Networked-Signal Processing in OAI

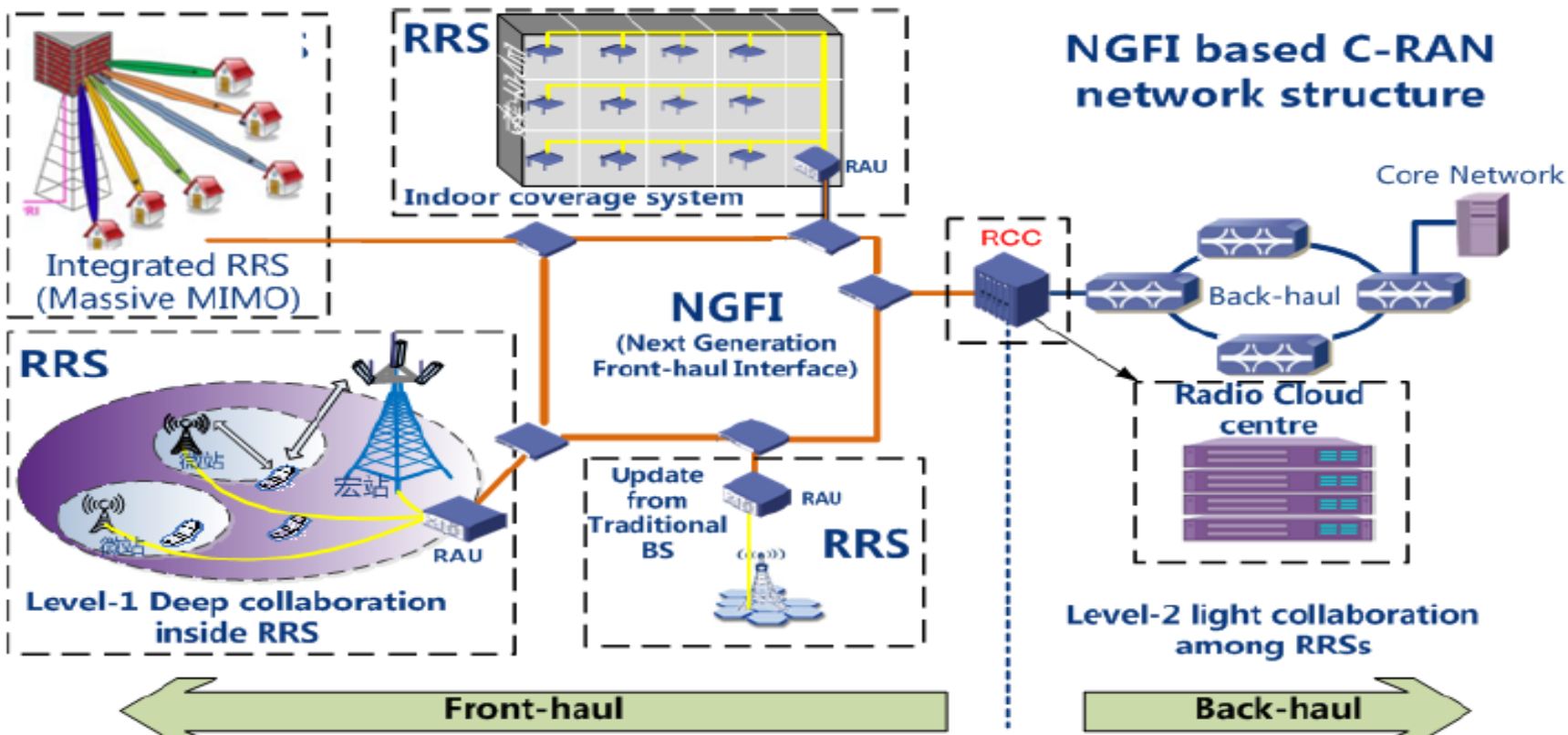
- Address the trend to *split/slice/distribute* the radio and protocol stack between network entities
  1. spatial signal-processing (distributed MIMO)
  2. network function virtualization
  3. data-center processing
- current implementation and what's coming
- some notes on EURECOM deployment
- some generic information from OAI community

Raymond Knopp
Open5G Lab, EURECOM
Sophia Antipolis, FRANCE

04.02.2017, FOSDEM
(Brussels)

NGFI Harmonization in OAI

- New descriptions for OAI RAN infrastructure Node Functions
  - `NGFI_RCC` : Radio Cloud Center
  - `NGFI_RAU` : Radio Aggregation Unit
  - `NGFI_RRU` : Remote Radio Unit
  - `3GPP_BBU` : Baseband Unit
  - `3GPP_eNodeB` : Complete eNodeB



Source: China Mobile - `http://www.windriver.com.cn/windforum/download/wf2015_networking03.pdf`

NGFI split points

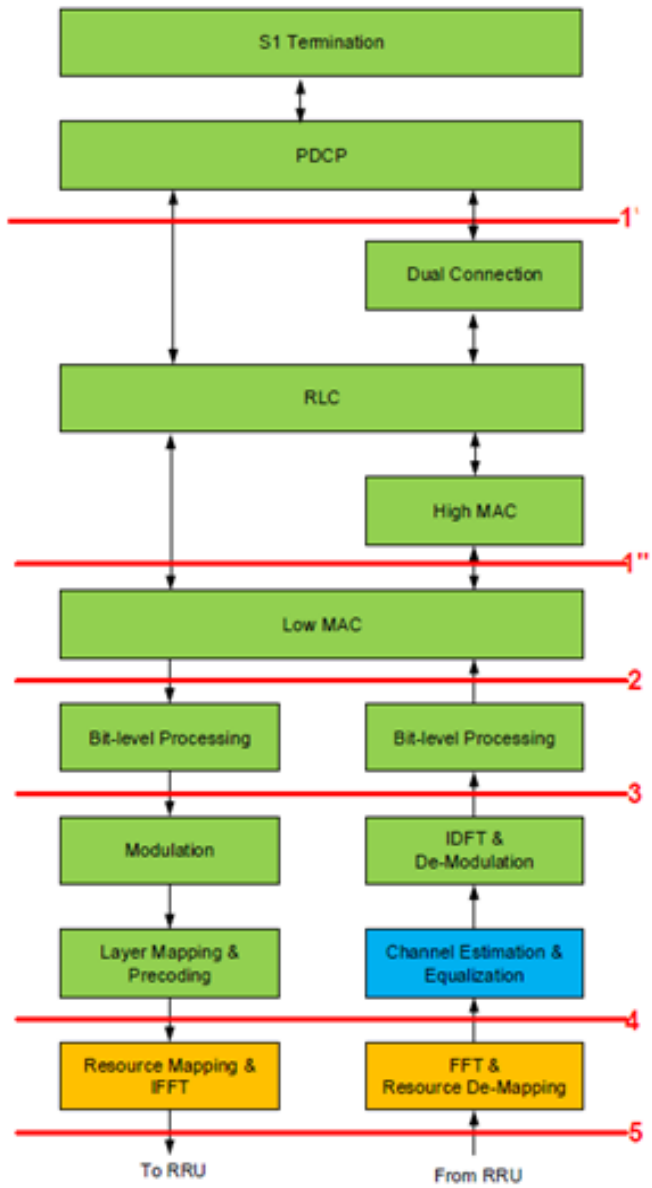
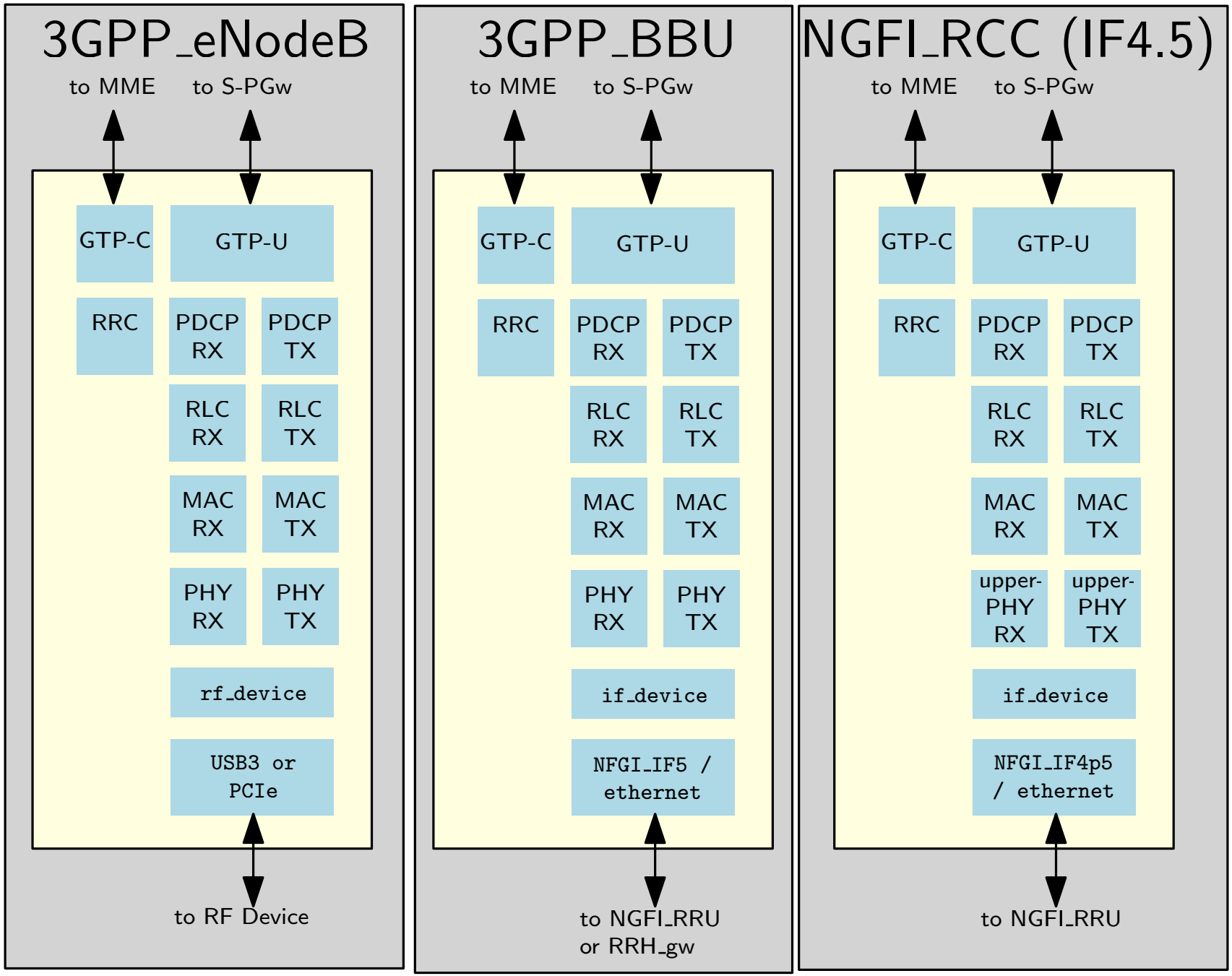


