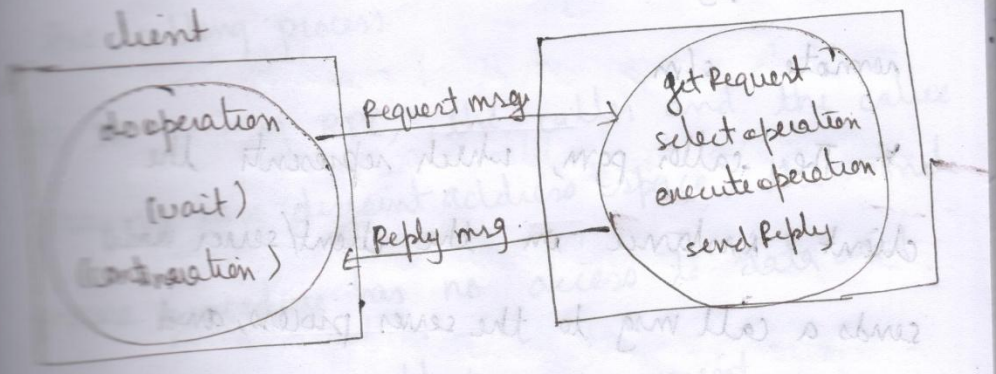
a) The client sends a request msg

b) The server executes the requested operation

47

c) The server responds with a reply msg.

- The following fig. shows the general model of request-reply protocols



operation	Description
doOperation	– The client sends requests and returns answer to the applⁿ pgm.
getRequest	– The server gets the request from the client.
sendReply	– The server sends a reply to the client

48

11) REMOTE PROCEDURE CALL

* RPC is an appln programming Interface (API) available for developing distributed appln. It allows pgms to execute subroutines on a remote s/m.

* The caller pgm, which represents the client instance in the client/server model sends a call msg to the server process, and waits for a reply msg.

* Traditionally the calling procedure is known as the client and the called procedure is known as the server.

* RPC Model

i) RPC model is similar to the well known and well understood procedure call model used for transfer of ctl and data within the pgm

RPC mechanism is an extension of the procedure call mechanism in the sense that it enables a call to be made to a procedure that does not reside in the address space of the calling process.

In case of RPC, the caller and the callee processes have disjoint address space, the remote procedure has no access to data and variables of the caller's environment

Transparency of RPC

→ A transparent RPC mechanism is one in which local procedures and remote procedures are indistinguishable to programmers.

→ RPC uses two types of transparency:

1) syntactic Transparency
2) semantic transparency

50

* Stub Generation

→ stub stubs can be generated in two ways: manually and automatically

→ In manually mtd, user can construct the stubs using set of translation functions provided by RPC implementer.

→ Most commonly used mtd for stub generation is automatic mtd. It uses Interface Definition Language (IDL). IDL is used to define the interface b/w client and a server.

* Server management:

→ servers are of two types: stateful server and stateless server. classification of server is based on implementation of the server.

→ stateful server: It maintains the client's state information for one remote procedure call to the next. Following are

the file operation supported by this server.

1) open
2) Read
3) write
4) seek
5) close

→ stateless server. This server does not maintain any client state information. Following are the file operations supported by this server.

1) Read
2) write

* RPC Pblm.

→ RPC is works really well if all the m/c are homogeneous

→ complications arise when the two m/c use different character encoding.

52

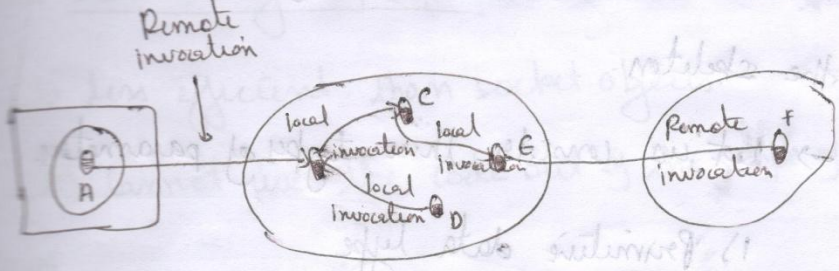
12) REMOTE METHOD INVOCATION

* RMI stands for Remote Method Invocation, is an object-oriented implementation of the remote procedure call model. It is an API for java programs only

* RMI is the OO equivalent of remote mtd calls. In this model, a process invokes the mtds in an object, which may reside in a remote host.

* RMI also supports registry, which allows client to perform lookups for a particular servic.

* Remote object must be written in java. Remote object are accessed through their interfaces only



Design Issues for RMI

1) RMI invocation semantics

2) Level of transparency

RMI invocation semantics

→ Maybe, at-least-once and at-most-once suffer from crash failures when the server containing the remote object fails.

