JOVIAL MESSAGING SYSTEM

FOR SCALABILITY, RELIABILITY AND HARDWARE/NETWORK FAILURE RESISTANCE

Jovial
{chat_system}

By

Rishav Bhowmik

Roll No. :1706349

Objectives

- Reliable messaging system
- Support a variety of devices and platforms
- Hardware & Network Failure resistance
- Offline Functionality
- Strict Queuing
- Strict identification of clients

This project is about

Conceptual & Practical Research of Jovial Messaging System

- Perform a Conceptual Study on requirements
- Identify requirements of Messaging System in Modern applications
- Identify challenges faced by Industry while implementing a Messaging System
- Prepare a suitable model that can be implemented in the maximum number of platforms, concerning their Hardware & Software Environments.

This project is about

Hosting Jovial Messaging System independently

- The Backend of Jovial Messaging System is not a fully Managed service.

  (NOT_YET!)
- For real world applications to Utilize this system we require a well designed ready to use Microservices.
- We also ensure that dependencies of this Project are open source and can be deployed independent.

  (Such as NodeJS, Rust, Rocket, Cassandra & MongoDB)

This project is about

Client Drivers for a variety of devices and Platforms which include

- Web Browsers
- Commonly used Linux distributions
- Linux subsystems such as Docker
- RIOT OS, Arduino & Lightweight Linux distributions

This project is about

Develop a Framework to fulfill the needs of modern industries

- Upcoming Boom number of IoT
- Utilize the revolutionary 5G networking
- Easy to Scale
- Fast Efficient & Cost effective

This project is about

A Framework to reduce development time of Applications

- Well tested open source libraries
- Easy to use High level abstraction
- Libraries for customizable usage
- Flexible design model to support updates in the application

Covering the gaps in existing systems

MQTT

- Message delivery NOT Guarantied
- NO User verification
- Limited/Unreliable client device Load balancing
- No offline functionality

Jovial Messaging System

- Message delivery Guarantied
- Mandatory User verification
- Load balancing with Guarantied message delivery
- Performs message question when network is down

