Synchronous Microservices : Fundamental Principle[1/2]

[Quick Bite] This is one of the key aspect in Microservice Architecture learning, so focus today on it.

It is revision for those who are already familiar with it otherwise learn as a concept. Disclaimer, it is not the exhaustive discussion, it is just to share you some quick indicator for the topic.

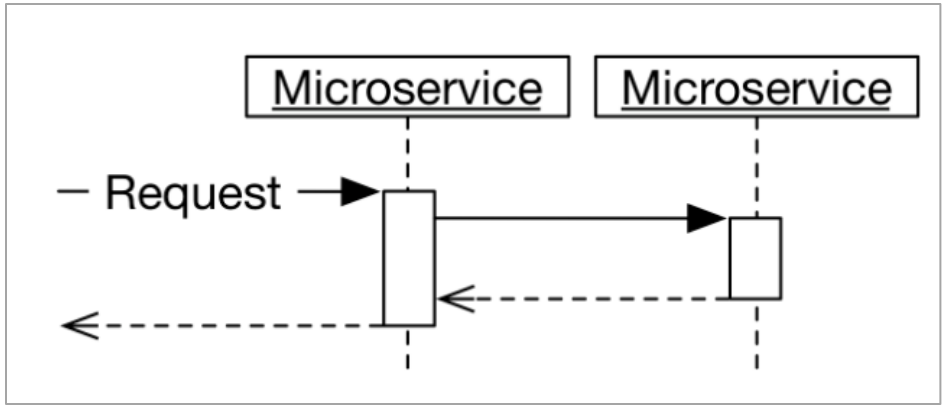
What is meant by synchronous communication between microservices?

What the architecture of a synchronous microservices system can look like?

Which advantages and disadvantages are associated with synchronous communication between microservices?

Def : A microservice is synchronous if it makes a request

to other microservices while processing requests and waits for the result.



Synchronous and asynchronous communication according to this definition are

independent of the communication protocol.

Synchronous protocols

A synchronous communication protocol means that a request returns a result.

For example, a REST or HTTP GET returns a result in an HTTP status, a JSON document, or an HTML page. If a system processes a REST request, makes a REST request itself to another system, and waits for the response, it is synchronous.

Asynchronous protocols

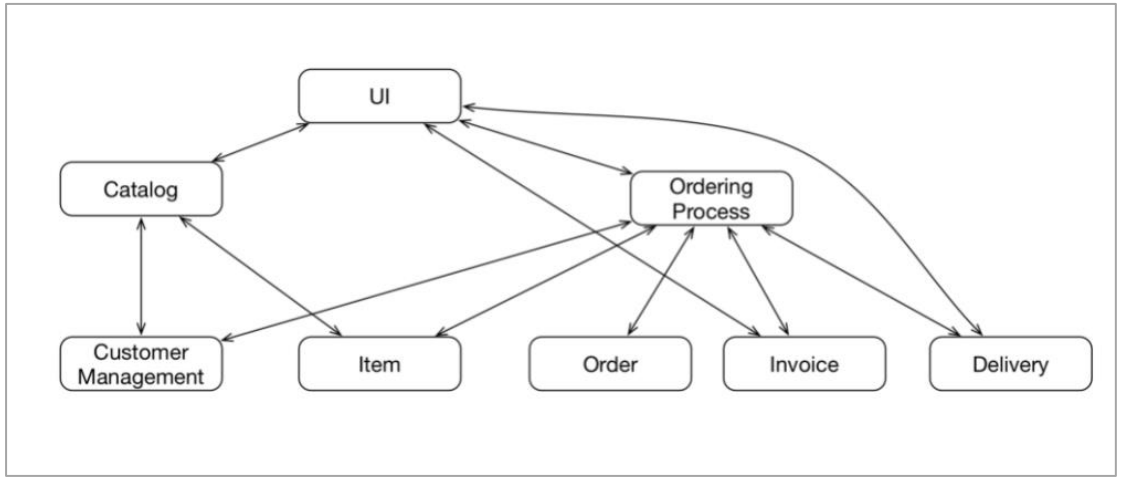
Asynchronous communication protocols, on the other hand, send messages to which the recipients react. There is no direct response.

Synchronous communication with an asynchronous protocol occurs when one system sends a message with an asynchronous communication protocol to another system and then waits to receive a response with an asynchronous communication protocol.

