3.3 Inter-process communication (IPC) - Introduction, shared memory system and message passing system.

Inter-Process Communication (IPC)

Inter-Process Communication (IPC) is a mechanism that allows processes to communicate and synchronize their actions while running in an operating system. IPC is essential for coordinating activities between processes, sharing data, and achieving concurrency.

Different Perspectives on IPC

- **Definition**:
  IPC is a set of techniques that enable processes to exchange data and synchronize their actions.
- **Simple Usage**:
  IPC is like two friends talking to each other through either a shared notebook (shared memory) or by sending notes (message passing).
- **Scenario**:
  Imagine two programs running on a computer: one generates data (a producer), and the other consumes it (a consumer). They need a way to share this data without interfering with each other.
- **Example**:
  In a multi-user operating system, a web server and a database server may need to communicate to provide web pages that display data from the database.

Introduction to IPC Methods

IPC is broadly categorized into two types:

1. **Shared Memory System**
2. **Message Passing System**

Shared Memory System

- **Definition**:
  A shared memory system is an IPC mechanism where multiple processes can access the same segment of memory to communicate. The operating system allows these processes to map a memory region into their address space.

- **Simple Usage**:
  Imagine a large whiteboard in a common room that multiple people can write on or read from simultaneously.

- **Explanation**:
  In shared memory, processes exchange information by reading and writing to a common area in memory. This method is efficient because data does not need to be copied between processes. However, synchronization mechanisms (like semaphores or mutexes) are required to manage access and avoid conflicts.

- **Advantages and Disadvantages of Shared Memory System**

Criterion	Advantages	Disadvantages
Speed	Faster communication due to direct memory access.	Requires synchronization, which can be complex.
Data Sharing	Ideal for sharing large amounts of data.	Processes must be on the same machine.
Efficiency	Minimal overhead since no data copying is required.	Security risks as processes access the same memory.
Control	Processes have complete control over data format and access.	Debugging can be challenging due to shared access.

Message Passing System

- **Definition**:
  A message passing system is an IPC mechanism where processes communicate by sending and receiving messages, typically through the operating

system's kernel or using network protocols.

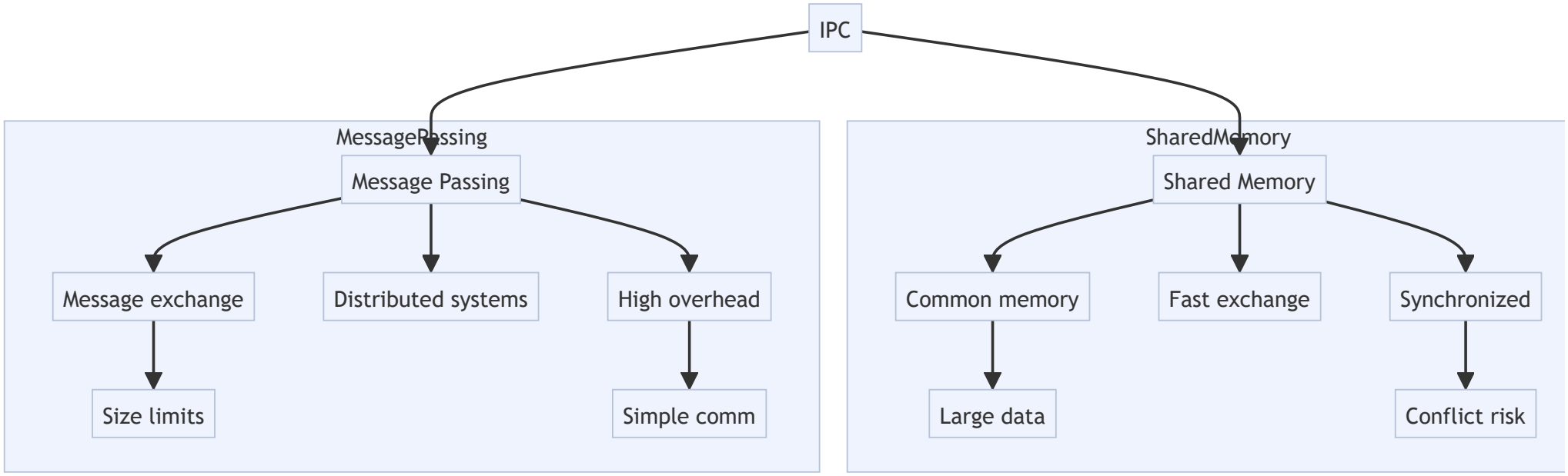
- **Simple Usage**:
  Think of message passing as sending letters via mail. One person writes a letter and sends it, and the other receives and reads it.

- **Explanation**:
  In a message-passing system, processes exchange information by sending messages to each other. These messages can be sent over a network or locally through the operating system's IPC facilities. This method is suitable for distributed systems where processes run on different machines.

- **Advantages and Disadvantages of Message Passing System**

Criterion	Advantages	Disadvantages
Simplicity	Easier to implement and use for simple communication.	Can be slower due to message copying and system calls.
Modularity	Facilitates modular design, ideal for distributed systems.	Limited by message size and system constraints.
Isolation	Processes are isolated, reducing the risk of data corruption.	Requires handling of message loss or duplication in networks.
Flexibility	Works across different machines or networks.	Higher overhead compared to shared memory for large data transfers.

Diagram: IPC Methods Overview



Summary Table: IPC Methods

Aspect	Shared Memory System	Message Passing System
Data Sharing	Direct access to shared memory.	Indirect access via message exchanges.
Speed	Faster due to direct memory access.	Slower due to message copying and handling.
Implementation Complexity	More complex due to synchronization.	Easier to implement for simple communication.
Communication Scope	Limited to processes on the same machine.	Supports communication across different machines.
Overhead	Minimal overhead once setup is complete.	Higher overhead due to kernel involvement.
Security	Less secure due to shared memory access.	More secure due to isolation of processes.
Use Case	Suitable for large data transfer within the same system.	Ideal for modular and distributed system designs.
Concurrency	Efficient but requires synchronization.	Easier to manage concurrency but with potential delays.

Conclusion

Inter-Process Communication (IPC) is crucial for processes to exchange data and synchronize their actions. The choice between a shared memory system and a message-passing system depends on the specific requirements, such as speed, data size, complexity, and the need for distributed communication. Understanding both methods helps in designing efficient and effective inter-process communication strategies in an operating system.