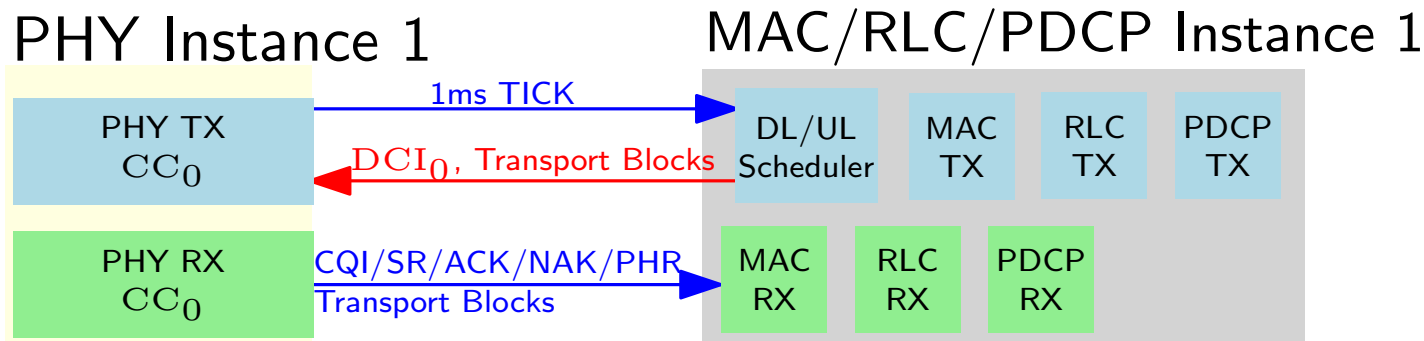
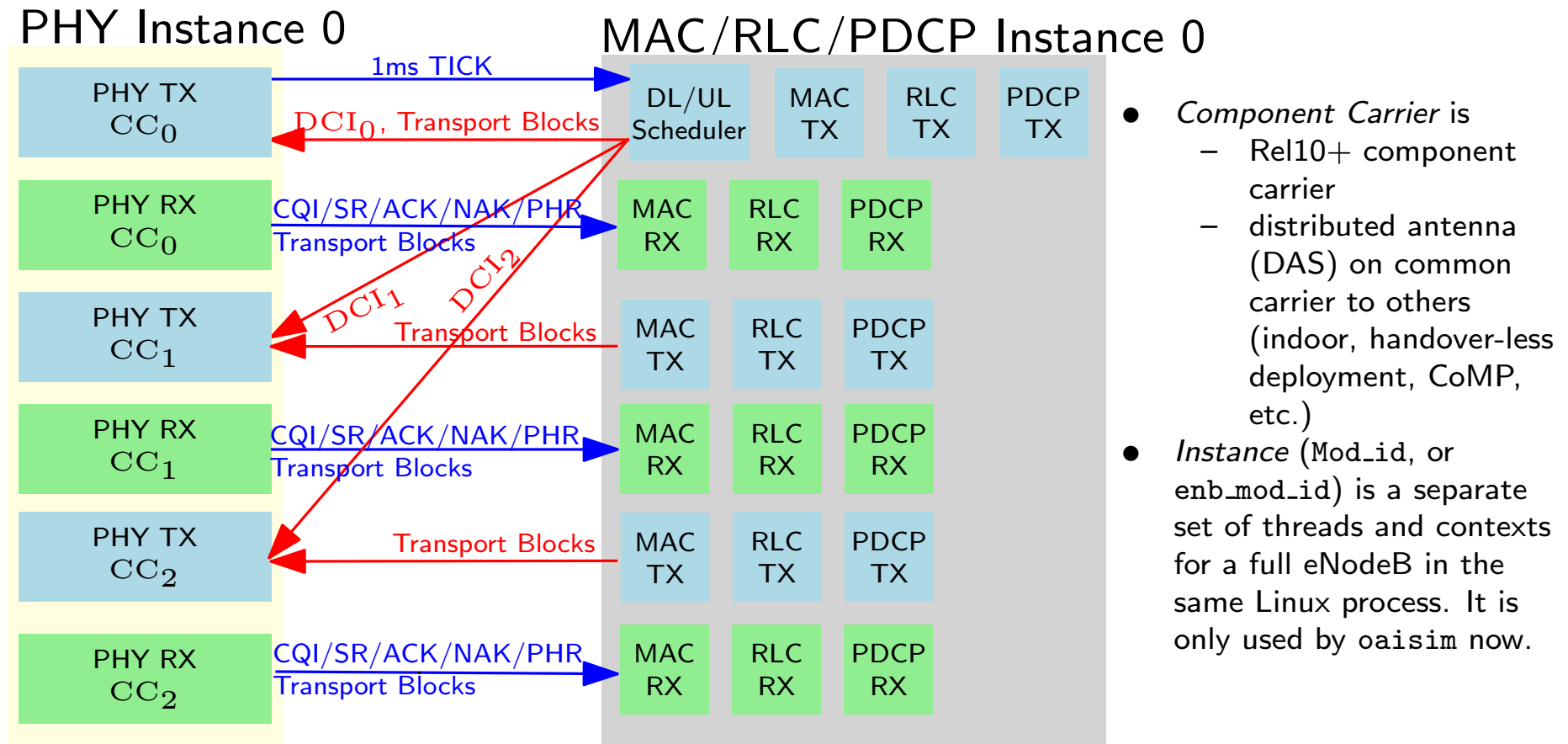


