Faculty of Engineering and Applied Science

SOFE 4790U Distributed Systems

Homework: Gateways

Mamun Hossain - 100553073

The idea of gateway is central to a distributed systems. Please read the following articles and try to come with a descriptions for;

The requirements of a gateway, and a high level design of the various components needed for a gateway to operate.

The goal of a gateway is to send requests to the backend server and then send that request back to the client. This is very crucial for a distributed system. Let's consider the gateway routing pattern, and think about the problem at hand: a client would need to consume different services, multiple, or both at once. We can also consider the gateway offloading pattern which has things such as services that would require configuration, management, and maintenance. For example, being able to handle the security issues at hand, says ass SSL certificate management. Even the gateway aggregation pattern uses a gateway to aggregate multiple individual requests to turn it into a single request, but how can that all be done?

Let us begin with the **gateway routing pattern**: The main requirement is having a client to have the need to consume different services, and gateway routing pattern ensures a way to have different endpoints handle the request and routes, as we can see in the image below.

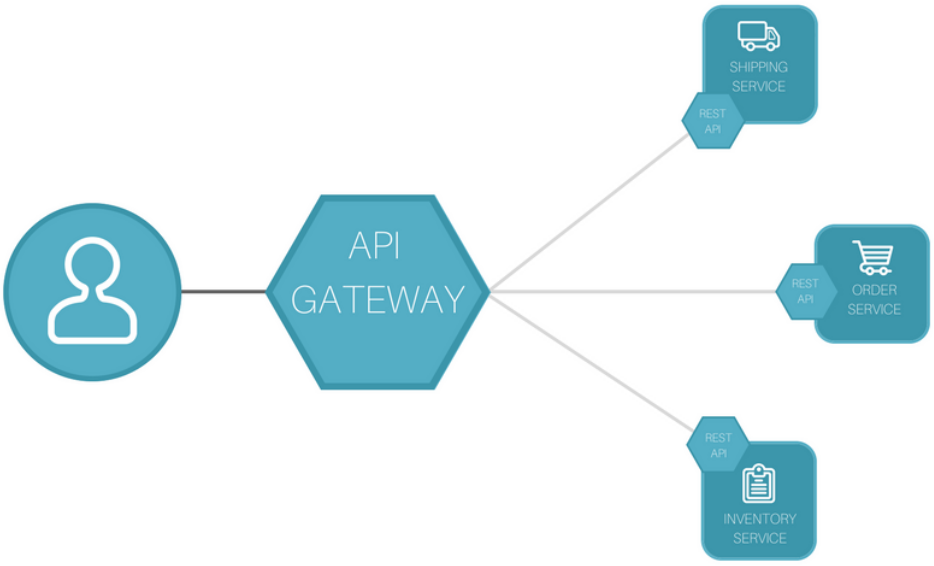

Figure 1: High level Gateway Routing Pattern

The solution to this would be to simply apply a gateway to the front of the application or services that are being used. When we consider the OSI layers, we would would Layer 7, the application layer, to route the requests to the required instances. With this, the client will only need to know about the single endpoint they interact with.

Now consider the **gateway offloading pattern**: the requirements that are required for this pattern would be that the gateway must be highly available and cannot be easily prone to failure. You will also want to ensure that the gateway is scalable for the aforementioned requirements and the application and its endpoints, and you do not want the gateway to become a bottleneck for the application. Below is an image that depicts this.

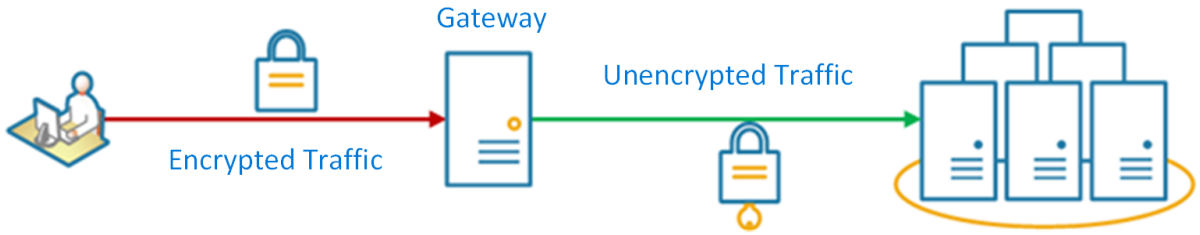

Figure 2: High level Gateway Offloading Pattern

You will want to use this pattern when a deployment has a concern with security, for example, a concern over the SSL certifications for the endpoints. If you use this pattern to track transactions, it would be highly beneficial to generating correlation IDs for logging purposes. That being said, this pattern is not always acceptable, for example it would not be suitable if there is coupling across different services.

