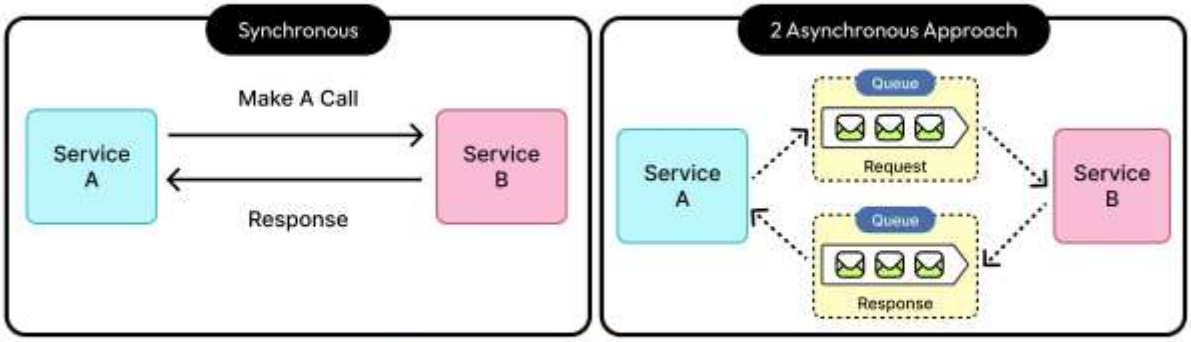
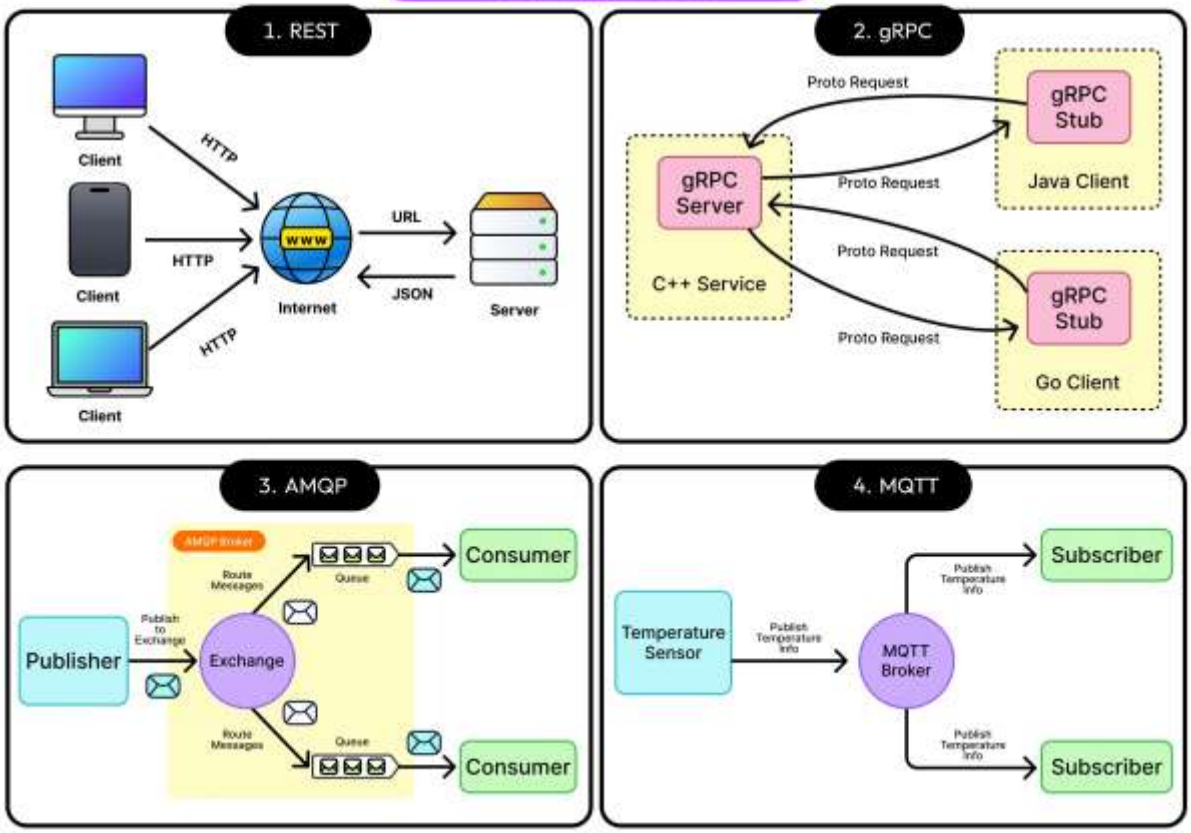
Synchronous vs Asynchronous Communication

