D2D setup: Instruction Guide

1. Hardware setup

In this document we provide the installation and execution instructions for two D2D scenarios: the *off-network scenario* where none of the UEs in connected to any network infrastructure and the *relay network scenario* where at least one UE is connected to the network and acts as a relay for the external (e.g., internet) traffic from/to the off-net UE(s). Here, we provide some information about our hardware setup for these two scenarios.

- **Off-network scenario**

For this scenario, each UE node consists of a NUC PC or laptop (tested with 8 CPU cores, 8GB RAM) running **Ubuntu 16.04 or 18.04** on a **low latency kernel.** Each machine is connected with a *USRP B200 mini* or *USRP B210* RF front end. The 2 USRPs need to be connected with an external 10 MHz frequency reference in order to achieve sidelink synchronization between the communicating UEs. In our current setup, we have tested using either a signal generator or an octoclock as the external frequency reference (Figure 1). For this purpose, GPS-disciplined oscillators could also be used instead. This setup has been tested in Band 7 and Band 14 so far.

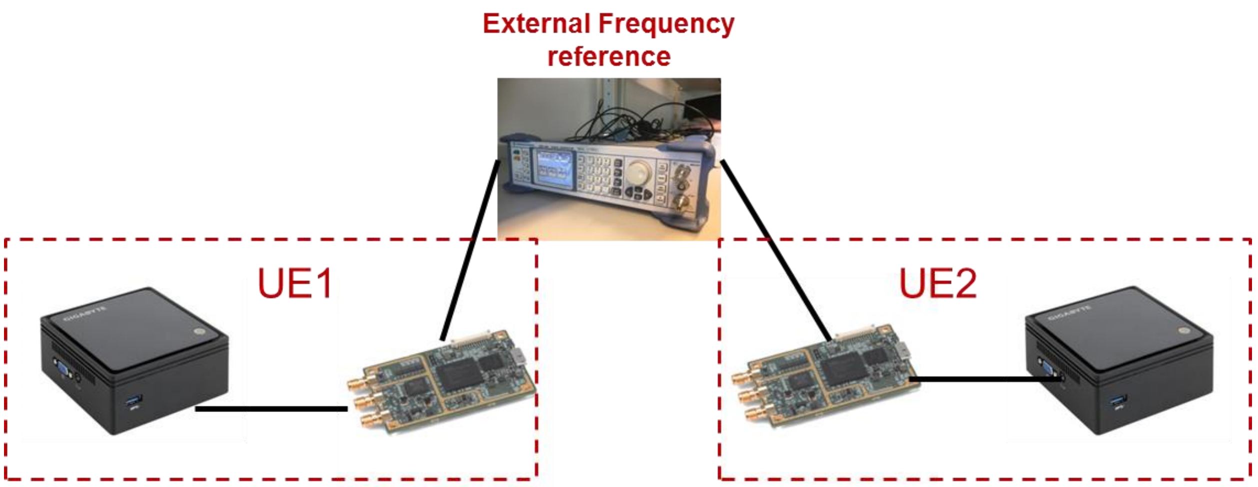

Figure 1 Off-network RF setup with 2 UEs

- **Relay network scenario**

For this scenario, we also need a node (PC + USRP) acting as the eNB, as well as an additional USRP at the relay UE node which will be connecting to the network. The USRP of the eNB node also needs to be connected with the external frequency reference. Regarding the relay node, one of the USRPs is used for the Uplink, Downlink and Sidelink Tx directions and the second USRP is used for Sidelink Rx only. The two USRP devices need to be synchronized in time and as a result a common external PPS (Pulse per second) reference has to be used (Figure 2). For this reason we have been using two USRP B210 devices at the relay node, which provide an input for the PPS reference.

