Sample Questions for Autumn Mid Semester Examination 2024-25

Distributed Operating System (CS30009)

Topic:Reliable and unreliable primitives and message passing(client server model implementation)

Short Question (1 mark)

Q1. How does a client typically initiate communication with a server?How does a client-server application handle multiple simultaneous client connections?

Answer:

By sending a request over a network to the server's IP address and port number.By using techniques such as multi-threading or asynchronous I/O to manage and respond to multiple client requests concurrently.

Q2. Discuss the differences between reliable and unreliable primitives in distributed systems. Provide examples of each and explain how these primitives influence the design of distributed algorithms.

Answer:

Reliable Primitives:
·Definition: Primitives that ensure guaranteed delivery and correct ordering of messages or operations.
Examples: TCP (Transmission Control Protocol) for communication, Two-Phase Commit (2PC) for distributed transactions.
Influence on Design: Simplify algorithm design by reducing the need for custom error handling and recovery mechanisms. Algorithms can assume that messages will arrive in order and be delivered correctly, leading to more straightforward implementation of consensus and coordination.

· Unreliable Primitives:
· Definition: Primitives that do not guarantee message delivery or order, which can lead to issues such as message loss or duplication.
Examples: UDP (User Datagram Protocol) for communication, direct hardware access without guarantees.

Influence on Design: Require additional mechanisms to handle errors and ensure reliability. Designers must implement custom protocols for acknowledgment, retransmission, and ordering, increasing the complexity of the system.

****

Long Question (2.5 marks)

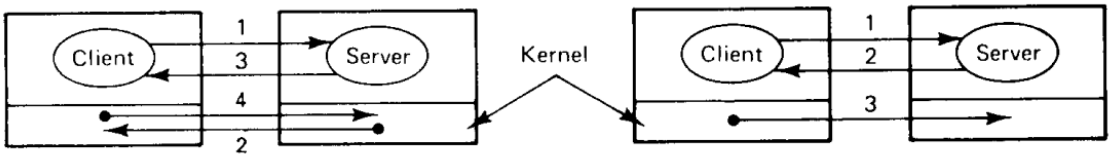
Q1. State the approaches used for reliable communication in distributed operating system

Answer:

Three approaches are used

The first one is to redefine the semantics of send to be unreliable.The system gives no guarantee about messages being delivered.

The second approach is to require the kernel on the receiving machine to send an acknowledgement back to the kernel on the sending machine. Only when this acknowledgement is received will the sending kernel free the user (client) process. The acknowledgement goes from kernel to kernel; neither the client nor the server ever sees an acknowledgement. Just as the request from client to server is acknowledged by the server's kernel, the reply from the server back to the client is acknowledged by the client's kernel. Thus a request and reply now take four messages, as shown in Fig. 2-13(a).



1. Request (client to server)
2. ACK (kernel to kernel)
3. Reply (server to client)
4. ACK (kernel to kernel)

1. Request (client to server)
2. Reply (server to client)
3. ACK (kernel to kernel)

Fig. 2-13. (a) Individually acknowledged messages. (b) Reply being used as the acknowledgement of the request. Note that the ACKs are handled entirely within the kernels.

The third approach is to take advantage of the fact that client-server communication is structured as a request from the client to the server followed by a reply from the server to the client. In this method, the client is blocked after sending a message. The server's kernel does not send back an acknowledgement. Instead, the reply itself acts as the acknowledgement. Thus the sender remains blocked until the reply comes in. If it takes too long, the sending kernel can resend the request to guard against the possibility of a lost message. This approach is shown in Fig. 2-13(b).

Q2. Describe how a server can manage multiple client requests simultaneously in a distributed system. What are the main techniques used, and what are their advantages and disadvantages?

Answer:

Multi-threading: Each client request is handled by a separate thread within the server process.

Advantages: Allows simultaneous processing of multiple requests. Provides isolation between requests.

Disadvantages: Can lead to increased memory usage and context-switching overhead.

Asynchronous I/O: The server uses non-blocking I/O operations and event-driven programming to handle multiple requests.

Advantages: Efficient resource use and scalability, as the server can handle many requests with a small number of threads.

Disadvantages: More complex to implement and manage compared to multi-threading.

Process-based Concurrency: Each client request is handled by a separate server process.

Advantages: Provides strong isolation between requests and fault tolerance.

Disadvantages: Higher memory and process management overhead.

****