A V2I BASED APPROACH TO MULTICAST IN VEHICULAR NETWORKS

Ravi Tomar [1*], Hanumat G Sastry [2], Manish Prateek [3]

[1,2,3]School of Computer Science, University of Petroleum & Energy Studies, Dehradun, India

Email: ravitomar7@gmail.com[1], hsastry@ddn.upes.ac.in[2], mprateek@ddn.upes.ac.in[3] (corresponding author)

DOI: https://doi.org/10.22452/mjcs.sp2020no1.7

ABSTRACT

The present research work proposes a novel approach for efficient information dissemination in Vehicular Networks using V2I based communication. This approach makes use of the V2I communication to establish the connectivity between the cloud server and the vehicle. The approach is focused on finding the utility of existing IoT protocols in the Vehicular Networks. The foundation of this work relies in the (Message Queue Telemetry Transport Protocol) MQTT protocol, IoT and IaaS. The paper presents the background technologies used in the work and propose the approach. The proposed approach has been experimentally tested and found significant results.

Keywords*: IaaS, MQTT, IoT, VANETs, V2I*

1.0 INTRODUCTION

VANTEs has been an area of interest for researchers and is continuously evolving[1]. The VANETs evolved as a subclass of (Mobile Ad-Hoc Network) MANETs bearing special characteristics like high mobility and dynamic topology[2]. Initially, (Vehicular Ad-Hoc Network) VANET research was focused towards the V2I based communication but as new technologies are emerging and availability of better infrastructure in smart cities, VANETs are becoming more of V2I based approach. This work is focused on the integration of various new technologies to incorporate in vehicular networks. To get a better understanding how this approach works, let us imagine an accident event has occurred on the road. The vehicles engaged in accident and the following vehicle senses the event, and further update other vehicles about the event. This timely information helps approaching vehicles in making a better decision to apply brake or choosing alternate route. The scenario is presented in the figure below:

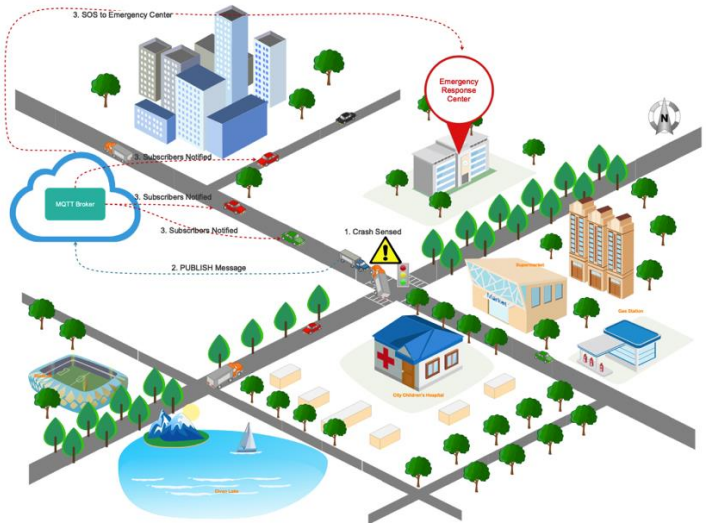

Fig. 1: Scenario depicting the working of MQTT

All the vehicles are equipped with the capable hardware to communicate with each other through MQTT protocol, the MQTT broker resides as a central server for entire network. The information about the event is sensed by some or more

vehicles, they publish the information to the broker under the appropriate topic. After publishing the information, the broker forwards it to the subscribers who are subscribed under the topic. The following section discuss about the multicasting in VANETs followed by the various background technologies used in this approach, the detailed explanation of the proposed approach and concludes with the experimental setup of the proposed approach.

1.1 Multicast in VANET

In multicast Information dissemination techniques for VANETs, the messages travels from a single sender to a group of interested vehicles[2]. Multicasting reduces the transmission overhead, control overhead and power consumption by sending the copies of messages to various vehicles simultaneously. The MANET based multicast approaches are not well suited for VANETs due to its characteristics like unpredictable topology, and high mobility etc., and hence multicast information dissemination are the most active research area due to their efficiency and mobility within a dynamic environment like VANET. The present research made use of latest technologies such as Infrastructure as a Service(IaaS)[3], Internet of Things(IoT)[4] and Message Queue Telemetry Transport protocol (MQTT)[5] to implement the multicasting in VANETs. The following section explain background technologies used specifically in this work.

2.0 BACKGROUND TECHNOLOGIES

In this section, we discuss the background technologies which are used in this work. These technologies integrate to form a foundation for the proposed approach.

2.1 Cloud Computing

Cloud computing is the on-demand delivery of compute power, database, storage, applications, and other IT resources via the internet with pay-as-you-go pricing[6]. The cloud computing provides user access to servers, storage, databases, and an expansive set of application services over the Internet. The following figure presents the overall architecture of cloud computing.

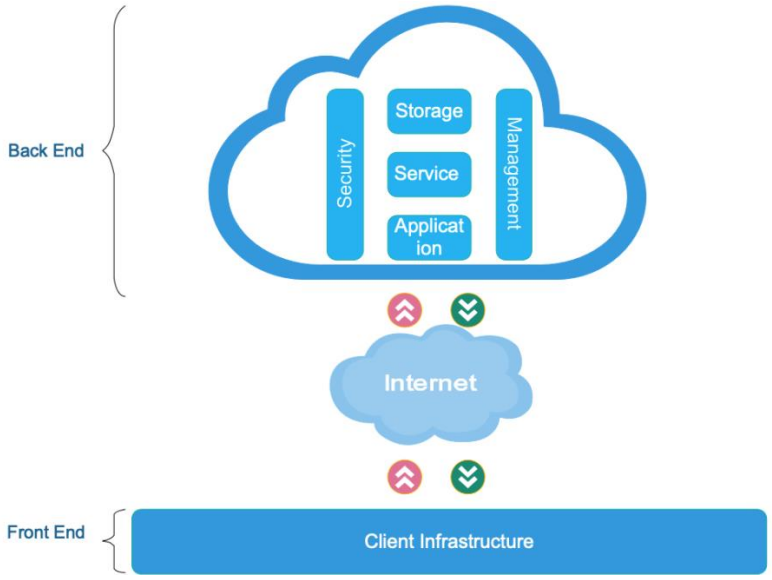

Fig. 2: Cloud Computing Architecture

The cloud services provider such as Amazon Web Services(AWS) [6] and Google cloud [7] owns and maintains the network-connected hardware required for these application services, while you provision and use what you need via a web application. Due to its simplicity, cost effectiveness, speed, efficiency, productivity, performance and security features cloud computing is a popular option for people and businesses. The following figure presents the cloud services offered versus on premises hardware[8].