Now for the **gateway aggregation pattern:** it is similar to the requirements of the gateway routing pattern, however it is also adding aggregation of services. What this means is that it offers a gateway service that aggregates multiple internal requests to different services and/or applications but it only exposes a single request to the client. So the requirements are the same as the routing, with the extra step. Below are high level designs that describe the way this gateway would operate.

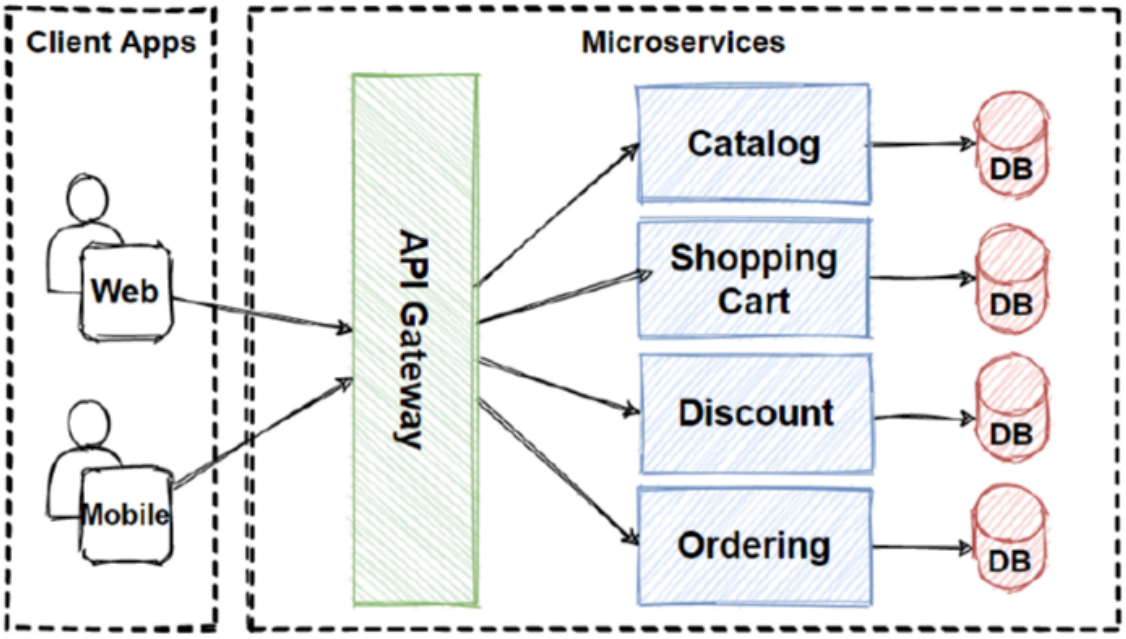

Figure 3 - Multiple clients

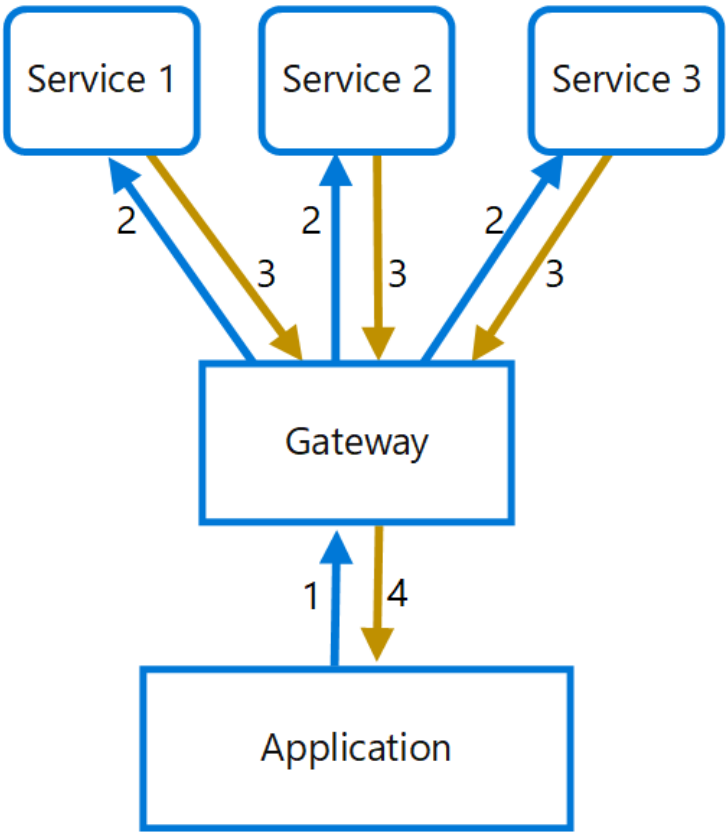

Figure 4: Single Application

The solution for this pattern is to include a gateway that reduces the chattiness between the client and the service. This way, the gateway will be able to retrieve the client requests individually, but will also be able to send the requests to different backend servers and then will aggregate the result that, which in turn, ends up sending the request back to the client. This also helps improve application performance over high-latency networks.

