2017 International Conference on Emerging Trends & Innovation in ICT (ICEI)
Pune Institute of Computer Technology, Pune, India, Feb 3-5, 2017

Scalable Tracking System for Public Buses using IoT Technologies

Jay Lohokare[1], Reshul Dani[2*], SumedhSontakke[3^],Asst. Prof. Rahul Adhao[4*]*
[*]Department of Computer and IT, [^]Department of Electrical Engineering
College of Engineering, Pune
[1]lohokarejs13.comp@coep.ac.in, [2]reshulsd13.comp@coep.ac.in, [3]sontakkesa15.elec@coep.ac.in, [4]rba.comp@coep.ac.in

***Abstract*— Reliability in public transport is of great importance today. Millions of people travelling by public buses waste a lot of time waiting at bus stops. This paper focuses on presenting a solution to tackle the said problem by harnessing IoT technology stack. If the people travelling get accurate real time location of the buses along with estimate time for arrival at bus stop based on the real time traffic conditions, it will facilitate an overall increase in reliability on the public buses. The solution proposed in this paper involves using the existing internet enabled devices on the bus (like the e-ticketing system) or a simple android tablet to capture the real time location and send to the servers. Accessing this location data from servers will be facilitated by Representational State Transfer (REST) APIs which users can access through android application, SMS or web-portals. The system proposed will have distributed architecture in order to tackle high number of requests from users.Although there are existing solutions which harness the use of Global Positioning System (GPS) for bus tracking, they aren't ready to handle high demand on the backend which will exist in the near future. We have addressed this problem. The primary contribution of this paper is that it shows that a backend based on Message Queue Telemetry Transport (MQTT) instead of the traditionally used Hypertext transfer protocol (HTTP) based REST will be light weight, data efficient and scalable. We have proposed and implemented the backend as well as the front end required for the tracking system and presented the improvements.**

Keywords*— *Internet of Things, Smart transport, Location based services, Android, MQTT, HTTP, GPS, smart phones, public transport, bus tracking

I. INTRODUCTION

Transportation is a major pain area for cities today. With the ever increasing load on public transport systems, it is really necessary to increase efficiency in these systems. Due to extreme traffic conditions, over-crowding and many other similar issues, public buses lack punctuality and reliability. There is a dire need to tackle this issue. The required solution should not only facilitate improvement in the services, but should also be a driving factor for increase in trust on the public bus transport systems.

Reliability in public transport will be facilitated when the traveller accurately knows when a bus will arrive to the bus stop or when will the bus reach the destination.The solution proposed in this paper harnesses the real time location of the buses to calculate the estimated time for reaching a particular position. By saving location data on the server along with corresponding timestamps, we can estimate the time for the bus to arrive as a bus stop, or time to reach a destination, using services like Google maps [1].

This paper presents a simple and cost effective solution to make public transportation services 'smart'. The paper will present the concept, technology stack, components and the outcomes of implementing the solution. The primary goal of the proposed solution is to minimize the costs involved in implementation and to create a backend that can scale up easily with increase in demand.

The location data collected from the buses should be accessible by users as well as developers to harness the collected data to create more value from collected data.

II. EXISTING SOLUTIONS

Typically, bus tracking systems show data related to the arriving buses, time to arrival and estimated departure time based on location of bus that is obtained via Radio Frequency (RF) transceiver [2]. There are such transceivers on every bus and bus stops so that the buses communicate the location to bus stops. The microprocessors at bus stop then calculate estimated time for arrival and display it on a screen.

SMS services on GSM are used in some solutions as means for communication [3]. There are GSM modules with Global Positioning System (GPS) on the bus. These periodically send the location to databases through SMS. The location data can then be accessed by sending SMS to receivers on the database servers.

Location data can be obtained by GPS based tracking devices mounted on vehicles. These systems useHyper Text Transfer Protocol (HTTP) to send location data to databases [4]. Android devices in the buses stream location data to servers, which can be accessed through android applications or web-portals.

