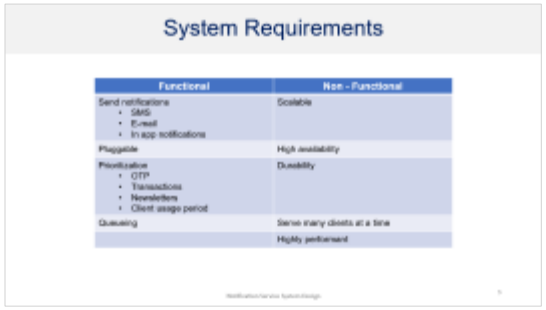
Notification Service System Design for Irembo

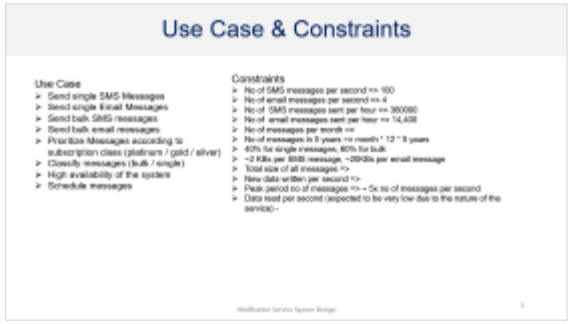
13 Oct 2020

Oladipo Oyekanmi

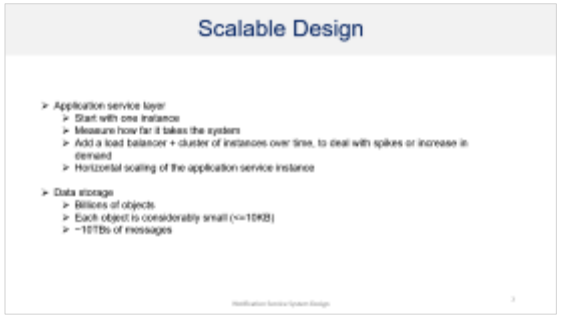
Outline

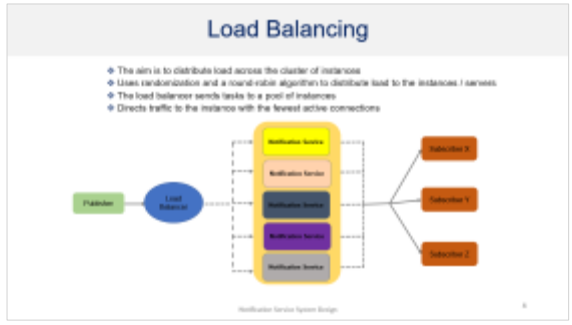

System Requirements
Functional
Non - Functional
Scalable
High availability
Durability
Queueing
Highly performant

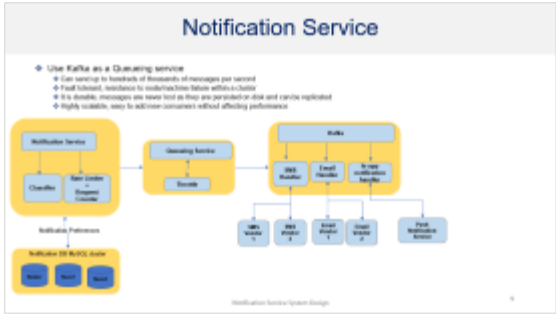

Problem Statement
Publisher
Notification Service
Subscriber X
Subscriber Y
Subscriber Z

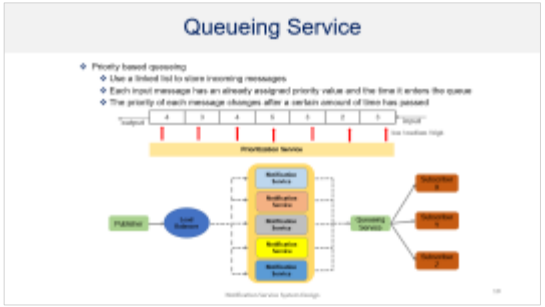

Use Case & Constraints
Use Case
Send single SMS Messages
Send single Email Messages
Send bulk SMS messages
Send bulk email messages
Prioritize Messages according to subscription class (platinum / gold / silver)
Classify messages (bulk / single)
High availability of the system
Schedule messages
Constraints

