

Project Report

on

MEC API Integration with 5GC

for

location based services in V2X infrastructure

Topics in Wireless Networks – CS6260

**Arun Kant Dubey** (CS20MTECH12008)

**Madhuri Annavazzala** (CS21MTECH02003)

**Guided by: Mentored by:**

**Dr.** [Bheemarjuna Reddy Tamma](mailto:tbr@cse.iith.ac.in) **Supriya Dilip Tambe**

**Index**

| Slno. | Title | Page no. |
| --- | --- | --- |
| 1 | Introduction | 2 |
| 2 | Problem Statement | 2 |
| 3 | System Architecture   1. UE(SUMO/Mobile UE) 2. RAN+UE Emulator 3. 5G Core 4. MEC Server and Location Service API 5. V2X Application Server 6. Screenshots of the SUMO scenario | 3  4  4  4  4  4  7 |
| 4 | Implementation   1. Core side 2. MEC side 3. V2X server 4. Application | 11  11  11  11 |
| 5 | Performance Analysis | 12 |
| 6 | Challenges | 13 |
| 7 | Conclusion | 13 |
| 8 | Future Extensions | 14 |
| 9 | References | 14 |

**1.Introduction**

The past few years have proved that Autonomous driving is no longer a technology out of sci-fi, but a brutal reality. There are a lot of mobile manufacturers, OEMs, network providers etc working in association with the standardization organizations to build up standards that could be followed widely across the globe. Like all technologies, Autonomous Vehicles also come with some requirements followed by some challenges that need to be taken care of. Autonomous vehicles are being designed with the concepts such as safety, efficiency and convenience in mind and is the main reason why Intelligent Transport Systems are in the first place designed.

Most Autonomous Vehicular applications have stringent requirements in terms of time due to the criticality of the functions they offer and the importance of synchronization on road. The information that is passed from one vehicle to another is quite critical in keeping the roads safer for the users. Information such as the coordinates of other vehicles on the road need to be passed on with as little latency and as high-precision accuracy as possible.

This requirement of latency has directed us to add the information nearby the user to reduce the latency, hence the introduction of Multi-Access Edge Cloud(MEC).

Previously, this information had to be fetched from the cloud every time thereby increasing the round trip latency.

In this report, we describe how we have leveraged the MEC location API to get the location details of the nearby vehicles and to alert any vehicle that is within 100 m range of the target vehicle. We also later show that there is a huge improvement in the latency using this approach when compared to the traditional cloud variant of it.

**2.Problem Statement**

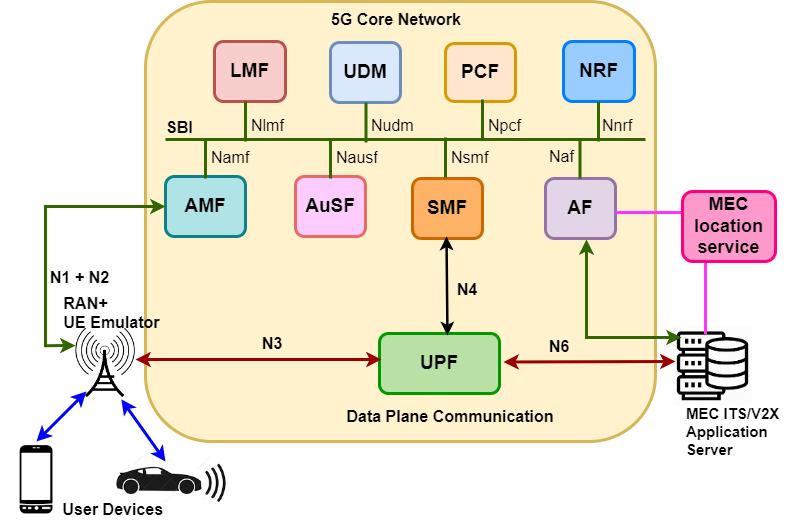
The problem statement involves the following parts-

* To integrate the MEC Location service API with the 5G Core.
* To get the location information from the MEC location service API.
* Perform the calculations at MEC and send an alert to the user if the nearby vehicle is within 100m range of the target vehicle.