However, none of these systems can handle an increase in requests. The data consumption for sending the data from buses to servers is high. The backend will crash if there is an increase in the number of users using the service. Therefore there is a need to explore a solution that will be scalable and efficient.

978-1-5090-3404-8/17/$31.00 ©2017 IEEE

III. Overview of Smart Tracking System With Scalable Backend

The population is constantly rising today throughout the world. The number of people that rely on public transport will increase multi-fold in the years to come. With rise in population, there will be increase in the load on the public transport systems which will require increase in number of buses. The increase in the number of buses will result to a tremendous increase in the amount of data generated real time. The location data of these thousands of buses will need to be handled by the backend reliably. There indeed is a need for a scalable backend for systems. The traditionally used HTTP protocol needs to be replaced by a better and light weight protocol. Using Message Queue Telemetry Transport (MQTT) will not only result inlessinternet bandwidth consumption [5, 6] compared to other protocols, but will also increase the flexibility of the solution. MQTT has been proven to require minimum bandwidth even in a constrained network[7]. Use of MQTT protocol will enable the use of a distributed backend. With broker clustering in MQTT, the system will be highly available, fault tolerant and scalable.

The location data collected from buses can be used to build data analytics applications. There can be a lot of useful insights related to traffic conditions based on this data. Hence, it is important to make this data available to developers via APIs to encourage its use for creating applications for the benefit of society.

Fig. 1 shows the model of the proposed solution. The solution is divided into three parts. The first part is the collection of location and related data from buses. The second part is handling this data on a distributed IoT backend. The third part is making this data available to users through various platforms.

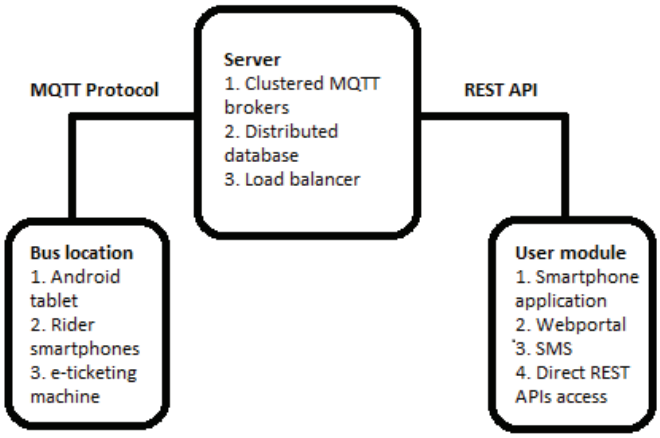

Fig. 1. Overview of the proposed solution

IV. Collection of Data From Bus

Collecting real time location data is the major issue to tackle due to involvement of hardware which will need heavy investment. The traditional solution is installing GPSand General Packet Radio Service (GPRS) enabled devices in every bus. These devices though costly, serve as one time investment to provide services of importance.

The following are the cheaper alternatives available:

A. *E-ticketing systems*

Today, most buses in cities like Pune, India already have the necessary hardware proving the means for real time location tracking. Having an e-ticketing systemis the solution to prevent the installation of new costly hardware. These systems are GPRS enabled. By network triangulation, it is possible to get real time location of the devices. Thus, with location available and internet access, they can be used to share location data with users with minor changes.

B. *Android tablet phones*

There are many cheap android devices available in market with internet and GPS. These tablets can double as infotainment systems in the bus. They can display basic information like route number, destination, next bus stop, time and also information related to weather, nearby tourist spots. The public transport agencies can show advertisements on the tablets.

C. *Crowd sourcing location data*

This method involves no GPS/GPRS devices installed in the bus. Most of the people have smartphones today. Smart phones of people travelling by a particular bus can act as source of location data for that bus. Thus, the data can be ‘crowd sourced’.

