ESTABLISHMENT OF EFFICIENT CONGESTION CONTROL IN VEHICULAR AD-HOC NETWORKS

Report submitted to the SASTRA Deemed to be University
as the requirement for the course

CSE302: COMPUTER NETWORKS

Submitted by

BUDDIPATI YATEESH VARDHAN
(Reg. No.: 124158085, COMPUTER SCIENCE & ENGINEERING (IOTA))

December 2022

SCHOOL OF COMPUTING

THANJAVUR, TAMIL NADU, INDIA – 613 401

SASTRA

ENGINEERING · MANAGEMENT · LAW · SCIENCES · HUMANITIES · EDUCATION

DEEMED TO BE UNIVERSITY

(U/S 3 of the UGC Act,1956)

THINK MERIT | THINK TRANSPARENCY | THINK SASTRA

SCHOOL OF COMPUTING
THANJAVUR – 613 401

Bonafide Certificate

This is to certify that the report titled "**Establishment of efficient congestion control in vehicular ad-hoc networks**" submitted as a requirement for the course, **CSE302: COMPUTER NETWORKS** for B.Tech. is a bonafide record of the work done by **Shri. BUDDIPATI YATEESH VARDHAN (Reg. No.124158085, CSE(IOTA))** during the academic year 2022-23, in the School of Computing.

Project Based Work *Viva* voce held on ___ – December – 2022

Examiner 1 **Examiner 2**

ACKNOWLEDGEMENT

At the foremost, we thank Lord Almighty for the bountiful blessings he showered and for being our shield and fortress.

I express our heartfelt gratitude to our beloved **Vice-Chancellor Dr.S. Vaidhya Subramaniam** for giving us an opportunity to be the student of renowned institution.

I also express our thanks to **Dr.R. Chandramouli, Registrar** for giving us an opportunity to do our B.Tech degree in CSE.

I are greatly indebted to **Dr.Umamakeswari A, Dean, SOC(School of Computing)** for allowing us to utilize all the facilities in the campus.

I express our wholehearted thanks to **Dr.Shankar Sriram V S, Associate Dean** , for his valuable inputs.

My heartful thanks to **Dr.Vidivelli S, Professor**, for her constant support and guidance.

I also convey our deep sense of gratitude and profound thanks to all Teaching and non-
Teaching staff of our department who have directly and indirectly helped for successful completion of the project.

TABLE OF CONTENTS

List of Figures

List of Tables

List of Graphs

Abbreviations

Acronym	Abbreviation
VANET	Vehicular Ad-hoc Network
MANET	Mobile Ad-hoc Network
VEINS	Vehicles In Network Simulation
RSU	Road Side Unit
V2V	Vehicle to Vehicle
V2I	Vehicle to Infrastructure

Notations

Notation	Meaning
γ	Congestion level rate
γ_{max}	Maximum congestion level rate
γ_{before}	Congestion level rate just before updating
V_{inst}	Instantaneous speed of the vehicle
$V_{threshold}$	Threshold speed of the vehicle

ABSTRACT

Vehicular Ad-hoc Network (VANET) is a fast-emerging technology derived from ad-hoc networks that provide intelligent communication between vehicles and other external devices. The fact that the VANET deals with high velocity movements of vehicles and the unpredictability of vehicles makes VANET unique from the rest of networks. The primary objective of VANET is to provide on road safety for the vehicles. These safety measures are achieved by making use of two types of message passing techniques viz.,

i) Periodic safety messages

ii) Event-driven messages

Periodic messages are used in order to exchange status information of the vehicle e.g., coordinates, velocity etc. whereas Event-driven messages are broadcasted whilst emergency situations e.g., accidents, harsh braking etc. These two types of messages are broadcasted over same channel. In case of heavy traffic conditions, these techniques are at a risk of failure since the periodic messages might consume the entire channel bandwidth leading to a congested channel. Hence, there is an absolute need for establishing a congestion control algorithm. In addition to that, using external infrastructure is an expensive process. In order to make it cost efficient, we need to reduce the usage of external infrastructure. In this project, we enhance the regular VANET's by making use of only vehicle to vehicle(V2V) communication and reducing vehicle to Infrastructure communication(V2I), thereby reducing the cost.

In this model, we mainly focus on the following:

a) Calculating congestion level of road by each vehicle.
b) Transmitting obtained values to other vehicles

CHAPTER 1

1.1 INTRODUCTION

These days, transportation plays a significant role in people's lives. The reason for this is that people must regularly travel from one location to another. To meet the needs of the population, there has been a great increase in the number of automobiles. The result of all these developments is an increase in traffic in vehicles. This increase in traffic has a significant influence on the environment, public health, and the nation's economy. Hence, it is necessary to regularly track and assess the state of the traffic. We can lessen the negative consequences of traffic on society by doing this

VANETs, which stands for Vehicular Ad-hoc Networks and is a subset of MANETs, are a specific kind of network that are used to monitor the traffic conditions of the automobiles. Vehicles exchange data with one another and with the Road Side Units via this network. V2V (vehicle to vehicle) communication is the sharing of data between vehicles, and V2I (vehicle to infrastructure) communication is the exchanging of data between vehicles and roadside units. Later, this information is employed to calculate how congested the roads are. The vehicles choose the best course to follow based on the information to avoid traffic and get to their destination in time. As a result, a vehicle's travel time is shortened.

VANETs are incredibly beneficial, but they are also overpriced. The use of external units RSUs significantly increases the costs. Furthermore, data transmission using RSUs increases network communication overhead, which has an impact on sensitive data that is prone to transmission delays. We determine the congestion levels and transmit the data using V2V communication to get around all of these problems.

Fig 1.1: Representation of V2V communication among vehicles.

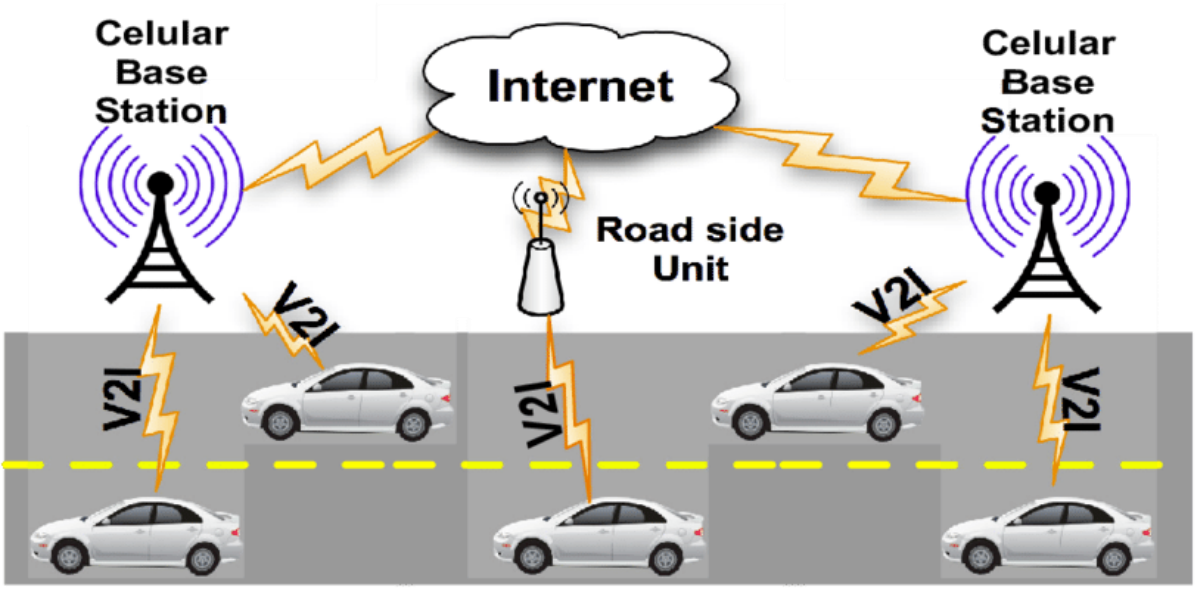


Fig 1.2: Representation of V2I communication between vehicles and Infrastructure

1.2 PROPOSED WORK

In order to cut down on the average journey time of vehicles, we suggest a method in this project for estimating the level of traffic congestion on the roadways. Also, we solely use V2V communication in this project. As a result, the equipment needed for V2I communication is not required, which lowers the cost of setting up the Road side units.