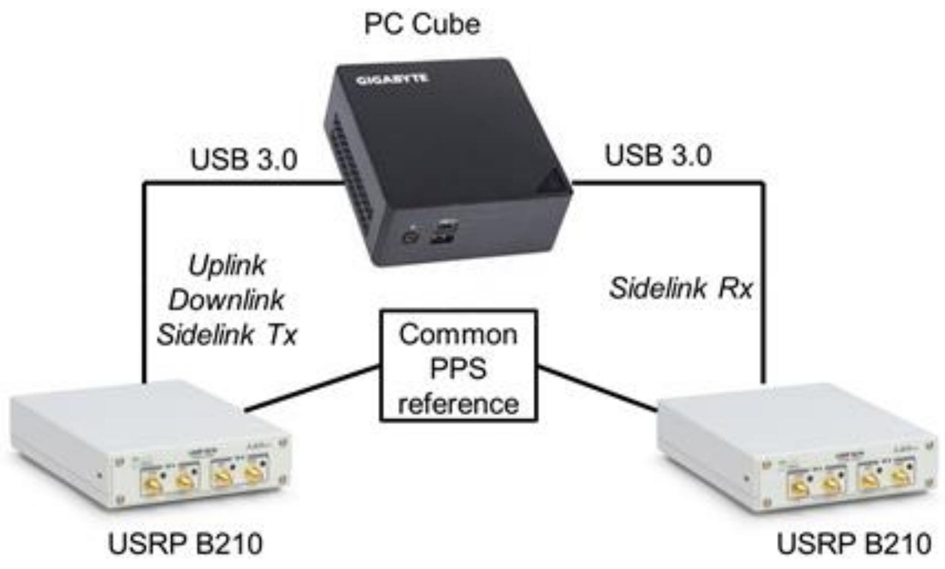

Figure 2 Relay node RF setup

2. UE installation and execution

To get the code from the OAI repository please run in a new folder:

```
git clone https://gitlab.eurecom.fr/oai/openairinterface5g.git
```

Then switch to the LTE-sidelink branch :

```
cd openairinterface5g
git checkout LTE-sidelink
```

Before building the UE, if this UE is intended to be an on-network UE (i.e., that will attach to the core network), you should make sure that the IMSI used for the UE should match an entry in the HSS. The file named *openair3/NAS/TOOLS/ue_eurecom_test_sfr.conf* can be modified for the IMSI entry corresponding to HSS entry HPLMN+MSIN=208930000000001 and HSS OP key used to determine the OPC key below as follows:

```
SIM: {
        MSIN=”0000000001”;
        USIM_API_K= "fec86ba6eb707ed08905757b1bb44b8f";
        OPC="c42449363bbad02b66d16bc975d77cc1";
        MSISDN="33611123456";
    };
    # Home PLMN Selector with Access Technology
    HPLMN= "20893";
```

To build the UE, please refer to the following, starting from the parent directory

```
source oaienv
cd cmake_targets
./build_oai –I –w USRP    #Only the first time to install software dependencies
./build_oai --UE -w USRP -C
```

Then, the routing kernel module responsible for handling the IP traffic should be mounted.

```
cd tools
source init_nas_s1 UE
ifconfig #To verify that the new ip interfaces (oip0, oip1) corresponding to OAI are
         #present
```

*Repeat the above procedure for the second UE.

In order to launch a scenario with 2 off-net UEs, you should run the following (in this example using band 14) :

```
# The first UE will act as a synchronization reference, so the –ue-synchref option
should be present
cd openairinterface5g/cmake_targets/lte_build_oai/build
sudo ./lte-uesoftmodem --ue-synchref --ue-sl-only --ue-enable-sl -C 763000000 -r 50 -
-ue-rxgain 100 --ue-txgain 0 --usrp-clksrc external | tee /tmp/oai.log
```

```
# The second UE will act as a synchronization receiver, so the –ue-synchref option
should be absent
```

```
cd openairinterface5g/cmake_targets/lte_build_oai/build
sudo ./lte-uesoftmodem --ue-sl-only --ue-enable-sl -C 763000000 -r 50 --ue-rxgain 100
--ue-txgain 0 --usrp-clksrc external | tee /tmp/oai.log
```

After running the above commands on the 2 UEs you should see that the non-synch-ref can synchronize to the synch-ref, given that they have a common 10MHz external reference.

3. D2D-application installation and execution

The D2D Application implements the basic ProSe functionalities related with all the PC5 types of traffic. This allows for testing separately all the D2D related features (**i.e., multicast traffic, discovery, 1-to-1 connection establishment and Unicast traffic, Relay traffic**). The D2D application is publicly available on a separate repository: https://gitlab.eurecom.fr/tien-thinh.nguyen/d2d-l3-stub

In the following, the instructions for getting the D2D application code, installing it, configuring the environment and launch the distinct features are provided.

To get the code please run:

```
git clone https://gitlab.eurecom.fr/tien-thinh.nguyen/d2d-l3-stub
```

To install the code:

```
gcc -I . d2d_app.c -o d2d_app –lpthread
```

Afterwards, some network configuration steps are required for each UE (assuming scenarios with two sidelink enabled UEs). These steps are described below analytically but instead of manually applying them you can also run the scripts *relay_UE_routing.sh* and *remote_UE_routing.sh* for the synch-ref UE and the non-synchref UE respectively, considering a scenario with 2 UEs:

```
#UE 1 (synchronization reference UE)
#Configure the IP interface corresponding to sidelink
sudo ifconfig oip0 10.0.0.1
sudo ifconfig oip0 hw ether 00:00:00:00:00:01
#Add ARP entry corresponding to the IP and MAC address of the other UE
sudo ip neigh add 10.0.0.2 lladdr 00:00:00:00:00:02 dev oip0 nud permanent
#or (optional, in case ARP fails to work with oip interface):
sudo ip neigh change 10.0.0.2 lladdr 00:00:00:00:00:02 dev oip0 nud permanent
```

```
#UE 2
#Configure the IP interface corresponding to sidelink
sudo ifconfig oip0 10.0.0.2
sudo ifconfig oip0 hw ether 00:00:00:00:00:02
#Add ARP entry corresponding to the IP and MAC address of the other UE
sudo ip neigh add 10.0.0.1 lladdr 00:00:00:00:00:01 dev oip0 nud permanent
#or (optional, in case ARP fails to work with oip interface):
sudo ip neigh change 10.0.0.1 lladdr 00:00:00:00:00:01 dev oip0 nud permanent
```

