

E-UTRAN USER GUIDE

Editor:	EURECOM
Deliverable nature:	Public
Due date:	July, 2015
Delivery date:	July, 2015
Version:	0.1
Total number of pages:	27
Reviewed by:	
Keywords:	LTE, eNB, UE, S1AP, GTP

List Of Authors

Company	Authors
Eurecom	Christian BONNET, Lionel GAUTHIER, Rohit GUPTA, Florian KALTENBERGER, Raymond KNOPP, Navid NIKAIEN, Cedric ROUX.

Revision History

The following table is a record of the main modifications done to the document since its creation.

Version	Author	Date	Description
1.0	Navid Nikaein	20/07/2015	Initial Draft
2.0	Rohit Gupta	14/11/2016	Revised Draft

Executive Summary

The deliverable presents the eNB, UE developed by EURECOM.

The document presents the deployment scenarios of the E-UTRAN, its configuration, installation and running.

Note: While we keep the document up to date, but it is not possible all the time due to frequent updates to our code. Gitlab wiki content takes precedence over this document. If you find errors with this document, please help us fix it.

1 Introduction

The EURECOM eNB is a bundle of software components that provides the eNB functions of the LTE on both radio interface (i.e. Uu) and core network interfaces (i.e. S1-C and S1-U).

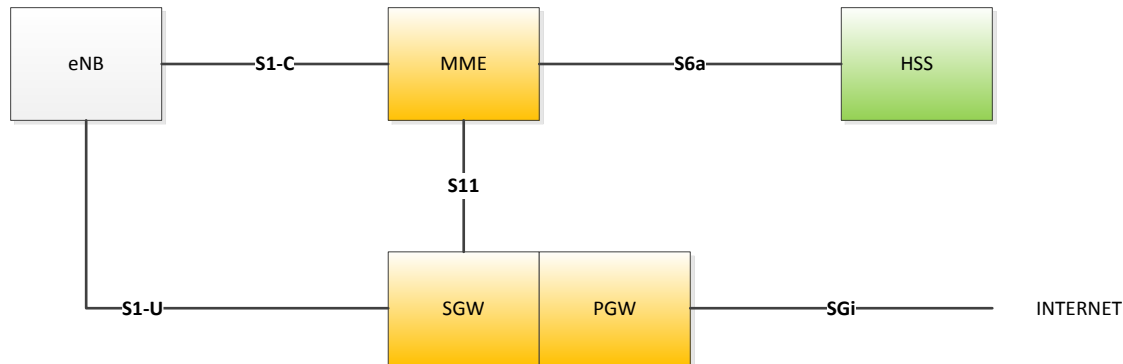


Figure 1 LTE overview

1.1 Deployment scenarios

Different deployment scenarios can be considered with the EURECOM eNB and UE as follows [1]:

- Commercial UE <-> OAI eNB + Commercial EPC
- Commercial UE <-> OAI eNB + OAI EPC
- Commercial UE <-> Commercial eNB + OAI EPC
- OAI UE <-> Commercial eNB + OAI EPC (experimental)
- OAI UE <-> Commercial eNB + Commercial EPC (experimental)
- OAI UE <-> OAI eNB + Commercial EPC (experimental)
- OAI UE <-> OAI eNB + OAI EPC
- OAI UE <-> OAI eNB

Below we present few of them.

1.2 eNB without S1 interface

In this deployment, S1AP and GTP protocols are bypassed, and thus eNB exchange the IP packets with the upper layer through the OAI network device driver called nasmesh.

To setup a radio link, you require OAI UE with the network device driver and without the NAS protocol.

One example scenario here is LTE-u.

1.3 ENB with S1 Interface

In this deployment, eNB is built with the S1AP and GTP protocols and interacts with EPC. Different EPC may be connected to the OAI eNB in addition to the OAI EPC.

Below, we provide few examples with OAI EPC.

1.3.1 eNB with all-in-one OAI EPC platform

The following picture depicts a EURECOM eNB and EPC providing MME and GW functions, and interact with the EURECOM HSS. In this deployment scenario, the S11 interface is virtual in the sense that S11 messages do not go through the network layer but through an inter-task interface message passing middleware (ITTI).