V. IoT Backend Architecture

The data collected from a source needs to be sent to servers efficiently and reliably. It should be stored in a way such that users can query it and should be available even when there is a heavy load on the servers due to huge number of people querying it. Fig. 2 shows the proposed architecture.

The solution involvesa light weight protocol to send the collected data to databases. There are many protocols that can be used to achieve this. However, MQTT will best serve this use-case due to its low bandwidth usage and support scalability.MQTT backend can be easily scaled up by clustering of its brokers. Once the server gets the data, it will store it in NoSQL database like MongoDB [8], which will be distributed and have multiple instances to enable load balancing. Users can then query bus locations from these databases.

In MQTT protocol, there is a broker(server) that contains ‘topics’.Topics are the way to determine who receives the data generated by a sender. A receiver has to ‘subscribe’ to a topic to receive data ‘published’ with that topic by any sender [9]. The location data published over MQTT from the buses will have the route number as topic. Location data comprising of the current latitude and longitude of a bus will be published at a frequency of 5 seconds and will still consume less bandwidth because of the use of MQTT. To make the system scalable, the MQTT brokers

will be clustered as shown in Fig. 3. The clustered MQTT brokers act together as one broker. Every broker in the cluster will maintain the topics list. When one broker is unavailable, another broker handles the request. This way, the brokers balance within themselves. This clustering ensures that the system is highly available. Even if one server goes down, other servers will be there in its place to handle the requests and data.

The server will receive the location data through MQTT and will store it in a database collection for that particular route number. This can be done by directly modifying the MQTT broker to write to database when data is received or by creating a MQTT client to do so.

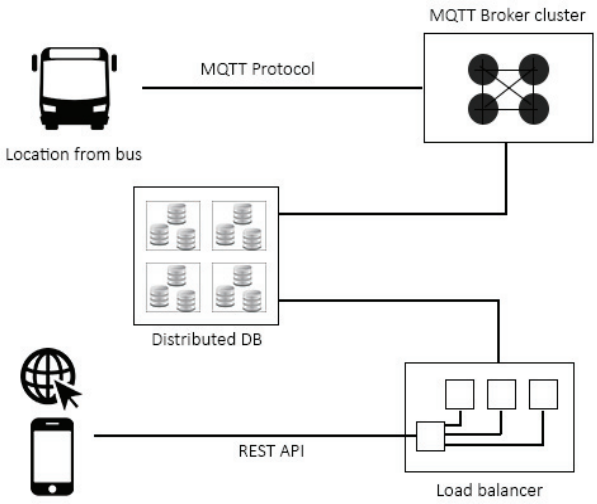

Fig. 2. Architecture Components

The documents in the database will contain location of the bus and timestamp along with route number and bus registration number. Bus registration number is an example for unique ID a bus can have. At the same time, the same data will also be put into a central database maintaining locations of all buses

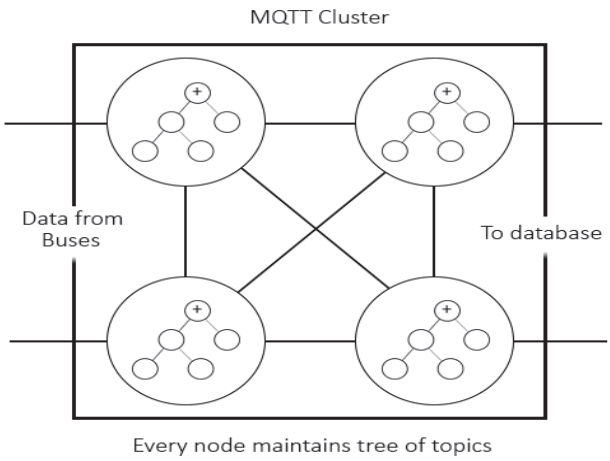

Fig. 3. Cluster of MQTT brokers

Example document in the database:

{

“Latitude”: 17,

“Longitude”: 23,

“bus_registration_number”: “MH04A1234”,

“bus_route_number”: 100;

“direction”: 0,

“timestamp”: 211005102016

}

Here, direction indicates which way the bus is going for the particular route (A to B or B to A).

