



## AMARI UE Simbox User guide



Version: 2025-12-12

# Table of Contents

<b>1</b>	<b>Introduction .....</b>	<b>2</b>
<b>2</b>	<b>Hardware setup .....</b>	<b>4</b>
2.1	Power supply .....	4
2.2	RF connection .....	4
2.2.1	RF connection for one Cell MIMO 2x2 in FDD mode .....	5
2.2.2	RF connection for two Cells MIMO 2x2 in FDD mode .....	5
2.2.3	RF connection for one Cell MIMO 2x2 in TDD mode .....	6
2.2.4	RF connection for two Cells MIMO 2x2 in TDD mode .....	7
2.2.5	RF connection for one Cell MIMO 4x4 in FDD mode .....	8
2.2.6	RF connection for one Cell MIMO 4x4 in TDD mode .....	9
<b>3</b>	<b>Connection to AMARI UE Simbox .....</b>	<b>10</b>
3.1	Connect your Simbox to the local network .....	10
3.2	Login.....	10
3.3	Web GUI.....	10
3.3.1	Set a password.....	10
<b>4</b>	<b>LTE service .....</b>	<b>12</b>
4.1	Manage LTE automatic service .....	12
4.1.1	Status .....	12
4.1.2	Stop .....	12
4.1.3	Start .....	12
4.1.4	Disable .....	12
4.1.5	Enable.....	12
<b>5</b>	<b>Initial setting.....</b>	<b>13</b>
5.1	Selection of configuration file.....	13
5.2	Frequency and bandwidth configuration .....	13
5.3	Multiple cells configuration .....	13
5.3.1	Example : 2 LTE cells from two different eNodeB .....	13
5.3.2	Example : 2 LTE cells from same eNodeB .....	14
5.3.3	Example : 1 LTE and 1 NR cell (NSA mode) .....	14
5.4	TX/RX gain setting .....	15
5.4.1	Wired test .....	15
5.4.2	Wireless test .....	15
5.5	Multiple UE mode and timing advance .....	15
5.6	Reload config files .....	16
<b>6</b>	<b>Create scenario using WebGui.....</b>	<b>17</b>
6.1	Configuration.....	17
6.2	Create scenario .....	18
6.2.1	Create UEs tab .....	19
6.2.2	PDN tab .....	20
6.2.3	Channel simulation tab .....	20
6.2.4	Power on/off tab.....	20

6.2.5	Simulations tab .....	21
6.3	Export scenario .....	23
<b>7</b>	<b>Run UE scenarios with WebGui .....</b>	<b>24</b>
7.1	UE simulator tab .....	24
7.1.1	Actions buttons .....	24
7.1.2	UE list .....	24
7.1.3	Statistics .....	24
7.2	Scenario example .....	25
7.3	Executing scenario tab .....	26
7.4	Example .....	26
<b>8</b>	<b>Create scenario without WebGui .....</b>	<b>29</b>
8.1	Multiple UEs .....	29
8.2	Examples of configuration file .....	29
8.2.1	Power_ON-OFF .....	29
8.2.2	Power_ON-PING .....	30
8.2.3	Power_ON-2xUEs-UDP_UL .....	30
8.2.4	Uplink_Traffic_Specific-APN .....	31
8.3	Remote control .....	32
<b>9</b>	<b>Downlink and Uplink transfer using iperf .....</b>	<b>33</b>
9.1	Introducing Linux Network Namespaces .....	33
9.2	Uplink test .....	33
9.3	Downlink test .....	33
<b>10</b>	<b>NB-IOT devices .....</b>	<b>35</b>
10.1	Stand alone cell .....	35
10.2	Guard band and in band cell .....	35
<b>11</b>	<b>Channel simulator .....</b>	<b>36</b>
11.1	Cell configuration .....	36
11.2	UE parameters .....	37
11.3	Channel simulation scenario using WebGui .....	37
11.4	Channel simulation scenario through remote API .....	39
11.4.1	Cell configuration parameters to be provided: .....	39
11.4.2	UE configuration parameters to be provided: .....	39
11.4.3	Command: "ue_move" .....	40
<b>12</b>	<b>logging .....</b>	<b>41</b>
12.1	Logging through WebGui .....	41
12.2	Command line monitor .....	41
12.2.1	t spl command .....	41
12.2.2	t cpu command .....	42
12.2.3	t command .....	42
<b>13</b>	<b>Advanced options .....</b>	<b>44</b>
13.1	Remote UE .....	44
13.2	External application .....	45
13.3	External SIM .....	45

<b>14</b>	<b>Troubleshooting</b>	<b>46</b>
14.1	No SIB found	46
14.2	SIB found but attach fails	46
14.3	Downlink peak throughput not achieved	47
14.4	RRC_STATE locking value	48
14.5	Connection to webGui fails	48
14.6	TRX discontinuity too wide	48



# 1 Introduction

Amari UE Simbox is a UE simulator acting as a 3GPP compliant LTE and 5G NR UE (in NSA or SA mode).

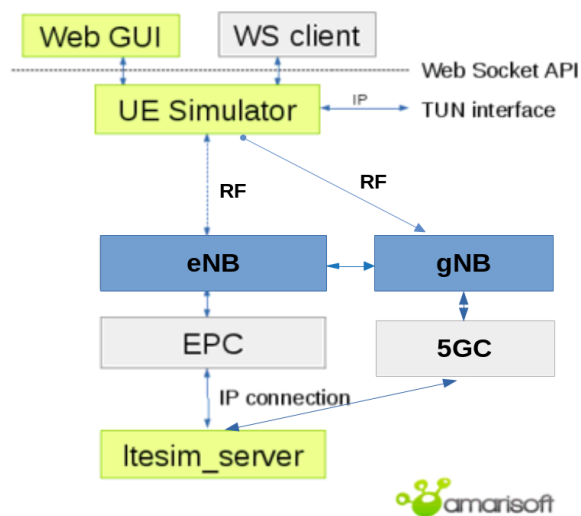
It can simulate up to 1000 UEs sharing the same spectrum with different types of traffic within multiple cells. It is also supporting NB-IoT and LTE-M.

The offer is completed by an integrated IP simulation package for reproducible test results.



The UE Simbox is a turnkey solution running on Fedora 42 operating system. It embeds four SDR cards, all software components and license required to emulate 4G/5G UEs.

The UE Simbox can be connected to the network over the air or in conducted mode on one side. On the other side it can be configured and controlled using either configuration files, WebSocket API or via a web based GUI. IP traffic may be reachable with a Linux TUN network interface.



This document describes the first steps to start and configure your Amarisoft Simbox. For advanced configurations and testing, please refer to the application notes and other documents available under [extranet.amarisoft.com](http://extranet.amarisoft.com) or under the `/root/ue/doc/` folders of your Simbox.



## 2 Hardware setup

### 2.1 Power supply

Plug the external power cable provided with the Simbox in the **AC-in** Connector and place the toggle switch located in the rear panel on position "1", then press the **Power on** button on the top of the front panel to turn the system on.

The led will light up blue once Simbox is powered ON .

Power adapter specifications:

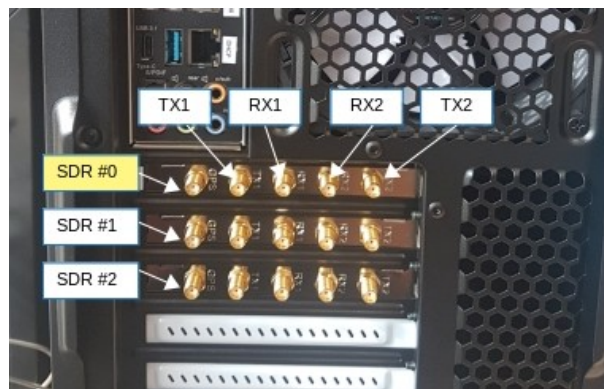
Input: 100~240 V AC, 50~60 Hz

### 2.2 RF connection

Before initiating any testing with the LTE UE simulator, first thing to do is to connect it correctly to the eNodeB/gNodeB.

Your setup is composed of four SDR cards that have two RX, two TX plus one GPS SMA connectors.

- **RX1** is the main receive antenna port. Must be connected to eNodeB/gNodeB main transmit antenna
- **RX2** is the diversity receive antenna port. It is needed when diversity or MIMO transmission modes are used. Must be connected to eNodeB/gNodeB secondary transmit (Diversity) antenna
- **TX1** is the main transmit antenna port. Must be connected to eNodeB/gNodeB main receive antenna
- **TX2** is not used. (the current version only supports a single uplink antenna)
- **GPS** is used for connecting an external GPS clock. See `trx_sdr.pdf` file for more details.



Each SDR card can support one cell MIMO 2x2. By consequence, for scenarios such as MIMO 4x4 or multiple cells, several SDR cards are required. See [4x4 MIMO RF connection], page 8, or See [multiple cells RF connection], page 5, for more details.

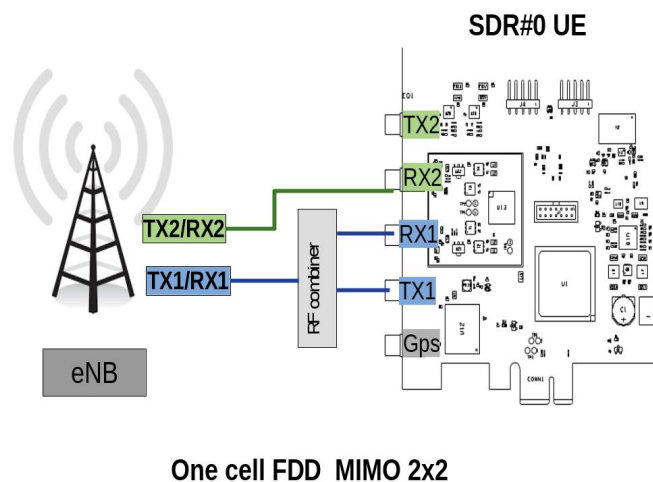
LTE UE simulator can be connected to the eNodeB either using RF cables or in wireless way using LTE antennas.

- When LTE antennas are used, RF connection is simple and just consist in connecting the four antennas provided to TX1, TX2, RX1 and RX2 SMA connectors.

- When RF cables are used, cabling must follow the instruction provided in the following sections. TX and RX gains must be adjusted accordingly. See [TX:RX gain setting], page 15, for more details

Note: When using RF combiners, unused ports of the combiners must be terminated with RF terminator or absorber especially when using frequency above 2.5GHz. Otherwise big insertion losses may happen and downgrade the RF performances.

### 2.2.1 RF connection for one Cell MIMO 2x2 in FDD mode



Note: For SISO test, connection between TX2/RX2 and UE (RX2) is not required

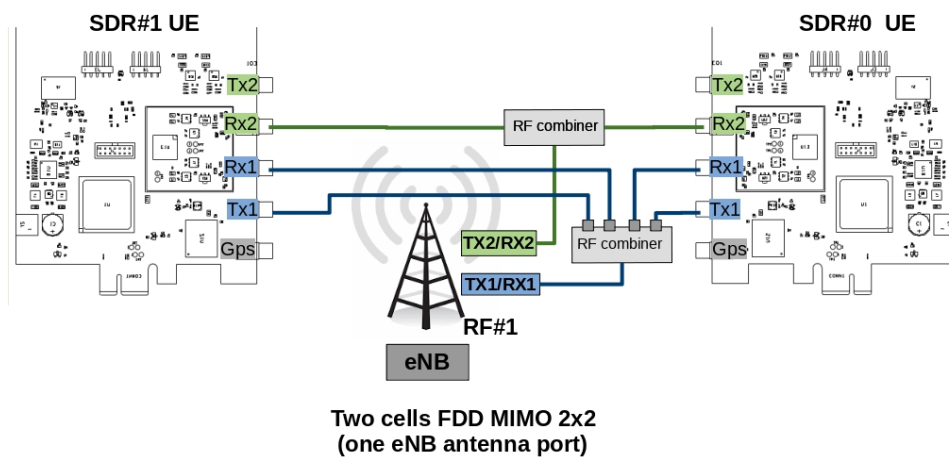
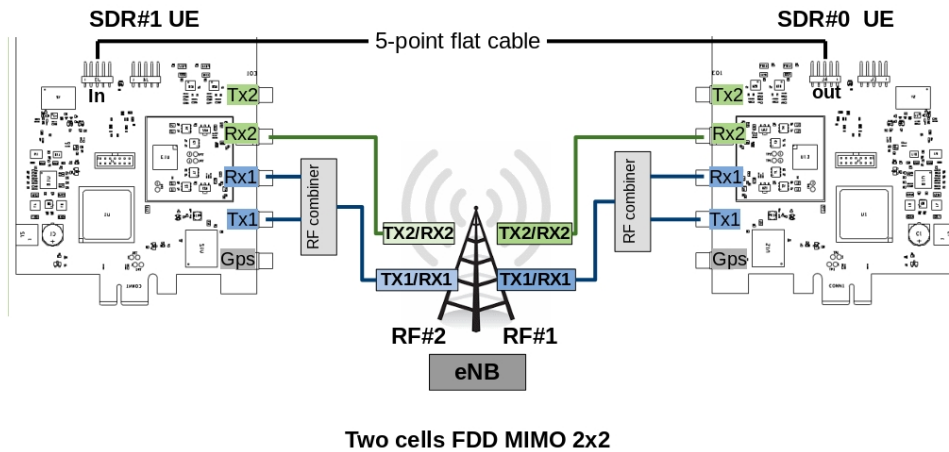
### 2.2.2 RF connection for two Cells MIMO 2x2 in FDD mode

When two cells are configured at eNodeB side, two SDR cards are required at UE simulator side even if both cells are intrafreq cells or use the same RF antenna port. Each SDR card is dedicated to one specific cell.