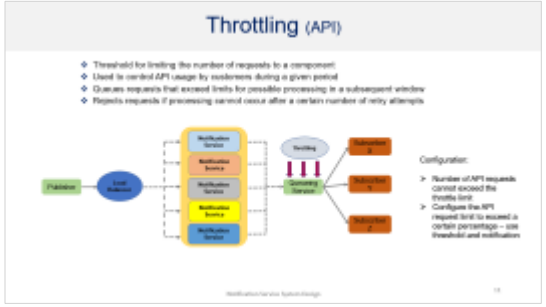

Abstract Design / Bottlenecks
Application service layer (serves the requests)
Authentication / subscription service
Queueing service
Throttling service
Load balancing service
Publishing service
Classifying service
Publishing service
Data storage layer (keeps track of all messages sent and user details including date and time)
Scalability

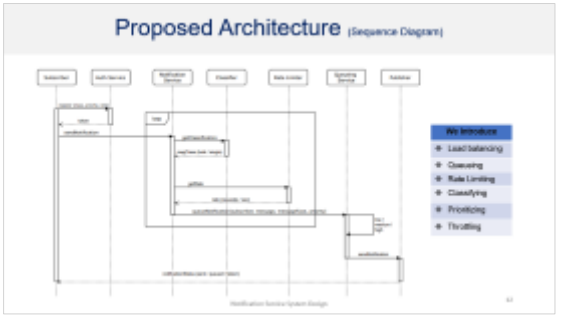

Scalable Design
Application service layer
Start with one instance
Measure how far it takes the system
Add a load balancer + cluster of instances over time, to deal with spikes or increase in demand
Horizontal scaling of the application service instance
Data storage
Billions of objects
Each object is considerably small (<=10KB)
~10TBs of messages

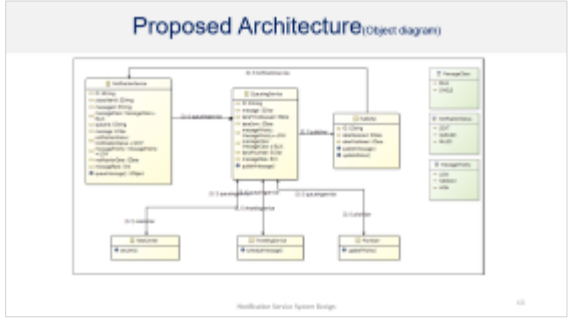

Load Balancing
The aim is to distribute load across the cluster of instances
Uses randomization and a round-robin algorithm to distribute load to the instances / servers
The load balancer sends tasks to a pool of instances
Directs traffic to the instance with the fewest active connections
Publisher
Load Balancer
Subscriber X
Subscriber Y
Subscriber Z


Notification Service
Use Kafka as a Queueing service


Queueing Service
Priority based queueing
Use a linked list to store incoming messages
Each input message has an already assigned priority value and the time it enters the queue
The priority of each message changes after a certain amount of time has passed
Publisher
Load Balancer


Throttling (API)
Threshold for limiting the number of requests to a component
Used to control API usage by customers during a given period
Queues requests that exceed limits for possible processing in a subsequent window
Rejects requests if processing cannot occur after a certain number of retry attempts
Publisher
Load Balancer
Throttling
Queueing Service
Configuration:


Proposed Architecture (Sequence Diagram)
We introduce
Load balancing
Queueing
Rate Limiting
Classifying
Prioritizing
Throttling