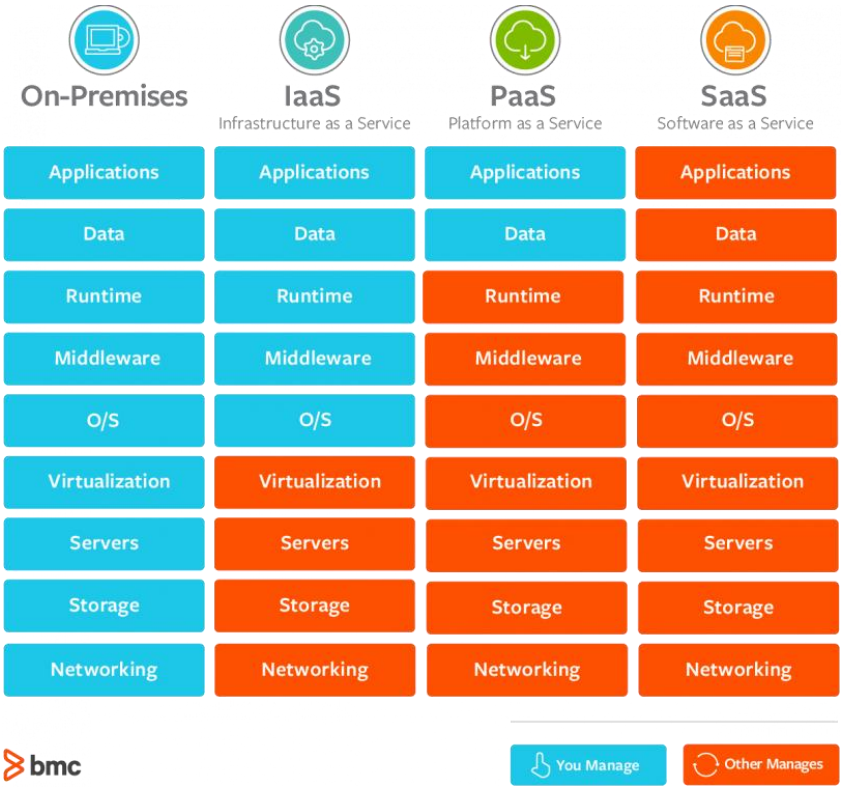

Fig. 3: Different types of Cloud Services[8]

The cloud services are broadly classified into three categories, i.e.

a) Infrastructure as a Service (IaaS)
b) Platform as a Service (PaaS)
c) Software as a Service (SaaS)

a) Infrastructure as a Service (IaaS): This type of cloud service provides raw computing hardware over the internet such as storage servers. The user do not require to maintain the physical hardware and no high upfront cost is required. This is also known as utility computing and examples of IaaS are Amazon EC2[9], Windows Azure[10], Rackspace[11], Google Compute Engine[12] etc.

b) Platform as a Service (PaaS): This type of cloud service provides the platform which comprises of all the hardware and software for any specific development process. The examples of PaaS are AWS Elastic Beanstalk[13], Heroku[14], Apache Stratos[15] etc.

c) Software as a Service (SaaS): This type of cloud service the complete software runs on a remote machine and the user can use it through the web browser or Linux terminal. The examples of SaaS are Google Apps[16], Microsoft Office 365[17].

In the present research program, the proposed approach for V2I communication has been built upon IaaS service EC2[9] from AWS. This IaaS service provides optimized network solution, all other required software and its deployment are configured.

2.2 Internet of Things (IoT)

The Internet of Things (IoT) is a system of interrelated computing devices with unique identifiers (UIDs), having the ability to communicate and transfer data with each other without requiring the human intervention[18].

