ISSN:0975-9646

Cloud based Magixphone and Web gaming Platform

Aditi Kore, Komal Lunkad, Pranjali Ingole, Sonia Gupta

Computer Science Department, Pune Institute of Computer Technology
Pune University Dhankwadi, Pune- 411043, Maharashtra, India.

Abstract — This paper describes the development of platform which will support messaging and gaming together. The cloud server will have the capability to hold both the messaging and gaming service. Currently, Google Cloud Messaging (GCM) is used. The messages are pushed to a large number of people who are subscribed to the service in a very less time. The delivery of messages by GCM is very unpredictable. Hence, there is a need of a reliable connection to Google GCM. This platform will include of following functionalities: 1) sending messages from different clients to the server 2) server will display the message on the dashboard 3) motion points will be transferred to the server and mapped 4) server will display the gaming objects on the dashboard.

Keywords — game service platform, user engagement, web messaging

I. Introduction

The design for a cloud based messaging and web gaming platform for customer engagement is proposed. The complete backend architecture can handle above operation and a simple front end operation to demonstrate the system. The Client-Server architecture is used. The server machine will offer two basic services: Real time messaging and Web gaming. The server will be connected to a dashboard where messages and game objects will be displayed. Clients devices can be laptop or smart phones where services can be used via web browsers or applications. For motion gaming, client device must have accelerometer and gyro meter. Server and clients are connected to each other by short ranged wireless network which can be private (not necessarily connected to internet). Client can send messages to server which will be displayed on dashboard in real time. The game data and control points in case of motion games will be transferred to server through wireless network in real time where these points will be mapped to game object displayed on dashboard. Users can control game objects on dashboard using their client devices such as smart phones.

II. Architecture

A. *Design*

Phase 1:

Designing of the backend for the system is to be done in this phase and it is based on any choice of framework. The backend should have following functionalities:

I. It will be able to handle multiple requests coming to the system.
II. Client connects to local server using its cell phone or laptop.
III. Client can subscribe for a service (messaging or gaming).
IV. The system will display the subscribed service onto the screen automatically.
V. The system will be able to handle simultaneous connections to it gracefully.
VI. It will have support of multi-player gaming.
VII. Client can logout gracefully
VIII. It will have a secondary database to store information for analytics of all the connections and game play.

It will work with all smart phones.

Phase 2:

Development of sample front end application will be done in this phase.

I. The sample content can be an Effect/Game/Information dispenser or combinations of any two or all.

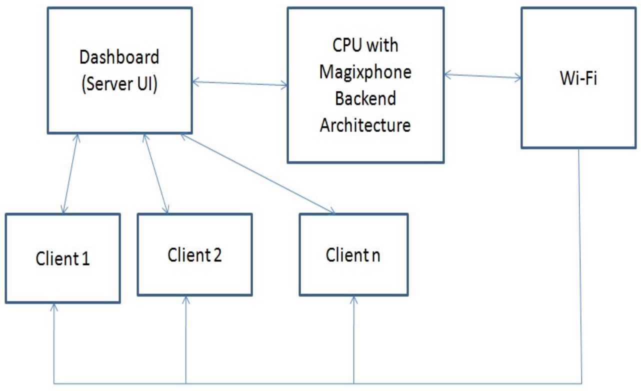

Fig. 1: Low level architecture

B. *Technology Stack*

Node.js: Node.js is a JavaScript runtime built on Chrome's V8 JavaScript engine. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient. As an asynchronous event driven framework, Node.js is designed to build scalable network applications.

Express: Express is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications.

Sails: Sails makes it easy to build custom, enterprise-grade Node.js apps. It is designed to support the requirements of modern apps: data-driven APIs with a scalable, service-oriented architecture. It's especially good for building chat, real time dashboards, or multiplayer games.

Mongodb: MongoDB is an open-source, document database designed for ease of development and scaling.

SocketIO: Socket.IO enables real-time bidirectional event-based communication. It works on every platform, browser or device, focusing equally on reliability and speed.