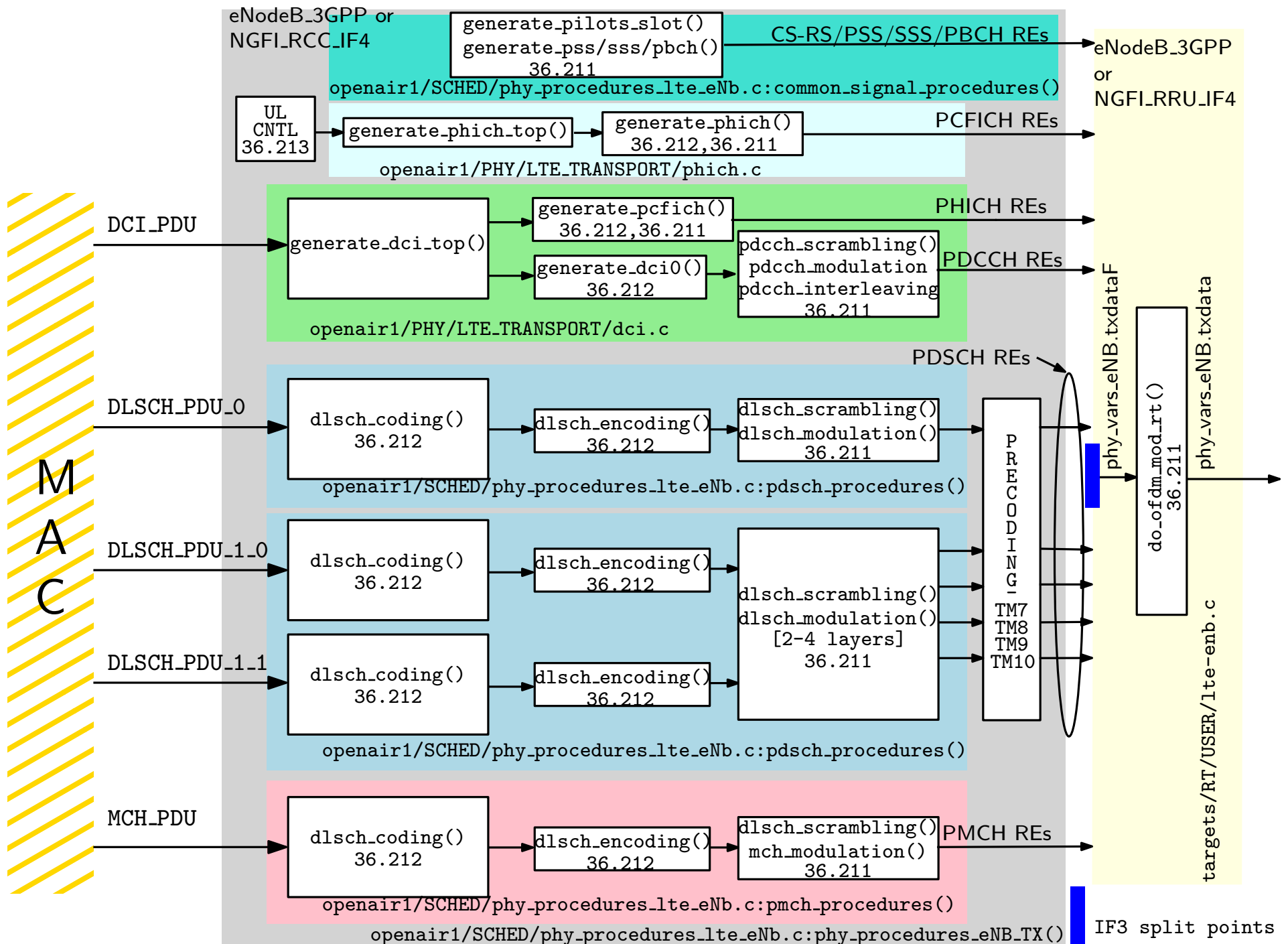
Some Preliminaries

- This description corresponds to the enhancement-10-harmony branch
- **purpose 1**: clean up interfaces within L1. L1/L2 are being done by others in the community (e.g. Bell Labs)
- **purpose 2**: harmonize HW platforms (hence the name)