1) Maybe semantics

2) At-least once semantics

3) At-most once semantics

Passing Parameters

→ when a client invokes a remote mtd with parameters, passing parameters are

54

handled under the covers by the stub and the skeleton.

→ Let us consider three types of parameters.

1) Primitive data type

2) Local object type

3) Remote object type

* RMI Registry

A simple directory service called the RMI registry, rmiregistry, which is provided with the Java S/W development kit (SDK). The RMI registry is a service whose server, when active, runs on the object server's host m/c

* Advantage of RMI

1) simple & clean to implement

2) It is possible to create zero-install client for the users

3) No client installation is needed.

55

* Disadvantage of RMI

1) Less efficient than socket objects.

2) Cannot use the code out of the scope of Java.

3) security issues need to be maintained more closely

13) CASE STUDY: JAVA RMI

* Java RMI is a mechanism that allows a java program running on one computer to apply a mtd to an object living on a different computer.

* Java RMI provides the following elements:

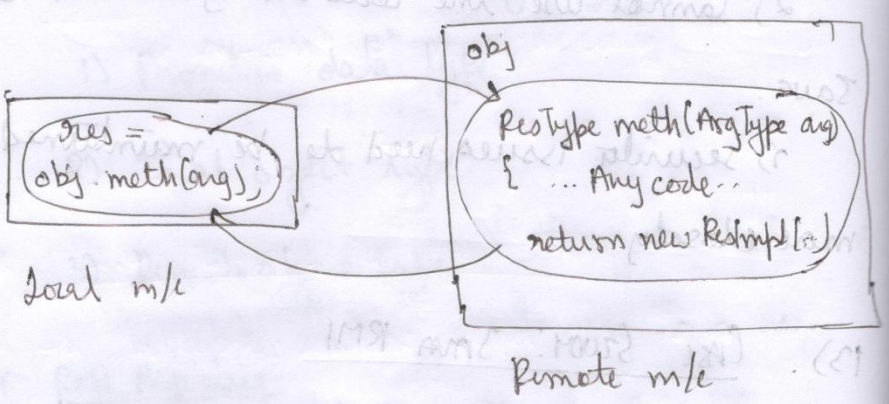
a) Remote object implementations

b) client interfaces, or stubs

c) A remote object registry for finding objects in the n/w

d) A facility for automatically creating

remote objects on - demand.



* Advantage of RMI

1. Since RMI is a java native, integration of its remote object facilities into a java appln is seamless
2. RMI is platform - independent
3. RMI is easy to learn

14) GROUP COMMUNICATION

* Groups are dynamic. They may be created or destroyed. Processes may join

- leave groups and processes may belong to multiple groups.
- Group communication can be implemented several ways.
  1. one to many
  2. Many to one
  3. Many to many

* One to many

→ Msg is sent by one sender to multiple receivers

→ Group Management:

↳ Here receiver processes forms a group. groups are of two types:

1) closed group
2) open group

→ In one-to-many communication, the degree of flexibility is normally expressed in following forms:

58

a) 0-reliability

b) 1-reliability

c) m-out-of-n reliability

d) All-reliable

* Many-to-one Communication

→ Multiple senders send msgs to a single receiver. single receiver may be selective or non-selective

→ Many-to-one scheme is non-determinism.

* Many-to-many

→ multiple sender send msgs to multiple receiver

→ ordered msg. delivery ensures that all msgs are delivered to all receivers in an order acceptable to the application

