Vehicular communication
Laboratory Assignment

Assignment
The main goal of this exercise is to Inspect the impact of channel quality on the usability of the modulation scheme. This is also about the desired packet loss. A pure simulation measures channel quality characteristics first, then the Wi-Fi implementation using SDR, and finally, the mobile communication technology - V2X LTE-sidelink.

In this lab, you will use GNU Radio, a tool to control and process Software Defined Radio (SDR), as the lab is about emulation of vehicular communication based on Wireless LAN, i.e., IEEE 802.11p[1] and Cellular V2X, i.e., 4G. Moreover, you will inspect the impact of Modulation and Code Rate (MCS).

Task 1. - Measure the dependency of coding rate and modulation on SNR
For each modulation scheme, find the **smallest SNR** for **lossless** communication. Inspect the error rate and search for persistent zero loss. Observe the constellation diagram as well since the error rate is not computed for a disrupted channel and may show zero despite no communication.

Measurement instructions
1. Start **GNURadio companion** (topmost icon on the left)
2. ALL code is located in folder ***pybombs/src/gr-ieee-80211/examples***
3. Open ***wifi_loopback.grc*** and start it (with the green play button)
4. Set the modulation and the coding rate and **find the required SNR** (required to get the desired quality, i.e., zero packet loss). **Record your findings in the measurement report.**

Notes:

- **Reasonable SNR accuracy is about 1 dB**. Do not spend time tweaking it to higher accuracy (0.5 or even 0.1 dB) as it takes too much time.
- This task is a pure simulation, no actual radio hardware or radio transmission is used.
- This part can be done on both PCs simultaneously. Split the tasks properly in your group so you can work more effective.
- Do not change the other (despite changeable) values (*pdu_length*, *interval*, *epsilon*, *chan_est*) since they are not necessary for this task.

[1] https://www.wime-project.net/

Task 2. - Transmitting real data over 802.11 with minimal loss

For each modulation find the proper TX and RX gain values for loss-less communication and observe the impact of packet fragmentation on round trip time.

Measurement instructions

Here we are using GNU radio SDR as a simplified Wi-Fi transceiver. The implementation is not ideal, and some tweaks and workarounds are needed for the available implementation to work. The task is based on using the virtual TAP network device as the source of data for the USRP.

- Prepare the system environment by running script ***sudo bash wifi_transceiver_env.sh***. Script can be found in lab. PCs (directory *pybombs/src/gr-ieee-80211/examples*) and Appendix I contains the detailed description of its functionality.
- Open ***wifi_transceiver.grc*** in GNURadio companion on both PCs.
    a. Configure **MAC addresses in WI-FI MAC block.**
        - **Source MAC** is your PC tap0 interface MAC (**ip a show tap0 | grep link**).
        - **Destination MAC** is your colleagues tap0 MAC.
    b. Run via green play button.
    c. Select the unused channel in band 5 GHz.
- **Open Wireshark** and start collecting packets on tap0 interface.
    a. Open terminal and run ***sudo wireshark***
    b. You should be able to observe outgoing and incoming packets as well as their eventual fragmentation (once the GNUradio starts to send them).
- Run proper command to ping the peer PC. This creates packets at the application level, the kernel routes them to the tap device (thanks to your previous setup) where USRP will take the data and try to transmit it over your Wi-Fi software radio implementation.
    a. ***ping 192.168.ROW_NUMBER.PC_OTHER_IP -s #ICMP_SIZE -c 30 -i 0.3***
        - **-s** number of bytes send per ICMP request
        - **-c** controls total number of ICMP requests
        - **-i** interval between ICMP requests
    b. Find/measure **tx_gain** and **rx_gain** required for a dependable transmission (better than 5% packet loss)
        - Note: **Sufficient accuracy is about 0.1. Do not spend time tweaking it to higher accuracy (0.05 or even 0.01) as it takes too much time.**
    c. Compute or observe the margin where your packets start to get fragmented, for both one and two fragments per ICMP request.
    d. **Do** your RTT measurement for fragmented and unfragmented ICMP requests **to fill the table**.