- **purpose 3**: make the interfaces flexible for functional splits between RU/SU and local radio gateways
- **purpose 4**: harmonize lte-softmodem (lte-ue/lte-enb) and oasisim

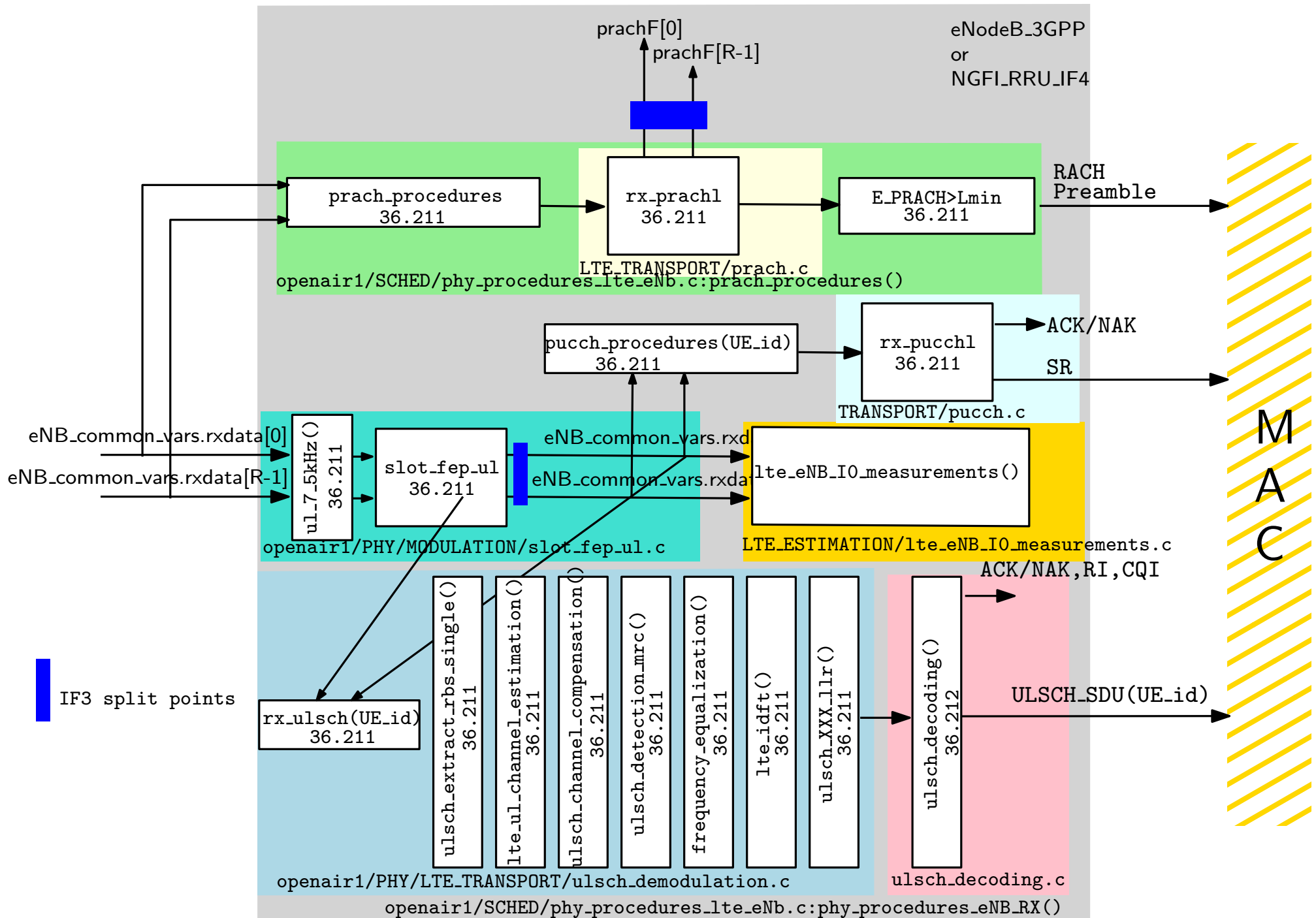
eNodeB Threads, Instances and Component Carriers