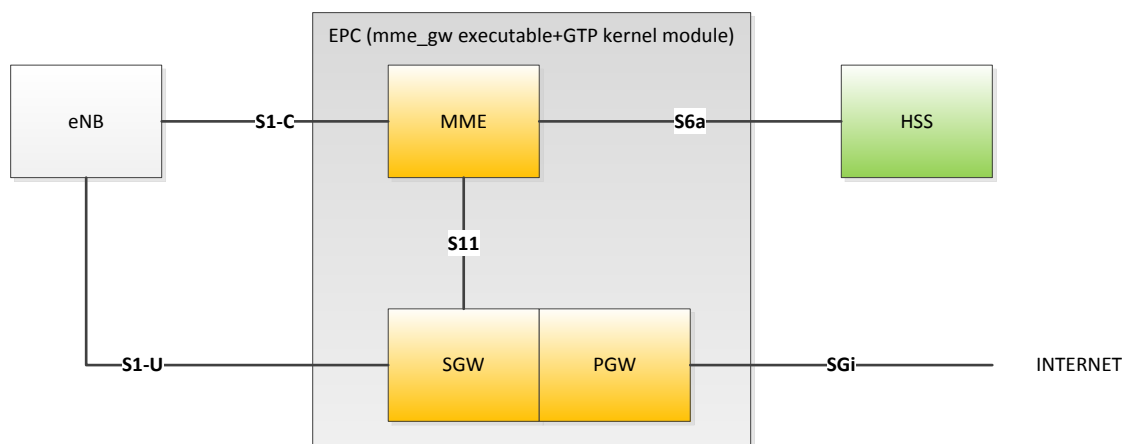
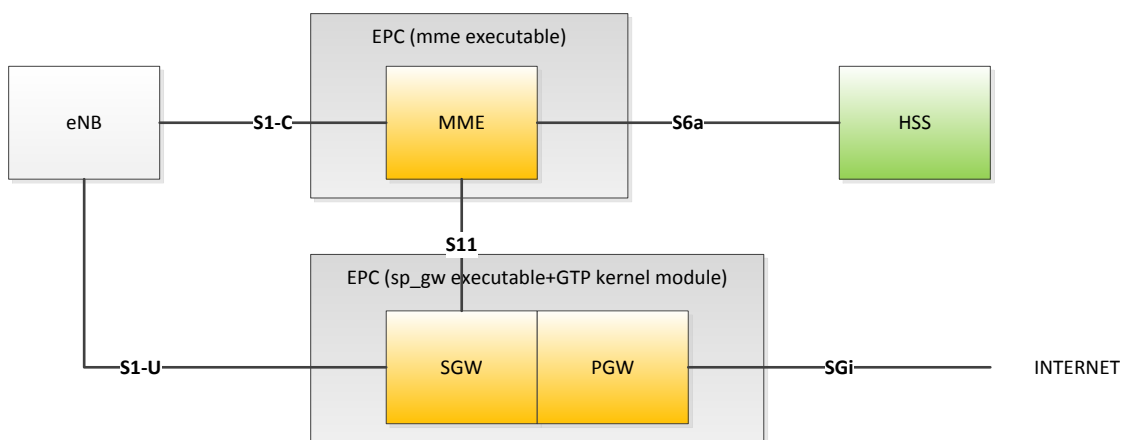


Figure 2 OAI eNB with all-in-one OAI EPC

The EPC can be deployed on the same host as OAI eNB host or on its own host.

1.3.2 eNB with separate EPC platform

Actually this deployment scenario is under development and cannot be demonstrated yet.



1.4 eNB on virtualized environments

See [2] [5].

1.4.1 Containers (LXC and Dockers)

See [2] [5].

1.4.2 KVM

See [2] [5].

2 eNB Installation

The eNB and UE software have only been tested on UBUNTU 14.04x64, and UBUNTU 14.10x64 LINUX distributions on Intel x86 64 bits platforms, and to less extend on Debian.

In addition, low latency kernel is required. For kernel installation, please refer to [3].

If you want to try another LINUX distribution, it is mandatory to have a 64 bits LINUX distribution.

2.1 eNB source code

The OpenAirInterface software can be obtained from our gitlab server. You will need a Git client to get the sources (on Ubuntu Linux the client can be install using the command "apt-get install git"). The openair5G repository is currently used for main developments.

Depending on what is recommended on the openair mailing list

<https://gitlab.eurecom.fr/oai/openairinterface5g/wikis/MailingList>, you should use the develop branch.

If git is not installed on your computer, execute in a shell the following command:

```
user@host:~$ sudo apt-get install git
```

Then to retrieve the source code, if you have read-only access, execute in a shell the following command:

```
user@host:~$ git clone https://gitlab.eurecom.fr/oai/openairinterface5g.git
```

The source code in a release directory or in the trunk directory is organized as follow:

- cmake_targets : Openair build system (latest)
- common : Common code to all layers
- openair1 : Physical layer source code
- openair2 : Layer 2(MAC, RLC, RRC, PDCP) source code

- `openair3` : Middleware code (mainly unused).
- `targets` : Specific code for executables (may contains unsupported old build system).

Important!

- In this document `OPENAIR_DIR` is the path to the openair working directory

2.2 eNB additional software

Some software installations have to be done prior to build the EURECOM eNB/UE.

In `OPENAIR_DIR/cmake_targets` directory, execute the following command:

```
user@host:~/openairinterface5g/cmake_targets$ ./build_oai-I
```

Optionaly add: `--install-optional-packages -w USRP` (For USRP Specific Installation)

Refer to Gitlab Wiki, <https://gitlab.eurecom.fr/oai/openairinterface5g/wikis/OpenAirSoftwareSupport> for HW/SW requirements and updated installation instructions

This command will update the software source list of your Ubuntu installation. It will install miscellaneous software packages.

2.3 eNB configuration

OpenAirInterface (OAI) offers configuration files for facilitating the determination of the parameters for each component.

The top level parameters configuration for multiple eNBs is divided in 6 main sections:¹

- **Main parameters:** Configuration of the base station ID, tracking area code (TAC), and mobile country code (MCC) and mobile network code (MNC).
- **PHY parameters:** Configuration of the physical layer parameters (i.e., frequency, power control, tx/rx number of antennae, tx/rx gain, hopping etc.)
- **SRB parameters:** Configuration of special radio bearers parameters (i.e., poll retransmission timer, reordering timer etc.)
- **MME parameters:** Configuration of MME parameters (i.e., IPv4/IPv6 addressing etc.)
- **Network interfaces:** Configuration of network interfaces (i.e., eNB S1-U IPv4 address, eNB S1-MME IPv4 address etc.)
- **log config:** Configuration of logger's level and verbosity by taking into account all the layers and components of network (i.e., PHY, MAC,RLC, PDCP, HW etc.)