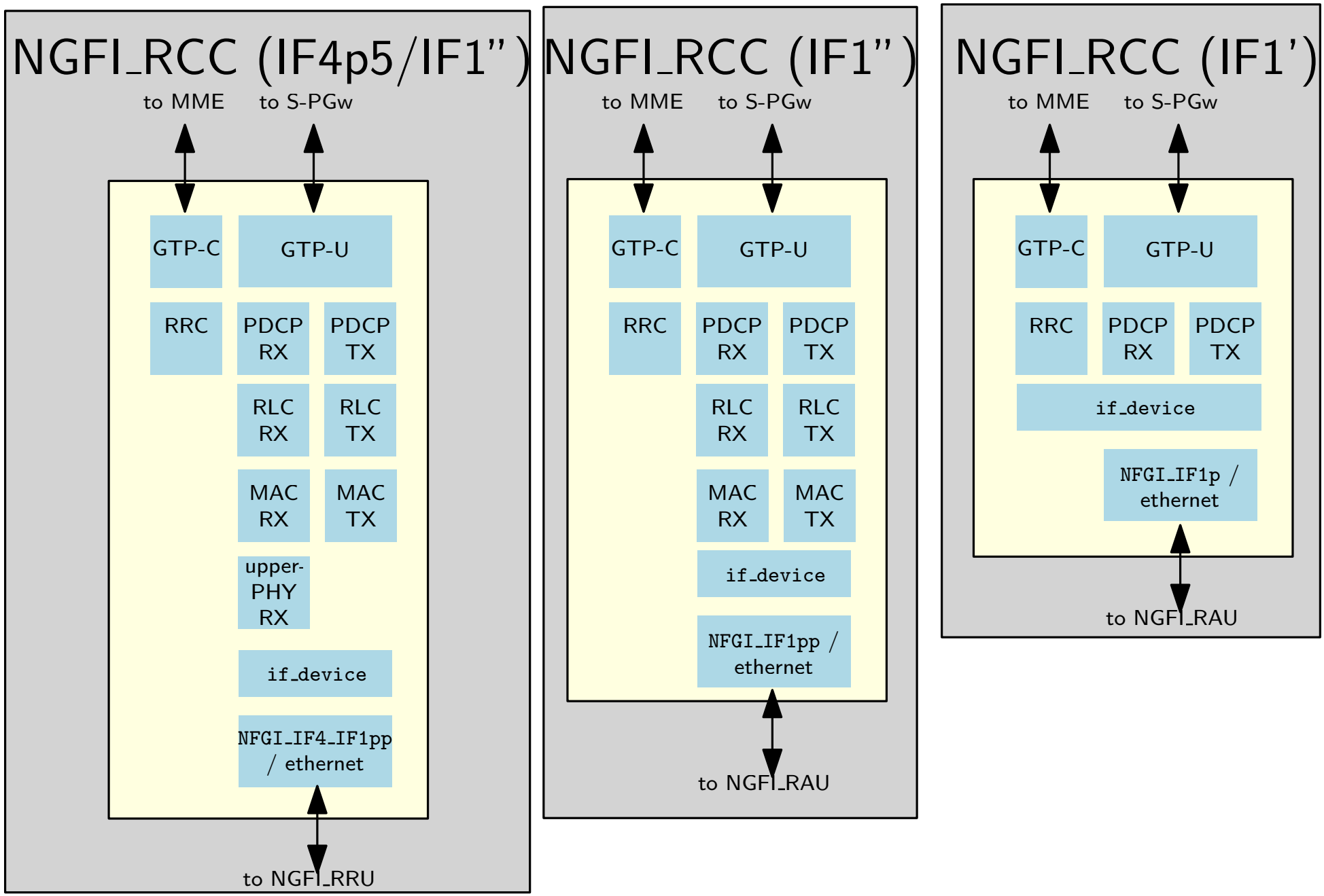
Figure 3-1: Division Plans for the RCC-RRS Interface

- Current OAI implementation (RRU/RCC) supports either
  - IF5 time-domain fronthaul (¿ 1 GbE required)
  - IF4.5 split (FFTs) (280 Mbit/s/antenna port fronthaul 20 MHz carrier) per carrier/sector
  - Soon IF2 (NFAPI) IF1 for super-PDCP soon