Consider a bus with registration number ‘MH 12A3023’ servicing route ‘100’, the location data received from the bus will be stored in NoSQL collection named ‘100’ as well as a central collection ‘Buses’. Thus, when a user wants to know what the live locations of buses on route 100 are, the solution will search in collection named ‘100’. When the user queries the location of a particular bus, the collection named ‘Buses’ will be searched.Fig. 4 shows the collections in which the location information of two buses on route number 100 and 200 will be stored.

The e-ticketing machines or android tablets in the bus will publish data over MQTT with the route number as the topic. In most cases, these machines or tablets remain same for a particular bus. So they will have the bus registration number pre-fed in

them. In case of streaming location data from smartphone of bus riders, there will be an application where the user can set the route number and bus registration number to start streaming location of that bus to server. To remove the process of manually entering the numbers, there can be a NFC tag or Bluetooth beacon in the bus [10]. Scanning the tag will automatically start streaming the location data with correct credentials.

The high number of user queries can be handled by distributing the NoSQL database. For example, NoSQL collection for bus route '100' and bus route '200' will be on different servers 'A' and 'B'

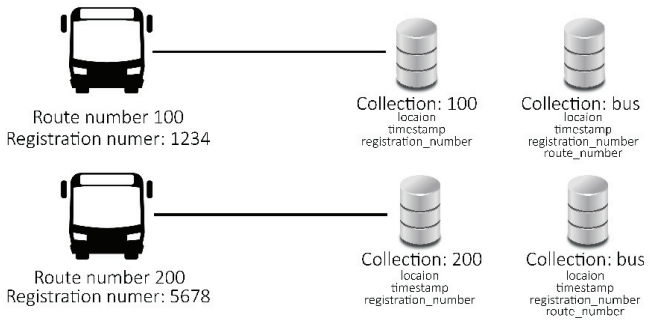

Fig. 4. Database collections

Thus, queries for bus route 100 will be directed to server A and that for route 200 to server B thereby distributing loads.Also, maintaining instances of same database on multiple servers will help in managing the load. For example, there will be multiple instances of database collection A on multiple servers. If there are a high number of requests for route 100, the load balancer will automatically route the requests to the instance of collection A with minimum load. Consider 3 servers maintaining collection A. Let S1, S2, S3 be the 3 servers. Load on S1 is 'x', on S2 is 1.5x, on S3 is 1.2x. When a request comes for collection A, the load balancer will automatically route it to the server with least load, in this case, S1.

When MQTT writes to the databases, the data will be modified on all the instances to make sure that all instances have latest data.

The load distribution will be achieved using tools like HAProxy[11]. The requests from a user will come to an intermediate server which will then route it to the appropriate server based on the request query.

VI. Data Access Points for Users

The users should be able to access location data of the buses real time through various platforms like web, smartphones or SMS. The proposed solution will make this possible by means of REST APIs. The user will send HTTP requests to the servers which will then fetch required data based on context of the user.

The users can access the data through the user module (web-portal, SMS or smartphone application). This location data will be shown to the users using Google maps or equivalent map services. The user module will allow the user to search a bus by bus registration number directly. However if registration number isn't known, the user module will first ask to select the nearest boarding point (bus stop) to the user. This process can be automated by maintaining a database of bus stops along with their corresponding location data. The user's location will be compared with locations of the bus stops and distance will be calculated by using the Haversine formula.

La1: User latitude

Lo1: User longitude

La2: Bus stop latitude

Lo2: Bus stop longitude

Dlon: Lo2 – Lo1

Dlat: La2 – La1

R: Radius of earth

$$A = (\sin(Dlat/2))^2+\cos(La1)*\cos(La2)*(\sin(DLon/2))^2$$

$$C = 2 * \tan^{-1}(\text{sqrt}(A),\text{sqrt}(1-A))$$

$$\text{Distance} = R * C \qquad (1)$$

Using (1), the bus stop with shortest distancewill be the nearest bus stop.The user module will then ask the bus route number to be selected or the destination bus stop.

