Lab 4: Cellular Vehicle-to-Everything (C-V2X)

Introduction

C-V2X is rapidly growing in popularity and usage around the world. Several auto manufacturers (e.g., Ford, Cadillac, GM) have announced intentions to deploy C-V2X on their models within the next five years, and C-V2X forms the backbone of China's nation-wide V2X deployment that is currently underway. While DSRC remains popular in Europe and Japan, most industry and academic authorities believe C-V2X (particularly augmented with 5G technology) will be the primary V2V technology of the future. In this lab, you will take a closer look at C-V2X from a low-level perspective and explore some of the properties that make C-V2X such a desirable protocol. You will particularly focus on the physical (PHY) and MAC layers of C-V2X, as they differ substantially from DSRC. This lab makes use of both the V2Verifier testbed and srsRAN, an open-source cellular project that V2Verifier makes partial use of.

This lab particularly focuses on LTE-V2X, the current generation of C-V2X technology. While 5G C-V2X (NR-V2X) equipment is not yet on the market, many of the physical- (PHY) and MAC-layer components of NR-V2X evolved from LTE-V2X; thus, studying LTE-V2X in detail will provide you with an understanding of the core elements of both protocols.

Required Equipment

- 2x USRP B210 SDRs
    - 2x TXCO GPSDO module (installed in B210s)
    - 2x 3V GPS antennas
    - 2x 5V DC power adapter for USRP
- 2x antennas capable of operation on 5.9 GHz
- 2x PCs (or virtual machines) with USB 3.0 ports
- 1x LimeSDR
- (Optional) 1x Cohda MK6C EVK

Operating System

This lab is designed to be used with Ubuntu 18.04. These instructions may not work properly with older or newer versions. An Ubuntu 18.04 ISO is available at http://releases.ubuntu.com/18.04/. Instructions for installing Ubuntu on a PC are available here. Although a PC is recommended, if using VMs you may refer to Appendix 1 for Ubuntu installation instructions.

Activity 1 – Install the Operating System

Make sure you complete this activity on two PCs**.** If you are using VMs[1], you may choose to create one VM and then clone it for use on a separate PC (to save time).

1. Install Ubuntu 18.04 LTS. You can get an ISO from https://releases.ubuntu.com/18.04/.

    - If you are using a PC, follow the Ubuntu installation instructions.

    - If you are using a VM:

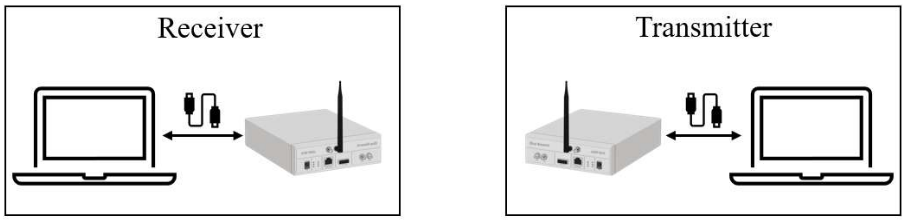
        i. Create a new **Ubuntu 18.04 LTS** virtual machine using VMware Workstation[2]. Do not attempt to download or use an existing Ubuntu VM, you **must** create a new VM for these instructions to work properly.

        ii. Give the VM a minimum of 4GB memory and 20GB storage and leave the remaining installation settings as their default values. Once installed, wait for the OS to automatically update, then reboot.

2. Run `sudo apt update` and `sudo apt -y upgrade` to make sure all installed software packages are up to date. Do not upgrade to a newer Ubuntu version if you are prompted to do so. When the update process completes, restart the PCs (or VMs) and proceed to the next activity.

Activity 2 – Installing the V2Verifier Testbed

This lab assumes that you have completed Lab 1 and either already have, or are already familiar with, the necessary steps to set up and configure the V2Verifier testbed for use with C-V2X. If you have not completed Lab 1, previously used DSRC, or just need a refresher, refer to Activity 2 in Lab 1 for detailed setup instructions. **Make sure to set up two (2) PCs** for this lab. PCs will be referred to according to the diagram below as receiver and transmitter in the remainder of this lab.



[1] This lab requires installing the complete srsRAN project, a task which may require up to several hours if executed on a virtual machine. We strongly encourage you to use real PCs rather than VMs; however, if you must use VMs then we recommend spreading this lab over at least two days in order to allow this installation to complete between lab sessions.
[2] If VMware Workstation is unavailable, we recommend using VirtualBox as a (free) alternative.

Activity 3 – Installing srsRAN with srsGUI Support

Although V2Verifier incorporates code from the srsRAN project to support its C-V2X implementation, some useful features are missing. Therefore, it is desirable to install the actual srsRAN project to support these additional features. You will also install srsGUI, the graphical interface for srsRAN, in order to view some interesting features of C-V2X (i.e., the constellation diagrams for C-V2X channels). **Complete this activity <u>only</u> on the receiver PC**.