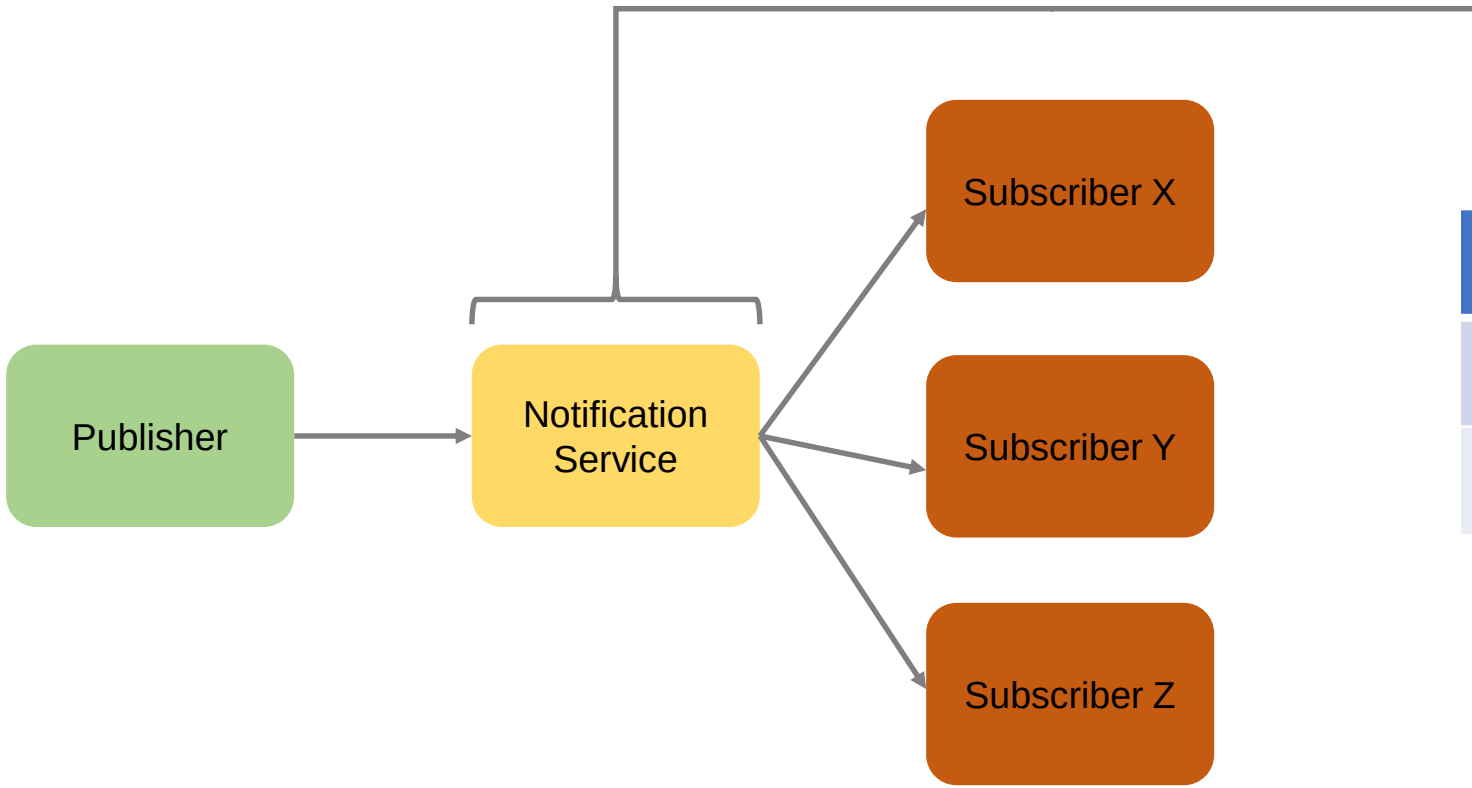

Proposed Architecture(Object diagram)


Summary
Load Balancing
Queueing
Rate Limiting
Throttling
Scalability
High Availability
Scalable, Highly performant, Highly available and Durable

System Requirements

Functional	Non - Functional
Send notifications • SMS • E-mail • In app notifications	Scalable
Pluggable	High availability
Prioritization • OTP • Transactions • Newsletters • Client usage period	Durability
Queueing	Serve many clients at a time
	Highly performant

Problem Statement



- **Too many requests from a specific client monthly**
- Too many requests across the entire system
- Too many requests within the same time window from a client

Use Case & Constraints

Use Case

- Send single SMS Messages
- Send single Email Messages
- Send bulk SMS messages
- Send bulk email messages
- Prioritize Messages according to subscription class (platinum / gold / silver)
- Classify messages (bulk / single)
- High availability of the system
- Schedule messages

Constraints

- No of SMS messages per second => 100
- No of email messages per second => 4
- No of SMS messages sent per hour => 360000
- No of email messages sent per hour => 14,400
- No of messages per month =>
- No of messages in 5 years => month * 12 * 5 years
- 40% for single messages, 60% for bulk
- ~2 KBs per SMS message, ~20KBs per email message
- Total size of all messages =>
- New data written per second =>
- Peak period no of messages => ~ 5x no of messages per second
- Data read per second (expected to be very low due to the nature of the service) -

Abstract Design / Bottlenecks

- Application service layer (serves the requests)
  - Authentication / subscription service
  - Queueing service
  - Throttling service
  - Load balancing service
  - Publishing service
  - Classifying service
  - Publishing service
- Data storage layer (keeps track of all messages sent and user details including date and time)
  - Scalability

