Trinity College Dublin

The University of Dublin

COMPUTER NETWORKS

CSU33D03-202122

PROJECT – 2

202122-P1–GROUP 4

VEHICULAR COORDINATION USING P2P NETWORK

Name: Mudit Garg

TCD ID: 21355125

Mail ID: gargmu@tcd.ie

Mobile: +353 892050748

Name: Vanshika Sinha

TCD ID: 21355135

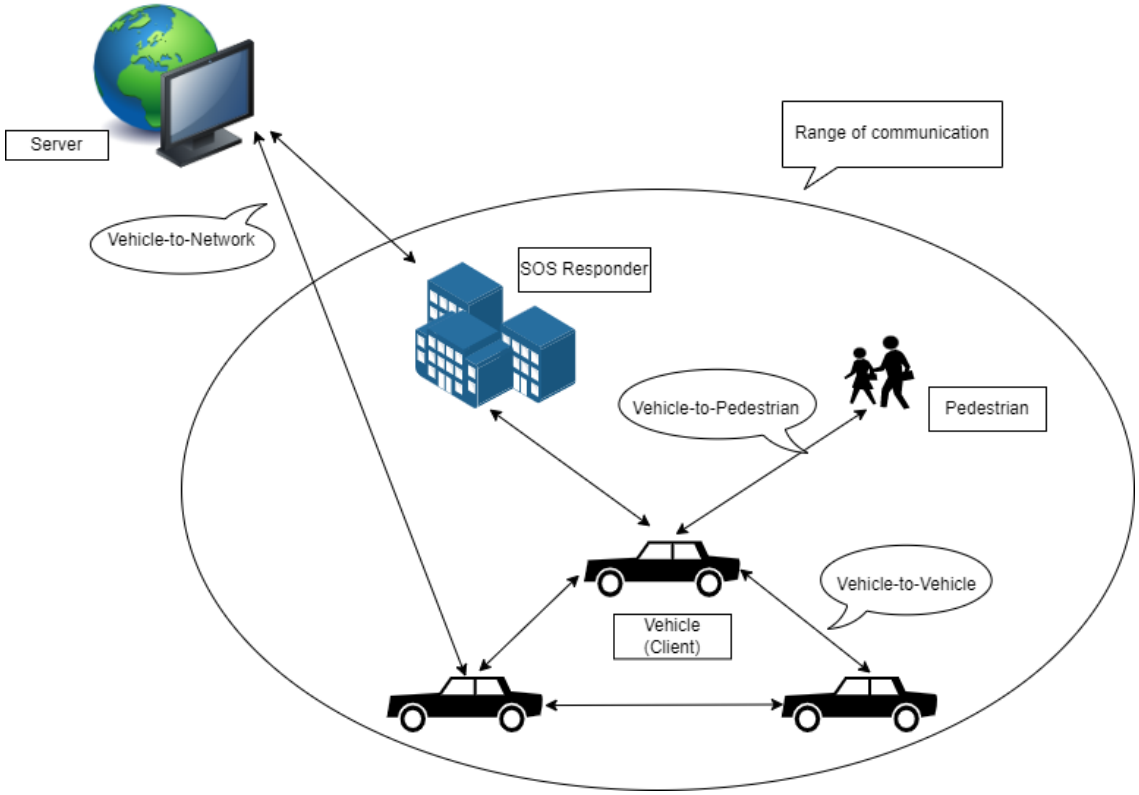
Mail ID: sinhav@tcd.ie

Mobile: +353 894915291

A. Use Case:

In the recent years, vehicular communication has become a platform to maintain smart transportation system to enhance traffic control and safety as well as SOS control. The potential of sharing data such as position, speed, and location of the surroundings vehicles and infrastructures provides significant data for reducing accidents in real time scenarios. Considering the scenario of two cars turning at the same time on a narrow road with a blind turn, greatly increases the possibility of a collision. Thus, a P2P network between vehicles minimizes the risk of accidents by increasing coordination.

B. Logical Diagram:



C. Structure:

In our model, we have incorporated best of both worlds - **Client-Server and Peer-to-Peer**.

The Client-Server connection only exists while a new client is getting added to the network. This handshake is required for successful connection of the new client to the network via the server. This connection is also required for flagging a particular client as a vehicle or a SOS responder. This will be helpful in the future for broadcasting SOS alerts to only the SOS responders.

The Peer-to-Peer connection is present between every client (vehicles and pedestrians included). This connection will always be active and will listen to any messages sent by any one of them. Please note, server will only come in between to decide if the message is a SOS from any one of the clients. This is to inform the SOS responders in case of that emergency. The server can be removed as the middle-man but for the sake of convenience and ease-to-understand, the code implements the path of server as a middle-man.

D. Network model:

The V2V (Vehicle to Vehicle) communication uses the Peer-to-Peer model wherein every node connection is possible via **one-to-one handshake** between each node with every nearby compatible node while incorporating network protocols like –

- **DSRC (Dedicated Short-Range Communication)** to enable nearby peer communication.
- **TCP (Transmission Control Protocol)** to incorporate server to peer connection.

