MESSAGE BROKERS

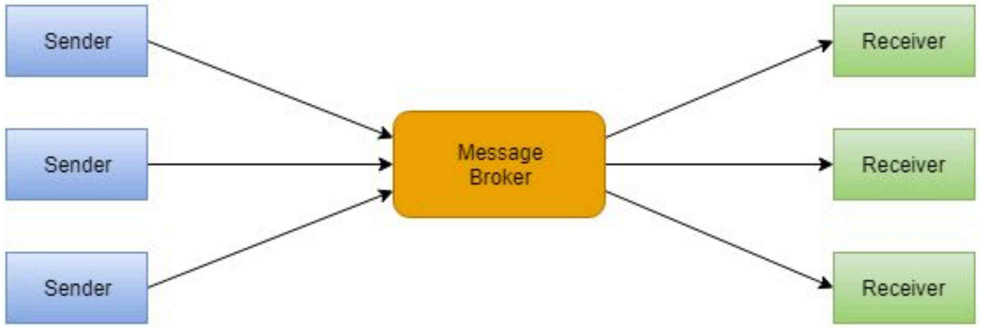
What Is A Message Broker?

Generally speaking, a broker is an individual or firm who arranges transactions between a buyer and a seller. An example could be a real estate agent that mediates the communication between the buyer and seller. In the healthcare space, there are companies that function as intermediaries that forward claims information from healthcare providers to insurance payers, also known as clearinghouses.
Likewise, if we want to exchange messages between two distributed software components, we need a mediator. This mediator is known as the message broker. It receives incoming messages from a sender and sends them to a receiver. This way the sender and receiver can be totally decoupled.

A message broker acts as an intermediary platform in communication between two applications.
It provides a way for exchanging messages from sending to receiving points. Messaging enables distributed communication that is loosely coupled. A message is sent to a destination by a sender, and can be retrieved from the destination by the receiver. The sender and the receiver don’t have to be available at the same time in order to communicate. A good example of messaging from the real world, that enables communication between people, is email.

Microservices, Internet of Things and Message Brokers

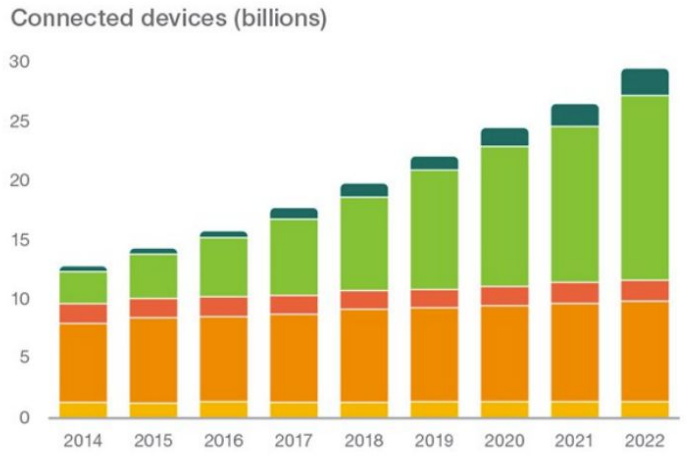
There are many use cases where the message broker is a good fit for a system, including currently popular topics in IT - Internet of Things and Microservices. Therefore, the usage of a message broker can also be beneficial in the healthcare industry. Microservice architecture is used more often in healthcare ecosystems, as healthcare applications are becoming more and more complex. Also, healthcare is one of the application domains where IoT is of enormous interest. When connected to the internet, ordinary medical devices can collect invaluable additional data, give extra insight into symptoms and trends, enable remote care, and generally give patients more control over their lives and treatment.

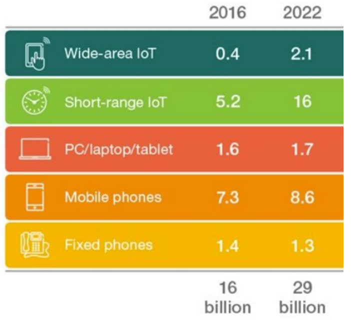


Message Broker In IoT

Today we have around 22 billion smart devices connected to the internet. Considering the fact that about 7.7 billion people live on our planet, it means that we have almost 3 smart devices per person. This number is getting bigger every day and will increase significantly in the years to come.