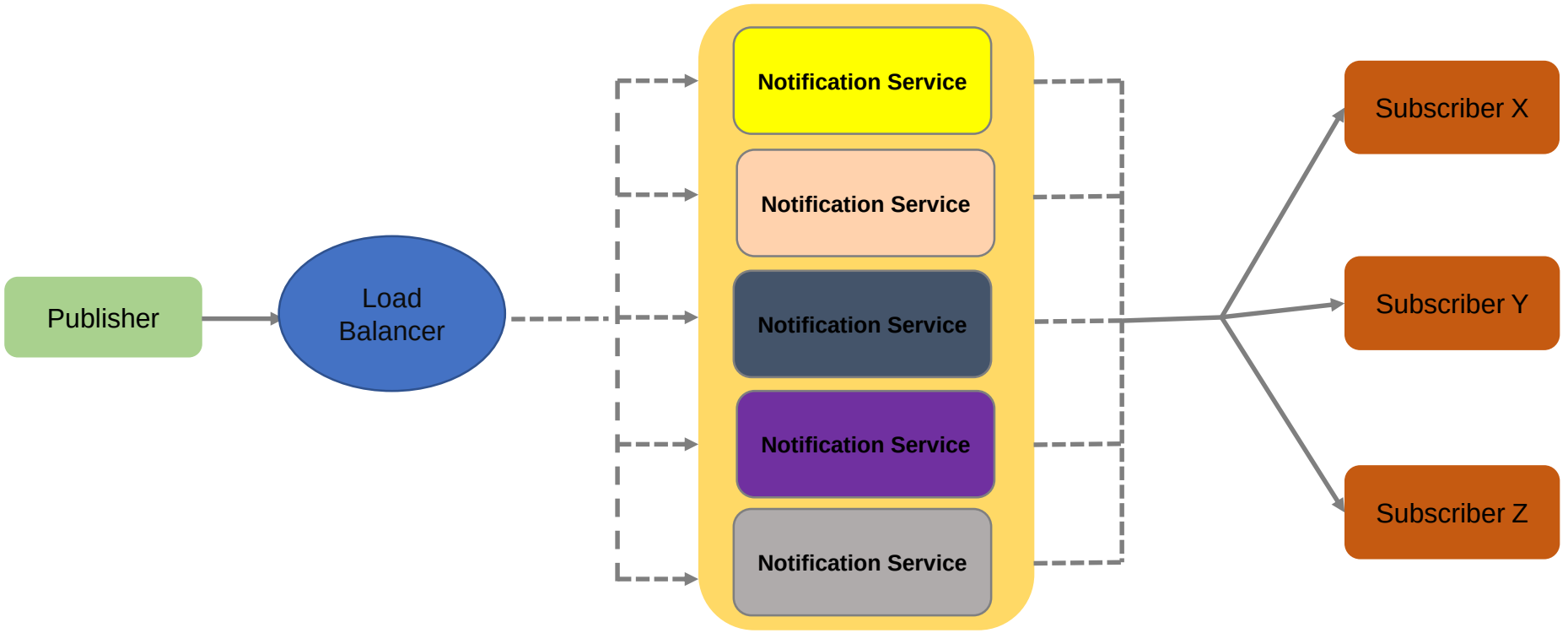
Scalable Design

- Application service layer
  - Start with one instance
  - Measure how far it takes the system
  - Add a load balancer + cluster of instances over time, to deal with spikes or increase in demand
  - Horizontal scaling of the application service instance

- Data storage
  - Billions of objects
  - Each object is considerably small (<=10KB)
  - ~10TBs of messages