1. Install the required libraries for srsGUI

```
sudo apt install -y libboost-system-dev libboost-test-dev libboost-
thread-dev libqwt-qt5-dev qtbase5-dev
```

2. Download and build srsGUI

```
cd ~
git clone https://github.com/srsLTE/srsGUI.git
cd srsGUI
mkdir build
cd build
cmake ../
make
sudo make install
sudo ldconfig
```

3. With srsGUI installed, you can proceed to install srsRAN. First, install the necessary libraries.

```
sudo apt install -y build-essential cmake libfftw3-dev libmbedtls-
dev libboost-program-options-dev libconfig++-dev libsctp-dev
```

4. Now, download and build the srsRAN project. This may take up to 20 minutes to complete, so be patient!

```
cd ~
git clone https://github.com/srsRAN/srsRAN.git
cd srsRAN
mkdir build
cd build
cmake ../
make
make test
sudo make install
sudo ldconfig
```

Activity 4 – Exploring the C-V2X PHY Layer

1. Ensure that both USRPs are connected to their PCs (transmitter and receiver) and synchronized with GPS.

2. On the receiver PC, `cd` to `~/srsRAN/build/lib/examples`

3. Launch the srsRAN C-V2X receiver with the `-v` option for verbose output

   ```
   ./pssch_ue -a clock=gpsdo -v
   ```

   If you followed the instructions about and installed srsGUI, you should see console output as well as a srsGUI window displaying two plots labeled PSSCH and PSCCH. Recall that these are the data and control channels, respectively, for C-V2X.

4. On the transmitter PC, `cd` to `~/v2verifier/cv2x`

5. Launch the V2Verifier C-V2X transmitter.

   ```
   ./build/cv2x_traffic_generator
   ```

6. On the receiver PC, you should see console output as well as some visual activity in the srsGUI window. Answer the following questions:

   - In the srsGUI window, what do you see in the PSSCH and PSCCH windows? (Hint: think back to some of your background in wireless communication, specifically regarding modulation schemes).

   - In the srsRAN terminal (console) output, you should also see quite a bit of activity. There are several fields (e.g., RRI, RTX) for each incoming C-V2X message. List those fields in your report, as well as what you think the information provided in those fields means. Google may be your friend here if you encounter an acronym that you are unfamiliar with!

   - Based on what you observed and (maybe) researched in order to answer the previous question, what do you think are the most significant and/or most useful features of the C-V2X PHY layer that are not present in DSRC? Consult the lecture slides and previous labs as necessary to answer this question.

7. In the terminal window on the receiver PC, stop srsRAN with Ctrl+C (you may need to hit this a few times). srsRAN automatically creates a packet capture file in `/tmp/ue.pcap.` Open this file in Wireshark.

   - If you receive a warning similar to "undefined encapsulation protocol," you need to tell Wireshark how to read the captured packets. Go to Edit → Preferences → Protocols → DLT User. Click "Edit" and add a new parser with the protocol for DLT=147 set to "mac-lteframed" and then save and close this window. Wireshark should now be able to read the capture file.

- Look at the PHY and MAC fields for one of the captured LTE-V2X packets. What do you see that is different from previous (DSRC) capture files that you have viewed? Does any of the visible information support or contradict your previous conclusions about the C-V2X PHY layer. Make note of any relevant observations for your report.

- Do you see any security features (e.g., a signature) in this packet? Remember, this is just a C-V2X packet without 1609.2 security. Based on your answer, do you think C-V2X could benefit from 1609.2 security in a similar manner to DSRC?

Activity 5 – (Optional, if equipment available) Commercial C-V2X Equipment

This activity is optional and depends on whether you have access to a Cohda MK6C device. Note that links below to the Cohda support site may only be accessible to you if you have purchased a MK6C device and consequently been provided with a support account by Cohda Wireless. Unfortunately, due to restrictions on sharing proprietary material, we are unable to share those instructions here.

1. On the transmitter PC, connect to a Cohda MK6C device. Instructions on how to do this are available from the Cohda Wireless support site. Ensure that the Cohda device obtains GPS lock from the same source as your USRPs.

2. On the Cohda device, install and launch the example application for transmitting BSMs with 1609.2 security enabled. Again, instructions on how to do this are available on the Cohda support site.

8. On the receiver PC, launch the srsRAN C-V2X receiver with the `-v` option for verbose output and the `-w` option to disable srsGUI plots (for faster processing).

```
./pssch_ue -a clock=gpsdo -v -w
```

3. After capturing several messages from the Cohda device, stop the srsRAN receiver with Ctrl+C.

4. Open the packet capture file at `/tmp/ue.pcap`. Use the contents to answer the following questions:

   - As commercial devices, the MK6C units are equipped with 1609.2 security and use digital certificates obtained through the Aerolink security suite to issue and verify digital signatures. Find the certificate data inside the 1609.2 data block and take note of the contents. What do you think are the most significant/important elements of the certificate?

   - What are some of the other security elements from 1609.2 that you can see in the captured BSMs which were not included in V2Verifier? For any elements you identify, what do you think is their purpose and why do you think they are important?

   - At the application layer, the MK6Cs use the commercial standard for BSMs, SAE J2735 This standard cannot be implemented in open-source because it is proprietary, so using commercial equipment gives you a look at this real V2V feature which is not included in V2Verifier or srsRAN. In Wireshark, take a look at the contents of the SAE J2735 BSM payload for one of the captured BSMs. What are the contents of a commercial BSM like this one?