华为技术有限公司	版本	密级
	BoudicaV100R100C10B650SP8	内部公开
	文档编号	共37页

BoudicaV100R100C10B650SP8 Debug Manual

项目名称 ：Neul-Hi2110

华为技术有限公司

Revision Record 修订记录

Date 日期	Revision Version 修订版本	CR ID CR号	Sec No. 章节	Change Description 修改描述	Author 作者
2016-11-22	0.1			初稿建立	陈贞

目录

1. 目的和范围

1.1. 目的

本文档用于指导研发和各条测试线人员初步定位 NB-IOT 协议侧的问题。主要描述了协议各层 NAS/RRC/L2/L1 的关键日志和常见问题以及常见问题的定位步骤。

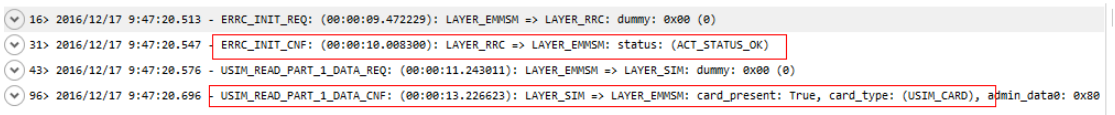
1.2. 范围

本文适用于研发内部、系统测试及外场测试人员。

2. 初步排查手册

2.1. 开机流程

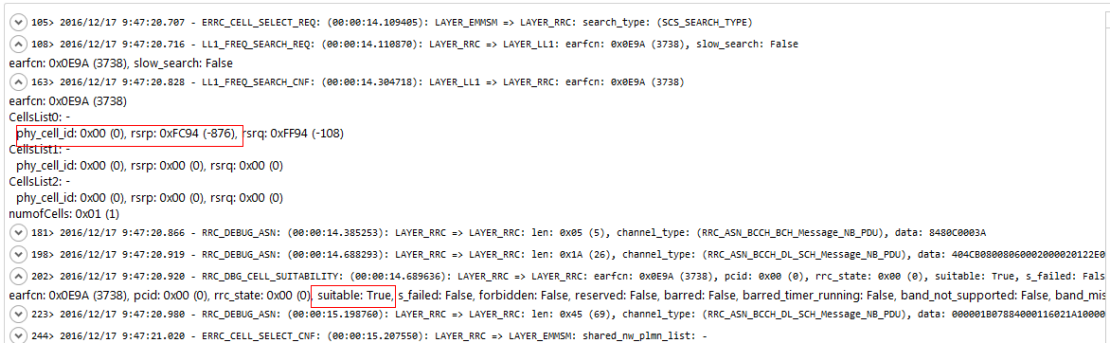
正常的开机流程如下图所示，如果出现下面的 log 说明开机流程正常。过滤条件：ERRC_INIT、usim_read。需要注意卡状态和卡类型选项。



这个流程中常见的问题是没有读 SIM 卡的回复消息，出现这种情况一线可以反馈 log 给家里，并说明是 SIM 没回复。

2.2. 搜网流程

一定要注意的是发送注册的 AT 命令要在开机流程完成之后(等到 log 出现 USIM_READ_PART_1_DATA_CNF 之后)。搜网流程的完整 log 如下。过滤条件为：ERRC_CELL_SELECT、LL1_FREQ_SEARCH、cell_suit、asn。其中 ASN 可以查看读到的 MIB/SIB 消息。

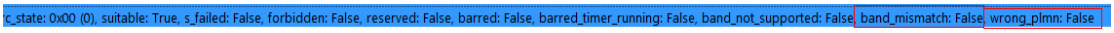


这个流程常出现的问题：

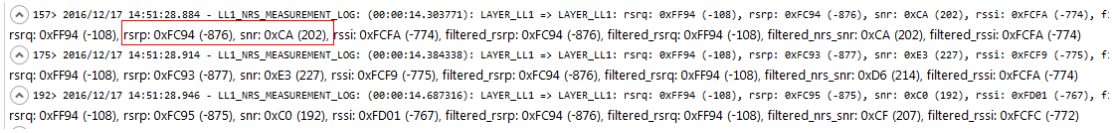
1、开机 AT 命令发送之后等待时间太短就发起注册的 AT 命令，导致失败；

2、搜网过程中一直没搜到合适的小区，上面 log 中 LL1_FREQ_SEARCH_CNF 的 numofCells 会是 0，这个时候一线同事需要排查信号和频段；

3、搜网搜到了小区但是小区不可用，上面 log 中可以通过查看 suitable 是否为 true 来判断，如果不为 true，log 的后面会打印出到底是什么原因，如果下面红框内容出现 true，一线需要排除 plmn 和 band 是否合适。其他情况一线同事可以反馈 log 给家里，注明小区不合适驻留。

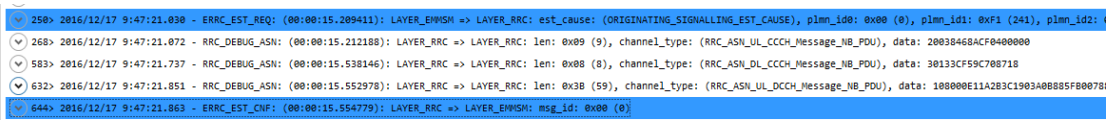


4、搜网失败很可能是由于信号太差导致，通过日志过滤 ll1_nrs 可以看到当前信号质量。里面有两个关键参数：RSRP 和 SNR（需要除 10 进行折算）。RSRP 小于-120 认为信号已经不是很好，SNR 小于-10 认为干扰已经很强。



2.3. 建链流程

建链流程的完整 log 如下。过滤条件为： ERRC_EST、asn。其中 ASN 是 msg3，msg4，msg5。

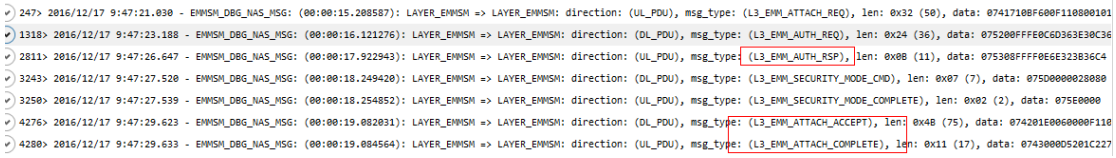


这个流程常出现的问题：

1、建链过程中出现多条 msg3、msg4，如果最后能够出现 ERRC_EST_CNF，这个是正常的重传。如果没有收到，需要一线反馈 log 到家里，同时注明现象。

2.4. 附着流程

附着流程的完整 log 如下。过滤条件为：dbg_N。



这个流程常出现的问题：

1、如果没有收到网测发起的鉴权和安全流程，这种问题需要一线排查核心网的安全和鉴权开关有没有开。

2、如果出现鉴权被拒，同时被拒原因值是 mac_failure，需要一线排查 SIM 卡的开户信息是否正确。

3、如果出现 attach 被拒，需要一线反馈 log 到家里，同时注明现象。

2.5.数传流程

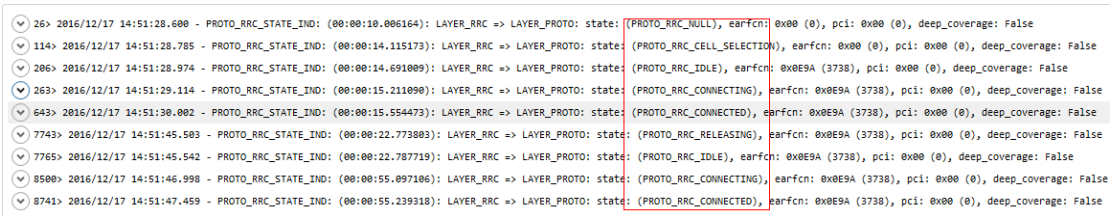
数传流程的完整 log 如下。过滤条件有 3 个，如下所示：
pdh_data_req 、
(UL_PDU), msg_type: (L3_ESM_DATA_TRANSPORT)、
(UL_PDU), msg_type: (L3_EMM_CONTROL_PLANE_SERVICE_REQ)。下图中每个红色框代表一包数据，每个数据包都有一个 pdh_data_req 消息。

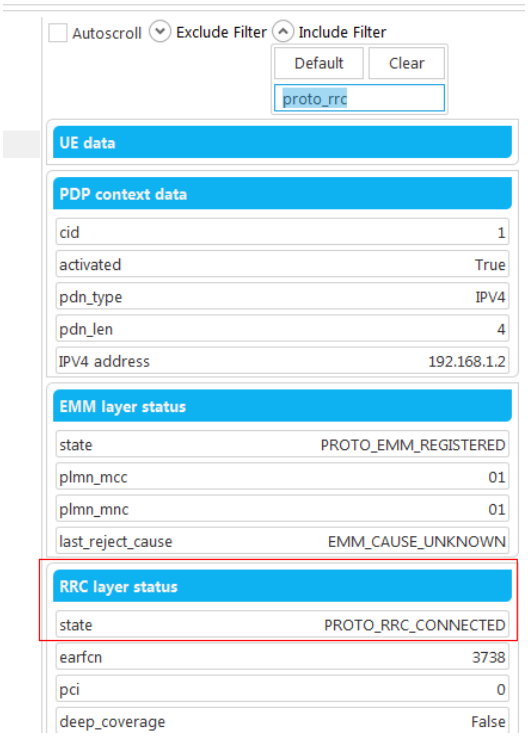
这个流程常出现的问题：

1、串口发送数据之后首先需要在 log 过滤查看数据是否到达芯片协议栈，可以通过 log 中 PDH_DATA_REQ 确认，如果没有该 log，说明数据没有发送成功，需要排查串口、模组以及 AT 命令式否正确；

2、如果在日志中出现 PDH_DATA_REQ，需要一线同事反馈数传失败 log 以及串口操作日志给家里，同时请注明问题时间点。

3、如果想要知道当前是 idle 态还是 connect 态，可以通过过滤 proto_rrc 进行查看，或者在运行时查看 uelogview 的右下方的红色框处。。





3. NAS 问题定位指导

3.1. 附着的概念

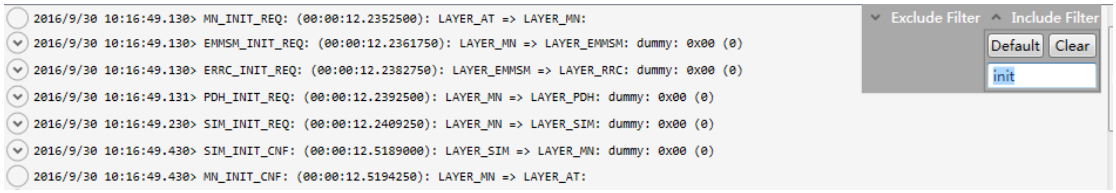
NAS 层属于层 3，在 RRC 层之上。UE 要获取 EPS 服务，必须 ATTACH 到 EPC，ATTACH 成功后 UE 与 MME 之间有相同的移动性上下文，UE 和 PDN GW 之间建立起缺省承载。通过 EPS ATTACH 流程，UE 还可以获取到网络分配的 IP 地址。

3.2. Log 查看

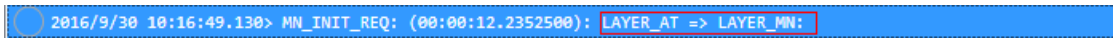
在查看 log 时我们需要从开机到搜网到附着逐步查看，在开机成功的条件下看搜网，在搜网成功的条件下看附着消息。下面从这两方面进行说明。

3.2.1. 开机消息查看

检索关键词: INIT
作用：在 NAS 层我们需要查看终端的开机是否成功，如下图所示：
以下每一条消息都是必选消息，缺一不可。



1) 第一条消息是 AT 模块下发的开机消息



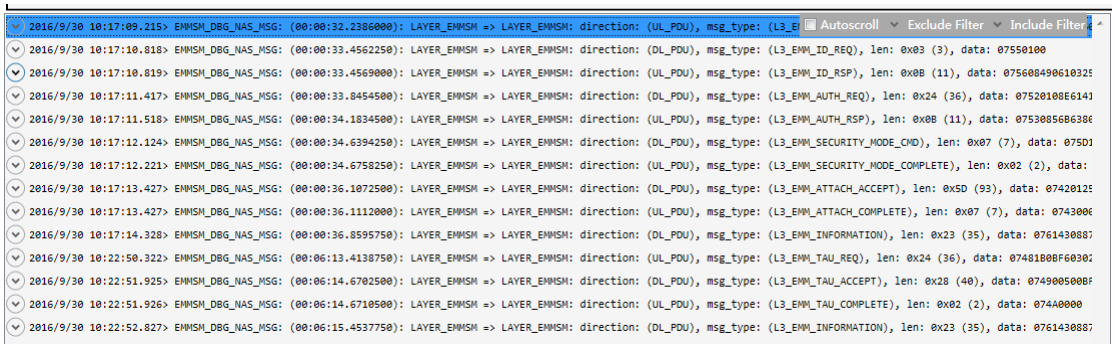
上图可以看到消息的方向是 AT 到 MN

2) 第二条消息是 MN 下发给 NAS 层的开机请求消息
3) 第三条消息是 NAS 下发给 RRC 的开机请求消息
4) 第四条消息是 MN 下发给 PDH 模块的开机请求消息
5) 第五条消息是 MN 下发给 SIM 卡的开机请求消息
6) 第六条消息是 SIM 卡回复给 MN 的开机确认消息
7) 第七条消息是 MN 给 AT 的开机确认消息，**看到该 log 表示 UE 开机完成，可以进行后续的流程**。

3.2.2. 空口消息查看

1. 过滤关键字 emmsm_dbg_nas_msg

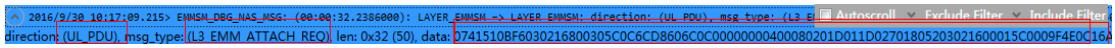
过滤后可以看到如下图所示：



下面说明这里面每一条 Log 的含义

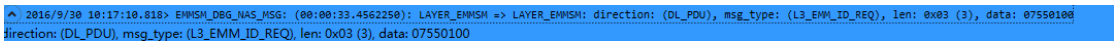
红色加粗的消息表示每一次单板进行附着流程必选的消息，其它消息均为可选消息。

1） **点开第一条 Log 如下图所示：**



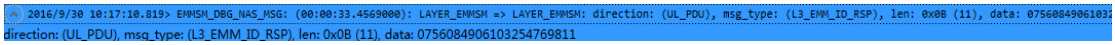
框住的三部分分别是表示上下行消息(direction)、该消息的名称(msg_type)和具体的码流(data)，可以看到 Log 里已经说明了该消息是 ATTACH REQ 消息，UL 表示上行即从 UE 发送的核心网。

2） 第二条 log 如下图所示：



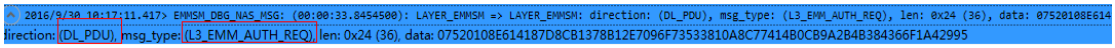
该消息是身份验证消息，该消息是可选的，不是必选消息，属于正常消息，是核心网对 UE 的身份验证消息，从图中可以看到 DL，即从核心网到 UE。

