Distributed Systems. DPU

Practical No. 1.

Problem: Implement multi-threaded client/server Process communication using RMI.

Tools: Java environment Environment, jdk 1.8. rmiregistry.

Related Theory: RMI provides communication between java applications that are deployed on different servers & connected remotely using objects called stub & skeleton. This communication architecture makes a distributed application seem like a group of objects communication across a remote connection. These objects are encapsulated by exposing an interface, which helps access the private state & behaviour of an object through its methods.

RMI Registry is a remote object registry, a Bootstrap naming service, that is used by RMI Server on the same host to bind remote objects to names. Clients on local & remote hosts then look up the remote objects & make remote method invocations.

Key terminologies of RMI.

The following are some of the important terminologies used in a Remote Method Invocation. Remote object: This is an object in a specific JVM whose methods are exposed so they could be invoked by another program deployed on a different JVM.

Remote interface: This is a Java interface that defines the methods that exist in a

remote object. A remote object can implement more than one remote interface to adopt multiple remote interface behaviours.

RMI: This is a way of invoking a remote object's methods with the help of a remote interface. It can be carried with a syntax that is similar to the local method invocation.

Stub: This is a Java object that acts as an entry point for the client object to route any outgoing requests. It exists on the client JVM & represents the handle to the remote object.

If any object invokes a method on the stub object, the stub established RMI by following these steps:

1. It initiates a connection to the remote machine JVM.
2. It marshals the parameters passed to it via the remote JVM.
3. It waits for a response from the remote object & unmarshals the returned value or exception, then it responds to the caller with that value or exception.

Skeleton: This is an object that behaves like a gateway on the serve side. It acts as a remote object with which the client objects interacts through the stub. This means that any requests coming from the remote client are routed through it. If the skeleton receives a request, it establishes RMI through

these steps:
1. It reads the parameter sent to the remote method.
2. It invokes the actual remote object method.
3. It marshals the result back to the caller.

Designing the Solution:

The essential steps that need to be followed to develop a distributed application with RMI are as follows:
1. Design & implement a component that should no only be involved in the distributed application, but also the local components.
2. Ensure that the components that participate in the RMI calls are accessible across networks.
3. Establish a network connection between applications that need to interact using the RMI.

Remote interface definition: The purpose of defining a remote interface is to declare the methods that should be available for invocation by a remote client.
Programming the interface instead of progamming the component implementation is an essential design principle adopted by all modern Java frameworks, including Spring. In the same pattern, the definition of a remote interface

takes importance in RMI design as well.

2. Remote object implementation: Java allows a class to implement more than one interface at a time. This helps ~~of~~ remote objects implement one or more remote interfaces. The remote object class may have to implement other local interfaces & methods that it is responsible for. Avoid adding complexity to this scenario, in term of how the arguments or return parameter values of such component methods should be written.

3. Remote client implementation: Client objects that interact with remote server objects can be written once the remote interfaces are carefully defined even after the remote objects are deployed.

Conclusion: Remote method Invocation allows you to build Java applications that are distributed among several machines. Remote Method Invocation allows a Java object that executes on one machine to invoke a method of a java object that execute on another machine. This is an important feature, because it allows you to build distributed applications.

© Vaishnavi 11/2/25

Distributed System.

DPU

Practical No. 2

Problem Statement: To develop any distributed application with CORBA program using JAVA IDL.

Tools: Java programming Environment, JDK 1.8.

Related Theory: Common Object Request Broker Architecture (CORBA):

CORBA is an acronym for common object request broker Architecture. It is an open source, vendor-independent architecture & infrastructure developed by the object Management Group (OMG) to integrate enterprise application across a distributed network. CORBA specifications provide guidelines for such integration applications. based on the way they want to interact, irrespective of technology, hence all kind of technologies can implement these standards using their own technical implementations.

When two application systems in a distributed enviroment interact with each other, there are quite a few unknown between those applications system, including the technology they are developed in, the base operating system they are running on, or system configuration. They communicate mostly with the help of each other's network address or through a naming service. Due to this, these

applications end up with quite a few issues in integration, including content mapping mismatches.

An application developed based on CORBA standards with standard Internet Inter-ORB Protocol (IIOP), irrespective of the vendor that develops it, should be able to smothly integrate & operate with another application developed based on CORBA standards through the same or different vendor.

Except legacy applications, most of application follow common standards when it comes to object modeling, for example. All applications related to say, 'HR & Benifit'. maintain an object model with details of the organization employees with demographic information, benefits, payroll, & deductions. They are only different in the way they handle the details based on the country and region, they are operating for. For each object type, similar to the HR & Benifits system, we can define an interface using the Interface Definition language (OMG IDL) The contract between these applications is defined in terms of a interface for the server objects that the clients can call. This IDL interface is used by each client to indicate when they should call any particular method to marshal. The target object is going to use the same.

interface definition when it receives the request from the client to unmarshael in order to execute the method that was requested by the client operation. Again during response handling, the interface definition is helpful to marshael & unmarshael arguments on the client side once received. The IDL interface is a design concept that works with multiple programming languages including C, C++, Java, Ruby, python & IDL script. This is close to writing a program to an interface, a concept we have been discussing that most recent programming languages & frameworks such as spring. The interface has to be defined clearly for each object. The system encapsulate the actual implementation along with their respective data handling & processing, and only the methods are available to the rest of world through the interface. Hence, the clients are forced to develop their invocation logic for the IDL interface exposed by the application they want to connect to with the method parameters advised by the interface operation.

