教育部「5G行動寬頻人才培育跨校教學聯盟計畫」

5G行動網路協定與核網技術聯盟中心

「5G行動寬頻協同網路」課程模組

單元7
開源碼行動通訊暨5G模擬平台

副教授：吳俊興

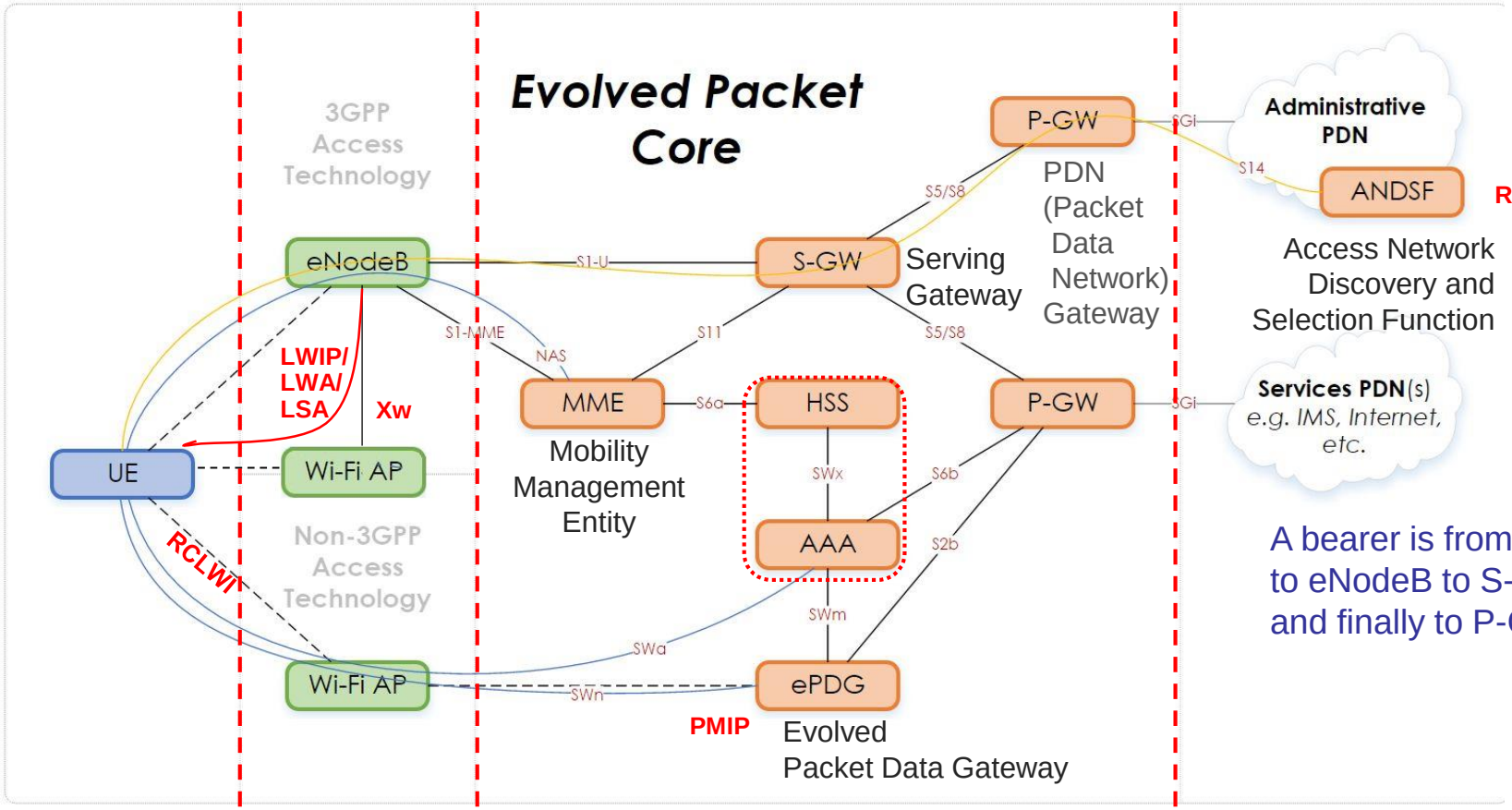
助教：吳振宇、林原進

國立高雄大學 資訊工程學系

Outline

- 3GPP 4G/5G System Architecture
- 4G/5G Open Source Platform
  - OpenLTE
  - OpenAirInterface
  - RECO
  - Free5GC
- srsLTE
  - srsUE, srsENB, srsEPC
- nukLWA/nukxDC

4G EPS (Evolved Packet System) / SAE

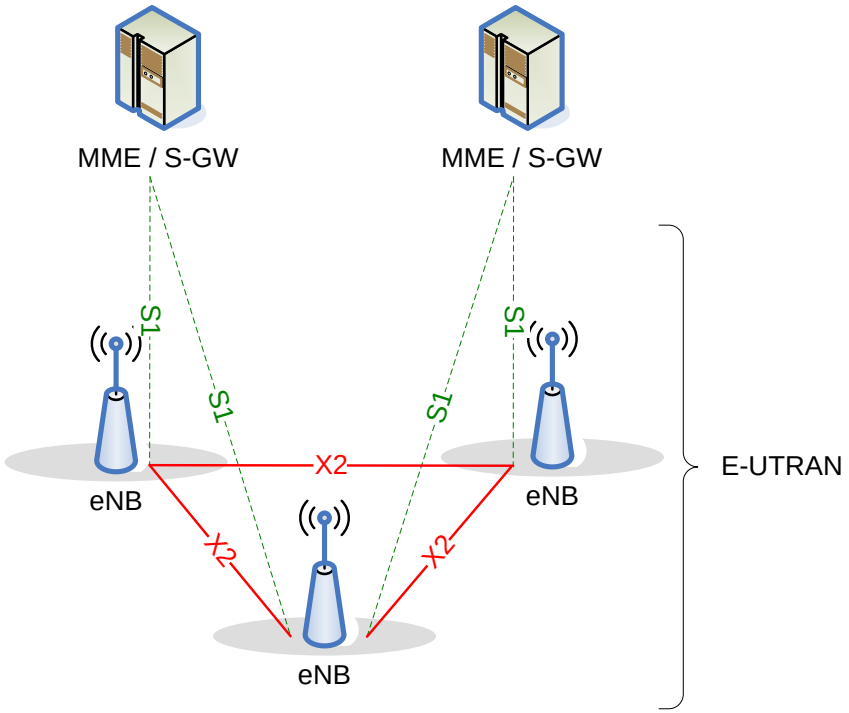


A bearer is from UE to eNodeB to S-GW and finally to P-GW

- EPC (Evolved Packet Core): main component of EPS, includes
  - MME: key control-node for LTE – UE paging; chooses S-GW for UE during attach and handover
    - Authenticating the user (by interacting with HSS - Home Subscriber Server)
  - S-GW: manages and stores UE contexts; routes and forwards user data packets
  - P-GW: provides connectivity from the UE to external packet data networks
  - ePDG: secures data transmission with UE connected to EPC over *untrusted non-3GPP* access
  - ANDSF: provides information to UE to discover available access networks (either 3GPP or not)

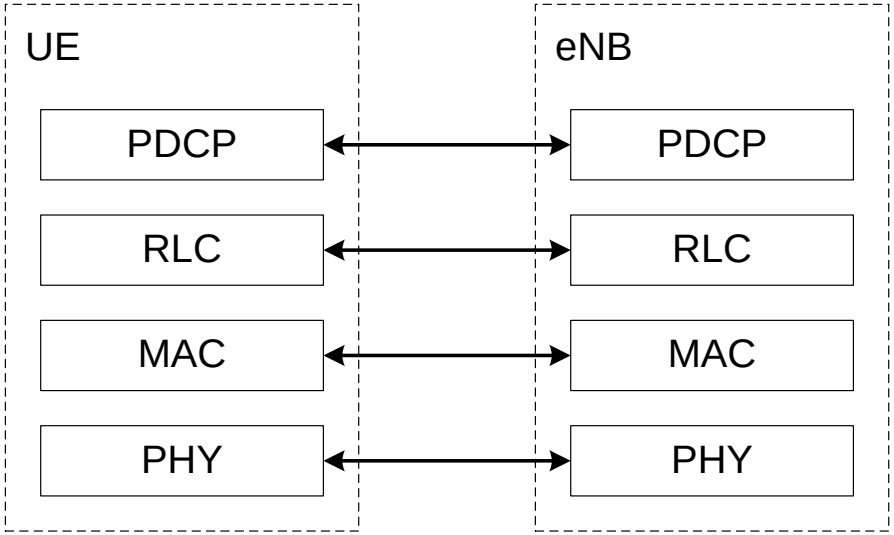
4G - E-UTRAN Overall Architecture

- The E-UTRAN consists of eNBs, providing the E-UTRA user plane (PDCP/RLC/MAC/PHY) and control plane (RRC) protocol terminations towards the UE
- The eNBs are interconnected with each other by means of the X2 interface
- The eNBs are also connected by means of the S1 interface to the EPC (Evolved Packet Core), more specifically to the MME (Mobility Management Entity) by means of the S1-MME interface and to the Serving Gateway (S-GW) by means of the S1-U interface
- The S1 interface supports a many-to-many relation between MMEs / Serving Gateways and eNBs

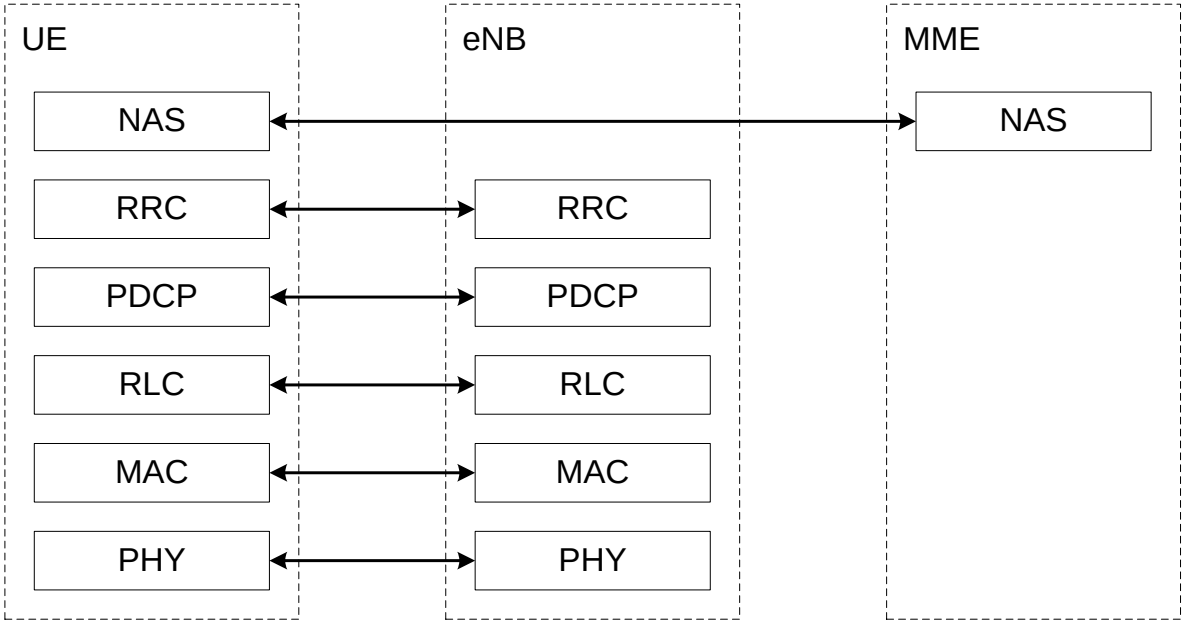


Radio Protocol Architecture of E-UTRAN

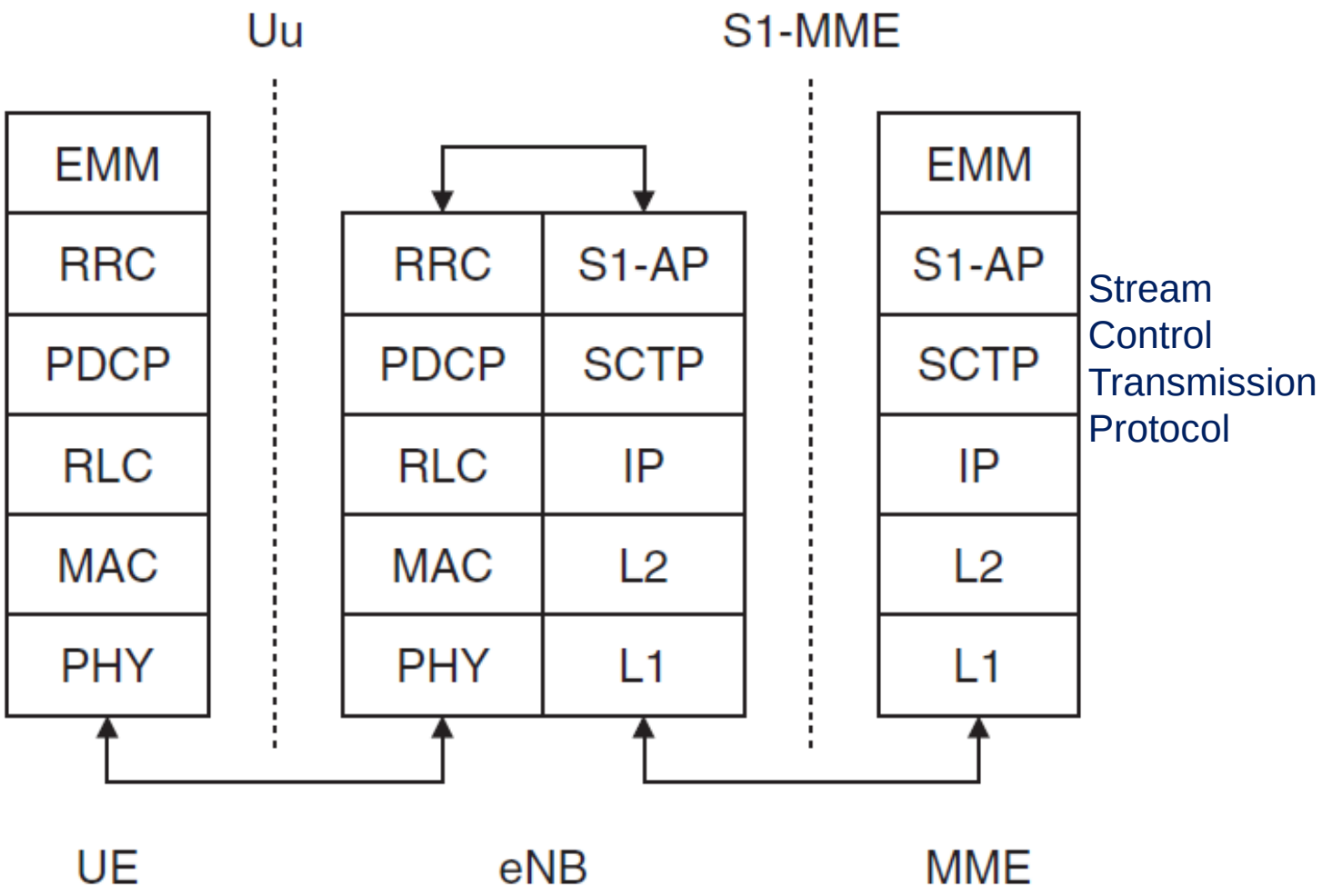
User Plane



Control Plane

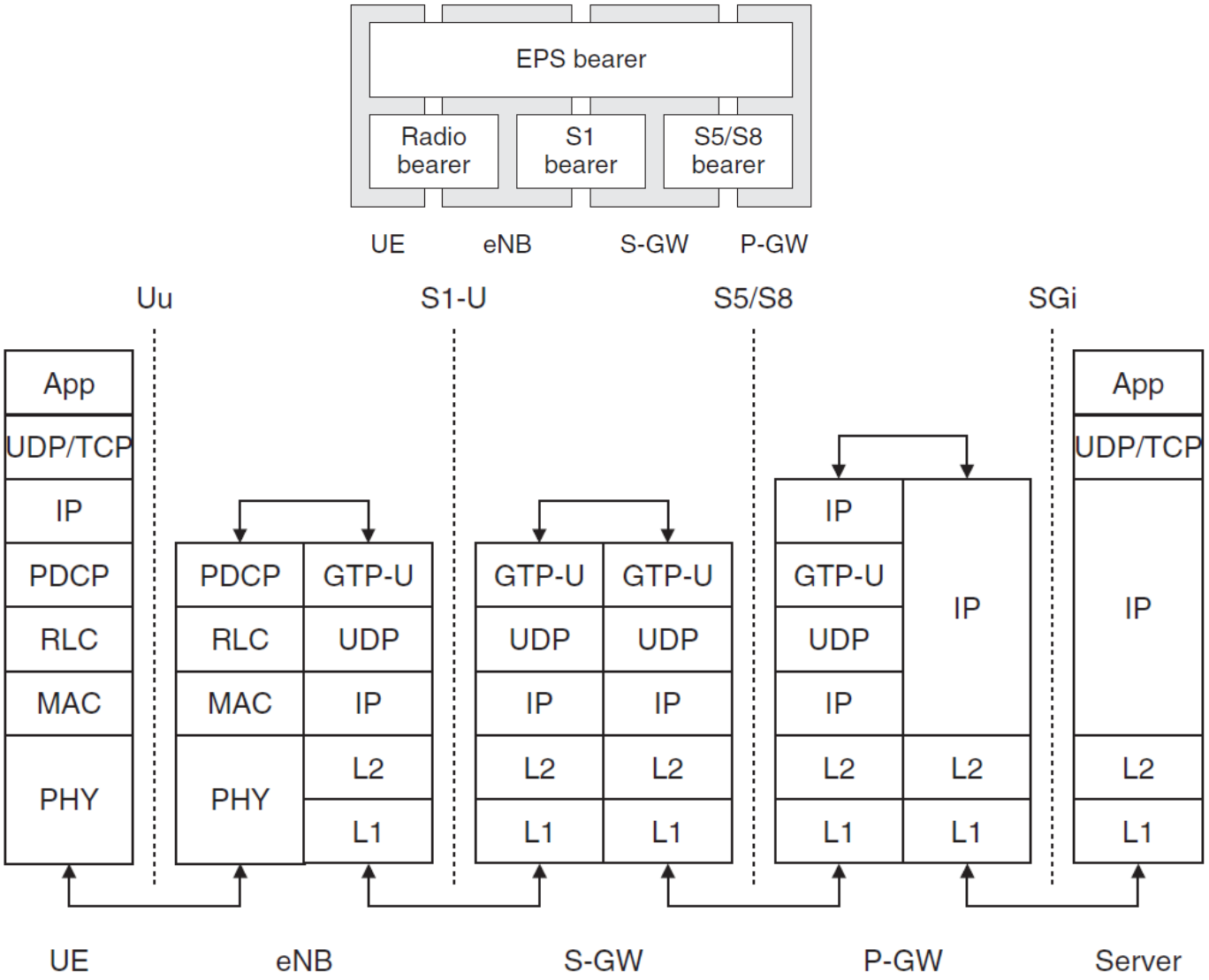


Protocol Stack to Exchange Control Signaling



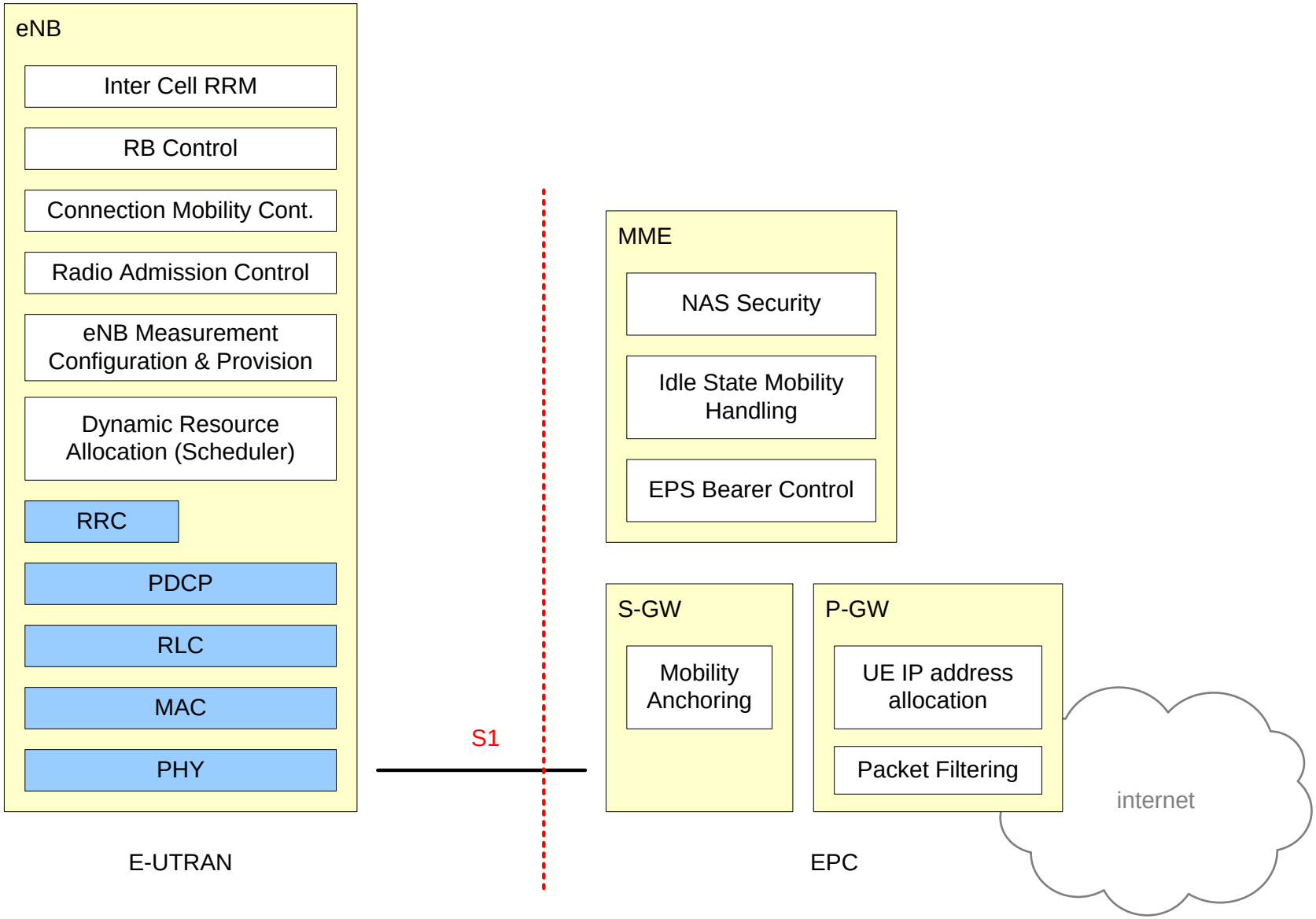
TS 23.401

Bearer Implementation (Using GTP)