Because gateways are of such importance, they are typically using a clustering technique, where multiple instances of the gateways are deployed. Why would you do that?

The reason why you would multiple instances that insinuate a clustering technique is because you can enable many high-availability clusters of gateway installations, that will help the client and the services use data resources from different cloud services that can help. With these cloud services (eg. Power BI), because gateways are so important, clusters are used to avoid any singular failure point when accessing any on-premise data resource. The gateway cloud service will be used in the primary gateway in a cluster unless the gateway isn't able to be used, which then allows the service to access the next available gateway in the cluster.

A technique to get the cluster to operate is to use the Leader Election Pattern:

To describe this technique is that it is used as a method to ensure that the shared resources in a distributed system have as minimal conflict as possible. The way this works is through an election of task instances that is picked to be the leader, and the leader will then coordinate the other instances in the protocol that needs to be administered for that system. However, if all the task instances are running the same code, then they are all capable of being the leader.
With this leader election pattern, we can be sure that we can avoid problems such as writing data to the same entry at the same time. However, some issues that arise are the following: being able to detect when the leader has failed, or just unavailable.
The main difficulty with this pattern is that the leader must be robust and must work without any failures. We would tend to use this pattern to avoid making a leader a bottleneck in the system so the work can be coordinated efficiently.

Can you draw UML sequence diagrams for the leader election algorithm? and a sequence diagram for gateway use cases using clusters of instances?