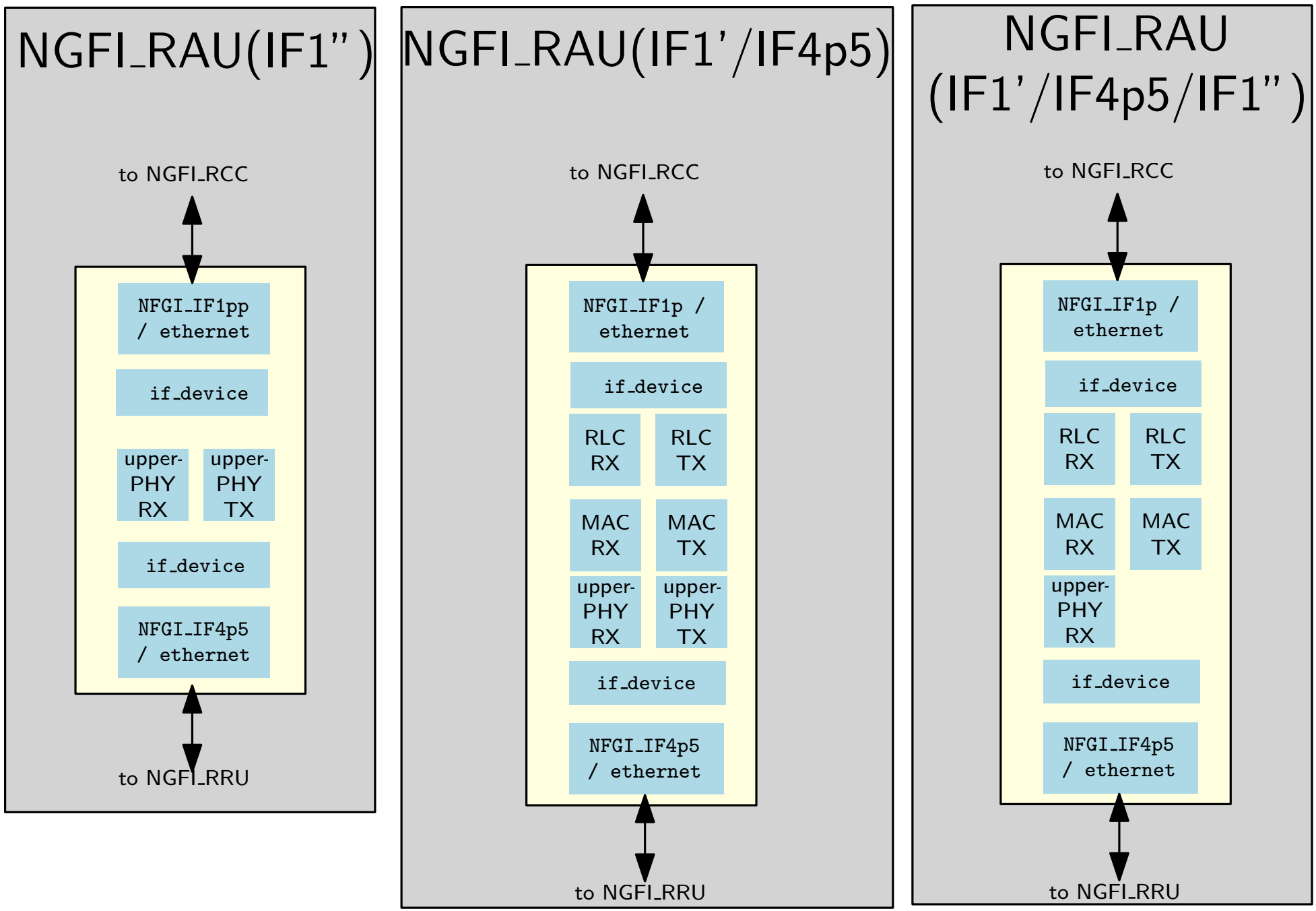
Functional Splits



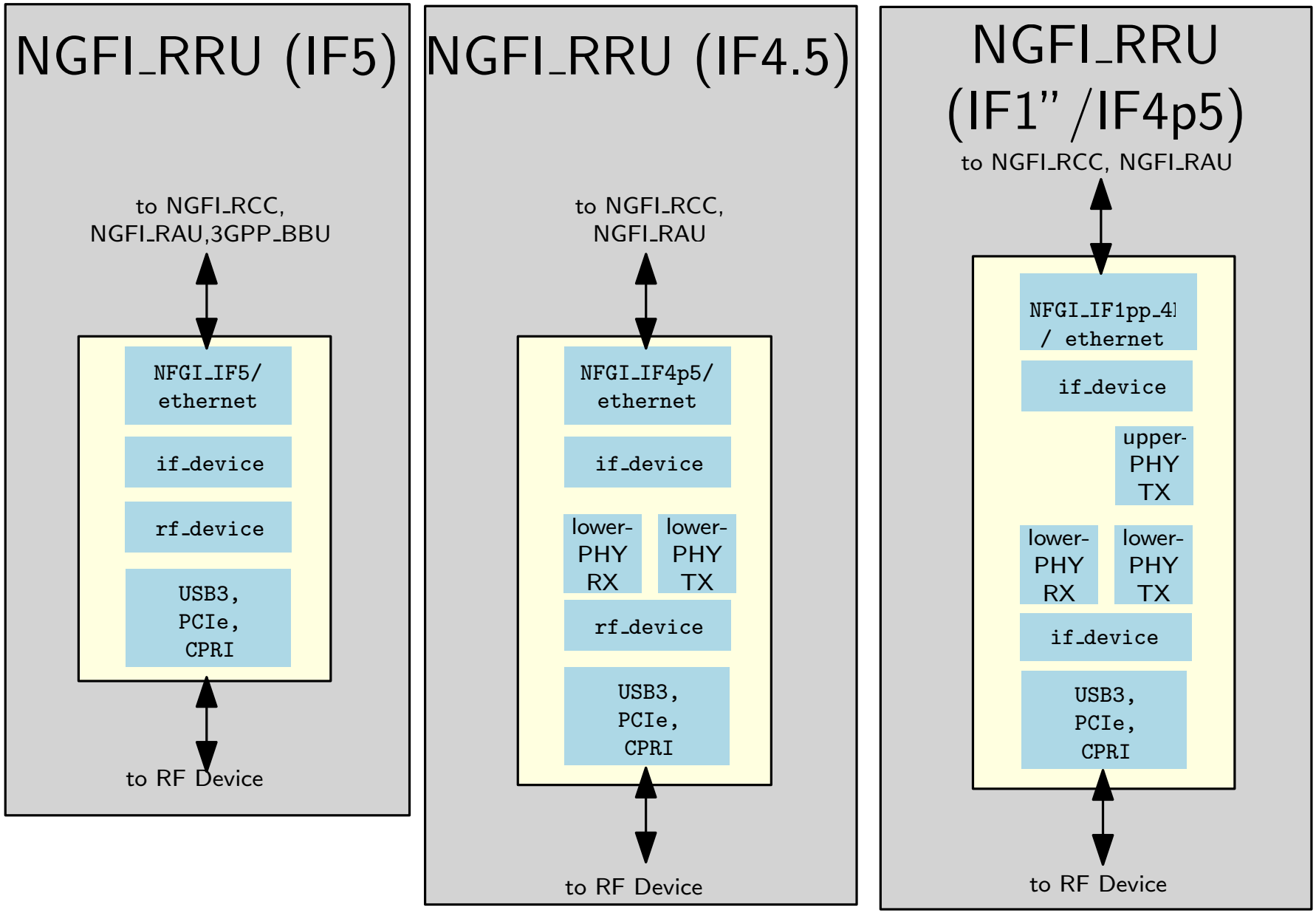
Functional Splits



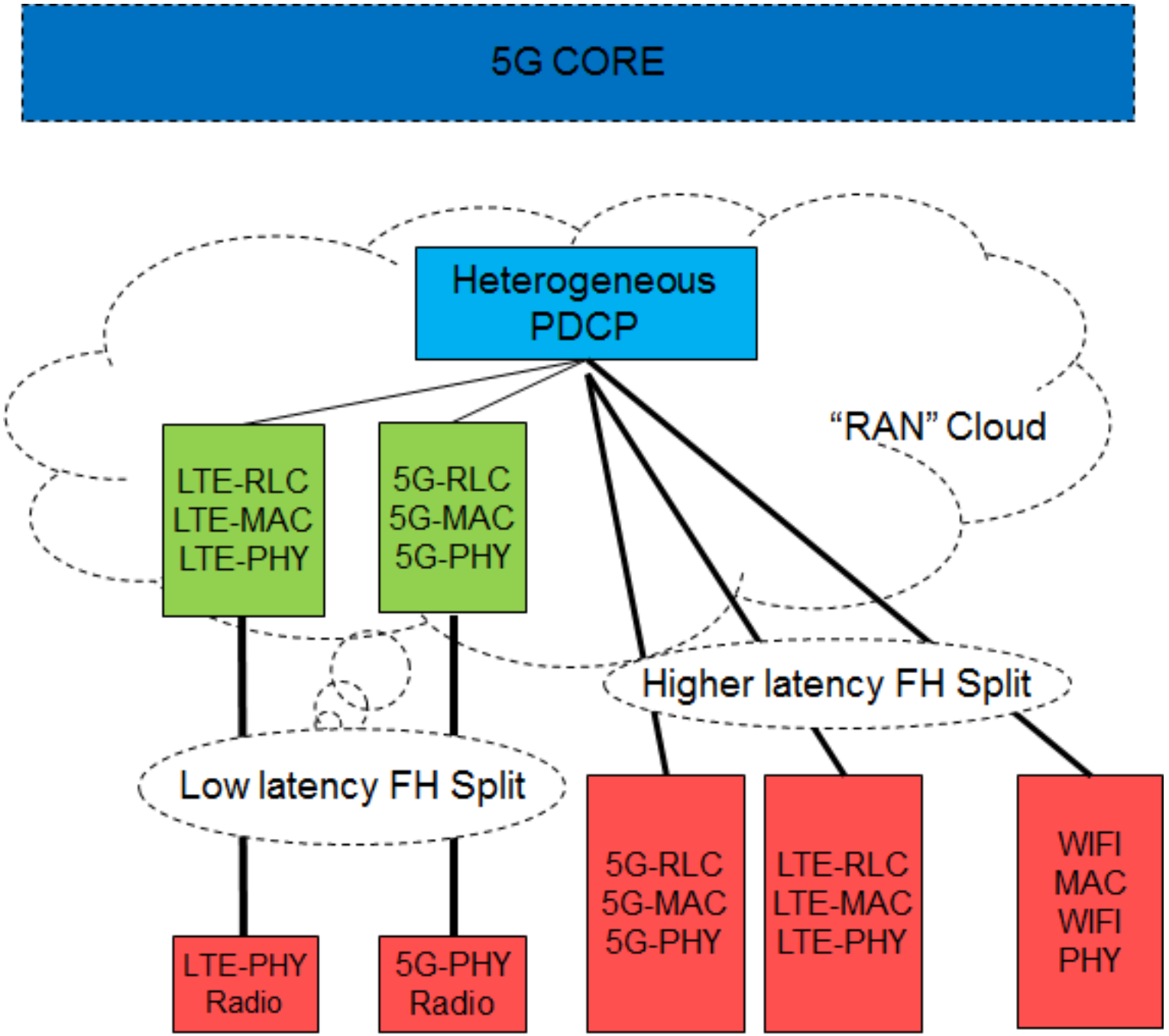
Functional Splits



Functional Splits

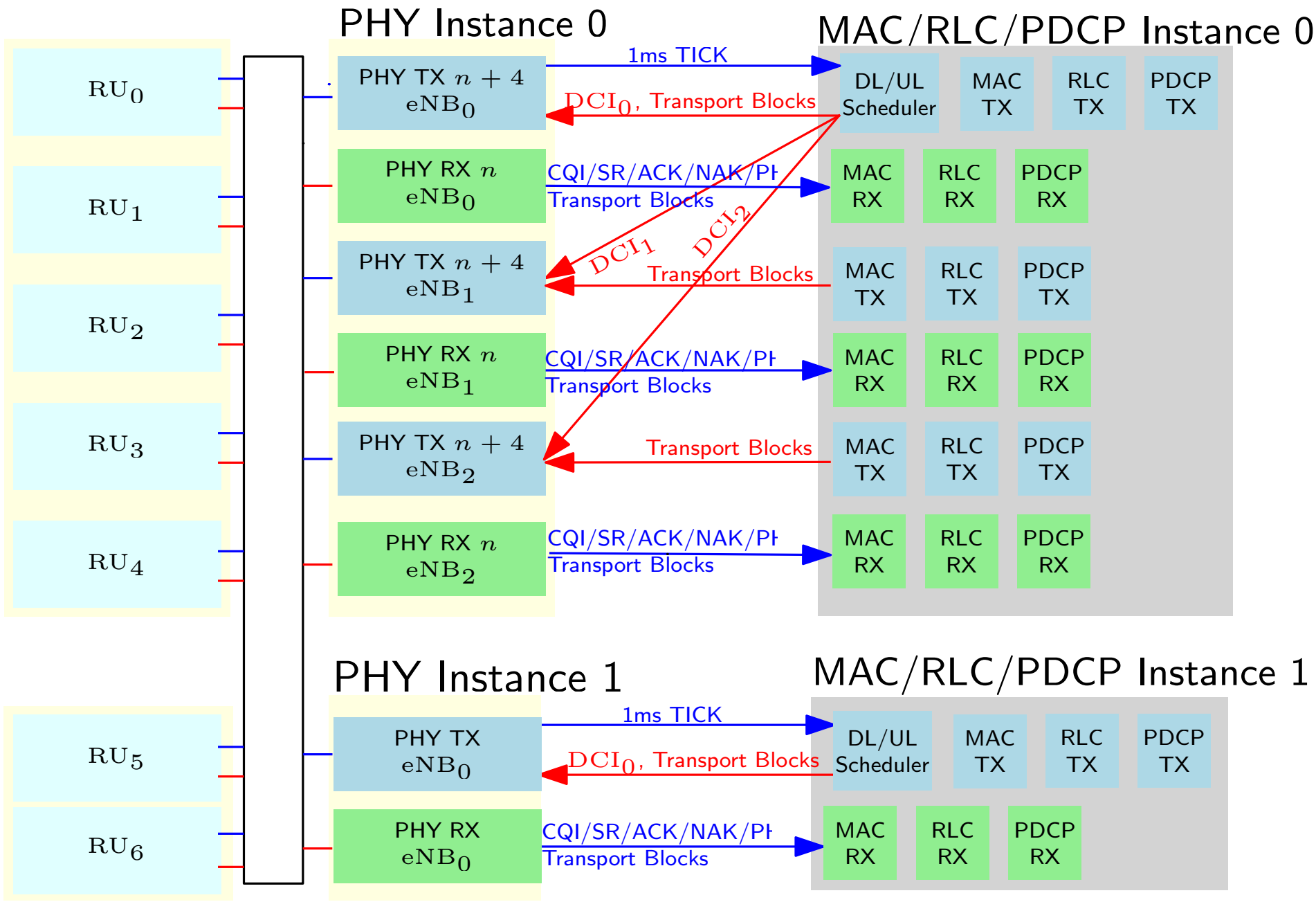


Some Notes on usage of splits



OAI RAN Software Architecture

RU/eNodeB Instances and Component Carriers



RU/eNodeB Instances and Component Carriers

- *Radio Unit* (RU) is
  - an entity managing a set of **physical** antennas. It can have a *local RF unit* or *remote RF unit*
  - performs precoding of multiple eNB TX streams and OFDM modulation (TX) and demodulation (RX) (part of 36.211)
- *eNB Instance* (indexed by `Mod_id`, or `enb_mod_id`) is a separate set of threads and contexts for a full eNodeB, or at least the MAC/RLC entities, in the same Linux process. There is one MAC/RLC entity associated to all component carriers of a single eNB instance.
- *eNB Component Carrier* (indexed by `CC_id`) is
  - a software entity managing the L1 procedures (36.213,36.212,36.211) and can act on
    - sectored antenna component
    - Rel10+ component carrier
    - virtual cell for DAS or Massive-MIMO array
  - each eNB is managed by one or two threads which operate on a subframe (TX and RX) and can have a *local RU* or *remote RU*
  - if a remote radio unit the eNB performs the 36.213 specifications only (HARQ, etc.) and connects to the remainder via the IF2 xhaul interface.

RU/eNodeB Instances and Component Carriers

- RU may have both an `if_device` for fronthaul and an `rf_device` for interconnection with a local RF unit
- if the `rf_device` is absent, it must have a southbound fronthaul interface (either IF5 or IF4p5) depending on the local processing of the remote RU
- if the `if_device` is absent, it must have a southbound RF interface and `rf_device`.
- three types of L1 processing are performed by the RU
  - subset of common L1 procedures from 36.211 specifications
  - fronthaul compression/decompression
  - framing
- on TX
  - A-law compression for (NGFI_RAU_IF4p5, NGFI_RAU_IF5)
  - A-law decompression (for NGFI_RRU_IF4p5 and NGFI_RRU_IF5)
  - OFDM modulation and cyclic prefix insertion (for NGFI_RRU_IF4p5,NGFI_RAU_IF5,3GPP_eNodeB_BBU,3GPP_eNodeB)
  - Precoding (for NGFI_RAU_IF5, NGFI_RAU_IF4p5,3GPP_eNodeB_BBU,3GPP_eNodeB)

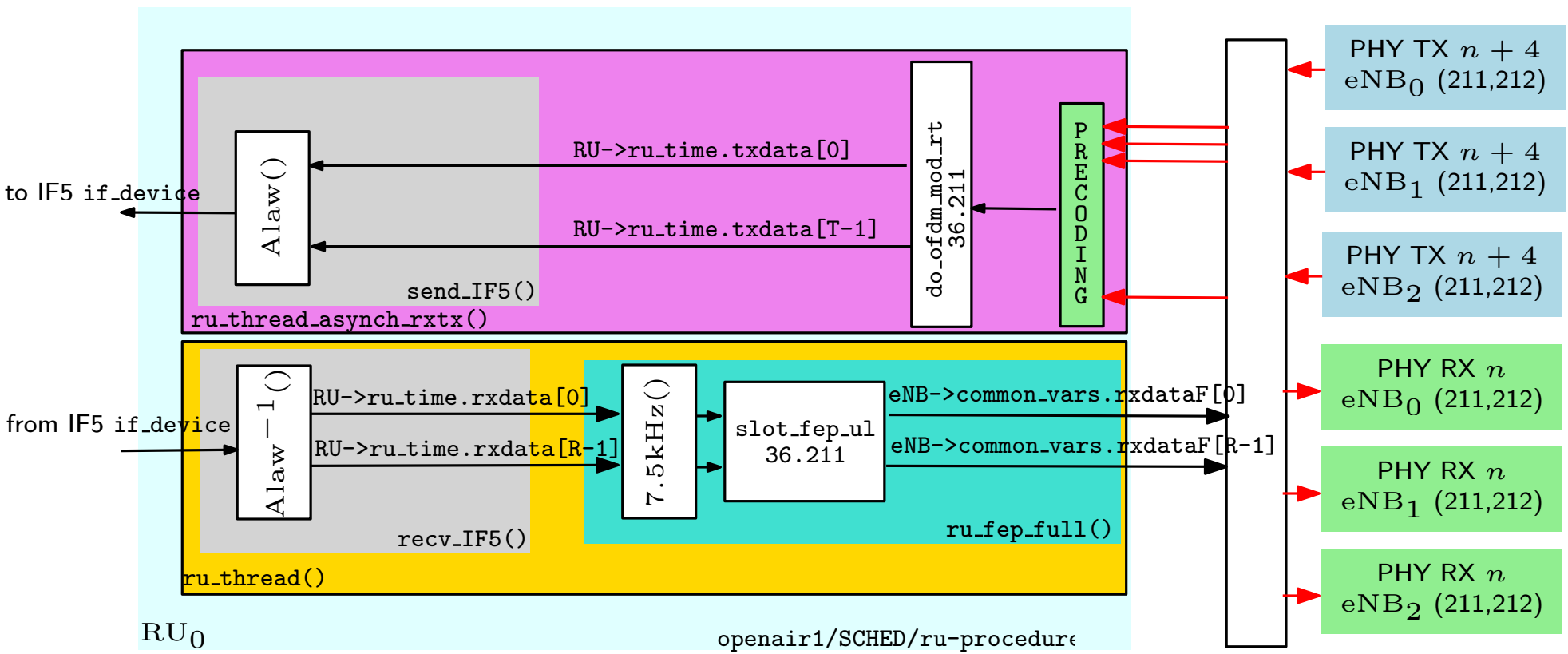
RU/eNodeB Instances and Component Carriers

- on RX
  - A-law compression for (NGFI_RRU_IF4p5, NGFI_RRU_IF5)
  - A-law decompression (for NGFI_RAU_IF4p5 and 3GPP_eNodeB_BBU)
  - cyclic prefix removal, frequency-shifting, OFDM demodulation, PRACH DFT (for NGFI_RRU_IF4p5, NGFI_RAU_IF5, 3GPP_eNodeB_BBU, 3GPP_eNodeB)

RU Fronthaul interfaces (NGFI_IF5)

- IF5 transports packets of size equal to a subframe and corresponding to a 1ms chunk of signal in the time-domain. This is done via the functions send_if5 and recv_if5, in the layer1 transport procedures (`openair1/PHY/LTE_TRANSPORT/if5_tools.c`). A timestamp is given along with the samples, corresponding to the time (in samples) of the first sample of the packet.
- each block can be compressed with A-law compression, yielding a compression rate of .5.