The following diagram shows a single-process ORB CORBA architecture with the IDL configured as client stubs with object skeletons. The objects are written

and a client for it as represented in the diagram. The client & server use stubs & skeletons as proxies, respectively. The IDL interface ~~allow~~ follows a strict definition, & even though the client & server are implemented in different technologies, they should integrate smoothly with the interface definition strictly implemented. In CORBA, each object instance aquires an object reference for itself with the electronic token identifier.

Conclusion: CORBA provides the network transparency, Java provides the implementation transparency. CORBA complements the Java™ platform by providing a distributed object framework, services to support, that framework & interoperability with other languages. The Java platform complements CORBA by providing a portable, highly productive implementation environment. The combination of Java and CORBA allow you to build more scalable & more capable applications than can be built using the JDK alone.

Mukunda 21/12/25

DPU

Practical No. 3

Aim: To develop any distributed application using Message Passing Interface (MPI)

Tools: Java Programming Environment, JDK 1.8 or higher, MPI library (mpi.jar), MPJ Express

Theory: Message passing is a popularly renowned mechanism to implement parallelism in applications; it is also called MPI. The MPj interface for Java has a technique for identifying the user & helping in lower startup overhead. It also helps in collective communication & could be executed on both shared memory & distributed system. MPJ is a familiar Java API for MPI implementation. mpi Java is near flexible Java binding for MPJ standards. Currently developers can produce more efficient & effective parallel applications using message passing.

A basic prerequisite for message passing is a good communication API. Java comes with various ready-made package for communication notably an interface to BSD sockets, & the Remote Method invocation mechanism. The parallel computing world is mainly concerned with 'symmetric communication, occuring in groups of interacting peers. This symmetric model of communication is captured in the successful Message PIs

DPU

Message-Passing Interface Basics:

Every MPI program must contain the preprocessor directive:

#include <mpi.h>

The mpi.h file contains the definitions & declarations necessary for compiling an MPI program.

MPI_Init initializes the execution environment for MPI. It is a 'share nothing' modality in which the outcome of any one of the concurrent process can in no way be influenced by the intermediate results of any of the other processes. Command has to be called before any other MPI, call is made, and it is an error to call it more than a single time within the program. MPI_Finalize cleans up all the extraneous mess that was first put into place by MPI_Init.

The principal weakness of this limited form of processing is that the processes on different nodes run entirely independent of each other. It cannot enable capability or coordinated computing. To get the different process to interact, the concept of communication is needed. MPI programs are made up of concurrent executing at the same time that in almost all cases are also communicating with each

other. To do this, an object called the 'communicator' is provided by MPI. Thus the user may specify any number of communicators within an MPI program, each with its own set of process. "MPI_COMM_WORLD" communicator contains containing all the concurrent processes making up an MPI program.
The size of a communicator is the number of processes that makes up the particular communicator. The following function call provides value of the number of processes of the specified communicator.

MPJ Express: is an open source Java message passing library that allows application developers to write & execute parallel applications for multicore processors & compute clusters / clouds.
The software is distributed under the MPJ licence. MPJ Express is a message passing passing library that can be used by application developers to execute their parallel Java applications on compute clusters or networks of computers.
MPJ Express is essentially a middleware that supports communication between individual processors of cluster.
The programming model followed by MPJ Express is Single Program Multiple

DPU

Date:

Conclusion: There has been a large amount of interest in parallel programming using Java. mpj is an MPI binding with Java along with the support for multicore architecture so that user can develop the code on its own laptop or desktop. This is an effort to develop & run parallel programs according to MPI standard.

Shrivinder
27/2/25

DPU

Practical No. 4

Title: Implement Berkeley algorithm for clock synchronization

Tools: Anaconda, Python language.

Theory: Berkeley's Algorithm is an algorithm that is used for clock synchronization in distributed system. This algorithm is used in cases when some or all system of the distributed network have one of these issues.
A. The machine does not have an accurate time source.
B. The network or machine does not have a UTC server.
The algorithm is designed to work in a network where clocks may be running at slightly different rates & some computers may experience intermittent communication failures.
The basic idea behind Berkely's Algorithm is that each computer in the network periodically sends its local time to designed "master" computer, which then computes the correct time for the network based on the received timestamps. The master computer then sends the correct time back to all the computers in the network, & each computer set its clock to the received time.
Distributed system contains multiple nodes that are physically separated but are linked together using a network.
Berkeley's algorithm.
In this algorithm, the system chooses a node as

master / leader node. This is done from pool nodes in the server.

The algorithm is -

- An election process chooses the master node in the server.
- The leader then polls followers that provide their time in a way similar to cristian's Algorithm, this is done periodically.
- The leader then calculates the relative time that other nodes have to change or adjust to synchronize to the global clock time which is the average of times that are provided to the leader node.

Let's summ up steps followed to synchronized the clock using the Berkeley algorithm, Nodes in the distributed system with their clock timings -

N1 - 14:00 (master node)

N2 → 13:46

N3 → 14:15

Step 1: The leader is elected, node N1 is the master in the system.

step2: leader requests for time from all nodes.

N1 - time - 14:00

N2 → time: 13:46

N3 → time : 14:20

step3 → The leader averages the times & sends the correction time back to the nodes.

N1 → Corrected Time 14:02 (+2)

N2 → Corrected time 14:02 (+16)

N3 → Corrected time 14:02 (-18)

This shows how the synchronization of nodes of a distributed system is done using Berkely's algorithm.