In the picture below, both eNodeB cells use a different antenna port (RF#1 and RF#2). This cabling is relevant for two neighbor cells or for a Carrier Aggregation test.

Note: When both cells are using the same eNodeB antenna port, please refer to the second picture (one eNodeB antenna port) for the RF connection.

Note: This cabling also corresponds to a NSA cell configuration with one LTE cell 2x2 MIMO and one NR cell 2x2 MIMO



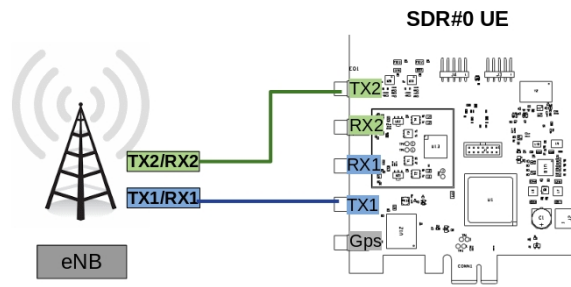
### 2.2.3 RF connection for one Cell MIMO 2x2 in TDD mode

In TDD mode, as both TX and RX are sent on the same channel, only TX1 port is needed for the main antenna and TX2 for the diversity/MIMO antenna.

However, with Amarisoft SDR card, it is possible to force the DL signal reception on RX port and Uplink signal transmission on TX port when `rx_antenna: "rx"` is set in the RF driver configuration file.

This improves the performance of the SDR card as we don't have to switch the RF several times per radio frame.

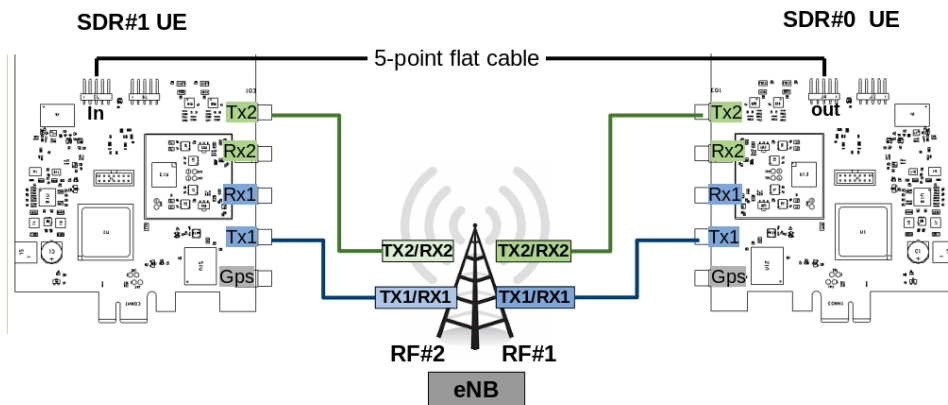
When `rx_antenna: "rx"` is set, RF connection is the same as for FDD.



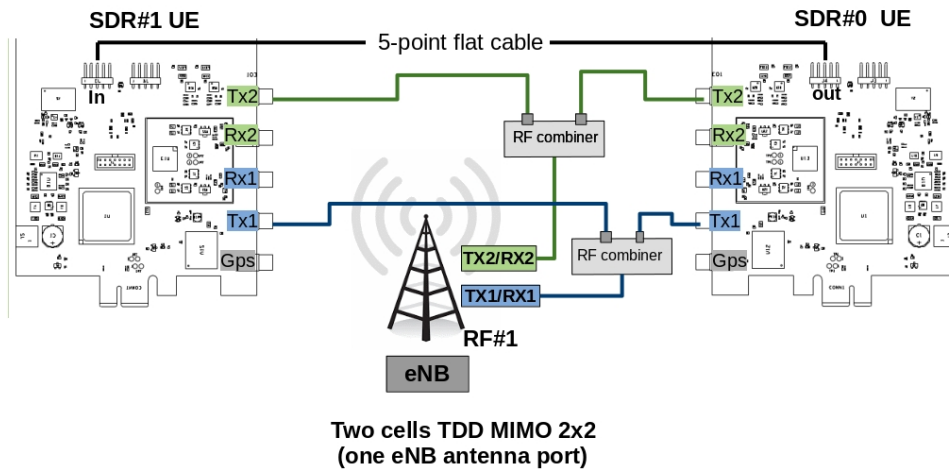
One cell TDD MIMO 2x2

## 2.2.4 RF connection for two Cells MIMO 2x2 in TDD mode

As for FDD, in the picture below each eNodeB cell uses a different antenna port (RF#1 and RF#2). This configuration can be used for 2 neighbor cells or for a Carrier Aggregation test. When both cells are using the same eNodeB antenna port, please refer to second picture (one eNodeB antenna port) for the RF connection.



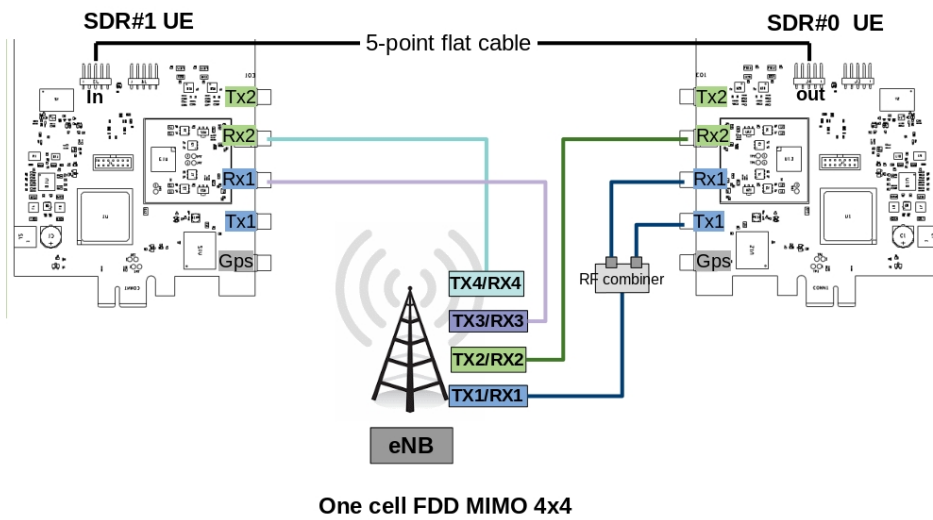
Two cells TDD MIMO 2x2



When `rx_antenna: "rx"` is set, RF connection is the same as for FDD. See section 2.2.3 for more details about `rx_antenna: "rx"`

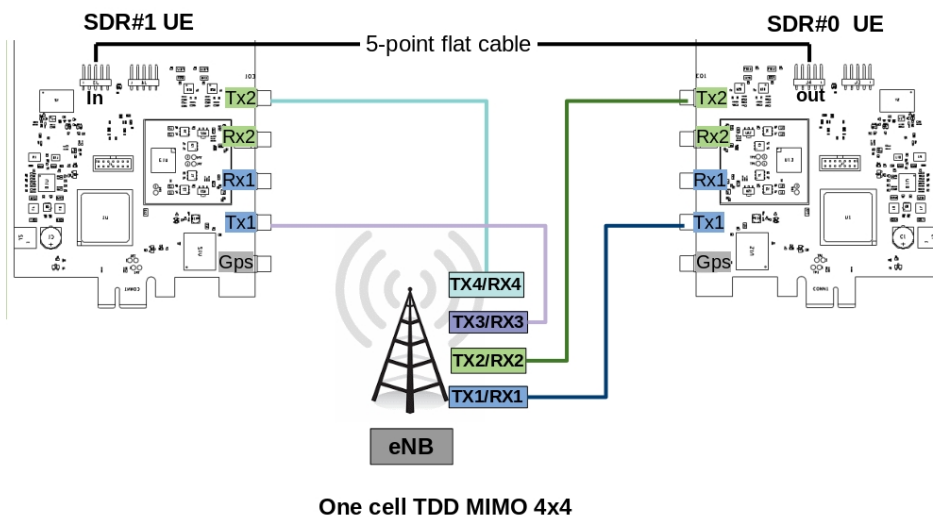
### 2.2.5 RF connection for one Cell MIMO 4x4 in FDD mode

In the case of MIMO 4x4, two SDR cards are required on the UE simulator side. Both SDR cards are combined in order to provide four TX and RX antenna ports.



### 2.2.6 RF connection for one Cell MIMO 4x4 in TDD mode

As for FDD, two SDR cards are required on the UE simulator side. Both SDR cards are combined in order to provide four TX and RX antenna ports.



## 3 Connection to AMARI UE Simbox

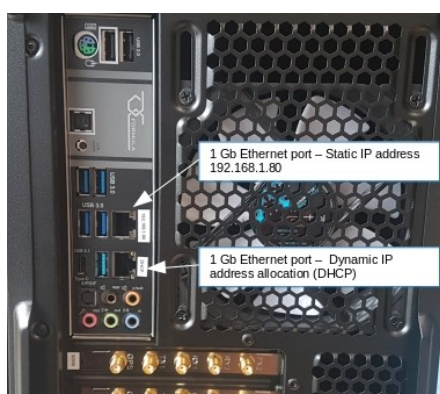
The AMARI UE Simbox does not have a built-in graphical card however you can remotely connect to the callbox by using one of the Ethernet ports.

### 3.1 Connect your Simbox to the local network

The AMARI UE Simbox has two 1Gb Ethernet ports

The first one (on the top) is configured with a static IP address 192.168.1.80.

The second one is configured in DHCP mode and will get automatically an IP address within the range defined on your router.



Once you are remotely connected, you can modify the static IP address allocated by editing the file `/etc/sysconfig/network-scripts/ifcfg-enp7s0`. Thus, future connections to the Simbox can be made using this static address. Note: for Fedora version  $\geq 42$  the file to edit is `/etc/NetworkManager/system-connections/ifcfg-enp7s0.nmconnection`.

### 3.2 Login

To login as root, please use `root/toor` as login / password.

There is also a user account with `user/resu` as login/password.

Note: The Amarisoft software suit is installed and executed under root account.

### 3.3 Web GUI

The web GUI to display logs is available through web access at the following URL:  
`http://<IP address>/`

#### 3.3.1 Set a password

To add authentication to the web portal:

Edit `/etc/httpd/conf/httpd.conf` and look for `<Directory "/var/www/html">` section. Inside it, add `AuthConfig` to `AllowOverride` directive so that it becomes `AllowOverride None AuthConfig`.