3） 第三条 Log 如下图所示：



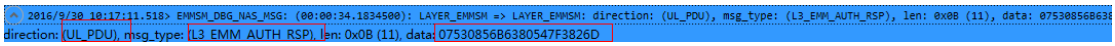
该消息是 UE 对身份验证的回复消息。

4） 第四条消息是鉴权请求消息，如下图所示：

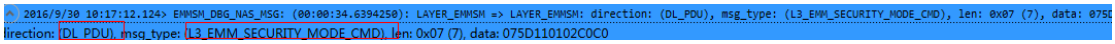


鉴权消息是核心网发给 UE 的下行消息，用来确认 UE 的合法性。

5） 第五条消息是 UE 对核心网鉴权消息的回复消息，如下图所示：

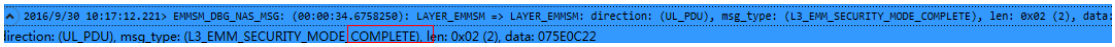


6） 第六条消息是核心网下发给 UE 的安全模式命令消息：

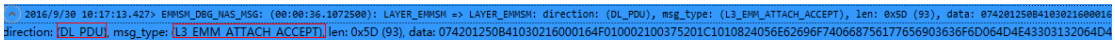


该消息是核心网和 UE 协商安全算法的消息，通过该消息 UE 和核心网建立了完整的安全上下文，后续的消息会经过加密处理。

7） 第七条消息是 UE 对核心网的安全模式命令消息的回复消息

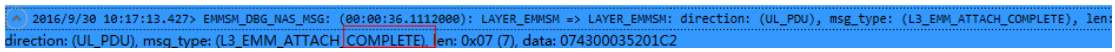


8） 第八条消息是核心网对第一条 ATTACH REQ 的回复消息



收到该消息说明核心网接收 UE 的 ATTACH 请求，允许 UE 进行附着业务。

9） 第九条消息是 UE 发给核心网的 ATTACH 完成消息



该消息表示附着流程已经完成，后续 UE 可以进行正常的业务。

3.3. 常见问题及解决方法

常见的需要注意的事项

A. XML 文件需要放到 C:\Program Files (x86)\Neul\UE Log Viewer\Decoders 目录下

B. 串口 J10 适用 AT 命令，J11 适用打 LOG，注意不要搞错

常见的问题及排查解决方法

1) 开机失败

解决方法：

1、 检查板子上电情况

2、 检查 sim 卡是否插入

3、 检查跳线帽是否插入是否安装正确

2） 搜网失败

引起搜网失败的原因比较多，主要是接入层的问题，这里主要在一些操作上给一点定位方法

1、 检查 AT 命令是否输入正确

2、 检查板子是否连接上信号源

3） ATTACH 失败

ATTACH 失败主要由两大方面引起

a） 随机接入失败导致

1、随机接入失败主要是由 msg1 到 msg5 其中的某条消息处理有问题导致，这里不做具体的说明。

b） 信令处理失败导致

鉴权失败

1、 检查卡的鉴权安全开关在核心网是否打开

2、 检查卡的 KI/OP 是否和核心网设置的一样

3、 检查 SIM 卡是否是 4G 的 SIM 卡，目前只支持 4G 的 sim 卡

4、 检查卡是否烧坏（sim 卡不能热拔插，不当的操作可能导致 SIM 卡烧坏）

5、 检查 S1 口的安全配置是否正确（目前版本只能支持到 EEA1/EIA1）

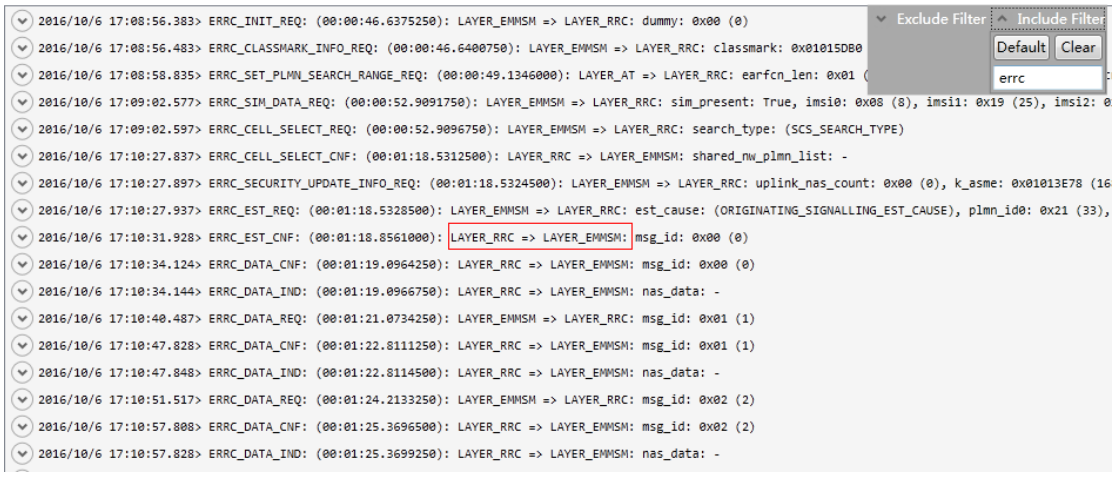
4. RRC 问题定位指导

4.1. 流程梳理用的 LOG

4.1.1. 定位消息流向和行为

检索关键词：ERRC

作用：可确认 Layer_RRC 层与其他层之间的消息定义和消息流向



较为常用的 LOG 意义解释：

ERRC_CELL_SELECT_REQ/CNF/REJ

NAS 给 RRC 发送指定 PLMN 的搜网请求/响应/拒绝

ERRC_CELL_SELECT_IND

RRC 层在离开 Connect 态之后主动驻留的小区指示

ERRC_PLMN_SEARCH_REQ/CNF/REJ

NAS 给 RRC 发送搜索 PLMN 的请求/响应/拒绝

ERRC_PLMN_SEARCH_IND

RRC 搜索到一个 PLMN 并上报

ERRC_EST_REQ/CNF/REJ

NAS 给 RRC 发送建立链路的请求/响应/拒绝

ERRC_DATA_REQ/CNF/REJ

NAS 给 RRC 发送上行数据发送请求/响应/拒绝

ERRC_DATA_IND

RRC 给 NAS 发送下行数据指示

ERRC_REL_REQ/CNF/REJ

NAS 给 RRC 发送释放链路请求/响应/拒绝

ERRC_REL_IND

RRC 层上报（基站发起或因 UE 异常发起）释放指示

以上消息可用于梳理 RRC 层的消息流程，一个 REQ 对应一个 CNF 或者 REJ，IND 消息为 RRC 主动上报给上层；如果消息没有成对，在确认 Log 没有丢失的情况下，可帮助缩小定位范围；

4.1.2. 定时器超时场景

检索关键词：RRC_TIMER

作用：可确认 RRC 层协议是否存在协议定时器超时

较为常用的 LOG 意义解释：

RRC_TIMER_T300_EXPIRY_IND：RRC 连接建立失败

RRC_TIMER_T310_EXPIRY_IND：失同步，会释放连接

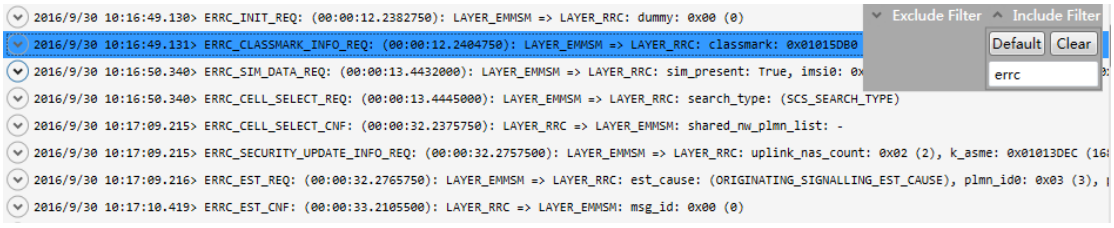
RRC_TIMER_BARRING_EXPIRY_IND：BAR 掉的一个小区或频点超过 300s

4.1.3. 关键流程分析

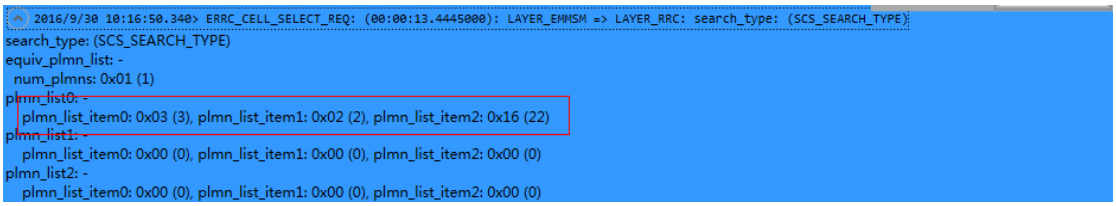
1. 搜网流程

检索关键字：ERRC

在 UE 开机完成后要想获取服务需要驻留在某一个小区，并进行随机接入流程，如下图所示：



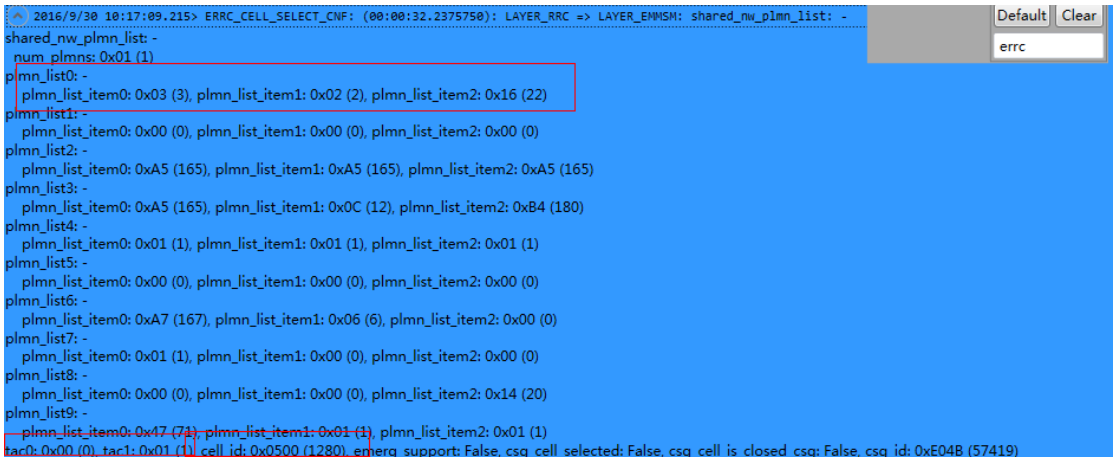
1) 第 4 条消息是 NAS 给 RRC 下发的小区搜索消息，在 AT 命令中我们指定了 PLMN 的条件下 NAS 会直接下发该消息，如下图所示：



上图中框起来的是 PLMN，PLMN 是 BCD 编码，具体的编码格式这里不再说明有兴趣的可以

下去研究，我这里的 PLMN 是 302610，在不同的基站下，AT 下发的 PLMN 发生变化。

2) 第 5 条消息是 RRC 给 NAS 上报的小区搜索回复消息

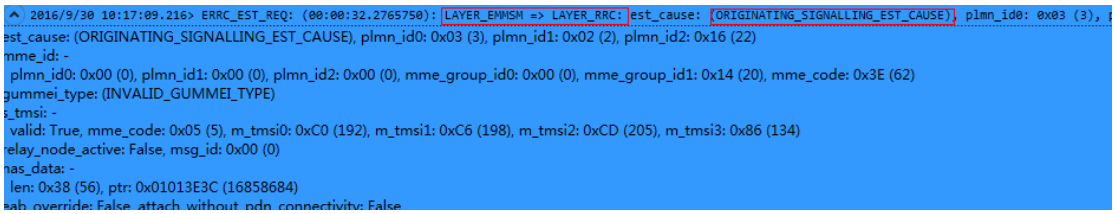


该消息中包含当前小区的 PLMN,TAC 以及小区 ID，这里小区 ID 一般取后两位，这里显示的是 80。

2. 连接建立流程

当小区选择成功，即 UE 驻留到小区后 UE 会发起附着流程，触发接入层做随机接入，在 NAS 层看来是就是需要建立连接（空口信令连接），这是 NAS 层发起建链请求，同时触发下层开始做随机接入。

1) 这时在 NAS 层我们看到了第七条 Log，如下图所示：



上图框住的部分分别表示该消息的方向即从 NAS 到 RRC，发起建立连接的原因初始信令原因。

2) 第八条 Log 表示 RRC 对连接建立的回复或者成功或者失败

上图中框住的部分是该消息的名称，看到该消息说明底层随机接入成功，在 NAS 层看来就是建链成功了。

同时需要注意的是如果收到了 ERRC_EST_REJ 消息，说明底层随机接入失败，建链不成功，这时 UE 的 NAS 层会继续发起第七条消息，进行再次建链。

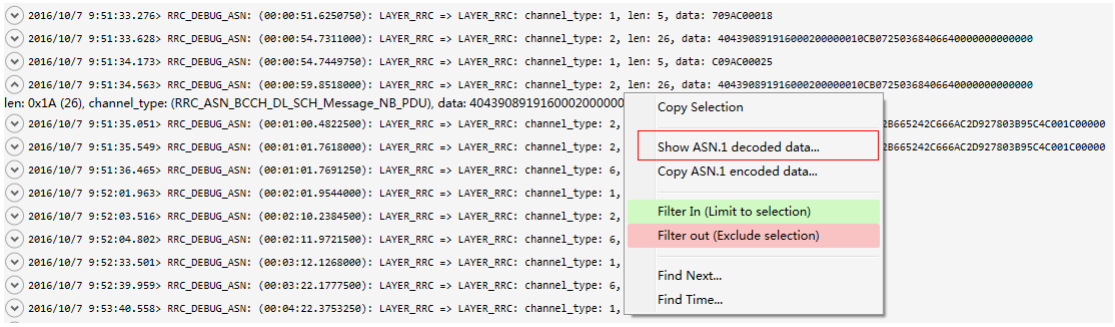
4.2. 获取参数信息的重要 Log

4.2.1. UU 口 ASN 编码数据

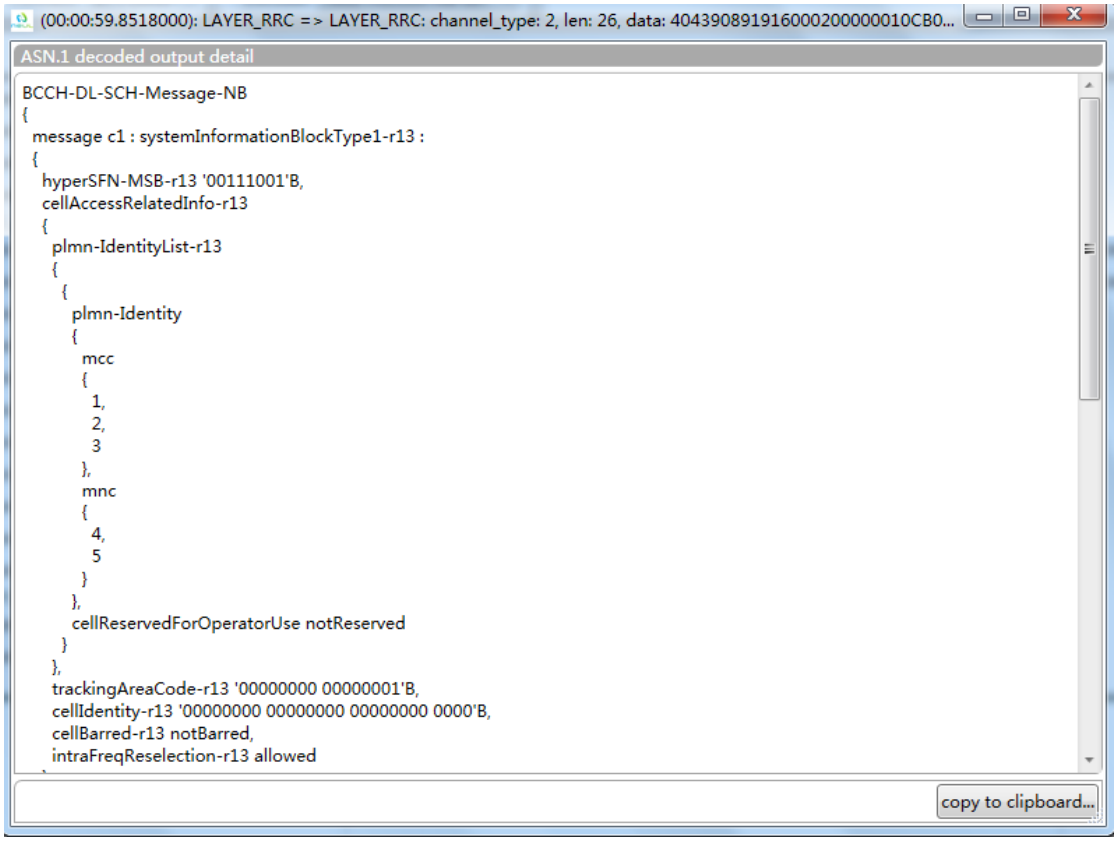
关键字：ASN

查看方法:

1、选中想要查看的 ASN 数据，右键选择 Show ASN.1 decoded data



2、查看解码后信元参数

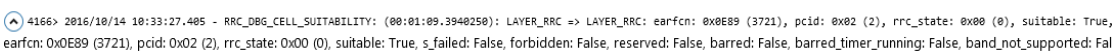


通过 ASN 解码可以完全跟踪 UU 口的数据和行为，是 UE 侧确认绝大多数参数配置和交互的优先手段；

4.2.2. 查看搜索到的小区是否能够驻留

关键字：

RRC_DBG_CELL_SUITABILITY

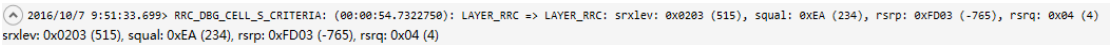


如果在小区选择中，看到 ERRC_CELL_SELECT_REJ,则可以通过 RRC_DBG_CELL_SUITABILITY 得知小区不能驻留的原因，而且也可以获得目标小区 earfcn，pci 等综合信息，可以结合查看