**3.System Architecture**

The System Architecture for the project includes the following components-

* UE( SUMO/ Mobile UI)
* RAN+UE Emulator
* 5G Core
* MEC Server and Location Service API
* V2X Application Server

****

**Fig 1: System Architecture**

3.1 UE(SUMO/Mobile UE)

Vehicular traffic is generated using Simulation of Urban mobility(SUMO) , an open source traffic simulation package. A segment of road is simulated from IIT-Hyderabad campus where vehicles are running in a loop at different speeds to provide the situation to demonstrate the desired scenario.

3.2 RAN+UE Emulator

The RAN+UE Emulator emulates the Radio Access Network in 5G System as well as the User Equipment. It gets the location information from the simulated scenario in SUMO and encodes it into an LPP PDU and passes it onto the Core.

3.3. 5G Core

The 5G Core supports the location based services with the help of a network function called Location Management Function(LMF). The LMF decodes the LPP PDU and provides the location response. This location response is sent over to Application Function(AF) via Access and Mobility Function(AMF) after finding out if any instance of AF has subscribed for such location services or not.

3.4 MEC Server and Location Service API

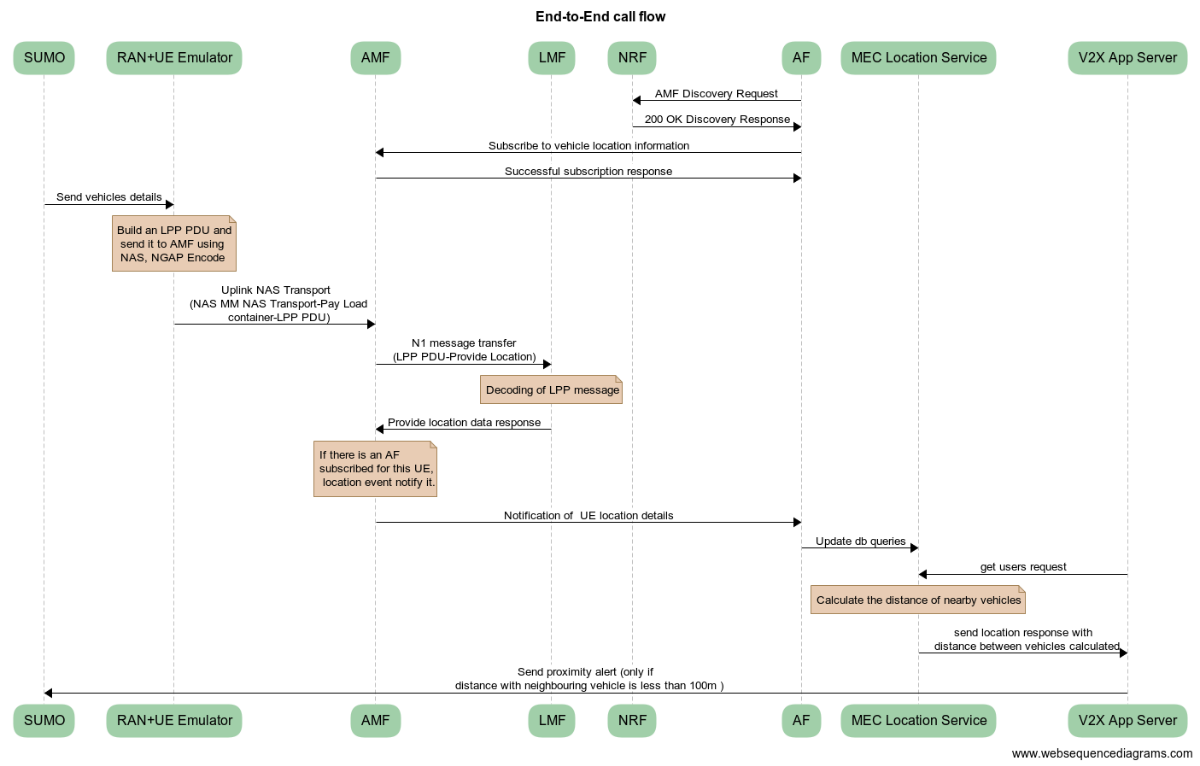
The MEC Server performs all the heavy lifting in the application, such as calculation of distances whenever a new location coordinate has been added to the database. It also creates a list of vehicles that are nearby(within a 100m range of the target vehicle) and only sends this relevant information to the V2X application server.

The MEC Location Service API has been defined by the ETSI forge standards and has been used in the project. It serves as the access point for the MEC Server. It helps in fetching the location details.