ByteByteGo



The Key Protocols



1. Synchronous Communication

- **Diagram**: Service A calls Service B and waits for a response.
- **Real-World Example**:
  - **Scenario**: A user logs into a website.
  - **Details**: The frontend (Service A) sends the username and password to the backend server (Service B). The frontend waits for a response before proceeding (e.g., showing a welcome message or an error).
  - **Analogy**: It's like calling a friend and waiting on the phone until they answer your question.

---

2. Asynchronous Communication

- **Diagram**: Service A sends a request via a queue to Service B. The response may also be delivered through another queue.
- **Real-World Example**:
  - **Scenario**: Uploading a video on YouTube.
  - **Details**: Once you upload a video (Service A), YouTube puts it in a processing queue. You don't wait for it to finish; you get notified later (Service B sends the response asynchronously).
  - **Analogy**: It's like sending a letter by post and getting a reply later.

The Key Protocols

---

1. REST (Representational State Transfer)

- **Diagram**: Client → HTTP Request → Internet → Server (returns JSON).
- **Real-World Example**:
  - **Scenario**: Checking the weather on a weather app.

- **Details**: The app (client) sends a REST API request to a weather server with your city. The server responds with the weather in JSON format.
- **Synchronous by nature.**

---

2. gRPC (Google Remote Procedure Call)

- **Diagram**: Proto Requests from various clients (Java, Go) to gRPC Server (C++).
- **Real-World Example**:
  - **Scenario**: Internal services in a company talking to each other, like a recommendation service talking to a user service.
  - **Details**: Faster than REST, gRPC uses protocol buffers and is ideal for microservices in distributed systems.
  - **Mostly synchronous but supports async too.**

---

3. AMQP (Advanced Message Queuing Protocol)

- **Diagram**: Publisher → Exchange → Queues → Consumers.
- **Real-World Example**:
  - **Scenario**: E-commerce site order processing.
  - **Details**: When you place an order, it's published to a message broker **(like RabbitMQ).** Different services consume it: one processes payment, another updates inventory, etc.
  - **Fully asynchronous communication.**

Kafka in Real Life (Real-World Example)

Example: User Activity Tracking (e.g., LinkedIn or Netflix)

- **Scenario**: When a user watches a video or clicks a button, an event is generated.

- **Details**:
  - Microservices (or apps) **publish** these events to Kafka.
  - Other services (analytics, recommendation engine, fraud detection, etc.) **subscribe** to the relevant Kafka topics and process data asynchronously.
  - There is **no waiting** for a response; the event is simply logged and processed by downstream consumers.
  - **Analogy**: It's like dropping a message in a suggestion box — different departments read the box when they need to.

---

4. MQTT (Message Queuing Telemetry Transport)

- **Diagram**: Temperature Sensor → MQTT Broker → Subscribers.
- **Real-World Example**:
  - **Scenario**: Smart home temperature monitoring.
  - **Details**: IoT sensors publish temperature updates to an MQTT broker. Apps or devices (subscribers) receive updates in real-time.
  - **Asynchronous and lightweight, ideal for IoT.**

Summary Table:

Type	Protocol	Real-Life Example	Sync/Async
REST	REST	Checking weather via mobile app	Sync
gRPC	gRPC	Microservices talking in an enterprise app	Sync/Async
Messaging Queue	AMQP	E-commerce order processing system	Async
IoT Messaging	MQTT	Smart thermostats sending temperature data	Async

---