Tracking System of Multiple Mobile Devices & Sensors using MQTT

Beena M. Patel[1]Bhavisha R. Suthar[2]
[1,2]Department of Computer Engineering
[1,2]U. V. Patel College of Engineering, Ganpat University, India

Abstract*—Dreaming on the tremendous potential in developing smarter mobile devices has lead the world towards the golden era of Android Operating System and approximately over two billion users have been witness during the last decade. Throughout this duration, such mobile technology has played vital role to facilitate the internet and other respective benefits to reach each and every individual throughout the globe. As the result, the use of Internet and hence consumption of data has been increased drastically and demanding more and more day by day. No matter whatever we will develop in Mobile Broadband Internet, we will be always in short of supply. The future is demanding to initiate the precise as well concise solution of network traffic. Also as per current demand, tracking of devices and people are also a future requirement i.e if small kids studying in long distance schools in metro city then for parents it is require that their kids are away in school but within a safe circle and they (kids) are as per their device location. So if Message Queuing Telemetry Transport moreover known as MQTT would be the eventual solution in the same direction to handle conventionally used Hypertext Transfer Protocol based request/response using client-server computing model. This lightweight messaging protocol can be used to establish connectivity and data sharing between multiple clients as well as between client and server with low internet bandwidth.*

Keywords—Android, MQTT, HTTP, GPS, Fuse Location, Proximity Sensor, Gyroscope Sensor

I. INTRODUCTION

In a recent year, the demand for capturing real data continuously is an essential need for each field. Every field/client wants Real data updation to plan their field outcomes in a significant way. In current scenario, these needs are fulfilling through client/server architecture or say through HTTP. We are using Internet of things for accessing data when we want to collect as per our need. We are requesting to the server and server responds as request given through HTTP. The connected devices need a protocol using which they could communicate only when it is required. In the client-server computing model the function of HTTP protocol is to request and response. HTTP request send by clients to the server. On behalf of client, the server provides resources such as HTML files, other content or performs other functions and returns a response message to the client [1].

In this solution more data required more sensors input require for gathering the real time updating. So, eachsensor continuously sending request to server and at the end server loaded and stuck up due to heavy network traffic. With the ever-increasing load of network traffic, these systems efficiency necessarily should be increase. The proposed solution, utilize the real time location/position of devices (Mobiles) and also, we can achieve the raw data generated from the device sensors. To overcome this issue MQTT is a best solution. Parallelly Internet is growing and gaining popularity, but new protocols are being introduced to make development and implementation easier. The goal of this was to have a bandwidth-efficient protocol that uses as little battery power as possible. Instead of the typical HTTP request/response paradigm MQTT protocol uses a publish/subscribe architecture. Publish/Subscribe is an event driven design that enables messages to be sent to clients, with the focal correspondence point being the MQTT broker which incorporates a theme into the message. Every client that needs to get messages subscribes to a specific theme, and after that the broker will convey every one of the messages with a coordinating point to the client. With this stated, the client do not need to know one another, they just need to discuss over the point. This procedure and architecture allows for incredibly extensible arrangements with the standard conditions between data, customers, and procedures.

This is as opposed to HTTP where a client will pull the information it needs. Instead, the broker send the information to the customer when that is slightly new and has a for all time open TCP association. If this event is intruded on, the broker will buffer the messages and sit tight for a steady connection to send them to the client.

MQTT [18] is an ISO standard [7] which is publish-subscribe-based messaging protocol and work on top of TCP/IP protocol. It is designed for connections with remote locations. There is a "small code footprint" required or the network bandwidth is limited. Publish-subscribe messaging pattern require a message broker [17].

MQTT is intended for obliged gadgets and high latency, lower bandwidth or unreliable network. It was outlined as a great degree of lightweight publish/subscribe messaging transport. For instance, it has been utilized as a part of sensors imparting to a dealer by means of a satellite connection, over intermittent dial-up connections with healthcare providers, and in a scope of home automationand little gadget situations [6].