¹ An example of eNB config file is located at:

<https://gitlab.eurecom.fr/oai/openairinterface5g/blob/develop/targets/PROJECTS/GENERIC-LTE-EPC/CONF/enb.band7.tm1.usrpb210.conf>

A detailed description of the parameters that are configured is given in Annex A (eNB configuration content).²

Figure 3 shows the view of the build process of OAI eNB, and how configuration and binary files are generated

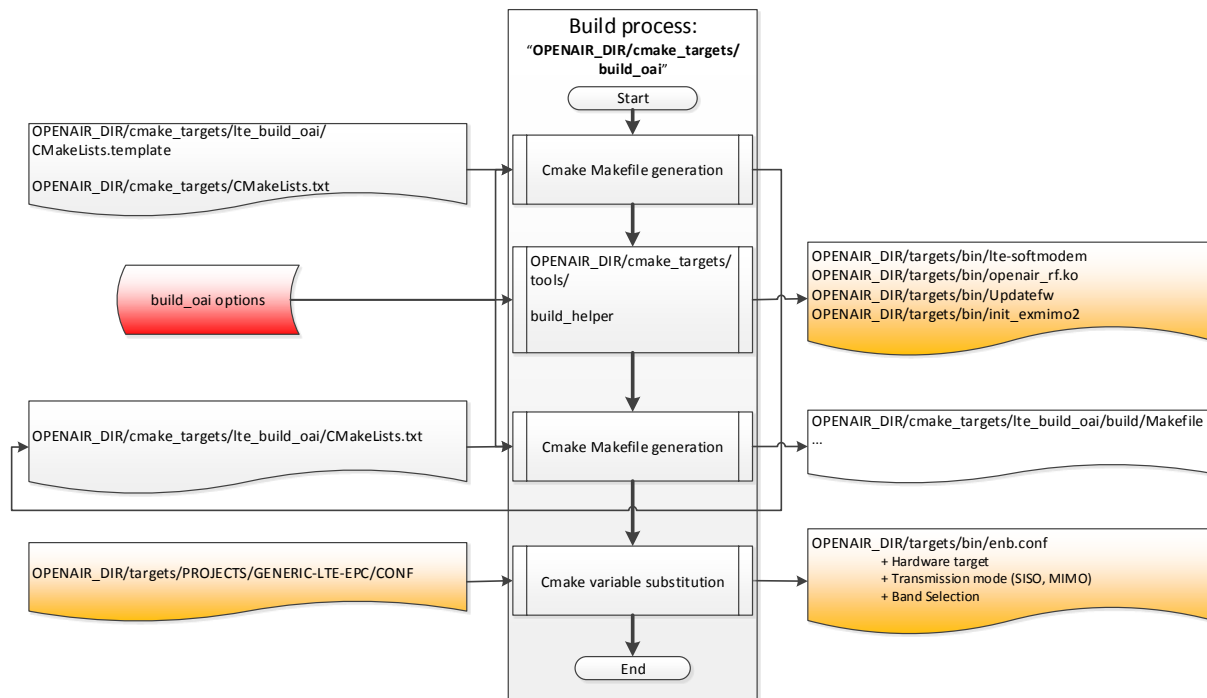


Figure 3 OAI eNB build process and configuration

Inputs files and parameters are on the left part of the figure, the build process is in the center part and output configuration files are on the right side of the figure.

eNB configuration file contents are detailed in Appendix A.

Configuration files have to be filled prior to compilation.

2.4 eNB build and Run

open a shell, and run

- user@host:~/openairinterface5g/cmake_targets\$./build_oai -h

Different targets are available as follows:

² The type and the declaration of the parameters as variables can be found in the corresponding C file:

https://gitlab.eurecom.fr/oai/openairinterface5g/blob/develop/openair2/ENB_APP/enb_config.c

2.4.1 EXMIMO Target

To build (for EXMIMO):

- user@host:~/openairinterface5g/cmake_targets\$./build_oai -eNB -w EXMIMO -x -c

To run: (for EXMIMO)

- user@host:~/openairinterface5g/targets/bin\$./init_exmimo2
- user@host:~/openairinterface5g/targets/bin\$ sudo -E ./lte-softmodem.Rel10-O ..
../PROJECTS/GENERIC-LTE-EPC/CONF/enb.band7.tm1.exmimo2.conf -S -d

for Help

- user@host:~/openairinterface5g/targets/bin\$ sudo -E ./lte-softmodem.Rel10-h

2.4.2 USRP Target

To build:

- user@host:~/openairinterface5g/cmake_targets\$./build_oai -g -eNB -X -w USRP

To run :

- user@host:~/openairinterface5g/targets/bin\$ sudo -E ./lte-softmodem.Rel10-O ..
../PROJECTS/GENERIC-LTE-EPC/CONF/enb.band7.tm1.usrp210.conf -S -d

for Help:

- user@host:~/openairinterface5g/targets/bin\$ sudo -E ./lte-softmodem.Rel10-h

2.5 UE build and Run

2.5.1 EXMIMO

To build:

- user@host:~/openairinterface5g/cmake_targets\$./build_oai --eNB -UE -x -c -w EXMIMO

Note: The flag -UE builds the UE Sim Card parameters. The UE softmodem is also built when you pass -eNB parameter.