Working principle of the Berkeley Algorithm.

Step 1: Time collection

- The master node sends time requests to all the slave nodes in the system.
- Each slave responds with its current local time.

Step 2: Time adjustment Calculation.

- The master node calculates the average clock time by considering all the responses.
- The master ignore faulty or significantly deviating clocks to prevent errors.
- The master then computes the clock offset (difference between each node's time & the average time).

Step 3: Clock synchronization.

- The master sends time adjustment (offset) to each slave, instructing them to speed up or slow down their clocks accordingly.
- Instead of sending the absolute time, the master only sends the required adjustment, minimizing errors due to network delays.

Applications:

- Distributed Computing systems.
- Cloud computing
- Financial & banking system where precise time synchronization is required.

Advantages:

DPU

- Decentralized.
- Fault ~~tole~~ tolerant.
- Efficient.

Conclusion: Thus we studied, how to implement Berkeley algorithm for clock synchronization.

Attended
27/2/25

DPU

Practical No. 5

Aim: Implement token ring Based mutual exclusion algorithm.

Theory: In a distributed system, multiple processes or nodes often require access to shared resources. Mutual exclusion ensures that only one process can access the critical section (CS) at a time to prevent data inconsistency & race conditions.

The Token Ring Algorithm is a well-known distributed mutual exclusion that ensures safe access to shared resources. It operates on a logical ring topology, where a special message is passed among the processes to grant access to the critical section.

Key Concept of Token Ring Algorithm.

1. Logical Ring Structure.
   - The processes are arranged in a logical ring, where each process has a unique ID and knows its next neighbor in the ring.
   - The last process in the ring connects back to the first process, forming a closed loop topology.
2. Token (Control Message).
   - A unique token is generated in the system.
   - The token is a small data packet used as a control mechanism to grant access to the critical section.
   - Only the process holding the token can enter the critical section.
3. Critical Section Execution.
   - A process can enter the critical section only if it has the token.

DPU

• After executing the critical section, the process forwards the token to the next process in the ring.

4. Token Circulation:

• If a process does not need the token, it simply forwards it to the next process.

• The token keeps circulating in the ring, ensuring that all processes get a fair chance to enter the critical section.

Working of the Token Ring Algorithm.

1. Initialization

• The processes are organized in a logical ring structure.

• One token is initially assigned to a process to start the execution.

2. Requesting the Critical Section.

• A process waits for the token.

• If it receives the token, it enters the critical section.

• If it does not have the token, it waits for its turn.

3. Executing in the Critical Section.

• The process executes the critical section while holding the token.

• It performs the necessary operations on the shared resources.

4. Releasing the Critical Section.

• After completing its execution, the process forwards the token to the next process in the ring.

5. Token passing

• The token is passed sequentially from one process to another in a circular manner.

• The cycle repeats, ensuring fair & deadlock-free

execution:

Example of Token Ring Algorithm Execution.

Consider five processes (P0, P1, P2, P3, P4) arranged in logical ring:

P0 → P1 → P2 → P3 → P4 → P0

1. Initially, P0 holds the token & enters the critical section.
2. After executing its task, P0 release the token & passes it to P1.
3. P1 receives the token & executes its critical section.
4. P1 passes the token to P2, & the process continues.
5. If P3 does not need the token, it forwards it to P4.
6. Once P4 finishes, it passes the token back to P0.
7. The token continues circulating in the ring, ensuring fairness

Advantage of Token Ring algorithm

1. Fairness:
   - → Every process eventually get the token, preventing starvation.
   - → No process, can jump ahead in priority, ensuring an ordered execution.
2. Deadlock Prevention:
   - Since only one token exists, there are no circular waits, avoiding deadlock.
3. Starvation-free.
   - → As the token moves sequentially, each process gets a turn within finite time.
4. Scalability for small system.

DPU

- Works efficiently for small to medium-sized distributed system.

Conclusion: Thus we studied about token ring based mutual exclusion.

C

Unlocked
7/4/25

DPU

Practical NO:6

Title: Implement Bully & Ring algorithm for leader election.

Tools: Java Programming Environment, JDK 1.8 Eclipse Neon.

Related Theory: Election Algorithm.

1] Many distributed algorithms require a process to act as a coordinator.
2] The coordinator can be any process that organizes actions of other processes.
3] A coordinator may fail.
4] How is a new coordinator chosen or elected?

Assumptions:
Each process has a unique number to distinguish them. Processes know each other's process number.
There are two types of Distributed Algorithms.
1] Bully Algorithm.
2] Ring Algorithm.

Bully Algorithm:
A. When a process, P, notice that the coordinator is no longer responding to request, it initiates an election.
1. P send an Election message to all processes with higher numbers.
2. If no one responds, P wins the the election & becomes a coordinator.
3. If one of the higher-up's answers, it takeover,

P's job is done.

B. When a process get an Election message from one of its lower-numbered colleagues.

1. Receiver send an OK message back to the sender to indicate that he is alive & will take over.
2. Eventually, all processes give up apart of one, & that one is the new coordinator.
3. The new coordinator announces its victory by sending all processes a Co-ordinator message telling them that it is the new coordinator.

C. If a process that was previously down comes back:

1. It holds an election.
2. If it happen to be highest process currently running, it will win the election & take over the coordinators job.

"Biggest guy" always win & hence the name bully algorithm.