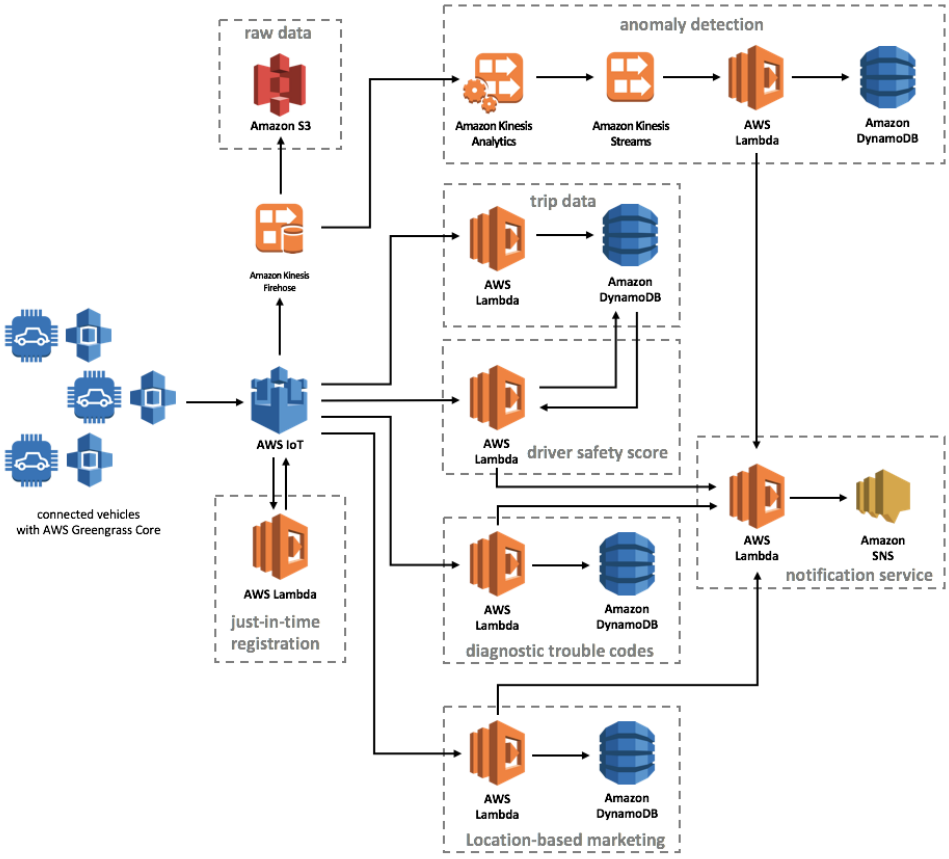

Fig. 4: AWS architecture for Connected Vehicles in IoT[reff-awsiot]

These computing devices are coupled with mechanical machines, people, objects etc. to provide a connected ecosystem. The IoT is relevant to Vehicular Networks as vehicles becomes the objects and obtains the capability to communicate with surroundings. This leads to new dimensions in safety, security, infotainment and traffic updates. The IoT is not a single technology but a convergence of multiple technologies like machine learning, sensors and embedded systems. One such example of integration of IoT with existing cloud services is presented in the Fig. 4. The Fig. 4 presents the AWS architecture[19] for connected vehicles, this can be observed from the figure that the convergence of many services leads to an overall implementation of V2I based services.

2.3 Message Queue Telemetry Transport (MQTT)

Message Queue Telemetry Transport or MQTT[5] for short is an ISO standard for networking, i.e. ISO/IEC PRF 20922[5], introduced by IBM (1999) and later on standardized by OASIS (2013). MQTT is a lightweight networking protocol based on a publish-subscribe model[20]. It aims to connect applications and networks in a comparatively faster manner as compared to the standard TCP/IP protocol. The following figure presents the MQTT PUB-SUB architecture.

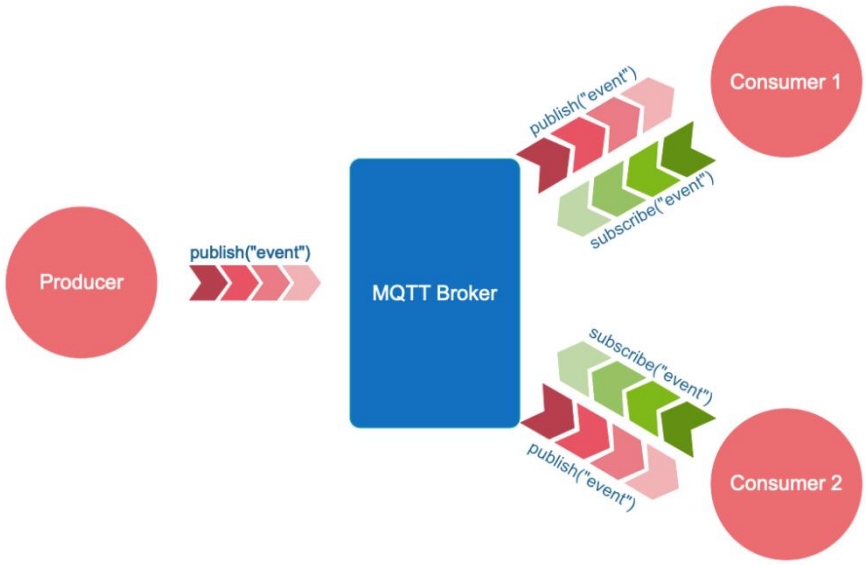

Fig. 5: MQTT publish subscribe architecture

MQTT is mainly designed to provide faster communication in remote locations with limited bandwidth and a small code footprint [5]. In the middle of the publish-subscribe model sits a message broker. Publishers are mostly lightweight clients submitting information related to some topic or another. Subscribers are the entities interested in the topics. The broker provides communication between the two or more parties involved in communication and is also responsible for classifying messages into topics. MQTT is based on an event-driven architecture where a topic of interest between parties is chosen as a channel for communication. Publishers publish data concerning a topic and subscribers subscribe to it to receive information. MQTT protocols are proven efficient and scalable and thus work as a solution for servicing Vehicular Ad-hoc Networks. The architecture is such that it enables rapid scalability and the ability to deliver information without getting parties to interact directly. The following figure presents the message format of MQTT protocol.

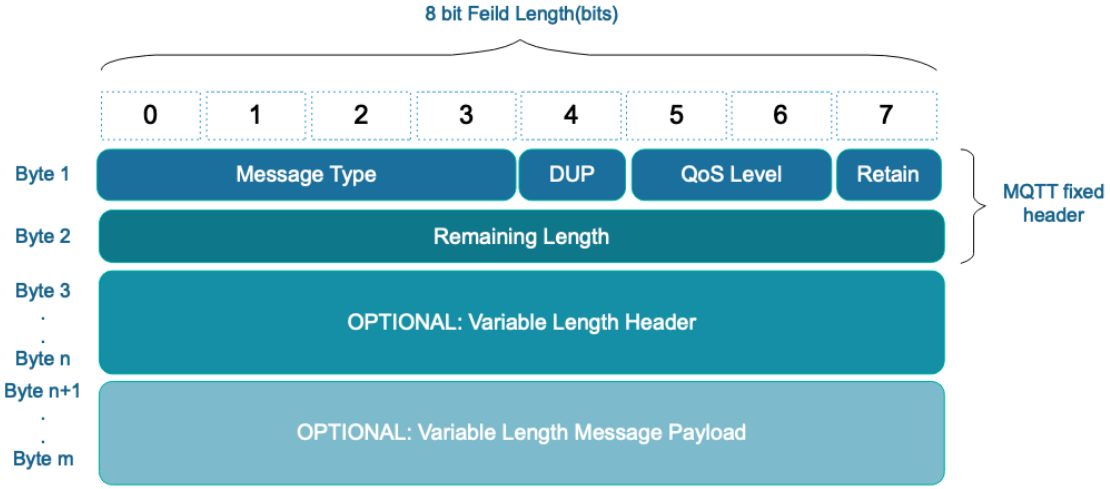

Fig. 6: MQTT Protocol message format

The MQTT protocols also provides reliability through the QoS settings, the following subsection describes the QoS parameters in MQTT protocol.