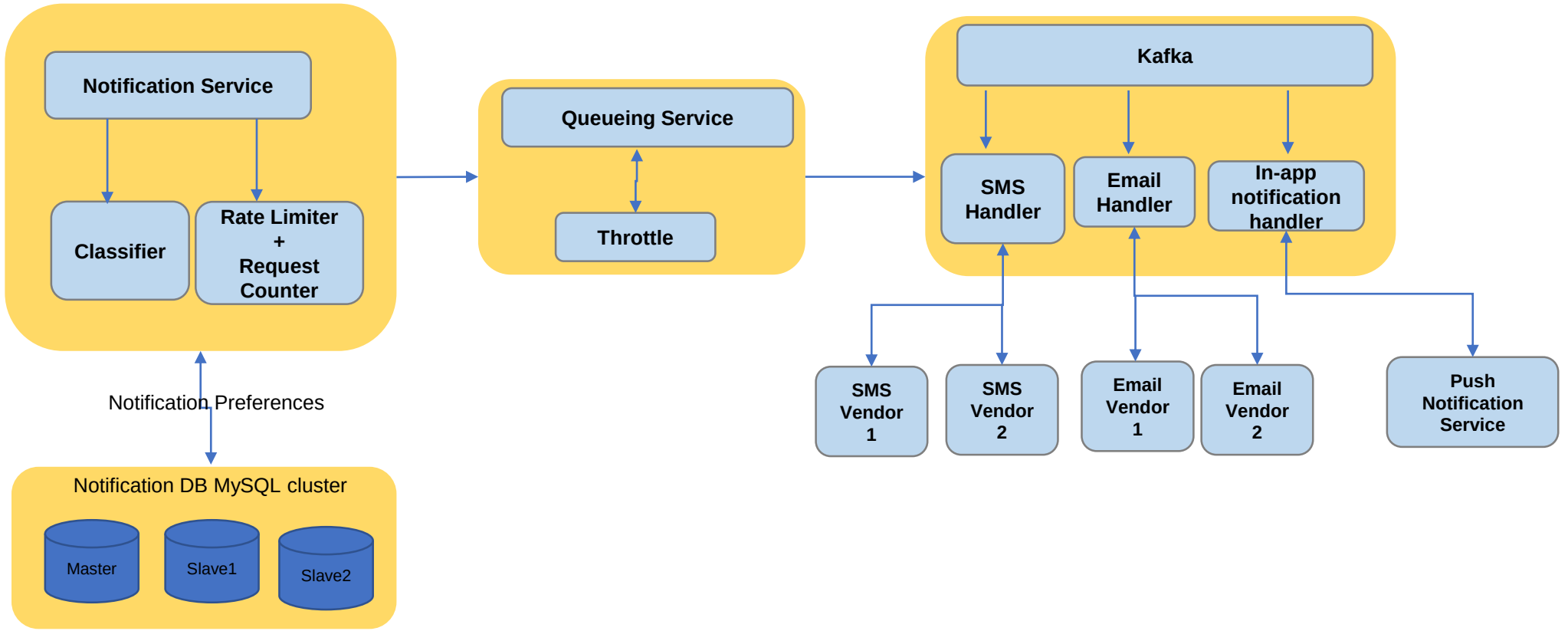
Load Balancing

- The aim is to distribute load across the cluster of instances
- Uses randomization and a round-robin algorithm to distribute load to the instances / servers
- The load balancer sends tasks to a pool of instances
- Directs traffic to the instance with the fewest active connections



