MICHAEL ORINA
SCII/00825/2019

DISTRIBUTED SYSTEM CAT 1 2023

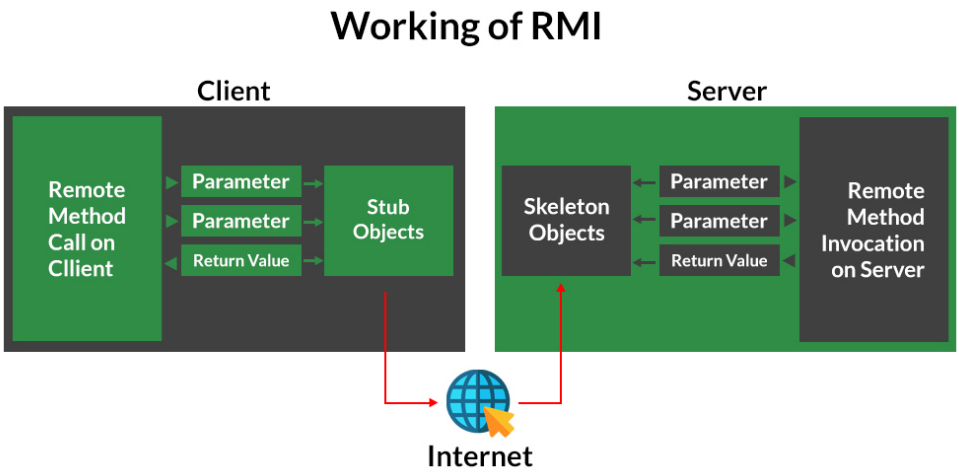
1

(a) Briefly discuss Clock synchronization.

In distributed systems, clock synchronization is the process of aligning the clocks of all nodes (computers, devices, etc.) to a common reference time. This is important for maintaining the consistency and accuracy of time-sensitive operations such as coordination, causality, and ordering. Without clock synchronization, it can be difficult to determine the order of events or distinguish between cause-and-effect relationships.
In distributed systems, clock synchronization algorithms like NTP (Network Time Protocol) or PTP (Precision Time Protocol) are used to achieve this goal. These algorithms exchange time information between nodes and use a combination of local clock measurements and network communication to converge to a common time. The accuracy of the synchronized clocks depends on the stability and precision of the reference time source, as well as the network conditions and the performance of the synchronization algorithm.

(b) With aid of a diagram, differentiate between:

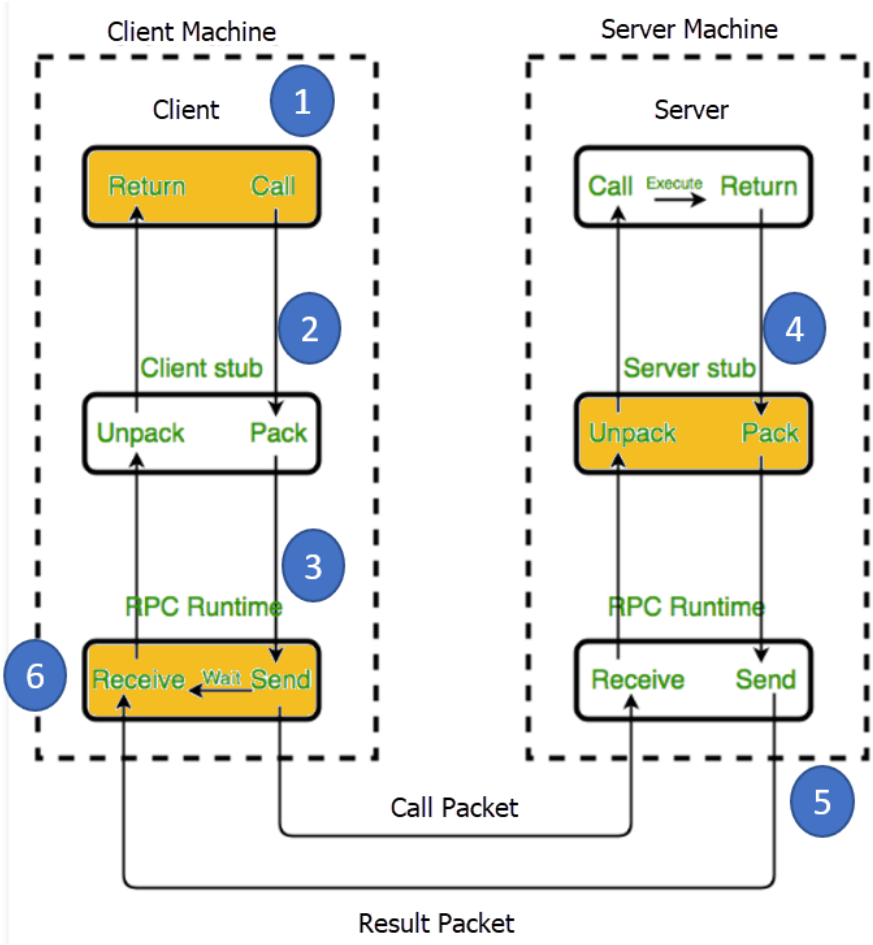
Remote Method Invocation (RMI)



Remote Method Invocation (RMI) is a Java-based technology used for building distributed applications. It allows a Java object running in one Java Virtual Machine (JVM) to invoke methods on an object in another JVM. The objects can be located on the same machine or on different machines connected by a network.
RMI works by defining interfaces that describe the remote methods, and then implementing these interfaces on the remote objects. The client makes a request to the remote object by

invoking one of its methods as if it were a local object. RMI takes care of the details of communicating the request to the remote object and returning the results.
RMI provides a way for Java objects to communicate with each other in a distributed system, and it is widely used for building client-server applications, where the client invokes methods on remote servers. It is also used for building peer-to-peer applications, where objects on different machines communicate directly with each other.

