IETF-112
IPWAVE Hackathon Project

1-5 November 2021

Champion: Jaehoon (Paul) Jeong

Department of Computer Science and Engineering at SKKU

pauljeong@skku.edu

I E T F

IP Wireless Access in Vehicular Environments (IPWAVE) Basic Protocols Project

Champion: Jaehoon (Paul) Jeong (SKKU)

IETF-112 IPWAVE Hackathon Project:
Context-Aware Navigator Protocol (CNP)

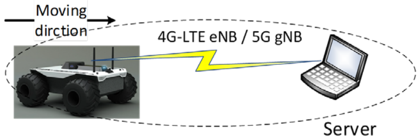
Professors:

- **Jaehoon (Paul) Jeong (SKKU)**
- **Younghan Kim (SSU)**

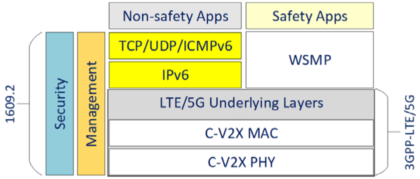
Students:

- **Bien Aime Mugabarigira (SKKU)**
- **Junhee Kwon (SKKU)**
- **Yiwen (Chris) Shen (SKKU)**
- **Hyeonah Jeong (SKKU)**
- **Gilteun Choi (PNU)**

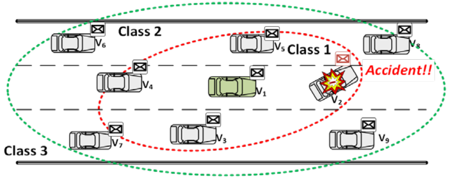
Implementation Setup



Protocol Stack



IPv6 ND Option



IPv6 ND with Cooperation Context Message (CCM)

IPv6 ND with Emergency Context Message (ECM)

Objectives

- To Demonstrate IPWAVE Basic Protocols
- To Send vehicle mobility information option to server
- Simulation of Context-Aware Navigation Protocol over C-V2X
- To Discover technology gaps for IPWAVE

Where to get source code:

- GitHub: https://github.com/ipwave-hackathon-ietf

How to set up an environment:

(a) Implementation

- Hardware

- Robot car (AION robotics R1) and Desktop (AWS server)

- Software

- OS: Ubuntu 18.04.5 LTS, ROS-melodic
- AWS linux remote server: ubuntu 20.04.2 LTS

(b) Simulation

- Software

- OS: Ubuntu 16.04
- Others: OpenCV2X, SUMO 1.0.0, OMNeT++ 5.4.1, GNU GCC7.3, Veins 5.0 and INET 3.6.6

Implementation Contents:

- To Support IETF Vehicular Mobility Information (VMI) Option in IPv6-based vehicular networks over C-V2X.
  - ✓ VMI: https://datatracker.ietf.org/doc/draft-jeong-ipwave-context-aware-navigator/
- To Develop a vehicular communication system for safe and secure driving using IETF IPWAVE protocols and W3C Vehicle Information Service Specification (VISS)
  - ✓ Core: https://www.w3.org/TR/viss2-core/
  - ✓ Transport: https://www.w3.org/TR/viss2-transport/

성균관대학교 SUNG KYUN KWAN UNIVERSITY

Hackathon Plan

- **Part 1: Simulation**
  - To test the applicability of IPWAVE protocols in C-V2X
  - To simulate Context-Aware Navigation Protocol (CNP) with C-V2X

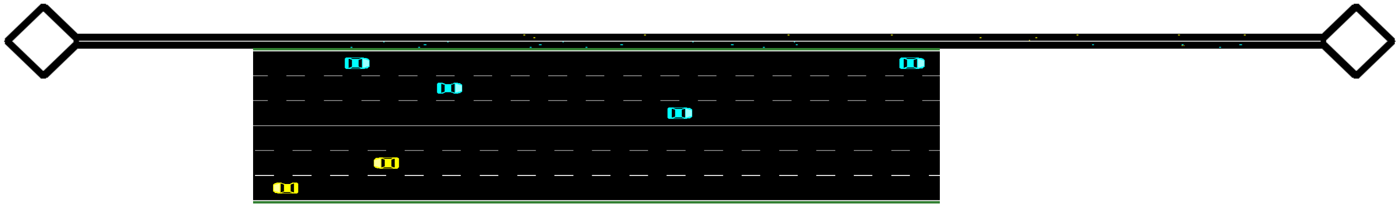
- **Part 2: Robot Car Implementation**
  - To test vehicle communication standards (i.e., VISS and VSS) using Robot Car (i.e., AION Robot R1)
    - ✓ VISS: W3C Vehicle Information Service Specification
    - ✓ VSS: GENIVI Vehicle Signal Specification