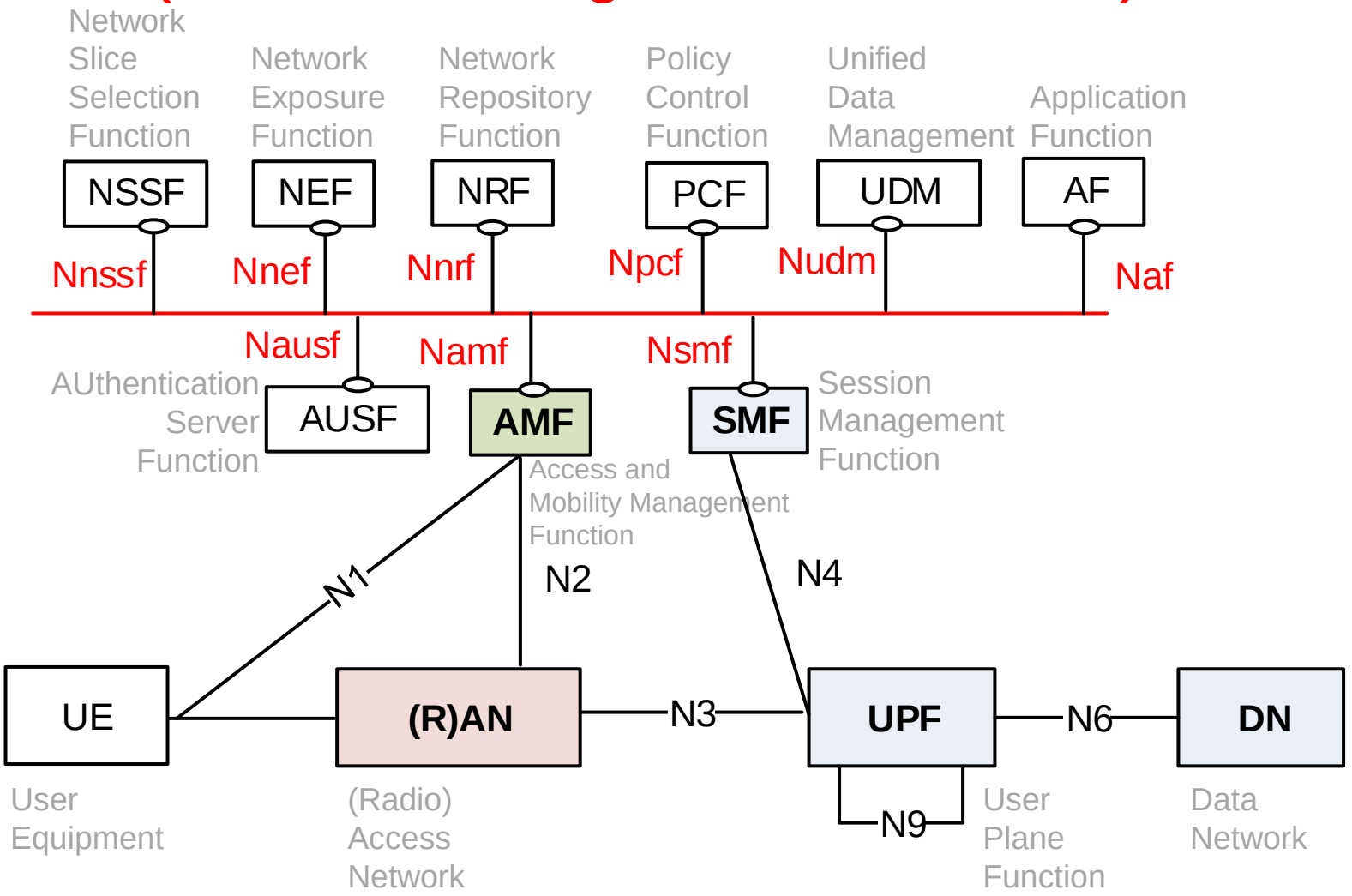


TS 23.401

Functional Split between E-UTRAN and EPC



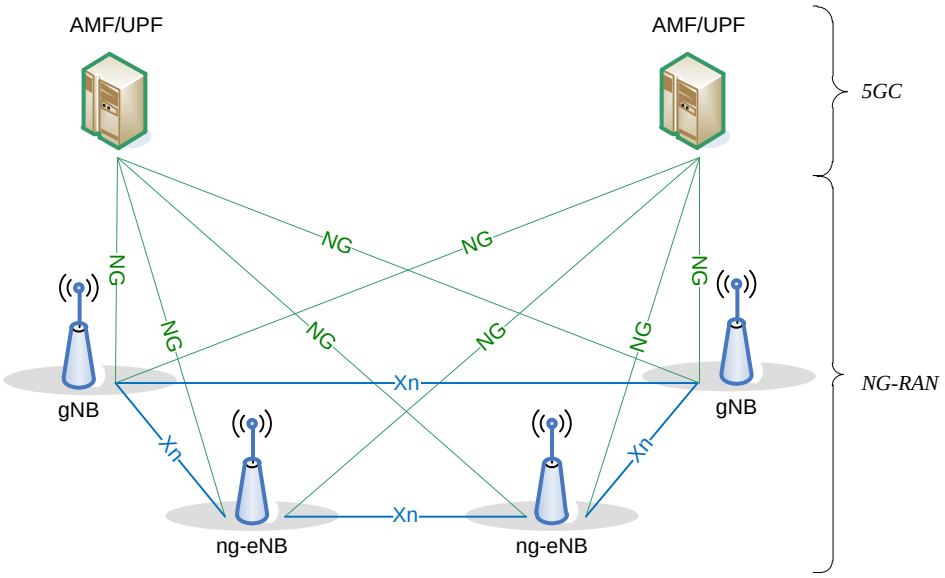
5G System Architecture (Non-Roaming Service-Based)



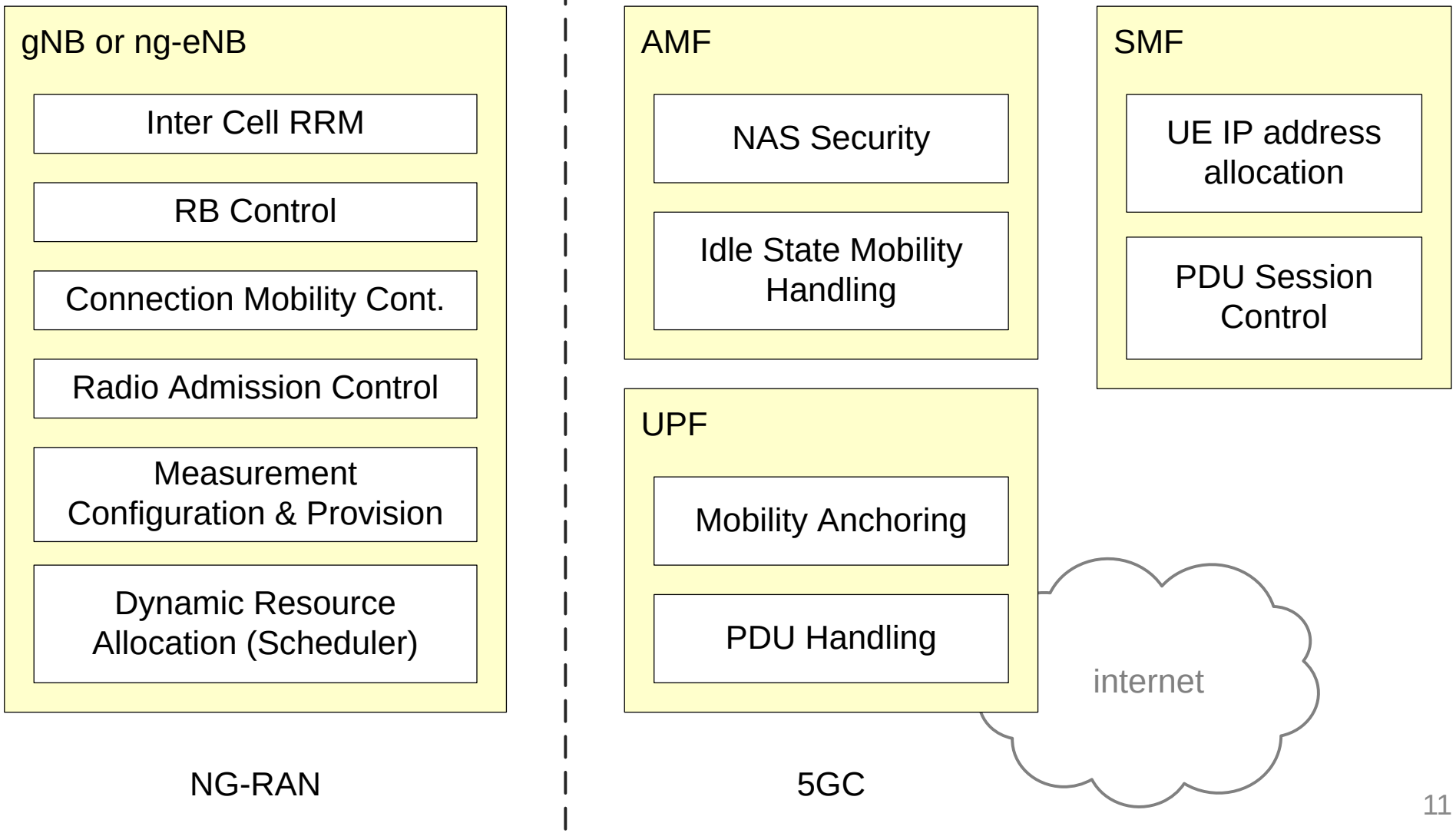
The non-roaming reference architecture with service-based interfaces used within the Control Plane (TS23.501)

NG-RAN Overall Architecture

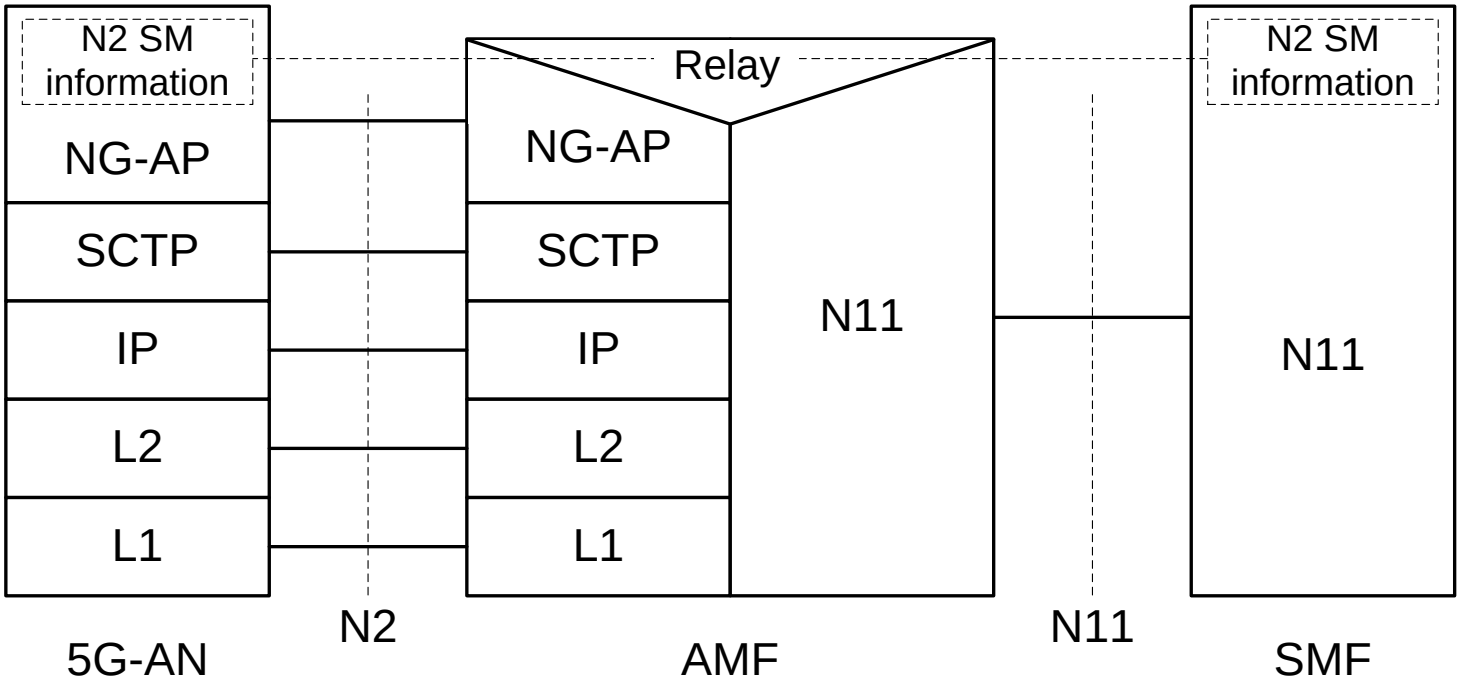
- An NG-RAN node is either
  - A gNB, providing NR user plane and control plane protocol terminations towards the UE
  - An ng-eNB, providing E-UTRA user plane and control plane protocol terminations towards the UE
- The gNBs and ng-eNBs are interconnected with each other by means of the Xn interface
- The gNBs and ng-eNBs are also connected by means of the NG interfaces to the 5GC, more specifically to the AMF (Access and Mobility Management Function) by means of the NG-C interface and to the UPF (User Plane Function) by means of the NG-U interface (see 3GPP TS 23.501)
- The architecture and the F1 interface for a functional split are defined in 3GPP TS 38.401



Functional Split between NG-RAN and 5GC

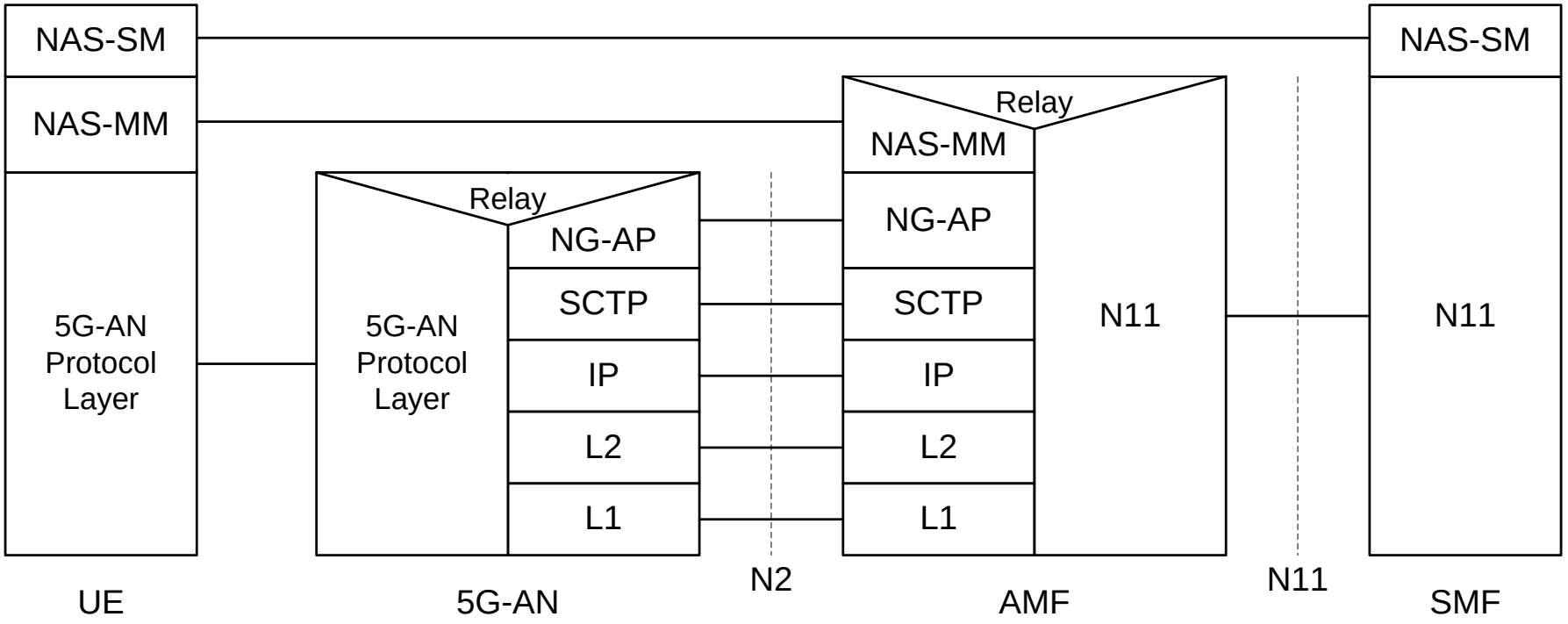


Control Plane between the AN and the SMF



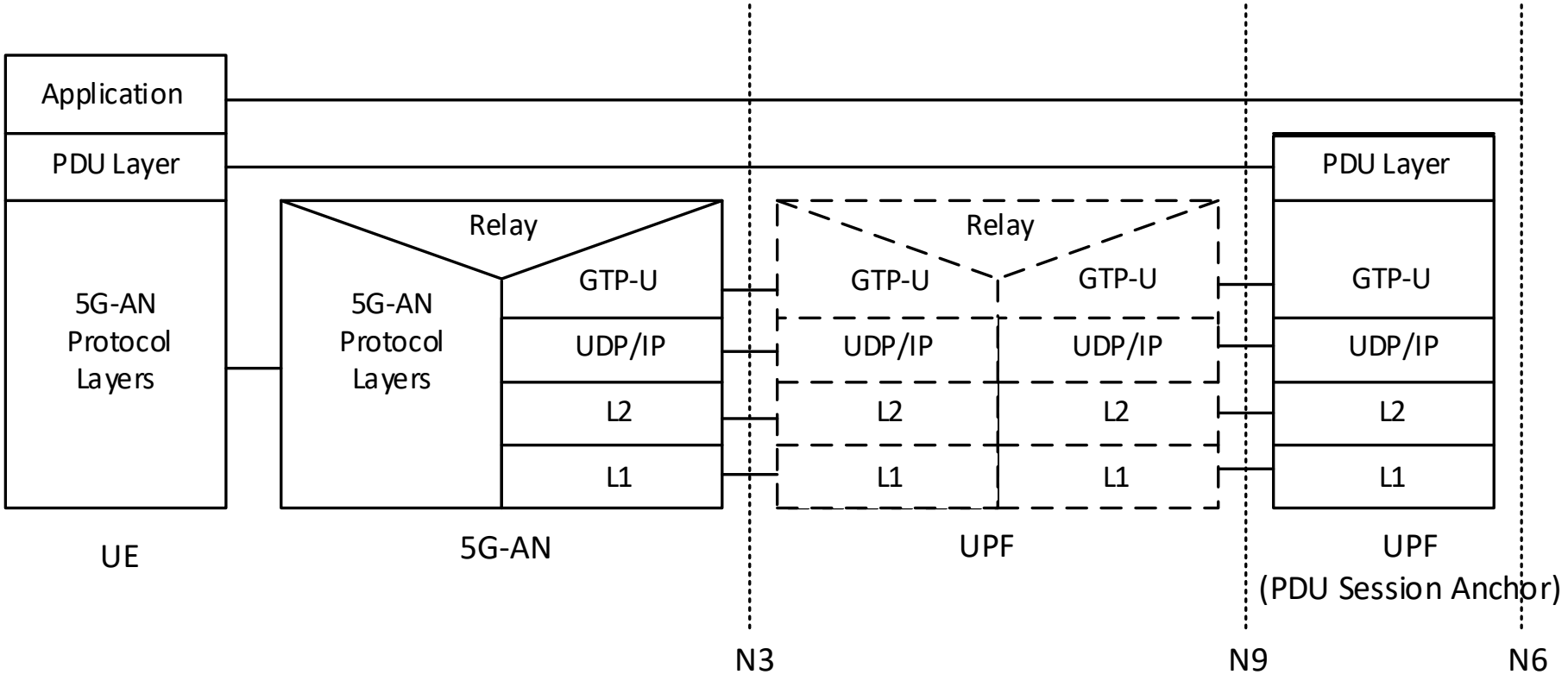
- N2 SM information: the subset of NG-AP information that the AMF transparently relays between the AN and the SMF
  - Included in the NG-AP messages and the N11 related messages
  - From the AN perspective, there is a single termination of N2 i.e. the AMF
  - For the protocol stack between the AMF and the SMF, see TS 23.501 clause 8.2.3

Control Plane Protocol Stack between the UE and the SMF

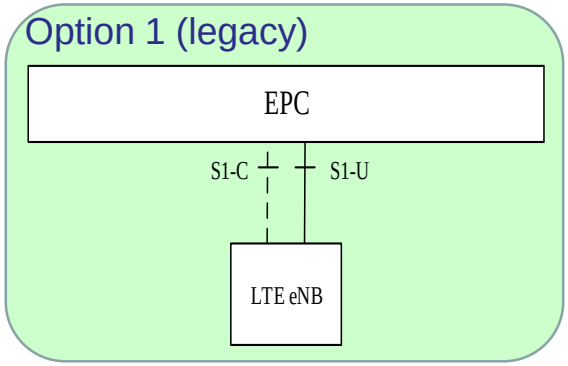


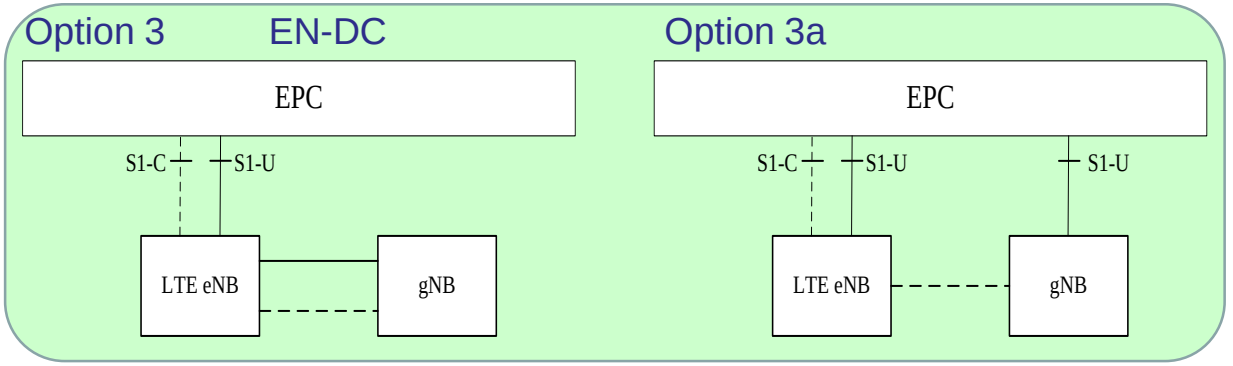
- NAS-SM: The NAS protocol for SM functionality supports user plane PDU Session Establishment, modification and release
  - It is transferred via the AMF, and transparent to the AMF. 5G NAS protocol is defined in TS 24.501
- NAS-SM supports the handling of Session Management between UE and the SMF
- The SM signalling message is handled, i.e. created and processed, in the NAS-SM layer of UE and the SMF
  - The content of the SM signalling message is not interpreted by the AMF

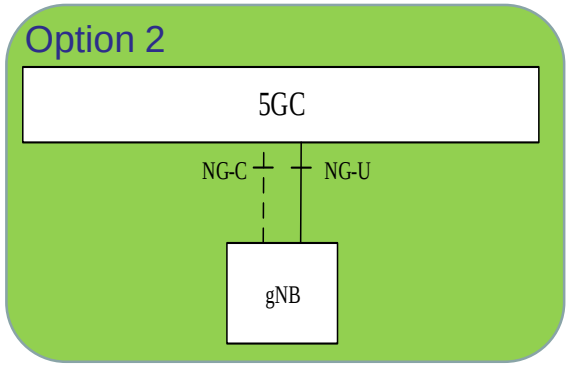
User Plane Protocol Stack for 3GPP Access

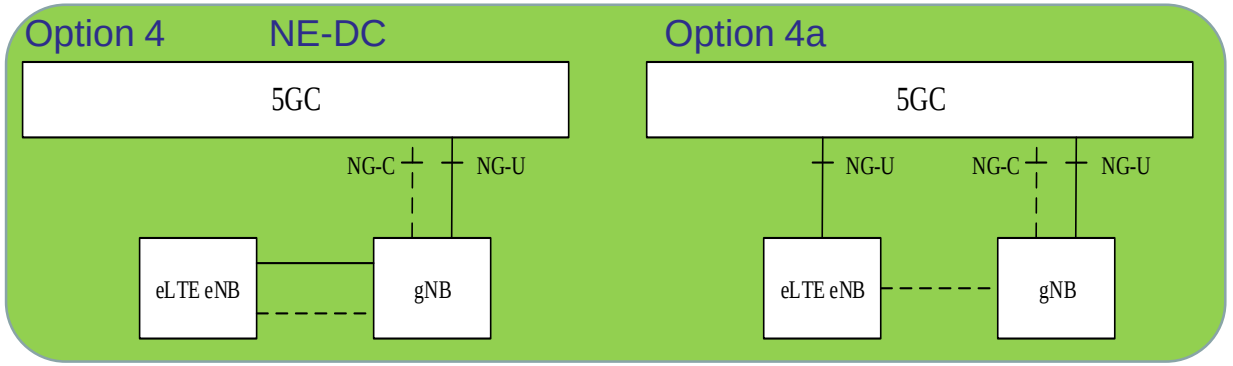


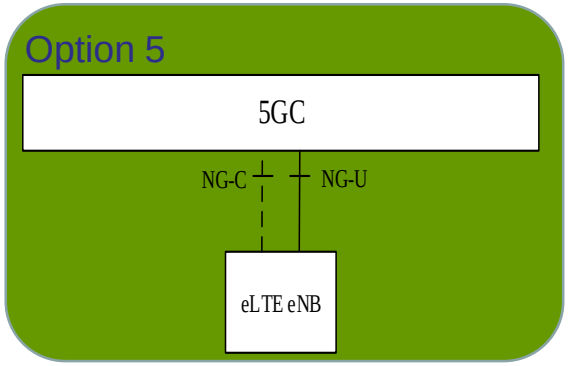
5G Architecture Options (TR38.801)