2.3.1 Quality of Service (QoS) in MQTT

The MQTT protocol is scalable and efficient but the objective of MQTT protocol is also to ensure expected delivery of messages. The need of application is to get a reliable delivery even through the unreliable network. However, as the network scales, number of messages generated and received increases exponentially. This abrupt increase in messages may create server overhead and thereby introduce some latency in the network and thus resulting in delay of critical information. This reliability and latency can be taken care through the QoS levels provided in the MQTT protocol. The MQTT protocol provides three Quality of Service levels and are used to categorize the messages. The following figure presents the various QoS levels and are explained as:

- QoS (0): The message is delivered at most once with no rebroadcasting and this mode is enabled by default. QoS(0) is the fastest mode but also the most unreliable one to deliver information.
- QoS (1): This mode ensures that the information is delivered at least once by requesting a PUBACK acknowledgement packet in return. The overhead of PUBACK exchange introduces a little overhead, and also the same packet can be delivered multiple times to the subscriber.
- QoS(2): This is the most reliable and the slowest mode to deliver information as it comprises of a 4-way message exchange.

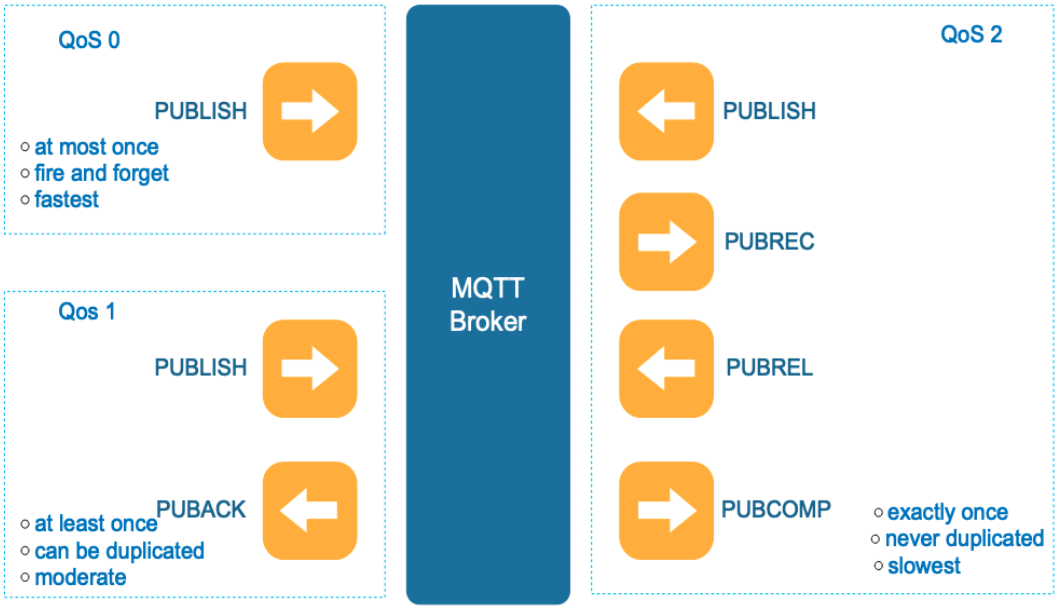

Fig. 7: QoS level in MQTT protocol

2.3.2 The Downgrading of QOS

MQTT protocol-based information exchange is a many to many communication mechanisms and both the subscribers and the publishers has to agree on some QoS to reliably deliver the messages. One of the characteristics of MQTT is that sender and receiver need not to know each other and they do not directly communicate. This lack of direct communication raises a concern regarding the consensus on QoS level. This is where the MQTT QoS provides the downgrading mechanism. In the downgrading process, if a publisher publishes some information using the QoS (2) or QoS(1) and the subscriber is not accepting the same QoS, then downgrading of QoS modes takes place until both publishers and subscribers are at the same mode. In present work we set the QoS level to 2, so we can achieve maximum reliability with permissible latency. The following figure presents the QoS downgrading process.