Synchronous Microservices : Fundamental Principle[2/2]

Let's focus the example shown in right side of the page here,

In this diagram, Nodes are microservice and name chosen to reflect the Purpose of the microservice briefly. Also, Edges here to show the Communication required to handle the dependency among the Microservices.



Here we have micro services like,

UI microservice, which is primarily interacting with the micro service like **Catalog**, **Ordering Process**, **Invoice** and **Delivery**

Now further , explore the Microservice like **Catalog** and **Ordering Process**.

Catalog microservice is primarily depend on **Customer Management** & **Item**

Another microservice, Ordering Process which depends on **Customer Management**, **Item**, **Order**, **Invoice** and **Delivery**

Synchronous Microservices : Advantage & Disadvantage

Advantage

Consistency

The architecture of this system ensures that all microservices always display the same information about a product or order because they use synchronous calls to access the respective microservices in which the data is stored.

There is only one place where each piece of the data is stored. Every microservice uses that data and therefore shows the latest data.

Intuitive for API Endpoint Service like system

Synchronous communication is a natural approach if the system is to offer an API. Any microservice can implement part of the API. The API can be the product, or the API can be used by mobile applications.

Easy to migrate from Monolithic Architecture

From Component structure like Monolithic architecture based Webservice, can be easier to migrate into such an architecture.

For example, the current architecture can already have a division into different synchronous communication endpoints or teams can exist for each of the functionalities.

Easy to understand for developer

Calling methods, procedures, or functions in a program is usually synchronous. Developers are familiar with this model so they can understand it more easily.

Disadvantage

One central database

This single central database, sometime raise the concern of SPF(Single Point of failure) for the system.

This will be handled using the replication and master selection protocol like Paxos or Raft etc…

Service discovery and Load balancing is concerned here in-case of synchronous microservice

Low Performance

Performance may also suffer because the microservices must wait for results from many other microservices.

Failure propagation

The vulnerability of the microservices and the additional waiting times can prevent the reliable operation of microservices systems with synchronous communication. These problems have to be solved when a microservices system uses synchronous communication.

Resiliency

Again, resiliency needs special attention and effort , without it gets hit and system could be considered as unreliable.

Higher level of dependency

synchronous communication can create a higher level of dependency in the domain logic.

Synchronous Microservices : Challenges and Technical Solution

1. Service discovery
The microservices must know how they can communicate with other microservices. Usually, this requires an IP address and a port.
Service discovery serves to find the port and IP address of a service. So, incase of synchronous microservice communication, you must discover the other side microservice before communication. It's meaning that , you should implement something like DNS(Domain Name System) for your service cluster. This is primarily not a concern if you implement Event based asynchronous microservice system.

2. Resilience
When communication is synchronous, microservices must be prepared for the failure of other microservices. It must be prevented that the calling microservice fails as well. Otherwise, there will be failure cascade: First one microservice fails. Then other microservices call this microservice and fail. In the end, the entire system will be down.

3. Load balancing
Each microservice should be scalable independently of the other microservices. Load must be distributed between microservices.

This does not only pertain to access from the outside, but also to internal communication. Therefore, there must be a load balancing for each microservice.

4. Routing
Finally, every access from the outside should be forwarded to the responsible microservices. This requires routing.

- ✓ A user might want to use the catalog and the order process.
- ✓ While they are separate microservices, they should appear as parts of the same system to the outside.

Consequently, the technologies for synchronous microservices which are discussed in the following chapters have to offer solutions for **service discovery**, **resilience**, **load balancing**, and **routing**.

5. API gateways

For complex APIs, complex routing of requests to the microservices might be needed. API Gateways offer additional features:
- Most of the API gateways can perform user authentication.
- They can throttle the network traffic for individual users to support a high number of users at the same time. This can be supplemented by centralized logging of all requests or caching.
- API gateways can also solve aspects like monitoring, documentation, or mocking.

The implementations of Apigee, 3scale by Red Hat, apiman, or cloud products like the Amazon API Gateway and the Microsoft Azure API Gateway are examples of API gateways.

The examples in this course do not offer public REST interfaces, only REST interfaces for internal use. The public interfaces are just web pages and the examples do not use API gateways.

Synchronous Microservices : Testing of Synchronous microservice

Test Framework, designed for Synchronous Microservice?

One option as a test framework for synchronous microservice is Pact (https://pact.io/) .

One sentence description for Pact is Fast, easy and reliable testing for integrating web apps,

APIs and microservices.

Documentation for Pact : https://docs.pact.io/

Consumer testing