What got done (1/3)

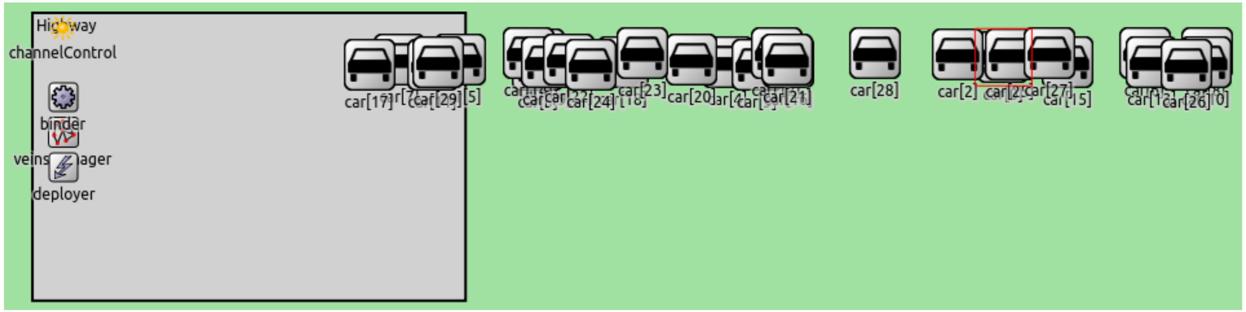
- Simulation implementation of IPWAVE Context-Aware Navigation Protocol (CNP) with C-V2X
  - Adaptation of Cooperation Context Message (CCM) by V2V within the Mode4 application.
  - Exchange of Emergency Context Message (ECM) with higher priority over CCM within AODV-based ad hoc routing.
- Implementation of IPWAVE Draft and W3C Standard on Robot Car
  - We applied both IPWAVE CNP draft and W3C standard (VISS) to extend its scope from server(car)-client scheme to V2I scheme.
  - Communication format and data structure of VISS are applied.

What got done (2/3)

- Simulation Environment:
  - A SUMO highway scenario road network with 3 lanes is used.

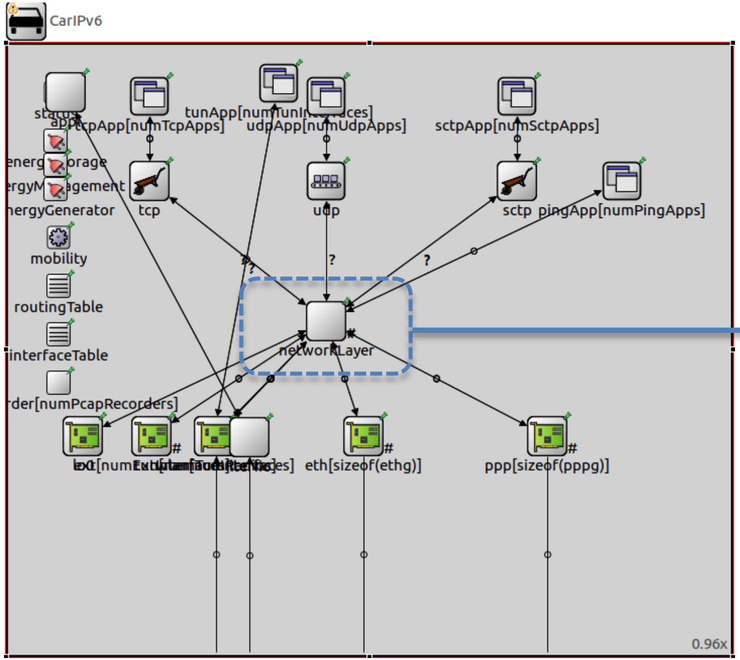


  - Running vehicles move in C-V2X vehicular network in OMNeT++.