4.2.3. 查看搜索到的小区 S 准则是否满足

关键字：

RRC_DBG_CELL_S_CRITERIA

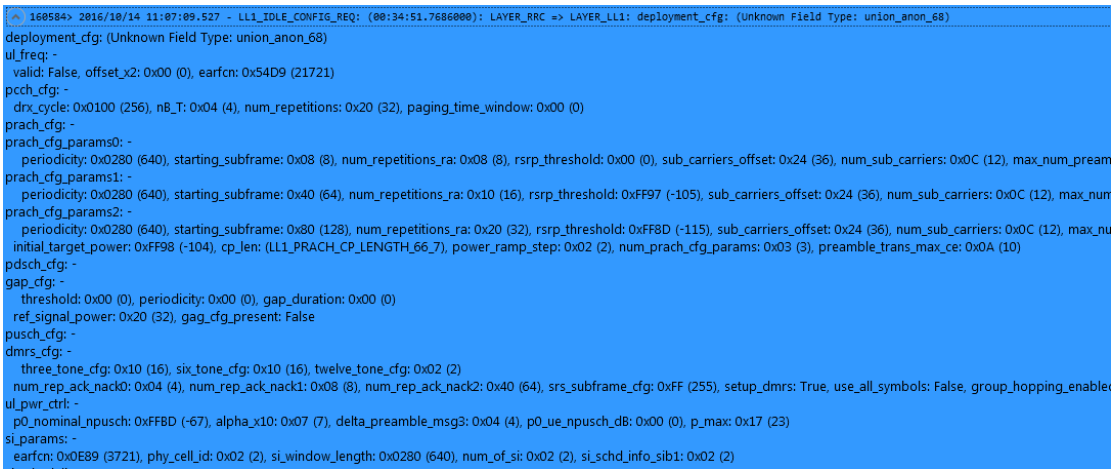


RRC_DBG_CELL_S_CRITERIA 可以获取当前目标小区的 rsrp 及 rsrq 值；只有 Srxlev 及 Squal 均大于 0，才能驻留小区；

4.2.4. 查看 RRC 对 L2 及 L1 的参数配置

关键字：

LL1_IDLE_CONFIG_REQ



LL1_CONNECTED_CONFIG_REQ

```
2016/10/15 16:26:56.832 - LL1_CONNECTED_CONFIG_REQ: (00:08:27.6010330): LAYER_RRC => LAYER_LL1: deployment_cfg: (Unknown Field Type: union_anon_68)
deployment_cfg: (Unknown Field Type: union_anon_68)
ul_freq: -
  valid: False, offset_x2: 0x00 (0), earfcn: 0x549C (21660)
prach_cfg: -
prach_cfg_params0: -
    periodicity: 0x0280 (640), starting_subframe: 0x08 (8), num_repetitions_ra: 0x08 (8), rsrp_threshold: 0x00 (0), sub_carriers_offset: 0x24 (36), num_sub_carriers: 0x0C (12), max_num_preamble_attempt_ce: 0x04 (4), num_rep_per_preamble
prach_cfg_params1: -
    periodicity: 0x00 (0), starting_subframe: 0x00 (0), num_repetitions_ra: 0x00 (0), rsrp_threshold: 0x00 (0), sub_carriers_offset: 0x00 (0), num_sub_carriers: 0x00 (0), max_num_preamble_attempt_ce: 0x00 (0), num_rep_per_preamble_attemp
prach_cfg_params2: -
    periodicity: 0x00 (0), starting_subframe: 0x00 (0), num_repetitions_ra: 0x00 (0), rsrp_threshold: 0x00 (0), sub_carriers_offset: 0x00 (0), num_sub_carriers: 0x00 (0), max_num_preamble_attempt_ce: 0x00 (0), num_rep_per_preamble_attemp
  initial_target_power: 0xFF98 (-104), cp_len: (LL1_PRACH_CP_LENGTH_66_7), power_ramp_step: 0x02 (2), num_prach_cfg_params: 0x01 (1), preamble_trans_max_ce: 0x0A (10)
pdcch_cfg: -
  max_num_repetitions: 0x08 (8), start_sf_uss_x2: 0x08 (8), pdcch_offset_uss: (LL1_PDCCH_OFFSET_ZERO)
pusch_cfg: -
dmrs_cfg: -
    three_tone_cfg: 0x00 (0), six_tone_cfg: 0x00 (0), twelve_tone_cfg: 0x00 (0)
  num_rep_ack_nack0: 0x20 (32), num_rep_ack_nack1: 0x20 (32), num_rep_ack_nack2: 0x20 (32), srs_subframe_cfg: 0xFF (255), setup_dmrs: False, use_all_symbols: True, group_hopping_enabled: False, group_assignment: 0x00 (0)
pdsch_cfg: -
gap_cfg: -
    threshold: 0x00 (0), periodicity: 0x00 (0), gap_duration: 0x00 (0)
  ref_signal_power: 0x1B (27), gag_cfg_present: False
ul_pwr_ctrl: -
  p0_nominal_npusch: 0xFFBD (-67), alpha_x10: 0x07 (7), delta_preamble_msg3: 0x04 (4), p0_ue_npusch_dB: 0x00 (0), p_max: 0x17 (23)
si_params: -
  earfcn: 0x0E4C (3660), phy_cell_id: 0x03 (3), si_window_length: 0x0280 (640), num_of_si: 0x02 (2), si_schd_info_sib1: 0x02 (2)
si_scheduling0: -
    sib_mapping_bitmap: 0x01 (1), si_periodicity: 0x80 (128), si_tbs_size: 0xD0 (208), repetition_pattern: 0x04 (4), radio_frame_offset: 0x00 (0)
si_scheduling1: -
    sib_mapping_bitmap: 0x02 (2), si_periodicity: 0x0100 (256), si_tbs_size: 0x38 (56), repetition_pattern: 0x04 (4), radio_frame_offset: 0x00 (0)
si_scheduling2: -
    sib_mapping_bitmap: 0x00 (0), si_periodicity: 0x00 (0), si_tbs_size: 0x00 (0), repetition_pattern: 0x00 (0), radio_frame_offset: 0x00 (0)
si_scheduling3: -
    sib_mapping_bitmap: 0x00 (0), si_periodicity: 0x00 (0), si_tbs_size: 0x00 (0), repetition_pattern: 0x00 (0), radio_frame_offset: 0x00 (0)
si_scheduling4: -
    sib_mapping_bitmap: 0x00 (0), si_periodicity: 0x00 (0), si_tbs_size: 0x00 (0), repetition_pattern: 0x00 (0), radio_frame_offset: 0x00 (0)
si_scheduling5: -
    sib_mapping_bitmap: 0x00 (0), si_periodicity: 0x00 (0), si_tbs_size: 0x00 (0), repetition_pattern: 0x00 (0), radio_frame_offset: 0x00 (0)
si_scheduling6: -
    sib_mapping_bitmap: 0x00 (0), si_periodicity: 0x00 (0), si_tbs_size: 0x00 (0), repetition_pattern: 0x00 (0), radio_frame_offset: 0x00 (0)
si_scheduling7: -
    sib_mapping_bitmap: 0x00 (0), si_periodicity: 0x00 (0), si_tbs_size: 0x00 (0), repetition_pattern: 0x00 (0), radio_frame_offset: 0x00 (0)
ll1_valid_subframes: -
    num_valid_subframes: 0x00 (0), valid_subframes0: 0x00 (0), valid_subframes1: 0x00 (0), valid_subframes2: 0x00 (0), valid_subframes3: 0x00 (0), valid_subframes4: 0x00 (0)
```

L2_UL_CONFIG_REQ

L2_DL_CONFIG_REQ

```
49059> 2016/10/14 10:43:40.019 - L2_UL_CONFIG_REQ: (00:11:22.0744000): LAYER_RRC => LAYER_L2_UL: pdcp_cfg: -
pdcp_cfg: -
header: -
    message_id: 0x00 (0), src: (LAYER_INVALID), dest: (LAYER_INVALID), length: 0x00 (0)
srb1_cfg: -
    dummy_parameter: False
  srb0_present: True, srb1_present: False
rlc_cfg: -
msg_hdr: -
    message_id: 0x00 (0), src: (LAYER_INVALID), dest: (LAYER_INVALID), length: 0x00 (0)
srb1_cfg: -
rlc_cfg: -
ul: -
        poll_retransmit_timer: 0x00 (0), max_retx_thres: 0x00 (0)
dl: -
        enable_sr_sngap: False
    srb_id: (L2_SRB0), action_type: (RLC_ACTION_SETUP)
  srb0_present: True, srb1_present: False
mac_cfg: -
header: -
    message_id: 0x00 (0), src: (LAYER_INVALID), dest: (LAYER_INVALID), length: 0x00 (0)
config: -
mac_sch_cfg: -
      periodicBsrTimer: 0x00 (0), retxBSRTimer: 0x00 (0)
drb_lchs0: -
      lcid: 0x00 (0), priority: 0x00 (0), sr_prohibit_enabled: False, lch_sr_mask: 0x00 (0)
```

以上信息的参数配置主要来源于 SIB2 及 rrcConnectSetup 消息中，可通过比对 ASN 排查 RRC 层下发参数是否有误；也可快速获取 L1L2 的参数配置信息；

4.2.5. 查看当前 UE 支持的 BAND 信息

关键字：LL1_NVCONFIG_SUPPORTED_BANDS

```
11930> 2016/10/14 10:34:28.761 - LL1_NVCONFIG_SUPPORTED_BANDS: (00:02:10.6987250):  => : num_supported_bands: 0x01 (1), band_ids0: 0x08 (8)
num_supported_bands: 0x01 (1), band_ids0: 0x08 (8)
```

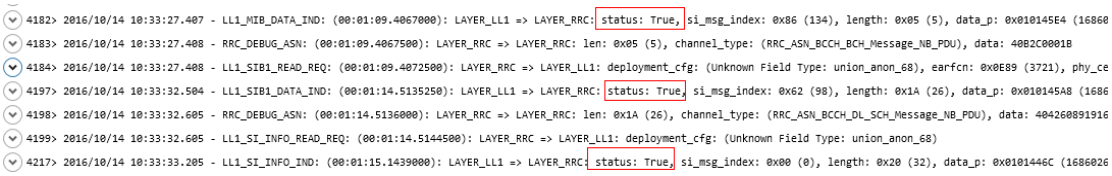
4.2.6. 系统消息的接收

关键字：

LL1_MIB_DATA_IND

LL1_SIB1_DATA_IND

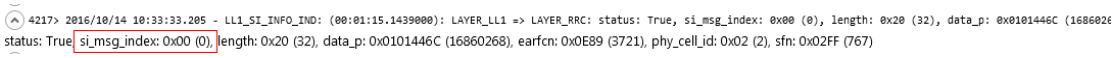
LL1_SI_INFO_IND



满足小区驻留的必要条件是正确接收到 MIB，SIB1，SIB2；可通过查看 log 中的 status 是否为 True 确认接收是否正确；一旦 MIB、SIB1 接收出现错误，RRC 会再重新接收一次，如果仍然错误会将该小区加入 Bar 列表，即不再驻留该小区。Bar 时常为 300s

SI 接收错误，则直接无视该 SI_INFO_IND

特别注意 LL1_SI_INFO_IND 消息：



可通过比对 SIB1 的 ASN 中的 SI 调度信息及 LL1_SI_INFO_IND 消息的 si_msg_index，确认该消息接收的是哪个 SI；

5. L2 问题定位指导

5.1. 目的

旨在指导测试和一线人员能够快速确定为 L2 问题。

1. 了解你的工具

介绍了 UE viewer 中与 L2 相关的主要 log，建议再定位问题前熟悉这些 log 并知道他们所包含的意义。这些 log 就是定位 L2 问题的眼和耳，了解他们并且正确地选择合适 log 可以帮助你快速解释现象。

2. 知道什么是对的

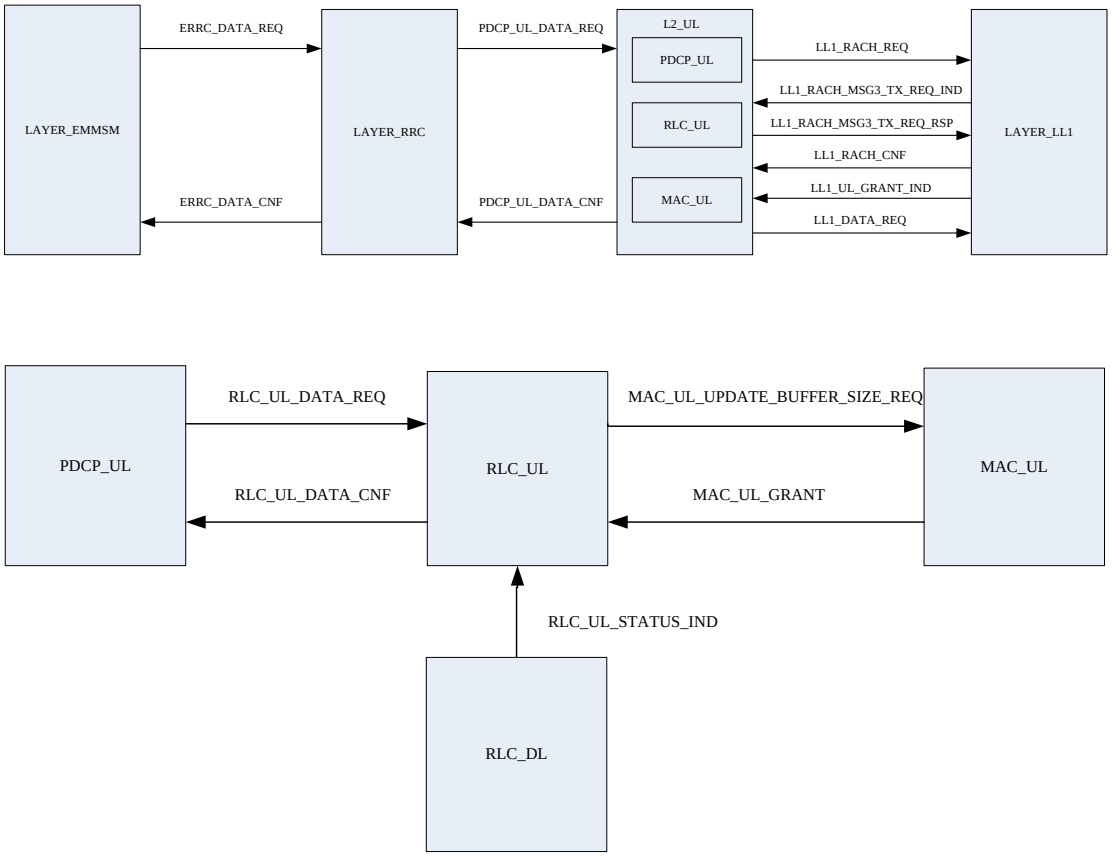
分别介绍了上行和下行数传的正常流程。一旦你知道了什么是正常流程，那么就能快速地察觉出哪里产生了异常。否则，当问题发生了你可能都未曾察觉。

3. 分而治之

介绍定位的问题时分析的步骤。该步骤唯一目的就是缩小问题的所搜范围，向目标逼近，最终确定目标范围。

5.2. 上行数传

5.2.1. 上行数传主要消息



5.2.1.1. PDCP_UL_DATA_REQ

```
2016/9/28 10:05:12.887> PDCP_UL_DATA_REQ: (00:25:24.8324250): LAYER_RRC => LAYER_PDCP_UL: pdu: 0x010131BC (
pdu: 0x010131BC (16855484), pdulen: 0xC6 (198), rb_id: (L2_SRB1), proc_id: 0x47 (71)
```

1. 说明：RRC层发送上行PDCP PDU至PDCP UL层的消息。
2. 等级：VERBOSE
3. 包含参数：

   pdu: PDCP PDU 的指针

   pdu_len: PDCP PDU 的长度

   rb_id:承载，有以下两种

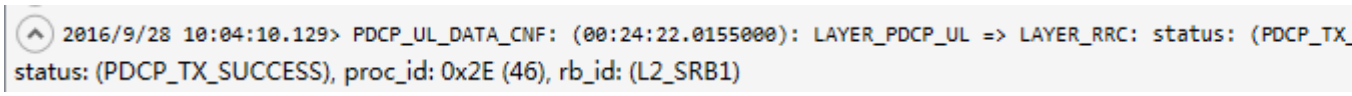
   L2_SRB1：SRB1，使用 AM 模式进行传输，进入 CONNECTED 连接态后均采用该方式进行上行传输。