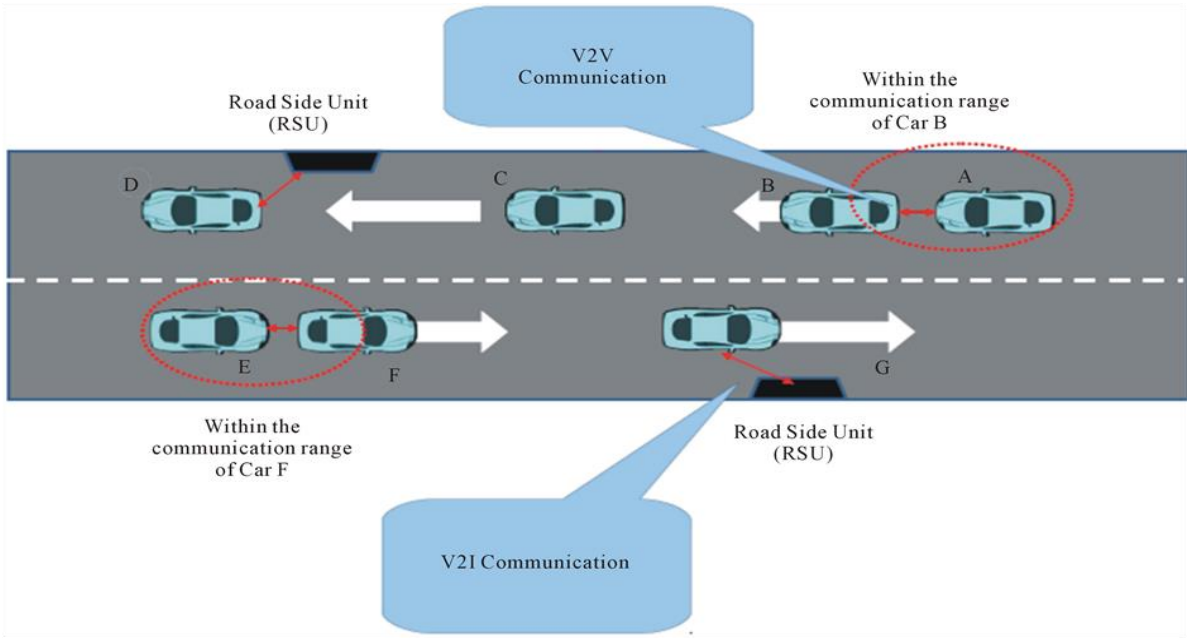


Fig 1.3: Communication in VANET network

The proposed strategy can be achieved in following steps:

- Detection of traffic congestion level.
- Broadcasting the traffic congestion level among other vehicles.