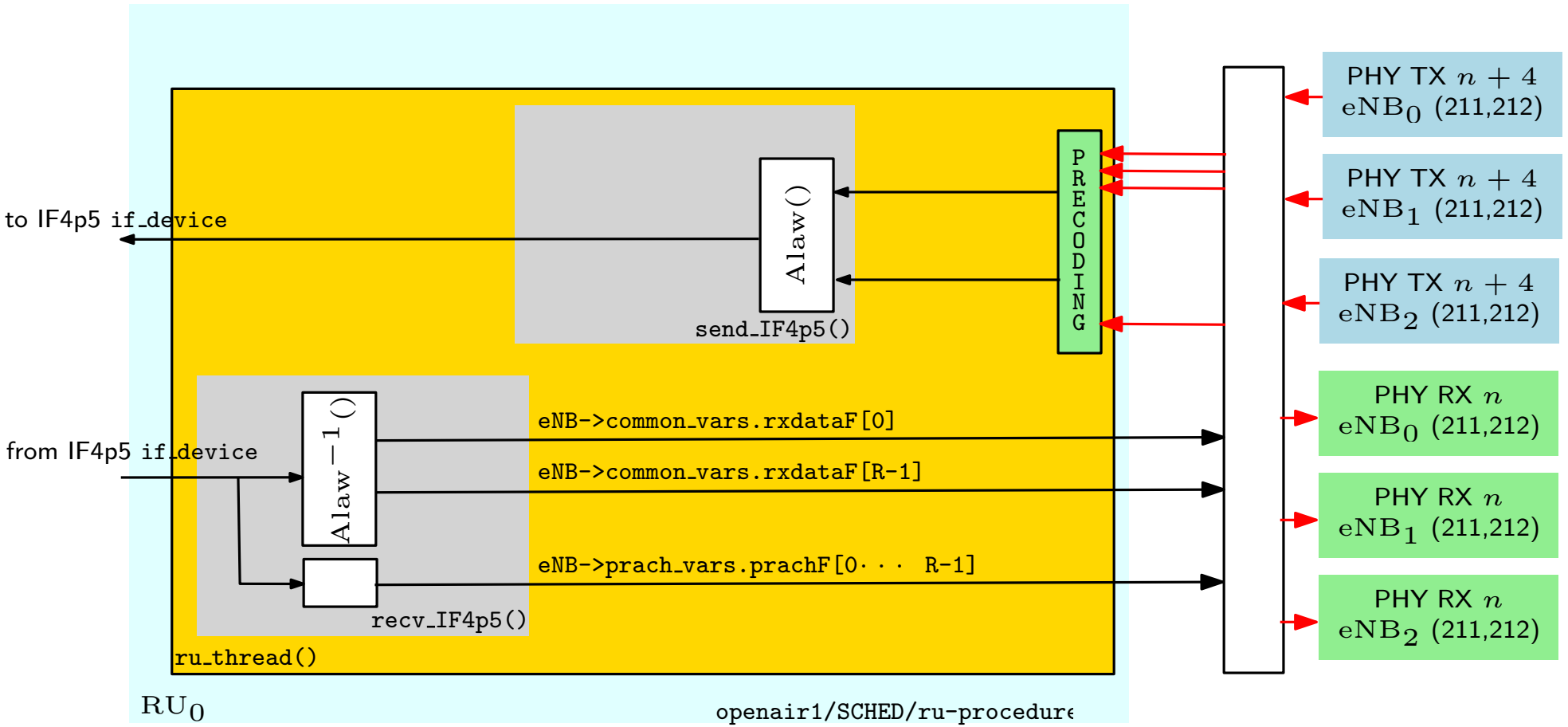
IF5 RU eNB end



RU Fronthaul interfaces (NGFI_IF4p5)

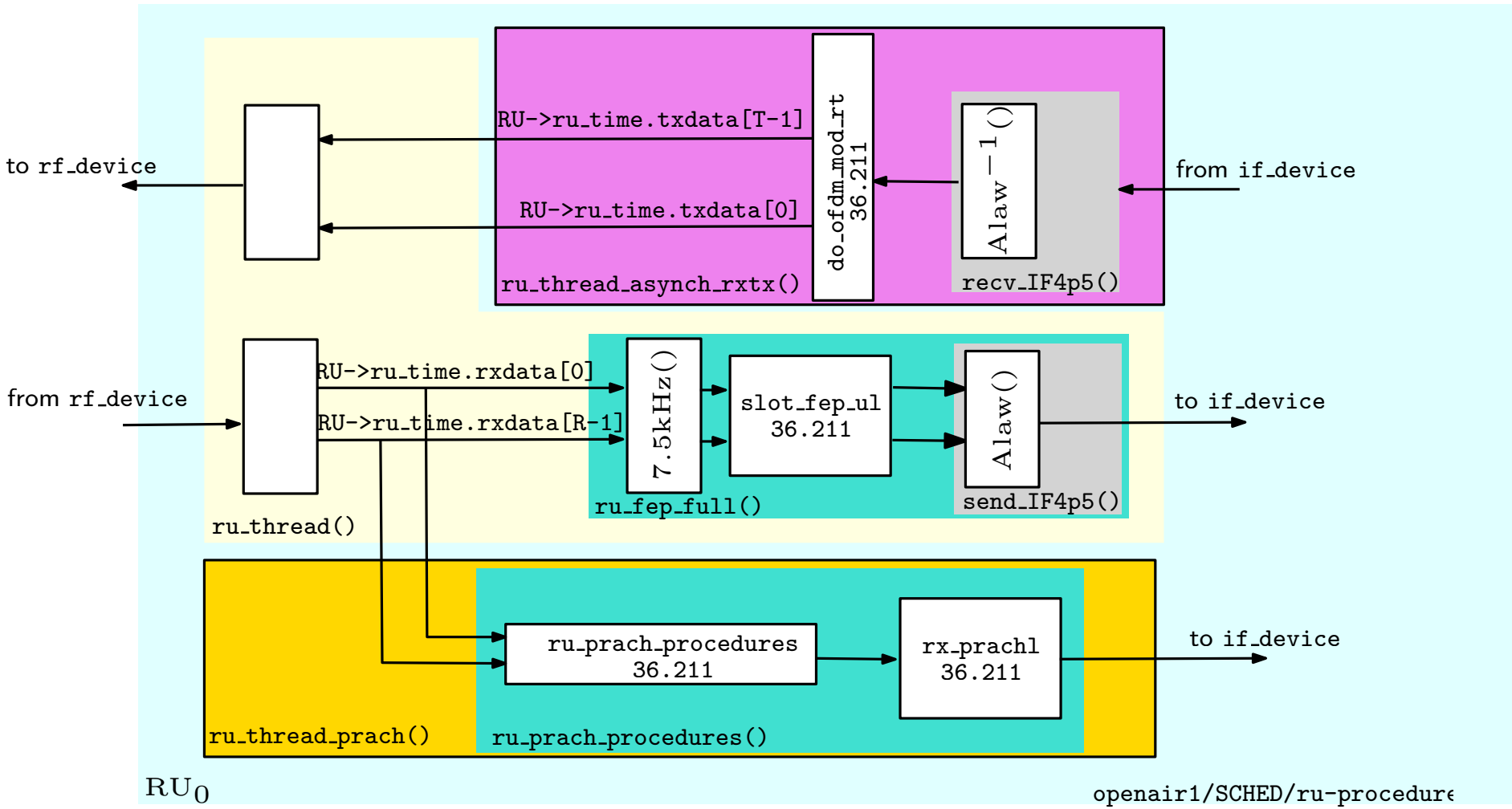
- IF4p5 transports packets of size equal to an OFDM symbol (for DLRE and ULRE) indexed by the symbol, subframe and frame number. This is done via the functions send_if4p5 and recv_if4p5, in the layer1 transport procedures (`openair1/PHY/LTE_TRANSPORT/if4_tools.c`).
- each block are compressed with A-law compression, yielding a compression rate of .5.

IF4p5 RU eNB end



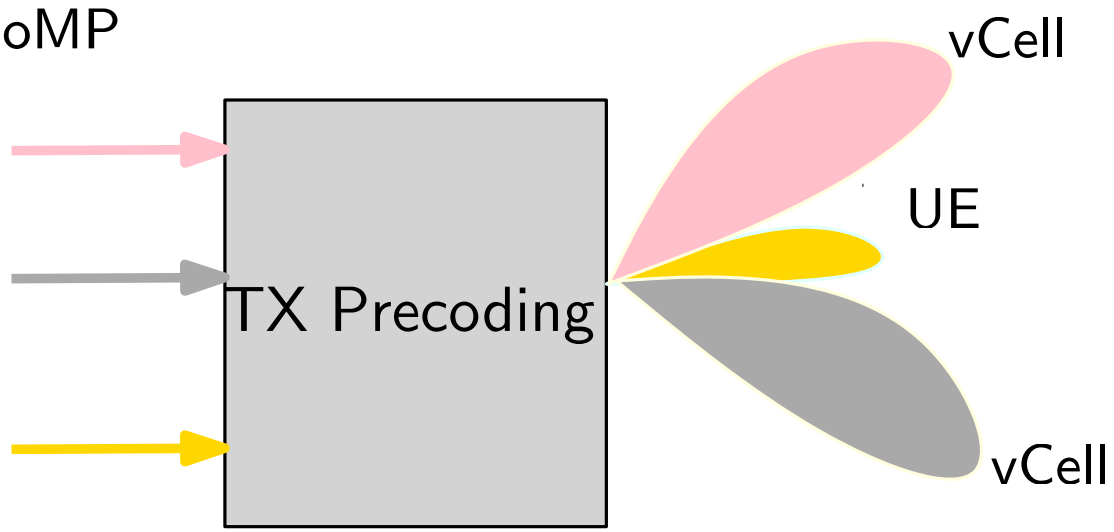
RU Fronthaul interfaces (NGFI_IF4p5)