   L2_SRB0：SRB0，使用 TM 模式进行传输，只有在 IDLE 空闲态下发送 RRC_CONNECT_REQ

才会使用该方式进行上行传输。

proc_id：process id，在 PDCP_UL_DATA_CNF 中有相同字段，用于其指示上行传输成功的 PDCP PDU。

5.2.1.2. PDCP_UL_DATA_CNF



1. 说明：PDCP UL发送至RRC层指示对应的PDCP PDU发送状态
2. 等级：VERBOSE
3. 包含参数：

status: PDCP PDU 发送状态

PDCP_TX_SUCCESS：发送成功

PDCP_MAX_RLC_RETRX_ERR：重传超过最大次数后失败

PDCP_RACH_FAILURE：重同步随机接入失败导致的传输失败

proc_id: PDCP PDU 的 process id，与 PDCP_UL_DATA_REQ 中的 proc_id 相对应

rb_id:承载，有以下两种

L2_SRB1：SRB1，使用 AM 模式进行传输，进入 CONNECTED 连接态后均采用该方式进行上行传输。

L2_SRB0：SRB0，使用 TM 模式进行传输，只有在 IDLE 空闲态下发送 RRC_CONNECT_REQ 才会使用该方式进行上行传输。

proc_id：process id，在 PDCP_UL_DATA_CNF 中有相同字段，用于其指示上行传输成功的 PDCP PDU。

5.2.1.3. LL1_RACH_REQ

1. 说明：MAC UL层向LL1发起RACH请求。
2. 等级：NORMAL
3. 包含参数

initiator:发起 RACH 的实体

LL1_RACH_MAC_INITIATED：MAC 层发起的随机接入，通常为重同步。

LL1_RACH_L3_INITIATED：L3 层发起的随机接入，初始随机接入。

LL1_RACH_PDCCH_ORDER_INITIATED：PDCCH ORDER 发起的随机接入。

5.2.1.4. LL1_RACH_MSG3_TX_REQ_IND

```
2016/9/28 10:04:38.375> LL1_RACH_MSG3_TX_REQ_IND: (00:24:50.3002250): LAYER_LL1 => LAYER_MAC_UL: tb_size: 0
tb_size: 0x0B (11), crnti: 0xF011 (61457), ph: (LL1_PH3)
```

1. 说明：LL1层请求MAC UL层组包MSG3 MAC PDU
2. 等级：NORMAL
3. 包含参数

tb_size: MSG3 的 TB size

crnti: 如果是 L3 发起的随机接入，则该 crnti 为 T-CRNTI，MAC UL 层会保存该值，并在竞争决议成功之后作为 C-RNTI；如果是 PDCCH ORDER 或者 MAC 层发起的随机接入，则该 crnti 为 C-RNTI，MAC UL 会忽略该值，因为 MAC UL 已经保存。

ph: power headeroom 的等级，用于指示在 MAC PDU 中 PHR control element 的取值。

LL1_PH0：等级 0；
LL1_PH1：等级 1
LL1_PH2：等级 2
LL1_PH3：等级 3
LL1_PHR_NOT_USED：不在 MAC PDU 中包含 PHR control element。

5.2.1.5. LL1_RACH_MSG3_TX_REQ_RSP

```
2016/9/28 10:44:36.819> LL1_RACH_MSG3_TX_REQ_RSP: (00:39:23.1754750): LAYER_MAC_UL => LAYER_LL1: data: 0x01
data: 0x010120F7 (16851191), free: 0x010120E0 (16851168)
```

1. 说明：MAC UL层发送给LL1的MSG3 MAC PDU
2. 等级：NORMAL
3. 包含参数

data: MSG3 的数据指针
free:释放指针

5.2.1.6. LL1_RACH_CNF

```
2016/9/28 10:44:37.015> LL1_RACH_CNF: (00:39:23.2960500): LAYER_LL1 => LAYER_MAC_UL: cause: (LL1_RACH_SUCCE
cause: (LL1_RACH_SUCCESS)
```

1. 说明：LL1层完成RACH流程后发给MAC UL层告知结果
2. 等级：NORMAL
3. 包含参数：

cause: RACH 完成的结果，取值如下：

LL1_RACH_SUCCESS

LL1_RACH_ERROR_MAX_NUM_PREAMBLE_ATTEMPTS

LL1_RACH_CONTENTION_RESOLUTION_MISMATCH:

LL1_RACH_ERROR

LL1_RACH_CANCEL

LL1_RACH_INTERRUPTED_BY_RRC_RELEASE

LL1_RACH_INTERRUPTED_BY_MAC_RESET

5.2.1.7. LL1_UL_GRANT

```
2016/9/28 10:44:11.394> LL1_UL_GRANT_IND: (00:38:57.6929250): LAYER_LL1 => LAYER_MAC_UL: tb_size: 0x11 (17)
tb_size: 0x11 (17)
```

1. 说明：LL1层发送给MAC UL层的UL grant，MAC UL收到该消息后进行MAC PDU的组包
2. 等级：NORMAL
3. 包含参数：

   tb_size: UL grant 的 TB size

5.2.1.8. LL1_DATA_REQ

```
2016/9/28 10:46:44.984> LL1_DATA_REQ: (00:41:31.3138000): LAYER_MAC_UL => LAYER_LL1: data: 0x0100F19A (1683
data: 0x0100F19A (16839066), free: 0x0100F180 (16839040)
```

1. 说明：MAC UL层完成MAC PDU的组包之后将之发送给LL1
2. 等级：NORMAL
3. 包含参数：

   data: MSG3 的数据指针

   free:释放指针

5.2.1.9. RLC_UL_DATA_REQ

同 PDCP_UL_DATA_REQ。

5.2.1.10. RLC_UL_DATA_CNF

同 PDCP_UL_DATA_CNF。

5.2.1.11. MAC_UL_UPDATE_BUFFER_SIZE_REQ

```
2016/9/28 10:05:20.065> MAC_UL_UPDATE_BUFFER_SIZE_REQ: (00:25:32.0308500): LAYER_RLC_UL => LAYER_MAC_UL: rb_id: (L2_SRB1)
rb_id: (L2_SRB1)
data_len: -
  srb0_datalen: 0x00 (0)
srb1_datalen: -
    tx_len: 0x33 (51), tx_header_len: 0x04 (4), retx_bs_num: 0x00 (0), retx_bs_p: 0x00 (0), total_retx_len: 0x00 (0), sr_len: 0x02 (2), total_len: 0x35 (53)
  drb_datalen: 0x00 (0)
```

1. 说明：RLC层更新RLC BUFFER状态
2. 等级： VERBOSE
3. 包含参数：

total_len:上行待传输的数据的总字节数；
sr_len:上行待传输的状态报告的字节数；
total_retx_len:待上行重传数据的总字节数；
retx_bs_p：待上行重传的各个 RLC PDU 的长度数组；
retx_bs_num：待上行重传的 RLC PDU 的个数；
tx_header_len:在包含所有的上行新传数据情况下的 RLC PDU header 的长度；
tx_len:待新传的上行数据长度。

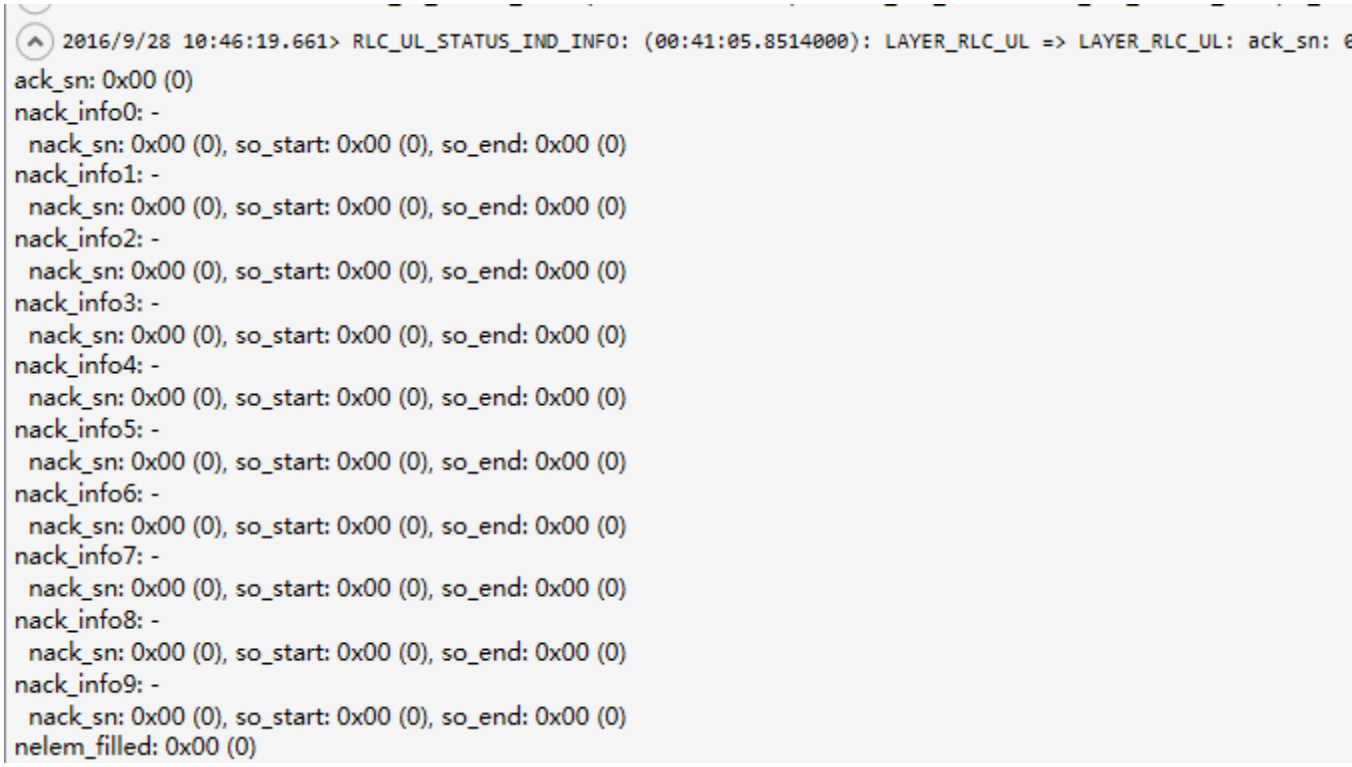
5.2.1.12. MAC_UL_GRANT

```
2016/9/28 10:44:36.818> MAC_UL_GRANT_IND: (00:39:23.1745500): LAYER_MAC_UL => LAYER_RLC_UL: ptr: 0x010120FD
ptr: 0x010120FD (16851197), size: 0x05 (5), rsize: 0x00 (0), rb_id: (L2_SRB1)
```

1. 说明：MAC UL层请求RLC UL层进行RLC PDU组包
2. 等级： NORMAL
3. 包含参数：

   ptr:请求的 RLC PDU 的起始位置；
   size:RLC PDU 的最大 size；
   rsize:未使用，请忽略；
   rb_id：承载， L2_SRB1 或者 L2_SRB0；

5.2.1.13. RLC_UL_STATUS_IND_INFO



1. 说明：RLC DL接收到指示上行数据的RLC SR状态报告后转发给RLC UL，RLC UL根据其中信息释放RLC PDU，并指示PDCP UL发送结果。
2. 等级： NORMAL
3. 包含参数：

ack_sn: ACK SN 表示所以小于该值的 SN RLC PDU，除了下面 nack_sn 之外的 RLC PDU 都确认被接收到了；

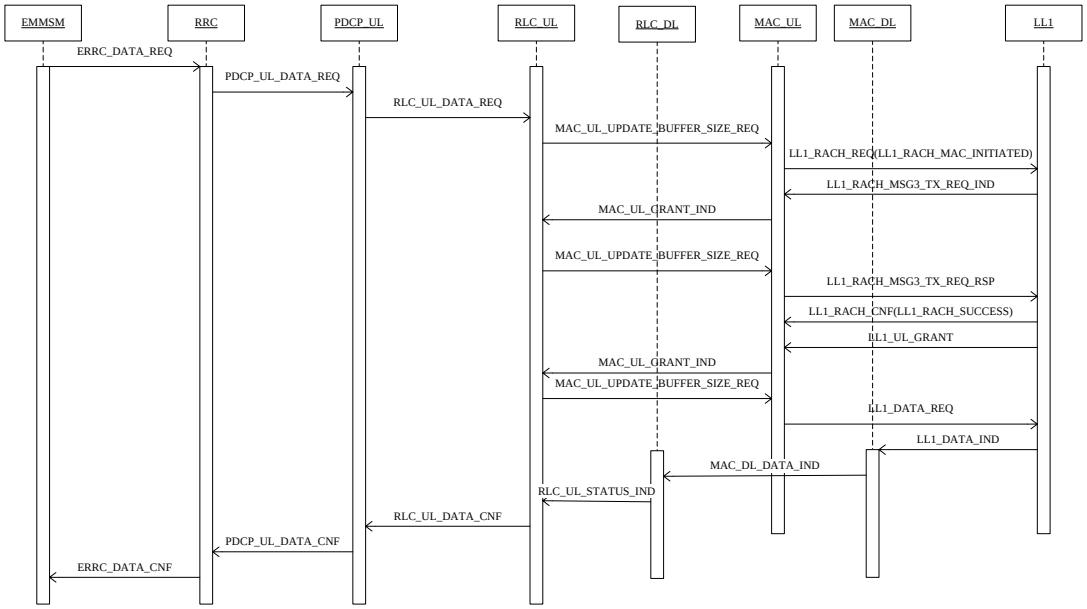
nack_sn: NACK SN 表示未被成功接收到的 SN；

so_start:未成功接收的 SN 中丢失分片的起始字节；

so_end: 未成功接收的 SN 中丢失分片的结束字节；

nelem_filled: nack element 的数量，表示 RLC 丢包的数量

5.3.上行正常流程



1. ERRC_DATA_REQ

   EMMSM 将上行 NAS 消息发送至 RRC 层；

2. PDCP_UL_DATA_REQ

   RRC 层添加头并将上行数据发送至 PDCP UL 层；

3. RLC_UL_DATA_REQ

   PDCP UL 层收到的上行数据作为一个 PDCP PDU 发送至 RLC UL 层赋值一个 proc id，RLC UL 层将数据保存下来，并记录 proc id。

4. MAC_UL_UPDATE_BUFFER_SIZE_REQ

   RLC UL 通知 MAC UL 层更新 RLC BUFFER。由于之前已无上行数据 RLC BUFFER 为空，所以会触发一个 regular BSR。

5. LL1_RACH_REQ(LL1_RACH_MAC_INITIATED)

   在 regular BSR 触发后，并且是由于由上行数据要发送。那么如果没有配置 SrProhibitTimer，MAC UL 则会立即向 LL1 发送 RACH 请求。

6. LL1_RACH_MSG3_TX_REQ_IND

   LL1 收到 RACH_REQ 之后就会立即去执行 RACH 流程。LL1 在成功接收到 RAR 之后拿到 UL grant 就会发送 LL1_RACH_MSG3_TX_REQ_IND 请求 MAC 层发送 MSG3。MAC UL 层收到该消息后就会开始组包 MAC PDU，首先 MAC 层会将 CRNTI 以及 BSR 等 control elements 组进 MAC PDU。

7. MAC_UL_GRANT_IND

   MAC UL 层请求 RLC UL 层组包一个 RLC PDU，通常在 MSG3 中去掉 BSR 以及 CRNTI 之后 MAC 层只能提供 5 个字节给 RLC 进行组包，所以通常在此 RLC 会对上行数据进行第一个分

片，此时 BSR 会指示剩余的数据量，BTS 根据此值提供上行调度。

8. MAC_UL_UPDATE_BUFFER_SIZE_REQ

RLC 在完成 RLC PDU 组包之后会更新 RLC BUFFER。

9. LL1_RACH_MSG3_TX_REQ_RSP

MAC 拿到 RLC PDU 后完成 MAC PDU 的组包，则将 MSG3 发送至 LL1。由 LL1 在 PUSCH 上传输至 BTS。

10. LL1_RACH_CNF(SUCCESS)

LL1 在 MSG3 之后收到 BTS 提供的以 C-RNTI 加扰的 UL grant，则成功完成 RACH 流程，发送 LL1_RACH_CNF(SUCCESS)通知 MAC 层。

11. LL1_UL_GRANT

LL1 接受到新的 UL grant 之后告知 MAC 层，请求进行 MAC PDU 组包；

12. MAC_UL_GRANT_IND

MAC UL 层拿到上行授权之后会请求 RLC UL 层组包 RLC PDU，一个上行 PDCP 可能会被分片为若干的 RLC PDU。

13. MAC_UL_UPDATE_BUFFER_SIZE_REQ

RLC UL 完成 RLC PDU 组包之后通知 MAC UL 层更新 RLC BUFFER。

14. LL1_DATA_REQ

MAC UL 层将组好的 MAC PDU 发送至 LL1，LL1 通过 PUSCH 将其发送至 BTS。

步骤 11 至 14 可能会重复多次，根据该上行 PDCP PDU 大小决定，在最后一次的流程中，MAC_UL_UPDATE_BUFFER_SIZE_REQ 消息中的 total_len 字段应该被更新为 0，同时再最后一个 RLC PDU 中 poll 会被设置为 ture，MAC PDU 中 BSR 会等于 0.