To run:

- user@host:~/openairinterface5g/targets/bin\$./init_exmimo2
- user@host:~/openairinterface5g/targets/bin\$ sudo -E ./lte-softmodem.Rel10-U -C
2680000000 -r25 --ue-scan-carrier -d

2.5.2 USRP

To build:

- user@host:~/openairinterface5g/cmake_targets\$./build_oai --UE=eNB -x -c -w USRP

To run:

- user@host:~/openairinterface5g/targets/bin\$ sudo -E ./lte-softmodem.Rel10-U -C 2680000000 -r25 --ue-scan-carrier -d

2.6 OAISIM build and Run

To build:

- user@host:~/openairinterface5g/cmake_targets\$./build_oai -x -c --oaisim

to Run:

- With local S1: user@host:~/openairinterface5g/targets/bin\$ sudo -E ./oaisim.Rel10-O ../PROJECTS/GENERIC-LTE-EPC/CONF/enb.band7.generic.oaisim.local_mme.conf
- Without S1: user@host:~/openairinterface5g/targets/bin\$ sudo -E ./oaisim.Rel10-O ../PROJECTS/GENERIC-LTE-EPC/CONF/enb.band7.generic.oaisim.local_no_mme.conf

2.7 Test OAI

user@host:~/openairinterface5g/cmake_targets\$./build_oai-s

The test cases can run for long time. It is best to run specific tests. For more information, see

<https://gitlab.eurecom.fr/oai/openairinterface5g/wikis/OAITestSetup>

3 eNB Monitoring tools

OAI provides monitoring tools such as network protocol analyzers, loggers, performance profilers, timing analyzers and command line interfaces for performing the intended measurements and monitoring the network. Specifically, the supported monitoring tools are:

- OAI Soft Scope and Status can be read remotely.
- Wireshark / PCAP Interface:
 - L2/L3 and S1 interface.
 - Localhost or remote host access.
 - Example can be found at :
- OAI Logger and dissector (called itti_analyzer):
 - Remote log file retrieval
- OpenAirInterface performance profiler: for processing time measurement.
- OpenAirInterface timing analyzer: build the VCD
 - Convert the ascii variables and function calls output to vcd format (value change data) allowing to view the temporal function and variable changes in a vcd viewer, such as gtkwave.
- OAI message sequence chart (MSC) : <http://www.mcternan.me.uk/mscgen/>.

- CLI interface.

The figures below depict the provided monitoring tools and their functionality via typical measurement and packet capture examples. In addition, MSC examples are illustrated.

OAI Soft Scope provides plots for received signal power, channel impulse response, channel frequency response, channel frequency response, LLRs, throughput and I/Q components (e.g., 4-QAM constellation). This tool offers a complete overview of the PHY layer characteristics, an example is shown in Figure 4.