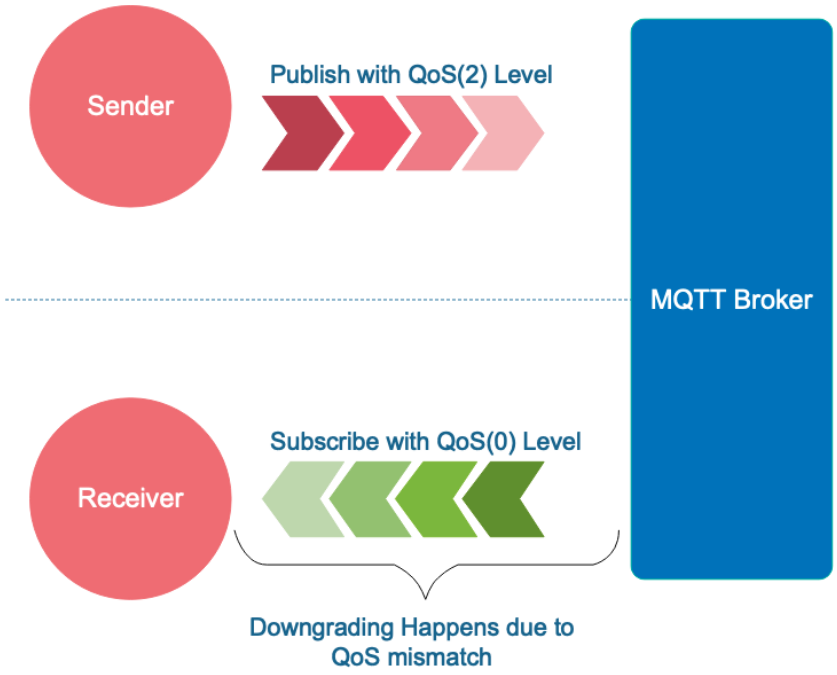

Fig. 8: QoS Downgrading process

3.0 PROPOSED APPROACH

This section explains the proposed novel approach to multicast in VANETs. This approach has been experimentally devised and tested in real world implementation. This approach make use of V2I based communication and has shown significant results in real world testing. The following section presents the overall working and implementation of the approach.

The proposed approach towards efficient information dissemination in VANETs, uses the IaaS cloud services and IoT technology hardware and the MQTT protocol. The MQTT broker server is deployed on the cloud in the application layer and cloud network infrastructure is used to enable communication among various nodes. The scenario explained in the Fig. 1 used as a base for showcasing and implementing our approach. It is assumed that all the vehicles in the network are equipped with an On-Board Unit(OBU) with Global Positioning System(GPS) module. OBU is essentially one of the main components of vehicles in VANETs, it is responsible for gathering and processing the information sensed by multiple sensors within the vehicle. GPS module is used to provide coordinates of the vehicle in real-time. OBU leverages GPS to provide the exact location (latitude and longitude) in the form of a set of coordinates i.e.

$$p \in C \text{ with } C \in \Re 2$$

This approach proposes a separate layer on the top the existing OBU in every vehicle. This layer manages the V2I communication. The transmission and reception of information is handled by this layer. In-vehicle to infrastructure communication, the broker is deployed on the cloud server and enables the reliable and efficient information delivery to vehicles. The information is disseminated to vehicles based on topics to which they have subscribed themselves. The working of our model can be better understood by the following three phases:

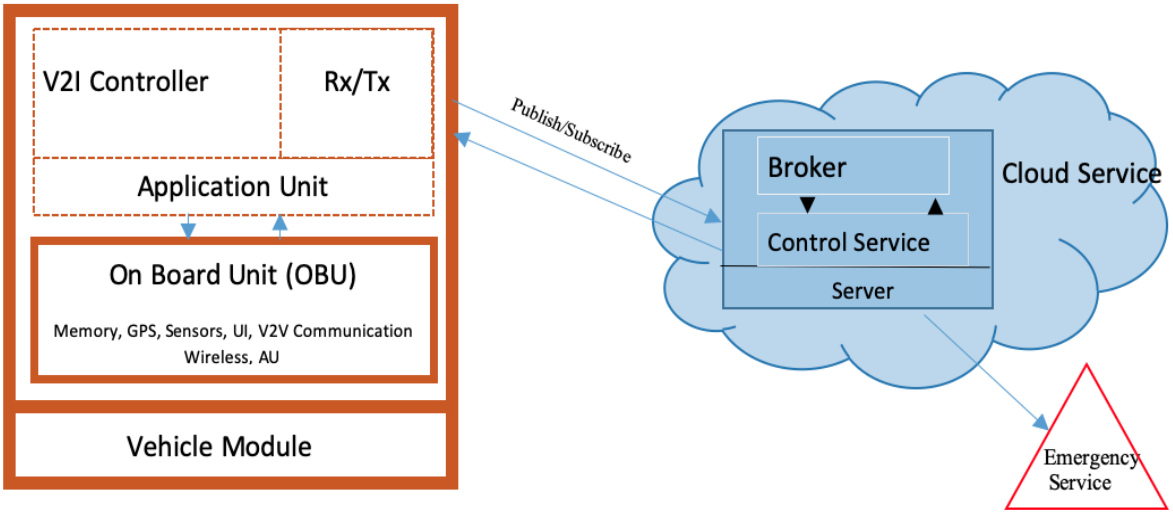

Fig. 9: Internal structure of each module

3.1 Subscription Phase

The subscription phase is the first phase in this approach. This phase begins by requesting the set of coordinates (C) from the OBU of vehicles. The coordinate set comes in the form of latitude and longitudes, these are used to identify the ID of the road on which the vehicle is moving (Road_id). Using this Road_id parameter, a common property is established for the vehicles moving on the same road. A Google-based API[21] is used to resolve these coordinates into meaningful information. The API takes coordinate set of vehicles as input and generates a JSON based output which defines minute details such as route, region, locality, country etc. The Road_id is used as the route name as it is a static and unique string which can be used throughout the working of the model. This process is repeated at regular interval to keep vehicle update every time. The Road_id parameter is passed on to the subscribe function of MQTT Server and this further delivers the message to all the vehicles subscribed to the Road_id. The system ensures to unsubscribe to the old Road_id topic as it enters the new Road_is, this avoids the delivery of outdated messages. The following figure gives the flow chart depicting the working of subscription phase.

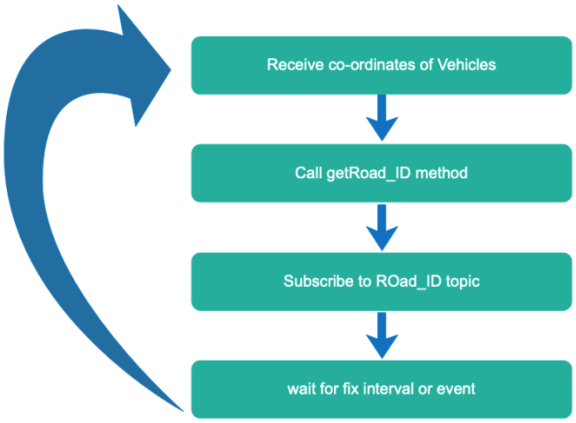

Fig. 10. Subscription phase

3.2 Information Dissemination Phase

The Information dissemination or the publish phase is when some event takes place in the network on the road segment. The vehicles nearby the event will gather data via OBU and send/publish it to the MQTT broker. These vehicles also assign a priority to the messages in the form of QoS. The message is published along with the QoS level and the Road_id

as the topic. All the subscribers to that particular Road_id will receive the message broadcasted by the broker. The flow chart for this phase is given as:

Fig. 11: Information dissemination phase

3.3 Information Reception Phase

The information reception phase is the reception of published message from the previous phase. In this phase all the subscribed vehicles receive the message and further action is taken.