In software architecture, the messaging pattern is publish-subscribe where the messages send by senders called publishers and messages receive by the receivers called subscribers but they do not program the messages. Instead of classify the published messages into specific classes without knowledge of any subscribers. Likewise, subscribers state their interest in one or more classes but receive messages only from area of interest, without knowledge of any publishers. Publish–subscribe is a sibling of the message line concept and is usually one part of a larger message-oriented middleware system. Generally messaging system supports both one publish and subscribe. Then message queue models in their API, like Java Message Service. This pattern provides large network scalability and more dynamic network topology [7].

Multiple sensors provide the signals to mobile Apps in the devices to find out device location. Though, it is not simple to choose the true combination of signals for a

specific task in specific conditions. Even more complicated is to find a solution about battery efficiency. In Google Play services the fused location provider is a location API that is smartly combines different signals to provide the location detail which is your app needs. The underlying location technologies like GPS, Wi-Fi managed by the fused location provider and provides a simple API that people can use to specify the required quality of service. For instance, people request the most precise data available, or the best accuracy possible with no extra power consumption [8].

Mainly Android powered devices have built-in sensors which measure movement, direction, and different environmental positions. The sensor provides raw data with high accuracy. Those data are useful to people when they want to observe three-dimensional device movement or its position, or people want to observe changes in the ambient environment near a device. For instance, from a device's gravity sensor a game might track readings to infer complex user gesture and movements, such as tilt, shake, rotary motion, or roll. Motion sensors measure acceleration forces and rotational forces along three axes. This group includes accelerometers, gravity sensors, gyroscopes, and rotational vector sensors [9].

II. EXISTING SOLUTIONS

The Fused area data would then be able to be gotten to by sending SMS to receivers on the database servers. Area data can be acquired by GPS based tracking Android gadgets. These frameworks utilize Hypertext Transfer Protocol (HTTP) to send area data to databases. Android gadgets send area data to servers that can be gotten to through online interfaces. In any case, none of these frameworks can deal with an expansion in demands. The data utilization for sending the data from gadgets to servers is high. The backend will crash if there is an expansion in the quantity of clients utilizing the service. In this way, there is a need to investigate solution that will be scalable and efficient [4].

HTT5P/1.0 relates acutely with TCP. It acquires frequent round-trip delays because of connection establishment, and also executes slow start in both directions for little period of connections, and incurs heavy latency penalties due to the mismatch of the classic access profiles with the particular request per transaction model. HTTP/1.0 also requests busy servers to dedicate resources to keep TIME_WAIT detail for large amount of closed connections [11].

If the client's requests are more than one for different network bandwidth then it special effects on the act of HTTP protocol. So, with number of node traffic the delay will increase and as per requests increase. Moreover, it depends on increase in the number of client nodes, an increasing of an end-to-end queuing delay of the client node to the server. When the link bandwidth is increase, the decreasing of the end-to-end queuing delay of the client node to the server. When the number of the clients are increasing from ten to 20, then the end-to-end queuing delay increased by a big margin, but when the number of the clients are increasing from 20 to 30, the end-to-end queuing delay increasing relatively less [12].

III. BACKGROUND

A. MQTT Publish Subscribe Architecture

MQTT is intended to overcome the difficulties of connecting the quickly extending physical world of telephones, tablets, sensors, and actuators with built up software processing technologies. These standards additionally end up making this protocol perfect for the rising M2M or IoT world of connected gadgets where bandwidth and battery control are at a premium. The MQTT messages are distributed asynchronously ("push") via publish subscribe architecture. The MQTT protocol exchange a series of MQTT control packets in a well-defined way. Each and every control packet has particular purpose and every bit in the packet is sensibly made to decrease the data transmitted on the network. A MQTT topology has a MQTT client and server and both are communicate through different control packets [13].

As shown in Fig. 1, the MQTT Protocol is base on top of TCP/IP and both client and broker need to have a TCP/IP stack.

