MIVA OPEN UNIVERSITY

DISTRIBUTED COMPUTING

Weekly Summary

2

School of Computing

Communication in Distributed Systems

Communication is essential in distributed systems, allowing different components (nodes or processes) to exchange information and work together. The communication process in distributed systems involves message passing (IPC), where processes interact across different devices, possibly located far apart.

Basic Communication Models

Message Passing

Involves sending and receiving messages between processes, often used in scenarios where services like Paystack or Flutterwave coordinate transactions in real time.

Remote Procedure Call (RPC)

A process invokes a function on another process, even if it's on a different machine, creating the illusion of local execution. Google Maps uses RPC for location data retrieval from distributed servers.

Communication Mechanisms

Sockets

Like phone lines, sockets allow real-time communication by opening and closing connections for continuous data exchange.

Message Queue

Used for asynchronous communication, where messages are queued and processed later, ensuring system efficiency (e.g., Flutterwave).

Publish-Subscribe Systems

Used for broadcasting updates, where subscribers receive notifications automatically, like YouTube notifications for new videos.

Types of Communication

Synchronous Communication

Both sender and receiver must be ready at the same time, like a phone call. It's faster but fragile in high-latency networks.

Asynchronous Communication

The sender sends a message without waiting for a response, and the receiver processes it later, making it more resilient in modern systems.

Communication Challenges in Distributed Systems

Latency

Delays between sending and receiving messages, a common issue even in high-speed networks like AWS data centers.

Message Loss

Loss of messages during transmission can occur, requiring systems to retry messages or use timeouts.

Failure Detection

Determining whether a process has failed or is just slow is challenging, particularly in unreliable networks.

Real-life Applications

Healthcare Systems: A distributed hospital system uses message queues for alerts, RPC for updating pharmacy inventories, and publish-subscribe for sharing lab results.

Fintech: Platforms like Paystack and Flutterwave rely on message-passing to coordinate real-time transactions across microservices.

Ride-hailing: Platforms like Bolt and Telegram use sockets for real-time communication, such as location updates and messages.

Message Passing vs Shared Memory

Message Passing

Processes communicate by sending messages, ideal for independent processes with separate memory spaces. It has overhead due to network transmission and is typically synchronous or asynchronous.

Shared Memory

Allows processes to directly access and modify common memory spaces, offering faster communication but requiring synchronization mechanisms like semaphores or locks to avoid inconsistencies.

Advantages of Message Passing

- Best for independent processes with distinct memory spaces. Reliable communication can be ensured even with failures.
- Suitable for systems that need high reliability and network-based communication.