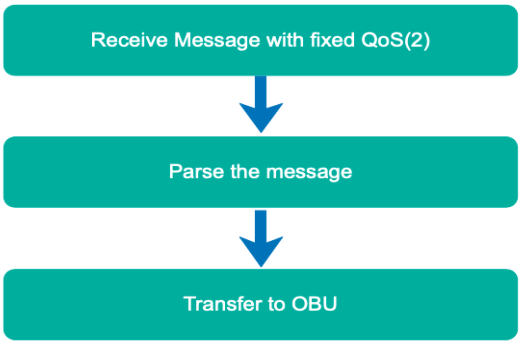

Fig. 12: Information Reception Phase

4.0 EXPERIMENTAL SCENARIO

The proposed approach has been implemented with the following hardware and software.

Table 1: Hardware and Software Configuration

HARDWARE		
No.	Component	Configuration
1.	NODEMCU V3[22]	Microcontroller: Tensilica 32-bit RISC CPU Xtensa LX106 Operating Voltage: 3.3V
2.	BREAKOUT BOARD[23]	DC Input voltage: 6V to 24V Onboard 5V / 1A DC-DC step-down converter circuit. Lead out the pins of 5V and 3.3V power supply
3.	UBLOX NEO6MV2 GPS[24], [25]	Receiver Type : 50 Channels, GPS L1 frequency, C/A Code, SBAS: WAAS, EGNOS, MSAS
SOFTWARE		
1.	AWS EC2 INSTANCE[9]	ECU: Variable Memory: 2 GB Network: up to 5 Gigabit
2.	MOSQUITTO MQTT BROKER[26]	Protocol: mqtt Port: 1883 Listener: 1884

The experiment has been carried out using EC2 cloud service from AWS and deploying mosquitto MQTT broker[26] on it. Every vehicle in the network has an ESP826612E microcontroller onboard along with Ublox Neo6mv2 for GPS coordinates. To implement and evaluate the results, we build two working prototypes and got satisfactory results. Every vehicle subscribes to the topic based on Road_id to which they belong at any given point of time. For testing purpose, we generated overspeed alert above 50kmph and the vehicle sense the occurrence of an event, gather the data and publish it under a suitable topic along with a chosen priority. The broker then sends the messages to the subscribers based on the QoS level chosen. The approach works on PubSub model and is thus implicitly multicast. Apart from this information exchange, we also provide the ability to publish emergency messages to the Emergency channel along with the level/mode of QoS decided by the sender. It ensures fast and assured delivery of emergency information. The following image shows the microcontroller used in our experimental setup.

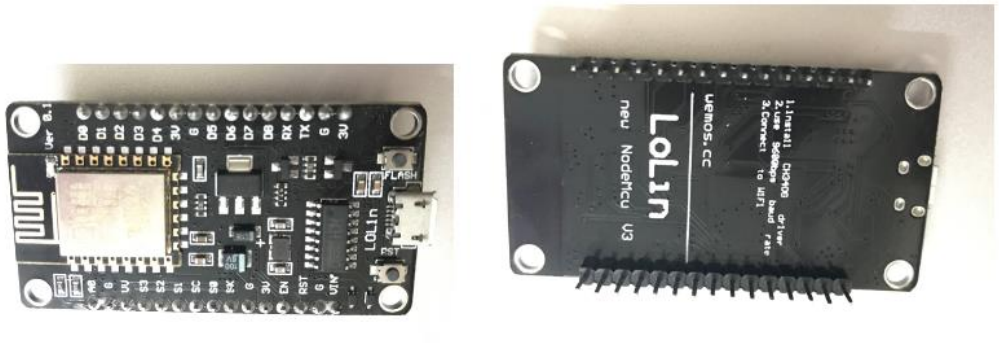

Fig. 13: NodeMCU with ESP826612E Module

The following image in Fig. 14(left) presents the breakout board for NodeMCU, this board is used to receive 12V power supply from the car battery and power up the microcontroller module.

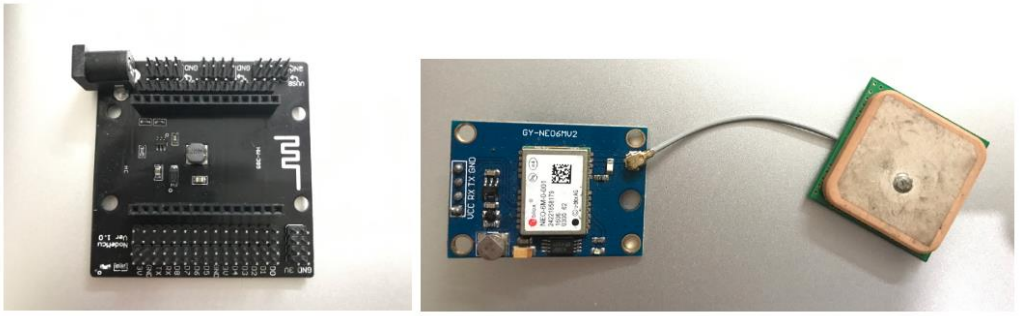

Fig. 14: (left)Breakout Board for NodeMCU with 12v supply support, (right) Ublox Neo6mv2 GPS device

The image in Fig. 14 (right)presents the GPS device used to gather the coordinates of the vehicle at any point of timeandtThe following image in Fig. 15 presents the fully working hardware integrated in lab.

