Distributed Systems Viva Question Bank - Detailed Answers

Assignment 1: Remote Method Invocation (RMI)

- **What is RMI?** Remote Method Invocation (RMI) is a Java API that allows an object residing in one Java Virtual Machine (JVM) to invoke methods on an object running in another JVM. It is used for building distributed applications in Java and supports object-to-object communication over a network.
- **What is the basic principle of RMI architecture?** The basic principle behind RMI is to provide remote communication between Java programs. RMI enables the execution of methods across JVMs by abstracting the details of network communication, using a combination of stubs and skeletons to invoke remote methods.
- **What are the layers of RMI Architecture?**
    1. **Stub and Skeleton Layer**: Provides the interface for client and server communication.
    2. **Remote Reference Layer**: Maintains the reference of remote objects.
    3. **Transport Layer**: Responsible for setting up connections, managing communication, and transferring data.
- **What is the role of Remote Interface in RMI?** The remote interface defines the methods that can be called remotely. It must extend the java.rmi.Remote interface. Classes implementing this interface must handle the actual method logic.
- **What is the role of the java.rmi.Naming class?** This class provides methods to bind, unbind, rebind, lookup, and list remote objects in the RMI registry. It allows clients to obtain references to remote objects using a URL-like naming convention.
- **What is meant by binding in RMI?** Binding is the process of associating a name with a remote object so that clients can look it up in the RMI registry. Binding is done using the Naming.bind() or rebind() methods.
- **What are the steps involved to make an RMI program work?**
    1. Define a remote interface.
    2. Implement the interface in a remote object class.
    3. Compile the code and generate stubs using the rmic tool.

4. Start the RMI registry (rmiregistry).
5. Register the remote object.
6. Run the client to invoke remote methods.

- **What is the role of a stub in RMI?** A stub is a proxy on the client side that represents the remote object. It forwards the client's method invocation to the actual remote object located on the server.
- **Explain Marshalling and Demarshalling.** Marshalling is the process of converting data or object parameters into a byte stream for transmission. Demarshalling is the reverse process—reconstructing the original object from the received byte stream.
- **Explain Serialization and Deserialization.** Serialization is the process of converting an object into a byte stream to save it to a file or send it over a network. Deserialization is the process of converting the byte stream back into a copy of the original object.
- **What method is used by the RMI client to connect to remote RMI servers?** The Naming.lookup("rmi://host:port/serviceName") method is used to obtain a reference to the remote object registered with the RMI registry.
- **Define the following terms:**
    - **Remote Object**: An object whose methods can be invoked from another JVM.
    - **Server Object**: The actual implementation of the remote interface hosted on the server.
    - **rmiregistry**: A registry where remote objects are bound to names.
    - **rmic**: A compiler tool to generate stub and skeleton classes from compiled remote object classes.
- **Why are stubs used in RMI?** Stubs hide the complexity of remote communication. They make a remote method call appear like a local call by handling parameter marshalling, communication, and receiving results.
- **What is the function or role of skeleton in RMI?** The skeleton resides on the server side and receives method calls from the stub. It unpacks the request and invokes the appropriate method on the actual remote object. (Note: Skeletons were used in earlier versions of Java.)
- **How does communication with remote objects occur in RMI?** The client uses the stub to invoke a method, which is marshalled and sent over the network. The

skeleton receives the request, demarshals it, executes the method, and returns the result to the stub.

- **What is a Remote Object & Reference?** A remote object resides on a server and implements a remote interface. A remote reference is the reference used by a client to invoke methods on this remote object.
- **What is a Remote Interface?** A remote interface defines methods that can be called remotely. It acts as a contract between the client and the server.
- **What is binding in Client-Server?** Binding in a client-server model refers to registering a server object with a naming service or registry so that clients can look up the service using a known name.

