UNIT-I

CHARACTERIZATION OF DISTRIBUTED SYSTEMS

1) Define Distributed System & discuss its characteristics. Give examples for Distributed Systems.

Distributed system is the one in which hardware or software components located at networked computers communicate and coordinate their actions only by passing messages.

Characteristics of Distributed Systems are:

Concurrency: Concurrent programs execution is the norm. I can do my work on my computer while you do your work on yours, sharing resources such as web pages or files when necessary. Resources can be added or removes easily.

✓ No global Clock: Close coordination often depends on a shared idea of the time at which the programs' actions occur. But it turns out that there are limits to the accuracy with which the computers in a network can synchronize their clocks – there is no single global notion of the correct time.

✓ Independent Failure: Faults in the network result in the isolation of the computers that are connected to it, but that doesn't mean that they stop running.

Typical examples of Distributed systems are,

- The Internet
- Intranets
- Mobile and Ubiquitous computing

2) List the challenges in distributed systems. Explain in detail any two of them.

HETEROGENEITY:

The Internet enables users to access services and run applications over a heterogeneous collection of computers and networks.

Openness:

✓ The openness of a computer system is the characteristic that determines whether the system can be extended and reimplemented in various ways.

✓ The openness of distributed system determined primarily by the degree to which new resource sharing devices can be added

Security:

✓ Their security is therefore of considerable importance.

✓ Security for information resources has three components:

✓ confidentiality (protection against disclosure to unauthorized individuals),

✓ integrity(protection against alteration or corruption), and

✓ availability (protection against interference with the means to access the resources).

Scalability:

✓ A system is described as scalable if it will remain effective when there is a significant increase in the number of resources and the number of users.

Failure Handling:

✓ Failures in a distributed system are partial – that is, some components fail while others continue to function.

✓ The following are techniques for dealing with failures. ▪ Detecting failures ▪ Masking failures ▪ Tolerating failures ▪ Recovery from failure ▪ Redundancy

Concurrency:

✓ Both services and applications provide resources that can be shared by clients in a distributed system.

✓ There is therefore a possibility that several clients will attempt to access a shared resource at the same time.

SYSTEM MODELS

1) Define Architecture Model. Mention its goal & explain the following with an example.
i) Mobile Code ii) Mobile Agent iii)Proxy Server & Cache

Architectural model is concerned with the placement if its components and the relationships between them.

i)Mobile Code:

• It is used to refer to code that can be sent from one computer to another and run at the destination.

• Example : java applets

ii)Mobile agent:

• Mobile agent is a running program that travels from one computer to another carrying out a task to someone's behalf, such as collecting information, eventually returning with the results.(eg: Google form)

• Mobile agent is a complete program(including both code & data) that can work independently.

• Mobile agent can invoke local resources/data.

iii)Proxy & cache:

• proxy servers are u sed to increase availability and performance of the services by reducing the load on the network and web-server.

• Proxy server provides copies(replications) of resources which are managed by other server

Cache:

A store of recently used data objects that is closer to the client process than those remote objects

2. Summarize the following design requirements for Distributed Architectures

i) Performance Issues ii) Quality of Service

1.Performance Issues

■ Responsiveness : The speed of a remote invocation depends on:

- ■ The load and performance of the server and network
- ■ Delays in all the software components
- ❑ Transfer of data is slow

■ Throughput:

- ❑ Rate at which computational work is done
- ❑ Fairness

■ Balancing of Computational Loads

- ❑ Applets remove load from the server
- ❑ Use several computers to host a single service

2.Quality of Service :

- ❑ Reliability & Security :
  - ❑ ability of a system to perform and maintain its function in every circumstance
- ❑ Performance :
  - ❑ Ability to meet Timeliness guarantees
- ❑ Adaptability:
  - ❑ the ability of a system to adapt itself efficiently and fast to changed circumstances

3. Explain the Failure Model. With the help of a tabular column describe the various classes of Arbitrary, Omission & Timing failures

Failure model :

- In a distributed system both processes and communication channels may fail.
- Failure model defines the types of failure,
  - Omission failure
  - Arbitrary Failure
  - Timing Failure

Omission and arbitrary failures

Class of failure	*Affects*	*Description*
Fail-stop	Process	Process halts and remains halted. Other processes may detect this state.
Crash	Process	Process halts and remains halted. Other processes may not be able to detect this state.
Omission	Channel	A message inserted in an outgoing message buffer never arrives at the other end's incoming message buffer.
Send-omission	Process	A process completes a*send*, but the message is not put in its outgoing message buffer.
Receive-omission	Process	A message is put in a process's incoming message buffer, but that process does not receive it.
Arbitrary (Byzantine)	Process or channel	Process/channel exhibits arbitrary behaviour: it may send/transmit arbitrary messages at arbitrary times, commit omissions; a process may stop or take an incorrect step.