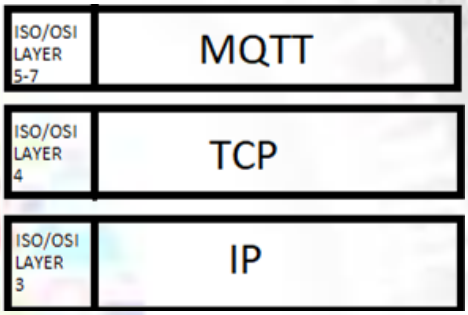

Fig.1: MQTT connection [13]

B. Ideal for Constrained Networks (Low Bandwidth, High Latency, Data Limits, & Fragile Connections)

MQTT control packet headers are reserved as minor as possible. It is having three parts such as variable header, a fixed header and payload. Every MQTT control packet has a 2-byte fixed header. It is not necessary that all the control packet has the variable headers and payload. A variable header consist of the packet identifier if used by the control packet. A payload up to 256 MB could be attached in the packets. This protocol having a small header overhead makes it appropriate for IoT by decreasing the amount of data transmitted over conditioned networks [13].

C. Quality of Service (QoS) for MQTT

Quality of service (QoS) levels find out how each MQTT message is delivered and necessarily be stated that for every message sent through MQTT. It is essential to select the proper QoS value for each messages, because this value determines how the messages are delivered by communication between client and server. Using MQTT three QoS for message delivery could be achieved:

1) QoS 0 (At most once)

According to the best efforts of the operating environment the messages are delivered. In this delivery message loss can be happen.

2) QoS 1 (At least once)

Where the messages are must be arrive but duplicate messages can also be received.

3) QoS 2 (Exactly once)

- Where the messages must be arrived exactly once.
- When considering performance impact of QoS there is a simple rule. It is "The higher the QoS, the lower the performance". MQTT affords flexibility to the IoT devices, to choose appropriate QoS they would need for their functional and environment requirements [13].

IV. IMPLEMENTATION

A. Connecting with MQTT

```
mqttAndroidClient = new mqttAndroidClient(mcontext, "tcp://broker.hivemq.com:1883",clientId);
mqttAndroidClient.connect();
```

B. Subscribe to Particular Topic

```
try
{
mqttAndroidClient.subscribe(mqttSubscribe Topic, qos);
}
catch(Exception e)
{
e.printStackTrace();
}
```

C. Connect Google Client for Location Service

```
mGoogleApiclient = new GoogleApiClient.Builder(this).addApi(LocationServices.API).build();
mGoogleApiClient.connect();
```

D. Register Location Updates

```
mLocationRequest = LocationRequest.create();
mLocationRequest.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);
mLocationRequest.setInterval(5000);
LocationServices.FusedLocationApi.requestLocationUpdates(mGoogleApiClient, mLocationRequest, listener);
registerRequestUpdate(listener);
```

E. Callback from Location Updates & MQTT Message Triggering

```
Gsongson = new Gson();
String payload="";|
LocationDatalocationData = new LocationData();
locationData.setmLatitude(getFusedLatitude());
locationData.setmLongitude(getFosedLongitude());
String currentDate = getDateCurrentTimeZone(System.currentTimeMillis());|
locationData.setCurrentDate(currentDate);
payload=gson.toJson(locationData);
mqttHandler.publishMQTTMessage(payload);
```

F. Gyroscope Sensor Callback

```
gyroSensorManager = (SensorManager)getSystemService(SENSOR_SERVICE);
gyroscopeSensor = gyroSensorManager.getDefaultSensor(Sensor.TYPE_GYROSCOPE);
//Create a listener
gyroscopeSensorListener = new SensorEventListener()
{
@Override
public void onSensorChanged(SensorEventsensorEvent)
{
if(sensorEvent.value[2]>0.5f)
{
//this mean device is rotating
//in anticlockwise direction
}
else if(sensorEvent.values[2]<0.5f)
{
//this mean device is rotating
//in clockwise direction
}
}
};
```

G. Proximity Sensor Callback

```
proxySensorManager = (SensorManager)getSystemService(SENSOR_SERVICE);
proximitySensor = gyroSensorManager.getDefaultSensor(Sensor.TYPE_PROXIMITY);
proximitySenesorListener = new SensorEventListener()
{
@override
public void onSensorChanged(SensorEventsensorEvent)
{
if(sensorEvent.values[0]<proximitySensor.getMaximumRange())
{
//Detected something nearby
}
else
{
//Nothing is nearby
}
}
};
//Register it, specifying polling interval in microseconds,
proxySensorManager.registerListener(proximitySensorListener, proximitySensor, 2*1000*1000);
```