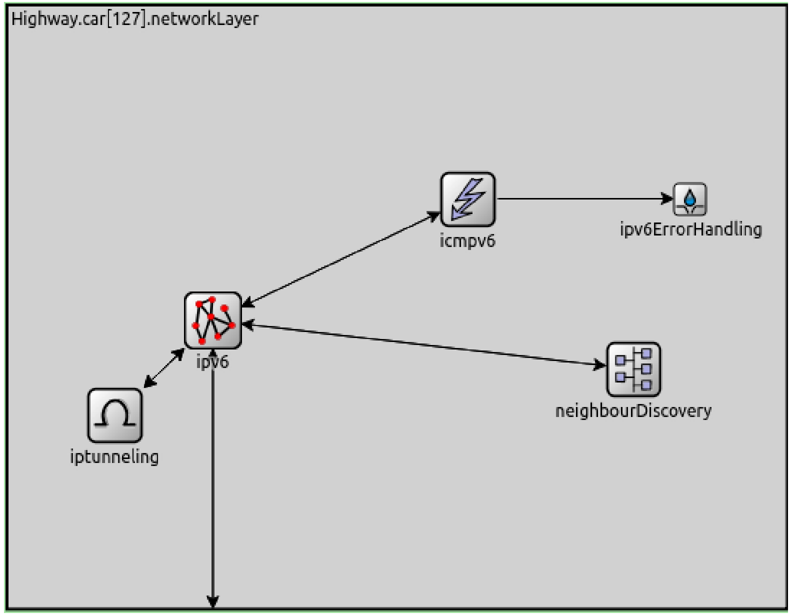


What got done (3/3)

- IPv6 Car Simulation:



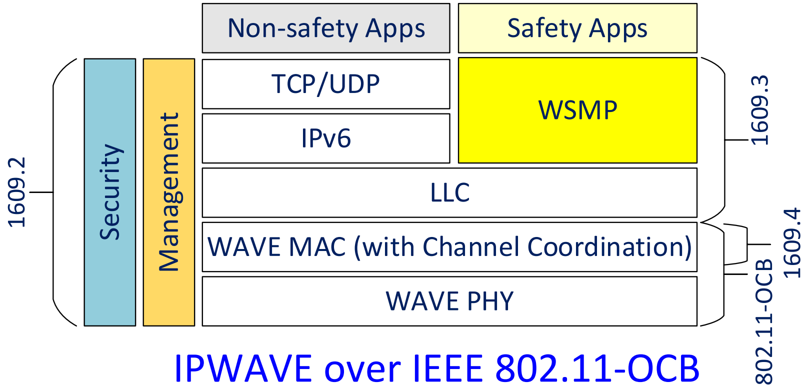
Vehicle structure



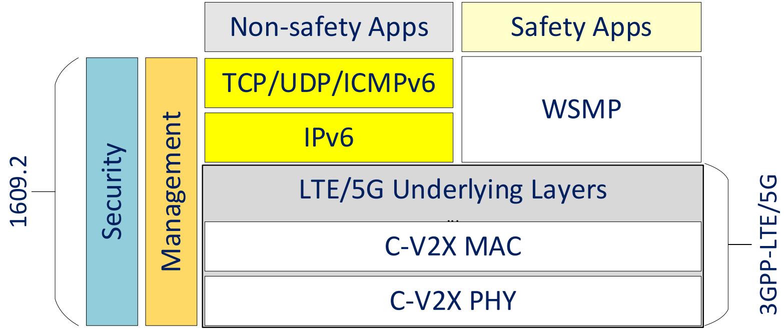
IPv6 network layer on top of 3GPP LTE/5G

What we learned (1/2)

