Faculty of Engineering and Applied Science

SOFE 4790U Distributed Systems

Homework #4

Name	Student #
Owen Musselman	100657709

Part I:

Come up with descriptions for; The requirements of a gateway, and a high level design of the various components needed for a gateway to operate.

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.

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.

Need to use application layer 7 so that requests are routed to the correct instance to be dealt with, this is gateway routing. With gateway routing there can be multiple disparate services, where the client can utilize various services, where the client is able to keep making requests, and the gateway will change its routing, allowing the client to not have to change anything. With gateway routing, there can be various instances of the same service, where the services connected to the gateway can be increased or decreased depending on the traffic to save resources. There can also be multiple versions of the same service, where the gateway can route the requests to a service that can serve them (one version can do something but another cannot)Can use something like nginx that will move SSL operations to one of various HTTP servers to be dealt with accordingly, this is gateway offloading. In order to implement this, gateways should have the gateway close to the physical location of the backend services in an attempt to reduce latency. Design should also be adequate and have gateway performance with meet growth and usage. Aggregation should be built into the gateway as it could have an effect on the routing and offloading of the gateway, and this will mitigate any inconsistencies in the offloading and routing. Through the use of aggregation it will decrease the amount of requests from the application to the services in the backend. Asynchronous input and output should be used to make sure there are small performance issues for the application. Additionally, it should have a robust design utilizing circuit breakers, timeouts, and bulkheads to make the system more available, and reliable. The design of gateways could have a single point of failure, and the gateway should be designed to accommodate the availability of the system.

Part II:

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? A technique to get the cluster to operate is to use the Leader Election pattern.

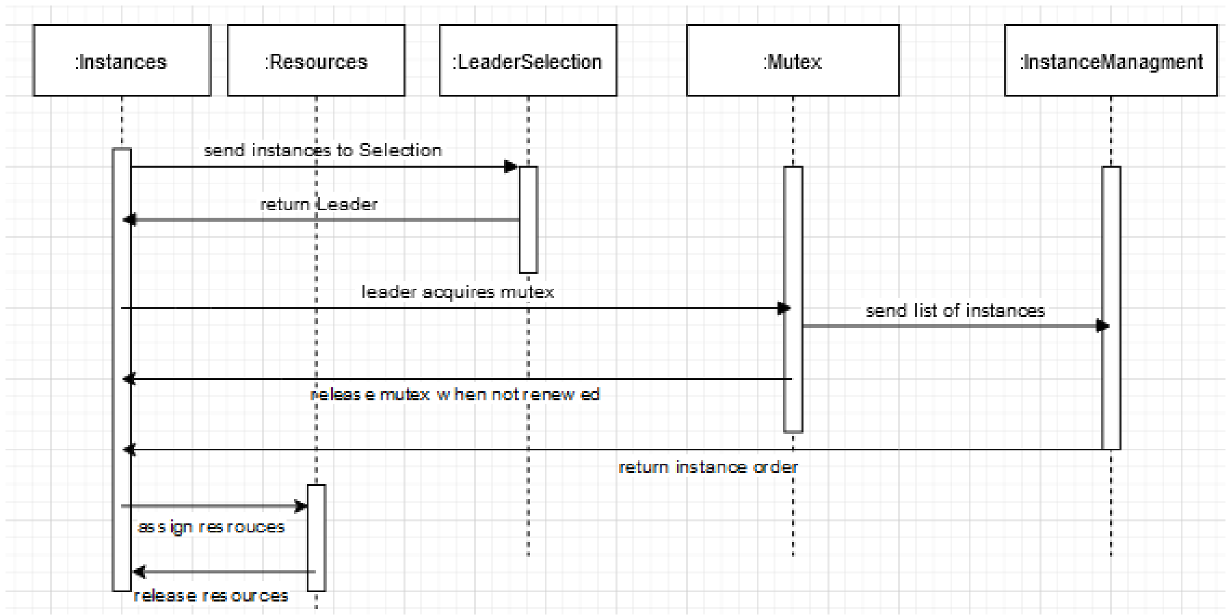
The use of clustering a gateway is an important one. By clustering gateways incoming requests are sent to the main gateway instance within the cluster. In the event that this main gateway is not operational, the incoming requests will be sent to another gateway within the cluster. This is accomplished through patterns like the Leader Election pattern, where the main component (gateway) is chosen.