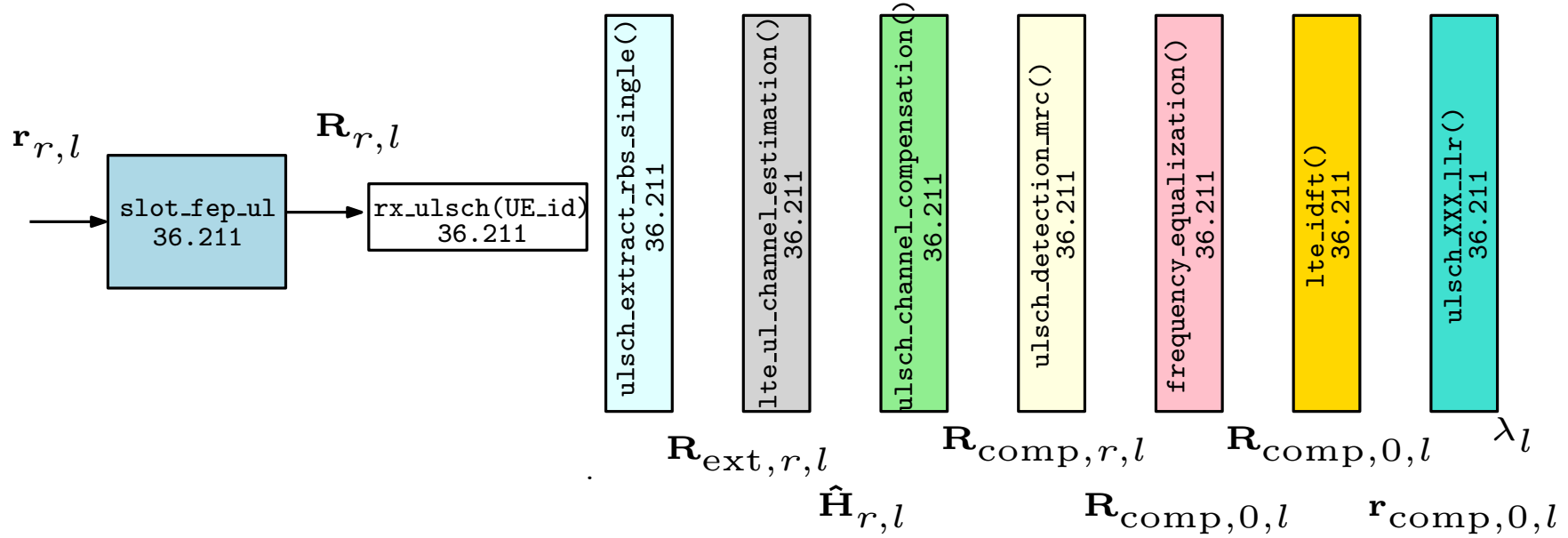
eNodeB PHY TX Procedures



eNodeB PHY RX Procedures



eNodeB ULSCH Demodulation



$$\mathbf{R}_{r,l} = \text{DFT}_{N_{\text{fft}}}(\mathbf{r}_{r,l} \odot \mathbf{F}_{7.5}), r = 0, 1, \dots, R-1, l = 0, 1, \dots, N_{\text{symb}} - 1 \text{ (eNB_common_vars} \rightarrow \text{rxdataF}[\text{[]}] \text{)}$$

$$R_{\text{ext},r,l}(n) = R_{r,l}(12\text{firstPRB} + n), n = 0, 1, \dots, 12N_{\text{PRB}} - 1 \text{ (eNB_pusch_vars} \rightarrow \text{ulsch_rxdataF_ext}[\text{[]}] \text{)}$$