Then restart HTTP daemon:

```
service httpd restart
```

Create `/var/www/html/lte/.htaccess` file with following content:

```
AuthType Basic
```

```
AuthName "Amarisoft Web GUI"  
AuthUserFile /etc/httpd/.htpasswd  
Require valid-user
```

To add a user and its password, type the following command:

```
htpasswd -c /etc/httpd/.htpasswd <username>
```



## 4 LTE service

The UE Simbox is configured to provide an automatic LTE service. At each reboot of the PC, LTE UE service is turned on automatically.

The default config is :

- One LTE cat12 UE configured to camp on LTE cell Band7 (EARFCN 3350) SISO 5mhz bandwidth.

### 4.1 Manage LTE automatic service

#### 4.1.1 Status

You can check the LTE service status this way:

```
service lte status
```

The command will return "active (running)" status if service is running

#### 4.1.2 Stop

You can stop all LTE components this way:

```
service lte stop
```

#### 4.1.3 Start

You can start them again this way:

```
service lte start
```

#### 4.1.4 Disable

You may also prevent them to start at boot time:

```
systemctl disable lte
```

NB: lte service remains enable until next reboot

NB2: this command is not available on Ubuntu version <= 14

#### 4.1.5 Enable

You may enable service at boot time this way:

```
systemctl enable lte
```

NB: lte service remains disable until next reboot

NB2: this command is not available on Ubuntu version <= 14

## 5 Initial setting

### 5.1 Selection of configuration file

The default file used by LTE automatic service is `ue.cfg` (available under `/root/ue/config` directory). This file aims to configure the UE simulator parameters such as UE category, EARFCN, IMSI and others.

To change the UE configuration, update this file and restart the LTE service with `service lte restart` command.

Note: Some examples of configuration file (`ue-nbiot.cfg`, `ue-catm1.cfg`) are provided in Amarisoft releases as a starting point.

To quickly swap between different configuration, you can stop the automatic service using `service lte stop` command and run manually the command line `./lteue config/xxx.cfg` under `/root/ue` directory.

### 5.2 Frequency and bandwidth configuration

Once configuration file has been selected, frequency and bandwidth must be set to match eNodeB/gNodeB setting as both parameters are static at UE simulator side.

The parameters to be changed are:

- `dl_earfcn`
- `bandwidth`

### 5.3 Multiple cells configuration

When multiple cells are configured at network side, each of them must be declared in `cell_groups` parameter.

A cell groups references the configuration of 1 or more cells of the same type (LTE, Cat-M1, NB-IoT, or 5G NR). See `lteue.pdf` for more details

Note: Cells within same group must be synchronized at subframe/frame level.

#### 5.3.1 Example : 2 LTE cells from two different eNodeB

The example below represents a scenario with 2 cells from two different eNodeB (not synchronized)

```
cell_groups: [{
  group_type: "lte",
  multi_ue: true,
  cells: [{
    rf_port: 0,
    bandwidth: 20,
    dl_earfcn: 6300,
    n_antenna_dl: 1,
    n_antenna_ul: 1,
  }]
}, {
  group_type: "lte",
```

```

multi_ue: true,
cells: [{
  rf_port: 1,
  bandwidth: 20,
  dl_earfcn: 3350,
  n_antenna_dl: 1,
  n_antenna_ul: 1,
}],
}],

```

### 5.3.2 Example : 2 LTE cells from same eNodeB

The example below represents a scenario with 2 cells from same eNodeB (synchronized)

```

cell_groups: [{
  group_type: "lte",
  multi_ue: true,
  cells: [{
    rf_port: 0,
    bandwidth: 20,
    dl_earfcn: 6300,
    n_antenna_dl: 1,
    n_antenna_ul: 1,
  },
  {
    rf_port: 1,
    bandwidth: 20,
    dl_earfcn: 3350,
    n_antenna_dl: 1,
    n_antenna_ul: 1,
  },
]
}
],

```

### 5.3.3 Example : 1 LTE and 1 NR cell (NSA mode)

The example below represents a scenario with one LTE and one NR cell

```

cell_groups: [{
  group_type: "lte",
  multi_ue: false,
  cells: [{
    rf_port: 0,
    bandwidth: 20,
    dl_earfcn: 300,
    n_antenna_dl: 1,
    n_antenna_ul: 1,
  }]
}, {
  group_type: "nr",
  multi_ue: false,
  cells: [{
    rf_port: 1,
    band: 78,

```

```

        bandwidth: 20,
        dl_nr_arfcn: 632628,
        ssb_nr_arfcn: 632256,
        subcarrier_spacing: 30,
        n_antenna_dl: 1,
        n_antenna_ul: 1,
    }],
}],

```

## 5.4 TX/RX gain setting

TX and RX gain values must be fine tuned depending on your setup (conducted vs wireless conditions, physical attenuator used, combiner/divider, etc..) as dynamic power control is not handled at UE side (excepted if channel simulation mode is enabled) .

TX and RX gain values are defined in RF configuration file located under /root/ue/config/rf\_driver directory. To know which files is used by LTE service, just look at ue.cfg file. Example:

```
include "rf_driver/config.cfg",
```

### 5.4.1 Wired test

In wired test conditions, it's recommended to put attenuator between eNodeB and LTE UE simulator in order not to damage SDR cards and avoid saturation.  
On UE Simulator side, max SDR input is -10 dBm, max SDR output is 5dBm.

Recommended values for PCIe SDR cards are :

```
tx_gain: 60.0, /* TX gain (in dB) */
rx_gain: 0.0, /* RX gain (in dB) */
```

Note: If physical attenuators are used, equivalent gain must be added to these default values

### 5.4.2 Wireless test

In Wireless test conditions, the recommended values for PCIe SDR cards are :

```
tx_gain: 90.0, /* TX gain (in dB) */
rx_gain: 60.0, /* RX gain (in dB) */
```

## 5.5 Multiple UE mode and timing advance

To activate the simulation of multiple UEs, the parameter `multi_ue` must be set to `true`.  
The "multiple UE" mode aims to simulate hundreds of devices on the same PC sharing the same physical layer. As a consequence all UE have the same timing advance and this value must be defined in the ue.cfg file.

There are two ways to define the timing advance for multiple UE.

- It can be set automatically by using the `global_timing_advance=-1`. If automatic mode is set, the UE simulator uses the timing advance from the first received RAR for all UEs. This is the default behaviour.
- It can be set manually by using the `global_timing_advance=x`, ( $x$  = TA of the uplink relative to the downlink in 1/1.92 us unit)

We recommend to use the automatic mode. However, if UEs encounter difficulties to synchronize with eNodeB signal you may have to adjust manually the timing advance. See [SIB found but attach fails], page 46, for more details.

## 5.6 Reload config files

Once UE configuration is completed, you can restart your service to take these modifications into account with command `lte service restart`, and switch to LTE screen using `screen -x lte` command.

As a result, you should see **SIB found** message displayed. This means that eNodeB cell has been detected and UE simulator is ready to be used !

If SIB found message is not displayed, See [No sib found], page 46, for more details.

Type `ue` in the monitor to see the list of UEs and their status.

To power on a UE, just type `power_on ueid` command where `ueid` is the UE ID listed in `ue` command result.

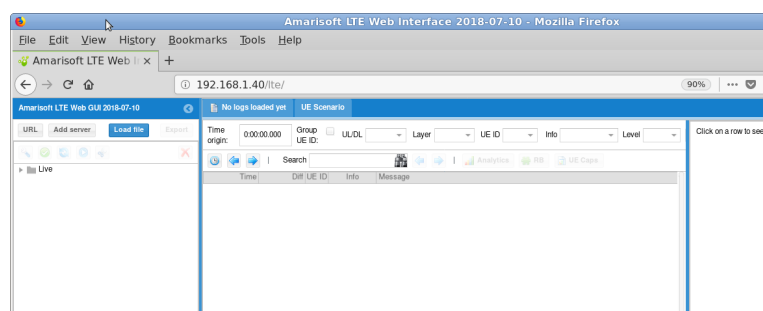
## 6 Create scenario using WebGui

Once your UE simulator has been started and has decoded eNodeB system information (Cell 0: SIB found message is displayed) you can start using it to create devices, generate end to end traffic and simulate different channel conditions in order to test your eNodeB and Core network. These scenarios can be generated automatically using the WebGui as described below.

Note: More complex scenarios can be created by writing your own script manually and controlled using LTEUE Remote API. See [Create scenario without WebGui], page 29, for more details.

### 6.1 Configuration

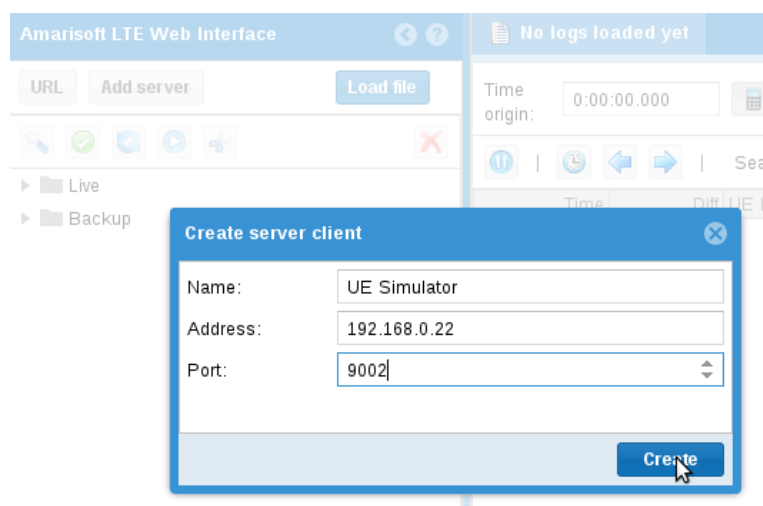
Open your web browser and connect to LTE UE IP address followed by /lte. (example 192.168.1.40/lte). The following main page will open.



Note: If the page can't be loaded, please check httpd service is running on LTEUE side (service httpd status). Also, make sure no firewall or other security software is blocking the communication

Create a websocket and connect a client server :

- Click on **Add server** button on the left panel
- Define a name , as instance: **UE Simulator**
- Set IP address and com Port of LTE UE. By default com port is 9002. The value is defined in ue.cfg file, parameter `com_addr`



Once the server client is created and get connected to the LTE UE service, a new window will appear where logging level per layer can be set

**Please configure logs**

Name: Client

Address: 192.168.1.40

Port: 9002

Log buffer count: 8192

Signal: ☐

CCH: ☐

BCCH: ☐

Layer	Filter	Level	Max size	Payload	
PHY	warn	error	0	<input type="checkbox"/>	⚠️
MAC	warn	error	0	<input type="checkbox"/>	⚠️
RLC	warn	error	0	<input type="checkbox"/>	⚠️
PDCP	warn	error	0	<input type="checkbox"/>	⚠️
RRC	debug	debug	1	<input type="checkbox"/>	⚠️
NAS	debug	debug	1	<input type="checkbox"/>	⚠️
IP	warn	error	0	<input type="checkbox"/>	⚠️

⚠️ ⚠️ ⚠️

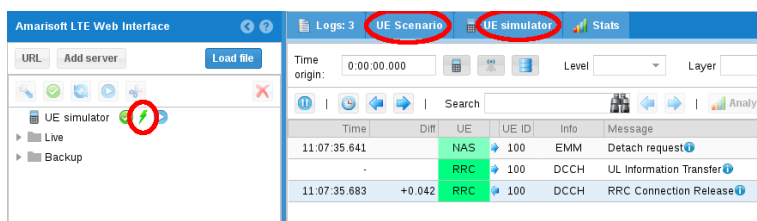
**Update**

For more details about logging configuration, see ltewww.pdf documentation

Note : If this page can't be loaded, please check that lte service is running (`service lte status`)

You should see now (as shown in the picture below)

- A yellow lightning icon on left panel
- Two new tabs on center panel: **UE Scenario** & **UE Simulator**