IF4p5 RU remote-end

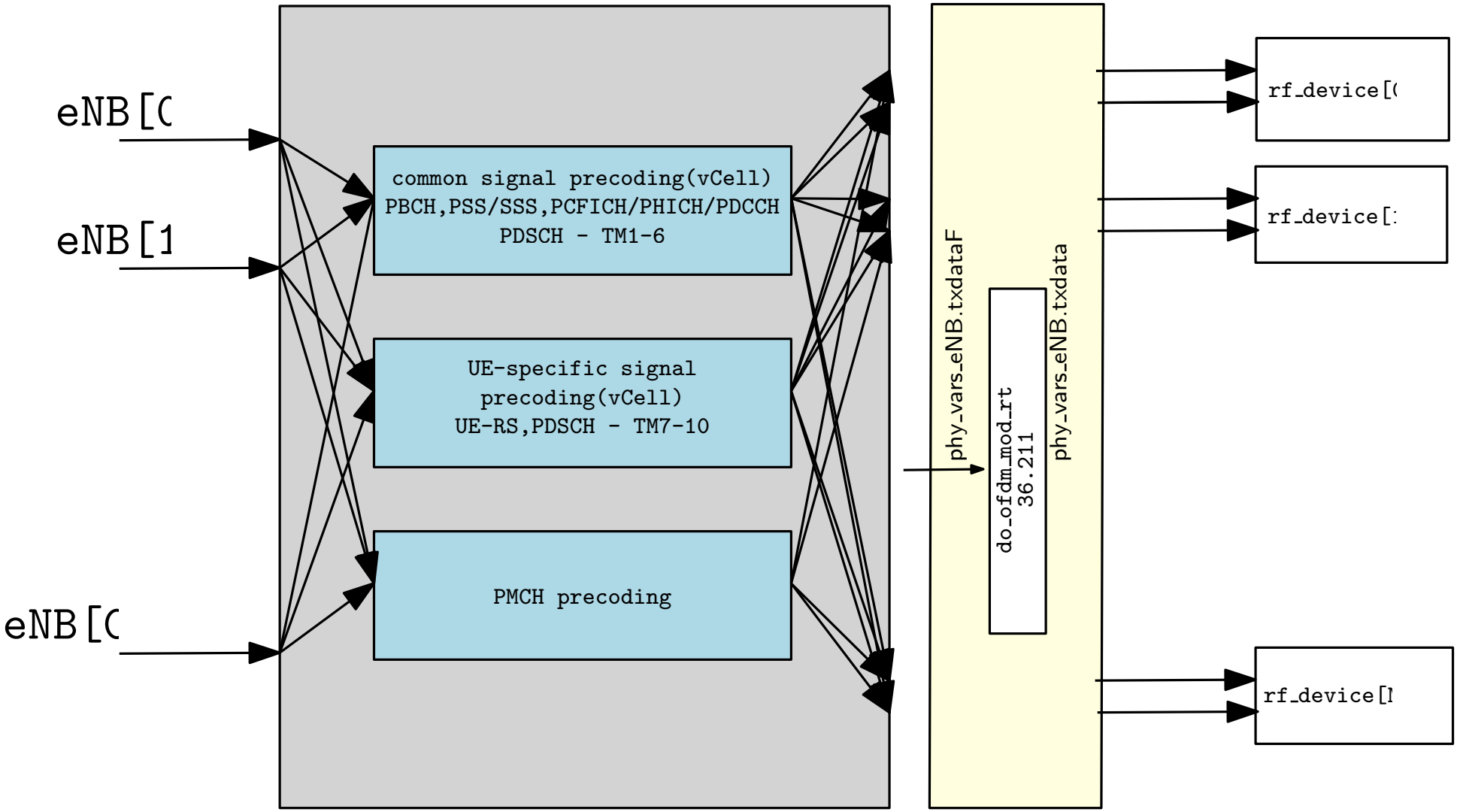


TX Precoding

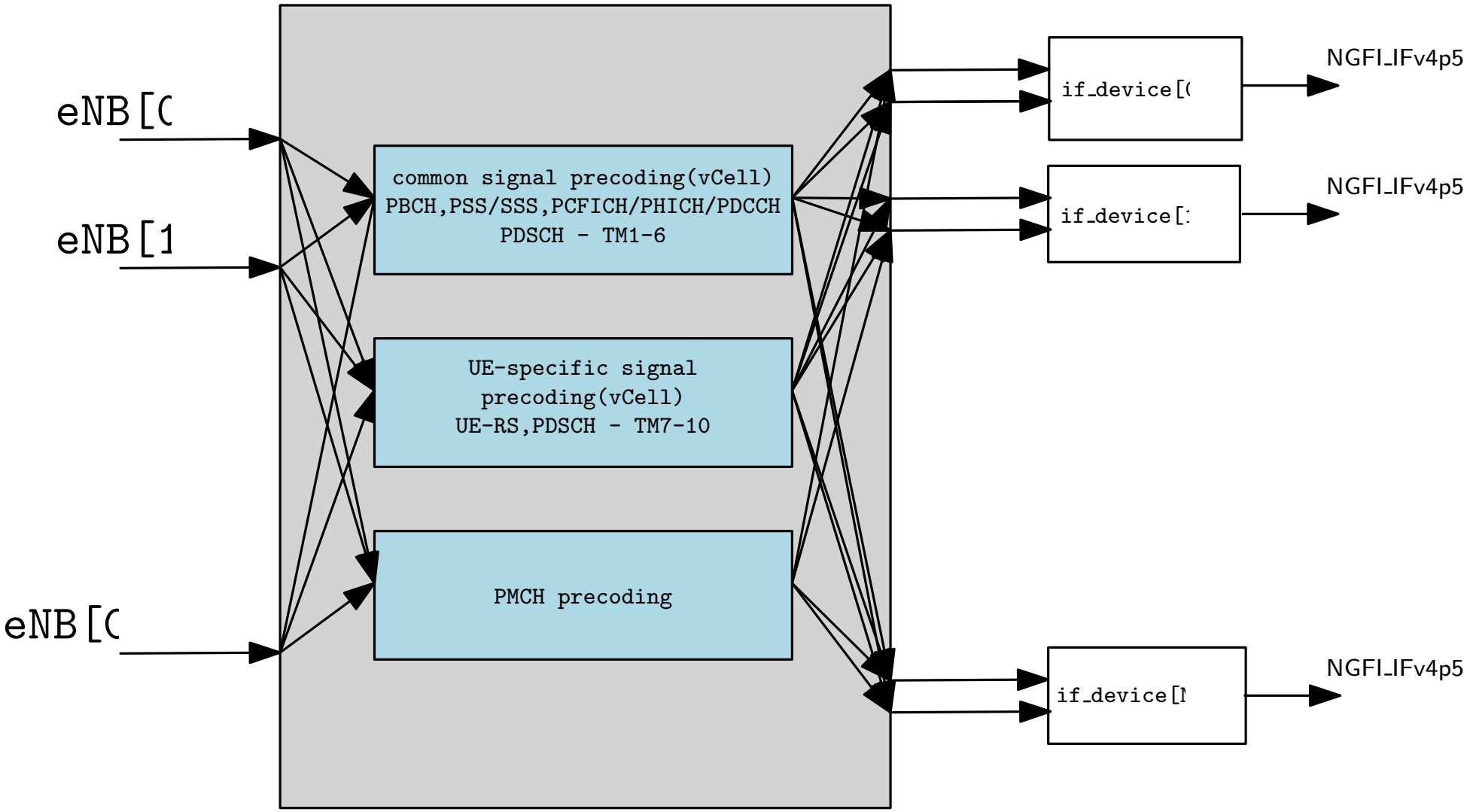
- Spatio-temporal filtering for muli-cell (vCell) and multi-user transmission. Input and output are frequency-domain signals.
- can be applied to Rel-10/11/12/13 physical channels and Rel-8 common channels
  - UE-specific precoding (TM7-10)
  - vCell-specific precoding (PDCCH + TM1-6) for groups of UEs
  - PMCH vCells
- Precoding applicable to
  1. indoor DAS
  2. outdoor co-localized arrays (e,g, Massive-MIMO)
  3. outdoor CoMP



TX Precoding (to RF device)

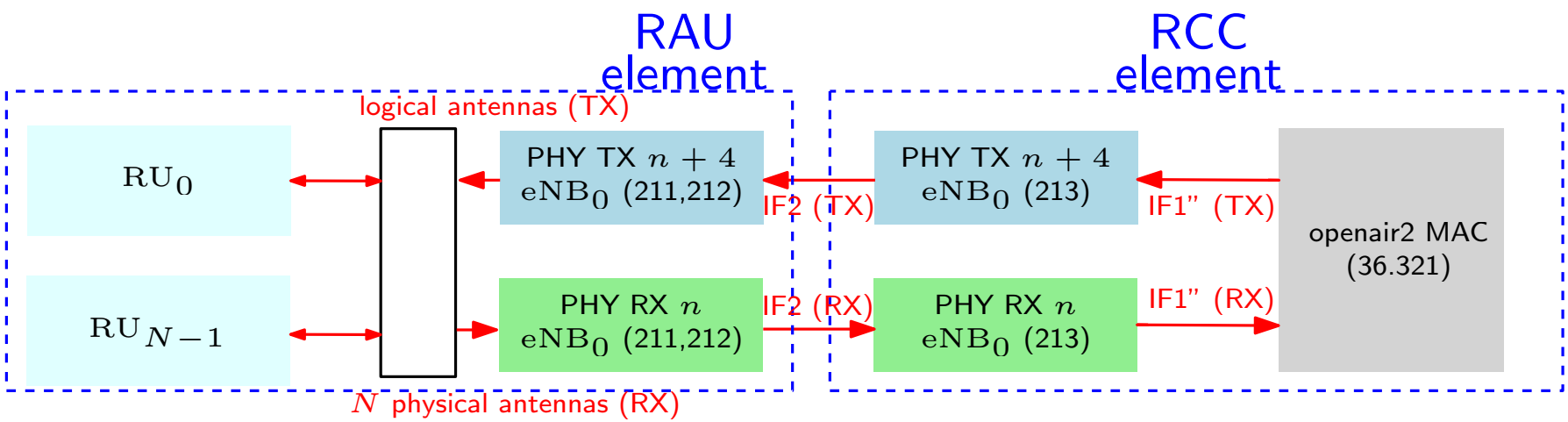


TX Precoding (to IF device, NGFI_IFv4p5)



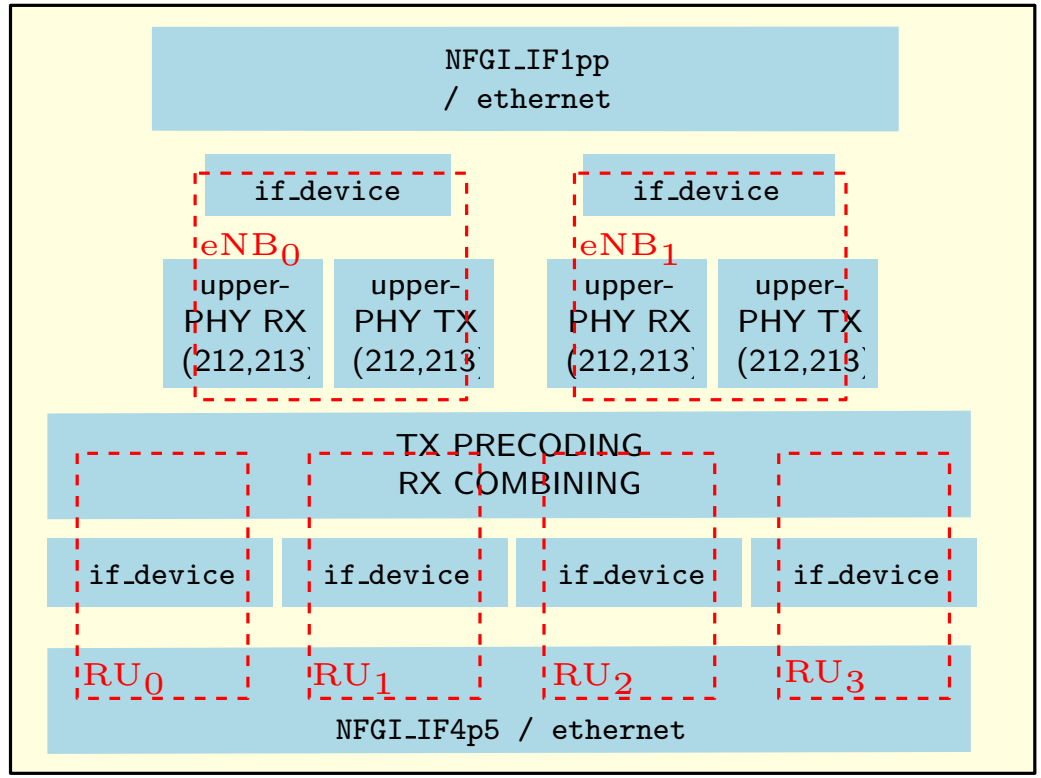
RU/eNodeB Instances and Component Carriers