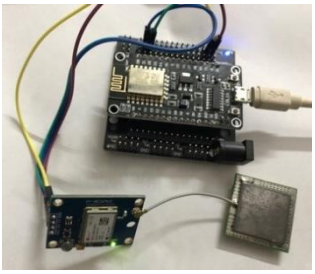

Fig. 15: Lab View Running Module

The screenshot in Fig. 16 provides the debug console snippet of connecting the microcontroller to the WIFI, MQTT server, sensing the coordinates, setting the topic and finally publishing it.

```
Connecting to RaviTomar
.....
WiFi connected
IP address:
192.168.1.123
Attempting MQTT connection...connected
System Ready, connected to 5 sattelites
Getting Topic.....
found, now setting Topic to Sewla Kalan Majra, Dehradun
****Message arrived****
at SewlaKalanMajraDehradun
{"lat":30.29916,"lon":78.00322,"tst":1572619900}
```

Fig. 16: Clipped Debug Console

The following image presents the final ready hardware, ready to be mounted in the vehicle.

Fig. 17: Full Prototype Assembled

The following image demonstrate the deployed hardware on car dashboard and updating the real time location on the cloud server. The mobile hotspot device is provided for internet connectivity and power is connected to the 12 Volt supply through 12 V to 5V converter. The system gets stable connection to internet and GPS coordinates in approximately 1 minute. The system is ready to transmit and receive information on the go. The further sections displays the dashboard of the information exchange between the device.

Fig. 18: Mounted in Car Cabin

The following image presents the screenshot of the dashboard of running vehicles on the road while sharing their normal running "OK" messages with other vehicles. The dashboard shows, precise location of vehicle, number of vehicle on map, latitude, longitude, time and date, topic subscribed and the last message received.

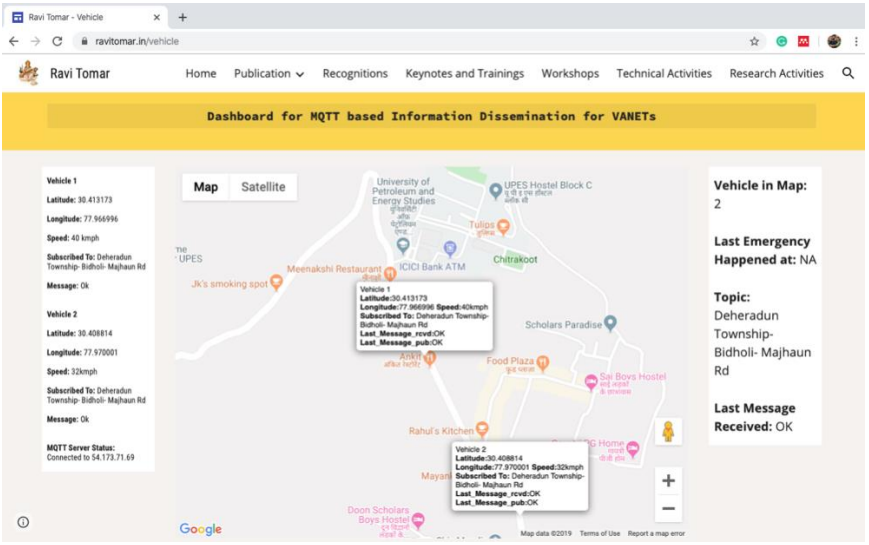

Fig. 19: Screenshot of dashboard showing running vehicle with no emergency.

The following figure presents the screenshot of the dashboard of running vehicles on the road while sharing the alert message of speed above than 50kmph with other vehicles. The dashboard shows, precise location of vehicle, number of vehicles on map, latitude, longitude, time and date, topic subscribed, and the last message received.

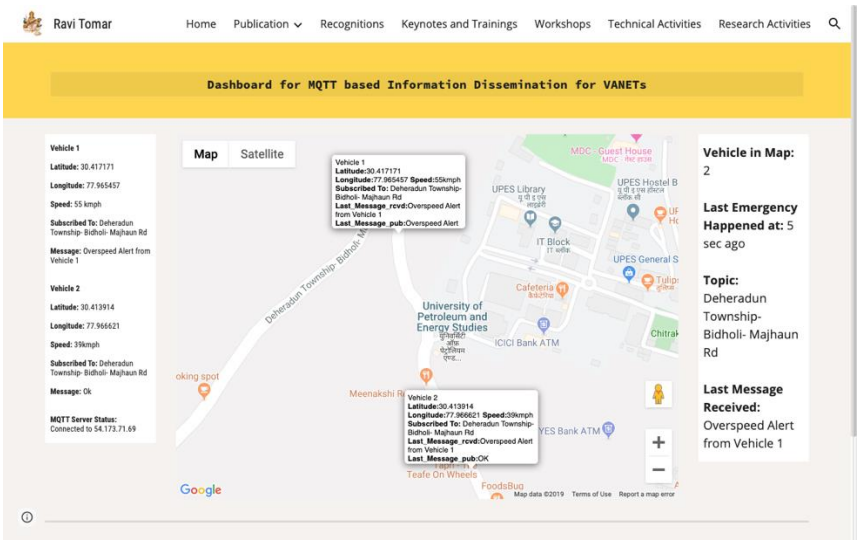

Fig. 20: Screenshot of dashboard showing running vehicle with emergency message

5.0 DATA INTERPRETATION

The data collected during the experimental setup has been analyzed and summarized in the table below. The data of 300 seconds has been randomly picked for both vehicles. The Table 2 presents the time taken by the single message to be delivered to all other vehicles after publishing to the broker. The data is for latency in seconds for a single message. We could see the results are significantly good, as MQTT protocol is highly scalable so the same message can be delivered to large number of vehicles in almost same latency without having much computational and communication delay. We further investigated the amount of byte transmitted over the network, one publishes per second for 300 seconds resulted in 300 packets and 3,22,200 bytes of data. The whole transmission took 5. 989 seconds and this includes the processing at publish node. Table 3 presents this summarized data.