**UE Scenario** will be used to define scenariii.

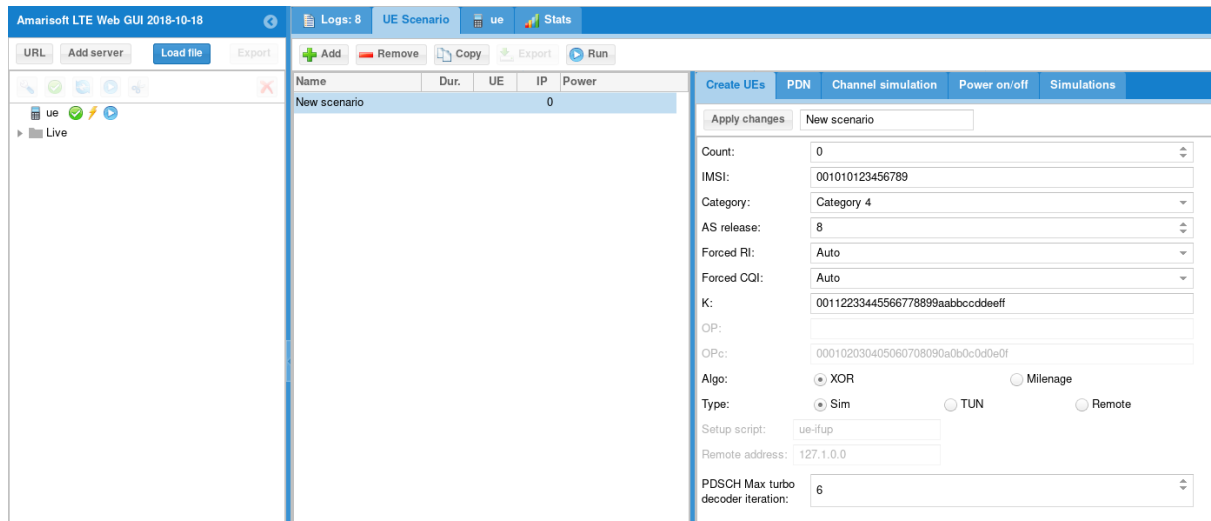
**UE Simulator** aims to control your UE simulator instance.

## 6.2 Create scenario

Once the webGui is up and running, you can create scenarios.

- In the main panel, click on the **UE Scenario** tab.

- Then click on **Add** button. A new scenario named "New scenario" appears now in the main panel
- Select **New scenario** to display the scenario panel on the right side.
- Define a name for this scenario and configure parameters for all tabs as described below.



### 6.2.1 Create UEs tab

**Count** Number of UE to create. If set to 0, the scenario shall only be applied to an already created UE.

**IMSI** IMSI of each UE. The special character \$ can be added to differentiate each UE. It will be replaced by the UE index.

NB: Same IMSI can be used for all UEs if EPC can handle it and differentiate UEs based on their IMEI as instance. This is actually supported by Amarisoft EPC when `multi_sim` parameter is true in user database file (`ue-db-ims.cfg`).

IMSI can be generated by introducing `${F(i)}` inside IMSI digits where `F` is a Javascript mathematical function and `i` the index of UE. The absolute integer rounded result padded with 0 will be put inside IMSI.

Example:

```
00101${1000 * i + Math.sqrt(i)}
```

Will produce 001010000001001, 001010000002001, 001010000003001, 001010000004002...

**category** UE category.

**Forced RI** Force RI returned by UE to base station. 1 for SISO, 2 for MIMO

**Forced CQI**

Force CQI returned by UE to base station. If set to 0, UE will estimate it. It's recommended to set it at 15 for testing at max throughput in wired condition

**K/OP/OPC/Algo :**

Secret key parameters. To be aligned with IMSI parameters stored in HSS so that the UEs can be authenticated correctly. See `lteue.pdf` for more details

**Type** Allow to select simulation mode between **SIM**, **TUN** and **Remote**



**Sim** mode: Uplink and downlink traffic flow of each UE is managed by an embedded traffic generator called "lte simserver" running on EPC side. No logical TUN interface is created per UE.

**TUN** mode: Create a linux TUN interface per PDN for each UE.

This allows to communicate with each UE independently through their network namespaces.

Lte\_simserver is not required in that case. Iperf command can be used for generating or receiving IP traffic as instance. See [Downlink and Uplink transfer using iperf], page 33, for more details.

**Remote** mode: Used with remote UE mode. See [Remote UE], page 44, for more details.

#### Setup script:

Select script used for TUN and Remote mode.

### 6.2.2 PDN tab

PDN parameters are optional. If the tab remains empty, the UE(s) will attach to the network and request a PDN connectivity without specifying the APN. Network will provide the default APN in that case.

To request a PDN with a specific APN and change the PDN type, a new PDN profile has to be created using the following parameters :

<b>Name</b>	Access point name (APN) requested in the PDN connectivity request
<b>Type</b>	PDN type. Can be IPv4 only, IPv6 only, IPv4v6 or Ethernet
<b>connect</b>	Define when PDN connectivity request will be triggered. Can be done during the UE attach procedure or on demand, i.e when the simulation will start the transfer on this specific APN.

Note : PDN tab only aims to declare and configure the PDN. To trigger a PDN with the specific APN declared, a simulation must be created and refer to this APN. See [Simulations tab], page 21,

### 6.2.3 Channel simulation tab

Channel simulation parameters are optional. If the tab remains empty, RX and TX signals respectively received and transmitted by the UE(s) will remained unchanged. No simulation will be applied.

In order to simulate different channel conditions, See [Channel simulator], page 36, for more details

### 6.2.4 Power on/off tab

Power on/off parameters are optional. If the tab remains empty, the UE(s) declared in the create UEs tab will be created but won't be turned ON automatically.

If **enabled** box is checked, simulation will generate on and off periods for each UE and run simulations defined in **Simulations** tab.

**Duration** Duration of the simulation in seconds. All simulations and power off/on commands will be over before this duration. It can be seen as the maximum simulation duration.

**Connection attempts/s**

Number of maximum UE connection attempt per second.

**Max simultaneous connected UE**

Maximum number of simultaneously connected UE. Simulation will avoid any power on until this limit is reached, in other words, next power on will occur after new power off.

**Power on duration**

Duration in seconds of power on period. UE will remain powered on during this time and then will power off, allowing a new UE to connect.

**Power off duration**

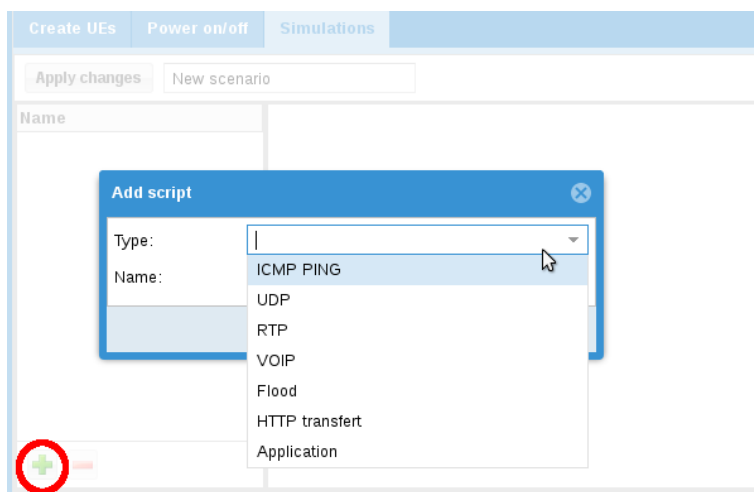
Minimum duration in seconds of power off period. When powered off, a UE will remain powered off at least this time before being candidate to power on again.

Note: UE simulator will try to put as many simulation as possible within the **duration** period, depending on parameters.

## 6.2.5 Simulations tab

The simulation tab aims to create different type of IP traffic simulation. You can add several simulations per scenario. Each simulation will be placed inside each power on period of each UE.

Click on "+" button and select simulation type:



You can then choose the following IP traffic:

**ICMP Ping** Perform ICMP Ping request.

**UDP** Send UDP constant bitrate traffic.

**RTP** Send RTP constant bitrate traffic.

**VOIP** Simulate voice RTP traffic using statistical model.

**Flood** Send UDP packet burst

**HTTP transfert**  
Send HTTP requests.

**Application**  
Launches an external application. See [External application], page 45, for more details

Once IP traffic selection has been done , additional parameters can be configured:

The screenshot shows the 'Simulations' configuration window. At the top, there are tabs: 'Create UEs', 'PDN', 'Channel simulation', 'Power on/off', and 'Simulations'. Below the tabs, there is a button 'Apply changes' and a text input 'New scenario'. A 'Simulations:' dropdown shows 'UDP #0' with a green plus icon and a red minus icon. The configuration fields below are:

- Type: UDP
- Name: UDP #0
- Probability: 1
- APN: (empty)
- Start delay: 0, +/- 0
- Duration: 5, +/- 0
- Payload size: 1000
- Bitrate : 1000000
- Destination: 192.168.3.1
- Direction: ☐ DL/UL, ☒ DL, ☐ UL

**Name** Name of simulation

#### Probability

Between 0 and 1. When scenario is running, probability that a UE instance runs the simulation depends on probability (1 = 100% probability, 0 means no chance, 0.5 means one out of two) .

**APN** Allow to request a specific Access Point Name in the PDN connectivity request. The APN name defined here must be declared previously in the PDN tab. See [PDN tab], page 20,

#### Start delay

Script start delay in seconds. If power off/on procedure is not activated, script starts after this delay. If power on/off procedure is activated, you should always set a delay as power on procedure may take a while unless it is what you want to do.

**Duration** Duration of the script in seconds.

#### Payload size

Size of the IP packet in Bytes

**Bitrate** Traffic Bitrate in bit per second

#### Destination

Destination address of IP traffic.

In **sim** mode, ltesim\_server peer entity running on EPC must listen to the same IP address ( `./ltesim_server -a 192.168.3.1`)

#### Direction

IP traffic direction. Can be downlink only, Uplink only or both Uplink/downlink

Once all parameters have been configured, click on **Apply changes** to commit them. Your scenario will now appear in the center panel and is ready to be used !

### 6.3 Export scenario

Once a scenario has been created, it's possible to export it. In the center panel, the **Export** button generates a json config file that you can directly integrate in your UE configuration file for running it through command line instead of using the WebGUI.

It also generates a MME config file that can be loaded on Amarisoft EPC to fill automatically the ue\_db of MME component.

## 7 Run UE scenarios with WebGui

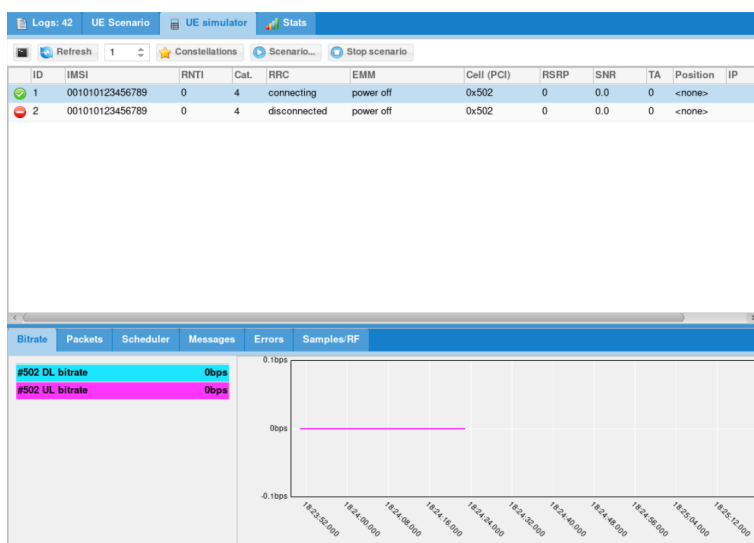
Once scenario have been created (as described in section above), you can now run them to test your eNodeB and EPC.

In the center panel, select the scenario and click on **run** button

Once started, two tabs **UE simulator** and **UE simulator:scenarioName** can be used to monitor the UE status, visualize chart and trigger manual actions

### 7.1 UE simulator tab

When selecting simulator UE tab, several panels are displayed.



#### 7.1.1 Actions buttons

Panel on top aims to trigger actions:

- **monitor** button: open linux monitor
- **Refresh** button will force refresh of UE list. Else it is done regularly and refresh period is defined by the number field on the right.
- **Start** button will allow you to start predefined scenario. Note that only scenario that create UEs will be proposed.
- **Stop** button will stop any pending simulation on UE simulator.
- **constellations** button: show real time constellations (when **signal** check box is enabled in logging server configuration)

#### 7.1.2 UE list

Panel in the middle lists all UE created and their status.