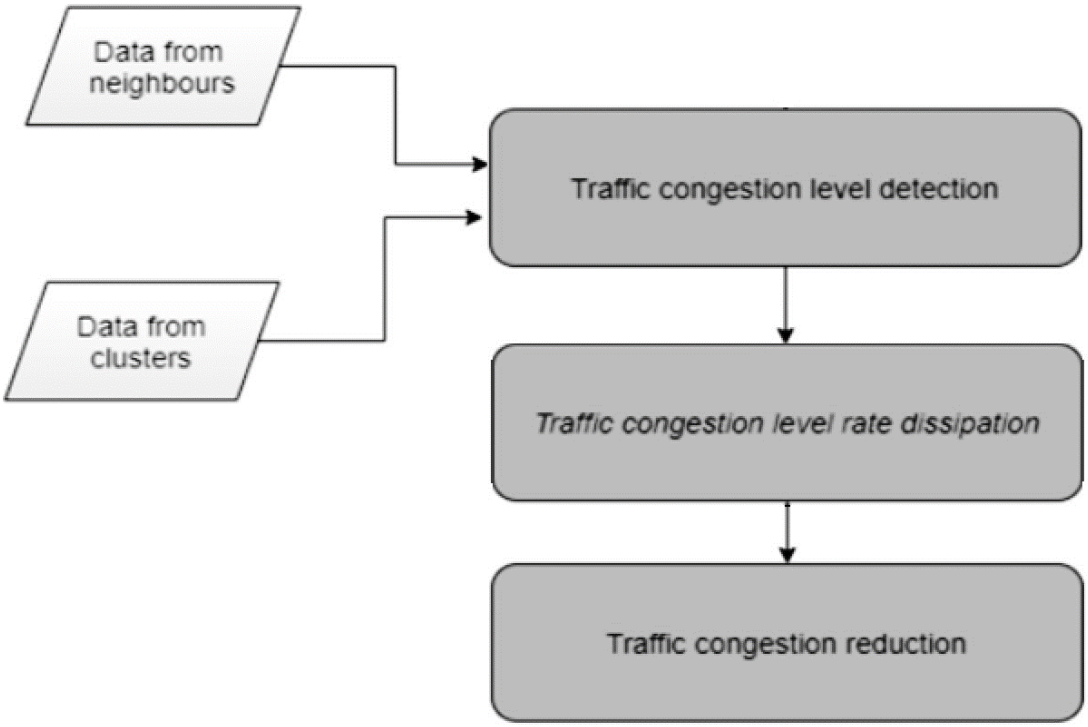


Fig 1.4: Workflow

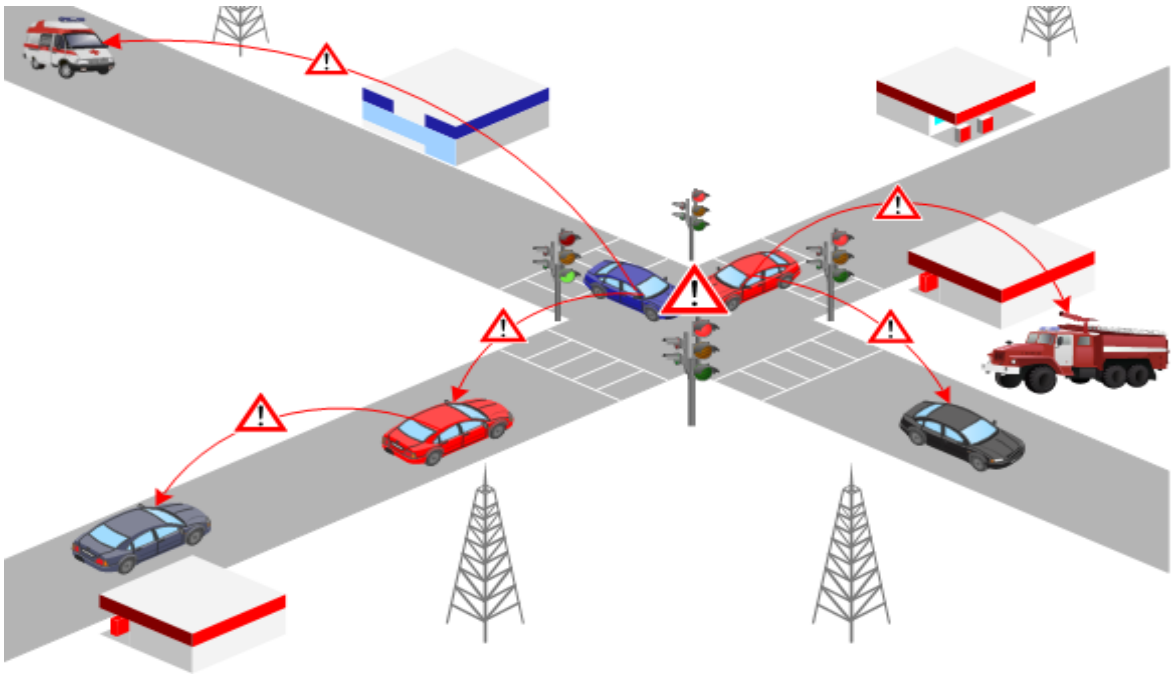


Fig 1.5: Congestion control in VANET

1.3 TRAFFIC CONGESTION LEVEL DETECTION

Estimating traffic congestion is the first and most crucial stage in this approach. Travel time or delay that is longer than what is typically experienced by vehicles on free-flowing or congestion-free roads is referred to as traffic congestion. Numerous studies have demonstrated and found that a decrease in vehicle speed is caused due to traffic congestion. In this context, we determine the amount of traffic congestion based on the speed of the moving vehicles. Here, we introduce a new traffic measure called traffic congestion rate, denoted by γ which represents how congested the road is. Higher the congestion, higher is the value of γ and vice versa. γ ranges between 1 and γ_{max} where, γ =1 represents congestion less zone. With increase in traffic, γ value also increases till γ_{max}. Each road has $V_{threshold}$ which represents the threshold speed that is used to quantify the congestion level. Here, we assume that $V_{threshold}$ is $1/3^{rd}$ of maximum speed of vehicle allowed on the road. Also, we initialize the global variables: lastChangeAt=0, lastChangeCongAt=0, congested=false. The instantaneous speed of the vehicle is given by V_{inst}.

We implement algorithm1 to calculate and update the congestion level.

Parameter	Initialized value
γ (Local congestion level)	1
T_{cong}	10 seconds
T_{free}	20 seconds
$V_{threshold}$	4.63 m/s
congested	False
lastChangeAt	0
lastChangeCongAt	0

Table 1.1: Parameters present in algorithm 1

Algorithm 1:

procedure updateCongestionLevel
  if congested **then**
    if $V_{inst} \leq$ Vthreshold **then**
      lastChangeAt $\leftarrow$ 0.
      $\gamma \leftarrow \min(\gamma_{max}, \ 1+\frac{\text{currentTimestamp} - \text{lastChangeCongAt}}{\text{Tcong}})$
    else if lastChangeAt = 0 **then**
      lastChangeAt $\leftarrow$ currentTimestamp
    else if currentTimestamp − lastChangeAt $\geq T_{free}$ **then**
      congested $\leftarrow$ false
      $\gamma \leftarrow 1$
      lastChangeAt $\leftarrow$ 0
    end if
  else if$V_{inst} \leq$ Vthreshold **then**

      lastChangeAt $\leftarrow$ currentTimestamp
    else if currentTimestamp − lastChangeAt $\geq T_{cong}$ **then**
      congested $\leftarrow$ true
      lastChangeCongAt $\leftarrow$ lastChangeAt
      lastChangeAt $\leftarrow$ 0
      $\gamma \leftarrow \min(\gamma_{max}, \ 1+\frac{(\text{currentTimestamp} - \text{lastChangeCongAt})}{\text{Tcong}})$
    end if
  else
    lastChangeAt $\leftarrow$ 0
  end if
end procedure

Explanation:

If $V_{inst} < V_{threshold}$ for more than T_{cong} seconds, then we set congested as “true” and update the congestion level rate. If $V_{inst} > V_{threshold}$ for more than T_{free} seconds and congested=true, then γ becomes 1. Thus, the vehicles estimate the local congestion level of the roads. Also, when a vehicle moves to another road, timers and congestion level rates are not reset since there is a high possibility that newly entered road might also have similar congestion level rate.

1.4 TRAFFIC CONGESTION LEVEL DISSIPATION/BROADCAST

It is necessary for vehicles to communicate with one another about the level of traffic congestion once it has been estimated. Each vehicle will be able to comprehend the level of congestion in their immediate area and will also have an understanding about the traffic conditions by exchanging information among themselves. The data provided from other vehicles allows the vehicles to select the quickest path to reach their destination.

We implement the following algorithm to broadcast the traffic congestion level rates among other vehicles.

Algorithm 2:

```
procedure performBroadcast
    γ before ← γ
    γ ← updateCongestionLevel
   if γ ≠ γbefore then
          create packet(γ)
          broadcast(congestionData)
          end if
   end if
end procedure
```

Explanation:

Initially, all vehicles traffic congestion level rates are initialized to 1. With increase in number of vehicles, the congestion rate also increases. Now the changed congestion rate must be broadcast to other vehicles. Whenever, there is a change in traffic congestion, we create a data packet with the changed congestion rate and broadcast this packet to all the other vehicles.

1.5 MERITS

- Allows short and medium range communication.
- It doesn’t need any roadside infrastructure.
- Less cost and easy to implement.
- Supports short message delivery.
- Minimizes latency in communication link.
- It is fast and reliable and provides real time safety.
- Protect vehicles from potential road hazards and improve safety.

1.6 DEMRITS

- Frequent topology partitioning due to high mobility.
- Problems in long range communication.
- Problem in broadcasting under environmental forces.

CHAPTER 2

2.1 SOURCE CODE

Source code for algorithm 1: *To update the congestion level of vehicles*

```
double updateCongestionLevel()
{
        conglvl_before = conglvl;
       if(congested)
       {
              if(mobility->getCurrentVelocity().length() <= vthreshold)
              {
                lastChangeAt = 0;
                currentTimestamp = simTime().dbl()*1000 ;
                if(maxconglvl >= (1+(( simTime().dbl()*1000 -
lastChangeCongAt)/tcong)))
                {
                      conglvl = 1+(( simTime().dbl()*1000 -lastChangeCongAt)/tcong);
                }
                 else
                {
                      conglvl = maxconglvl;
                }
              }
              else if(lastChangeCongAt == 0)
              {
                      currentTimestamp = simTime().dbl()*1000;
                      lastChangeAt = currentTimestamp;
              }
              else if(simTime().dbl()*1000-lastChangeAt >= tfree)
              {
                      congested = false;
                      conglvl = 1;
                      lastChangeAt = 0;
               }
       }
       else if(mobility->getCurrentVelocity().length() <= vthreshold)
       {
              if(lastChangeAt == 0)
              {
                      lastChangeAt = simTime().dbl()*1000;
              }
              else if((simTime().dbl()*1000 - lastChangeAt) >= tcong)
```

```
                {
                        congested = true;
                        lastChangeCongAt = lastChangeAt;
                        lastChangeAt = 0;
                        currentTimestamp = simTime().dbl()*1000;
                        if(maxconglvl >= (1+((currentTimestamp-
lastChangeCongAt)/tcong)))
                        {
                             conglvl = 1+(( simTime().dbl()*1000 -
lastChangeCongAt)/tcong);
                        }
                        else
                        {
                                conglvl = maxconglvl;
                        }
                }
        }
 else
{
        lastChangeAt = 0;
}
return conglvl;
}
```