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

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

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

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

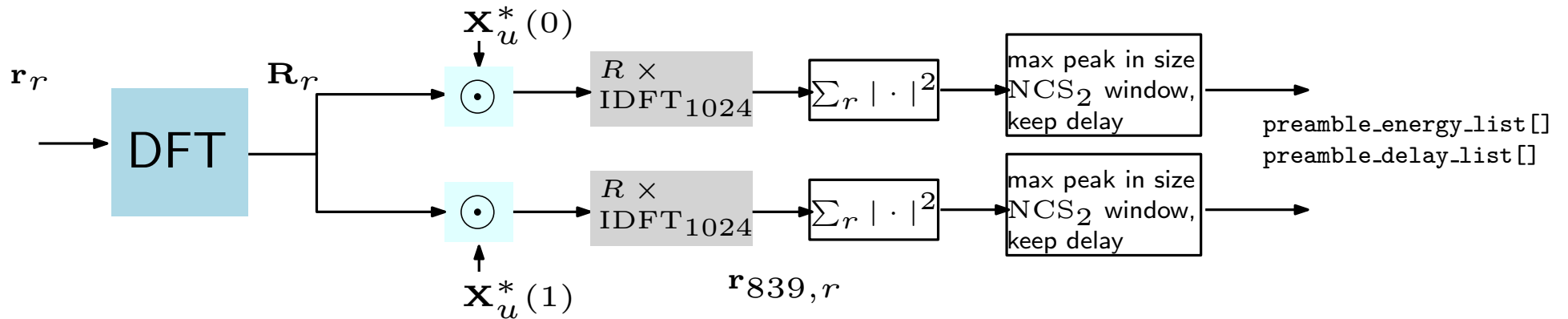
$$\mathbf{r}_{\text{comp},0,l} = \text{DFT}_{12N_{\text{PRB}}}(\mathbf{R}_{\text{comp},0,l})$$

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

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

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

eNodeB PRACH Detection



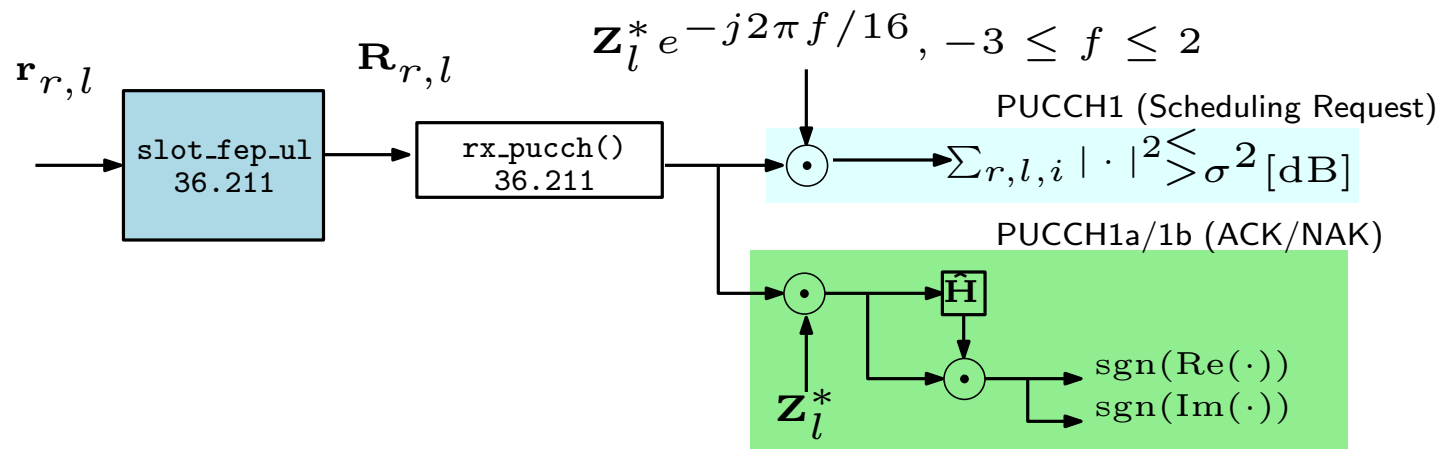
$\mathbf{R}_r = \text{DFT}_{N_{\text{PRACH}}}(\mathbf{r}_r), r = 0, 1, \dots, R - 1$ (lte_eNB_prach_vars \rightarrow rxsigF[])

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

$\mathbf{r}_{839,r} = \text{IDFT}_{1024}(\mathbf{R}_{\text{comp},r}), r = 0, 1, \dots, R - 1$ (lte_eNB_prach_vars \rightarrow prach_ifft[])