3.5 V2X Application Server

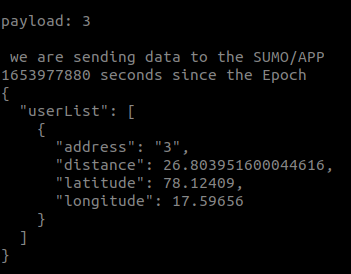
The V2X application server is responsible for receiving the relevant list of information from the MEC Server and simply forwarding it to the related vehicles as an alert.

The sequence diagram depicting the actions among various actors in the end-to-end system is as depicted below-



**Fig 2: End-to-end call flow between the actors for location services in autonomous vehicles**

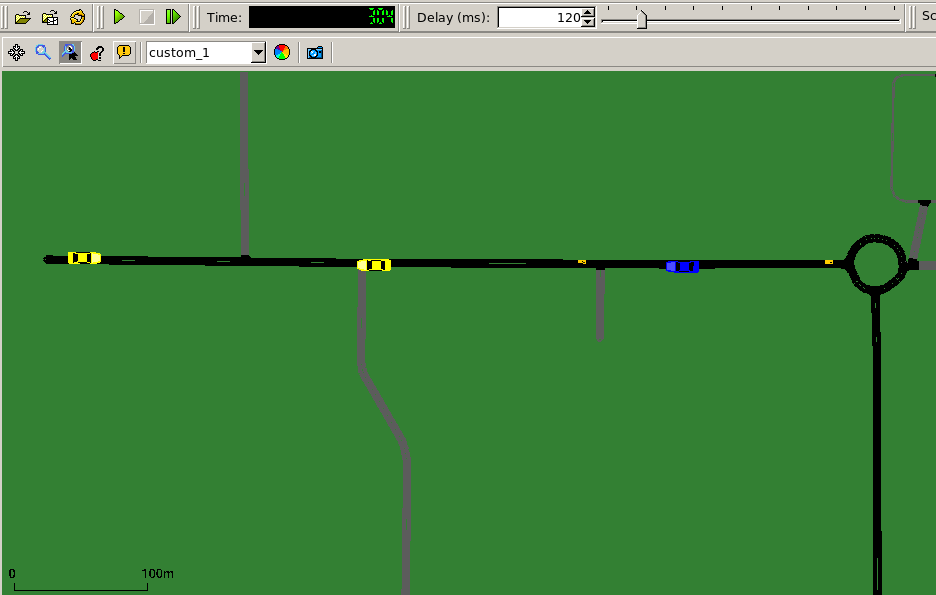
* SUMO to RAN: sendto() is used to send vehicle information over the socket, this includes vehicle ID, location latitude and location longitude.
* RAN to AMF : In In5gtRanNew.cpp, the function called **in5gtProvideLocationtoAmf()** is used to send the location details to AMF.
* AMF to AF : In In5gtAmfNew.cpp the function named in5gtNotifyAf() is used to send the location details to AF after being configured into the required message format in the function named  **in5gtN1MessageNotify()**.
* AF to MongoDB(location collection in MEC Server): In In5gtAFServer.cpp the **serverhandleNotify()** manages the location details. We created a mongo thread that will manage the inserts into the DB present in the MEC Server.
* MEC Server (via MEC API) to V2X Application Server: MEC Server has a method called **users\_get()** in the location\_controller.py that is called whenever a request is invoked by the V2X application server. It fetches the latest location coordinates for each vehicle and computes the distance between them. And the relevant message details are forwarded to the V2X application server(in the following format) to notify the vehicle.



