IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 5, Issue 2, May 2021
Impact Factor: 4.819

Interservice Communication in Microservices

Christy Sibi Pachikkal
Department of Information Technology
Sir Sitaram and Lady Shantabai Patkar College of Arts and Science, Mumbai, India

Abstract: *Microservices based applications are distributed and for each service can run on a different machine. Due to its distributed pattern, one of the key challenges when designing applications is the mechanism by which services communicate with each other. Communication among microservices must be efficient and robust. With many small services interacting to complete a single business activity, this can be a task . If we are talking about communication styles, it is possible to classify them in two axes. In this article, we look at the compromises between asynchronous messaging versus synchronous communication. Since it is common for each service instance to work as a separate process, services need to work using a process communication protocol.*

Keywords: Approaches, One to One, One to Many Communication Styles, Interservice Communication Patterns

I. INTRODUCTION

In the Microservice architecture pattern, a distributed system is running on a number of different machines, and each service is a component or process of an enterprise application. That means these services working at the multiple machines must handle requests from the clients of this enterprise application. Sometimes all these services work together to handle those requests. So, all services interact using an inter-service communication method. But in case of the Monolithic application, all components are the part o f the same application and run on the same system. So, Monolithic application does not require a communication which has inter-service mechanism. Microservices architecture implies of having a collection of loosely coupled services. These services can be deployed individually from each other and provide great flexibility and productivity. The goal of the microservices is to sufficiently decompose and decouple the application into loosely coupled services structured around business capabilities. A monolithic application has all its components combined as a single object and deployed to a single machine. One component call another using language-level method calls. However, in the microservice architecture, all components of the applications run on several machines as a process or service, and they use inter-service communication to interact with each other. Microservices frameworks usually execute a consumer grouping mechanism whereby different instances of a single application have been placed in a competing consumer relationship in which only one instance is expected to handle an incoming message . In Microservice Architecture, we can classify our inter-service communication into two approaches like the following:

1. Synchronous communication style
2. Asynchronous communication style

II. APPROACHES FOR DIFFERENT MICROSERVICE INTERSERVICE COMMUNICATION PATTERNS

Decisions related to such a division require knowledge about the business aspects of a system, but communication standards can be easily defined, and they are unchangeable no matter which approach to architecture we decide to implement. If we are talking about communication styles, it is possible to classify them in two different forms. The first step is to define whether a protocol is synchronous or asynchronous communication.

2.1. Synchronous Inter-service Communication Style

Synchronous communication is often regarded as request/response interaction style and pattern. One microservice makes a request to another service and waits for the services to process the result and send a response back. In this style, it is common that the requester blocks its operation while waiting for a response from the remote server. In this

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 5, Issue 2, May 2021
Impact Factor: 4.819

type of communication style, the client service expects a response within time and wait for a response by blocking a while. This style can be used by simply using HTTP protocol usually in the form of REST. It is the simplest feasible solution for microservices inter-service communication to interact with services. The client can make a REST call to interact with other services. The client sends a request to the server and waits for a response from the service (JSON over HTTP). Representational state transfer (REST) application programming interfaces: API and gRPC are the most common framework for implementing synchronous form of communication style in microservices.

- **REST API:** REST is an architectural style that is most commonly used for designing APIs for modern web services. In a system which uses REST API for its IPC i.e. (inter process communication), each service normally has its own web server up and running on a specific port such as 8080 or 443, and each service displays a set of endpoints to enable the interactions with other microservices and exchange of communication between them. The server that interacts directly with client through its interface can also be identified as WebAPI
- **gRPC:** gRPC is an open -source high performance RPC framework designed and developed by Google. Remote procedure call (RPC) is a mechanism used in several distributed applications to expedite inter service communication. RPC was first implemented by Birrell and Nelson and it has been regarded as a protocol that enables a communication exchange between two process with characteristics of low overhead, simplicity, and transparency. By default, if when a client sends a request to a server it halt the process and waits for the results to be returned. RPC or Remote Procedure call is therefore considered as synchronous form of communication.

Example: if we look into any e-commerce application. When a customer searches for a specific product to purchase, then that product's availability needs to be confirmed in the inventory by making a request to product convenience service. Because customer should know about what the current availability of the product to place the order. In this case, we can use synchronous form of communication to know about the product's real - time availability in inventory and price information.