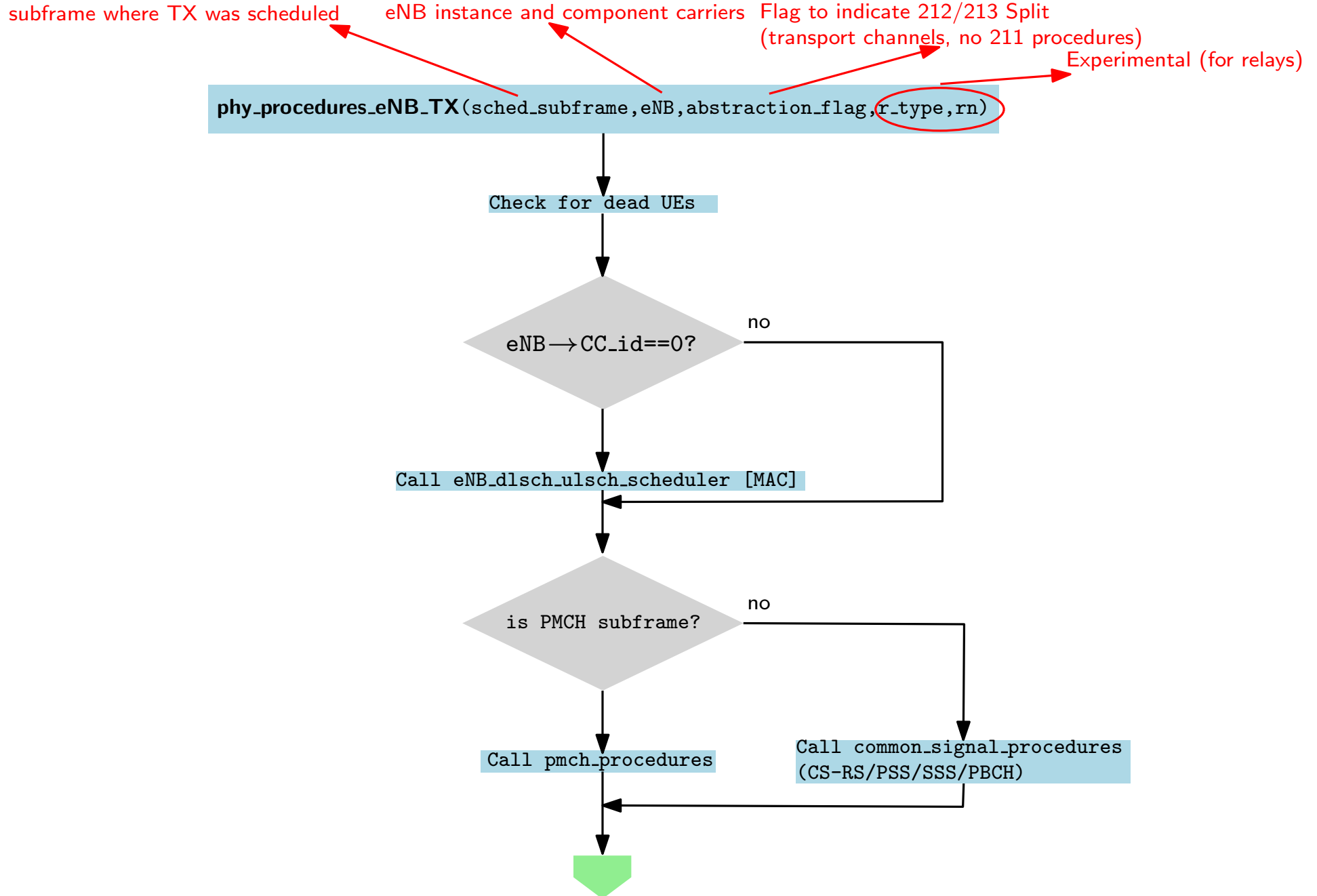
- PRACH detection is a quasi-optimal non-coherent receiver for vector observations (multiple antennas)
- correlation is done in the frequency-domain, number of correlations (in the example above 2) depends on *zeroCorrelationConfig* configuration parameter
- peak-detection (for delay estimation) is performed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