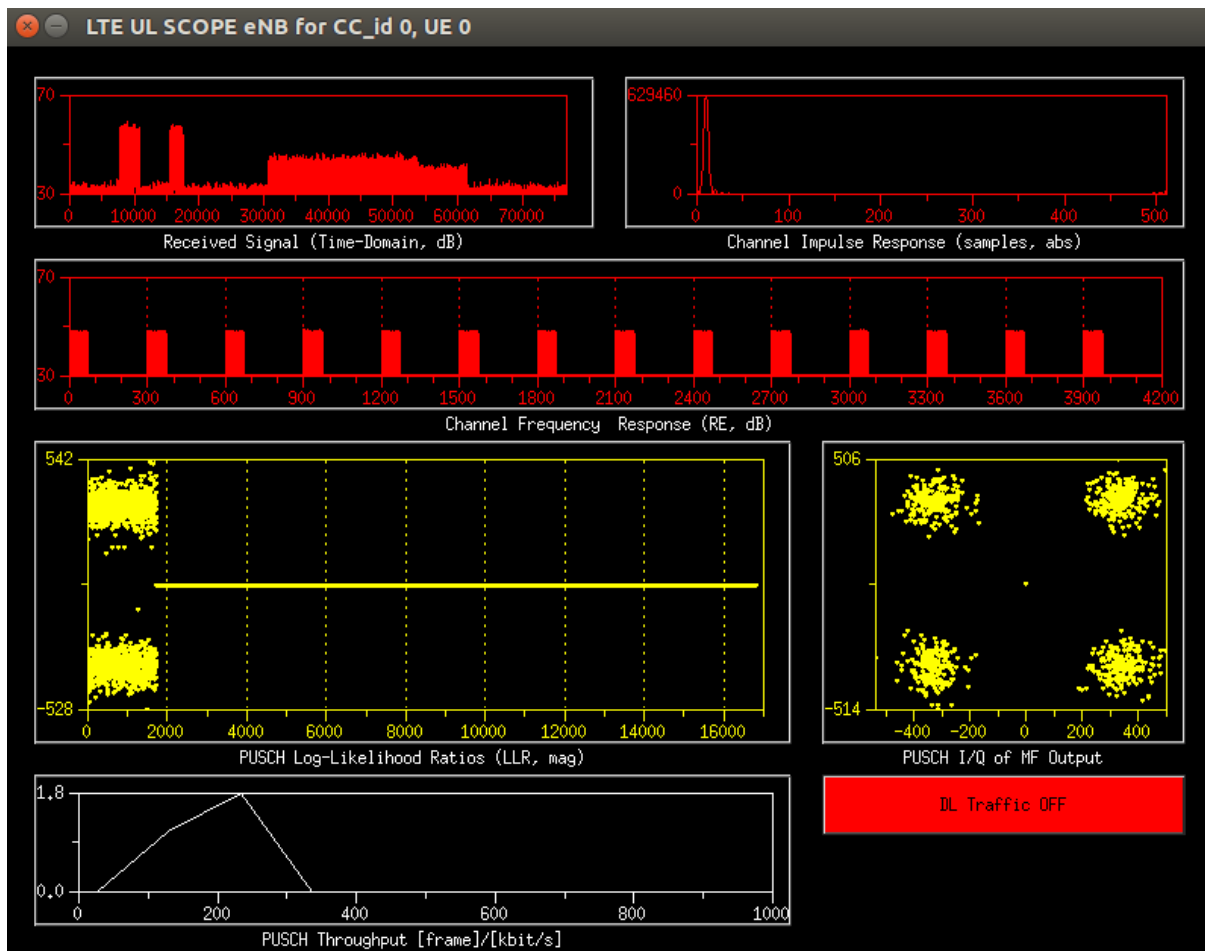


Figure 4: OAI Soft Scope: Received Signal, Channel Impulse Response, LLRs, I/Q and Throughput plots

Wireshark is a powerful tool widely used for capturing packets, offering integrated sorting and filtering options. In addition to layer 3 protocol dissection, OpenAirInterface implements the Wireshark interface

The image displays the Wireshark 2.12.1 network protocol analyzer interface. The top pane shows a list of 26 captured packets, primarily RRC (Radio Resource Control) messages between 127.0.0.1 and 127.0.0.1. The middle pane provides a detailed view of the selected packet (packet 26, RRCConnectionSetupComplete), showing its structure and fields such as rrc-TransactionIdentifier, criticalExtensions, and dedicatedInfoR8. The bottom pane displays the raw packet data in hexadecimal and ASCII format.

The itti_analyzer tool can be used to analyze the exchanges between RRC<->S1AP, RRC<->PDCP, PDCP<->S1. In addition to the protocol information, the itti_analyzer analyses all the messages exchanged between different protocols. It allows message filtering on per time, channel (CCCH, DCCH, DTCH), sender, and receiver basis allowing to monitor and follow the protocol functionalities. Figure 6 depicts an example for RRCConnectionReconfiguration message capture using the itti_analyzer.

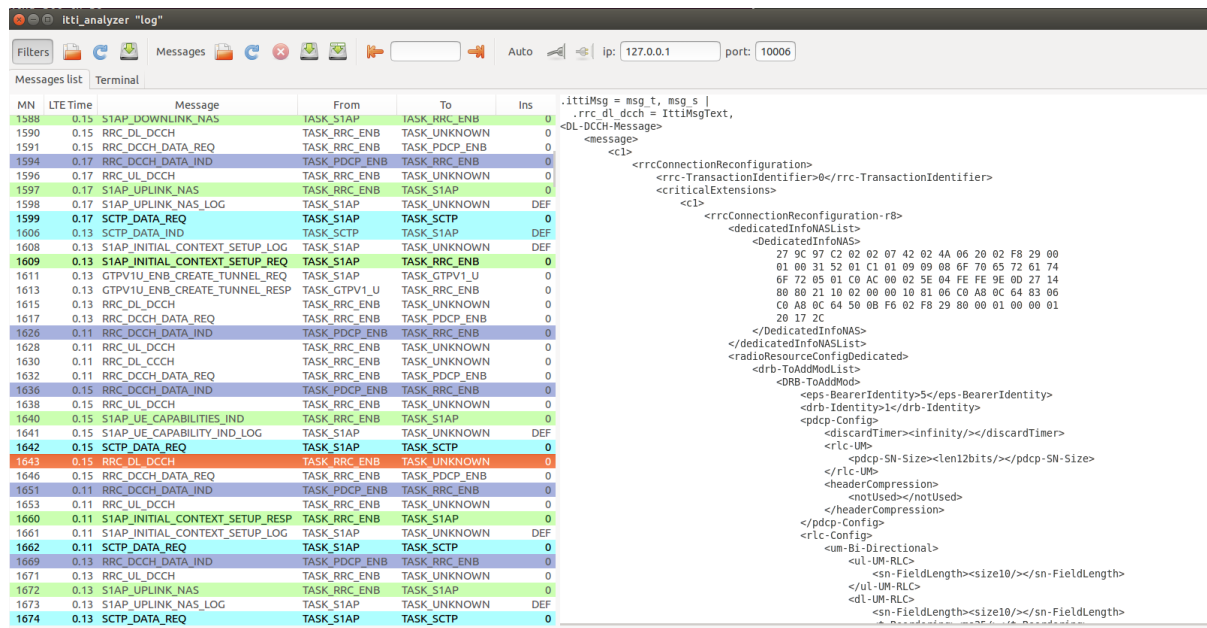


Figure 6: ITTI analyser: RRCConnectionReconfiguration message capture

OAI provides a logger exploited for code testing, debugging, checking protocols functionality and storing messages information. The OAI logger can be configured in the eNB configuration file. The logs have the following format:

[COMP][LOG LEVEL][FUNC][ID][FRAME NUM][CONTENT] as described below:

- COMP: represents the log component and can be S1AP, GTP, RRC, PDCP, RLC, MAC, PHY.
- LOG LEVEL: represents the level and verbosity of the logs, and can have the following values: Emerge, Alert, Critic, Error, War, Notice, Info, Debug, Trace.
- FUNC: represents the name of the function inside which the log is called (This is an optional block).
- ID: represents the eNB Identifier.
- FRAME NUM: represents the frame counter.
- CONTENT: shows the content of the log message.