- On TX path
  - eNB instances/component carriers operate on a set of logical antenna ports (0-3 for TM1-6, 4 for eMBMS, 5 for TM7, 6 for positioning, 7-8 for TM8, etc.)
  - each eNB has a list of RUs and the logical antenna ports are mapped to the physical antennas attached to the RUs via the precoding function
  - if the eNB has a remote RU then it is connected by an xhaul interface (IF2) to the eNB instance/component carrier in another unit which contains the RU. Here the eNB processing is split.
- on RX the eNB instance/component carrier has access to **all** RU signals for processing



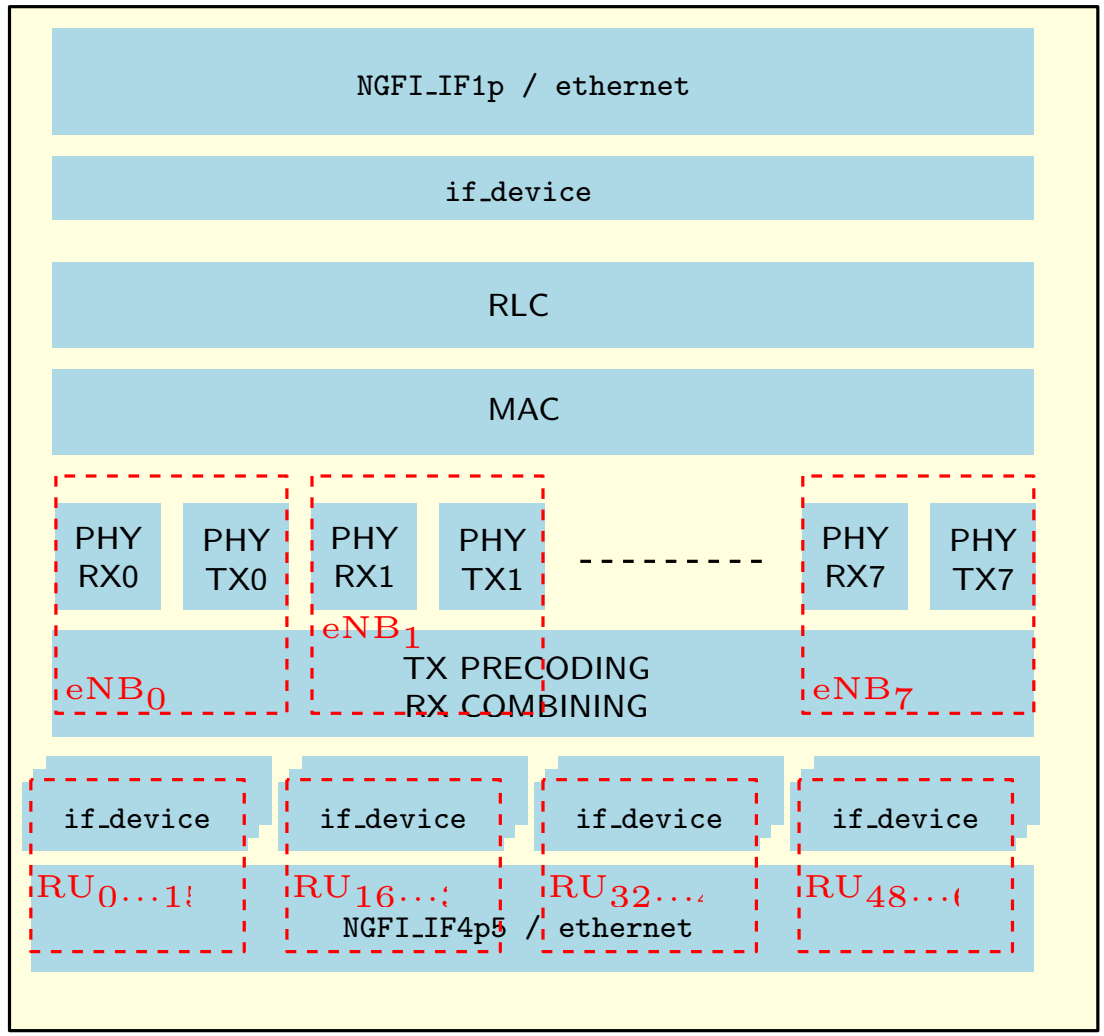
RAU Example (DAS)

- Example: RAU with `NGFI_IF1pp` xhaul (MAC/PHY split) northbound, `NGFI_IF4p5` fronthaul southbound, 2 vCell logical interfaces (2 L1/L2 instances, or 1 L2 instance and 2 CCs), 4 RRUs with `NGFI_IF4p5`



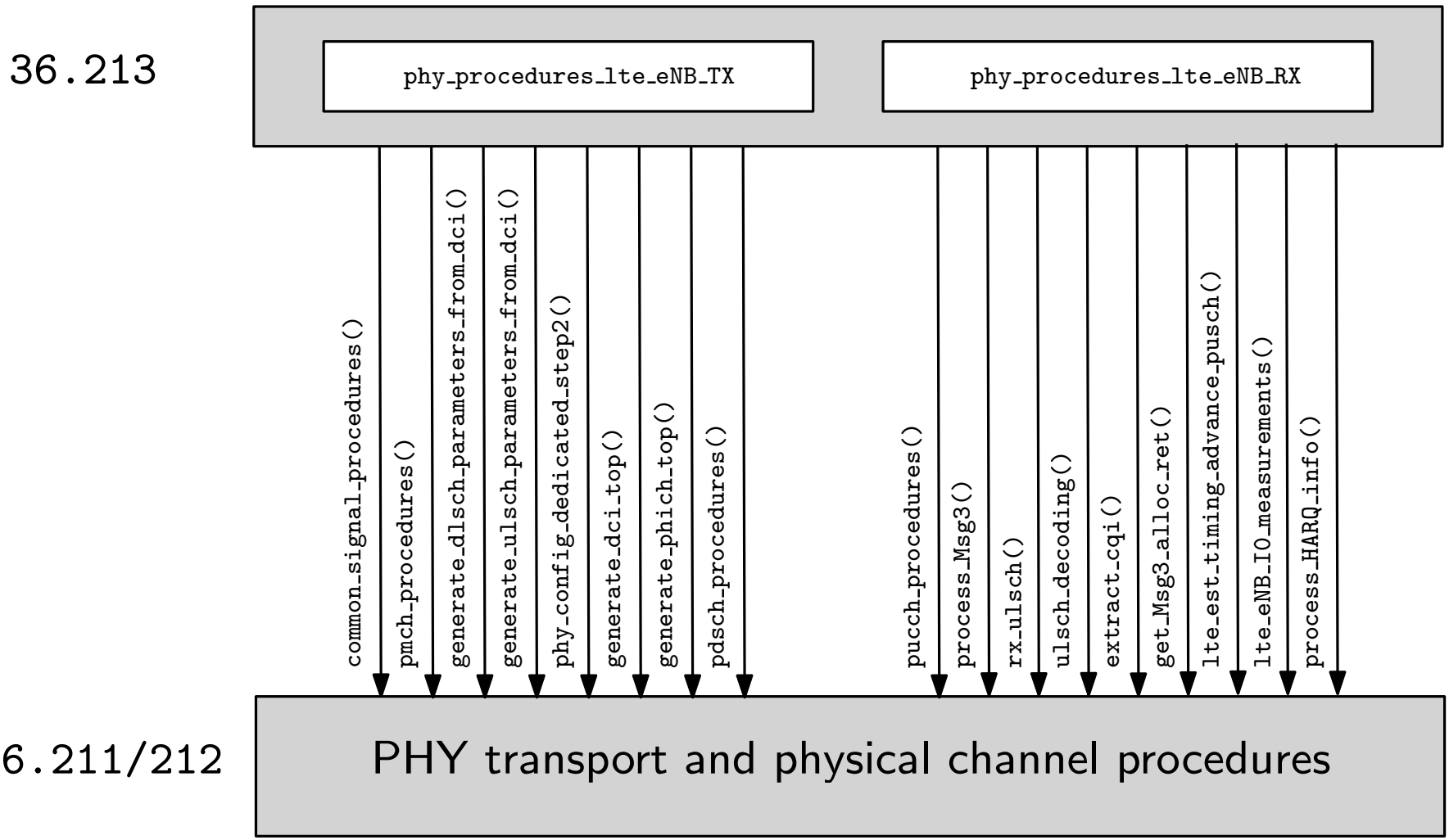
RAU Example (Massive-MIMO)

- Example: massive-MIMO RAU with `NGFI_IF1p` fronthaul northbound, 8 L1 component carriers,1 L2 instances, many local RRUs with `NGFI_IF4p5` southbound