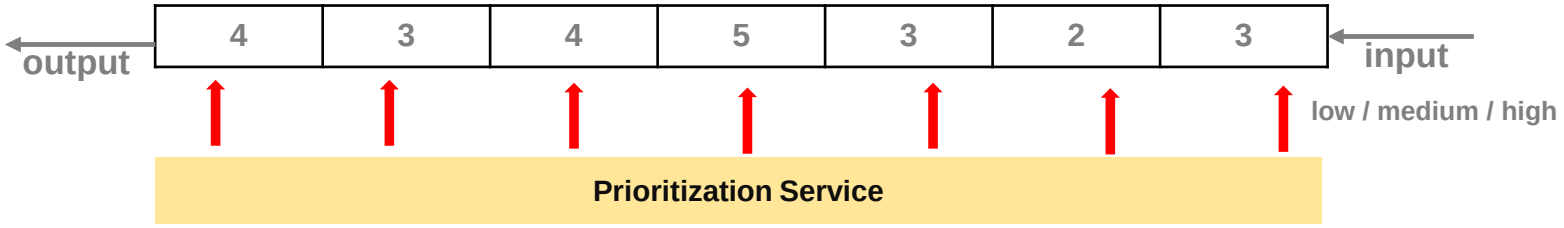
Notification Service

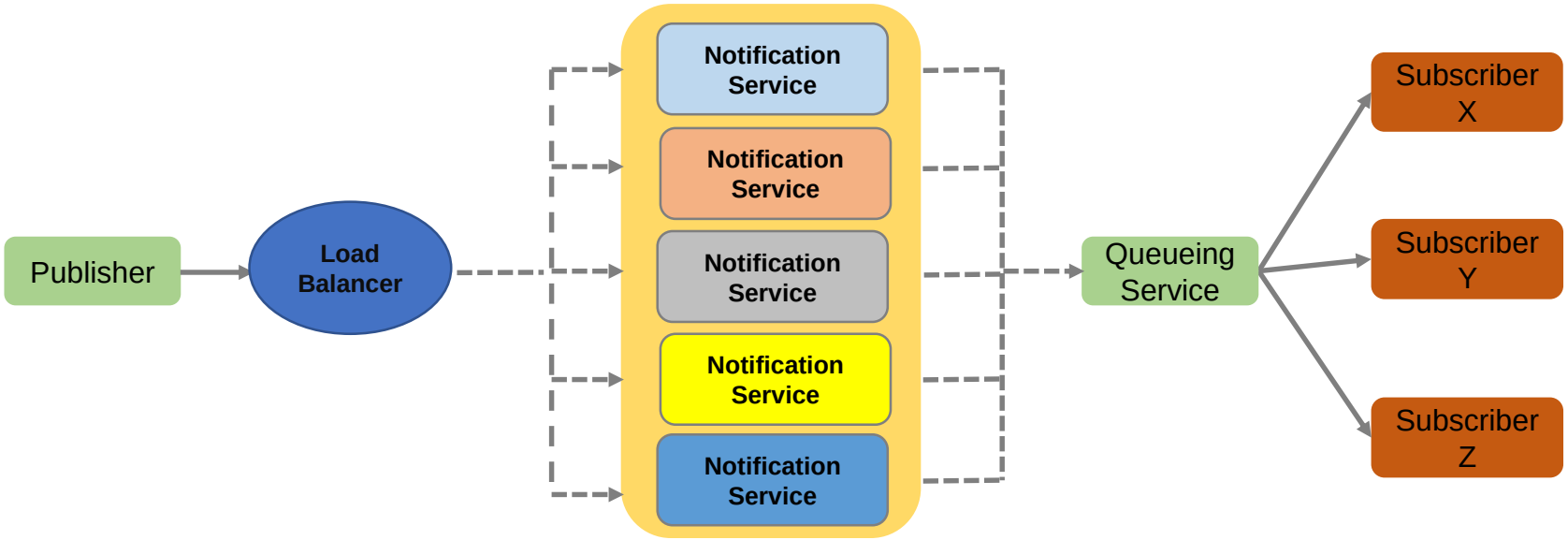
- Use Kafka as a Queueing service
  - Can send up to hundreds of thousands of messages per second
  - Fault tolerant, resistance to node/machine failure within a cluster
  - It is durable, messages are never lost as they are persisted on disk and can be replicated
  - Highly scalable, easy to add new consumers without affecting performance



Queueing Service

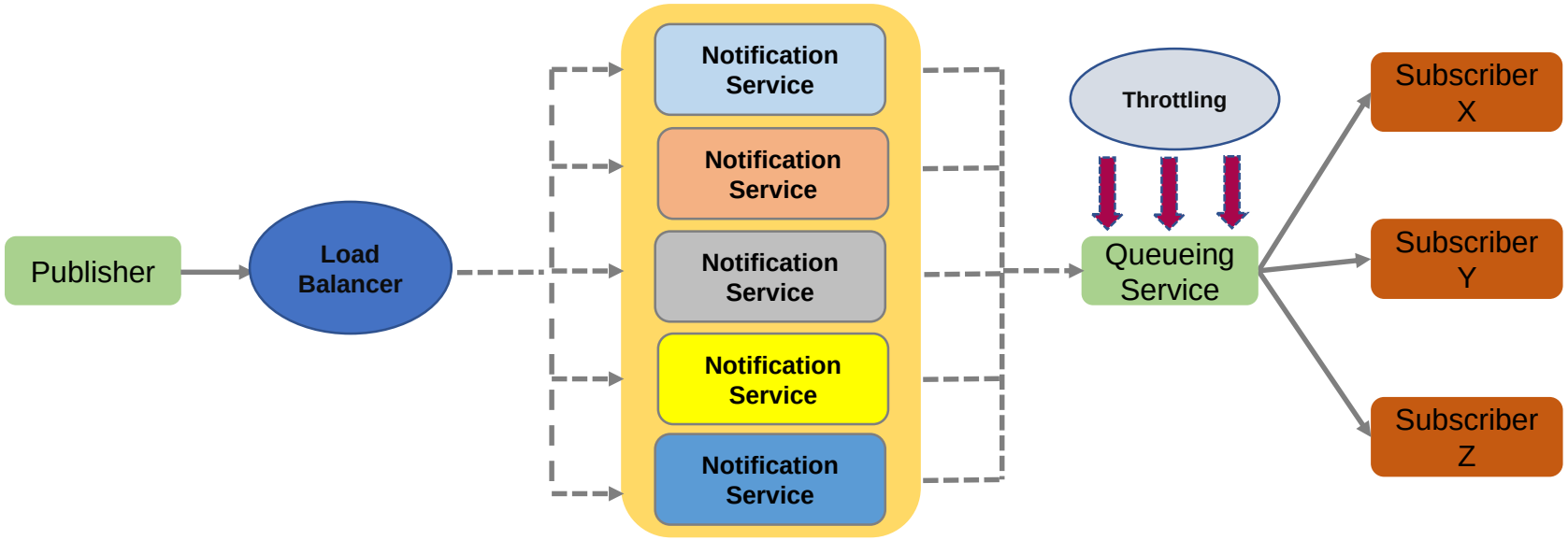
- Priority based queueing
  - Use a linked list to store incoming messages
  - Each input message has an already assigned priority value and the time it enters the queue
  - The priority of each message changes after a certain amount of time has passed