The growing amount of equipment connected to the internet has led to a new term, Internet of Things (IoT). Simply put, IoT is a set of devices that are able to interact with each other. With development of new IoT devices, like smart houses and other automatic systems, our everyday life becomes more and more digitized.This way of technological development has brought many advantages, but also some challenges, like the problem of successful data exchange.





	2016	2022
Wide-area IoT	0.4	2.1
Short-range IoT	5.2	16
PC/laptop/tablet	1.6	1.7
Mobile phones	7.3	8.6
Fixed phones	1.4	1.3
	16 billion	29 billion

Benefits of Message Brokers in IoT

IoT devices generate a lot of traffic. They send different types of information, like status reports and environmental measurements. They also receive data, including instructions and data from other devices.

Most of the IoT Implementations today use REST over HTTP based connectivity from the client to the server. REST has certain limitations that show up when a solution scales up to the larger number of devices and more transactions per second.

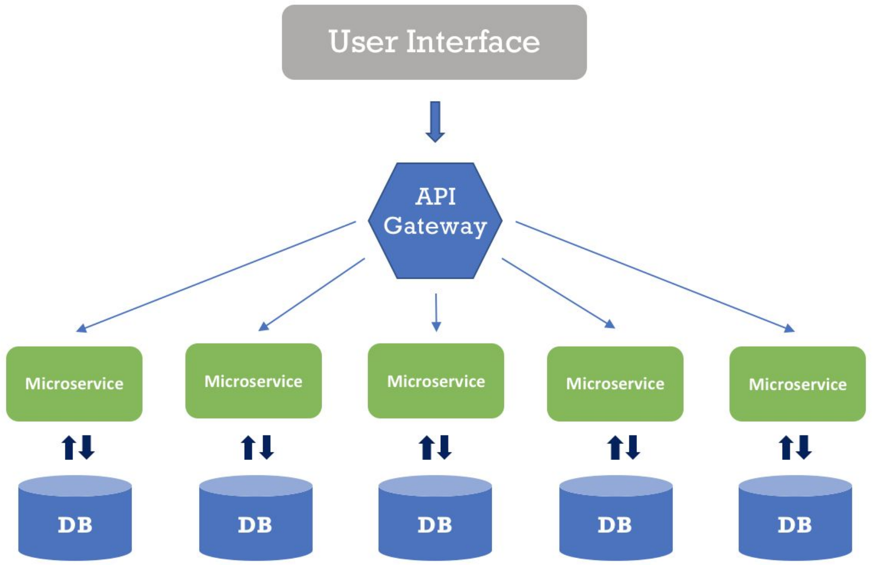
REST is a one-way connection. The client connects to the server in order to send or retrieve data. The server is not able to contact the client directly, it needs to wait for the client to connect. This causes a delay in performing an action.

Let’s take an example of activating an air conditioner from a mobile app. The message from the mobile will hit the server instantly. But the message from the server to the client needs to wait for the client to connect to the server. Imagine that instead of having your phone ring when someone is trying to call you, you had to pick up the phone every few seconds just to check if somebody is on the line waiting to talk to you - this is the same use case.

...25 times faster than REST Calls.

Message broker allows the client to always stay connected providing two way communication between the client and the server. This allows the server to push the message to the client device making the device respond to the command instantly as expected.
According to analysis and test reports, Message broker can transfer data at a rate of 20 to 25 times faster than REST Calls.

Although there are many ways to handle IoT messaging, message brokers are proving to be one of the best options.
The Internet of Things and cloud computing can be advantageous for healthcare applications. Typical IoT topology in healthcare has 3 parts. A set of connected sensors or handheld devices that are recording the patient’s vital signals, electrocardiogram (ECG), electromyography (EMG), body temperature, blood glucose (BG) and sending this information to broker. Message broker redirects this data to cloud instances that analyze and store processed data to a cloud storage. Finally the user can access and monitor patient data from any location.