- From IEEE 802.11-OCB-based IPWAVE to C-V2X-based IPWAVE



IPWAVE over IEEE 802.11-OCB

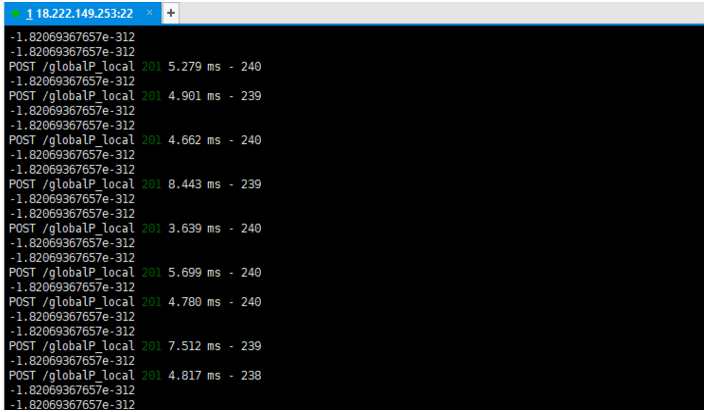


IPWAVE over C-V2X

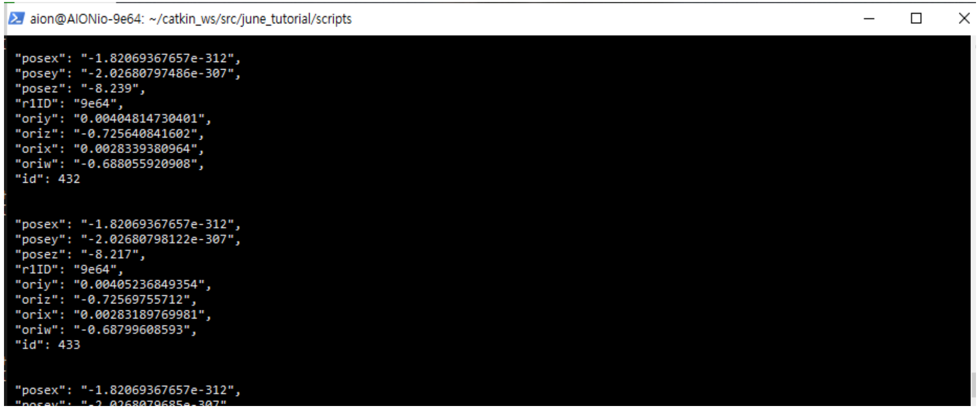
- We tested the Cooperation Context Message (CCM) and Emergency Context Message (ECM) for CNP application in 3GPP-LTE mode 4.
- Although the vehicular protocol stack has been defined with IEEE 802.11-OCB, it can be adapted to the C-V2X access layer.

What we learned (2/2)

- We implemented an AWS remote server and R1 program to send its Vehicle Mobility Information (VMI) to the server.



Remote Server

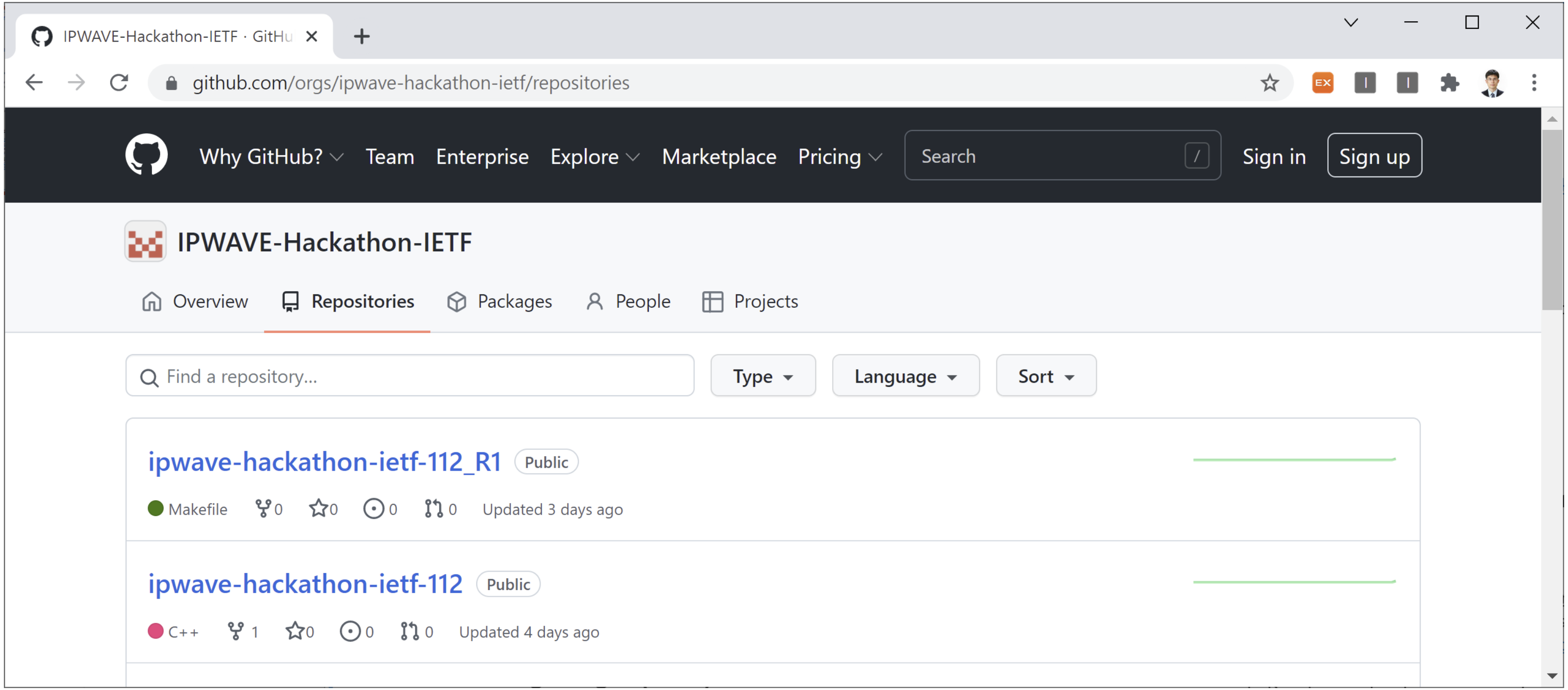


Vehicle

- We applied the W3C standard (i.e., VISS) to wireless communication.
- Cars from various manufactures can have different specifications or sensors, so we learned that GENIVI VSS standard should consider such variables.