15. LL1_DATA_IND

LL1 把接收到的下行 MAC PDU 发送至 MAC DL 层处理。

当该 PDCP PDU 的所有 RLC PDU 发送完成之后，后续会接收到一个 RLC SR 状态报告的下行数据包。

16. MAC_DL_DATA_IND

MAC DL 层从下行的 MAC PDU 中解析出 RLC PDU 发送至 RLC DL 层进行处理。

17. RLC_UL_STATUS_IND

RLC DL 解析出 RLC PDU 为一个 SR 状态报告。RLC DL 将该 SR 状态中的信息（包括 ACK_SN 以及 NACK elements）发送至 RLC UL 进行处理。

18. RLC_UL_DATA_CNF

RLC UL 收到 SR 状态信息，将已经发送完的 RLC PDU 释放掉，并通知 PDCP UL 由 proc id 标识的 PDCP PDU 已经发送成功。

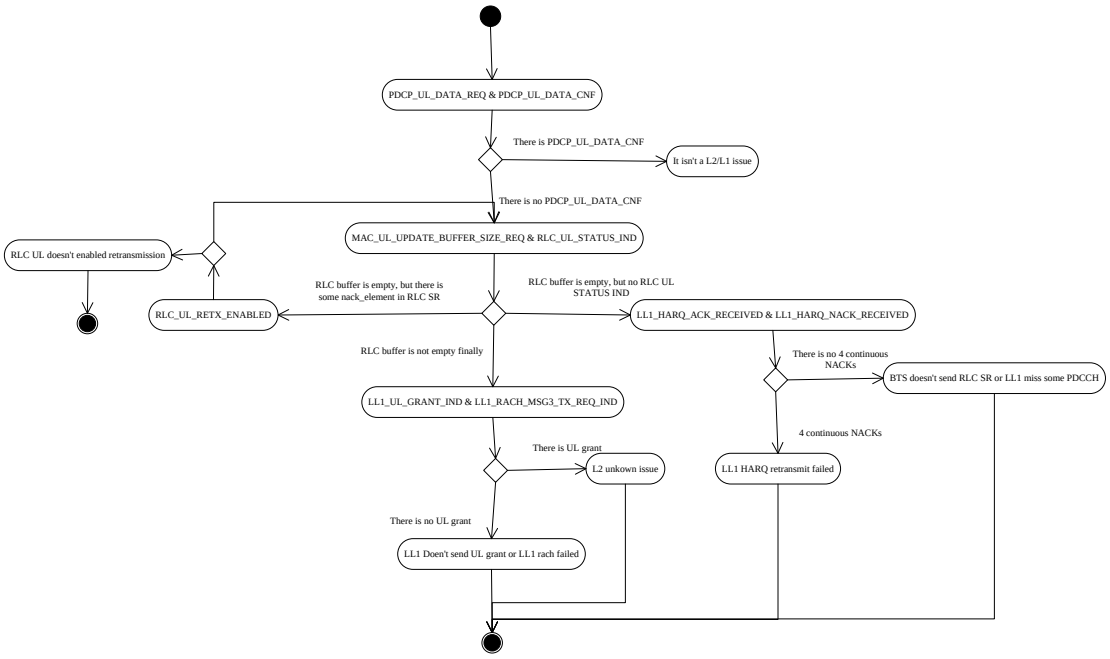
19. PDCP_UL_DATA_CNF

PDCP UL 发送消息通知 RRC 层数据发送成功。

20. ERRC_DATA_CNF

RRC 层发送消息通知 EMMSM 层数据发送成功。

5.3.1. 上行定位步骤



步骤 1：

过滤 log：

PDCP_UL_DATA_REQ

PDCP_UL_DATA_CNF

观察：

观察每一个 PDCP_UL_DATA_REQ 是否都有与之对应的 PDCP_UL_DATA_CNF.

正常期望：

每一个 PDCP_UL_DATA_REQ 都有对应的 PDCP_UL_DATA_CNF，则说明所以上层发往 L2 的上行数据包都发送成功。

异常情况:

1. 发现某PDCP_UL_DATA_REQ没有对应的PDCP_UL_DATA_CNF，则说明该 PDCP_UL_DATA_REQ对应的上行数据包没有发送成功。则进入步骤2.

步骤 2：

过滤 log：

MAC_UL_UPDATE_BUFFER_SIZE_REQ

RLC_UL_STATUS_IND

观察：

观察 MAC_UL_UPDATE_BUFFER_SIZE_REQ 中的数据是否都已经成功发送，并观察接下来的 RLC_UL_STATUS_IND 是否指示 BTS 都已经接收成功。

正常期望：

MAC_UL_UPDATE_BUFFER_SIZE_REQ 中的 total_len 是逐渐减少为 0，直至 total_len 为 0 为止，接下来的有 RLC_UL_STATUS_IND，并且当中 nelem_filled 应该为 0 则说明已发送的上行数据均被 BTS 确认接收成功。

异常：

1. 最后MAC_UL_UPDATE_BUFFER_SIZE_REQ中total_len始终不为0，并没有逐渐减少，则继续步骤3；
2. MAC_UL_UPDATE_BUFFER_SIZE_REQ中total_len最终减为0，并且后续也有RLC_UL_STATUS_IND_INFO，但是发现其中的nelem_filled，则继续步骤4；
3. MAC_UL_UPDATE_BUFFER_SIZE_REQ中total_len最终减为0，但是后续没有发现RLC_UL_STATUS_IND_INFO，则继续步骤5；

步骤 3：

过滤 log:

LL1_HARQ_ACK_SENT

LL1_HARQ_NACK_SENT

观察：

观察 LL1 是否有 HARQ 重传失败导致 RLC 丢包。

正常期望：

无 LL1_HARQ_NACK_SENT,或者连续的 LL1_HARQ_NACK_SENT 少于 4 次。则说明无 MAC PDU 丢包。

异常：

1. 有连续4次的LL1_HARQ_NACK_SENT，则说明该MAC PDU丢包。

步骤 4：

过滤 log:

RLC_UL_RETX_ENABLED

观察：

观察 RLC 是否使能了上行 RLC PDU 的重传；

正常：

能够看到 RLC_UL_RETX_ENABLED 说明 RLC UL 使能了重传，则继续步骤 2，观察重传数据是否发送成功。

异常：

未能看到 RLC_UL_RETX_ENABLED 说明 RLC UL 没有使能重传。

步骤 5：

过滤 log:

LL1_UL_GRANT

LL1_RACH_REQ

LL1_RACH_MSG3_TX_REQ

LL1_RACH_CNF

观察：

观察在发送上行数据的过程中，LL1 是否有 RACH 失败或者未能收到 UL grant.

正常期望：

如果有 LL1_RACH_REQ，则期望看到 LL1_RACH_CNF(LL1_RACH_SUCCESS)，说明

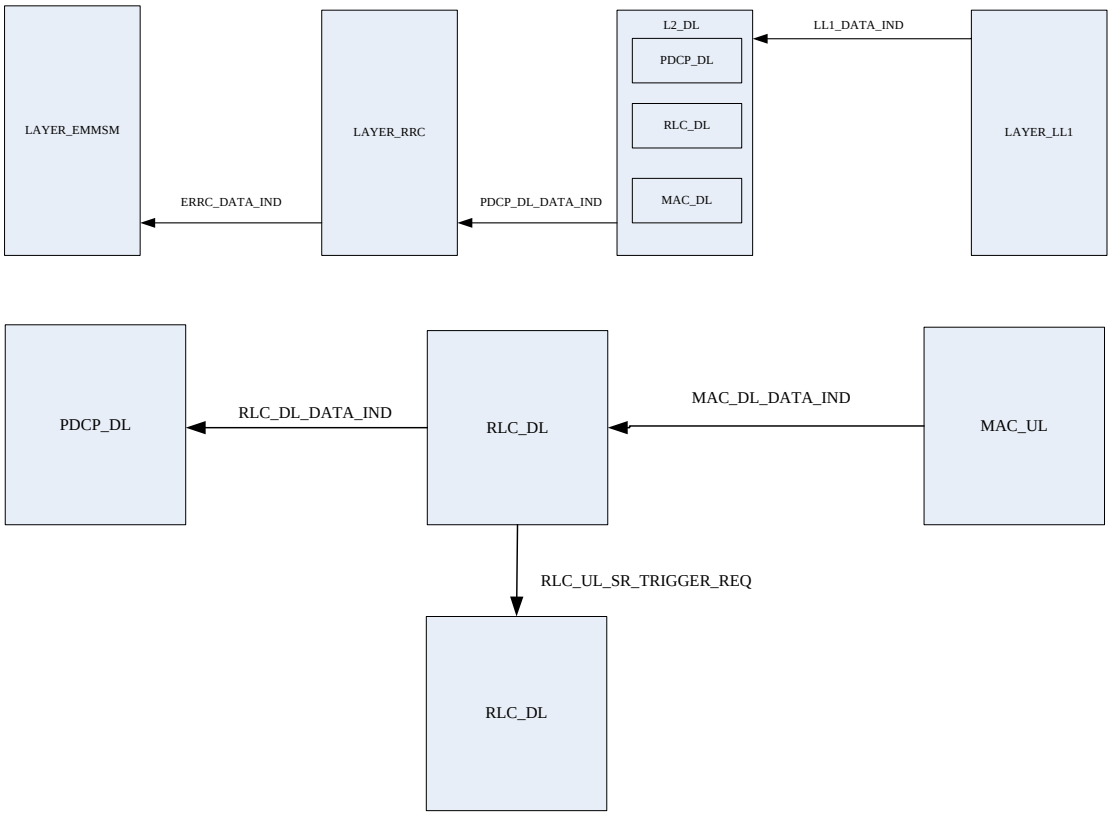
RACH 成功。如果后续还能观察到 LL1_UL_GRANT 则说明 LL1 能够持续拿到 UL grant 直到 MAC_UL_UPDATE_BUFFER_SIZE_REQ 中 total_len 为 0.那么一切正常。

异常：

1. 有LL1_RACH_REQ，但是无法观察到LL1_RACH_CNF(LL1_RACH_SUCCESS)，则说明LL1 RACH失败。
2. MAC_UL_UPDATE_BUFFER_SIZE_REQ中的total_len不为0，但是后续又看不到LL1_UL_GRANT，则说明LL1没有上报UL grant。

5.4. 下行数传

5.4.1. 下行数传主要消息



5.4.1.1. PDCP_DL_DATA_IND

```
2016/9/26 17:26:50.567> PDCP_DL_DATA_IND: (00:14:36.2311250): LAYER_PDCP_DL => LAYER_RRC: pdcppdu: 0x0100C810 (16828432), pdcppdu_len: 0x014E (334), rb_id: (L2_SRB1)
pdcppdu: 0x0100C810 (16828432), pdcppdu_len: 0x014E (334), rb_id: (L2_SRB1)
```

1. 说明：PDCP DL 层成功接收到下行PDCP PDU后发送至RRC层

2. 等级： NORMAL

3. 包含参数：

Pdcppdu:下行接收到的 PDCP PDU 的指针；
Pdcppdu_len：下行接收到的 PDCP PDU 的长度；
rb_id：承载。

5.4.1.2. RLC_DL_DATA_IND

同 PDCP_DL_DATA_IND

5.4.1.3. MAC_DL_DATA_IND

```
2016/9/26 17:26:07.853> MAC_DL_DATA_IND: (00:13:53.5869250): LAYER_MAC_DL => LAYER_RLC_DL: pdu_ptr: 0x0100F
pdu_ptr: 0x0100F5E8 (16840168), rb_id: (L2_SRB1), pdu_len: 0x51 (81)
```

1. 说明：MAC DL层接收到MAC PDU后将解析出的RLC PDU发送至RLC DL层处理

2. 等级： NORMAL

3. 包含参数：

Pdu_ptr:pdu 的指针；
pdu_len：PDU 的长度；
rb_id：承载。

5.4.1.4. RLC_UL_SR_TRIGGER_REQ

```
2016/9/26 17:26:31.461> RLC_UL_SR_TRIGGER_REQ: (00:14:17.1293750): LAYER_RLC_DL => LAYER_RLC_UL: rb_id: (L2
rb_id: (L2_SRB1)
sr_info: -
  ack_sn: 0x01C4 (452), nack_info_list: 0x00 (0), num_nack_elem: 0x00 (0)
```

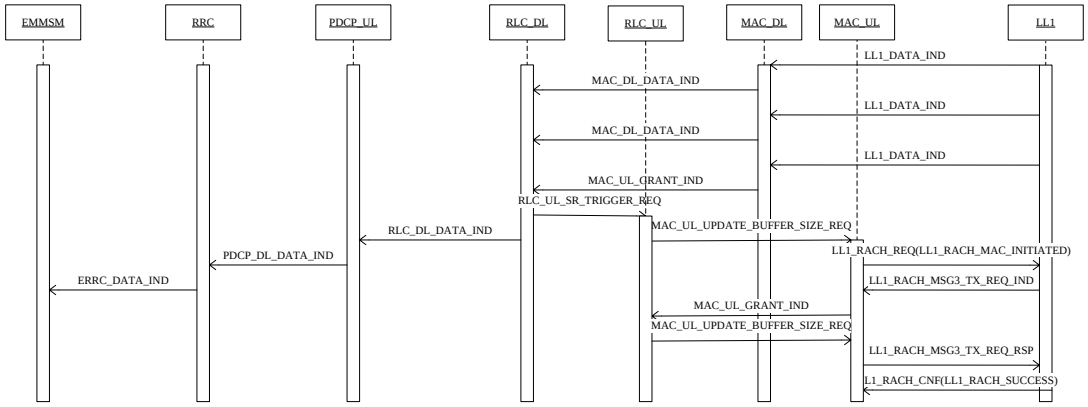
1. 说明：RLC DL向RLC UL发送触发发送针对下行RLC PDU的SR状态报告的消息

2. 等级： NORMAL

3. 包含参数：

ack_sn: ACK SN 表示所以小于该值的 SN RLC PDU，除了下面 nack_sn 之外的 RLC PDU 都确认被接收到了；
nack_info_list:nack elements 数值指针
num_nack_elem: nack element 的数量，表示 RLC 丢包的数量

5.4.2. 下行数传正常流程



1. LL1_DATA_IND

   LL1 把接收到的下行 MAC PDU 发送至 MAC DL 层处理。

2. MAC_DL_DATA_IND

   MAC DL 层将从 MAC PDU 从解析出来的 RLC PDU 发送至 RLC DL 层进行处理。
   通常步骤 1~2 会重复若干次，根据下行 PDCP PDU 大小决定会有多少个分片。

3. RLC_UL_SR_TRIGGER_REQ

   在最后一个分片的 RLC PDU 中 poll bit 会被设置为 true，要求 UE 发送一个 SR 状态报告给 BTS。因此，RLC DL 会发送 RLC_UL_SR_TRIGGER_REQ 至 RLC UL 要求其发送一个上行的 SR 状态报告。