Remote Procedure Call (RPC).



Remote Procedure Call (RPC) is a protocol for making a request from one computer to another computer to execute a procedure or function. It is used for building distributed systems, where a client program running on one machine wants to request a service from a server program running on another machine.
In RPC, the client program calls a procedure that is executed on the server, just as if the procedure were local. The client program does not need to know that the procedure is executed on a remote machine. The RPC mechanism takes care of the details of transmitting the request to the server and returning the results to the client.
RPC is widely used in distributed systems and has been implemented in many programming languages, including C, C++, Java, and Python. It is a popular way of building client-server

applications, where the client invokes procedures on a remote server. The main advantage of RPC is that it provides a simple and convenient way to build distributed systems, allowing developers to focus on the application logic rather than the details of inter-process communication.

(c) Consider a simple server that carries out client requests without accessing other servers. Explain why it is generally not possible to set a limit on the time taken by such a server to respond to a client request.

It is generally not possible to set a limit on the time taken by a simple server to respond to a client request because of the unpredictable nature of computation and resource utilization. The server may need to perform a complex calculation or access a large dataset, which can take a varying amount of time depending on the input data and the current system load. In addition, the server may be subject to external factors such as network congestion or hardware failures that can impact its performance.
In general, it is difficult to predict the exact amount of time a server will take to respond to a client request, and it is therefore not feasible to set a strict limit on response time. However, some strategies can be used to mitigate the impact of long-running requests and improve the overall performance of the server. For example, a server could implement timeouts or prioritize requests based on their importance, or it could use techniques like load balancing or caching to distribute the workload and reduce response times.

(d) Explain the following Algorithms.

(i) Ring algorithm.
The Ring Algorithm is a distributed algorithm used in computer networks for resource allocation and node communication. It is used to arrange nodes in a logical ring topology, where each node is connected to exactly two other nodes, one on its left and one on its right. The nodes in the ring communicate with each other to coordinate the distribution of resources, such as processing tasks or network bandwidth.
The Ring Algorithm operates as follows: each node in the ring has a unique identifier and maintains a list of its neighbors. When a node wants to send a message or request a resource, it sends the request to its right neighbor. The right neighbor checks its own resources and, if it is unable to fulfill the request, it passes the request on to its right neighbor, and so on, until the request reaches a node that can fulfill it. The response then travels back along the ring in the opposite direction, from the node that fulfilled the request to the node that initiated the request.
The Ring Algorithm is efficient and simple to implement, making it a popular choice for distributed resource allocation and node communication. However, it has some limitations, such as the possibility of bottlenecks at nodes with high resource utilization and the lack of fault tolerance in case of node failures.

(ii) Bully election algorithm
The Bully Algorithm is a distributed algorithm used in computer networks to elect a coordinator or leader among a group of nodes. It is used in situations where there is a need for a single

node to act as the coordinator for a group of nodes, such as in a distributed database or a cluster of computing nodes.

The Bully Algorithm operates as follows: each node in the group has a unique identifier, and the node with the highest identifier is elected as the coordinator. If the coordinator fails or becomes unavailable, the other nodes in the group can detect the failure and initiate an election process. During the election process, each node sends an "are you there?" message to its neighbors. If a node does not receive a response from its neighbors, it assumes that they are unavailable and declares itself as the coordinator. The other nodes in the group receive the declaration and, if they are available, they send a challenge message to the declaring node. The node with the highest identifier among the challenger nodes is elected as the new coordinator.

The Bully Algorithm is simple to implement and provides a way for nodes in a distributed system to elect a coordinator in a decentralized manner. However, it can be subject to performance issues, such as high network traffic and long election times, in large groups of nodes. To address these issues, other algorithms, such as the ring algorithm and the Paxos algorithm, have been developed to improve the performance and reliability of distributed systems.

(e) Discuss different types of security threats in any distributed system and measures taken against them.

Distributed systems are vulnerable to a wide range of security threats, including unauthorized access, data theft, and system malfunctions. Some common types of security threats in distributed systems include:

1. Malicious attacks: This includes attacks such as denial of service (DoS), where an attacker deliberately overwhelms a network or system with traffic, or man-in-the-middle (MitM) attacks, where an attacker intercepts and manipulates communication between two nodes in a network.
2. Data theft: This includes unauthorized access to sensitive information stored on nodes in a network, such as passwords, financial information, or personal data.
3. System malfunctions: This includes failures in network components, such as nodes or routers, that can disrupt communication and lead to data loss or corruption.

To address these security threats, a variety of measures can be taken, including:

1. Authentication and authorization: Implementing secure authentication and authorization mechanisms, such as password protection or two-factor authentication, can prevent unauthorized access to nodes and systems in a network.
2. Encryption: Encrypting sensitive information and communication between nodes can prevent data theft and unauthorized access to sensitive information.
3. Network security: Implementing firewalls, intrusion detection systems, and other network security measures can prevent malicious attacks and system malfunctions.
4. Regular security audits and updates: Regularly auditing network security and updating software and systems can help prevent vulnerabilities and reduce the risk of security breaches.

5. Disaster recovery and backup: Having a disaster recovery and backup plan in place can help protect against data loss or corruption in the event of a system failure.