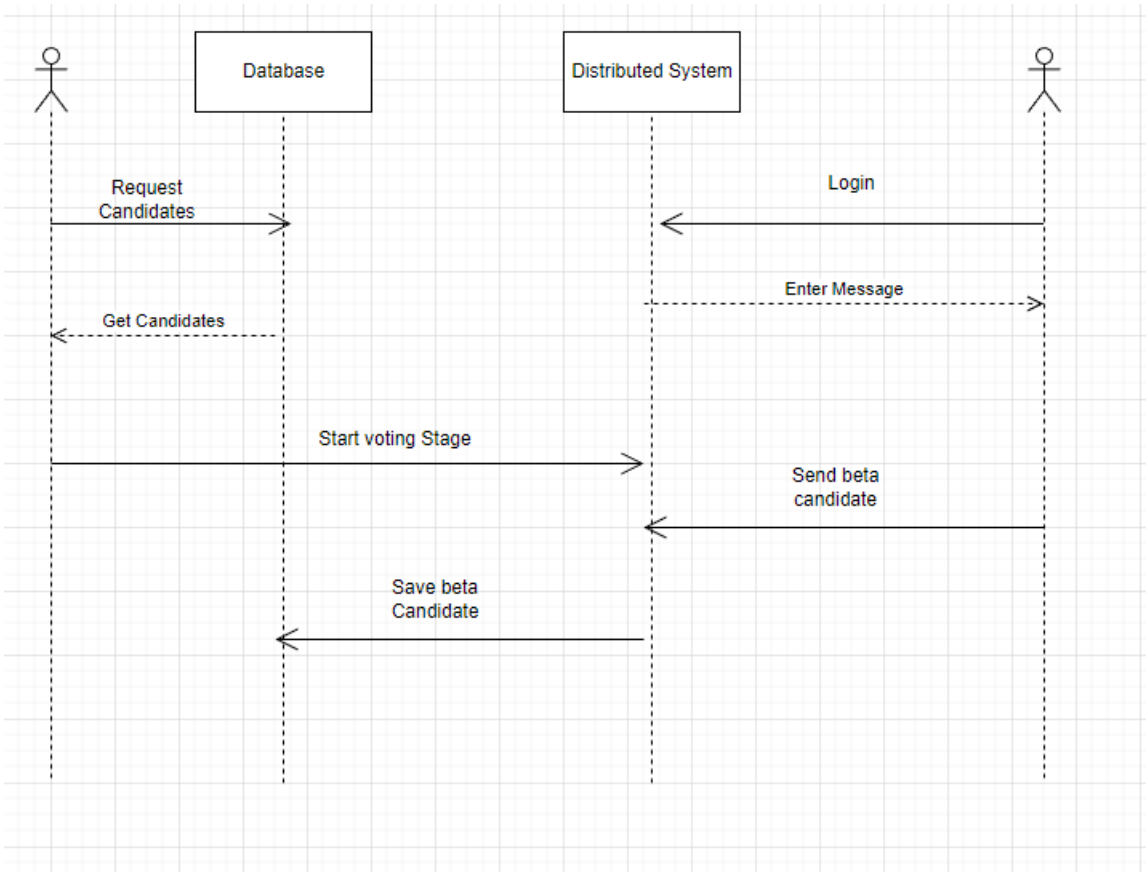

Figure 5: Leader Election Algorithm sequence diagram

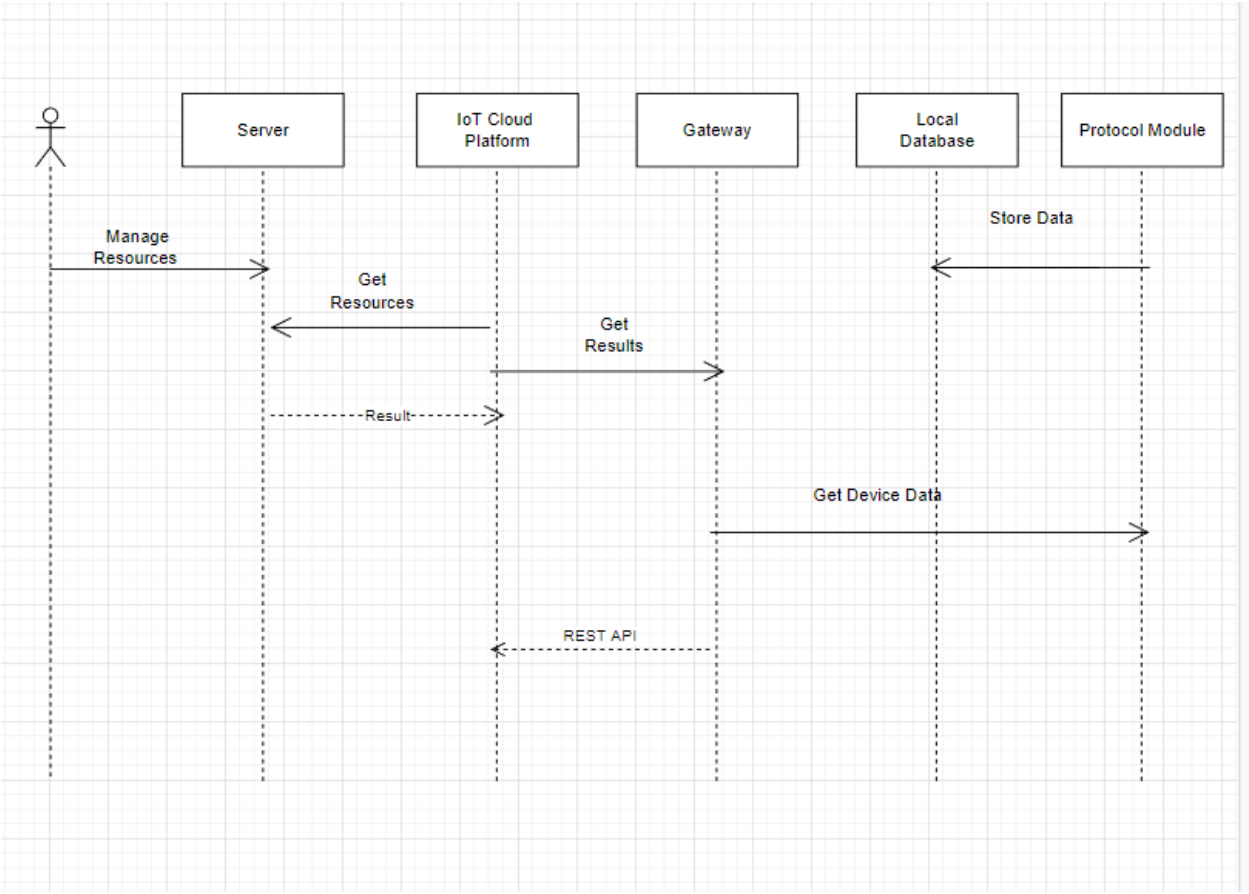

Figure 6: Sequence Diagram for Gateway Use Cases