Faculty of Engineering & Applied Science

SOFE4790U – Distributed Systems

Homework: Gateways

Due Date: 11/06/2022

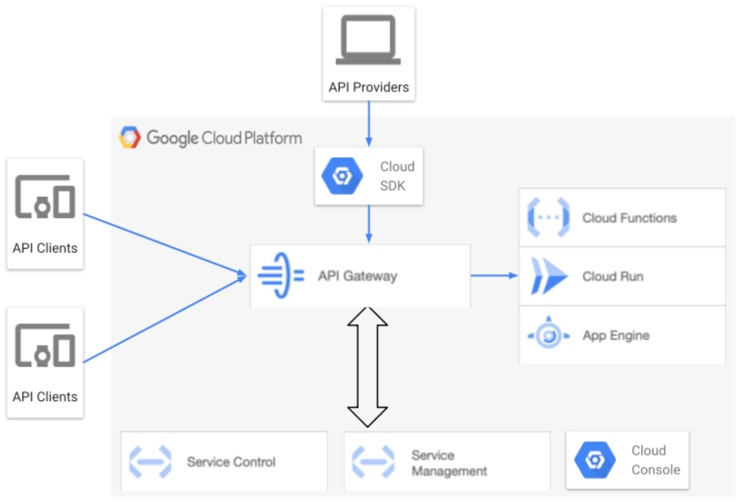
First Name	Last Name	Student ID
Abdul	Bhutta	100785884

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.

Gateway Components

Components	Description
Gateway	Managing the services
Control	Applying management rules
Management	Managing and configuration
Console	Deploy, manage, and monitor

An example of a high-level design or architecture for a gateway is shown below, which is used to implement a Google API Gateway. The main components are the gateway API, service control, service management, google cloud CLI, and cloud console.



Gateway Requirements

Requirement	Description
Frontend IP	Ip address to access the gateway
Port	To identify and listen to the request
Certificate	Provides security and privacy
Listeners	Used to listen to the request and logically send the request to the backend services.
Protocols	A type of communication among the components. (HTTP, HTTPS)

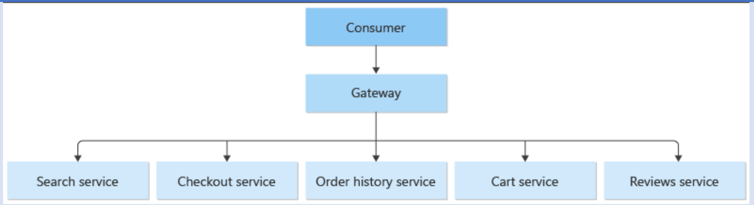
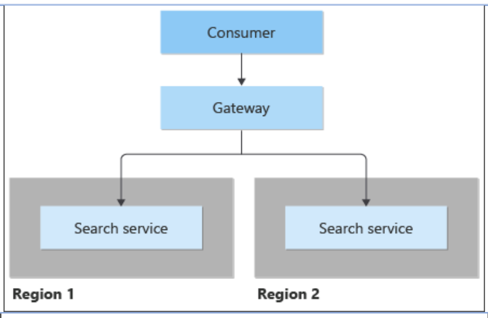
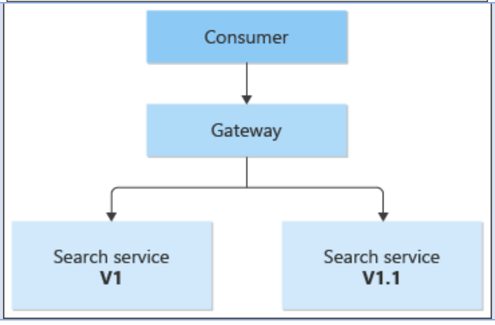
Requirements for Gateways

- *Reliable*: Gateway needs to be reliable, sending requests to the proper services, and being able to function correctly with heavy traffic loads.
- *Scalable*: Gateways can be scaled up and down with the traffic coming in, making for a more efficient system.
- *Availability*: Gateway should have a high availability to ensure clients can use the system at all times.
- *Performance*: The gateways should perform their tasks (Routing, Aggregation, Offloading) in an acceptable time.
- *Life-Time Costs*: The implementation of the gateway should have a lower cost when it is working for the system.
- *Reusability*: Reusability should be high for the implementation of the gateways, as this can be used in many other systems, reducing design and implementation times for future systems.
- *Maintainability*: The gateway's maintainability should be high, as in the case of a failure in a gateway it should be easy to mend the problem and get it back up and running quickly.