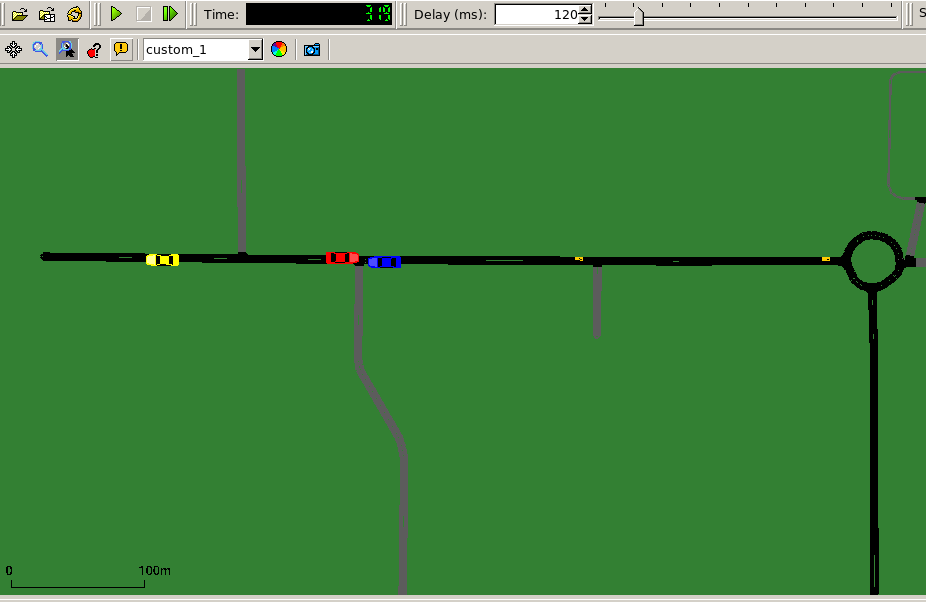
**Fig: JSON format consisting of vehicle ID, latitude, longitude and its distance from the target vehicle(vehicle ID 1).**

* V2X application Server to SUMO: In modAI.cpp, we have a function named send() that is used to send the alerts onto SUMO.

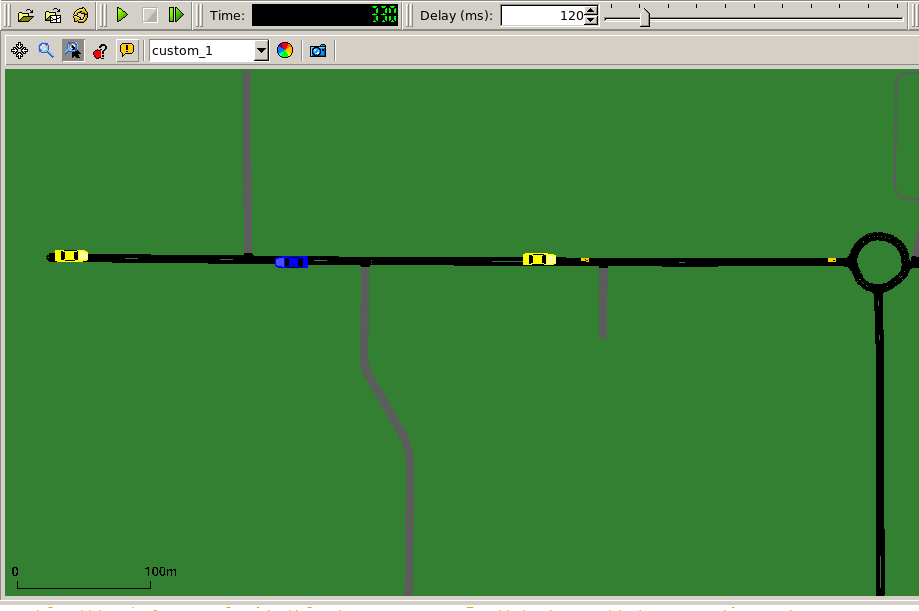
The following images present the end SUMO output we receive. A Blue vehicle in the scenario depicts the target vehicle. A target vehicle is the vehicle which will receive the alert if any other vehicle is within 100 m of its range. A yellow vehicle depicts all other vehicles present in the scenario which are not in the 100 m range. Once the yellow vehicle comes within the range of 100m of the target vehicle, it turns red(this calculation after being done at the MEC side).