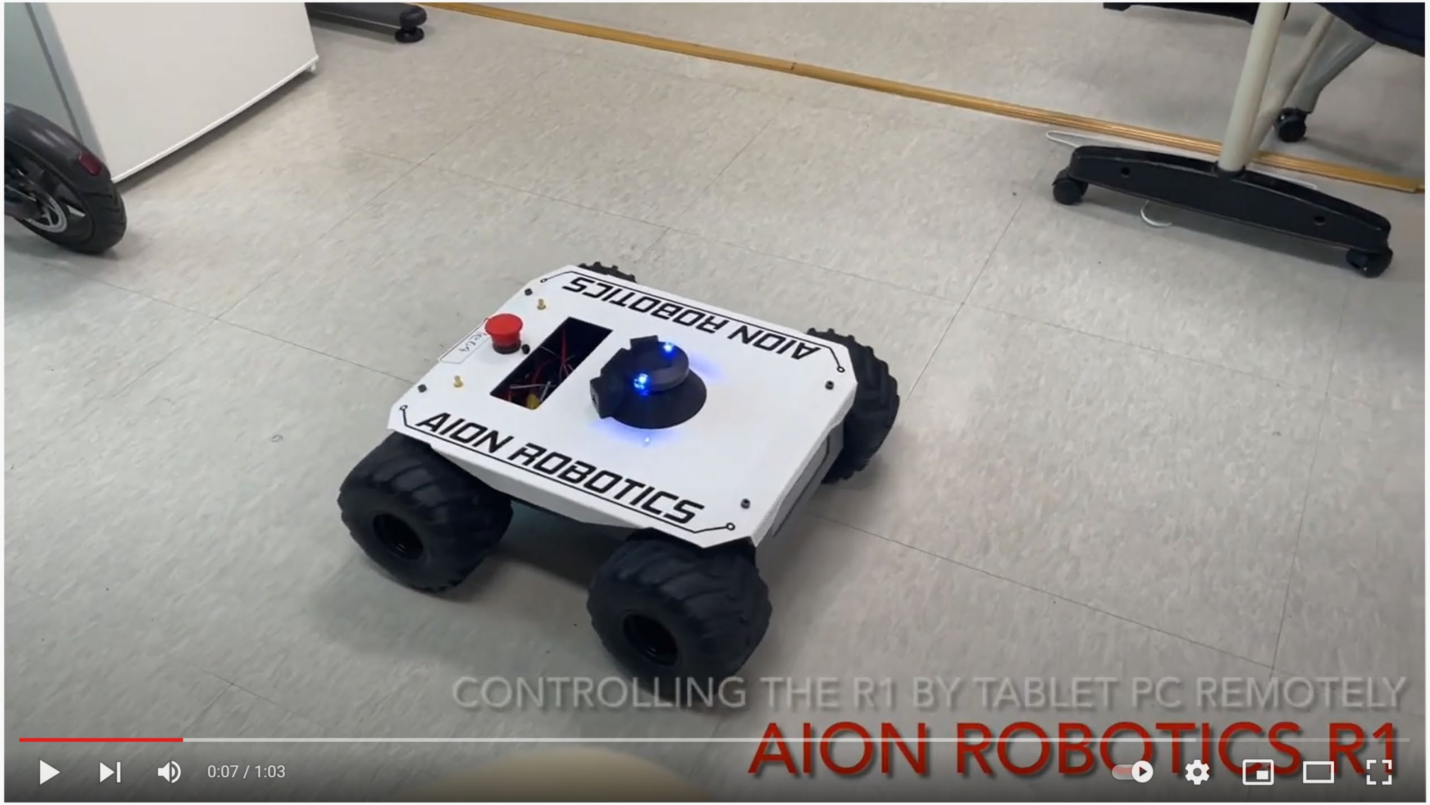
Open Source Project at GitHub

URL: https://github.com/ipwave-hackathon-ietf/ipwave-hackathon-ietf-112
URL: https://github.com/ipwave-hackathon-ietf/ipwave-hackathon-ietf-112_R1