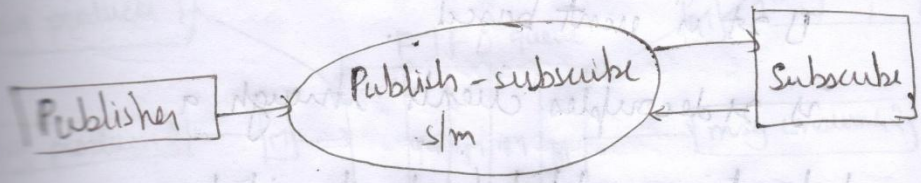
→ some msg ordering required

1. Absolute ordering

2. Consistent / Total ordering
3. Causal ordering
4. FIFO ordering.

2) PUBLISH - SUBSCRIBE SYSTEM

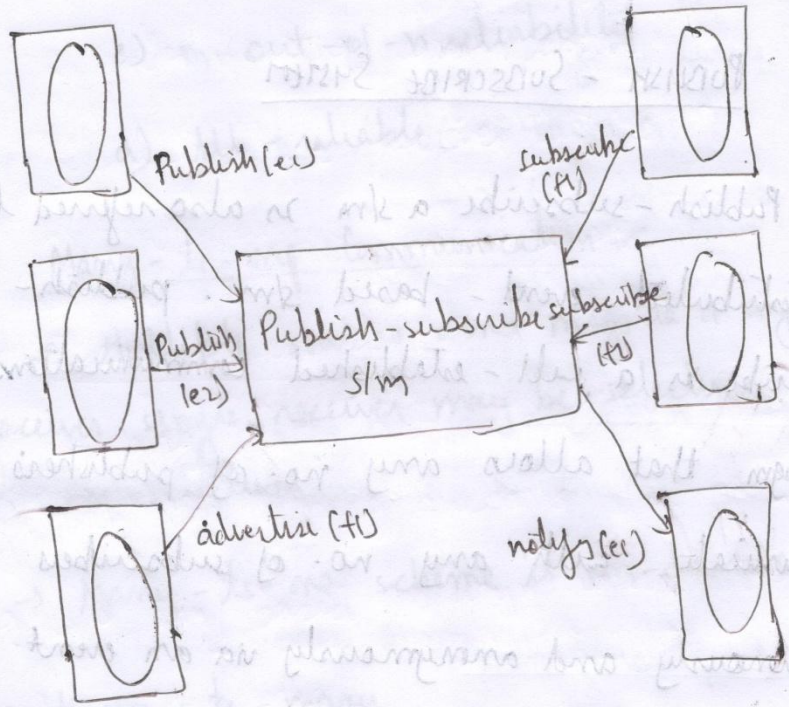
* Publish - subscribe a s/m is also referred to distributed event - based s/m. publish-subscribe is a well-established communications paradigm that allows any no. of publishers to communicate with any no. of subscribers asynchronously and anonymously via an event channel



* characteristics of publish - subscribe s/m
1. Asynchronous communication
2. many to many interaction
3. Heterogeneous

60

* Programming Model



* Publish/ subscribe Benefits

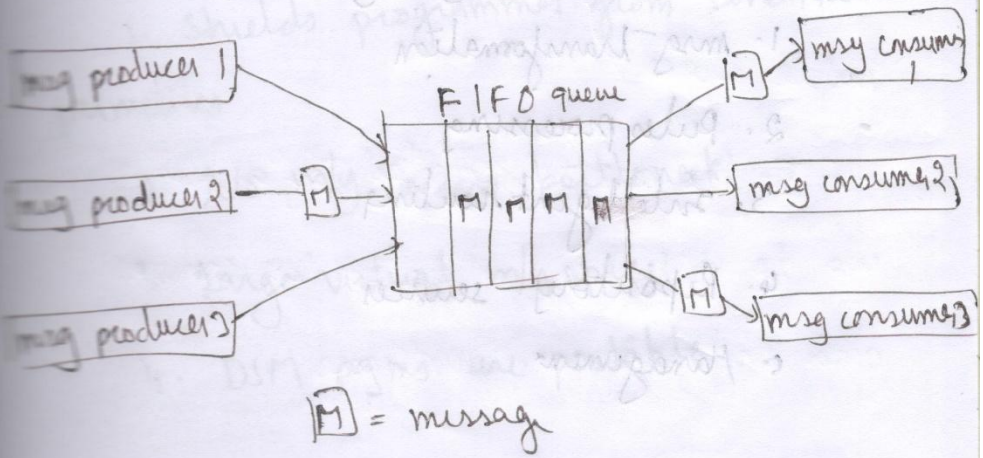
1) It is event based
2) It decouples clients through a declarative publish/ subscribe interface
3) interactions and queries are declarative

* Disadvantages of Publish/subscribe

1) No strong guarantee on brokers to deliver content to subscribers
2) Potential bottle neck
3) Security an issue.

2) Message Queues

* Msg queues are also referred to as msg-oriented middleware (MOM). msg brokers or traditional, msg-oriented middleware are much better tools for the simple sharing of information at the data level or the applⁿ interface level.



62

* sender and receiver execution mode are as follows:

1) sender running, receiver running

2) sender running, receiver passive

3) sender passive, receiver running

4) sender passive, receiver passive

* Message brokers:

→ msg broker act as an appln level gateway in a msg queuing s/m. It is a s/w s/m based on asynchronous, store-and-forward msg.

* → services provided by msg brokers

1. msg transformation
2. Rules processing
3. Intelligent routing
4. Repository services
5. Management

7) SHARED MEMORY APPROACHES

* The DSM implements the shared m/y model in distributed s/ms., which have no physical shared m/y. The shared m/y model provides a virtual address space shared b/w all nodes.

* DSM allows pgms running on seperate computers to shared data w/o the programmer having to deal with sending msg. Instead underlying technologies with send the msgs to keep the DSM consistent b/w computers.