Throttling (API)

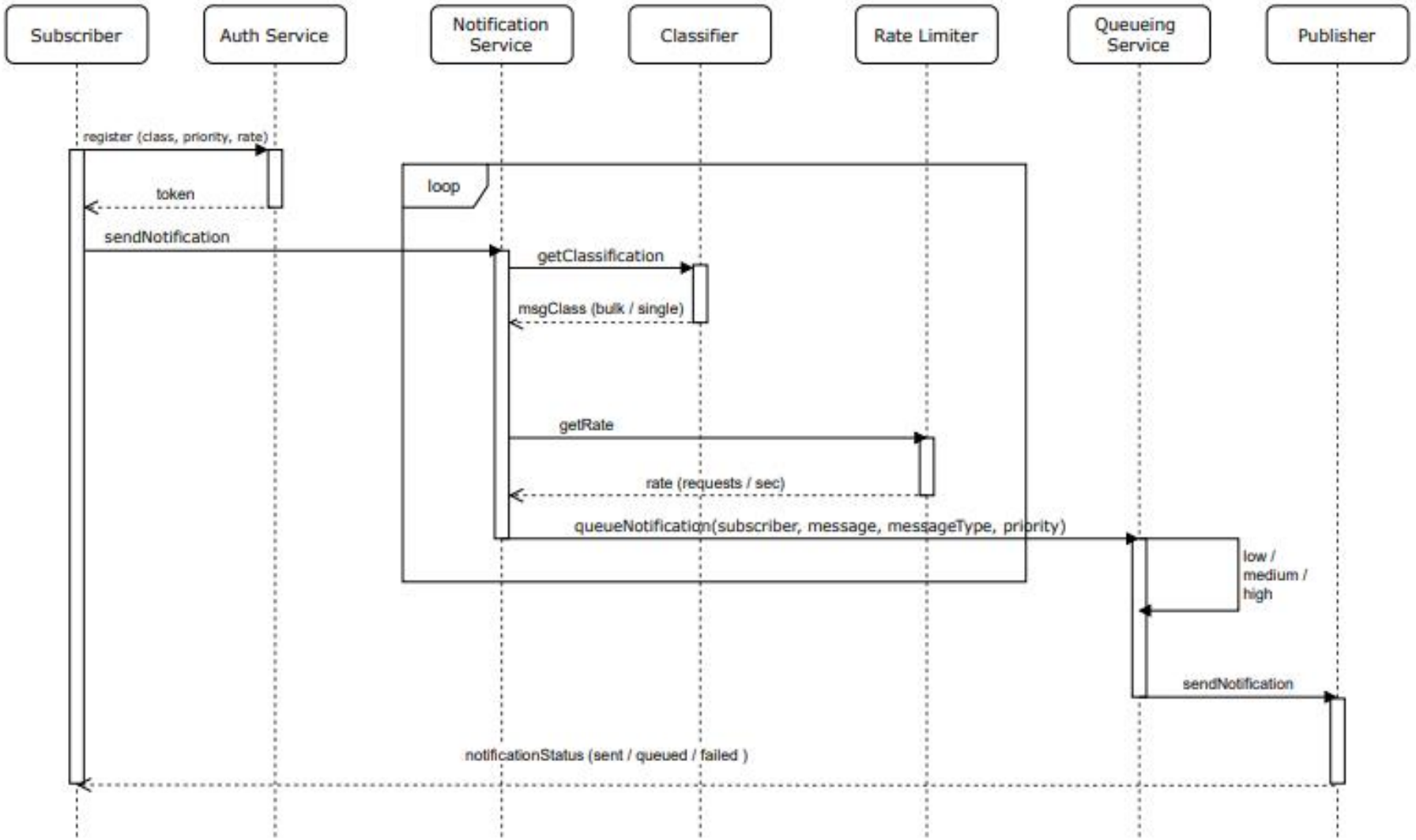
- Threshold for limiting the number of requests to a component
- Used to control API usage by customers during a given period
- Queues requests that exceed limits for possible processing in a subsequent window
- Rejects requests if processing cannot occur after a certain number of retry attempts