Right click on any UE to perform more actions:

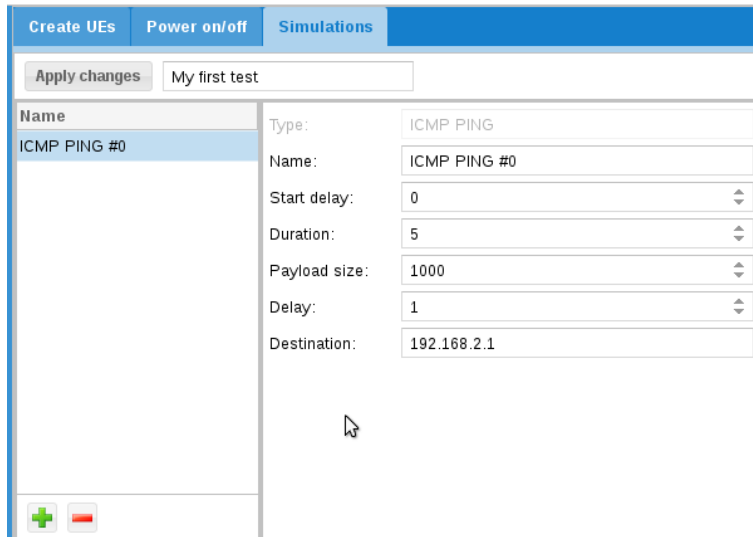
- Power on/off
- Connect pdn: enter APN for PDN to connect
- Scenario: apply scenario on this UE (Only scenario without UE creation can be used).

#### 7.1.3 Statistics

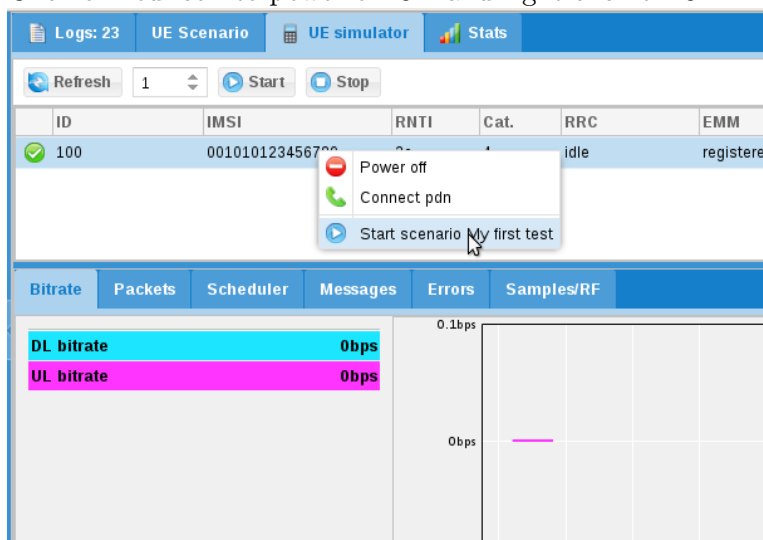
Panel at the bottom provides various real time charts for debug and tracking purpose

## 7.2 Scenario example

- First create a scenario in UE scenario tab and call it **My first test**.
- Select **Simulations** tab and add ICMP ping:

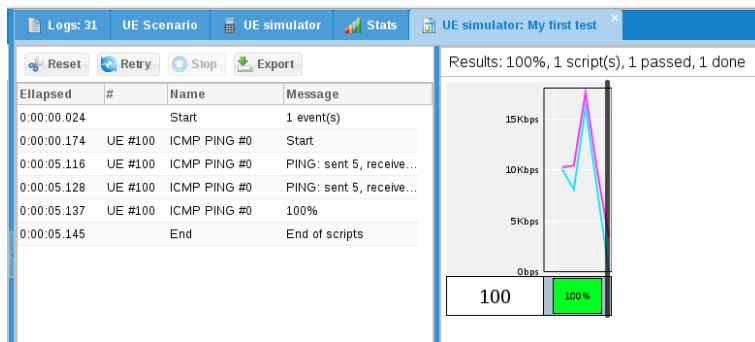


- Click on **Apply changes**
- Go to **UE Simulator** tab.
- Click on red icon to power on UE and right click on UE:



- Select **My first test**. A new tab is created to follow scenario.
- Select **scenario** tab

### 7.3 Executing scenario tab



Following buttons are available:

- Reset will flush logs
- Retry will start scenario again
- Stop will stop current scenario
- Export will export in a CSV file scenario results

### 7.4 Example

Let's try the following exercise:

- 100 UE have to be connected simultaneously.
- 20 UE will connect every second.
- Each UE will stay connected 10s
- Each UE can't be powered on less than 10s
- Scenario will last 1 minute.
- UE will perform HTTP transfer and pings.

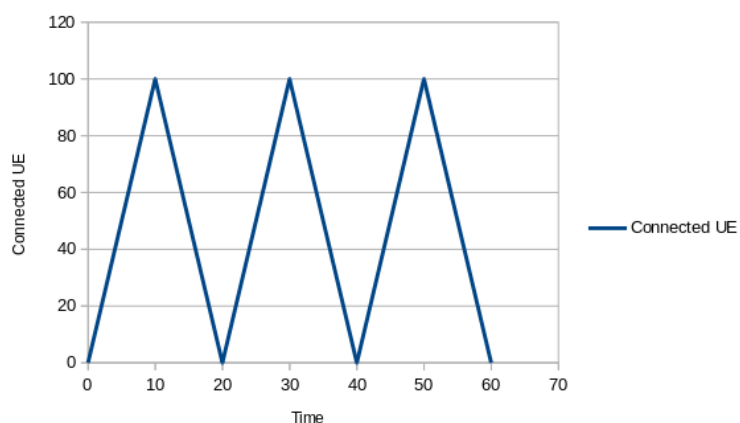
Let's create a new scenario.

First, we need to estimate the amount of necessary UE.

If we set only 100 UE:

- The first one will connect at  $t=0s$  and disconnect at  $t=10s$
- The last one will connect at  $t=5s$  (100 UE will take 5s to connect at 20 caps).
- From  $t=10s$ , UE will start to disconnect but there will be no non connected remaining UE to connect again as UE have to stay disconnected at least 10s.

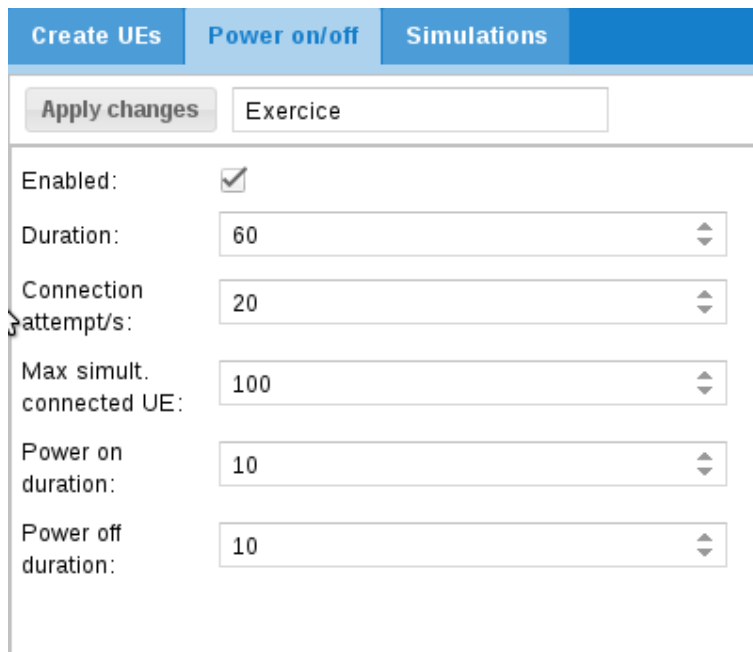
This implies between  $t=10s$  and  $t=20s$ , total amount of connected UE will decrease to 0 at  $t=20s$ . As a result:



So what can we do ?

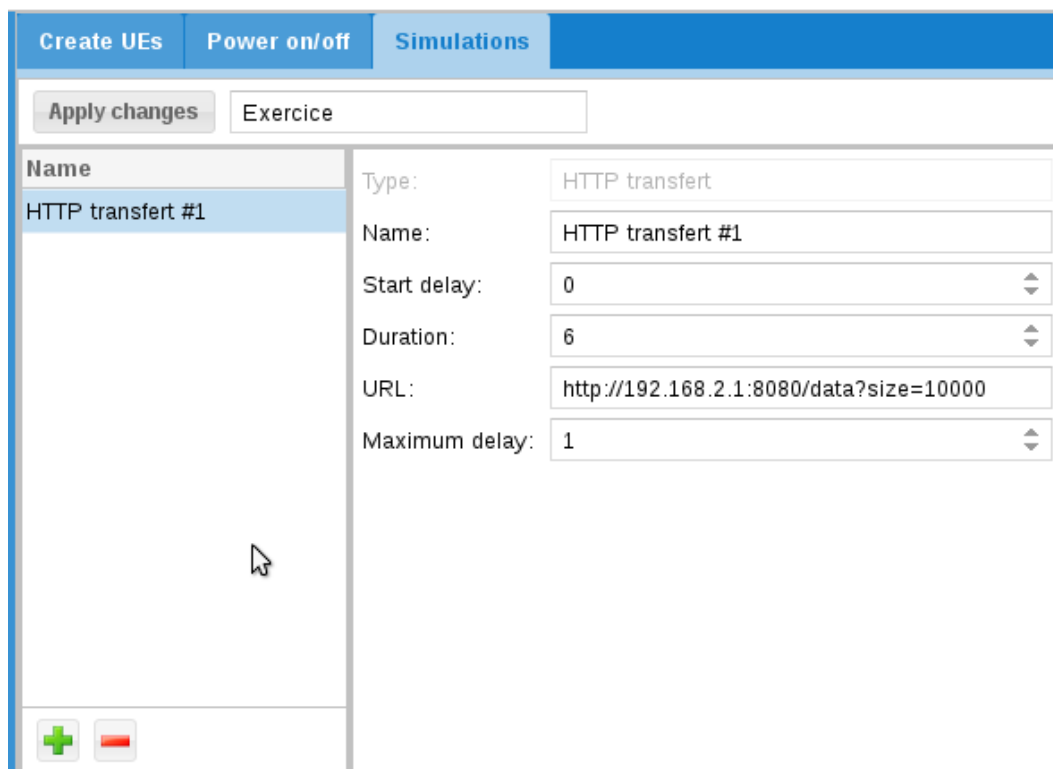
We can reduce power off duration but this will imply all UE will stay disconnected 0s !  
And we can increase the amount of UE to have a constant pool of disconnected UE.

Let's do this:



Create UEs		Power on/off	Simulations
<b>Apply changes</b>		Exercice	
Enabled:	<input checked="" type="checkbox"/>		
Duration:	60		
Connection attempt/s:	20		
Max simult. connected UE:	100		
Power on duration:	10		
Power off duration:	10		

Then we can add our scripts:



Create UEs		Power on/off	Simulations
<b>Apply changes</b>		Exercice	
Name	Type:	HTTP transfert	
HTTP transfert #1	Name:	HTTP transfert #1	
	Start delay:	0	
	Duration:	6	
	URL:	http://192.168.2.1:8080/data?size=10000	
	Maximum delay:	1	

With this configuration, HTTP transfer will last 6s.

As power on duration is 10s, it means HTTP transfer will start 2s after power on and will stop 2s before power off.