Timing failures

Class of Failure	*Affects*	*Description*
Clock	Process	Process's local clock exceeds the bounds on its rate of drift from real time.
Performance	Process	Process exceeds the bounds on the interval between two steps.
Performance	Channel	A message's transmission takes longer than the stated bound.

UNIT-II

INTER PROCESS COMMUNICAITON

1. Explain the characteristics of IPC & With a neat diagram explain sockets

1. Synchronous and asynchronous communication :

➢ In the synchronous form, both send and receive are blocking operations. Eg.Continous Chatting

➢ In the asynchronous form, the use of the send operation is non-blocking and the receive operation can have blocking and non-blocking variants. Eg.WhatsApp

2. Message destinations:

➢ A local port is a message destination within a computer, specified as an integer.

➢ A port has an exactly one receiver but can have many senders.

3.Reliability :

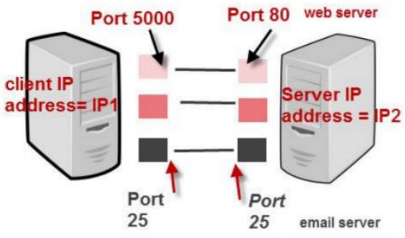
➢ A reliable communication is defined in terms of validity and integrity.

➢ A point-to-point message service is described as reliable if messages are guaranteed to be delivered despite a reasonable number of packets being dropped or lost.

➢ For integrity, messages must arrive uncorrupted and without duplication.

4.Ordering: Some applications require that messages to be delivered in sender order.

Sockets:



IP Address + Port number = Socket

TCP/IP Ports And Sockets

▪ A socket can be thought of as an endpoint in a two-way communication channel. Eg: telephone Call

▪ Internet IPC mechanism of Unix and other operating systems (BSD Unix, Solaris, Linux, Windows NT, Macintosh OS) Processes can send and receive messages via a socket.

▪ Sockets need to be bound to a port number and an internet address in order to send and receive messages

. ▪ Each socket has a transport layer protocol (TCP or UDP).

▪ Messages sent to some internet address and port number can only be received by a process using a socket that is bound to this address and port number.

2. Compare & Contrast between Synchronous & Asynchronous communication in the context of IPC.

3. Explain Java API for the following.
    - UDP datagrams
    - TCP streams

▪ The Java API provides datagram communication by two classes:

➢ Datagram Packet :

❖ It provides a constructor to make an array of bytes comprising:

• Message content • Length of message • Internet address • Local port number

❖ It provides another similar constructor for receiving a message.

➢ Datagram Socket:

❖ This class supports sockets for sending and receiving UDP datagram.

❖ It provides a constructor with port number as argument.

❖ Datagram Socket methods are:

• send and receive • setSoTimeout • connect

▪ Java API for TCP streams :

➢ The Java interface to TCP streams is provided in the classes:

❖ ServerSocket • It is used by a server to create a socket at server port to listen for connect requests from clients.

❖ Socket:

• It is used by a pair of processes with a connection.

• The client uses a constructor to create a socket and connect it to the remote host and port of a server.

• It provides methods for accessing input and output streams associated with a socket.

4. Discuss issues relating to datagram communication.

▪ Issues related to datagram communications are:

➢ Message size:

❖ IP allows for messages of up to 2 16 bytes.

❖ Most implementations restrict this to around 8k bytes.

❖ Any application requiring messages larger than the maximum must fragment.

❖ If arriving message is too big for array allocated to receive message content, truncation occurs.

➢ Blocking :

❖ Send: non-blocking

• upon arrival, message is placed in a queue for the socket that is bound to the destination port.

❖ Receive: blocking:

• Pre-emption by timeout possible

• If process wishes to continue while waiting for packet, use separate thread

➢ Timeout

➢ Receive from any

5 Discuss the Characteristics and issues related to stream communication.

➢ Characteristics of the stream abstraction:

❖ Message sizes: Application dependent

❖ Lost messages :Ack ,Seq No.

❖ Flow control: Window size

❖ Message duplication & Ordering : Seq No

❖ Message destination: time with IP &Port no.

▪ Issues related to stream communication:

➢ Matching of data items:

❖ data interpretation error w.r.t. use of data stream

➢ Blocking: Use of Queue destination Socket

➢ Threads: To avoid delay in handling clients

5. Define marshalling and unmarshalling.

▪ Marshalling ▪ Marshalling is the process of taking a collection of data items and assembling them into a form suitable for transmission in a message.

▪ Unmarshalling ▪ Unmarshalling is the process of disassembling a collection of data on arrival to produce an equivalent collection of data items at the destination.

6. Explain CORBA CDR with an example

- CORBA Common Data Representation (CDR):

➢ CORBA CDR is the external data representation defined with CORBA 2.0.