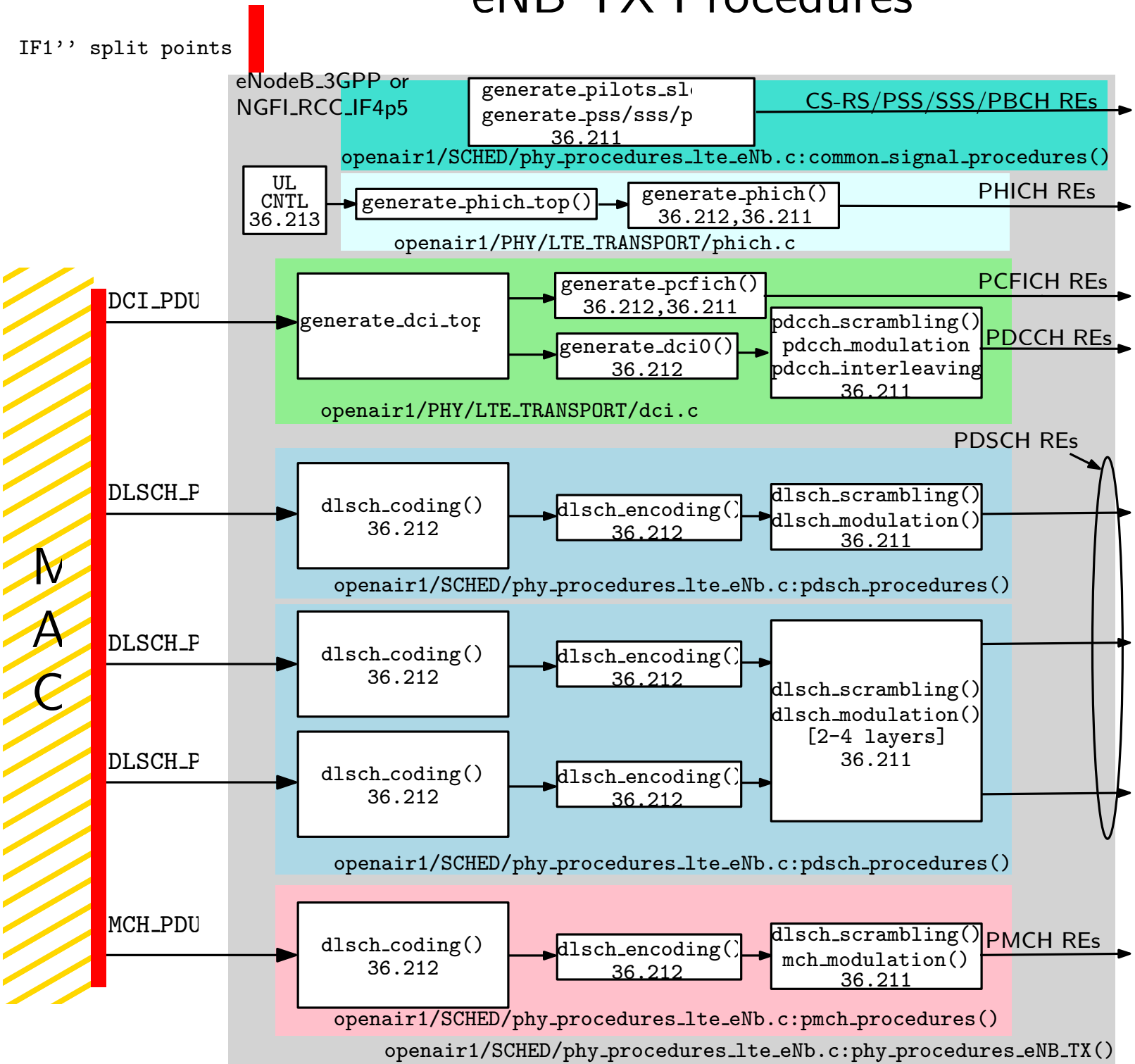


OAI IF2 Interface

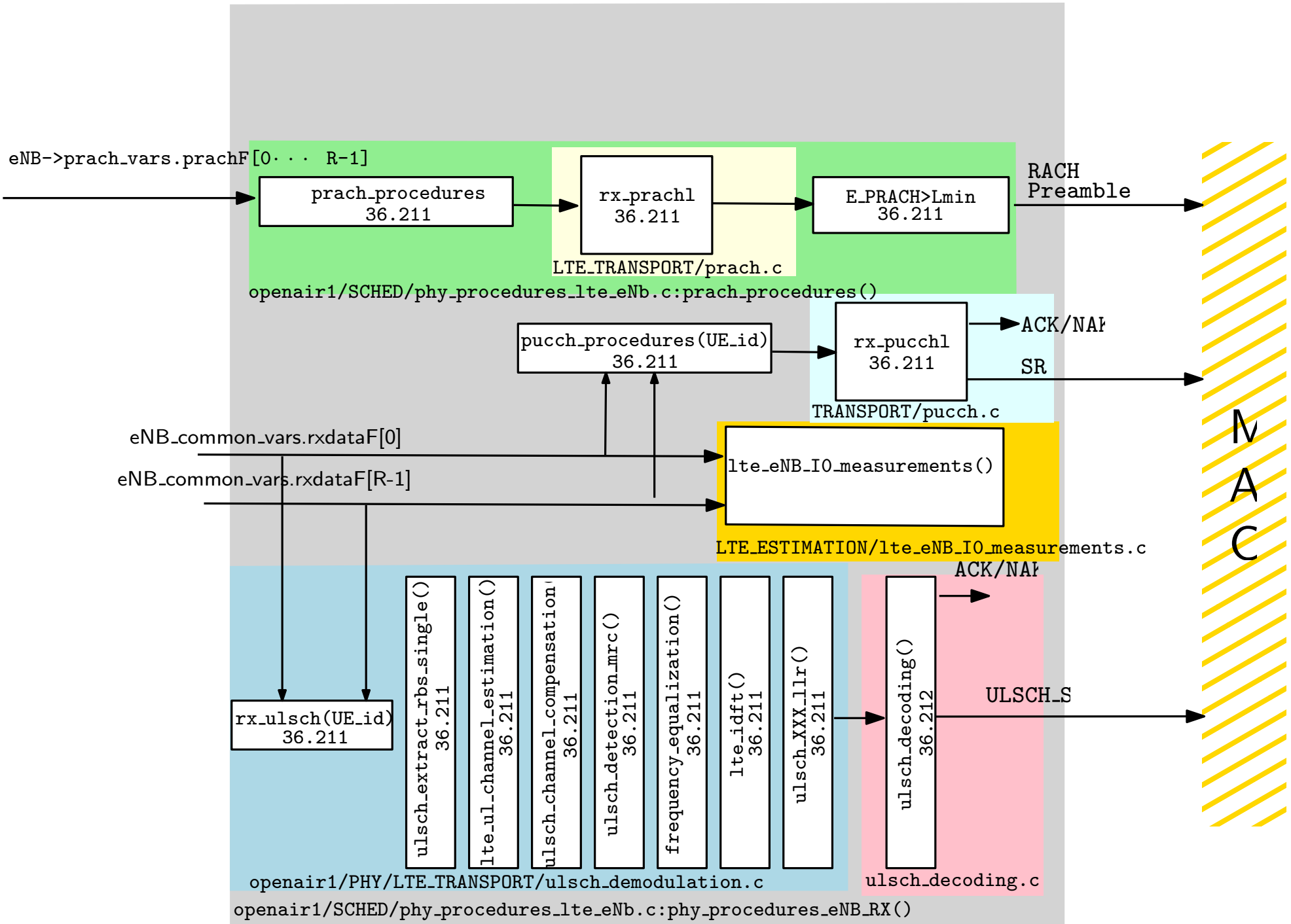
- OAI IF2 is the interface between the 36.213 Physical Layer Procedures (HARQ, SR, CSI, etc.) and the transport/physical channel processing
- it can be networked, although this is not used as an xhaul interface at the moment. Considerations for using FAPI/NFAPI P7 are underway.



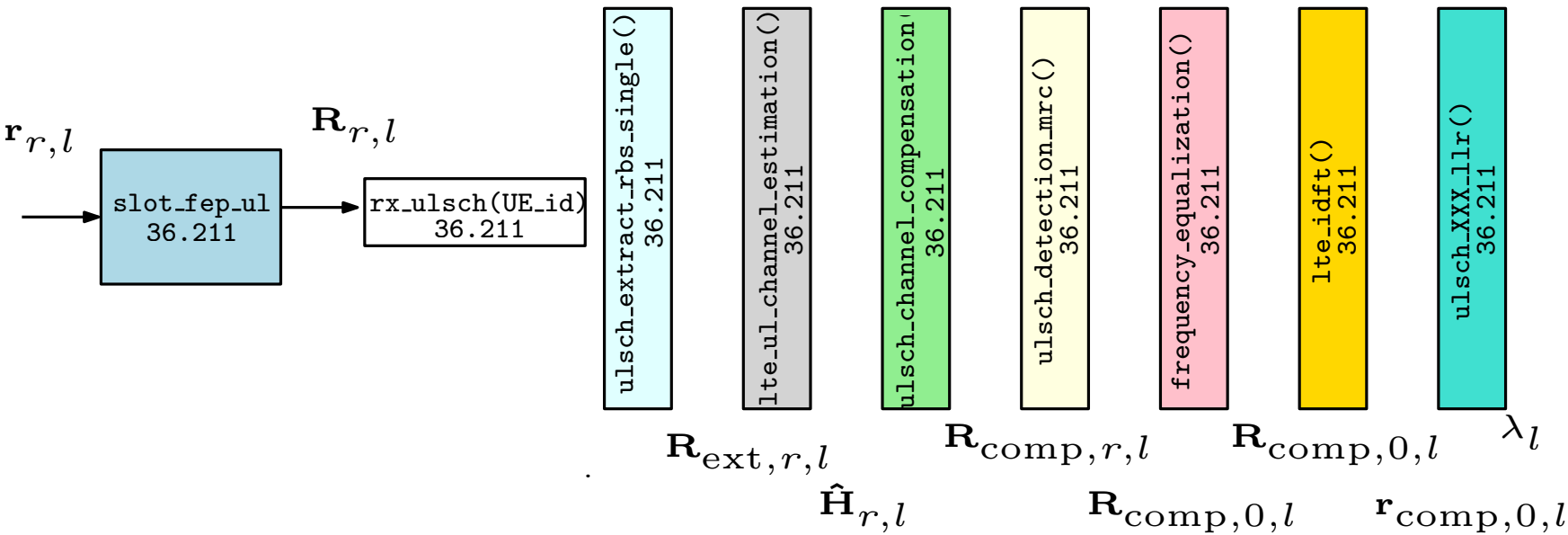
eNB TX Procedures



eNB PHY RX Procedures



eNB ULSCH Demodulation



$\mathbf{R}_{r,l} = \mathrm{DFT}_{N_{\mathrm{fft}}}(\mathbf{r}_{r,l} \odot \mathbf{F}_{7.5}), r = 0, 1, \cdots, R-1, l = 0, 1, \cdots, N_{\mathrm{symb}} - 1$ (**eNB_common_vars→rxdataF[][]**)

$R_{\mathrm{ext},r,l}(n) = R_{r,l}(12\mathrm{firstPRB} + n), n = 0, 1, \cdots, 12N_{\mathrm{PRB}} - 1$ (**eNB_pusch_vars→ulsch_rxdataF_ext[][]**)

$\hat{\mathbf{H}}_{r,l} = \mathbf{R}_{\mathrm{ext},r,l} \odot \mathbf{DRS}_l^*(\mathrm{cyclicShift}, n_{\mathrm{DMRS}(2)}, n_{\mathrm{PRS}})$, (**eNB_pusch_vars→drs_ch_estimates[]**)

$\mathbf{R}_{\mathrm{comp},r,l} = \hat{\mathbf{H}}_r \odot \mathbf{R}_{\mathrm{ext},r,l} 2^{-\log_2 |H_{\max}|}, \hat{\mathbf{H}}_r = \frac{1}{2}(\hat{\mathbf{H}}_{r,3} + \hat{\mathbf{H}}_{r,10})$
(**eNB_pusch_vars→ulsch_rxdataF_comp**)

$\mathbf{R}_{\mathrm{comp},0,l} = \frac{1}{R}\sum_{r=0}^{R-1} \mathbf{R}_{\mathrm{comp},r,l}$

$R_{\mathrm{comp},0,l}(n) = R_{\mathrm{comp},0,l}(n)\dot{Q}_8\left(\frac{1}{|\hat{H}(n)|^2 + I_0}\right), \hat{H}(n) = \sum_{r=0}^{R-1} \hat{H}_r(n)$

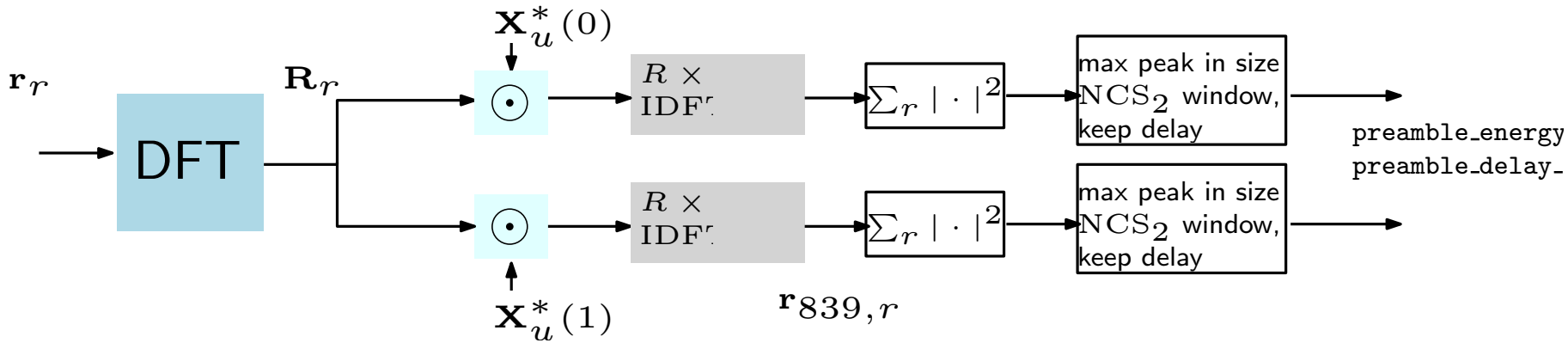
$\mathbf{r}_{\mathrm{comp},0,l} = \mathrm{IDFT}_{12N_{\mathrm{PRB}}}(\mathbf{R}_{\mathrm{comp},0,l})$

QPSK : $\lambda_l(2n) = \mathrm{Re}(r_{\mathrm{comp},0,l}(n)), \lambda_l(2n+1) = \mathrm{Im}(r_{\mathrm{comp},0,l}(n))$ (**eNB_pusch_vars→ulsch_llr**)

16QAM : $\lambda_l(4n) = \mathrm{Re}(r_{\mathrm{comp},0,l}(n)), \lambda_l(4n+2) = \mathrm{Im}(r_{\mathrm{comp},0,l}(n))$