Take a look at URL: <http://192.1.168.4.1:8080/data?size=10000> This URL will be interpreted by `ltesim_server` embedded HTTP server as a transfer of 10000 byte(s). Note that `ltesim_server` must be started with HTTP server enabled:

```
sudo ./ltesim_server -a 192.168.4.1 -H 8080
```

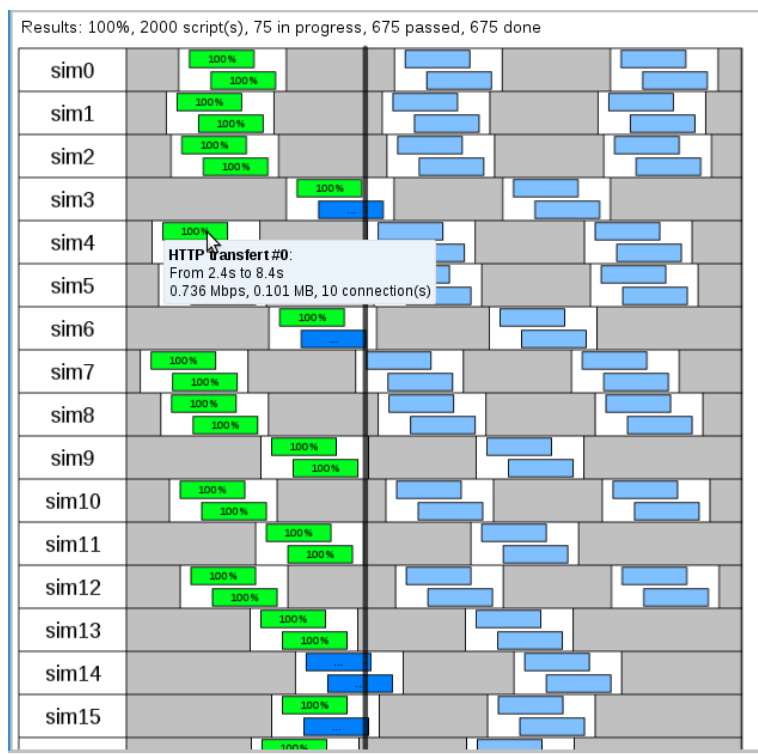
Then add ping

The screenshot shows the 'Simulations' tab of a software interface. On the left, there is a list of simulations: 'HTTP transfert #0' and 'ICMP PING #1'. The 'ICMP PING #1' simulation is selected. On the right, the configuration for this simulation is displayed in a form:

- Type: ICMP PING
- Name: ICMP PING #1
- Start delay: 3
- Duration: 6
- Payload size: 1000
- Delay: 1
- Destination: 192.168.2.1

At the bottom left, there are two small icons: a green plus sign and a red minus sign.

And start it



## 8 Create scenario without WebGui

In order to ease automation, UE scenario can be directly defined in the `ue.cfg` file and run automatically when `lte` service is started.

The exhaustive list of parameters that can be used are described in `lteue.pdf` documentation, chapter 7.3.

For creating configuration files including scenario, you can start from one of the files provided in the amarisoft releases (available under `/root/ue/config` folder) as a starting point and modify it as described in the following sections in order to create multiple UE.

Note: as described in "Export scenario" section above, another possibility consists in creating the scenario through the Web GUI and exporting it. The json config file generated can be integrated into the config file.

Note : Export button is active only when number of UE created in scenario is not Null

### 8.1 Multiple UEs

It's possible to create multiple UEs by using either the `ue_count` parameter or by . This is convenient when high number of UEs must be created manually.

When defined, `ue_count` instances of UE are created. Each of them has a unique IMEI and K values (starting from IMSI and K values and incremented by one) .

All others properties remain the same for each UEs.

When using Amarisoft EPC solution, `count` parameter can be used as well to create n user entries in EPC by incrementing the IMSI and K values

### 8.2 Examples of configuration file

#### 8.2.1 Power\_ON-OFF

In the example below, UE with IMSI 001010000000200 will automatically power on after 3 seconds and power off after 3000 seconds.

```
ue_list: [
  {
    as_release: 8,
    ue_category: 4,
    imsi: "001010000000200",
    sim_events: [
      {
        "start_time": 3,
        "event": "power_on"
      },
      {
        "start_time": 3000,
        "event": "power_off"
      }
    ]
  }
]
```

### 8.2.2 Power\_ON\_PING

The example below shows the different steps of a PING scenario for UE with IMSI 001010123456789:

- UE creation
- Power on
- Start Ping traffic after 5 seconds
- Report a (forced) CQI= 15
- Report a (forced) RI= 2
- Stop Ping after 300 seconds (305-300)

```
ue_list: [
  {
    "ue_id": 88,
    "imsi": "001010123456789",
    "K": "00112233445566778899aabbccddeeff",
    "forced_cqi": 15,
    "forced_ri": 2,
    "sim_events": [
      {
        "start_time": 0,
        "event": "power_on"
      },
      {
        "start_time": 5,
        "end_time": 305,
        "dst_addr": "192.168.3.1",
        "payload_len": 1400,
        "delay": 1,
        "id": 36457,
        "event": "ping"
      }
    ]
  }
],
```

### 8.2.3 Power\_ON\_2xUEs\_UDP\_UL

The example below shows the different steps of a UDP UL scenario with 2 UEs:

- UE creation
- Power on
- Start Uplink UDP transfer after 5 and 10 second respectively
- Stop transfer after 300 seconds

```
ue_list: [
  {
    "ue_id": 1,
    "imsi": "001010123456789",
    "imeisv": "8682430000000101",
    "K": "00112233445566778899aabbccddeeff",
    "sim_algo": "xor",
    "ue_category": 4,
    "pdsch_max_its": 6,
```

```

    "as_release": 8,
    "sim_events": [
      {
        "start_time": 0,
        "event": "power_on"
      },
      {
        "start_time": 5,
        "end_time": 305,
        "dst_addr": "192.168.3.1",
        "payload_len": 1400,
        "bit_rate": 15000000,
        "type": "udp",
        "event": "cbr_send"
      }
    ]
  },
  {
    "ue_id": 2,
    "imsi": "001010123456789",
    "imeisv": "8682430000000201",
    "K": "00112233445566778899aabbccddeeff",
    "sim_algo": "xor",
    "ue_category": 4,
    "pdsch_max_its": 6,
    "as_release": 8,
    "sim_events": [
      {
        "start_time": 0,
        "event": "power_on"
      },
      {
        "start_time": 10,
        "end_time": 310,
        "dst_addr": "192.168.3.1",
        "payload_len": 1400,
        "bit_rate": 15000000,
        "type": "udp",
        "event": "cbr_send"
      }
    ]
  }
],

```

#### 8.2.4 Uplink\_Traffic\_Specific\_APN

The example below shows the different steps for sending UDP packets with iperf using a specific APN IMS

- UE creation
- Power ON after 5s
- Trigger PDN connectivity request with APN "IMS" after 10s
- Run Uplink traffic on IMS bearer after 15s using iperf

```

ue_list: [
  {
    as_release: 8,
    ue_category: 4,
    imsi: "001010000000200",
    K: "00112233445566778899aabbccddeeff",
    /* Enable it to create a TUN interface for each UE PDN */
    tun_setup_script: "ue-ifup",

    sim_events: [
      {
        event: "power_on",
        start_time: 5,
      },
      {
        "apn": "ims",
        "pdn_type": "ipv4v6",
        "start_time": 10,
        "event": "pdn_connect"
      },
      {
        "start_time": 15,
        "event": "ext_app",
        "prog": "ext_app.sh",
        "args": ["iperf -c 192.168.4.1 -i 1 -u -b 1M -t 100"],
      },
    ],
  }
],

```

For more details about event type (flood, UDP uplink, UDP downlink, http, others.. ) and parameters please refer to lteue.pdf document, chapter 6.7 IP simulation messages

### 8.3 Remote control

Once scenario is running, it's possible to remotely control the UEs and trigger some actions (such as turning off UEs or changing rf gains) either by using the command line monitor or through WebSocket.

For more details about command line and remote API, please refer to lteue.pdf section Remote API and Command line monitor.

Example of remote command :

```
./ws.js 192.168.1.11:9002 '{"message": "power_on","ue_id":2}'
```

## 9 Downlink and Uplink transfer using iperf

For running Downlink and Uplink test, iperf can be used on both sides: Core network and LTE UE simulator.

Note that by default network namespace (netns) is required at UE side.

### 9.1 Introducing Linux Network Namespaces

A network namespace is logically another copy of the network stack, with its own routes, firewall rules, and network devices. In UE config file, when parameter `tun_setup_script: "ue-ifup"` is present, each UE gets its own (logical) Tun interface created.

You can then use it to select the UE PDN to be used for transmitting and receiving data, as described in the section below

The useful commands are :

- `ip netns list` - show all of the named network namespaces
- `ip netns exec ueX cmdX` - Execute a command within the namespace of a UE.

### 9.2 Uplink test

For sending uplink traffic from multiple UEs with iperf:

- Set `tun_setup_script: "ue-ifup"`, under `ue_list` (in `ue.cfg` file), for each UE in order to create a TUN interface for each UE PDN
- Start LTE UE service (`service lte start`), switch to LTE screen (`screen -x lte`) and power\_on UEs (`power_on <UE_ID>`)
- On EPC side, run iperf command `iperf -s -u -i 1`
- On UE side, in linux shell, type `ip netns list` to list all network namespaces  

```
[root@localhost ~]# ip netns list
ue2
ue1
```
- Select the targeted UE and generate uplink IP traffic from this UE by running the iperf command within its namespace  
 Example: `ip netns exec ue2 iperf -c 192.168.3.1 -u -b 150M -i 1 -t 100`

### 9.3 Downlink test

For sending downlink traffic to multiple UE using iperf:

- Set `tun_setup_script: "ue-ifup"`, under `ue_list` (in `ue.cfg` file), for each UE in order to create a TUN interface for each UE PDN
- Start LTE UE service (`service lte start`), switch to LTE screen (`screen -x lte`) and power\_on UEs (`power_on <UE_ID>`)
- Once registered, get eRAB IP address of each UE by running `ue` command in LTE screen  

```
(ue) ue
# UE_ID CL RNTI      RRC_STATE      EMM_STATE #ERAB IP_ADDR
0      1  0   51        idle        registered      1 192.168.2.2
1      2  0   50        idle        registered      1 192.168.2.6
```
- On EPC side, run iperf command to start Downlink transfer and select the targeted UE by using its eRAB IP address. Example :  
`iperf -c 192.168.2.2 -u -b 150M -i 1 -t 100`

- On UE side, in linux shell, type `ip netns list` to list all network namespaces  
`[root@localhost ~]# ip netns list`  
`ue2`  
`ue1`
- Select the targeted UE and execute the iperf command within its namespace  
Example: `ip netns exec ue1 iperf -s -u -i 1`

Note: any command can be run in the namespace, such as ping, ifconfig, etc..

Note: ifconfig command can be useful to link a UE network namespace to its eRAB IP address as intance

## 10 NB-IOT devices

### 10.1 Stand alone cell

When using standalone NB-IOT cell, only `dl_earfcn` has to be set in `ue.cfg` file. The `ul_earfcn` is automatically deduced

Note : If Uplink Downlink spacing doesn't follow standard values, `ul_earfcn` must be set as well

### 10.2 Guard band and in band cell

In in-band and guard band modes, RB used for NB-IOT in uplink and downlink are configured by the network. Uplink EARFCN and frequency offset values used are broadcasted via SIB2. UE SIM can not change dynamically the UL Freq based on SIB2 parameters. It's then required to set those values (`ul_earfcn` and `ul_carrier_freq_offset`) manually in the `ue.cfg` file

Example of SIB2 message:

```

    },
    uplinkPowerControlCommon-r13 {
        p0-NominalNPUSCH-r13 -80,
        alpha-r13 a109,
        deltaPreambleMsg3-r13 0
    }
},
ue-TimersAndConstants-r13 {
    t300-r13 ms2500,
    t301-r13 ms2500,
    t310-r13 ms200,
    n310-r13 n6,
    t311-r13 ms10000,
    n311-r13 n5
},
freqInfo-r13 {
    ul-CarrierFreq-r13 {
        carrierFreq-r13 24318,
        carrierFreqOffset-r13 v0
    },
    additionalSpectrumEmission-r13 1
},
timeAlignmentTimerCommon-r13 infinity
},
}

```

Note : If Amarisoft eNodeB is used, command `cell` can be used to display `dl_earfcn`

Example of `ue.cfg` file:

```

bandwidth: 1.4,
multi_ue: true,
multi_ue_type: "nbiot",
rel13_5: true,
cells: [
    {
        dl_earfcn: 6309, /* 806 MHz (Band 20) */
        n_antenna_dl: 1, /* number of downlink antennas */
        n_antenna_ul: 1,
        ul_earfcn: 24318,
        ul_carrier_freq_offset: 0,
    }
]

```



## 11 Channel simulator

The LTE UE simulator embeds a channel simulator that can be used to alter UE measurements and simulate different radio conditions.