eNodeB PUCCH Detection

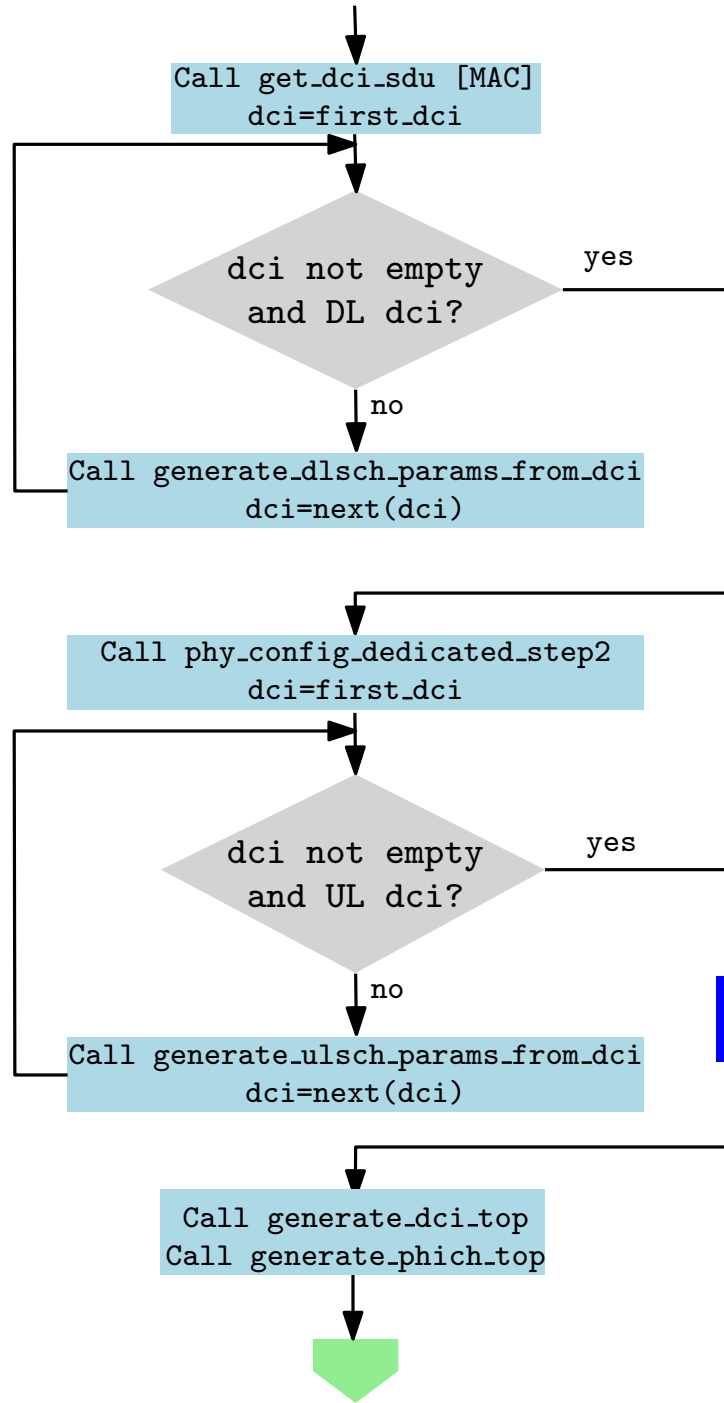


- PUCCH1 detection is a quasi-optimal non-coherent receiver (energy detector) for vector observations (multiple antennas) for scheduling request. Care is taken to handle residual frequency-offset.
- PUCCH1A/1B detection is quasi-coherent based on a rough channel estimate obtained on the 3 symbols without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