Advantages of Shared Memory

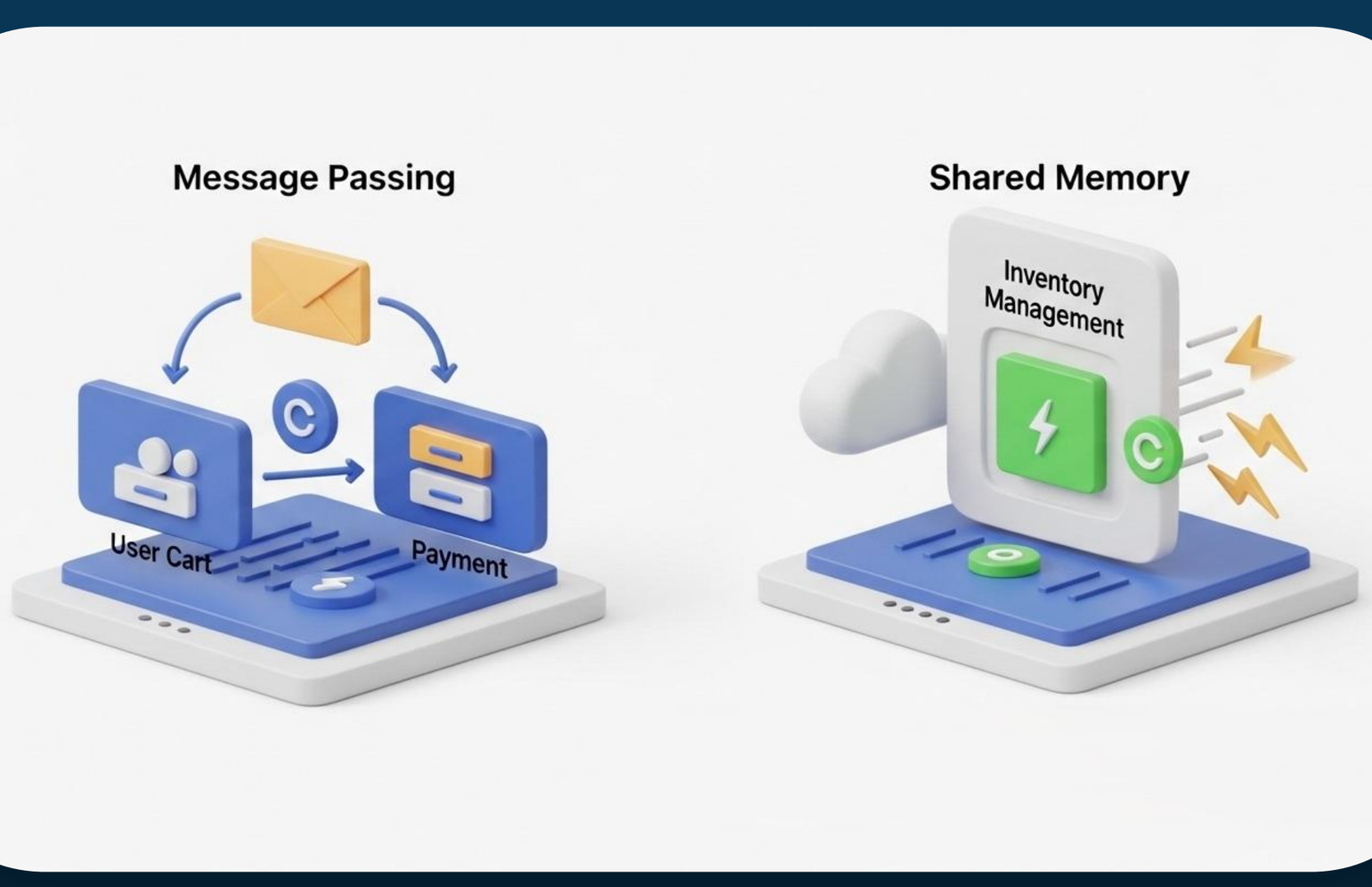
- Offers fast communication and less overhead compared to message passing. It’s ideal for closely connected processes within the same physical machine.
- Requires careful synchronization to prevent data inconsistencies and maintain consistency.

Choosing Between Message Passing and Shared Memory

1	2
Message Passing	**Shared Memory**
Suited for systems with independent processes or distributed nodes.	Best for tightly coupled systems with high communication demands and minimal latency.

Practical Scenario

For an online shopping platform, message passing is ideal for inter-service communication, while shared memory could be used for real-time inventory updates within a single service.



Reflection Questions

Reflection Questions

- How do message passing and shared memory differ in terms of performance and scalability in large distributed systems?
- When should message passing be preferred over shared memory in distributed systems, and vice versa?
- What are the potential trade-offs between synchronous and asynchronous communication in distributed environments?
- How can latency and message loss affect the overall performance of distributed systems, and what measures can mitigate these issues?
- In what scenarios would the use of shared memory introduce challenges related to data consistency and synchronization?

Call to Action

Review the core concepts of communication mechanisms and assess the communication model best suited for different system architectures.

Consider practical implementations of message passing and shared memory in various distributed systems, from fintech to healthcare.

Questions to Ponder On

- How does message passing ensure reliability in systems with distributed services?
- What role does latency play in choosing between message passing and shared memory for system communication?
- How can shared memory lead to synchronization issues, and what tools can help resolve these?

What would be the best communication mechanism for a real-time distributed application like a ride-hailing service?

- In what cases would message queues be a better choice than direct message passing for communication in distributed systems?

Skills and Competencies Acquired After This Lesson

- Analytical skills
- System performance evaluation
- Distributed system architecture design
- Fault tolerance mechanisms
- Communication protocol analysis
- Scalability assessment
- Synchronization techniques
- Asynchronous communication implementation