Users can then query live location of a bus based on the route number, target destination or bus registration number. For registration number entered, the user API will return the last known location of the bus from the central buses collection.

When the user selects the destination bus stop, the mapping of boarding point – destination point to route number can be done by maintaining a simple database of all bus stops on all routes, as shown in Table I. So, after selecting boarding point and destination, the user will get live location of all buses on routes taking to the destination. The 1st row of Table I indicates the route numbers. The corresponding columns contain the list of bus stops on that route. When the user queries data by boarding and destination bus stops, the data is checked in the table. Consider 'A' and 'B' are the boarding and destination bus stops respectively. The query checks the table to see if 'A' occurs in the table and is followed by 'B'. Bus Stop 'B' need not immediately follow 'A'. Routes 100, 200, 300 have such condition true. The query will then fetch the live locations of buses on these three routes and return them to the user.

The queries from the user module will be same across all platforms (Smartphone or web). The SMS service deployment will be by setting a server to receive SMS queries and sending the results back as SMS.

The HTTP GET requests can also be made by developers to get access to the data. The output for the requests will be following format:

A. *Query by registration number*

Will return JSON response fetching data from 'buses' collection with following parameters:

1. Latitude
2. Longitude
3. Bus route number
4. Bus registration number
5. Time stamp
6. Direction

B. *Query by route number*

This query will return a JSON array containing objects. Each object will give details of active buses running on that route. The data fetched will be from collection with the route number as its name.

TABLE I. BUS STOPS ON EACH ROUTE

100	200	300	400	500
A	A	Z	A	Q
B	B	A	S	A
C	H	B	D	C
D	L	G	F	S
E	R	H	G	D

C. *Query by boarding point and destination*

This query will return a JSON array containing objects for active buses on every route taking from the boarding point to the destination. This data will be fetched from various collections based on which routes are applicable.

VII. PREDICTION OF ARRIVAL TIME AND TIME TO REACH DESTINATION

In order to predict accurately the arrival time of bus at the bus-stop and time to reach destination, it is necessary to consider real time traffic. Historical data can be useful to predict arrival time [12]. Using artificial neural networks (ANN) on historical data is one way of estimating the required time [13]. However, the ANN would not consider real time traffic. Hence, it will be appropriate to use Map services like that of Google to provide estimated arrival time. Google Maps considers real time traffic while predicting time for travel. The time for arrival thus can be calculated accurately by giving the live location of bus and location of bus stop to google maps.

To handle the incorrect data received from buses, radial bias function neural networks can be used. These will be able to recognize faulty location data and neglect delete it from the databases in order to avoid incorrect information retrieval [14].

VIII. RESULTS

The solution proposed will enable users to easily fetch the live locations of the buses and thus know the estimated time for arrival and also the time to reach the destination. By knowing this time, users will benefit as they can now plan their travel very accurately. The public bus service will become more efficient and reliable due to availability of real time data. The users can also access this service from a website or through SMS.

Fig. 5 and Fig. 6are the screenshots of the android application. Fig. 5 shows users being able to get location of bus based on route number. Fig. 6 shows users getting bus location data based on destination bus stop. The application will automatically select nearest bus stop as boarding point (Shivagingar is selected as boarding point in the above screenshots).

The solution constructed hasErlang MQTT Broker(eMQTTD) as MQTT broker. eMQTT brokers support clustering and are scalable, with every single node in the cluster supporting up-to 1.3 million concurrent MQTT connections[15]. When there is increase in number of users, there is just a need to add more brokers to this cluster. The clustered nodes in eMQTTD cluster together act as one MQTT broker and distribute the load within them. eMQTT has modules for connecting to MongoDB. Thus, the solution created is highly scalable and can easily sustain increase in load on servers.