**Fig: Target vehicle(Blue) and other vehicles in the scenario(yellow)- near Hostel circle, IIT Hyderabad**



**Fig: Nearby vehicle turns red(earlier yellow) as soon as it comes within 100 m range of the target vehicle( in blue)**

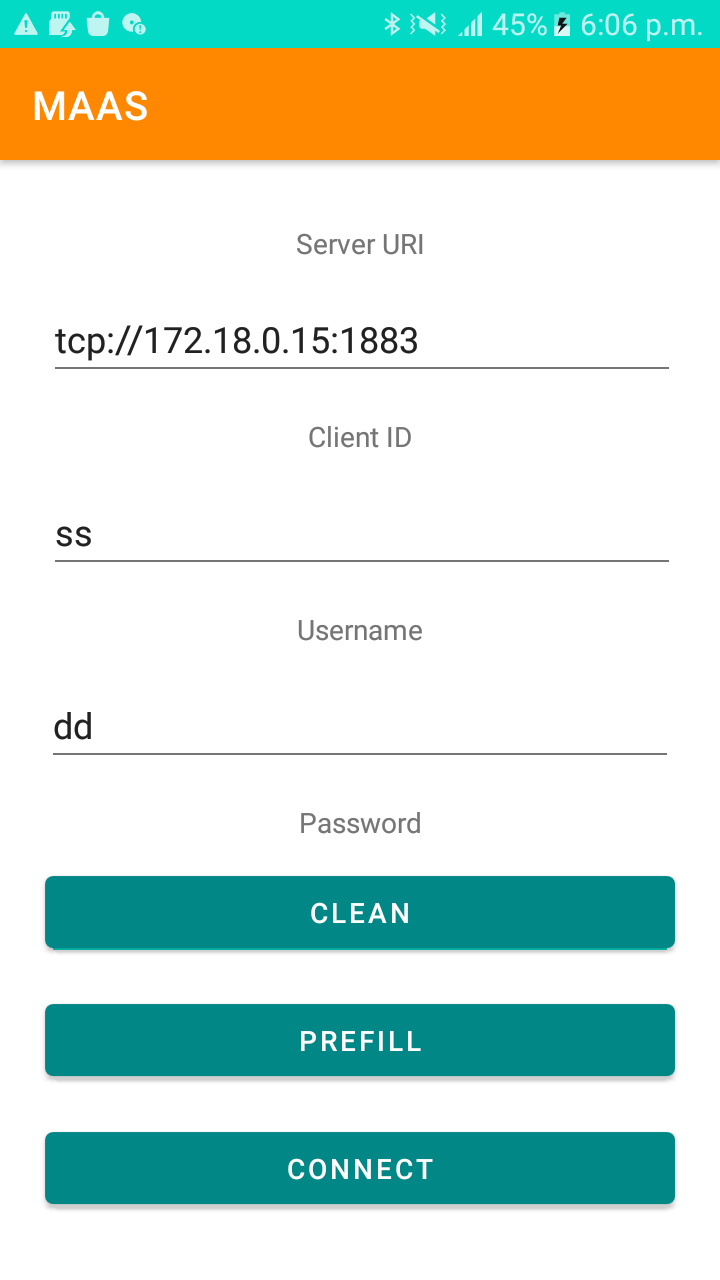


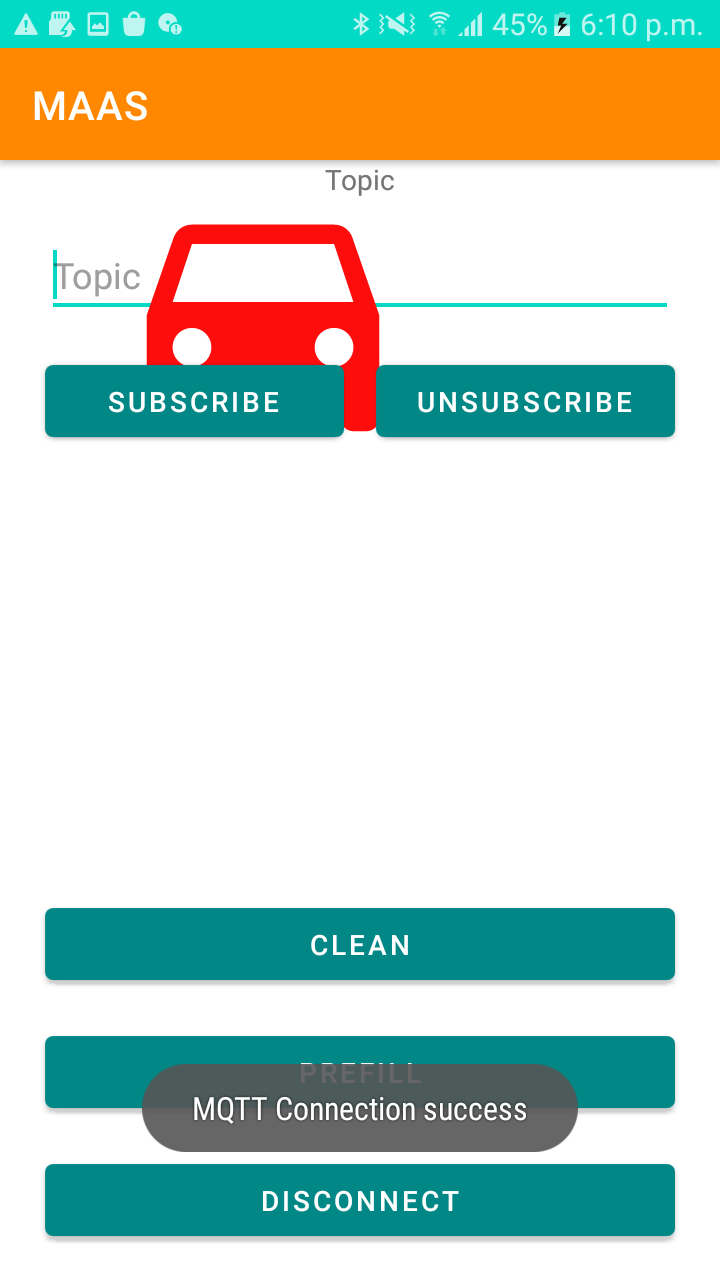
**Fig: The nearby vehicle turns back to yellow as soon as it leaves the 100m range of the target vehicle(in blue)**