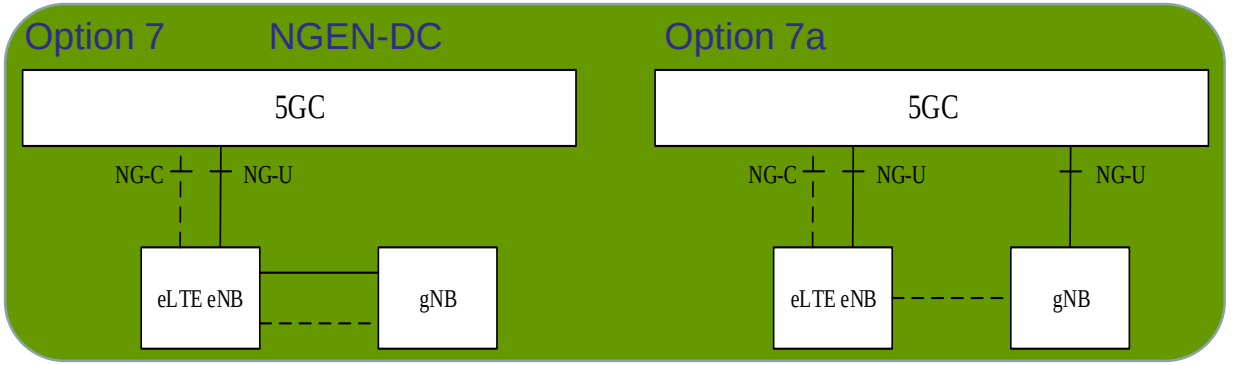






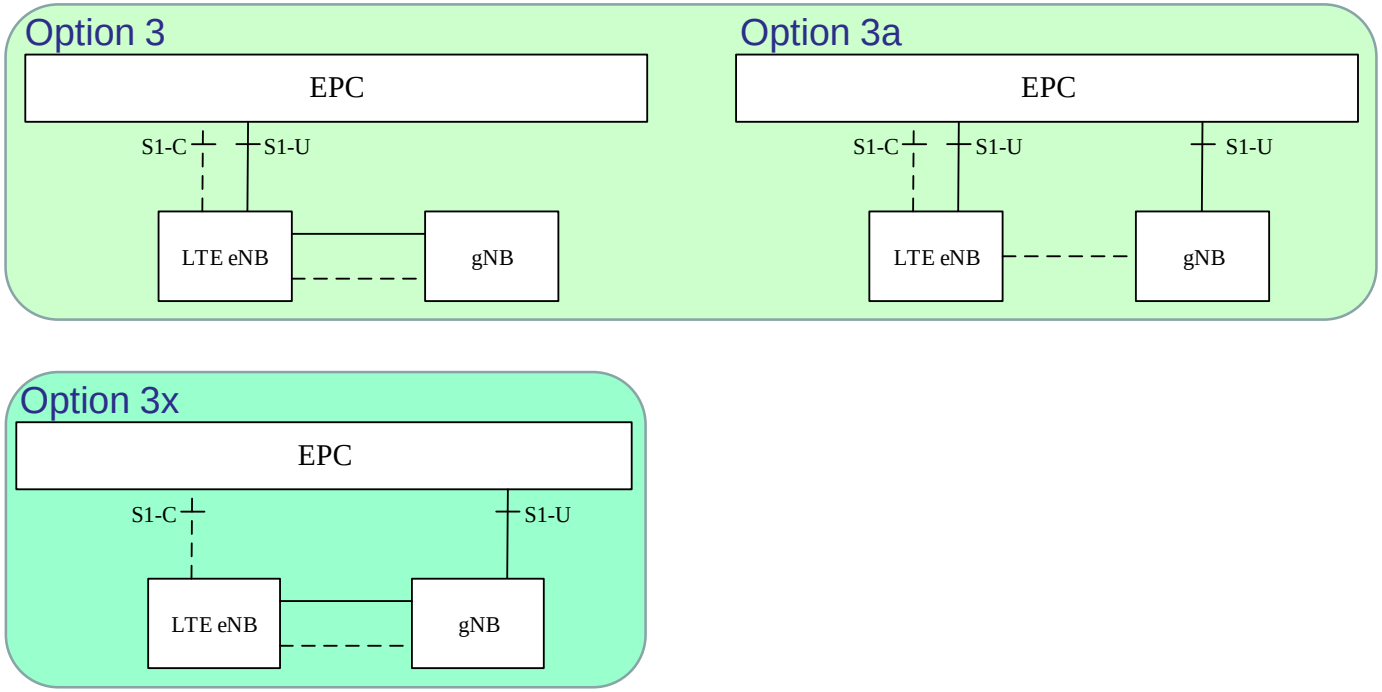






5G NSA (Release 15)

- Three models of NSA with EN-DC
  - Option 3 – Traffic split at eNB
  - Option 3a – Traffic split at S-GW
  - Option 3x – Traffic split at gNB

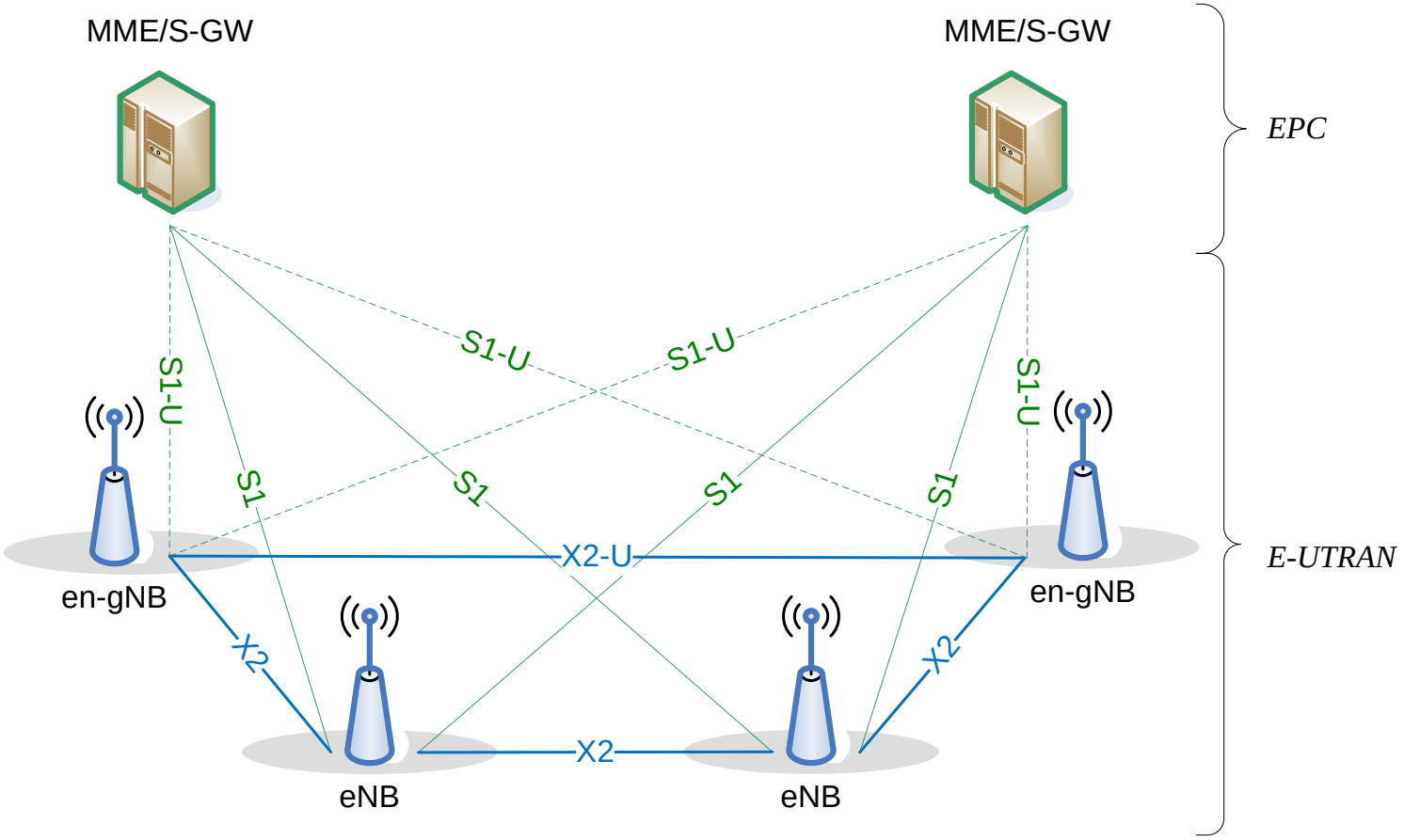


Common MR-DC Principles

- Multi-Radio Dual Connectivity (MR-DC) is a generalization of the Intra-E-UTRA Dual Connectivity (DC)
- A multiple Rx/Tx capable UE may be configured to utilise resources provided by two different nodes connected via non-ideal backhaul
- One providing NR access and the other one providing either E-UTRA or NR access
  - One node acts as the Main Node(MN) connect to the core network
  - The other node acts as the Second Node(SN)

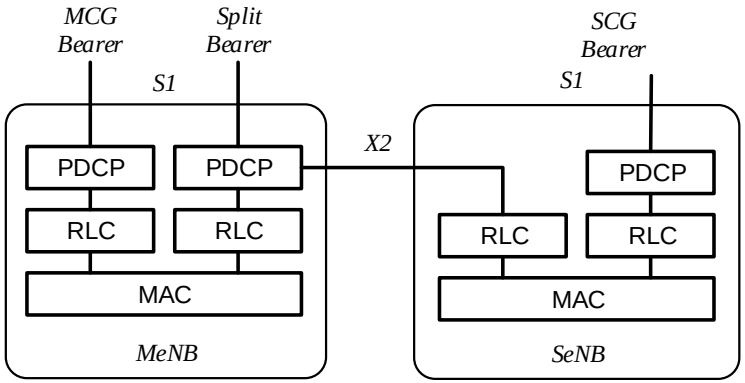
EN-DC Overall Architecture

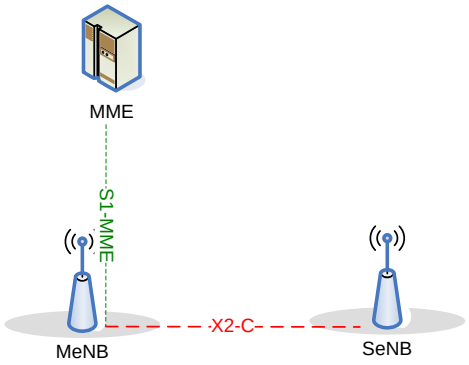
- E-UTRAN supports MR-DC via E-UTRA-NR Dual Connectivity (EN-DC)
- UE is connected to one eNB that acts as a MN and one en-gNB that acts as a SN
- The eNB is connected to the EPC via the S1 interface and to the en-gNB via the X2 interface
- The en-gNB might also be connected to the EPC via the S1-U interface and other en-gNBs via the X2-U interface

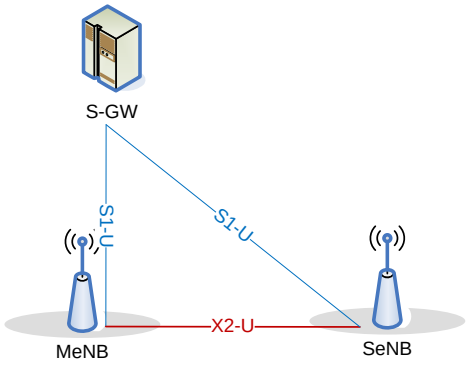


DC and EN-DC

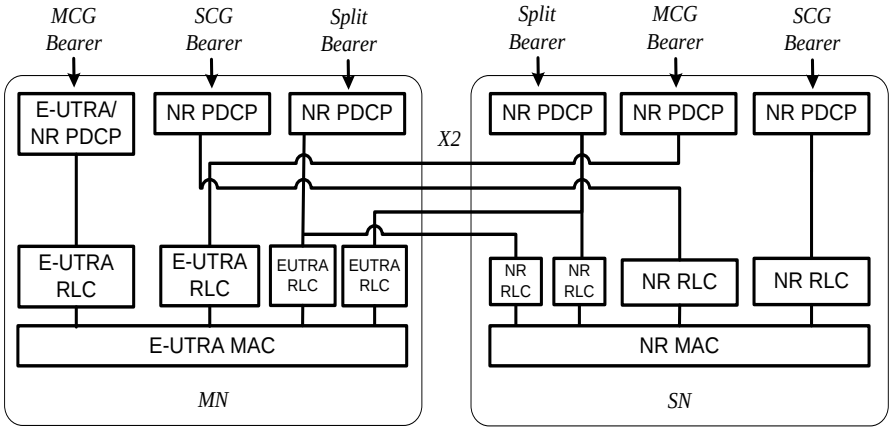
DC

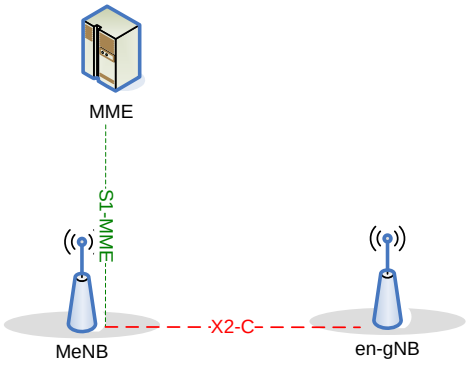


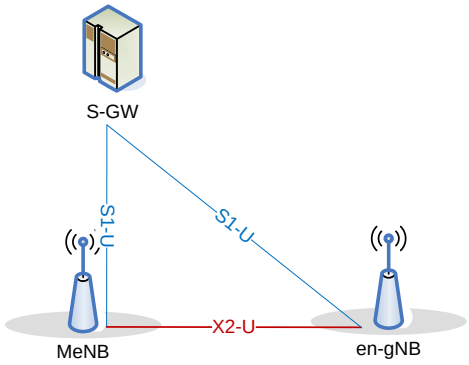




EN-DC





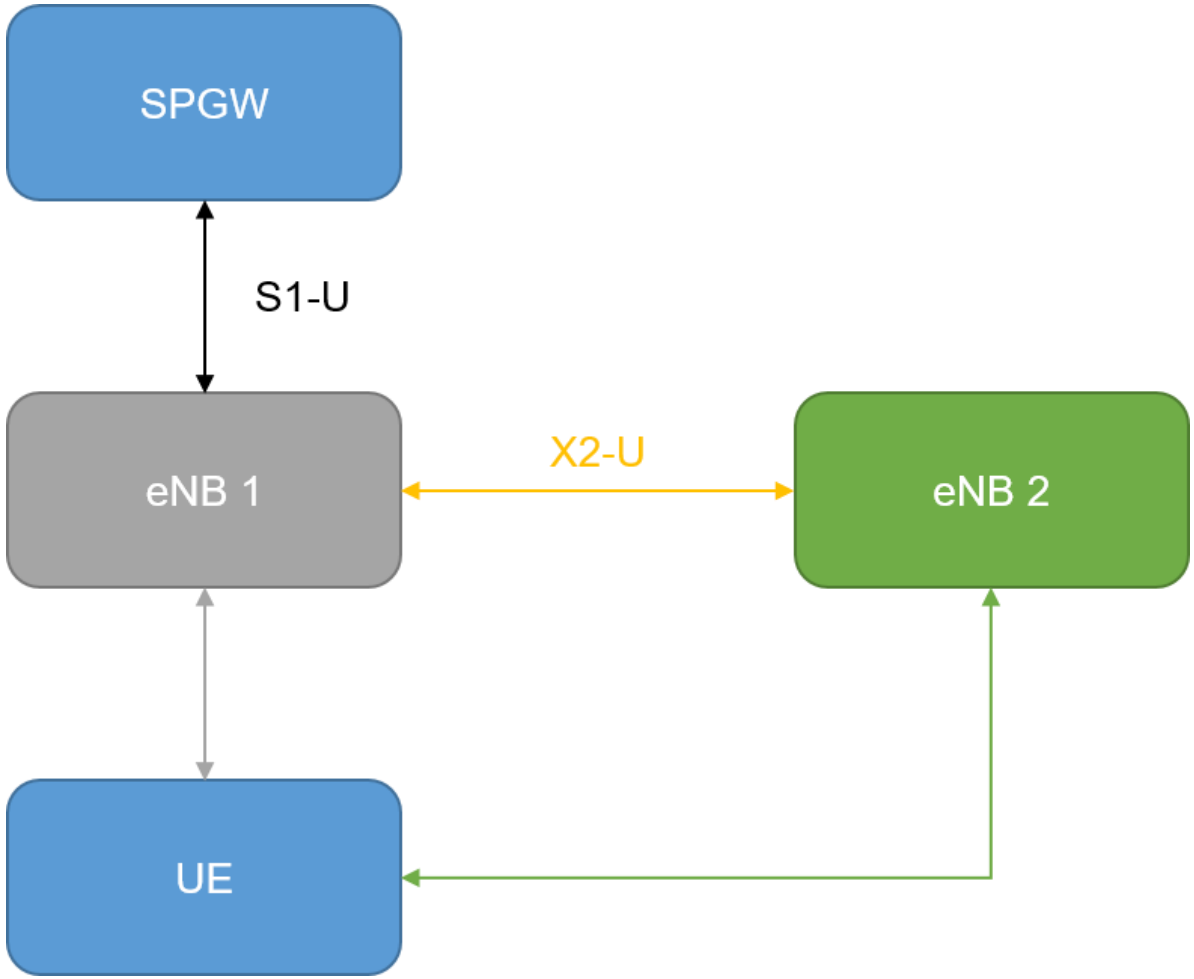


Radio Protocol Architecture

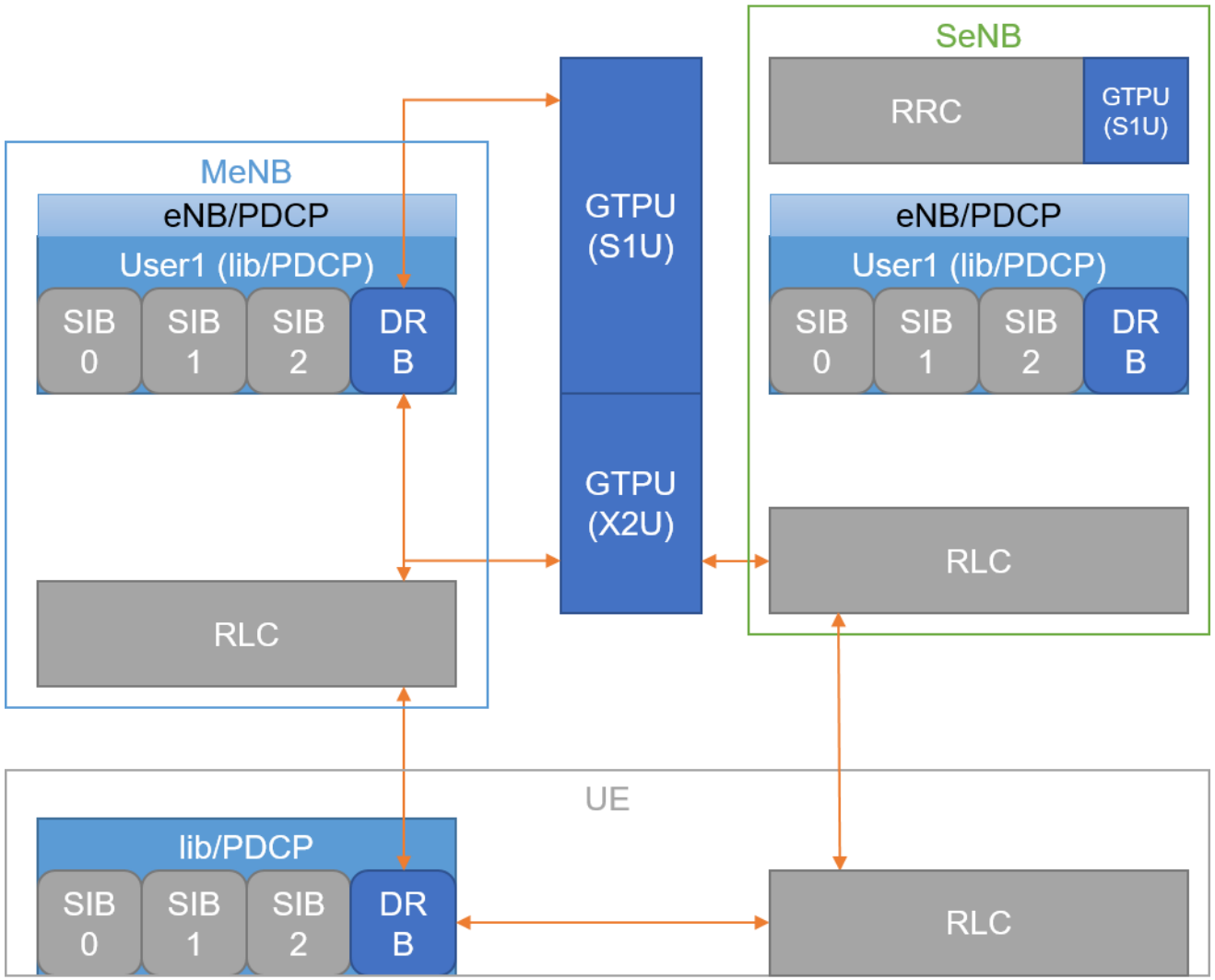
C-Plane

U-Plane

Dual Connectivity Architecture (User Plane)



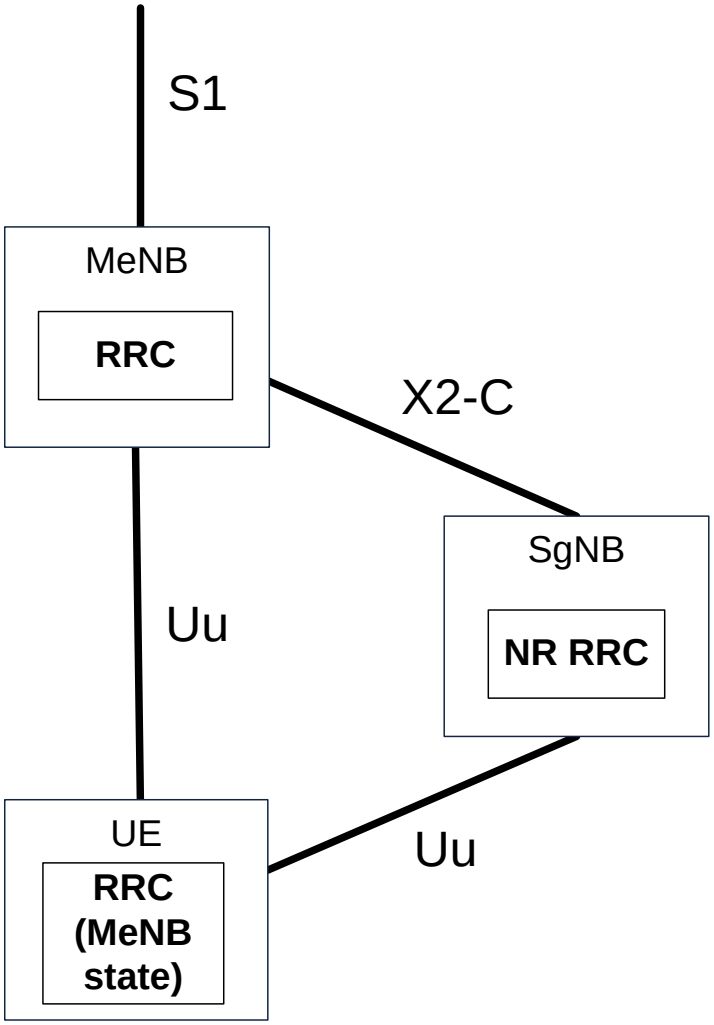
Dual Connectivity (User Plane)