* Advantages of DSM

1. shields programmes from send/Receive primitives
2. No m/y access bottleneck
3. Large virtual m/y space
4. DSM pgms are portable

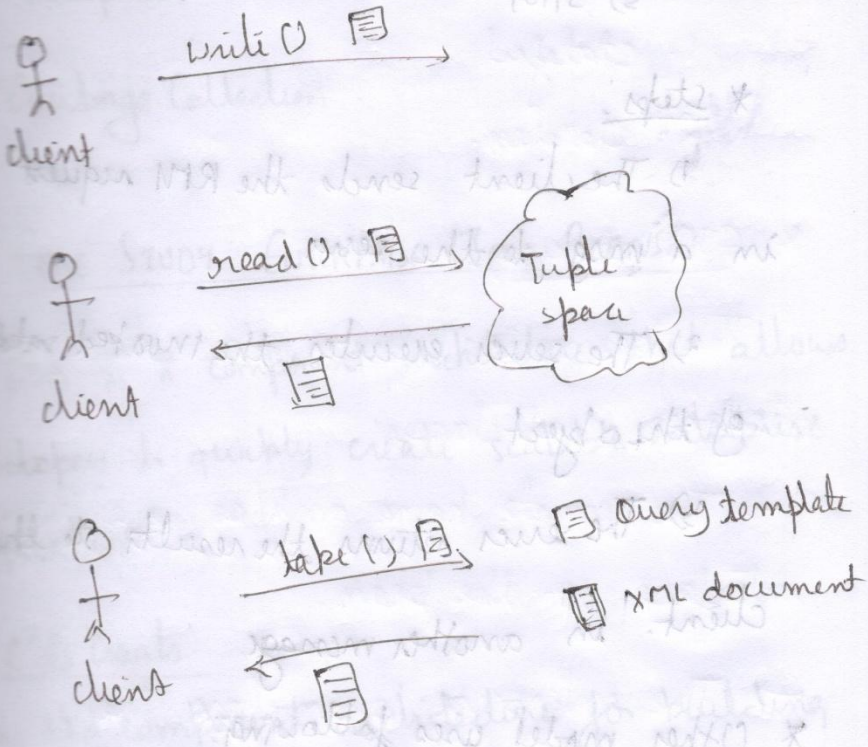
* Disadvantage of DSM

1) May incur a performance penalty
2) Must provide for protection against simultaneous access to shared data
3) Little programmer ctrl over actual msg being generated
4) Performance of irregular pblms in particular may be difficult

* Eg Tuple space communication:

→ Tuple is ordered collection of typed data. It is fundamental data structure of tuple space

→ Tuples are represented by a list of upto 16 fields, seperated by commas and enclosed in parentheses



7) DISTRIBUTED OBJECT

* Object has a unique identifier. It may have many different references that refer to the object. It has a set of attributes whose names denote values.

* Distributed object technologies:
  1) Distributed Common Object Model (DCOM)
  2) CORBA

3) JINI

* Steps:

1) The client sends the RMI request in a msg to the server

2) The server executes the invoked method of the object

3) The server returns the results to the client in another message

* Other model uses following:

a) Chains of related invocations

b) Object replication

c) Object migration

* Five Parts of Distributed Object Model:

1. Remote Object References
2. Remote Interfaces
3. Actions

67

4. Exceptions

5. Garbage Collection.

(A) CASE STUDY: ENTERPRISE JAVA BEANS

* EJB is a component architecture that allows developers to quickly create scalable enterprise application.

* EJB Goals:

1. Std component architecture for building distributed business applns. in java
2. Interoperability b/w enterprise beans

* Types of EJB

→ Types of EJB are as follows:

1. Session beans
2. Message driven beans
3. Entity beans

* To accommodate the different ways clients interact with applications, session

68

beans come in two varieties:

1. stateful
2. stateless.

* EJB Working

The bean developer must create these interfaces and classes:

→ The remote home and/or local home interface for the bean.

→ The enterprise bean class. The enterprise bean class implements the business logic for the bean. The client accesses these methods through the bean's remote interface.

* Home object

→ The client cannot instantiate an EJB object directly because EJB objects could exist on a different m/c than

the one the client is on. Similarly, EJB promotes location transparency, so clients should never be aware of exactly where EJB object reside.

* The chief responsibilities of home objects are to do the following:

1. create EJB object
2. Finding existing EJB objects
3. Remove EJB objects

* Advantages of EJB

1) Helps to write scalable, reliable and secure applns.

2) Thinner clients can be developed

3) It is agreed upon by industry

4) A new EJB can be constructed from existing EJBs.