Then, the different test scenarios can be launched as follows (assuming L2Ids of UE1, UE2 are 0x01, 0x02 respectively. GroupL2Id is set to 0x03.):

- **Test One-to-many scenario**

```
#Run UE1 then UE2:

 - UE1 OAI (synch-ref): using the command configuration described in section 1
 - UE1 D2D App (on a separate terminal):
     ./d2d_app -g 0x01 0x03

 # A SLRB shall be established for this communication (check SLRB_ID corresponding to
group communication with ID 0x03 from d2d_app's terminal)

 - sudo iptables -A POSTROUTING  -t mangle -o oip0 -d 224.0.0.3 -j MARK --set-mark
XXX #XXX is SLRB_ID

 - UE2 OAI: using the command configuration described in section 1
 - UE2 D2D App (on a separate terminal):
     ./d2d_app -g 0x02 0x03

  # a SLRB shall be established for this communication (check SLRB_ID corresponding
to group communication with ID 0x03 from d2d_app's terminal)

 - sudo iptables -A POSTROUTING  -t mangle -o oip0 -d 224.0.0.3 -j MARK --set-mark
XXX #XXX is SLRB_ID

#Validation using ping (example with UE2 sender – UE1 receiver)
- Sender – UE2: ping -I oip0 224.0.0.3
- Receiver – UE1: using wireshark

#Validation using iperf
-Sender – UE2:   iperf -c 224.0.0.3 -u -b 1M --bind 10.0.0.1 -t 100
-Receiver – UE1: iperf -s -i 1 -u -B 224.0.0.3
```

- **Test PC5-S (UE2 – sender initiating Direct communication, UE1 – receiver accepting Direct communication) and PC5-U for One-to-One and One-to-Many scenarios**

```
#Step 1: Launch OAI at the 2 UEs

  - UE1 OAI (synch-ref): using the command configuration described in section 1
  - UE2 OAI: using the command configuration described in section 1

#Step 2: Launch D2D App for PC5-S receiver (in this example this is UE 1)

  - UE1: ./d2d_app -r 0x01 0x02 #listen to PC5-S communication and incoming PC5-U
packet from UE2, 0x01 - UE1-L2Id, 0x02 - UE2-L2Id

# Mark the packets with the corresponding SLRB
  # 2 SLRBs will be established: 1 for unicast communication and 1 for multicast
(check SLRB_ID from d2d_app's screen)

  - sudo iptables -A POSTROUTING -t mangle -o oip0 -d 10.0.0.2 -j MARK --set-mark XXX
  #XXX is SLRB_ID for unicast communication with UE 2
  - sudo iptables -A POSTROUTING -t mangle -o oip0 -d 224.0.0.3 -j MARK --set-mark
  XXX #XXX is SLRB_ID for multicast communication

#Step 3: Launch D2D App for PC5-S sender (in this example this is UE 2)

  - UE2: ./d2d_app -s 0x02 0x01 #initiating PC5-S communication and establish 1-1
  communication, 0x02 - UE2-L2Id, 0x01 - UE1-L2Id

  # Mark the packets with the corresponding SLRB
  # 2 SLRBs will be established: 1 for unicast communication and 1 for multicast
  (check SLRB_ID from d2d_app's screen)

  - sudo iptables -A POSTROUTING -t mangle -o oip0 -d 10.0.0.1 -j MARK --set-mark XXX
  #XXX is SLRB_ID for unicast communication with UE 1
  - sudo iptables -A POSTROUTING -t mangle -o oip0 -d 224.0.0.3 -j MARK --set-mark
  XXX #XXX is SLRB_ID for multicast communication

#Validation using ping
- Unicast (UE2): ping –I oip0 10.0.0.1
- Multicast (UE2): ping –I oip0 224.0.0.3
```

- **Test PC5-D (Discovery)**

```
#Step 1:
- UE1 OAI: using the command configuration described in section 5.3.4.2
- UE1: ./d2d_app -d #send a PC5-Discovery-Announcement via PC5D
```

```
#Step 2:
- UE2 OAI: using the command configuration described in section 5.3.4.2
- UE2: ./d2d_app -d #send a PC5-Discovery-Announcement via PC5D
```