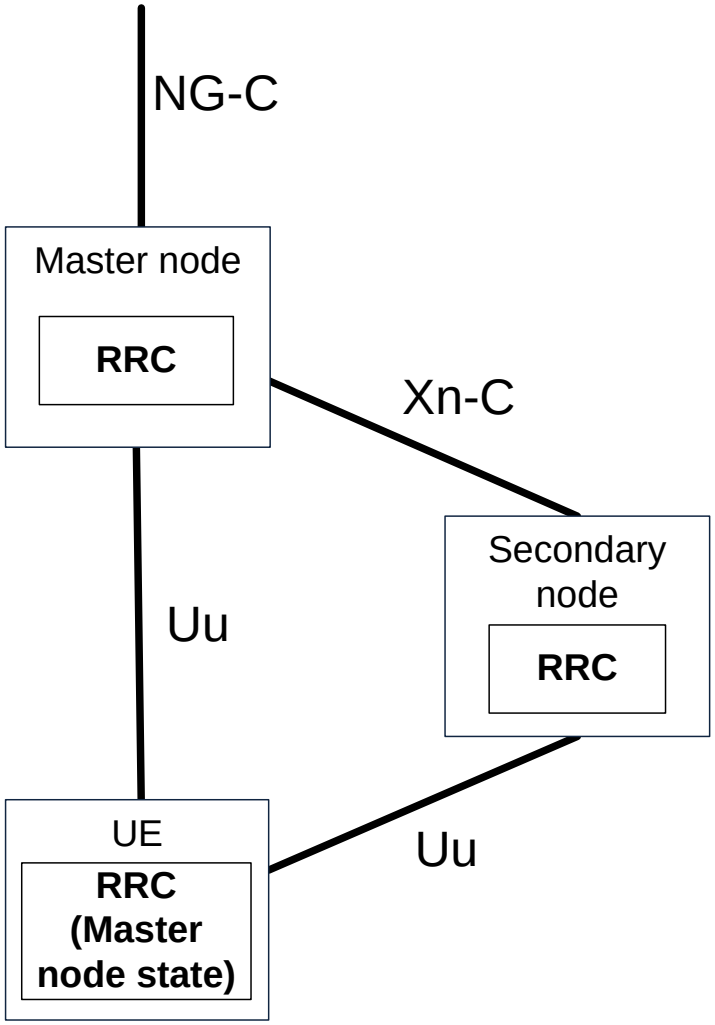
MR-DC with the 5GC

- NG-RAN supports these types of Dual Connectivity
- E-UTRA-NR Dual Connectivity(NGEN-DC)
  - UE is connected to one ng-eNB that acts as a MN and one gNB that acts as a SN
- NR-E-UTRA Dual Connectivity(NE-DC)
  - UE is connected to one gNB that acts as a MN and one ng-eNB that acts as a SN
- NR-NR Dual Connectivity(NR-DC)
  - UE is connected to one gNB that acts as a MN and another gNB that acts as a SN

Control Plane Architecture for EN-DC and MR-DC with 5GC

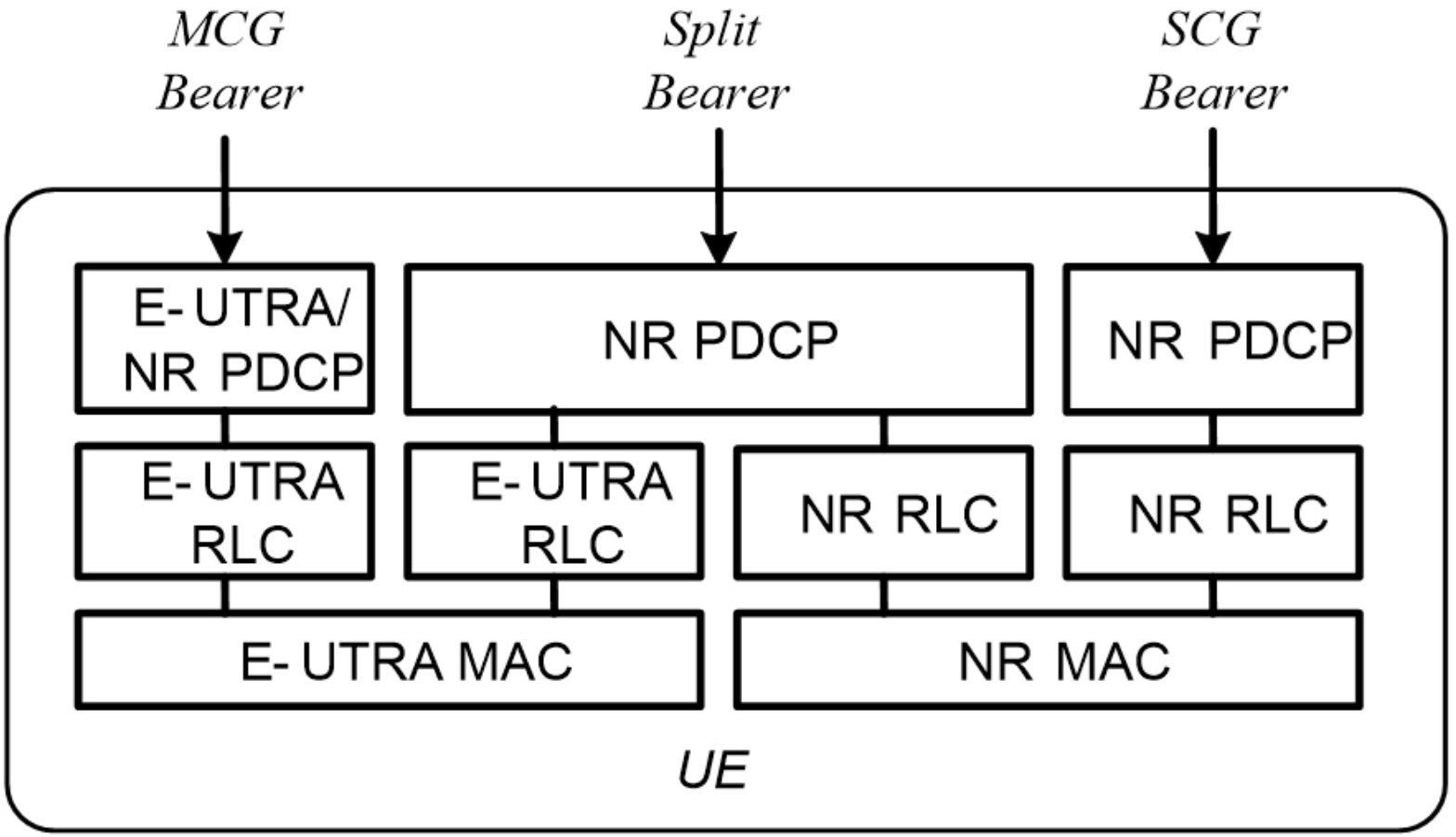


EN-DC

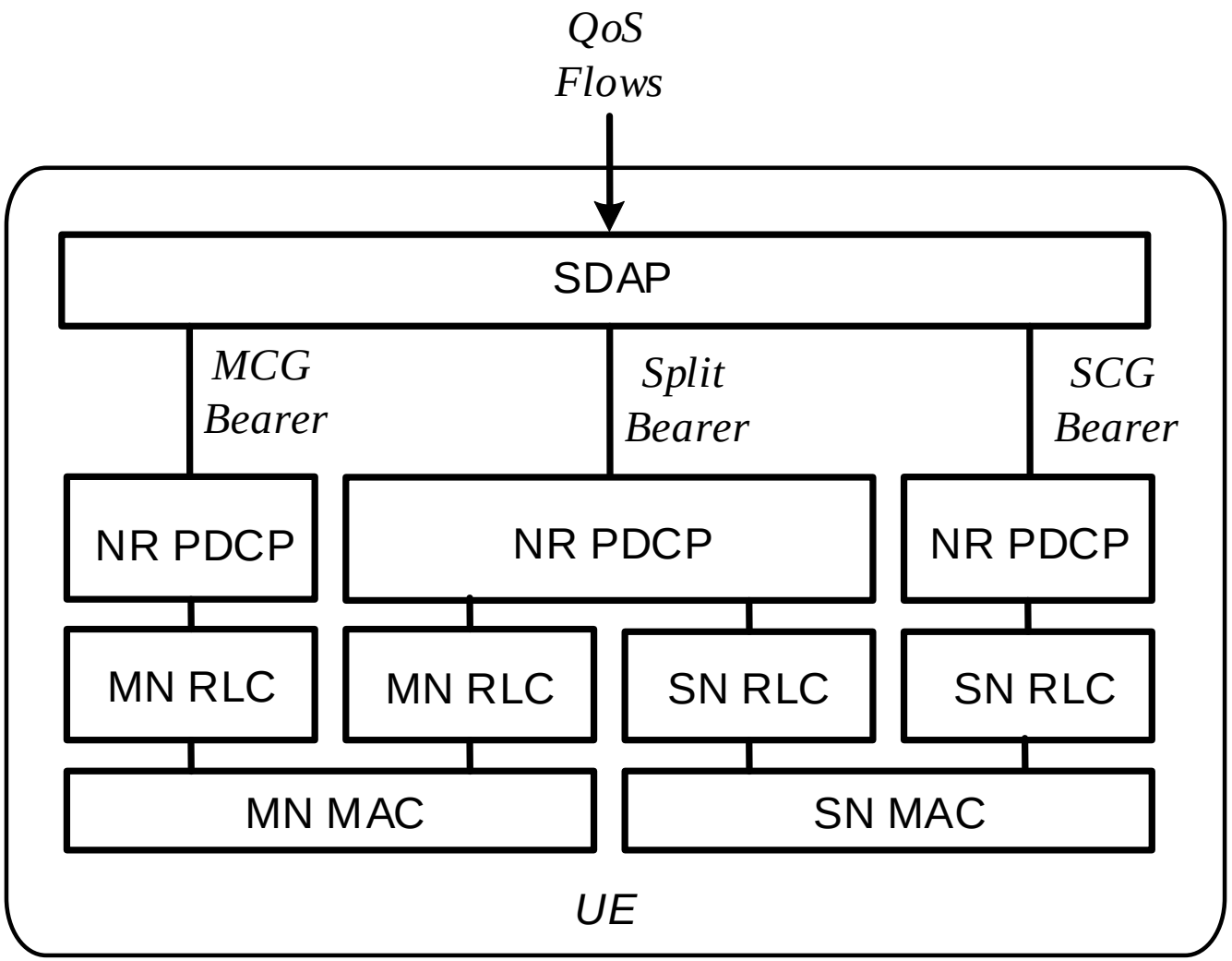


MR-DC

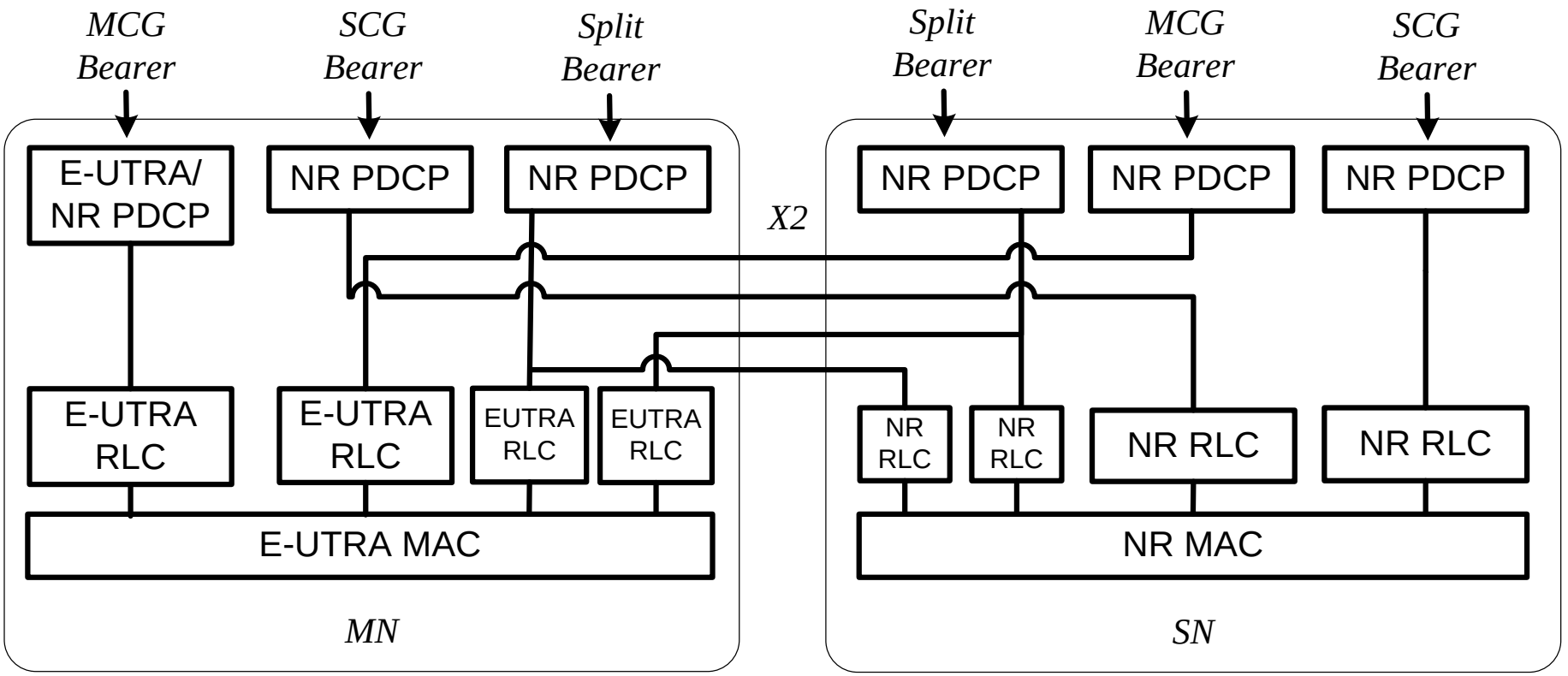
Radio Protocol Architecture in MR-DC with EPC



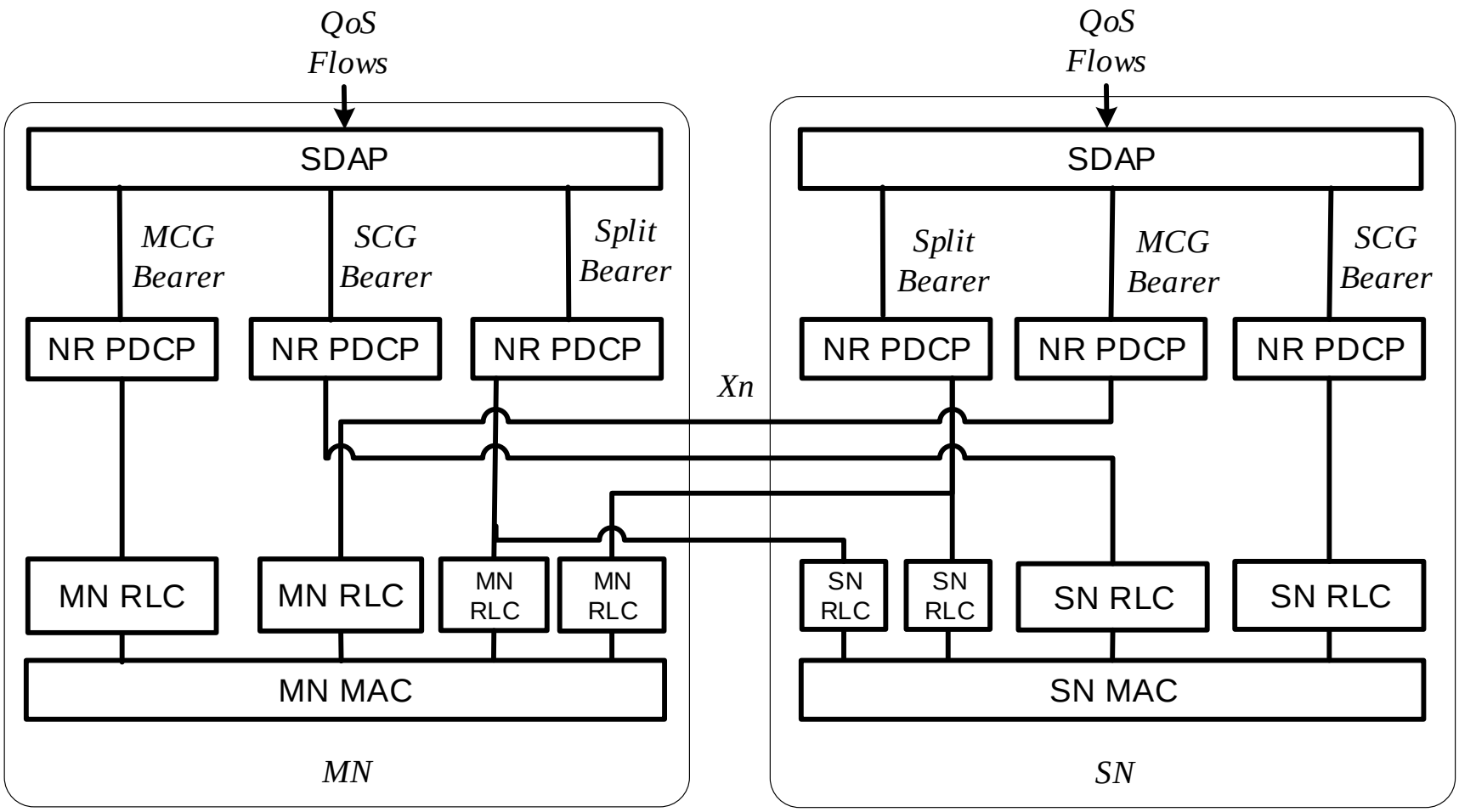
Radio Protocol Architecture in MR-DC with 5GC



Network Side Protocol in MR-DC with EPC



Network Side Protocol in MR-DC with 5GC



Outline

- 3GPP 4G/5G System Architecture
- 4G/5G Open Source Platform
  - OpenLTE
  - OpenAirInterface
  - RECO
  - Free5GC
- srsLTE
  - srsUE, srsENB, srsEPC
- nukLWA/nukxDC

OpenLTE

- OpenLTE is an open source implementation of the 3GPP LTE specifications
  - Used by OAI, srsLTE, etc
- Currently, octave code is available for test and simulation of
  - downlink transmit and receive functionality and
  - uplink PRACH transmit and receive functionality
- In addition, GNU Radio applications are available for
  - downlink transmit and receive to and from a file,
  - downlink receive using rtl-sdr, HackRF, or USRP B2X0,
  - LTE I/Q file recording using rtl-sdr, HackRF, or USRP B2X0, and
  - a simple eNodeB using USRP B2X0
- The current focus is on
  - extending the capabilities of the GNU Radio applications and
  - adding capabilities to the simple base station application (LTE_fdd_enodeb)

Reference: https://sourceforge.net/projects/openlte/

GNU Radio

- A free & open-source software development toolkit that provides signal processing blocks to implement software radios
  - It can be used with
    - readily-available low-cost external RF hardware to create software-defined radios, or
    - without hardware in a simulation-like environment
  - It is widely used in hobbyist, academic and commercial environments to support both wireless communications research and real-world radio systems
- A software radio is a radio system which performs the required signal processing in software instead of using dedicated integrated circuits in hardware
  - The benefit is that since software can be easily replaced in the radio system, the same hardware can be used to create many kinds of radios for many different transmission standards
  - Thus, one software radio can be used for a variety of applications

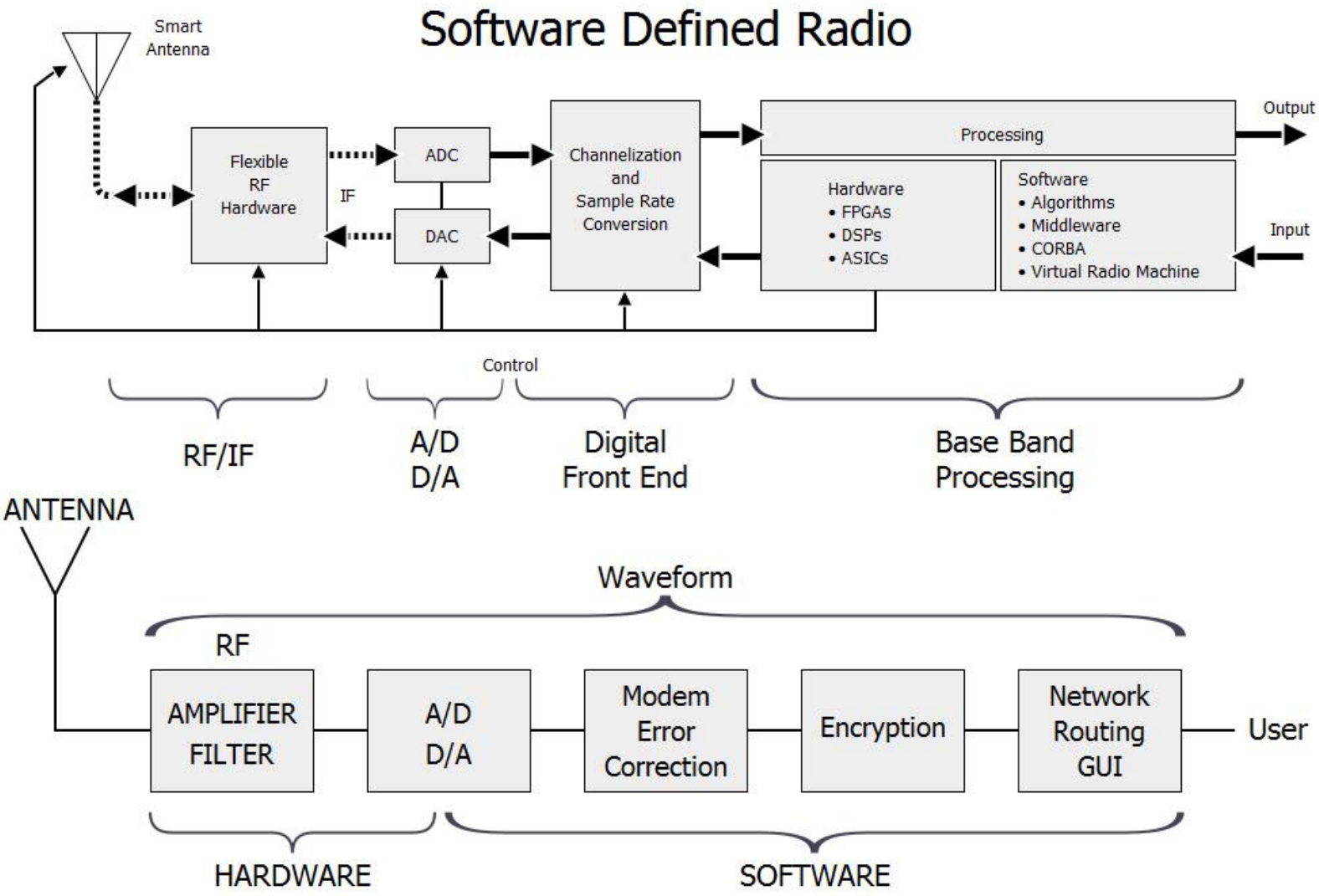
https://www.gnuradio.org/

Software-Defined Radio (SDR)

- Software-defined radio (SDR) is a radio communication system where components that have been traditionally implemented in hardware are instead implemented by means of software on a personal computer or embedded system
- A basic SDR system may consist of a personal computer equipped with a sound card, or other analog-to-digital converter, preceded by some form of RF front end
- Significant amounts of signal processing are handed over to the general-purpose processor, rather than being done in special-purpose hardware
- Such a design produces a radio which can receive and transmit widely different radio protocols based solely on the software used

https://en.wikipedia.org/wiki/Software-defined_radio

Software Defined Radio Concept



By SDR_et_WF.JPG: Topituukderivative work: McSush (talk) - SDR_et_WF.JPG, Copyrighted free use

Operating Principles

- The ideal receiver scheme would be to attach an analog-to-digital converter to an antenna
  - A digital signal processor would read the converter, and then its software would transform the stream of data from the converter to any other form the application requires
- An ideal transmitter would be similar
  - A digital signal processor would generate a stream of numbers. These would be sent to a digital-to-analog converter connected to a radio antenna.