Source code for algorithm 2: *For message broadcast among vehicles*

```
void broadcast()
{
        auto payload = makeShared<VeinsInetSampleMessage>();
        timestampPayload(payload);
        payload->setChunkLength(B(100));
        payload->setRoadId(traciVehicle->getRoadId().c_str());

        auto packet = createPacket(std::to_string(conglvl));
        packet->insertAtBack(payload);
        sendPacket(std::move(packet));
}
```

Overall source code:

//In src folder:

VeinsInetSampleAppliation.h

```
#pragma once
#include "veins_inet/veins_inet.h"
#include "veins_inet/VeinsInetApplicationBase.h"
class VEINS_INET_API VeinsInetSampleApplication : public
veins::VeinsInetApplicationBase {
protected:
        bool haveForwarded = false;
protected:
        bool congested;
        double vthreshold;
        double vinst;
        double conglvl;
        double conglvl_before;
        double maxconglvl;
        double lastChangeAt;
        double lastChangeCongAt;
        double currentTimestamp;
        int tcong;
        int tfree;
        int cnt;
protected:
        double updateCongestionLevel();
        void broadcast();
protected:
        virtual bool startApplication() override;
        virtual bool stopApplication() override;
        virtual void processPacket(std::shared_ptr<inet::Packet> pk) override;
public:
        VeinsInetSampleApplication();
        ~VeinsInetSampleApplication();
};
```

VeinsInetSampleAppliation.cc

```
#include "veins_inet/VeinsInetSampleApplication.h"
#include "inet/common/ModuleAccess.h"
#include "inet/common/packet/Packet.h"
#include "inet/common/TagBase_m.h"
#include "inet/common/TimeTag_m.h"
#include "inet/networklayer/common/L3AddressResolver.h"
#include "inet/networklayer/common/L3AddressTag_m.h"
#include "inet/transportlayer/contract/udp/UdpControlInfo_m.h"
#include "veins_inet/VeinsInetSampleMessage_m.h"
using namespace inet;
Define_Module(VeinsInetSampleApplication);
VeinsInetSampleApplication::VeinsInetSampleApplication()
{
}

void VeinsInetSampleApplication::broadcast()
{
        cnt+=1
        auto payload = makeShared<VeinsInetSampleMessage>();
        timestampPayload(payload);
        payload->setChunkLength(B(100));
        payload->setRoadId(traciVehicle->getRoadId().c_str());

        auto packet = createPacket(std::to_string(conglvl));
        packet->insertAtBack(payload);
        sendPacket(std::move(packet));
}
double VeinsInetSampleApplication::updateCongestionLevel()
//double updateCongestionLevel()
{
         conglvl_before = conglvl;
        if(congested)
        {
                if(mobility->getCurrentVelocity().length() <= vthreshold)
                {
                  lastChangeAt = 0;
                  currentTimestamp = simTime().dbl()*1000 ;
                  if(maxconglvl >= (1+(( simTime().dbl()*1000 -
lastChangeCongAt)/tcong)))
                   {
                        conglvl = 1+(( simTime().dbl()*1000 -lastChangeCongAt)/tcong);
                   }
                    else
```

```
                {
                        conglvl = maxconglvl;
                }
            }
            else if(lastChangeCongAt == 0)
            {
                        currentTimestamp = simTime().dbl()*1000;
                        lastChangeAt = currentTimestamp;
            }
            else if(simTime().dbl()*1000-lastChangeAt >= tfree)
            {
                        congested = false;
                        conglvl = 1;
                        lastChangeAt = 0;
             }
        }
        else if(mobility->getCurrentVelocity().length() <= vthreshold)
        {
                if(lastChangeAt == 0)
                {
                        lastChangeAt = simTime().dbl()*1000;
                }
                else if((simTime().dbl()*1000 - lastChangeAt) >= tcong)
                {
                        congested = true;
                        lastChangeCongAt = lastChangeAt;
                        lastChangeAt = 0;
                        currentTimestamp = simTime().dbl()*1000;
                        if(maxconglvl >= (1+((currentTimestamp-
lastChangeCongAt)/tcong)))
                        {
                             conglvl = 1+(( simTime().dbl()*1000 -
lastChangeCongAt)/tcong);
                        }
                        else
                        {
                                conglvl = maxconglvl;
                        }
                }
        }
 else
{
        lastChangeAt = 0;
}
```

```
return conglvl;
}
bool VeinsInetSampleApplication::startApplication()
{

 if(getParentModule()->getIndex()>=0)
 {
        traciVehicle->setMaxSpeed(13.89);
        congested=false;
        vthreshold=4.63;
        tfree=20000;
        conglvl=1;
        maxconglvl=9999;
        lastChangeAt=0;
        lastChangeCongAt=0;
        tcong=10000;
        cnt=0;
        auto call=[this]()
        {
                EV_INFO<<getParentModule()->getIndex()<<":"<<
                this->updateCongestionLevel();
                /* if(this->conglvl_before!=this->conglvl)
                {
                        this->broadcast();
                }*/
        };
                auto Timerspec=veins::TimerSpecification(call).interval(0.1);
                timerManager.create(Timerspec,"update");
        auto call2=[this]()
        {
                this->broadcast();
        };
                auto Timerspec=veins::TimerSpecification(call2).interval(1);
                timerManager.create(Timerspec,"broadcast");
        }

        return true;
}
bool VeinsInetSampleApplication::stopApplication()
{
        return true;
}
VeinsInetSampleApplication::~VeinsInetSampleApplication()
{}
```

```
void VeinsInetSampleApplication::processPacket(std::shared_ptr<inet::Packet> pk)
{
        auto payload = pk->peekAtFront<VeinsInetSampleMessage>();
        EV_INFO << "Received packet: " << payload << endl;
        getParentModule()->getDisplayString().setTagArg("i", 1, "green");
        traciVehicle->changeRoute(payload->getRoadId(), 999.9);

        if (haveForwarded) return;

        auto packet = createPacket("relay");
        packet->insertAtBack(payload);
        sendPacket(std::move(packet));

        haveForwarded = true;
}
```

//In simulations folder:

omnetpp.ini:

```
[General]
network = Scenario
sim-time-limit = 25s
debug-on-errors = true
cmdenv-express-mode = true
image-path = ../../../../images

# UDPBasicApp
*.node[*].numApps = 1
*.node[*].app[0].typename =
"prj_1.veins_inet.VeinsInetSampleApplication"
*.node[*].app[0].interface = "wlan0"

# Ieee80211Interface
*.node[*].wlan[0].opMode = "a"
*.node[*].wlan[0].radio.bandName = "5.9 GHz"
*.node[*].wlan[0].radio.channelNumber = 3
*.node[*].wlan[0].radio.transmitter.power = 20mW
*.node[*].wlan[0].radio.bandwidth = 10 MHz

# HostAutoConfigurator
*.node[*].ifConfig.interfaces = "wlan0"
*.node[*].ifConfig.mcastGroups = "224.0.0.1"

# VeinsInetMobility
*.node[*].mobility.typename = "VeinsInetMobility"

# VeinsInetManager
*.manager.updateInterval = 0.1s
*.manager.host = "localhost"
*.manager.port = 9999
*.manager.autoShutdown = true
*.manager.launchConfig = xmldoc("sumoprjt.launchd.xml")
*.manager.moduleType = "prj_1.veins_inet.VeinsInetCar"
**.vector-recording = true
```