Figure 7 shows an instance of messages exchange that is stored in OAI logger. This information is used for further study by the experimenters and code analysis/debugging for the developers.

```
cigale@cigale: ~/openair4G/common/utis/itti_analyzer
cigale@cigale: ~/openair4G/common/utis/itti_analyzer x cigale@cigale: ~/openair4G/targets/RT/USER x

</BandInfoEUTRA>
<BandInfoEUTRA>
  <InterFreqBandList>
    <InterFreqBandInfo>
      <InterFreqNeedForGaps><true/></InterFreqNeedForGaps>
    </InterFreqBandInfo>
    <InterFreqBandInfo>
      <InterFreqNeedForGaps><true/></InterFreqNeedForGaps>
    </InterFreqBandInfo>
    <InterFreqBandInfo>
      <InterFreqNeedForGaps><true/></InterFreqNeedForGaps>
    </InterFreqBandInfo>
  </InterFreqBandList>
</BandInfoEUTRA>
</bandListEUTRA>
</measParameters>
<featureGroupIndicators>
  01011110000011011101100010000000
</featureGroupIndicators>
<interRAT-Parameters>
</interRAT-Parameters>
</UE-EUTRA-Capability>
[RRC][I]RRCCONNECTIONRECONFIGURATION Encoded 813 bits (102 bytes)
[RRC][I][eNB 0] Frame 2178, Logical Channel DL-DCCH, Generate RRCCONNECTIONRECONFIGURATION (bytes 102, UE id c7b4)
[SCPT][I][sctp_send_data] Successfully sent 45 bytes on stream 1 for assoc_id 40
[RLC][I][FRAME 02178][eNB][MOD 00][RNTI c7b4][SRB AM 01] RLC_AM_DATA_REQ size 107 Bytes, NB SDU 1 current_sdu_index=6 next_sdu_index=7 conf 0 mul
2
[RRC][N][eNB 0] Frame 132: received a DCCH 1 message on SRB 0 with Size 2 from UE c7b4
[RRC][I][FRAME 02180][eNB][MOD 00][RNTI c7b4] Received on DCCH 1 RRC_DCCH_DATA_IND
[RLC][I][FRAME 02180][eNB][MOD 00][RNTI c7b4] [DRB 1] rrc_rlc_add_rlc DRB
[RRC][I][eNB 0] Frame 2180 : Logical Channel UL-DCCH, Received RRCCONNECTIONRECONFIGURATIONCOMPLETE from UE rnti c7b4, reconfiguring DRB 1/LCID 3
[RRC][I][eNB 0] Frame 2180 : Logical Channel UL-DCCH, Received RRCCONNECTIONRECONFIGURATIONCOMPLETE from UE 0, reconfiguring DRB 1/LCID 3
[MAC][I][rrc_mac_config_req] [CONFIG][eNB 0/0] Configuring MAC/PHY for UE 0 (c7b4)
```

Figure 7: OAI logger: RRCConnectionReconfigurationComplete reception

OAI offers online statistics for the status of the network (e.g., successful transmissions, errors per HARQ per round, average throughput etc.). Figure 8 depicts the UL-SCH/DL-SCH errors per HARQ process (8 in LTE FDD) per round (4 is maximum). The provided statistics can be used in experimental measurements for performance evaluation of the system.

RAN Timing

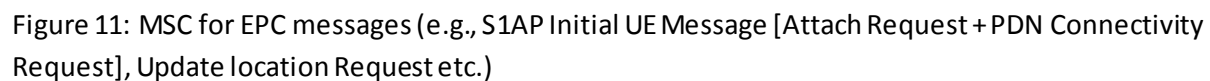
- RX processing subframe n (eNB_thread_rx)
- `trx_read(1 ms)`
 - `slot_fep` for slots $2n$ and $2n + 1$ [`phy_procedures_eNB_common_rx()`]
 - launch PRACH RX thread (subframe n)
 - `prach_procedures`
 - `phy_procedures_eNB_rx()` (subframe n)

- TX processing subframe $n + 4$ (eNB_thread_tx)
- `phy_procedures_eNB_tx`
 - `ofdm_mod`
 - `trx_write(1 ms)`



Figure 9: Timing analyser [6]

OAI uses MSC for representing network entities/interfaces (i.e., eNB, UE, MME, HSS, S6A etc.) and their interactions in a diagrammatic form. Arrows direction defines the sender/receiver and explanatory comments are included. The examples in Figure 10 and Figure 11 show an instance of E_UTRAN and EPC messages exchange, respectively.