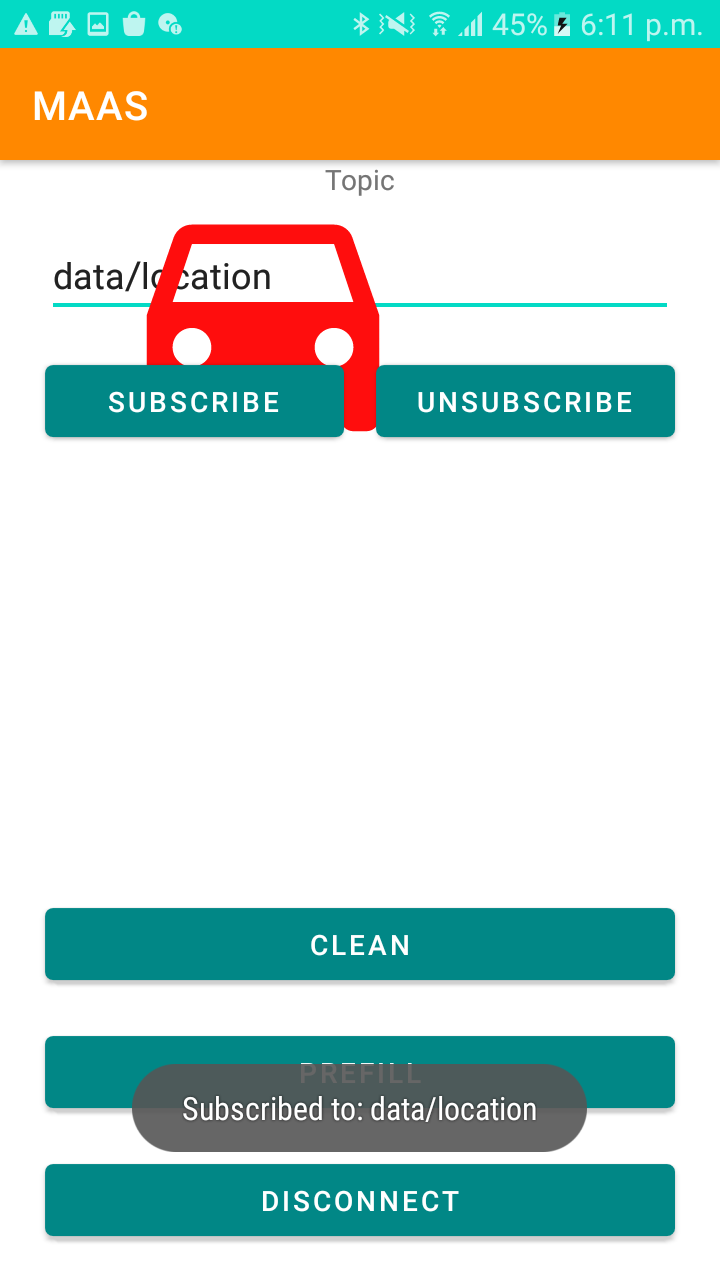
* V2X application Server to Android application:

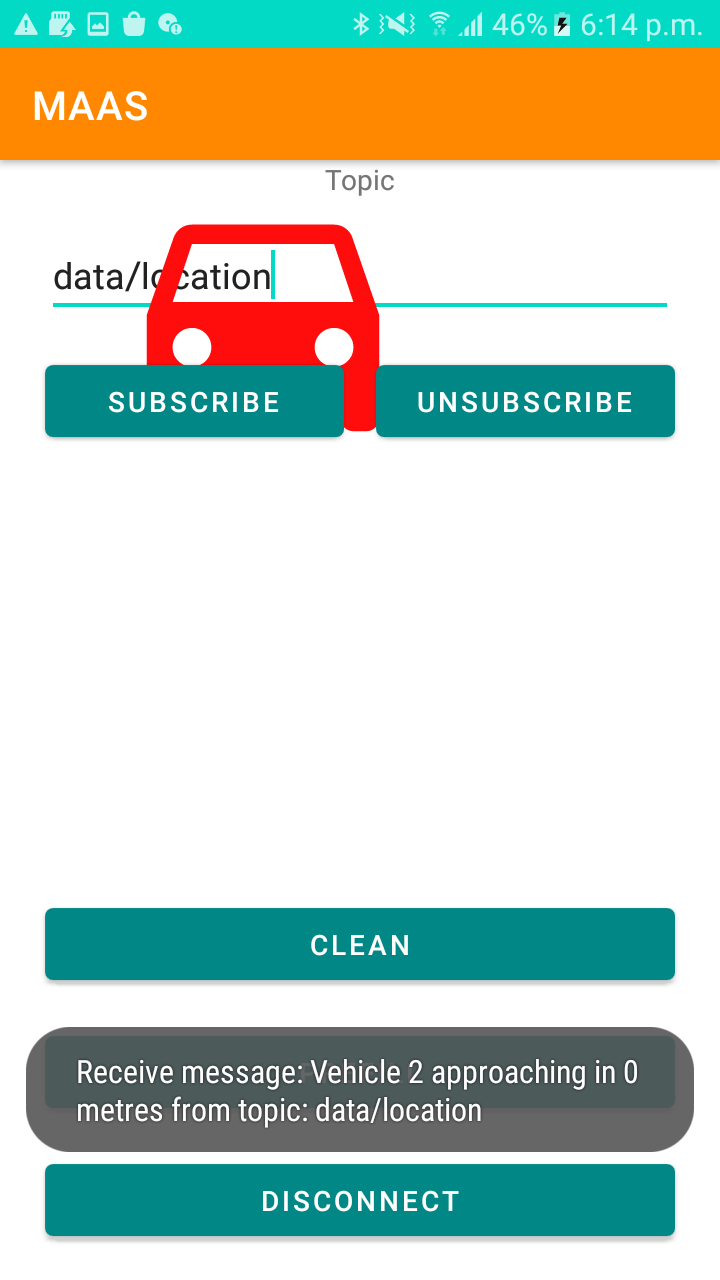
To receive alerts based on location of the nearby vehicle an android application is created that uses MQTT protocol to get alerts from the V2X application server. A pop-up comes specifying the nearby vehicles list. The latency for this setup is around **1841 ms.**

Below are the screenshots of the mobile application:

****

****

****

****

**4.Implementation**

The following details give an overview of the changes done to implement the above system architecture to provide the location-based services in 5G Core integrated with the MEC Location Service API-

4.1 SUMO side

A vehicular network is created where a road segment is simulated using simulation of urban mobility (SUMO). The vehicles are made to run in a loop at varying speed to demonstrate the desired scenario, as the vehicle density in the network increases the vehicle starts receiving alert and turns blue, the neighboring vehicles to the target vehicle in consideration change color when within 100 meters.

4.2 Core side

Changes have been done in Application Function(AF). It has been made capable of receiving the location details from AMF and to convert it into the JSON format that is acceptable by the MEC API. This format is then inserted into a database that is present in the MEC Server.

4.3 MEC Server and Location Service API

The MEC Location Service API when invoked with a curl request from the V2X Application server, fetches the data from the Database and gives it to the MEC Server for further processing.