Ring algorithm

Initiation:

1. When a process notices that coordinator is not functioning:
2. Another process initiates the election by sending "Election" message.

Leader Election:

3. Initiator sends the message to it's successor
4. At each step, sender adds it own its own process number to the list in the message.

5. When the message get back to the process that started it all: The message comes back to initiator. In the queue the process with maximum ID Number wins.
Initiator announces the winner by sending another message around the ring.

Designing the solution:
A. For Ring:
Algorithm Initiation:
1. Consider the Process 4 understand that process 7 is not responding.
2. Process 4 initiates the Election by sending "Election" message to it's successor with It's ID.
Leader Election:
3. Message comes back to initiator. Here the initiator is 4.
4. Initiator announce the winner by sending another message around the ring. Here the process with highest process ID is 6.
The initiator will announce that Process 6 is Coordinator.

Implementing the Solution.
1] Creating class for process which includes.
   i] state: Active/Inactive.
   ii] Index: store index of process
   iii] ID: process ID.
2] Import Scanner class for getting input from

Console.
3] Getting input from User for number of processes & store them into object of classes.
4. Sort there objects on the basis of process id.
5. make the last process id as "inactive".
6. Ask for menu .1. Election 2. Exit
7. Ask for initializing election processes.
8. These inputs will be used by Ring algorithm.

Conclusion : Election algorithm are designed to choose a coordinator. We have two election algorithms for two different configuration of distributions of distributed system.
The Bully algorithm applies to system where every process can send a message to every other process in the system & the Ring algorithm applies to system organized as a ring. In this algorithm we assume that the links between the process are unidirectional & every process can message to the process on its right only.

Utkarsh 7/4/25

DPU

Practical No: 7

Title: To create a simple web service & write any distributed application to consume the web service.

Tools: Java programming Environment, JDK & Net beans IDE with Glassfish Server.

Theory:

Web Service:

A web service can be defined as a collection of open protocols & standard for exchanging information among system or applications.

A service can be treated as a web service if:

- The service is discoverable through a simple lookup.
- It uses a standard XML format for messaging.
- It is available across internet / intranet networks.
- It is a self-describing service through a simple XML syntax.
- The service is open to, and tied to, any operating system/programming language.

Types of Web Service:

There are two types of web services:

1. SOAP: SOAP stand for simple object Access Protocol. SOAP is an XML based industry standard protocol. For designing & developing web services. Since it's XML based, it's platform and language independent. So, our server can be based on

Java & client can be on .NET, PHP etc & vice versa.
2. REST: REST (Representational state transfer) is an architectural style for developing web services. It's getting popularity recently because it has small learning curve when compared to SOAP. Resources are core concepts of RestFul web services & they are uniquely identified by their URIs.

Web service architecture:
As part of a web service architecture, there exist three major roles.
Service Provider is the program that implements the service agreed for the web service & exposes the service over the internet / intranet for other applications to interact with.
Service Requestor: is the program that interacts with the web service exposed by the service. Provider. It make an invocation to the ~~ser~~ web service over the network to the Service Provider & exchanges information.
Service Registry acts as the directory to store references to the web services.
The following are the step involved in a basic SOAP web service operational behaviour.
1. The client program that wants to interact with another application prepares its request content as a SOAP message.
2. Then, the client program sends this SOAP message to the server web service as an HTTP.

DPU

POST request with the content passed as the body of the request.

3. The web service plays a crucial role in this step by understanding the SOAP request & converting it into a set of instruction that the server program can understand.

4. The server program processes the request content as programmed & prepares the output as the response to the SOAP request.

5. Then, the web service takes this response content as a SOAP message & reverts to the SOAP HTTP request invoked by the client program with this response.

6] The client program web service reads the SOAP response message to receive the outcome of the server program for the request content it sent as a request.

SOAP web service:

Simple Object Access Protocol is an XML-based protocol for accessing web services. It is a W3C recommendation for communication between two applications, and it is a platform- and language-independent technology in integrated distributed application. While XML & HTTP together make the basic platform for web services, the following are the key components of standard SOAP web services.

Universal Description, Discovery & Integration UDDI is an XML based framework for describing, discovering & integrating web services. It acts as directory of web service interface described in the WSDL language.

Web Services Description language : WSDL in an XML document containing information about web services, such as the method name.

Conclusion : The assignment, described the web services approach to the Service Oriented Architecture concept. Also, described the Java APIs for programming web services & demonstrated examples of their use by providing detailed step-by-step examples of how to program web services in Java.

Distributed System.
Assignment-1

1] What are the general characteristics of inter-process communication.

→ Inter-process communication refers to the mechanism that allow processes to communicate & synchronize with each other. Here are its general characteristics.

1] Synchronization: Prevents race condition when multiple process access shared resources. Ensures orderly execution of dependent processes.

2] Data Exchange: Allow process to share information efficiently. It can be done using shared memory.

3] Communication Mechanism: Includes pipes, message queues, shared memory & socket. Each mechanism has different use cases based on speed & complexity.

4) speed: Shared memory is faster since it avoids kernel involvement. Message passing is slower due to system call overhead.

5] Security: Access control ensures only authorized processes communicate. Data integrity must be maintained to prevent corruption.

6] Resource sharing: Allows processes to share memory, files & devices. Helps reduce redundancy & improve efficiency.

7] Scalability: Can work between processes on the same or different systems. Supports distributed computing & multi-threading.

8] Reliability: Ensures message reach the correct recipient without loss. Uses acknowledgement & error handling for consistency.