CHAPTER 3

3.1 SNAPSHOTS

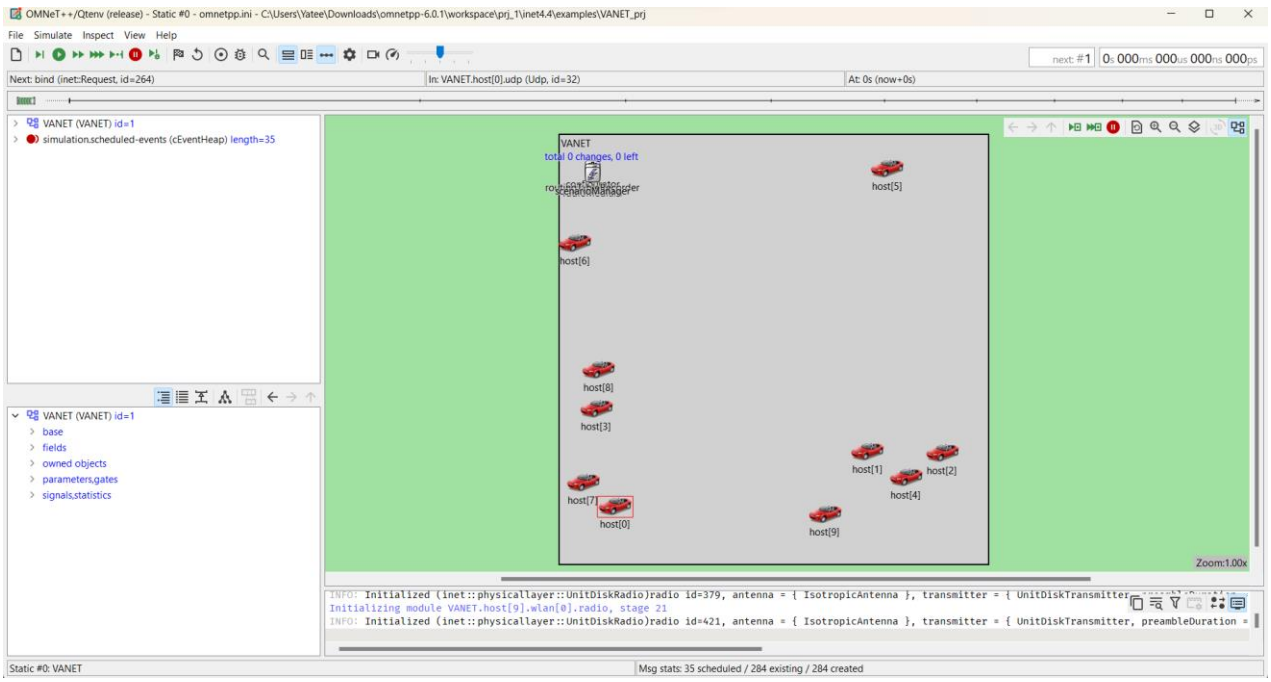


Fig 3.1: Nodes (10 in number) in the simulation setup.

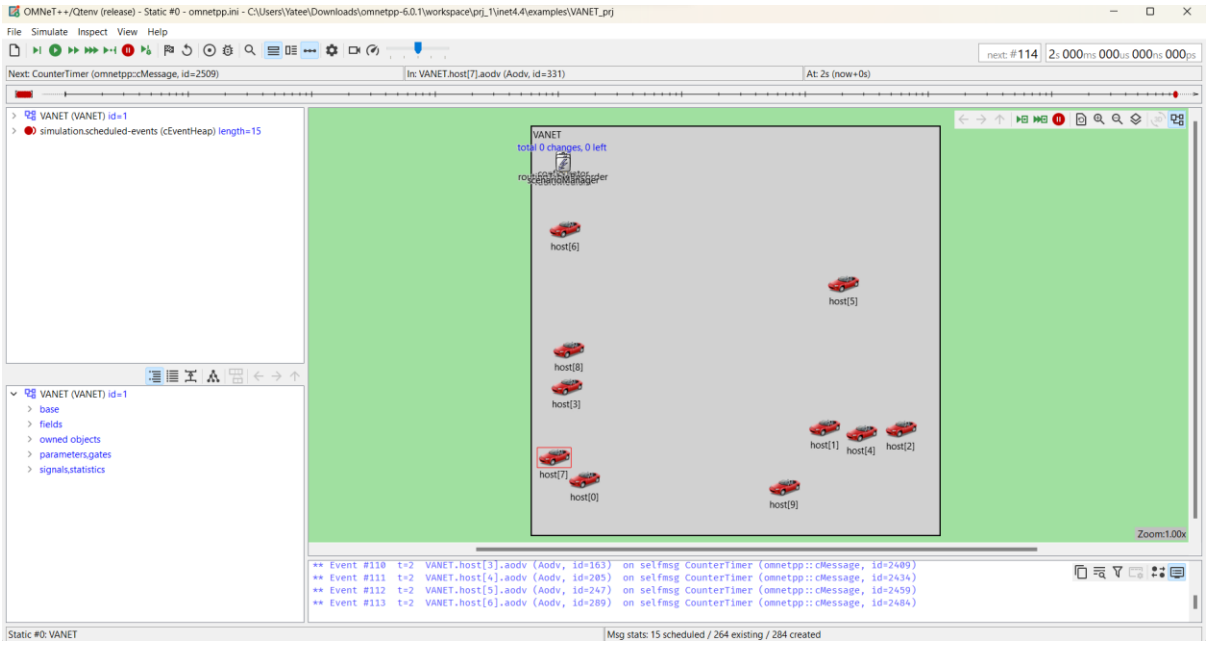


Fig 3.2: Mobile nodes changing their position.