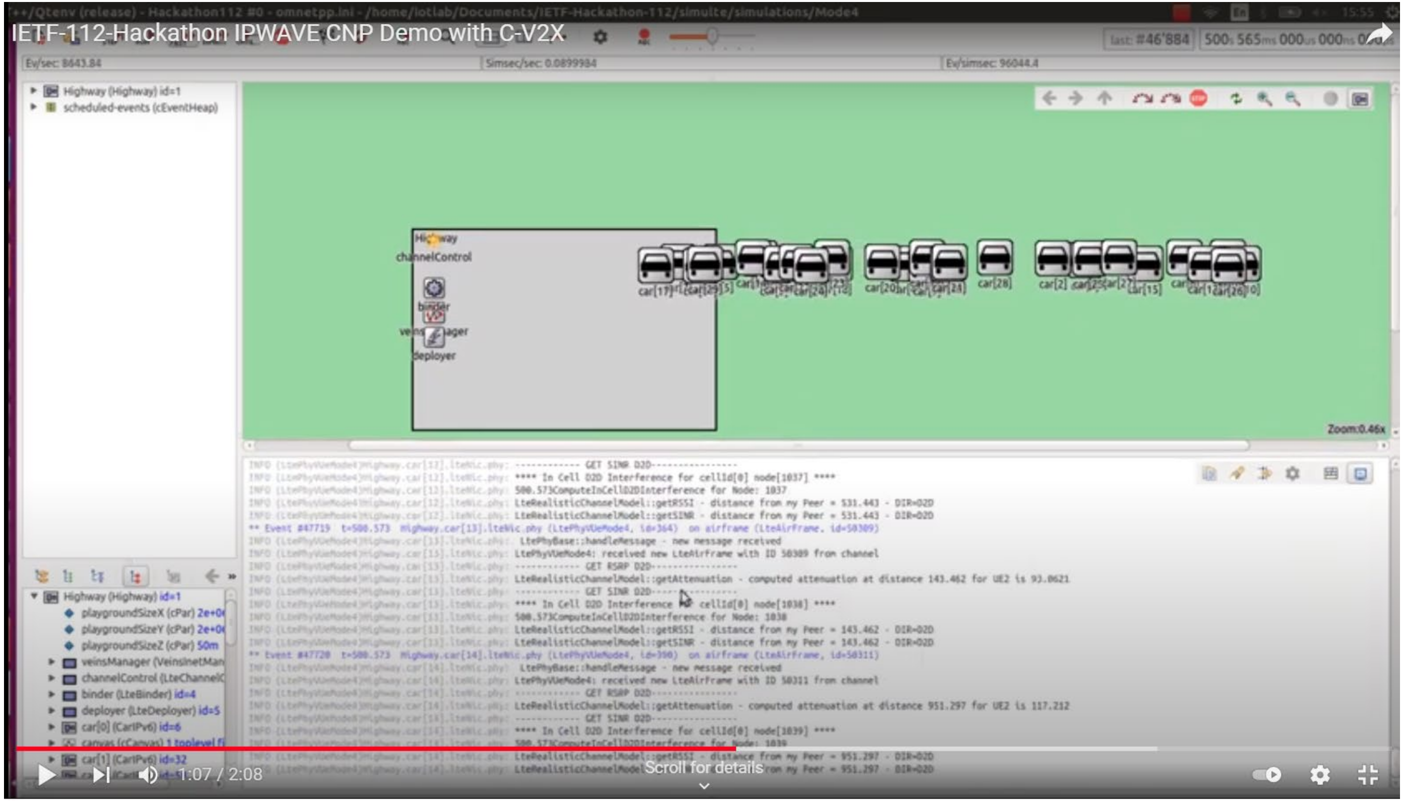
Demonstration Video Clip at YouTube (1/2)

URL: https://www.youtube.com/watch?v=c4HZw88u6Fs



Demonstration Video Clip at YouTube (2/2)

URL: https://youtu.be/1ZzeXCOA1Ww



Wrap Up (1/2)

- IETF Hackathon-112 introduction meeting on Monday 11/01/2021
- IPWAVE Team worked with cooperation with I2NSF and BMWG Teams.

Wrap Up (2/2)

Professors:
- Jaehoon (Paul) Jeong (SKKU)
- Younghan Kim (SSU)

Team members:
- Bien Aime Mugabarigira (SKKU)
- Junhee Kwon (SKKU)
- Yiwen (Chris) Shen (SKKU)
- Hyeonah Jeong (SKKU)
- Gilteun Choi (PNU)

Hackathon Team Photo

IETF-112 Hackathon Korea Teams

Sponsors

NRF National Research Foundation of Korea

IITP Institute of Information & Communications Technology Planning & Evaluation

TTA 한국정보통신기술협회 Telecommunications Technology Association

SUNGKYUNKWAN UNIVERSITY 1398 성균관대학교

숭실대학교 Soongsil University

ETRI 한 국 전 자 통 신 연 구 원 Electronics and Telecommunications Research Institute

K-IETF KOREA IETF FORUM

Appendix 1 for C-V2X Simulation

(1) Simulation Environments

(2) Configurations in OMNeT++ and SUMO

(3) Project References in OMNeT++ and SUMO

(4) Reference to Original Open C-V2X

Simulation Environments

- OS: Ubuntu 16.04

- Simulators:
  - SUMO 1.0.0
  - OMNeT++ 5.4.1
- GNU GCC 7.3

- Open Sources:
  - https://github.com/ipwave-hackathon-ietf/ipwave-hackathon-ietf-112
    - Based on OpenCV2X
  - Veins 5.0
  - INET 3.6.6