2] What are various types of Synchronous & asynchronous

communication in IPC & why it is blocking receive has no disadvantages in Java.

→ Types: 1] Synchronous Communication (Blocking)
- Sender & receiver must wait for each other to complete communication

Example: Blocking message passing: The sender waits until the receiver acknowledges the message.
- Synchronous RPC (Remote Procedure Call): The client waits for the server to process & return a response before proceeding.

2] Asynchronous Communication (Non-blocking):
- The sender & receiver can continue execution without waiting.

Example: Message queues: The sender adds messages to a queue, & the receiver processes them later.
- Non-blocking Sockets: Data transmission happens in the background, allowing other tasks to continue.

In Java, blocking receive is not a major disadvantage because:

1] Threading Support: Java provides multi-threading allowing one thread to block while others continue execution

2] Efficient Resource Management: Java handles I/O operations efficiently with synchronized I/O streams & sockets, minimizing CPU wastage.

3] What is Network Virtualization? What are the types of overlay networks?

→ Network Virtualization:

Network Virtualization is the process of creating multiple virtual networks on top of a physical network infrastructure. It allows multiple users or applications to share network resources efficiently while maintaining isolation & security.

- Key features: Decouples network functions from hardware.
- Enhance scalability, security & flexibility.
- Support multi-tenancy in cloud environments.

Types of Overlays Networks:

1) peer-to-peer: Direct communication between nodes without a central server.
2) Virtual private Network: Encrypts & tunnels the network traffic over a public network.
3) Software-Defined Networking (SDN) overlays.
- Uses a centralized controller to manage virtual networks.
4) Content Delivery Networks (CDNs)
- Distributes content across multiple servers for faster delivery.
5) Blockchain & Decentralized Networks: Distributed ledger-based networks for secure transactions.

4) What are the various advantages of overlay networks.

→ 1) Scalability: Support large-scale network by adding virtual layers without modifying the underlying infrastructure.
2) Security: Provides encryption, access control, & isolation from the physical network.
3) Flexibility & Customization.
- Allows customized routing & traffic management without changing physical hardware.

4] Resilience & fault tolerance:
- Can dynamically reroute traffic incase of network failures.

5] Efficient Resource Utilization: Multiple logical networks can run on the same physical infrastructure, reducing costs. Example. VXLAN enables multi-tenant cloud environment.

6] Improved Performance: Optimize routing & reduces latency for global applications.

3] Describe lamport's clock algorithm & give an example of how to order events in a distributed system. What are the algorithm's limitations?

→ lamport's clock algorithm.
- lamport's logical clock is timestamping mechanism used to established an order of events in a distributed system where there is no global clock.

Algorithm steps:
1] Initialization: Each process P_i maintains a logical clock LC_i, initially set to 0.
2] Event Occurrence: Internal event: increment the logical clock $LC_i = LC_i + 1$
- message sending: Attach the current timestamp LC_i to the message.
- message Receiving: Update the local clock:
  $LC_i = \max(LC_i, LC_{received}) + 1$
- Ensures causality

Limitations:
1] Does not captures Concurrency:
2] Requires Additional Mechanism for total ordering.

Assignment 12

1] Why do we need an election algorithm? Using appropriate diagram. Describe the bully algorithm step by step?

→ → Election algorithm are used in distributed system to select a leader when the current leader fails. The leader is responsible for critical tasks like resource allocation, synchronization & decision-making.

Algorithm: The bully algorithm is a leader election algorithm in a distributed system where processes have unique numerical IDs & the highest-numbered process is elected as the new coordinator.

Key steps: i) Failure detection: A process detects that the current coordinator has failed.
- The failure can be detected via timeout.

2] Election initiation: The detecting process starts an election by sending an Election message to all processes with a higher ID.

3) Response from higher Processes: If a higher-numbered process is alive, it responds with an OK message.
- The responding process takes over the election & starts a new election.

4] Declaring the New Coordinator: The highest surviving process sends a coordinator message to all other processes, announcing itself as the new leader.

2] What are the key issues in replica Management?

Explain the following with respect to the content replication & placement with suitable diagram:
⇒ 1] permanent 2] Server-Initiated 3] Client-Server

→ Replica Management in distributed system deals with creating, maintaining, & synchronizing copies of data across multiple location to improve performance, availability & fault tolerance.

1] permenant Replicas: Permenant replicas are predefined & do not change over time.
: These replicas are placed on fixed servers for high availability & reliability.

2] Server-Initiated Replicas:
→ The server dynamically creates replicas based on demand patterns. When a specific data item is requested, the server replicates it to reduce latency.

3] Client-Initiated replicas:
: The client requests a local copy of data for faster access. Typically used in browser caching & offline storage.

3] What is a primary-based protocol in a consistency protocol? Explain the working of replicated write protocol with active replication.

→ A primary-based consistency protocol is a method for ensuring data consistency in distributed systems where a primary replica is responsible

for handling all write operations, other replicas receive update from the primary.
→ A replicated write protocol ensures that write operations are performed on all replicas, maintaining consistency across distributed nodes.
How replication Works: 1) Client sends a Write request.
→ A Client sends a write request to all replicas.
- The request is broadcasted using a totally ordered multicast to ensure all replicas receive updates in the same order.
2- Replicas Executes the Operation: Each replica independently processes the write request.
- Since they execute the same operation in the same order, they remain consistent
step 3: Response to the client: Each replica sends back a confirmation.