4. RLC_DL_DATA_IND

   RLC DL 层将接收到的分片 RLC PDU 进行串联，组成完整的下行 PDCP PDU 发送至 PDCP DL 层。

5. MAC_UL_UPDATE_BUFFER_SIZE_REQ

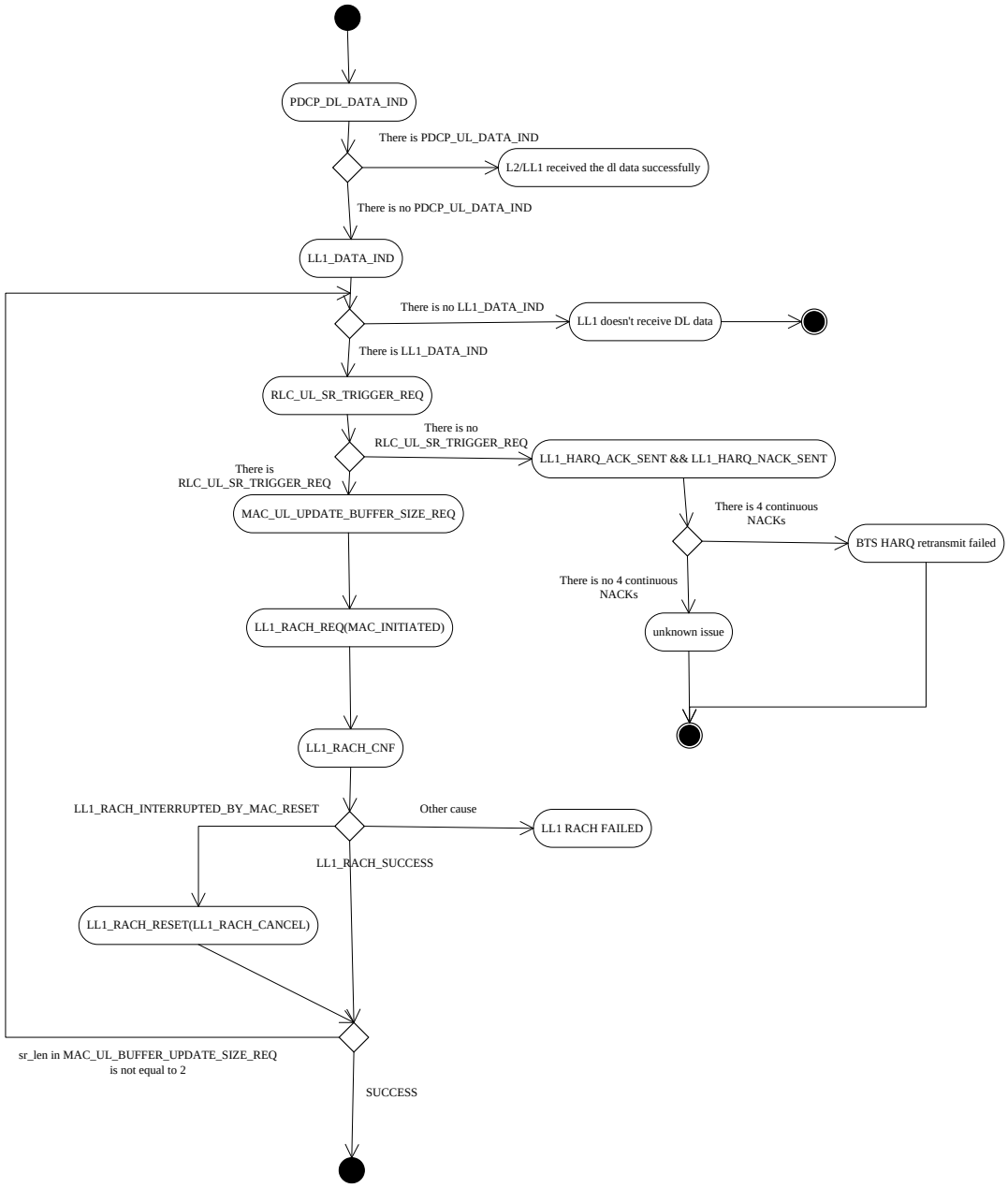
   当 RLC UL 接受到 SR 触发请求之后，会更新 RLC BUFFER 并通知 MAC UL。由于之前的 MAC BUFFER 为空，所以会触发一个 regular BSR。

6. LL1_RACH_REQ(LL1_RACH_MAC_INITIATED)

   由于 regular BSR 被触发，且 RLC buffer 中有等待传输的数据。在 SrProhibitTimer 没有配置的情况下，MAC UL 会立即向 LL1 发送 RACH_REQ。

   后续的随机接入流程中，在接收到 RAR 之前，有可能 BTS 会主动提供上行 UL grant，在 LL1 收到之后会发送 LL1_UL_GRANT 至 MAC UL 层，MAC UL 则会发送 LL1_RACH_RESET(CANCEL) 至 LL1 层取消 RACH 流程。

5.4.3. 下行定位步骤



步骤 1：

过滤 log：

PDCP_DL_DATA_IND

观察：

观察是否有成功接收下行的 PDCP PDU

正常期望：

每一个 PDCP_DL_DATA_IND 都表示 L2/LL1 成功接收到一个完整的 PDCP PDU。

异常情况:

1. 在期望有下行数据的时候，没有发现有PDCP_DL_DATA_IND，则说明L2/LL1

接收下行数据存在问题，则继续步骤2.

步骤 2：

过滤 log：

LL1_DL_DATA_IND

观察：

观察 LL1 是否有接收到下行的 MAC PDU

正常期望：

LL1_DL_DATA_IND 表示 LL1 下行接收到了下行的 MAC PDU，继续步骤 3；

异常情况:

1. 没有LL1_DL_DATA_IND则表示LL1没有接收到任何的下行MAC PDU，则需要LL1继续定位。

步骤 3：

过滤 log：

RLC_UL_SR_TRIGGER_REQ

观察：

RLC 是否触发发送 RLC SR 至 BTS。

正常期望：

能够观察到 RLC_UL_SR_TRIGGER_REQ，并且当中的 nack_element 等于 0，说明 RLC DL 已经完整接收到一个下行 PDCP PDU。下一步需要发起随机接入来发送 RLC SR 至 BTS。

异常情况:

1. 没有观察到RLC_UL_SR_TRIGGER_REQ，则说明UE丢失掉了该下行的PDCP PDU的最后一个分片，继续进行步骤4。
2. 观察到RLC_UL_SR_TRIGGER_REQ，但是其中的num_nack_element不等于0，则说明有数据需要重传。继续步骤5.

步骤 4：

过滤 log：

LL1_HARQ_ACK_SENT

LL1_HARQ_NACK_SENT

观察：

LL1 是否有 HARQ 丢包导致触发 UE 发送 RLC SR。

正常期望：

能够观察到 4 个连续的 LL1_HARQ_NACK_SENT，说明是 LL1 的 HARQ 丢包导致的 RLC 没有发送 SR。

异常情况:

1. 没有观察到4个连续的LL1_HARQ_NACK_SENTR，说明是未知原因导致的RLC UL没有发送SR。

步骤 5：

过滤 log：

MAC_UL_UPDATE_BUFFER_SIZE_REQ

LL1_RACH_REQ

LL1_RACH_CNF

观察：

观察 RLC SR 状态报告是否成功发送至 BTS

正常期望：

能够顺序观察到 MAC_UL_UPDATE_BUFFER_SIEZE_REQ 、 LL1_RACH_REQ 以及 LL1_RACH_CNF 三条 log，并且 LL1_RACH_CN 中的 cause 为 LL1_RACH_SUCCESS 或者 LL1_RACH_INTERRUPTED_BY_MAC_RESET.前者表示 RACH 成功，后者表示 RACH 由于拿到 BTS 上行的 UL 而被取消掉了。如果观察到 MAC_UL_UPDATE_BUFFER_SIZE_REQ 中的 sr_len 大于 2，则继续返回至步骤 2 等待接收 BTS 重传下行数据。如果 sr_len 等于 2，则成功发送 SR 并且无丢失 RLC PDU。

异常情况:

1. 没有观察到LL1_RACH_CNF，或者观察到其为其他cause，其说明LL1 RACH失败。

6. L1 问题定位指导

6.1. 随机接入失败问题

6.1.1. 随机接入正常流程

随机接入有两种流程：初始接入和重同步。注意初始接入一共有 5 步，msg1 到 msg5；重同步一共有 3 步，msg1 到 msg3。这里只介绍初始接入 msg1-msg4，重同步与初始接入前三步的 log 相似。

下图为初始接入成功的 LOG，过滤关键字：

ll1_rach_req

ll1_rach_cnf

nprach_req

npdcch_search

npdsch_req

npdsch_cnf

dci

rar

msg3

contention

npusch_req

npusch_cnf

```
115145> 2016/12/22 19:07:07.823 - LL1_RACH_REQ: (00:14:10.155792): LAYER_MAC_UL => LAYER_LL1: initiator: (LL1_RACH_L3_INITIATED)
115163> 2016/12/22 19:07:08.103 - DSP_NPRACH_REQ: (00:14:10.453308): LAYER_LL1 => LAYER_DSP: cell_id: 0x14 (20), selectedStartSubcarrier: 0x02 (2), repetition: 0x02 (2), start_subframe:
115180> 2016/12/22 19:07:08.144 - DSP_NPDCCH_SEARCH_REQ: (00:14:10.503967): LAYER_LL1 => LAYER_DSP: samples_offset_ms: 0x00 (0), max_samples: 0x00 (0), search_space_duration: 0x08 (8),
115185> 2016/12/22 19:07:08.144 - DSP_NPDCCH_SEARCH_CNF: (00:14:10.516693): LAYER_DSP => LAYER_LL1: quality: 0x0C80 (3200), pdcch_timing: 0x00 (0), pdcch_repetitions: 0x01 (1), pdcch_el
115194> 2016/12/22 19:07:08.163 - LL1_DCI_FORMAT_N1_NORMAL: (00:14:10.518035): LAYER_LL1 => LAYER_LL1: dci_n1: -
115198> 2016/12/22 19:07:08.163 - DSP_NPDSCH_REQ: (00:14:10.518463): LAYER_LL1 => LAYER_DSP: samples_offset_ms: 0, max_samples: 0, start_subframe: 1317, cell_id: 20, crnti: 33, repetiti
115203> 2016/12/22 19:07:08.163 - DSP_NPDSCH_CNF: (00:14:10.522430): LAYER_DSP => LAYER_LL1: quality: 0x0900 (2304), length: 0x12 (18), pdsch_valid_sf: 0x01 (1), pdsch_elapsed_sf: 0x01
115207> 2016/12/22 19:07:08.173 - LL1_LOG_DEBUG_VALUE: (00:14:10.522888): LAYER_LL1 => LAYER_LL1: value: 0x50 (80), tag: (LL1_DEBUG_TIMING_ADVANCE_FROM_RAR), line: 0x49 (73)
115209> 2016/12/22 19:07:08.173 - LL1_RAR_UL_GRANT: (00:14:10.523071): LAYER_LL1 => LAYER_LL1: rar_pdu: -
115213> 2016/12/22 19:07:08.173 - LL1_RACH_MSG3_TX_REQ_IND: (00:14:10.523529): LAYER_LL1 => LAYER_MAC_UL: tb_size: 0x08 (11), crnti: 0xB2C6 (45766), ph: (LL1_PH0)
115219> 2016/12/22 19:07:08.173 - LL1_RACH_MSG3_TX_REQ_RSP: (00:14:10.524291): LAYER_MAC_UL => LAYER_LL1: data: 0x010156A3 (16864931), free: 0x01015688 (16864904)
115222> 2016/12/22 19:07:08.173 - DSP_NPUSCH_REQ: (00:14:10.524536): LAYER_LL1 => LAYER_DSP: crnti: 0xB2C6 (45766), cell_id: 0x14 (20), repetition_number: 0x01 (1), start_subframe: 0x05
115234> 2016/12/22 19:07:08.233 - DSP_NPUSCH_CNF: (00:14:10.540954): LAYER_DSP => LAYER_LL1:
115246> 2016/12/22 19:07:08.233 - DSP_NPDCCH_SEARCH_REQ: (00:14:10.543151): LAYER_LL1 => LAYER_DSP: samples_offset_ms: 0x00 (0), max_samples: 0x00 (0), search_space_duration: 0x08 (8),
115251> 2016/12/22 19:07:08.234 - DSP_NPDCCH_SEARCH_CNF: (00:14:10.550781): LAYER_DSP => LAYER_LL1: quality: 0x0C80 (3200), pdcch_timing: 0x00 (0), pdcch_repetitions: 0x01 (1), pdcch_el
115259> 2016/12/22 19:07:08.234 - LL1_DCI_FORMAT_N1_NORMAL: (00:14:10.551971): LAYER_LL1 => LAYER_LL1: dci_n1: -
115263> 2016/12/22 19:07:08.234 - DSP_NPDSCH_REQ: (00:14:10.552398): LAYER_LL1 => LAYER_DSP: samples_offset_ms: 0, max_samples: 0, start_subframe: 1351, cell_id: 20, crnti: 45766, repet
115268> 2016/12/22 19:07:08.243 - DSP_NPDSCH_CNF: (00:14:10.558654): LAYER_DSP => LAYER_LL1: quality: 0x1080 (4224), length: 0x29 (41), pdsch_valid_sf: 0x02 (2), pdsch_elapsed_sf: 0x02
115276> 2016/12/22 19:07:08.243 - LL1_RACH_CONTENTION_RESOLUTION_IND: (00:14:10.560180): LAYER_LL1 => LAYER_MAC_DL: length: 0x29 (41), data: 0x01015E70 (16866928)
115279> 2016/12/22 19:07:08.243 - MAC_UL_RACH_CONTENTION_IND: (00:14:10.560791): LAYER_MAC_DL => LAYER_MAC_UL: contention_resolution_id0: 0x20 (32), contention_resolution_id1: 0x03 (3),
115281> 2016/12/22 19:07:08.243 - LL1_RACH_CONTENTION_RESOLUTION_RSP: (00:14:10.561187): LAYER_MAC_UL => LAYER_LL1: status: (LL1_RACH_SUCCESS)
115298> 2016/12/22 19:07:08.244 - LL1_RACH_STATS_IND: (00:14:10.576538): LAYER_LL1 => LAYER_LL1: rach_stats: -
115306> 2016/12/22 19:07:08.244 - LL1_RACH_CNF: (00:14:10.577453): LAYER_LL1 => LAYER_MAC_UL: cause: (LL1_RACH_SUCCESS)
```

图中红色方框标注即为 MSG1-MSG4，第一行 LOG 可以看到接入类型，最后一行 LOG 可以看到接入结果。这样初步可以看出问题出现在哪一步。

6.1.2. LOG 过滤

如果怀疑是随机接入失败，无论是初始随机接入还是，重同步的随机接入，先在 UElogviewer 中输入以下过滤信息：

全面版	简化版
Cell_select	*ll1_rach_req*
pusch	*ll1_rach_cnf*
pdsch	*nprach_req*
pdcch	*npdcch_search*
pucch	*npdsch_req*
prach	*npdsch_cnf*
dci	*dci*
asn	*rar*
ll1_rach_req	*msg3*
ll1_rach_cnf	*contention*
	npusch_req

6.1.3. 初步分析

分别找到如下部分 log，如果有哪一部分没有找到，或者跟列出问题相同，则可以初步定位到是哪一步 msg 错误与错误的信道。

注意初始接入一共有 5 步，msg1 到 msg5；重同步一共有 3 步，msg1 到 msg3。这里只介绍初始接入 msg1-msg4，重同步与初始接入前三步的 log 相似。

6.1.3.1. MSG1

```
LL1_RACH_REQ: (00:01:21.2069250): LAYER_MAC_UL => LAYER_LL1: initiator: (LL1_RACH_L3_INITIATED)
LL1_LOG_ECL_INFO: (00:01:21.2071500): LAYER_LL1 => LAYER_LL1: current_ecl: 0x00 (0), ecl_selected_by: (LL1_RACH_
LL1_NPRACH_START_TIME: (00:01:21.2074500): LAYER_LL1 => LAYER_LL1: hfn: 0x0391 (913), sfn: 0xC0 (192), sf: 0x08
LL1_LOG_ECL_INFO: (00:01:21.5141250): LAYER_LL1 => LAYER_LL1: current_ecl: 0x00 (0), ecl_selected_by: (LL1_RACH_
DSP_NPRACH_REQ: (00:01:21.5143000): LAYER_LL1 => LAYER_DSP: cell_id: 0x47 (71), selectedStartSubcarrier: 0x02 (2
DSP_NPRACH_CNF: (00:01:21.5629750): LAYER_DSP => LAYER_LL1:
```

从这里开始接入流程，这是一次层三发起的初始随机接入，可以看到 ll1 给 dsp 发出的 msg1 的发送请求与回复。

【重要】如果需要与基站校对 msg1 的各项参数（其余信道找对应的 req,如 dsp_pdsch_req），请点击上图中 DSP_NPRACH_REQ 消息，展开后可以看到所有参数。

```
DSP_NPRACH_REQ: (00:01:21.5143000): LAYER_LL1 => LAYER_DSP: cell_id: 0x47 (71), selectedStartSubcarrier: 0x02 (2), repetiti
tartSubcarrier: 0x02 (2), repetition: 0x02 (2), start_subframe: 0x0788 (1928), subcarrierAllocation: 0x0C (12), subcarrierOffset: 0x24 (3
```