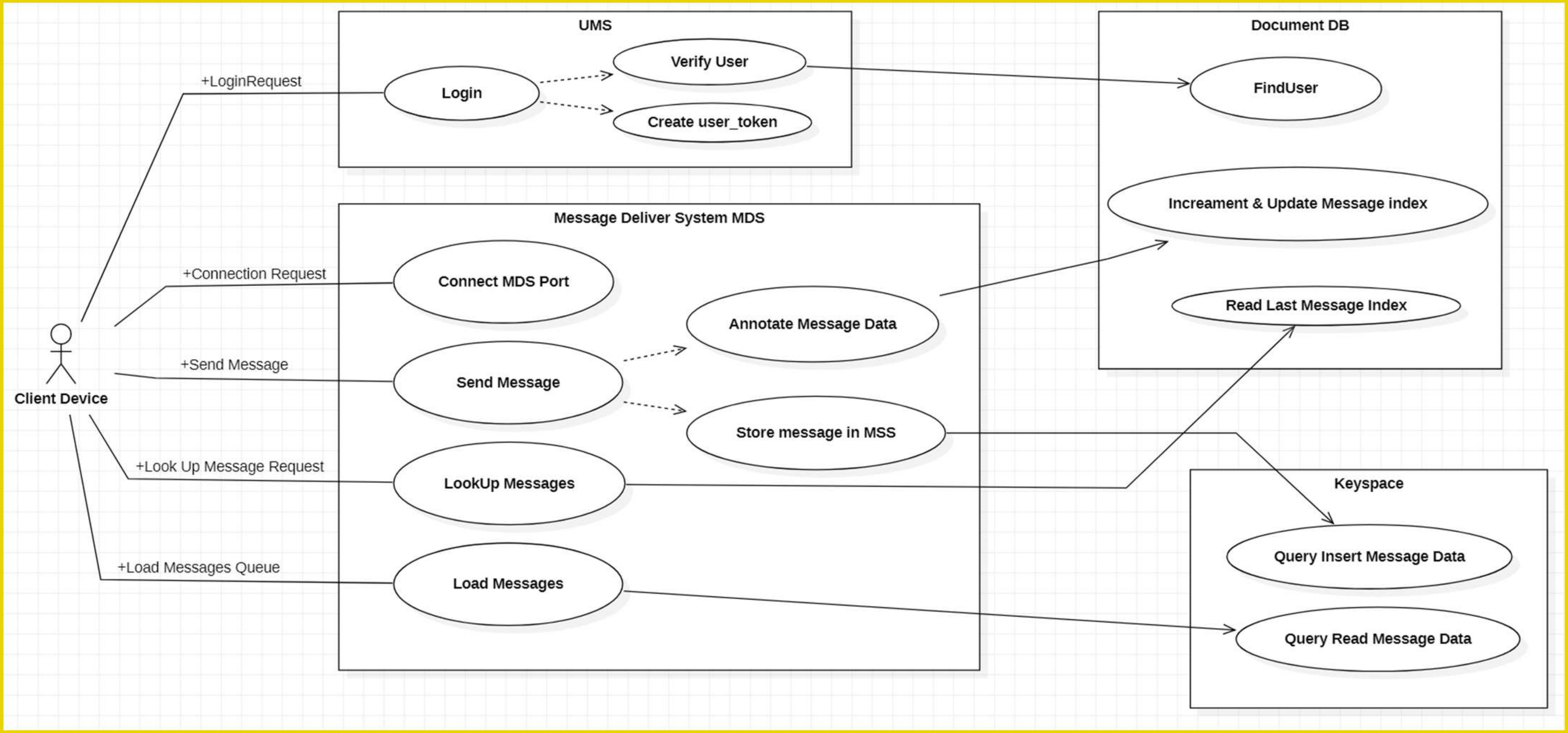
Basic Architecture

Jovial Chat System is designed for a Microservices architecture

The Backend Micro Services Involved :

1. UMS (User Management System)
2. MDS (Message Delivery System)
3. MSS (Message Storage System)

Basic Architecture



Client Device

The client is any physical entity that can send or receive messages using the system.

A **client** participates in the Message Delivery Process, using a **Client Device** or **A set Client Devices** to interact with ***Message Delivery System***.

Client Device

A Client Device has the following operations:-

- **Queueing**
- **Sending**
- **Receive**
- **Local Storage**

Client Device /User Device Set

A User can utilize **multiple devices** as **Use Device Set**.

A User Device Set performs **load balancing** on devices which come under it. *Which means,* any device can send a message from anywhere. And the devices in set will receive messages as per their **set_sub_factor**.

Client Device / Sending and Queuing

Creating New Message “Hi”

Creating New Message “Bye”

queue_index	message_index	sender_id	reciver_id	data
1	NULL	abcd88u21noinoin3332	abcd88u21noinoin8787	<Buffer..."Hi"...101>
2	NULL	abcd88u21noinoin3332	abcd88u21noinoin8787	<Buffer..."Bye"...102>

Message sent to MDS and stored in MSS

queue_index	message_index	sender_id	reciver_id	data
1	7	abcd88u21noinoin3332	abcd88u21noinoin8787	<Buffer..."Hi"...101>
2	11	abcd88u21noinoin3332	abcd88u21noinoin8787	<Buffer..."Bye"...102>