4] What is check pointing in a distributed system? Explain the working of coordinated check pointing recovery mechanism.

→ Checkpointing is a technique used in distributed system to periodically save the state of processes so that the system can recover from failure without restarting from scratch.
- Mechanism: Coordinated checkpointing ensures that all processes in a distributed system take a checkpoint together, ensuring a globally consistent state.
How it works:

1] Initiation: A coordinator Initiater the checkpointing process; It sends a checkpoint request to all other processes.
2] message Handling: Each process pauses communication & saves its state to stable storage.
3] Acknowledgement: Once all processes ~~that~~ take their checkpoints, they send a CHECKPOINT DONE acknowledgement to the coordinator.
4] Commit & Rollback: If all processes successfully take a checkpoint the system commits the checkpoint.

5] What are two primary reason for replication? Explain Casual consistency model with suitable example using distributed shared database.

→ 1] Fault Tolerance: If one replica fails, others can serve request, ensuring high availability
2] Performance Improvement: Reduces latency by storing replicas closer to users.
Casual Consisteng:
· Casual Consistency ~~enter~~ ensures that operations that are casually related appear in the same order across all replicas, but concurrent operations can be seen in different orders.
Key properties: If operation A influence operation B, then A must be seen before B.
If operations are independent, they may be seen in any order.
Example: Social media Post & Comments.

Unicircle 18/2/25

Assignment-3

1] Describe the architecture of Sun Network file system in details.

→ The Sun Network file system is a distributed system protocol that allows a client to access files on a remote server as if they were local. It operate over TCP/IP network providing file sharing in heterogeneous environment.

NFS Architecture:

1) NFS server: The server provides access to files stored in directories shared over the network.
2) NFS client: The client request file access operations such as reading, writing, or modifying files.
3) Remote Procedure Call: NFS relies on RPC for communication between client & server. Each NFS operation is an RPC request.
4) File system: On the server side, NFS allows sharing files stored on local file systems.
5) Mounting: A client mounts a remote directory to make it appear as a part of its local file system.

2] What is a Directory Service? What is the difference between DNS & X.500? Describe in detail the components of X.500 service architecture.

→ A directory service is a software application or system that provides access to a distributed directory across a network.

It is primarly used for storing, retrieving & managing information in a distributed manner.

Components of Directory services:

- Directory Information Tree: Represent the hierarchical structure of the directory.
- Directory Server: Manages the directory & process queries.
- Directory client: makes queries to the directory services.
- Access Control: Ensures secure access to the directory based on user permissions.
- X.500 Service Architecture:
- Directory Information Tree: Represent a hierarchical structure for storing directory entries.
- Directory System Agent: (DSA): The DSA is a server responsible for storing & managing directory information.
- Directory Access Protocol: The protocol used by DUAs to interact with DSAs.

3] Design the design of a peer to peer download system designed to support very large multimedia files. How does the Bit Torrent protocol operate.

→ A peer to peer download system allows users to share large multimedia files directly with each other without requiring a centralized server. Key components of P2P system include:

1. Peers: Each participant in the network that acts both as a client & a server. They download & upload parts of the file.
2. File chunking: The large multimedia file is divided into smaller chunks, allowing different peers to download parts of the file concurrently.
3. Tracker: A central server maintain information about which peers have which chunks of the file. It helps peers find each other.
4. Swarming: Peers download chunks from multiple sources simultaneously improving speed & efficiency.

Bittorrent Protocol: is a popular P2P protocol for distributing large files. The key operations,

1) File chunking: The file is broken into small chunks.
2) Peer Discovery: Clients find peers via a tracker.

4] What are web services? Describe with a suitable diagram the general organization of the Apache web server

→ Web services allow applications to communicate with each other over a network using standardized protocol. They enable interoperability between different platform & technologies.

• Apache Web Server Organization.

The Apache web service is a widely used open-source web server software that serves web content over HTTP. The general organization of Apache includes.

1) Client request: The client sends an HTTP request to the Apache server.
2. Document Root: Apache retrieves file from the document root directory.
3- Modules: Apache uses modules to process requests
4. Response: The server returns the requested content.
5. logging: Apache log requests in access log for monitoring & debugging

5] Explain the following in brief.

3] i) Wearable devices:
- Wearable devices are portable, small electronic gadget designed to be worn on the body. These devices monitor health, fitness, & environment. (eg. smartwatches, fitness trackers, health)

2) PVM: (parallel Virtual Machine):
PVM is a software tool that enables a distributed computing environment, allowing programs to run on multiple machines simultaneously, making use of parallel processing to solve complex computational problem.

5] JINI: is a Java-based technology for building distributed system. It supports dynamic discovery & interaction of networked devices & services, providing a framework for building distributed scalable system.

Invisible 27/2/25

Assignment-4

1] Explain with an application key component of Service Oriented Architecture.

→ Service-Oriented architecture is a design pattern that allows different services to communicate with each other over a network. It enables interoperability, reusability & scalability in distributed system.

Components of SOA.

1) Service Provider: A service provider hosts & offers service that client can use. Ex. A travel booking company provides services for flight booking, hotel reservations & car rentals.

2] Service Consumer: A service consumer is any client application that request a services from the provider. Ex. A user visits a travel booking website or mobile app to search for flights & hotel.

3] Service Registry: A directory where service providers register their services, allowing consumers to find the

4] Service Contract: A formal aggreement between the provider & consumer that defines how the service is used.

2] Explain in brief the following Distributed System monitoring tools.