【重要】如果要对与基站校对帧号（所有信道都一样），请根据上图中 START_TIME 这条消息来看，

```
hfn: 0x0391 (913), sfn: 0xC0 (192), sf: 0x08 (8), sf_from_now: 0x0159 (345)
```

由于 MSG1 是上行数据，无法看出基站是否收到，主要通过看是否收到 MSG2 前的 pdcch 来判断。

```
LL1_NPDCCH_START_TIME: (00:01:21.5640750): LAYER_LL1 => LAYER_LL1: hfn: 0x0391 (913), sfn: 0xC2 (194), sf: 0x04
DSP_NPDCCH_SEARCH_REQ: (00:01:21.5647750): LAYER_LL1 => LAYER_DSP: samples_offset_ms: 0x00 (0), max_samples: 0x0
DSP_NPDCCH_SEARCH_CNF: (00:01:21.5693000): LAYER_DSP => LAYER_LL1: quality: 0x0A80 (2688), pdcch_timing: 0x07 (7
LL1_NPDSCH_START_TIME: (00:01:21.5707000): LAYER_LL1 => LAYER_LL1: hfn: 0x0391 (913), sfn: 0xC3 (195), sf: 0x01
LL1_DCI_FORMAT_N1_NORMAL: (00:01:21.571100): LAYER_LL1 => LAYER_LL1: dci_n1: -
```

如果看到 dci_n1 的信息说明 pdcch 已经解到，msg1 基本可以认定收到

6.1.3.2. MSG2

MSG2 失败的情况一般有三种：RAR 超时（没有收到 DCI），MSG2 接收失败和 MSG2 解析失败。

6.1.3.2.1. RAR 超时

RAR 超时，可以从 LOG 中看到如下 LOG：

```
121670> 2016/12/22 18:01:05.556 - DSP_NPRACH_REQ: (00:14:26.270660): LAYER_LL1 => LAYER_DSP: cell_id: 0x64 (100), selectedStartSubcarrier: 0x0A (10), repetition: 0x04 (4), start_subframe: 0x02CE
121683> 2016/12/22 18:01:05.592 - DSP_NPDCCH_SEARCH_REQ: (00:14:26.309570): LAYER_LL1 => LAYER_DSP: samples_offset_ms: 0x00 (0), max_samples: 0x00 (0), search_space_duration: 0x20 (32), cell_id:
121688> 2016/12/22 18:01:05.664 - DSP_NPDCCH_SEARCH_CNF: (00:14:26.396972): LAYER_DSP => LAYER_LL1: quality: 0x00 (0), pdcch_timing: 0x00 (0), pdcch_repetitions: 0x00 (0), pdcch_elapsed_sf: 0x0E
121697> 2016/12/22 18:01:05.683 - DSP_NPDCCH_SEARCH_REQ: (00:14:26.398712): LAYER_LL1 => LAYER_DSP: samples_offset_ms: 0x00 (0), max_samples: 0x00 (0), search_space_duration: 0x20 (32), cell_id:
121702> 2016/12/22 18:01:05.788 - DSP_NPDCCH_SEARCH_CNF: (00:14:26.522949): LAYER_DSP => LAYER_LL1: quality: 0x00 (0), pdcch_timing: 0x00 (0), pdcch_repetitions: 0x00 (0), pdcch_elapsed_sf: 0x0E
121709> 2016/12/22 18:01:05.802 - DSP_NPDCCH_SEARCH_REQ: (00:14:26.524505): LAYER_LL1 => LAYER_DSP: samples_offset_ms: 0x00 (0), max_samples: 0x00 (0), search_space_duration: 0x20 (32), cell_id:
121714> 2016/12/22 18:01:05.923 - DSP_NPDCCH_SEARCH_CNF: (00:14:26.654998): LAYER_DSP => LAYER_LL1: quality: 0x00 (0), pdcch_timing: 0x00 (0), pdcch_repetitions: 0x00 (0), pdcch_elapsed_sf: 0x0E
121724> 2016/12/22 18:01:05.942 - DSP_NPDCCH_SEARCH_REQ: (00:14:26.656738): LAYER_LL1 => LAYER_DSP: samples_offset_ms: 0x00 (0), max_samples: 0x00 (0), search_space_duration: 0x20 (32), cell_id:
121729> 2016/12/22 18:01:06.048 - DSP_NPDCCH_SEARCH_CNF: (00:14:26.780029): LAYER_DSP => LAYER_LL1: quality: 0x00 (0), pdcch_timing: 0x00 (0), pdcch_repetitions: 0x00 (0), pdcch_elapsed_sf: 0x0E
121738> 2016/12/22 18:01:06.066 - DSP_NPDCCH_SEARCH_REQ: (00:14:26.781799): LAYER_LL1 => LAYER_DSP: samples_offset_ms: 0x00 (0), max_samples: 0x00 (0), search_space_duration: 0x20 (32), cell_id:
121743> 2016/12/22 18:01:06.222 - DSP_NPDCCH_SEARCH_CNF: (00:14:26.932006): LAYER_DSP => LAYER_LL1: quality: 0x00 (0), pdcch_timing: 0x00 (0), pdcch_repetitions: 0x00 (0), pdcch_elapsed_sf: 0x0E
121745> 2016/12/22 18:01:06.227 - LL1_RAR_TIMER_EXPIRY_IND: (00:14:26.932281): LAYER_LL1 => LAYER_LL1: timer_handle: 0x00 (0)
```

说明 UE 发出 PRACH 后，在 RAR 的窗口内没有收到 RAR 的 DCI。出现这种情况后，请确认信号强度是否合理，过滤 NRS_MEASUREMENT 可以查看各个参数，RSRP 不低于-140 左右，SNR 不低于-12.5。**最好有基站跟踪的 LOG，这类问题需要同时参考 UE LOG 和基站 LOG 定位**。

6.1.3.2.2. MSG2 接收失败

若 LOG 中能够看到收到了 MSG2 的 DCI(NPRACH 之后的 DCI_N1)，但是接着又发起了 NPRACH_REQ 的情况，极大可能是在 NPDSCH 信道上解析 MSG2 失败，可以看到 NPDSCH_CNF 的 crc_ok 为 0.

```
2016/12/22 18:21:15.680 - DSP_NPDSCH_REQ: (00:07:15.841949): LAYER_LL1 => LAYER_DSP: samples_offset_ms: 0, max_samples: 0, start_subframe: 1698, cell_id: 100, crnti: 60057, repetition_number: 1, tbs_size_bytes: 85, nrs_crs_pow
2016/12/22 18:22:55.656 - DSP_NPDSCH_CNF: (00:08:55.841583): LAYER_DSP => LAYER_LL1: quality: 0x2700 (9984), length: 0x55 (85), pdsch_valid_sf: 0x04 (4), pdsch_elapsed_sf: 0x05 (5), crc_ok: 0x00 (0)
```

出现此种情况后，需要核对参数以及帧号，应同时提供基站跟踪和 UE LOG。

6.1.3.2.3. MSG2 解析失败

MSG2 的解析结果可以通过过滤 RAR_UL_GRANT 查看。一般出错在 Preamble ID 不匹配之类的。同样，需要基站的 LOG 跟踪。

```
LL1_NPDSCH_START_TIME: (00:01:21.5707000): LAYER_LL1 => LAYER_LL1: hfn: 0x0391 (913), sfn: 0xC3 (195), sf: 0x01 (1), sf_from_now: 0x05 (5)
LL1_DCI_FORMAT_N1_NORMAL: (00:01:21.5711000): LAYER_LL1 => LAYER_LL1: dci_n1: -
DSP_NPDSCH_REQ: (00:01:21.5714500): LAYER_LL1 => LAYER_DSP: samples_offset_ms: 0, max_samples: 0, start_subframe: 1951, cell_id: 71, crnti
DSP_NPDSCH_CNF: (00:01:21.5772500): LAYER_DSP => LAYER_LL1: quality: 0x0800 (2048), length: 0x12 (18), crc_ok: 0x01 (1)
LL1_RAR_UL_GRANT: (00:01:21.5780000): LAYER_LL1 => LAYER_LL1: rar_pdu: -
```

这里看到 RAR 的话，说明 msg2 已经解到，点击这条消息可以进一步确认 rar 解码情况，确认是否是给当前 UE 的调度。

```
2016/9/30 8:19:51.894> LL1_RAR_UL_GRANT: (00:01:21.5780000): LAYER_LL1 => LAYER_LL1: rar_pdu: -
rar_pdu: -
 subcarrier_ind_nsc: 0x01 (1), pusch_start_subframe: 0x0C (12), modulation_coding_scheme_tbs: 0x58 (88),
result: (LL1_STORE_RAR_SUCCESS), expected_rapid: 0x26 (38)
```

看到 success 说明解码成功，如果看到 no_matching_rapid 或者其他关键字，则说明 rar 解码失败。

6.1.3.3. MSG3

```
LL1_NPUSCH_START_TIME: (00:01:21.5797250): LAYER_LL1 => LAYER_LL1: hfn: 0x0391 (913), sfn: 0xC4 (196), sf: 0x04 (4), sf_from_now: 0x08 (8)
LL1_PUSCH_CALC_TX_POWER: (00:01:21.5798500): LAYER_LL1 => LAYER_LL1: tx_power: 0xFEFC (-260), tx_power_db: 0xFFE6 (-26)
DSP_NPUSCH_REQ: (00:01:21.5799250): LAYER_LL1 => LAYER_DSP: crnti: 0xB341 (45889), cell_id: 0x47 (71), repetition_number: 0x01 (1), start_s
DSP_NPUSCH_CNF: (00:01:21.5958250): LAYER_DSP => LAYER_LL1:
```

上行信道，同理看不出发送结果，只能通过点击 req 消息来展开确认各个参数，或者根据 start_time 来核对发送帧号

6.1.3.4. MSG4

MSG4 与 MSG2 同样，存在窗口，若在窗口内没有收到，可以看到如下的 LOG：

```
14911> 2016/12/22 15:53:55.661 - DSP_NPDCCH_SEARCH_CNF: (00:00:46.777893): LAYER_DSP => LAYER_LL1: quality: 0x00 (0), pdcch_timing: 0x00 (0), pdcch_repetitions: 0x00 (0), pdcch_elap
14912> 2016/12/22 15:53:55.661 - LL1_LOG_FSM_EVENT: (00:00:46.777954): LAYER_LL1 => LAYER_LL1: fsm: (LL1_FSM_CONNECTED), state: 0x08 (8), event: 0x01400010 (20971536)
14913> 2016/12/22 15:53:55.662 - LL1_MAC_CONTENTION_RESOLUTION_TIMER_EXPIRY_IND: (00:00:46.778137): LAYER_LL1 => LAYER_LL1: timer_handle: 0x00 (0)
```

正常情况下，LOG 如下：

```
LL1_NPDCCH_START_TIME: (00:01:21.5965250): LAYER_LL1 => LAYER_LL1: hfn: 0x0391 (913), sfn: 0xC6 (198), sf: 0x04 (4), sf_fr
DSP_NPDCCH_SEARCH_REQ: (00:01:21.5970250): LAYER_LL1 => LAYER_DSP: samples_offset_ms: 0x00 (0), max_samples: 0x00 (0), sea
DSP_NPDCCH_SEARCH_CNF: (00:01:21.6093250): LAYER_DSP => LAYER_LL1: quality: 0x0A80 (2688), pdcch_timing: 0x00 (0), pdcch_r
LL1_NPDSCH_START_TIME: (00:01:21.6102250): LAYER_LL1 => LAYER_LL1: hfn: 0x0391 (913), sfn: 0xC7 (199), sf: 0x01 (1), sf_fr
LL1_DCI_FORMAT_N1_NORMAL: (00:01:21.6105250): LAYER_LL1 => LAYER_LL1: dci_n1: -
DSP_NPDSCH_REQ: (00:01:21.6108000): LAYER_LL1 => LAYER_DSP: samples_offset_ms: 0, max_samples: 0, start_subframe: 1991, ce
DSP_NPDSCH_CNF: (00:01:21.6195250): LAYER_DSP => LAYER_LL1: quality: 0x0F80 (3968), length: 0x29 (41), crc_ok: 0x01 (1)
DSP_NPUCCH_CNF: (00:01:21.6368000): LAYER_DSP => LAYER_LL1:
LL1_RACH_STATS_IND: (00:01:21.6372250): LAYER_LL1 => LAYER_LL1: rach_stats: -
```

与 msg2 相同是下行信道，所以确认 pdcch 与 pdsch 的 crc 正确后，msg4 收到。

这里注意所有下行的信道（pdcch/pdsch）都可以通过看 crc_ok 来确认 UE 的基带是否收到信息（先不论内容对错）。

Crc_ok = 1 成功

Crc_ok = 0 失败

6.1.4. 尝试定位

如果通过以上分析发现如下问题，可以参考下述手段来定位，并同时记录问题向家里报告。

- Msg2没有收到，对应的前面的dsp_pdcch_search_cnf返回的crc_ok = 0

  该问题会导致本次随机接入失败，并不断重试，直到超过最大重复次数。

1. 请与基站同事抓数确认pdcch调度帧号，UE侧请参照上文msg1中所写方法确认。基站侧请确认是否与UE开始搜索的pdcch帧号一致的位置发送。

   如果基站没有调度，请继续回溯确认 msg1 基站是否收到，msg1 的对数对参方法同样请参照上文 msg1 所写。

- Msg4没有收到，对应的前面的dsp_pdcch_search_cnf返回的crc_ok = 0

  同上第 2 点，直接对数确认 pdcch 本身，如果没有调度请回溯确认基站有没收到 msg3。

  最关键的是，能够同时提供 UE LOG 和基站跟踪给家里，方便家里确认调度是否正确。

6.2. 系统消息收不全问题

6.2.1. LOG 过滤

Cell_select

pdsch

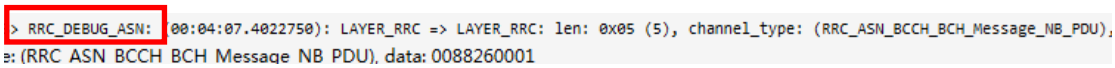
Asn

si_info_read

si_info_ind

6.2.2. 初步分析

看系统消息与随机接入略有不同，需要结合 ASN 消息看 RRC 收到的结果，注意，所有 ASN 消息都可以点击展开查看消息内容!