- The ideal scheme is not completely realizable due to the current limits of the technology
  - The difficulty of conversion between the digital and the analog domains at a high enough rate and a high enough accuracy at the same time, and
  - without relying upon physical processes like interference and electromagnetic resonance for assistance

Prerequisites

- Hardware
  - USB3 interface
  - Modern multicore CPU (Intel Core i5, Core i7 or equivalent with SSE4.1 SSE4.2 and AVX support)
  - UHD driver installed (for Ettus SDRs)
  - GNURADIO

Setup

- OpenLTE is not only requiring a huge amount of processing power, but it also requires a very low latency due its need to transmit/receive a radio frame every 1ms
  - If there is any delay in the processing, the system will not going to be able respond in time and will lose samples
- It is recommended to switch of any CPU and/or system features (mostly in your BIOS) which can cause any delays or can slow down the so called context switching time
  - Intel SpeedStep, deep and deeper sleep states etc. should be turned off. Especially with high bandwidth setups (10, 15 and 20MHz)
  - it is recommended to swtich off the GUI on linux
    - There is also a low latency edition of the linux kernel, but at this point there is no absolute proof that it actually helps with OpenLTE
- With an Ettus radio (B200, B210) you will need the latest UHD driver besides GNURadio

About OpenAirInterface™ Source Code

- The OpenAirInterface™ (OAI) source code is split into two projects
- OAI Radio Access Network (OAI-RAN)
  - This project implements 4G LTE and 5G Radio Access Network
  - Both NodeB and User Equipment (UE) are implemented
  - Code Repository: The code resides in this GITLAB (https://gitlab.eurecom.fr/oai/openairinterface5g/)
  - License: This code is distributed under the OAI 5G Public License
- OAI Core Network (OAI-CN)
  - This project implements 4G LTE Evolved Packet Core (EPC) and 5G Core Network
  - Code Repository: The code resides in this GITHUB (https://github.com/openairinterface)
  - License: This code is licensed under the Apache V2.0 License

https://www.openairinterface.org/

OAI-RAN Features (PHY)

- PHY
  - The Physical layer implements 3GPP 36.211, 36.212, 36.213 and provides the following features
    - LTE release 8.6 compliant, and implements a subset of release 10
    - FDD (stable) and TDD configurations 1 and 3 (experimental)
    - Bandwidth: 5, 10 (stable), and 20 MHz (experimental)
    - Transmission modes: 1, 2 (stable), 3 (eNB only, experimental), 5, and 6 (experimental)
    - CQI/PMI reporting: aperiodic, feedback mode 3-0 and 3-1
    - All downlink (DL) channels are supported: PSS, SSS, PBCH, PCFICH, PHICH, PDCCH, PDSCH, PMCH
    - All uplink (UL) channels are supported: PRACH, PUSCH, PUCCH (format 1/1a/1b), SRS (experimental), DRS
    - HARQ support (UL and DL)
    - Highly optimized base band processing (including turbo decoder)

OAI-RAN Features (MAC)

- MAC
  - The MAC layer implements a subset of the 3GPP 36-321 release v8.6 in support of BCH, DLSCH,RACH, and ULSCH channels
  - The eNB MAC implementation includes
    - RRC interface for CCCH, DCCH, and DTCH
    - Schedulers
    - DCI generation
    - HARQ Support
    - RA procedures and RNTI management
    - RLC interface (AM, UM)
    - UL power control
    - Link adaptation
  - UE MAC implementation includes
    - PDU formats: all control elements and logical channels
    - RLC interface AM,UM, TM
    - RRC transparent interface for CCCH and BCCH
    - Buffer status reporting and scheduling request procedures
    - Power headroom reporting

OAI-RAN Features (RLC)

- The RLC layer implements a full specification of the 3GPP 36-322 release v9.3 for allthe three mode: transparent mode (TM), unacknowledged mode (UM), and acknowledge mode (AM)with the following characteristics
  - RLC TM (mainly used for BCCH and CCCH)
    - Neither segment nor concatenate RLC SDUs
    - Do not include a RLC header in the RLC PDU
    - Delivery of received RLC PDUs to upper layers
  - RLC UM (mainly used for DTCH)
    - Segment or concatenate RLC SDUs according to the TB size selected by MAC
    - Include a RLC header in the RLC PDU
    - Duplication detection
    - PDU reordering and reassembly
  - RLC AM, compatible with 9.3
    - Segmentation, concatenation, and reassembly
    - Padding
    - Data transfer to the user
    - RLC PDU retransmission in support of error control and correction
    - Generation of data/control PDUs

OAI-RAN Features (PDCP)

- The current PDCP is header compliant with 3GPP 36-323 Rel 10.1.0 and implement the following functions
  - User and control data transfer
  - Sequence number management
  - RB association with PDCP entity
  - PDCP entity association with one or two RLC entities
  - Integrity check and encryption using the AES and Sonw3G algorithms

OAI-RAN Features (RRC)

- The RRC layer, shared between the UE and the ENB, performs the control of the radio interface. It is based on 3GPP 36.331 v9.2.0
- The control procedures available in the LTE platformare the following
  - System Information broadcast (SIB 1, 2, 3, and 13)
  - RRC connection establishment
  - RRC connection reconfiguration (addition and removal of radio bearers, connection release)
  - RRC connection release
  - Inter-frequency measurement collection and reporting at UE and eNB
  - eMBMS for multicast and broadcast

5G NR Development and Releases

- Development process
  - Main branch for development is oai/develop-nr
  - When a new feature for 5G-NR is developed, a feature branch is created starting from the latest oai/develop-nr
  - When finished it is merged back to oai/develop-nr
  - Further, oai/develop-nr is regularly updated with oai/develop
- Hardware requirements
  - PC
    - The processing requirements for 5G-NR are much higher as for 4G, so a high end PC or server is needed
    - Intel Core i7 6900K (8 cores), 16GB DDR, 480GB SSD (~2200€)
      - Allows SW LDPC on 3 cores (1 segment per slot, 3 slots decoded in parallel, up to 30Mb/s) or LDPC on FPGA (up to 300Mb/s on 80MHz SISO)
    - Intel Core i9 7980EX (18 cores) shall support future configurations incl. MIMO (~3700 €)
      - Allows parallel LDPC SW decoder on 9 or 12 cores, up to 3 (tested) or 5 (ongoing) segments per slot or 140Mb/s or LDPC on FPGA (same as above)
    - Intel(R) Xeon(R) Gold 6154 CPU @ 3.00GHz with 18 cores and 2x10Gbit Ethernet

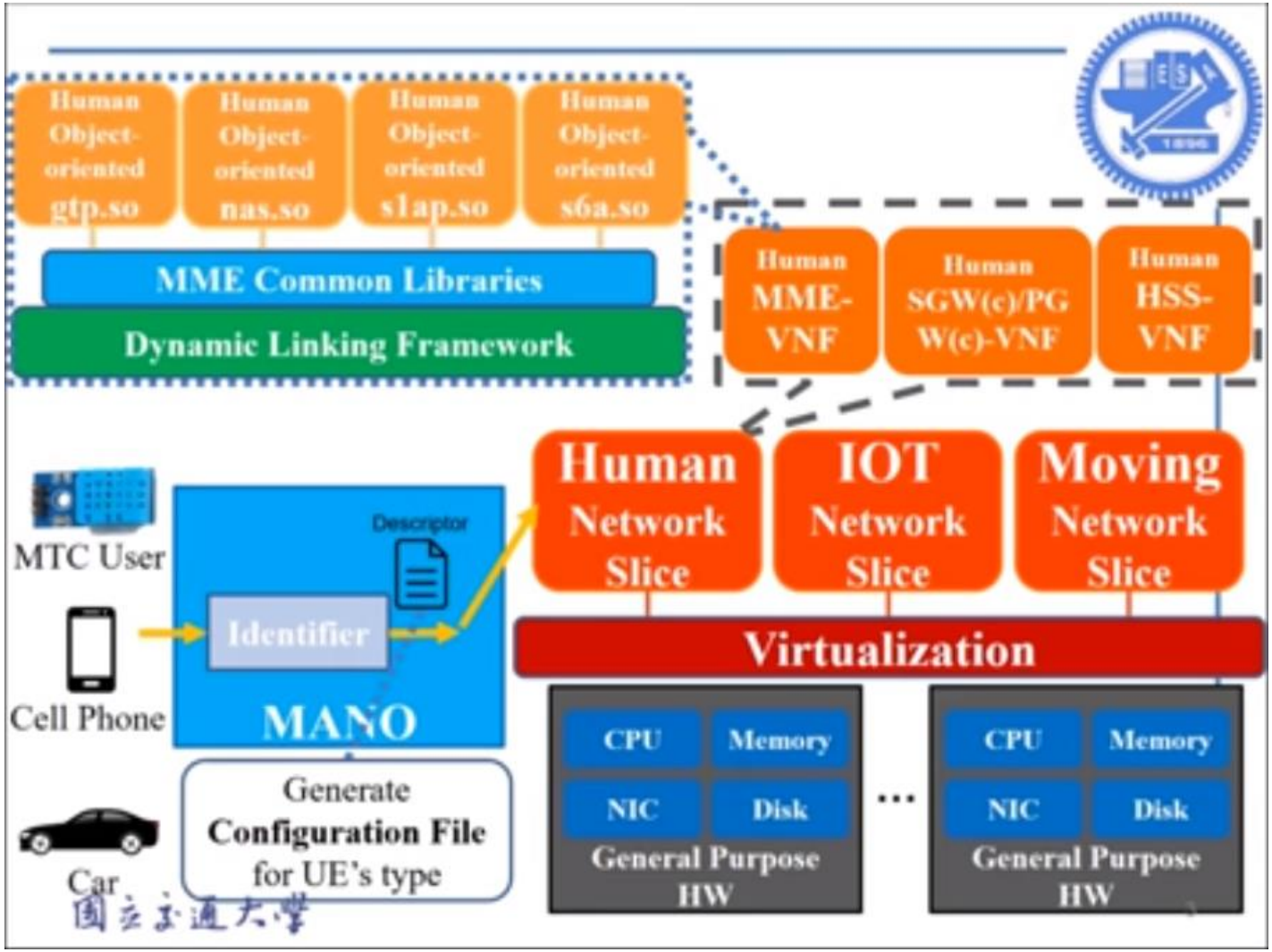
5G NR Development and Releases (Cont.)

- SDR
  - USRP N310
    - This is the latest version of the USRP designed specifically for 5G-NR
    - It will support bandwidth up to 100MHz
  - USRP X310
    - This older platform will also work with 5G-NR, but only supports bandwidths up to 80MHz with 3/4 sampling
  - SYRTEM platform
    - The SYRTEM platform is based on the Xilinx EVALUATION KIT ZYNQ-7000 ZC706 (ex. supplier Digikey ref: 122-1904-ND, 2427€)and the Analog Device Transceiver type ADRV9371 (ex. supplier : Digikey ADRV9371-W/PCBZ-ND ('WIDE TUNING RANGE 300MHZ-6HGZ' version) 1169€)
- LDPC offload
  - As a high performance alternative to the software LDPC decoder included in this release, the decoder can also be offloaded to another FPGA board (EVALUATION KIT ZYNQ-7000 ZC706)
  - The FPGA binary image provided by Creonic (https://www.creonic.com) while the drivers are provided by Syrtem (http://www.syrtem.com/)
- Operating system
  - Ubuntu 16.04 or 17.10 with the lowlatency kernel option
  - CentOS Linux release 7.4.1708 (Core)

RECO

- A Reconfigurable Core Network for Future 5G Communication Systems
  - RECO is an implementation of reconfigurable core network that demonstrates how to implement customized virtual network entities efficiently to suit for different types of users with different characteristics
  - It is mainly modified from openair-cn, a core network developed by EURECOM
  - http://www.reconet.org/reco/

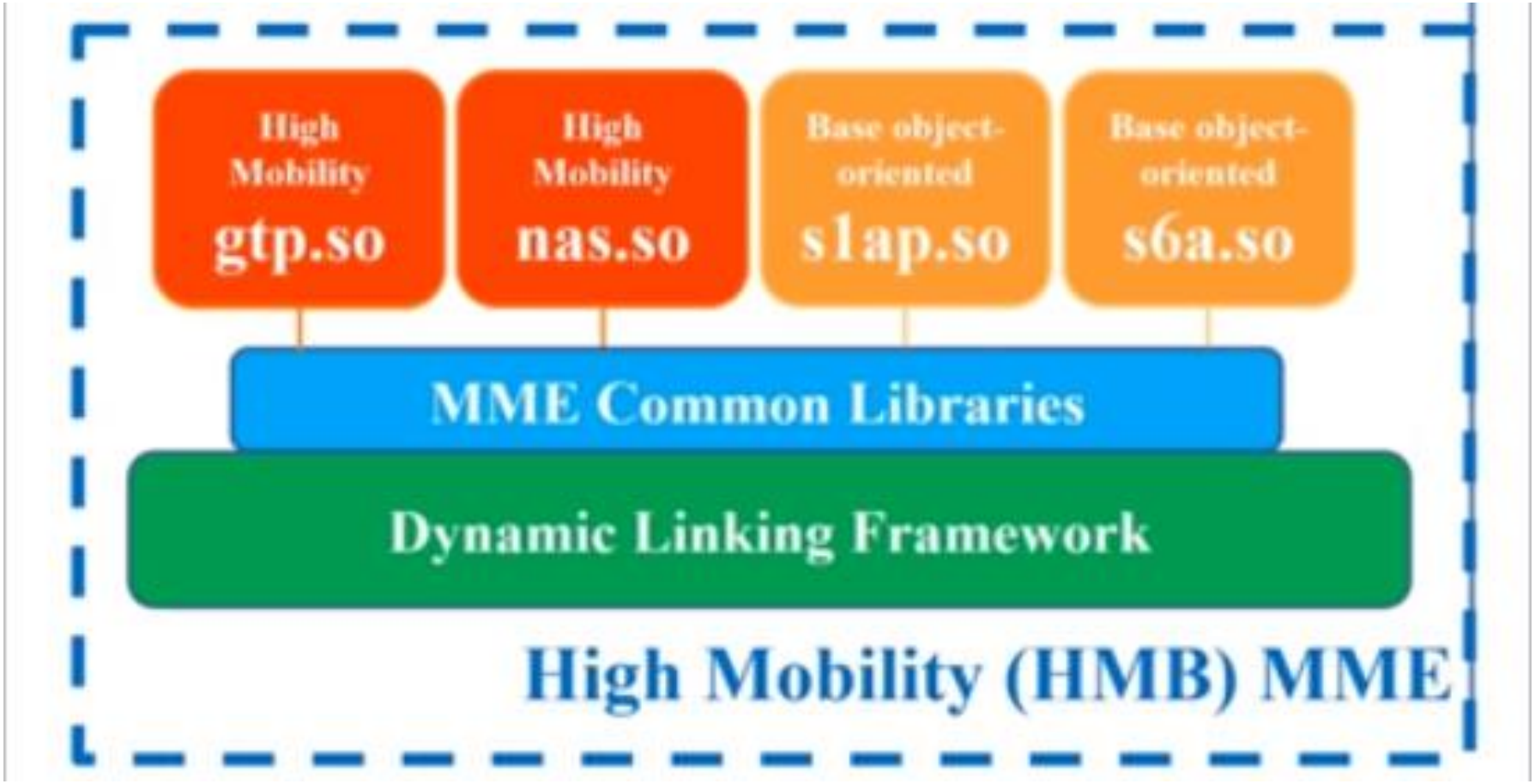
Overview of RECO



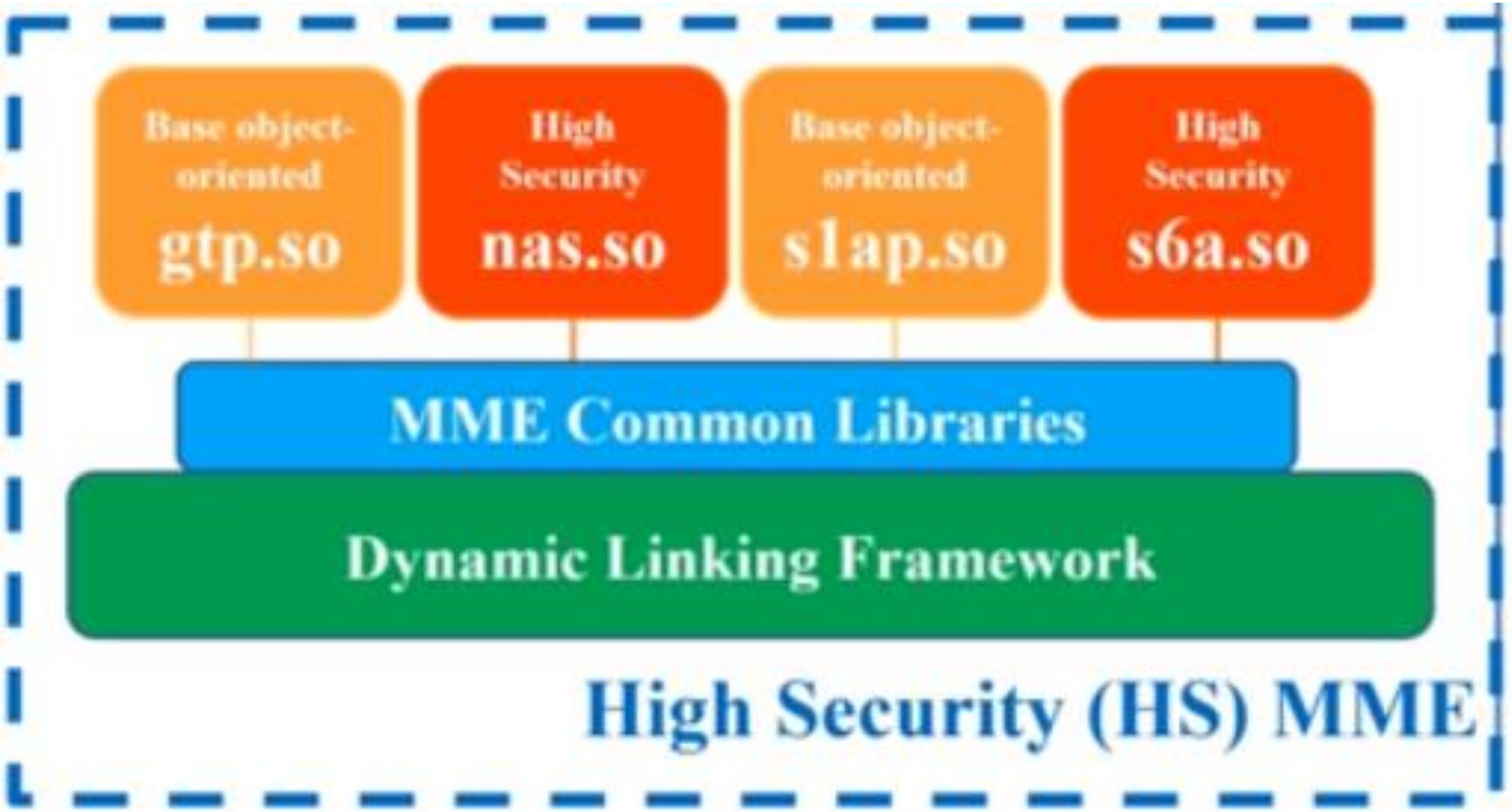
Reconfigure MME

- Dynamic Linking Framework
  - Pares configuration file, load and initialize corresponding modules
- MME Common Libraries
  - Common MME libraries which different types of users share
  - e.g., UDP, SCTP, hash table
- Object-oriented modules
  - Customized modules which differ between different types of users

Example: MME for High Speed Rail



Example: MME for eHealth

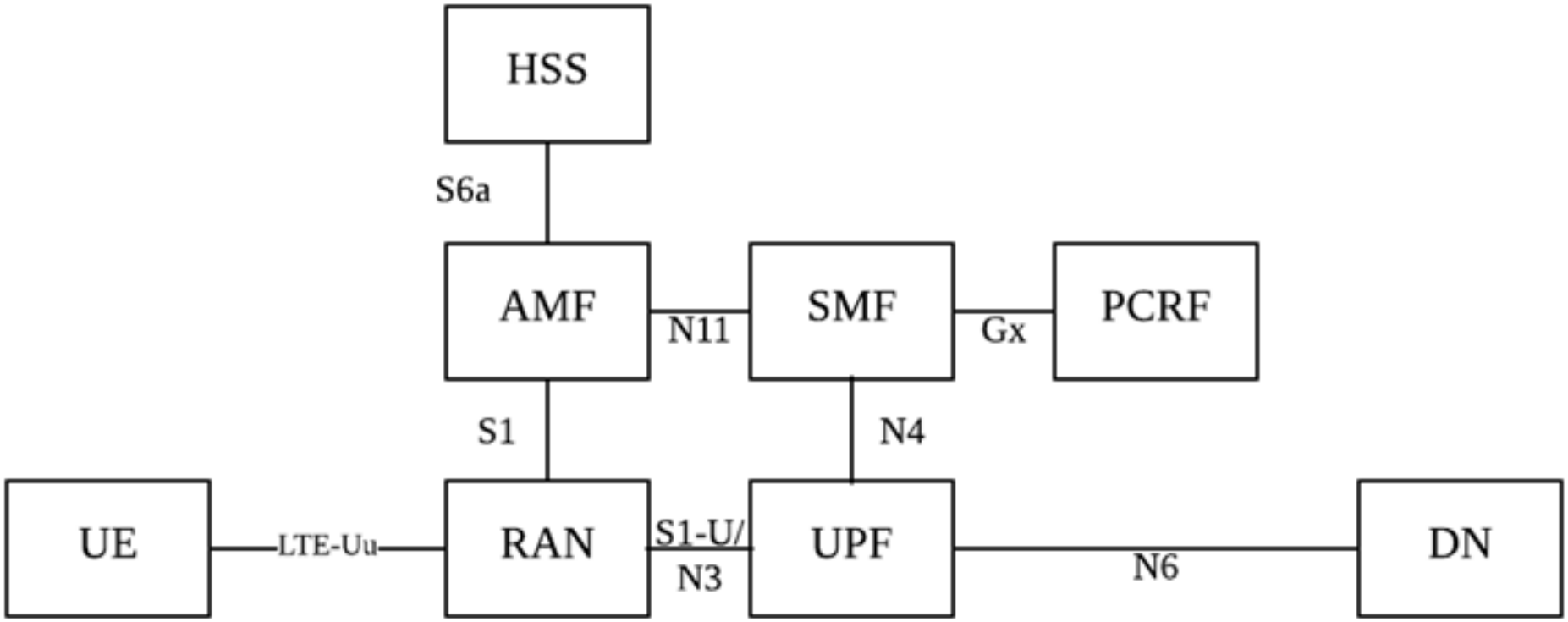


Free5GC

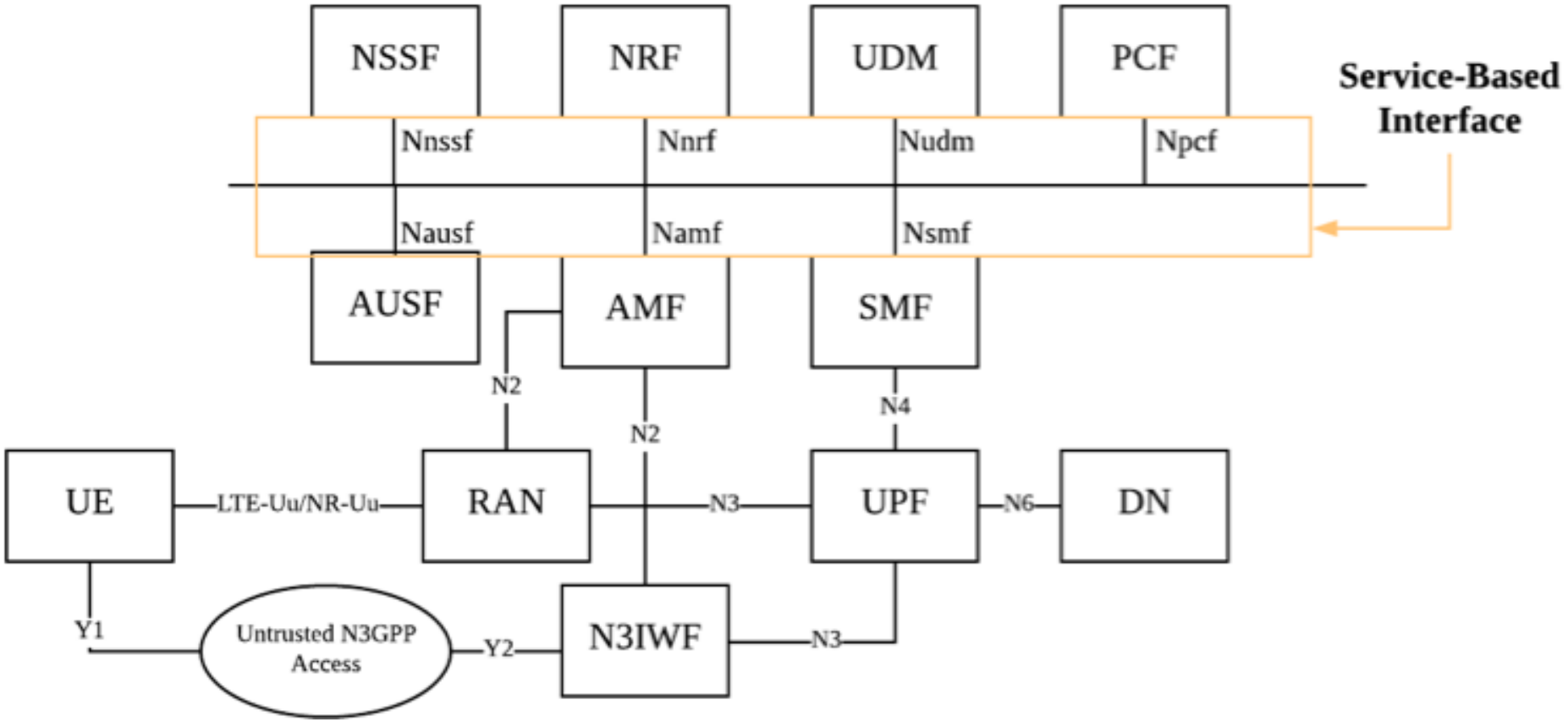
- An open-source project for 5th generation (5G) mobile core network
- The ultimate goal of this project is to implement 3GPP Release 15 (R15) and Release 16 (R16) 5G core network (5GC)
  - current version only implements three most important components in 5GC, mainly for the enhance Mobile Broadband (eMBB)
    - Access and Mobility Management Function (AMF)
    - Session Management Function (SMF)
    - User Plane Function (UPF)
- Major contributors are with National Chiao Tung University (NCTU)