The main parameter to activate the channel simulation is: `channel_sim`

If set to "true", the UE channel simulator is enabled.

Note: Channel simulator is only available in multi UE mode (`multi_ue = true`).

On the downlink side, depending on the simulated UE path loss, the channel simulator modifies the PER (Packet Error Rate) of PDSCH and PDCCH and updates the measured RSRP and CQI.

On the uplink side, The signal level is modified accordingly. The path loss of each UE is computed according to the corresponding UE and cell positions, channel and antenna models.

### 11.1 Cell configuration

When the UE channel simulator is enabled (`channel_sim = true`), the following additional parameters must be provided (see `lteue.pdf` file for more details):

- position
- antenna
- ref\_signal\_power
- ul\_power\_attenuation

Here is an example of cell configuration defined in `ue.cfg` file:

```
bandwidth: 20,
multi_ue: true,
channel_sim: true,

cells: [
  {
    dl_earfcn: 3350, /* 2680 MHz (band 7) */
    n_antenna_dl: 2, /* number of downlink antennas */
    n_antenna_ul: 1,
    global_timing_advance: -1,
    /* position (in meters) */
    position: [0, 0],
    ref_signal_power: 5, /* in dBm */
    /* Attenuation between the UE TX and eNodeB RX in dB. In
    LTESIM, it is equal to the path loss of the UE group. */
    ul_power_attenuation: 108,
    antenna: { type: "isotropic" },
  }
],
```

## 11.2 UE parameters

When the UE channel simulator is enabled (`channel_sim = true`), the following UE parameters must be provided (see `lteue.pdf` file for more details):

- position
- initial\_radius
- speed
- direction
- noise\_spd
- channel

Those parameters can be set manually in the `ue.cfg` file or configured automatically with the WebGui as described below.

Scenario creation through WebGui is the easiest way to simulate noise and mobility. Based on input parameters, WebGui will generate a random scenario and static channel simulation parameters.

However, if you need to get deeper control of UE position, speed and direction or set specific path loss parameters, you can set the initial value of UE list parameters in the `ue.cfg` file (See [example of `ue.list` cfg file], page 38) and use the remote API as described below to move the UE position.

## 11.3 Channel simulation scenario using WebGui

The WebGui (`ltewww`) provides options to create scenario including channel simulation.

WebGui interface :

Create UEs	Channel simulation	Power on/off	Simulations
<input type="button" value="Apply changes"/> <input type="text" value="channel simulation"/>			
Channel type:		Additive white Gaussian noise	
Max distance (m):		500	
Max speed (km/h):		50	
Noise spectral density in dBm/Hz:		-174	

To activate channel simulation through WebGui :

- Select **Channel type** = **Additive white Gaussian noise**
- Set **Min distance**: minimum allowed distance from coordinates [0,0] in meters.
- Set **Max distance**: maximum allowed distance from coordinates [0,0] in meters.
- Set **Speed**: UE speed in km/h.
- Set **Noise spectral density in dBm/Hz** : define Noise density (default value is -174).

Note that UE will be randomly placed inside the ring delimited by minimum and maximum distance

Example :

Let's define the same values as the picture above :

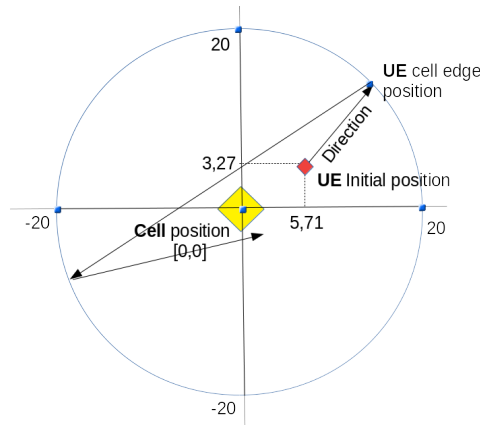
- **Channel type** = **Additive white Gaussian noise**

- Min distance = 0
- Max distance = 20
- Speed = 5
- Noise spectral density in dBm/Hz= -174

This generates the following ue\_list parameters :

```
ue_list: [
  {
    "ue_id": 1,
    "imsi": "001010123456789",
    "imeisv": "278088000000101",
    "K": "00112233445566778899aabbccddeeff",
    "sim_algo": "xor",
    "opc": "000102030405060708090A0B0C0D0E0F",
    "ue_category": 4,
    "min_distance": 0,
    "max_distance": 20,
    "noise_spd": -174,
    "position": [
      5.711861020335772,
      3.274739949279177
    ],
    "direction": 42.431151896147014,
    "speed": 5,
    "channel": {
      "type": "awgn"
    },
    /* If set, will override global parameter: channel_sim */
    "channel_sim": true,
    "pdsch_max_its": 6,
    "as_release": 8,
    "sim_events": []
  }
]
```

As a result, with this scenario, UE will move from its initial position (5.71,3.27) following the direction(42.43) at the speed of 5 km/h. Once UE has reached the max distance (20 meters) or min distance (if set) to origin, it will bounce according to the **bounce** parameter.



## 11.4 Channel simulation scenario through remote API

To get deeper control of UE position, speed and direction it's possible to use the remote API to control the UE position. First of all, ue.cfg file must be filled correctly

### 11.4.1 Cell configuration parameters to be provided:

See [cell configuration parameters], page 36,

### 11.4.2 UE configuration parameters to be provided:

- Position
- initial\_radius
- speed
- direction
- noise\_spd
- channel

see lteue.pdf for more details about these parameters.

Example of UE configuration file

```
ue_list: [
{
  /* USIM card data */
  imsi: "0010112345600",
  K: "00112233445566778899aabbccdde00",
  count: 10,
  /* UE channel simulator parameters */
  position: [15, 0], /* starting position (in meters) */
  speed: 0, /* speed (km/h) (default = 0) */
  direction: 0, /* direction (degrees) (default = 0) */
  channel: {
```

```
        type: "awgn",  
    },
```

Once configured, the scenario can be run and UE parameters changed on the fly through the remote API

### 11.4.3 Command: "ue\_move"

Move a UE to a specific position . Includes the following parameters :

- ue\_id
- position
- speed
- direction

Example of Websocket command :

```
./ws.js 192.168.1.11:9002 '{"message": "ue_move","ue_id":1,"position": 10,"speed": 5, ,"dir
```

## 12 logging

### 12.1 Logging through WebGui

For logging LTE UE messages through webGui, please refer to ltewww.pdf documentation . Default LTE UE com addr is 9002

Note: ue.log file can be fetched under your PC /tmp folder without using the WebGui

### 12.2 Command line monitor

As described in lteue.pdf command line can be used to display logging information. `t help` command provides the list of options available

#### 12.2.1 t spl command

t spl command provides key information about the UE signals

```
--TX 1----- --TX 2----- dBFS --RX 1----- --RX 2-----
  RMS   MAX   RMS   MAX   SAT   RMS   MAX   RMS   MAX
-31.4 -16.7 -32.3 -382.3   0 -40.6 -28.4 -41.3 -29.2
-31.4 -16.7 -32.3 -382.3   0 -40.6 -28.4 -41.3 -29.1
-31.5 -16.7 -32.3 -382.3   0 -40.6 -28.6 -41.3 -28.3
-31.5 -16.7 -32.3 -382.3   0 -40.6 -28.3 -41.3 -28.8
-31.4 -16.7 -32.3 -382.3   0 -40.6 -28.5 -41.3 -28.7
```

Where, for TX:

- RMS means Root-Mean-Square. This is the mean value in dB FS (full Scale) of the TX samples for each antenna. In the example above, mean transmit power is -31.4 dBFS on TX1 and -382.3 on TX2 (the current version only supports a single uplink antenna)  
Note: `tx_gain` command can be used to decrease or increase the TX gain at UE side. Initial value is set in rf\_driver configuration file.
- MAX means Maximum. Displays the maximum sample value during the measurement interval period
- SAT means Saturation. Displays the number of saturation events during the measurement interval period

And for RX:

- RMS means Root-Mean-Square. This is the mean value in dB FS (Full Scale) of the RX samples for each antenna.
- MAX means Maximum. Displays the maximum sample value during the measurement interval period  
Note: The higher is the value, the better will be the reception on UE side. The maximum theoretical value is 0. However, if you see 0 on RX 1 or RX2 this may likely signify that eNodeB signal is too strong and saturates the RX reception at UE side. Note: `rx_gain` command can be used to decrease or increase the RX gain at UE side. Initial value is set in rf\_driver configuration file.

### 12.2.2 t cpu command

t cpu command provides key informations about the CPU load.

-Proc-	---RX-----	---TX-----	---- TX/RX diff (ms)		
CPU	MS/s	CPU	MS/s	CPU	min/avg/max sigma
51.8%	23.040	6.6%	23.040	1.6%	2.23/2.8/3.3 0.2
52.9%	23.040	6.6%	23.039	1.6%	1.97/2.8/3.3 0.2
52.3%	23.040	6.6%	23.041	1.5%	2.20/2.8/3.3 0.2
51.4%	23.040	6.6%	23.040	1.6%	2.20/2.8/3.3 0.2
50.8%	23.040	6.6%	23.039	1.6%	2.19/2.8/3.3 0.2

t cpu shows CPU consumption for the main LTE task (Proc), the reception chain (RX) or transmission chain (TX). Units are MS/s (Million of sample per second)

On top of that, t cpu command provides informations on TX-RX delay: The min, average and max values are given in milliseconds. To make it simple, in LTE FDD mode as instance, UE is granted in Uplink 4 TTI (4 ms) before data have to be sent. UE has by consequence 4ms to generate the UL samples. The RX/TX delay is the remaining time before data are processed and time they must be sent in UL. If this value decreases to zero, this means that PC is too slow and physical layer is running out of CPU to process the data within this period. The higher is the TX/RX value is , the better it is.

### 12.2.3 t command

t command (without argument) provides key informations about the Uplink and downlink transfer

-----Hz-----ppm-----dB-----dBm-----DL-----															
UE_ID	RAT	CL	RNTI	CFO	SRO	SINR	RSRP	mcs	retx	rxfail	txok	brate	#its	mcs	ta
1	LTE	00	003d	-898	-3.5	32.1	-44.0	27.8	0	0	6192	209M	1/3.9/6	20.0	0
1	LTE	00	003d	-900	-3.5	32.1	-44.0	27.8	0	0	6208	210M	1/3.9/6	20.0	0
1	LTE	00	003d	-900	-3.5	32.1	-44.0	27.8	0	0	6192	209M	1/3.9/6	20.0	0
1	LTE	00	003d	-901	-3.5	32.1	-44.0	27.7	14	4	6204	208M	1/3.9/6	21.0	0

Where :

- UE\_ID is the UE ID
- RAT is the access technology (LTE, NB-IoT or NR)
- CL is the Cell ID
- RNTI is the (C)RNTI of each UE
- CFO is the Carrier frequency offset in Hertz
- SRO is the Sample Rate Offset in ppm
- SINR (Signal-to-Interference Ratio) is an indicator of the signal quality while the rsrp is an indicator of the signal strength. SINR is the ratio of Signal Power and Noise Power. Unit is in dB.
- RSRP (Reference signal received power), is the linear average over the power contributions of the resource elements that carry cell-specific reference signals within the considered measurement frequency bandwidth. Unit is in dBm
- mcs (Modulation Coding Scheme) is the average MCS value used in downlink during the measurement interval period. More details about MCS range are available in 3GPP TS 36.213
- retx (retransmission) is the number of retransmission sent by UE in Uplink or received by eNodeB in downlink. This parameter gives indications about the channel quality.
- rxfail is the number of received downlink transport blocks with CRC errors.
- rkok is the number of received downlink transport blocks without CRC error.