1. MIB

MIB 消息与 SIB1 消息之后会进行系统消息的搜索， MIB+SIB1 消息的读取会有两次，第二次的 SIB1 之后才会进行 SI 系统消息读取，找到第二次的 MIB

```
3> RRC_DEBUG_ASN: (00:04:07.4022750): LAYER_RRC => LAYER_RRC: len: 0x05 (5), channel_type: (RRC_ASN_BCCH_BCH_Message_NB_PDU),
e:(RRC_ASN_BCCH_BCH_Message_NB_PDU), data: 0088260001
```

2. SIB1

随后找到 SIB1，由于也是 pdsch 下行消息，通过 CRC 确认成功与否

```
DSP_NPDSCH_REQ: (00:04:07.4034250): LAYER_LL1 => LAYER_DSP: samples_offset_ms: 0, max_samples: 0, start_subframe: 2574, cell_id:
DSP_NPDSCH_CNF: (00:04:12.4375000): LAYER_DSP => LAYER_LL1: quality: 0x4A8580 (4883840), length: 0x1A (26) crc_ok: 0x01 (1)
RRC_DEBUG_ASN: (00:04:12.4385000): LAYER_RRC => LAYER_RRC: len: 0x1A (26), channel_type: (RRC_ASN_BCCH_DL_SCH_Message_NB_PDU), d
```

3. SI配置确认

找到 ll1_si_info_read_req，这是 RRC 下发 SI 搜索指令

```
2016/9/30 9:38:15.124> LL1_SI_INFO_READ_REQ: (00:04:12.4393750): LAYER_RRC => LAYER_LL1: deployment_cfg: (Unknown Field Type: union_anon_68)
deployment_cfg: (Unknown Field Type: union_anon_68)
si_params: -
  earfcn: 0x0DEB (3563), phy_cell_id: 0x47 (71), si_window_length: 0x03C0 (960), num_of_si: 0x02 (2), si_schd_info_sib1: 0x02 (2)
si_scheduling0: -
  sib_mapping_bitmap: 0x01 (1), si_periodicity: 0x0200 (512), si_tbs_size: 0x0148 (328), repetition_pattern: 0x04 (4), radio_frame_offset: 0x00 (0)
si_scheduling1: -
  sib_mapping_bitmap: 0x02 (2), si_periodicity: 0x0800 (2048), si_tbs_size: 0x38 (56), repetition_pattern: 0x04 (4), radio_frame_offset: 0x00 (0)
si_scheduling2: -
  sib_mapping_bitmap: 0x00 (0), si_periodicity: 0x00 (0), si_tbs_size: 0x00 (0), repetition_pattern: 0x00 (0), radio_frame_offset: 0x00 (0)
si_scheduling3: -
  sib_mapping_bitmap: 0x00 (0), si_periodicity: 0x00 (0), si_tbs_size: 0x00 (0), repetition_pattern: 0x00 (0), radio_frame_offset: 0x00 (0)
si_scheduling4: -
  sib_mapping_bitmap: 0x00 (0), si_periodicity: 0x00 (0), si_tbs_size: 0x00 (0), repetition_pattern: 0x00 (0), radio_frame_offset: 0x00 (0)
si_scheduling5: -
  sib_mapping_bitmap: 0x00 (0), si_periodicity: 0x00 (0), si_tbs_size: 0x00 (0), repetition_pattern: 0x00 (0), radio_frame_offset: 0x00 (0)
si_scheduling6: -
  sib_mapping_bitmap: 0x00 (0), si_periodicity: 0x00 (0), si_tbs_size: 0x00 (0), repetition_pattern: 0x00 (0), radio_frame_offset: 0x00 (0)
si_scheduling7: -
  sib_mapping_bitmap: 0x00 (0), si_periodicity: 0x00 (0), si_tbs_size: 0x00 (0), repetition_pattern: 0x00 (0), radio_frame_offset: 0x00 (0)
ll1_valid_subframes: -
  num_valid_subframes: 0x00 (0), valid_subframes0: 0x00 (0), valid_subframes1: 0x00 (0), valid_subframes2: 0x00 (0), valid_subframes3: 0x00 (0), valid_subframes4: 0x00 (0)
si_msg_required_bitmap: 0x03 (3), mode: (LL1_DEPLOYMENT_MODE_INBAND_SAMEPCI)
```

可以看到 RRC 下发了对 SI0 和 SI1（第一条和第二条调度消息分别对应，其余 SI 以此类推）搜索的参数配置与搜索指令，接下来 LL1 会开始 SI 搜索。

4. SI搜索结果

```
DSP_NPDSCH_CNF: (00:04:28.7385250): LAYER_DSP => LAYER_LL1: quality: 0x1B6480 (1795200), length: 0x29 (41), crc_ok: 0x01 (1)
DSP_NPDSCH_REQ: (00:04:28.7397000): LAYER_LL1 => LAYER_DSP: samples_offset_ms: 0, max_samples: 0, start_subframe: 961, cell_id:
LL1_SI_INFO_IND: (00:04:28.7409500): LAYER_LL1 => LAYER_RRC: status: True, si_msg_index: 0x00 (0), length: 0x29 (41), data_p: 0x
x00 (0), length: 0x29 (41), data_p: 0x010145D8 (16860632), earfcn: 0x0DEB (3563), phy_cell_id: 0x47 (71), sfn: 0x029D (669)
RRC_DEBUG_ASN: (00:04:28.7410250): LAYER_RRC => LAYER_RRC: len: 0x29 (41), channel_type: (RRC_ASN_BCCH_DL_SCH_Message_NB_PDU), d
(RRC_ASN_BCCH_DL_SCH_Message_NB_PDU), data: 0000033138E38B0B4603A8664992B665212C666A92B5278003B9444C062A1DB41C
```

先找到 ll1_si_info_ind 消息，对应 status 为 true 说明解到一条 SI，对应 si_msg_index 为 0 说明，是上文中的第一条调度消息，也就是 SIB2(SI 和 SIB 的包含关系根据基站配置决定)成功收到并上报。往上找到第一条 dsp_npdsch_cnf 的 CRC 为 1 也说明了信道上 CRC 解码成功。
接着可以点开后面的 ASN 来确认 SIB2 消息的内容。
同理，校验其余 SIB。注意，目前收到 SIB2 后 RRC 就会停止搜索 SI 并发起建链请求。

6.2.3. 尝试定位

发现哪一条 SI 消息 CRC 为 0 出现多次，RRC 在 LL1 多次搜索失败后就会视为当前 cell_select 结果为 reject，出现这样的情况，就可以记录下来给家里分析了。当前在弱信号下 SI 消息有可能搜索不到。

6.3.覆盖等级和测量值

6.3.1. LOG 过滤

覆盖等级的过滤为 ecl
测量值的过滤为 nrs_measure

6.3.2. 覆盖等级 log 分析

当过滤 ecl 时，可以看到 LL1_LOG_ECL_INFO 这条消息，log 如下：

```
· LL1_LOG_ECL_INFO: (00:10:12.527313): LAYER_LL1 => LAYER_LL1: current_ecl: 0x00 (0), ecl_selected_by: (LL1_RACH_ECL_SELECTED_BY_MEASUREMENT), rsrp: 0xFC92 (
· LL1_LOG_ECL_INFO: (00:10:14.837127): LAYER_LL1 => LAYER_LL1: current_ecl: 0x01 (1), ecl_selected_by: (LL1_RACH_ECL_SELECTED_NEXT_COVERAGE_LEVEL), rsrp: 0xF
```

- current_ecl 就是当前选的覆盖等级，上面的两个覆盖等级就分别为 0 和 1。
- ecl_select_by 是覆盖等级获得的方式，最常见的是如下两种覆盖等级获得方式。
  ①LL1_RACH_ECL_SELECTED_BY_MEASUREMENT 就是通过测量的 rsrp 和门限来获得。
  LL1_LOG_ECL_INFO 这条消息后面两个参数是门限，`threshold0: 0xFB6E (-1170), threshold1: 0xFB0A (-1270)`
  这两个门限就分别是覆盖等级 1 和 2 的 rsrp 门限。以前是通过 rsrp 和门限就能获取当前的覆盖等级。但是从 B650SP6 开始，覆盖等级的的算法多了一个 SNR，SNR 的两个门限分别为 30 和-70。只要 rsrp 和 SNR 中的一个满足覆盖等级 1 或 2 的门限，就选择覆盖等级 1 或 2.
  ②LL1_RACH_ECL_SELECTED_NEXT_COVERAGE_LEVEL 是覆盖等级攀升，当我们用测量获得的覆盖等级做 rach 失败几次以后，会提升覆盖等级，尝试用下一个覆盖等级接入。

6.3.3. 测量值 log 分析

当过滤 nrs_measure 时，可以看到 LL1_NRS_MEASUREMENT_LOG 这条消息，log 如下：

```
LL1_NRS_MEASUREMENT_LOG: (00:07:41.808410): LAYER_LL1 => LAYER_LL1: rsrq: 0xFF94 (-108), rsrp: 0xFC91 (-879), snr: 0x012C (300),
rssi: 0xFCCF (-817), filtered_rsrp: 0x00 (0), filtered_rsrq: 0x00 (0), filtered_nrs_snr: 0x00 (0),
```

其中 rsrp，rsrq，snr 是本次的测量值。
filtered_rsrp，filtered_rsrq，filtered_nrs_snr 是取最近 4 次的平均值，更准确。但是它们只在 cell_select 之后才有效，cell_select 之前这些值为 0.

6.4.上行发射功率

6.4.1. LOG 过滤

tx_power_dbm

6.4.2. 上行发射功率分析

在 LOG 中找到想观测的那一条上行信道 prach/pusch，再通过过滤可以找到如下 LOG：

```
- DSP_NPUSCH_REQ: (00:00:43.486175): LAYER_LL1 => LAYER_DSP: crnti: 0x3A31 (14897), cell_id: 0x2B
- DSP_LOG_IPC_DATA: (00:00:43.486267): LAYER_DSP => LAYER_LL1: data0: 0x3D (61), data1: 0x23 (35)
- LL1_LOG_RF_START_TIME_CALC: (00:00:43.486328): LAYER_LL1 => LAYER_LL1: rf_tick_ref: 0x637E6437
- LL1_RF_TX_START: (00:00:43.487152): LAYER_LL1 => LAYER_LL1: start_time_ticks: 0x6387C3E7 (16698
```

在 LL1_RF_TX_START 的最右边可以找到

```
configure_time: 0x0266D9 (157401), ready_time: 0x01F88D (129165), tx_power_dbm: 0xFFF9 (-7)
```

这就是当前这次上行的发射功率

6.5. IDLE 态接收 Paging 问题

6.5.1. 注意事项：

1， 确认是否使能 eDRX 特性，eDRX 特性对 CN, eNodeB, UE 版本有限制。

2， 需要确保 UE 成功接入后回到 IDLE 状态，并且 UE 不能进入 PSM 状态。测试时可以看到 UE 收到”release”并且 RRC 下发 ”IDLE_CONFIG_REQ” 之后再进行 paging。

```
25456> 2016/12/24 11:56:45.089 - RRC_DEBUG_ASN: (00:01:18.331298): LAYER_RRC => LAYER_RRC: len: 0x02 (2), channel_type: (RRC_ASN_DL_DCCH_Message_NB_PDU), data: 2002
25627> 2016/12/24 11:56:52.764 - RRC_DEBUG_ASN: (00:01:18.613830): LAYER_RRC => LAYER_RRC: len: 0x05 (5), channel_type: (RRC_ASN_BCCH_BCH_Message_NB_PDU), data: 9C8AC00018
25636> 2016/12/24 11:56:52.813 - LL1_IDLE_CONFIG_REQ: (00:01:18.615844): LAYER_RRC => LAYER_LL1: deployment_cfg: (Unknown Field Type: union_anon_116)
25647> 2016/12/24 11:56:54.044 - LL1_IDLE_CONFIG_CNF: (00:01:18.623107): LAYER_LL1 => LAYER_RRC: status: (LL1_CONFIG_STATUS_SUCCESS)
```

3， 在使能 eDRX 特性时，由于 eDRX 周期较长以及时钟同步问题，CN 可能很久之后才下发 paging 消息，并且 eNodeB 有的可能丢掉核心网下发的 paging 消息，因此 paging 没有成功时请先确认 eNodeB 是否下发 paging。

4，

2016-12-24 08:22:31(104)	RRC_PAGING_NB	eNB发送给UE	80
2016-12-24 08:22:31(924)	RRC_CONN_REQ_NB	UE发送给eNB	80
2016-12-24 08:22:31(926)	RRC_CONN_SETUP_NB	eNB发送给UE	80
2016-12-24 08:22:32(181)	RRC_CONN_SETUP_CMP_NB	UE发送给eNB	80

5， 对于系统消息更新触发 paging 的场景，因为系统消息不会立刻更新，因此 UE 可能延迟一段时间后再去更新。

6.5.2. LOG 过滤：

asn

idle_config

paging

n2_

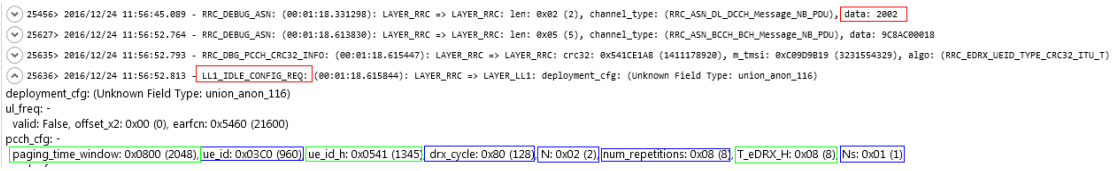
crc32

psm

pdcch_search_req

6.5.3. 初步分析：

1， 确认配置参数是否一致：
通过看 release 之后的”LL1_IDLE_CONFIG_REQ” 中 DRX 参数（图中蓝色出）和 eDRX 参数（图中绿色出）是否正确。



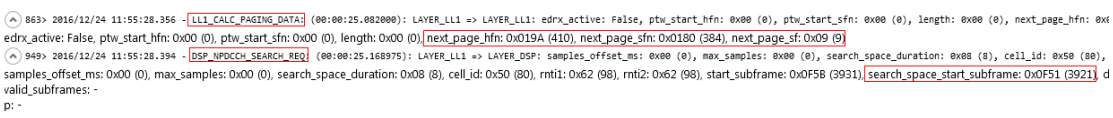
其中 DRX 参数通过 SIB2 配置：

```
pcch-Config-r13
{
  defaultPagingCycle-r13 rf128,
  nB-r13 one64thT,
  npdcch-NumRepetitionPaging-r13 r8
},
```

eDRX 参数通过 Attach Accept 信令配置：

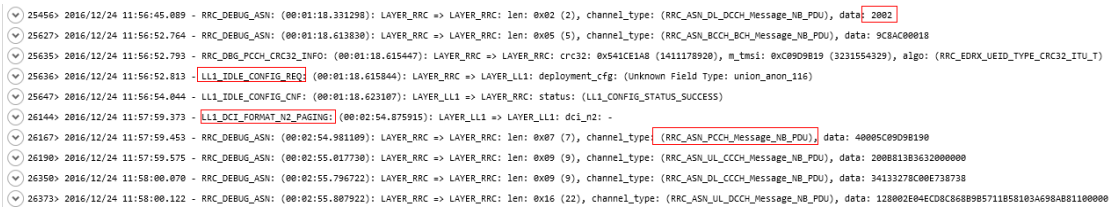
```
Extended DRX Parameters
    Element ID: 0x6e
    Length: 1
    0111 .... = Paging Time Window: Iu: 7 s / WB-S1: 10.24 s / NB-S1: 20.48 s (0x7)
    .... 0101 = eDRX value: GERAN: 48.96 s / UTRAN: 327.68 s / E-UTRAN: 81.92 s (0x5)
```

2， 确认 PO 是的正确：
通过 ”LL1_CALC_PAGING_DATA” 和 ”DSP_NPDCCH_SEARCH_REQ” 中看到下次 PO 的 HFN, SFN, SF 位置，以及避让后真正的解析位置与理论计算或基站 log 进行比对。



3， Paging 场景下，UE 解析到 Paging DCI 之后会打出 Paging DCI 和 Paging 消息，之后进

行接入，如果没有接入可在 Paging 消息中确认 s_TMSI 是否匹配。




```
PCCH-Message-NB
{
 message c1 : paging-r13 :
 {
  pagingRecordList-r13
  {
   {
    ue-Identity-r13 s-TMSI :
    {
     mmec '00000101'B,
     m-TMSI '11000000 10011101 10011011 00011001'B
    }
   }
  }
 }
}
```

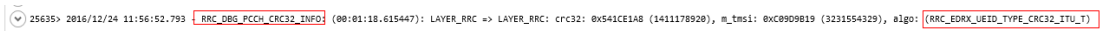
4， 系统消息变更场景下，UE 会解析到多条 Paging DCI，之后 UE 会读取系统消息但不会接入。

6.5.4. 有无解决/规避方案。

eDRX 参数 UE_ID_H 与基站不一致导致 UE 收不到 Paging 消息：

先确认 CN，eNodeB，UE 版本是否正确。

版本正确后再确认 UE 使用的 CRC 算法是否一致。新版 eNodeB 使用 ITU 标准：



如果 UE 不是使用该算法请使用 NV 配置命令修改算法：

```
stackmsg.exe send 4294967295 RRC_NVCONFIG_CRC_CFG_TYPE LAYER_RRC 1 2
```

7. 应用核错误码打印

测试时设置 AT+CMEE=1，错误时能够返回错误码，错误码信息非常重要能够快速定位是否是问题，也能很快的定位出相关问题。

例如：当需要用 AT+NCDP=138.3.13.5 来设置 CoAP 服务器地址时，如果没有预先设置 IMEI，就会返回 ERROR，如果我们设置了 AT+CMEE=1，就会返回：
+CME ERROR：256
这时我们就能知道设置 NCDP 的前置条件没有满足。
目前常见的错误码:

```
{AT_CAUSE_CMD_IN_PROGRESS, 3},          //operation not allowed
{AT_CAUSE_PROGRESS_ERROR, 4},           //operation not supported
{AT_CAUSE_MEMORY_ERROR, 23},            //memory failure
{AT_CAUSE_NOT_NUMERIC, 50},             //Incorrect parameters
{AT_CAUSE_PARAM_MISSING, 50},           //Incorrect parameters
{AT_CAUSE_FLOW_CONTROL, 159},           //Uplink Busy   功能正常，发生流控
{AT_CAUSE_PARAM_UNCONFIG, 256},         //Required Parameter Not Configured
{AT_CAUSE_KV_NOT_ENOUGH_SPACE, 257},    //kv has not enough space
```