Gateway	**Description**
Gateway Routing	Route the various requests coming into various services into a single endpoint.
Gateway Offloading	Push shared service functionalities to a gateway proxy. Where it moves shared functionalities from applications to the gateway.
Gateway Aggregation	The gateway combines multiple incoming requests into a single request. Better when the client makes many requests to various backend systems.

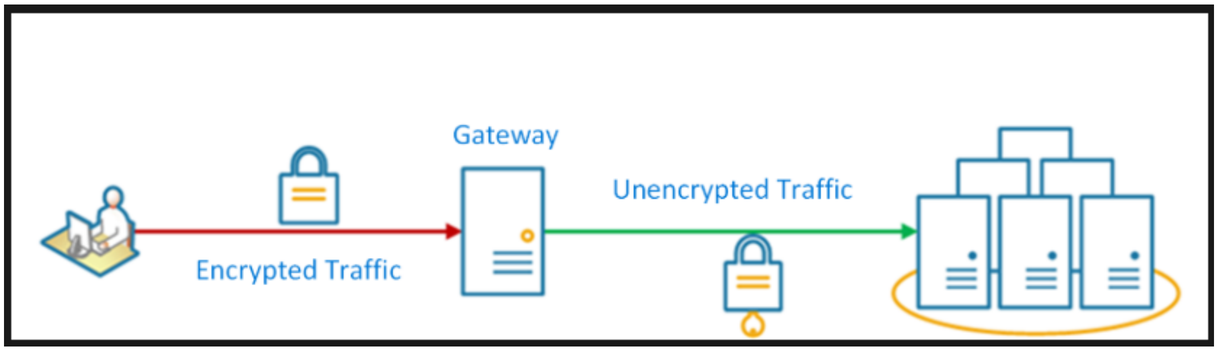
Gateway Routing Pattern

The concern is raised when multiple services, multiple instances of the same service, or multiple versions of the same service are updated, the client will be required to be updated as well. The solution to these concerns is deploying a gateway routing pattern which places a gateway at the endpoint and uses the application layer to route the traffic. This pattern allows the client to communicate with a single endpoint and hides the database or backend infrastructure. Examples of gateway routing patterns integrated within the applications are provided below.

Scenarios	Gateway Implentation
Multiple Services	 Consumer Gateway Search service Checkout service Order history service Cart service Reviews service
Multiple Instances of the the same servces	 Consumer Gateway Search service Search service Region 1 Region 2
Multiple Versions	 Consumer Gateway Search service V1 Search service V1.1

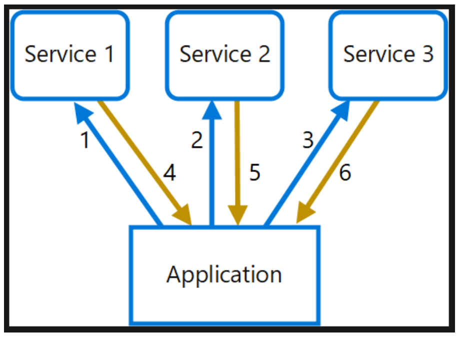
Gateway Offloading Pattern

As there are many applications which require services to be shared and deploying a service in each application causes overhead or deployments error. Some examples of services are security, authentication, and monitoring. Implementing each service with the application is complex, and the gateway offloading pattern is used to solve this issue. The solution is to add the services within the gateway and allowing to simplify the deployments. An example of a gateway offloading pattern is shown below.

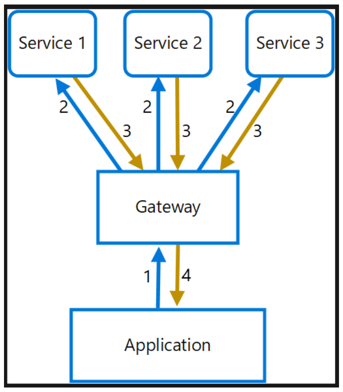


Gateway Aggregation Pattern

The gateway aggregation pattern is applied within applications which require communication with multiple requests to different databases or backend systems. When the client is sending multiple requests to the database or when it requires many services to perform one functionality, it will require the use of many resources, which will impact the performance and scalability of the application. An example of provided below where the application is requesting 3 services over a cellular network. The cellular network is unreliable, inefficient and has high latency, which may result in failure due to connectivity issues.