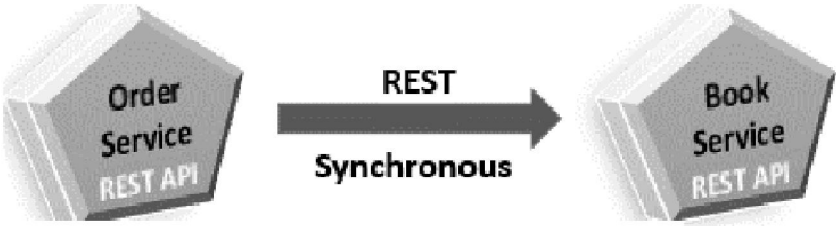

Figure 1: Synchronous communication style

2.2 Synchronous One-to-One Interservice Communication Style

Most synchronous communications are one-to -one style. In synchronous one-to -one communication, you can also use numerous instances of a service to scale the service. However, if you try to do, you will have to use a load-balancing mechanism or methods on the client side. Each service includes meta-information about all instances of the calling service. For one-to-one synchronous services, the same can be achieved with a load-balancing method performed on the client side. Each service has information about the location addresses of all instances that are calling services. This information can be taken from a se rvice discovery server or may be provided manually in configuration properties. Each service is having a built-in routing client that can choose one instance of a target service, using the right algorithm, and send a request there. As you can see in the following diagram, we have multiple instances of a specific service, but the services are still communicating one-to -one. That means that each service communicates to an instance of an alternative service. The load balancer selects which method should be called. The following is a list of some of the most commonly used load -balancing methods:

- **Round-Robin:** This is the simplest method among all that routes requests across all the instances in sequence. Client requests are allocated to application servers in rotation. For example, if you are having three application servers: the first client request to the first application server in the list, the second client request to the second

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 5, Issue 2, May 2021
Impact Factor: 4.819

application server in the list , the third client request to the third application server in the list , the fourth to the first application server and so on by continuing the process.

- **Least Connections:** This is a method in which the requests goes to the instance that has the fewest number of connections at the time. In this, the request goes to the instance that is processing with the least number of active connections at the existing time. In cases where application servers have similar specifications, an application server may be overloaded due to extended long lived connections; at that time this algorithm takes only the active connection as load into consideration.
- **Weighted Round-Robin:** This is an algorithm that allocates a weight to each particular instance and forwards the connection according to this weight. Weighted Round Robin are builds on the simple Round-robin load balancing algorithm to account for differing application server characteristics. The administrator assigns a weight for each application server based on criteria of their choosing to demonstrate the application servers' traffic handling.

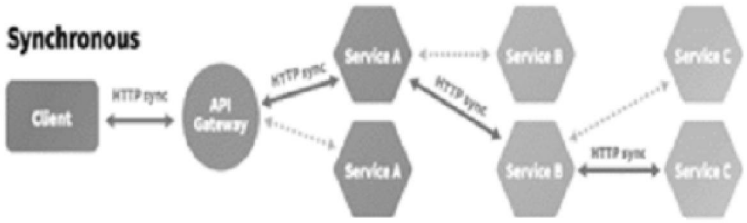

Figure 2: Synchronous one-to-one communication style

- **IP Hash:** This is a t ype of method that generates a unique hash key from the source IP address and defines which instance takes the request. It combines source and destination IP addresses of the client and server to produce a unique hash key. The key is used to allocate the client to a specific server. As this key can be regenerated if the session is broken, the client request is directed to the same server it was using previously. This is useful if it is important when a client should connect to a session that is still active after a disconnection.

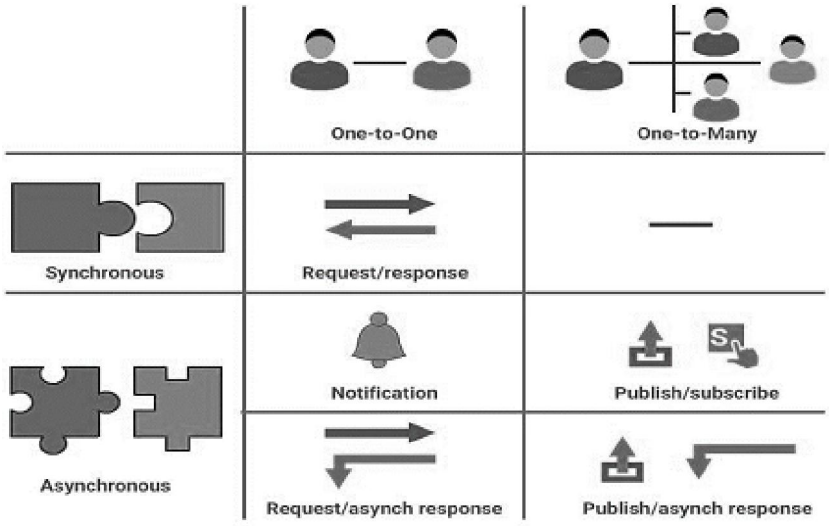

Figure 3: Interservice Communication Interaction Types

2.3 Asynchronous Inter-Service Communication Style

The asynchronous form of communication can be implemented in microservices when services exchange messages with each other through a message broker. In this form of communication, the message broker acts as an intermediary