The MEC Server, gets the latest coordinates of each vehicle and calculates the distance between the target vehicle to that of the other vehicles in the scenario and if it appears to be within the range of 100m of another vehicle, it sends a notification to the V2X application server. We have calculated the Euclidean distance between the coordinates given, by first converting the given degree latitude and longitude into coordinates.

4.4 V2X Application Server

The V2X application server has been built to invoke the MEC API with a curl request. The response is given whenever a certain condition(Vehicle within 100m range of target vehicle) is met. It also then forwards the same to the user(target) vehicle as and when it receives a response from the MEC Server.

**5. Performance Analysis**

The performance analysis of the system has been done by taking the values of latency at each component involved in providing the location services. We define the end-to-end latency as the time taken from sending a set of coordinates from SUMO to receiving an alert in SUMO.

This involves the sum of latencies from a varied set of components in the setup which are as follows:

* Time taken to send coordinates to RAN(+UE emulator) from SUMO
* Time taken to pass on the values to AF via AMF from RAN
* Time taken to Update the values at the DB
* Time taken to calculate the distance at MEC Server
* Time taken to send the alert to the UE from V2X application server

| **TASK** | **Avg latency (in milliseconds)** |
| --- | --- |
| **Time taken to send coordinates to RAN from SUMO** | **1** |
| **Time taken to pass on the values to AF via AMF from RAN(check in RAN, AMF, AF)** | **279** |
| **Time taken to send the alert to the UE(end-to-end)** | **361** |

We observed a drastic reduction in the latency when MEC was introduced to provide the location based services. We outsourced the heavy computation such as calculation of distances at each instance and to observe if any vehicle is within the 100m range to MEC. And only the relevant list of vehicles is sent to the V2X application server to forward the alerts. This we believe has helped us reduce the latency to a minimum.

**6. Challenges**

* One of the very first challenges we faced during the tenure of the project was resolving the errors in the YAML of the MEC Location Service API. The literature and the community support online is quite limited due to the fact that the APIs are relatively new and their usage in research was rare.
* While integrating the MEC Location Service API with the existing 5G Core we had to make the changes in format of the location details so that it is acceptable as a valid format by the API.
* An array of discussions were conducted on the format in which the alerts to the user had been conveyed. We initially developed a mobile application and explored the option to send the location alerts using websockets. However, not having the websockets in cpp was a challenge. We later switched to MQTT to send and receive alerts. This added an extra latency of the WiFi involved, so we switched to displaying the alert in the form of change in color in the SUMO interface.
* To demonstrate the use-case we created an android application, where the vehicles were supposed to update their position and color dynamically based on the location of the vehicle in consideration but to demonstrate that we faced issues with synchronization of location information with its representation using the google map sdk.

**7.Conclusion**

In this project, we implemented an end-to-end setup that provides location services in the vehicular networks by leveraging MEC Location Service API, integrated with the 5G Core. We observed that inclusion of MEC for outsourcing the heavy-lifting (such as computation of distances at every instance, to determine the nearby vehicles) can help reduce the latency and can be useful in mission critical applications such as providing emergency alerts in cases of Collision Avoidance etc. We also further note that the project has immense potential for optimization. We plan on optimizing the current code to further reduce the latency to a minimum.

**8. Way Forward**

The project has immense potential for optimization. We plan on exploring other APIs such as VIS(Vehicle Information Service) to generate a network congestion map that will help predict the network conditions and make prior informed decisions of resource allocation as the vehicle moves. Introduction of distributed MEC that can coordinate with each other to perform calculations and provide the output to the nearby vehicle can be another way to go forward. This can be applied in scenarios where there is huge need for heavy computations such as in HD map rendering. A vehicle can fetch the local map from an MEC Server near it and then all the MEC Servers can communicate with each other to generate a global map.

**References**

[**https://forge.etsi.org/rep/mec/gs013-location-api**](https://forge.etsi.org/rep/mec/gs013-location-api)

[**https://www.etsi.org/deliver/etsi\_gs/MEC/001\_099/013/02.02.01\_60/gs\_MEC013v020201p.pdf**](https://www.etsi.org/deliver/etsi_gs/MEC/001_099/013/02.02.01_60/gs_MEC013v020201p.pdf)

[**https://www.eclipse.org/sumo/**](https://www.eclipse.org/sumo/)