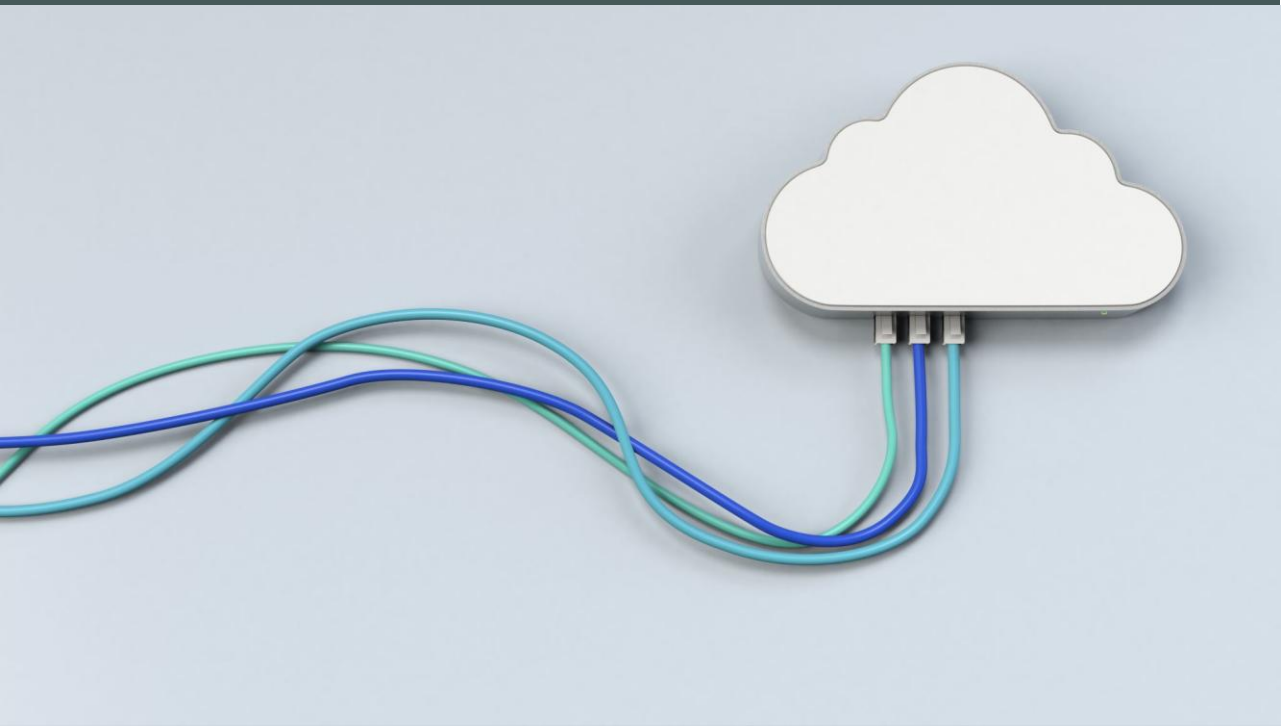
Lecture # 5

What is a Distributed System?



A **Distributed System** is a collection of **independent computers** that appears to its users as a **single coherent system**.

Examples:

- Google Search
- Distributed Databases (e.g., Cassandra, MongoDB)
- Online Banking Systems
- Cloud Computing Platforms

Why Use Distributed Systems?

- Scalability
- Fault tolerance
- Resource sharing
- Performance improvements

Communication in Distributed Systems

Two Models of Communication:

- Synchronous Communication
- Asynchronous Communication

Communication Types:

- Point-to-point (1-to-1)
- Broadcast or multicast (1-to-many)

Synchronous vs Asynchronous Communication

Feature	Synchronous	Asynchronous
Definition	Sender waits for reply	Sender continues without waiting
Latency	Lower control, predictable	Higher variability, less predictable
Error Handling	Easier	More complex
Use Cases	RPCs, banking systems	Email systems, message queues
Blocking?	Yes	No