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 5, Issue 2, May 2021
Impact Factor: 4.819

between services to coordinate the request and responses. One of the fundamental differences in asynchronous communication as compared to the synchronous mode is that in asynchronous communication the client no longer makes a direct call to the server and expect an immediate answer. Instead, other services subscribe to the same broker to pick -up the available requests and process them further before placing them back to the message queue. In this communication style, the client service doesn't wait for the response coming from another service. So, here the client does not block a thread while it is waiting for a response from the server. Such type of communications is possible by using lightweight messaging brokers. The message producer service does not wait for a response. It just generates message and sends message to the broker; it waits for the only acknowledgement from the message broker to know the message has been received from message broker or not. Failure isolation is well in asynchronous communication. Because if a message broker fails to send the message, once the message broker gets retrieved it will resend the message again. We can see in the following fig:

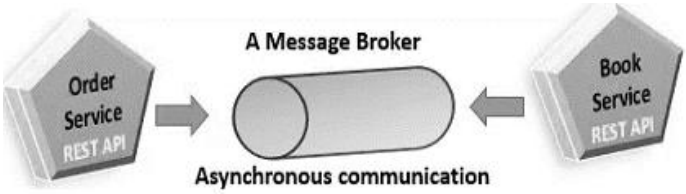

Figure 4: Asynchronous Communication Style

As you can see in the following diagram, the Order Service produces a message to a Message Broker and then forgets about it. The Book Service which subscribes to a topic is fed with all the messages belonging to that topic. The services do not need to know each other at all, they just need to know that messages of a specific type exist with a certain payload. There are various tools to support lightweight messaging, you can choose one of the following message brokers that is delivering your messages to consumers running on individual microservices. The most popular protocol for this type of communication is AMQP (Advanced Message Queuing Protocol), which is supported by many operating systems and cloud providers. An asynchronous messaging system can be implemented as a one-to -one (queue) or one-to -many (topic) mode. The most popular message brokers based on the AMQP protocol are RabbitMQ and ApacheKafka.

- **RabbitMQ:** It is considered as a stable, open-source message broker application. RabbitMQ promotes the usage of a protocol named as AMQP (Advanced messaging queuing protocol) as the wire level protocol or network protocol for exchanging messages. It is a binary protocol that deals with the low-level details of encoding and marshaling of message contents.
- **Apache Kafka:** Kafka is the most popular open source distributed publish-subscribe streaming platform that can handle millions of messages per minute. Apache Kafka is a distributed streaming platform that was initially conceived as a message queue. Kafka uses a binary TCP-based protocol that is optimized for the efficiency and depend on a "message set" abstraction that eventually groups messages together to reduce the overhead of the network round trip.

For example, in a banking domain, loan request should be processed and needs approval at multiple levels. So, in this case, when a user raises a request for the loan, then the loan request service will provide some reference number instantly . Once all the approvals have been done, it will store the loan request details in the database. So, in this situation, we can use asynchronous communication. The two properties of the asynchronous communication can be described as follows:

A. Asynchronous One-to -One Communication Style

In this communication approach, each request of a service client is processed by one instance of a service. There are the following kinds of one-to -one interactions. Moreover, the message queues up if the receiving service is down & proceeds later when they are up. This is particularly important from the perspective of loose coupling, multi -service communication, and coping up with partial server failure.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 5, Issue 2, May 2021
Impact Factor: 4.819

- **Notification:** It is considered as a one-way request. A client transmits a request to a service; however, a reply is not expected/ sent.
- **Async/Request Response:** In this, the client transmits a request to service, which responds asynchronously. The client does not block while waiting and is designed with th e belief that the response might not arrive for a while.