https://www.free5gc.org/

Stage 1 Architecture of Free5GC

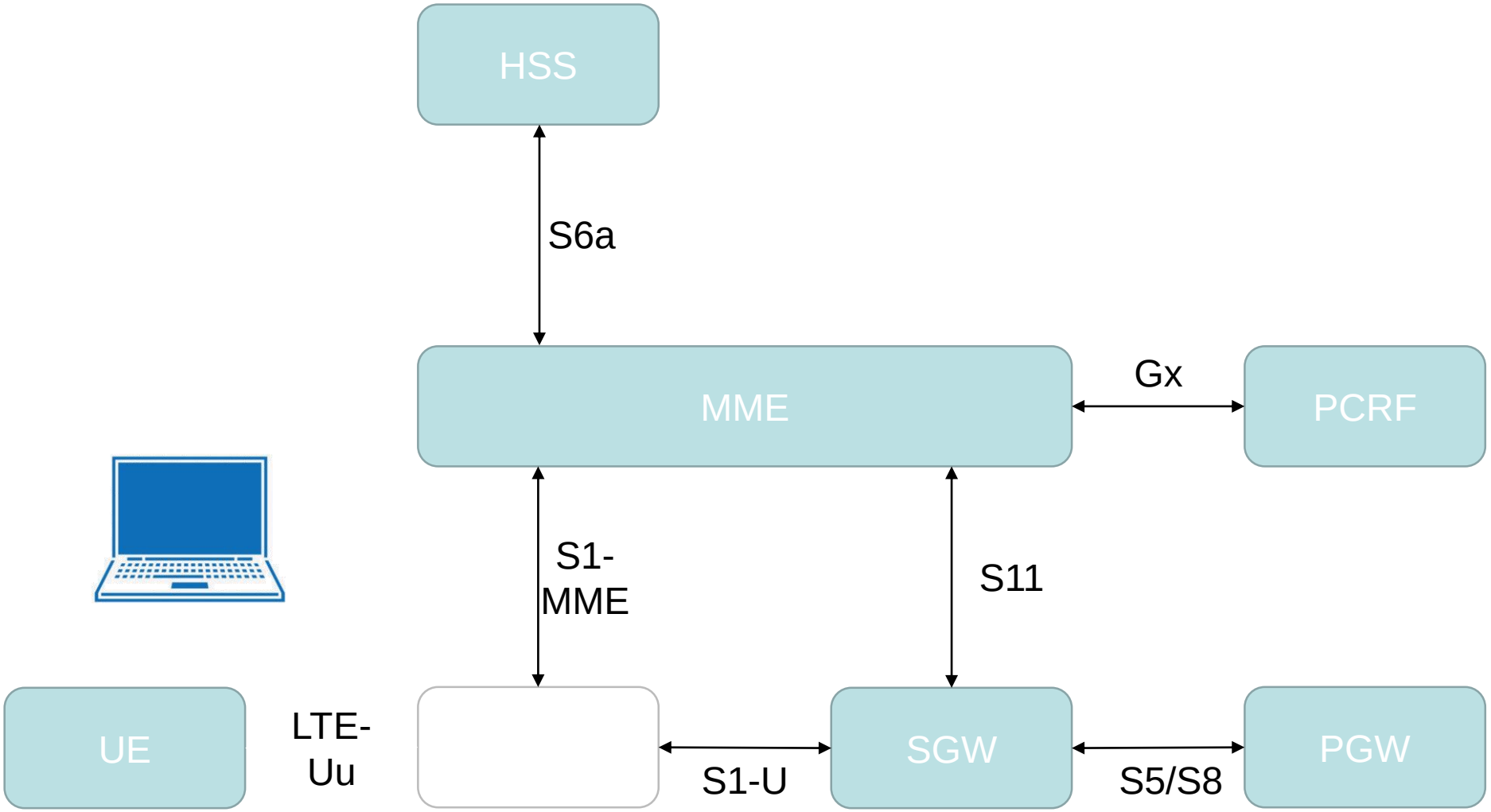


Stage 2 Architecture of Free5GC

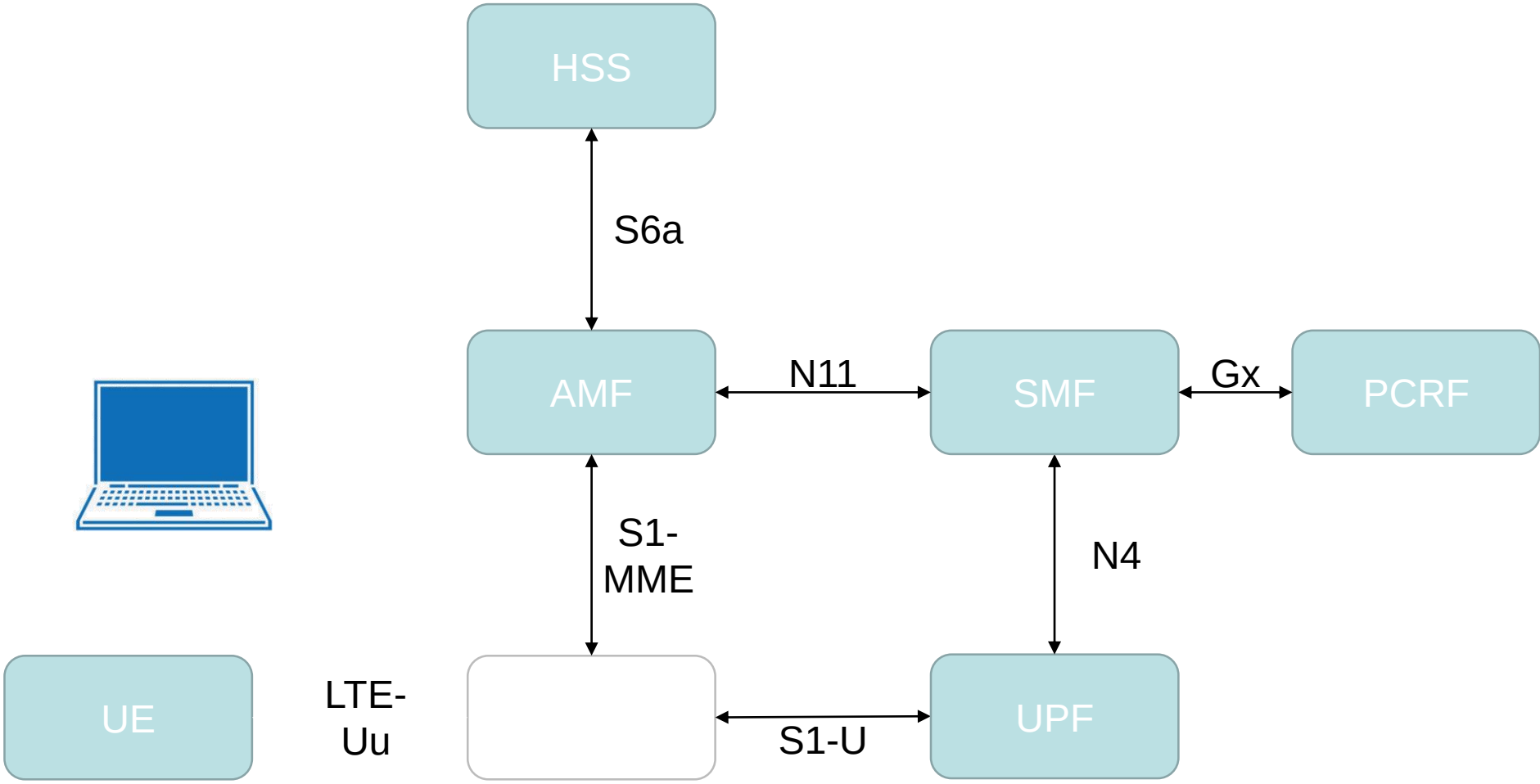


- Will be released in September 2019

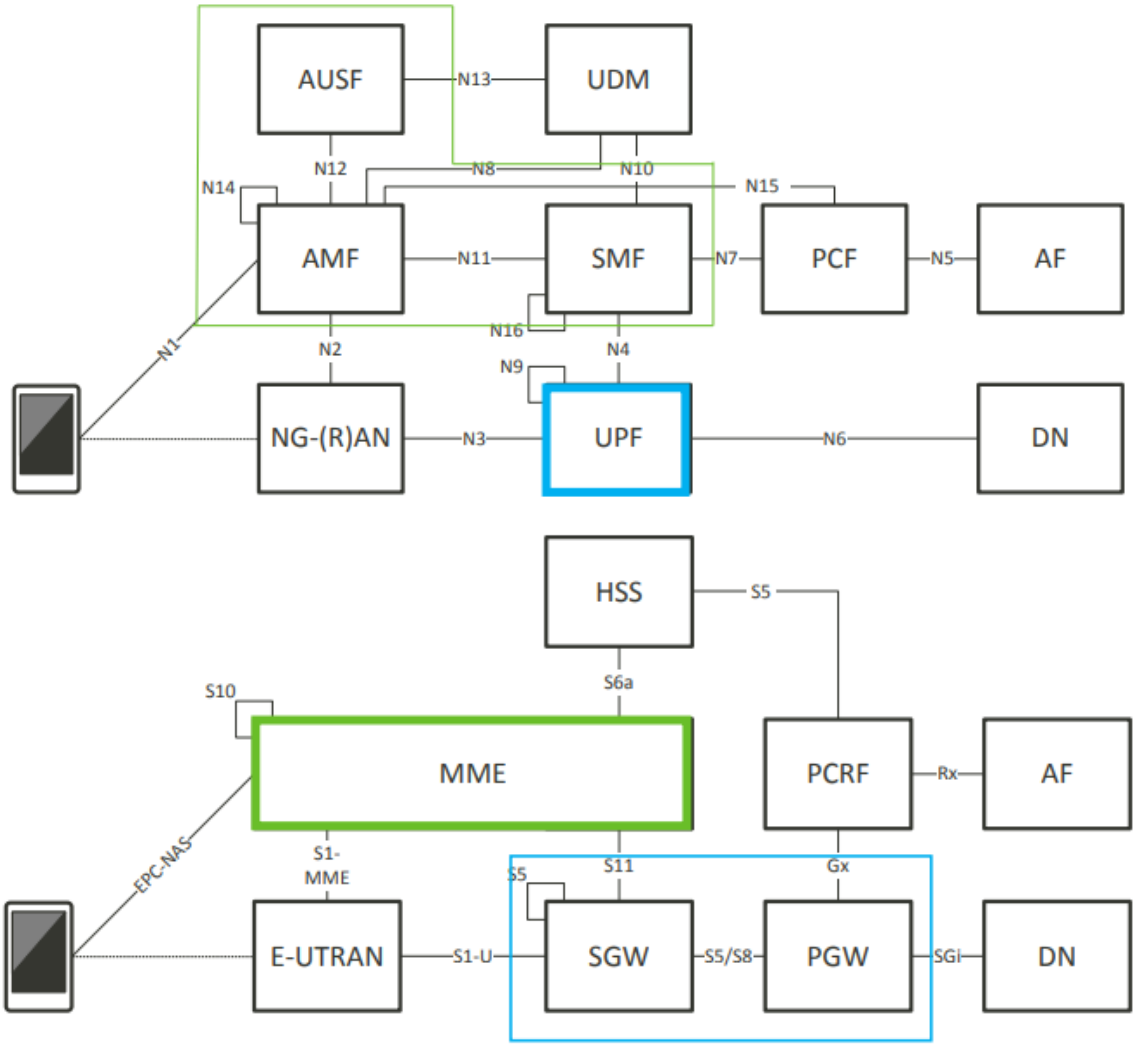
LTE-A Laboratory Platform



Free5GC Laboratory Platform



EPC <> 5GC


AUSF
N13
UDM
N12
N8
N10
N14
N15
AMF
N11
SMF
N7
PCF
N5
AF
N16
N1
N2
N4
N9
NG-(R)AN
N3
UPF
N6
DN
HSS
S5
S6a
S10
MME
PCRF
Rx
AF
EPC-NAS
S1-
MME
S5
S11
Gx
E-UTRAN
S1-U
SGW
S5/S8
PGW
SGi
DN

EPC<>5GC Correspondence

EPC <> 5GC Correspondence

EPC		5GC
MME	↔	AMF + SMF + AUSF
SGW	↔	UPF
PGW	↔	UPF
PCRF	↔	PCF
HSS	↔	UDM

EPC		5GC
EPC-NAS	↔	N1
S1-MME	↔	N2
S1-U	↔	N3
S11	↔	N4
Rx	↔	N5
SGi	↔	N6

Outline

- 3GPP 4G/5G System Architecture
- 4G/5G Open Source Platform
  - OpenLTE
  - OpenAirInterface
  - RECO
  - Free5GC
- srsLTE
  - srsUE, srsENB, srsEPC
- nukLWA/nukxDC

srsLTE

- srsLTE is a free and open-source LTE software suite developed by SRS (www.softwareradiosystems.com)
- srsLTE includes
  - srsUE - a complete SDR LTE UE application featuring all layers from PHY to IP
  - srsENB - a complete SDR LTE eNodeB application
  - srsEPC - a light-weight LTE core network implementation with MME, HSS and S/P-GW
  - A highly modular set of common libraries for PHY, MAC, RLC, PDCP, RRC, NAS, S1AP and GW layers
- srsLTE is released under the AGPLv3 license and uses software from the OpenLTE project (http://sourceforge.net/projects/openlte) for some security functions and for RRC/NAS message parsing

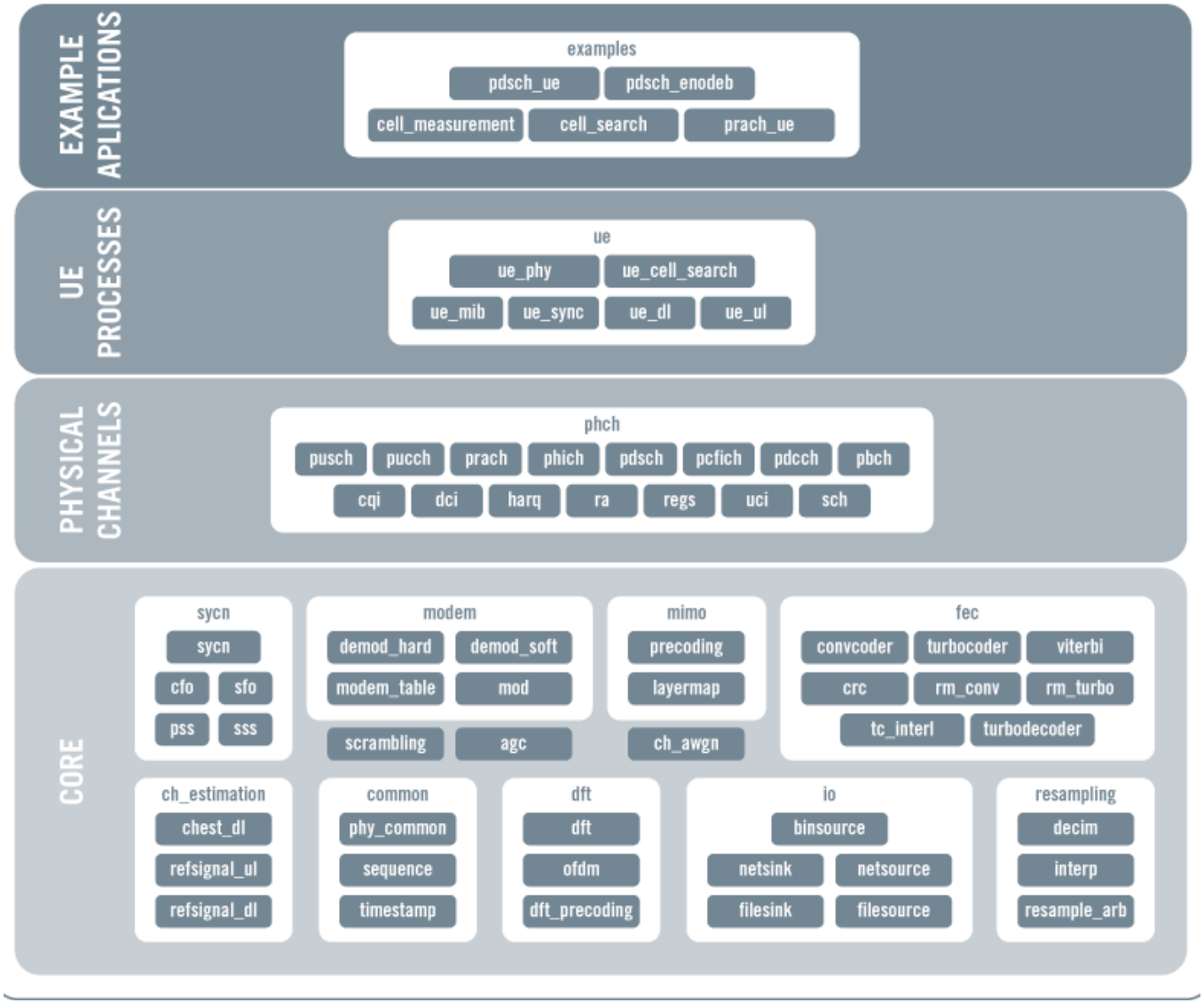
Reference: https://github.com/srsLTE/srsLTE

srsLTE Common Features

- LTE Release 10 aligned
- Tested bandwidths: 1.4, 3, 5, 10, 15 and 20 MHz
- Transmission mode 1 (single antenna), 2 (transmit diversity), 3 (CCD) and 4 (closed-loop spatial multiplexing)
- Frequency-based ZF and MMSE equalizer
- Evolved multimedia broadcast and multicast service (eMBMS)
- Highly optimized Turbo Decoder available in Intel SSE4.1/AVX2 (+100 Mbps) and standard C (+25 Mbps)
- MAC, RLC, PDCP, RRC, NAS, S1AP and GW layers
- Detailed log system with per-layer log levels and hex dumps
- MAC layer wireshark packet capture
- Command-line trace metrics
- Detailed input configuration files
- Channel simulator for EPA, EVA, and ETU 3GPP channels
- ZeroMQ-based fake RF driver for I/Q over IPC/network

Reference: https://github.com/srsLTE/srsLTE

Overview of srsLTE



Reference: http://www.softwareradiosystems.com

srsUE Features

- FDD and TDD configuration
- Carrier Aggregation support
- Cell search and synchronization procedure for the UE
- Soft USIM supporting Milenage and XOR authentication
- Hard USIM support using PCSC framework
- Virtual network interface *tun_srsue* created upon network attach
- QoS support
- 150 Mbps DL in 20 MHz MIMO TM3/TM4 configuration in i7 Quad-Core CPU.
- 75 Mbps DL in 20 MHz SISO configuration in i7 Quad-Core CPU.
- 36 Mbps DL in 10 MHz SISO configuration in i5 Dual-Core CPU.

srsENB Features

- FDD configuration
- Round Robin MAC scheduler with FAPI-like C++ API
- SR support
- Periodic and Aperiodic CQI feedback support
- Standard S1AP and GTP-U interfaces to the Core Network
- 150 Mbps DL in 20 MHz MIMO TM3/TM4 with commercial UEs
- 75 Mbps DL in SISO configuration with commercial UEs
- 50 Mbps UL in 20 MHz with commercial UEs
- User-plane encryption