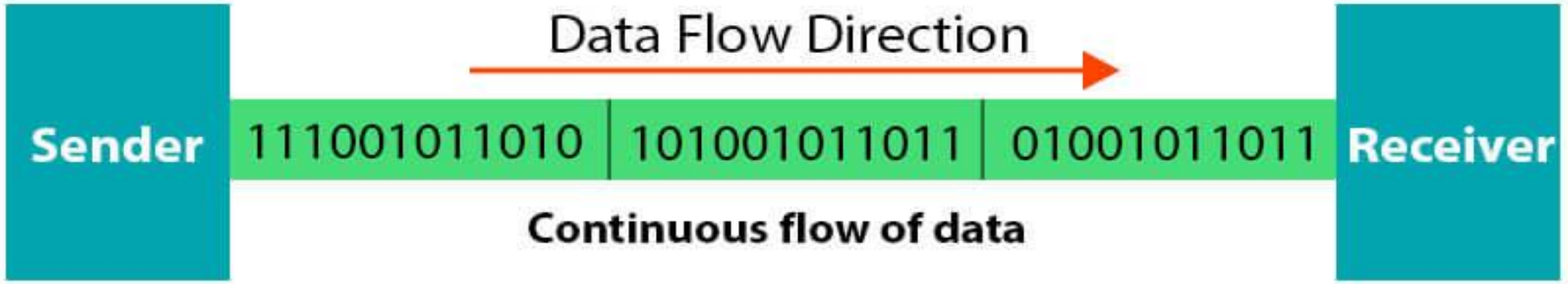
Synchronous vs. Asynchronous Communication

Communication is **at the heart** of distributed systems. Nodes (or processes) must exchange messages to coordinate tasks, share resources, and ensure consistency. The nature of this communication—**synchronous** or **asynchronous**—has a profound impact on **system design, performance, and complexity**. (Synchronous & Asynchronous)

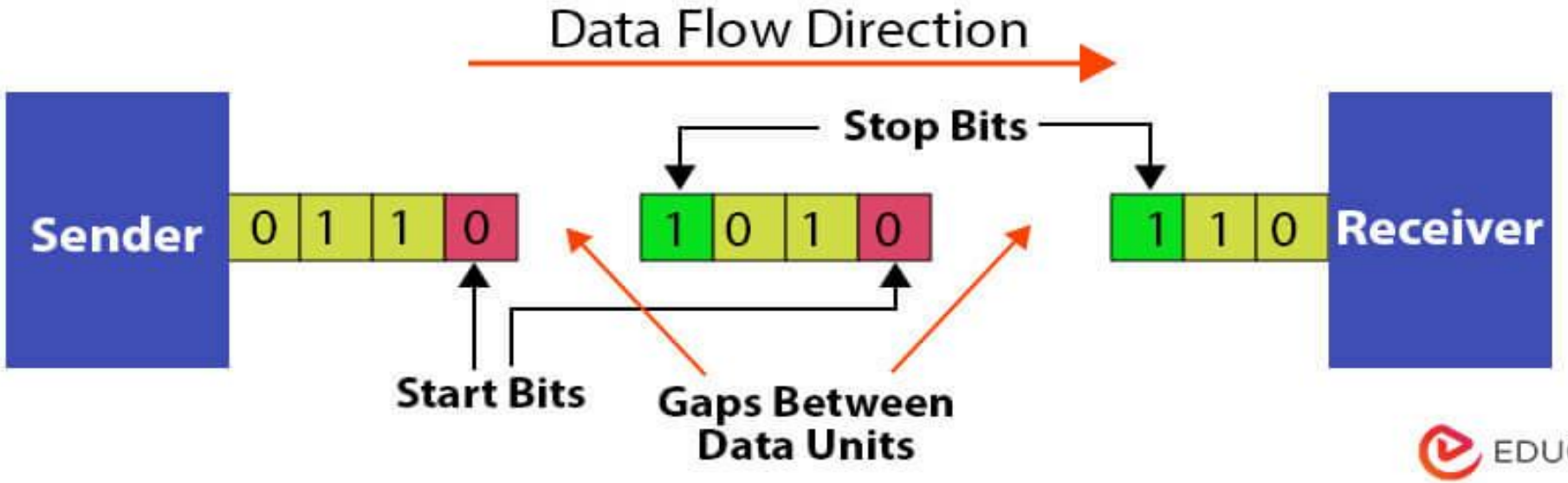
Scenario	Analogy
Synchronous	Phone call – both must be online and respond in real time.
Asynchronous	Email – send anytime; receiver reads/responds later.

Synchronous and Asynchronous Transmission

Synchronous Transmission



Asynchronous Transmission



EDUCBA

Comparison Table

Feature	Synchronous	Asynchronous
Blocking	Yes (sender waits)	No (sender continues)
Message Handling	Immediate	Delayed/Queued
Coupling	Tight temporal coupling	Loose temporal coupling
Scalability	Lower	Higher
Reliability Dependency	High on receiver availability	Less dependent on receiver state
Complexity	Lower (easier to reason)	Higher (requires queues, handling)
Typical Use Cases	RPCs, DB queries	Message Queues, Microservices, IoT

Choosing Between Them

Application Requirement	Recommended Model
Real-time interaction	Synchronous
High throughput and decoupling	Asynchronous
Strict consistency	Synchronous
Fault tolerance and resilience	Asynchronous