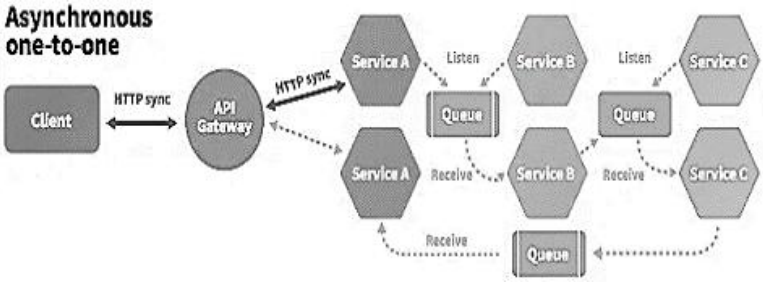

Figure 5: Asynchronous one-to-one communication style

As you can see in the following diagram, in one-to -one style, each service has only one instance and the services are communicating through the message broker queue.

B. Asynchronous One-to -Many Communication Style

In this communication approach, each request of a service client is administered by multiple service instances. The client does not block while waiting for a response, and the response is not certainly sent immediately. The following kinds of one-to -many interactions are as follows :

- **Publish/subscribe:** In this approach, a client publishes a notification message that is used by zero or multiple interested services.
- **Publish/async responses:** In this approach, a client then publishes a request message and waits a certain amount of time for responses to come from related services.

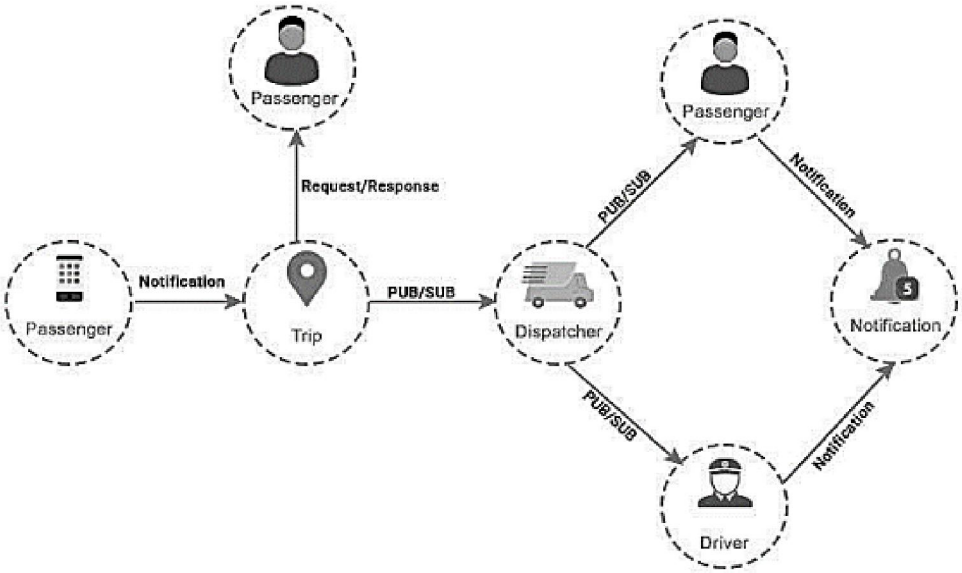

Figure 6: Asynchronous one-to-many communication style

In the above following diagram, we can see that there are multiple instances of each service. Here, a client service publishes a notification message as a topic and this topic is consumed by one or more instances of the interested services.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 5, Issue 2, May 2021
Impact Factor: 4.819

III. Conclusion

Inter-service communication in microservices is essential for building a microservices-based application. Services are independent and the only way which they can interact and formulate business functionality is through interservice communication. In Microservices architecture style, communication between services will play an important role when it comes to performance. In this paper, we compared synchronous and asynchronous inter-service methods with regards to performance. So based on your requirement, you have to choose the right approach for inter-service communication. Therefore, both synchronous and asynchronous type of communication patterns has to be adopted according to the functional and non -functional requirements of the specific components. I hope this article was informative and would be helpful and leaves you with a better understanding of interservice communication in Microservices.

References

[1] https://www.dineshonjava.com/microservices-interservice-communication
[2] https://walkingtreetech.medium.com/inter-servicecommunication-in -microservices-c54f41678998
[3] https://www.researchgate.net/publication/346943854_Evaluating_the_Impact_of_Inter_Process_Communication_in_Microservice_Architectures
[4] https://www.chakray.com/microservices-communication methods-types-and-styles/
[5] https://dzone.com/articles/communicating-betweenmicroservices
[6] https://docs.microsoft.com/enus/azure/architecture/microservices/design/
[7] https://softteco.com/blog/inter-service -communication -in microservices
[8] https://blog.devgenius.io/microservice-architecture communication-design-patterns-70b37beec294