The solution is to implement a gateway in the middle to reduce the communication between the application and backend services. In the figure below, the gateway will receive the requests from the application, and the gateway will forward the requests to the services. The services will send back a response, and the gateway will combine the responses into a single response and forward it to the application.



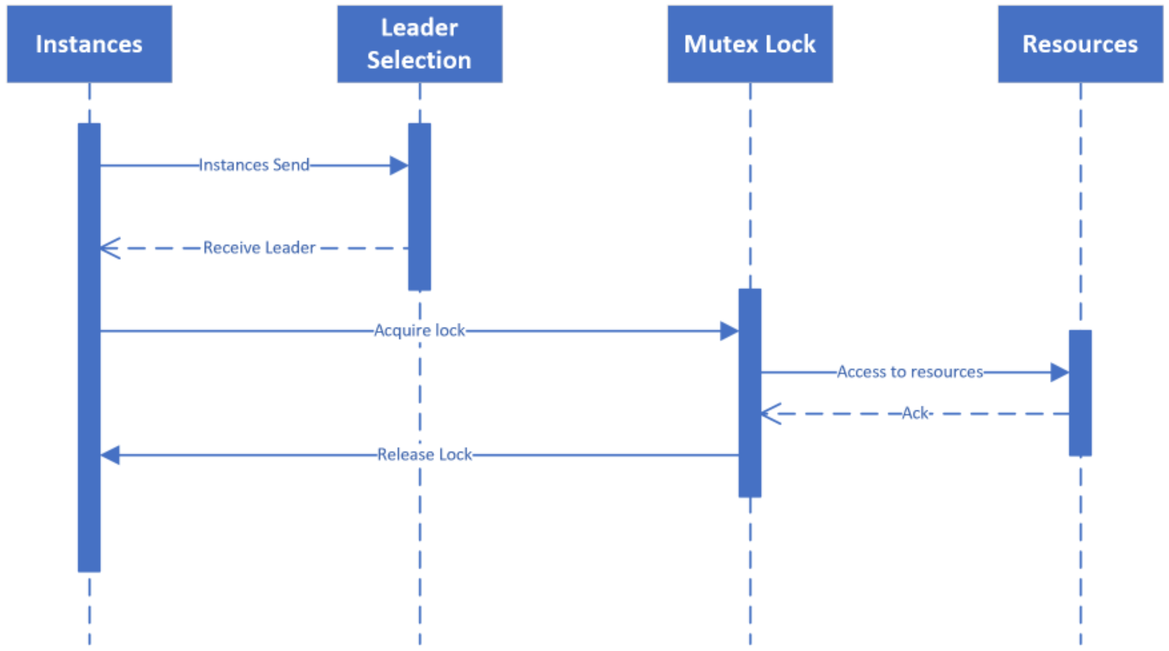
Because gateways are of such importance, they are typically using a clustering technique, where multiple instances of the gateways are deployed. Why would you do that?

The reason why you would have 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. Gateways use a clustering technique to allow redundancy, load balancing, and failover. This allows the use of another gateway in case one fails. The incoming requests are forwarded to the main gateway of the cluster, allowing for better performance and scalability.