IX. SCREENSHOTS

Fig. 5. Search bus by route number

Fig. 6. Search bus by destination

X. Conclusion

We have proposed and implemented a new way of tracking public buses by using MQTT instead of traditional HTTP. The proposed solution is scalable and future ready. The use of proposed architecture allows horizontal scalability for the solution. Use of MQTT will result to less consumption of bandwidth and on demand access to location data of the buses. Use of MQTT makes the whole solution data efficient and allows light weight data transfer. The distributed backend to handle increase in number of users makes it scalable. This systemwill help in making the public transport systems reliable and efficient.

Acknowledgment

We implemented and demonstrated the solution proposed by us at the Digital Pune Hackathon,October 12, 2015 conducted by Persistent Systems and won the second runner up prize. We received incubation support and encouragement to further refine the solution to make it market ready so as to be implemented in Pune city in collaboration with Pune Mahanagar Parivahan Mandal Ltd (PMPML). We sincerely thank the CEO of PMPML, Ms. Mayura Shindekar for her valuable inputs about the working of the e-ticketing systems and the problems regarding public transport in Pune city. We thank her for the encouragement provided to develop the solution so that it can be deployed in the real world.

References

[1] Google maps,https://developers.google.com/maps/

[2] Chheda Gaurav, Gajra Niket, Chhaya Manal, Deshpande Jitesh, GhargeSaylee, “Real Time Bus monitoring and Passenger Information System”, in“International Journal of Soft Computing and Engineering (IJSCE)”, ISSN:2231-2307, Volume-1, Issue-6, January 2012 34

[3] Maruthi R.,Jayakumari C., “SMS based Bus Tracking System using OpenSource Technologies”, in International Journal of Computer Applications (0975 – 8887) “, Volume 86 – No 9, January 2014

[4] Salim A., Idrees Ibrahim , “Design and Implementation of Web-Based GPS-GPRS Vehicle Tracking System”, in International Journal of Computer Science and Information Technologies , Vol 3, Issue 12, 443-448, December 2013.

[5] Vasileios Karagiannis, Periklis Chatzimisios, Francisco Vazquez-Gallego and Jesus Alonso-Zarate, “A Survey on Application Layer Protocols for the Internet of Things”, Transaction on IoT and Cloud computing, 2015

[6] Z. Wei, Y. Song, H. Liu, Y. Sheng and X. Wang, "The research and implementation of GPS intelligent transmission strategy based on on-board Android smartphones," *Computer Science and Network Technology (ICCSNT), 2013 3rd International Conference on*, Dalian, 2013, pp. 1230-1233

[7] Y. Chen and T. Kunz, "Performance evaluation of IoT protocols under a constrained wireless access network," *2016 International Conference on Selected Topics in Mobile & Wireless Networking (MoWNeT)*, Cairo, 2016, pp. 1-7

[8] MongoDB,https://docs.mongodb.com/

[9] MQTT , http://mqtt.org/documentation, mqtt.org

[10] K. Tanaka and K. Naito, "Demo: Implementation of unconscious bus location sensing system with smartphone devices and beacon devices," *2016 13th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, Las Vegas, NV, 2016, pp. 280-281.

[11] HAProxy, http://www.haproxy.org/

[12] J. Gong, M. Liu and S. Zhang, "Hybrid dynamic prediction model of bus arrival time based on weighted of historical and real-time GPS data," *2013 25th Chinese Control and Decision Conference (CCDC)*, Guiyang, 2013, pp. 972-976

[13] L. Singla and P. Bhatia, "GPS based bus tracking system," *Computer, Communication and Control (IC4), 2015 International Conference on*, Indore, 2015, pp. 1-6.

[14] Ahmed N.Abdalla, Azher Fakharuddin, Muhammad Rauf and NikM.Kamal ,“Design of Intelligent GPS Navigation System for Bus Monitoring and Station Reporting”,National Conference on Postgraduate Research, 2009.

[15] EMQTT, http://emqtt.io/docs/v2/index.html