Client Device /Receiving and Storing

Get updated last message index

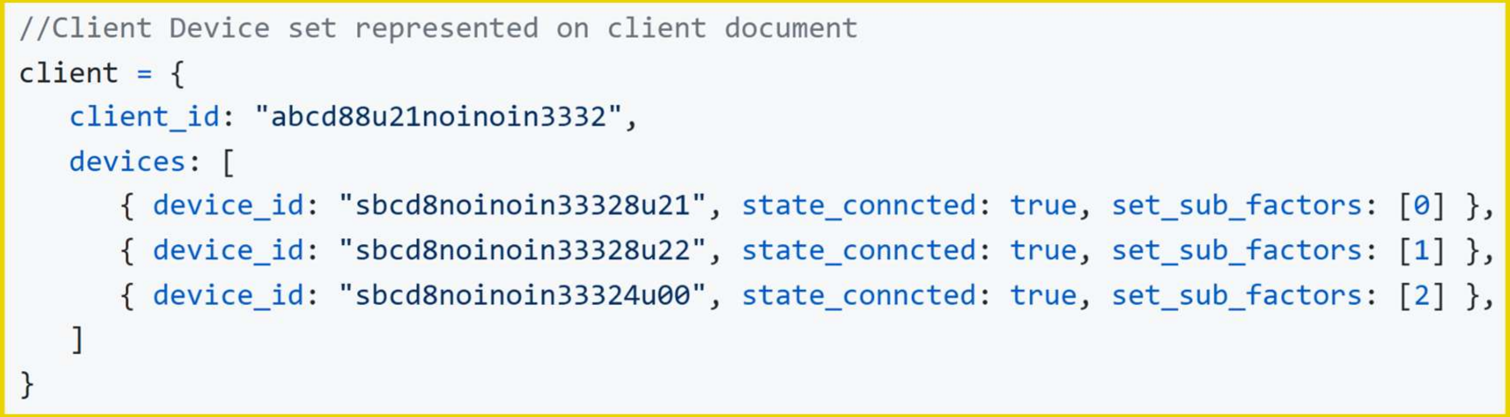
message_index	reciver_id	sender_id	data
1	null	null	null
2	null	null	null
3	null	null	null

Load Message with message index: [1,2,3]

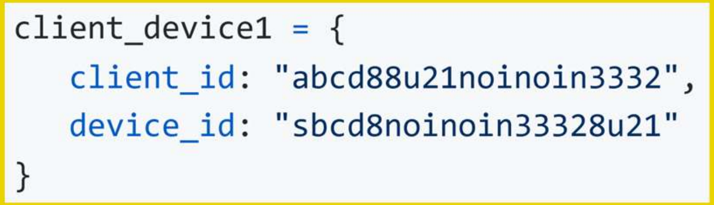
message_index	reciver_id	sender_id	data
1	abcd88u21noinoin3332	abcd88u21noinoin8787	<Buffer...403>
2	abcd88u21noinoin3332	abcd88u21noinoin8787	<Buffer...102>
3	abcd88u21noinoin3332	abcd88u21noinoin8787	<Buffer...134>

Client Device Set / Representation

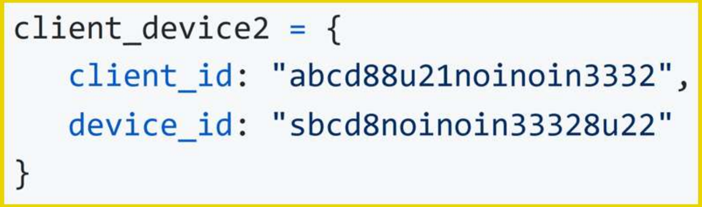
Device set for Client with
client_id: “abcd88u21noinoin3332”



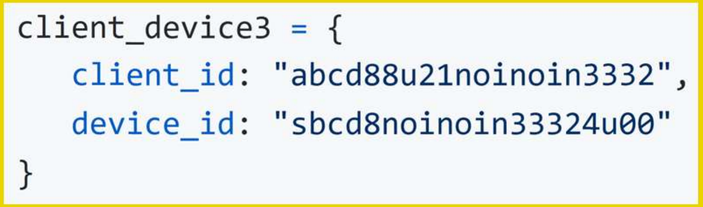
```
//Client Device set represented on client document
client = {
   client_id: "abcd88u21noinoin3332",
   devices: [
      { device_id: "sbcd8noinoin33328u21", state_conncted: true, set_sub_factors: [0] },
      { device_id: "sbcd8noinoin33328u22", state_conncted: true, set_sub_factors: [1] },
      { device_id: "sbcd8noinoin33324u00", state_conncted: true, set_sub_factors: [2] },
   ]
}
```



```
client_device1 = {
   client_id: "abcd88u21noinoin3332",
   device_id: "sbcd8noinoin33328u21"
}
```



```
client_device2 = {
   client_id: "abcd88u21noinoin3332",
   device_id: "sbcd8noinoin33328u22"
}
```



```
client_device3 = {
   client_id: "abcd88u21noinoin3332",
   device_id: "sbcd8noinoin33324u00"
}
```