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

Leader Election Pattern:

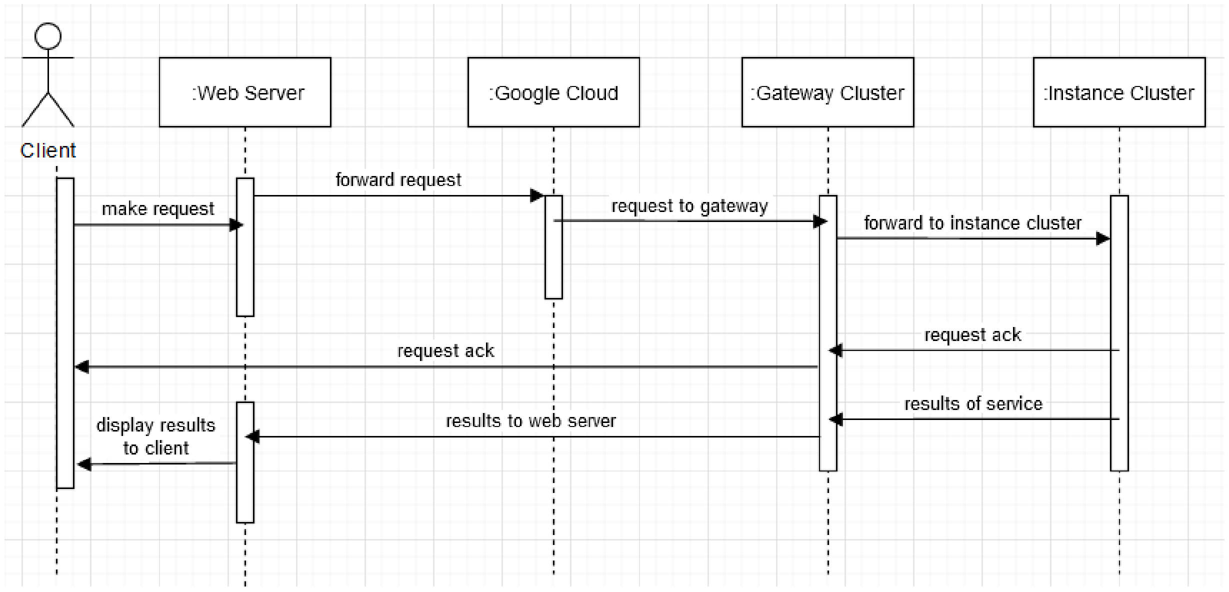
The Leader Election pattern harmonizes the operations that are done by a set of instances that are working together in a distributed application. This pattern will elect one of the instances of the distributed application as the leader, in which the leader is in charge of managing the other instances. This makes it so that the instances will not conflict with one another, like fighting over resources or other instance’s work.

UML Sequence Diagram - Leader Election



The set of instances is sent to a leader selection algorithm (could be the ring algorithm) where the leader is decided. Once decided that instance will acquire the mutex that belongs to the leader. When the mutex is acquired, the instances are sent to an Instance Management method where the instances are arranged in a way where they do not conflict with each other or shared resources, where the proper instances are assigned the resources, and then release them when they are done.

UML Sequence Diagram - Gateway use cases using clusters of instances



The client sends its request to the web server. This web server utilizes google cloud platform for clustering capabilities. The google cloud platform is just depicted to show that is what is being used in this case, but requests would be forwarded to the clusters. The client's request is then forwarded to the gateway cluster where a main gateway has already been chosen using a pattern like the Leader Election pattern. From the gateway cluster the request is sent to the instance cluster where the request is processed. When the request reaches the instance cluster in this case, an acknowledgement is sent back to the client. When the request is served, the results are then sent back to the web server where it is displayed to the client. With the use of these clusters the availability, reliability, scalability, and performance are increased for example.