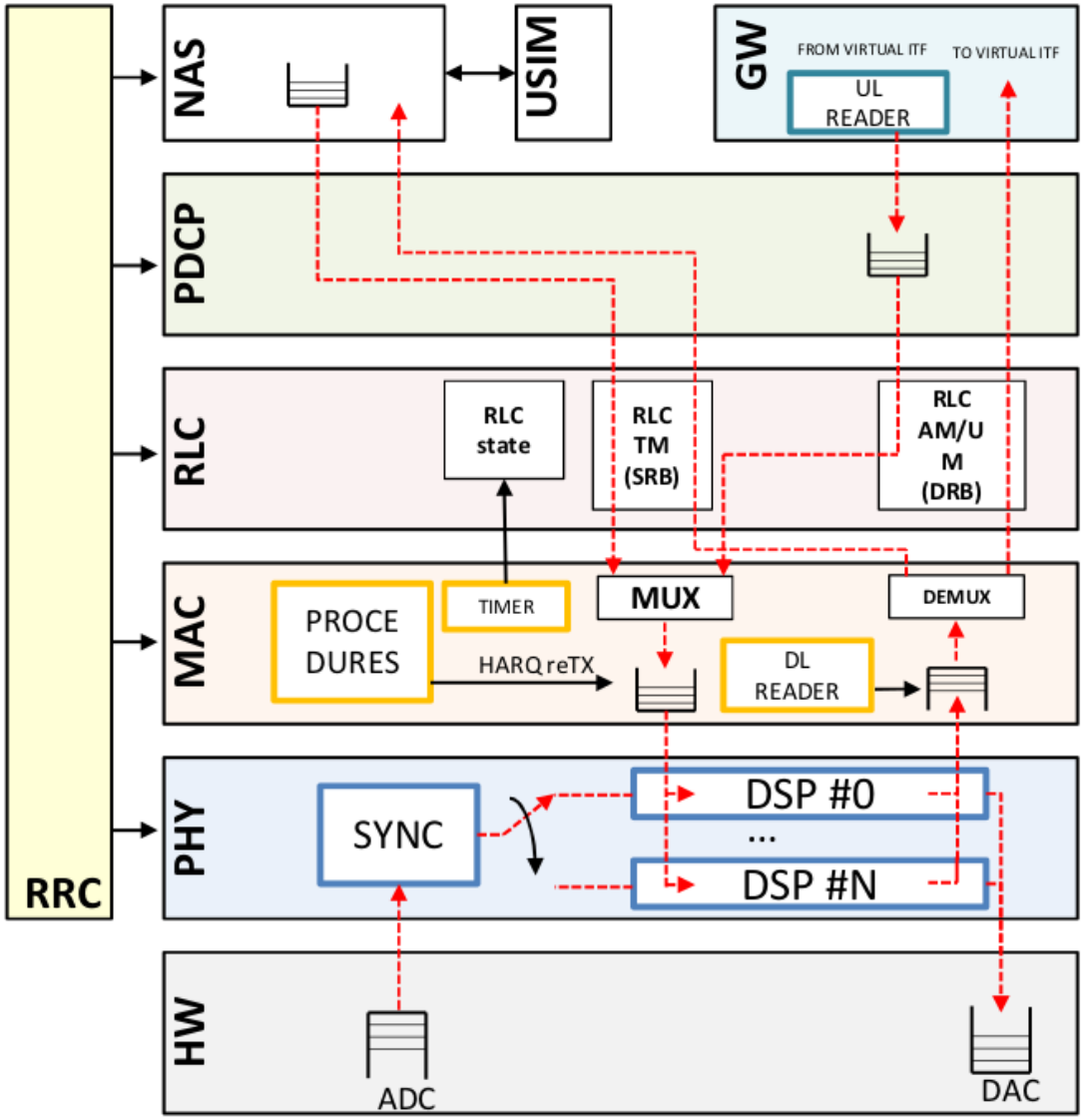
srsEPC Features

- Single binary, light-weight LTE EPC implementation with:
  - MME (Mobility Management Entity) with standard S1AP and GTP-U interface to eNB
  - S/P-GW with standard SGi exposed as virtual network interface (TUN device)
  - HSS (Home Subscriber Server) with configurable user database in CSV format
- Support for paging

Hardware

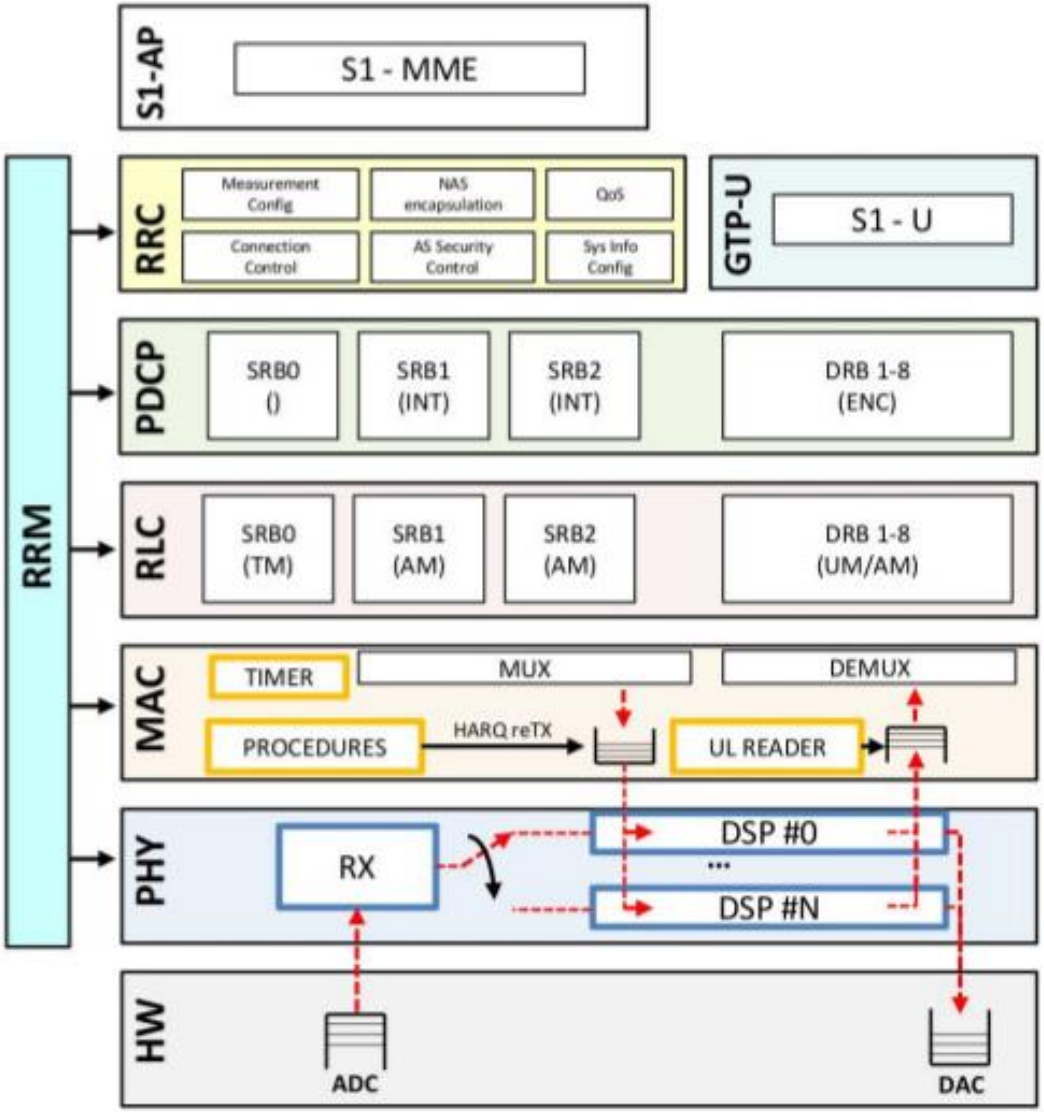
- The library currently supports the Ettus Universal Hardware Driver (UHD) and the bladeRF driver
- Thus, any hardware supported by UHD or bladeRF can be used
- There is no sampling rate conversion, therefore the hardware should support 30.72 MHz clock in order to work correctly with LTE sampling frequencies and decode signals from live LTE base stations

Overview of srsUE



Reference: http://www.softwareradiosystems.com

Overview of srsENB



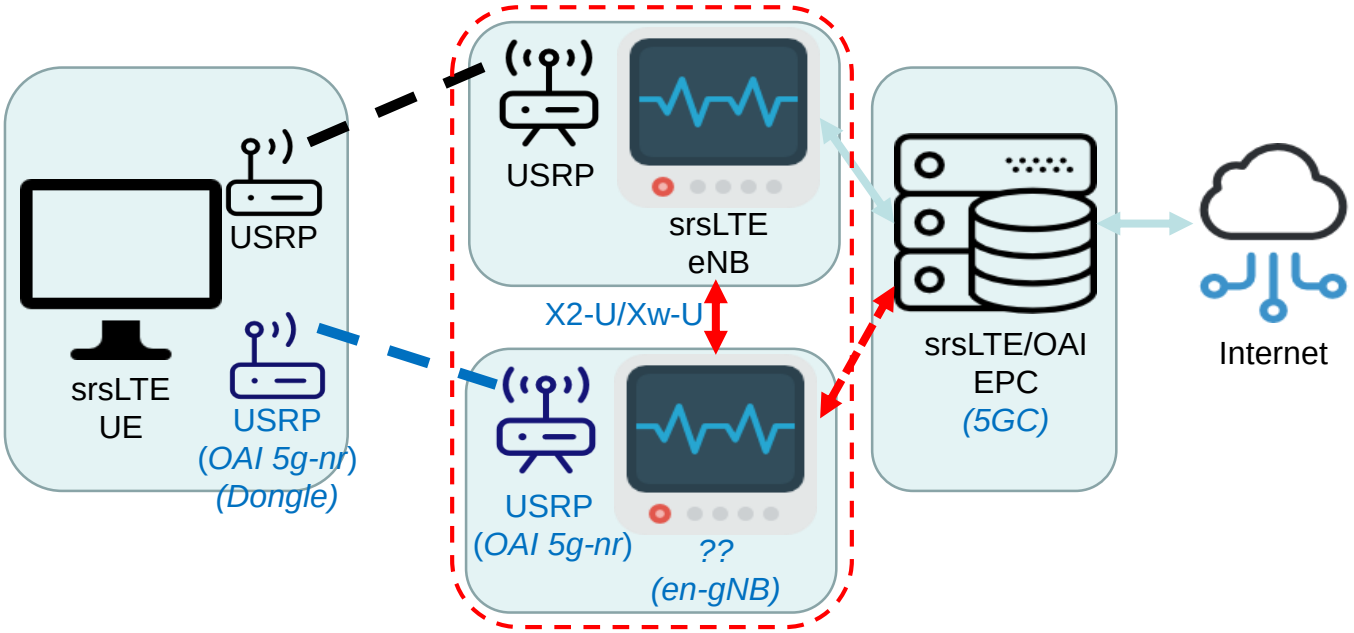
Reference: http://www.softwareradiosystems.com

Outline

- 3GPP 4G/5G System Architecture
- 4G/5G Open Source Platform
  - OpenLTE
  - OpenAirInterface
  - RECO
  - Free5GC
- srsLTE
  - srsUE, srsENB, srsEPC
- nukLWA/nukxDC