eNodeB TX Flowchart

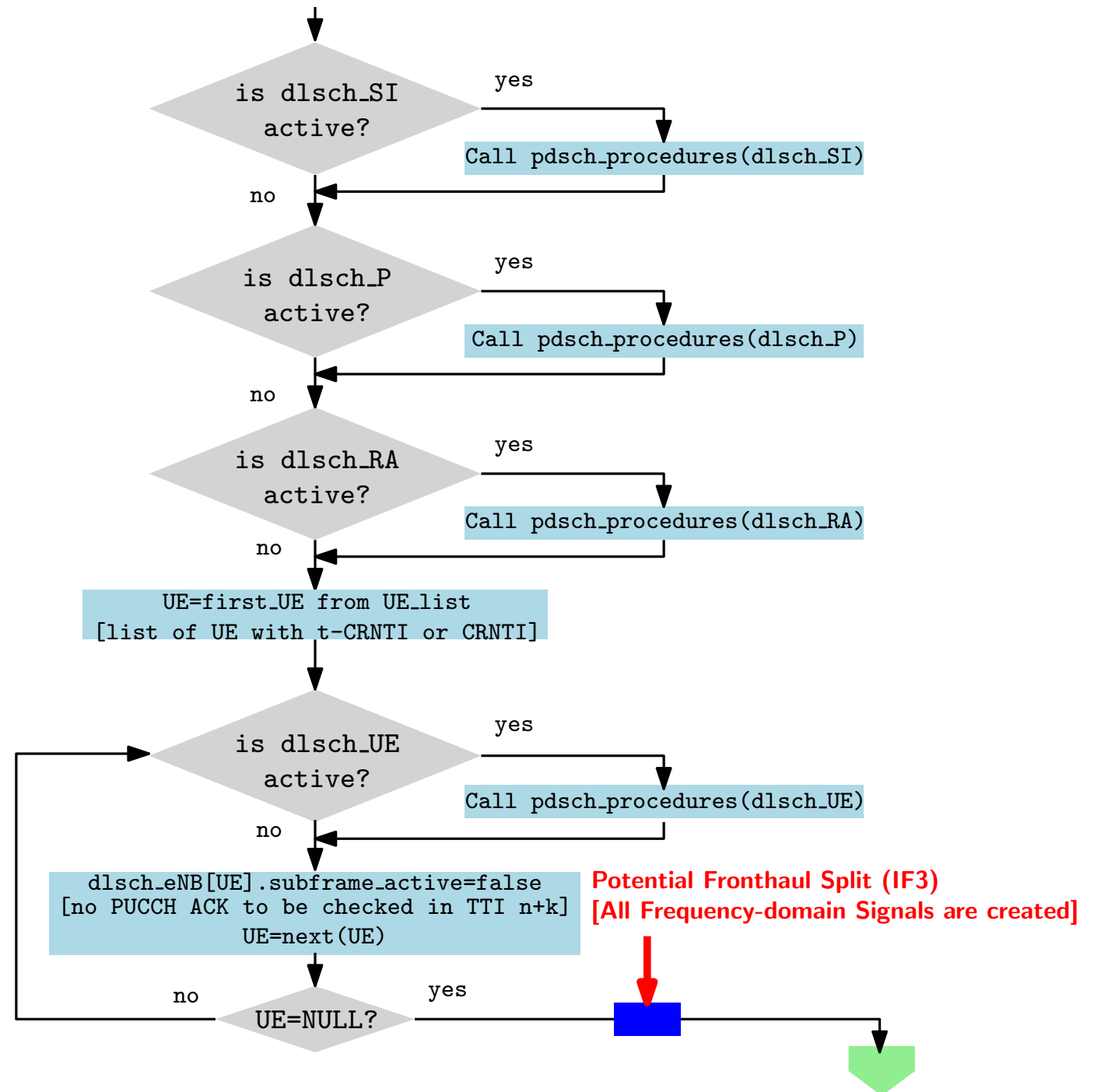


eNodeB TX Flowchart

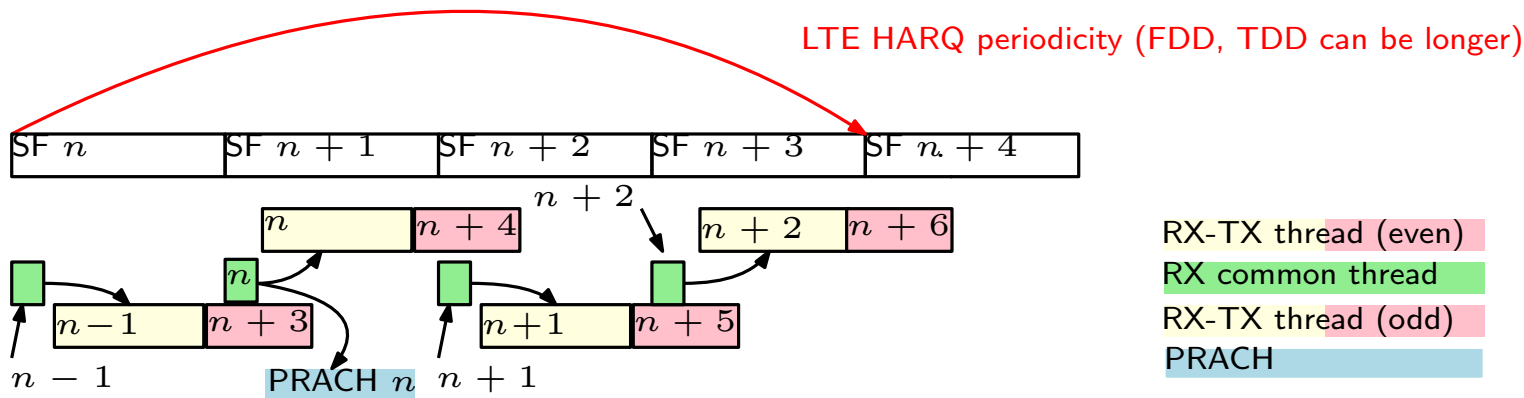


Potential Fronthaul Split
(a bit like NodeB-RNC interface)

eNodeB TX Flowchart



eNodeB Timing



- 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.
- four threads, RX common, RX-TX even, RX-TX odd and PRACH. RX common processes the portion of the subframe common to all UEs and then wakes up one of the 2 RX-TX threads and the PRACH thread if necessary. The RX-TX thread first performs RX ue-specific processing for subframe n and then TX common and ue-specific processing for subframe $n+4$. 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.

eNodeB 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)`