Configurations in OMNeT++ and SUMO

- Install OMNeT++ by following the procedure in the installation manual: https://doc.omnetpp.org/omnetpp/InstallGuide.pdf
- Install proper SUMO version
- Import projects in OMNeT++ workspace
  - Import INET by

–File → Import → General → Existing projects into workspace

  - Similarly, as INET, import SimuLTE
  - Import veins:

»Specifically, search for nested project and install both veins and veins_inet3 projects.

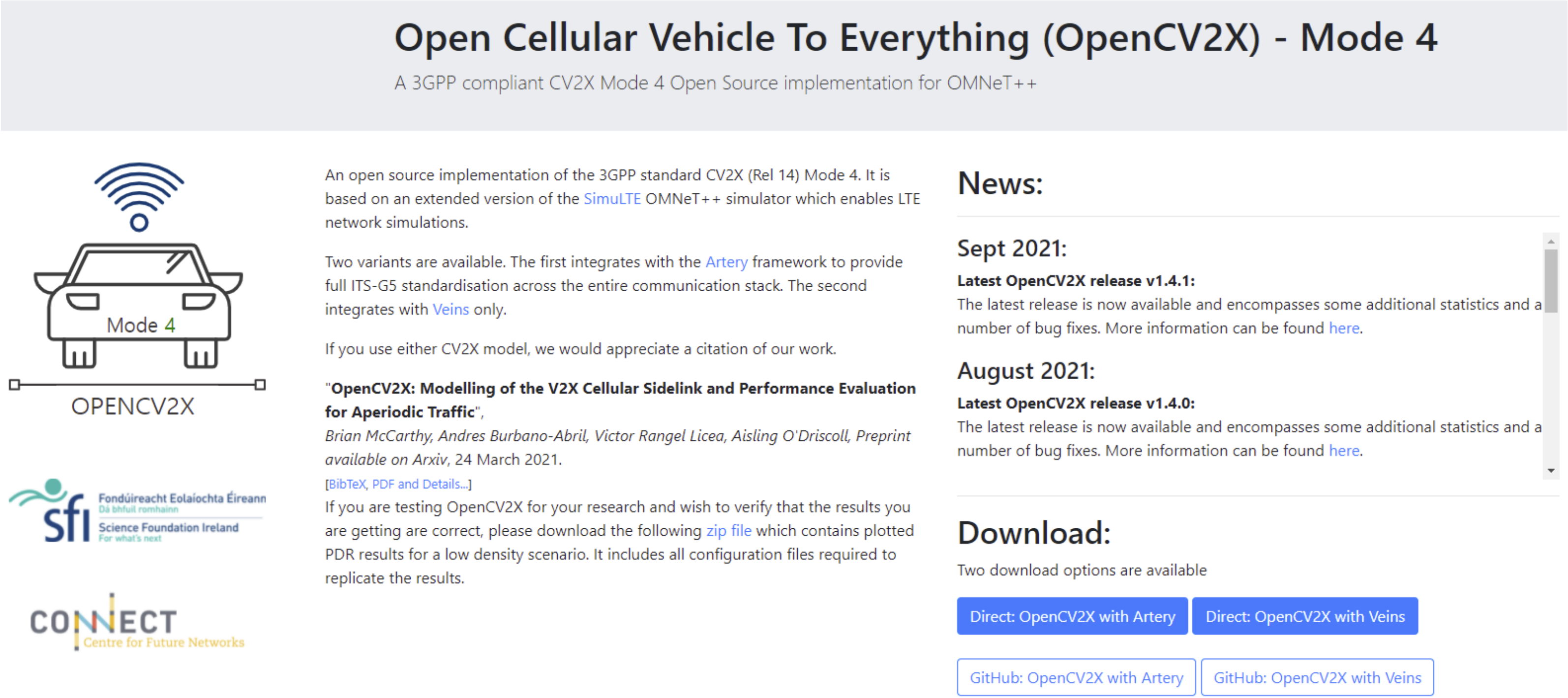
Project References in OMNeT++ and SUMO

- Activate project features to ensure SimuLTE runs correctly.
- Right-click on lte project and choose Properties
- Then, Project References and tick inet, veins and veins_inet3
- Run the scenario in veins:
- *python2 sumo-launchd.py*
- Run the simulation by:

– *lte → simulations → mode4 → omnetpp and in set inifile configuration, choose Hachathon112*

Reference to Original Open C-V2X

- URL: http://www.cs.ucc.ie/cv2x/media/OpenCV2X_Documentation.pdf



Open Cellular Vehicle To Everything (OpenCV2X) - Mode 4

A 3GPP compliant CV2X Mode 4 Open Source implementation for OMNeT++

Mode 4

OPENCV2X

An open source implementation of the 3GPP standard CV2X (Rel 14) Mode 4. It is based on an extended version of the SimuLTE OMNeT++ simulator which enables LTE network simulations.