Configuration:

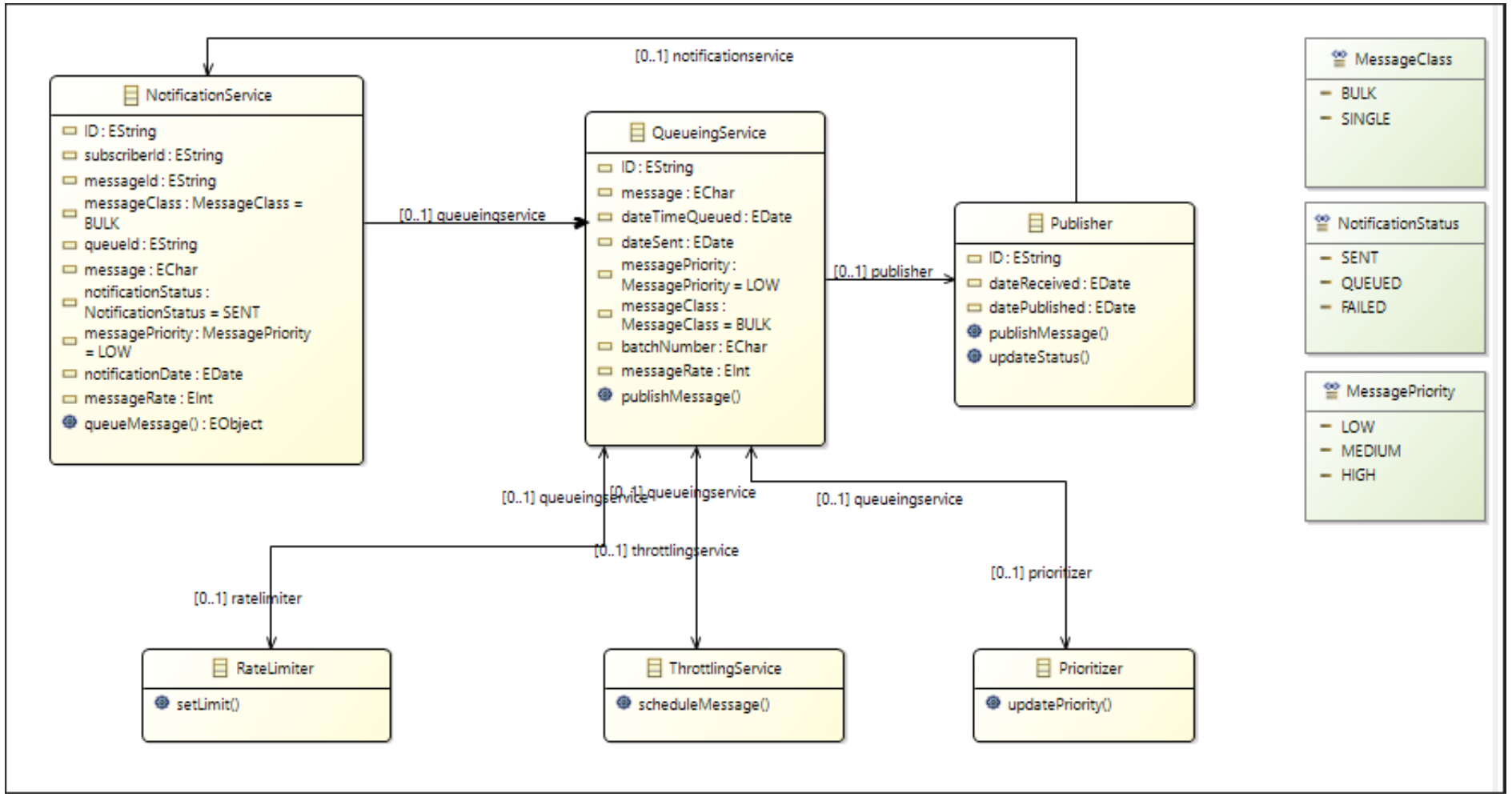
- Number of API requests cannot exceed the throttle limit
- Configure the API request limit to exceed a certain percentage – use threshold and notification

Proposed Architecture (Sequence Diagram)



We Introduce
❖ Load balancing
❖ Queueing
❖ Rate Limiting
❖ Classifying
❖ Prioritizing
❖ Throttling

Proposed Architecture(Object diagram)



Summary

- Load Balancing
- Queueing
- Rate Limiting
- Throttling
- Scalability
- High Availability

Scalable, Highly performant, Highly available and Durable