The V2P (Vehicle to Pedestrian) communication uses the **same protocols as the V2V** connection as this model, when incorporated, will consider the pedestrian node to be the same as a vehicle just without input to the network (**Audience Node**).

The N2S (Node to Server) communication uses protocols like **TCP/IP** to enable communication from server to node (i.e., vehicles and pedestrian alike). The choice of this protocol is due to its scalable nature and characteristic.

E. Other considerations e.g., Security, SW Eng., etc.:

The **Pedestrian node can also have the capability to send an SOS signal** in case of emergency.
In this model, **a direct participation of vehicles can be implemented** (which can be used as relays) for the exchange and transfer of information without the involvement of server. This can make the network work even in remote locations (indirectly making it a decentralised network).
In this model, **time is of essence.** So the choice of any future protocol or algorithm must revolve around this major goal that we need to reduce time latency and increase coordination without any data loss or leak.
This model will **never be in charge of controlling the working of the car**. The only purpose of this network is to send and receive data to and from other peers of the network and provide it to the user for the user to take specific and necessary actions. If this network model wants to be a part of the “self-driving cars” revolution, there is a huge security side of this model that has not been given enough thought. There are various technologies like cryptography, blockchain, token verification etc that can be implemented in the data security while transmission of data to ensure that correct and untampered data exchange takes place between the peers and risk of loss remains minimum to zero.

F. Implementation details:

Implementation details are shared below in the Code Instruction section along with its setup.

G. Unique Selling Point(s):

The fact that this model can be upscaled further to various other sectors like vehicle-to-infrastructure (V2I), vehicle-to-pedestrian (V2P) and vehicle-to-network (V2N) communication, apart from vehicle-to-vehicle (V2V) communication makes this project unique from others. In the proposed model, vehicles can communicate on their own without the need of a centralized server, in a Peer-to-Peer fashion using various protocols like DSRC in one of the modules.

H. Advances and Improvements:

i. The proposed model in the previous submission was majorly focused on the Vehicle-to-Vehicle communication, but we have upscaled our research and implementation to vehicle-to-pedestrian (V2P) and vehicle-to-network (V2N) communication.

ii. We have extended our communication model to Client-Server taking into consideration the Peer-to-Peer model as well.

I. Code Instruction:

Note, Python is a prerequisite for the system that is going to run the script(s). To install python, follow the instruction after downloading the setup from - https://www.python.org/downloads/release/python-3102/

This brief tutorial will help the user simulate the server and client(s) on the same computer using different instances of the terminal posing as different clients (some being vehicles, some being pedestrians, some being SOS responders).

Step 1: Extract the .zip file provided. This zip file, when extracted, shall result in a folder containing the following files – *server.py , client.py , ip_port_gen.py, README.md*

Step 2: Open one instance of the terminal in the same folder in which the above-mentioned files are present. Run the command "***python server.py***".

If you get the following message –

"*Server is listening to incoming connection requests. Currently on standby!*"

… your code has successfully run and the instance of the terminal is acting as a server and is waiting for the clients to join.

Step 3: Run another instance of the terminal in the same folder and run the command "***python client.py***".

If you get the following message –

"*Vehicle has started connection to the server ...*"

…. your code has successfully created an instance of a client. It will now ask if you want this instance of the client to a SOS responder or not. Now that is up to you. If you say Y (yes), any SOS message sent by any instance of the client will be visible to these SOS responders only (to prevent panic in the real-world scenario). If you say N (no), it will act as a normal vehicle that can send and receive messages to and from the network.

Step: To create another instance of a client, just repeat Step 3 and answer the SOS responder question based on your need.

Detail: There are specific messages which when sent from a non-SOS Responder client, will trigger the SOS Alert that will be broadcasted to all the SOS Responders. Check the code for further clarity.

Test Cases: Try the following keywords (case sensitive) in order to check if the code is behaving as it should.

1. ACCIDENT - the code should recognise this as a SOS alert and will display a message on the server screen and the screen of every instance of the client that is a SOS responder only (message should not be displayed on the screen of the non SOS responders) (to prevent panic among the others)

2. MALFUNCTION - the code should recognise this as a SOS alert and will display a message on the server screen and the screen of every instance of the client that is a SOS responder only (message should not be displayed on the screen of the non SOS responders) (to prevent panic among the others)

3. TRAFFIC_AHEAD - the code should not recognise this as a SOS alert and will display this same message on the screen of every instance of the client that is not a SOS responder only (message should not be displayed on the screen of the server and the SOS responder) (for privacy & security and to prevent the redundancy of data for others)

4. DIVERSION_AHEAD - the code should not recognise this as a SOS alert and will display this same message on the screen of every instance of the client that is not a SOS responder only (message should not be displayed on the screen of the server and the SOS responder) (for privacy & security and to prevent the redundancy of data for others)

J. Sources and References:

- https://ieeexplore.ieee.org/document/1041239
- http://section.iaesonline.com/index.php/IJEEI/article/viewFile/2736/601
- https://medium.com/@shivamrawat_756/guide-to-p2p-over-internet-9f7cf41470bd
- https://medium.com/hackernoon/socket-programming-in-python-client-server-and-peer-examples-a25c9782b584