C. ***High level architecture***

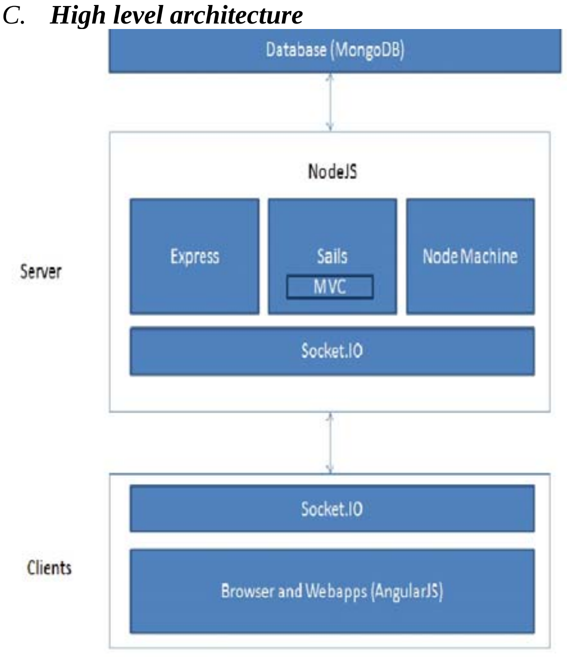

Fig. 2: High Level Architecture

Basically, there are three layers in the architecture diagram which are client-side, server and database (If required for future extension). The database used will be mongodb. This will support the non-structured form of data to be stored. The server side will be developed using the NodeJs and Sails. The client side application will be developed using Socket.IO. The web apps and the browser functionality will be extended using AngularJs.

III. PROCEDURE

1. Server establishment:

Node.js server is started and kept running on server machine. It will handle client connections and requests using SocketIO.

2. Connection and subscription:

Initially, users have to login with the authenticated user id and password. The users have to subscribe to whichever service they want (messaging or gaming).

3. Message handling and displaying:

Many clients will send message to the server, which are to be displayed on dashboard. Server will display those messages on the dashboard according to sequence in which they are received.

The size of the messages on the dashboard will be decided dynamically according to the total number of messages. If the limit of the number of messages on the dashboard exceeds then messages which are being displayed for the longest time will be removed.

4. Accessing control points and mapping:

The game control points are obtained from accelerometer and gyro meter readings of client devices. These points are transferred to server via wireless network in real time. These points are mapped to the game object displayed on dashboard to control motion of that objects.

IV. MATHS

Let S be the System set represented as follows:.

S={s, e, X, Y, NDD, DD, Fme, Sc, Fc}

s: start state ,Wi-Fi Connectivity and Subscription to messaging or service is checked.
e: end state. Logout and Unsubscribe the service.

X= set of inputs where, X={x1, x2}
x1= message text
x2= gaming controls such as accelerometer and gyro meter supported by IMO Controller.

Y= {Y1, Y2}
where,
Y1= Messages are displayed on the dashboard or on client devices.
Y2= The controller movements are mapped from the mobile device to the dash-board and accordingly the object on dashboard is moved.

Fme: set of main functions
Fme= {Fsubscribe, Finputmessage, Fmap, Fsendmessage, Fcreateobj, Fmoveobj, Fstorecloud, Funsubscribe} where,
Fsubscribe: Function to subscribe to the service i.e. either messaging or gaming.
Finputmessage: Function to type message
Fsendmessage: Function to send message
Fmap: Function to map movements from client device to server.
Fcreateobj: Function to create object which will move.
Fmoveobj: Function to move objects on the display board
Fstorecloud: Function to store services on cloud.
Funsubscribe: Function to unsubscribe to the service, either messaging or gaming.
DD: Deterministic Data
DD= Input Set
DD: Non-Deterministic Data
DD= Wi-Fi is turn on, the devices are not in the given wifi zone.
Sc: Success Case
Sc= Successful delivery of messages and mapping the movements from mobile devices to dash boards.
Fc: Failure Case
Fc= If the message is not delivered properly or congestion in the network.

V. Conclusions

The above platform can be used for development of various messaging applications and motion games where smart phone or any client device can turn into a remote control for game. The platform can have use cases like Shopping malls, Quizzes, Restaurants, Coffee shops, Auction places, Social gatherings where people will interact with each other for short time. Real time matches and tournaments also can make use of it.

Acknowledgment

We are thankful to our internal guides Mr B. D. Zope and Mrs Wakode for their helpful guidance .We are also thankful to our external mentor Mr Akshay Dekhate for his technical guidance.

References

[1] Miranda Zhang, Rajiv Ranjan, Michael Menzel, Surya Nepal, Peter Strazdins, S. Chen, "An Infrastructure Service recommendation System for Cloud Applications with Real-time QoS Requirement Constraints," IEEE SYSTEMS JOURNAL, April 11, 2015.

[2] Yavuz Selim Yilmaz, Bahadir Ismail Aydin "Google Cloud Messaging (GCM): An Evaluation," 978-1-4799-3512 IEEE, March 14, 2014.

[3] Shaoxuan Wang, Sujit Dey , "Modeling and Characterizing User Experience in a Cloud Server Based Mobile Gaming Approach," 978-1-4244-4148 IEEE, September 8, 2009.

[4] Weiqing Zhao, " Improving Computer Basis Teaching Through Mobile Communication and Cloud Computing Technology," 978-1-4244-6542 IEEE, February, 2010.