Assignment 2: Common Object Request Broker Architecture (CORBA)

- **What is CORBA?** CORBA stands for Common Object Request Broker Architecture. It is a standard defined by the Object Management Group (OMG) that enables communication between distributed objects written in different programming languages and running on different platforms.
- **Where to use CORBA?** CORBA is typically used in distributed systems that require interoperability between components developed in various languages. It's suitable for large-scale enterprise systems, such as telecom, financial, and aerospace applications.
- **What are the advantages of using CORBA?** CORBA provides platform and language independence, supports complex object interactions, and promotes reuse of components. It also has built-in features like transaction management and security.
- **What is ORB?** The Object Request Broker (ORB) is middleware that facilitates communication between client and server objects in CORBA. It locates the object, sends the request, and returns the result.
- **Explain the architecture of CORBA.** CORBA's architecture consists of clients, server objects, the ORB, stubs, skeletons, and the Interface Definition Language (IDL). Clients use stubs to send requests to server objects via the ORB, and server-side skeletons receive and execute the requests.
- **How does Java support CORBA?** Java supports CORBA through the Java IDL API, which allows Java programs to interface with CORBA objects using IDL. Java provides tools like idlj and packages such as org.omg.* for this purpose.

- **What is idlj?** The idlj tool is a compiler that takes IDL files as input and generates Java stubs and skeletons. These classes are used to implement and invoke CORBA objects in Java.
- **How to create CORBA objects using Java IDL?**
    1. Define the remote interface using IDL.
    2. Compile it using idlj to generate Java classes.
    3. Implement the server object.
    4. Start the ORB and register the object.
    5. Create a client to access the CORBA object.

Assignment 3: MPI / MPJ Express - Distributed Sum

- **What is MPI?** MPI (Message Passing Interface) is a standard for communication among processes running on a distributed memory system. It provides message-passing libraries for parallel computation.
- **Where to use MPI?** MPI is used in high-performance computing applications like weather modeling, scientific simulations, and parallel processing tasks requiring distributed computing resources.
- **What are the advantages of using MPI?** MPI is highly scalable, portable, and efficient. It allows complex data communication patterns and supports both synchronous and asynchronous communication.
- **What is MPJ Express?** MPJ Express is a Java-based MPI-like library that allows parallel programming in Java. It provides message-passing features similar to MPI and is used for academic and research projects.
- **What is parallel programming and how is MPI used in it?** Parallel programming involves dividing a task into subtasks that run simultaneously. MPI facilitates this by allowing multiple processes to communicate and coordinate, sharing intermediate results or data.
- **What is SPMD?** SPMD (Single Program Multiple Data) is a parallel programming model where each processor runs the same program but on different portions of data. It is commonly used with MPI.
- **What is a multiprocessor and multicomputer system?**
    - A **multiprocessor system** has multiple CPUs sharing a common memory and OS.

- A **multicomputer system** has independent computers connected via a network, each with its own memory and OS.

- **What is Unicast & Multicast?**
  - **Unicast** is one-to-one communication between a single sender and receiver.
  - **Multicast** is one-to-many communication where data is sent from one sender to multiple receivers.

- **Steps in implementing the MPI program in a multicore environment:**
  1. Initialize the MPI environment.
  2. Determine the number of processes and their ranks.
  3. Divide and distribute data.
  4. Perform local computations.
  5. Use collective communication (e.g., reduce, gather) to combine results.
  6. Finalize the MPI environment.

- **How to compile and run MPI program:**
  - Compilation: mpicc program.c -o program
  - Execution: mpirun -np 4 ./program (runs with 4 processes)

- **Different features of MPI:**
  - Portability
  - Scalability
  - Point-to-point and collective communication
  - Process synchronization
  - Support for heterogeneous networks

Assignment 4: Clock Synchronization using Berkeley Algorithm

- **What is a logical clock?** A logical clock is an abstract concept used in distributed systems to order events without relying on physical time. It assigns a numerical value to events to maintain a consistent ordering, e.g., Lamport Timestamps.