Client Device Set / Receiving message

Messages **Received** by the Client

message_index	**reciver_id**	**sender_id**	**data**
1	abcd88u21noinoin3332	abcd88u21noinoin8787	<Buffer...400>
2	abcd88u21noinoin3332	abcd88u21noinoin8787	<Buffer...430>
3	abcd88u21noinoin3332	abcd88u21noinoin8787	<Buffer...100>
4	abcd88u21noinoin3332	abcd88u21noinoin8787	<Buffer...230>
5	abcd88u21noinoin3332	abcd88u21noinoin8787	<Buffer...550>
6	abcd88u21noinoin3332	abcd88u21noinoin8787	<Buffer...430>

Client Device Set /Receiving message

```
( message_index % set_device_count ) === set_sub_factor
```

Client's device 1

set_sub_factor = **0**

message_index	reciver_id	sender_id	data
3	abcd88u21noinoin3332	abcd88u21noinoin8787	<Buffer...100>
6	abcd88u21noinoin3332	abcd88u21noinoin8787	<Buffer...430>

Client's device 2

set_sub_factor = **1**

message_index	reciver_id	sender_id	data
1	abcd88u21noinoin3332	abcd88u21noinoin8787	<Buffer...400>
4	abcd88u21noinoin3332	abcd88u21noinoin8787	<Buffer...230>

Client's device 3

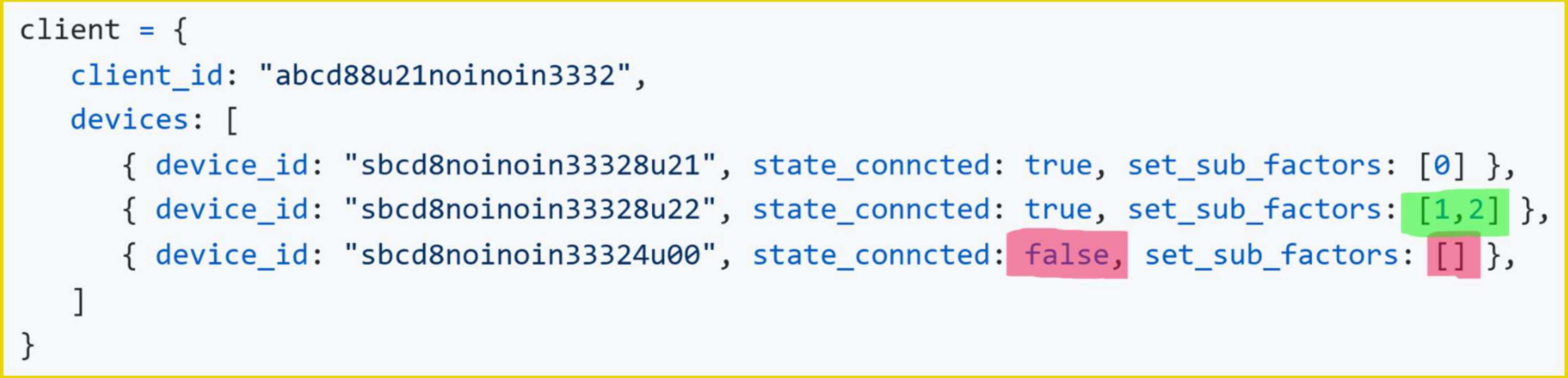
set_sub_factor = **2**

message_index	reciver_id	sender_id	data
2	abcd88u21noinoin3332	abcd88u21noinoin8787	<Buffer...430>
5	abcd88u21noinoin3332	abcd88u21noinoin8787	<Buffer...550>

Client Device Set /Manage Hardware Failure

If a Device of Client's device set goes down

Let's assume 'client_device_3' goes down



```
client = {
   client_id: "abcd88u21noinoin3332",
   devices: [
      { device_id: "sbcd8noinoin33328u21", state_conncted: true, set_sub_factors: [0] },
      { device_id: "sbcd8noinoin33328u22", state_conncted: true, set_sub_factors: [1,2] },
      { device_id: "sbcd8noinoin33324u00", state_conncted: false, set_sub_factors: [] },
   ]
}
```

Client Device Set / Manage Hardware Failure

Client's device 1

set_sub_factor = **0**

message_index	reciver_id	sender_id	data
3	abcd88u21noinoin3332	abcd88u21noinoin8787	<Buffer...100>
6	abcd88u21noinoin3332	abcd88u21noinoin8787	<Buffer...430>