H. Publish Message using MQTT

```
Byte[] encodedPayload = new byte[0];
try
{
encodedPayload = payload.getBytes("UTF=8");
MqttMessage message=new MqttMessage(encodedPayload);
mqttAndroidClient.publish(mqttSubscribeTopic, message);
}
catch(Exception e)
{
e.printStackTrace();
}
```

I. GPS Location Data Publishing &Receiving

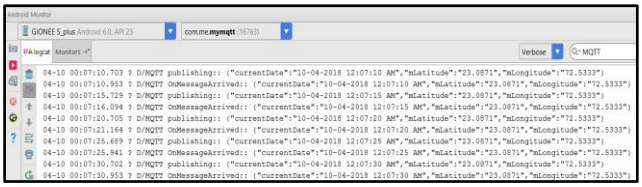

Fig.2: Tracking Results

V. CONCLUSIONS

In this paper, we have proposed how to minimise server loading by using MQTT instead of HTTP. The proposed solution is scalable and will provide less consumption of bandwidth and on demand access to location data. This solution allows light weight data transfer.

REFERENCES

[1] https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol#Technical_overview.
[2] Nut Aroon,"Study of using MQTT cloud platform for Remotely Control robot and GPS tracking" in Electrical Engineering/Electronics, Computer, Telecommunications and information technology (ECTI-CON), 13th International conference on IEEE, 2016, pp. 1-6.
[3] Banks, A. & Gupta, R. MQTT Version 3.1. 1. OASIS standard, 2014.
[4] Jay Lohokare, Reshul Deni, SumedhSontakke, Asst. Prof. Rahul Adhao, " Scalable Tracking System for public Buses using IoT Technologies" in International conference on emerging trends & innovation in ICT (ICEI),IEEE conference on IEEE,2017, pp. 104-109.
[5] Lampkin, V., Leong, W. T., Olivera, L., Rawat, S., Subrahmanyam, N., Xiang, R. & Locke, D., Building smarter planet solutions with mqtt and ibmwebspheremq telemetry. IBM Redbooks, 2012.
[6] https://www.dialogic.com/glossary/mqtt-protocol.
[7] https://en.wikipedia.org/wiki/Publish-subscribe_pattern.
[8] https://developers.google.com/location-context/fused-location-provider/.
[9] https://developer.android.com/guide/topics/sensors/sensors_overview.html.
[10] Singh, Meena, et al. "Secure mqtt for internet of things (iot)." Communication Systems and Network Technologies (CSNT), 2015 Fifth International Conference on. IEEE, 2015.
[11] https://www.w3.org/Protocols/HTTP-NG/http-prob.html.
[12] Pan Wei, Zhiguo Hong; Minyong Shi, "Performance analysis of HTTP and FTP based on OPNET" in 15th international conference on computer and information science (ICIS), 2016 (pp. 1-4), IEEE.
[13] https://www.ibm.com/developerworks/community/blogs/5things/entry/5_things_to_know_about_mqtt_the_protocol_for_internet_of_things?lang=en.
[14] https://github.com/eclipse/paho.mqtt.android.
[15] Z. Wei, Y. Song, H. Liu, Y. Sheng and X. Wang, "The research and implementation of GPS intelligent transmission strategy base on on-board Android smartphones," Computer science and network technology (ICCCSNT), 2013 3rd International conference on , Dalian, 2013, pp.1230-1233.
[16] Kraijak, S., &Tuwanut, P. (2015, October). A survey on internet of things architecture, protocols, possible applications, security, privacy, real-world implementation and future trends. In Communication Technology (ICCT), 2015 IEEE 16th International Conference on (pp. 26-31). IEEE.
[17] https://en.wikipedia.org/wiki/MQTT.
[18] http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-3.1.1.html.