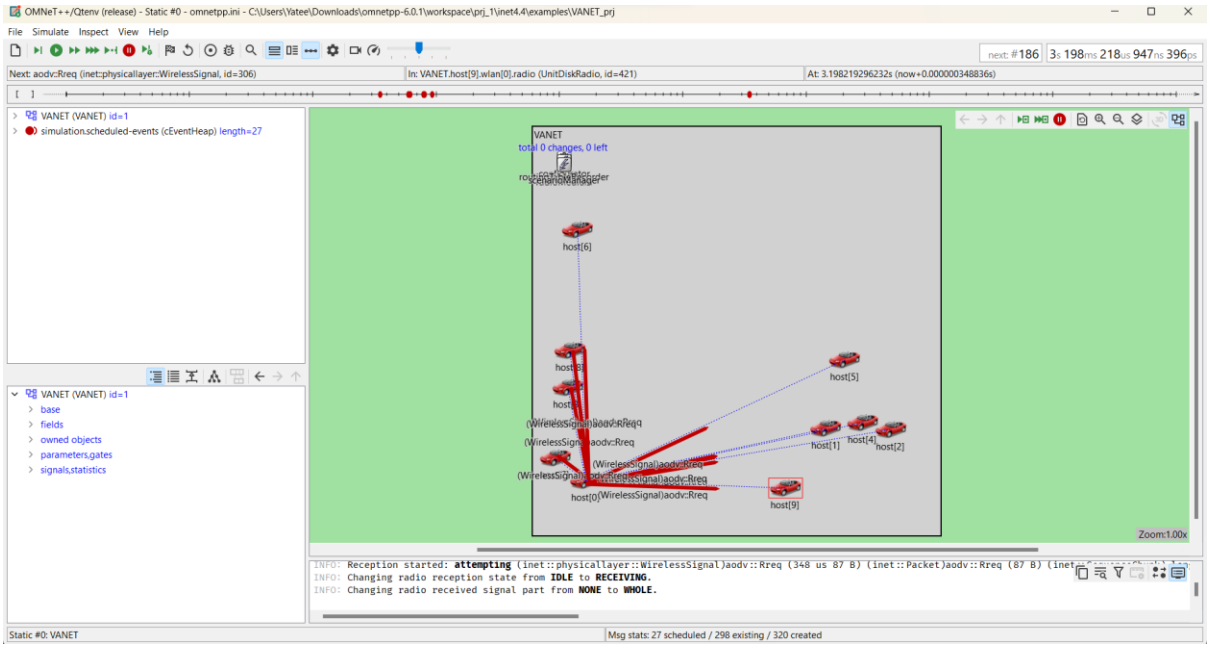


Fig 3.3: Message broadcast from one vehicle to others

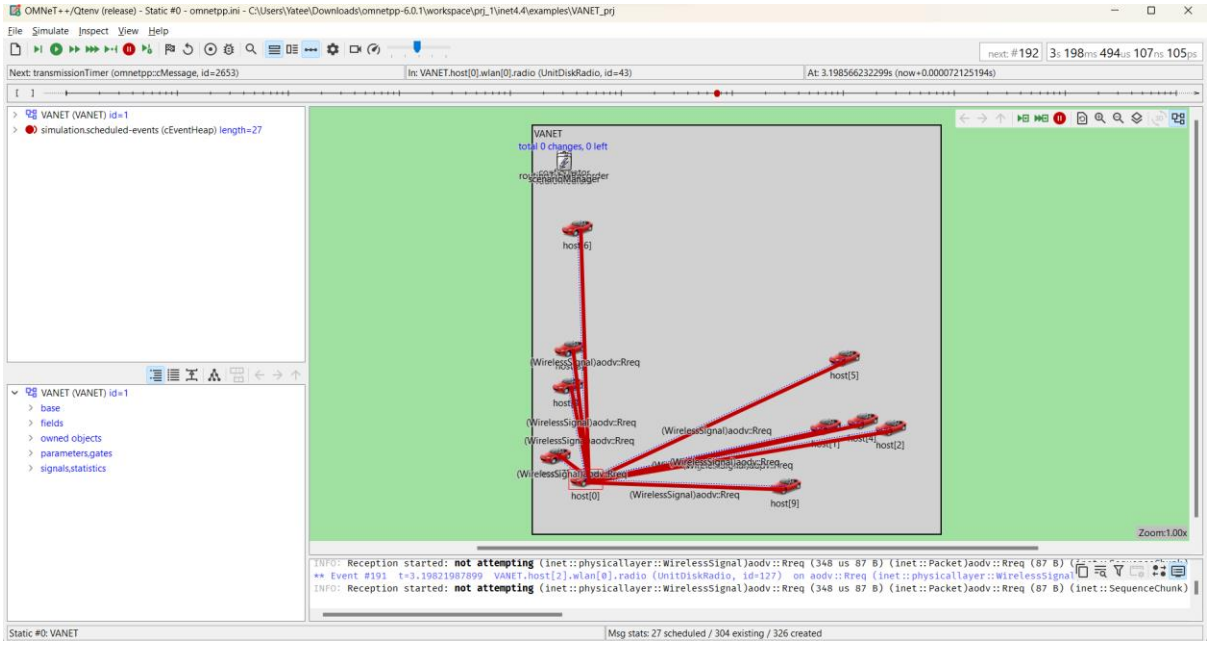


Fig 3.4: Message received by other vehicles

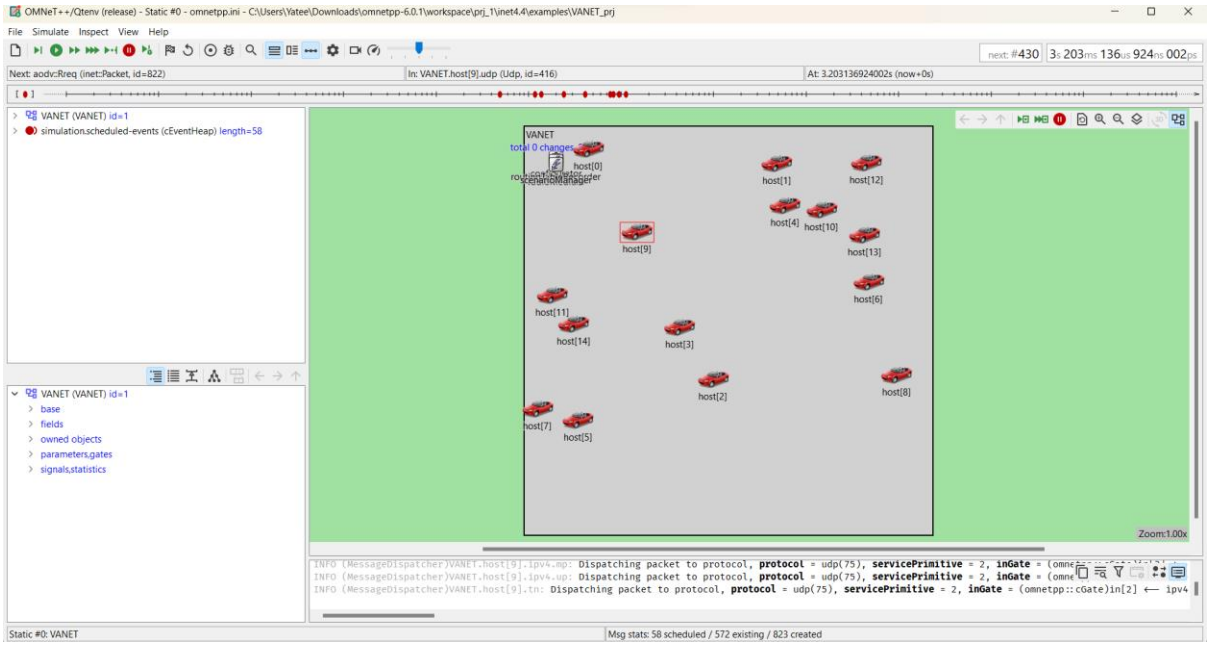


Fig 3.5: Additional nodes