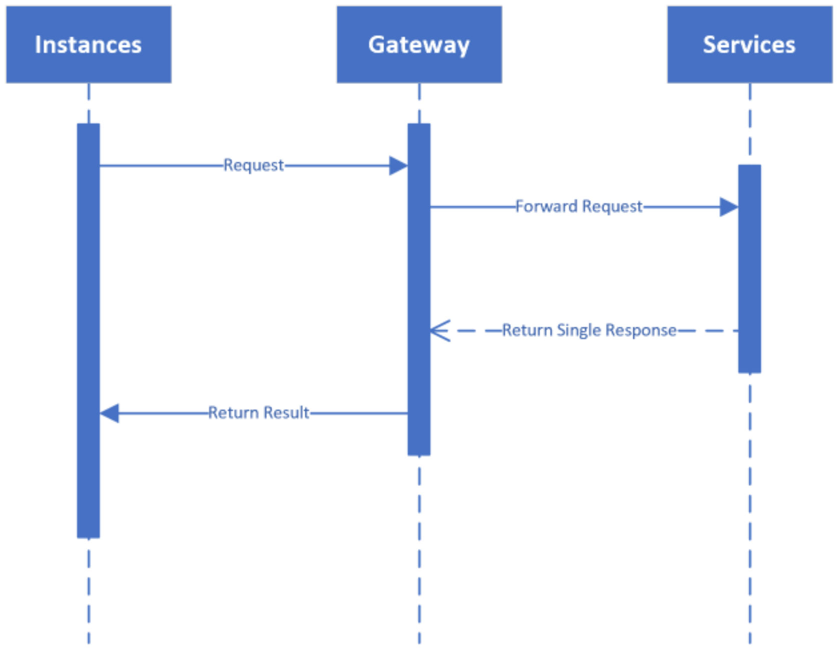
Leader Election Pattern

A typical cloud application consists of many tasks which may use the same resources in which a distributed system elects a leader to manger the other services. In a cloud application where multiple instances are provided to different users may be performing a task and needs to write to a database which is being shared by all the users. This may cause an issue as one task may overwrite the previous data. The leader election pattern is applied to coordinate and manage the instances.

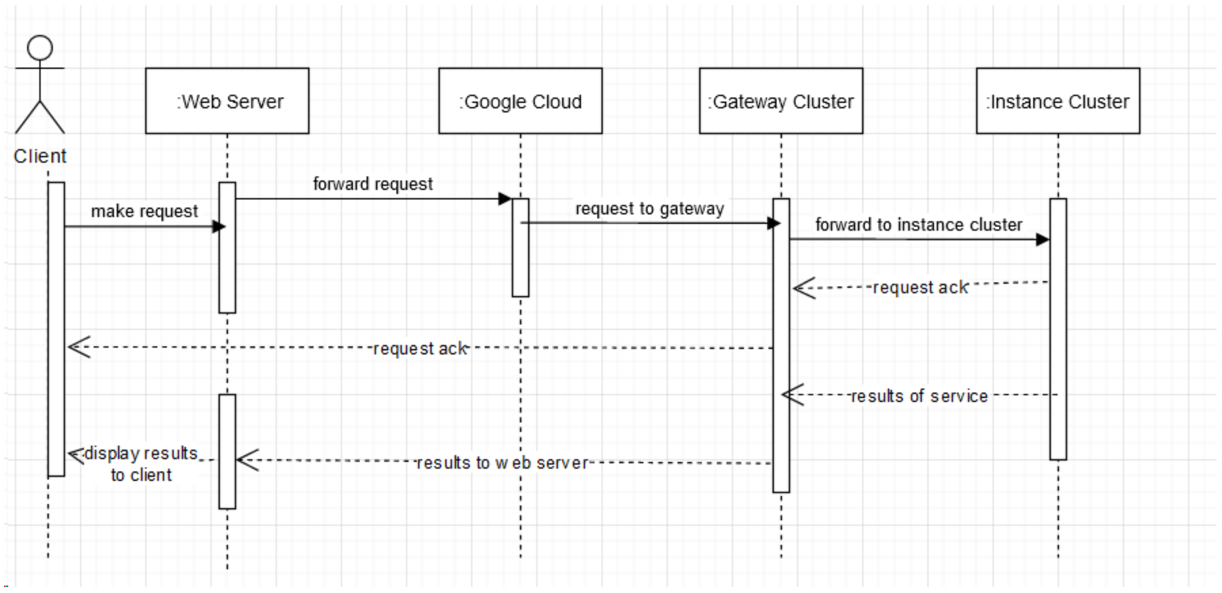
Sequence Diagram for Leader Election Algorithm



Sequence Diagram for Gateway and multiple Instances



Not sure which one is right


Client
:Web Server
:Google Cloud
:Gateway Cluster
:Instance Cluster
make request
forward request
request to gateway
forward to instance cluster
request ack
request ack
results of service
display results to client
results to w eb server