4 References

- [1] Nikaein, Navid; Knopp, Raymond; Kaltenberger, Florian; Gauthier, Lionel; Bonnet, Christian; Nussbaum, Dominique; Ghaddab, Riadh, "OpenAirInterface: an open LTE network in a PC", url: <http://www.eurecom.fr/publication/4371>
- [2] Nikaein, Navid; Knopp, Raymond; Gauthier, Lionel; Schiller, Eryk; Braun, Torsten; Pichon, Dominique; Bonnet, Christian; Kaltenberger, Florian; Nussbaum, Dominique, "Closer to cloud-RAN: RAN as a service", url: <http://www.eurecom.fr/publication/4632>
- [3] OpenAirInterface Wiki, URL: <https://gitlab.eurecom.fr/oai/openairinterface5g/wikis/home>
- [4] OpenAirInterface repository: <https://gitlab.eurecom.fr/oai/openairinterface5g>
- [5] OpenAirInterface Tutorials: <https://gitlab.eurecom.fr/oai/openairinterface5g/wikis/OpenAirUsage>
- [6] OAI L1L2 Procedures: https://gitlab.eurecom.fr/oai/openairinterface5g/blob/develop/targets/DOCS/oai_L1_L2_procedures.pdf

Annex A eNB configuration

The eNB configuration parameters are presented here. All the parameters are divided in the following 6 main sections:

A.1 Main section

Parameter	Type	
real_time	String	Real time compilation
eNB_ID	Integer	eNB identifier
cell_type	String	Type of the cell
eNB_name	String	Name of eNB
tracking_area_code	String	Tracking area code (TAC)
mobile_country_code	String	Mobile country code (MCC)
mobile_network_code	String	Mobile network code (MNC)

Table 1 eNB configuration main section

A.2 Physical parameters section

Parameter	Type	
frame_type	String	Type of the frame (e.g., FDD,TDD)
tdd_config	Integer	
timer_status_prohibit	Integer	
tdd_config_s	Integer	
prefix_type	String	Cyclic prefix
eutra_band	Integer	EUTRA band (e.g., 7, 13)
downlink_frequency	Double	Downlink frequency

uplink_frequency_offset	Integer	Uplink frequency offset
Nid_cell	Integer	Physical layer cell identity
N_RB_DL	Integer	Number of resource blocks (RBs) in Downlink
Nid_cell_mbsfn	Integer	
nb_antennas_tx	Integer	Number of Tx antennae
nb_antennas_rx	Integer	Number of Rx antennae
tx_gain	Integer	Tx gain
rx_gain	Integer	Rx gain
prach_root	Integer	
prach_config_index	Integer	Parameter that defines exactly when UE should send RACH in frequency/time grids (Details TS36.211 Table 5.7.1-2)
prach_high_speed	String	
prach_zero_correlation	Integer	The zero correlation zone is used to guarantee orthogonality of generated sequences. The value depends on particular condition in the cell
prach_freq_offset	Integer	With this information, cell informs UE and other neighbor cells know about which PRB is available for RACH access
pucch_delta_shift	Integer	
pucch_nRB_CQI	Integer	Number of resource blocks for channel quality indicator (CQI) periodic reporting
pucch_nCS_AN	Integer	Cyclic shift used for PUCCH1
pucch_n1_AN	Integer	PUCCH to be used for HARQ (Rel 10)
pdsch_referenceSignalPower	Integer	This defines the energy per resource element for the reference signal using a range from -60 to 50 dBm
pdsch_p_b	Integer	It is used to calculate the power difference between

		PDSCH and Reference Signal. Value is from 0 to 3
pusch_n_SB	Integer	Number of subbands (range 1 to 4)
pusch_enable64QAM	String	64QAM (enable/disable)
pusch_hoppingMode	String	Hopping mode can be inter-subframe, intra or inter-subframe
pusch_hoppingOffset	Integer	Offset values range from 1 to 98
pusch_groupHoppingEnabled	String	Group hopping (enable/disable)
pusch_groupAssignment	Integer	Parameter that gives sequence shift pattern for group hopping (0 to 29)
pusch_sequenceHoppingEnabled	String	Sequence hopping (enable/disable)
pusch_nDMRS1	Integer	
phich_duration	String	
phich_resource	String	
srs_enable	String	
pusch_p0_Nominal	Integer	Impacts the calculation of PUSCH transmit power and applicable to non-persistent scheduling only (-126 to 24 dBm)
pusch_alpha	String	Impacts the calculation of PUSCH transmit power and also scales the contribution of path loss. Possible values are 0, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9 and 1
pucch_p0_Nominal	Integer	Impacts the calculation of PUSCH transmit power and applicable to non-persistent scheduling only (-126 to 24 dBm)
msg3_delta_Preamble	Integer	Impacts the transmit power of PUSCH when responding to random access response grant (-1 to 6dB)
pucch_deltaF_Format1	String	Uplink power control parameter
pucch_deltaF_Format1b	String	Idem above