Client's device 2

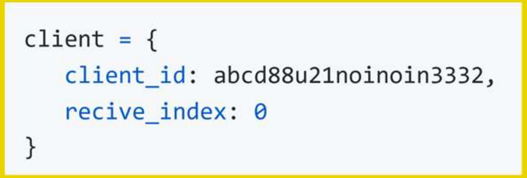
set_sub_factor = **1, 2**

message_index	reciver_id	sender_id	data
1	abcd88u21noinoin3332	abcd88u21noinoin8787	<Buffer...400>
2	abcd88u21noinoin3332	abcd88u21noinoin8787	<Buffer...430>
4	abcd88u21noinoin3332	abcd88u21noinoin8787	<Buffer...230>
5	abcd88u21noinoin3332	abcd88u21noinoin8787	<Buffer...550>

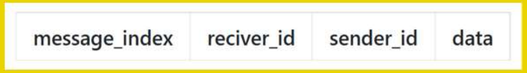
Client's device 3

Message Delivery System/ Sending a Message

Before Sending Message

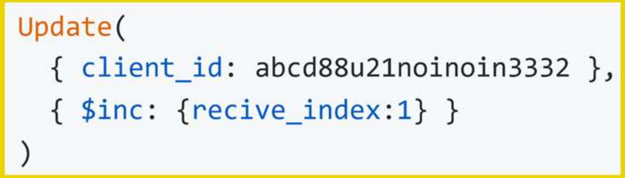


```
client = {
   client_id: abcd88u21noinoin3332,
   recive_index: 0
}
```

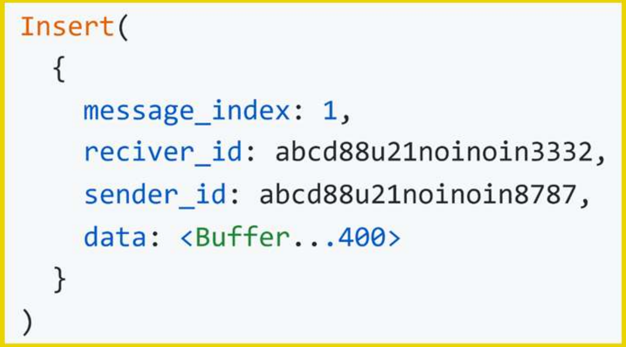


message_index	reciver_id	sender_id	data

While Sending Message

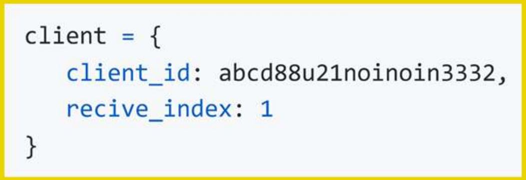


```
Update(
  { client_id: abcd88u21noinoin3332 },
  { $inc: {recive_index:1} }
)
```

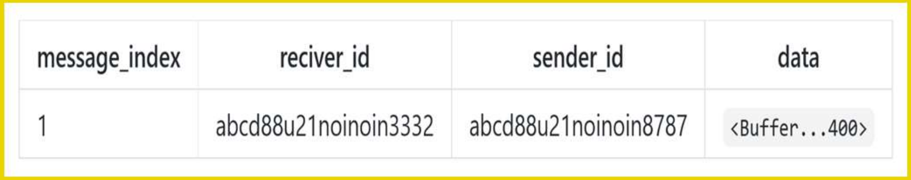


```
Insert(
  {
    message_index: 1,
    reciver_id: abcd88u21noinoin3332,
    sender_id: abcd88u21noinoin8787,
    data: <Buffer...400>
  }
)
```

Message when Added and Stored



```
client = {
   client_id: abcd88u21noinoin3332,
   recive_index: 1
}
```



message_index	reciver_id	sender_id	data
1	abcd88u21noinoin3332	abcd88u21noinoin8787	<Buffer...400>

THANK YOU

PROJECT'S GITHUB SPACE: HTTPS://GITHUB.COM/JOVIALCHAT

By

Rishav Bhowmik

Roll No. :1706349