➢ It consists 15 primitive types:

➢ Short (16 bit) ➢ Long (32 bit) ➢ Unsigned short ➢ Unsigned long
➢ Float(32 bit) ➢ Double(64 bit) ➢ Char ➢ Boolean(TRUE,FALSE)
➢ Octet(8 bit) ➢ Any(can represent any basic or constructed type)

example: struct with value {'Smith', 'London', 1934}

index in sequence of bytes	←4 bytes→	*notes on representation*
0–3	5	*length of string*
4–7	"Smit"	*'Smith'*
8–11	"h___"	
12–15	6	*length of string*
16–19	"Lond"	*'London'*
20–23	"on__"	
24–27	1934	*unsigned long*

Figure 9. CORBA CDR message

7. Explain Java object serialization with an example.

➢ In Java RMI, both object and primitive data values may be passed as arguments and results of method invocation.

➢ An object is an instance of a Java class.

➢ Example, the Java class equivalent to the

Person struct Public class Person implements Serializable {

Private String name;

Private String place;

Private int year;

Public Person(String aName ,String aPlace, int aYear)

{ name = aName;

place = aPlace;

year = aYear;

}

//followed by methods for accessing the instance variables }

8. Define Marshalling. Construct a marshalled form that represents a Organization with instance variable values :{ 'KLSGIT','BELGAUM', 1979, 590008} by using CORBA-CDR & Java Serialization.

9. Analyze the failure model of Request/Reply protocol in client-server Communication using UDP

▪ Failure model of the request-reply protocol

➢ If these three primitives are implemented over UDP they have the same communication failures

▪Omission failure (link failures, drops/losses, missed/corrupt addresses)

▪Out-of-order delivery

▪Node/process down

Solved by

▪Timeouts with retrans until reply is received/confirmed

▪Discards of repeated requests by requestId (by server process)

▪On lost reply messages, server repeats idempotent operations(eg.adding an element to set)

▪Maintain history (reqid, message, client-id) or buffer replies and retrans – memory intensive

10. Discuss the drawbacks of UDP over TCP stream to implement the request-reply protocol

11. Explain request-reply communication with the neat diagram and specify the operations of the same.

▪ The request-reply protocol was based on a trio of communication primitives:

doOperation, getRequest, and sendReply

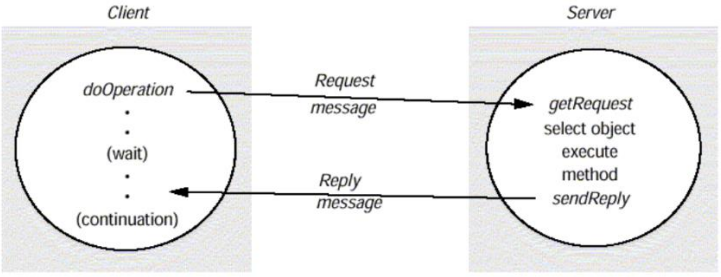


Figure 12. Request-reply communication

42

Three Premitives:

1.doOperation method

▪ used by the client to invoke remote operations

▪Arguments→ object & method to be invoked

▪Its result is an RMI Reply

▪Client calling doOperation marshals the arguments into an array of bytes & unmarshals the results from the array of bytes

▪Client doOperation is blocked until remote object in the server performs the requested operation & transmits a reply msg back

2.GetRequest

▪-used by a server process to acquire service requests

▪-when server has invoked the method in the object it then uses

3.SendReply is used to send reply to client.

▪-when reply msg is received doOperation is unblocked & client continues to execute

12. List and explain RPC exchange protocols.

▪ RPC exchange protocols(failure handling)

➢ Three protocols are used for implementing various types of RPC.

❖ The request (R) protocol.

❖ The request-reply (RR) protocol.

❖ The request-reply-acknowledge (RRA) protocol.

Name	Messages sent by		
	Client	Server	Client
R	Request		
RR	Request	Reply	
RRA	Request	Reply	Acknowledge reply

Figure 15. RPC exchange protocols

▪ In the R protocol, a single request message is sent by the client to the server.

▪ The R protocol may be used when there is no value to be returned from the remote method

. ▪ The RR protocol is useful for most client-server exchanges because it is based on request-reply protocol

. ▪ RRA protocol is based on the exchange of three messages: request-reply-acknowledge reply.

13. Explain HTTP request and reply message format.

▪ HTTP methods

▪ Client Rqst=method+URL

➢ GET

❖ Requests the resource, identified by URL as argument.

❖ If the URL refers to data, then the web server replies by returning the data

❖ If the URL refers to a program, then the web server runs the program and returns the output to the client.

method	*URL*	*HTTP version*	*headers*	*message body*
GET	//www.dcs.qmw.ac.uk/index.html	HTTP/ 1.1		

Figure 16. HTTP request message

➢ A reply message specifies

❖ The protocol version

❖ A status code

❖ Reason

❖ Some headers

❖ An optional message body

HTTP version	*status code*	*reason*	*headers*	*message body*
HTTP/1.1	200	OK		resource data

Figure 17. HTTP reply message