- **What is a global clock?** A global clock is a theoretical unified time source shared by all nodes in a distributed system. It is used to maintain a common time reference, although it is not physically feasible and must be approximated through synchronization.
- **Which algorithm is used for clock synchronization?** Clock synchronization in distributed systems can be achieved using the Berkeley algorithm, Cristian's algorithm, or the Network Time Protocol (NTP), depending on the system's requirements and environment.
- **How does the Berkeley algorithm achieve synchronization?** A master node polls all other nodes for their local times, averages the responses (excluding faulty values), and then sends the adjustment value to each node. This approach avoids dependency on a single time server and improves fault tolerance.
- **What is the purpose of clock synchronization in a distributed system?** Synchronizing clocks ensures that all nodes in a distributed system can agree on the ordering of events, maintain consistency, detect causality, and perform coordinated actions, which are essential for correct operation.
- **Explain clock skew or drift.** Clock skew is the difference in time between clocks in different systems. Clock drift refers to the gradual divergence of a clock's time from the correct time due to variations in oscillator frequency.
- **Explain any two physical clock synchronization methods:**
    1. **Cristian's Algorithm** – A client sends a request to a time server and adjusts its clock based on the server's response and estimated round-trip delay.
    2. **NTP (Network Time Protocol)** – Uses a hierarchical structure of time servers and statistical techniques to synchronize system clocks accurately over the Internet.

Assignment 5: Mutual Exclusion using Token Ring Algorithm

- **What is mutual exclusion?** Mutual exclusion is a property that ensures that multiple processes do not enter their critical sections simultaneously, preventing data inconsistency and race conditions in distributed systems.
- **What are the requirements of mutual exclusion?**
    1. **Safety** – Only one process should be allowed in the critical section.
    2. **Liveness** – Requests should eventually be granted.
    3. **Fairness** – Requests should be handled in the order they are made.

- **How can a mutual exclusion algorithm be implemented?** It can be implemented using centralized algorithms, distributed algorithms, or token-based algorithms such as Ricart-Agrawala or token ring.
- **Explain token and non-token based algorithms:**
    - **Token-based**: A special token is passed between processes. Only the process holding the token can enter the critical section.
    - **Non-token-based**: Processes request access and use a voting mechanism (e.g., Ricart-Agrawala) without any token.
- **What are the benefits of the token ring algorithm?**
    - Ensures fairness as the token circulates in a fixed order.
    - Reduces message complexity in low-contention situations.
    - Provides bounded time to access the critical section.
- **What are the different reasons a token may be lost?**
    - Node crash or network failure during token transmission.
    - Token corruption due to hardware/software error.
    - Improper handling or programming bugs.
- **What will happen if the token is lost?** If the token is lost, mutual exclusion is broken. A token regeneration algorithm must be invoked to detect the loss and generate a new token to restore normal operation.

Assignment 6: Leader Election – Bully and Ring Algorithm

- **What is the Ring Election Algorithm?** In the Ring Algorithm, processes are arranged in a logical ring. A process starts an election by sending a message around the ring. Each process appends its ID to the message. When it returns, the process with the highest ID is chosen as the coordinator.
- **What is the Bully Election Algorithm?** In the Bully Algorithm, when a process notices the coordinator has failed, it sends election messages to all processes with higher IDs. If no response is received, it declares itself as the leader. Otherwise, a higher process continues the election.