Table 2: Latency Summary

Minimum	Average Latency	Maximum
0.0314	0.033	0.0347

Table 3: Exchange Summary

Byte Transmitted	Number of packet	Time in Seconds
322200	300	5.989

6.0 CONCLUSION AND FUTURE DIRECTION

In the present work we proposed a novel approach to multicast in Vehicular network using V2I based communication. The paper started with introduction to V2I and multicast in VANETs, and further discussed other relevant technologies required for the proposed work such as cloud computing, IoT and MQTT. Upon identification of the needs and the available technologies, the system model is presented. To validate our approach, we built two prototypes for testing and received significant results. This approach works on Pub/Sub Model and hence is the multicast-based approach. As a future work, the vehicle may subscribe to the approaching road, so it is informed about the events happening on approaching road in advance. More work can be done to optimize the Road_id finding method by locally providing the data from specific city.

REFERENCES

[1] G. G. Suresha, "Internet Access in Vehicular Ad-Hoc Networks Based on Infrastructure and LTE Communications Masters Thesis in Communication and Signal Processing," No. January, 2017.

[2] W. Farooq, M. A. Khan, S. Rehman, and N. A. Saqib, "A Survey of Multicast Routing Protocols for Vehicular Ad Hoc Networks," *International Journal of Distributed Sensor Networks*, Vol. 2015, 2015.

[3] M. Y. Pandith, "Data Security and Privacy Concerns in Cloud Computing," *Internet of Things and Cloud Computing*, Vol. 2, No. 2, p. 6, 2015.

[4] R. Tomar, M. Prateek, and H. G. Sastry, "A novel approach to multicast in VANET using MQTT," *Ada User Journal*, Vol. 38, No. 4, pp. 231–235, Dec. 2017.

[5] "MQTT." [Online]. Available: http://mqtt.org/. [Accessed: 30-Oct-2019].

[6] "What is Cloud Computing." [Online]. Available: https://aws.amazon.com/what-is-cloud-computing/?nc1=f_cc. [Accessed: 30-Oct-2019].

[7] "Google Cloud Computing, Hosting Services & APIs | Google Cloud." [Online]. Available: https://cloud.google.com/gcp/. [Accessed: 30-Oct-2019].

[8] "SaaS vs PaaS vs IaaS: What's The Difference and How To Choose – BMC Blogs." [Online]. Available: https://www.bmc.com/blogs/saas-vs-paas-vs-iaas-whats-the-difference-and-how-to-choose/. [Accessed: 01-Nov-2019].

[9] "Amazon EC2." [Online]. Available: https://aws.amazon.com/ec2/. [Accessed: 06-Nov-2019].

[10] "Microsoft Azure Cloud Computing Platform and Services." [Online]. Available: https://azure.microsoft.com/en-in/. [Accessed: 06-Nov-2019].

[11] "Rackspace: Managed Dedicated & Cloud Computing Services." [Online]. Available: https://www.rackspace.com/. [Accessed: 06-Nov-2019].

[12] "Compute Engine | Google Cloud." [Online]. Available: https://cloud.google.com/compute/. [Accessed: 06-Nov-2019].

[13] "AWS Elastic Beanstalk – Deploy Web Applications." [Online]. Available: https://aws.amazon.com/elasticbeanstalk/. [Accessed: 06-Nov-2019].

[14] "Cloud Application Platform | Heroku." [Online]. Available: https://www.heroku.com/. [Accessed: 06-Nov-2019].

[15] "Apache Stratos." [Online]. Available: https://stratos.apache.org/. [Accessed: 06-Nov-2019].

[16] "G Suite: Collaboration & Productivity Apps for Business." [Online]. Available: https://gsuite.google.com/intl/en_in/. [Accessed: 06-Nov-2019].

[17] "Microsoft 365 | Combines Office 365, Windows 10, Security." [Online]. Available: https://www.microsoft.com/en-in/microsoft-365. [Accessed: 06-Nov-2019].

[18] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Generation Computer Systems*, Vol. 29, No. 7, pp. 1645–1660, 2013.

[19] "Architecture Overview - AWS Connected Vehicle Solution." [Online]. Available: https://docs.aws.amazon.com/solutions/latest/connected-vehicle-solution/architecture.html. [Accessed: 01-Nov-2019].

[20] "What is Pub/Sub Messaging?" [Online]. Available: https://aws.amazon.com/pub-sub-messaging/. [Accessed: 30-Oct-2019].

[21] "Get Started | Geocoding API | Google Developers." [Online]. Available: https://developers.google.com/maps/documentation/geocoding/start. [Accessed: 01-Nov-2019].

[22] "NodeMcu ESP8266 V3 Lua CH340 Wifi Dev. Board - Robu.in | Indian Online Store | RC Hobby | Robotics." [Online]. Available: https://robu.in/product/nodemcu-esp8266-v3-lua-ch340-wifi-dev-board/. [Accessed: 06-Nov-2019].

[23] "NodeMCU ESP8266 Serial Port Baseboard Lua WIFI Development Board - Robu.in | Indian Online Store | RC Hobby | Robotics." [Online]. Available: https://robu.in/product/nodemcu-baseboard-nodemcu-lua-wifi-development-board-esp8266-serial-port-baseboard/. [Accessed: 06-Nov-2019].

[24] "(No Title)." [Online]. Available: https://www.u-blox.com/sites/default/files/products/documents/NEO-6_DataSheet_(GPS.G6-HW-09005).pdf. [Accessed: 06-Nov-2019].

[25] "NEO-6 u-blox 6 GPS Modules Data Sheet NEO-6-Data Sheet This document applies to the following products: Name Type number ROM/FLASH version PCN reference," 2011.

[26] "Eclipse Mosquitto." [Online]. Available: https://mosquitto.org/. [Accessed: 06-Nov-2019].