$\lambda_l(4n+1) = |\mathrm{Re}(r_{\mathrm{comp},0,l}(n))| - 2\overline{|h(n)|}, \lambda_l(4n+3) = |\mathrm{Im}(r_{\mathrm{comp},0,l}(n))| - 2\overline{|h(n)|}$

eNB PRACH Detection



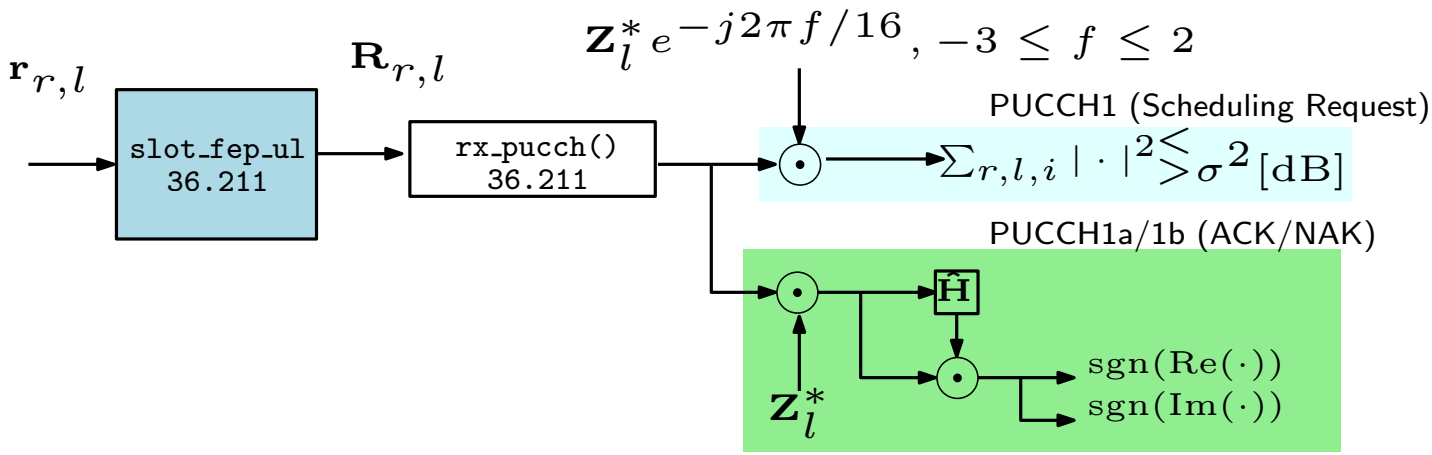
$\mathbf{R}_r = \mathrm{DFT}_{N_{\mathrm{PRACH}}}(\mathbf{r}_r), r = 0, 1, \cdots, R-1$ (lte_eNB_prach_vars→rxsigF[])

$\mathbf{R}_{\mathrm{comp},r} = \mathbf{R}_r \odot \mathbf{X}_u^*[i], r = 0, 1, \cdots, R-1$ (lte_eNB_prach_vars→prachF[])

$\mathbf{r}_{839,r} = \mathrm{IDFT}_{1024}(\mathbf{R}_{\mathrm{comp},r}), r = 0, 1, \cdots, R-1$ (lte_eNB_prach_vars→prach_ifft[])

- PRACH detection is a quasi-optimal non-coherent receiver for vector observations (multiple antennas)
- correlation is done in the frequency-domain, number of correlations (in the example above 2) depends on *zeroCorrelationConfig* configuration parameter
- peak-detection (for delay estimation) is performed in each NCS time-window

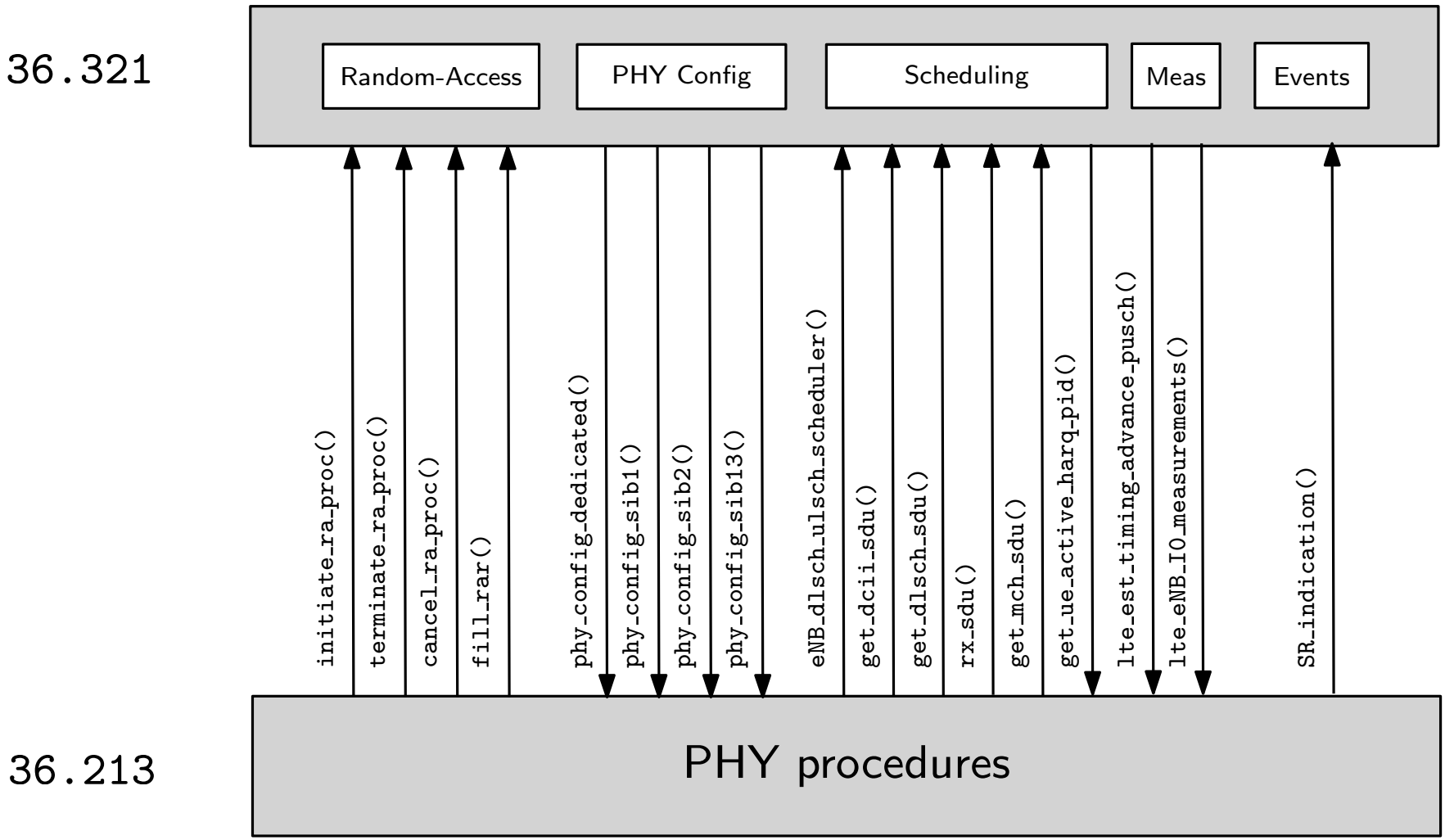
eNB PUCCH Detection



- PUCCH1 detection is a quasi-optimal non-coherent receiver (energy detector) for vector observations (multiple antennas) for scheduling request. Care is taken to handle residual frequency-offset.
- PUCCH1A/1B detection is quasi-coherent based on a rough channel estimate obtained on the 3 symbols without data modulation.
- In both cases, correlation is done in the frequency-domain

OAI IF1" Interface

- OAI IF1" is the interface between the 36.321 Medium-Access (MAC) Layer Procedures and the 36.213 Physical Layer Procedures
- it cannot be networked today, Considerations for allowing an xhaul networking of this interface are underway.



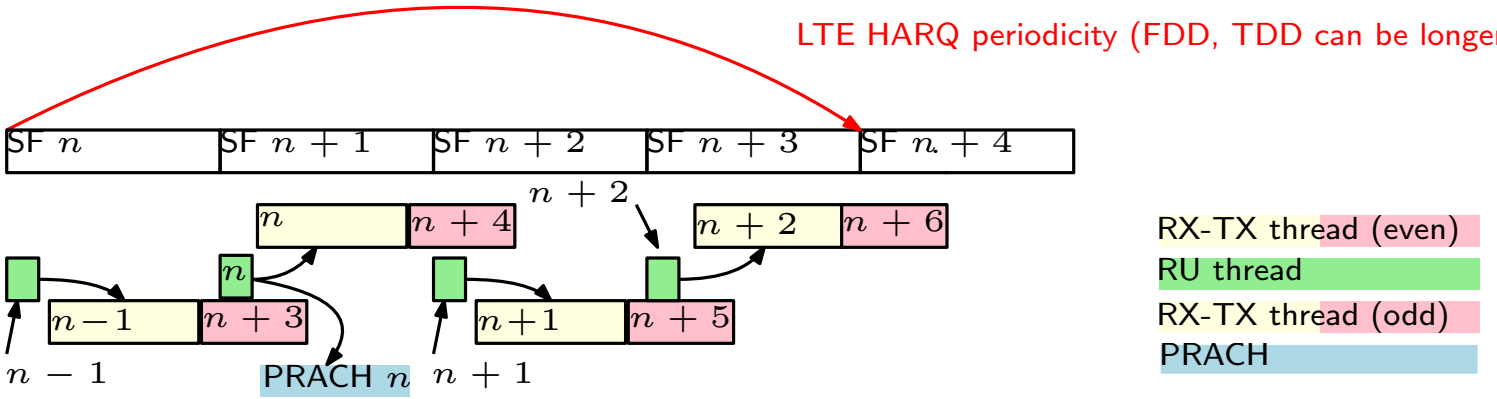
RU Threads

- Threads (all in `targets/RT/USER/lte-ru.c`)
  - `ru_thread`: Thread per RU which sequentially performs
    - read from south interface (RF or IF fronthaul)
    - RX processing for subframe n (if necessary).
    - wakeup eNBs that are waiting for signal (if necessary)
    - wait for eNB task completion (if necessary)
    - do TX processing for subframe $n+4$ (if necessary). Note that this can spawn multiple worker threads for very high order spatial processing (e.g. massive-MIMO or DAS for UDN)
    - do outgoing fronthaul (RF or IF fronthaul)
  - `ru_thread_prach`: Thread for PRACH processing in remote RU (DFT on RX, IF4p5 RRU)
  - `ru_thread_asynch`: Thread for asynchronous reception from fronthaul interface (TX direction in RRU). Allows for some jitter so as not to block the RRU real-time processing.
- Synchronization on fronthaul interface
  - *synch_to_ext_device* : synchronizes to incoming samples from RF or Fronthaul interface using blocking read
  - *synch_to_other* : synchronizes via POSIX mechanism to other source

eNB Threads

- Threads (all in `targets/RT/USER/lte-enb.c`)
  - multi RX/TX thread mode (optional)
    - `eNB_thread_rxtx`: 2 threads per CC/Instance which do both RX procedures for subframe n and TX procedures for subframe $n+4$. One operates on even subframes, one on odd. This allows 1ms subframe processing to use multiple-cores.
  - single RX/TX thread mode (default)
    - `eNB_thread_single`: Thread per CC/Instance which sequentially
      - blocks on signal from RU
      - RX/TX processing for subframe n and $n+4$
      - signals completion to RU
  - `eNB_prach`: Thread per CC_id/Instance for PRACH processing

eNB Timing (multi-thread mode)



- The current processing requires approximately 1ms peak in each direction (basically 1 core RX, 1core TX). The current architecture will work on a single core if the sum of RX and TX procedures is limited to 1ms. It can fit on 2 cores if the sum of RX,TX and PRACH is less than 2ms.
- three threads, RX-TX even, RX-TX odd and PRACH. RX-TX blocks until woken by the RU thread with a new RX subframe n that is linked to this eNB process. The RX-TX thread performs ue-specific processing for subframe n and then TX common and ue-specific processing for subframe $n+4$ (frequency-domain generation only). This insures the data dependency between TX $n+4$ and RX n is respected. The duration of this thread should be less than 2ms which can compensate some jitter on the RX processing.

eNB 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)