- **Explain with example the concept of Ring and Bully Algorithm:**
    - *Ring*: P3 detects failure, starts election. The message goes P3 → P4 → P5 → P3. P5 has the highest ID, so it becomes the new coordinator.
    - *Bully*: P2 notices failure, sends to P3, P4, P5. P5 responds and starts its own election, eventually becoming coordinator.
- **What are the different types of distributed algorithms for leader election?**
    1. Ring Algorithm
    2. Bully Algorithm
    3. Chang and Roberts Algorithm
    4. Invitation Algorithm
- **What is the Bully Algorithm for electing a leader?** It is an election algorithm where the process with the highest ID becomes the coordinator. It "bullies" lower ID processes into accepting it as the leader.
- **What is the algorithm for leader election?** The election algorithm involves detecting failure, initiating election, comparing process IDs, and choosing the process with the highest priority (usually ID) to become the leader.
- **Which algorithm is best – Bully or Ring?**
    - **Bully** is faster and requires fewer messages when higher-ID processes are active.
    - **Ring** is simpler, fairer in token passing but slower in large systems.
- **What is the difference between Ring and Bully Election Algorithm?**
    - **Ring** uses a circular message passing pattern and includes all nodes.
    - **Bully** uses direct messages to higher-ID processes and elects the highest available one quickly.

Assignment 7: Web Services

- **What is a Web Service?** A web service is a standardized way for applications to communicate over the Internet using protocols like HTTP and formats like XML or JSON. It allows interoperability between different systems.

- **Enlist a few examples of web services:**
    - Weather data APIs (e.g., OpenWeatherMap)
    - Currency exchange rate services
    - Payment gateways (e.g., PayPal API)
    - Social media APIs (e.g., Twitter API)
- **How to consume web services?** A client can consume a web service using HTTP requests through tools like Postman or programming libraries like HttpURLConnection in Java or requests in Python. The response is usually in XML or JSON.
- **Types of web services:**
    1. SOAP (Simple Object Access Protocol)
    2. REST (Representational State Transfer)
- **What is SOAP?** SOAP is a protocol for accessing web services. It uses XML for messaging and operates over various transport protocols like HTTP, SMTP. It is standards-based and supports security and transactions.
- **What is REST?** REST is an architectural style that uses standard HTTP methods (GET, POST, PUT, DELETE). It is stateless, cacheable, and works with resources represented by URIs.
- **Explain web service architecture in general:** The architecture includes a service provider (offers the service), service registry (directory for finding services), and service consumer (client). Communication happens via protocols like HTTP and message formats like XML/JSON.
- **Explain SOAP web service architecture:** SOAP architecture includes a client, SOAP request/response messages, WSDL (Web Services Description Language), and a SOAP server. Messages are exchanged over HTTP with strict standards and error handling.
- **Explain REST web service architecture:** RESTful architecture uses HTTP methods to perform actions on resources represented by URIs. It is lightweight, human-readable, and widely used in web and mobile apps.
- **Differentiate between SOAP and REST:**
    - SOAP uses XML only; REST can use XML, JSON.
    - SOAP is heavier and supports advanced features; REST is lighter and faster.

    - SOAP uses WSDL; REST uses HTTP verbs and URIs.
- **Java API used for SOAP:** Java provides JAX-WS (Java API for XML Web Services) for creating and consuming SOAP-based services.
- **Java API used for REST:** Java offers JAX-RS (Java API for RESTful Web Services) to create and consume RESTful services.
- **How to compile and run a web service application program:**
    - Compile: Use javac for Java source files.
    - Deploy: Use a web server or servlet container like Apache Tomcat.
    - Access: Use a browser or client tool to send requests to service endpoints.
- **Explain step-by-step web service implementation:**
    1. Define the service interface.
    2. Implement the service logic.
    3. Deploy on a server.
    4. Expose the endpoint.
    5. Create and run the client.
- **What is a traditional web-based system?** A traditional web-based system includes static/dynamic web pages served through HTTP. It typically uses web servers, browsers, and back-end databases.
- **What is a web server?** A web server is software (e.g., Apache, Nginx) that serves HTTP requests from clients by delivering HTML files or handling API calls.
- **How is fault tolerance handled in web-based systems?** Fault tolerance is achieved using redundancy, load balancing, failover servers, replication, and backup systems to ensure availability during failures.