1] Zabbix: Zabbix is an open-source monitoring tool for networks, servers, applications & cloud services. It collect real-time data using agents, SNMP or IPMI & provides

visualization through dashboards & alerts. It supports auto-discovery, triggers based notifications & integrations with third-party tools.

2. iNagios: iNagios is a web-based front end for Nagios, a widely used open-source monitoring tool. It simplifies monitoring by providing an intuitive UI for managing alerts, hosts & services. iNagios retains Nagios core capabilities such as plugin support & customizable alerts, while enhancing usability & accessibility.

3] Explain in brief following Microkernels.
i] mach ii] chorus:

→ mach: Mach is a microkernel developed at Carnegie Mellon University that provides minimal core functionalities like interprocess communication, memory management & task scheduling. It supports message-passing between processes, enabling modular OS design. Mach was used as foundation for operating system like NEXTSTEP, macOS, & influenced windows NT.

ii] CHORUS: is a distributed real-time microkernel designed for scalability & fault tolerance. It provides lightweight processes & message-based IPC for building distributed system. Initially

developed by INRIA & later commercialized, CHORUS influenced Unix System V & was used in telecommunications & embedded system.

4] Why is computer clock synchronization necessary? Describe the design requirements for a system to synchronize the clocks in a distributed system.

→ In distributed system, each computer maintains its own local clock. However, due to clock drift the clocks of different nodes may become unsynchronized over time. This can lead to various issues such as:

- Inconsistent Event Ordering - If events are timestamped incorrectly, causality may be violated, leading to incorrect execution sequences.
- Data Inconsistency - In database system, an incorrect order of transaction can cause integrity issues.
- Security Vulnerabilities → Authentication protocols relying on time-based tokens may fail due to clock mismatches.
- Failure Recovery Issues: Distributed logs & backups require accurate timestamps for recovery.
- Coordination & Communication Issues: Applications such as distributed file system, cloud computing & IoT require synchronized clocks for proper coordination.
- Design Requirements for Clock Synchronization System
1] Accuracy: The system should ensure that all clocks

remain within a small acceptable deviation from the reference time.
- Fault Tolerance: The system should handle failures of nodes or communication links without significant degradation in synchronization.
- Scalality: The system should efficiently synchronize clocks even as the number of nodes increases.

5] What is NTP? With the help of diagram, describe how NTP works.

→ Network Time protocol is widely used clock synchronization protocol that synchronizes computer system clocks across a network. It operates over UDP & ensures that all system in a network maintain accurate time by synchronizing with a reference time source.

How NTP works?

NTP follows a hierarchical structure consisting of different levels of time sources, called Stratum levels:

1. Stratum 0: Reference clocks.
2. Stratum 1: NTP server directly connected to stratum 0 clocks.
3. Stratum 2: Servers synchronizing with stratum 1 server
4. Stratum 3: lower level clients synchronizing with higher-stratum servers.

- NTP synchronization Process:

1. Client Request, 2. Timestamp Exchange, 3. Clock Adjustment 4. Periodic Synchronization.

Assignment-5

1] Why is the Berkeley algorithm used? Describe how it works using pseudocode.

→ The Berkeley algorithm is used for clock synchronization in distributed system where no machine has a perfect clock. It is particularly useful when:
1. No global time source:
2. Nodes have different clock drifts overtime.
3. A master node synchronize all nodes by averaging their clocks.

How the Berkeley algorithm works:
1. A master node initiates synchronization.
2. It requests the current time from all participating slave nodes.
3. Each slave node sends its local time to the master.
4. The master :) Computes the average clock time.
:) Calculates adjustments for each node.
5. The master sends the adjustments to each slave.
6. Each slave modifies its clock accordingly.

This ensures all nodes synchronize to a common time improving coordination in distributed system.

2] What is MPI? Describe the point-to-point communication in MPI with suitable diagram Explain in brief, the following send operations in MPI: MPI_Ssend, MPI_Bsend, MPI_Rsend, MPI_Isend.

→ MPI is a standardized & portable message-passing library designed for parallel computing

in distributed memory systems. It enables communication between processes running on multiple nodes in a cluster or supercomputer. MPI supports both point-to-point & collective communication.

Point-to-Point Communication in MPI: Point-to-Point communication in MPI involves data transfer between two processes:

- Sender Process: Sends a message.
- Receiver Process: Receives the message.
- User blocking & non-blocking communication.
- MPI_Ssend: Ensures that the message is received before returning.
- MPI_Bsend: User a user-provided buffer for sending data.
- MPI_Rsend: Assumes that the receiver is ready to receive the message.
- MPI_Isend: Initiates the send operation & returns immediately.

3] What are the key issues in Replica Management? Explain Management? the following with respect to content replication & placement with suitable diagram. Permanent Replicas, Server-Initiated Replicas, Client-Initiated Replicas.

→ Replica management is distributed system deals with maintaining multiple copies of data across different location to improve availability, fault tolerance & performance. The key issues includes

1. Consistency: Ensuring all replicas have the same data & remain updated.
2. Synchronization: Managing updates across replicas efficiently.
3. Scalability: Handling large number of replicas efficiently.
4. fault tolerance: Ensuring system reliability in case of failure.

1] Permanent Replicas: These are fixed replicas that do not change location frequently.
2] Server-Initiated Replicas: The server decides when & where to create new replicas
3] Client Initiated Replicas: Client request local copies of data based on usage patterns.