Task 3. - Transmitting real data over Cellular V2X (Device to Device communication)

In this task we are using the same USRP hardware, but now to implement V2X cellular communication. We use Frauhofer implementation of V2X LTE-sidelink.

On one PC **start master**, that controls communication and on other PC **start client** that connects to master.

A. Open terminal window and go to ***fraunhofer-sidelink*** directory.
B. Edit plain-text file configuration located at ***fraunhofer-sidelink/srssl/ue.conf.example****)*
   - Set uplink (ul_freq) and downlink (dl_freq) frequency based on your row

Row	**1**	**2**	**3**	**4**	**5**
Frequency [MHz]	3510	3520	3530	3540	3560

C. On PC with master run ***sh start.sh*** and wait for initialization before proceeding with the next steps
D. On PC with client run ***sh start_client.sh***
E. On both PCs set Tx and Rx gains to 55/60 via command. **Replace ?** with **1 for master** and **2 for client**:
   - *curl -X PUT -H "Content-Type: application/json" -d '{"rx_gain":60, "tx_gain":55}' localhost:1300**?**/phy/gain*
F. Check the gains with command (Replace ? with 1 for master and 2 for client):
   - ***curl -X GET localhost:1300?/phy/metrics***
G. In a new terminal window on **client** start ping
   - ***ping 10.0.2.11 -s #ICMP_SIZE -c #COUNT***
     - -s controls number of bytes send
     - -c controls number of repetitions
H. To set MCS (***#MCS_VALUE***) and TBS on **client** use command:
   - ***curl -X PUT -H "Content-Type: application/json" -d '{"pssch_fixed_i_mcs":#MCS_VALUE,"pssch_min_tbs":100 }' localhost:13002/phy/repo***
I. Select a reasonable number of ping repetitions for valid average values computation and repeat for required MCS.
J. Record your findings in the report table.

Appendix I

Task 2 environment configuration script

GNU radio is the user space application here which takes the packets from tap device and sends them over the radio, then injects the received packets on the other side. The implementation is not ideal, we have to use a few workarounds for everything to work together:

1. Add the TAP device.
2. Change its MAC.
3. Set the MTU for the device (note that this is not the actual limitation of V2X Wi-Fi IEEE 802.11p standard, but just like the later artificial delay it is needed to improve the performance of USRP).
4. Add the IP address.
5. Set the MTU for the route advertised to the OS (this is to impose the IP fragmentation to IP stack).
6. Artificial packet delay is imposed here as well, this is to give the USRP device time to process everything.
7. The "layer 2 driver" with proper ARP implementation does not exist here, so we need to fill the ARP cache manually.

The upper points are shown in the following bash commands **but just for the illustration.**

The actual PC setup will be done via a script that the lecturer will give to you.

```
sudo ip tuntap add dev tap0 mode tap
sudo ifconfig tap0 down
sudo ifconfig tap0 hw ether 12:34:56:78:90:ab # replace ab by a = row number b = 1 for left PC, b = 2 for right PC
sudo ifconfig tap0 mtu 440
sudo ifconfig tap0 up
sudo ifconfig tap0 192.168.ROW_NUMBER.PC_IP # ROW_NUMBER = 120 + row number, PC_IP is 1 for left PC and 2 for right PC
sudo route del -net 192.168.ROW_NUMBER.0/24 # Note that route may not exists previously
sudo route add -net 192.168.ROW_NUMBER.0/24 mss 400 dev tap0
sudo tc qdisc del dev tap0 root # Note that device may not exists previously
sudo tc qdisc add dev tap0 root netem delay 10ms
sudo arp -s 192.168. ROW_NUMBER. PC_OTHER_IP 12:34:56:78:90:ab # replace ab by a = row number !! b = 1 for right PC, b = 2 for left PC !!
```

Once the PC networking is set, you can review your setup by commands:

- **ip address** – see the device setup on IP layer, note the MTU and IP address.
- **ip route** – see the routing table.
- **tc qdisc show** – see queue, imposing the artificial delay.