Microservices and Message Broker

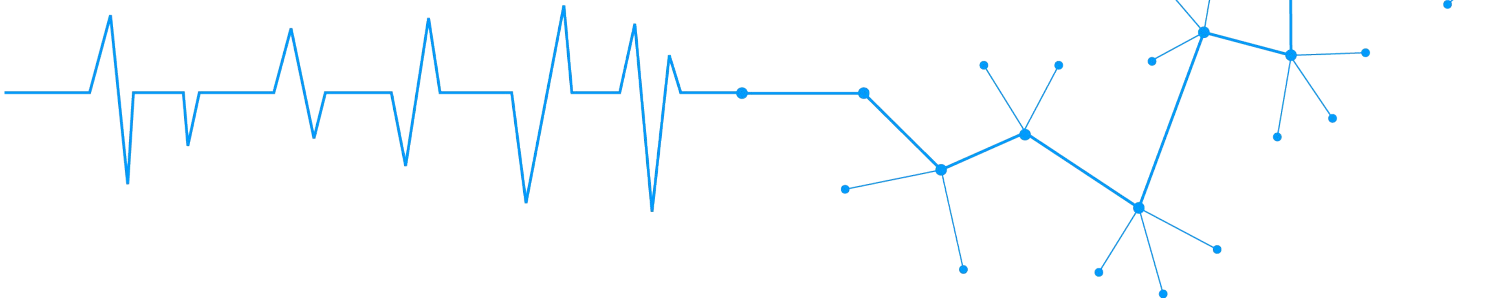
Microservices are an important innovation in application development and deployment. The microservice application architecture represents a new approach to software development where developers build the application as a set of modular components or services, each with a specific task or business targets. Each of these modular components is known as a microservice.

Microservices have gained popularity in recent years as an alternative to the more traditional monolithic software architectures. Software developers have adopted the microservices architecture as a means of improving the process of developing, testing and delivering software.

…fully asynchronous communications.

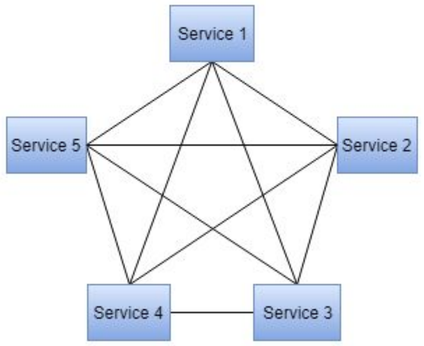
When using microservices, one of the biggest challenges is to decide how services should communicate with each other.
Most developers choose to design REST API that each service exposes and then have the other services invoke that API with a regular HTTP client.
This has some advantages, but also many drawbacks.
For example, imagine that called service has crashed and can’t respond. Your client service has to implement some kind of reconnection or failover logic, otherwise, you risk to lose requests and pieces of information.
Choosing message broker, on the other hand, is a good solution because it adds a layer of abstraction between loosely coupled services and allows for fully asynchronous communications.

Benefits of Message Broker in Microservices

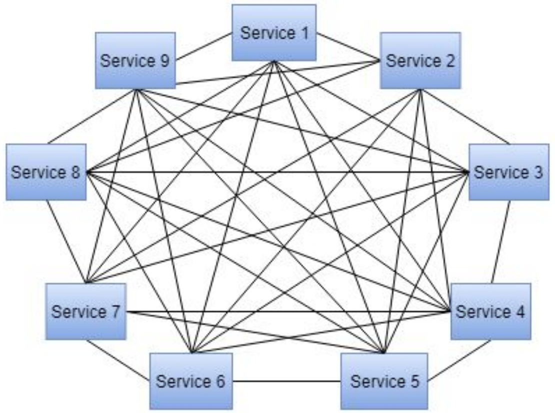


Scalable microservice connectivity.

Without a message broker and queues or topics, producer services talk directly to consumer applications through TCP or HTTP/REST. This works fine when few services are talking to each other, but it becomes a problem as we add more services to the system. Each new application adds N number of new potential connections that need to be incorporated within the system logic. Moreover, to cover service failure, you'll have to build in a way for the system to know how to find a new instance if an application in your system or your server fails. In short, TCP and HTTP/REST doesn't scale in terms of application connectivity, and the management of the communication between the connections doesn't scale either.