- **Test traffic relaying scenario**

In order to be able to relay external IP traffic (target destination is 8.8.8.8 in this example) from/to the remote UE through the connected UE, the instructions provided for the previous scenario on one-to-one communication should be combined with the following ip table configuration, considering UE 1 (synch-ref) as the relay UE and UE 2 as the remote UE:

```
#UE1 (relay UE)

ip route add 8.8.8.8 dev oip1

# Applying NAT so that the Remote UE originating/destined traffic does not get
blocked at the PGW.
sudo iptables -t nat -A POSTROUTING -o oip1 -j MASQUERADE
```

```
#UE2 (remote UE)

 # Associate destination address (e.g. 8.8.8.8) with the MAC address of the
relay UE at the neighbor table
sudo ip neigh add 8.8.8.8 lladdr 00:00:00:00:00:01 dev oip0 nud permanent

# Mark outgoing packets for specific destination address (e.g. 8.8.8.8) with
SLRB id
sudo iptables -A POSTROUTING  -t mangle -o oip0 -d 8.8.8.8 -j MARK --set-mark
XXX
```

Once these configuration steps are applied, the relay UE can be launched with slightly modified configuration comparing to section 1, as follows:

```
cd openairinterface5g/cmake_targets/lte_build_oai/build
sudo ./lte-uesoftmodem --ue-synchref --ue-enable-sl -C 763000000 -r 50 --ue-rxgain
100 --ue-txgain 0 --ue-max-power 0 --usrp-clksrc external | tee /tmp/oai.log
```

4. eNB installation and execution

The OAI code can be downloaded from gitlab as follows:

```
git clone https://gitlab.eurecom.fr/oai/openairinterface5g.git
```

Once the OAI code is downloaded, the develop branch needs to be checkout as follows:

```
cd openairinterface5g
git checkout develop
```

The following commands can be used to build the OAI code. The -I option should only be used the first time to install all the dependencies.

```
cd cmake_targets
./build_oai –I #Only the first time to install the software dependencies
./build_oai --eNB -w USRP -C
```

An example eNB configuration file is shown below. The file can be found at *targets/PROJECTS/GENERIC-LTE-EPC/CONF/enb.band14.tm1.50PRB.usrpb210.conf.* The fields highlighted in yellow may need to be modified. The MCC, MNC, and TAC need to match the settings at the EPC. The downlink frequency and uplink frequency offset depends on the band used. In this example, we used Band 14. The rx_gain and RUs settings may need to be lowered or increased depending on the specific testbed configuration, if wired connections or antennas are used, distance between antennas, attenuators, etc. Please note that the sidelink code only works at 10MHz (i.e., 50 PRBs).

```
eNBs =
(
 {
    …
    tracking_area_code  =  1;
    plmn_list = ( { mcc = 208; mnc = 93; mnc_length = 2; } );
    component_carriers = (
      {
      …
      downlink_frequency                                    = 763000000L;
      uplink_frequency_offset                               = 30000000;
      …
      rx_gain                                                                       = 100;
      }
    );
    ////////// MME parameters:
    mme_ip_address      = ( { ipv4       = "10.10.10.6";
                              ipv6       = "10:10:30::17";
                              active     = "yes";
                              preference = "ipv4";
                            }
                          );

    ///X2
    enable_x2 = "no";
    t_reloc_prep      = 1000;      /* unit: millisecond */
    tx2_reloc_overall = 2000;      /* unit: millisecond */

    NETWORK_INTERFACES :
    {
        ENB_INTERFACE_NAME_FOR_S1_MME            = "eno1";
        ENB_IPV4_ADDRESS_FOR_S1_MME              = "10.10.10.5/24";
        ENB_INTERFACE_NAME_FOR_S1U               = "enx00249b29e9a3";
        ENB_IPV4_ADDRESS_FOR_S1U                 = "10.10.20.5/24";
        ENB_PORT_FOR_S1U                         = 2152;

        ENB_IPV4_ADDRESS_FOR_X2C                 = "10.10.20.5/24";
        ENB_PORT_FOR_X2C                         = 36422;
    };

  }
);
…
RUs = (
    {
       local_rf       = "yes"
         nb_tx          = 1
         nb_rx          = 1
         att_tx         = 0
         att_rx         = 0;
         bands          = [14];
         max_pdschReferenceSignalPower = -24;
         max_rxgain                    = 100;
         eNB_instances  = [0];
```

The eNB can be started with the following command line statement. The file used should be the eNB configuration file modified in the previous section:

```
cd /openairinterface5g/cmake_targets/lte_build_oai/build
sudo ./lte-softmodem -O ../../targets/PROJECTS/GENERIC-LTE-
EPC/CONF/enb.band14.tm1.50PRB.usrpb210.conf --clock 1 | tee oai_enb.log
```