4] What is a primary-based protocol in a consistency protocol? Explain the working of primary-backup protocol with suitable diagram.

→ A primary-based protocol is a consistency mechanism used in distributed system where one designated node, called the primary, is responsible for coordinating updates to a shared resources. The primary ensures that all replicas remain consistent by controlling write operation before propagating changes to other replicas.
primary-backup protocol: The primary-Backup protocol is a fault tolerant mechanism in which one server acts as the primary, while on or more servers acts as backups. The primary server

processes all write operation & propogates the updates to the backups.

Working of primary-Backup protocol:

1. Read Operation: Client can directly read data from any replica. No need for coordination, as backups always have consistent data.
2. Write Operation: The client sends a write request to the primary server. The primary updates its own data & then forward the updates to all backup servers.
3. Failure Handling: If the primary fails, a backup is promoted to become the new primary.

5] What is the distribution commit problem? Discuss how this problem is solved using the two-phase commit protocol with suitable diagram.

→ The distributed commit problem occurs in a distributed database or system when multiple nodes must agree on whether to commit or abort a transaction. The main challenge is ensuring atomicity - either all nodes commit the transaction or all abort it, even in the presence of failures.

Two phase Commit Protocol:

1. Phase 1: Prepare (Voting phase): The coordinator sends a "prepare" message to all participating nodes. Each participant checks if it can commit the transaction & replies with either "Yes" or "No".
2. Phase 2: Commit: If all participants vote "Yes" the coordinator sends a "Commit" message to all & they finalize the transaction.

Assignment - 6

1] What are the key design issues for distributed file system? Describe the requirement for distributed file system.

→ Key Design Issues for Distributed File System.

* ~~A Distr~~ i] Transparency: Access Transparency: Users should access remote files the same way as local files.

2] Location Transparency: Users should not need to know where a file is physically stored.

2] Concurrency Control: Ensuring multiple users can access or modify a file without conflicts. Preventing race conditions and maintaining consistency.

3] Fault Tolerance & Reliability: The system should recover from hardware or network failure.

4] Scalability: The system should efficiently handle growing users & data without performance degradation.

5] Security & Access Control: User authentication and authorization to prevent unauthorized access.

6] performance & efficiency: Minimizing access latency by ~~a~~ caching frequently used data.

2] Why Quality of Service Management is important in Distributed multimedia system? Describe QoS manager responsibilities using suitable graphical representation.

→ In Distributed multimedia system applications, such as video conferencing, online streaming, & VoIP require high-quality real-time data transmission.

Quality of Service management ensures:
1. Timely Delivery: Prevent delays in audio/vedio transmission.
2. Bandwidth Management: Allocates resources efficiently to avoid congestion.
3. Jitter Reduction: Maintains smooth playback by minimizing variations in delays.

Responsibility of QoS Manager.
1. Admission Control: Determines if new user can be accommodated without degrading existing sessions.
2. Resource Reservation: Allocates bandwidth processing power & storage before service starts.
3. Traffic shaping & Scheduling: Control data flow to prevent congestion & ensure smooth multimedia streaming.
4. Error Detection & Recovery: Implements techniques like forward error correction & retransmission to minimize packet loss.
5. Adaptive QoS mechanism: Dynamically adjusts quality based on network conditions.

3] Describe, using the appropriate diagram, how web service is implemented in horizontal distribution using web server clusters.

→ Horizontal distribution involves addition multiple web servers to distribute the load efficiently. This is implemented using a web server cluster, where multiple servers handle incoming requests in parallel, improving scalability, fault tolerance & performance.

Architecture of a web service.

1. Load balancer:
   - Distributes incoming client request across multiple web servers.
   - Ensures efficient resource utilization & prevents server overload.
2. Web server Cluster:
   - multiple web server handle HTTP requests, can be stateless or stateful.
3. Application Servers: If needed, application logic can be processed separately from the web server layer.
4. Database Server: Stores dynamic content. Can be replicated for high availability.

4) Explain in brief, the two places of client-side web caching & explain cooperative caching with suitable diagram.

→ Client-side web Caching

client-side web caching improves performance & reduces bandwidth usage by storing copies of web resources locally. The two primary places where caching occurs on the client side are:

1. Browser Cache: The web browser stores static resources like HTML, CSS, Javascript, image & videos. When the user revisits a website, the browser loads cached content instead of requesting it again from the server.
2. Proxy Cache: A caching server sits between client & web servers. It stores frequently accessed web page & serves them directly, reducing load on the origin server.

eg. Squid Proxy, NGINX.

Cooperative Caching:

- Cooperative Caching: is a distributed caching approach where multiple caching nodes share cached data, improving efficiency across a network. Instead of each client, they cooperate by forwarding cache requests to nearby peers before fetching from origin server.
- Benifits:
  - Reduce duplicate downloads
  - Minimize latency by serving cached content from a nearby node
- Balance load across the ~~network~~ caching network.

5] What is Service Oriented Architecture? Explain the various SOA components. How does it differe from traditional software architecture?

→ Service-Oriented Architecture is a software design approach where applications are built as collection of loosely coupled services that communicate over a network. Each service performs a specific business function and can be reused across different applications.

- Key Component of SOA:

1. Service provider: Develops & offers services.
   - Registers services in a service directory.
2. Service consumer: Request & uses services provided by the service provider.
3. Service Registry: Acts as a centralized directory where available services are listed.
4. Service Contract: Defines how service interact using WSDL

Unicode 24/3/25