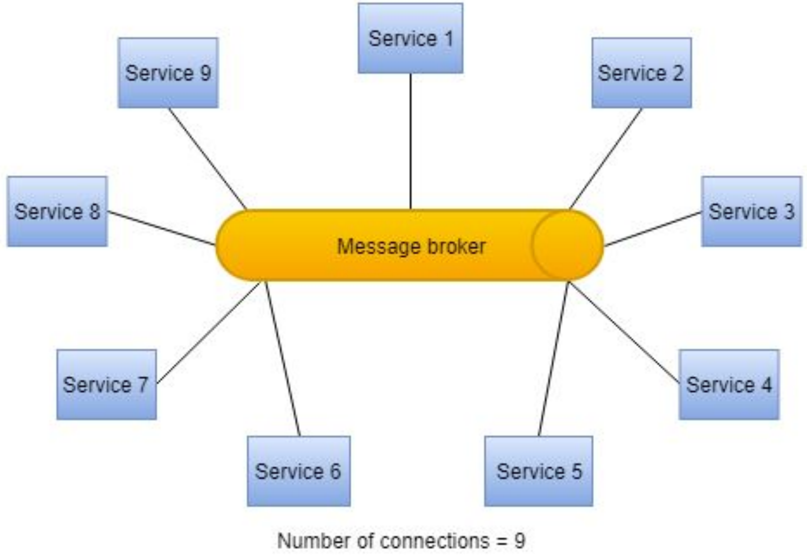


Number of connections = 10

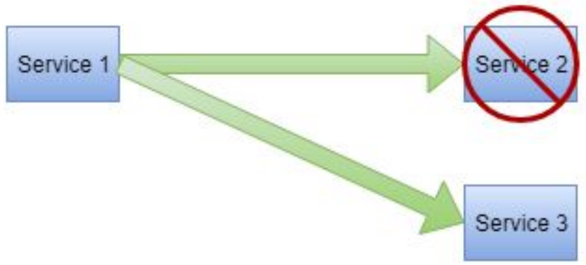


Number of connections = 36

By contrast, in event-driven microservice architectures supported by messaging, services produce events and subscribe to the events they want to use.
In this system services are not directly connected, there is a message broker between them, so adding new service will not increase communication complexity.



Recovery from error



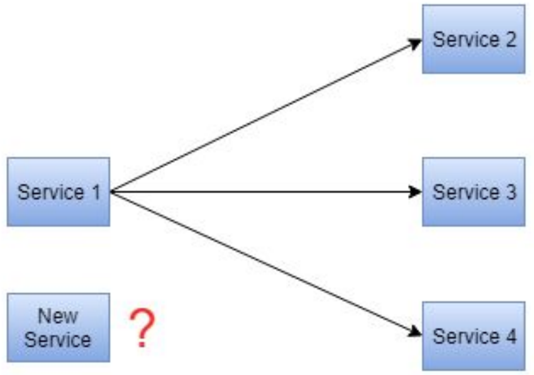
In a non messaging environment, if the receiving service fails, then ideally a new instance of that service will spin off, the sending service will have to detect the failure and after a certain number of tries it will go on the next instance. This perfect scenario is complex to engineer in the real world. Now think how much more complex this gets as new applications are added to the system.

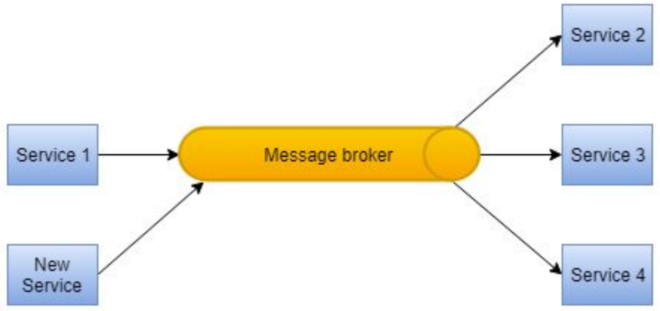
In an event-driven message broker environment, services connect to the broker, not to one another. Sending services emit events, and new receiving applications connect to the broker and subscribe to events. When a new event message is emitted to the broker, the messaging system knows which applications are active and which are not and the messages are sent only to active applications. In case of failure, messages will arrive and stay in the queue, until the new application is ready to receive them.