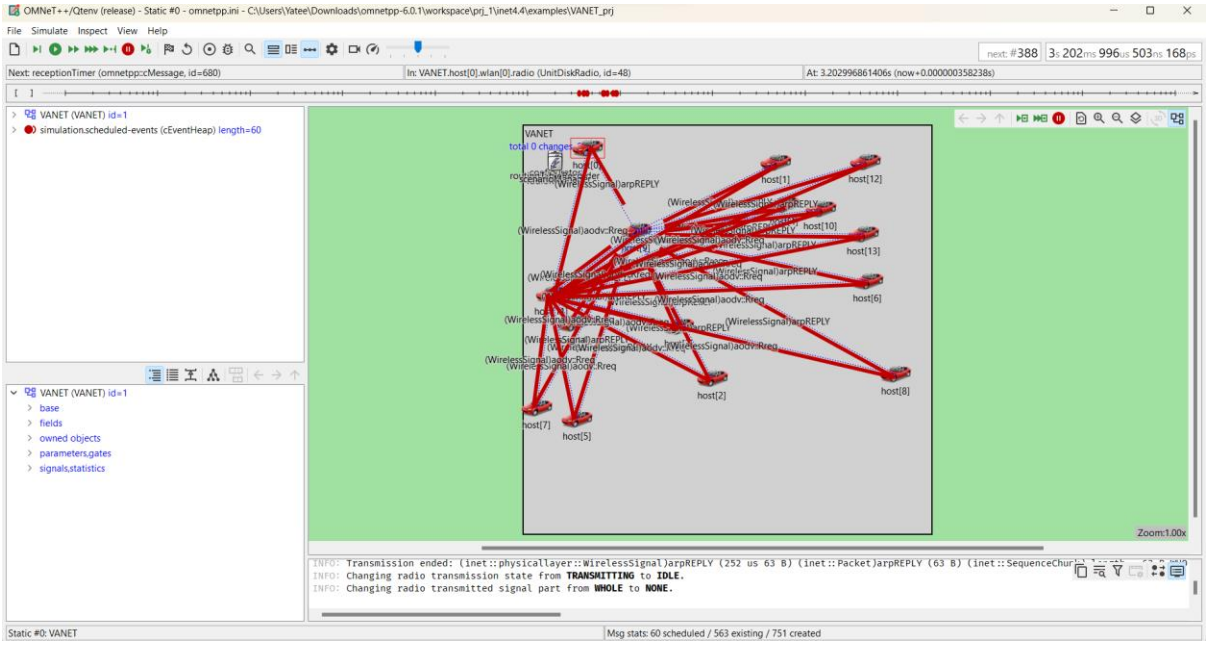


Fig 3.6: Multiple message broadcast among vehicles (1)

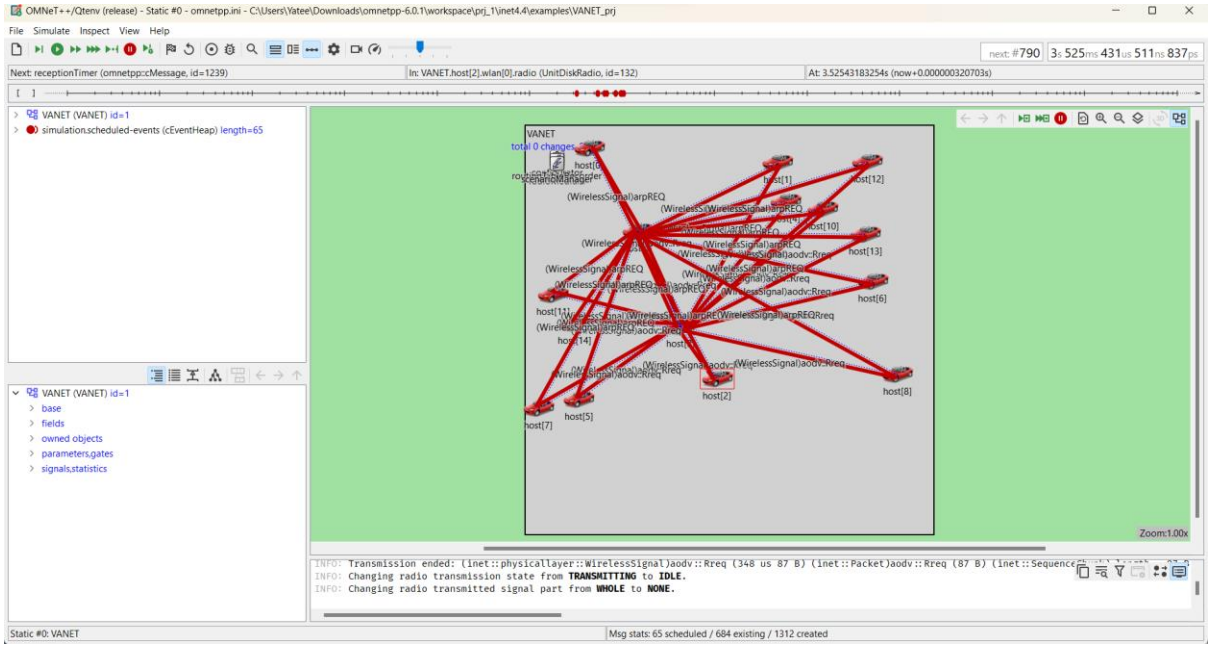
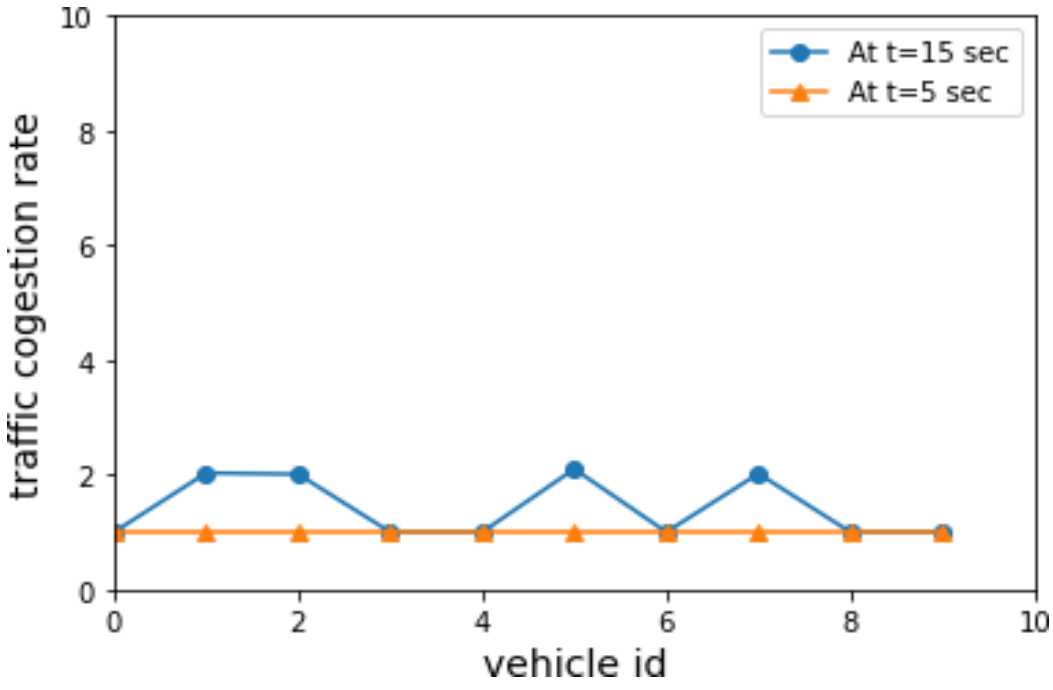


Fig 3.7: Multiple message broadcast among vehicles (2)