nukLWA/nukxDC實驗平台

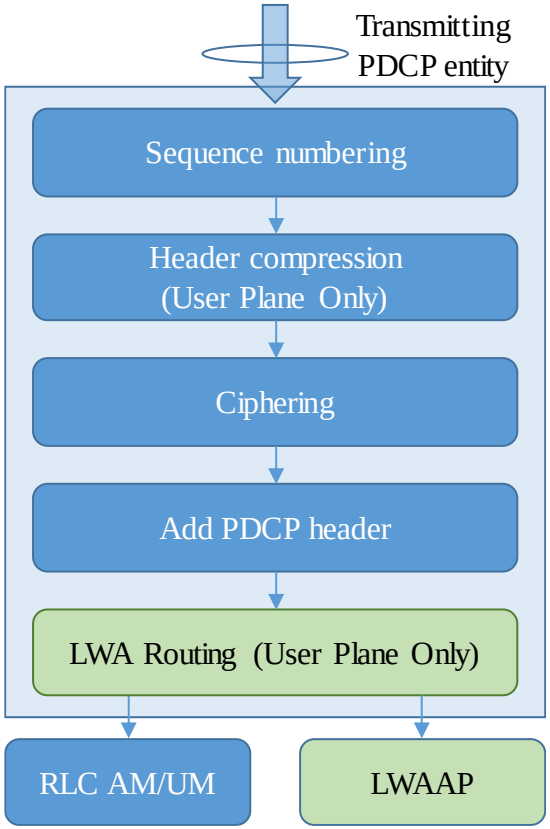
nukxDC



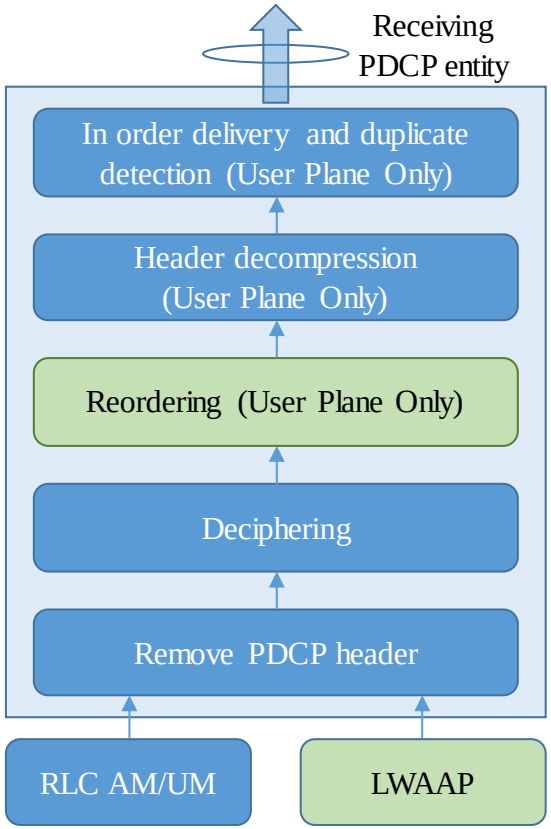
Access Network

1. Ethernet for Emulator
2. WiFi for LWA (nukLWA)
3. LTE for eNB (srsLTE)
4. OAI 5g-nr for gNB (Pending)

nukLWA/nukxDC



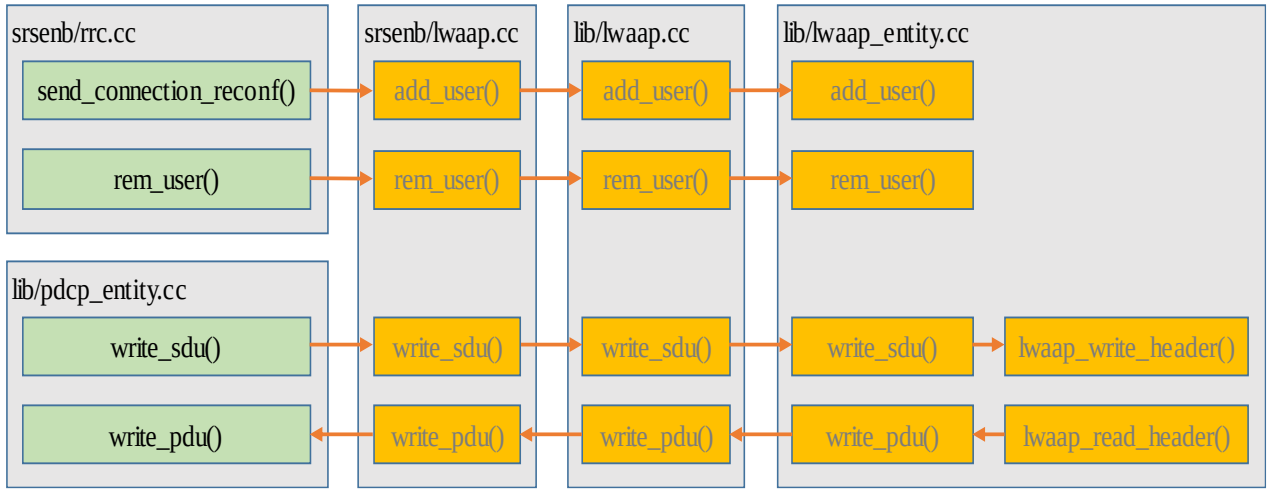
eLWA雙介面傳送模組



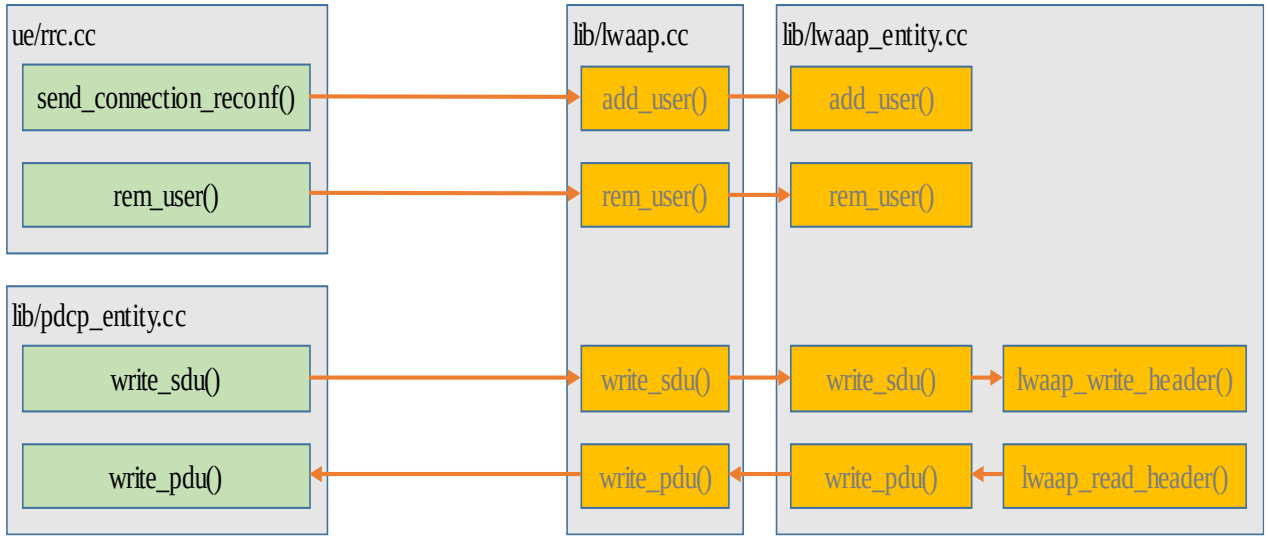
eLWA雙介面接收模組

WLAN介面傳送與接收模組的程式架構

eNB端

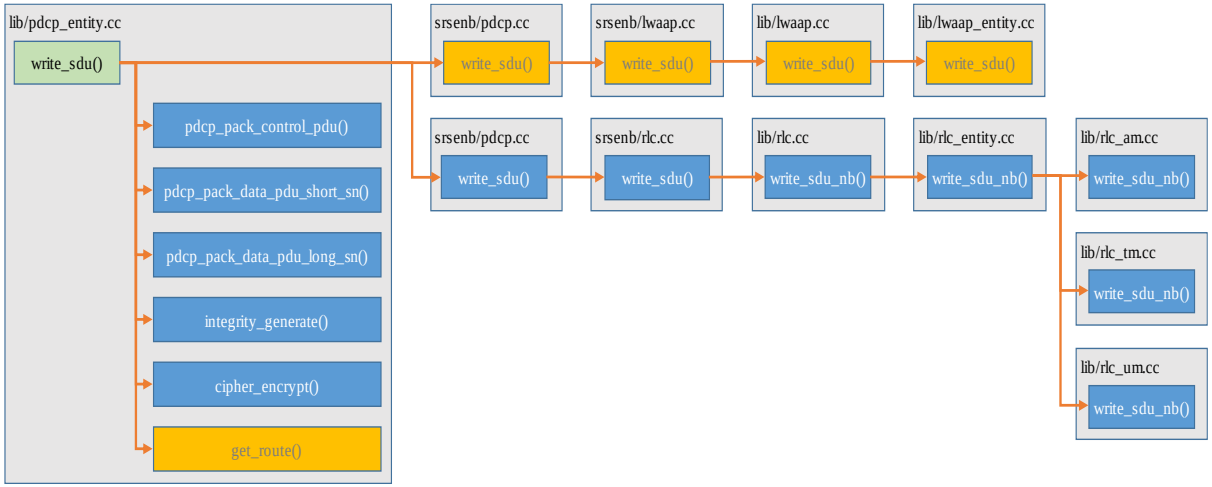


UE端

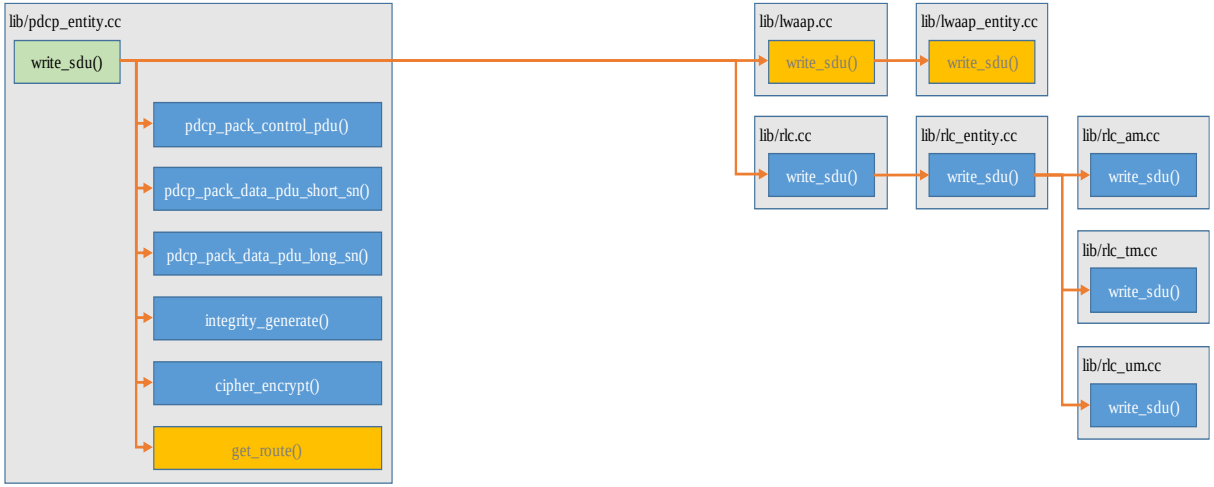


雙介面傳送模組的程式架構

eNB端

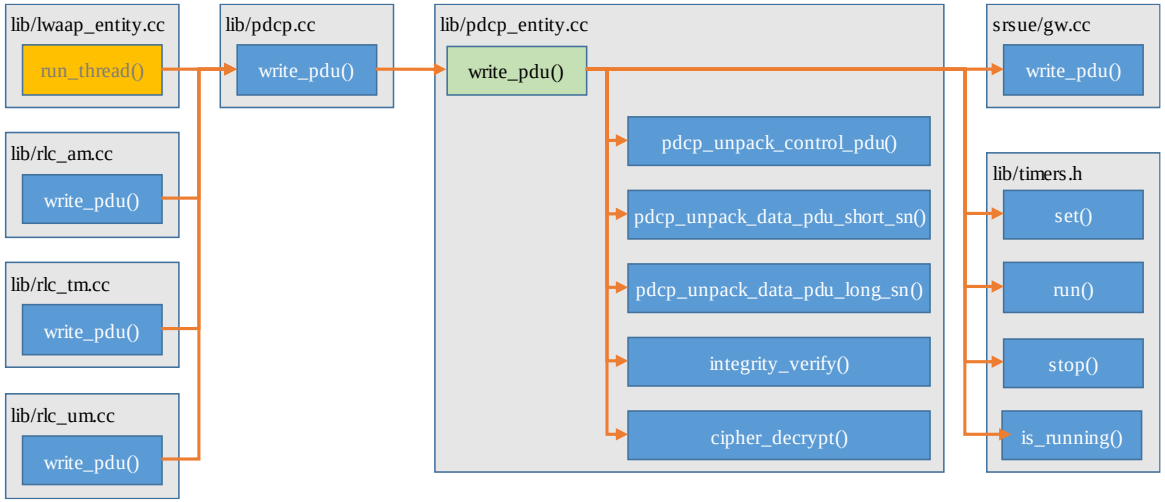


UE端

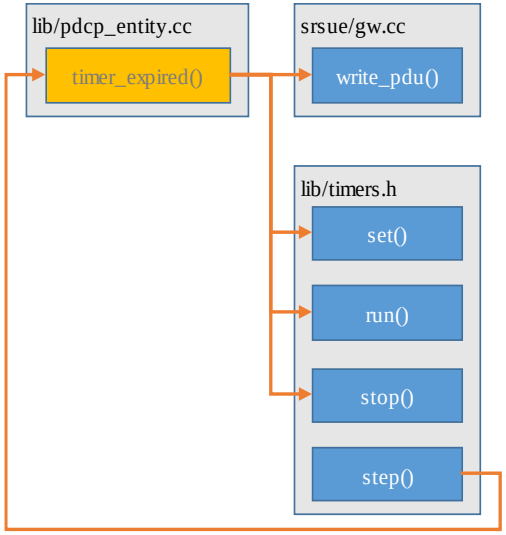


雙介面接收模組的程式架構

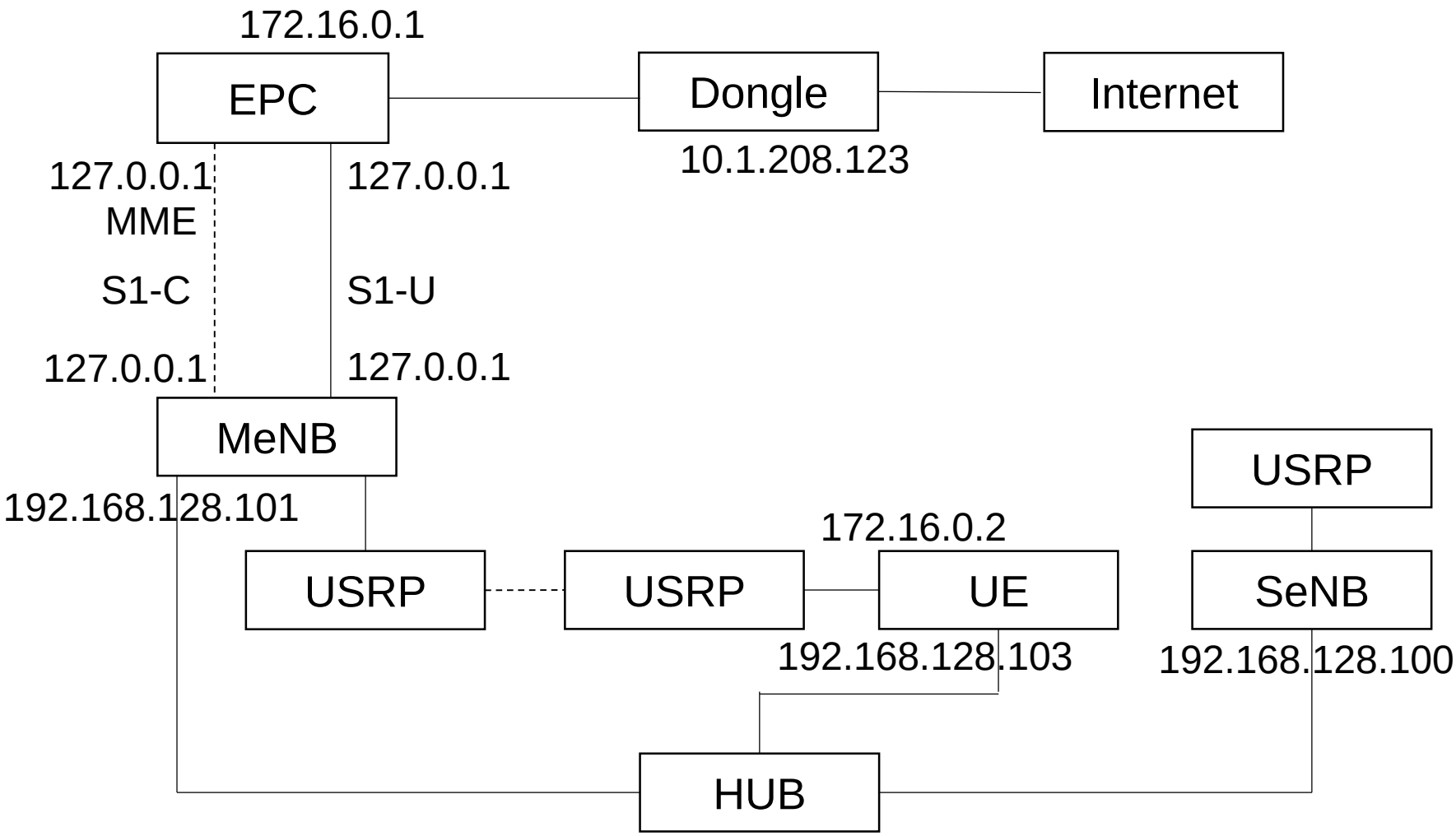
eNB端



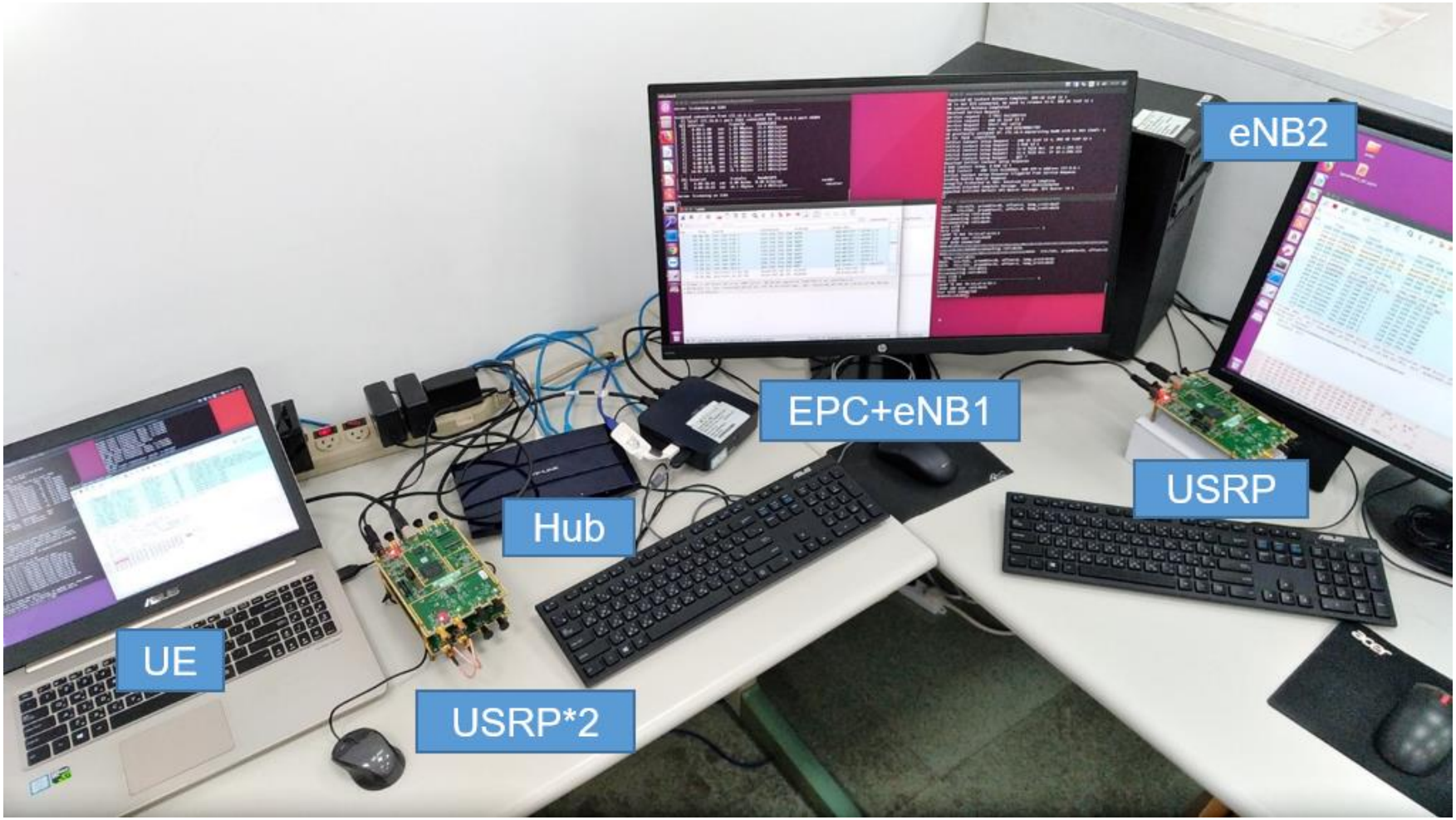
UE端



srsLTE/nukxDC 實驗架構



實驗環境



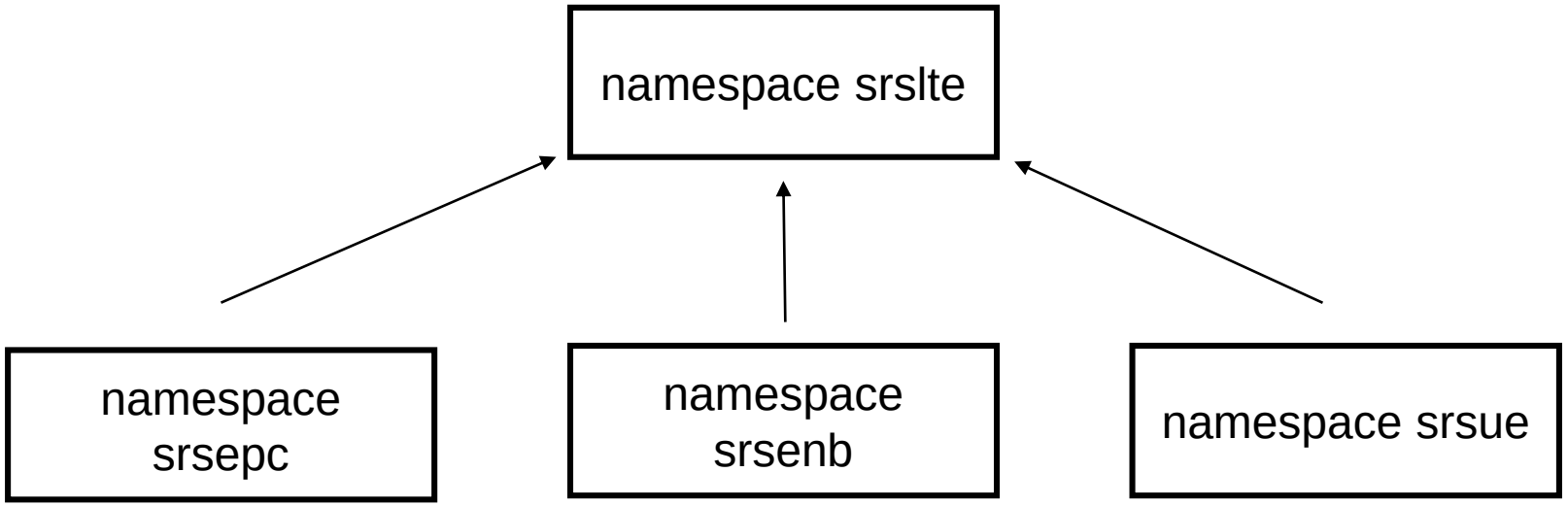
軟硬體環境－硬體

名稱	規格	數量	目的
EPC+eNB1	電腦型號 : ASUS VivoMini UN65H	1	啟動MME,S-GW,P-GW
	Ethernet Network Cards	2	一張連接內部網路 (PCI-E : Realtek RTL8111/8168) 一張連接外部網路 (USB : TP-LINK UE300)
	USRP B210	1	啟動srsLTE eNB
eNB2	電腦規格 : CPU : i7-6700 , RAM : 32G	1	模擬第二個基地站
	USRP B210	1	啟動srsLTE eNB
UE	電腦型號 : ASUS NB M580V	1	模擬 UE
	USRP B210	1	啟動srsLTE UE
Hub	型號 : TP-LINK WR1043ND	1	分配內部網路

軟硬體環境 – 軟體

名稱	軟體	版本
EPC	OS : Ubuntu	Ubuntu 16.04
		Kernel : 4.15.0-041500-lowlatatency
	srsLTE EPC	srsLTE 18.06.1 470953bf9c5875646e4d5049c8f213d202fa84fd
eNB	OS : Ubuntu	Ubuntu 16.04
		Kernel : 4.15.0-041500-lowlatatency
	srsLTE eNB	srsLTE 18.06.1 470953bf9c5875646e4d5049c8f213d202fa84fd
UE	OS : Ubuntu	Ubuntu 16.04
		Kernel : 4.15.0-041500-lowlatatency
	srsLTE UE	srsLTE 18.06.1 470953bf9c5875646e4d5049c8f213d202fa84fd

srsLTE 結構

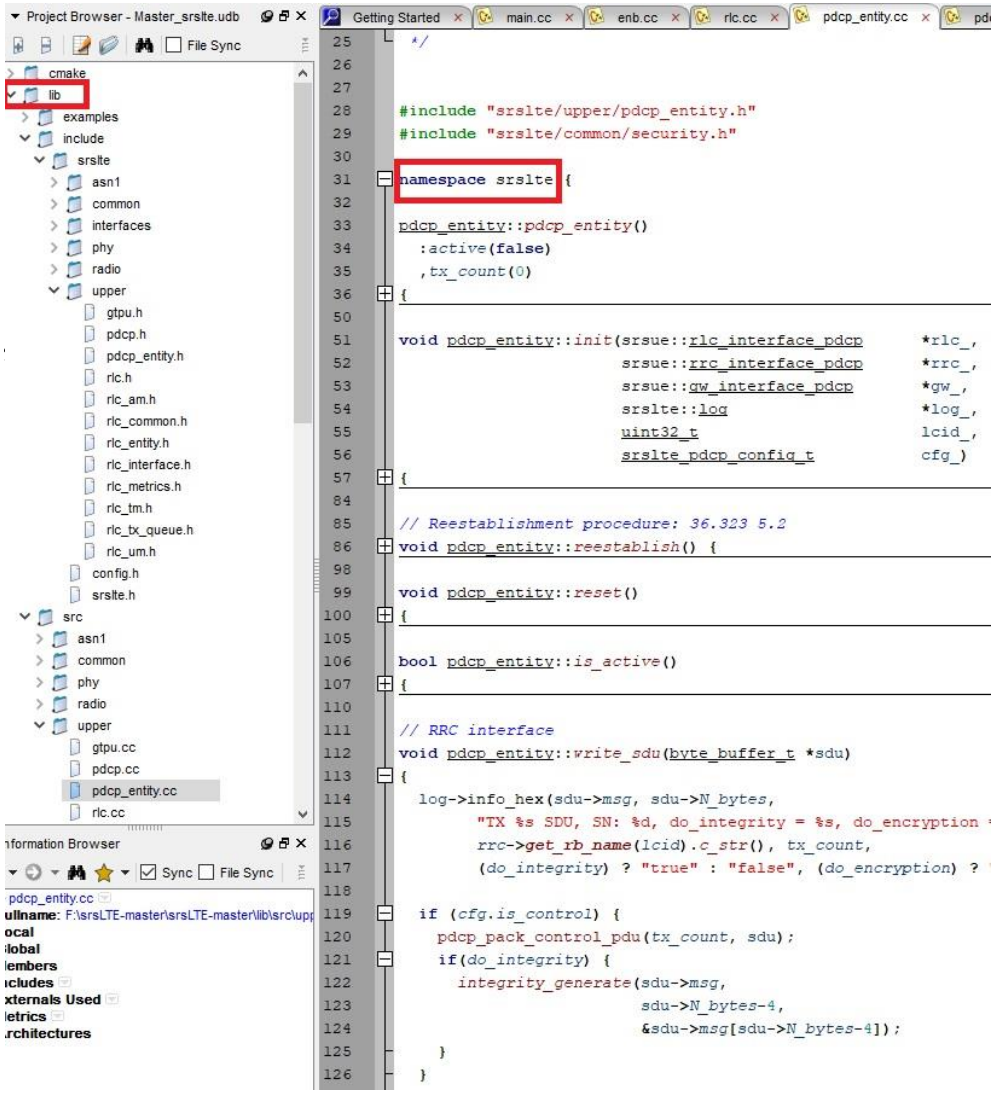


srsepc、srsenb、srsue有一些程式碼是共用的，共用的程式碼會寫在namespace srslte，當有需要使用的時候會呼叫srslte裏面的程式碼

namespace srslte

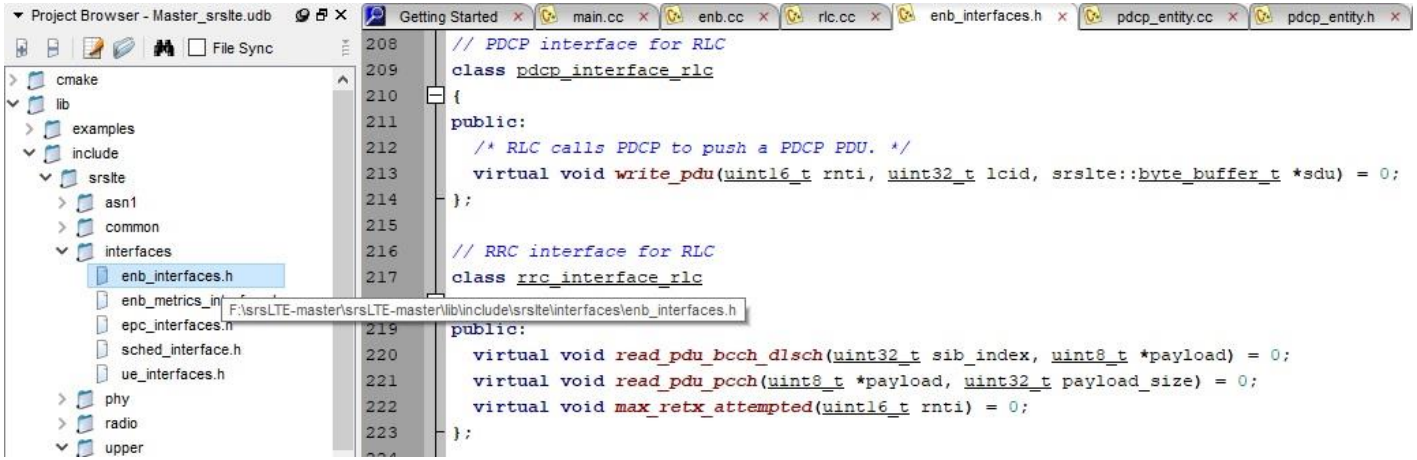
namespace srslte
{
// pdcp all function
class pdcp{…};
}

//eNB及UE共用pdcp的功能



enb_interface.h

```
namespace srsenb
{
        class pdcp_interface_rlc{…};      // pdcp function for rlc
        class rlc_interface_pdcp{…};      // rlc function for pdcp
}
//eNB所有界面會寫在enb_interfaces.h
```



pdcp init

srsenb\src\upper\pdcp.cc

```
void pdcp::init(rlc_interface_pdcp* rlc_, …, srslte::log* pdcp_log_)
{
        rlc   = rlc_;
        rrc   = rrc_;
        gtpu  = gtpu_;
        log_h = pdcp_log_;
        pool = srslte::byte_buffer_pool::get_instance();
        pthread_rwlock_init(&rwlock, NULL);
}
```

//srsenb界面的運作部份會寫在srsenb\src\，同時如果srsenb的界面有需要使用其他層的function，它在初始化的時候會取得其他層的界面，有需要時再呼叫其他層的界面來使用其他層的function。

srsenb\src\enb.cc

```
namespace srsenb
{
        bool enb::init(all_args_t *args_)
        {
                pdcp_log.init("PDCP ", logger);
                pdcp_log.set_level(level(args->log.pdcp_level));
                pdcp_log.set_hex_limit(args->log.pdcp_hex_limit);
                pdcp.init(&rlc, &rrc, &gtpu, &pdcp_log);
        }
}
```

//srsenb所有程式的界面初始化會在srsenb\src\enb.cc開始

pdcp init

srsenb\src\upper\pdcp.cc

```
void pdcp::init(rlc_interface_pdcp* rlc_, …, srslte::log* pdcp_log_)
{
        rlc   = rlc_;
        rrc   = rrc_;
        gtpu  = gtpu_;
        log_h = pdcp_log_;
        pool = srslte::byte_buffer_pool::get_instance();
        pthread_rwlock_init(&rwlock, NULL);
}
```

//上一頁的pdcp.init會呼叫srsenb\src\upper\pdcp.cc裏面的init()

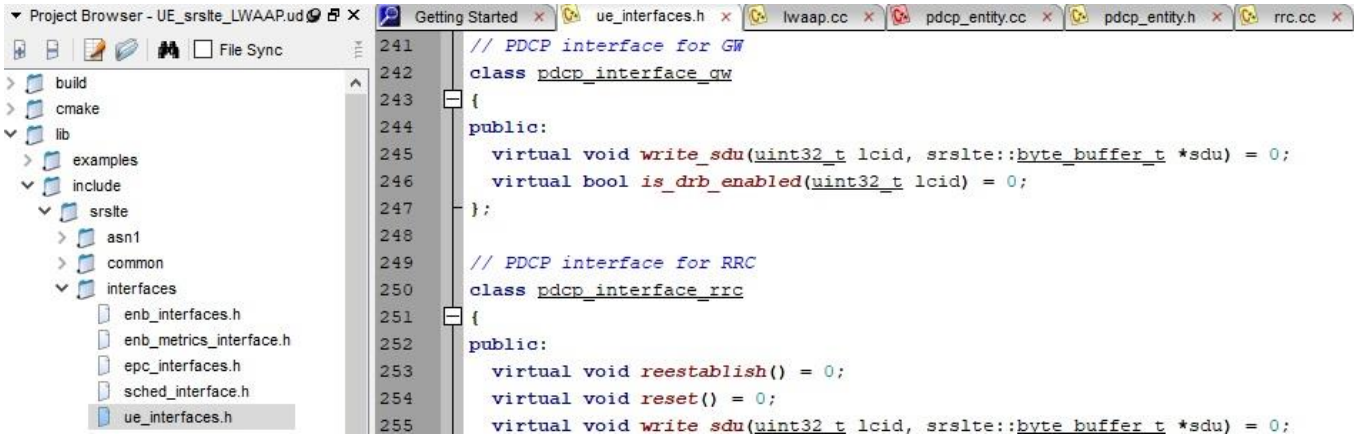
eNB封包流程

```
UE -> eNB -> EPC                          //eNB收到從UE收到封包，rlc層收到封包
srsenb::pdcp::write_pdu()                 //rlc呼叫pdcp的界面把封包送到pdcp層
|-> srslte::pdcp::write_pdu()             //enb的pdcp界面呼叫srslte的pdcp界面
    |-> srslte::pdcp_entity::write_pdu()  // srslte的pdcp界面再呼叫運作程式
        |-> srsenb::gtpu::write_pdu()     //pdcp再呼叫gtpu界面

UE <- eNB <- EPC                          //eNB收到從EPC收到封包，gtpu層收到封包
srsenb::gtpu::run_thread(){ recv(); }     //gtpu層收到EPC送來的封包
|-> srsenb::pdcp::write_sdu()             //gtpu層呼叫pdcp的界面把封包送到pdcp層
    |-> srslte::pdcp::write_sdu()         //enb的pdcp界面呼叫srslte的pdcp界面
        |-> srslte::pdcp_entity::write_sdu()  //srslte的pdcp界面再呼叫運作程式
            |-> srsenb::rlc::write_sdu()     //pdcp再呼叫rlc界面
                |->…
```

ue_interface.h

namespace srsue
{
	class pdcp_interface_rrc{…};	// pdcp function for rrc
	class rlc_interface_pdcp{…};	// rlc function for pdcp
}
//UE所有的界面會寫在ue_interfaces.h



srsue\src\ue.cc

```
namespace srsue
{
        bool ue::init(all_args_t *args_)
        {
                pdcp_log.init("PDCP ", logger);
                pdcp_log.set_level(level(args->log.pdcp_level));
                pdcp_log.set_hex_limit(args->log.pdcp_hex_limit);
                pdcp.init(&rlc, &rrc, &gw, &pdcp_log, …);
        }
}
```

//srsue所有程式的界面初始化會在srsue\src\ue.cc開始

UE封包流程

UE -> eNB　　//UE把封包送到eNB
srsue::gw::run_thread()　　//UE取得封包
|-> srslte::pdcp::write_sdu()　　//gw層呼叫pdcp界面並把封包送到pdcp層
　　|-> srslte::pdcp_entity::write_sdu()
　　|->…

UE <- eNB　　//UE底層收到eNB的封包
|->...
　|-> srslte::pdcp::write_pdu()　//底層呼叫pdcp界面並把封包送到pdcp層
　　|-> srslte::pdcp_entity::write_pdu()//pdcp界面呼叫pdcp運作程式
　　|-> srsue::gw::write_pdu() { write(); }//pdcp呼叫gw層的界面

Summary

- Brief review of 3GPP 4G/5G system architecture
- Several open-source platforms are introduced
  - OpenLTE
  - OpenAirInterface
  - RECO
  - Free5GC
  - srsLTE (srsUE, srsENB, srsEPC)
- nukLWA/nukxDC
  - srsLTE of SRS as a basis for the developed 5G emulator
  - nukxDC supports DC emulation with Ethernet and WiFi