Two variants are available. The first integrates with the Artery framework to provide full ITS-G5 standardisation across the entire communication stack. The second integrates with Veins only.

If you use either CV2X model, we would appreciate a citation of our work.

"**OpenCV2X: Modelling of the V2X Cellular Sidelink and Performance Evaluation for Aperiodic Traffic**",
Brian McCarthy, Andres Burbano-Abril, Victor Rangel Licea, Aisling O'Driscoll, Preprint available on Arxiv, 24 March 2021.
[BibTeX, PDF and Details...]

If you are testing OpenCV2X for your research and wish to verify that the results you are getting are correct, please download the following zip file which contains plotted PDR results for a low density scenario. It includes all configuration files required to replicate the results.

Fondúireacht Eolaíochta Éireann
Dá bhfuil romhainn
Science Foundation Ireland
For what's next

CONNECT
Centre for Future Networks

News:

Sept 2021:

Latest OpenCV2X release v1.4.1:
The latest release is now available and encompasses some additional statistics and a number of bug fixes. More information can be found here.

August 2021:

Latest OpenCV2X release v1.4.0:
The latest release is now available and encompasses some additional statistics and a number of bug fixes. More information can be found here.

Download:

Two download options are available

Direct: OpenCV2X with Artery | Direct: OpenCV2X with Veins

GitHub: OpenCV2X with Artery | GitHub: OpenCV2X with Veins

Appendix 2 for Robot Car Implementation

(1) Implementation Environments

(2) Code for Server and AION R1 Robot Car

Implementation Environments

- AWS linux remote server OS: Ubuntu 20.04.2 LTS
- R1’s OS : Ubuntu 18.04.5 LTS
- ROS : melodic

- Open Sources:
  - https://github.com/ipwave-hackathon-ietf/ipwave-hackathon-ietf-112_R1
  - http://wiki.ros.org/

Code for Server and AION R1 Robot Car

```javascript
const jsonServer=require('json-server');
const server=jsonServer.create();
const path=require('path');
const router=jsonServer.router(path.join(__dirname,'db.json'));
const middlewares=jsonServer.defaults();

server.use(middlewares);

server.use(jsonServer.bodyParser);

server.use(router);

let port=5056;
server.listen(port,()=>{
        console.log(`JSON server is running on port ${port}`)
})

/////////////////////////////////////////////////////////////

const dgram=require('dgram');
const server_ws = dgram.createSocket('udp4');

server_ws.on('error', (err) => {
        console.log(`server error:\n${err.stack}`);
        server_ws.close();
});

server_ws.on('message', (msg, rinfo) => {
        const json=JSON.parse(msg.toString());
        console.log("restAPI");
        console.log(json["value"]);
});

server_ws.on('listening', () => {
        const address = server_ws.address();
        console.log(`server listening ${address.address}:${address.port}`);
});

server_ws.bind(9998);
```

AWS remote server receiving Robot car message

```python
def globalP_local_callback(data):

        newItem={
                "r1ID": "9e64",
                "value":str(
                    {"posex":data.pose.pose.position.x,
                    "posey":data.pose.pose.position.y,
                    "posez":data.pose.pose.position.z,
                    "orix":data.pose.pose.orientation.x,
                    "oriy":data.pose.pose.orientation.y,
                    "oriz":data.pose.pose.orientation.z,
                    "oriw":data.pose.pose.orientation.w}
                    ),
                "timestamp":str(get_now())
                }

    serverAddr=("18.222.149.253",9998)
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    client_socket.sendto(json.dumps(newItem),serverAddr)

    url_items="http://18.222.149.253:5056/globalP_local"
    print("\nSending globalPosition_local")
    print(newItem)
    response=requests.post(url_items, data=newItem)


def listener():
        rospy.init_node('listener', anonymous=True)
        rospy.Subscriber("/mavros/global_position/local",Odometry,globalP_local_callback)
        rospy.spin()

if __name__ == '__main__':
    listener()
```

Listener on Robot car's onboard computer