3.2 PERFORMANCE EVALUATION



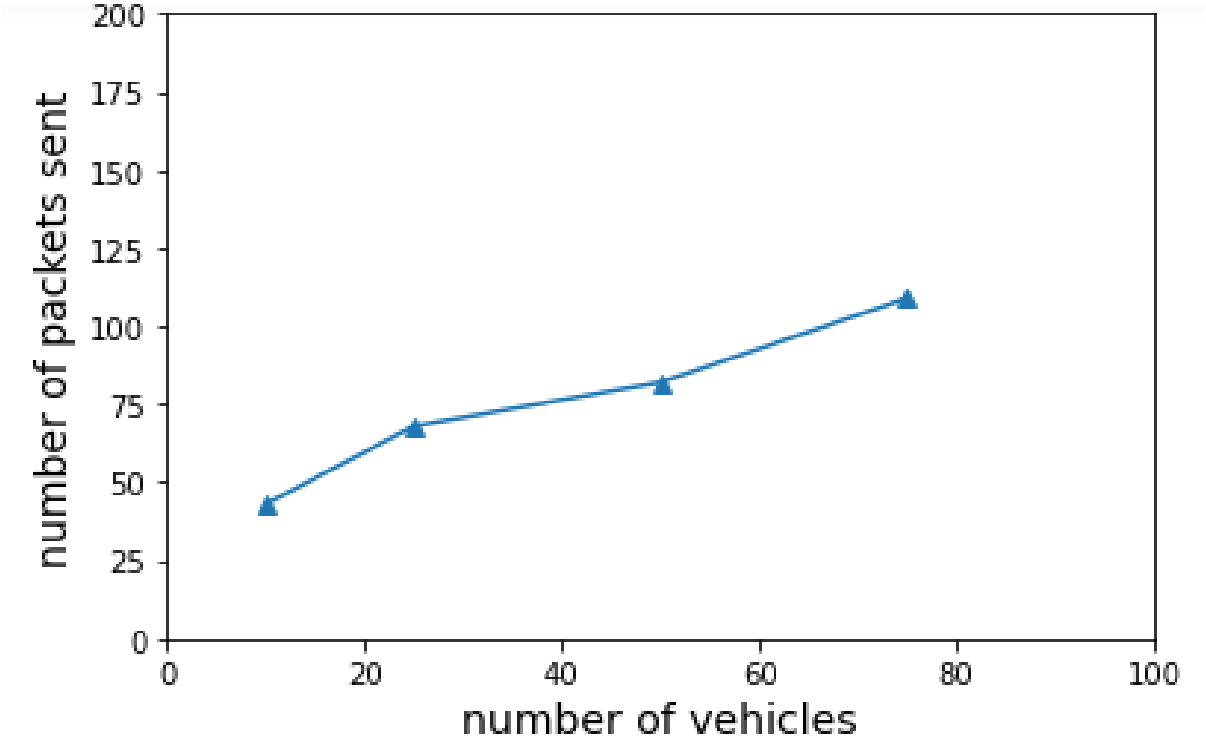
Graph 3.1: graph plotted between vehicle id and its congestion level rate

In the above graph, corresponding traffic congestion level rates are plotted at two different simulation timings.

At t=5 sec, all vehicles congestion level rate is 1 which represents that all the vehicles are in congestion less zones.

At t=20 sec, some of the vehicles started moving towards other vehicles, as a result of which congestion rates increased.

From this we can conclude that, as the vehicles start moving towards their destinations, the traffic congestion increases due to increase in vehicular density around a place.



Graph 3.2: graph plotted between number of vehicles and number of packets sent

In the above graph, number of messages that are broadcasted between the vehicles is plotted.

With an increase in number of vehicles, the number of message broadcasts also increases. This is because, with increase in number of vehicles, there is a high probability for rapid change in congestion level rates of the vehicles.

CHAPTER 4

4.1 CONCLUSION

In this project, we omitted V2I communication and solely depended on V2V communication. This decrease in the use of V2I connectivity allowed us to reduce the cost to establish external units like RSUs. V2V communication implementation is very quick and easy. Since there is no V2I communication, there is no risk in transferring sensitive data such as event-driven messages like accident alerts, and the communication overhead is also minimal. This strategy assisted us in cutting both the amount of gasoline used and the total amount of time that vehicles spent travelling. Consequently, the protocol used in this project is profitable, simple to install, and cost-effective.

4.2 FUTURE WORKS

With this project, we were only able to use a certain number of vehicles and routes. Additionally, we mostly concentrated on the vehicles that are inside the communication range in this project. By establishing the communication outside of the available range, we can extend this work. We can employ clustering techniques, where the cars are separated into clusters, to establish contact with the vehicles that are out of range. One vehicle can serve as the cluster head for each cluster and communicate the level of congestion to neighboring clusters. Additionally, by creating much more efficient algorithms for shortest path discovery between source and destination cars, we can improve this task and minimize the message transmission times.

CHAPTER 5

5.1 REFERENCES

- (Aissaoui, 2014) Aissaoui, R., Menouar, H., Dhraief, A., Filali, F., Belghith, A., & Abu-Dayya, A. (2014, June). Advanced real-time traffic monitoring system based on V2X communications. In *2014 IEEE International Conference on Communications (ICC)* (pp. 2713-2718). IEEE.

- (Akinlade, 2019) Akinlade, O., Saini, I., Liu, X., & Jaekel, A. (2019, May). Traffic Density Based Distributed Congestion Control Strategy for Vehicular Communication.

- (Bouassida,2008) Bouassida, M. S., & Shawky, M. (2008, November). On the congestion control within VANET. In *2008 1st IFIP Wireless Days* (pp. 1-5). IEEE

- (Djahel,2012) Djahel, S., & Ghamri-Doudane, Y. (2012, April). A robust congestion control scheme for fast and reliable dissemination of safety messages in VANETs. In *2012 IEEE Wireless Communications and Networking Conference (WCNC)* (pp. 2264-2269). IEEE.

- (Facchina,2020) Facchina, C. (2020). *Adaptive Transmission Power Level with Vehicle Speed Approximation of Density for VANET Congestion Control* (Doctoral dissertation, University of Windsor (Canada))

- (Jain, 2018) Jain, R. (2018). A congestion control system based on VANET for small length roads. *arXiv preprint arXiv:1801.06448*.

- (Konur, 2011) Konur, S., & Fisher, M. (2011, May). Formal analysis of a VANET congestion control protocol through probabilistic verification. In *2011 IEEE 73rd Vehicular Technology Conference (VTC Spring)* (pp. 1-5). IEEE.

- (Letswamotse,2018) Letswamotse, B. B., Malekian, R., Chen, C. Y., & Modieginyane, K. M. (2018). Software defined wireless sensor networks and efficient congestion control. *IET Networks*, *7*(6), 460-464

- (Noori, 2013) Noori, H., & Valkama, M. (2013, December). Impact of VANET-based V2X communication using IEEE 802.11 p on reducing vehicles traveling time. In *2013 International Conference on Connected Vehicles and Expo (ICCVE)* (pp. 654-661). IEEE.

- (Pan,2016) Pan, J., Popa, I. S., & Borcea, C. (2016). Divert: A distributed vehicular traffic re-routing system for congestion avoidance. *IEEE Transactions on Mobile Computing*, *16*(1), 58-72.

- (Rath,2019) Rath, M., Pati, B., & Pattanayak, B. K. (2019). Mobile agent-based improved traffic control system in VANET. In *Integrated Intelligent Computing, Communication and Security* (pp. 261-269). Springer, Singapore.

- (Sharma,2019) Sharma, S., Chahal, M., & Harit, S. (2019, February). Transmission Rate-based Congestion Control in Vehicular Ad Hoc Networks. In *2019 Amity International Conference on Artificial Intelligence (AICAI)* (pp. 303-307). IEEE

- (Sankaranarayanan,2017) Sankaranarayanan, M., Mala, C., & Mathew, S. (2017, March). Congestion rate estimation for VANET infrastructure using fuzzy logic. In *Proceedings of the 2017 International Conference on Intelligent Systems, Metaheuristics & Swarm Intelligence* (pp. 98-102).

- (Sousa,2019) de Sousa, R. S., Boukerche, A., & Loureiro, A. A. (2019, June). DisTraC: A Distributed and Low-Overhead Protocol for Traffic Congestion Control Using Vehicular Networks. In *2019 IEEE Symposium on Computers and Communications (ISCC)* (pp. 1-6). IEEE.