

# Synchronous vs Asynchronous Communications

## Master System Design

Progress

42/130 chapters



Ashish Pratap Singh



3 min read

Q Search topics...



### Databases & Storage

4/12

Imagine you're at a coffee shop and order a coffee. You wait in line, place your order, and wait for the barista to prepare your coffee. Once it's ready, they hand it to you, and you pay.

### Database Scaling Techniques

2/8

This is a **synchronous** interaction - you wait for the coffee before proceeding.

### Caching

5/6

Now, imagine ordering coffee online for delivery. You place your order, pay, and continue with your day. Later, the coffee is delivered to your doorstep.

### Asynchronous Communications

3/4

This is an **asynchronous** interaction - you didn't wait for the coffee to be prepared and delivered before moving on with your day.

### Tradeoffs

7/9

- Vertical vs Horizontal Scaling
- Concurrency vs Parallelism
- Long Polling vs WebSockets

## Get Premium

Subscribe to unlock full access to all premium content

Subscribe Now

Reading Progress

0%

On this page

1. Synchronous Communications

2. **Asynchronous Communications**

3. Factors to Consider

✓ Stateful vs Stateless Architecture

✓ Strong vs Eventual Consistency

✓ Push vs Pull Architecture

✓ Monolith vs Microservices

• Synchronous vs Asynchronous Communications

○ REST vs GraphQL

Distributed System Concepts

0/10 ▾

Architectural Patterns

0/5 ▾

Microservices

0/6 ▾

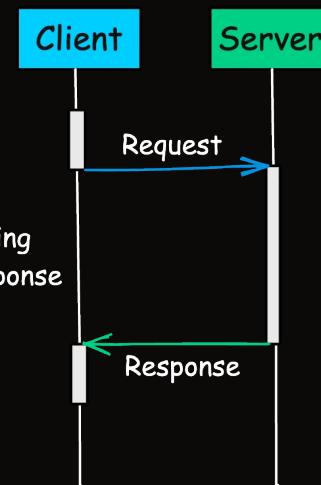
Big Data Processing

0/5 ▾

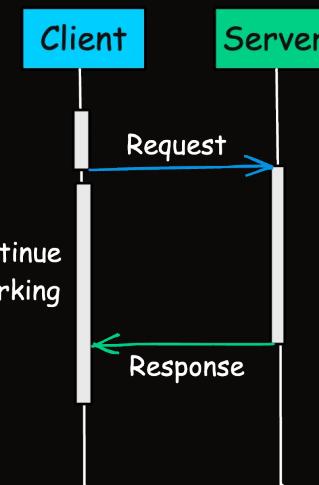
Observability

0/3 ▾

## Synchronous



## Asynchronous



In this article, we'll explore the differences between synchronous and asynchronous communications, their advantages, disadvantages, and use cases.

# I. Synchronous Communications

In synchronous communication, the sender waits for the receiver to acknowledge or respond to the message before proceeding.

It's like a phone call: you speak, the other person listens and responds, and the conversation progresses sequentially.

When a client (such as a user's web browser or a service in a microservices architecture) makes a synchronous request, it waits until the response is received.

The workflow remains **blocked** during this waiting period, unable to perform other tasks.

### **Key Characteristics of Synchronous Communication**

- **Blocking:** The client cannot continue until a response is received. This blocking behavior can create delays in user-facing applications, especially if the server response time is unpredictable.
- **Immediate Feedback:** The client receives feedback about success or failure right away. For example, when logging in, a user gets immediate confirmation if their credentials are correct or incorrect.
- **Tight Coupling:** Since the client and server must remain connected during the entire request-response cycle, they are often tightly coupled. This can simplify development but may reduce flexibility.

### **Advantages:**

- **Immediate Feedback:** Synchronous communications provide instant feedback, allowing for swift error detection and correction.
- **Simple Implementation:** Synchronous designs are often

straightforward to implement, as the request and response occur in a single, continuous transaction.

- **Consistency:** Data consistency is easier to manage because updates are processed in order.

### **Disadvantages:**

- **Blocking:** The sender is blocked until a response is received, potentially leading to resource waste and decreased system performance.
- **Tight Coupling:** Synchronous communications can create tight coupling between components, making it challenging to evolve or replace individual components without affecting the entire system.
- **Resource Intensive:** Each request must be fully processed before moving to the next, potentially leading to resource underutilization.

### **Use Cases:**

- **Low-latency applications:** Synchronous communications are suitable for applications requiring real-time responses, such as video streaming or online gaming.
- **Simple transactions:** Synchronous designs are ideal for straightforward transactions, like querying a database or fetching cached data.

## 2. Asynchronous Communications

Asynchronous communication is a communication pattern where the sender does not wait for the receiver to process the message and can continue with other tasks. The receiver processes the message when it becomes available.

### Advantages:

- **Non-Blocking:** The sender does not block and can continue executing other tasks after sending the message, reducing resource waste and improving system performance.
- **Loose Coupling:** The sender and receiver are loosely coupled, allowing them to operate independently.
- **Scalability:** Asynchronous communication enables better scalability as the sender and receiver can process messages at their own pace.
- **Resilience:** Failures in one part of the system do not necessarily cripple the entire operation.

### Disadvantages:

- **Complex Implementation:** Asynchronous designs can be more challenging to implement, as they require additional mechanisms for handling responses and errors.

- **Delayed Feedback:** Asynchronous communications may introduce delayed feedback, making error detection and correction more complex.
- **Data Consistency:** Ensuring data consistency across different parts of the system can be more complex.

### Use Cases:

- **High-throughput applications:** Asynchronous communications are suitable for applications requiring high throughput, such as message queues or task processing.
- **Decoupled systems:** Asynchronous designs are ideal for systems with multiple, independent components, like microservices architecture.
- **Long-running tasks:** Offloading non-urgent tasks to an asynchronous queue, like image processing or report generation, is ideal.
- **Event-driven architectures:** Asynchronous communication shines in systems where components react to real-time events, such as notifications.

## 3. Factors to Consider

When deciding between synchronous and asynchronous communication, consider the following factors:

1. **Performance:** Asynchronous communication can lead to better performance and throughput as the sender and receiver can work independently.
2. **Scalability:** Asynchronous communication allows for better scalability as the system can handle a higher load by processing messages concurrently.
3. **Reliability:** Asynchronous communication can provide better reliability through message persistence and retries in case of failures.
4. **Complexity:** Asynchronous communication introduces additional complexity in terms of message ordering, error handling, and coordination between components.
5. **Real-time requirements:** If the system requires real-time interactions or immediate responses, synchronous communication may be more suitable.

[< Prev: Monolith vs Microservices](#)[!\[\]\(d84e7ea36f695d92cb39ec32c307ac93\_img.jpg\) Take Notes](#)[!\[\]\(feabb98897b440bc8695a03336a6e2df\_img.jpg\) Star](#)[!\[\]\(9dfdaff1d86ba3c1f8353b4d1b61b8c5\_img.jpg\) Mark as Complete](#)[!\[\]\(83f22ed94ec5517769dd76d702c6bfd8\_img.jpg\) Ask AI](#)[Next: REST vs GraphQL >](#)