- brate is the average bitrate (at the MAC layer), in bits per second.
- ta is the Average of the uplink timing advance in TA units.

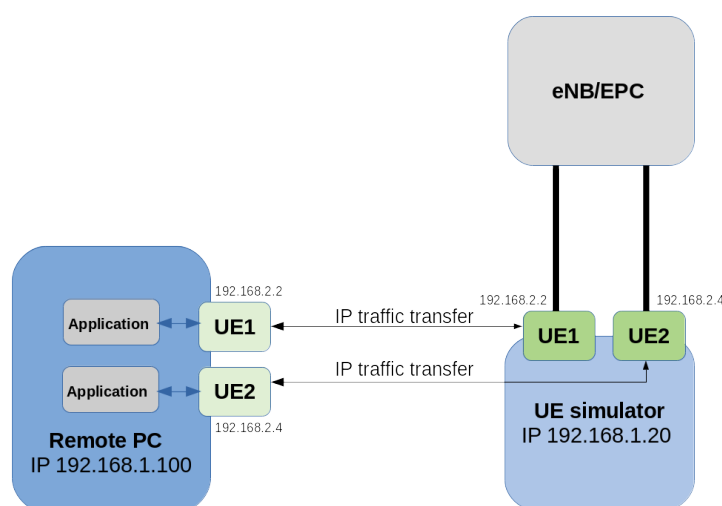
Information about the logging tool are provided in `ltewww.pdf` document. It describes how to setup the web interface and apply filter to get logs displayed.



## 13 Advanced options

### 13.1 Remote UE

When using tunnel interface with external program, you may want external program to be run on a different PC. The Remote UE tool allows you to transfer IP traffic from each UE to a remote entity.



On the remote PC :

- First copy `lterue`, `config/rue.cfg`, `libnopoll.so`, `libcrypto.so.1.1`, `libssl.so.1.1`, `config/ue-ifup` and `config/ext_app.sh` on your remote PC.
- In `rue.cfg`, change the bind address to the IP address of your remote PC (example 192.168.1.100 here)
- As root, run `lterue config/rue.cfg` command

Note : You dont need any specific license. `lterue` uses GTP over SCTP to communicate with LTEUE.

On the UE simulator PC:

- In `ue.cfg`, set `rue_bind_addr` parameter to the IP address of UE sim PC (example 192.168.1.20 here)
- As root, run `lteue config/ue.cfg` command

Setup is now configured and ready for running scenario either manually or with Webgui

When using WebGui:

- Create your scenario and in **Create UEs** tab set **type** = Remote
- Keep **setup script** value = `ue-ifup`
- Set **Remote address** value = IP address of remote PC ( example 192.168.1.100 here)
- Run the scenario

When scenario is run manually:

In `ue.cfg` file, add for each UE, under `ue_list`:

- `tun_setup_script`: "ue-ifup"
- `rue_addr`: "192.168.1.100", where 192.168.1.100 is the remote PC IP address

Now, If you go back to remote PC:

- In LTE terminal you should see your UE using `ue` command
- In another shell run `ip netns list` command. You should see the networks namespace created for each UE .See [Linux Network Namespaces], page 33, for more details
- You can run now any command within the UE Network Namespaces using command `ip netns ueX commandX`. Example: `ip netns exec ue1 ping 192.168.3.1`

## 13.2 External application

You can replace predefined simulation by a custom application. For this, choose **Application** in IP simulation list.

When started, the external application will fork a process and return its standard output and error.

To handle dedicated application, please take a look at `libsim_custom.js` file in `LTEWWW` component.

You can add specific result handler using `tag` for association.

Note that it requires associated UE to be configured in tunnel mode or with remote UE mode and thus IP simulations can't be mixed.

## 13.3 External SIM

External SIM card can be used using PCSC-Lite lib, which is supported on Amarisoft LTE UE simulator

To do so :

- Install PCSC-Lite lib on your machine (`pcsc-lite-devel`)
- In `ue.cfg` file, under `ue_list`, set `external_sim`: `true`
- If several cards are used, set `sim_reader_index`. This is optional.

## 14 Troubleshooting

### 14.1 No SIB found

When running LTE UE simulator, before creating any UE or starting scenario, the first output that should be displayed on the shell is "(ue) Cell 0: SIB found". This means that LTE UE simulator has detected the eNodeB and can read the SIB. This is a prerequisite before running any tests.

If this message is not displayed, here is a check list that can help to find the root cause:

- Check that LTE UE config file parameters match with eNodeB configuration: bandwidth, dl\_earfcn, n\_antenna\_dl
- Check correct SDR card is being used (For PCIe SDR cards , check rf\_driver config file : args: "dev0=/dev/sdr0").
- Check rx\_gain value. Should be around 60dB in wireless connection or 0dB in wired connection
- Check received signal strength with `t spl` command. See [t spl command], page 41, for more details.
- Test in wireless condition instead of conducted one. This can help to verify that RF cabling is not the culprit (see RF connection section for TF and RX gain impact)

If all checks above haven't been enough to figure out why eNodeB signal is not detected, you can also use the spectrum analyzer function embedded in the UE simulator to verify the LTE signal quality received in downlink. See `trx_sdr.pdf` documentation for more details.

### 14.2 SIB found but attach fails

When multiple UE mode is enabled, UE simulator may have difficulties to synchronize with eNodeB signal. If such a case occurs, you should see that UE is able to receive SIBs but further communications fails with bad CRC on physical layer.

This means that you should adjust the parameter `global_timing_advance` in your configuration file. The `global_timing_advance` parameter can be set automatically by using the special value -1 (`global_timing_advance:-1`). If automatic mode is set, the UE simulator uses the timing advance from the first received RAR for all UEs. This is the default behavior.

You can also manually adjust the timing advance for all UEs in case you still experience CRC errors with automatic mode. You can check TA value on eNodeB side and set it to minus 1 in UE

(`global_timing_advance = TA[enb] - 1`). if you are using simulator with Amarisoft eNodeB, you can type `t` at eNodeB screen and look at PRACH traces.

Then, use `ta` value minus one as `global_timing_advance`.

```
PRACH: cell=01 seq=17 ta=2 snr=18.5 dB
```

```
PRACH: cell=01 seq=22 ta=2 snr=18.0 dB
```

```
PRACH: cell=01 seq=23 ta=2 snr=18.5 dB
```

```
PRACH: cell=01 seq=29 ta=3 snr=17.6 dB
```

In this example, adjust `global_timing_advance` to 1.

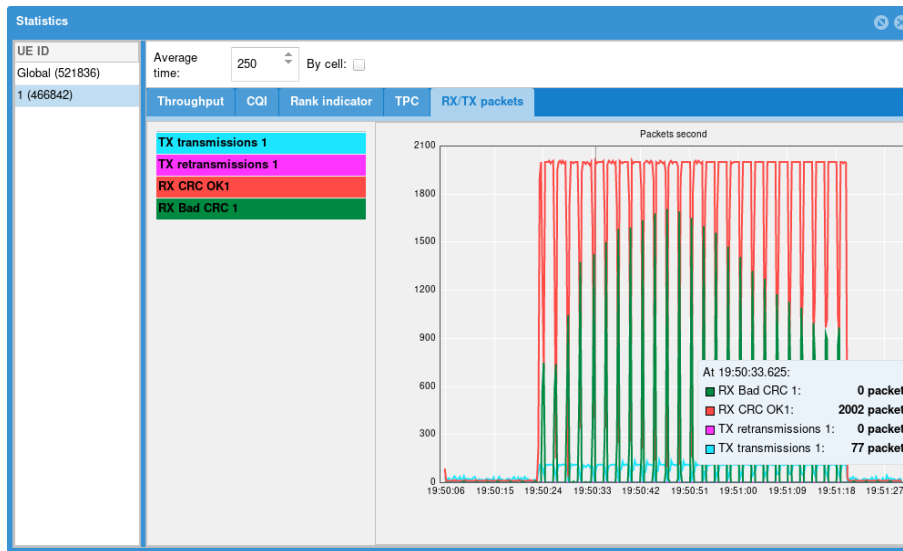
If you are using another eNodeB and you do not have access to eNodeB logs and information, you can enable the PHY and MAC layer logs in UE simulator and look for `ta` value in MAC traces

```
12:13:37.086 [MAC] - 0001 ta=13 ul_grant=128768 c_rnti=0x0047
```

In this example, you should set the `global_timing_advance` to 12.

### 14.3 Downlink peak throughput not achieved

If peak throughput is not achieved in Downlink, a first investigation can be done at physical layer using the webGui. In the main panel, click on **analytics** button. A new window will pop up. Note : physical layer trace level must be set to **debug** prior to the testing.



In this new window, several statistics are displayed and provide useful information about UE performance

#### Throughput

Displays Downlink and Uplink throughput at physical layer

#### CQI

CQI values reported by the UE to the eNodeB. A low CQI value means that Downlink channel quality is low and results in low throughput in downlink as eNodeB will use small MCS

#### Rank indicator

Rank Indicator values reported by UE to the eNodeB. A RI of 0 means that UE is not able to receive data in MIMO mode (TM3) and results in lower throughput . This can be caused by a low SNR, or a high correlation between two antenna stream or simply because the `n_antenna_dl` parameter as been set to 1 in the `ue.cfg` file

#### TPC

TPC (Transmit power control) average values received by the eNodeB. Note: When channel simulation is disabled, TPC command are not taken into account by the UE. The transmit power is set in the configuration file

#### RX/TX packets

Display number of packet received and sent with good and bad CRC (cyclic redundancy check) This graphic give a good overview of the downlink/uplink performance. In perfect radio conditions, no packet with bad CRC should be received by UE or eNodeB. Having a high number of bad CRC indicates that RF path is not optimal or that UE is running out of CPU.

`t cpu` command provides CPU indication at UE side.

- If webGui is used for downlink, verify that `ltesim_server` service is running on eNodeB (`./ltesim_server -a 192.168.3.1`) and check that scenario created under webGui uses the same IP address.
- Force CQI and RI values (RI reported value is not dynamic and forced CQI value may give better throughput than the one calculated by the LTE UE simulator based on downlink signal quality)

- If Downlink transfer is run with iperf, verify that bitrate set in iperf is higher than expected throughput (at least 10%)

## 14.4 RRC\_STATE locking value

When running NB-IOT test, the RRC state may report "Locking" value . This is visible when typing "ue" command in LTE UE screen. This state means that LTE UE has not been able to synchronize on eNodeB side. This is likely due to wrong EARFCN or RX signal level

## 14.5 Connection to webGui fails

If your browser fails to load the webGui page (xx.xx.xx.xx/lte), check that :

- httpd service is running on LTE UE SIM side (service httpd status)
- Test with another web browser (Firefox, Chrome or Internet Explorer). Avoid beta version of web browser

If the main page can be loaded but connection to server fails, check that

- LTE service is running on LTE UE side (service lte status)
- Server configuration is correctly set, i.e ip address and port com. Port com default value is 9002
- No firewall blocks the IP traffic between PC and LTE setup

## 14.6 TRX discontinuity too wide

TRX discontinuity too wide means that frames number on cell 1 and 2 are misaligned . This may have several root causes.

- Simply reboot your setup. This can solve the problem otherwise, follow steps below.
- If you have more than one SDR cards, check that SDR cards are synchronized with each other with the USB cable. You must also identify the sdr mapping to know which connector shall be used (input or output). Please see installguide.pdf for more details
- Check that both eNodeB cells are synchronized (frame boundary and Frame number) . LTE UE simulator doesn't support asynchronous cells.
- If problem occurs during data transfer test, this can happen when PC is running out of CPU. To check it:
  - Run lte\_init.sh script available under /root/ue directory.
  - Run the Linux top - H command to check if unexpected process are loading the CPU.
  - Remove unnecessary drivers and peripherals such as WiFi dongle.
  - If you use a HDMI monitor, disconnect it and access the system remotely by SSH for instance.
  - Ensure that your CPU is fast enough. It should be at least a Intel 4 Core i7 with a clock running at 3.6 GHz.
  - Remove unused daemons or cron jobs.