pucch_deltaF_Format2	String	Idem above
pucch_deltaF_Format2a	String	Idem above
pucch_deltaF_Format2b	String	Idem above
rach_numberOfRA_Preambles	Integer	Total number of random access preambles available for contention based random access. Since there are maximum 64 preambles sequences available, others could be reserved by eNB for Non-Contention based random access. Range of this parameter is 4 to 64
rach_preamblesGroupAConfig	String	
rach_powerRampingStep	Integer	Power ramping step size with possible values of 0, 2, 4 or 6 dB
rach_preambleInitialReceivedTargetPower	Integer	Preamble initial received target power with values from -120 dBm to -90 dBm with step size of 2 dBm
rach_preambleTransMax	Integer	Maximum number of preambles transmissions. Possible values are 3, 4, 5, 6, 7, 8, 10, 20, 50, 100, 200
rach_raResponseWindowSize	Integer	Duration of RA response window. RA response window size is in unit of subframes (2, 3, 4, 5, 6, 7, 8, or 10 subframes)
rach_macContentionResolutionTimer	Integer	Mac contention resolution timer in unit of subframes
rach_maxHARQ_Msg3Tx	Integer	Maximum number of HARQ retransmissions for message 3 of RACH process (contention-based Random access) with possible values from 1 to 8 in step of 1
pcch_default_PagingCycle	Integer	The default DRX cycle in idle mode in unit of radio frames
pcch_nB	String	Parameter used in finding the actual paging frames and paging occasions in RRC idle mode
bcch_modificationPeriodCoeff	Integer	The value (2,4,6,8) of this parameter is multiplied with default DRX cycle (e.g., 320ms, 640ms) to generate the BCCH modification period

ue_TimersAndConstants_t300	Integer	Starts at the RRC Connection REQ transmit
ue_TimersAndConstants_t301	Integer	Starts at the RRC Connection Re-establishment REQUEST
ue_TimersAndConstants_t310	Integer	Starts when UE detects PHY layer related problems (when it receives N310 consecutive out-of-sync INDs from lower layers)
ue_TimersAndConstants_t311	Integer	Starts while initiating Connection Re-establishment procedure

Table 2 eNB configuration subsection SRB1 parameters

A.3 SRB1 parameters section

Parameter	Type	
timer_poll_retransmit	Integer	Timer for poll retransmission
timer_reordering	Integer	RLC AM reordering timer
timer_status_prohibit	Integer	
poll_pdu	Integer	
poll_byte	Integer	
max_retx_threshold	Integer	

Table 3 eNB configuration subsection SRB1 parameters

A.4 MME parameters section

Parameter	Type	
ipv4	String	IPv4 address
ipv6	String	IPv6 address

active	String	Activation (yes/no)
preference	String	IPv4 or IPv6

Table 4 eNB configuration subsection MME parameters

A.5 Network interfaces section

Parameter	Type	
ENB_INTERFACE_NAME_FOR_S1_MME	String	Interface name for S1-MME (e.g., eth1)
ENB_IPV4_ADDRESS_FOR_S1_MME	String	eNB IPv4 subnet for S1-MME
ENB_INTERFACE_NAME_FOR_S1_U	String	Interface name for S1-U (e.g., eth1)
ENB_IPV4_ADDRESS_FOR_S1_U	String	eNB IPv4 subnet for S1-U
ENB_PORT_FOR_S1_U	String	eNB port for S1-U

Table 5 eNB configuration subsection Network interfaces

A.6 Log config

Parameter	Type	
global_log_level	String	Global logger level (e.g., emerg, alert, crit, error, warn, notice, info, debug, trace)
global_log_verbosity	String	Global logger verbosity level (e.g., none, low, medium, high, full)
hw_log_level	String	HW logger level (e.g., emerg, alert, crit, error, warn, notice, info, debug, trace)
hw_log_verbosity	String	HW logger verbosity level (e.g., none, low, medium, high, full)
phy_log_level	String	PHY logger level (e.g., emerg, alert, crit, error, warn, notice, info, debug, trace)
phy_log_verbosity	String	PHY logger verbosity level (e.g., none, low, medium, high, full)

mac_log_level	String	MAC logger level (e.g., emerg, alert, crit, error, warn, notice, info, debug, trace)
mac_log_verbosity	String	MAC logger verbosity level (e.g., none, low, medium, high, full)
rlc_log_level	String	RLC logger level (e.g., emerg, alert, crit, error, warn, notice, info, debug, trace)
rlc_log_verbosity	String	RLC logger verbosity level (e.g., none, low, medium, high, full)
pdcp_log_level	String	PDCP logger level (e.g., emerg, alert, crit, error, warn, notice, info, debug, trace)
pdcp_log_verbosity	String	PDCP logger verbosity level (e.g., none, low, medium, high, full)
rrc_log_level	String	RRC logger level (e.g., emerg, alert, crit, error, warn, notice, info, debug, trace)
rrc_log_verbosity	String	RRC logger verbosity level (e.g., none, low, medium, high, full)
gtpu_log_level	String	GTPU logger level (e.g., emerg, alert, crit, error, warn, notice, info, debug, trace)
gtpu_log_verbosity	String	GTPU logger verbosity level (e.g., none, low, medium, high, full)
udp_log_level	String	UDP logger level (e.g., emerg, alert, crit, error, warn, notice, info, debug, trace)
udp_log_verbosity	String	UDP logger verbosity level (e.g., none, low, medium, high, full)
osa_log_level	String	OSA logger level (e.g., emerg, alert, crit, error, warn, notice, info, debug, trace)
osa_log_verbosity	String	OSA logger verbosity level (e.g., none, low, medium, high, full)

Table 6 eNB configuration subsection Log config