System Scalability

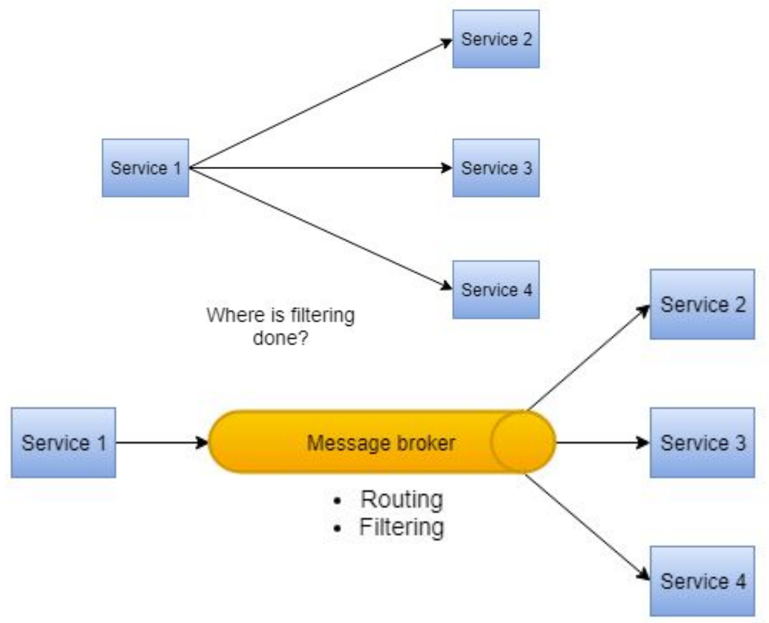
In the microservices where applications are directly connected, they can affect the efficiency of one another. For example, if a sending service is creating information at a rate that is too high for a receiving service to process, a situation will arise where the sending service accumulates messages while it waits for the slow receiving service to be ready. If you want to increase the load for either type of services, you will have to add a logic for spreading the load between the services.

Alternatively, in an event-driven messaging environment, all the communication will go through the broker. You can track the rate of messages being sent and received and also the size of the queue, and you can determine is there a need to increase the number of services. Adding new service of any type is easy, as they only have to connect to a message broker and the services automatically share the workload. This is incredibly useful in cloud applications where you can spin up applications in busy times and spin them down when it gets quiet. The message broker manages this process without the services having to know about one another.





Message Filtering and Routing



In situations where you have an application producing a number of events, and particular events need to go to multiple services, then doing this using HTTP/Rest or TCP is not easy. You would need to build a complex communication system. Further, if there are different types of events, and every service is consuming a certain subset of these events, you would have to do filtering on sending or receiving end, either way it would require too much work.

By contrast, in the event-driven messaging environments filtering and routing of the information is built into the system. Services don't have to know about each other. All they have to be aware of is the event and how to ask for it. The message broker can also contain rules about which applications can subscribe to what type of messages, as well as what applications are allowed to send specific types of events. The broker authenticates services, and grants publishing and consumption permissions based on that authentication.

In Conclusion

Message broker is not a new concept, but it is gaining in popularity quickly. It can be used in many systems to improve communication between services. Two interesting examples where messaging is a good fit are Internet of Things and Microservices. We’ve seen that message broker can introduce many benefits. It offers bidirectional communication. Performances are better compared to HTTP/REST. It’s easier to scale up the system or to recover from failure. It adds a layer of abstraction between services and allows for fully asynchronous communications, and so on.

Message broker systems are also becoming more common in healthcare applications, as they are introduced with Microservices and IoT. As the current trend of data growth continues, message broker solutions will be used more often in the future.

ENABLING THE DIGITAL HEALTH REVOLUTION!

CONTACT

www.vicert.com
info@vicert.com
@vicert_inc

SAN FRANCISCO

1355 Market Street, Suite 488
San Francisco, CA 94103, USA

+1.415.495.7700

VICERT